Abstract

Glenn A. Thesing. Analysis and Local Estimation of Spatial Data Under the Intrinsic Hypothesis of Spatial Random Fields. (Under the direction of Dr. George Christakos.)

The intrinsic hypothesis of random fields is applied in the analysis of spatially distributed point-value data. In particular, the intrinsic hypothesis is coupled with a kriging point estimator to study the local behavior of non-homogeneous natural processes over two-dimensional space and to produce maps of degree of spatial trend, spatial correlation features, and point estimates.

An automated computer algorithm is implemented to conduct the analysis at a local scale. A rectangular grid is assigned throughout the data domain. Trend assessment, covariance modeling, and estimation are conducted at each grid node using local subsets of data. The algorithm is applied to a set of soil moisture content measurements. Sensitivity analyses are conducted and indicate estimation accuracy is dependent on the size of the local subset and the covariance model used.

Table of Contents

Section 1.0
Introduction

In the sciences which study environmental, hydrologic, atmospheric and other phenomena, an investigator may often collect a set of data consisting of the measured values of some parameter (e.g., atmospheric deposition of sulfate) at various discrete locations in space. If the process is transient in time such as phreatic ground water elevations, the parameter values may be assumed to have been measured simultaneously, and if the process may be considered invariant in time such as bedrock surface elevations, the measurements need not be simultaneous. The problem which this work addresses is to *estimate* the parameter at a location for which there is no measurement by using the data provided. The problem is referred to as point estimation and may be approached with a number of different methodologies (see Chpt. 11, Isaaks, 1989). This work implements an unbiased linear estimation method which may be grouped within the category of kriging methods. The method used here is referred to as *intrinsic kriging*.

Since the given data set provides information about the phenomenon only at the locations of the data points, it is necessary to introduce a model to estimate the phenomenon at unmeasured locations. Given that the phenomena usually studied are the result of complex interactions of many processes, it is unlikely that useful deterministic models which preserve the complexity of the process have been developed, or if they have, they may require the estimation of an impractically large number of parameters. The scientific understanding of process interactions may be too simplistic to permit such an approach without introducing assumptions which oversimplify the model, resulting in an analytically tractable but potentially inaccurate representation. A probabilistic model may be more appropriate in such cases. Within the framework of a probabilistic model, the sample data are viewed as the result of a physical process which, due to the complexity of its interacting components and the lack of understanding of those components, appears to

contain an element of statistical randomness. In this work, a probabilistic model is chosen with the following considerations and requirements:

- the phenomenon under study is modeled as a statistically correlated spatial random field (SRF);
- the set of sample data are assumed to represent one *realization* from an infinite set of possible realizations of the random field;
- the modeled random field is permitted to exhibit non-homogeneity in its statistical mean and/or covariance;
- the characteristics of non-homogeneity are analyzed using the tools provided by the intrinsic hypothesis (see Christakos and Thesing, 1993);
- characterization of the non-homogeneous process, point estimation, and calculation of estimation error variance are all conducted at a *local* scale using a local subset of the complete data set.

As mentioned previously, this work implements a form of the point estimation method known as kriging. Ordinary kriging -- the conventional form of the kriging methods -- is an unbiased estimation method which defines a point estimator as a linear combination of weighted values of the known data points. The derivation of the method and its application are readily found in the literature (e.g., Chpt. 12, Isaaks, 1989). At the core of the kriging process is the fitting of a covariance or semivariogram model to the data. The principle weakness of ordinary kriging is that it assumes the random field represented by the data to be homogeneous in space. However, rarely are spatial data of natural processes homogeneous, and a non-constant mean is generally the source of non-homogeneity (Cressie, 1986). As mentioned previously, the ability to accept and characterize non-homogeneous data is one of the primary considerations in the development of this work. Thus, a more general approach is required.

Matheron's (1973) theory of the intrinsic spatial random field (ISRF) offers a framework in which to build models capable of accepting non-homogeneous processes.

The theory is derived from the basic principle that a non-homogeneous spatial random field can be mathematically transformed into an analog which is homogeneous. The mathematical analog is comprised of spatial increments where each increment is a linear combination of values from the original SRF. In constructing these increments, the spatial trend, which renders the original SRF non-homogeneous, is removed by way of mathematical cancellation or filtering. The resulting SRF of increments is , thus, homogeneous.

Non-homogeneous data may be accepted under the intrinsic hypothesis without making the assumption that the data are locally or quasi-homogenous. The assumptions that *are* made apply to the nature of the intrinsic model, not the data. This characteristic is an important motivation for this work because it removes the often-unrealistic restriction of data homogeneity. However, the generality of the intrinsic hypothesis makes it equally applicable to homogeneous processes.

One of the integral components of ISRF theory is the *order of intrinsity*, ν. Qualitatively, ν represents the complexity of trend in the mean of a non-homogeneous SRF. The higher the value of ν, the more complex is the trend. Another important aspect is that the ordinary covariance for a non-homogeneous SRF may be decomposed into a homogeneous component -- the generalized spatial covariance -- and a non-homogeneous component usually represented by a polynomial of degree ν. It turns out that under the intrinsic hypothesis, the order of intrinsity ν and the generalized covariance function for a non-homogeneous (or homogeneous) SRF provide a complete stochastic characterization of the SRF's spatial correlation structure. In addition to the data themselves, these parameters are necessary and sufficient inputs to the spatial estimation procedure developed for use under the intrinsic hypothesis.

The objective of this work has been to develop an automated computer program to conduct the analysis of degree of trend and covariance modeling on a local scale using proximal subsets of the complete data set. This analysis is followed by estimation of the

phenomenon via kriging at unmeasured locations. By considering local subsets of the data instead of the entire set, we allow the model to depict spatial variability of trend and correlation characteristics which may be present in the actual phenomenon. Conducting the procedure at numerous points within some domain of interest results in graphical depictions of the spatial variability of ν, covariance parameters, point estimates, and estimation error variances. These depictions then provide insight into the behavior of the phenomenon across the algebraic structure of space.

Section 2.0
Introductory SRF Concepts and Theory

Motivation

Empirical analysis of data from spatially distributed natural phenomena -- such as atmospheric deposition of sulfate, or hydraulic conductivity in an aquifer system -- often indicates that the spatial variablility of these processes has two important descriptive features, each of which is associated with a pariticular scale of observation. On a regional or *macroscopic* level, there may be a well-defined spatial structure resulting from trend or, as in time series analysis, cyclic changes. On a local or *microscopic* scale, the process may exhibit irregular, erratic fluctuations characteristic of unstructured random behavior. The coexistence of both these macroscopic and microscopic properties is termed the *macro-micro duality* and is essential to the core of spatial random field theory.

In constructing a model to study these phenomena, both features should be addressed. Classical theory of probability could be implemented by considering each data observation to be the outcome of a random variable. However, this platform alone fails to account for the spatial structure and correlation among the collective of random variables. An improvement upon this concept is the spatial random field (SRF) in which the behavior of the phenomenon is conceptually (and eventually mathematically) disassociated into a deterministic trend component describing the macroscopic structural features and a random residual accounting for erratic fluctuations on the micro-scale. The following discussion will expand some fundamental concepts underlying SRF theory. Throughout the discussion, lower case letters x, y, etc. will denote random variables; upper case letters X, Y, etc. will denote random fields; and lower case Greek letters χ, ψ, etc. will denote the values of the random variables and spatial random fields.

Characteristics and Definitions

In the most basic sense, a SRF may be conceptualized as a collection of random variables $x_i(\xi)$ in an n-dimensional domain where each $x_i(\xi)$ is associated with a location $\mathbf{s}_i$ in $\mathbf{R}^n$; $\mathbf{s}_i = (s_1,\ldots,s_n)_i$, and ξ is an elementary event from the sample space Ω. Associated with each $x_i(\xi)$ is a probability density $f_{x_i}(\chi)$ which ascribes a probability law to the random variable. Figure 1 illustrates this concept in $\mathbf{R}^1$. Such a collection of random variables in $\mathbf{R}^n$ is referred to as a *spatial random field* (SRF) $X(\xi,\mathbf{s})$. When this construction is restricted to one spatial dimension, it is referred to as a *random process.* The configuration of the probability distributions -- their relative arrangement -- for the collective of random variables is what imparts a spatial structure the phenomenon on a macro-scale. The outcome of the SRF at any location is a random outcome subject to the constraints of the governing probability law at that location. Thus, random behavior is possible at any location in the SRF.

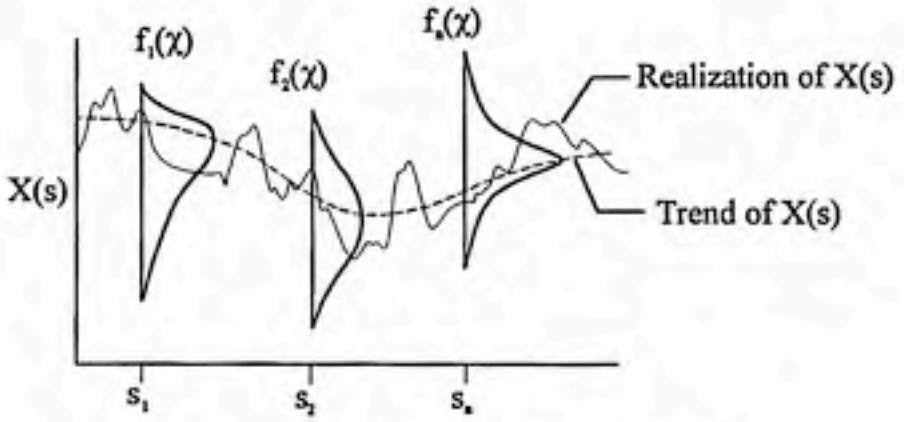


Figure 1. A one-dimensional SRF.

Spatial random fields exhibit variability with the spatial coordinates s and the state variable ξ. If s is fixed, then $X(\xi,\mathbf{s}_0)$ becomes a random variable specific to location $\mathbf{s}_0$. If ξ is fixed, then $X(\xi_1,\mathbf{s})$ becomes a *realization* of X across the ensemble of random

variables (Figure 1). For notational convenience, ξ is generally omitted from this representation: $X(\xi,\mathbf{s}) = X(\mathbf{s})$.

Classifications

A SRF can be qualitatively described by the following classifications:

- discrete or continuous in **s**;
- discrete or continuous in χ;
- vector or scalar **s** and $X(\mathbf{s})$;
- Gaussian or non-Gaussian probability densities;
- homogeneous or non-homogenous;
- isotropic/anisotropic;
- degree of memory of $X(\mathbf{s})$;
- spatial/spatio-temporal.

Within the context of this investigation, only classifications based on discrete/continuous, vector/scalar, and homogeneous/non-homogenous parameters are relevant. The SRFs considered here will be continuous in both **s** and χ; 2-dimensionally vector in spatial location **s**; and scalar in the SRF value $X(\mathbf{s})$. Regarding homogeneity, the SRFs considered here may be homogeneous or non-homogeneous. Consider the following example of a waste disposal site with chemically impacted soils. The spatial location vector **s** is continuously defined over the site and is comprised of the components s_1 and s_2 in $\mathbf{R}^2$, while the values of chemical concentrations represented by the SRF $X(\mathbf{s})$ are scalar magnitudes over some range $[0, \chi_{max}]$. The SRF of concentrations may be homogeneous or non-homogeneous; the intrinsic method employed here will accept either.

Homogeneity of a Spatial Random Field

Homogeneity is an integral consideration in the modeling of SRFs. Whether or not a physical process can be appropriately modeled as a homogeneous SRF is one of the key

questions an analyst must answer before constructing a model to represent the process. Within the context of geostatistical and stochastic modeling of spatial phenomena, there are two classes of homogeneity. The first class is *strong homogeneity* and refers to the condition in which the individual probability densities in the ensemble of random variables that comprise $X(\mathbf{s})$ do not vary with translation across $\mathbf{s}$. The second class is *weak homogeneity*, also known as second-order homogeneity, and refers to the condition in which only the first two moments of the probability densities--the mean and the covariance--are invariant with translation across the ensemble of random variables in $\mathbf{s}$. That is, the mean value is given by

$$E[X(\mathbf{s})] = m \qquad [1]$$

where m is constant across all $\mathbf{s}$. The covariance is a measure of the correlation between two random variables at different points in space and is defined as

$$c(\mathbf{s},\mathbf{s}+\mathbf{r}) = E\{[X(\mathbf{s})-m(\mathbf{s})][X(\mathbf{s}+\mathbf{r})-m(\mathbf{s}+\mathbf{r})]\} \qquad [2]$$

where $\mathbf{r}$ is the spatial vector, or *lag*, separating the locations of the two random variables under consideration. For a homogeneous process $m(\mathbf{s}) = m(\mathbf{s}+\mathbf{r}) = m$, and thus the covariance between any two locations $\mathbf{s}$ and $\mathbf{s}+\mathbf{r}$ is given by

$$c(\mathbf{s},\mathbf{s}+\mathbf{r}) = c(\mathbf{r}). \qquad [3]$$

Note that for the homogeneous case, the covariance is a function only of the lag $\mathbf{r}$ separating the two points, not of the actual locations of those points. Furthermore, a SRF may be non-homogeneous in its mean or covariance or both. Strong homogeneity necessarily includes weak homogeneity, although a weakly homogeneous SRF is not necessarily strongly homogeneous. In the geostatistical literature, the term "homogeneity" is generally used to indicate weak homogeneity; this paper will follow that convention.

In practice, data sets from physical processes, especially natural processes, rarely exhibit homogeneity. There is usually some trend which precludes a constant mean. Thus, in analyses of real data, the assumption of a homogeneous SRF is often unrealistic.

A significant amount of research in geostatistical and stochastic methods has focused on the analysis of homogeneity/non-homogeneity and the development and implementation of methods that can effectively model non-homogeneous phenomena (e.g., Bilonick, 1985; Christensen, 1990; Cressie, 1986; Venkatram, 1988).

Statistical inference of a covariance or semivariogram function is critical to the success of kriging estimation and other geostatistical methods. The SRF approach holds that spatial phenomena are described by the underlying macroscopic trend and by locally-based random fluctuations around the base trend. For homogeneous data, the data themselves may be used to calculate an estimate of the constant mean. Thus, the disassociation of the realization (as represented by the data) into a deterministic trend component and a random component is straightforward. However, the mean for non-homogeneous data changes with location. The disassociation is no longer clear-cut, since the mean is not well defined. This uncertainty leads to difficulty in statistical inference of an appropriate covariance function. Furthermore, a non-constant trend may lead to serious bias in the experimental semivariogram (Cressie, 1991)

$$\gamma_x(r) = \frac{1}{2} Var\left[X(s+r) - X(s)\right] \qquad [4]$$

leading to errors in parameter estimation for a semivariance model, which can ultimately lead to misinterpretations regarding the underlying structure of the phenomenon.

These difficulties raise the question of how to address the estimation of non-homogeneous data. When the estimation approach is kriging, several methods are used frequently. One approach is to conduct variogram or covariance inference and estimation with conventional, homogeneous methods but on a local scale using proximal subsets of data. The rationale is that a non-homogeneous process defined over a large region will exhibit less absolute trend and a more homogeneous correlation structure over a subregion relative to the entire domain. This concept is termed *quasi-homogeneity* and is implemented by Haas (1990) in his study of atmospheric sulfate deposition. Another

approach, termed *universal kriging* (Matheron, 1969), is an extension of ordinary kriging and involves the modeling of the trend as a polynomial of finite (and unknown) degree. Defining the trend analytically allows it to be accounted for in the analysis. Intrinsic kriging is of course another approach and will be explored in detail in this paper.

Section 3.0
Overview of ISRF-ν Theory

This section will present the mathematical theory underlying the intrinsic spatial random field of order ν (ISRF-ν). The discussion will begin with definitions of two of the integral components of ISRF-ν theory: order of intrinsity ν and the spatial increment of order ν. These fundamental components are requisite to the definition of the ISRF-ν. The generalized spatial covariance is integral to the spatial estimation procedure implemented in this work and will be introduced following the ISRF-ν discussion. Within the context of this discussion, SRF will refer to any random field, and ISRF-ν will refer to any SRF which can be mathematically transformed into a homogeneous analog.

Order of Intrinsity ν

The order of intrinsity ν of an ISRF is analogous to the degree of a polynomial which describes the spatial trend in the mean of the SRF. For example, an ISRF of order ν=0 refers to a SRF with constant mean; an ISRF of order ν=1 refers to a SRF with a linear trend in the mean; an ISRF of order ν=2 refers to a SRF with quadratic trend in the mean; and so on. The higher the representative value of ν, the more complex is the trend. Figure 2 illustrates this concept. The order ν provides insight into the complex spatial structure of the ISRF. The non-homogeneous mean is mathematically described in terms of a polynomial of order ν.

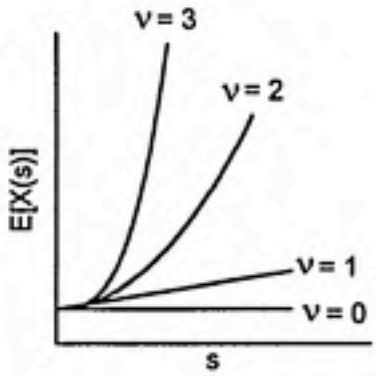


Figure 2. Conceptual illustration of ν.

Spatial Increment of Order ν

Given the spatial random field $X(\mathbf{s})$, the linear combination of SRF values

$$Y_q(\mathbf{s}) = \sum_{i=1}^{m} q_i X(\mathbf{s}_i) \qquad [5]$$

is defined to be a generalized *spatial increment of order* ν (SI-ν) if and only if the coefficients q_i meet the conditions

$$\sum_{i=1}^{m} q_i g_\kappa(\mathbf{s}_i) = 0. \qquad [6]$$

The q_i are real-valued weights and $g_\kappa(\mathbf{s}_i) = s_{i_1}^{\rho_1} \ldots s_{i_n}^{\rho_n}$ are monomial products of the spatial coordinates in n dimensions. The exponents ρ_i are non-negative integers such that $\sum_{i=1}^{n} \rho_i \le \nu$. The subscript κ takes the values $\kappa = 1, 2, \ldots, \alpha(\nu)$ where

$$\alpha(\nu) = \binom{\nu+2}{\nu} = \frac{(\nu+1)(\nu+2)}{2} \qquad [7]$$

and assigns the number of possible permutations of $g_\kappa(\mathbf{s}_i)$ where the exponents $\rho_1, \rho_2, \ldots, \rho_n$ are varied. If the conditions [6] are met for an order of intrinsity ν, then $Y_q(\mathbf{s})$ is itself a zero-mean homogeneous SRF. That is, the algebraic combinations of the

appropriate q_i and the surrounding values $X(s_i)$ "filter out" the ν^{th} order trend in the original SRF.

Consider a two-dimensional ISRF-ν: $X(\mathbf{s}) = X(s_1, s_2) = X(x, y)$. Here, x and y are Cartesian coordinates. Equation [6] thus yields the set of monomials shown in Table 1.

Table1. Spatial Monomials of $g_\kappa(s_i)$.

$\nu = 0$	$\nu = 1$	$\nu = 2$
$\alpha(0) = 1$	$\alpha(1) = 3$	$\alpha(2) = 6$
$g_1(\mathbf{s}) = x^0 y^0 = 1$	$g_1(\mathbf{s}) = x^0 y^0 = 1$	$g_1(\mathbf{s}) = x^0 y^0 = 1$
	$g_2(\mathbf{s}) = x^1 y^0 = x$	$g_2(\mathbf{s}) = x^1 y^0 = x$
	$g_3(\mathbf{s}) = x^0 y^1 = y$	$g_3(\mathbf{s}) = x^0 y^1 = y$
		$g_4(\mathbf{s}) = x^2 y^0 = x^2$
		$g_5(\mathbf{s}) = x^0 y^2 = y^2$
		$g_6(\mathbf{s}) = x^1 y^1 = xy$

Therefore, $Y_q(\mathbf{s})$ is a SI-ν if the following conditions hold:

$$\text{if } \nu = 0, \text{ then } \sum_{i=1}^{m} q_i = 0;$$

$$\text{if } \nu = 1, \text{ then } \sum_{i=1}^{m} q_i = 0, \ \sum_{i=1}^{m} q_i x_i = 0, \ \sum_{i=1}^{m} q_i y_i = 0;$$

$$\text{if } \nu = 2, \text{ then } \sum_{i=1}^{m} q_i = 0, \ \sum_{i=1}^{m} q_i x_i = 0, \ \sum_{i=1}^{m} q_i y_i = 0, \ \sum_{i=1}^{m} q_i x_i^2 = 0, \ \sum_{i=1}^{m} q_i y_i^2 = 0, \ \sum_{i=1}^{m} q_i x_i y_i = 0. \quad [8]$$

An example is illustrated in Figure 3. Five points are indicated in $\mathbf{R}^2$ with point $\mathbf{s}_1$ centered at (x_0, y_0). Assume that weights q_i are associated with each of these points so that the spatial increment at $\mathbf{s}_1$ is given by

$$Y_q(\mathbf{s}_1) = 4X(\mathbf{s}_1) - 1X(\mathbf{s}_2) - 1X(\mathbf{s}_3) - 1X(\mathbf{s}_4) - 1X(\mathbf{s}_5). \quad [9]$$

The highest order of intrinsity for which the conditions of Equation [6] are met is ν=1. The conditions break down for the second-order terms. Therefore, Equation [9] is a SI-1.

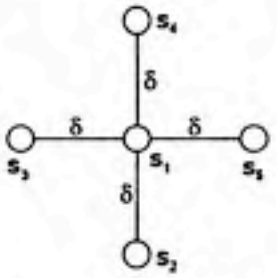


Figure 3. Spatial increment of order 1.

Intrinsic Spatial Random Field of Order ν

Having defined the generalized spatial increments of $X(\mathbf{s})$, we may now state the following definition: An *intrinsic spatial random field of order* ν is a random field for which the SI-ν are weakly homogeneous. The following definitions supplement the above.

- By convention, a homogeneous SRF is an ISRF-(-1).
- The spatial increments of any SRF which is already homogeneous are also homogeneous. A homogeneous SRF is also an ISRF-ν for all values of ν. Additionally, any ISRF-η is also an ISRF-ζ where $\zeta > \eta$, although the reverse is not generally true. For example, if an ISRF has homogeneous increments of order 2, it will also have homogeneous increments of orders 3, 4, et cetera.

Generalized Spatial Covariance of Order ν

The spatial correlations of natural processes which are modeled as SRFs are quantitatively defined by the covariance function. Within the modeling framework, the covariance describes the correlation in SRF behavior between two points in space. One of the integral components of the ISRF theory is that the ordinary covariance for an ISRF satisfies

$$c_X(\mathbf{s}_i,\mathbf{s}_j) = k_X(\mathbf{r}_{ij}) + p^{\nu}(\mathbf{s}_i,\mathbf{s}_j) \qquad [10]$$

where $k_X(\mathbf{r}_{ij})$, $\mathbf{r}_{ij} = \mathbf{s}_i - \mathbf{s}_j$ is the *generalized spatial covariance of order* ν (GSC-ν) and $p^{\nu}(\mathbf{s}_i,\mathbf{s}_j)$ is a νth-order polynomial in s with variable coefficients. This decomposition principle is an important theoretical construct. Under the intrinsic hypothesis, the conditions [6] placed on the spatial increment filter out the polynomial component of the ordinary covariance in the spatial increment, eliminating the need to quantify this component. Additionally, the ordinary covariance of the homogeneous $Y_q(\mathbf{s})$ is related to the GSC-ν of the non-homogeneous $X(\mathbf{s})$ by the expression

$$c_Y(\mathbf{r}_{ij}) = \sum_{\alpha=1}^{m}\sum_{\beta=1}^{m} q_{i\alpha}q_{j\beta}k_X(\mathbf{r}_{\alpha\beta}) \qquad [11]$$

where the weights q_i take values subject to constraints [6].

Section 4.0
Intrinsic Kriging System

As mentioned in Section 2.0, a SRF is represented by the function $X(\xi,\mathbf{s})$, where ξ is a random component accounting for fluctuations in X at a smaller scale and $\mathbf{s}$ is the spatial vector component accounting for the macro-evolution of the process on a larger scale. The problem we will solve is to calculate estimates of the SRF using the information provided by the measurements χ_i of the physical process at the locations $\mathbf{s}_i$, where i=1,2,...,m and represents the local sample set used for estimation. In creating an optimal estimate $\hat{X}(\mathbf{s})$ at some location $\mathbf{s}_k$, we want to satisfy three conditions.

- The estimate should be a weighted linear combination of the available data. Thus, the point estimate is given by

$$\hat{X}(\mathbf{s}_k)=\sum_{i=1}^{m}\lambda_i X(\mathbf{s}_i) \qquad [12]$$

  where the λ_i are real-valued kriging weights determined during the estimation process.

- The estimator $\hat{X}(\mathbf{s}_i)$ should possess the property of unbiasedness such that the expected value of the estimation error $\varepsilon=\hat{X}(\mathbf{s}_k)-X(\mathbf{s}_k)$ is zero:

$$E\left[\hat{X}(\mathbf{s}_k)-X(\mathbf{s}_k)\right]=0. \qquad [13]$$

- Finally, the variance of the estimation error $\sigma_\varepsilon^2(\mathbf{s}_k)$ should be minimized with respect to the kriging weights $\{\lambda_i,\ i=1,2,...,m\}$ to provide an optimal estimate.

These conditions are used to develop the system of kriging equations. We begin by linking the kriging estimator to the ISRF-ν theory. This linkage is accomplished through the kriging weights. To derive a solution to the unknown weights λ_i, we minimize the estimation error variance under the same constraints used to define the SI-ν,

namely those given by [6]. In other words, we want to define a SI-ν, $Y_q(\mathbf{s}_k)$, where the kriging weights λ_i are valid coefficients q_i such that the constraints

$$\sum_{i=1}^{m,k} \lambda_i g_\kappa(\mathbf{s}_i) = 0 \qquad [14]$$

are met with $\lambda_k = -1$. In this case the constraints may also be written as

$$\sum_{i=1}^{m} \lambda_i g_\kappa(\mathbf{s}_i) = g_\kappa(\mathbf{s}_k). \qquad [15]$$

When the λ_i are valid SI-ν coefficients, Equation [5] may be rewritten as

$$Y_q(\mathbf{s}_k) = \sum_{i=1}^{m,k} \lambda_i X(\mathbf{s}_i)\ , \qquad [16]$$

and recalling $\lambda_k = -1$ and the form of the linear estimator given by [12] we can write

$$Y_q(\mathbf{s}_k) = \sum_{i=1}^{m} \lambda_i X(\mathbf{s}_i) - X(\mathbf{s}_k) = \hat{X}(\mathbf{s}_k) - X(\mathbf{s}_k). \qquad [17]$$

Note that the latter is the expression for estimation error $\varepsilon(\mathbf{s}_k)$ so that under these conditions the estimation error is a valid SI-ν.

Now we must develop an expression for the estimation error variance

$$\sigma_\varepsilon^2(\mathbf{s}_k) = E\left[(\varepsilon - m_\varepsilon)^2\right] \qquad [18]$$

where m_ε is the mean value of the estimation errors. Applying the condition of unbiasedness ($m_\varepsilon = 0$), we get

$$\sigma_\varepsilon^2(\mathbf{s}_k) = E\left[(\varepsilon)^2\right] = E\left[\left(\hat{X}(\mathbf{s}_k) - X(\mathbf{s}_k)\right)^2\right]. \qquad [19]$$

Substituting the linear estimator from the first condition above, this equation becomes

$$\sigma_\varepsilon^2(\mathbf{s}_k) = E\left[\left(\sum_{i=1}^{m} \lambda_i X(\mathbf{s}_i) - X(\mathbf{s}_k)\right)^2\right]$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{m} \lambda_i \lambda_j E\left[X(\mathbf{s}_i)X(\mathbf{s}_j)\right] - 2\sum_{i=1}^{m} \lambda_i E\left[X(\mathbf{s}_i)X(\mathbf{s}_k)\right] + E\left[\left(X(\mathbf{s}_k)\right)^2\right] \qquad [20]$$

where $\lambda_k = -1$. It is important to note that, since $\varepsilon = Y_q(\mathbf{s}_k)$, this expression for estimation error variance given by [20] is equivalent to the variance for the SI-ν, which is by definition a zero-mean homogeneous SRF. The link between ISRF-ν theory and the kriging estimator lies in this equivalence. The terms $E\left[X(\mathbf{s}_i)X(\mathbf{s}_j)\right]$ and $E\left[X(\mathbf{s}_i)X(\mathbf{s}_k)\right]$ in [20] are covariance expressions for a homogeneous zero-mean process. Under the intrinsic hypothesis these expressions are represented by the homogeneous component, the GSC-ν, of the ordinary covariance for a non-homogeneous SRF. Similarly, the term $E\left[\left(X(\mathbf{s}_k)\right)^2\right]$ is a variance expression for a homogeneous zero-mean process and is also represented by the GSC-ν at lag zero. With these observations we can rewrite [20] as

$$\sigma_\varepsilon^2(\mathbf{s}_k) = \sum_{i=1}^{m}\sum_{j=1}^{m}\lambda_i\lambda_j k_X(r_{ij}) - 2\sum_{i=1}^{m}\lambda_i k_X(r_{ik}) + k_X(0). \qquad [21]$$

Now we have an expression for error variance in terms of kriging weights and the generalized covariance. The object is to minimize this expression with respect to the weights by taking the partial derivatives and setting them to zero:

$$\frac{\partial\sigma_\varepsilon^2(\mathbf{s}_k)}{\partial\lambda_i} = 0\ ,\ \forall i. \qquad [22]$$

Doing this under the constraints of [15] yields the kriging equations:

$$\mathbf{K}\mathbf{w} = \mathbf{k} \qquad [23]$$

where

$$\mathbf{K}=\begin{bmatrix} k_X(r_{11}) & k_X(r_{12}) & \cdots & k_X(r_{1n}) & g_1(\mathbf{s}_1) & g_2(\mathbf{s}_1) & \cdots & g_\alpha(\mathbf{s}_1) \\ k_X(r_{21}) & k_X(r_{22}) & \cdots & k_X(r_{2n}) & g_1(\mathbf{s}_2) & g_2(\mathbf{s}_2) & \cdots & g_\alpha(\mathbf{s}_2) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ k_X(r_{n1}) & k_X(r_{n2}) & \cdots & k_X(r_{nn}) & g_1(\mathbf{s}_n) & g_2(\mathbf{s}_n) & \cdots & g_\alpha(\mathbf{s}_n) \\ g_1(\mathbf{s}_1) & g_1(\mathbf{s}_2) & \cdots & g_1(\mathbf{s}_n) & 0 & 0 & \cdots & 0 \\ g_2(\mathbf{s}_1) & g_2(\mathbf{s}_2) & \cdots & g_2(\mathbf{s}_n) & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ g_\alpha(\mathbf{s}_1) & g_\alpha(\mathbf{s}_2) & \cdots & g_\alpha(\mathbf{s}_n) & 0 & 0 & \cdots & 0 \end{bmatrix}, \quad \mathbf{w}=\begin{Bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \\ \mu_0 \\ \mu_1 \\ \vdots \\ \mu_\alpha \end{Bmatrix}, \quad \mathbf{k}=\begin{Bmatrix} k_X(r_{1k}) \\ k_X(r_{2k}) \\ \vdots \\ k_X(r_{nk}) \\ g_1(\mathbf{s}_k) \\ g_2(\mathbf{s}_k) \\ \vdots \\ g_\alpha(\mathbf{s}_k) \end{Bmatrix}.$$

In this system, $k_X(r_{ij})$ is the GSC-ν, and the argument r_{ij} is the isotropic lag:

$$r_{ij}=|\mathbf{s}_i-\mathbf{s}_j|=\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}\,. \qquad [24]$$

Note that $r_{ij}=r_{ji}$ so that the matrix **K** is symmetric; additionally, $r_{ii}=0$. The $g_\kappa(\mathbf{s}_i)$ are monomials of the spatial coordinates $\mathbf{s}_i$ and are equivalent to the monomials which define the SI-ν of $X(\mathbf{s})$. Here κ takes the values $\kappa=1,2,\ldots,\alpha$ with $\alpha=(\nu+1)(\nu+2)/2$. For example, the case of ν=1 would yield $g_1(\mathbf{s}_i)=x_i^0y_i^0=1$, $g_2(\mathbf{s}_i)=x_i^1y_i^0=x_i$, and $g_3(\mathbf{s}_i)=x_i^0y_i^1=y_i$. The λ_i are the kriging weights, and the μ_κ $(\kappa=1,2,\ldots,\alpha)$ are Lagrange multipliers which arise from the minimization of the error variance under the constraints of [15].

The solution to the optimal kriging weights is given by $\mathbf{w}=\mathbf{K}^{-1}\mathbf{k}$ and provides values for the weights used to determine $\hat{X}(\mathbf{s}_k)$. The weights are also used to determine $\sigma_\varepsilon^2(\mathbf{s}_k)$, which provides an indication of the accuracy of the estimate. Equation [21] may be used, although a more useful expression is provided by

$$\sigma_\varepsilon^2(\mathbf{s}_k)=k_X(0)-\mathbf{w}^{\mathrm{T}}\mathbf{k}. \qquad [25]$$

Note that the kriging system [23] does not depend on the data values $X(\mathbf{s}_i)$ but only their spatial locations. This property is termed *data independence* and is one of the advantages of kriging estimators.

