Spherekit:

The Spatial Interpolation Toolkit

Developed by:

Robert G. Raskin
Jet Propulsion Laboratory, Pasadena, CA

Christopher C. Funk
University of California, Santa Barbara, CA

Scott R. Webber
University of Delaware, Newark, DE

Conceived by:

Cort J. Willmott
University of Delaware, Newark, DE

Technical Report 97-4

November 1997

Preface

Spherekit is a spatial interpolation toolkit that was initiated by Cort Willmott and Mike Goodchild at the NCGIA. Development of a 'user-friendly' and spherically-based interpolation and mapping package had been a goal of Willmott's for some years, which became feasible during his 1995 sabbatical leave at UCSB. Willmott and Goodchild outlined the desired program structure and functionality and allocated the necessary funding as part of NCGIA's Initiative 15: Multiple Roles of GIS in the U.S. Global Change Program. Robert Raskin joined the team and expanded upon the conceptual structure and software design. Raskin contributed to the development, modification, and testing of the algorithms and program. Chris Funk was retained as the primary programmer on the project, and he ulitmately modified, wrote, integrated and/or tested the wide array of subprograms and interfaces that now comprise Spherekit. In the end, Chris contributed much to Spherekit's conceptual base, as well as to the programming. Programs and data also were provided by Scott Robeson (Indiana University), Scott Webber (University of Delaware), and Robert Renka (University of North Texas). Webber additionally contributed to program testing and refinement during the summer of 1996. Spherekit relies on GMT (Wessell et al., 1995) for its map and other graphics, and the authors are grateful to the GMT developers for freely making their programs available to the scientific community. Spherekit remains a work in progress. Its developers are continuing to improve it and they welcome feedback from users.

Table of Contents

1. Overview

Spherekit is a spatial interpolation software toolkit developed at NCGIA as part of Initiative 15: Multiple Roles of GIS in U.S. Global Change Research. This package features several unique capabilities and is freely distributed over the internet at:

http://whizbang.geog.ucsb.edu/spherekit/

Spherekit allows interpolation over continental or global scales by computing distances and orientations (among data and interpolation points) from geodesics on the surface of the globe. Conventional interpolations typically are based upon Euclidean distance in Cartesian 2-space which involve planar projections that produce distortions of some kind. In Spherekit, projections are applied only for display purposes after the interpolation has been carried out using spherical geometry. Users can select from several interpolation algorithms that have been adapted to the sphere: inverse distance weighting, thin plate splines, multiquadrics, triangulation, and kriging. Portions of the GSLIB package have been modified for the sphere and are used in Spherekit to compute variograms for the kriging algorithms.

Spherekit enables the user to incorporate knowledge or information about the processes that produce the underlying spatial variations into the interpolation model. A built-in equation editor and a collection of nonlinear transforms allow the user to create and experiment with new, physically meaningful variables from the independent and dependent variables available. This "smart" interpolation capability allows Spherekit to intelligently interpolate using auxiliary information. One use of the smart interpolation feature is to incorporate elevation information when interpolating variables that are correlated with height. A digital elevation model (DEM) is included with the package for this purpose.

Error analysis is an integrated component of Spherekit making the package particularly useful for comparing interpolation methods and parameters. Interpolation method performance is measured using cross-validation. Cross-validation error is defined at each observation point as the difference between its actual value and its interpolated value estimated from the remaining points. The resulting error field can be displayed either at the data points or by interpolating to a regular grid to reduce spatial biases. Error difference fields, comparing methods or parameter settings, can be created and displayed with ease.

The Spherekit package has been compiled and tested on SGI and DEC Alpha workstations and could be ported to other computers running the UNIX operating system. The software uses Tcl/Tk for its Graphical User Interface (GUI), Generic Mapping Tools (GMT) for display of output fields, Ghostview for display of PostScript files, and netCDF for storing the DEM. All of these required auxiliary packages can be downloaded together with Spherekit.

This report is a reference guide to the software and is not intended to be read from cover to cover. Chapter 2 presents four examples of the use of the software. Chapter 3 takes a single example and presents the steps required to carry out an interpolation. Chapter 4 describes each of the menu options available in Spherekit. Chapter 5 provides a technical description of the interpolation algorithms. Finally, Chapter 6 describes the download and installation procedures.

2. Examples

2.1 "Smart" interpolation

"Smart" interpolation improves the performance of traditional interpolations by using knowledge of the processes that produce the spatial variations (Willmott and Matsuura, 1995). In this example we use the general physical principle that temperature decreases with altitude in the troposphere. This decrease can be characterized roughly by the mean lapse rate. Standard and topographically informed interpolations are compared using a sparse network of 160 weather stations in China. The data set is deficient in that high altitude locations in the Himalayan mountains are underrepresented.

Figure 2.1 shows a standard interpolated temperature field using the multiquadric method. This estimate of the temperature field fails to accurately represent the relatively undersampled area of colder temperatures associated with the greatest elevations.

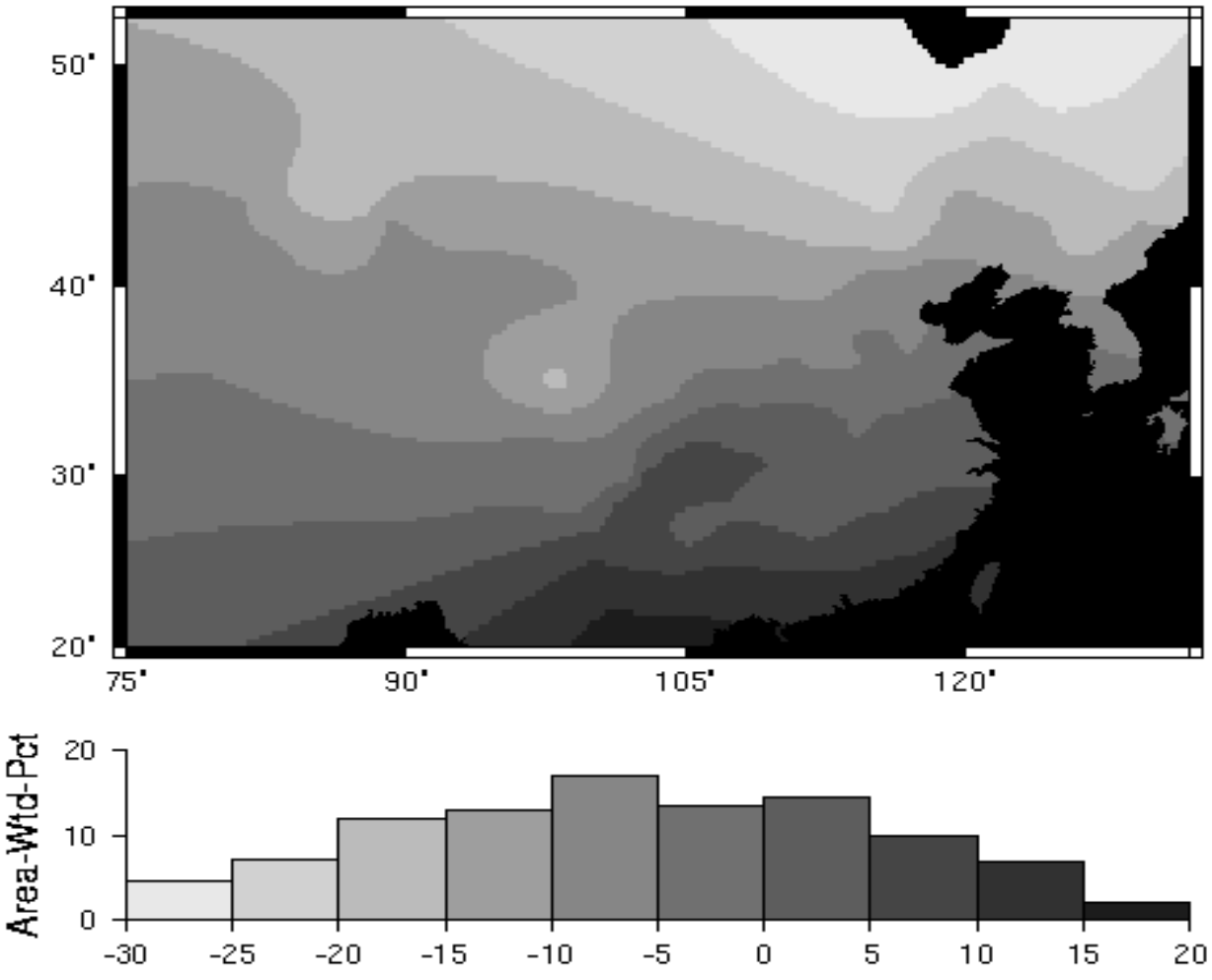


Figure 2.1 Standard Multiquadric Interpolation (Degrees C)

Figure 2.2 shows the corresponding topographically informed interpolation (Willmott and Matsuura, 1995). This interpolation was performed using the following steps:

1. Reduce temperatures to sea-level using the mean environmental lapse rate.
   (Sea Level Temp= Temp + Mean Environ Lapse Rate * Elevation)

2. Interpolate the "sea-level" temperature field to a one-degree grid
   using the multiquadric method.

3. Reintroduce the elevation effect on the interpolated field.
   (Temp= Sea LevelTemp - Mean Environ Lapse Rate * Elevation)

   This final step was carried out automatically by Spherekit by
   inverting the operations in Step 1.

Key differences between Figure 2.1 and Figure 2.2 result from the incorporation of the first-order temperature-elevation relationship (given by the mean environmental lapse rate) into the interpolator. That is, the "smart" interpolation captures the climatological influences of topography. Low temperatures associated with the mountains of western China are now visible, despite the sparsity of high altitude temperature stations.

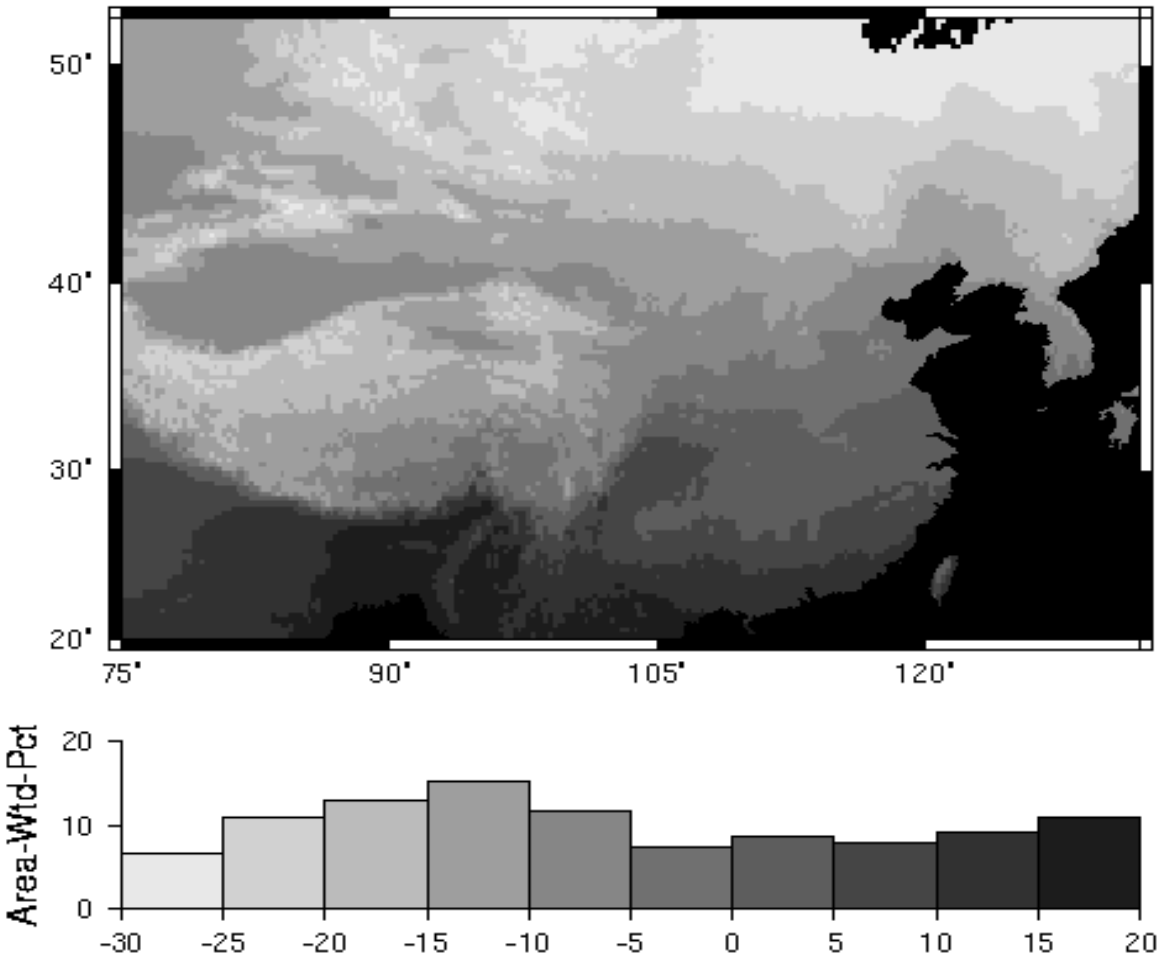


Figure 2.2 Topographically Informed Interpolation (Degrees C)

2.2 Global interpolation

This example focuses on interpolation over the entire globe. A 2456 station subset of the Global Historical Climatology Network (GHCN) (Vose et al., 1992) temperature observations for January 1990 were interpolated to a one-degree global grid. Sparsely populated regions end to be undersampled, resulting in a station network that is not uniformly distributed around the globe. Note that the use of spherical surface geometry ensures realistic patterns of isotherms throughout the polar region (Figure 2.3).

