Schlumberger

Introduction to CPS-3

Lecture Notes

GeoFrame 4

June 12, 2002

Copyright Notice

Disclaimer

Use of this product is governed by the License Agreement. Schlumberger makes no warranties, express, implied, or statutory, with respect to the product described herein and disclaims without limitation any warranties of merchantability or fitness for a particular purpose. Schlumberger reserves the right to revise the information in this manual at any time without notice.

Trademark Information

GeoFrame™, CPS-3™ and certain other software applications mentioned in this material are trademarks of Schlumberger.

All other products and product names are trademarks or registered trademarks of their respective companies or organizations.

Contents

LECTURE for Topic 2
GullFaks Training Data - Inventory and Description

Overview

In this chapter, we'll make an inventory of data in the **GeoFrame GullFaks** project which is used by this class.

The GullFaks field is well-documented and information on this field is available via the Schlumberger intranet.

This course uses only some of the data available in the project and includes

- Well top locations, bottom locations, and well paths
- Geologic markers
- 3D seismic interpretation, both horizons and fault segments
- Fault boundaries
- Layer-based net, gross, porosity and saturation averages

Other data used in the class includes

- 2D seismic line data from ASCII files
- Lease block polygon data from ASCII files

Velocity functions have been provided for the seismic cube, so that both seismic and well data are available in time and depth. In this training course, we will use **depth** data.

Table 2.1- Seismic Horizon Data and Geological Marker Names

Geological Marker in GeoFrame	Seismic Horizon in IESX	CPS Horizon Name	Fault Poly	Interpretation Density / Description
BUNNKRITT	BUNNKRITT	Bunnkritt	no	unfaulted/asap(1X1)
TARBERT	TARBERT	Tarbert	yes	sparse (20X20)
NESS	NESS	Ness	yes	dense/2d3d + asap (1.x1)
RANNOCH	RANNOCH	Rannoch	no	sparse (20x20)
DRAKE	DRAKE	Drake	no	dense (1x1)

Naming Conventions

Be aware that since projects can be shared by several persons in different disciplines, different names for different versions of interpretation, names of **GeoFrame** containers, marker names, and the like must be coordinated among those working in the project.

Volumetrics Notes

In this class, we will compute volume between the Tarbert and the Ness horizons. Because of large erosion zones in the Tarbert caused by the Bunkritt unconformity, the top of the reservoir must be a merging of the Tarbert and the unconformity.

Most of the data for this course will come from the GeoFrame data base. There will be several methods for gaining access to it from CPS-3. In addition, some of the data will come from outside of the project. Regardless of its origin, the next section provides an inventory of the data sets which will be used in CPS-3 for this course.

Data Inventory for GullFaks CPS-3 Training

Location Data

2D Locations

•mm_2d_gullfaks_shtpt.dat (ASCII)

3D Locations

- mm_3d_85acip_survey
- mm_3d_g1_survey
- mm_3d_offset_survey

Well Top Locations

- mm_Well_Locations_wtloc

Well Bottom Locations

- mm_Well_Locations_wbloc

Well Paths

- mm_Boreholes_Depth_wpath

Interpretation

Horizons/Fault Polygon Sets (Time)

BUNKRITT

- BUNKRITT_time_intrp

TARBERT

- TARBERT_time_intrp
- TARBERT_time_intrp_fpolys

NESS

- NESS_time_intrp
- NESS_time_intrp_fpolys

RANNOCH

- RANNOCH_time_intrp
- RANNOCH_time_intrp_fpolys

DRAKE

- DRAKE_time_intrp
- DRAKE_time_intrp_fpolys

Horizons/Fault Polygon Sets (Depth)

BUNKRITT

- mm_BUNNKRITT-1_BU-285_Depth_intrp

TARBERT

- mm_TARBERT_smooth_Depth_intrp
- mm_Tarbert

NESS

- mm_NESS_smooth_Depth_intrp
- mm_Ness

RANNOCH

- mm_RANNOCH_smooth_Depth_intrp

DRAKE

- mm_DRAKE_smooth_Depth_intrp

Fault Segments (Time)

- F2...etc...
- F2a
- F3
- F4
- F5
- F6
- F6a
- F7
- F7a
- F8
- F9
- F11
- F12
- F13
- F14

- F15
- F15a
- F16
- F17
- F18
- F19
- F20
- F21

Fault Segments (Depth)

- mm_F2_Depth_fsegs
- mm_F2a_Depth_fsegs
- mm_F3_Depth_fsegs
- mm_F4_Depth_fsegs
- mm_F5_Depth_fsegs
- mm_F6_Depth_fsegs
- mm_F6a_Depth_fsegs
- mm_F7_Depth_fsegs
- mm_F7a_Depth_fsegs
- mm_F14_Depth_fsegs
- mm_F15_Depth_fsegs
- mm_F15a_Depth_fsegs
- mm_F16_Depth_fsegs
- mm_F17_Depth_fsegs
- mm_F18_Depth_fsegs
- mm_F20_Depth_fsegs
- mm_F21_Depth_fsegs

Well Marker Sets (Depth)

- mm_BUNNKRITT_Depth_wmrkr
- mm_TARBERT_Depth_wmrkr
- mm_NESS_Depth_wmrkr
- mm_RANNOCH_Depth_wmrkr
- mm_DRAKE_Depth_wmrkr

Well Marker Sets (Time)

- BUNNKRITT_Time_wmrkr
- TARBERT_Time_wmrkr
- NESS_Time_wmrkr
- RANNOCH_Time_wmrkr

Interval Definitions

Zone Versions

- (none at present)

Zones

- (none at present)

Properties

Net-to-Gross Thickness (Data set)

- mm_TARBERT_NESS_net-gross

Net Pay Porosity (Data set)

- mm_TARBERT_NESS_Porosity

Net Pay Water Saturation (Data set)

- mm_TARBERT_NESS_WSat

Lease Information

Lease polygons

- mm_north_leases.ply (ascii)
- mm_North_Leases

Project Coordinate System Information

Datum: European 1950, Norway and Finland

Ellipsoid: International 1924

Projection: UTM, Zone 31, CM = 3.0

Hemisphere: Northern

Project Units

Metric

Project Location

North Sea

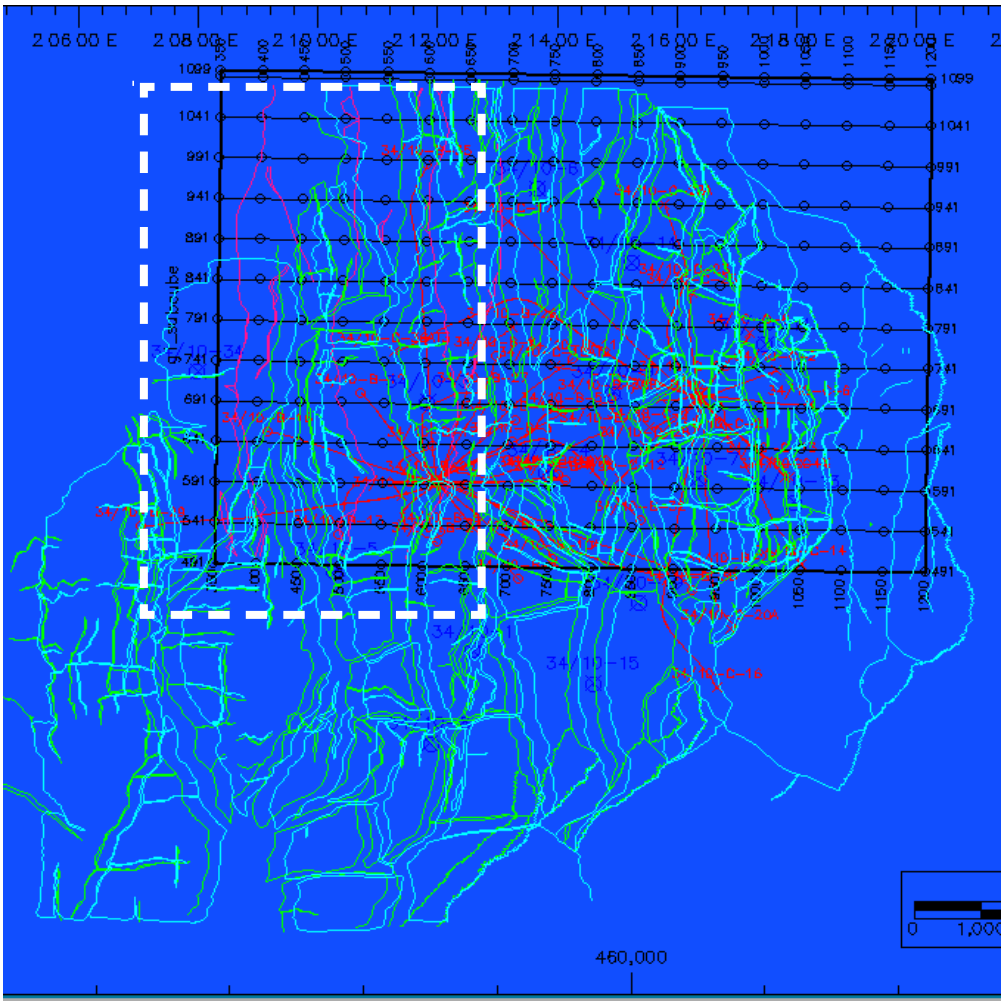


Figure 2.1 GullFaks Fault Patterns and 3D survey

Above, you can see **fault patterns** over the area as well as the limits of the **3D survey** in black. Two platforms are the source of many **well trajectories** shown in red. The white dotted rectangle shows the approximate interpretation coverage of the highest of the horizons - the **Tarbert**. Subsequently lower horizons will cover more and more area towards the East. Well coverage for the higher horizons such as the Tarbert are restricted to the lower SE quadrant of the dotted rectangle

The interpretation for the upper unconformity (Bunnkritt) covers all of the 3D rectangular area except for a small portion in the NW corner.

The figure below shows a smaller area which is focused on the extent of the well paths. The Bunkritt interpretation is in white, and covers just about all the 3D survey. The Tarbert interpretation is shown in grey and covers only the Western half. The Eastern platform has more and better distributed wells, but does not overlap with the best seismic.

.