Section 5.0
Intrinsic Kriging Algorithm

Now that the details of the mathematical theory have been addressed in previous sections, we will move on to a description of the algorithm used to implement kriging under the intrinsic hypothesis. The intrinsic kriging algorithm described here is also introduced in Delfiner (1976). However, a number of modifications have been made from the Delfiner algorithm, and these will be included in the following discussion.

Data Preparation

The first step in the process is data preparation. The point estimation method uses spatial data located in $\mathbf{R}^2$, so a data file should consist of some number of measured values each consisting of an X-coordinate, a Y-coordinate, and a parameter measurement. The set of measurements should be checked, and duplicate or near-duplicate points should be declustered to prevent possible numerical problems with closely spaced points.

Kriging Grid

In short, the method of kriging takes a set of spatial measurements and calculates an estimate of the measured parameter at a location $s_k = (x_k, y_k)$ where no measurement was taken. In a typical analysis, this estimation is conducted over a large number of spatial points to provide a discretized representation of the estimates. If a graphical portrait of the estimated process is desired, then the usual approach is to define a rectangular grid of nodes--the kriging grid--and conduct the estimation calculations at each node. The results can then be used in a graphics routine to visualize the surface of the estimates and the estimation error variances.

In defining the grid system, it is important to remember that kriging methods are linear interpolators. Therefore, an attempt should be made to define the grid system such

that the boundaries do not lie too far outside the group of data, and it is preferable to keep the grid system completely within the range of the data.

Neighborhood Selection

In this work the analysis of spatial trend, the determination of a covariance model, and the calculations of the kriging estimate and error variance are all conducted on a local scale. These tasks are carried out using localized subsets of the complete data set (Figure 4). The subsets are referred to as neighborhoods, and $\hat{X}(\mathbf{s}_k)$ and $\sigma_e^2(\mathbf{s}_k)$ are based on the SRF properties exhibited by the local neighborhoods surrounding each grid node.

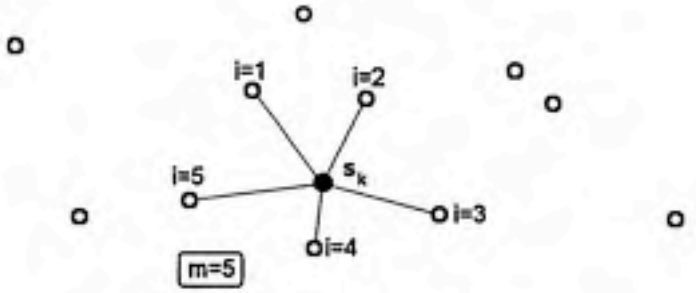


Figure 4. Local subset of five data.

When defining a neighborhood, one of several criteria may be followed. In one case, the neighborhood could be defined by all the data points that lie within a given fixed radius, *r*. Since the generalized covariances considered in this work are isotropic, the vector direction of the points chosen is irrelevant; thus, all points within the scalar radius *r* are suitable for the neighborhood. This approach may be appropriate when the data are spaced uniformly throughout the domain under consideration.

In a second case, the neighborhood could be defined by a fixed number *m* of closest data points. Again the use of an isotropic generalized covariance allows the

selection of nearest neighbors regardless of vector location. This approach may be appropriate when the data occur more densely in some areas of the domain than in others.

In this work, we have implemented the criterion of a fixed number m of nearest neighbors; thus, the m closest points constitute the neighborhood. For the case of intrinsic kriging, neighborhood sizes of 8-16 data points are usually adequate.

The kriging estimator is an exact interpolator, meaning the estimates are faithful to the data. Thus in the event that a kriging node $\mathbf{s}_k$ falls exactly on the location of a data point -- a condition refered to here as "bull's-eye" -- the estimate for that point will be equal to the data value. (This assumes there is no measurement error.) Intuitively, there is no estimation error in such a case, so the error variance is zero. These characteristics are accounted for in the kriging algorithm. For each kriging node, a check is performed to determine if the node coincides with a data point. If it does, $\hat{X}(\mathbf{s}_k)$ is given the value of that data point, $\sigma_e^2(\mathbf{s}_k)$ becomes zero, and the estimation procedure is bypassed. However, we are still interested to see what the order of intrinsity and covariance parameters are for this node. These portions of the analysis are still conducted, although when defining the neighborhood, the bull's-eye data point is excluded from the neighborhood. The neighborhood is, thus, based on the m closest data points located a distance $r > 0$ away from the node.

Local Order of Intrinsity ν

The next step in the algorithm is to determine the order of intrinsity which best fits the trend characteristics of the local neighborhood. This step is based on the accurate estimation of the data points themselves and, therefore, involves the assumption of a representative covariance model for the neighborhood. The objective is to chose the value of ν from the set {0,1,2} which best describes the trend in the data. Although ν can take on any non-negative integer value, the set {0,1,2} is usually sufficient to represent most local trends. The procedure is described in the following steps.

(1) Each data point $X(\mathbf{s}_i)$ in the neighborhood of m points is "removed" one at a time.

(2) The removed point $X(\mathbf{s}_i)$ is estimated from the remaining m-1 points assuming in turn that $\nu = 0$, 1, and 2. A linear covariance model $k_X(r) = -r$ is used to construct the kriging system [23] which is solved for the kriging weights λ_i, which are in turn used to calculate $\hat{X}(\mathbf{s}_i)_\nu$ by [12]. The linear covariance originates from the polynomial model $k_X(r) = a_0\delta(r) - c_0 r + c_1 r^3 - c_2 r^5$ where $a_0 = c_1 = c_2 = 0$ and $c_0 = 1$. Although the model is not optimized to the correlation structure of the neighborhood, it is a valid model for all values of ν.

(3) At each point i removed , the estimation error is determined for the three cases of ν = 0, 1, and 2 by the following

$$\varepsilon_{i,\nu} = \left| \hat{X}(\mathbf{s}_i)_\nu - X(\mathbf{s}_i) \right|. \qquad [26]$$

(4) For each i, the estimation errors $\varepsilon_{i,\nu}$ for $\nu = 0$, 1, and 2 are ranked as 1, 2, and 3 from smallest error to largest error, respectively. The rationale here is that the smallest error corresponds to the best estimate.

(5) The error ranks $\rho_{i,\nu}$ are then summed for each case of ν:

$$\Omega_{\nu=0} = \sum_{i=1}^{m} \rho_{i,\nu=0}, \quad \Omega_{\nu=1} = \sum_{i=1}^{m} \rho_{i,\nu=1}, \quad \Omega_{\nu=2} = \sum_{i=1}^{m} \rho_{i,\nu=2} \qquad [27]$$

where m is the number of points in the neighborhood.

(6) The ν-value corresponding to the smallest Ω_ν value is chosen as the order of intrinsity for the neighborhood. In the event that two Ω_ν values are equal and less than the third, the lower ν-value from the tied rank sums is taken as the order of intrinsity for the neighborhood. This response reflects intrinsic random field theory in that an ISRF-η is also an ISRF-ζ where $\zeta > \eta$, although the reverse is not generally true. Thus, when the above procedure produces two equally favorable possibilities for ν, the more conservative choice is to pick the lower ν-value.

Note that the ranks $\rho_{i,v}$ of the estimation errors $\varepsilon_{i,v}$ are used to determine v rather than the $\varepsilon_{i,v}$ themselves. Ranking makes the procedure resistant against the bias of outlier estimation errors. At this stage, the order of intrinsity of the neighborhood has been assigned. The next phase will identify the covariance model which best represents the correlation structure of the neighborhood.

Covariance Parameter Estimation

The fitting of a covariance model to the neighborhood of data constitutes the most critical step in the estimation procedure. The GSC function mathematically defines the correlation structure of the SRF and, as is evident from inspection of the kriging equation [23], it is the platform upon which the kriging estimations are built. It is also the most numerically intensive component of the computer program, since it requires the optimization of a nonlinear function.

Three different GSC-v models have been implemented in this work: polynomial, polynomial-spline, and poly-exponential models. Each model contains unknown coefficients whose values are determined based on the correlation structure of the neighborhood. The values of these coefficients must obey permissibility conditions which are assigned to ensure the GSC is conditionally positive definite, a requirement for generalized covariance functions (Christakos, 1984). In this sense, it leads to legitimate second moments of the generalized spatial increments. Permissibility values are obtained through spectral analysis and are unique to a particular GSC model. The objective of the covariance modeling portion of the algorithm is to calculate the optimal values for these coefficients subject to the constraints of permissibility. A least squares method which minimizes -- with respect to the coefficients -- the sum of the squared differences between the expected values of the estimation errors and their actual values is applied. Although the least squares method used here is a suitable approach, several other optimization methods may be employed for the inference. These include maximum likelihood

estimation and minimum variance unbiased quadratic estimation (Kitanidis, 1983). Since the procedures for determining a suitable covariance function are different for the different general forms, they are discussed separately.

Polynomial Covariance Model

The polynomial generalized covariance function is perhaps the most common GSC-ν used with intrinsic kriging estimators and is frequently indicated in the literature (e.g. Delfiner, 1976; Kitanidis, 1983; Christakos, 1992). The general form of this function is given by

$$k_X(r) = a_0\delta(r) + \sum_{i=0}^{\nu}(-1)^{i+1}c_i r^{2i+1} \qquad [28]$$

where a_0 and c_i are unknown constants, r is the lag between two points, and $\delta(r)$ is Kronecker's delta function: $\delta(r) = 1$ for $r = 0$, and $\delta(r) = 0$ for $r > 0$. Note that the polynomial degree and number of terms in the function depend on the given order of intrinsity, ν. The term $a_0\,\delta(r)$ describes the nugget effect. Additionally, [28] is linear in its parameters a_0 and c_i, a condition which facilitates inference. Practical experience with the intrinsic hypothesis has shown that, in most cases, values of ν from the set {0, 1, 2} will suffice for characterizing non-homogeneity. The values of the unknown coefficients are restricted to the permissibility conditions listed in Table 2. These conditions vary with the number of spatial dimensions of the data set. Note that this model is isotropic in $\mathbf{R}^n$ so that r is scalar. Since its coefficients can take values of zero, the general equation [28] submits 3, 7, and 15 distinct covariance functions for ν-values of 0, 1, and 2, respectively, as shown in Table 3. The form $k_X(r) = a_0\delta(r)$ represents a completely random process (white noise or pure nugget effect) with no correlation in behavior between different points in space. All other forms represent a process with significant correlation.

Table 2. Permissibility conditions for polynomial GSC-ν coefficients.

$\nu=0:$	$a_0 \geq 0$	$c_0 \geq 0$		
$\nu=1:$	$a_0 \geq 0$	$c_0 \geq 0$	$c_1 \geq 0$	
$\nu=2:$	$a_0 \geq 0$	$c_0 \geq 0$	$c_1 \geq -\sqrt{20c_0c_2(n+3)/3(n+1)}$	$c_2 \geq 0$

Table 3. Polynomial covariance models.

ν=0	ν=2
$k_x(r)=a_0\delta(r)-c_0r$	$k_x(r)=a_0\delta(r)-c_0r+c_1r^3-c_2r^5$
$k_x(r)=a_0\delta(r)$	$k_x(r)=a_0\delta(r)-c_0r+c_1r^3$
$k_x(r)=-c_0r$	$k_x(r)=a_0\delta(r)-c_0r-c_2r^5$
	$k_x(r)=a_0\delta(r)+c_1r^3-c_2r^5$
	$k_x(r)=-c_0r+c_1r^3-c_2r^5$
ν=1	$k_x(r)=a_0\delta(r)-c_0r$
$k_x(r)=a_0\delta(r)-c_0r+c_1r^3$	$k_x(r)=a_0\delta(r)+c_1r^3$
$k_x(r)=a_0\delta(r)-c_0r$	$k_x(r)=a_0\delta(r)-c_2r^5$
$k_x(r)=a_0\delta(r)+c_1r^3$	$k_x(r)=-c_0r+c_1r^3$
$k_x(r)=-c_0r+c_1r^3$	$k_x(r)=-c_0r-c_2r^5$
$k_x(r)=a_0\delta(r)$	$k_x(r)=c_1r^3-c_2r^5$
$k_x(r)=-c_0r$	$k_x(r)=a_0\delta(r)$
$k_x(r)=c_1r^3$	$k_x(r)=-c_0r$
	$k_x(r)=c_1r^3$
	$k_x(r)=-c_2r^5$

The method for finding a suitable GSC-ν begins with a known value of ν. The coefficients for each of the possible forms are determined and tested for permissibility. If any coefficient value does not pass the permissibility test, that form is not considered any further. Those forms which pass are tested for goodness of fit. The passing form which provides the best fit is then used for the estimation and error variance calculations.

To conduct the least squares routine which optimizes the values of the coefficients for a given GSC-ν form, we must first define several functions. The SI-ν defined previously as [16] is now written for a data point removed from a local neighborhood:

$$Y_q(\mathbf{s}_i) = \sum_{j=1}^{m} \lambda_{ij} X(\mathbf{s}_j) \qquad [29]$$

where m is the number of points in the neighborhood, i refers to the index of the point currently removed from the neighborhood, and λ_{ij} are the kriging weights associated with the remaining points. Note that $\lambda_{ii} = -1$ and that $Y_q(\mathbf{s}_i)$ is the estimation error of point i since

$$Y_q(\mathbf{s}_i) = \hat{X}(\mathbf{s}_i) - X(\mathbf{s}_i). \qquad [30]$$

Next we define the expected value of the squared estimation error for point i removed:

$$E\left[\left(\hat{X}(\mathbf{s}_i) - X(\mathbf{s}_i)\right)^2\right]. \qquad [31]$$

Substituting [12] for the estimate $\hat{X}(\mathbf{s}_i)$ into [30] and subsequently substituting this into expression [31], we obtain

$$E\left[Y_q(\mathbf{s}_i)^2\right] = \sum_{\alpha=1}^{m}\sum_{\beta=1}^{m} \lambda_{i\alpha}\lambda_{i\beta} E\left[X(\mathbf{s}_\alpha)X(\mathbf{s}_\beta)\right] \qquad [32]$$

where $\lambda_{ii} = -1$, and $E\left[X(\mathbf{s}_\alpha)X(\mathbf{s}_\beta)\right]$ is the covariance for a zero-mean homogeneous SRF -- the generalized spatial covariance. The resulting equation is

$$A_i = \sum_{\alpha=1}^{m}\sum_{\beta=1}^{m} \lambda_{i\alpha}\lambda_{i\beta} k_X(r_{\alpha\beta}) \qquad [33]$$

where $\lambda_{i\alpha}$ and $\lambda_{i\beta}$ are kriging weights and $r_{\alpha\beta}$ is the scalar distance between data points $\mathbf{s}_\alpha$ and $\mathbf{s}_\beta$. Now an objective function to be minimized with respect to the coefficients may be defined:

$$F = \sum_{i=1}^{m}\left[Y_q(\mathbf{s}_i)^2 - A_i\right]^2. \qquad [34]$$

The function F applies to a given neighborhood with m data points and is the sum of the squared differences between the actual estimation error squared and the expected estimation error squared.

Note that through the $k_X(r_{\alpha\beta})$ term in A_l, F depends on the values of the coefficients a_0, c_0, c_1, and c_2. The objective function is then minimized through taking partial derivatives with respect to the coefficients and setting them to zero:

$$\frac{\partial F}{\partial a_0} = \frac{\partial F}{\partial c_0} = \frac{\partial F}{\partial c_1} = \frac{\partial F}{\partial c_2} = 0. \qquad [35]$$

This operation yields the following set of equations:

from $\frac{\partial F}{\partial a_0} = 0$, [36]

$$a_0 \sum_{i=1}^{m} \left(\Lambda_i^{(0)}\right)^2 - c_0 \sum_{i=1}^{m} \Lambda_i^{(0)} \Lambda_i^{(1)} + c_1 \sum_{i=1}^{m} \Lambda_i^{(0)} \Lambda_i^{(3)} - c_2 \sum_{i=1}^{m} \Lambda_i^{(0)} \Lambda_i^{(5)} = \sum_{i=1}^{m} Y_q(\mathbf{s}_i)^2 \Lambda_i^{(0)};$$

from $\frac{\partial F}{\partial c_0} = 0$,

$$a_0 \sum_{i=1}^{m} \Lambda_i^{(1)} \Lambda_i^{(0)} - c_0 \sum_{i=1}^{m} \left(\Lambda_i^{(1)}\right)^2 + c_1 \sum_{i=1}^{m} \Lambda_i^{(1)} \Lambda_i^{(3)} - c_2 \sum_{i=1}^{m} \Lambda_i^{(1)} \Lambda_i^{(5)} = \sum_{i=1}^{m} Y_q(\mathbf{s}_i)^2 \Lambda_i^{(1)};$$

from $\frac{\partial F}{\partial c_1} = 0$,

$$a_0 \sum_{i=1}^{m} \Lambda_i^{(3)} \Lambda_i^{(0)} - c_0 \sum_{i=1}^{m} \Lambda_i^{(3)} \Lambda_i^{(1)} + c_1 \sum_{i=1}^{m} \left(\Lambda_i^{(3)}\right)^2 - c_2 \sum_{i=1}^{m} \Lambda_i^{(3)} \Lambda_i^{(5)} = \sum_{i=1}^{m} Y_q(\mathbf{s}_i)^2 \Lambda_i^{(3)};$$

from $\frac{\partial F}{\partial c_2} = 0$,

$$a_0 \sum_{i=1}^{m} \Lambda_i^{(5)} \Lambda_i^{(0)} - c_0 \sum_{i=1}^{m} \Lambda_i^{(5)} \Lambda_i^{(1)} + c_1 \sum_{i=1}^{m} \Lambda_i^{(5)} \Lambda_i^{(3)} - c_2 \sum_{i=1}^{m} \left(\Lambda_i^{(5)}\right)^2 = \sum_{i=1}^{m} Y_q(\mathbf{s}_i)^2 \Lambda_i^{(5)}.$$

As they appear in the above equations, the following are defined:

$$\Lambda_i^{(0)} = \sum_{j=1}^{m} \lambda_{ij}^2 \qquad \Lambda_i^{(3)} = \sum_{\alpha=1}^{m}\sum_{\beta=1}^{m} \lambda_{i\alpha}\lambda_{i\beta} r_{\alpha\beta}^3$$

$$\Lambda_i^{(1)} = \sum_{\alpha=1}^{m}\sum_{\beta=1}^{m} \lambda_{i\alpha}\lambda_{i\beta} r_{\alpha\beta} \qquad \Lambda_i^{(5)} = \sum_{\alpha=1}^{m}\sum_{\beta=1}^{m} \lambda_{i\alpha}\lambda_{i\beta} r_{\alpha\beta}^5$$

where the parenthetical superscripts in the left-hand-side terms are indices which identify the different functions.

Given a set of kriging weights $\{\lambda_{ij}\}$ for a neighborhood, [36] are solved simultaneously as a system of linear equations for the unknown coefficients a_0, c_0, c_1, and c_2. Because F is non-linearly dependent on the covariance coefficients, the minimization of F is an iterative process. For a given value of ν, the optimization of F is conducted for each of the GSC-ν models shown in Table 3. The process is described in the following steps.

(1) Since [36] require a set of weights $\{\lambda_{ij}\}$ for solution, and since a GSC-ν function is required for the solution of these weights from the kriging system [23], an initial guess for the GSC-ν is required to start the iterative solution to $\{a_0, c_0, c_1, c_2\}^{(l+1)}$ where l is an iteration counter referring to the current iteration. In this algorithm, we chose an initial GSC-ν function of

$$k_X^{(l=0)}(r) = -r \qquad [37]$$

since it is a permissible function for all values of ν. Note that this form is used only to initiate the iteration cycle. After this initial guess, one of the forms from Table 3 is used.

(2) Given a covariance model from Table 3 (or the initial GSC if l=0) and its most recent coefficient set $\{a_0, c_0, c_1, c_2\}^{(l)}$, each point $X(s_i)$ in the neighborhood is removed one at a time and the kriging system [23] is constructed using the remaining points as the

neighborhood and $X(\mathbf{s}_i)$ as the point to be estimated. Equation [23] is solved for the set $\{\lambda_{ij};\ j=1,2,\ldots,m;\ j\neq i\}^{(l)}$. When all the neighborhood points have been removed and the weights calculated, they may be represented by the following matrix:

$$\left[\begin{array}{c|cccc} & j=1 & j=2 & \cdots & j=m \\ \hline i=1 & -1 & \lambda_{12} & \cdots & \lambda_{1m} \\ i=2 & \lambda_{21} & -1 & \cdots & \lambda_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ i=m & \lambda_{m1} & \lambda_{m2} & \cdots & -1 \end{array}\right].$$

Note that in the above matrix, the weights λ_{ii} are assigned values of -1.

(3) With $\{\lambda_{ij}\}^{(l)}$ known, equations [36] are constructed and solved simultaneously for $\{a_0, c_0, c_1, c_2\}^{(l+1)}$.

(4) The permissibility of the solution set $\{a_0, c_0, c_1, c_2\}^{(l+1)}$ is verified with the conditions of Table 2. If the solution is not permissible, the current GSC model is no longer considered and the cycle begins at step (1) with another covariance model from Table 3. If the solution is permissible, the cycle proceeds to the next step.

(5) Coefficient solutions which pass the permissibility test are checked for convergence at each iteration. A percent relative difference criterion is used. The values

$$\rho_1 = \left|\frac{a_0^{(l+1)} - a_0^{(l)}}{a_0^{(l+1)}}\right| \qquad \rho_2 = \left|\frac{c_0^{(l+1)} - c_0^{(l)}}{c_0^{(l+1)}}\right|$$

$$\rho_3 = \left|\frac{c_1^{(l+1)} - c_1^{(l)}}{c_1^{(l+1)}}\right| \qquad \rho_4 = \left|\frac{c_2^{(l+1)} - c_2^{(l)}}{c_2^{(l+1)}}\right| \qquad [38]$$

are calculated, and the largest of the four is compared against a maximum allowable error criterion (e.g. $\rho_{max} = 10^{-5}$).

(6) If $\rho_{largest} \leq \rho_{max}$ the solution has converged, and the iterative cycle is broken. The next model of the GSC-ν from Table 3 is used, and the process restarts with step (1).

If $\rho_{largest} > \rho_{max}$ the solution has not converged and another iteration is required. The solution set is updated so that $\{a_0, c_0, c_1, c_2\}^{(l+1)}$ is used for $\{a_0, c_0, c_1, c_2\}^{(l)}$ in the next iteration beginning with step (2).

Once all the appropriate models have been considered, those which provide permissible solutions must be compared against one another to determine which one best represents the correlation structure of the neighborhood. To this end, we use a goodness-of-fit indicator defined by

$$\eta_\mu = \frac{\sum_{i=1}^{m} Y_q(\mathbf{s}_i)^2}{\sum_{i=1}^{m} A_i} \qquad [39]$$

where μ is an integer index identifying a particular covariance form optimized for the curret neighborhood. Note that η represents the ratio of the *actual* squared SI-ν values to the *expected* squared SI-ν values, and the case $\eta = 1$ indicates an exact least-squares fit of the covariance to the correlation structure of the neighborhood. Thus, η_μ is calculated for all GSC-ν_μ, and the covariance which produces the value closest to 1 is chosen as the model with the best fit. This covariance is used in the subsequent estimation and error variance calculations. Of course, in the event only one of the possible covariance forms results in a permissible GSC-ν, the goodness-of-fit test is unnecessary.

Polynomial-Spline Model

The polynomial-spline model is another common covariance (de Marsily, 1986; Cressie, 1987) and is similar to the polynomial model but with a smoothing spline term. In general, the function is

$$k_X(r) = a_0\delta(r) - c_0 r + c_1 r^3 + c_s r^2 \ln r \qquad [40]$$

where a_0, c_0, c_1 and c_s are unknown constants. Table 4 lists the permissibility conditions on the values of these coefficients for the case of $\mathbf{R}^2$. Because of the logarithm term, the spline model possesses smoothing properties not present in the polynomial model. Like the polynomial model, the unknown coefficients can take values of zero. Thus, there are several covariance forms for each value of ν. For the case of $\nu = 0$, only the first two terms are valid so that the polynomial-spline model is identical to the polynomial model; the appropriate forms are given in Table 3. For the case $\nu \geq 1$, all four terms are valid; thus, there are 15 possible permutations of the covariance form.

Table 4. Permissibility conditions for polynomial-spline GSC-ν coefficients in. $\mathbf{R}^2$

$\nu = 0$:	$a_0 \geq 0$	$c_0 \geq 0$		
$\nu = 1, 2$:	$a_0 \geq 0$	$c_0 \geq 0$	$c_1 \geq 0$	$c_s \geq -1.5\sqrt{c_0 c_1}$

The procedure for stochastic inference of the coefficient values is the same as for the polynomial model with several minor differences. The same objective function [34] used for the polynomial model is used for the polynomial-spline model; however, the c_2 term is not present and is substituted by the c_s term. Thus, the partial derivative

$$\frac{\partial F}{\partial c_s} = 0 \qquad [41]$$

replaces the $\frac{\partial F}{\partial c_2}$ term in [36] with the following:

$$a_0\sum_{i=1}^{m}\Lambda_i^{(s)}\Lambda_i^{(0)}-c_0\sum_{i=1}^{m}\Lambda_i^{(s)}\Lambda_i^{(1)}+c_1\sum_{i=1}^{m}\Lambda_i^{(s)}\Lambda_i^{(3)}+c_2\sum_{i=1}^{m}\left(\Lambda_i^{(s)}\right)^2=\sum_{i=1}^{m}Y_q(s_i)^2\Lambda_i^{(s)} \qquad [42]$$

where

$$\Lambda_i^{(s)}=\sum_{\alpha=1}^{m}\sum_{\beta=1}^{m}\lambda_{i\alpha}\lambda_{i\beta}r_{\alpha\beta}^2\ln r_{\alpha\beta}.$$

The steps to determine the coefficient values for the polynomial model are also used for the polynomial-spline model. The only differences are those noted above.

Poly-Exponential Covariance Model

The poly-exponential model was developed by Christakos (1992) and in general form is given by

$$k_{X,\nu}(r)=\frac{(-1)^{\nu+1}}{b^{2\nu+2}e^{br}}\left[1-e^{br}\sum_{i=0}^{2\nu+1}\frac{(-br)^i}{i!}\right], \quad b>0 \qquad [43]$$

where b is an unknown constant taking real values greater than zero, and r is the scalar distance between the two points under consideration. Unlike the previous covariances, the poly-exponential model has one unknown coefficient and does not possess a nugget term (although a nugget term could be included if desired). Additionally, there is only one possible form for each value of ν. The specific forms are shown as follows:

$$\text{for } \nu=0, \quad k_{X,0}(r)=\frac{-1}{b^2e^{br}}+\frac{1}{b^2}-\frac{r}{b}; \qquad [44]$$

$$\text{for } \nu=1, \quad k_{X,1}(r)=\frac{1}{b^4e^{br}}-\frac{1}{b^4}+\frac{r}{b^3}-\frac{r^2}{2b^2}+\frac{r^3}{6b}; \qquad [45]$$

$$\text{for } \nu=2, \quad k_{X,2}(r)=\frac{-1}{b^6e^{br}}+\frac{1}{b^6}-\frac{r}{b^5}+\frac{r^2}{2b^4}-\frac{r^3}{6b^3}+\frac{r^4}{24b^2}-\frac{r^5}{120b}. \qquad [46]$$

Given a neighborhood of points, the method for optimizing the value of b such that the GSC best represents the local correlation structure is similar to the method used for the polynomial and polynomial-spline covariance models. Equation [34] is again defined as an

objective function to be minimized with respect to b. Taking the derivative and equating to zero yields

$$\frac{dF}{db} = \sum_{i=1}^{m} \left[Y_q(\mathbf{s}_i)^2 - A_{i,\nu} \right] \frac{dA_{i,\nu}}{db} = 0 \qquad [47]$$

where i is the point removed and the subscript ν indicates that A_i depends on ν.

Newton-Raphson iteration is used to solve for b as a root to equation [47]. For this one variable case, the method is developed as follows. First, define the equation

$$f(b) = \frac{dF}{db} = 0 \qquad [48]$$

such that

$$f(b) = \sum_{i=1}^{m} \left[Y_q(\mathbf{s}_i)^2 - A_{i,\nu} \right] \frac{dA_{i,\nu}}{db}. \qquad [49]$$

The Newton-Raphson formula is then given by

$$b^{(l+1)} = b^{(l)} - \frac{f\left(b^{(l)}\right)}{f'\left(b^{(l)}\right)} \qquad [50]$$

where the superscript l is an iteration index and refers to the current iteration. Also

$$f'\left(b^{(l)}\right) = \frac{df}{db} = \sum_{i=1}^{m} \left[\frac{d^2 A_{i,\nu}}{db^2} \left(Y_q(\mathbf{s}_i) - A_{i,\nu} \right) - \left(\frac{dA_{i,\nu}}{db} \right)^2 \right] \qquad [51]$$

where

$$\frac{dA_{i,\nu}}{db} = \sum_{\alpha=1}^{m} \sum_{\beta=1}^{m} \lambda_{i\alpha} \lambda_{i\beta} \frac{d}{db} k_{X,\nu}\left(r_{\alpha\beta}\right) \quad \text{and} \quad \frac{d^2 A_{i,\nu}}{db^2} = \sum_{\alpha=1}^{m} \sum_{\beta=1}^{m} \lambda_{i\alpha} \lambda_{i\beta} \frac{d^2}{db^2} k_{X,\nu}\left(r_{\alpha\beta}\right).$$

The general form of the covariance first derivative is given by

$$\frac{dk_X}{db} = \frac{(-1)^{\nu+2}}{b^{2\nu+2}} \left[\frac{2\nu+2}{b} \left(e^{-br} - \sum_{i=0}^{2\nu+1} \frac{(-br)^i}{i!} \right) + \left(re^{-br} - \sum_{i=0}^{2\nu+1} \frac{ir^i}{i!} (-b)^{i-1} \right) \right], \qquad [52]$$

while the specific forms of the second derivatives are given for $\nu = 0$, 1, and 2 as follows:

$$\frac{d^2k_{X,\nu=0}}{db^2} = \frac{-6}{b^4e^{br}} + \frac{6}{b^4} - \frac{4r}{b^3e^{br}} - \frac{2r}{b^3} - \frac{r^2}{b^2e^{br}}; \quad [53]$$

$$\frac{d^2k_{X,\nu=1}}{db^2} = \frac{20}{b^6e^{br}} + \frac{8r}{b^5e^{br}} + \frac{r^2}{b^4e^{br}} - \frac{20}{b^6} + \frac{12r}{b^5} - \frac{3r^2}{b^4} + \frac{r^3}{3b^3}; \quad [54]$$

$$\frac{d^2k_{X,\nu=2}}{db^2} = \frac{-42}{b^8e^{br}} - \frac{12r}{b^7e^{br}} - \frac{r^2}{b^6e^{br}} + \frac{42}{b^8} - \frac{30r}{b^7} + \frac{10r^2}{b^6} - \frac{2r^3}{b^5} + \frac{r^4}{4b^4} - \frac{r^5}{60b^3}. \quad [55]$$

Note that A_i and its derivatives are functions of b since they contain the covariance function or one of its derivatives. For a given value of ν, the optimal solution of b is conducted as follows.

(1) Start with an initial guess $b^{(l=0)}$ arbitrarily chosen within the range of permissibility of b.

(2) Using the appropriate covariance model and its most recent coefficient $b^{(l)}$, each point $X(\mathbf{s}_i)$ in the neighborhood is removed one at a time and the kriging system [23] is constructed using the remaining points as the neighborhood and $X(\mathbf{s}_i)$ as the point to be estimated. Equation [23] is solved for the set $\{\lambda_{ij};\ j=1,2,\ldots,m;\ j\neq i\}^{(l)}$.

(3) Use $b^{(l)}$ and $\{\lambda_{ij}\}$ to solve the Newton-Raphson equation [50] for $b^{(l+1)}$. A globally convergent routine by Press et al, (1992) is impemented in the actual computer code. In order to enforce the permissibility condition that $b>0$, the solution of b is constrained to meet permissibility with the addition of a penalty function $p(b)$ to the objective function:

$$p(b) = \frac{10^{-12}}{b}. \quad [56]$$

(4) At each iteration, check $b^{(l+1)}$ for convergence by calculating the relative error

$$\rho = \left| \frac{b^{(l+1)} - b^{(l)}}{b^{(l+1)}} \right| \qquad [57]$$

and comparing against the maximum allowable error (e.g. $\rho_{max} = 10^{-7}$).

(5) If $\rho \leq \rho_{max}$ the solution has converged, and the iterative cycle is broken. The GSC has been determined and is ready for use in the estimation of the kriging node.

If $\rho > \rho_{max}$ the solution has not converged and another iteration is required. The solution set is updated so that $b^{(l+1)}$ is used for $b^{(l)}$ in the next iteration beginning with step (2).

If the number of iterations has reached the maximum allowable without converging, another initial guess may be required. However, this scenario is not likely with the globally convergent Newton solver.

At the conclusion of these steps, the GSC is ready for use in calculation of the unknown point estimate and estimation error variance.