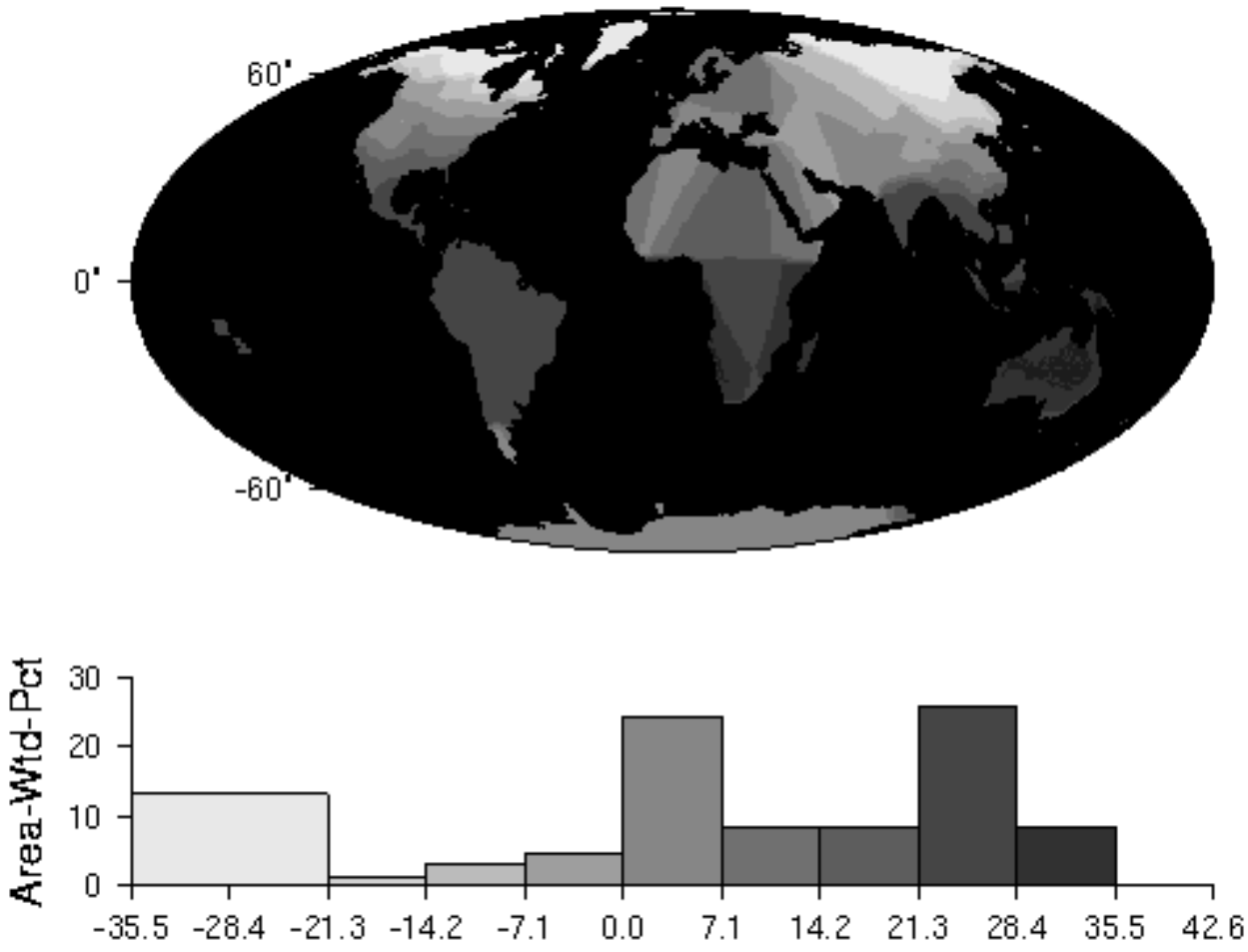


Figure 2.3 Global Interpolation of GHCN Data (Degrees C)

The interpolation method used is inverse distance weighting. Users specify which of several inverse distance functions to use and how the neighborhood of influence is determined. A neighborhood can be defined by a distance (radius) or by a number of points (nearest neighbors). An extrapolation correction based upon local gradients of the observed field is available to prevent local extrema (or peaks or valleys) from occurring only at the data points. A spatial bias correction

can be invoked to adjust the distance weights based upon the angular distribution of the nearest neighbors. The influence of both the extrapolation and spatial bias corrections may be independently scaled.

The orthographic projection of the interpolated temperature field shown in Figure 2.3 represents a reimplimentation of Shepard's (1968) interpolator with an average of seven nearest neighbors, a limited extrapolation capability, and an angular correction.

2.3 Error analysis

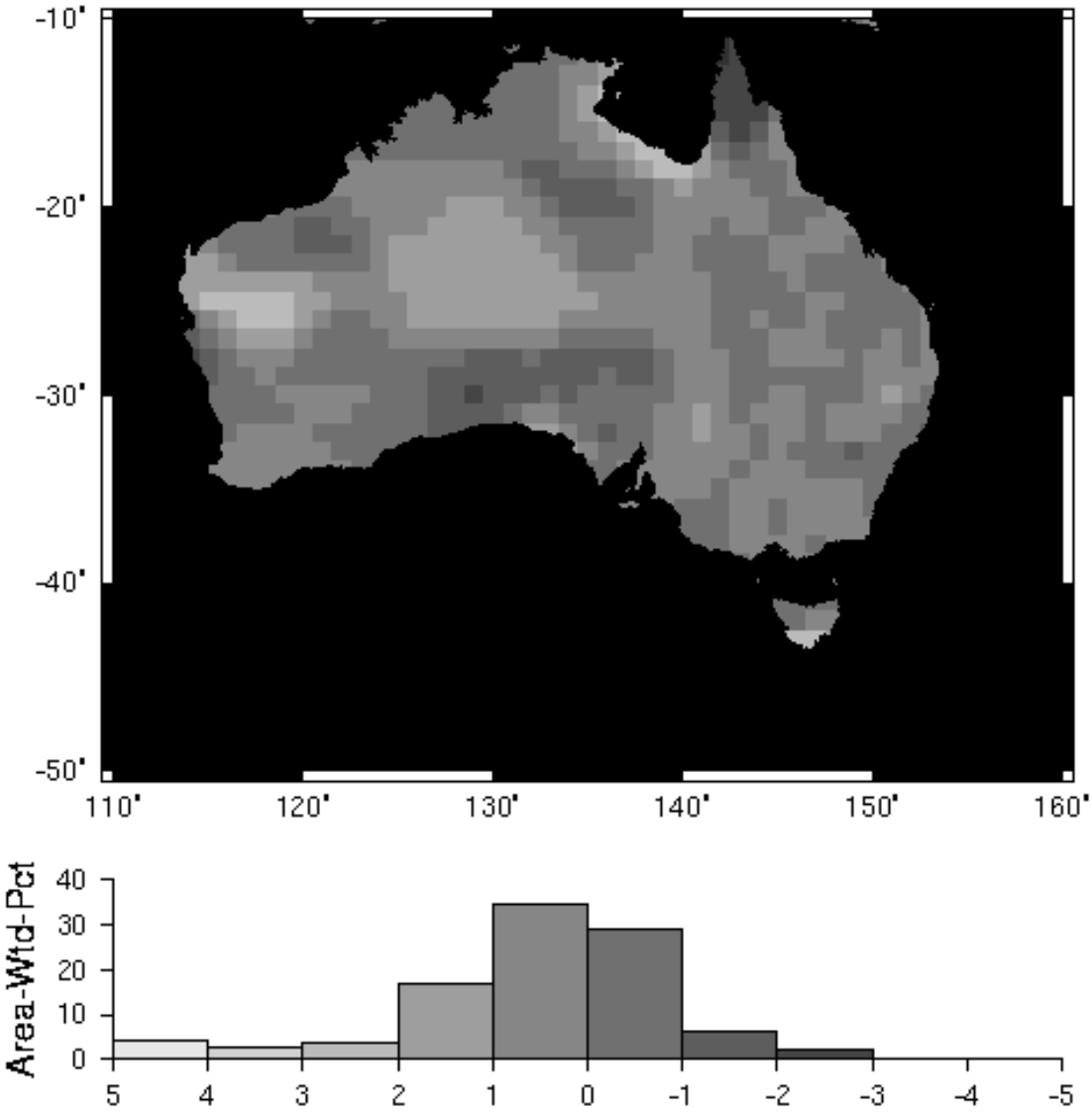
Spherekit provides researchers with the ability to quickly and easily compare interpolation methods and interpolation parameter settings. This example compares two methods: thin-plate spline and Cressman (an inverse distance weighted method). Figures 2.4 and 2.5 present error analyses of these methods applied to the Australia portion of the GHCN dataset used in Example 2. Cross validation is used to generate error estimates at each observation point. These errors are then interpolated to a grid, reducing spatial bias. The plots below are gridded, revealing the one-degree granularity of the interpolation.

Further analyses may be carried out using Spherekit's matrix math capabilities. Figure 2.6 displays the difference field (Cressman-Spline) of the above error plots. It is interesting to note that the standard deviation of the difference field is greater than both mean absolute errors.

2.4 Spatial variability

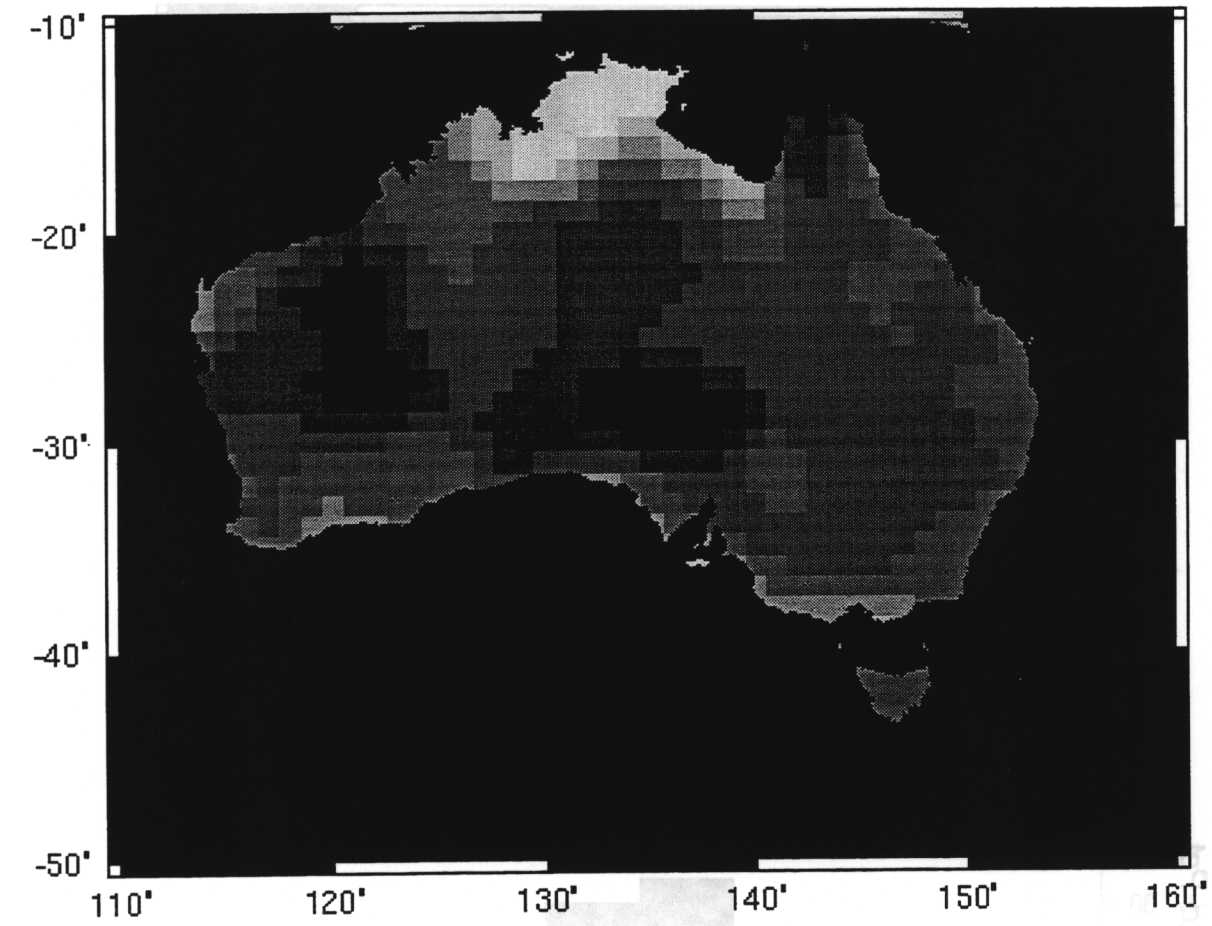
Several exploratory data analysis tools are available to examine the spatial variability of a dataset. Several of the features of the GSLIB software library are integrated into Spherekit. The GHCN temperature dataset of Example 2 is used again to demonstrate the long-distance correlations present in climate data. As Spherekit computes distances using great circle distances, distances at continental and global scales are computed correctly.