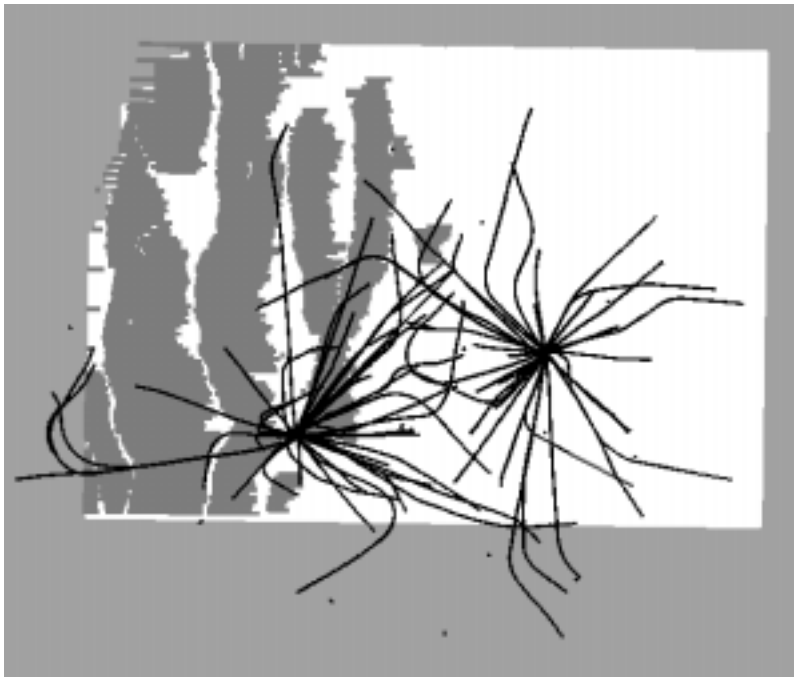
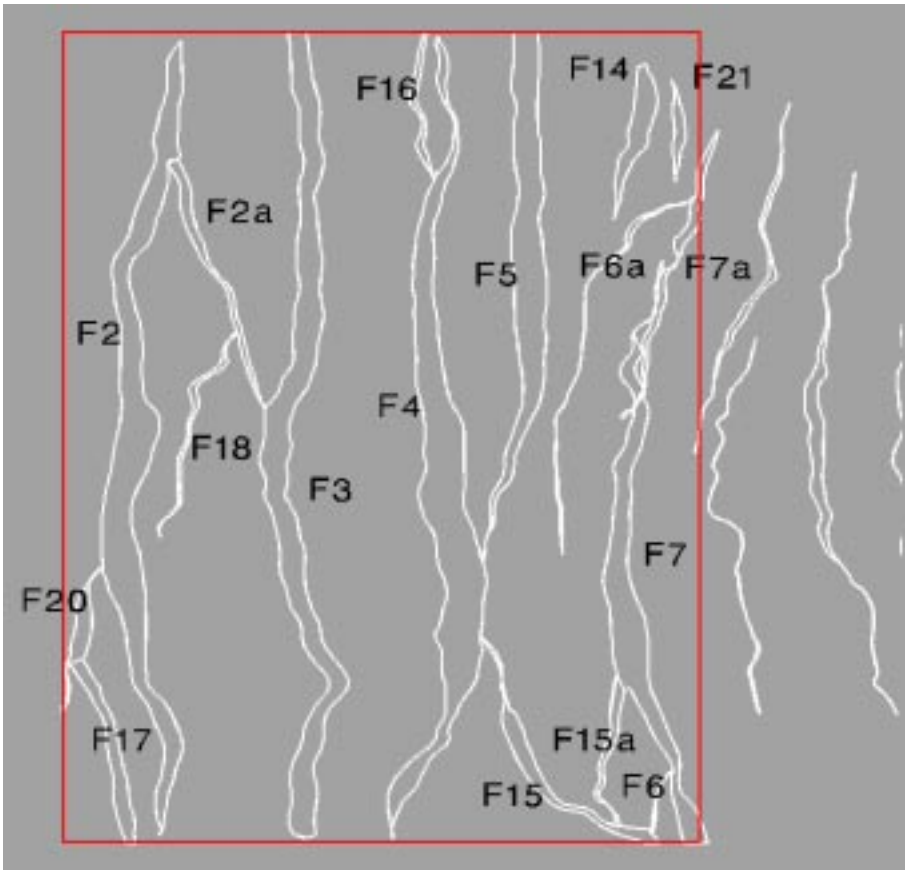


Figure 2.2 Well paths, Bunkritt interpretation, and Tarbert interpretation

Selected Fault Patterns

For purposes of the CPS-3 training, we will use only the larger faults as shown below.



Reservoir Geometry

The area of the GullFaks field which we will be mapping consists of a series of tilted fault blocks with the upper horizon of the group, the Tarbert, containing erosion zones where no interpretation exists above the Bunnkritt unconformity.

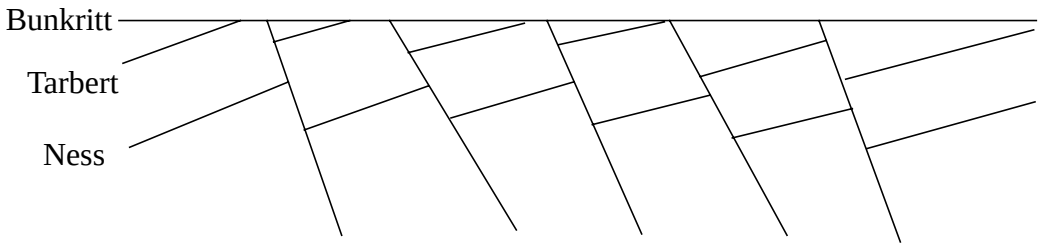


Figure 2.3 Reservoir profile displaying the stratigraphic relationship of the horizons

LECTURE for Topic 4
CPS-3 Menu Organization and Capabilities Overview

Overview

This chapter will give you a brief overview of all functionality provided in **CPS-3**, and how it is organized. An overview of the menu hierarchy of the **CPS-3 Main Module** is also provided.

The full capability of **CPS-3** is divided among the following independent modules which are described in this chapter:

- **Main Module** - modeling and mapping tools
- **Map Editor** - simple graphic editing of Map sets
- **Model Editor** - comprehensive grid and data editing
- **Color Palette Editor** - customize/create palettes

In addition, the following data management tools are available from the **Main Module**:

- **Control Point Editor** - interactive spreadsheet editor for **Data** sets
- **Subset Reorganizer** - a fault management tool
- **Map Layer Manager** - a tool for reorganizing graphic layers of a map

In this chapter we also present a convenient "**How To...**" **Matrix** which cross-references common mapping operations with the menu navigation instructions for how to get there. This cross-reference is at the end of the chapter.

CPS-3 Main Module

The **Main Module** is where most of the traditional modeling and mapping tools are located, and where you will probably spend most of your time in **CPS-3**. In the previous chapter, we learned how to invoke the **Main Module** from either a stand-alone or a **GeoFrame** 3.0-or-later installation. In this chapter, we will learn many things about the environment of the **Main Module** - its conventions, resources, organization, and concepts.

Main Module Dialog Box

The figure on the opposite page displays the current **Main Module** graphic dialog box. Note the following features emphasized in the figure.

- **Pull-Down Menus**:

  All **CPS-3** functionality can be accessed through these pull-down cascading menus.

- **Icons**:

  For more convenient access to commonly used functions, these icons can be a shortcut instead of traversing the menu tree.

- **Icon Descriptions**:

  As the cursor is moved over the icons, a description of each is presented here.

- **Scroll Bars**:

  When zoomed in, use these slider bars for panning across the display.

- **Display Environment Box**:

  Every display environment you define in **CPS-3** contains the definition of an x, y, z box called the **Volume of Interest** (**VOI**). What is shown here is simply the x, y portion of the box. The system attempts to maximize the amount of canvas space allocated to your currently active x, y box.

- **Canvas**:

  This is simply the total potential screen area (or paper plot area, when plotting) where the x, y box can be located. The lower left corner of the canvas represents the origin(0,0) for the internal graphic coordinate system in inches or centimeters.

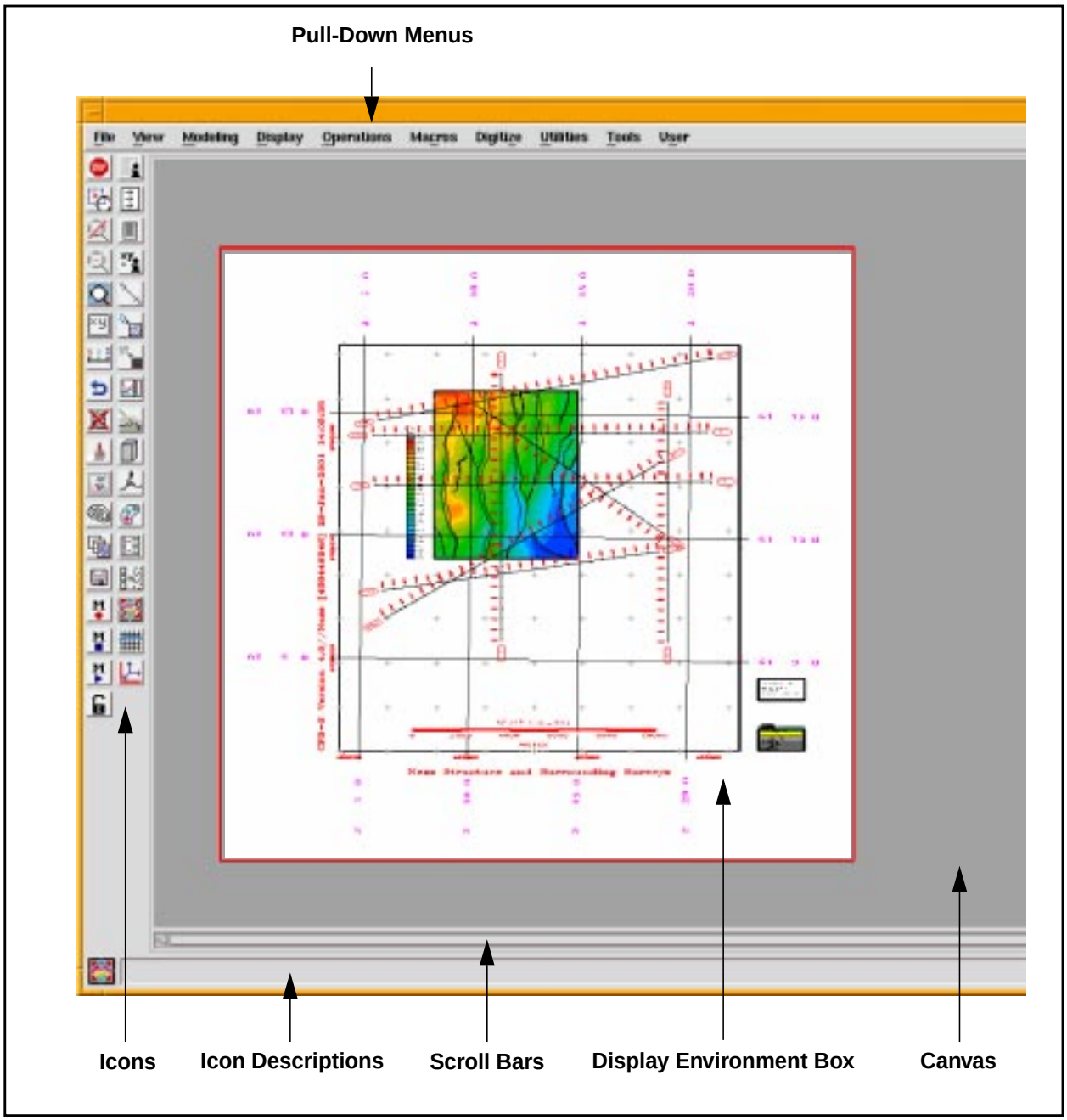


Figure 4.1 CPS-3 Main Module

X,Y Tracker Display

When moving the graphic cursor during zooming, the x,y location of the cursor is echoed at the bottom of the **Main Module** dialog box as shown in the following figure. The position is echoed in both **engineering** units and **plotter** (viewport) units.

The x,y tracking also occurs during screen digitizing initiated from the **Digitize** pull-down menu.