Point Estimate and Error Variance

At this stage in the algorithm, the order of intrinsity which best characterizes the trend in the neighborhood has been determined, and a GSC-ν model has been optimized for the correlation structure of the neighborhood. We are ready to proceed with the actual kriging estimation at the unknown point $\mathbf{s}_k$. The estimation begins with the construction of the system [23] using ν and the optimized GSC and $\mathbf{s}_k$ as the point to be estimated. The system is solved for the unknown kriging weights λ_i. The weights are then used in conjunction with the neighboring data to calculate the point estimate by [12].

Subsequently, [25] is used to calculate the estimation error variance $\sigma_\varepsilon^2(s_k)$. With this step, the algorithm is completed and may restart with the next point to be estimated.

Summary and Computer Implementation

Figure 5 provides a flow chart which outlines the intrinsic kriging algorithm just discussed.

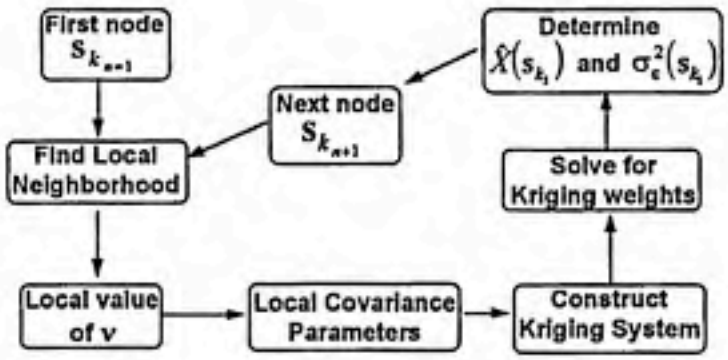


Figure 5. Intrinsic kriging algorithm.

Recall that a principle focus of this work was to develop an automated computer program to conduct this algorithm. The program structure follows directly from the algorithm described in this section. Because of the differences in numerical implementation of the three generalized covariances, three separate versions of the program have been written, and each version implements one covariance: polynomial, polynomial-spline, or poly-exponential. The polynomial and polynomial-spline versions are nearly identical, since these GSCs differ only in one term. However, the poly-exponential version is significantly different from the other two due to the uniqueness of this GSC function. All three programs are coded in FORTRAN 77. Commented hard-copies and tables of variable definitions from the polynomial and poly-exponential versions are contained in Appendices A and B respectively.

The algorithm described herein is well suited to parallel processing. The calculations of ν, covariance parameters, $\hat{X}(\mathbf{s}_k)$, and $\sigma_\varepsilon^2(\mathbf{s}_k)$ are specific to each node and depend only on the locations and values of the data points in the local neighborhood. Thus, the calculations at any node are completely independent of the calculations at every other node. If this algorithm was compiled for parallel processing and executed on a parallel machine, its processing speed could be significantly accelerated relative to the serial execution time.

Section 6.0
Application: Soil Moisture Content Data Set

General Information and Data Set

The automated computer program developed for the intrinsic kriging algorithm was applied with a set of soil moisture content data from a study area near Maricopa, Arizona. The site is located at the Maricopa Agricultural Center about 90 miles northwest of Tucson. The data consist of 75 soil moisture measurements collected in Spring of 1988 from a depth of 10 centimeters within a domain approximately 1500 meters long (East-West) and 250 meters wide (North-South). Figure 6 shows the sampling locations in plan view. Note there are three East-West transections of 15 data each and two areas of denser data clustering. Warrick et al (1990) designed and implemented the sampling plan as part of a project involving sampling strategies and geostatistical analysis of hydrologic properties in the upper vadose zone. The two areas of clustering were assigned on a quasi-random basis for the purpose of homogenizing lag class sizes for experimental variogram calculations. Each measurement consists of an X-Y coordinate location and the moisture content (ratio of weight of entrained water to weight of solids) of the extracted sample given in percent. Table 5 lists the measured values. Soil types in the area consist of the Casa Grande sandy clay loam and the Trix clay loam. We used the soil moisture data to calculate a set of estimations at gridded locations for graphical output as well as to conduct sensitivity analysis for key program parameters and to evaluate estimation error. Presently, we will discuss the initial set of gridded estimations.

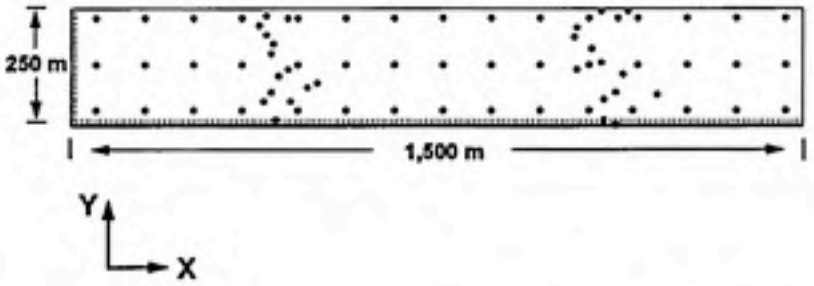


Figure 6. Locations of the 75 soil moisture content measurements, plan view. X is east, and Y is north.

Table 5. List of soil moisture content data.

Easting (meters)	Northing (meters)	Percent Moisture	Easting (meters)	Northing (meters)	Percent Moisture
50.0	50.0	17.2	865.0	50.0	17.2
50.0	150.0	21.9	865.0	150.0	20.1
50.0	250.0	24.6	865.0	250.0	21.5
150.0	50.0	18.0	965.0	50.0	20.1
150.0	150.0	23.6	965.0	150.0	20.2
150.0	250.0	24.5	965.0	250.0	19.7
250.0	50.0	19.6	1035.0	210.0	19.8
250.0	150.0	24.1	1040.0	140.0	19.9
250.0	250.0	24.2	1040.0	230.0	19.0
350.0	50.0	20.9	1065.0	50.0	21.8
350.0	150.0	26.3	1065.0	150.0	20.0
350.0	250.0	24.1	1065.0	250.0	18.3
385.0	235.0	22.8	1070.0	185.0	19.9
395.0	70.0	20.0	1090.0	155.0	21.4
400.0	215.0	20.4	1090.0	265.0	17.8
400.0	255.0	19.7	1095.0	30.0	20.1
410.0	90.0	22.2	1095.0	75.0	19.3
410.0	175.0	23.5	1120.0	20.0	18.5
415.0	195.0	22.7	1120.0	90.0	19.9
420.0	30.0	19.0	1125.0	250.0	18.8
425.0	125.0	23.2	1135.0	130.0	19.8
445.0	140.0	19.2	1145.0	265.0	17.9
445.0	250.0	17.4	1155.0	50.0	18.8
450.0	70.0	16.0	1165.0	150.0	23.1
465.0	50.0	14.2	1165.0	250.0	16.3
465.0	150.0	18.8	1205.0	10.0	17.5
465.0	250.0	18.4	1205.0	85.0	19.1
485.0	100.0	13.3	1225.0	10.0	17.2
505.0	110.0	13.2	1265.0	50.0	22.0
565.0	50.0	10.4	1265.0	150.0	22.7
565.0	150.0	15.5	1265.0	250.0	16.2
565.0	250.0	15.8	1365.0	50.0	21.8
665.0	50.0	14.9	1365.0	150.0	22.4
665.0	150.0	14.9	1365.0	250.0	18.3
665.0	250.0	20.9	1465.0	50.0	20.4
765.0	50.0	16.9	1465.0	150.0	19.6
765.0	150.0	20.3	1465.0	250.0	15.1
765.0	250.0	19.0			

Baseline Case

We constructed an estimation grid around the data for the first case. The rectangular domain extends in the X-direction (easting) from 0 to 1500 meters and in the

Y-direction (northing) from 20 to 270 meters. Node spacing was 10 meters in both directions for 151 nodes in X and 26 in Y. The total number of estimation points is 3926. For this case, we used the polynomial GSC model with a local neighborhood of 13 data points.

On output, the following parameters are provided for each node: order of intrinsity ν, covariance coefficient values, soil moisture estimate $\hat{X}(\mathbf{s}_k)$, and estimation error variance $\sigma_\varepsilon^2(\mathbf{s}_k)$. These results are graphically displayed to depict their spatial variation. Figure 7 depicts ν throughout the domain. The noteworthy contribution of this graphic is the spatial variability of ν. The order of intrinsity which *best* describes the trend characteristics for any local neighborhood varies from neighborhood to neighborhood. Similarly, Figure 8 depicts the spatial variability of the covariance coefficient values $\{a_0, c_0, c_1, c_2\}$ which provide the optimal (minimized) solution to the objective function [34]. Again, this variability indicates the correlation structure of the soil moisture SRF varies from neighborhood to neighborhood so that the optimal covariance parameters modeled for the correlation structure vary as well.

It is important to note that the values for ν and $\{a_0, c_0, c_1, c_2\}$ do not depend on the actual spatial coordinates of a given node but rather on the set of local data points which comprise the neighborhood around that node. Thus any two or more adjacent nodes which, because of their proximity, share the same neighborhood of m data will also have the same calculated values for ν and $\{a_0, c_0, c_1, c_2\}$. This characteristic results from the algorithm which employs only the local data points, removing them one at a time and estimating them from the remaining neighborhood points, to determine ν and the covariance parameters.

The calculated values of ν and $\{a_0, c_0, c_1, c_2\}$ at each node give rise to the estimates and estimation error variances for each node. Making the assumption that the estimation errors $\hat{X}(\mathbf{s}_k) - X(\mathbf{s}_k)$ are normally distributed, 95% confidence interval widths can be constructed using the values for $\sigma_\varepsilon^2(\mathbf{s}_k)$. The estimates and confidence interval widths are

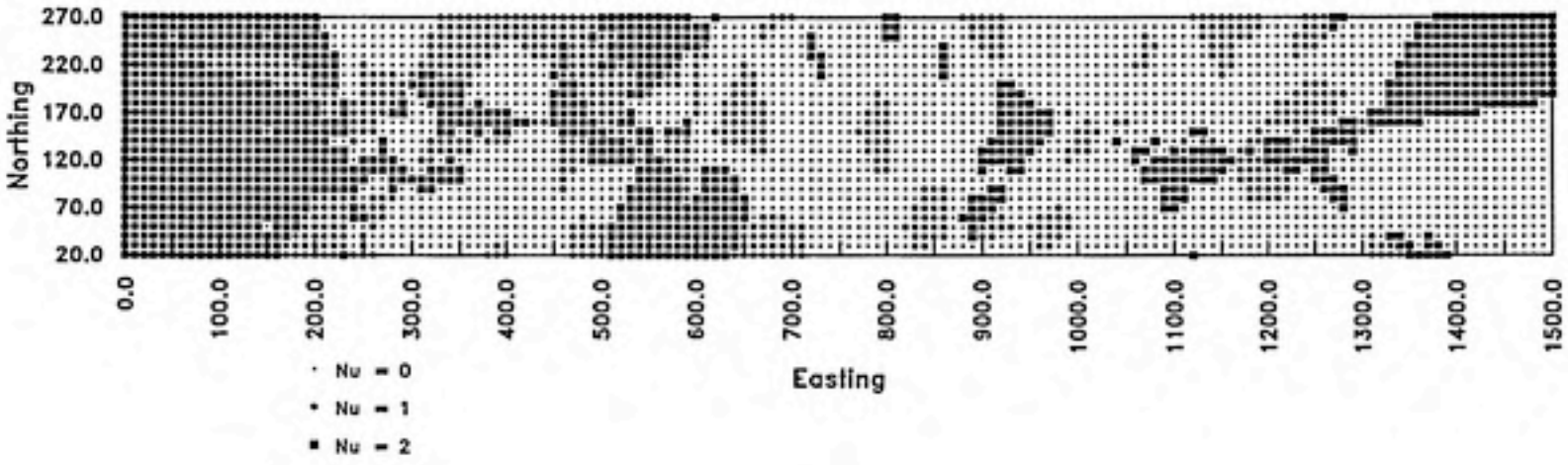


Figure 7. Order of intrinsity for baseline estimation case.

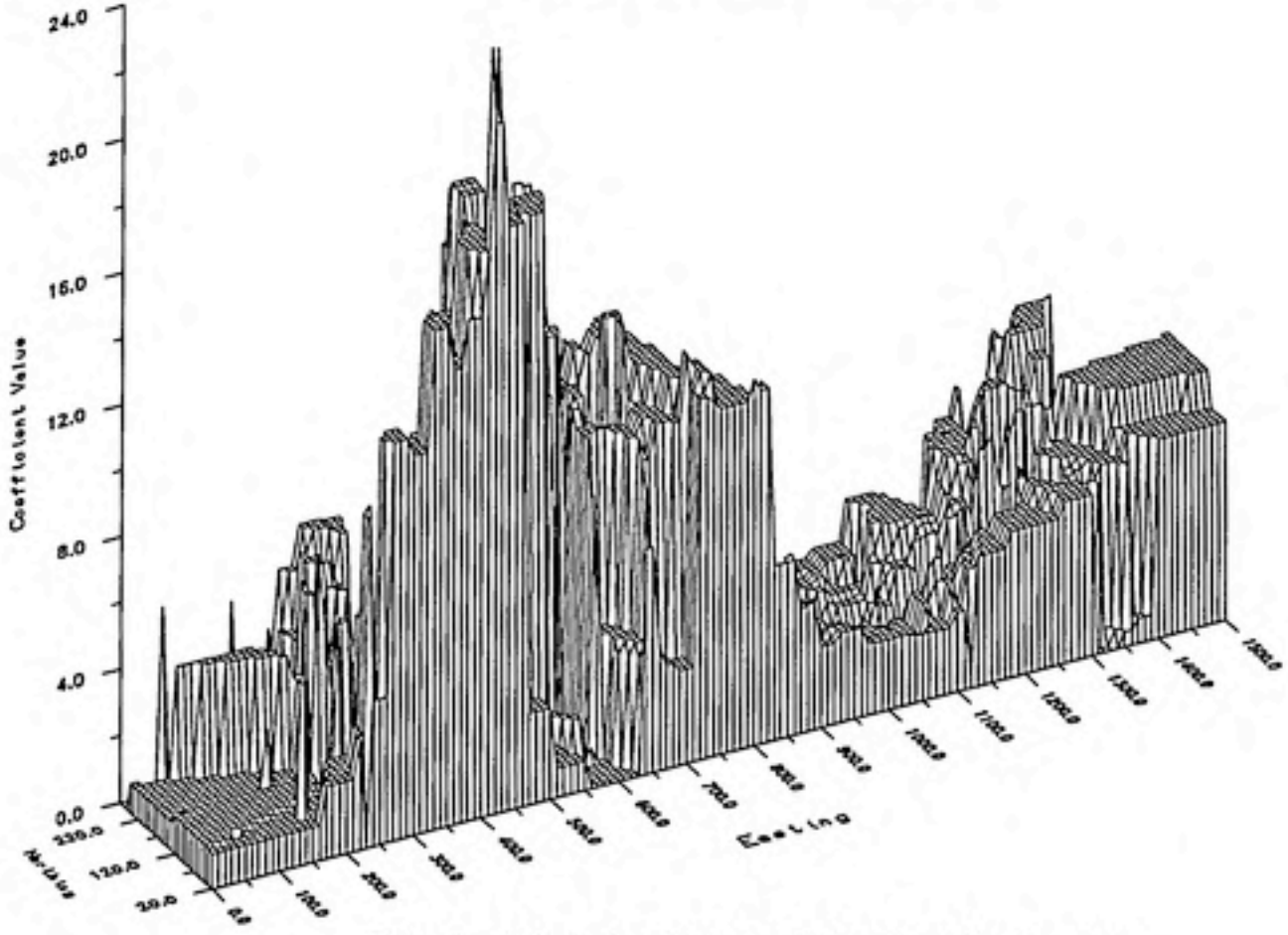


Figure 8a. Values of coefficient a_0 for baseline estimation case.

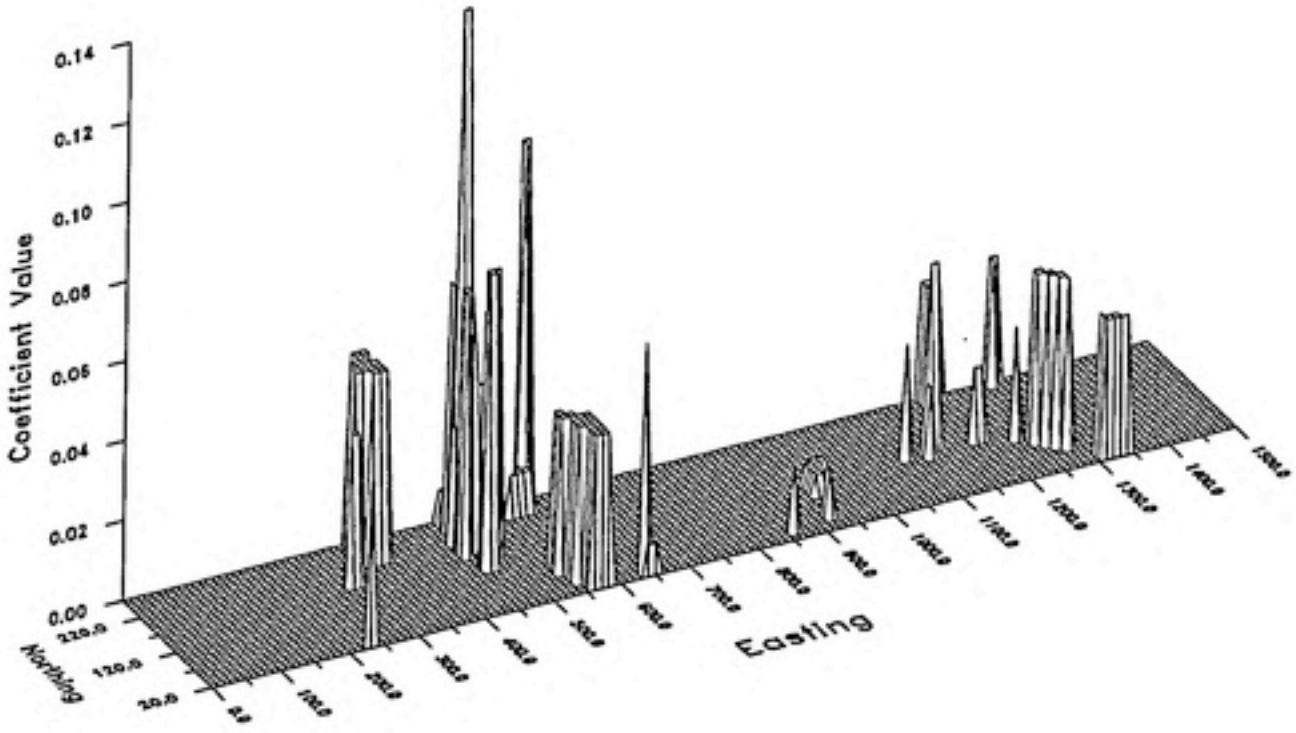


Figure 8b. Values of coefficient c_0 for baseline estimation case.

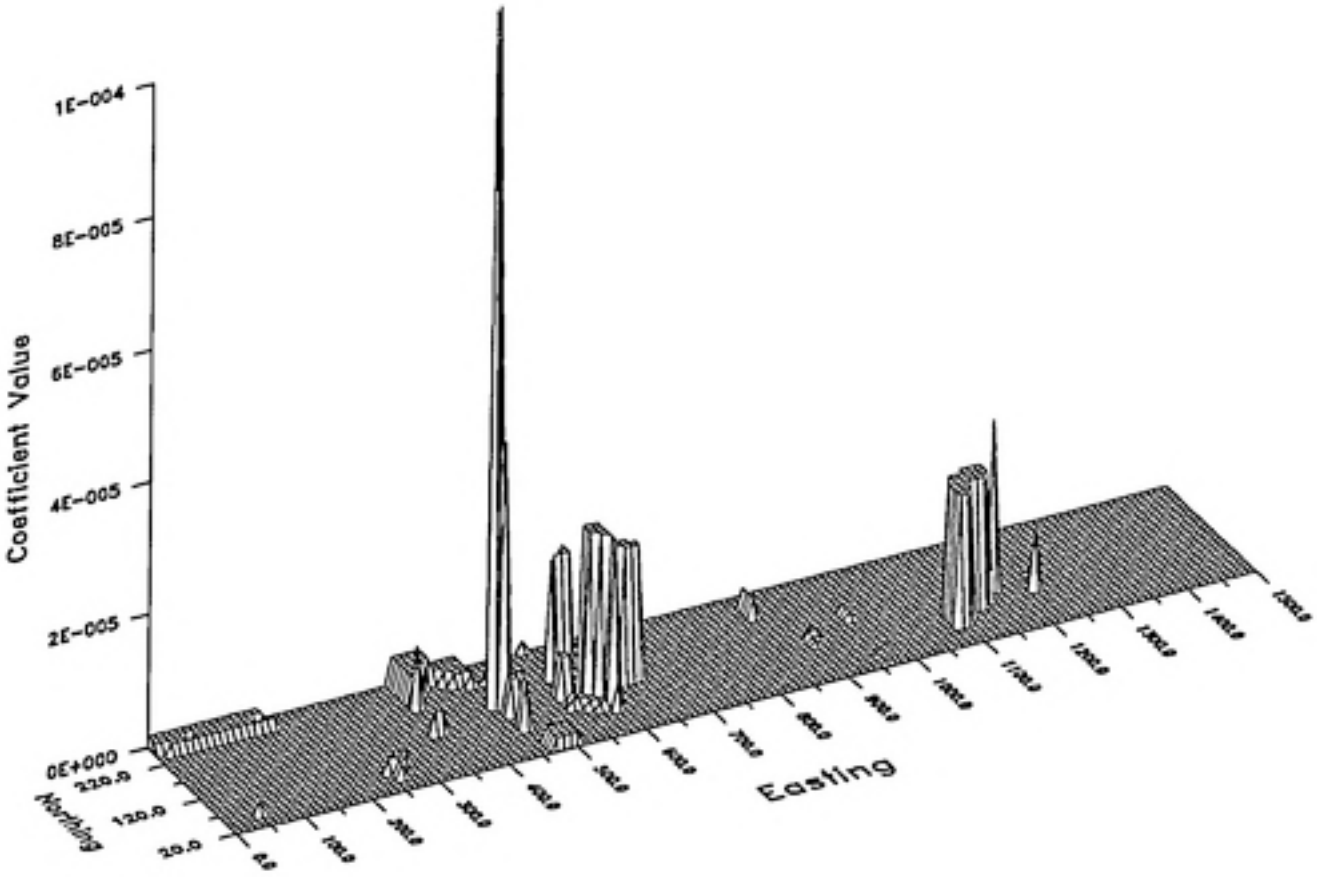


Figure 8c. Values of coefficient c_i for baseline estimation case.

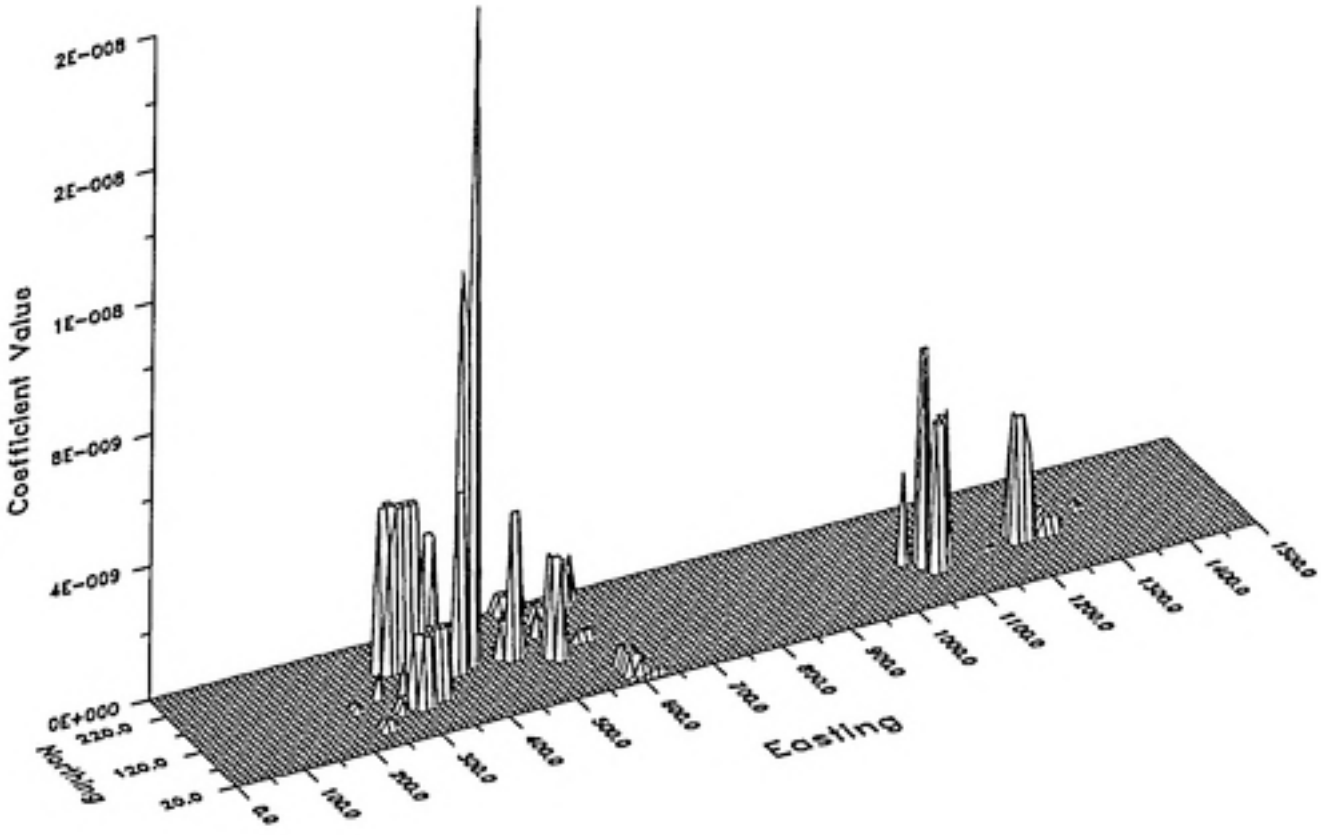


Figure 8d. Values of coefficient c_2 for baseline estimation case.

depicted graphically in Figures 9 and 10, respectively. Note that the estimation surface (Figure 9) is locally irregular and, on a more regional scale, depicts soil moisture content as a non-homogeneous process. This behavior is appropriate for a SRF approach to modeling and estimation.

Error and Sensitivity Analyses

We conducted error and sensitivity analyses to assess the performance of the algorithm. These consisted of cross-validating the 75 data -- that is, calculating estimates at each data point using surrounding points -- and determining the estimation error for various configurations of neighborhood size and generalized covariance model. Table 6 summarizes the scenarios considered. Appendix C contains histograms of estimation errors for each case as well as scatter plots with linear regression fits for the actual data values $X(\mathbf{s}_i)$ versus the estimated values $\hat{X}(\mathbf{s}_i)$. The linear regressions provide an indication of kriging prediction success with a regression model of $y = 1x$ and a regression coefficient of 1.0 indicating perfect prediction.

Table 6. Summary of cross-validation cases.

Case Number	Neighborhood Size	GSC Model	Nugget Term	Regression Coefficient
C1	10	P	Yes	0.5683
C2	13	P	Yes	0.4536
C3	16	P	Yes	0.4166
C4	10	PS	Yes	0.5725
C5	13	PS	Yes	0.5884
C6	16	PS	Yes	0.4763
C7	10	PE	(na)	0.4525
C8	13	PE	(na)	0.4729
C9	16	PE	(na)	0.0783
C10	10	P	No	0.6222
C11	13	PS	No	0.7100

P: Polynomial Model
PS: Polynomial-Spline Model
PE: Poly-Exponential Model

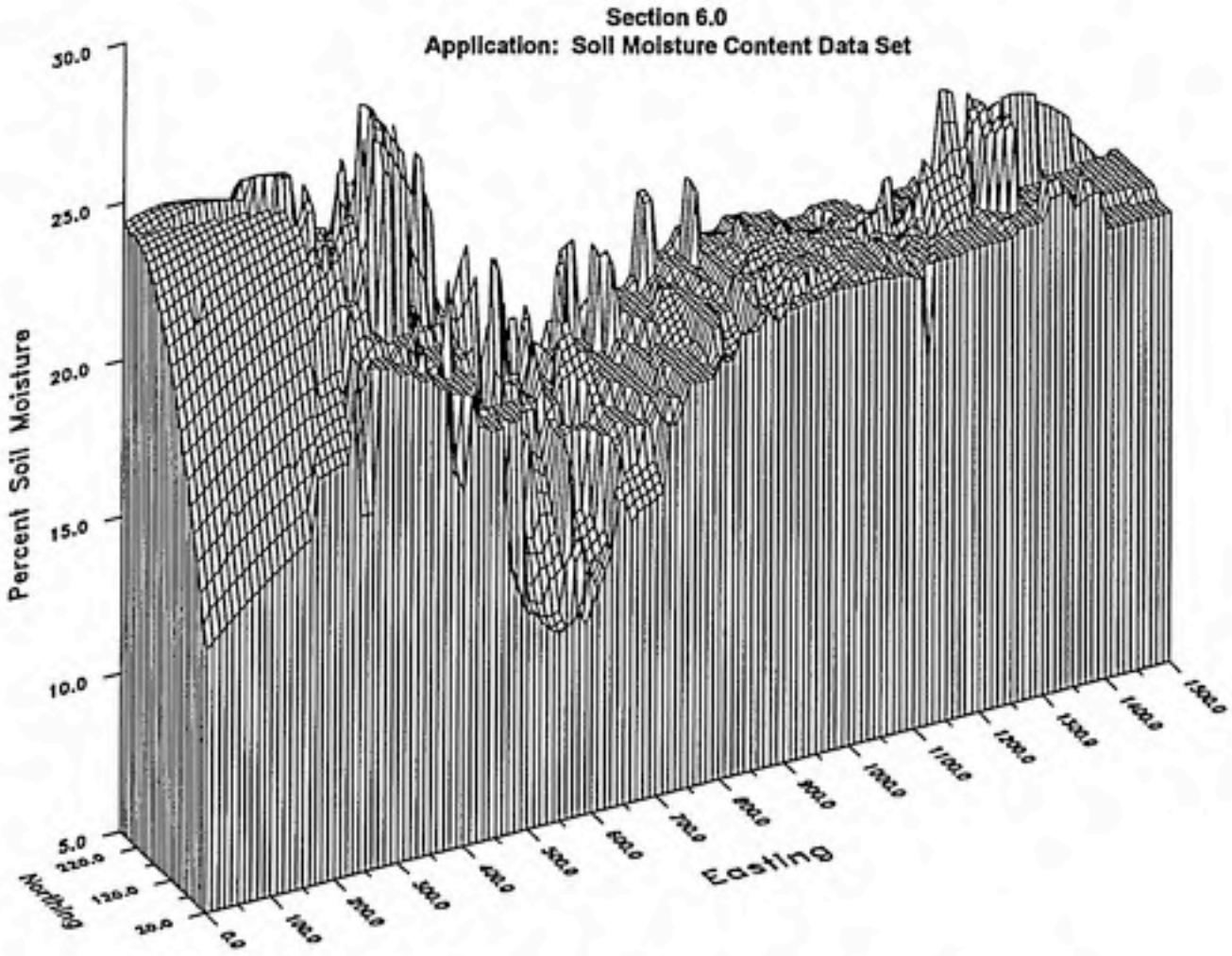


Figure 9. Estimated values of soil moisture content for baseline case.

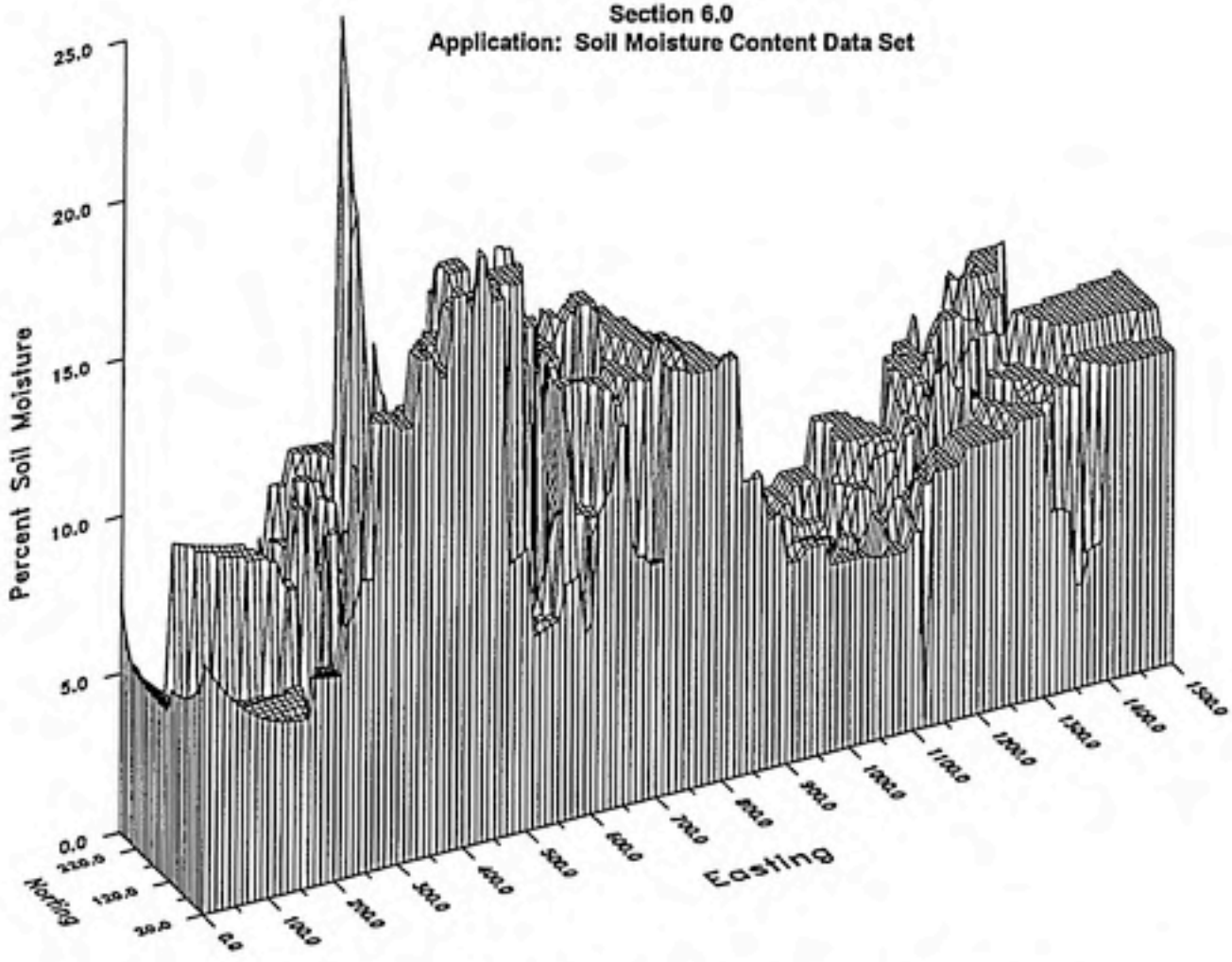


Figure 10. 95% confidence interval widths for estimates from baseline case.