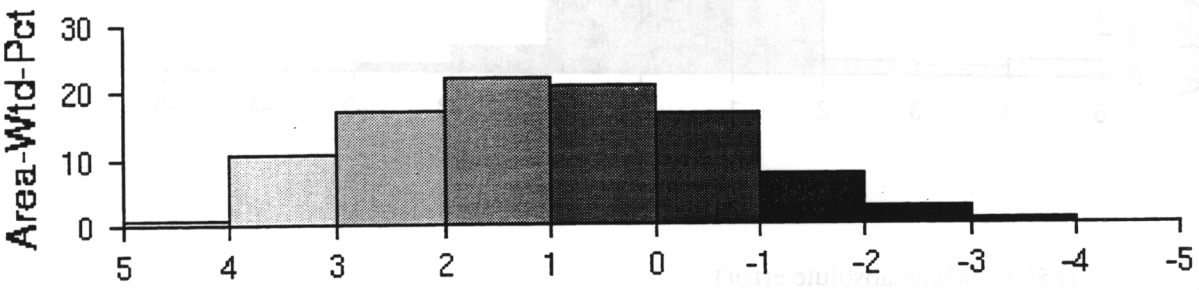
Figure 2.7 shows an isotropic semivariogram of the dataset. There is a plateau in the semivariogram in the 2000-4000 km range and a sharp rise thereafter. This calculation is repeated using anisotropic semivariograms in the east-west and north-south directions. Figure 2.8 (the east-west semivariogram) displays the plateau more prominently. This characteristic corresponds to the common notion that zonal variations in temperature are relatively small. The north-south variogram (Figure 2.9) shows a more rapid increase in variance with distance, as would be expected. Interestingly, the semivariogram falls after reaching a peak; presumably this is due to a return to the same latitude zone at these distances.



MAE	0.593	(Mean absolute error)
MBE	-0.13	(Mean bias error)
RMSE	0.873	(Root mean square error)
MIN	-4.568	
MAX	3.173	
COUNT	1,248	

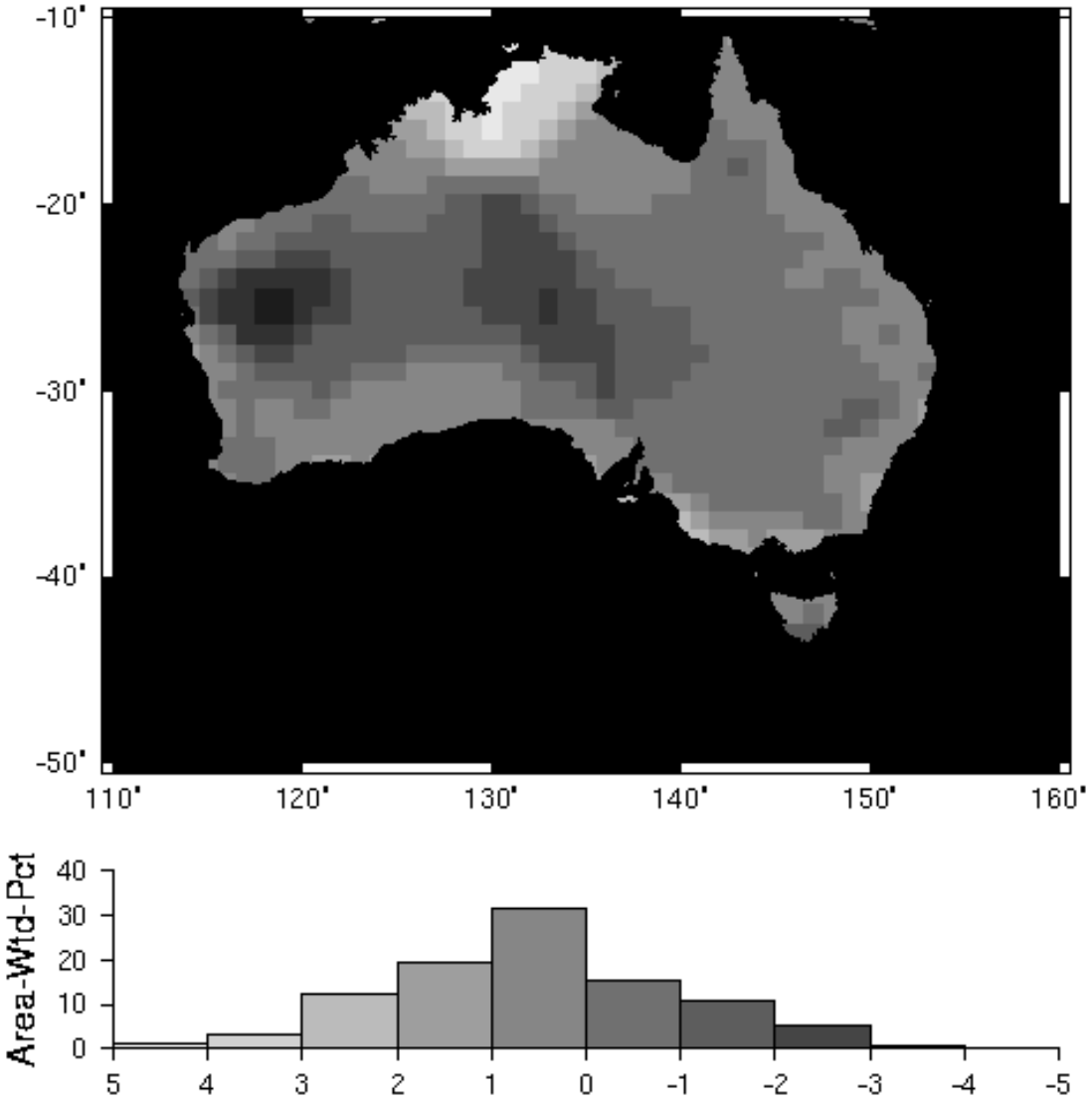
Figure 2.4 Thin-Plate Spline Cross-validation (Degrees C)





MAE	1.068
MBE	-0.335
RMSE	1.379
MIN	-4.407
MAX	3.437
COUNT	1,248

Figure 2.5 Cressman Cross-validation (Degrees C)



MIN	-4.999
MAX	4.176
AVG	-0.223
STD	1.410
COUNT	1,248

Figure 2.6 Difference field (Spline error - Cressman error) (Degrees C)

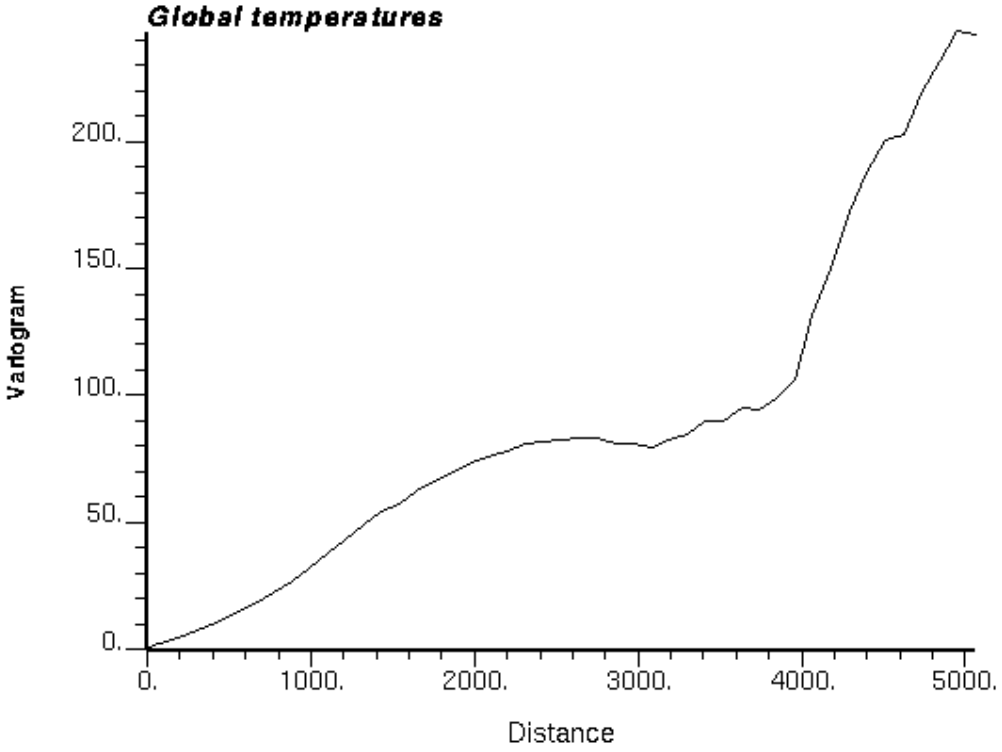


Figure 2.7 Isotropic semivariogram (Degrees C)^2 as a function of distance (km)

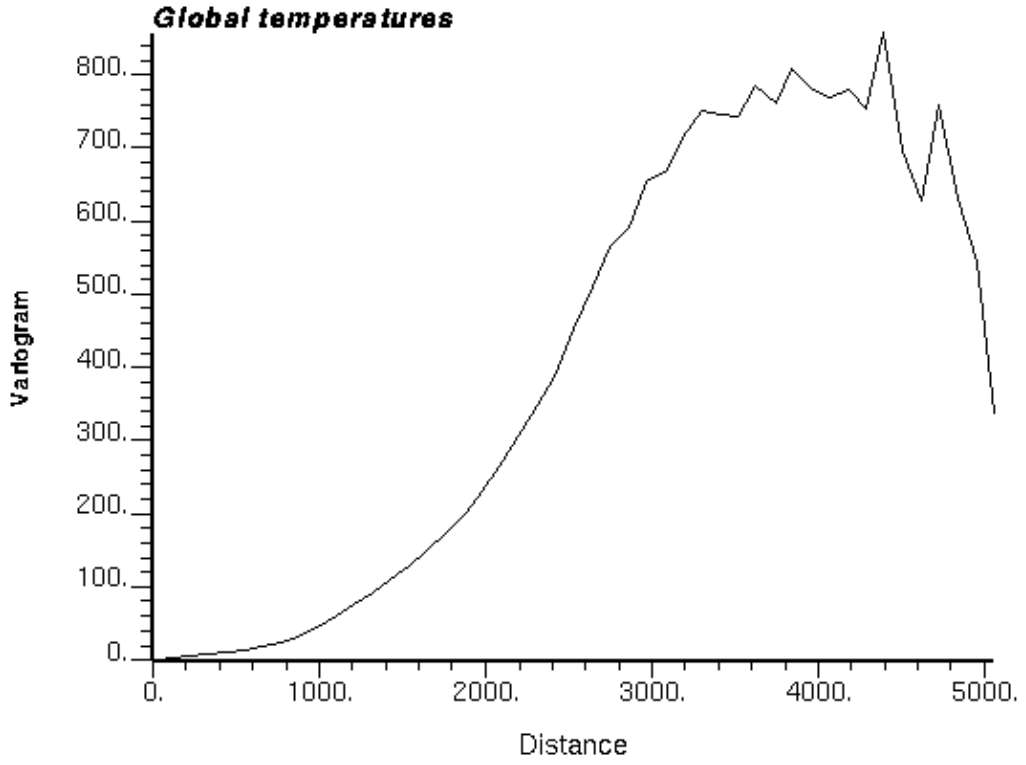


Figure 2.8 North-South semivariogram (Degrees C)^2 as a function of distance (km)