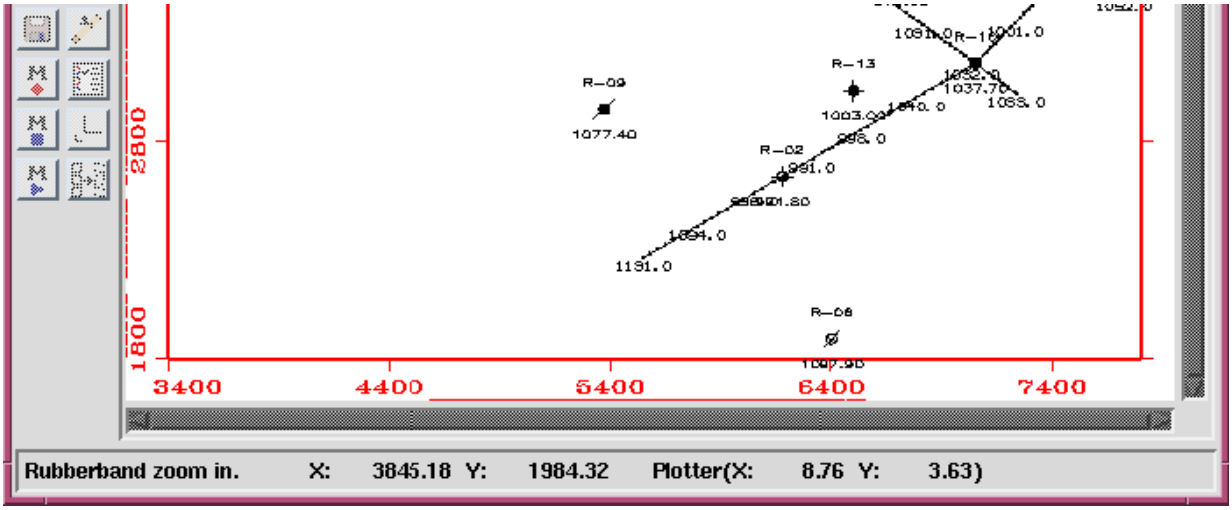


Figure 4.2 Main Module displaying x,y locations of graphic cursor

For convenience, the tracker may also be invoked at any time with the tracker icon:

Figure 4.3 Display x and y position at cursor icon (aka Tracker icon)

Measuring Tool

The measure distance icon in the following figure provides facilities for measuring distances and angles on the graphic display.

Figure 4.4 Measure distance icon

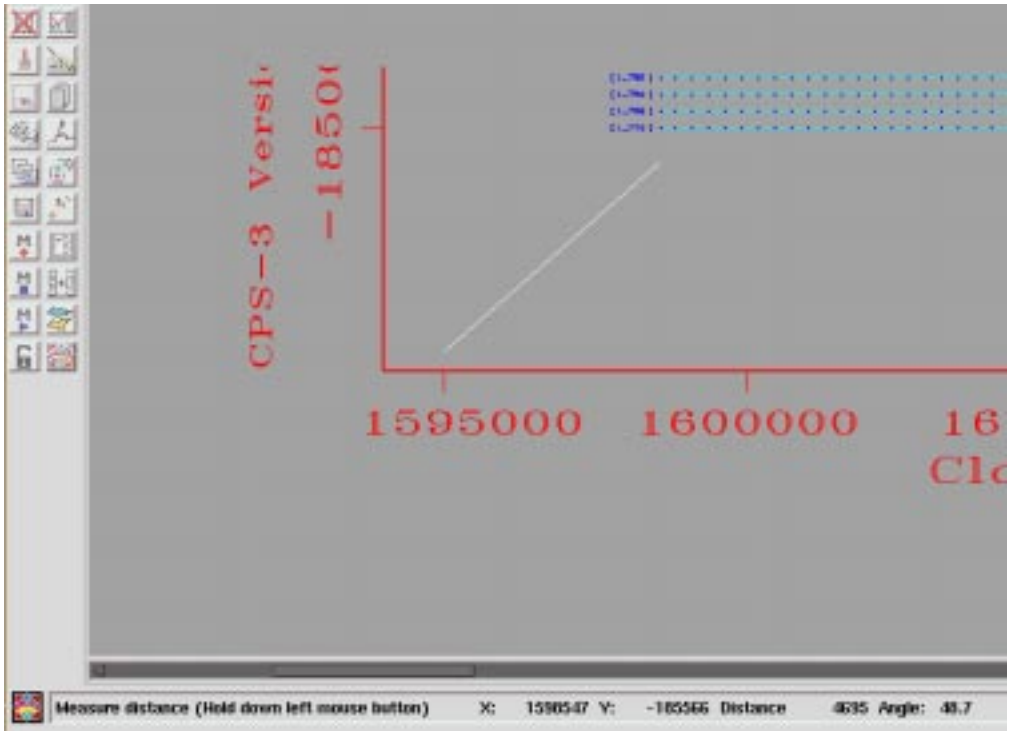


Figure 4.5 Determining distance using the Measure distance icon

Main Module Status Window

The figure on the next page provides an example of the **Main Module** status window, on which the following components should be noted:

- **Currently open Data set**:

  This table provides a list of the currently open sets by set type (**Data**, **Fault**, **Polygon**, **Surface**, and **Map**). This particular example shows one active **Data** set. Note that the table scrolls downward and has room for up to seven active sets per type.

- **Currently open Surfaces**:

  In this example, there is one open **Surface** set.

- **Native Command Entry**:

  By clicking the cursor in this box, you can type native commands here, as an alternative to menu or icon selection.

- **Online status report dialog**:

  This window displays a real-time status report of all operations you perform during the **CPS-3** session. This information is also written to a file called

  <username>.rep

  where **<username>** is your **login id**. This file is overwritten each time you start another **CPS-3** session.

- **Swap Screens Icon**:

  The **CPS-3 Status Information** window can be instantly moved from the current screen to the opposite screen by simply toggling the **Swap Screens** icon in the upper right of the display.

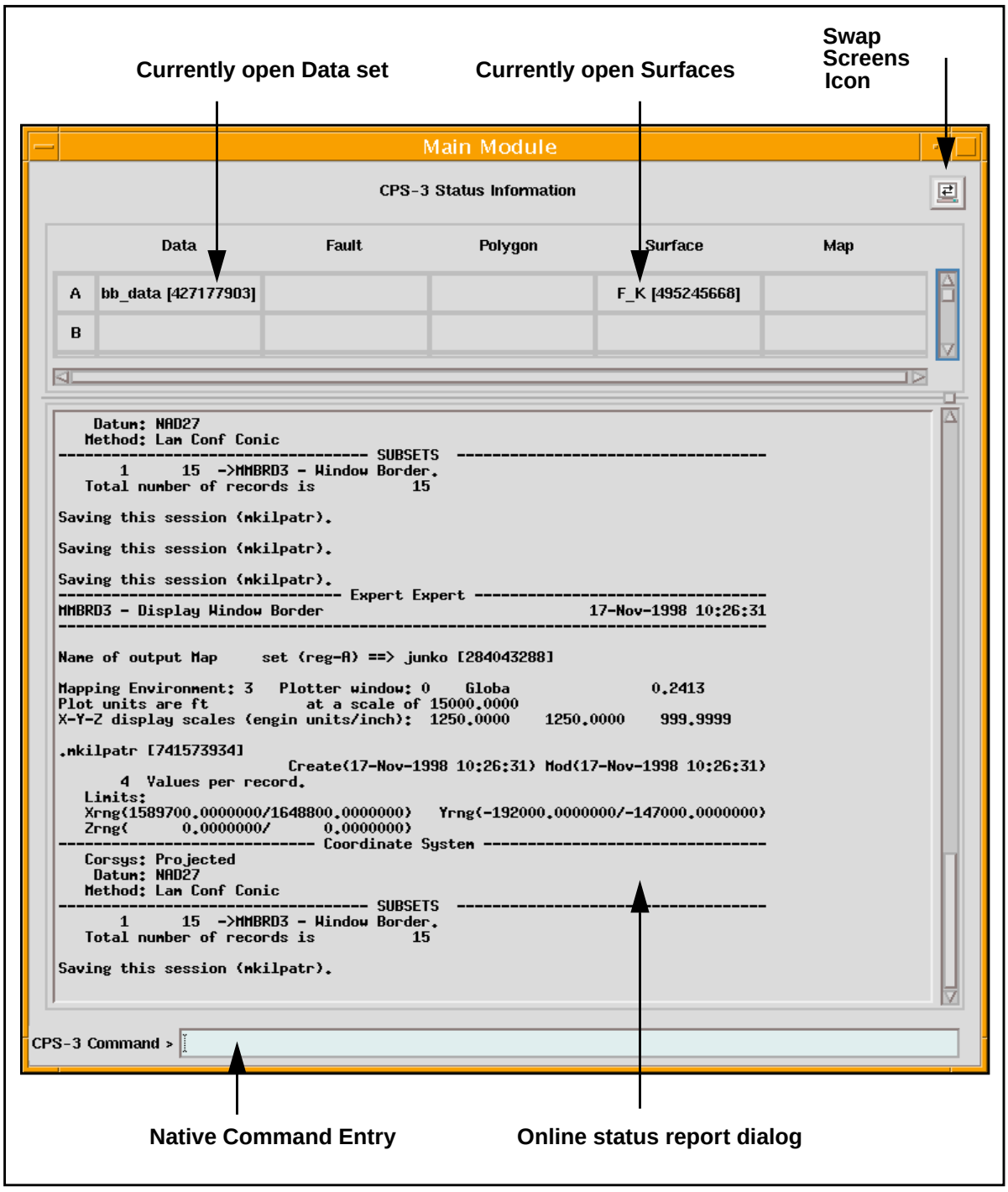


Figure 4.6 Main Module displaying CPS-3 Status Information window

UNIX Environment

In the **Main Module** go to **User > Show Environment** to bring up the **CPS-3 Environment** window. This window can remain visible while you perform other mapping operations. It's purpose is to provide you with information about where **CPS-3** is installed, and where the configurable resource files are located. It also serves to remind you of the path to your open project, so that copying of configuration files into your project area from an xterm is simplified.

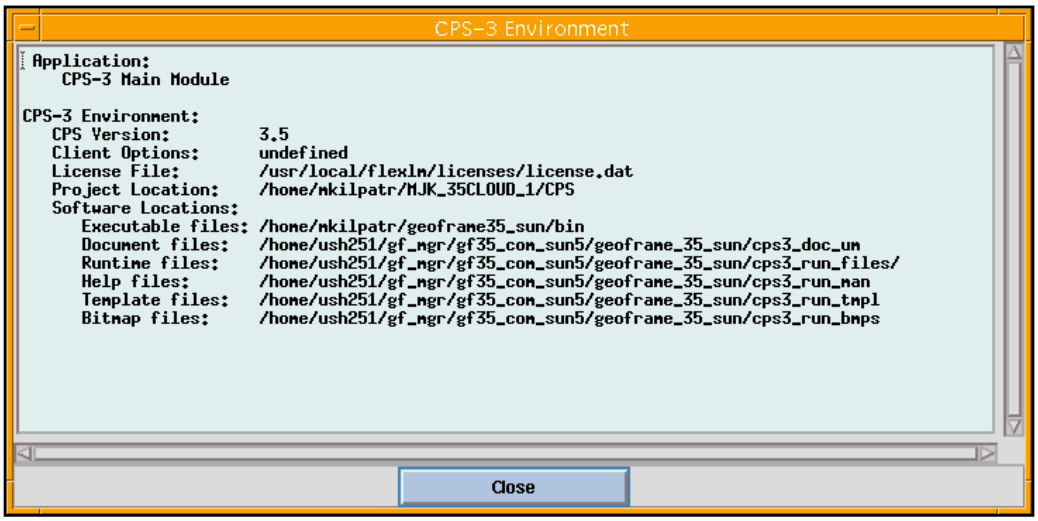


Figure 4.7 CPS-3 Environment window

CPS-3 Map Editor

The **Map Editor** in the following figure is a graphic editing tool used after creating the **Map Set** in the **CPS-3 Main Module**. Users can edit attributes such as colors and fonts. Overposting can be cleaned up interactively. Symbols and text can be added. Composite maps can also be created in the **Map Editor**.