In addition we have constructed cumulative distribution plots for the absolute errors $\left|\hat{X}(\mathbf{s}_i) - X(\mathbf{s}_i)\right|$. These plots can be contrasted against one another to provide an indication of relative overall accuracy. Figure 11 contrasts the absolute errors for the polynomial GSC given neighborhood sizes of 10, 13, and 16 points. Inspection suggests that the 10-, 13-, and 16-point neighborhoods are close in overall accuracy but that the 10-point subset provides slightly better overall accuracy than the 13-point which is slightly better than the 16-point. The corresponding linear regression coefficient values are 0.5683, 0.4536, and 0.4166 for 10, 13, and 16 points respectively. The decreasing values verify the suggestion of the cumulative plots, namely that the smallest neighborhood provides the most accurate results.

A similar plot of cumulative errors for the polynomial-spline GSC model is shown in Figure 12. Inspection suggests the 10- and 13-point neighborhood cases are very close in overall accuracy and both are slightly better than the 16-point case. The corresponding regression coefficient values are 0.5725, 0.5884, and 0.4763 for 10, 13, and 16 points respectively. The r^2 values echo the implication of the cumulative distribution graph that the smaller neighborhoods are better.

The same neighborhood-size cases were run with the poly-exponential model. Figure 13 is the cumulative plot of cross-validation errors. The 10- and 13-point neighborhoods provide approximately the same overall accuracy, and both are significantly better than the 16-point neighborhood. The corresponding regression coefficient values are 0.4525, 0.4729, and 0.0783 for 10, 13, and 16 points respectively. The extremely low r^2 value for the 16 point case was caused mostly by 3 extreme outlier errors, $|\varepsilon| > 10\%$ moisture. When these outliers are ignored, the regression coefficient improves to 0.4751.

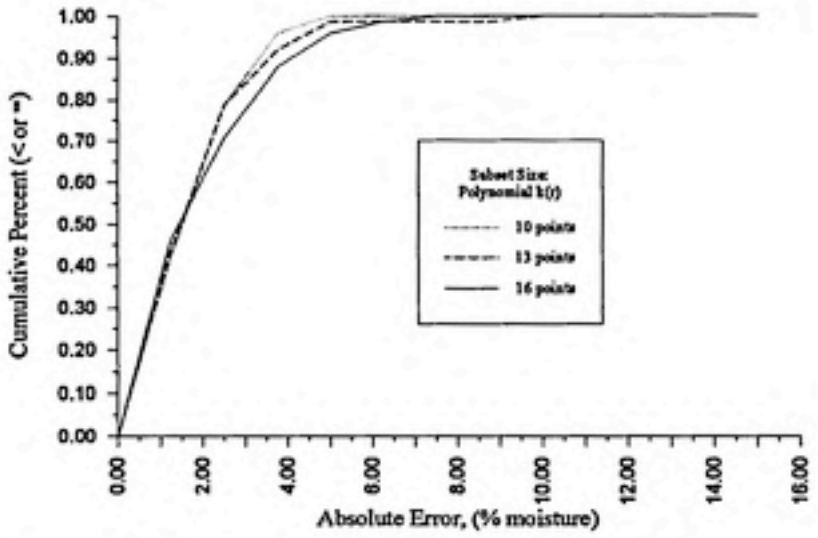


Figure 11. Cumulative frequency distribution of absolute errors for different neighborhood sizes using the polynomial GSC model.

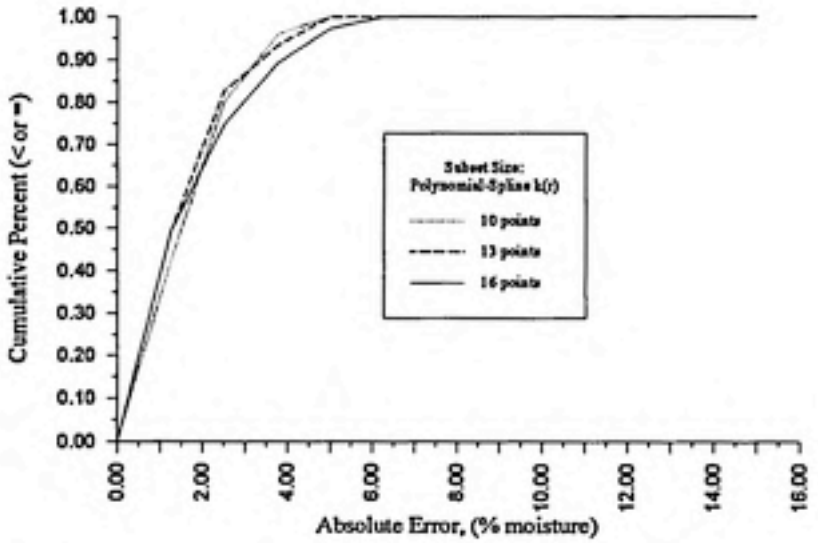


Figure 12. Cumulative frequency distribution of absolute errors for different neighborhood sizes using the polynomial-spline GSC model.

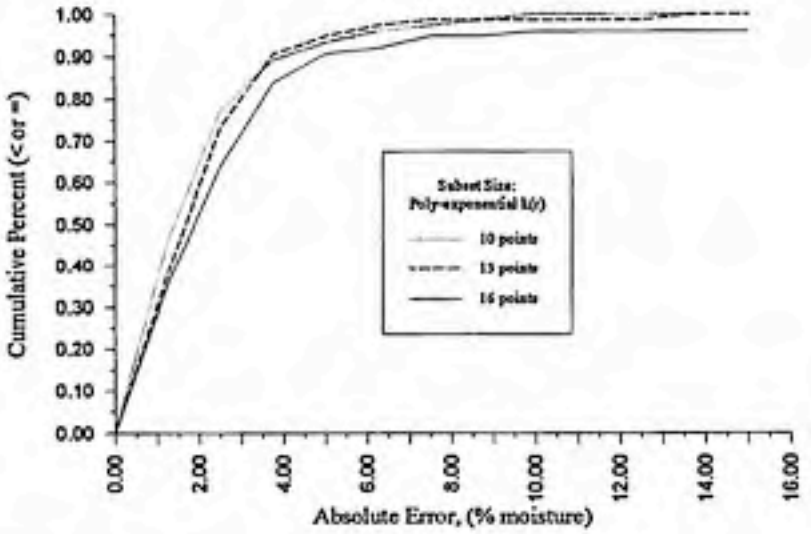


Figure 13. Cumulative frequency distribution of absolute errors for different neighborhood sizes using the poly-exponential GSC model.

It is apparent that neighborhood size has an effect on overall estimation accuracy. For all three GSC models, the smaller neighborhoods seemed to produce the best results with this data set. It should also be noted that the optimal GSC at a disproportionately large number of the grid locations in Figure 8 were chosen by the goodness-of-fit parameter [39] to be the pure nugget covariance $k_Y(r) = a_0\,\delta(r)$. Cressie (1986) made a similar observation of the geostatistical package BLUEPACK (Delfiner, Renard, and Chiles, 1978) which uses a similar algorithm for kriging under the intrinsic hypothesis. The declaration of a process as white noise runs contrary to the SRF concept of structural correlation. Thus, we reran the polynomial and polynomial-spline cross-validations without the GSC nugget term. The coefficient a_0 was simply forced to take the value 0.0. A 10-point neighborhood was used for the polynomial case, since 10 points produced the best results with nugget effect. Similarly 13 points were used for the polynomial-spline case.

Figure 14 shows the cumulative frequency distributions for the absolute errors from the polynomial case with and without a nugget term. Figure 15 is the same for the

polynomial-spline case. In both GSCs, the case without nugget effect appears to produce better overall accuracy. The linear regression coefficients for the polynomial model are 0.5683 with nugget term and 0.6222 without. For the polynomial-spline, the values are 0.5884 with nugget and 0.7100 without.

Estimation accuracy for this data set is improved using a GSC model without nugget effect. However, during the gridded estimation, the goodness-of-fit parameter [39] gave better values for the pure nugget GSC than for any other form the majority of the time. It is suggested that in subsequent versions of the kriging program an alternate goodness-of-fit evaluation (e.g. a jackknife estimator) should be used in place of [39] to correct this anomaly.

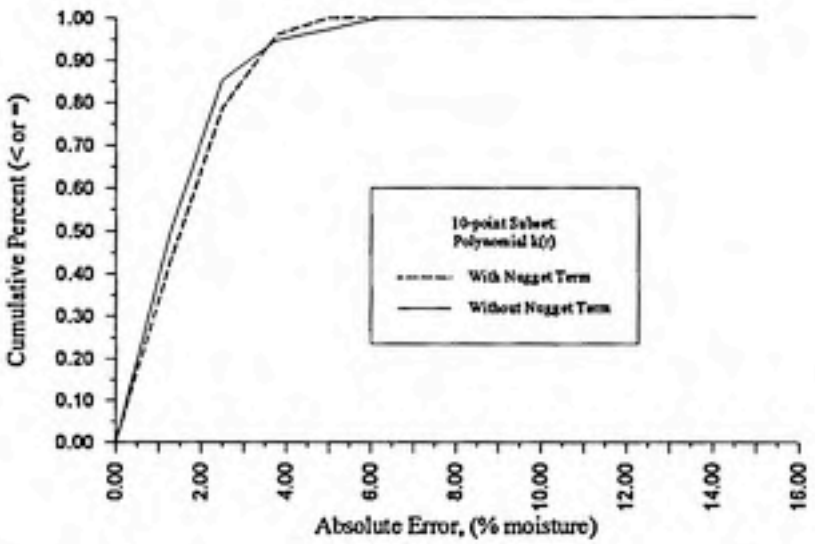


Figure 14. Cumulative frequency distribution of absolute errors, polynomial GSC model with and without nugget effect.

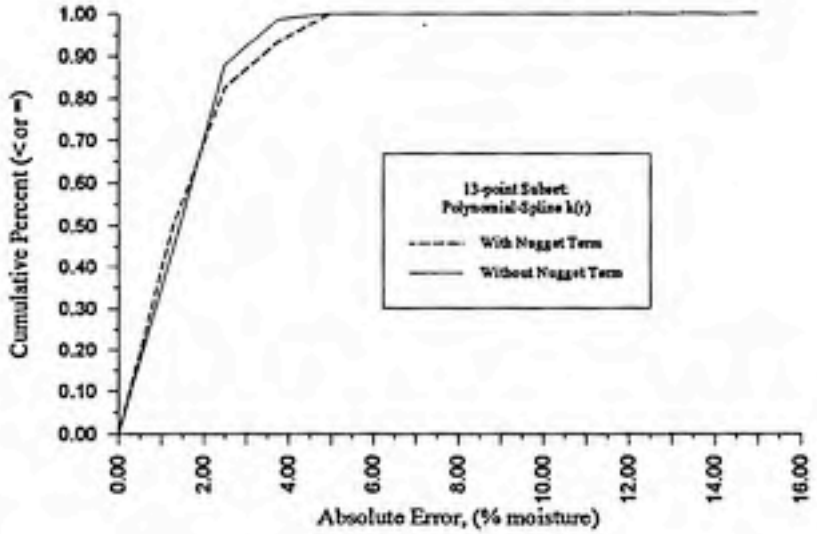


Figure 15. Cumulative frequency distribution of absolute errors, polynomial-spline GSC model with and without nugget effect.

One additional evaluation contrasts the three different GSC models: which one performs the best with this data set? Based on the correlation coefficients for the cross-validations above, the best performance is from the polynomial-spline model without nugget effect. The poly-exponential model proved the least accurate. However, the solution scheme for the poly-exponential coefficient is more complex than for the other GSCs and requires an initial coefficient guess as well as a penalty function appended to the objective function [34] to impose a solution constrain for permissibility of the GSC. Different combinations of initial guess and choice of penalty function might improve results.

Estimation Error Variance

One final topic concerns estimation error variance. It has been generally thought that the estimation error variance for a kriging estimate provides some indication of accuracy of the estimate. One would therefore expect high error variances to be associated with high errors. To check whether the kriging variances for both an ordinary

and universal kriging scheme could be used as measures of local accuracy, Journel and Rossi (1989) ranked the absolute errors from 100 cross-validated data and plotted them against the ranks of the corresponding error variances. They found no significant correlation (linear regression correlation coefficients < 0.1).

We conducted a similar exercise with our 75 data using the most successful case: polynomial-spline GSC and 13-point neighborhood. The scatter plot is shown in Figure 16. Our results confirm the findings of Journel and Rossi, namely that the relative magnitude of the estimation error variance is not correlated with the relative magnitude of the *actual* estimation error. It should be noted that solution of the kriging weights does not depend on the actual data values (data-independence property) but only on the covariance function and the relative positions of the data and estimation point. Thus, the error variance is not generally a measure of local accuracy. Rather, as noted by Journel and Rossi, it is simply a covariance-model-dependent ranking of configurations of data locations.

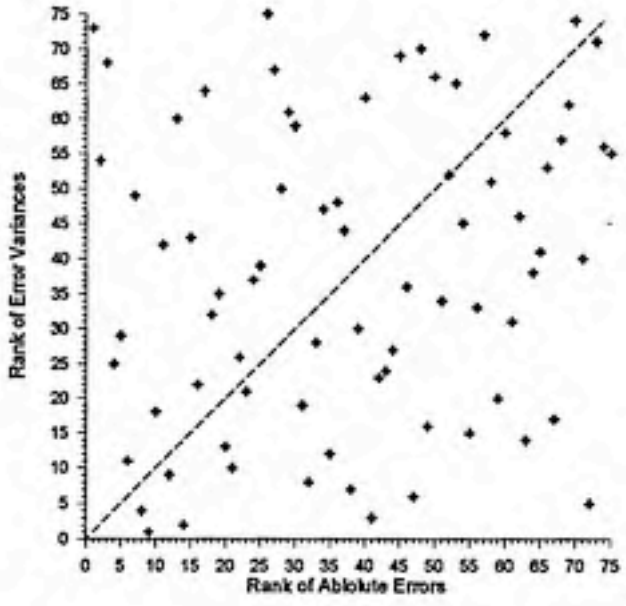


Figure 16. Ranks of absolute estimation errors vs. ranks of corresponding error variances. Correlation coefficient: r^2=0.0363.

We offer an additional explanation for this lack of correlation. Being a function of random variables, the estimation error is itself a random variable. As such it is characterized by a probability density function for which the variance parameter provides some indication of spread. The larger the variance, the wider the spread and the more probable is a large error value (i.e., inaccurate estimate). However, a large variance does not necessarily correspond to a large estimation error. Rather, it simply corresponds to a larger *range* of *possible* error values. It says nothing about the error values themselves. Being random variables, they can take on any value within the range assigned by the probability distributions. Thus, it is quite possible and not surprising to have a small error associated with a large error variance and a somewhat large error associated with a small variance. This condition would account for the lack of correlation between the relative magnitudes of the estimation errors and the error variances. It is suggested here that analysts using kriging should keep the above discussion in mind while assessing accuracy with the error variance.

Section 7.0
Summary

In this work we have applied, using local neighborhoods of data, a kriging estimator under the intrinsic hypothesis of SRFs to analyze trend characteristics and GSC parameters and to calculate point estimates of a spatially distributed parameter (in this case, soil moisture content). The intrinsic hypothesis studied here is more general in theory than commonly used conventional approaches. The non-homogeneous phenomenon is transformed into a homogeneous process of spatial increments of order ν. The resulting surface is trend-free. Subsequent analyses are performed using these increments rather than the original process. The assumption of data homogeneity which more conventional methods rely upon (e.g. ordinary kriging) is not necessary. The order of intrinsity parameter, ν, describes the degree of trend in the original process. This trend is filtered out in the increments. Additionally, the ordinary covariance of the original process may be decomposed into a homogeneous component (the GSC-ν) and a non-homogeneous component. As it turns out, ν and the GSC-ν provide a complete stochastic characterization of the process and are necessary and sufficient inputs to the kriging estimation scheme. As with other kriging estimators, the intrinsic kriging scheme is a linear, unbiased estimator with minimum estimation error variance.

We have implemented an automated algorithm which chooses the order of intrinsity and GSC-ν parameters that best suit the trend and correlation structure of the local neighborhoods. The point estimates are then based on these localized descriptions. When trend and covariance analyses are conducted over a grid, the results can be illustrated graphically to depict the spatial variations in trend and correlation structure of the original process.

The algorithm has been applied to a set of 75 soil moisture content measurements. We used three different GSC-ν models in various combinations with neighborhood size.

Comparison of the results from these combinations demonstrates that neighborhood size and GSC model both effect the overall accuracy of the estimates. The combination which produced the best overall results consisted of the polynomial-spline GSC model without the nugget term and a 13-point neighborhood.

References

Bilonick, R.A., "The space-time distribution of sulfate deposition in the northeastern United States," *Atmospheric Environment*, Vol. 19, No. 11, pp. 1829-1845, 1985.

Christakos, G., "On the problem of permissible covariance and variogram models," *Water Resources Research*, Vol. 20, No. 2, pp. 251-265, 1984.

Christakos, G., *Random Field Models in Earth Sciences*, Academic Press, Inc., San Diego, CA, 474 p., 1992.

Christakos, G. and G.A. Thesing, "The intrinsic random field model in the study of sulfate deposition processes," *Atmospheric Environment*, Vol. 27A, No. 10, pp. 1521-1540, 1993.

Christensen, R., "The equivalence of predictions from universal kriging and intrinsic random-function kriging," *Mathematical Geology*, Vol. 22, No. 6, pp. 655-664, 1990.

Cressie, N., "Kriging Nonstationary Data," *Journal of the American Statistical Association*, Vo. 81, No. 395, pp. 625-634, Sept. 1986.

Cressie, N., "A nonparametric view of generalized covariances for kriging," *Mathematical Geology*, Vol. 19, No. 5, pp. 425-449, 1987.

Cressie, N., *Statistics for Spatial Data*, J. Wiley, New York, 1991.

Delfiner, P., "Linear estimation of nonstationary spatial phenomena," *Advanced Geostatistics in the Mining Industry*, M. Guarascio et al, eds., D. Reidel Publishing Co., Dordrecht, Holland, pp. 49-68, 1976.

Delfiner, P., D. Renard, and J.P. Chiles, *BLUEPACK-3D Manual*, Fontainebleau, France: Centre de Geostatistique, 1978.

de Marsily, G., *Quantitative Hydrogeology*, Academic Press, Inc., San Diego, 440 p., 1986.

Haas, T.C., "Kriging and automated variogram modeling within a moving window," *Atmospheric Environment*, Vol. 24A, No. 7, pp. 1759-1769, 1990.

Isaaks, E.H. and R.M. Srivastava , *Applied Geostatistics,* Oxford University Press, New York, 561 p., 1989.

Journel, A.G., and M.E. Rossi, "When do we need a trend model in kriging," *Mathematical Geology*, Vol. 21, No. 7, pp. 715-739, 1989.

Kitanidis, P.K., "Statistical estimation of polynomial generalized covariance functions and hydrologic applications," *Water Resources Research*, Vol. 19, No. 4, pp. 909-921, 1983.

Matheron, G., *Le Krigeage Universel* (Cahiers du Centre de Morphologie Mathematique, No. 1), Fontainebleau, France: Centre de Morphologie Mathematique, 1969.

Matheron, G., "The intrinsic random functions and their applications," *Advances in Applied Probability*, Vol. 5, pp. 439-468, 1973.

Press, W.H., S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in FORTRAN*, Cambridge University Press, New York, 963 p., 1992.

Venkatram, A., "On the use of kriging in the spatial analysis of acid precipitation data," *Atmospheric Environment*, Vol. 22, No. 9, pp. 1963-1975, 1988.

Warrick, A.W., S.A. Musil, J.F. Artiola, D.M. Hendricks, and D.E. Myers of University of Arizona-Tucson, "Sampling strategies for hydrological properties and chemcial constituents in the upper vadose zone," unpublished Final Technical Report to Water Resources Research Grant Program (Section 105), U.S. Geological Survey, September 1990.

Appendix A:
Variable Definitions and Source Code for Polynomial GSC

Main Module

Name	Type	Description
A0, C0, C1, C2	Real variable	Local values of polynomial covariance coefficients.
Akm	Real array	Left-hand-side covariance matrix for system of intrinsic kriging equations.
Bkv	Real array	Right-hand-side covariance vector for system of intrinsic kriging equations.
Bkvdup	Real array	Dublicate of array Bkv.
Data	Real array	Spatial coordinates and parameter values of input data.
dX	Real variable	X-spacing between grid nodes.
dY	Real variable	Y-spacing between grid nodes.
Err	Real variable	Local value of estimation error variance.
Est	Real variable	Local value of point estimate.
ibul	Integer variable	Condition flag: 0 = no bull's-eye; 1 = bull's-eye.
icvg	Integer variable	Condition flag: 0 = non-converging covariance solutions; 1 = converging solutions.
Iext	Integer variable	Number of spatial monomials in system of kriging equations.
IGSC	Integer array	Mixed-integer template for various forms of the covariance function. Dimensioned 15x4x3: up to 15 possible polynomial covariance forms depending on ν; 4 coefficients in order as A0, C0, C1, and C2; 3 possible orders of intrinsity. An entry of 1 indicates a non-zero coefficient value for that covariance form and the given ν-value. Entries of 0 indicate the coefficient is not used in that form.
ispot	Integer variable	Index placeholder for data point which is bull's-eyed.
MAXdat	Integer variable	Dimension parameter for number of input data points.
MAXnhd	Integer variable	Dimension parameter for size of local neighborhoods.
Ncurnt	Integer variable	Counter for kriging node currently being processed.
ncvg	Integer variable	Counter for number of non-converging covariance coefficients solutions.
Ndata	Integer variable	Number of data points in the input file.
Ne	Integer variable	Number of equations in kriging system.
Nhood	Integer variable	Number of data points which comprise the local neighborhoods.
Nnx	Integer variable	Number of grid nodes in X-direction.
Nny	Integer variable	Number of grid nodes in Y-direction.
Nu	Integer variable	Local value of order of intrinsity, ν.
Radii	Real array	Matrix of scalar distances between all pairs of neighborhood points. Diagonal holds values between current node and all neighborhood points.
Sub1	Real array	Spatial coordinates and parameter values of input data.
WtsLin	Real array	Matrix of kriging weights resulting from estimation of each neighborhood point from all others using the linear covariance $k(r) = -r$.

Xo	Real variable	Starting X-value for grid.
Xsk	Real variable	X-value of current grid node.
Xvect	Real array	Left-hand-side vector of unknown kriging weights and Lagrange multipliers for system of kriging equations.
Yo	Real variable	Starting X-value for grid.
Ysk	Real variable	X-value of current grid node.

Subroutine Search

Name	Type	Description
Data	Real array	Spatial coordinates and parameter values of input data.
Iorder	Integer array	Index which keeps track of original order of sorted Radius values.
istep	Integer variable	Causes sorting of one additional Radius value when there is a bull's-eye.
MAXdat	Integer variable	Dimension parameter for number of input data points.
MAXnhd	Integer variable	Dimension parameter for size of local neighborhoods.
Ndata	Integer variable	Number of data points in the input file.
Nhood	Integer variable	Number of data points which comprise the local neighborhoods.
Radius	Real array	Distances between all data points and current grid node.
Subl	Real array	Spatial coordinates and parameter values of current neighborhood.
Xsk	Real variable	X-value of current grid node.
Ysk	Real variable	X-value of current grid node.

Subroutine CalcRd

Name	Type	Description
Datset	Real array	Spatial coordinates and parameter values of current neighborhood.
MAXnhd	Integer variable	Dimension parameter for size of local neighborhoods.
Nhood	Integer variable	Number of data points which comprise the local neighborhoods.
r	Real variable	Distance between two indicated points.
Radii	Real array	Matrix of scalar distances between all pairs of neighborhood points. Diagonal holds values between current node and all neighborhood points.
Xsk	Real variable	X-value of current grid node.
Ysk	Real variable	X-value of current grid node.

Subroutine Intrin

Name	Type	Description
Anu	Real array	Left-hand-side covariance matrix for system of intrinsic kriging equations constructed to estimate one data point from the others in the neighborhood.
Bnu	Real array	Right-hand-side covariance vector for system of intrinsic kriging equations constructed to estimate one data point from the others in the neighborhood..
Diff	Real array	Absolute values of estimation errors ε calculated for each removed data point and each value of Nu = 0,1,2.

Est	Real variable	Local estimate of data point removed from the neighborhood.
Ialfa	Integer variable	Number of spatial monomial in system of kriging equations.
Irank	Integer array	Rank values of estimation errors stored in Diff.
MAXnhd	Integer variable	Dimension parameter for size of local neighborhoods.
Ne	Integer variable	Number of equations in kriging system.
Nhood	Integer variable	Number of data points which comprise the local neighborhoods.
Nu	Integer variable	Local order of intrinsity, v.
Sub1	Real array	Spatial coordinates and parameter values of current neighborhood.
Sub2	Real array	Spatial coordinates and parameter values of current neighborhood with one point removed.
WtsLin	Real array	Matrix of kriging weights resulting from estimation of each neighborhood point from all others using the linear covariance k(r) = -r.
XAllNu	Real array	Same as WtsLin.
Xnu	Real array	Vector of solved kriging weights for a given point removed from the neighborhood and a given order of intrinsity.

Subroutine Remove

Name	Type	Description
IndxPt	Integer variable	Index value for neighborhood point currently removed.
MAXnhd	Integer variable	Dimension parameter for size of local neighborhoods
Nhood	Integer variable	Number of data points which comprise the local neighborhoods.
Sub1	Real array	Spatial coordinates and parameter values of current neighborhood.
Sub2	Real array	Spatial coordinates and parameter values of current neighborhood with one point removed.

Subroutine Rank

Name	Type	Description
Diff	Real array	Absolute values of estimation errors ε calculated for each removed data point and each value of Nu = 0,1,2.
Iorder	Integer array	Stores original (pre-ranked) orders of estimation errors in Diff.
Ipoint	Integer variable	Index value for neighborhood point currently removed.
Irank	Integer array	Stores rank for given estimation error for current point removed.
MAXnhd	Integer variable	Dimension parameter for size of local neighborhoods

Subroutine SetNu

Name	Type	Description
Iorder	Integer array	Stores original (pre-ranked) orders of estimation errors in Diff.
Ipoint	Integer variable	Index value for neighborhood point currently removed.
Irank	Integer array	Stores rank for given estimation error for current point removed.
Isum0,1,2	Integer variable	Summations of ranks associated with estimation errors from all points removed for a given value of Nu = 0,1,2.
MAXnhd	Integer variable	Dimension parameter for size of local neighborhoods
Minsum	Integer variable	Smallest of the three sums of ranks, Isum0, Isum1, and Isum2

Nhood	Integer variable	Number of data points which comprise the local neighborhoods.
Nu	Integer variable	Local value of order of intrinsity, ν.

Subroutine Covar

Name	Type	Description
A0, C0, C1, C2	Real variable	Local values of polynomial covariance coefficients.
BestCf	Real array	Set of coefficients returned by the goodness-of-fit test conducted by Select.
GSCcf	Real array	In COMMON block GSCval based in routine Covar. Holds values of polynomial covariance coefficients which are calculated in CalcCf. Dimensioned 15x4x3 entries: up to 15 possible polynomial covariance forms depending on ν; 4 coefficients in order as A0, C0, C1, and C2; 3 possible orders of intrinsity. Non-zero entries correspond to 1s in IGSC array in main module.
MAXnhd	Integer variable	Dimension parameter for size of local neighborhoods
Nhood	Integer variable	Number of data points which comprise the local neighborhoods.
Npass	Integer variable	Number of passing covariance forms for a given grid node.
Nu	Integer variable	Local value of order of intrinsity, ν.
Permis	Character array	Declared in COMMON block COPchk based in routine Covar. Entries correspond with a set of covariance coefficients for a given form and are either "Pass" or "Fail", indicating the status of the solution. Dimensioned 15x1x3 entries: 15 possible polynomial covariance forms depending on ν; 1 set of coefficients for each form; 3 possible orders of intrinsity.
Radii	Real array	Matrix of scalar distances between all pairs of neighborhood points. Diagonal holds values between current node and all neighborhood points.
Sub1	Real array	Spatial coordinates and parameter values of input data.
WtsLin	Real array	Matrix of kriging weights resulting from estimation of each neighborhood point from all others using the linear covariance $k(r) = -r$.

Subroutine CalcCf

Name	Type	Description
A0, C0, C1, C2	Real variable	Local values of polynomial covariance coefficients.
A0Li	Real variable	Value from function A0lamb.
BestCf	Real array	Set of coefficients returned by the goodness-of-fit test conducted by Select.
C0Li	Real variable	Value from function Cnlamb.
C1Li	Real variable	Value from function Cnlamb.
C2Li	Real variable	Value from function Cnlamb.
Csums	Real variable	Summartion counters of lambda function terms.
ErrChk	Character	Flag indicating if convergence test has passed for failed.
Errmax	Real variable	Maximum allowable error in any coefficient solution vector.
error	Real variable	Value from function Reldif.

GSCcf	Real array	In COMMON block GSCval based in routine Covar. Holds values of polynomial covariance coefficients which are calculated in CalcCf. Dimensioned 15x4x3 entries: up to 15 possible polynomial covariance forms depending on ν; 4 coefficients in order as A0, C0, C1, and C2; 3 possible orders of intrinsity. Non-zero entries correspond to 1s in IGSC array in main module.
icvg	Integer variable	COMMON flag indicating convergence ("1") or non-convergence ("0").
IGSC	Integer array	Declared in un-named COMMON block. Mixed-integer template for various forms of the covariance function. Dimensioned 15x4x3 entries: up to 15 possible polynomial covariance forms depending on ν; 4 coefficients in order as A0, C0, C1, and C2; 3 possible orders of intrinsity. An entry of 1 indicates a non-zero coefficient value for that covariance form and the given ν-value. Entries of 0 indicate the coefficient is not used in that form.
Iter	Integer variable	Iteration counter.
Maxit	Integer variable	Maximum number of iterations.
MAXnhd	Integer variable	Dimension parameter for size of local neighborhoods
More1	Integer variable	Iteration at which to begin the 1st level of successive under-relaxation..
More2	Integer variable	Iteration at which to begin the 2nd level of successive under-relaxation..
More3	Integer variable	Iteration at which to begin the 3rd level of successive under-relaxation..
More4	Integer variable	Iteration at which to begin the 4th level of successive under-relaxation..
ncvg	Integer variable	Counter for number of non-converging solutions.
Nforms	Integer variable	Number of possible GSC forms for a given value of ν.
Nhood	Integer variable	Number of data points which comprise the local neighborhoods.
Npass	Integer variable	Number of passing covariance forms for a given grid node.
Nu	Integer variable	Local value of order of intrinsity, ν.
Permis	Character array	Declared in COMMON block COPchk based in routine Covar. Entries correspond with a set of covariance coefficients for a given form and are either "Pass" or "Fail", indicating the status of the solution. Dimensioned 15x1x3 entries: 15 possible polynomial covariance forms depending on ν; 1 set of coefficients for each form; 3 possible orders of intrinsity.
Pvalu	Character	Flag indicating if permissibility test has passed or failed.
Radii	Real array	Matrix of scalar distances between all pairs of neighborhood points. Diagonal holds values between current node and all neighborhood points.
Sub1	Real array	Spatial coordinates and parameter values of input data.
sur1	Real variable	Successive under-relaxation factor for 1st level.
sur2	Real variable	Successive under-relaxation factor for 2nd level.
sur3	Real variable	Successive under-relaxation factor for 3rd level.
sur4	Real variable	Successive under-relaxation factor for 4th level.
WtsLin	Real array	Matrix of kriging weights resulting from estimation of each neighborhood point from all others using the linear covariance $k(r) = -r$.