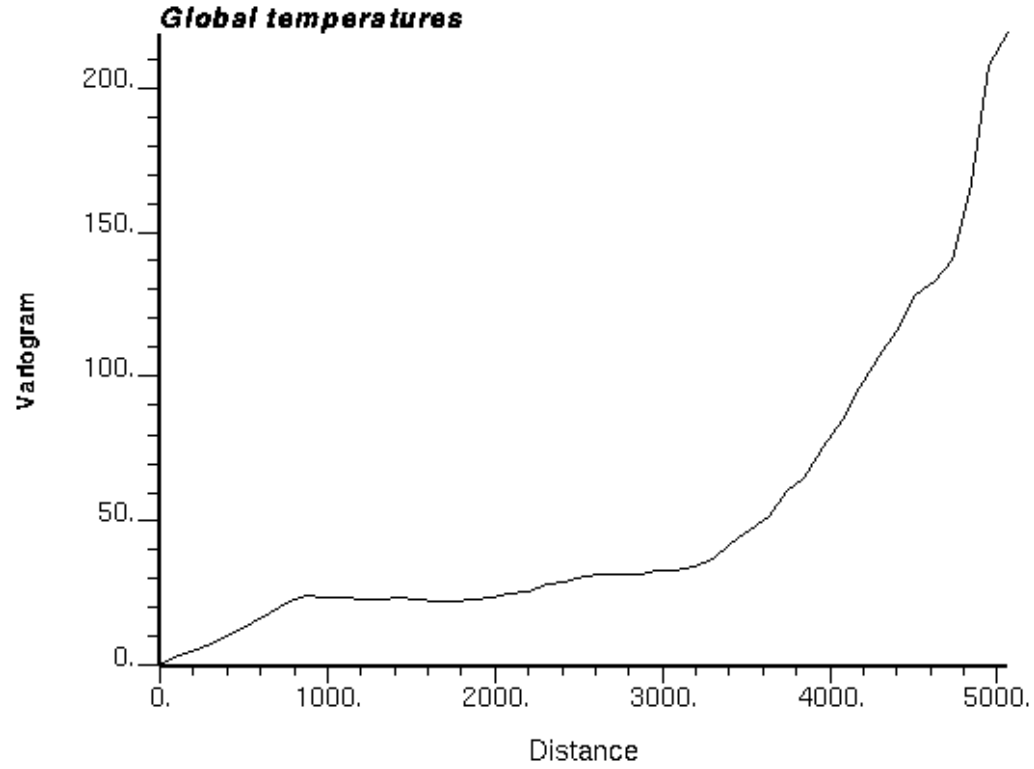


Figure 2.9 North-South semivariogram (Degrees C)^2 as a function of distance (km)

3. Tutorial

For this tutorial, you will use a dataset consisting of precipitation, elevation, and temperature values for 160 locations in China. Spherekit can produce a complete estimated field for these variables using various interpolation techniques. The package also can produce a field of error estimates at the estimated points. We will examine this process as well as the following interpolation methods: Inverse Distance Weighting, Kriging, Splines, and Multiquadric interpolation.

Getting The Test Data

Begin by downloading the file china.dat from www.ncgia.ucsb.edu/pubs/spherekit/main.html. Move this file to your Spherekit work directory. Examine the first few lines of this file by typing: head china.dat**.** The first few lines should look like this:

```
51.716667 126.650000      244.000000    3.0   -27.7
48.766667 121.916667      823.000000    4.0   -23.1
49.216667 119.750000      610.000000    2.0   -29.7
50.500000 121.466667     1067.000000    4.0   -30.0
49.166667 125.233333      305.000000    2.0   -28.6
47.383333 123.916667      152.000000    1.0   -23.3
47.433333 126.966667      244.000000    4.0   -22.9
47.233333 131.983333       91.000000    5.0   -20.3
46.816667 130.283333      122.000000    0.0   -22.7
45.283333 130.950000      152.000000    0.0   -20.2
```

Running Spherekit

From a command prompt, type 'sk'. This should bring up the Spherekit application, depicted below. If not, consult the README file for guidance. You should have a window that looks Figure 3.1. Spherekit allows you to manipulate eight different kinds of objects. Click on the little graphic icons that line up on the left of the Spherekit window. Read the messages for each of the eight objects. This should familiarize you with the Spherekit data types.

Creating a SK Project Directory

Spherekit allows you to organize your work into projects, each project corresponds to a sub-directory off of your Spherekit work directory. Create a new work directory by selecting:

`File->Environment->Create New Project`

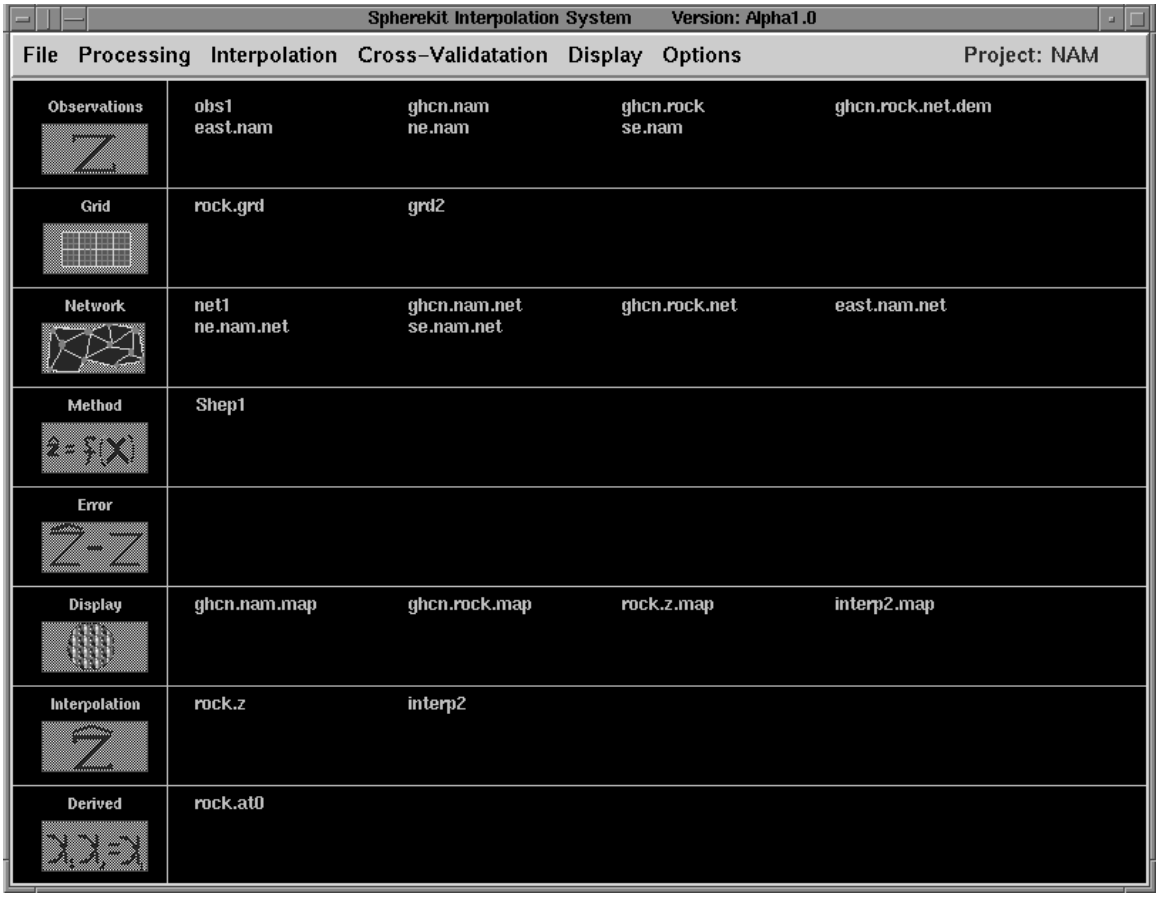


Figure 3.1 Main Screen

That is, select **File** from the Main Menu, **Environment** from the **File** submenu, **and Create New Project** from the **Environment** submenu. When prompted for a name, enter 'tutorial' and hit return. **tutorial** will now become the current project, and all data objects will be stored in this directory. If you would like to return to this workspace in a later session, choose:

```
File->Environment->Set Project
```

Next we will create a Location object. Spherekit stores locations independently of sets of values. This allows multiple sets of values to refer to the same set of locations. To load (import) a set of locations that are irregularly spaced, choose:

File->Import Data->Locations->Network

When the dialog box appears:

I. Enter China.net in the first field.

II.
   A. Click on the 2nd field, and use the file dialog to find the 'china.dat' file,
   B. Click OK.

III. Click Okay again.

This should generate a Network object called China.net that should appear in the Network section. Click on the object's name 'China.net', and examine its metadata. Every new object should be examined in this manner.

Now we're ready to read some data values. Go to:

File->Import Data->Values

Set the name field to 'Temperature'. Click on the **??????** in the locations field. **??????** is Spherekit's generic symbol for 'need this value'. Select china.net from the drop down menu. If the 'china.dat' file is not specified, select that file. Then click OK.

This should generate an Observations object called temperature. Click on **Temperature** and examine the metavalues.

Two “objects" should now be in the interface window. One is named "Temperature" which is a temperature dataset for 160 weather stations in China. The second is called "China.net" which is the latitude and longitude of the nodes of the weather station network.

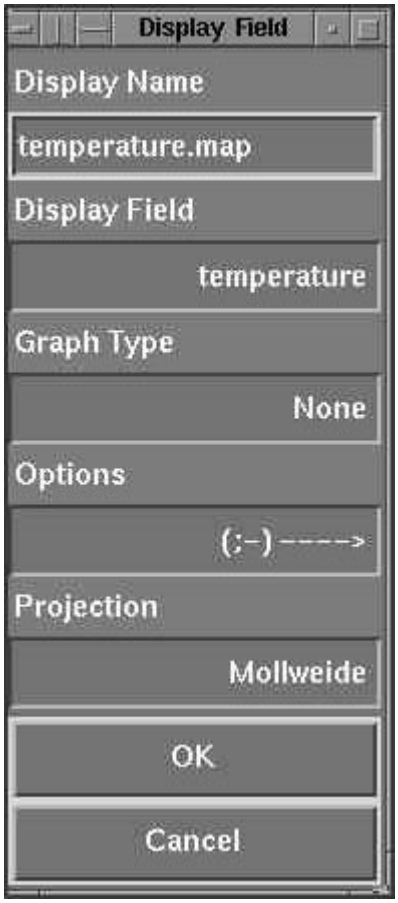
Generating a Temperature Plot

Now we will create a dot map of the temperature field. Click:

Display->Field

This opens a dialog box that looks like Figure 3.2.

Figure 3.2 Display Field Dialog Box



Under the Display Field heading click on the **??????** and select Temperature. Under the Projection heading Pick an appropriate projection for China. You may have to experiment with this parameter until you get a projection you think is good. Click OK when you have selected the display field and the projection. You should get a map of the Temperature data points for China that will look similar to the Figure 3.3.

This is the data from which you will be generating interpolated fields of Temperature. At this point you must create an **interpolation method** and a **grid**. Creating an interpolation method involves selecting one of several predefined methods, such as inverse distance, and specifying some parameters, such as the spatial extent of the data included in the estimation. For example, if we are interpolating temperature in the continental United States how many points do we want to use to interpolate a temperature value in Tucumcari New Mexico? Should we use data from Seattle? This problem is called the "Neighbor Selection problem", and the user provides parameters to identify the neighborhood size. Another parameter to be set is the exponent of the inverse distance weighting. The default exponent is 2.0 (in analogy to the effect of gravity), but you may experiment with the value of this parameter.

.