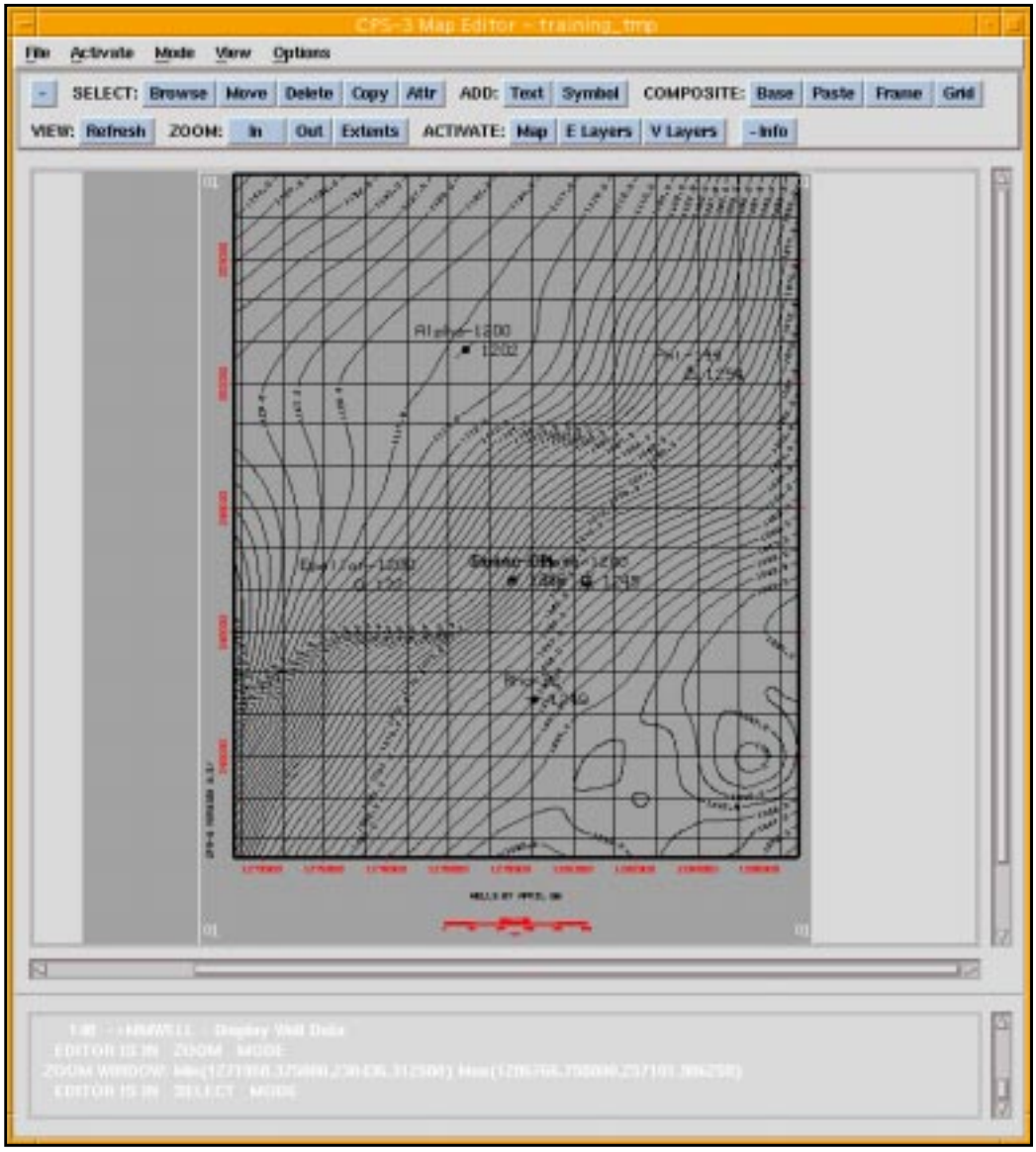


Figure 4.8 CPS-3 Map Editor

CPS-3 Model Editor

The **Model Editor** in the figure below is for editing surfaces, points, faults, polygons, and features. Users can make changes to the model (surface) by moving, deleting, or redrawing contours, modifying data (points), modifying fault data, using polygons for constraints, adding feature data, or by modifying the actual grid nodes.

After each edit, the model is regridded and saved upon completion of the editing. The final model (grid) is then brought back into the **CPS-3 Main Module**.

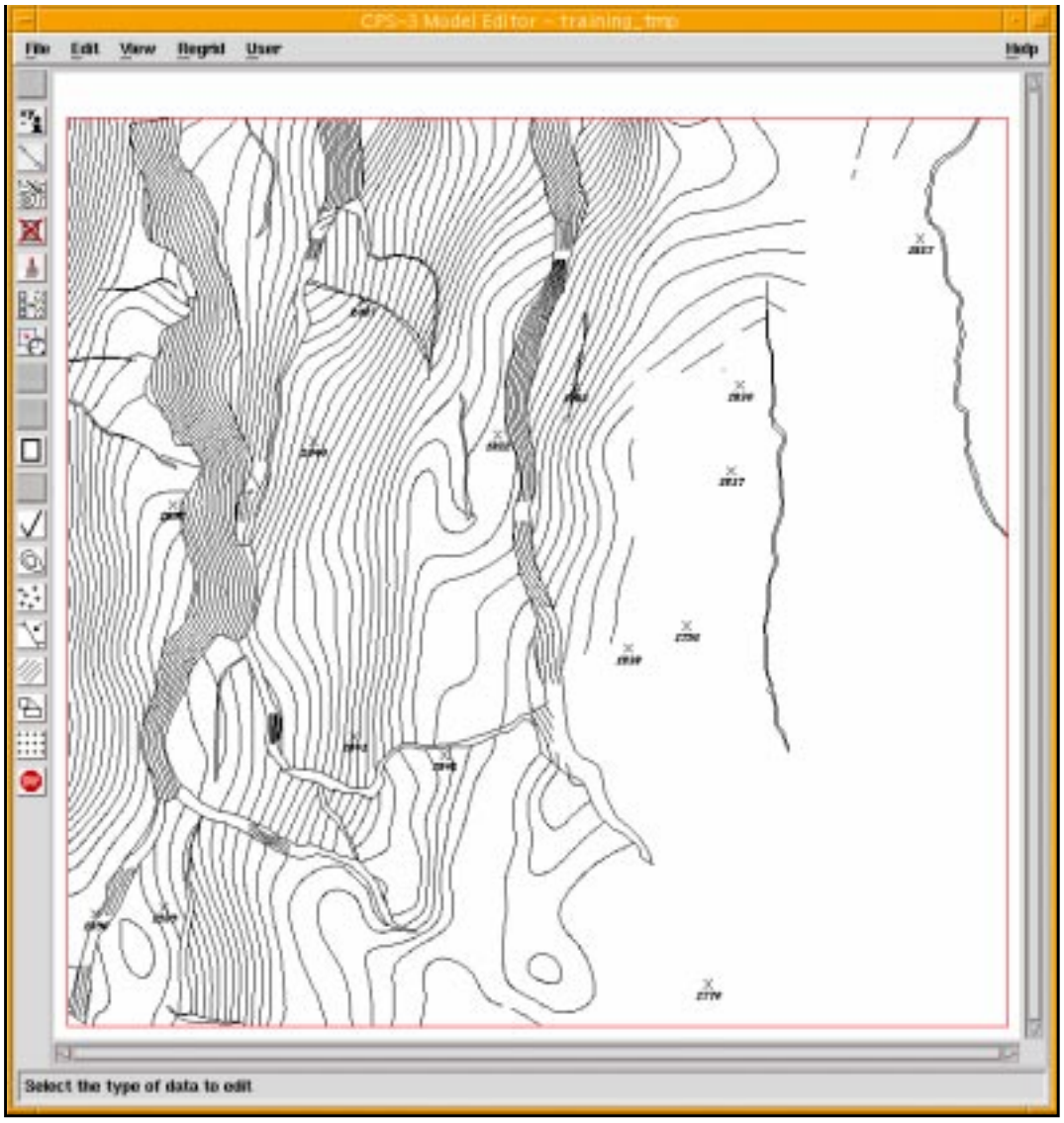


Figure 4.9 CPS-3 Model Editor

CPS-3 Color Palette Editor

The **Color Palette Editor** in the following figure allows users to define their own color palettes to use when drawing color shaded contour maps. Colors can be associated with a specific z-value. Each individual color can be set, or the system will interpolate between two colors set by the user. The color palettes are then saved to a color palette name in the user's project directory. These palettes can be accessed in the **Main Module** when specifying parameters for displaying color shaded contours.

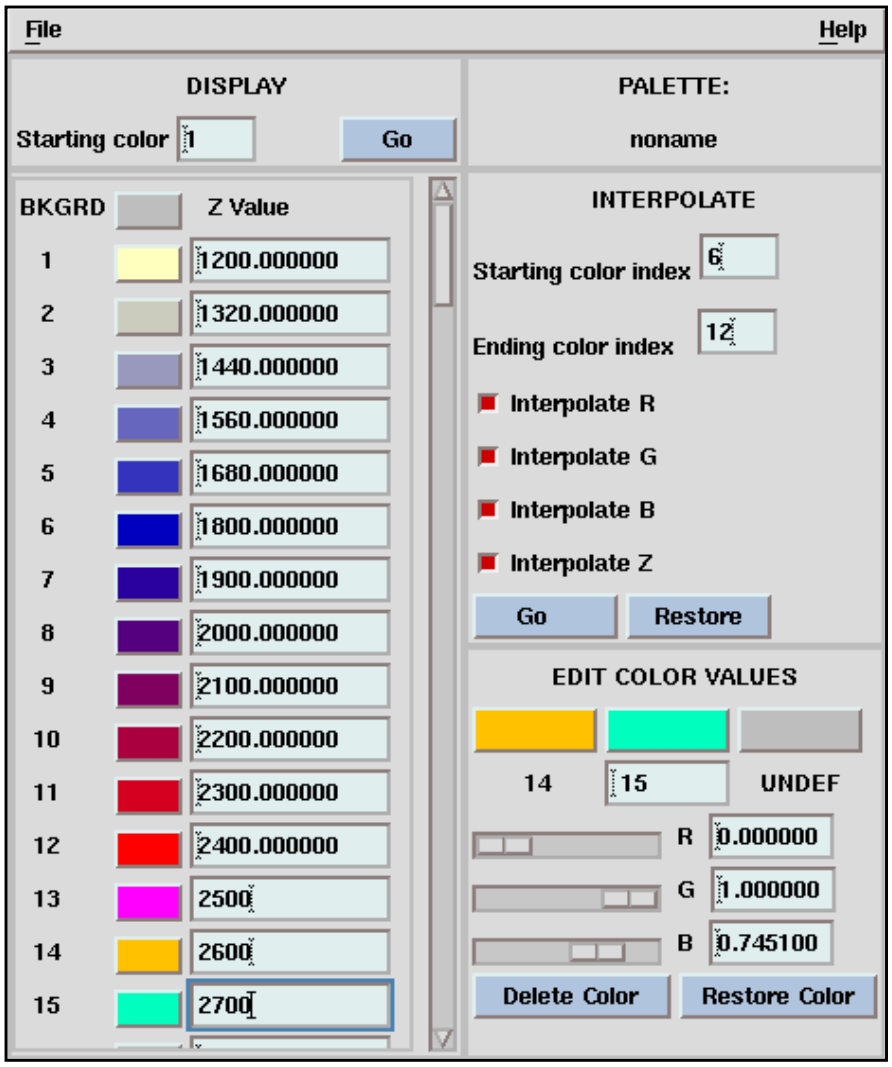


Figure 4.10 CPS-3 Color Palette Editor

CPS-3 Control Point Data Editor

In the **CPS-3 Main Module**, go to **Utilities** > **Sets** > **Edit Data Set** to access the spreadsheet-formatted data editor which is shown below. This feature is very useful for removing z-fields, changing subset names, and editing graphic symbology such as symbol codes, or even well names in data sets.