WtsNxt	Real array	Set of kriging weights calculated at each iteration for each point removed in the local neighborhood.
Xnew	Real array	New vector of coefficient values for a current iteration.
Xold	Real array	Old vector of coefficient values for previous iteration.
Yi	Real variable	Value of the spatial increment for a current data point removed.
Ysums	Real array	Right-hand-side vector for system of equations for unknown coefficients.

Subroutine PermQ

Name	Type	Description
A0,C0,C1,C2	Real variable	Value of the polynomial covariance coefficients.
Flag	Charater variable	Flag indicating whether coefficient set is permissible or not.
Iform	Integer variable	Index number for current form of the GSC being considered.
Nu	Integer variable	Value of order of intrinsity.

Subroutine NewWts

Name	Type	Description
A0,C0,C1,C2	Real variable	Value of the polynomial covariance coefficients.
Ak	Real array	Covariance matrix for the kriging system.
Bk	Real array	Covariance vector for the kriging system.
Coeffs	Real array	Array of polynomial GSC coefficients.
Ialfa	Integer variable	Number of spatial monomials in system of kriging equations.
Nhood	Integer variable	Number of data in neighborhood.
Nu	Integer variable	Value for order of intrinsity ν.
Sub1	Real array	Array of neighborhood point locations and values.
Sub2	Real array	Array of neighborhood point locations and values with one point removed.
WtSet	Real array	Set of kriging weights for all points in the neighborhood removed.
Xk	Real array	Unknown vector in kriging system: the kriging weights.

Subroutine Select

Name	Type	Description
Ai	Real variable	Value of function A_i.
BestCf	Real array	Values of coefficients with best goodness-of-fit.
Eta	Real array	Values of goodness-of-fit parameter η associated with corresponding GSC form.
Nhood	Integer variable	Number of data in neighborhood.
Nu	Integer variable	Value for order of intrinsity ν.
Passng	Real array	Values of coefficients for all GSC forms which gave viable solutions
Sub1	Real array	Array of neighborhood point locations and values.
WtSet	Real array	Set of kriging weights for all points in the neighborhood removed.
Yi	Real variable	Value of spatial increment.

Subroutines Kmatrx and Kvectr

Name	Type	Description
A0,C0,C1,C2	Real variable	Values of polynomial GSC coefficients.
AK	Real array	Values of covariance matrix.
BK	Real array	Values of right-hand-side covariance vector.
Datset	Real array	Spatial locations and values for neighborhood data.
Iext	Integer variable	Number of spatial monomials in sytem of kriging equations.
Ndim	Integer variable	Number of points in neighborhood.
Polynm	Real array	Values of polynomial monomials.

Subroutines Estmat and EstErr

Name	Type	Description
A0,C0,C1,C2	Real variable	Values of polynomial GSC coefficients.
BK	Real array	Values of right-hand-side covariance vector.
Datset	Real array	Spatial locations and values for neighborhood data.
Error	Real variable	Value of the estimation error variance.
Est	Real variable	Value of the kriging point estimate.
Iext	Integer variable	Number of spatial monomials in sytem of kriging equations.
Ndim	Integer variable	Number of points in neighborhood.
Wts	Real array	Values of kriging weights.

This version of code contains in-line comments to explain the program structure. It is not meant for machine use. The source code filename for this program is MAIN6.FOR.

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

C.....Set the run parameters for the case run.
      PARAMETER (Ndata=75, Nhood=13, Nnx=151, Nny=1, Xo=0.d0, Yo=20.d0,
     &           dX=10.d0, dY=10.d0)

C.....Dimension the arrays which are used in the main module.
      PARAMETER (MAXdat=100, MAXnhd=20)
      DIMENSION Data(MAXdat,3), Subl(MAXnhd,3), WtsLin(MAXnhd,MAXnhd),
     &          Radii(MAXnhd,MAXnhd), Akm(MAXnhd+6,MAXnhd+6),
     &          Bkv(MAXnhd+6), Bkvdup(MAXnhd+6), Xvect(MAXnhd+6)

C.....Declare and dimension the variables and arrays which are common.
      COMMON  IGSC(15,4,0:2), ncvg, icvg
      COMMON /Serch/ ibul, ispot
      INTEGER IGSC, ncvg, icvg, ibul, ispot

C.....Open the disk files used in input/output data transfers. By convention, file units
   in the 10s are for input, and file units in the 20s are for output. Unit 10 is the
   input file containing the point value data. Three output files are specified. Unit 20
   is the file receiving all the kriging results. Unit 25 contains locations of any nodes
   where covariance coefficient solututions did not converge. Unit 30 logs the execution
   of the run, keeping a record of input data and run parameters.

      OPEN (10,File='d:\work\data\soilwatr.txt',IOSTAT=io10)
      OPEN (20,File='d:\work\scratch\chk77-1.txt',IOSTAT=io20)
      OPEN (25,File='d:\work\scratch\chk77-2.txt',IOSTAT=io25)
      OPEN (30,File='d:\work\scratch\chk77-3.txt',IOSTAT=io30)
      WRITE (*,'(4(2x,i5))') io10, io20, io25, io30

C.....Read the input data set from the specified disk file. Note that the
      first column of numbers in the data file should consist of integer
      values which index the order of the data points. Currently, this
      index value is not used in the program, but is useful in the data
      file as a means of identifying where points may have been removed
      by declustering. The input data is echoed to screen and to the log
      output file.

      READ (10,*) (idumb, Data(I,1), Data(I,2), Data(I,3), I=1,Ndata)

      WRITE (*,2020) (Data(I,1), Data(I,2), Data(I,3), I=1,Ndata)
      WRITE (30,2020) (Data(I,1), Data(I,2), Data(I,3), I=1,Ndata)
 2020 FORMAT (2(2x,f16.8),2x,f10.3)

C.....Echo the run parameters to the log output file.
      WRITE (30,*) '  Parent    =   main6.for'
      WRITE (30,*) '  Nhood     =', Nhood
      WRITE (30,*) '  GSC       =   Polynomial'
```

```
      WRITE (30,*) '  Nugget?   =   Yes'
      WRITE (30,*) '  Errmax    =   1.0e-05'
      WRITE (30,*) '  Ndata     =', Ndata
      WRITE (30,*) '  Nnx       =', Nnx
      WRITE (30,*) '  Nny       =', Nny
      WRITE (30,*) '  dX        =', dX
      WRITE (30,*) '  dY        =', dY

C.....Initialize some values.  The subroutine Templt initializes the values
      of the array IGSC to either 0 or 1, providing a template of possible
      covariance forms.  See variable definitions for explanations of these
      terms.

      CALL Templt
      ncvg=0
      Nn=Nnx*Nny
      Ncurnt=1
      Ysk=Yo

C.....The following nested DO-loops for Iny and Inx conduct the main portion of the
   program.  They iterate through the grid of kriging nodes, moving across a row of nodes
   from smaller to larger X-values then up to the next      row.  The local intrinsic
   kriging algorithm is conducted within this nest for each node in the grid.

      DO Iny=1,Nny
         Xsk=Xo

         DO Inx=1,Nnx

            icvg=1
            ibul=0
            WRITE (*,1090) Ncurnt, Nn, ncvg
            WRITE (30,1090) Ncurnt, Nn, ncvg
1090        FORMAT (' Processing node ',i4,' of ',i4,' : Ncvg = ',i4)

            CALL Search (Ndata,Nhood,Data,Xsk,Ysk,Subl)
            CALL CalcRd (Nhood,Xsk,Ysk,Subl,Radii)
            CALL Intrin (Nhood,Subl,WtsLin,Nu)
            CALL Covar (Nhood,Nu,Subl,Radii,WtsLin,
     &                  A0,C0,C1,C2)

C.............Now the order and coefficients of the GSC-v have been found for the current
   kriging node (Xsk,Ysk).  The kriging point estimate and estimation error variance can
   now be determined.  First, check to see a bull's-eye has occurred.  If so, the estimate
   is given the value of the data point where the bull's-eye has occurred, and the
   estimation error variance is zero.  If not, the estimate and error variance are
   calculated normally.

            IF (ibul .EQ. 1) THEN
               Est=Data(ispot,3)
               Err=0.d0
```

```
            ELSE IF (ibul .EQ. 0) THEN
                Iext=Ialpha(Nu)
                CALL Kmatrx (Nhood,Iext,A0,C0,C1,C2,Sub1,Akm)
                CALL Kvectr (Nhood,Iext,Xsk,Ysk,A0,C0,C1,C2,
     &                       Sub1,Bkv)

C.................The b-vector for the kriging system is used in two places:  the
       kriging system itself and the estimation error variance.  In the
       solution of the kriging system, the b-vector is operated on and
       its values changed.  Therefore, the duplicate Bkvdup is utilized
       in the linear solver, and the original Bkv is used in the error
       variance calculation.

                DO m=1,Nhood+6
                    Bkvdup(m)=Bkv(m)
                END DO

                Ne=Nhood+Iext
                CALL SolvGJ (Akm,Ne,26,Bkvdup,1,1)

C.................The return from the linear solver SolvGJ is the solution vector
       the the system of kriging equations.  These values are passed
       back the the calling module through the array for the right-
       hand-side vector, in this case Bkvdup.  Therefore, the solution
       values are copied into the array Xvect to avoid confusion as to
       where they are.

                DO ko=1,Ne
                    Xvect(ko)=Bkvdup(ko)
                END DO

                CALL Estmat (Nhood,Sub1,Xvect,Est)
                CALL EstErr (Nhood,Iext,A0,C0,C1,C2,Xvect,Bkv,Err)
            END IF

C.............Analysis for the current node is now complete.  The results are
   written to the output file.  Next, icvg is checked to see if there were any non-
   converging coefficient solutions to the covariance parameters.  If so, the coordinates
   of the current node are written to file 25.

            WRITE (20,2000) Xsk,Ysk,Est,Err,Nu,A0,C0,C1,C2
 2000       FORMAT (2(f12.7,2x),2(e15.7,2x),i1,4(2x,e15.7))

            IF (icvg .EQ. 0) THEN
                WRITE (25,2010) Inx,Iny,Xsk,Ysk
 2010           FORMAT (2(2x,i4),2(2x,f12.2))
            END IF

            Ncurnt=Ncurnt+1
            Xsk=Xsk+dX
        END DO
```

```
            Ysk=Ysk+dY
        END DO

C.....The line above defines the end of the DO-nest.  The program is completed here.

        WRITE (20,'(a,i4)') 'Number of GSC models not converging:', ncvg
        WRITE (25,'(a,i4)') 'Number of GSC models not converging:', ncvg

        STOP
        END
C*****************************************************************

C.....This subroutine initializes the values of the array IGSC which provide a mixed-
   integer template of the various forms of the polynomial GSC function which are possible
   with different orders of intrinsity.  All the values in IGSC are initially set to zero.
   Then those elements which correspond to covariance coefficients with non-zero values
   are given values of 1.

        SUBROUTINE Templt

        COMMON  IGSC(15,4,0:2)
        INTEGER IGSC

C.....The first step is to initialize all elements in IGSC().
        DO i=1,15
            DO j=1,4
                DO k=0,2
                    IGSC(i,j,k)=0
                END DO
            END DO
        END DO

C.....Now, set the nonzero values for the case of v=0.

        IGSC(1,1,0)=1
        IGSC(1,2,0)=1

        IGSC(2,1,0)=1

        IGSC(3,2,0)=1

C.....Now, set the nonzero values for the case of v=1.

        IGSC(1,1,1)=1
        IGSC(1,2,1)=1
        IGSC(1,3,1)=1

        IGSC(2,1,1)=1
        IGSC(2,2,1)=1
```

```
      IGSC(3,1,1)=1
      IGSC(3,3,1)=1

      IGSC(4,2,1)=1
      IGSC(4,3,1)=1

      IGSC(5,1,1)=1

      IGSC(6,2,1)=1

      IGSC(7,3,1)=1

C.....Now, set the nonzero values for the case of v=2.

      IGSC(1,1,2)=1
      IGSC(1,2,2)=1
      IGSC(1,3,2)=1
      IGSC(1,4,2)=1

      IGSC(2,1,2)=1
      IGSC(2,2,2)=1
      IGSC(2,3,2)=1

      IGSC(3,1,2)=1
      IGSC(3,2,2)=1
      IGSC(3,4,2)=1

      IGSC(4,1,2)=1
      IGSC(4,3,2)=1
      IGSC(4,4,2)=1

      IGSC(5,2,2)=1
      IGSC(5,3,2)=1
      IGSC(5,4,2)=1

      IGSC(6,1,2)=1
      IGSC(6,2,2)=1

      IGSC(7,1,2)=1
      IGSC(7,3,2)=1

      IGSC(8,1,2)=1
      IGSC(8,4,2)=1

      IGSC(9,2,2)=1
      IGSC(9,3,2)=1

      IGSC(10,2,2)=1
      IGSC(10,4,2)=1

      IGSC(11,3,2)=1
```

```
      IGSC(11,4,2)=1

      IGSC(12,1,2)=1

      IGSC(13,2,2)=1

      IGSC(14,3,2)=1

      IGSC(15,4,2)=1

C.....All the nonzero values are now set.  Return to calling module.

      RETURN
      END
C*****************************************************************

C.....This subroutine conducts the search to identify the Nhood closest data points to a
   given kriging node.  The search is exhaustive among all data points and is based on the
   ranking of the scalar distances between a given node and the data points.  The routine
   also identifies bull's-eyes and sets      a series of flags if a bull's-eye occurs.

      SUBROUTINE Search (Ndata,Nhood,Data,Xsk,Ysk,Subl)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (MAXdat=100, MAXnhd=20)
      DIMENSION Data(MAXdat,3), Radius(MAXdat), Subl(MAXnhd,3)
      INTEGER Iorder(MAXdat)

      COMMON /Serch/ ibul, ispot
      INTEGER ibul, ispot

C.....Calculate the scalar distances between all the data points and the node.
      DO I=1,Ndata
          Radius(I)=Dist(Data(I,1),Data(I,2),Xsk,Ysk)
          Iorder(I)=I
      END DO

C.....Sort the Nhood+1 smallest values in Radius(), leaving the rest unsorted.
C     Although there are only Nhood points to comprising the neighborhood, one
C     extra value is sorted in the event that there is a "bull's-eye" and the
C     closest point cannot be used.

      DO I=1,Nhood+1
          DO J=I+1,Ndata
              IF (Radius(J) .LT. Radius(I)) THEN
                  CALL RSwap(Radius(I),Radius(J))
                  CALL ISwap(Iorder(I),Iorder(J))
              END IF
          END DO
      END DO
```

```
C.....Check for a bullseye in the smallest radius value, i.e. the 1st array elem.
C      ibul:  "0"=no bullseye; "1"=bullseye
C      istep:  "1" = don't include 1st pt. in nhood: "0" = do include 1st pt.
C      ispot:  this is the index # of bullseyed data pt.

C.....The bull's-eyes are identified by a distance of zero between a given pair of
        kriging node and data point.

       IF (Radius(1) .EQ. 0.d0) THEN
           ibul=1
           istep=1
           ispot=Iorder(1)
       ELSE
           istep=0
       END IF

C.....Now set the values of Subl() to the appropriate values from Data().

       DO I=1,Nhood
           Subl(I,1)=Data(Iorder(I+istep),1)
           Subl(I,2)=Data(Iorder(I+istep),2)
           Subl(I,3)=Data(Iorder(I+istep),3)
       END DO

       RETURN
       END
C--------------------------------------

C.....This routine swaps two real variable values.

       SUBROUTINE RSwap (arg1,arg2)

       IMPLICIT DOUBLE PRECISION (A-H,O-Z)

       temp=arg1
       arg1=arg2
       arg2=temp

       RETURN
       END
C--------------------------------------

C.....This routine swaps two integer variable values.

       SUBROUTINE ISwap (iarg1,iarg2)

       IMPLICIT INTEGER (I-N)

       itemp=iarg1
       iarg1=iarg2
       iarg2=itemp
```

```
      RETURN
      END
C*****************************************************************

C.....This routine calculates the distances between all pairs of data points in the
   current neighborhood and stores them is the array Radii for use in other routines.

      SUBROUTINE CalcRd (Nhood,Xsk,Ysk,Datset,Radii)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      DIMENSION Datset(MAXnhd,3), Radii(MAXnhd,MAXnhd)

C.....STEP 1:  Calculate the distances between the points in Datset and
C     the current kriging node.  Store these values on the diagonal of Radii().

      DO I=1,Nhood
          r=Dist(Xsk,Ysk,Datset(I,1),Datset(I,2))
          Radii(I,I)=r
      END DO

C.....STEP 2:  Calculate the distances between all pairs of points in Datset().
C     Note that corresponding entries in the upper and lower triangle of
C     the matrix are equivalent.

      DO I=1,Nhood-1
          DO J=I+1,Nhood
              r=Dist(Datset(I,1),Datset(I,2),
     &               Datset(J,1),Datset(J,2))
              Radii(I,J)=r
              Radii(J,I)=r
          END DO
      END DO

      RETURN
      END
C*****************************************************************

C.....This subroutine determines the order of intrinsity from the set {0,1,2} which best
   represents the trend characteristics of the current neighborhood. The algorithm used is
   defined in the text section on the intrinsic kriging algorithm.

      SUBROUTINE Intrin (Nhood,Sub1,WtsLin,Nu)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      DIMENSION Sub1(MAXnhd,3), Sub2(MAXnhd,3), Diff(0:2),
     &          Anu(MAXnhd+6,MAXnhd+6), Bnu(MAXnhd+6), Xnu(MAXnhd+6),
     &          XAllNu(MAXnhd,MAXnhd,0:2), WtsLin(MAXnhd,MAXnhd)
      INTEGER Irank(MAXnhd,0:2)
```

```
      Nh=Nhood-1

C.....The following DO-loop is the master loop which progresses through all the data
   points in the neighborhood and conducts the operations described in the text sections.

      DO I=1,Nhood
          CALL Remove (Nhood,I,Sub1,Sub2)

C.........The following DO cycles through the set of Nu-values (0,1,2) for each point
   removed and calculates the estimation error associated with that Nu-value.

          DO Inu=0,2
              Ialfa=Ialpha(Inu)
              CALL Kmatrx (Nh,Ialfa,0.d0,1.d0,0.d0,0.d0,Sub2,Anu)
              CALL Kvectr (Nh,Ialfa,Sub1(I,1),Sub1(I,2),
     &                     0.d0,1.d0,0.d0,0.d0,Sub2,Bnu)

              Ne=Nh+Ialfa
              CALL SolvGJ (Anu,Ne,26,Bnu,1,1)
              DO ko=1,Ne
                  Xnu(ko)=Bnu(ko)
              END DO

              DO M=1,Nh
                  XAllNu(I,M,Inu)=Xnu(M)
              END DO
              CALL Estmat (Nh,Sub2,Xnu,Est)
              Diff(Inu)=ABS(Est-Sub1(I,3))
          END DO

C.........The estimation errors in Diff are now ranked from lowest to highest.

          CALL Rank (I,Diff,Irank)
      END DO

C.....The lowest sum of ranks for a given Nu is now determined, and the neighborhood is
   given the corresponding order of intrinsity.

      CALL SetNu (Nhood,Irank,Nu)

C.....Now, for later use, save the sets of kriging weights which correspond
C     to the determined value of v for the neighborhood.  The IF block maps
C     the appropriate layer in XAllNu into WtsLin.  Note that these two
C     arrays have different numbers of columns; hence, the IF block is needed
C     to correctly map the entries.  This segment is verified, 6-25-92.□

      DO M=1,Nhood
          DO N=1,Nhood
              IF (N .LT. M) THEN
                  WtsLin(M,N)=XAllNu(M,N,Nu)
```

```
            ELSE IF (N .EQ. M) THEN
                WtsLin(M,N)=-1.d0
            ELSE IF (N .GT. M) THEN
                WtsLin(M,N)=XAllNu(M,N-1,Nu)
            END IF
        END DO
    END DO

    RETURN
    END
C-------------------------------------

C.....This routine removes a data point from a neighborhood and creates a new
      neighborhood Sub2 consisting of the old neighbors minus the removed point.

    SUBROUTINE Remove (Nhood,IndxPt,Sub1,Sub2)

    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    PARAMETER (MAXnhd=20)
    DIMENSION Sub1(MAXnhd,3), Sub2(MAXnhd,3)

    IF (IndxPt .EQ. 1) THEN
        DO I=2,Nhood
            Sub2(I-1,1)=Sub1(I,1)
            Sub2(I-1,2)=Sub1(I,2)
            Sub2(I-1,3)=Sub1(I,3)
        END DO
    ELSE
        Sub2(IndxPt-1,1)=Sub1(IndxPt-1,1)
        Sub2(IndxPt-1,2)=Sub1(IndxPt-1,2)
        Sub2(IndxPt-1,3)=Sub1(IndxPt-1,3)
    END IF

    RETURN
    END
C-------------------------------------

C.....This routine ranks the estimation errors from Nu-values (0,1,2) as 1st, 2nd,
      or 3rd from smallest error to largest error.

    SUBROUTINE Rank (Ipoint,Diff,Irank)

    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    PARAMETER (MAXnhd=20)
    DIMENSION Diff(0:2)
    INTEGER Irank(MAXnhd,0:2), Iorder(0:2)

C.....Initialize the array Iorder().
    Iorder(0)=0
    Iorder(1)=1
    Iorder(2)=2
```

```
C.....Sort the estimation errors in increasing order.
      DO I=0,2
          DO J=I+1,2
              IF (Diff(J) .LT. Diff(I)) THEN
                  CALL Rswap (Diff(I),Diff(J))
                  CALL Iswap (Iorder(I),Iorder(J))
              END IF
          END DO
      END DO

C.....Assign ranks according to the sorted variance values.
      Irank(Ipoint,Iorder(0))=1
      Irank(Ipoint,Iorder(1))=2
      Irank(Ipoint,Iorder(2))=3

      RETURN
      END
C-------------------------------------

C.....This routine sets the order of intrinsity for a given neighborhood based on the
   lowest sum of ranks for all the points removed and all the tested values of Nu.

      SUBROUTINE SetNu (Nhood,Irank,Nu)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      INTEGER Irank(MAXnhd,0:2)

C.....Initialize the summation variables.
      Isum0=0
      Isum1=0
      Isum2=0

C.....Add up the rank values for each order of intrinsity (0,1,2).
      DO I=1,Nhood
          Isum0=Isum0+Irank(I,0)
          Isum1=Isum1+Irank(I,1)
          Isum2=Isum2+Irank(I,2)
      END DO

C.....Test if two or more of the rank sums are equal and smaller than the other rank sum.
   If so, then assign v subjectively to the smallest possible order. This decision is
   based on the fact that an ISRF of order η is also an ISRF of order ζ where ζ > η
   and that the reverse is not necessarily true. Thus, by chosing the lower Nu-value in
   the event of a tie, the more conservative case is chosen.

      IF ((Isum0 .EQ. Isum1) .AND. (Isum0 .LT. Isum2)) THEN
          Nu=0
          RETURN
      ELSE IF ((Isum0 .EQ. Isum2) .AND. (Isum0 .LT. Isum1)) THEN
```

```
            Nu=0
            RETURN
        ELSE IF ((Isum1 .EQ. Isum2) .AND. (Isum1 .LT. Isum0)) THEN
            Nu=1
            RETURN
        ELSE IF ((Isum0 .EQ. Isum1) .AND. (Isum0 .EQ. Isum2)) THEN
            Nu=0
            RETURN
        END IF

C.....Determine which sum of ranks is the lowest and set Nu to the corresponding order of
   intrinsity.

        Minsum=MIN(Isum0,Isum1,Isum2)
        IF (Minsum .EQ. Isum0) THEN
            Nu=0
        ELSE IF (Minsum .EQ. Isum1) THEN
            Nu=1
        ELSE IF (Minsum .EQ. Isum2) THEN
            Nu=2
        END IF

        RETURN
        END

C*****************************************************************

C.....This routine controls the calculation of the polynomial covariance parameters
   {a0,c0,c1,c2}.  Given a value of Nu, each possible form of the covariance is analyzed.
   Those which provide permissible solutions for {a0,c0,c1,c2} are checked for goodness of
   fit, and the best of those is used in the kriging calculations.  The steps implemented
   here are discussed in the text section on the intrinsic kriging algorithm.

        SUBROUTINE Covar (Nhood,Nu,Sub1,Radii,WtsLin,A0,C0,C1,C2)

        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        PARAMETER (MAXnhd=20)
        DIMENSION Sub1(MAXnhd,3), Radii(MAXnhd,MAXnhd),
       &          WtsLin(MAXnhd,MAXnhd), BestCf(4)

        COMMON /GSCval/ GSCcf(15,4,0:2)
        COMMON /COPchk/ Permis(15,1,0:2)
        DOUBLE PRECISION GSCcf
        CHARACTER*4 Permis

C.....First, initialize the values of the /GSCval/ and /COPchk/ COMMON arrays.

        DO i=1,15
            DO j=1,4
                DO k=0,2
```

```
                GSCcf(i,j,k)=0.d0
                Permis(i,1,k)='Fail'
            END DO
        END DO
    END DO

C.....Now, begin the segment to yield the values of the GSC coefs.

    CALL CalcCf (Nhood,Nu,Subl,WtsLin,Radii,Npass)

C.....The variable Npass is returned from the routine CalcCf and is the number of
   coefficient forms which pass the permissibility criteria.  If this value is zero, a
   problem has occurred in CalcCf, since no solution has been found for {a0,c0,c1,c2}.  In
   this case, the linear covariance form k(r)=-r is used by        default for the
   kriging calculations.  If the value of Npass is 1, then only one form has passed and
   there is no need to conduct the goodness-of-fit test. If more than one form passes, the
   fit test is conducted to select the form which best describes the correlation structure
   of the given neighborhood.

    IF (Npass .EQ. 0) THEN
        WRITE (30,*) ' **ERROR**  Npass=0 in Covar().'
        A0=0.d0
        C0=1.d0
        C1=0.d0
        C2=0.d0
    ELSE IF (Npass .EQ. 1) THEN
        DO i=1,15
            IF (Permis(i,1,Nu) .EQ. 'Pass') THEN
                DO j=1,4
                    BestCf(j)=GSCcf(i,j,Nu)
                END DO
            END IF
        END DO
    ELSE IF (Npass .GE. 2) THEN
        CALL Select (Nhood,Nu,Npass,Subl,Radii,BestCf)
    END IF

    A0=BestCf(1)
    C0=BestCf(2)
    C1=BestCf(3)
    C2=BestCf(4)

    RETURN
    END
C*****************************************************************

C.....For a given value of Nu, this subroutine calculates the values of the covariance
   coefficients and determines whether or not they are permissible.  Embedded within the
   loop for Iform, which cycles through the possible GSC forms, is the DO WHILE loop which
   iterates on the coefficient solution set, checking for coefficient permissibility and
```

convergence at each iteration. Each solution set corresponds to a GSC form, and those sets which are permissible and converged are saved. If there is more than one such set, they are checked for goodness-of-fit from the calling module Covar.

```
      SUBROUTINE CalcCf (Nhood,Nu,Subl,WtsLin,Radii,Npass)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      DIMENSION WtsLin(MAXnhd,MAXnhd), WtsNxt(MAXnhd,MAXnhd),
     &          Radii(MAXnhd,MAXnhd), Subl(MAXnhd,3), Csums(4,4),
     &          Xnew(4), Xold(4), Ysums(4)
      CHARACTER*4 ErrChk, Pvalu

      COMMON  IGSC(15,4,0:2), ncvg, icvg
      COMMON /GSCval/ GSCcf(15,4,0:2)
      COMMON /COPchk/ Permis(15,1,0:2)
      DOUBLE PRECISION GSCcf
      INTEGER IGSC, ncvg, icvg
      CHARACTER*4 Permis

      PARAMETER (Errmax=1.0d-05, Maxit=25, More1=10, sur1=0.5d0,
     &           More2=12, sur2=0.1d0, More3=15, sur3=0.00001d0,
     &           More4=20, sur4=0.000001d0)

C.....Assign the number of possible GSC forms given the order v.
      IF (Nu .EQ. 0) Nforms=3
      IF (Nu .EQ. 1) Nforms=7
      IF (Nu .EQ. 2) Nforms=15
      Npass=0

C.....Begin the main loop which iterates through all possible forms of the GSC-v.

      DO 100 Iform=1,Nforms

C.........Now, equate values in WtsNxt with those in WtsLin so that the
C         initial set of weights comes from the case of k=(-r).  Note that
C         WtsNxt() is reset to this initial set of values for each new GSC
C         form.

          DO I=1,Nhood
              DO J=1,Nhood
                  WtsNxt(I,J)=WtsLin(I,J)
              END DO
          END DO

C.........Now, begin the DO WHILE loop which tests the convergence and
C         permissibility of the different GSC-v forms.  Some parameters are
C         initialized first.  Note that the parameters Xnew and Xold are
C         forced to zero by default for each new form of the GSC-v.  Non-zero
C         solutions from the solver will override the default values.
```

```
C.........Initialize some variables before beginning.

          ErrChk='Fail'
          Pvalu='Pass'
          Iter=0
          DO ni=1,4
              Xnew(ni)=0.d0
              Xold(ni)=0.d0
          END DO

C.........Begin the loop to iterate on the coeffs for a given GSC form.

          DO WHILE ((ErrChk .EQ. 'Fail') .AND. (Pvalu .EQ. 'Pass')
     &              .AND. (Iter .LT. Maxit))
              Iter=Iter+1

C.............STEP 1: construct the system of equations resulting from dF/dc.

              DO mi=1,4
                  Ysums(mi)=0.d0
                  DO mj=1,4
                      Csums(mi,mj)=0.d0
                  END DO
              END DO

              A0Li=0.d0
              C0Li=0.d0
              C1Li=0.d0
              C2Li=0.d0

              DO i=1,Nhood

                  IF (IGSC(Iform,1,Nu) .NE. 0)
     &                A0Li=A0lamb(Nhood,i,WtsNxt)
                  IF (IGSC(Iform,2,Nu) .NE. 0)
     &                C0Li=Cnlamb(Nhood,i,WtsNxt,Radii,1)
                  IF (IGSC(Iform,3,Nu) .NE. 0)
     &                C1Li=Cnlamb(Nhood,i,WtsNxt,Radii,3)
                  IF (IGSC(Iform,4,Nu) .NE. 0)
     &                C2Li=Cnlamb(Nhood,i,WtsNxt,Radii,5)
                  Yi=SpaInc(Nhood,i,WtsNxt,Sub1)

                  IF (IGSC(Iform,1,Nu) .NE. 0) THEN
                      Csums(1,1)=Csums(1,1)+A0Li**2
                      Csums(1,2)=Csums(1,2)-A0Li*C0Li
                      Csums(1,3)=Csums(1,3)+A0Li*C1Li
                      Csums(1,4)=Csums(1,4)-A0Li*C2Li
                      Ysums(1)=Ysums(1)+(Yi**2)*A0Li
                  ELSE
                      Csums(1,1)=1.d0
                  END IF
```

```
            IF (IGSC(Iform,2,Nu) .NE. 0) THEN
                Csums(2,1)=Csums(2,1)+C0Li*A0Li
                Csums(2,2)=Csums(2,2)-C0Li**2
                Csums(2,3)=Csums(2,3)+C0Li*C1Li
                Csums(2,4)=Csums(2,4)-C0Li*C2Li
                Ysums(2)=Ysums(2)+(Yi**2)*C0Li
            ELSE
                Csums(2,2)=1.d0
            END IF

            IF (IGSC(Iform,3,Nu) .NE. 0) THEN
                Csums(3,1)=Csums(3,1)+C1Li*A0Li
                Csums(3,2)=Csums(3,2)-C1Li*C0Li
                Csums(3,3)=Csums(3,3)+C1Li**2
                Csums(3,4)=Csums(3,4)-C1Li*C2Li
                Ysums(3)=Ysums(3)+(Yi**2)*C1Li
            ELSE
                Csums(3,3)=1.d0
            END IF

            IF (IGSC(Iform,4,Nu) .NE. 0) THEN
                Csums(4,1)=Csums(4,1)+C2Li*A0Li
                Csums(4,2)=Csums(4,2)-C2Li*C0Li
                Csums(4,3)=Csums(4,3)+C2Li*C1Li
                Csums(4,4)=Csums(4,4)-C2Li**2
                Ysums(4)=Ysums(4)+(Yi**2)*C2Li
            ELSE
                Csums(4,4)=1.d0
            END IF

        END DO

C.............STEP 2:  solve the current system of equations using Gauss-Jordan.

        CALL SolvGJ (Csums,4,4,Ysums,1,1)
        DO ki=1,4
            Xnew(ki)=Ysums(ki)
        END DO

C.............STEP 3:  use successive underrelaxation if the solution is not converging.