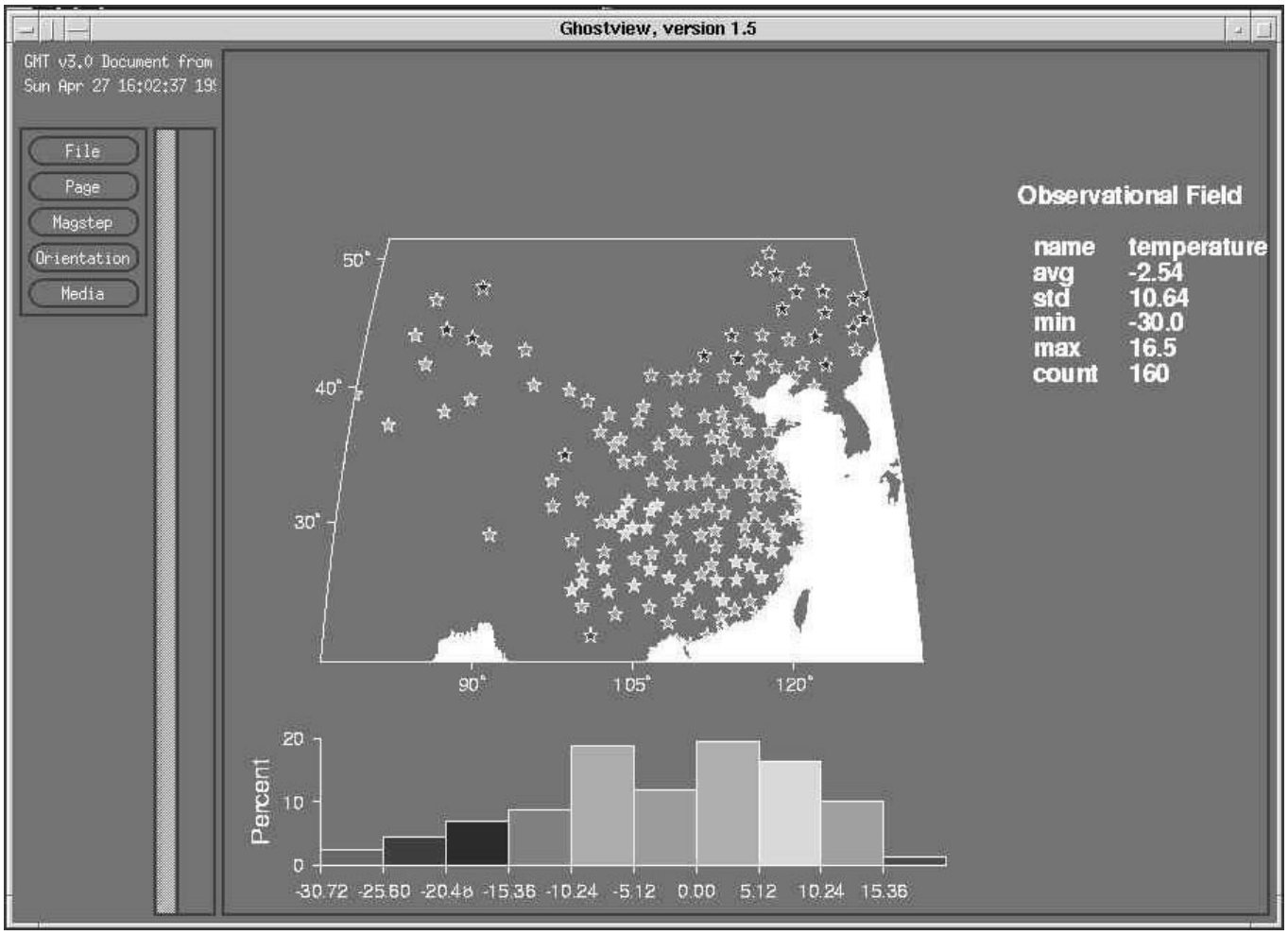


Figure 3.3 Display of original (pre-interpolated) data

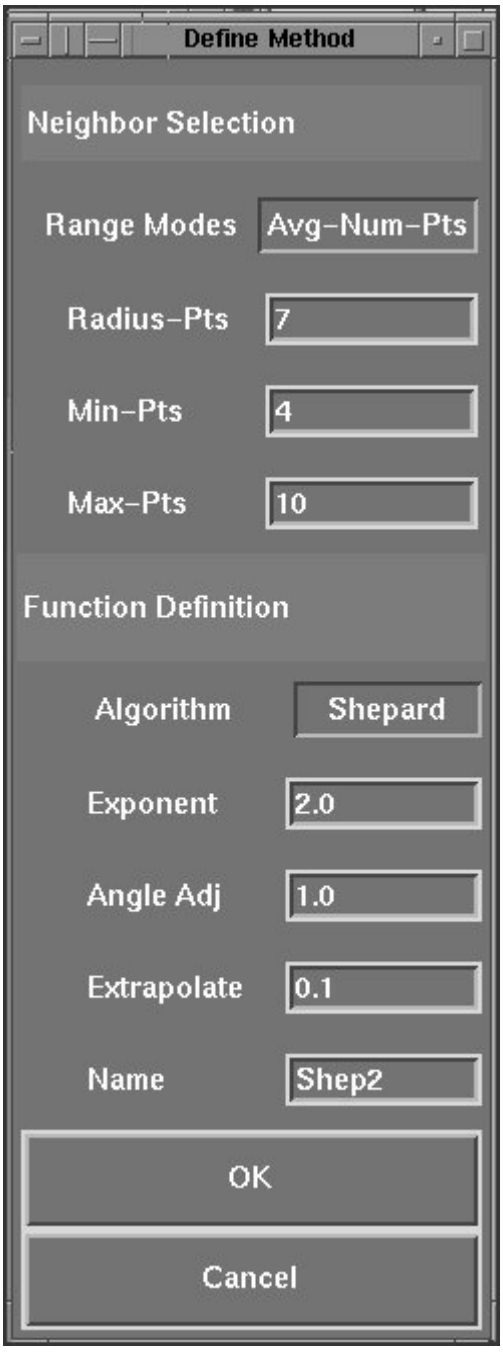
This figure shows a set of 160 weather stations from the Geophysical Historical Climate Network, for the month of December.

Creating an Interpolation Method

To create your interpolation method, select:

`Interpolation->Create Method->Inverse Distance Weighting`

This should produce a dialog box that looks like Figure 3.4.



The "Range Modes" identifies how the neighborhood size is specified. It is described in detail in Section 5.1. For now, use the default settings, which will select a neighborhood size of between 4 and 10 points, depending on the density of data in the vicinity of the interpolation point.

The "Function Definition" specifies the nature of the mathematical inverse distance function (see Section 5.1 for details). These defaults are fine. Click OK to accept them. This creates a new method called Shep1 that should appear in the Spherekit window.

Figure 3.4 Define Method Dialog Box

Creating an Interpolation Grid

In Spherekit a grid is simply a set of locations arranged at a regular distance (in degrees) from one another. To create a grid, select:

`Interpolate->Create Grid`

You should see the Grid Definition Dialog Box as shown in Figure 3.4.

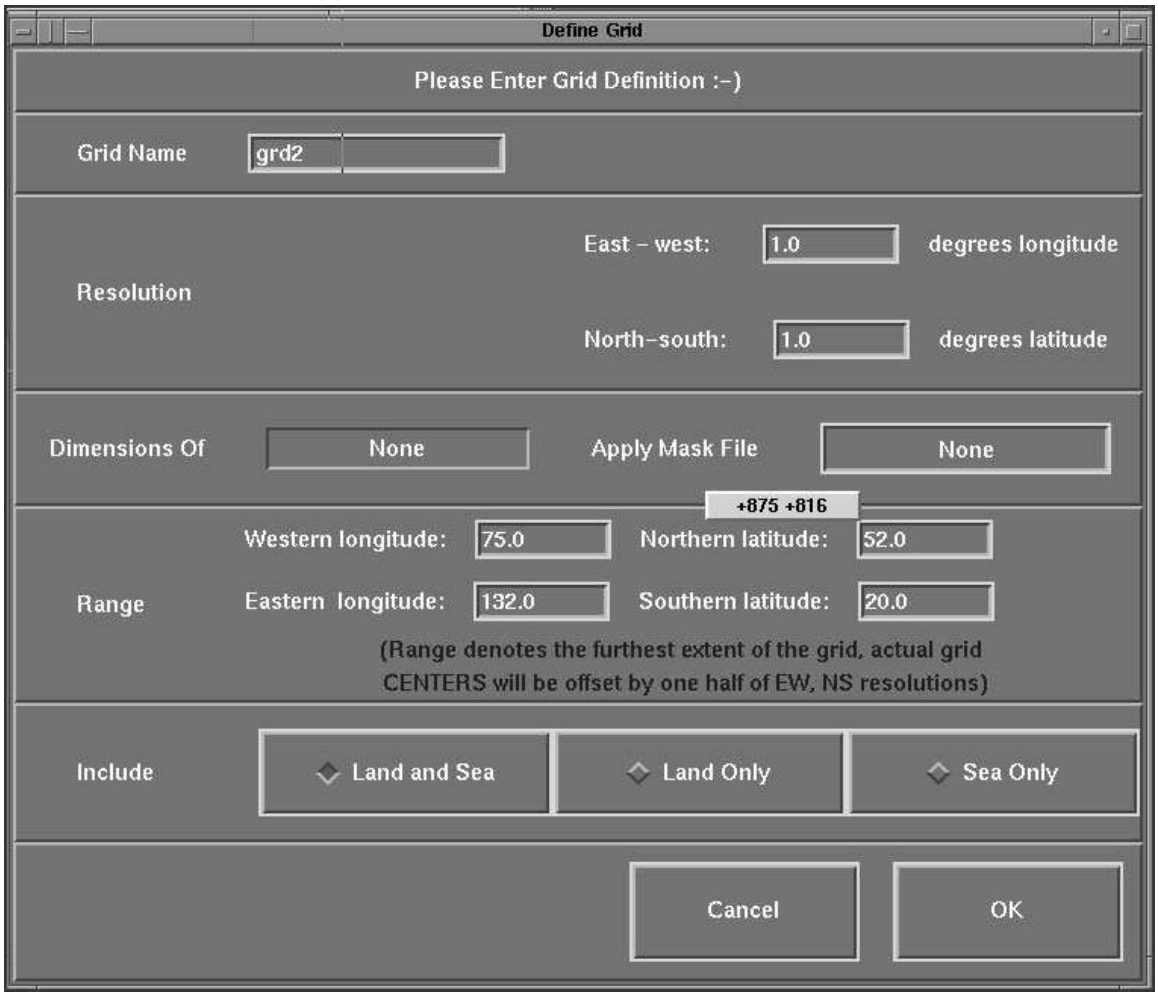


Figure 3.5 Define Method Dialog Box

You could play with these parameters, i.e. the resolution of the grid, the extent of area that you want to interpolate to, etc. For this example just accept the defaults by clicking OK. This will create an object called Grd1 that will show up in the Spherekit window.

Performing an Interpolation

Now we are ready to perform an interpolation. This will estimate a value at each of the cells in the grid you have just defined. To do this choose the following:

`Interpolation->Begin Interpolation`

This produces, you guessed it: yet another dialog box that looks like Figure 3.6.

Figure 3.6 Interpolation Dialog Box

Fill in this dialog as follows:

I. Choose a name that makes sense for the output field name. e.g. TempInvDist
II. Click on all the fields with **??????** in them and choose the appropriate objects.
III. Then click "OK". This will produce a new object with the output field name you supplied.

After interpolation occurs, the display dialog box will appear. Select a **Display Field** that is the interpolated field object that you just created (It should be the default), Select a **Graph Type** that appeals to you (fool around here, grid plots and Isolines are different). Grid plots simply provide the value of the grid cell, while a contour plot smooths the image based on a Delaunay triangulation procedure.

Now choose the **Options** that appeal to you. We suggest “Superimpose Data Values”, and “Base Scale on Source. Then click OK. An image will appear. This can be printed if you like, by selecting print from the ghostview file menu.

Generating Error Fields

Spherekit contains a rather limited set of online help topics. Begin this section by selecting:

`Display->Help`

This will bring up a list of the available help topics. Select **Cross-Validation-At-Net** and read the help message. Repeat this for **Cross-Validation-At-Grid**.

To make an image of the estimated error field select the following:

`Cross-Validation at Grid`

Complete these steps to fill in the field:

I. Enter a meaningful (but conscise!) name for this error field.
II. In the **interpolation** menu heading select one of your interpolation methods.
III. In the **interpolate from** menu select Temperature.
IV. In the **error interpolation method** select Shep1.
V. In the **interpolate errors to** field select the same grid you created earlier (probably Grd1).
VI. Click OK.

When the display dialog window shows up just click OK. This should bring up a ghostview representation of the estimated error.

4. Users' guide

This section describes screen displays and menu options that you will encounter when using the package. When starting Spherekit, the display is divided into eight types of variables: Observation, Grid, Network, Method, Error, Display, Interpolation, and Derived variables. These areas will be filled in with file names as fields of these types are read in or created. Clicking on any field name displays all known metadata for that field.

Throughout a Spherekit session, windows will pop out as options are selected. The method of closing windows will vary depending on the version of X-Windows that you are running. Generally, an icon in the upper left corner can be clicked to close or exit from the window.

The main screen of Spherekit is shown in Figure 4.1. Six main menu options appear along the upper periphery of the screen. Each of these options contains suboptions that are described in this section. The default Project Name: "Work" also appears on the upper right corner of the screen.

.

