IRIS Explorer User’s Guide
Release 3

Document Number NP2872
ISBN 1−85206−110−3

Chapter 1

Getting Started

This chapter introduces you to IRIS Explorer using five sample maps, each from a different scientific field. The examples cover:

- image processing
- numerical mathematics
- molecular chemistry
- atmospheric physics
- computational fluid dynamics

1.1. Overview

You can create maps for all occasions, for example, to generate a 3D image of vortices around hydro–electric turbines, or to visualize carbon dioxide concentration as a function of geographic location. These activities involve a number of separate tasks such as reading the data into the map, colormapping a gradient, extracting a 2D slice of a 3D dataset and visualizing the result. The modules in the map carry out tasks like these.

The five maps presented here use authentic data to illustrate the workings of IRIS Explorer. Each map deals with a specific issue, and the modules provide ways of altering or enhancing the data so as to make its significance clear, and perhaps to elicit correlations that might otherwise have gone unnoticed.

The maps are complete. You need only launch, or open, the map in the Map Editor, and then try varying the parameters as suggested in each example. Chapter 2 explains how you set about creating your own maps. **Table 1–1** defines some of the terms you will encounter when you start IRIS Explorer.

Term	Meaning
Map Editor	Work area for creating and modifying maps
Module Librarian	Lists maps and modules available for use
Module	Performs an algorithmic function on the data passing through it
Control Panel	Gives the user access to the module functions
Widgets	Dials, sliders, buttons, and text slots for changing parameters
Map	A collection of modules that carry out a series of related operations on a dataset
Image	A 2D array that can be viewed using the *DisplayImg* module
Geometric Object	A collection of points, lines, polygons or other shapes that can be displayed by the *Render* module

Table 1–1 Some Components of IRIS Explorer

1.2. Launching a Map

You can open maps in two ways: from the command line in the UNIX shell, and from the Module Librarian in IRIS Explorer.

1.2.1. Opening Your First Map

If IRIS Explorer is not running, you can use the command line in UNIX when you open a map for the first time. Change to */usr/explorer/maps* and type the IRIS Explorer command at the % prompt:

```
cd /usr/explorer/maps
explorer −map mapname.map
```

The *mapname.map* is the name of the map you want to open, for example, *chemistry.map*;. The name of each map shown in this chapter is given in the introduction to the map.

IRIS Explorer starts up and the map appears in the Map Editor. All the modules are *selected* (highlighted in white), and you can move the entire map by dragging on the title−bar of a single module.

Click on the background of the Map Editor to deselect the modules in the map before you experiment with individual modules. The highlighting disappears.

1.2.2. Opening Subsequent Maps

Once IRIS Explorer is running, you can use the Module Librarian to open a map. Follow these steps.

1. To clear the Map Editor of modules, click on the Edit menu and select "Destroy All."
2. Go to the Module Librarian next to the Map Editor and use the vertical scroll bars to locate the maps, which are shown in blue.
3. Select the map you want by placing the cursor on the map name and clicking the right mouse button, then selecting "Launch" from the pop−up menu.

   You can also hold down the left mouse button, drag the map over to the Map Editor and release the button.

When you release the mouse button, the map appears, module by module, on the screen. The menu bar displays a number in angle brackets, for example, <5>, which tells you initially how many modules are in the map; the number decreases as modules appear on screen, indicating how many modules remain to be launched.

Quitting IRIS Explorer

To quit IRIS Explorer, select "Quit" from the Admin menu in the Map Editor.

1.2.3. Opening Modules

You can expand the mini control panel (shown in **Figure 1−1**) of any module into the full−scale control panel by clicking on the square *Maximize* button at the right top corner of the control panel. When you do this, the parameter widgets and their values are displayed for easier manipulation.

Figure 1−1 Module Control Panel

1.3. Image Processing

This example illustrates how you can use IRIS Explorer modules to process an image. The *KoreaContour* map takes an aerial image of a region of the Korean landscape, renders it in 3D, and overlays a contour map showing altitudes. The resulting image is displayed on−screen. The map file is */usr/explorer/maps/KoreaContour.map.*

1.3.1. Opening the Image Map

To open the *KoreaContour* map, displayed in **Figure 1−2**, type

```
explorer −map koreaContour.map
```

The seven modules are connected to one another by blue wires. The two modules on the left read in the image files, and the module on the far right displays the processed image. In between these are four modules that manipulate the image data.