        IF ((Iter .GT. More1) .AND. (Iter .LE. More2)) THEN
            DO ir=1,4
                Xnew(ir)=sur1*Xnew(ir)+(1.d0-sur1)*Xold(ir)
            END DO
        ELSE IF ((Iter .GT. More2) .AND. (Iter .LE. More3)) THEN
            DO ir=1,4
                Xnew(ir)=sur2*Xnew(ir)+(1.0-sur2)*Xold(ir)
            END DO
        ELSE IF ((Iter .GT. More3) .AND. (Iter .LE. More4)) THEN
```

```
                DO ir=1,4
                    Xnew(ir)=sur3*Xnew(ir)+(1.0-sur3)*Xold(ir)
                END DO
            ELSE IF (Iter .GT. More4) THEN
                DO ir=1,4
                    Xnew(ir)=sur4*Xnew(ir)+(1.0-sur4)*Xold(ir)
                END DO
            END IF

C.............STEP 4:  check the permissibility of the new coeff values.
            CALL PermQ (Nu,Iform,Xnew(1),Xnew(2),Xnew(3),
     &                  Xnew(4),Pvalu)
            IF (Pvalu .EQ. 'Fail') THEN
                GOTO 200
            ELSE
                CONTINUE
            END IF

C.............STEP 5:  check if the new coeff solution set has converged.  If not, update
   the solution set and prepare another set of kriging weights using the updated GSC
   coefficients.

            error=Reldif(4,Xold,Xnew)
            IF ((error .LE. Errmax) .AND. (Iter .GT. 1)) THEN
                ErrChk='Pass'
            ELSE
                ErrChk='Fail'
                Xold=Xnew
                CALL NewWts (Nhood,Nu,Subl,Xold,WtsNxt)
            END IF

 200        CONTINUE
        END DO

        IF (Iter .GE. Maxit) THEN
            ncvg=ncvg+1
            icvg=0
            ErrChk='Fail'
        END IF

        GSCcf(Iform,1,Nu)=Xnew(1)
        GSCcf(Iform,2,Nu)=Xnew(2)
        GSCcf(Iform,3,Nu)=Xnew(3)
        GSCcf(Iform,4,Nu)=Xnew(4)

        IF ((ErrChk .EQ. 'Pass') .AND. (Pvalu .EQ. 'Pass')) THEN
            Permis(Iform,1,Nu)='Pass'
            Npass=Npass+1
        ELSE
            Permis(Iform,1,Nu)='Fail'
        END IF
```

```
  100 END DO

      RETURN
      END
C*****************************************************************

C.....This subroutine checks the permissibility of a set of covariance coefficients
   {a0,c0,c1,c2} against the required conditions.

      SUBROUTINE PermQ (Nu,Iform,A0,C0,C1,C2,Flag)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      CHARACTER*4  Flag

      COMMON  IGSC(15,4,0:2)
      INTEGER IGSC

      Flag='Pass'

      IF (A0 .LT. 0.d0) THEN
          Flag='Fail'
          RETURN
      ELSE IF (C0 .LT. 0.d0) THEN
          Flag='Fail'
          RETURN
      ELSE IF (C2 .LT. 0.d0) THEN
          Flag='Fail'
          RETURN
      END IF

      IF ((Nu .EQ. 2) .AND. (IGSC(Iform,3,Nu) .NE. 0)) THEN
          value= -DSQRT((100.d0/9.d0)*C0*C2)
          IF (C1 .LT. value) Flag='Fail'
      ELSE
          IF (C1 .LT. 0.d0) Flag='Fail'
      END IF

      RETURN
      END
C*****************************************************************

C.....Given a value of Nu, a neighborhood of data, and a set of covariance coefficients,
   this routine calculates a new set of kriging weights λ_ij for each point i removed from
   the neighborhood.

      SUBROUTINE NewWts (Nhood,Nu,Sub1,Coeffs,WtSet)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (MAXdat=100, MAXnhd=20)
```

```
      DIMENSION Sub1(MAXnhd,3), Coeffs(4), Sub2(MAXnhd,3),
     &          Ak(MAXnhd+6,MAXnhd+6), Bk(MAXnhd+6), Xk(MAXnhd+6),
     &          WtSet(MAXnhd,MAXnhd)

C.....Assign some initial values to local variables.

      Nh=Nhood-1
      Ialfa=Ialpha(Nu)

      A0=Coeffs(1)
      C0=Coeffs(2)
      C1=Coeffs(3)
      C2=Coeffs(4)

C.....Now, construct the matrix of kriging weights using the GSC passed in.

      DO i=1,Nhood
          CALL Remove (Nhood,i,Sub1,Sub2)
          CALL Kmatrx (Nh,Ialfa,A0,C0,C1,C2,Sub2,Ak)
          CALL Kvectr (Nh,Ialfa,Sub1(i,1),Sub1(i,2),
     &                 A0,C0,C1,C2,Sub2,Bk)

          Ne=Nh+Ialfa
          CALL SolvGJ (Ak,Ne,26,Bk,1,1)
          DO ko=1,Ne
              Xk(ko)=Bk(ko)
          END DO

          DO j=1,Nhood
              IF (j .LT. i) THEN
                  WtSet(i,j)=Xk(j)
              ELSE IF (j .EQ. i) THEN
                  WtSet(i,j)=-1.d0
              ELSE IF (j .GT. i) THEN
                  WtSet(i,j)=Xk(j-1)
              END IF
          END DO
      END DO

      RETURN
      END
C*********************************************************************

C.....This routine calculates the value of the function Λ_I^(s) given the set of kriging
   weights λ_IJ.

      FUNCTION A0lamb (Nhood,Irow,WtSet)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (MAXnhd=20)
```

```
      DIMENSION WtSet(MAXnhd,MAXnhd)

      Sum=0.d0

      DO M=1,Nhood
          Sum=Sum+(WtSet(Irow,M))**2
      END DO

      A0lamb=Sum

      RETURN
      END
C*********************************************************************

C.....This routine calculates the value of the functions Λ_i^(1), Λ_i^(3), and Λ_i^(5) given the set
   of kriging weights λ_ij and a neighborhood of data.

      FUNCTION Cnlamb (Nhood,Irow,WtSet,Radii,k)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      DIMENSION WtSet(MAXnhd,MAXnhd), Radii(MAXnhd,MAXnhd)

      Sum=0.d0

      DO M=1,Nhood
          DO N=1,Nhood
              IF (M .EQ. N) THEN
                  radius=0.d0
              ELSE
                  radius=Radii(M,N)
              END IF
              Term=WtSet(Irow,M)*WtSet(Irow,N)*(radius**k)
              Sum=Sum+Term
          END DO
      END DO

      Cnlamb=Sum

      RETURN
      END
C*********************************************************************

C.....This routine calculates the value of the spatial increment Y_e(s_i) given the set of
   kriging weights λ_ij.

      FUNCTION SpaInc (Nhood,Irow,WtSet,Datset)
```

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      DIMENSION WtSet(MAXnhd,MAXnhd), Datset(MAXnhd,3)

      Sum=0.d0

      DO M=1,Nhood
          Sum=Sum+WtSet(Irow,M)*Datset(M,3)
      END DO

      SpaInc=Sum

      RETURN
      END
C*****************************************************************

C.....This routine calculates the percent relative difference between two values X1 and
   X2.

      FUNCTION Reldif (Ndim,X1,X2)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (MAXnhd=20, MAXdim=4)
      DIMENSION  X1(MAXdim), X2(MAXdim)

      errmax=1.0d-20

      DO i=1,Ndim
          IF (X1(i) .EQ. 0.d0) THEN
              CONTINUE
          ELSE
              pdiff=ABS((X1(i)-X2(i))/X2(i))
              IF (pdiff .GT. errmax) errmax=pdiff
          END IF
      END DO

      Reldif=errmax

      RETURN
      END
C*****************************************************************

C.....This routine conducts a goodness-of-fit test on two or more coefficient solutions
   (i.e. covariance functions) to determine which one provides the best fit to the
   correlation structure of the given neighborhood.  The goodnes-of fit equation is
   described in the text section.

      SUBROUTINE Select (Nhood,Nu,Npass,Subl,Radii,BestCf)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (MAXnhd=20, MAXpas=15)
```

```
      DIMENSION Subl(MAXnhd,3), BestCf(4), Passng(MAXpas,4),
     &          Eta(MAXpas), WtSet(MAXnhd,MAXnhd), Cf(4)

      COMMON /GSCval/ GSCcf(15,4,0:2)
      COMMON /COPchk/ Permis(15,1,0:2)
      DOUBLE PRECISION GSCcf
      CHARACTER*4 Permis

C.....Assign the number of possible GSC forms given the order v.  (This does not mean the
   number of actual forms being tested.  That number is stored by the variable Npass.)

      IF (Nu .EQ. 0) Nforms=3
      IF (Nu .EQ. 1) Nforms=7
      IF (Nu .EQ. 2) Nforms=15

C.....Now, copy the passing forms of the GSC from the COMMON array GSCcf()
C     into the local array Passng().

      icount=0
      DO m=1,Nforms
          IF (Permis(m,1,Nu) .EQ. 'Pass') THEN
              icount=icount+1
              DO n=1,4
                  Passng(icount,n)=GSCcf(m,n,Nu)
              END DO
          ELSE
              CONTINUE
          END IF
      END DO
      IF (icount .NE. Npass) THEN
          WRITE (*,*)'  '
          WRITE (*,*)' **ERROR**  Values for icount and Npass in subroutine
     &                Select are not equal.'
          STOP
      END IF

C.....Now, determine the value of Eta for each of the passing GSC forms.

      DO m=1,Npass
          DO n=1,4
              Cf(n)=Passng(m,n)
          END DO
          CALL NewWts (Nhood,Nu,Subl,Cf,WtSet)
          Ysum=0.d0
          Asum=0.d0
          DO i=1,Nhood
              Yi=SpaInc(Nhood,i,WtSet,Subl)
              Ai=Afunc(Nhood,i,WtSet,Radii,Cf)
              Ysum=Ysum+Yi**2
              Asum=Asum+Ai
          END DO
```

```
            Eta(m)=Ysum/Asum
        END DO

C.....Now, determine which of the GSC forms produced the value of Eta closest
C       to one.  This form is the best of the set.

        difmin=1.0d20
        index=0
        DO m=1,Npass
            dif=ABS(1.d0-Eta(m))
            IF (dif .LT. difmin) THEN
                difmin=dif
                index=m
            END IF
        END DO

        DO n=1,4
            BestCf(n)=Passng(index,n)
        END DO

        RETURN
        END
C*****************************************************************
```

C.....This routine calculates the value of the the function A_i given the set of kriging weights $\lambda_{i,j}$, a neighborhood of data points, and a covariance function.

```
        FUNCTION Afunc (Nhood,Irow,WtSet,Radii,Coeffs)

        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        PARAMETER (MAXnhd=20)
        DIMENSION WtSet(MAXnhd,MAXnhd), Radii(MAXnhd,MAXnhd), Coeffs(4)

        A0=Coeffs(1)
        C0=Coeffs(2)
        C1=Coeffs(3)
        C2=Coeffs(4)
        Sum=0.d0

        DO m=1,Nhood
            DO n=1,Nhood
                IF (n .EQ. m) THEN
                    rlag=0.d0
                ELSE
                    rlag=Radii(m,n)
                END IF
                covar=GSC(A0,C0,C1,C2,rlag)
                Term=WtSet(Irow,m)*WtSet(Irow,n)*covar
                Sum=Sum+Term
```

```
            END DO
        END DO

        Afunc=Sum

        RETURN
        END
C*****************************************************************

C.....This routine constructs the left-hand-side covariance matrix of the system of
   kriging equations.

        SUBROUTINE Kmatrx (Ndim,Iext,A0,C0,C1,C2,Datset,AK)

        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        PARAMETER (MAXnhd=20)
        DIMENSION AK(MAXnhd+6,MAXnhd+6), Polynm(MAXnhd,6),
       &          Datset(MAXnhd,3)

C.....Zero the values of the elements in AK.
        DO i=1,MAXnhd+6
            DO j=1,MAXnhd+6
                AK(i,j)=0.d0
            END DO
        END DO

C.....STEP 1:  Calculate the GSC values which comprise the core of the K matrix.  The
   matrix is symmetric; therefore, redundant calculations are eliminated.

        Cov0=GSC(A0,C0,C1,C2,0.d0)
        DO I=1,Ndim
            AK(I,I)=Cov0
        END DO

        DO I=1,Ndim-1
            DO J=I+1,Ndim
                r=Dist(Datset(I,1),Datset(I,2),
     &                 Datset(J,1),Datset(J,2))
                AK(I,J)=GSC(A0,C0,C1,C2,r)
                AK(J,I)=AK(I,J)
            END DO
        END DO

C.....STEP 2:  Calculate the spatial monomials and enter them into
C       the appropriate locations in the K matrix.

        DO I=1,Ndim
            Polynm(I,1)=1.d0
            Polynm(I,2)=Datset(I,1)
            Polynm(I,3)=Datset(I,2)
            Polynm(I,4)=(Datset(I,1))**2
```

```
            Polynm(I,5)=Dataset(I,1)*Dataset(I,2)
            Polynm(I,6)=(Dataset(I,2))**2
        END DO

        DO I=1,Ndim
            DO J=1,Iext
                AK(I,J+Ndim)=Polynm(I,J)
                AK(J+Ndim,I)=Polynm(I,J)
            END DO
        END DO

C.....STEP 3:  Fill in the zero block of the K matrix.

        DO I=Ndim+1,Ndim+Iext
            DO J=Ndim+1,Ndim+Iext
                AK(I,J)=0.d0
            END DO
        END DO

        RETURN
        END
C*****************************************************************

C.....This routine constructs the right-hand-side covariance vector of the system of
   kriging equations.

        SUBROUTINE Kvectr (Ndim,Iext,Xsk,Ysk,A0,C0,C1,C2,Dataset,BK)

        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        PARAMETER (MAXnhd=20)
        DIMENSION BK(MAXnhd+6), Dataset(MAXnhd,3), Polynm(6)

C.....Zero the values of the elements in BK.
        DO i=1,MAXnhd+6
            BK(i)=0.d0
        END DO

C.....STEP 1:  Calculate the GSC segment of the vector.

        DO I=1,Ndim
            r=Dist(Dataset(I,1),Dataset(I,2),Xsk,Ysk)
            BK(I)=GSC(A0,C0,C1,C2,r)
        END DO

C.....STEP 2:  Calculate the polynomial segment of the vector.

        Polynm(1)=1.d0
        Polynm(2)=Xsk
        Polynm(3)=Ysk
        Polynm(4)=Xsk**2
        Polynm(5)=Xsk*Ysk
```

```
      Polynm(6)=Ysk**2

      DO I=1,Iext
         BK(I+Ndim)=Polynm(I)
      END DO

      RETURN
      END
C*****************************************************************
C.....This subroutine solves the system of linear equations A*x=b by Gauss-Jordan
   elimination with full pivoting.  The solution vector (x) is passed back to the calling
   program through the array which initially contains the right-hand-side vector, B.  The
   the routine was taken from Numerical Recipes (FORTRAN), 1992 edition.

      SUBROUTINE SolvGJ (A,N,NP,B,M,MP)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (NMAX=50)

      DIMENSION A(NP,NP),B(NP,MP),IPIV(NMAX),INDXR(NMAX),INDXC(NMAX)
      DO 11 J=1,N
        IPIV(J)=0
11    CONTINUE
      DO 22 I=1,N
        BIG=0.
        DO 13 J=1,N
          IF(IPIV(J).NE.1)THEN
            DO 12 K=1,N
              IF (IPIV(K).EQ.0) THEN
                IF (ABS(A(J,K)).GE.BIG)THEN
                  BIG=ABS(A(J,K))
                  IROW=J
                  ICOL=K
                ENDIF
              ELSE IF (IPIV(K).GT.1) THEN
                PAUSE 'Singular matrix'
              ENDIF
12          CONTINUE
          ENDIF
13      CONTINUE
        IPIV(ICOL)=IPIV(ICOL)+1
        IF (IROW.NE.ICOL) THEN
          DO 14 L=1,N
            DUM=A(IROW,L)
            A(IROW,L)=A(ICOL,L)
            A(ICOL,L)=DUM
14        CONTINUE
          DO 15 L=1,M
            DUM=B(IROW,L)
            B(IROW,L)=B(ICOL,L)
            B(ICOL,L)=DUM
```

```
15          CONTINUE
          ENDIF
          INDXR(I)=IROW
          INDXC(I)=ICOL
          IF (A(ICOL,ICOL).EQ.0.) THEN
            WRITE (*,*) 'Singular matrix.'
            STOP
          END IF
          PIVINV=1./A(ICOL,ICOL)
          A(ICOL,ICOL)=1.
          DO 16 L=1,N
            A(ICOL,L)=A(ICOL,L)*PIVINV
16        CONTINUE
          DO 17 L=1,M
            B(ICOL,L)=B(ICOL,L)*PIVINV
17        CONTINUE
          DO 21 LL=1,N
            IF(LL.NE.ICOL)THEN
              DUM=A(LL,ICOL)
              A(LL,ICOL)=0.
              DO 18 L=1,N
                A(LL,L)=A(LL,L)-A(ICOL,L)*DUM
18            CONTINUE
              DO 19 L=1,M
                B(LL,L)=B(LL,L)-B(ICOL,L)*DUM
19            CONTINUE
            ENDIF
21        CONTINUE
22      CONTINUE
        DO 24 L=N,1,-1
          IF(INDXR(L).NE.INDXC(L))THEN
            DO 23 K=1,N
              DUM=A(K,INDXR(L))
              A(K,INDXR(L))=A(K,INDXC(L))
              A(K,INDXC(L))=DUM
23          CONTINUE
          ENDIF
24      CONTINUE
        RETURN
        END
C*****************************************************************

C.....This routine solves the point estimate x̂(s) given a set of kriging weights.

        SUBROUTINE Estmat (Ndim,Datset,Wts,Est)

        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        PARAMETER (MAXnhd=20)
        DIMENSION Datset(MAXnhd,3),Wts(MAXnhd+6)
```

```
C  Calculate the estimate at the point Sk.
      Est=0.d0
      DO I=1,Ndim
         Est=Est+Datset(I,3)*Wts(I)
      END DO

      RETURN
      END
C*****************************************************************

C.....This routine solves the estimation error σ²ₑ(s).

      SUBROUTINE EstErr (Ndim,Iext,A0,C0,C1,C2,Wts,BK,Error)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      DIMENSION BK(MAXnhd+6), Wts(MAXnhd+6)

      Sum=0.d0
      DO I=1,Ndim+Iext
         Sum=Sum+Wts(I)*BK(I)
      END DO
      Cov0=GSC(A0,C0,C1,C2,0.d0)

      Error=Cov0-Sum

      RETURN
      END
C*****************************************************************

C.....This function calculates the scalar distance between two points in 2d space.

   FUNCTION Dist (X1,Y1,X2,Y2)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      Dist=DSQRT((X2-X1)**2+(Y2-Y1)**2)

      RETURN
      END
C*****************************************************************

C.....This function calculates the value of the polynomial GSC function given a lag and a
   set of coefficients {a₀,c₀,c₁,c₂}.

      FUNCTION GSC (A0,C0,C1,C2,rlag)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      IF (rlag .NE. 0.d0) THEN
         delta=0.d0
      ELSE
```

```
            delta=1.d0
         END IF
         GSC=(A0*delta)-(C0*rlag)+(C1*rlag**3)-(C2*rlag**5)

         RETURN
         END
C*****************************************************************

C.....This function calculates the value of the function α(ν) used to determine the number
   of monomial spatial coordinate functions needed in the system of kriging equations.

         FUNCTION Ialpha (nu)

         IMPLICIT INTEGER (I-N)
         Ialpha=((nu+1)*(nu+2))/2

         RETURN
         END
C*****************************************************************
C*****************************************************************
C*****************************************************************
```