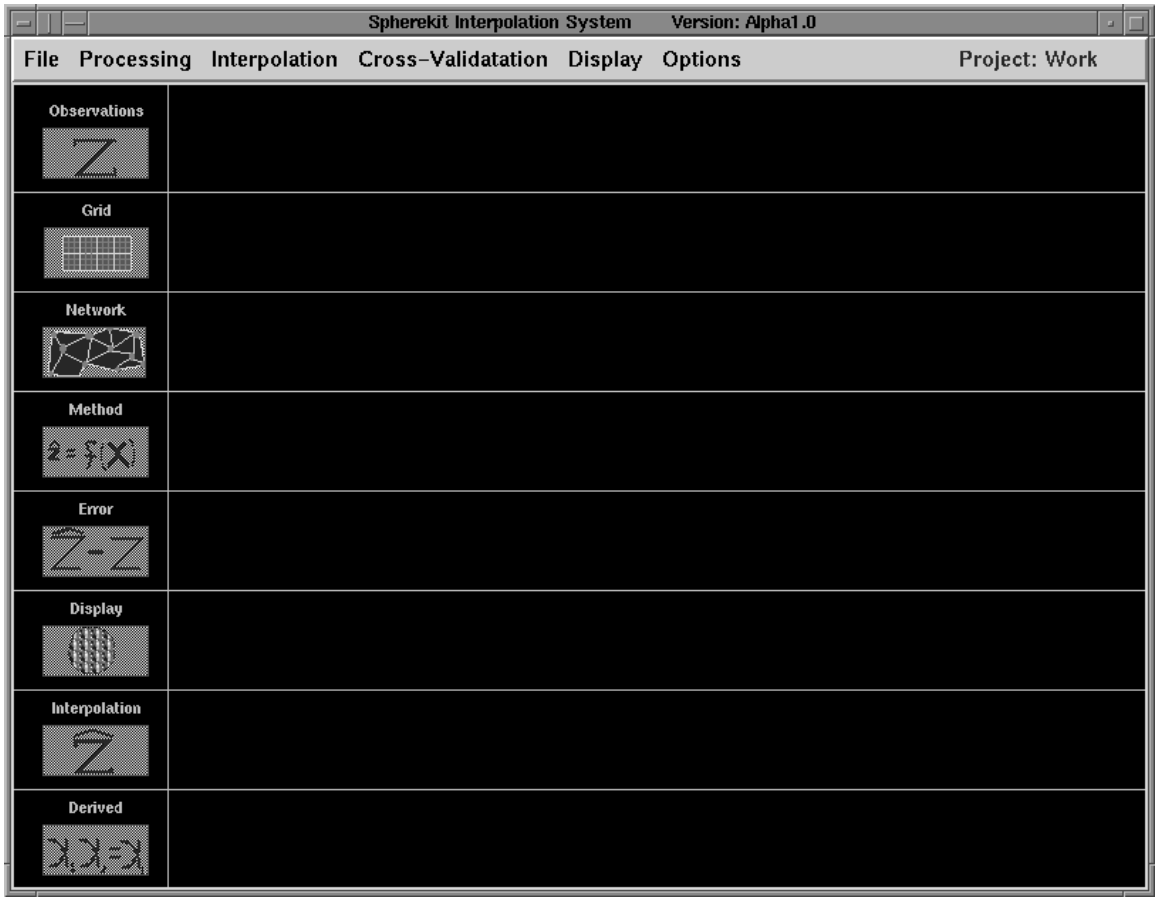
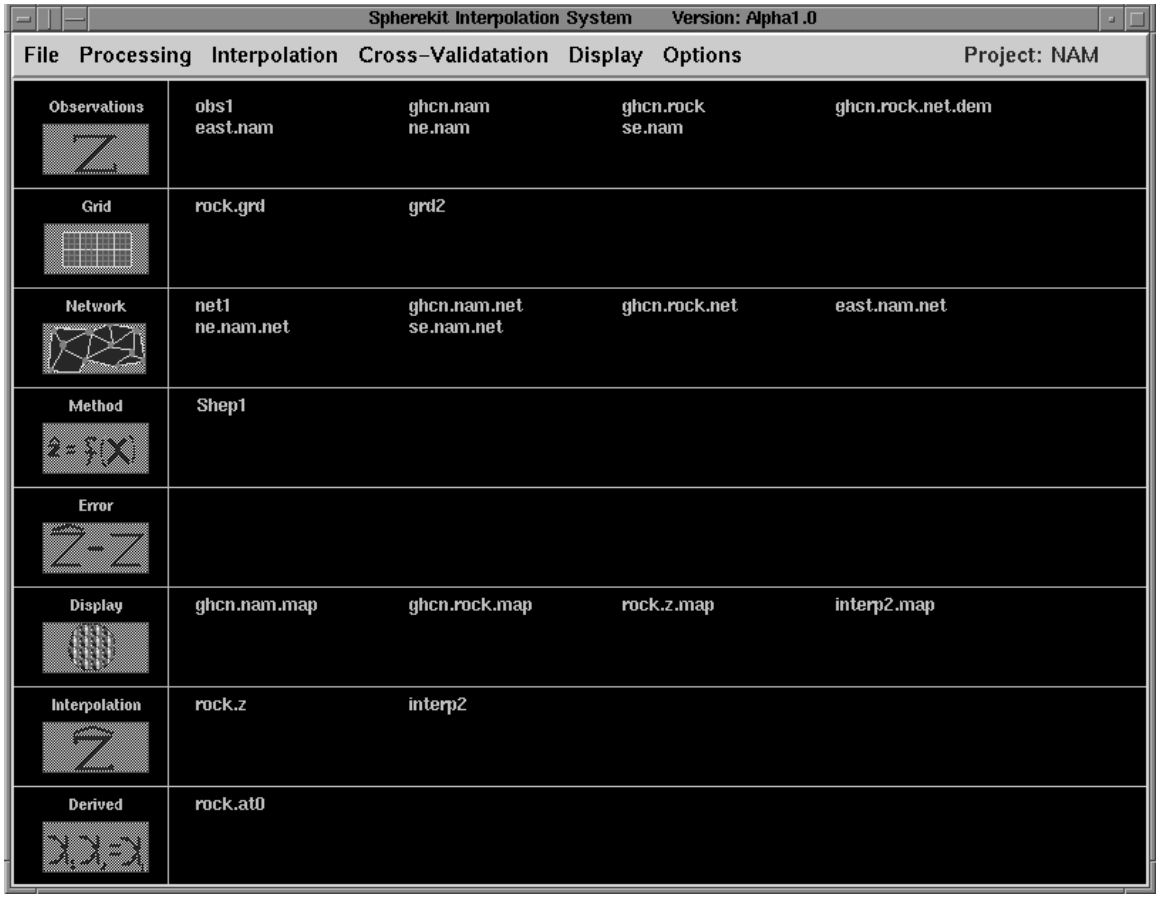


Figure 4.1 Main screen

As files are entered or created by the user, the filenames appear as shown in Figure 4.2. Clicking on any of the file names displays metadata associated with that file. The Figure shows a session with the Project Name: “NAM”. Seven data files (denoted “Observations”) have been loaded; two Grids and six Networks have been defined. These Grids and Networks may represent locations being interpolated from or interpolated to. One interpolation Method has been created. Four have been created in the example. The example also shows that two Interpolations and one Derived variable have been defined. The Interpolation files contain a description of the combination of Observations, a Grid or Network, and Method used to define the interpolation. The resulting output files appear as Display files with a default suffix of .map. The Display files can be converted to and saved as PostScript files. A Derived variable contains the output of a mathematical formula or function.



.

Figure 4.2 Main screen during sample session

In the remainder of this chapter, the menu options are explained; bold face is used to denote a specific menu item. The six major subheadings in this chapter directly correspond to the six main menu options in Spherekit. The submenu selections

from the main menu appear below with an arrow symbol ⇒ for each level below the main menu.

File

⇒Environment

⇒⇒Set Project Directory
This option allows the user to return to a working environment that was left at an earlier time. Upon return, all variables are reloaded and the status of the program is restored. To use this option, the user is prompted for the names of all saved projects.

⇒⇒Create Project Directory
This option creates a new workspace into which a user can load data. Upon return, all variables are reloaded and the status of the program is restored. To use this option, the user is prompted to enter a name of the current project. The initial project directory has the default name: Work.

⇒⇒Delete Project Directory
This option deletes a previously created working environment. The user is prompted for the name of an existing project environment.

⇒Import Data
Use this selection to load your data and/or the built-in DEM into Spherekit. These features are described below.

⇒⇒Locations
A set of locations is classified either as a network (irregularly spaced) or a grid (regularly spaced).

⇒⇒⇒Network (irregularly spaced points)
If you select Network, you are prompted for the column numbers of the latitudes and longitudes in the input file. You also are prompted for the file name and the format of the file (binary or ASCII). For ASCII files, you can enter a delimiter other than a blank space. A final option is available to read in a location index for each location. This allows the observation values to be referenced by location number. To use this option, you are prompted for the column number of the index in the location file.

⇒⇒⇒ Grid (regularly spaced points)
If you select Grid, you are prompted for the starting location and the grid resolution in east-west and north-south directions, either in km or degrees. You are asked for a name to assign to the grid, so that it can be referenced later.

⇒⇒Values
To load the data values associated with the grid or network that you've defined, enter the file name where they may be found and the column number. You are also asked for the associated grid or network name. If an indexed location is associated with each observation, select that option and enter the column number for the location index.

⇒⇒DEM
A digital elevation model (DEM) is built into Spherekit. Its coverage is global, including elevations both above and below sea level. To access the DEM, enter the name of the grid or network on which elevation values are desired. Assign a file name to the created values and use the data set as any other observational data set.

⇒Export Data
To save a field, enter the field name, the desired file format (ASCII or native binary), and the desired file name.

⇒Delete Variable
Spherekit saves all created objects in external files. Any of the following can be deleted with this selection: observation file, grid or network description, interpolated field, error field, or postscript display.

⇒Quit
Exits Spherekit and returns to the operating system.

Processing

⇒Derived variable
A wide range of mathematical transforms can be applied using this option. The selected operation is applied to all elements of the data set. The dataset may be observations, an interpolated surface, cross-validation errors, or a previously transformed variable.

⇒⇒Linear transform
Change the units or rescale the data by specifying a constant (offset) and slope (scale factor).

⇒⇒Nonlinear transform
This option can be used to reduce the skewness or kurtosis in the data. The inverse transform is applied automatically following interpolation as a default. This inverse operation can be disabled if desired when the Begin Interpolation option is selected. The available nonlinear transforms are:

Log [ln(x)]
Exponential [exp(x)]
Power [x^p]
Box-Cox [$(x^p)/p$]
Logistic [ln[x/(1-x)]]
Angular [arcsin(sqrt(x))]
Trigonometric [sin(x) or cos(x)]

⇒⇒Average

Use this option to average the corresponding values in multiple data objects (or files). Each file must have the same dimension. This option can be used to create a monthly or annual average of a quantity prior to (or subsequent to) interpolation.

⇒⇒Formula

Using a built-in equation editor, you can create new variables and perform the interpolation directly on these new variables. Each file must have the same dimension. The inverse transform of the created formula is applied automatically following interpolation as a default. This inverse operation can be disabled if desired when the Begin Interpolation is selected. An example of the use of the Formula operation is described in Section 2. The example discusses the creation of a variable "temperature reduced to sea level." The formula is used to obtain a corresponding sea level value for the interpolation. Following the interpolation, the formula is inverted to restore physical meaning to the interpolated field.

⇒Variograms

A variogram depicts the autovariability of a data set or the cross-variability of a pair of data sets. Viewing a semivariogram is the initial step in using the kriging interpolation method.

⇒⇒ Variogram Type

Several types of variograms can be computed. For complete details, see Deutsch and Journel (1992).

⇒⇒⇒Semivariogram

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n}\frac{\left[f(\mathbf{x}_i)-f(\mathbf{x}_j)\right]^2}{n(n-1)}$$

⇒⇒⇒Cross-semivariogram

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=i}^{n}\frac{\left[f_1(\mathbf{x}_i)-f_2(\mathbf{x}_j)\right]^2}{n^2}$$

⇒⇒⇒ **General relative semivariogram**

$$\frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}\left[f(\mathbf{x}_i)-f(\mathbf{x}_j)\right]^2}{n(n-1)\left\{\sum_{i=1}^{n}\sum_{j=i+1}^{n}\left[f(\mathbf{x}_i)+f(\mathbf{x}_j)\right]\right\}^2}$$

⇒⇒⇒ **Pairwise relative semivariogram**

$$\frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j=i}^{n}\frac{\left[f_1(\mathbf{x}_i)-f_2(\mathbf{x}_j)\right]^2}{\left[f_1(\mathbf{x}_i)+f_2(\mathbf{x}_j)\right]^2}$$

⇒⇒⇒ **Semivariogram of logarithm**

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n}\frac{\left[\log\ f(\mathbf{x}_i)-\log\ f(\mathbf{x}_j)\right]^2}{n(n-1)}$$

⇒⇒⇒**Rodogram**

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n}\frac{\sqrt{\left|f(\mathbf{x}_i)-f(\mathbf{x}_j)\right|}}{n(n-1)}$$

⇒⇒⇒**Madogram**

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n}\frac{\left|f(\mathbf{x}_i)-f(\mathbf{x}_j)\right|}{n(n-1)}$$

⇒⇒**Variable**

Enter the variable name.