Figure 1–2 Processing an Image

The seven modules in the map have the following functions:

- *ReadImg<2>* reads in the first image file (2D aerial image of a region of Korea).
- *ReadImg* reads in the second image file (altitude data for the same region).
- *DisplaceLat* displaces each node of the aerial image in the vertical direction according to the local value of the altitude.
- *DisplaceLat<2>* performs the same function as *DisplaceLat*, but with greater vertical distortion.
- *LatToGeom* creates polygons of the displaced data from *DisplaceLat*.
- *Contour* creates contours colormapped according to height, using the displaced data from *DisplaceLat<2>*.
- *Render* displays the output from *LatToGeom* and *Contour* as the surface with overlaid contours.

1.3.2. Experimenting with the Image Map

Both the aerial image and the altitude data go through *DisplaceLat* (see **Figure 1–3**), which displaces the image data so that topographical features become 3D.

- You can regulate the scale of the 3D rendering by turning the Scale dial on the module control panel.

  The result is a 3D version of the original 2D image, showing topographical features of the Korean countryside.

Figure 1–3 *DisplaceLat* Control Panel

The output of *DisplaceLat<2>* passes into *Contour* (see **Figure 1–4**), which creates a series of lines at varying heights above sea level just as in a topographic map. The scale factor for *DisplaceLat<2>* should be greater than that for *DisplaceLat*, to make the contour lines visible.

Figure 1–4 *Contour* Control Panel

To set the relationship between scale factors, the Scale parameter output port on *DisplaceLat* is wired to the Scale parameter input port on *DisplaceLat*<2>.

- You can alter the number of contour levels by moving the slider widget on the module control panel.
- You can also change the altitude range for which contours are shown by resetting the minimum and maximum levels.

The result is shown in *Render* (see **Figure 1–5**). You can open the *Render* window by clicking on the *Maximize* button and rotate the object in the window.

- To move the object, place the cursor on the object, hold down the left mouse button, and drag the object around. It continues to rotate when you release the mouse button.
- To stop the object, click on the background of the *Render* window.

Figure 1–5 The Contour Map in *Render*

For more information on using the *Render* menus to manipulate objects, see **"Visualizing Data"** in Chapter 3.

1.4. Numerical Mathematics

This example illustrates how to use IRIS Explorer modules to read in a short program script describing a mathematical function and visualize the data in the *Render* window. The mathematical function describes diffusive heat flow. It is a finite difference

Footnote: For more information on finite difference techniques, see Mitchell, A.R., and Griffiths, D.F., "The Finite Difference Method in Partial Differential Equations," John Wiley and Sons, 1980.

stencil for solving problems in parabolic differential heat conduction.

The form of the equation in this example is dimensionless; it serves as a normalized thermal model. A map using a

The form of the equation in this example is dimensionless; it serves as a normalized thermal model. A map using a specific form of such an equation with actual data can be used, for example, to test the conductivity of steel.

1.4.1. Opening the Heat−flux Map

To open the "heat−flux" map (see **Figure 1−6**), type

```
explorer -map heat-flux.map
```

Figure 1−6 Map for a Heat Flux

The map contains six modules, which have these functions:

- *LatFunction* reads in a program script that sets up a heat source and a heat sink, and generates the time−varying temperature distribution around them.
- *DisplaceLat* displaces the 2D lattice temperature data to create a 3D visualization of the way in which the temperature changes over time.
- *GenerateColormap* colormaps the temperature lattice by value.
- *LatToGeom* converts the displaced lattice data into a surface geometry.
- *Legend* produces a scale showing the temperature gradient.
- *Render* displays the surface and the legend.

Note: A *lattice* is an IRIS Explorer data type, which is used to store an ordered array of data. For more information, see **"Understanding IRIS Explorer Data Types"** in Chapter 2.

1.4.2. Experimenting with the Heat−flux Map

LatFunction reads in a short program written in the Shape language, which is described in detail in Chapter 10, "Module Prototyping with Shape" in the *IRIS Explorer Module Writer's Guide.* It is a C−like language that has capabilities for operating on whole lattices with a single statement. The program is saved in a file called *heat−flux* in */usr/explorer/maps* and can be run as often you choose.

The data passes from *LatFunction* to *Displacelat*, which displaces the 2D data into a 3D array to show the heat source as a peak and the heat sink as a trough. You can change the degree to which the peaks project from the surface by turning the dial on *Displacelat.* This causes the data to be displaced out of the plane to a greater or lesser degree.

GenerateColormap colormaps the heat−flux data according to temperature value. You can change the colors associated with each temperature value by using the option menus on the control panel. Refer to **"Creating Colormaps"** in Chapter 3 for more information.