Appendix B:
Source Code for Poly-Exponential GSC

This appendix contains the FORTRAN source code for the intrinsic kriging program which implements the poly-exponential GSC model. The code is nearly identical to that version of the program which uses the polynomial GSC model with the exception of those subroutines an functions which conduct the solution of the covariance coefficient value, *b*. Therefore, the code is sparsely commented. The reader is referred to the polynomial source code for detailed comments and variable definitions.

```
************************************************************************
C..This is the driving program for a set of routines which conducts
C  intrinsic kriging. The program advances through a user-defined grid of
C  kriging nodes.  At each node, the order of intrinsity, the coeffs
C  of the GSC-v, the kriging point estimate, and the estimation error
C  variance are calculated and saved in a data file.  This program
C  implements the poly-exponential GSC model.

C..The program conducts the following steps:
C     *Reads in the set of data used for kriging
C     *Assigns a kriging node (Sk)
C     *Finds the neighborhood of data points around Sk
C     *Determines the order of intrinsity of the neighborhood
C     *Determines the form and coeffient values for the GSC-v
C     *Calculates the kriging point estimate
C     *Calculates the estimation error variance
C     *Repeats the above steps for all points in the grid

$LARGE
$DEBUG
      IMPLICIT REAL*8 (A-H,O-Z)

C.....Set the run parameters for the diagnostic test runs.
      PARAMETER (Ndata=75, Nhood=13)
      PARAMETER (Nnx=151, Nny=26, Xo=0.d0, Yo=0.02d0)
      PARAMETER (dX=0.01d0, dY=0.01d0)

      PARAMETER (MAXdat=100, MAXnhd=20)
      REAL*8 Data(MAXdat,3), Sub1(MAXnhd,3),
     &       Radii(MAXnhd,MAXnhd), Akm(MAXnhd+6,MAXnhd+6),
     &       Bkv(MAXnhd+6), Bkvdup(MAXnhd+6), Xvect(MAXnhd+6)

      COMMON  ncvg, icvg    !Block based in main prgm
      COMMON /Serch/ ibul, ispot            !Block based in main prgm
      INTEGER*2 ncvg, icvg, ibul, ispot

      OPEN (10,File='b:\soilwtr2.txt',IOSTAT=io10)
      OPEN (20,File='b:\trash1.out',IOSTAT=io20)
      OPEN (25,File='b:\trash2.out',IOSTAT=io25)
      OPEN (30,File='b:\trash3.out',IOSTAT=io30)
      WRITE (*,'(4(2x,i5))') io10, io20, io25, io30
```

```
C.....Read the input data set from the specified disk file.
      READ (10,*) (idumb, Data(i,1), Data(i,2), Data(i,3), i=1,Ndata)

      WRITE (*,2020) (Data(I,1), Data(I,2), Data(I,3), I=1,Ndata)
 2020 FORMAT (2(2x,f16.8),2x,f10.3)

C.....Initialize some values and execute the main body of the program:
the
C     loop which iterates through the grid of kriging nodes.

      ncvg=0
      Nn=Nnx*Nny
      Ncurnt=1
      Ysk=Yo

      DO Iny=1,Nny
          Xsk=Xo

          DO Inx=1,Nnx

              icvg=1          !"1"=converged GSC coeffs; "0"=non-converged
              ibul=0          !"0"=no bullseye; "1"=bullseye
              WRITE (*,1090) Ncurnt, Nn, ncvg
              WRITE (30,1090) Ncurnt, Nn, ncvg
 1090         FORMAT (' Processing node ',i4,' of ',i4,' : Ncvg = ',i4)

              CALL Search (Ndata,Nhood,Data,Xsk,Ysk,Subl)
              CALL Rmatrx (Nhood,Xsk,Ysk,Subl,Radii)
              CALL Intrin (Nhood,Subl,Nu)
              CALL CovFnc (Nhood,Nu,Subl,Radii,b)

C.............Now the order and coefficient of the EGSC-v have been found
C             for the current kriging node (Xsk,Ysk).  The kriging point
C             estimate and estimation error variance can now be
determined.

              IF (ibul .EQ. 1) THEN   !ibul=1 indicates a bullseye
                  Est=Data(ispot,3)   !ispot is data pt wh/has been
bullseyed
                  Err=0.d0
              ELSE IF (ibul .EQ. 0) THEN
                  Iext=Ialpha(Nu)
                  CALL Kmatrx (Nhood,Iext,Nu,b,'Exp',Subl,Akm)
                  CALL Kvectr (Nhood,Iext,Nu,Xsk,Ysk,b,'Exp',Subl,Bkv)

                  DO m=1,Nhood+6
                      Bkvdup(m)=Bkv(m)          !Copy the original Bnu()
                  END DO

                  Ne=Nhood+Iext
                  CALL SolvGJ (Akm,Ne,26,Bkvdup,1,1)
                  DO ko=1,Ne
                      Xvect(ko)=Bkvdup(ko)
                  END DO

                  CALL Estmat (Nhood,Subl,Xvect,Est)
                  CALL EstErr (Nhood,Iext,Nu,b,Xvect,Bkv,Err)
              END IF

              WRITE (20,2000) Xsk,Ysk,Est,Err,Nu,b
 2000         FORMAT (2(f12.7,2x),2(e15.7,2x),i1,2x,e15.7)
```

```
                IF (icvg .EQ. 0) THEN
                    WRITE (25,2010) Inx,Iny,Xsk,Ysk
 2010               FORMAT (2(2x,i4),2(2x,f12.2))
                END IF

                Ncurnt=Ncurnt+1
                Xsk=Xsk+dX
            END DO

            Ysk=Ysk+dY
        END DO

        WRITE (20,'(a,i4)') 'Number of GSC models not converging:', ncvg
        WRITE (25,'(a,i4)') 'Number of GSC models not converging:', ncvg

        STOP
        END
C*****************************************************************

        SUBROUTINE Search (Ndata,Nhood,Data,Xsk,Ysk,Sub1)

        IMPLICIT REAL*8 (A-H,O-Z)
        PARAMETER (MAXdat=100, MAXnhd=20)
        REAL*8 Data(MAXdat,3), Radius(MAXdat), Sub1(MAXnhd,3)
        INTEGER*4 Iorder(MAXdat)

        COMMON /Serch/ ibul, ispot        !Block based in main prgm
        INTEGER*2 ibul, ispot

C  Calculate the distance between the data points and Sk.
        DO I=1,Ndata
            Radius(I)=Dist(Data(I,1),Data(I,2),Xsk,Ysk)
            Iorder(I)=I
        END DO

C  Sort the Nhood+1 smallest values in Radius(), leaving the rest
unsorted.
C  Although there are only Nhood points to comprising the neighborhood,
one
C  extra value is sorted in the event that there is a "bullseye" and the
C  closest point cannot be used.
        DO I=1,Nhood+1
            DO J=I+1,Ndata
                IF (Radius(J) .LT. Radius(I)) THEN
                    CALL RSwap(Radius(I),Radius(J))
                    CALL ISwap(Iorder(I),Iorder(J))
                END IF
            END DO
        END DO

C  Check for a bullseye in the smallest radius value, ie the 1st array
elem.
        IF (Radius(1) .EQ. 0.d0) THEN
            ibul=1                  !"0"=no bullseye; "1"=bullseye
            istep=1                 !"1"=don't include 1st data pt. in nhood
            ispot=Iorder(1)         !this is the index # of bullseyed data pt.
        ELSE
            istep=0                 !"0"=include 1st data pt. in nhood
        END IF

C  Now set the values of Sub1() to the appropriate values from Data().
```

```
      DO I=1,Nhood
          Sub1(I,1)=Data(Iorder(I+istep),1)
          Sub1(I,2)=Data(Iorder(I+istep),2)
          Sub1(I,3)=Data(Iorder(I+istep),3)
      END DO

      RETURN
      END
C-------------------------------------
      SUBROUTINE RSwap (arg1,arg2)

      IMPLICIT REAL*8 (A-H,O-Z)

      temp=arg1
      arg1=arg2
      arg2=temp

      RETURN
      END
C-------------------------------------
      SUBROUTINE ISwap (iarg1,iarg2)

      IMPLICIT INTEGER*4 (I-N)

      itemp=iarg1
      iarg1=iarg2
      iarg2=itemp

      RETURN
      END
C*********************************************************************

      SUBROUTINE Rmatrx (Nhood,Xsk,Ysk,Dataset,Radii)

      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      REAL*8 Dataset(MAXnhd,3), Radii(MAXnhd,MAXnhd)

C  STEP 1:  Calculate the distances between the points in Dataset and
C  the central node Sk.  Store these values on the diagonal of Radii().

      DO I=1,Nhood
          r=Dist(Xsk,Ysk,Dataset(I,1),Dataset(I,2))
          Radii(I,I)=r
      END DO

C  STEP 2:  Calculate the distances between all pairs of points in
Dataset().
C  Note that corresponding entries in the upper and lower triangle of
C  the matrix are equivalent.

      DO I=1,Nhood-1
          DO J=I+1,Nhood
              r=Dist(Dataset(I,1),Dataset(I,2),
     &                Dataset(J,1),Dataset(J,2))
              Radii(I,J)=r                    !Upper Triangle Entries
              Radii(J,I)=r                    !Lower Triangle Entries
          END DO
      END DO

      RETURN
      END
```

```
C*********************************************************************

      SUBROUTINE Intrin (Nhood,Sub1,Nu)

      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      REAL*8 Sub1(MAXnhd,3), Sub2(MAXnhd,3), Diff(0:2),
     &       Anu(MAXnhd+6,MAXnhd+6), Bnu(MAXnhd+6), Xnu(MAXnhd+6)
      INTEGER*4 Irank(MAXnhd,0:2)

      Nh=Nhood-1       ! Nh is the # of points in Sub2().

      DO I=1,Nhood

          CALL Remove (Nhood,I,Sub1,Sub2)

          DO Inu=0,2
              Ialfa=Ialpha(Inu)      !Determine the spatial polynom buffer
              CALL Kmatrx (Nh,Ialfa,Inu,0.d0,'Lin',Sub2,Anu)
              CALL Kvectr (Nh,Ialfa,Inu,Sub1(I,1),Sub1(I,2),
     &                     0.d0,'Lin',Sub2,Bnu)

              Ne=Nh+Ialfa
              CALL SolvGJ (Anu,Ne,26,Bnu,1,1)
              DO ko=1,Ne
                  Xnu(ko)=Bnu(ko)
              END DO

              CALL Estmat (Nh,Sub2,Xnu,Est)
              Diff(Inu)=ABS(Est-Sub1(I,3))    !Calc the error in Est
          END DO

          CALL Rank (I,Diff,Irank)
      END DO

      CALL SetNu (Nhood,Irank,Nu)

      RETURN
      END
C-------------------------------------

      SUBROUTINE Remove (Nhood,IndexPt,Sub1,Sub2)

      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      REAL*8 Sub1(MAXnhd,3), Sub2(MAXnhd,3)

      IF (IndexPt .EQ. 1) THEN
          DO I=2,Nhood
              Sub2(I-1,1)=Sub1(I,1)
              Sub2(I-1,2)=Sub1(I,2)
              Sub2(I-1,3)=Sub1(I,3)
          END DO
      ELSE
          Sub2(IndexPt-1,1)=Sub1(IndexPt-1,1)
          Sub2(IndexPt-1,2)=Sub1(IndexPt-1,2)
          Sub2(IndexPt-1,3)=Sub1(IndexPt-1,3)
      END IF

      RETURN
      END
C-------------------------------------
```

```
      SUBROUTINE Rank (Ipoint,Diff,Irank)

      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      REAL*8 Diff(0:2)
      INTEGER*4 Irank(MAXnhd,0:2), Iorder(0:2)

C  Initialize the array Iorder().
      Iorder(0)=0
      Iorder(1)=1
      Iorder(2)=2

C  Sort the estimation error variances in increasing order.
      DO I=0,2
          DO J=I+1,2
              IF (Diff(J) .LT. Diff(I)) THEN
                  CALL Rswap (Diff(I),Diff(J))
                  CALL Iswap (Iorder(I),Iorder(J))
              END IF
          END DO
      END DO

C  Assign ranks according to the sorted variance values.
      Irank(Ipoint,Iorder(0))=1
      Irank(Ipoint,Iorder(1))=2
      Irank(Ipoint,Iorder(2))=3

      RETURN
      END
C-------------------------------------

      SUBROUTINE SetNu (Nhood,Irank,Nu)

      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      INTEGER*4 Irank(MAXnhd,0:2)

C  Initialize the summation variables.
      Isum0=0
      Isum1=0
      Isum2=0

C  Add up the rank values for order of intrinsity.
      DO I=1,Nhood
          Isum0=Isum0+Irank(I,0)
          Isum1=Isum1+Irank(I,1)
          Isum2=Isum2+Irank(I,2)
      END DO

C  Test if two or more of the rank sums are equal and smaller than
C  the other rank sum.  If so, then assign v subjectively to the
C  smallest possible order.
      IF ((Isum0 .EQ. Isum1) .AND. (Isum0 .LT. Isum2)) THEN
          Nu=0
          RETURN
      ELSE IF ((Isum0 .EQ. Isum2) .AND. (Isum0 .LT. Isum1)) THEN
          Nu=0
          RETURN
      ELSE IF ((Isum1 .EQ. Isum2) .AND. (Isum1 .LT. Isum0)) THEN
          Nu=1
          RETURN
```

```
      ELSE IF ((Isum0 .EQ. Isum1) .AND. (Isum0 .EQ. Isum2)) THEN
          Nu=0
          RETURN
      END IF

C  Determine which rank sum is the lowest and set Nu to its corre-
C  sponding order of intrinsity.

      Minsum=MIN(Isum0,Isum1,Isum2)
      IF (Minsum .EQ. Isum0) THEN
          Nu=0
      ELSE IF (Minsum .EQ. Isum1) THEN
          Nu=1
      ELSE IF (Minsum .EQ. Isum2) THEN
          Nu=2
      END IF

      RETURN
      END

C*********************************************************************
C..This routine determines the value of the exponential GSC coeff
C  b by a Newton-Raphson algorithm to find the minimum value of
C  the function F wrt b.

      SUBROUTINE CovFnc (Nhood,Nu,Sub1,Radii,bfinal)

      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      REAL*8 Sub1(MAXnhd,3),Radii(MAXnhd,MAXnhd)
      LOGICAL check

      COMMON  ncvg, icvg                    !Based in main program
      INTEGER*2 ncvg, icvg

C.....Assign an initial guess for b.

      PARAMETER (b0=1.d0)

C.....Call the subroutine newt which finds a root to the dF/db=0 equation
C     by a glogally-convergent Newton's method.  The method is adapted
C     from Press, et al (1992).

      b=b0
      CALL newtz(Nhood,Nu,Radii,Sub1,b,1,check)
      WRITE (*,*) 'Passed newtz.'
      bfinal=b

      RETURN
      END
C*********************************************************************

      SUBROUTINE newtz(Nhood,Nu,Radii,Sub1,x,n,check)

      INTEGER n,nn,NP,MAXITS
      LOGICAL check
      DOUBLE PRECISION x(n),fvec,TOLF,TOLMIN,TOLX,STPMX
      PARAMETER (NP=40,MAXITS=120,TOLF=1.d-4,TOLMIN=1.d-6,TOLX=1.d-7,
     &           STPMX=100.d0,MAXnhd=20)
      COMMON /newtv/ fvec(NP),nn
      SAVE /newtv/
      INTEGER i,its,j,indx(NP)
```

```
      DOUBLE PRECISION d,den,f,fold,stpmax,sum,temp,test,fjac(NP,NP),
     &                 g(NP),p(NP),xold(NP),fminz,WtSet(MAXnhd,MAXnhd),
     &                 Radii(MAXnhd,MAXnhd),Subl(MAXnhd,3)
      EXTERNAL fminz

      CALL NewWts(Nhood,Nu,Subl,x,WtSet)
      WRITE (*,*) 'Passed NewWts.',WtSet(1,1),WtSet(1,2)

      nn=n
      f=fminz(Nhood,Nu,WtSet,Radii,Subl,x)
      WRITE (*,*) 'Passed fminz.',f
      test=0.
      do 11 i=1,n
        if(abs(fvec(i)).gt.test)test=abs(fvec(i))
11    continue
      if(test.lt..01*TOLF)return
      sum=0.
      do 12 i=1,n
        sum=sum+x(i)**2
12    continue
      stpmax=STPMX*max(sqrt(sum),float(n))
      do 21 its=1,MAXITS

        WRITE (*,'(i3,2x,f18.8)') its,x(1)

        call anljac(Nhood,Nu,WtSet,Radii,Subl,x,fjac)
        WRITE (*,*) 'Passed anljac.',fjac(1,1),fjac(1,2)

        do 14 i=1,n
          sum=0.
          do 13 j=1,n
            sum=sum+fjac(j,i)*fvec(j)
13        continue
          g(i)=sum
14      continue
        do 15 i=1,n
          xold(i)=x(i)
15      continue
        fold=f
        do 16 i=1,n
          p(i)=-fvec(i)
16      continue
        call ludcmp(fjac,n,NP,indx,d)
        WRITE (*,*) 'Passed ludcmp.'
        call lubksb(fjac,n,NP,indx,p)
        WRITE (*,*) 'Passed lubksb.'
        call linez(n,xold,fold,g,p,x,f,stpmax,check,fminz)
        WRITE (*,*) 'Passed linez.'
        test=0.
        do 17 i=1,n
          if(abs(fvec(i)).gt.test)test=abs(fvec(i))
17      continue
        if(test.lt.TOLF)then
          check=.false.
          return
        endif
        if(check)then
          test=0.
          den=max(f,.5*n)
          do 18 i=1,n
            temp=abs(g(i))*max(abs(x(i)),1.)/den
            if(temp.gt.test)test=temp
```

```
18        continue
          if(test.lt.TOLMIN)then
            check=.true.
          else
            check=.false.
          endif
          return
        endif
        test=0.
        do 19 i=1,n
          temp=(abs(x(i)-xold(i)))/max(abs(x(i)),1.)
          if(temp.gt.test)test=temp
19      continue
        if(test.lt.TOLX)return
21    continue
      pause 'MAXITS exceeded in newt'
      END
C  (C) Copr. 1986-92 Numerical Recipes Software #4-UZ2.
C*********************************************************************

      SUBROUTINE linez(n,xold,fold,g,p,x,f,stpmax,check,func)

      IMPLICIT DOUBLE PRECISION (A-H,O,Z)
      INTEGER n
      LOGICAL check
      DOUBLE PRECISION f,fold,stpmax,g(n),p(n),x(n),xold(n),
     &                 func,ALF,TOLX
      PARAMETER (ALF=1.d-4,TOLX=1.d-7)
      EXTERNAL func
CU    USES func
      INTEGER i
      DOUBLE PRECISION a,alam,alam2,alamin,b,disc,f2,fold2,rhs1,
     &                 rhs2,slope,sum,temp,test,tmplam

      check=.false.
      sum=0.
      do 11 i=1,n
        sum=sum+p(i)*p(i)
11    continue

      sum=sqrt(sum)
      if(sum.gt.stpmax)then
        do 12 i=1,n
          p(i)=p(i)*stpmax/sum
12      continue
      endif

      slope=0.
      do 13 i=1,n
        slope=slope+g(i)*p(i)
13    continue

      test=0.
      do 14 i=1,n
        temp=abs(p(i))/max(abs(xold(i)),1.)
        if(temp.gt.test)test=temp
14    continue
      alamin=TOLX/test
      alam=1.
1     continue
        do 15 i=1,n
          x(i)=xold(i)+alam*p(i)
```

```
15      continue
        f=func(x)
        if(alam.lt.alamin)then
          do 16 i=1,n
            x(i)=xold(i)
16        continue
          check=.true.
          return
        else if(f.le.fold+ALF*alam*slope)then
          return
        else
          if(alam.eq.1.)then
            tmplam=-slope/(2.*(f-fold-slope))
          else
            rhs1=f-fold-alam*slope
            rhs2=f2-fold2-alam2*slope
            a=(rhs1/alam**2-rhs2/alam2**2)/(alam-alam2)
            b=(-alam2*rhs1/alam**2+alam*rhs2/alam2**2)/(alam-alam2)
            if(a.eq.0.)then
              tmplam=-slope/(2.*b)
            else
              disc=b*b-3.*a*slope
              tmplam=(-b+sqrt(disc))/(3.*a)
            endif
            if(tmplam.gt..5*alam)tmplam=.5*alam
          endif
        endif
        alam2=alam
        f2=f
        fold2=fold
        alam=max(tmplam,.1*alam)
      goto 1
      END
C  (C) Copr. 1986-92 Numerical Recipes Software #4-UZ2.
C*****************************************************************

C.....This subroutine solves the value of the function f(b) given b.

      SUBROUTINE funcfb(Nhood,Nu,WtSet,Radii,Sub1,b,fb)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      DOUBLE PRECISION Sub1(MAXnhd,3),Radii(MAXnhd,MAXnhd),
     &                 WtSet(MAXnhd,MAXnhd)

      Sum=0.d0

      DO i=1,Nhood
          Yval=Yqi(Nhood,i,WtSet,Sub1)
          Aval=Ai(Nhood,i,Nu,WtSet,Radii,b)
          dlAval=dlAi(Nhood,i,Nu,WtSet,Radii,b)

          fbi=(Yval**2-Aval)*(-1)*dlAval
          Sum=Sum+fbi
      END DO

      fb=2.d0*Sum

C.....Add the terms of the penalty function.
      fb=fb+(-0.001d0/b**2)

      RETURN
```

```
      END
C*********************************************************************

C.....This subroutine solves the value of the function df/db given b.

      SUBROUTINE anljac(Nhood,Nu,WtSet,Radii,Sub1,b,d1fB)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      DOUBLE PRECISION Sub1(MAXnhd,3),Radii(MAXnhd,MAXnhd),
     &                 WtSet(MAXnhd,MAXnhd)

      Sum=0.d0

      DO i=1,Nhood
          Yval=Yqi(Nhood,i,WtSet,Sub1)
          Aval=Ai(Nhood,i,Nu,WtSet,Radii,b)
          d1Aval=d1Ai(Nhood,i,Nu,WtSet,Radii,b)
          d2Aval=d2Ai(Nhood,i,Nu,WtSet,Radii,b)

          d1fbi=((Yval**2-Aval)*(-1)*d2Aval)-((-1)*d1Aval**2)
          Sum=Sum+d1fbi
      END DO

      d1fb=2.d0*Sum

C.....Add the terms of the penalty function.
      d1fb=d1fb+(0.002d0/b**3)

      RETURN
      END
C*********************************************************************

      FUNCTION fminz(Nhood,Nu,WtSet,Radii,Sub1,x)

      IMPLICIT DOUBLE PRECISION (A-H,O,Z)
      INTEGER n,NP
      DOUBLE PRECISION fminz,x(*),fvec
      PARAMETER (NP=40,MAXnhd=20)
      DOUBLE PRECISION Sub1(MAXnhd,3),Radii(MAXnhd,MAXnhd),
     &                 WtSet(MAXnhd,MAXnhd)
      COMMON /newtv/ fvec(NP),n
      SAVE /newtv/
CU....USES funcv
      INTEGER i
      REAL sum

      call funcfb(Nhood,Nu,WtSet,Radii,Sub1,x,fvec)

      sum=0.
      do 11 i=1,n
        sum=sum+fvec(i)**2
11    continue

      fminz=0.5*sum

      RETURN
      END
C  (C) Copr. 1986-92 Numerical Recipes Software #4-UZ2.
C*********************************************************************

      SUBROUTINE ludcmp(a,n,np,indx,d)
```

```
      IMPLICIT DOUBLE PRECISION (A-H,O,Z)

      INTEGER n,np,indx(n),NMAX
      DOUBLE PRECISION d,a(np,np),TINY
      PARAMETER (NMAX=500,TINY=1.0e-20)
      INTEGER i,imax,j,k
      DOUBLE PRECISION aamax,dum,sum,vv(NMAX)
      d=1.
      do 12 i=1,n
        aamax=0.
        do 11 j=1,n
          if (abs(a(i,j)).gt.aamax) aamax=abs(a(i,j))
11      continue
        if (aamax.eq.0.) pause 'singular matrix in ludcmp'
        vv(i)=1./aamax
12    continue
      do 19 j=1,n
        do 14 i=1,j-1
          sum=a(i,j)
          do 13 k=1,i-1
            sum=sum-a(i,k)*a(k,j)
13        continue
          a(i,j)=sum
14      continue
        aamax=0.
        do 16 i=j,n
          sum=a(i,j)
          do 15 k=1,j-1
            sum=sum-a(i,k)*a(k,j)
15        continue
          a(i,j)=sum
          dum=vv(i)*abs(sum)
          if (dum.ge.aamax) then
            imax=i
            aamax=dum
          endif
16      continue
        if (j.ne.imax)then
          do 17 k=1,n
            dum=a(imax,k)
            a(imax,k)=a(j,k)
            a(j,k)=dum
17        continue
          d=-d
          vv(imax)=vv(j)
        endif
        indx(j)=imax
        if(a(j,j).eq.0.)a(j,j)=TINY
        if(j.ne.n)then
          dum=1./a(j,j)
          do 18 i=j+1,n
            a(i,j)=a(i,j)*dum
18        continue
        endif
19    continue
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software #4-UZ2.
C*********************************************************************

      SUBROUTINE lubksb(a,n,np,indx,b)
```

```
      IMPLICIT DOUBLE PRECISION (A-H,O,Z)

      INTEGER n,np,indx(n)
      DOUBLE PRECISION a(np,np),b(n)
      INTEGER i,ii,j,ll
      DOUBLE PRECISION sum
      ii=0
      do 12 i=1,n
        ll=indx(i)
        sum=b(ll)
        b(ll)=b(i)
        if (ii.ne.0)then
          do 11 j=ii,i-1
            sum=sum-a(i,j)*b(j)
11        continue
        else if (sum.ne.0.) then
          ii=i
        endif
        b(i)=sum
12    continue
      do 14 i=n,1,-1
        sum=b(i)
        do 13 j=i+1,n
          sum=sum-a(i,j)*b(j)
13      continue
        b(i)=sum/a(i,i)
14    continue
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software #4-UZ2.
C**********************************************************************

C..This function subprogram calculates the value of the spatial increment
C  for a given point i removed from the neighborhood.

      FUNCTION Yqi (Nhood,Irow,WtSet,Data)

      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      REAL*8 WtSet(MAXnhd,MAXnhd), Data(MAXnhd,3)

      Sum=0.d0

      DO M=1,Nhood
          Sum=Sum+WtSet(Irow,M)*Data(M,3)
      END DO

      Yqi=Sum

      RETURN
      END
C**********************************************************************
C..This function calculates the expected value of the square of the
spatial
C  increment, ie the square of the estimation error for the point i
removed
C  from the neighborhood.

      FUNCTION Ai (Nhood,Irow,Nu,WtSet,Radii,b)

      IMPLICIT REAL*8 (A-H,O-Z)
```

```
      PARAMETER (MAXnhd=20)
      REAL*8 WtSet(MAXnhd,MAXnhd), Radii(MAXnhd,MAXnhd)

      Sum=0.d0

      DO m=1,Nhood
          DO n=1,Nhood
              IF (n .EQ. m) THEN
                  rlag=0.d0
              ELSE
                  rlag=Radii(m,n)
              END IF
              covar=ExpGSC(Nu,b,rlag)
              Term=WtSet(Irow,m)*WtSet(Irow,n)*covar
              Sum=Sum+Term
          END DO
      END DO

      Ai=Sum

      RETURN
      END
C*********************************************************************
C..This function calculates the first derivative of expected value of the
C  square of the spatial increment, ie the square of the estimation error
for
C  the point i removed from the neighborhood.

      FUNCTION d1Ai (Nhood,Irow,Nu,WtSet,Radii,b)

      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      REAL*8 WtSet(MAXnhd,MAXnhd), Radii(MAXnhd,MAXnhd)

      Sum=0.d0

      DO m=1,Nhood
          DO n=1,Nhood
              IF (n .EQ. m) THEN
                  rlag=0.d0
              ELSE
                  rlag=Radii(m,n)
              END IF
              d1cov=d1EGSC(Nu,b,rlag)
              Term=WtSet(Irow,m)*WtSet(Irow,n)*d1cov
              Sum=Sum+Term
          END DO
      END DO

      d1Ai=Sum

      RETURN
      END
C*********************************************************************
C..This function calculates the second derivative of expected value of the
C  square of the spatial increment for the case of Nu=0.

      FUNCTION d2Ai (Nhood,Irow,Nu,WtSet,Radii,b)

      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      REAL*8 WtSet(MAXnhd,MAXnhd), Radii(MAXnhd,MAXnhd)
```

```
      Sum=0.d0

      DO m=1,Nhood
          DO n=1,Nhood
              IF (n .EQ. m) THEN
                  rlag=0.d0
              ELSE
                  rlag=Radii(m,n)
              END IF
C.............Determine which form of the 2nd derivative EGC to call.
              IF (Nu .EQ. 0) THEN
                  d2cov=d2EGC0(b,rlag)
              ELSE IF (Nu .EQ. 1) THEN
                  d2cov=d2EGC1(b,rlag)
              ELSE IF (Nu .EQ. 2) THEN
                  d2cov=d2EGC2(b,rlag)
              END IF
C.............Calculate the value for the current term.
              Term=WtSet(Irow,m)*WtSet(Irow,n)*d2cov
              Sum=Sum+Term
          END DO
      END DO

      d2Ai=Sum

      RETURN
      END
C**********************************************************************
C..This function calculates the value of the linear form of the
C  polynomial GSC for given values of C0 and rlag.

      FUNCTION GSClin (C0,rlag)

      IMPLICIT REAL*8 (A-H,O-Z)
      GSClin=(-1.d0)*C0*rlag

      RETURN
      END
C**********************************************************************
C..This function subprogram calculates the value of the exponential
C  form of the GSC-v function given the values of Nu, b, and rlag.

      FUNCTION ExpGSC (Nu,b,rlag)

      IMPLICIT REAL*8 (A-H,O-Z)

      p=b*rlag
      n=2*Nu+1
      sum=0.d0
      DO i=0,n
          fctrl=Factrl(i)
          sum=sum+((-p)**i/fctrl)
      END DO
      Term1=((-1.d0)**(Nu+1))/(b**(n+1)*DEXP(p))
      Term2=1.d0-DEXP(p)*sum

      ExpGSC=Term1*Term2

      RETURN
      END
C-------------------------------------
```

```
      FUNCTION Factrl(n)

      IMPLICIT REAL*8 (A-H,O-Z)

      Term=1.d0
      DO i=1,n
          Term=Term*DBLE(i)
      END DO
      Factrl=Term

      RETURN
      END
C*********************************************************************
C..This function subprogram calculates the value of the first derivative
C  of the exponential GSC function using a general form (ie one that can
C  be used for any value of Nu).  Values of Nu, b, and rlag are passed
C  to the subprogram from the calling module.

      FUNCTION d1EGSC(Nu,b,rlag)

      IMPLICIT REAL*8 (A-H,O-Z)

      n=2*Nu+1
      p=2.d0*DBLE(Nu)+2.d0
      sum1=0.d0
      sum2=0.d0

      DO i=0,n

C.........Calculate the first summation term.
          fctrl=Factrl(i)
          term1=(-b*rlag)**i/fctrl
          sum1=sum1+term1

C.........Calculate the second summation term.
          term2=((-1.d0)**(i-1))*DBLE(i)*(rlag**i)*(b**(i-1))/fctrl
          sum2=sum2+term2

      END DO

      term3=(-1.d0)**(Nu+2)/(b**p)
      term4=(p/b)*(DEXP(-b*rlag)-sum1)
      term5=rlag*DEXP(-b*rlag)-sum2

      d1EGSC=term3*(term4+term5)

      RETURN
      END
C*********************************************************************
C..This function subprogram calculates the value of the second derivative
C  of the expontential GSC for the case Nu=0.  Values of b, and rlag
C  are passed in from the calling module.

      FUNCTION d2EGC0(b,rlag)

      IMPLICIT REAL*8 (A-H,O-Z)

      p=b*rlag
      ep=DEXP(p)

      term1=(-6.d0)/(ep*b**4)
```

```
      term2=(-4.d0*rlag)/(ep*b**3)
      term3=6.d0/b**4
      term4=(-2.d0*rlag)/b**3
      term5=(-1.d0)*(rlag**2)/(ep*b**2)

      d2EGC0=term1+term2+term3+term4+term5

      RETURN
      END
C*****************************************************************
C..This function subprogram calculates the value of the second derivative
C  of the expontential GSC for the case Nu=1.  Values of b, and rlag
C  are passed in from the calling module.

      FUNCTION d2EGC1(b,rlag)

      IMPLICIT REAL*8 (A-H,O-Z)

      p=b*rlag
      ep=DEXP(p)

      term1=20.d0/(ep*b**6)
      term2=(8.d0*rlag)/(ep*b**5)
      term3=(rlag**2)/(ep*b**4)
      term4=(-20.d0)/b**6
      term5=(12.d0*rlag)/b**5
      term6=(-3.d0*rlag**2)/b**4
      term7=rlag**3/(3.d0*b**3)

      d2EGC1=term1+term2+term3+term4+term5+term6+term7

      RETURN
      END
C*****************************************************************
C..This function subprogram calculates the value of the second derivative
C  of the expontential GSC for the case Nu=2.  Values of b, and rlag
C  are passed in from the calling module.

      FUNCTION d2EGC2(b,rlag)

      IMPLICIT REAL*8 (A-H,O-Z)

      p=b*rlag
      ep=DEXP(p)

      term1=(-42.d0)/(ep*b**8)
      term2=(-12.d0*rlag)/(ep*b**7)
      term3=(-1.d0)*(rlag**2)/(ep*b**6)
      term4=42.d0/b**8
      term5=(-30.d0*rlag)/b**7
      term6=(10.d0*rlag**2)/b**6
      term7=(-2.d0*rlag**3)/b**5
      term8=rlag**4/(4.d0*b**4)
      term9=(-1.d0)*(rlag**5)/(60.d0*b**3)

      d2EGC2=term1+term2+term3+term4+term5+term6+term7+term8+term9

      RETURN
      END
C*****************************************************************

      SUBROUTINE PermChk (b,Flag)
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER*4  Flag

      Flag='Pass'

      IF (b .LE. 0.d0) THEN
          Flag='Fail'
          RETURN
      END IF

      RETURN
      END
C*****************************************************************

      SUBROUTINE NewWts (Nhood,Nu,Sub1,b,WtSet)

      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (MAXdat=100, MAXnhd=20)
      REAL*8 Sub1(MAXnhd,3), Sub2(MAXnhd,3), WtSet(MAXnhd,MAXnhd),
     &       Ak(MAXnhd+6,MAXnhd+6), Bk(MAXnhd+6), Xk(MAXnhd+6)

C.....Assign some initial values to local variables.

      Nh=Nhood-1              !Nh will be the # of points in Sub2()
      Ialfa=Ialpha(Nu)        !Determine the spatial monomial buffer

C.....Now, construct the matrix of kriging weights using the exponential
C     GSC passed in by the value of b.

      DO i=1,Nhood
          CALL Remove (Nhood,i,Sub1,Sub2)
          CALL Kmatrx(Nh,Ialfa,Nu,b,'Exp',Sub2,Ak)
          CALL Kvectr(Nh,Ialfa,Nu,Sub1(i,1),Sub1(i,2),b,'Exp',Sub2,Bk)

          Ne=Nh+Ialfa
          CALL SolvGJ (Ak,Ne,26,Bk,1,1)
          DO ko=1,Ne
              Xk(ko)=Bk(ko)
          END DO

          DO j=1,Nhood
              IF (j .LT. i) THEN
                  WtSet(i,j)=Xk(j)
              ELSE IF (j .EQ. i) THEN
                  WtSet(i,j)=-1.d0
              ELSE IF (j .GT. i) THEN
                  WtSet(i,j)=Xk(j-1)
              END IF
          END DO
      END DO

      RETURN
      END
C*****************************************************************
C..This function calculates the relative error between two scalar
C  values Xnew and Xold.

      FUNCTION Reldif (Xnew,Xold)

      IMPLICIT REAL*8 (A-H,O-Z)
      value=ABS((Xnew-Xold)/Xnew)  !Relative Err, Eq6.46 NumMeth
```

```
      Reldif=value

      RETURN
      END
C*********************************************************************
C..This function calcs the value of Eta for each of the exponential GSC
C  to be used in the estimation system and error variance calculation.

      FUNCTION EtaVal (Nhood,Nu,Sub1,Radii,b)

      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      REAL*8 Sub1(MAXnhd,3), WtSet(MAXnhd,MAXnhd)

      CALL NewWts (Nhood,Nu,Sub1,b,WtSet)
      Ysum=0.d0
      Asum=0.d0

      DO i=1,Nhood
          Yval=Yqi(Nhood,i,WtSet,Sub1)
          Aval=Ai(Nhood,i,Nu,WtSet,Radii,b)
          Ysum=Ysum+Yval**2
          Asum=Asum+Aval
      END DO

      EtaVal=Ysum/Asum

      RETURN
      END
C*********************************************************************

      SUBROUTINE Kmatrx (Ndim,Iext,Nu,b,switch,Dataset,AK)

      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      REAL*8 AK(MAXnhd+6,MAXnhd+6), Polynm(MAXnhd,6),
     &       Dataset(MAXnhd,3)
      CHARACTER*3 switch

C..STEP 1:  Calculate the GSC core values of the K matrix.  The matrix
C  is symmetric; therefore, redundant calculations are eliminated. An
C  IF statement is used to check whether the calling module needs
C  values calced with the linear GSC (switch='Lin') or the
C  exponential GSC (switch='Exp').

      IF (switch .EQ. 'Lin') THEN
          Cov0=0.d0
      ELSE
          Cov0=ExpGSC(Nu,b,0.d0)
      END IF

      DO I=1,Ndim
          AK(I,I)=Cov0                          !Diagonal Entries
      END DO

      DO I=1,Ndim-1
          DO J=I+1,Ndim
              r=Dist(Dataset(I,1),Dataset(I,2),
     &               Dataset(J,1),Dataset(J,2))
              IF (switch .EQ. 'Lin') THEN
                  entry=GSClin(1.d0,r)
              ELSE
```

```
                entry=ExpGSC(Nu,b,r)
            END IF
            AK(I,J)=entry       !Upper Triangle Entries
            AK(J,I)=AK(I,J)                 !Lower Triangle Entries
        END DO
    END DO

C   STEP 2:  Calculate the spatial polynomials and enter them into
C   the appropriate locations in the K matrix.
    DO I=1,Ndim
        Polynm(I,1)=1.d0
        Polynm(I,2)=Dataset(I,1)
        Polynm(I,3)=Dataset(I,2)
        Polynm(I,4)=(Dataset(I,1))**2
        Polynm(I,5)=Dataset(I,1)*Dataset(I,2)
        Polynm(I,6)=(Dataset(I,2))**2
    END DO

    DO I=1,Ndim
        DO J=1,Iext
            AK(I,J+Ndim)=Polynm(I,J)        !Upper Polynomial Block
            AK(J+Ndim,I)=Polynm(I,J)        !Lower Polynomial Block
        END DO
    END DO

C   STEP 3:  Fill in the zero block of the K matrix.
    DO I=Ndim+1,Ndim+Iext
        DO J=Ndim+1,Ndim+Iext
            AK(I,J)=0.d0                    !Zero Block
        END DO
    END DO

    RETURN
    END
C*********************************************************************

    SUBROUTINE Kvectr (Ndim,Iext,Nu,Xsk,Ysk,b,switch,Dataset,BK)

    IMPLICIT REAL*8 (A-H,O-Z)
    PARAMETER (MAXnhd=20)
    REAL*8 BK(MAXnhd+6), Dataset(MAXnhd,3), Polynm(6)
    CHARACTER*3 switch

C   STEP 1:  Calculate the GSC segment of the vector.
    DO I=1,Ndim
        r=Dist(Dataset(I,1),Dataset(I,2),Xsk,Ysk)
        IF (switch .EQ. 'Lin') THEN
            entry=GSClin(1.d0,r)
        ELSE
            entry=ExpGSC(Nu,b,r)
        END IF
        BK(I)=entry                     !GSC Segment
    END DO

C   STEP 2:  Calculate the polynomial segment of the vector.
    Polynm(1)=1.d0
    Polynm(2)=Xsk
    Polynm(3)=Ysk
    Polynm(4)=Xsk**2
    Polynm(5)=Xsk*Ysk
    Polynm(6)=Ysk**2
```

```
      DO I=1,Iext
          BK(I+Ndim)=Polynm(I)          !Polynomial Segment
      END DO

      RETURN
      END
C*********************************************************************
C..This subroutine solves the system of linear equations A*x=b by
C  Gauss-Jordan elimination with full pivoting.  The program originally
C  came from Numerical Recipes (FORTRAN), 1992 edition.

      SUBROUTINE SolvGJ (A,N,NP,B,M,MP)

      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NMAX=50)

      DIMENSION A(NP,NP),B(NP,MP),IPIV(NMAX),INDXR(NMAX),INDXC(NMAX)
      DO 11 J=1,N
        IPIV(J)=0
11    CONTINUE
      DO 22 I=1,N
        BIG=0.
        DO 13 J=1,N
          IF(IPIV(J).NE.1)THEN
            DO 12 K=1,N
              IF (IPIV(K).EQ.0) THEN
                IF (ABS(A(J,K)).GE.BIG)THEN
                  BIG=ABS(A(J,K))
                  IROW=J
                  ICOL=K
                ENDIF
              ELSE IF (IPIV(K).GT.1) THEN
                PAUSE 'Singular matrix'
              ENDIF
12          CONTINUE
          ENDIF
13      CONTINUE
        IPIV(ICOL)=IPIV(ICOL)+1
        IF (IROW.NE.ICOL) THEN
          DO 14 L=1,N
            DUM=A(IROW,L)
            A(IROW,L)=A(ICOL,L)
            A(ICOL,L)=DUM
14        CONTINUE
          DO 15 L=1,M
            DUM=B(IROW,L)
            B(IROW,L)=B(ICOL,L)
            B(ICOL,L)=DUM
15        CONTINUE
        ENDIF
        INDXR(I)=IROW
        INDXC(I)=ICOL
        IF (A(ICOL,ICOL).EQ.0.) PAUSE 'Singular matrix.'
        PIVINV=1./A(ICOL,ICOL)
        A(ICOL,ICOL)=1.
        DO 16 L=1,N
          A(ICOL,L)=A(ICOL,L)*PIVINV
16      CONTINUE
        DO 17 L=1,M
          B(ICOL,L)=B(ICOL,L)*PIVINV
17      CONTINUE
        DO 21 LL=1,N
```

```
          IF(LL.NE.ICOL)THEN
            DUM=A(LL,ICOL)
            A(LL,ICOL)=0.
            DO 18 L=1,N
              A(LL,L)=A(LL,L)-A(ICOL,L)*DUM
18          CONTINUE
            DO 19 L=1,M
              B(LL,L)=B(LL,L)-B(ICOL,L)*DUM
19          CONTINUE
          ENDIF
21      CONTINUE
22    CONTINUE
      DO 24 L=N,1,-1
        IF(INDXR(L).NE.INDXC(L))THEN
          DO 23 K=1,N
            DUM=A(K,INDXR(L))
            A(K,INDXR(L))=A(K,INDXC(L))
            A(K,INDXC(L))=DUM
23        CONTINUE
        ENDIF
24    CONTINUE
      RETURN
      END
C*********************************************************************

      SUBROUTINE Estmat (Ndim,Dataset,Wts,Est)

      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      REAL*8 Dataset(MAXnhd,3),Wts(MAXnhd+6)

C  Calculate the estimate at the point Sk.
      Est=0.d0
      DO I=1,Ndim
          Est=Est+Dataset(I,3)*Wts(I)
      END DO

      RETURN
      END
C*********************************************************************

      SUBROUTINE EstErr (Ndim,Iext,Nu,b,Wts,BK,Error)

      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (MAXnhd=20)
      REAL*8 BK(MAXnhd+6), Wts(MAXnhd+6)

      Sum=0.d0
      DO I=1,Ndim+Iext
          Sum=Sum+Wts(I)*BK(I)
      END DO

      Cov0=ExpGSC(Nu,b,0.d0)
      Error=Cov0-Sum

      RETURN
      END
C*********************************************************************
      FUNCTION Dist (X1,Y1,X2,Y2)

      IMPLICIT REAL*8 (A-H,O-Z)
      Dist=DSQRT((X2-X1)**2+(Y2-Y1)**2)
```

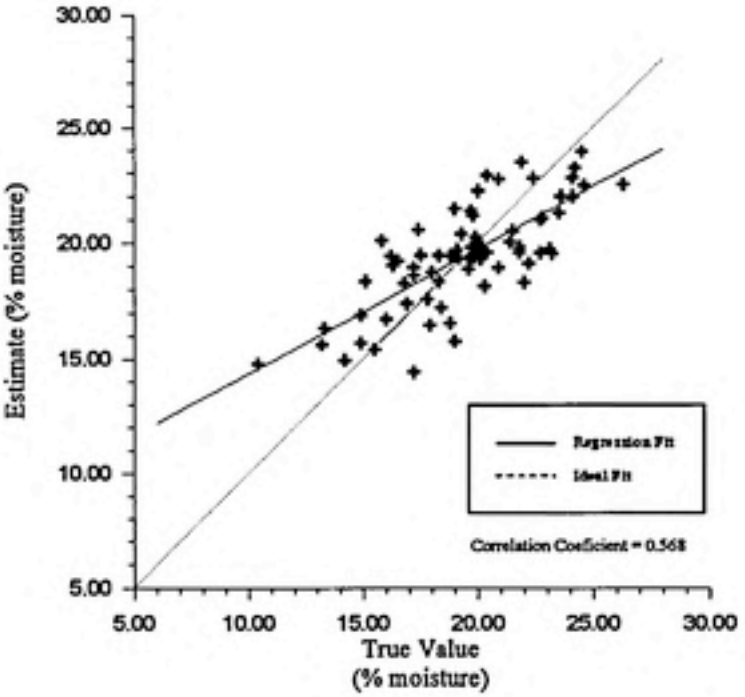
```
      RETURN
      END
C*********************************************************************
      FUNCTION Ialpha (nu)

      IMPLICIT INTEGER*4 (I-N)
      Ialpha=((nu+1)*(nu+2))/2

      RETURN
      END
C*********************************************************************
C*********************************************************************
C*********************************************************************

C  Changes made from parent program, MAIN5.FOR:
```