The data set shown below is a well marker data set, which was loaded as an **Extended** data set with three z-fields, a well name, and symbol code - all of which are available for editing.

The **Help** text for this dialog is very useful.

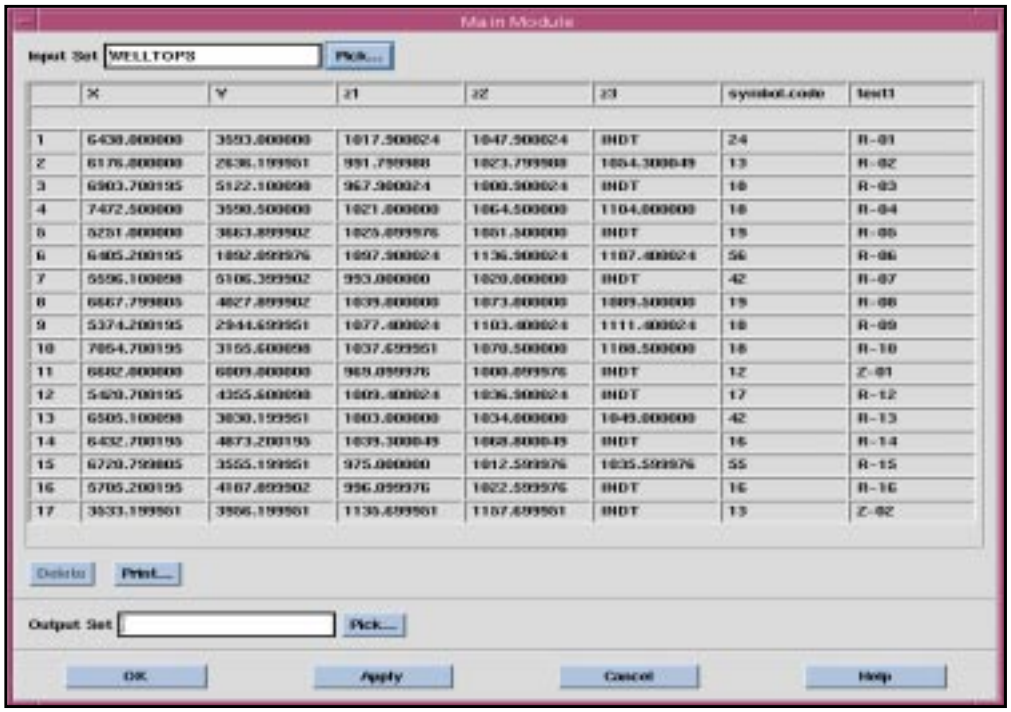


Figure 4.11 Main Module displaying spreadsheet-formatted data editor

CPS-3 Set-Subset Reorganizer

This utility performs data management functions among various **CPS-3** set types, reordering information from one domain into useful information in another. For example, assume that you mapped three surfaces, each with two major fault polygons having z-values attached. The three fault polygons migrate in x and y as they move downdip in each of the horizons. If you now wanted to create gridded surfaces for the two faults, it would be impossible with the fault sets because the x,y,z points are organized by horizon (Figure 4.12). What is needed is a resorting of these x,y,z points by fault, rather than by horizon (Figure 4.13). Individual files in these figures are indicated by separate fill patterns.

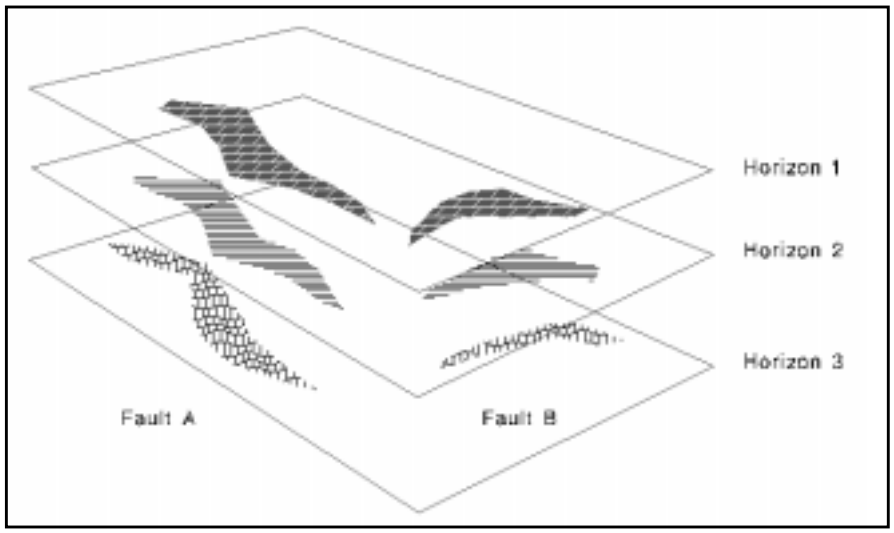


Figure 4.12 Sorted x, y, z points according to horizon

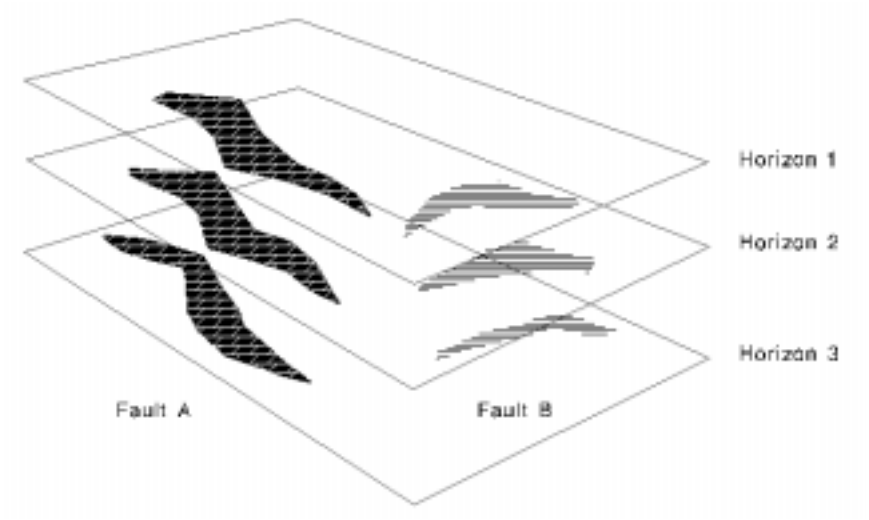


Figure 4.13 Sorted x, y, z points according to fault

The **Set/Subset Reorganizer** will perform this reordering for you. As the different shades of gray in the first diagram above indicate, x,y,z fault traces are grouped in **Fault** sets, one per horizon. In the **Set/Subset Reorganizer** dialog box, select the **Fault** sets for all horizons which contain valid x,y,z points to use in the gridding of the fault surfaces. From each of those selected fault sets, you can choose any or all of its subsets (individual fault traces for a single fault) which you want to be included in the output data sets.

In this example, we have asked for the output to be written to a **Data** set as shown below. There will be as many output **Data** sets as there are unique subset names (fault names) in the selected **Fault** sets. Each of the output **Data** sets can then be gridded to obtain a model of the fault surface.

The **Help** text for the **Set/Select Reorganizer** is very informative.

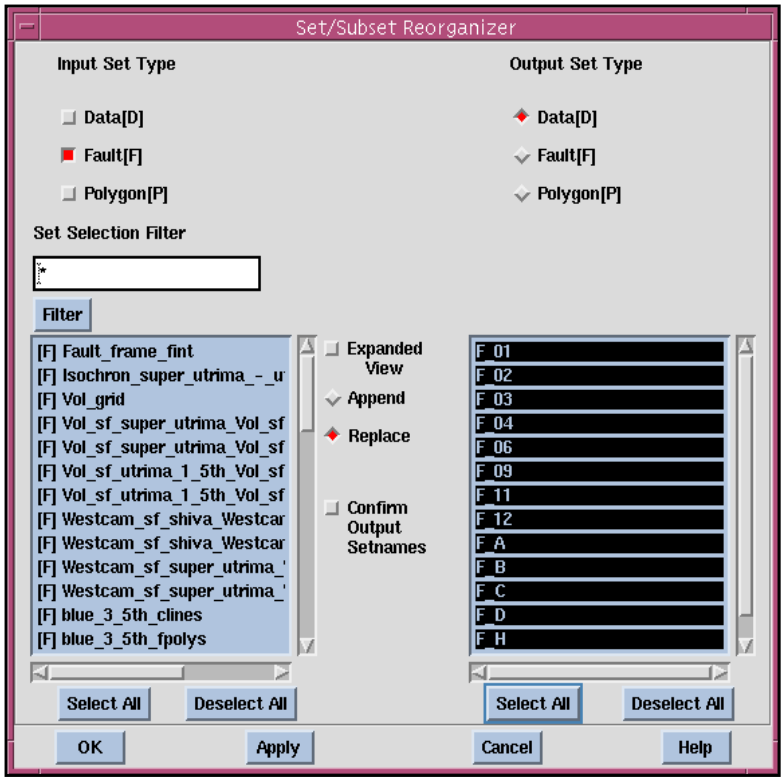


Figure 4.14 Set/Subset Reorganizer

CPS-3 Map Layer Manager

The **Map Layer Manager** (Figure 4.15) is an extremely useful tool for reordering the subsets stored in a map set. It is often the case that color shaded contours (Figure 4.16) obliterate other map elements simply because of the order in which they were displayed on the screen. The **Map Layer Manager** will allow you to reposition map layers (subsets) to optimize your graphic output without having to regenerate the graphics on the screen.