LatToGeom uses the heat−flux and colormap data (which are both IRIS Explorer lattices) to create geometry for display in *Render* (see **Figure 1−7**). *Render* can only display geometry, and all lattice data must be converted first.

Figure 1−7 Visualization of the Flux Peaks

Once the data is displayed in *Render*, you can change the background color of the window by selecting "Edit Background Color" from the Viewing menu. When the Background Color Picker appears:

- Click anywhere on the color wheel to select a color
- Drag the slider to increase the intensity of the selected color

1.5. Molecular Chemistry

This example illustrates how you can use IRIS Explorer modules to construct a 3D model of a complex organic molecule and then calculate the optimal radius and likely path for a probe atom investigating the target molecule.

1.5.1. Opening the Chemistry Map

To open the "chemistry" map (see **Figure 1–8**), type:

```
explorer -map chemistry.map
```

Figure 1–8 Visualizing a NutraSweet Molecule

The map takes configuration data for a NutraSweet® molecule and creates a ball–and stick representation of it and a dot surface that shows the region that is accessible by a probe atom.

The map contains seven modules:

- *FileList* lets you select which of the available chemistry molecule data files you want to visualize.
- *ReadPDB* reads in the molecular configuration data.
- *BallStick* creates the ball–and–stick representation of the molecule.
- *AtomicSurf* calculates a solvent–accessible surface around the molecule. It uses the USURF

  Footnote: Written by Joseph J. Moon and published under the auspices of the Quantum Chemistry Program Exchange (QCPE), Indiana University. Program #566

  program to define a solvent–accessible surface around the molecule.
- *GenerateColormap* produces a colormap which is used to color the atoms in the molecule and the dot surface.
- *LatToGeom* converts the surface data in IRIS Explorer lattice format into geometry.
- *Render* displays the ball–and–stick geometry of the molecule and the solvent–accessible surface data as a dot surface.

1.5.2. Experimenting with the Chemistry Map

The molecular configuration data is read in by *ReadPDB*. The input file is in the Brookhaven Protein DataBase (PDB) format, a commonly used format for saving descriptions of proteins, including atom positions and properties, and bond locations.

BallStick (see **Figure 1–9**) receives the data in pyramid form and generates sphere–and–cylinder geometry. (The pyramid is an IRIS Explorer data type.) The radius of each sphere is calculated according to the van der Waals' radius of the corresponding atom.

- You can increase or decrease the radii of all spheres in *BallStick* proportionately by turning the dial on the module control panel.

Figure 1–9 *BallStick* Control Panel

The molecule data also passes into *AtomicSurf* (see **Figure 1–10**). It uses the van der Waals' radius of each atom and adds a solvent probe radius to it. The surface points in this "solvation layer" represent the positions in which a solvent probe would be in contact with the target molecule.

Figure 1–10 *AtomicSurf* Control Panel

- You can change the probe radius for which the solvent–accessible surface is calculated by moving the slider on the control panel. The smaller the value, the finer the surface.

  A value of 1.5 Angstroms approximates the radius of a water molecule, a commonly used probe. Probes that are too small or too large are not useful, particularly as the probe size approaches that of the target molecule.
- You can also change the surface dot density by turning the control panel dial. The larger the value, the larger the number of dots produced.

  It is best to use lower densities for large molecules, otherwise the visualization becomes very cluttered. A density of about 10 dots/Angstrom^2 works well for molecules which have sizes comparable to that of NutraSweet.

GenerateColormap creates a default colormap for the molecule based on atomic number. The domain is set at a minimum of 0 and a maximum of 16 to accommodate the atoms, which are hydrogen, carbon, oxygen, and nitrogen.

It also colormaps the dots in the solvent–accessible surface according to the parent atom of each dot. This provides you

with information about which atoms a given probe will touch as it traverse the surface of the molecule.
- You can change the default color settings by using the option menus on the control panel. Refer to **"Creating Colormaps"** in Chapter 3 for more information.

You can use the capacities of *Render* to look at the NutraSweet molecule in great detail (see **Figure 1−11**). You can:
- *rotate* the molecule by holding down the left mouse button and dragging the molecule around with the cursor
- *translate* the molecule by holding down the middle mouse buttons and dragging the molecule around using the cursor
- *zoom* in and out on the molecule by holding down both the left and middle mouse buttons at the same time and moving the cursor up and down on the screen
- use the Lights menu to control the various light sources and highlight less distinct areas of the dot surface.

Figure 1−11 The Molecule in *Render*

For more information on using *Render*, read **"Visualizing Data"** in Chapter 3.