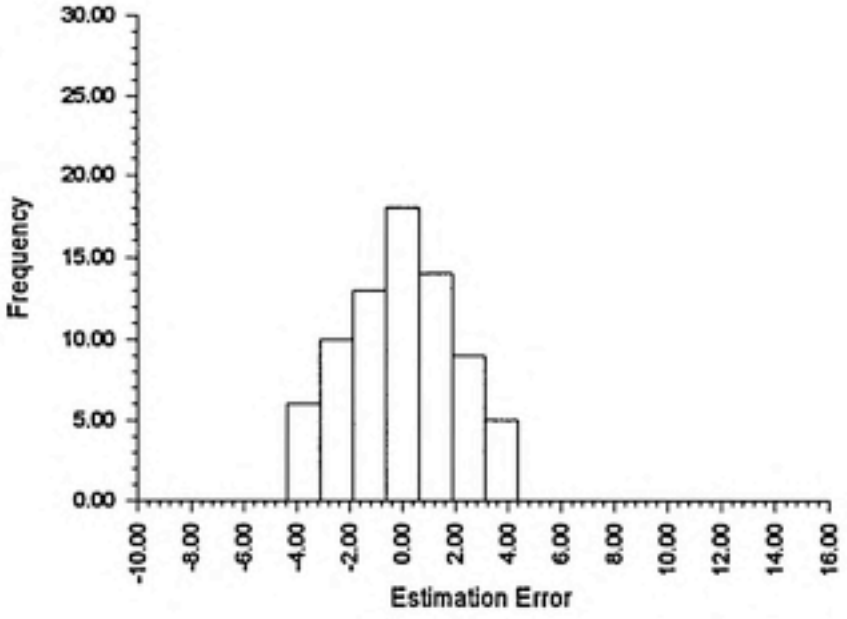
Appendix C:
Regression Plots and Error Histograms for Cross-validations

This appendix contains linear regression plots and estimation error histograms for the 11 cases of cross-validation of the soil moisture content data. The linear regressions describe the correlation between actual moisture content and the intrinsic kriging estimate for that value at each of the 75 data locations. The associated histograms describe the distribution of the estimation error

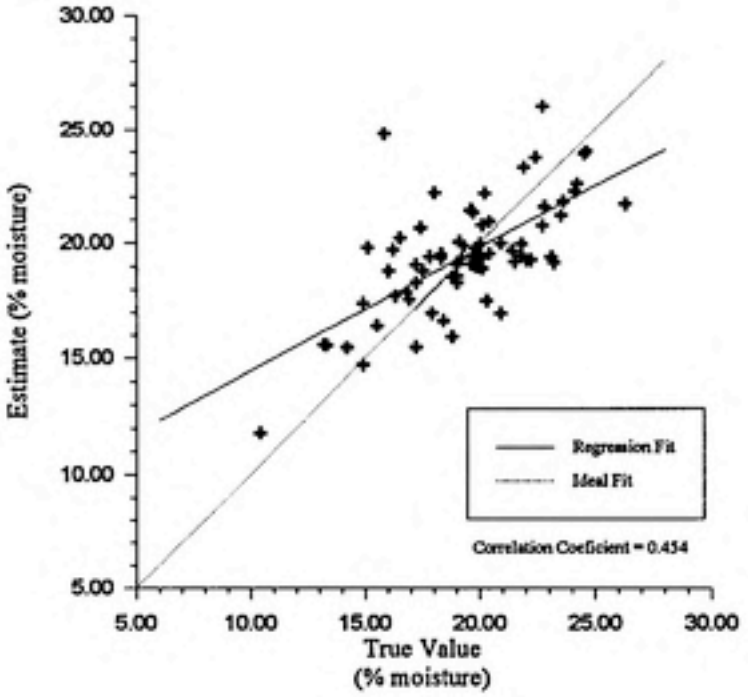
$$\varepsilon(\mathbf{s}_i) = \hat{X}(\mathbf{s}_i) - X(\mathbf{s}_i).$$

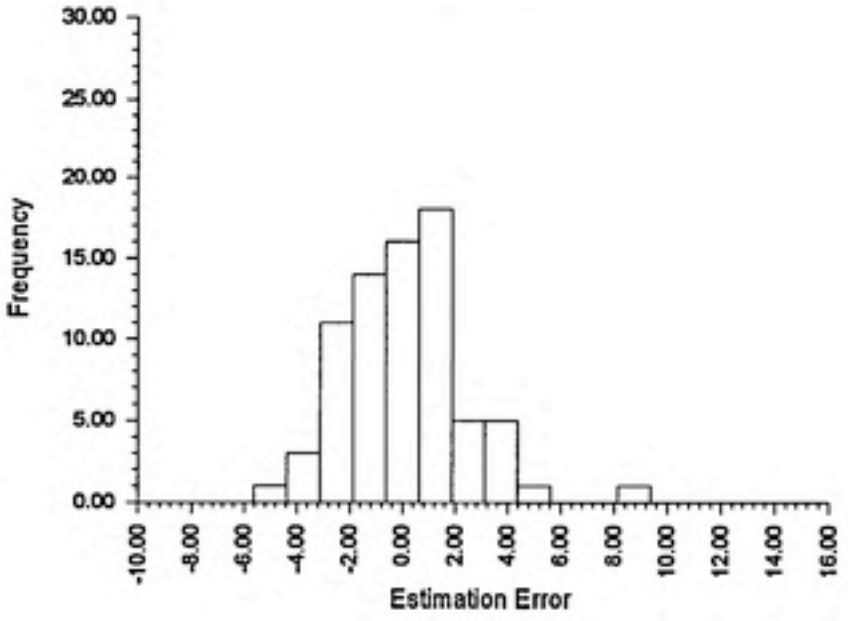
Case No.: C1
Neighborhood Size: 10
GSC Model: Polynomial
Nugget Term: Yes
Regression Coefficient, r^2: 0.5683



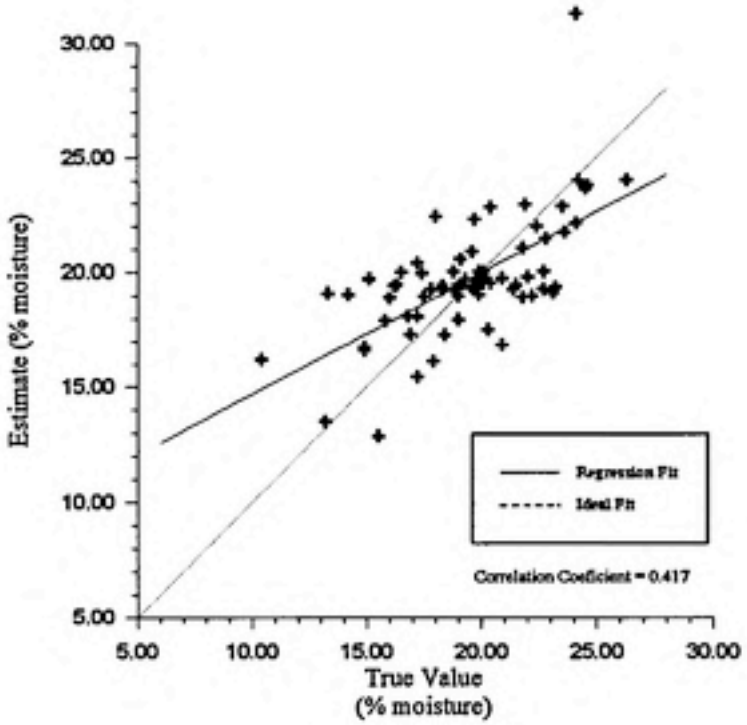


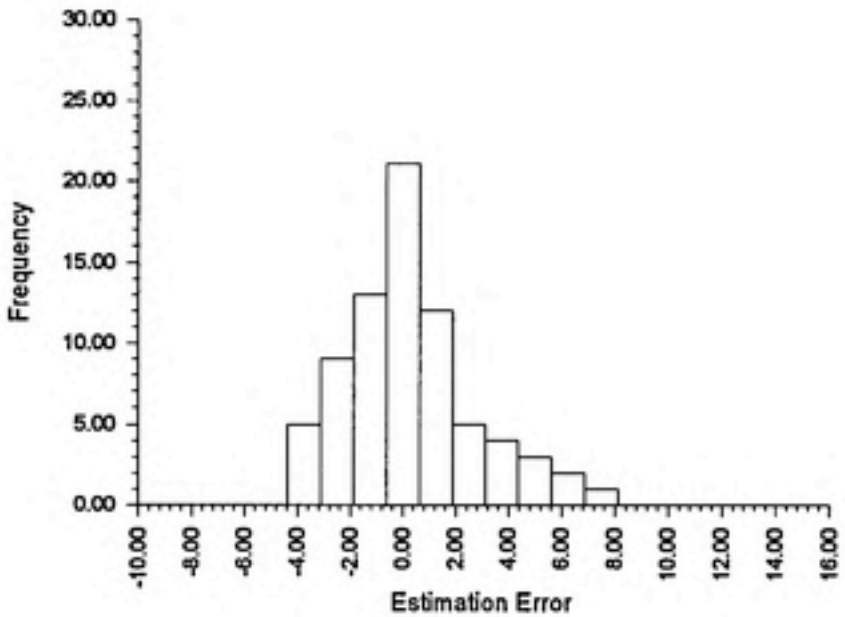
Case No.: C2
Neighborhood Size: 13
GSC Model: Polynomial
Nugget Term: Yes
Regression Coefficient, r^2: 0.4536



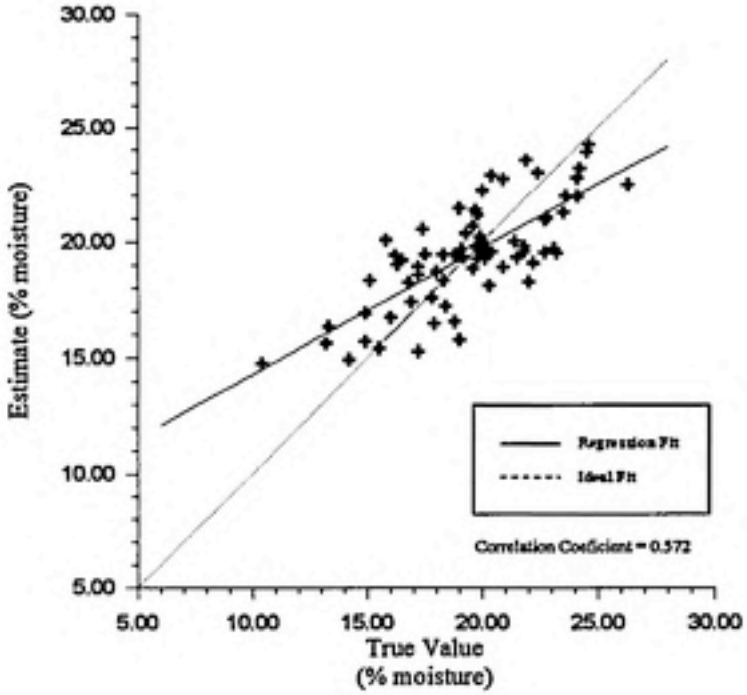


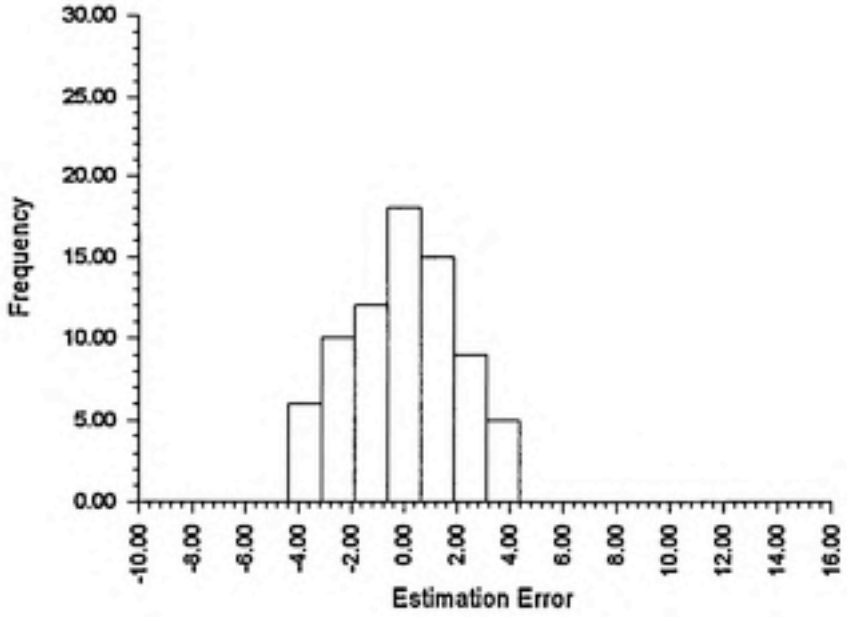
Case No.: C3
Neighborhood Size: 16
GSC Model: Polynomial
Nugget Term: Yes
Regression Coefficient, r^2: 0.4166



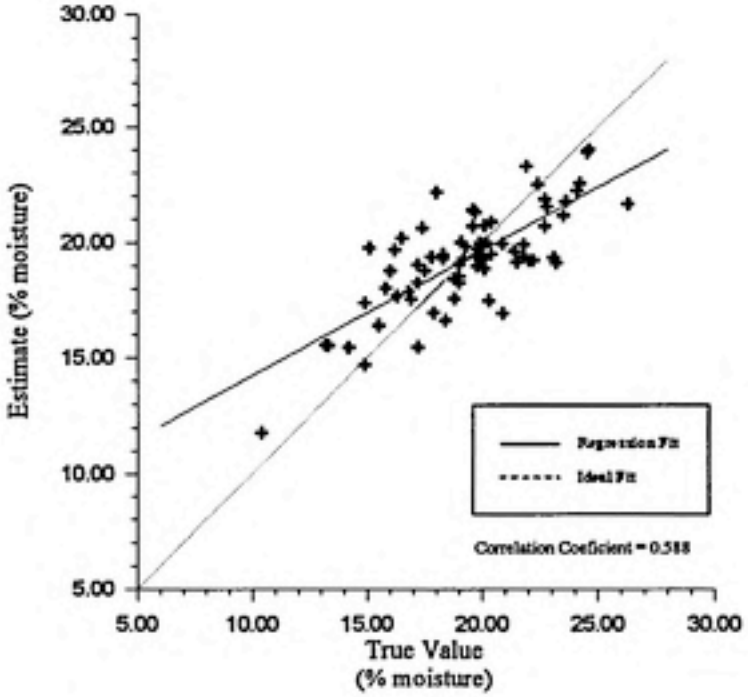


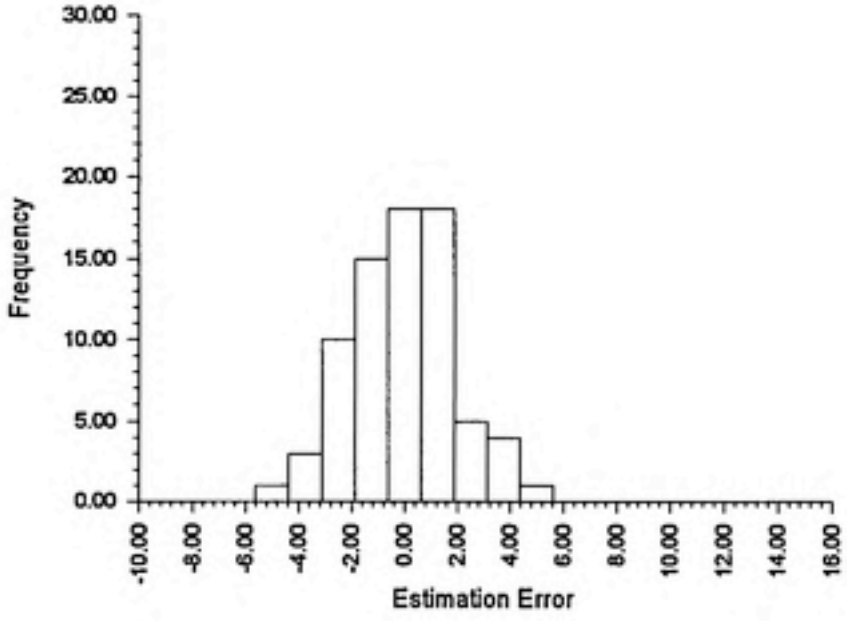
Case No.: C4
Neighborhood Size: 10
GSC Model: Polynomial-spline
Nugget Term: Yes
Regression Coefficient, r^2: 0.5725



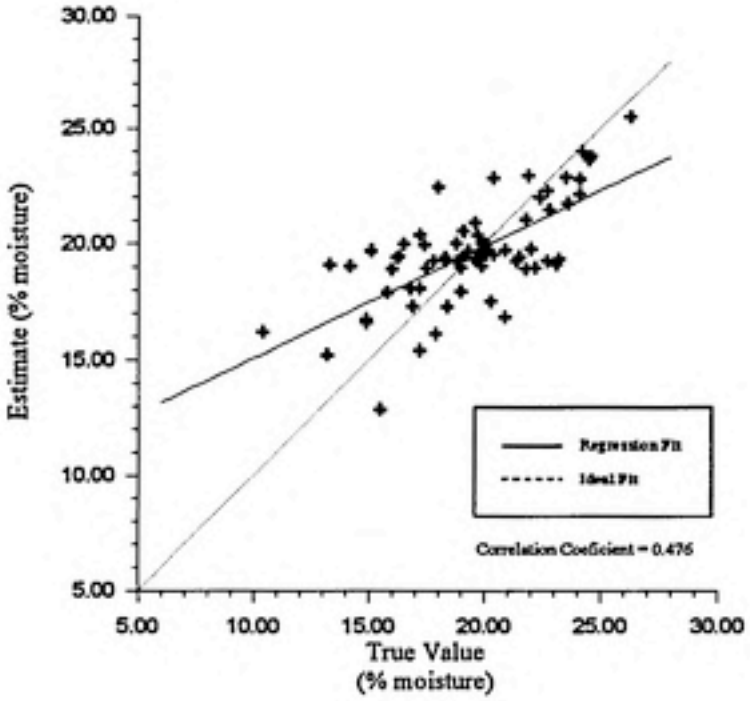


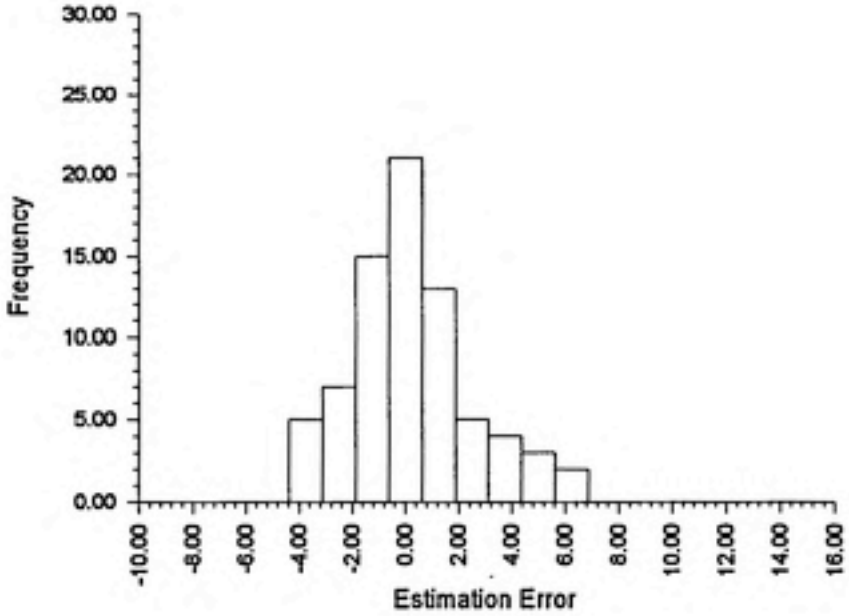
Case No.: C5
Neighborhood Size: 13
GSC Model: Polynomial-spline
Nugget Term: Yes
Regression Coefficient, r^2: 0.5884



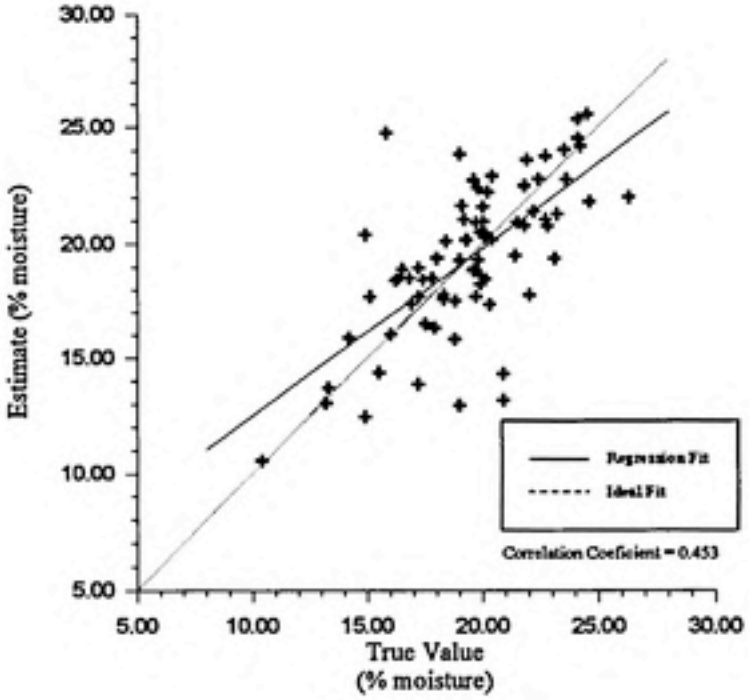


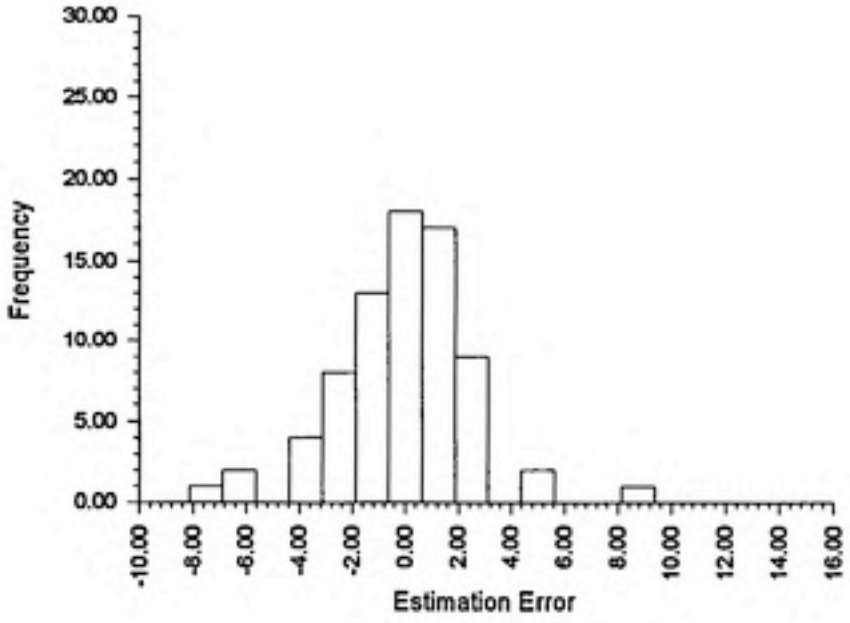
Case No.: C6
Neighborhood Size: 16
GSC Model: Polynomial-spline
Nugget Term: Yes
Regression Coefficient, r^2: 0.4763



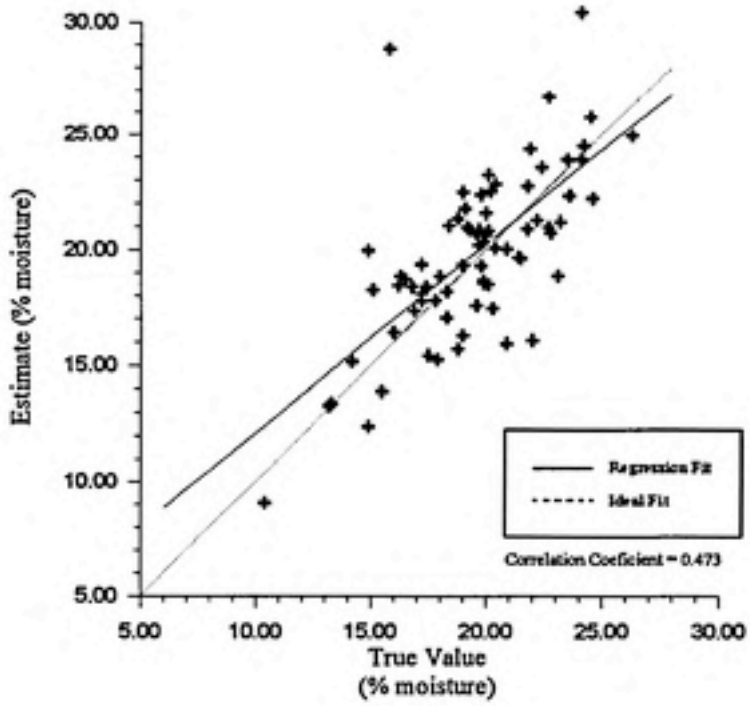


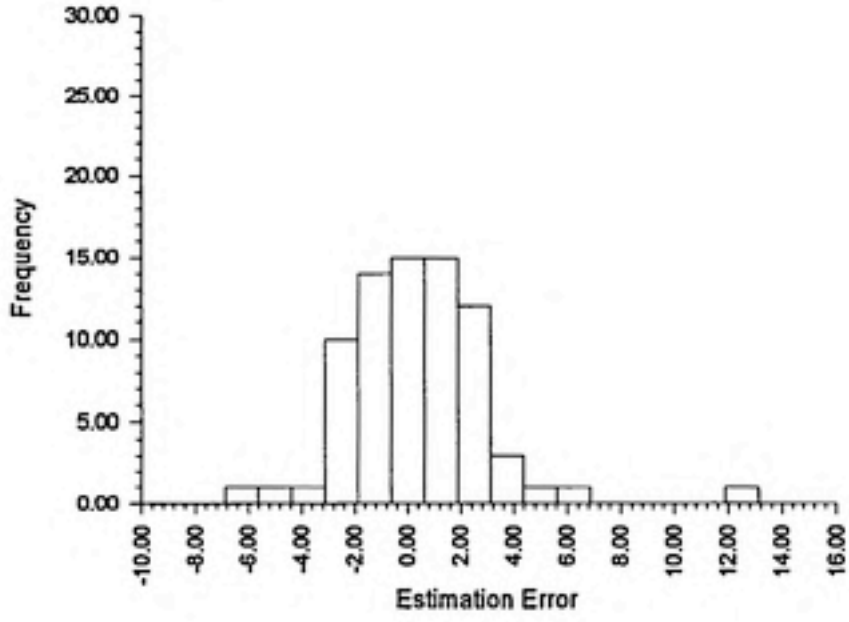
Case No.: C7
Neighborhood Size: 10
GSC Model: Poly-exponential
Nugget Term: (na)
Regression Coefficient, r^2: 0.4525



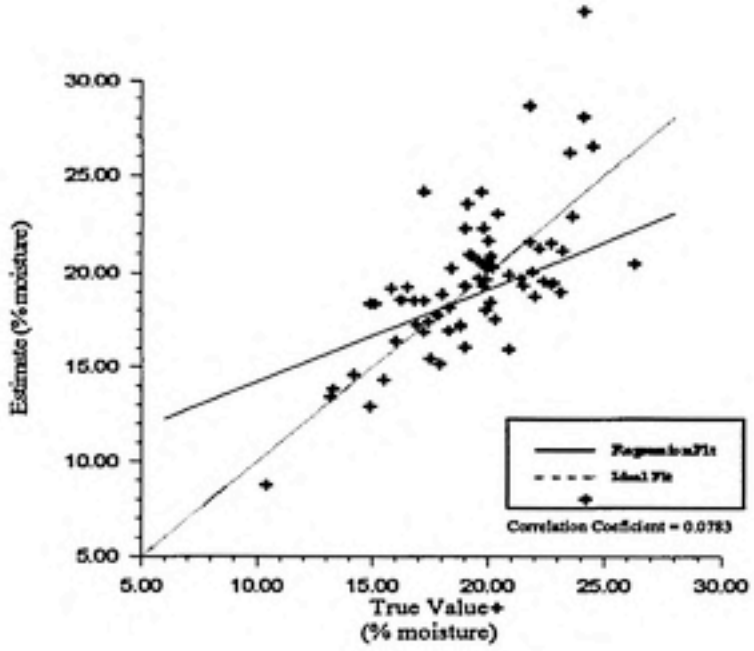


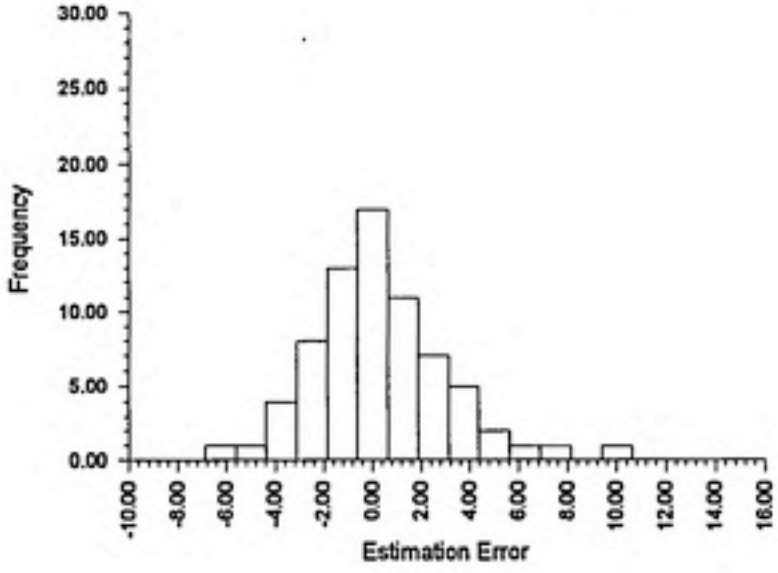
Case No.: C8
Neighborhood Size: 13
GSC Model: Poly-exponential
Nugget Term: (na)
Regression Coefficient, r^2: 0.4729



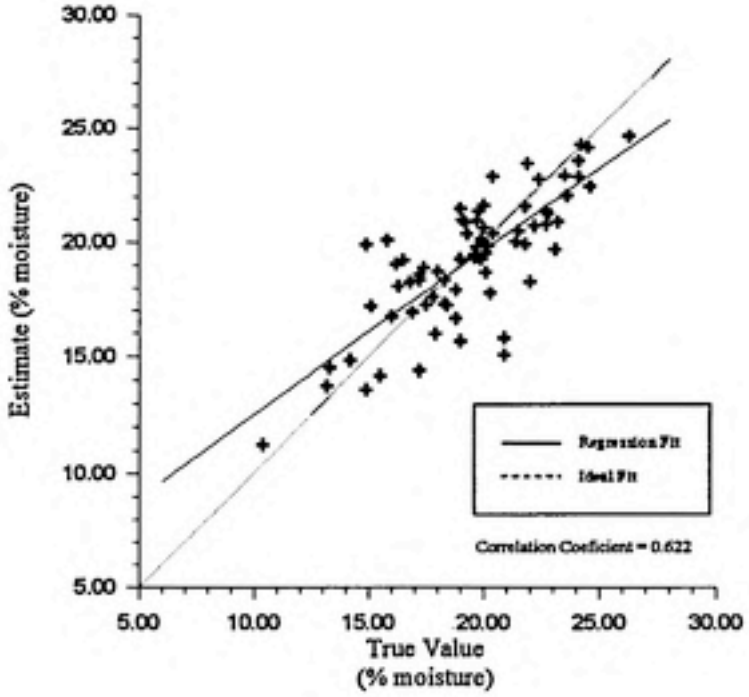


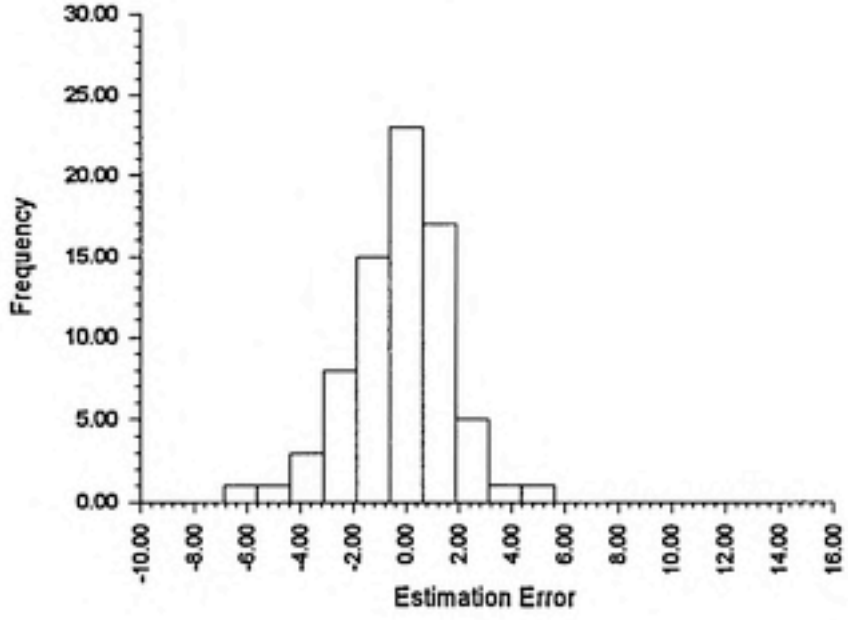
Case No.: C9
Neighborhood Size: 16
GSC Model: Poly-exponential
Nugget Term: (na)
Regression Coefficient, r^2: 0.0783





Case No.: C10
Neighborhood Size: 10
GSC Model: Polynomial
Nugget Term: No
Regression Coefficient, r^2: 0.6222





Case No.: C11
Neighborhood Size: 13
GSC Model: Polynomial-spline
Nugget Term: No
Regression Coefficient, r^2: 0.7100