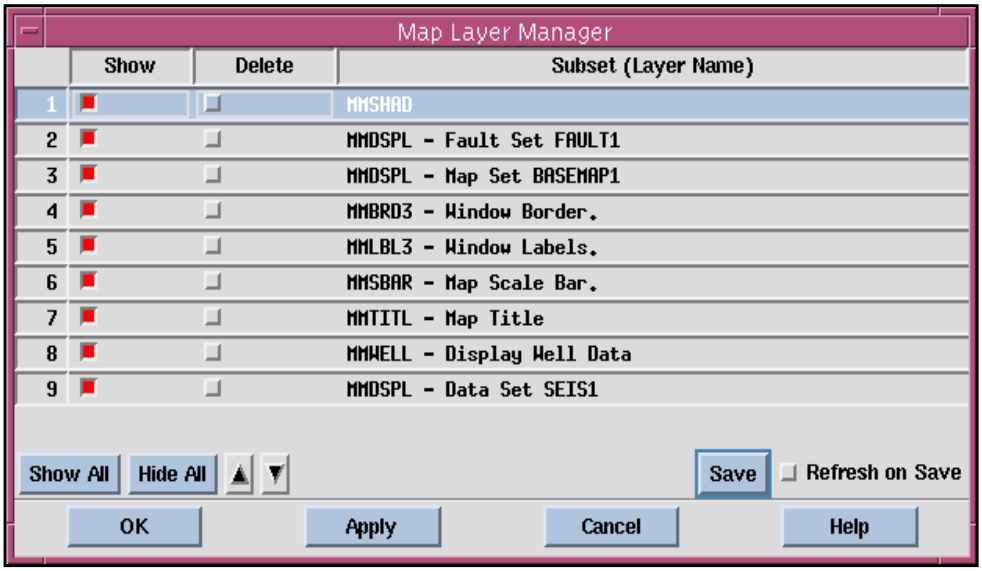


Figure 4.15 Map Layer Manager

In this example, the picture below and on your screen was generated with separate graphic commands; then the **Map Layer Manager** (Figure 4.15) was invoked with the **Manipulate current map layers** icon (Figure 4.17).

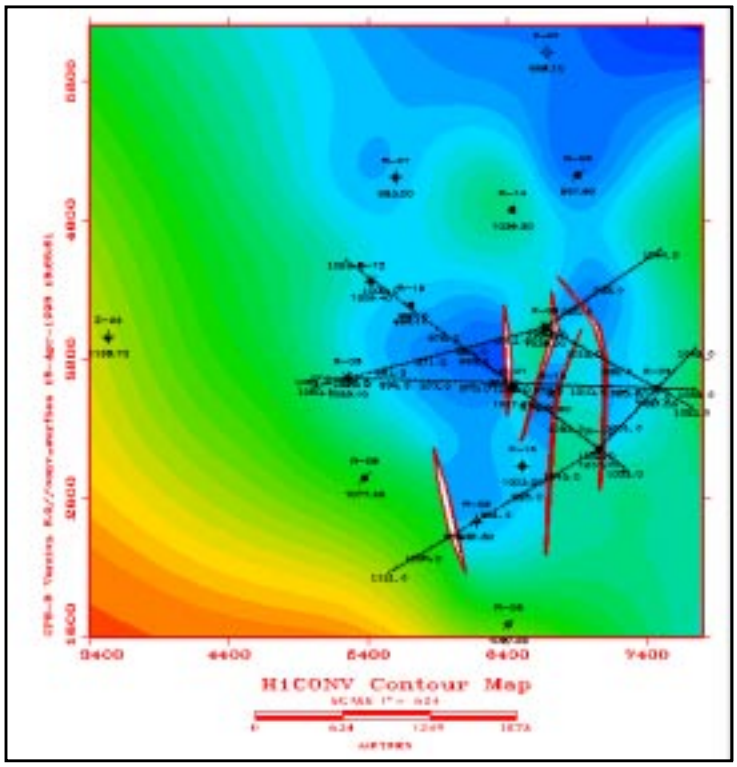


Figure 4.16 Example of colored contour map

Figure 4.17 Manipulate current map layers icon

Each layer on the screen is shown as a line in the dialog table.

To reorder the layers of an existing map, simply clear the screen, display the map, then invoke the **Map Layer Manager**.

Layers can be temporarily turned ***on*** or ***off***. Layers may also be deleted. As with other **CPS-3** dialogs, the **Help** text for this facility is very useful.

Menu Navigation by Topic

GENERAL

Set expert level - UTIL/ SYSTEM/ SET-EXPERT-LEVEL

Set control switches - UTIL/ SYSTEM/ SET-TOGGLE-SWITCHES

Find out the path to the CPS dsl - USER/ SHOW-ENVIRONMENT

See statistics on a CPS-3 set - UTIL/ SETS/ VIEW-CONTENTS-...

Look at the CPS-3 on-line User's Manual - TOOLS/ USER-MANUAL

DISPLAY

Create display environment -

- Sixth icon from top on left, then click CREATE on the Display row.

Display borders, labels, North arrow, titles, etc. - DISPLAY /BASEMAP

Display contours and color shaded maps - DISPLAY/ CONTOURS

Display cross-sections - DISPLAY/ 2D-XSECTION

Display color bar - DISPLAY/ COLOR-SHADING-PALETTE

Display ortho contours - DISPLAY/ CONTOUR/Orthocontours

Erase/delete/reorganize graphic layers - DISPLAY/ MAP-LAYERS

Save display as map set -

- Fifth icon from bottom on left

Zoom in- VIEW/ ZOOM-IN

Zoom out - VIEW/ ZOOM-OUT

Erase the screen -

- Big red X icon

Erase last graphic layer

- Blue back-arrow icon above big red X icon

Set graphic margins - DISPLAY/DISPLAY_FUNCTIONS/ SET_GENERAL_DISPLAY_PARAMETERS/

GRIDDING/MODELING

Create a modeling environment -

- Sixth icon from top on left/ then click CREATE on the Modeling row

Create a grid from data - MODELING/ SINGLE-SURFACE

Use faults during gridding -

- MODELING/ SINGLE-SURFACE/ Make sure Fault is clicked on and selected

Select the gridding algorithm -

- MODELING/ SINGLE-SURFACE/ Click on the Algorithm box

Conformal Gridding - MODELING/CONFORMAL_SURFACE

SINGLE GRID MODIFICATION

Refine a grid - OPERATIONS/ GRID/ REFINE

Smooth a grid - OPERATIONS/ GRID/ SMOOTH

Differentiate a grid - OPERATIONS/ GRID/ DIFFERENTIATE

Blank a grid - OPERATIONS/ GRID/ BLANK

Clip a grid -OPERATIONS/ GRID/ LOGICAL/ Use 1ST or 3rd Operation

Perform grid arithmetic - OPERATIONS/ GRID/ SINGLE-GRID

Tie a grid to data - OPERATIONS/GRID/TIE_GRID_TO_DATA

Chance a grid lattice - OPERATIONS/GRID/MODIFY_GRID_LATTICE

Extract a grid (Peek) - OPERATIONS/GRID/EXTRACT_GRID

Insert a grid (Poke) - OPERATIONS/GRID/INSERT_GRID

MULTIPLE GRID ARITHMETIC/LOGIC

Subtract two grids to create a thickness grid -

- OPERATIONS/ GRID/ MULTIPLE-GRIDS

Merge two grids -]

- OPERATIONS/ GRID/ Under MULTI-GRIDS, use Operations 1 - 4

MACROS

Run a macro - MACROS/ EXECUTE

Create a macro - MACROS/ CREATE

Edit a macro - Create a macro, then user the text editor of your choice

DIGITIZE

Digitize data, faults, polygons, text - DIGITIZE as needed

DATA POINT ARITHMETIC

Compute values from a grid at arbitrary (well) locations -

- OPERATIONS/ CONTROL POINTS/Interpolate from.....,

Perform arithmetic on control point z-fields -

OPERATIONS/ C.P./CONTROL-POINT-MATH

SET MANAGEMENT/SET MANIPULATION

Rename a z-field - UTILITIES/ SYSTEM/ MANAGE Z-FIELD NAMES

Delete a set - UTILITIES/ SETS/ DELETE

Copy a set - UTILITIES/ SETS/ COPY

Rename a set - UTILITIES/ SETS/ RENAME

Unlock a set - UTILITIES/ SETS/ UNLOCK

Edit a data set - UTILITIES/ SETS/ EDIT DATA SET

View all subsets of a set -
UTILITIES/SETS/VIEW_CONTENTS_&_STATISTICS/ LIST_SUBSETS

Print all values in a set -
UTILITIES/SETS/VIEW_CONTENTS_&_STATISTICS/ LIST_CONTENTS

OTHER CPS-3 APPLICATIONS

Run the Model Editor - TOOLS/MODEL-EDITOR

Run the Map Editor TOOLS/MAP-EDITOR

Run the Color Palette Editor TOOLS/ COLOR-PALETTE-EDITOR

Run SurfViz - TOOLS/ SURFVIZ

Run the GeoFrame Link - TOOLS/ GEOFRAME-LINK

Run the IESX/CPS Link - in Visualization Catalog under CPS-3

Run the Charisma/CPS Link - from Charisma menu

DATA IMPORT/EXPORT

Import/export ascii files - FILE/ IMPORT and EXPORT

Icon Definitions

Stop current process

Zoom in

Reveal all graphics

Unzoom to last zoom

DisplayEnv from Zoom

Select/edit environment

Set map scale

Undo last graphic display

Erase the screen

Refresh display

Basemap menu

Contour

Map layers

Save display as mapset

Record a macro

Stop recording macro

Execute a macro

Unlock a set

View set statistics

Subset utilities

List/Manage sets

Get x,y coordinates

Measure distance/angle

Quick map

Single surface gridding

2D Profiles

Borehole intersections

Volumetrics

Model editor

Color palette editor

Customize icon bar

Hide icons

GeoFrame Link

GF grid data manager

GF grid library data manager

LECTURE A for Topic 5
Current Integration Status of CPS-3 in GeoFrame

Overview

This chapter reviews the nature and extent of data integration for **CPS-3** within the **GeoFrame** data base.

In GF4.0, CPS-3 still maintains its own local data store, that subdirectory known as the CPS-3 DSL, where all CPS-3 binary sets are stored. However, much of the data in GeoFrame is visible to CPS-3, either directly, or via the GeoFrame data link, GFLink.

CPS Local Data Store (dsl) and GeoFrame Storage

Historically, **CPS-3** has used an internal data management system called the **Storage Manager**. This internal data store will eventually be replaced by the **GeoFrame Oracle** data base, but in **GeoFrame**, the **CPS-3** internal storage facility is still in place. Even so, there are some **CPS-3** sets which may, at the user's option, be stored in the **GeoFrame** data base at the time of their creation. In particular,

- **Data** sets may be stored in **GeoFrame** as scatter sets, however, subset organization is lost.
- **Fault** sets may be stored in **GeoFrame** with subset (fault name) organization maintained.
- **Surfaces** sets may be stored in **GeoFrame** with the fault set association maintained.

Controlling Where Sets are Stored or Retrieved

- **During Set Creation**:

  At the time any **CPS-3** set is created, the menus give you the choice of storing the data in either the **CPS** local data store or in **GeoFrame**.

- **During Set Selection**:

  At the time any existing **CPS-3** set is being selected**,** the menus show you the current storage location of all available sets (**CPS** local data store or **GeoFrame**), and allow you to select from either location.