You can use *FileList* to select another chemistry molecule to examine.

1.6. Atmospheric Physics

This example illustrates how you can use IRIS Explorer modules to visualize atmospheric data. Intense storms can produce tornadoes, high winds, and hail, and it is possible to simulate these storms by integrating a set of mathematical flow equations. These equations can, for example, predict values for the wind speed and direction, air temperature, humidity, pressure, and water content every 5 to 10 seconds on a lattice of grid points 500 to 1000 meters apart.

The data in the example map is taken from a simulation of a single severe storm made by the storm group at the University of Illinois

Footnote: Dr. Robert Wilhelmson, NCSA Research Scientist and Professor of Meteorology in the Department of Atmospheric Sciences (DAS) at the University of Illinois at Champaign−Urbana and Dr. Lou Wicker, post−doc, and Crystal Shaw, research programmer, both at NCSA and DAS.

. The map shows an isosurface of rain density and a volume rendering of air buoyancy. An isosurface is a surface which passes through all points in a 3−D dataset where the data has a particular value. The data is in the form of a uniform lattice.

1.6.1. Opening the Volume Map

To open the "volume" map (see **Figure 1−12**), type:

```
explorer −map volume.map
```

Figure 1−12 Simulating an Evolving Storm

The map contains eight modules:
- *ReadLat* reads in the rain density data.
- *ReadLat<2>* reads in the water buoyancy data.
- *ReadLat<3>* reads in the colormap for the water buoyancy data.
- *IsosurfaceLat* generates an isosurface of the rain density.
- *WireFrame* creates a frame that defines the boundaries of the storm data.
- *GenerateColormap* modifies the colormap for the buoyancy data.
- *VolumeToGeom* converts the volume data to geometry for *Render*.
- *Render* displays the rain density isosurface and the water buoyancy volume.

Note: For volume rendering techniques to display the volume as shown in this map, you require alpha blending hardware. Some workstations do not have this hardware, but they can display the volume adequately if the Splat type on *VolumeToGeom* is set to "Point" (see **"Experimenting with the Volume Map"** below).

1.6.2. Experimenting with the Volume Map

This map uses both surface rendering (the isosurface) and volume rendering (the haze) in one visualization to show the relationship between a number of variables which have been calculated in 3−D space during the modelling of the evolution of the storm.

IsosurfaceLat (see **Figure 1−13**) calculates an isosurface from the density data and outputs it as geometry. The threshold value is the density value for which an isosurface is created.

- You can change the threshold value by turning the dial on the control panel, and thereby change the shape of the isosurface. This indicates how the rain density varies throughout the storm space.

Figure 1−13 *IsosurfaceLat*Control Panel

VolumeToGeom volume−renders the buoyancy data as a hazy cloud surrounding the isosurface. The entities that make up the cloud are called splats, small planar shapes that simulate fuzzy blobs on screen.

- You can control the splat configuration by changing the splat type and size on the module control panel.

Rendering splats can may take a long time. You can set and adjust an error tolerance to get a balance between rendering time and splat quality.

- To set the error tolerance, use the dial on the control panel. A low error value produces an accurate volume display, but the display takes longer to generate.

ReadLat<3> reads the colormap settings for the volume rendering of the water buoyancy data into *GenerateColormap*. The current setting for the colormap shows buoyant air, containing little water, in red and the saturated air in blue.

Once the data is in *Render*, you can analyze it in detail. **Figure 1−14** displays the reflectivity from the storm

Footnote: R.B. Wilhelmson, B.F. Jewett, C. Shaw, L.J. Wicker, M. Arrott, C.B. Bushell, M. Bajuk, J. Thingvold, and J. B. Yost, "A Study of the Evolution of a Numerically Modeled Severe Storm," International Journal of Supercomputer Applications, Volume 4, No. 2, Summer, 1990 pp. 20−36.

. The blue−white surface encloses the area containing most of the large drops and hail. Several light sources illuminate the surface. You can see the overhang, observed by weather radar in many severe storms, at mid−level where the reflectivity surface comes out toward the viewer.

Figure 1−14 Storm Data in *Render*

Above the storm, there is a region of many colors volumetrically rendered from the buoyancy, a quantity used in the equations for vertical velocity. It represents the instantaneous effects of temperature, moisture, and water mixed with ice on the velocity acceleration, and is large at and above the top of the storm.

A wavy appearance is evident, indicating the presence of strong gravity waves. Yellow/green indicates relatively high buoyancy, blue indicates relatively low buoyancy. The effect is roughly similar to throwing a rock into a pond, but here it is the storm growing into the upper atmosphere.