⇒⇒ **Variable 2**

If cross-semivariogram has been selected above, enter the name of the second variable.

⇒⇒ **# lags**
The number of lags is the number of x-values along the ordinate of the variogram graph.

⇒⇒ **Length unit lag (km)**
The length is the increment or distance between points along the ordinate of the variogram graph.

⇒⇒ () **Isotropic** () **Anisotropic**
Check the isotropic box for a variogram based upon pairwise distances without orientation considerations. Check the anisotropic box if there are variations in correlation with direction. For example, some natural phenomena show greater variations in the north-south direction than in east-west because climatic variables often show greater variability along the meridians.

If anisotropic has been selected the following options apply:

⇒⇒ **Angle (Clockwise from N-S)**
The angle is the major axis of the anisotropy.

⇒⇒ **Cone (1/2 angle)**
This option allows you to exclude points lying outside of a cone centered on a point and whose center line is directed in the angle selected above.

⇒⇒ **Maximum distance (km)**
This option allows you to exclude points lying greater than the specified distance from the center line of the cone.

⇒⇒ **View Variogram**
Select this option when you have completed the entry of the variogram parameters.

⇒⇒ **Superimpose Model**
If you selected semivariogram, you have the option of superimposing a semivariogram model on the data. To fit a model, the name of the model (exponential, Gaussian, spherical, or linear), and a first guess at the range and sill. Spherekit will provide first guesses for these values.

⇒⇒⇒**Nugget**
The nugget is value of the model f(d) as d-> 0.

⇒⇒⇒**Model**
The choices are exponential, Gaussian, spherical, and linear. The range and sill are the parameters. For the linear model, slope and power are the parameters.

⇒⇒ ⇒ **Isotropic/Anisotropic**
For an anisotropic model, enter the angle offset for the major axis and the anisotropy factor.

⇒⇒⇒ **Add Additional Model**
Some datasets display very different spatial variability properties at close and far distances. To accommodate this characteristic, Spherekit permits a variogram function to be the sum of two models. Typically, the range parameter of the two models would be set quite differently in the two models.

⇒⇒ **Constrain Lat/Lon**
The domain of a dataset can be limited to a rectangle in the latitude-longitude plane. To use this option, the user enters a bounding pair of latitudes and longitudes.

⇒⇒ **Constrain Values**
The range of values in a dataset can be constrained to lie within specified minimum and maximum values. For example, chemical mixing rations can be constrained to lie between zero and one using this option.

Interpolation

⇒ **Create Grid/Locations:**

⇒⇒ **Grid** (regularly spaced set of points)
Enter the latitude and longitude of the northwest corner of the grid as well as the north-south grid spacing in degrees. The grid can be masked to exclude sites over the ocean and other water bodies or can be masked to exclude land sites.

⇒⇒ **Network** (irregularly spaced set of points)
Enter the file name containing the network locations.

⇒ **Create method**
Refer to Section 3 for a complete description of the interpolation methods and parameters available.

⇒ **Begin Interpolation**
Selection of this option initiates the interpolation. A check is performed to ensure that a method and grid/network have been chosen. You are asked to provide a name for the output field.

Cross-validation

Cross-validation refers to the process of removing one of the n observation points and using the remaining n-1 points to predict its value. This process is repeated at

each data point, with each estimate using n-1 points.

⇒Cross-validation to Network
The interpolation error at each data point is the difference between its observed and predicted values. This difference field can be displayed and manipulated as any other field in Spherekit. The user is prompted for a field name for the generated network. The display of an error field includes the error field, a table of error summary statistics, a histogram, and a scatterplot. The overall error is computed both as the mean absolute error (MAE) or the root mean square error (RMSE).

⇒Cross-validation to Grid
This option is used to interpolate the error field to a uniform grid (using a user-specified method). This second interpolation has the effect of reducing spatial bias in the estimate of the error. The output is an error field that can be displayed and manipulated as any other field in Spherekit. The display of an error field includes the error field, a table of error summary statistics, a histogram, and a scatterplot. The overall error is computed both as the mean absolute error (MAE) or the root mean square error (RMSE).

Display

⇒Field
Any field may be displayed using this selection. There are many different ways to display the field using the descriptions below:

⇒⇒ Display Name
Give a name to the display of the field so that it can be referenced at a later time.

⇒⇒ Display Field
Select the name of the field to be displayed.

⇒⇒ Graph Type

⇒⇒⇒Isoline
This graph type is a smoothed contour plot.

⇒⇒⇒ Grid Plot
This graph type is an unsmoothed plot that shows the resolution of the underlying data.

⇒⇒⇒None
Selecting this option will display no field. It is used to show data locations, triangulation, etc. (see Superimpose options) without any overlying data.

⇒⇒**Options**

⇒⇒⇒**Superimpose**
Any of the following can be superimposed on a displayed field. The default is no superimposed features.

⇒⇒⇒⇒ **Data Values**

⇒⇒⇒⇒**Triangulation**

⇒⇒⇒⇒ **Regular Grid**

⇒⇒⇒**Zoom**
The size of the displayed image can be set smaller or larger using this option.

⇒⇒⇒⇒**50%**

⇒⇒⇒⇒**100%** (default)

⇒⇒⇒⇒**150%**

⇒⇒⇒**Masking**
Ocean or water-covered locations can be masked out using this option. Alternatively, land areas can be masked.

⇒⇒⇒⇒ **No masking** (default)

⇒⇒⇒⇒ **Mask out sea**

⇒⇒⇒⇒ **Mask out land**

⇒⇒⇒**Palette**
The palette determines the range of color used in the display. Options are:

Topographic
Tan Blue
Tan Violet
Green Yellow
Red Blue
Blue Red
Chromatic (default)
Red
Blue

Green
Gray (for black and white display)

⇒⇒⇒ **# of Bins**

6
7
8
9
10 (default)

⇒⇒⇒ **# of Decimals**

0
1
2 (default)
3

⇒⇒⇒**Orientation**
This option permits a change from horizontal to portrait orientation.

⇒⇒⇒⇒**Landscape** (default)

⇒⇒⇒⇒**Portrait**

⇒⇒⇒**Scale**

⇒⇒⇒⇒ **Base Scale on Field** (default)
The scale will be selected based on the values of the interpolated field.

⇒⇒⇒⇒ **Base Scale on Source**
The scale will be selected based on the values of the observations.

⇒⇒⇒**Projections**
Select a projection appropriate to your data set. You will be prompted for any needed information associated with the projection.

Mercator
Orthographic
Mollweide
Albers Conic
Robinson
Plate Carree
Stereographic North Pole

Stereographic South Pole
Stereographic General

⇒Help
This selection accesses the on-line help for Spherekit

⇒Metadata
Using this selection, the metadata for any field can be displayed.

⇒Overview
Using this selection, the overview information for Spherekit is displayed.

⇒How To
This selection contains hints on performing certain functions in Spherekit.

⇒Read Me
This selection displays the README file that accompanies Spherekit.

⇒Log File
This selection displays the current log file.

⇒Credits
This selection displays the opening screen of Spherekit containing the credits.

Options

⇒Echo Commands
With this option disengaged, the commands that you select are written to the default window (the one from which you started Spherekit).

⇒Log Commands
With this option engaged, the commands that you select are written to a log file.

⇒Suppress GMT Commands
With this option engaged, all GMT commands are written to the window from which you started Spherekit. The default value is to suppress this display.

⇒Create Scale
This option allows the user to create a set of bins to use as a scale in the display of fields. To use this option, enter a desired number of bins and a desired minimum and maximum value associated with each bin.

5. Interpolation algorithms

The primary purpose of Spherekit is to estimate an interpolation function $\hat{f}(\mathbf{x})$ on a network or grid of points using observed values $f(\mathbf{x})$ at other points $\mathbf{x}$. The term interpolation implies that the estimated function agrees with the observations at those points:

$$\hat{f}(\mathbf{x}_i) = f(\mathbf{x}_i) \qquad \text{for all observations } i\,. \qquad (1)$$

Use of (1) is equivalent to assuming the observations to be error-free. If (1) were not to hold, the estimation function would be a smoothing rather than an interpolation function. Smoothing functions are not presently included in Spherekit.

Because of (1), Spherekit includes a preliminary check to determine if any interpolation point coincides with an observation point. If so, or if the two points lie within a small tolerance of one another, the interpolated value is set to that of the observation. Spherekit also will extrapolate, if an interpolation point is outside the convex hull of the observations. Extrapolation should be used with caution as accuracy will be poor in many instances.

Five classes of interpolation methods are available in Spherekit:

* Inverse distance weighting
* Triangulation
* Multiquadric
* Thin plate spline
* Kriging

Methods vary in terms of the number of observation points used in the estimate, the smoothness of the interpolation field, and the computational speed and storage requirements. Each method has been modified for use on the sphere by utilizing geodesic distance in lieu of Euclidean distance. Additional modifications for the sphere are used where possible.

5.1 Neighborhood size selection

Interpolation methods can be categorized as either global or local. A global method uses all observations to estimate an interpolated value. This strategy is not feasible when the number of observations is relatively large. A local method uses only observations lying within a specified neighborhood of the interpolation point. Most of the Spherekit interpolations can be carried out using either strategy.

The neighborhood size of a local method is specified as a radius (in kilometers) or a number of included points. An override can be invoked to put lower and upper bounds on the variable that was not specified. Use of the override will expand or

contract the neighborhood size if needed to meet the criteria. Spherekit also permits the specified radius to be computed as the average radius of a specified number of points.

5.2 Inverse distance weighting

Inverse distance weighting is the simplest interpolation method considered. In this method, a weighted average is taken of the observed values (neighbors) falling within the neighborhood of the interpolation point. The weights are an inverse function of distance. The user has control over the mathematical form of the weighting function, the size of the neighborhood, a bias correction, and an extrapolation/gradient correction. The description of the neighborhood size selection was given in Section 5.1.

The estimation function is obtained using:

$$\hat{f}(\mathbf{x}) = \frac{\sum_{i=1}^{m} w[d(\mathbf{x},\mathbf{x}_i)]f(\mathbf{x}_i)}{\sum_{i=1}^{m} w[d(\mathbf{x},\mathbf{x}_i)]} \qquad (2)$$

where $w[]$ is the inverse function of the geodesic distance d between the interpolation point $\mathbf{x}$ and one of its $\mathbf{x}_i$. There are m neighbors, where m usually is much less than n.

Weighting function

The simplest weighting function is inverse power:

$$w(d) = 1/d^p \qquad p>0$$

The value of p is specified by the user; $p= 2$ is a common choice. If p is too large, the interpolated value tends towards that of the nearest neighbor, with little influence of other points in the neighborhood. If p is too large, the interpolated value tends towards that of its average neighborhood value, with little variation within the neighborhood.

Shepard's weighting function (Shepard, 1968) differs from inverse power in the outer region of the neighborhood. For distances within 1/3 of the neighborhood radius, the function is inverse power with $p=2$. In the outer 2/3 of the neighborhood, the function is a quartic. The quartic produces a more rapid falloff than inverse square would produce in the outer region. The resulting function is continuous and differentiable at the region interface and goes to zero at the neighborhood boundary.

The other weighting function available is Cressman's function. This function differs from all others in as it is a smoothing, rather than an interpolative function. Its functional form is:

$$w(d)= (p^2 - d^2)/(p^2+ d^2)$$

where p is a user defined distance constant with $p>d$ for all d.

Anisotropy correction

The function in (2) is entirely distance-based; all points lying at a particular distance are given equal weight. No consideration is given to the angular distribution of neighbors relative to the interpolation point. Points that are angularly clustered are weighted nearly equally to points which are angularly separated. Neglecting the anisotropic distribution of neighbors can introduce spatial bias in the estimate.

Spherekit includes an anisotropy corrector to adjust weights based on their angular distribution. The correction is based on Shepard (1968) with spherical angles replacing planar angles. The modified weight w_i for the interpolation at point $\mathbf{x}_j$ is computed from:

$$w_i(d,\theta) = w_i(d)\left[1+\alpha\frac{\sum_{j=1}^{m} w_j[1-\cos(\angle \mathbf{x}_i\mathbf{x}\mathbf{x}_j)]}{\sum_{j=1}^{m} w_j}\right]$$

where $w_i(d)$ is the weight based upon distance alone. The parameter α is included for sensitivity studies and can be set to values between zero and one. A value of zero produces no correction, while one (the default value) produces its full effect.