GeoFrame data items are synonymous with **CPS-3** sets, but data items stored in **GeoFrame** must now also be identified with the following attributes:

- Container name
- Container type
- Property code
- Unit of measurement

Accessing Data in GeoFrame and IESX - Data Links and Menus

Binary Data Links for CPS-3

The previous IESX and CHARISMA Links for Seismic Locations and Interpretation are no longer necessary for CPS-3, since seismic data, navigation data, and interpretation are now stored directly in the GeoFrame data base and can be accessed by CPS-3 in other ways. The GeoFrame Link (**GFLink**), on the other hand, is still required, and has been expanded and improved for GF4.0.

How to access specific data types from CPS-3

Here is a brief summary of how selected data classes are accessed from CPS-3 in GF4.0:

2D survey location data are now accessed via the **GFLink** and can be posted with the existing Extended data seismic line posting feature.

3D survey location data is accessed and displayed via the **Display** menu. A new 3D seismic line posting feature is available.

Horizon interpretation is now seen from the CPS-3 file selection dialogs as grids in GeoFrame. Use the “Source” set attribute to distinguish actual interpretation grids from derivative grids. Note that in Modeling Office, Horizon Modeling has been modified to accept GeoFrame grids directly as input. In the CPS-3 Main Module, however, GeoFrame interpretation grids must be Copied to Data sets before using them in Single Surface Gridding.

Fault Segment Interpretation is accessed as Fault Cut Sets from **GFLink**.

Fault Boundary Interpretation is seen in the CPS-3 file selection dialogs as fault boundary sets in GeoFrame, just as before.

Fault Contact Interpretation is seen in the CPS-3 file selection dialogs as scatter Data sets in GeoFrame.

IESX Cartography, which will not reside in GeoFrame, will be brought into CPS-3 with a new Culture Loader which will be invoked from the CPS-3 menus and from the Visualization catalog in place of the old IESX Link.This facility will not be released in 4.0, but in one of the later versions such as 4.01.

Well location data, such as top location, bottom hole locations, and borehole trajectories are accessed from the **GFLink,** just as before.

Well markers for specific Horizons are accessed from the **GFLink,** just as before.

Property summations for specific Zones are accessed from the **GFLink**, just as before.

Tip Loops created by Framework 3D are stored in GeoFrame, but are visible from CPS-3 as **Polygons**.

All other **GeoFrame surfaces**, **scatter sets**, and **fault boundaries** can be seen directly in the CPS-3 file selection dialogs, and, if necessary, can be COPYed to the CPS-3 dsl, just as before.

Geoshare Links for Cartography

Although a **CPS-3 Geoshare** sender and receiver have been more or less unavailable in **GF3.5** through **GF3.8**, a facility exists in **GF3.8** to import.***rp66*** and .***gf66*** files which have been created by **Finder**, and contain cartographic information. Use the **File > Import > Geoshare Culture** menu path to this facility.

The new menu item **File/Import_IESX_Cartography** is now available and supersedes the previous IESX Link.

Geographic Coordinate Systems

Although the decision to store or not store data in **GeoFrame** is optional, the use of geographic coordinate systems is not. **Every set created in CPS-3 must be associated with a coordinate system which has been defined in GeoFrame.**

Sets are associated with a geographic coordinate system, or with the default coordinate system at the time of their naming and creation.

If two sets are associated with different coordinate systems, **CPS-3** will automatically perform the numerical conversions required during operations which use both sets. Specific rules for these conversions are covered in a later chapter.

Tip: *You may load data having any range of cartesian X,Y values into any **GeoFrame** project. As long as you reference the same GeoFrame coordinate system definition where necessary, the system will not attempt any kind of conversion. It will simply accept the data as it is.*

Rules of the Road for Automatic Coordinate System Conversion in CPS-3 Sets

Every set in **CPS-3** is now associated with a particular coordinate system by virtue of its assigned **Display** or **Modeling** environment under which it is created. Here are the basic rules governing how coordinate systems are initiated or modified under certain common conditions.

Best Practice: *Unless your project requires something different, the recommendation is to use one **coordinate system** for all sets in your project.*

Note that only **coordinate systems** (which includes **rotation**) are automatically converted in **CPS-3**. There are no ad hoc facilities for conversion of either **units** (feet, meters,...), or **domain** (time, depth,...).

- **Surfaces**:

  In practice, surfaces cannot be transformed from one coordinate system to another. However, graphic manifestations of a surface, such as contours or postings of nodes, can be transformed.

- **Reading ASCII data in geographic coordinates**:

  Data with geographic coordinates (degrees, minutes, seconds, or decimal degrees) will be transformed using the currently active **Display** environment.

- **Reading ASCII data in x, y coordinates**:

  In this instance, no conversion takes place. The data being loaded takes on the stamp of the active **Display** environment.

- **Surface Arithmetic and Surface Operations**:

  The environments of all surfaces in the operation must match the current **Modeling** environment.

- **Gridding**:

  During gridding, control points and fault traces are transformed to match the coordinate system of the active **Modeling** environment.

- **Graphic Display**:

  Any set displayed will be transformed to match the coordinate system of the currently active **Display** environment.

- **Data Links such as GFLink**:

  Any data moved into CPS-3 via these data links will be automatically transformed to match the coordinate system of the currently active **Display** environment. Note that **units** and **domain** of the output set is established by the **GeoFrame Display Units**, not the currently active **CPS-3 Display** environment.

Enhancements in CPS-3 for GeoFrame 4.0

In this technical note, we'll outline the enhancements added to CPS-3 in GeoFrame 4.0.

Enhancements in CPS-3 for GF4.0

1 Framework 3D Integrated into Modeling Office (MO)

- provides FW3D and P3D on same canvas
- reverse thrust fault capability added to Framework 3D
- ITC selection added to Model Editor for communication with Modeling Office

2 New CPS-3 Menus for Viewing FW3D output

- Display Framework contours including reverse fault contouring
- Display Framework Cross-sections
- Display Framework Allen diagrams

3 Model Editor enhanced with ITC for communication with MO

- surface changes in Modeling Office are seen in the Model Editor session

4 New modeling feature in Single Surface gridding allows conformal modeling

- uses upper and lower reference surfaces
- uses same algorithm as Horizon Modeling, but without fault framework

5 New surface operation allows "updating" of a grid with a new data set

- gives user ability to establish radius of influence for the data set

6 New versions of MSEDIT, MSPEEK, and MSPOKE are available and support rotated grids

7 Grid operations with automatic lattice matching to current modeling environment

8 New icon added to create display environment from current zoom window (GF3.8.1)

9 Macro facilities enhanced

- can now spawn a system task from a macro and wait (GF3.8)
- can now spawn a background task (GF3.8)
- access to macros from three categories - system, project, user
- can add ***descriptions*** to macros which are visible during macro selection
- define and assign macro ***categories*** which are displayed during selection
- new “Yes/No” prompt type
- new prompt facility - select from list of options
- prompt titles and prompt strings can now be variables
- can load extended data with existing format
- can define set attributes

10 Faster color-shading algorithm

11 New, faster contouring algorithm

12 Vector display function enhanced to accommodate rotated grids

13 Ability to delete rows in Data Editor

14 Polygon fill limit increased to 5000 vertices

15 Controls for posting fault names and z-values has been enhanced

16 Added creation/modification time and date as new set attributes

17 Added capability to select which attributes to display in the set selector dialog

18 Added set utility (copy, delete...) icons to List/Manage Sets dialog

19 “Expert Level” removed

20 Ability to customize the Display menu

21 GFLINK now has better borehole selection tools

22 GFLINK can now access fault cuts.

23 Specify line echo color in Digitizing dialog.

24 3D Seismic Survey Display

Examples of Macro Enhancements

The following macro excerpt shows how you can now use a variable string for the dialog title and the prompting text. It also shows a simple Unix command can be spawned.

```
declare $func as string
declare $title as string
declare $prompt1 as string

let $func = "ls /home/userA/data/*.dat > /home/userA/data/datlist"
let $title = "Prompt for Unix command string"
let $prompt1 = "Enter Unix command string"

begin_dlg
title $title
prompt $func "Enter command string"
end_dlg

spawn $func
```

Spawning detached tasks can also be done with the spawn command if you use the ampersand at the end.

```
spawn "xterm -l tbaker -n Macro Xterm ABC &"
```

The next example shows how to establish the set attributes for new sets.

```
M1OPEN "DA" "GF:f_azul" "UNKNOWN"
       "SurfaceName=f_azul"
       "SurfaceType=Fault"
       "SubType=Scattered Points"
       "PropertyCode=Depth"
       "ZUnit=m"
FDASCI 0 2 Computed 0 1E30 12
F1TRAS .
FDATTR 1 2 1 0
FDATNM 1 "" ""
FDAFIL 1 0 0
M1ASCI "d" "i" "/home/clumsden/Macros/Porton_Temp/f_azul.dat"
```

The example menu below shows that you may have three basic sources for collections of macros a **system**-wide collection, a **project**-related collection, and any number of "**user**" collections which can be identified simply by a path name.

In this menu, the **project**-based collection of macros has been selected and its macros listed. One of them has been highlighted which causes its internal **description** lines to be displayed.

The next figure shown an example of a "macro_index.txt" file, whose purpose is to subdivide a collection of macros into **categories**. In this example, each category is defined with a "CATEGORY:" keyword followed by the user-supplied category name. Following this, all macros belonging to that category are listed. The macros and the "macro_index.txt" file must be in the same directory.

```
CATEGORY: Prompts
prompttest.mac
CATEGORY: Logic
iftest.mac
whiletest.mac
CATEGORY: Misc
spawntest.mac
c_and_t_test.mac
```

In this example, the **System** macro group has been chosen, and the names of all macros in the “Display” **category** have been displayed for selection.

LECTURE B for Topic 5
Enhancements in CPS-3 for GeoFrame 4.0

Overview

In this technical note, we'll outline the enhancements added to CPS-3 in GeoFrame 4.0.