You can change the opacity values on the colormap to affect the look of the volume rendering (see Chapter 3, **"Creating Colormaps"**).

- You can lower the opacity of a range of colors to make them more transparent, and let you see other aspects more clearly.
- You can increase the opacity of a color to emphasize a feature.

1.7. Computational Fluid Dynamics

This example illustrates how you can use IRIS Explorer modules to study the dynamics of air flow over the nose of an aeroplane. The "cfd" map shows variations in air density surrounding the surface of the aeroplane.

The map generates an air flow field around the plane, extracts 2D slices of the volume, and colormaps them by density value, thus providing a means of examining different areas of the field.

The CFD group at Silicon Graphics created the plane from a model aeroplane by a digitized 3D scanning process, and the air flow data was generated from NASA's fluid dynamics program, *ARC3D*

Footnote: US Public Domain Software. For more information, see Pulliam, T. and Steger, J., "Implicit Finite−Difference Simulations of Three−Dimensional Compressible Flow," AIAA Journal, Vol 18, No. 2, Feb. 1980, pp. 159−167.

.

1.7.1. Opening the Cfd Map

To open the "cfd" map (see **Figure 1−17**), type:

```
explorer −map cfd.map
```

The map contains nine modules:

- *ReadLat* reads in the air density data file.
- *IsosurfaceLat* generates an isosurface from the air density data.
- *OrthoSlice* extracts the density near the plane's surface.
- *ProbeLat* creates non−orthogonal slices of the air density data.
- *TransformGen* sets the orientation for the *ProbeLat* slice.
- *PyrToGeom* converts the slice data from *ProbeLat* to geometry.
- *GenerateColormap* creates a colormap for the density data.
- *LatToGeom* converts the slice data from *OrthoSlice* to geometry.
- *Render* displays the isosurface and the slices.

1.7.2. Experimenting with the Cfd Map

OrthoSlice (see **Figure 1−15**) operates on the air density data to produce a curvilinear grid depicting the density field around the plane. The slice number is 1 (the minimum) and this index increases in a direction normal to the plane's surface. For $J = 1$, the grid actually lies along the surface of the plane.

Figure 1−15 *OrthoSlice* Control Panel

- You can change the slice number while holding the axis constant, and see the grid move out from the plane surface.

  As you vary the slice numbers, you get information about air densities at the plane surface and in its vicinity. High air densities occur at the nose when there is a massive deceleration of fluid. As the air flows away from the leading edge and accelerates over the wing, the density drops.
- You can also vary the axis. If you select the *K* axis, the slice is taken down the longitudinal axis of the wing and produces a cross−section of the wing.

ProbeLat and *IsosurfaceLat* complement each other. *IsosurfaceLat* generates an isosurface from the air density data. This tells you how all the air of a given density is distributed around the plane.

- You can change the threshold (the value at which the isosurface is calculated) by turning the dial on the control panel.

ProbeLat (see **Figure 1−16**) probes the curvilinear flow field along an arbitrary axis. The slice shows all density values in that region, colormapped by value.

Figure 1−16 *ProbeLat* Control Panel

- You can control the area of the slice by turning the Clip Size dial on the control panel.
- You can also change the form the probe takes, by selecting a type from the "Probe Type" option menu. In **Figure 1−17**, the selected probe type is a paddle.

Figure 1−17 Different Levels of Air Density

The axis along which the plane is cut is controlled by *TransformGen*. You can slice in any direction you choose by using the widgets on the *TransformGen* control panel to set a different axis. You can alter the orientation of the probe by using the mouse to move the object in *TransformGen*'s window.

GenerateColormap colormaps the grid produced by *OrthoSlice* according to its air density values. It is a scalar field, with one value at every point. The domain in *GenerateColormap* is set to a density minimum of about 0.7 and maximum of 1.1. This colormap is also used to color the probe surface according to air density values.

PyrToGeom and *LatToGeom* convert pyramid and lattice data respectively to geometry which can be displayed by *Render*.

Once the data is displayed in *Render* (see **Figure 1−18**), you can enhance the visualized data by using the *Render* menus. For example, click on the background of the *Render* window to bring up the *Render* pop−up menu. Select the "Draw Style" option, and then select "wireframe" from the cascading menu. To return all the surfaces to their original state, bring up the menu again and select the "as is" option.

Figure 1−18 The Rendered Flow Field

The grid, isosurface, and probe surface are separate objects, which can be separately treated in *Render*. For example, you could make the probe surface transparent, which would reveal the features behind it.