Gradient correction

A characteristic feature of the inverse distance weighting method is that the interpolation function is forced to have its local extrema at the observation points (or on a boundary). That is, the interpolation function has zero gradient at the observed points and nowhere else. Spherekit's gradient corrector adds a "trend" term $f^*(\mathbf{x}_i)$ to each observation value. The method is described below. Further justification for the steps can be found in Shepard (1968) and Willmott et al. (1985).

The trend term is obtained from the gradient of the observed field. The gradient at an observed point is estimated as:

$$\nabla f(\mathbf{x}_i) = \frac{\sum_{\substack{j=1 \\ j\neq i}}^{m} w_j \frac{(\mathbf{x}_i - \mathbf{x}_j)[f(\mathbf{x}_i) - f(\mathbf{x}_j)]}{\left[d(\mathbf{x}_i, \mathbf{x}_j)\right]^2}}{\sum_{\substack{j=1 \\ j\neq i}}^{m} w_j}$$

where w_j is the weighting factor determined above. If $\mathbf{x}$ is the interpolation point, the trend term associated with observation point $\mathbf{x}_i$ is:

$$f^*(\mathbf{x}_i) = \nabla f(\mathbf{x}_i) \bullet (\mathbf{x} - \mathbf{x}_i) \frac{v}{v + d(x, x_i)}$$

The final term on the right varies from zero to one for positive v which is determined from:

$$v = \alpha \frac{\max[f(\mathbf{x}_i)] - \min[f(\mathbf{x}_i)]}{\max|\nabla f(\mathbf{x}_i)|}$$

where the max and min functions are taken over all observation points, and α is a user-specified parameter. For α=0, there is no gradient correction, as $f^*(\mathbf{x})$= 0. For large α , the quotient in the definition of f* is one, and the gradient corrector has its full effect.

This correction method should be used with care because extrapolated estimates may exceed the range of realistic values when data networks are deficient. The default value of α is 0.1.

5.3 Multiquadrics and thin-plate splines

The next two interpolation methods produce estimates of a surface function $f(\mathbf{x})$ rather than estimates at individual points. These functions represent linear combinations of some function of distance to the data such that:

$$f(\mathbf{x}) = \sum_{i=1}^{m} w[d(\mathbf{x}, \mathbf{x}_i)] a_i \qquad (3)$$

where w[] is a specified function of the geodesic distance d between the interpolation point $\mathbf{x}$ and an observation point $\mathbf{x}_i$ and m denotes the number of points used in the estimate. The multiquadric and spline methods differ from one another in the functional form of w[]. The observed values do not appear explicitly in (3) Instead, they are reintroduced when the coefficients a_i are computed by Spherekit to satisfy (1). There are two distinct methods of fitting the surface to the observed values.

The "traditional" surface fit methods use all n observations ($m=n$) to obtain the interpolated field. Such global methods require the solution of n simultaneous equations and the inversion of an nxn matrix. The coefficients are computed once and can be stored for later use. To use the traditional method, select "Use all points" for the neighborhood size. Spherekit cannot perform the matrix inversion required by the traditional method when n is greater than several hundred. A warning is displayed if such an operation is requested by the user.

An alternative approach is to solve (3) using only observations within a specified neighborhood of each interpolation point. Using this local method, a new set of coefficients is obtained for each interpolation point and coefficients are not saved. Section 5.1 describes the neighborhood size selection procedure.

Multiquadrics

Multiquadric methods (Hardy and Gofert, 1975) express the interpolation function as a linear combination of quadric (e.g.. hyperbolic or conical) functions. Modification for the sphere has been formulated by Pottmann and Eck (1990). The functions in (3) are of the form:

$$w(d) = \sqrt{1 + R^2 - 2R\cos\ d}$$

where R is a user-specified tension parameter and d is the angular distance between the observation and interpolation points. Values equal to or slightly less than 1 are recommended.

An option is available to use reciprocal multiquadric interpolation. In this case, the above function is replaced by its reciprocal.

$$w(d) = \frac{1}{\sqrt{1 + R^2 - 2R\cos\ d}}$$

In most of our test cases, the standard multiquadric method outperformed the reciprocal multiquadric method.

Splines

In the thin-plate spline method, the functions in (3) can be any of the following forms:

$$w(d) = d^2 \log\ d$$
$$w(d) = d^2 \log\ d^2$$
$$w(d) = d^2(\log\ d - 1)$$

where d is the angular distance between the observation and interpolation points. A general reference is Watson (1992). More technical references are: Franke (1982), Harder and Desmarias (1972), and Sandwell (1987).

5.4 Kriging

Kriging differs from the other methods in that spatial variability information is used in the estimation procedure. Many forms of kriging are possible; Spherekit uses simple kriging, computed as:

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{m} w[d(\mathbf{x},\mathbf{x}_i)] f(\mathbf{x}_i)$$

where d is the angular distance between the observation and interpolation points. The weighting function is one or some combination of the following:

$$w(d) = a + \sigma^2 e^{-d/r} \qquad \text{(Exponential)}$$

$$w(d) = a + \sigma^2 e^{-(d/r)^2} \qquad \text{(Gaussian)}$$

$$w(d) = a + \sigma^2 \left(1 - \frac{3d}{2r} + \frac{d^3}{2r^3}\right) \qquad \text{(Spherical)}$$

$$w(d) = a - cd \qquad \text{(Linear)}$$

The free parameters a, σ^2 and r are known as the nugget, sill and range, respectively. For the linear model, c is the slope. All parameters are estimated through an iterative process. The resulting weights are determined by minimizing the variance of the estimate among all possible linear unbiased estimates. Such values are obtained using the covariance structure (or semivariogram) of the field.

The sill equals the variance of the observations at the limit of large distances. That is, the sill is the non-spatial component of the variability of the observations. The range refers to the range of distances for which there is significant covariance. The description of the variogram is given below in Section 5.2.

The semivariogram calculation is carried out using the GSLIB library software package (Deutsch and Journel, 1992). The semivariogram is defined as:

$$\gamma(s) = \frac{1}{2} \text{Var}[f(\mathbf{x}_2) - f(\mathbf{x}_1)] \qquad \text{such that } d(\mathbf{x}_1,\mathbf{x}_2) = s \ .$$

The kriging interpolation produces both the estimated field and the expected error about this estimate. The latter is the expected error assuming that the model specification is correct.

5.5 Triangulation

The triangulation method begins by connecting the observation points into a network of triangles. The purpose is to identify a neighborhood of nearby observation points to be used in the interpolation. There are many possible triangle networks that can be produced, among them the Delaunay triangulation. Spherekit uses Renka's algorithm and source code (Renka, 1984) to carry out a Delaunay triangulation (Okabe et al., 1992) of the observation points.

Once the triangulation is obtained, there are two ways that the interpolation can be performed. The simplest method is to take a linear interpolation of the values at the vertices of the triangulation. The resulting interpolation function is continuous, but not differentiable across an edge.

The second triangulation method uses a polynomial fit within each triangle (Renka, 1984). This method requires an intermediate step of fitting a cubic spline on each triangle edge. The interior values are taken as weighted averages of the edge values. In the intermediate step of fitting the spline, approximations to the gradient at each point must be computed. The user has the option to specify which of two gradient estimation methods is used. The local method uses only a few neighboring points, while the global method uses all points. The latter method may not be possible if the number of points is larger than a few hundred.

6. Download and installation

Spherekit is available for download from the Spherekit web page at:

http://www.ncgia.ucsb.edu/pubs/spherekit/main.html .

Binary versions currently are available for the DEC Alpha and SGI (IRIX 6.2). We expect to add other platforms in the future. Alternatively, the full source code is available for use on other platforms and for users who wish to modify the code. Several auxillary software packages are required to use Spherekit: GMT3.0, TclTk, ghostview, and netCDF, and these tools are included in both the binary and source code distributions.

An optional digital elevation model (DEM) also is available with the package. The DEM has global coverage over both land and sea, with 5 minute resolution. It is used for carrying out topographically assisted interpolation. The required Spherekit executable code takes up 20 Mb while the optional DEM requires an additional 37 Mb.

Binary Distributions (currently available only for DEC Alpha and SGI - IRIX 6.2)

To install the binary executables:

1. Download and uncompress the files for your system.
2. Run ./make_sk in the Spherekit directory
3. Run ./test_sk to ensure the accuracy of Spherekit's interpolations
4. Add the SK directory to your path statement
5. (Optional)
   a. Download the DEM file
   b. cd to the Spherekit directory and run set_sk_demdir

Source Code Distribution

With the source distribution you can compile Spherekit from the ground up. As a convenenience for you, we have also included the source files for netCDF, Tcl/Tk, a subset of GMT3.0, and ghostview1.5.

Since Spherekit has been tested with these modules, it is **strongly** suggested that you use the source files provided in this distribution. We have gone to a lot of work to bring these components together into one application framework. Please take advantage of this labor. Spherekit will probably **not** be kept forward compatible with later versions of these packages.

You will need to "make" these programs and libraries in addition to Spherekit.

The compilation process for the source distribution is comprised of these major steps:

I. Download and uncompress the source distribution file

II. cd to SKv1/netcdf-3.3/, read the README file there, and make the netCDF library

III. cd to SKv1/GMT3.0, read the README file there, and make the GMT3.0 executables

IV. cd to SKv1/TclTk, read the README file there, and make Tcl and then Tk

V. cd to SKv1/ghostview1.5, read the README file there, and make ghostview

VI. cd to SKv1, read the README file there, and make Spherekit

VII. run ./test_sk to test the interpolators

VIII. add the SK directory to your path statement

IX. (Optional) Load the DEM according to Step 5 in the binary distribution above.

Porting Spherekit to a new platform is a fairly involved task, requiring ANSII C and FORTRAN 77 compilers. If you port Spherekit, please let other users benefit from your efforts by contacting a member of the Spherekit team and making the binaries available to us. We will add your binaries to those listed above, and credit your work.

Questions or comments about the installation procedure should be directed to:
chris@geog.ucsb.edu .

7. References

Deutsch, C. V. and A. G. Journel (1992) GSLIB: Geostatistical Software Library and User's Guide, New York, Oxford University Press.

Fisher, N. I., T. Lewis, and B. J. J. Embleton (1987) Statistical Analysis of Spherical Data, Cambridge University Press, 329 pp.

Hardy, R. L. and W. M. Gofert (1975) Least squares prediction of gravity anomalies, geoidal undulations, and deflections of the vertical multiquadric harmonic functions, *Geophysical Research Letters*, 2, 423-426.

Pottmann, H. and M. Eck (1990) Modified multiquadric methods for scattered data interpolation over a sphere, *Computer Aided Design*, 7, 313-321.

Okabe, A., B. Boots, and K. Sugihara (1992) Spatial Tessellations, Wiley, 532 pp.

Raskin, R. G. (1994) Spatial Analysis on the Sphere, National Center for Geographic Information and Analysis Technical Report, 94-7, 44 pp.

Renka, R. J. (1984) Interpolation of data on the surface of a sphere, *ACM Transactions on Mathematical Software*, 10, 417-436.

Shepard, D. (1968) A two-dimensional interpolation function for irregularly-spaced data, Proc. 23rd National Conference ACM, ACM, 517-524.

Watson, D. F. (1992) Contouring: A Guide to the Analysis and Display of Spatial Data, Pergamon Press, 321 pp.

Willmott, C. J. On the Evaluation of Model Performance in Physical Geography, 1984. In Spatial Statistics and Models, G. L. Gaile and C. J. Willmott (eds.), Dordrecht, Holland: D. Reidel, 443-460.

Willmott, C. J., S. G. Ackleson, R. E. Davis, J. J. Feddema, K. M. Klink, D. R. Legates, J. O'Donnell and C. M. Rowe (1985) Statistics for the Evaluation and Comparison of Models, *Journal of Geophysical Research*, 90(C5), 8995-9005.

Willmott, C.J., C.M. Rowe, and W.D. Philpot (1985) Small-Scale Climate Maps: A Sensitivity Analysis of Some Common Assumptions Associated with Grid-Point Interpolation and Contouring, *American Cartographer*, 12, 5-16.

Willmott, C.J. and K. Matsuura (1995) Smart interpolation of annually averaged air temperature in the United States, *Journal of Applied Meteorology*, 34(12), 2577-2586.

Willmott, C.J. and S.M. Robeson (1995) Climatologically Aided Interpolation (CAI) of terrestrial air temperature, *International Journal of Climatology*, 15(2), 221-229.