Enhancements in CPS-3 for GF4.0

1 Framework 3D Integrated into Modeling Office (MO)

- provides FW3D and P3D on same canvas
- reverse thrust fault capability added to Framework 3D
- ITC selection added to Model Editor for communication with Modeling Office

2 New CPS-3 Menus for Viewing FW3D output

- Display Framework contours including reverse fault contouring
- Display Framework Cross-sections
- Display Framework Allen diagrams

3 Model Editor enhanced with ITC for communication with MO

- surface changes in Modeling Office are seen in the Model Editor session

4 New modeling feature in Single Surface gridding allows conformal modeling

- uses upper and lower reference surfaces
- uses same algorithm as Horizon Modeling, but without fault framework

5 New surface operation allows “updating” of a grid with a new data set

- gives user ability to establish radius of influence for the data set

6 New versions of MSEDIT, MSPEEK, and MSPOKE are available and support rotated grids

7 Grid operations with automatic lattice matching to current modeling environment

8 New icon added to create display environment from current zoom window (GF3.8.1)

9 Macro facilities enhanced

- can now spawn a system task from a macro and wait (GF3.8)
- can now spawn a background task (GF3.8)

- access to macros from three categories - system, project, user
- can add ***descriptions*** to macros which are visible during macro selection
- define and assign macro ***categories*** which are displayed during selection
- new “Yes/No” prompt type
- new prompt facility - select from list of options
- prompt titles and prompt strings can now be variables
- can load extended data with existing format
- can define set attributes

10 Faster color-shading algorithm

11 New, faster contouring algorithm

12 Vector display function enhanced to accommodate rotated grids

13 Ability to delete rows in Data Editor

14 Polygon fill limit increased to 5000 vertices

15 Controls for posting fault names and z-values has been enhanced

16 Added creation/modification time and date as new set attributes

17 Added capability to select which attributes to display in the set selector dialog

18 Added set utility (copy, delete...) icons to List/Manage Sets dialog

19 “Expert Level” removed

20 Ability to customize the Display menu

21 GFLINK now has better borehole selection tools

22 GFLINK can now access fault cuts.

23 Specify line echo color in Digitizing dialog.

Examples of Macro Enhancements

The following macro excerpt shows how you can now use a variable string for the dialog title and the prompting text. It also shows a simple Unix command can be spawned.

```
declare $func as string
declare $title as string
declare $prompt1 as string

let $func = "ls /home/userA/data/*.dat > /home/userA/data/datlist"
let $title = "Prompt for Unix command string"
let $prompt1 = "Enter Unix command string"

begin_dlg
title $title
prompt $func "Enter command string"
end_dlg

spawn $func
```

Spawning detached tasks can also be done with the spawn command if you use the ampersand at the end.

```
spawn "xterm -l tbaker -n Macro Xterm ABC &"
```

The next example shows how to establish the set attributes for new sets.

```
M1OPEN "DA" "GF:f_azul" "UNKNOWN"
       "SurfaceName=f_azul"
       "SurfaceType=Fault"
       "SubType=Scattered Points"
       "PropertyCode=Depth"
       "ZUnit=m"
FDASCI 0 2 Computed 0 1E30 12
F1TRAS .
FDATTR 1 2 1 0
FDATNM 1 "" ""
FDAFIL 1 0 0
M1ASCI "d" "i" "/home/clumsden/Macros/Porton_Temp/f_azul.dat"
```

The example menu below shows that you may have three basic sources for collections of macros a **system**-wide collection, a **project**-related collection, and any number of "**user**" collections which can be identified simply by a path name.

In this menu, the **project**-based collection of macros has been selected and its macros listed. One of them has been highlighted which causes its internal **description** lines to be displayed.

The next figure shown an example of a "macro_index.txt" file, whose purpose is to subdivide a collection of macros into **categories**. In this example, each category is defined with a "CATEGORY:" keyword followed by the user-supplied category name. Following this, all macros belonging to that category are listed. The macros and the "macro_index.txt" file must be in the same directory.

```
CATEGORY: Prompts
prompttest.mac
CATEGORY: Logic
iftest.mac
whiletest.mac
CATEGORY: Misc
spawntest.mac
c_and_t_test.mac
```

In this example, the **System** macro group has been chosen, and the names of all macros in the “Display” **category** have been displayed for selection.

LECTURE for Topic 6
Understanding CPS-3 Set Types in the CPS Partition (DSL)

Overview

In this lesson, we will describe the way in which **CPS-3** sets are organized within its local data store. This local data store is maintained by the **CPS-3 Project Manager** independently of the **GeoFrame** data base. Please refer to a previous lesson which discusses the integration status between **CPS-3** and **GeoFrame**.

A Typical CPS partition in a GeoFrame Project

The **CPS** partition is simply a pointer to a disk directory, for example:

/home/disk1/user1/projects/CLOUDSPIN/CPS

The first part of the path **/home/disk1/user1/projects** is determined by the **owner** of the project **CLOUDSPIN**.

The remainder of the path **/CLOUDSPIN/CPS** is determined by **GeoFrame**.

In this example, the full path shown above defines the location of the **CPS** local data store, or partition, within the **GeoFrame** project **Cloudspin**. In this directory, you will find everything which is managed by the **CPS Project Manager**.

As mentioned in the previous lesson about **GeoFrame** integration, it is possible in this release to optionally store some **CPS-3** sets in the **GeoFrame** data base, which is separate from this partition.

Ultimately, the data to which you have access in **CPS-3** will be stored in one or more of the file types outlined below. These files will reside in your **CPS dsl**. Note that each is distinguished by its **UNIX** file extension which is not seen in the **CPS-3** menus or dialogs.

- **.1cps** files - **session** files; files containing parameter values
- **.dcps** files - **data** files; x,y,z files
- **.fcps** files - **fault trace** files; x,y files; optionally, x,y,z files
- **.pcps** files - **polygon** files; x,y only
- **.scps** files - **surface** (grid) files; z only; x,y is inferred
- **.mcps** files - **map** files; picture files
- .**tcps** files - **table** files such as created by **Framework3D**

Data, **Fault**, and **Surface** sets may optionally be stored in either the local project files, as shown above, or in the **GeoFrame Oracle** data base, or both.

Each **CPS-3** set can consist of one or more named **subsets**.

Session Sets (or Session Files)

Usually, there is only one session set in your dsl location. It has the name **<your login id>.1cps** and contains the most recent values for all parameters you set during your **CPS-3** session. This includes definitions of your modeling and mapping environments. You will also see session files created for batch processes you initiate from **CPS-3**. Each will have a unique file name.

Data Sets (.dcps)

Description

Data sets contain information which is to be gridded or displayed, such as data points. Examples are well markers, seismic interpretation, and scatter data.

Data Types

CPS-3 uses a data set's **Data Type** to compute defaults for some of the modeling and display parameters. Most of the **Data Types** indicate a type of spatial distribution pattern which can be exploited by certain algorithms. Data types are shown in the following lists:

Data Types Having Z-values

- Scattered points
- Contours
- 2D Interpretation
- 3D Interpretation
- Fault Segments/Cuts
- Fault Contacts
- Pseudo Grid
- Feature Lines
- Borehold Trajectories
- Well Markers

Data Types Without Z-values

- 2D Lines (locations)
- 3D Lines (locations
- Well Surface Locations
- Well Bottom Hole Locations

Data Content - Z Fields

Each data point loaded into **CPS-3** must have at least an X and Y coordinate, but may also have up to 50 z-fields. These z-fields are numeric, and can represent any variable which is spatially distributed. A vertical well, for example, might have one z-field value for each horizon it passes through (markers), or a single well point may have one horizon depth value, and an associated thickness to the next layer. Z-fields may also contain dip and strike information for the horizon being mapped. There are no predefined combinations of z-fields which are required for **CPS-3**. Each z-field can represent anything you like. The most common data files loaded by **CPS-3** are data files containing x,y, and a z-value representing time or depth.

As you will see when loading data for the course, there are two attributes associated with each z-field:

- Z Field **Name** (z1, z2, top, bottom,...
- Z Field **Type** (depth, elevation, isochore,...

Data Content - Text Fields

Some data files contain textual information for each point which is to be posted on the map; for example, well name or operator name. There are two methods for storing **Data** sets in **CPS-3** - **Extended Data**, and **non-extended Data**.

Non-Extended Data

A data point in stored in this form may have up to 50 numeric z-fields, but no text fields. The only text allowed with this form of storage is a subset name. For well data, this can represent the well name if you select the **x,y,z + Name Field** option for loading. Use this type of **Data** for simple x,y,z points to be gridded when no special display capabilities are required.

Extended Data

This type of data allows all of the above, but also allows the storage and posting of up to 10 **text fields**, as well as the storage and utilization of **graphic symbology** attributes. Symbology attributes, such as symbol type, size, and color, can be associated with each data point. At present, **Extended Data** cannot be exported from **CPS-3**, nor can **Extended Data** be loaded by the use of macros.

Subsets

Data sets can be organized into subsets which makes certain processing easier. For example, line-oriented data (2D seismic, digitized contours) should be identified as such during **ASCII** loading so that each line becomes an identifiable subset in **CPS-3**. Several formats suitable for controlling line-oriented data will be illustrated below in the examples.

Fault Sets (.fcps)

Description

Fault sets in **CPS-3** store those polylines which are commonly referred to as fault **traces**, or in **GeoFrame** terminology, fault **boundaries**. To the **CPS-3** procedures, it does not matter how these polylines are geometrically defined. For example, a non-vertical fault pattern associated with a particular horizon can consist of two separate lines - an upthrown line and a downthrown line, or it may consist of a single closed polygon.

Types of Faults

Faults can be categorized by the user in **CPS-3** as either **vertical** or **non-vertical** at the time the faults are loaded. In other cases, some **CPS-3** operations in **Framework3D** automatically classify faults which they have created as being fault **traces**, fault **centerline**s, or fault **polygons**.

Fault Attributes

Each fault trace must have an x,y coordinate, but other information is optional. For example, if z-values are available on the trace, then these values should be loaded along with the x,y values to help in the gridding of the associated horizon. **Vertical and horizontal separation**, as well as **dip** values are also valid numeric fields which can optionally be loaded with fault traces.

Associations

Fault traces used as input to a gridding operation are automatically associated with the **surface** which is created. That is, the name of the fault set is stored in the parameter block of the surface set. This means that when contouring, you don’t have to remember which fault set to use for each surface.

Subsets

In a fault set for one horizon in **CPS-3**, there is typically one fault trace pattern for each named fault in the reservoir. The pattern for each named fault is stored as a separate subset in the fault set. Several loading formats will be shown in the examples to ensure that individual lines are identifiable in **CPS-3**.

Polygon Sets (.pcps)

Description

There are two main purposes for polygon sets - the first is to define some cartographic basemapping feature, such as a shoreline which is to be posted on a basemap. Note that unclosed polylines are also valid to store in a polygon set. The other use of polygon sets is to define some region in the modeling area within which operations are to be performed or excluded from being performed, for example, grid blanking or volumetrics calculation.

Types

Polygon sets can be typed as either **open** (polylines) or **closed** (polygons).

Subsets

A polygon set can consist of multiple polygons, or polylines, each of which is a separate subset.

Surface Sets (.scps)

Description

Surface sets hold the grids in **CPS-3**. Only the z-value at each grid node is stored. Parameters stored with the surface, such as the lower left corner of the grid, and the grid spacing in x and y, are used by the system to compute the x,y coordinates of each node location when needed.

Classification

Surfaces can be classified as one of the following:

- **horizon** surface
- **fault** surface
- **truncated** (**blanked**)
- **truncated** (**filled**)

These classifications are generated automatically by the software, in particular, the **Framework3D** operations.

Associations

As mentioned under **Fault Sets**, the input fault trace set is associated with the output surface in a gridding operation. In addition, there are spreadsheet-like operations in **CPS-3**, particularly in **Framework3D**, where groups of surfaces are defined. These groups of surfaces become associated during the creation of a **Table** set (a **.tcps** file in the project directory). For example, a fault framework **Table** set is created as the user loads surfaces in to the framework. If a surface in a table happens to be deleted, the system will recognize its absence the next time the table set is invoked.

Map Sets (.mcps)

Description

Essentially, a **Map** set is a picture file. The picture on the screen can be saved to a **Map** set at any time in **CPS-3**. Although it is possible, **Map** sets are not usually loaded into **CPS-3** from **ASCII** source files.

Subsets/Map Layers

All graphic components created by a single graphic operation are grouped as a separate subset in the saved map set. These separate graphic layers can be deleted, moved, or even hidden temporarily in **CPS-3** with the **Map Layer Manager**.

In the map set shown below, there are seven subsets, each of which is created by a separate display operation.

- **Border**
- **Border Labels**
- **Scale Bar**
- **Contours**
- **Wells**
- **Seismic Lines**
- **Fault Traces**

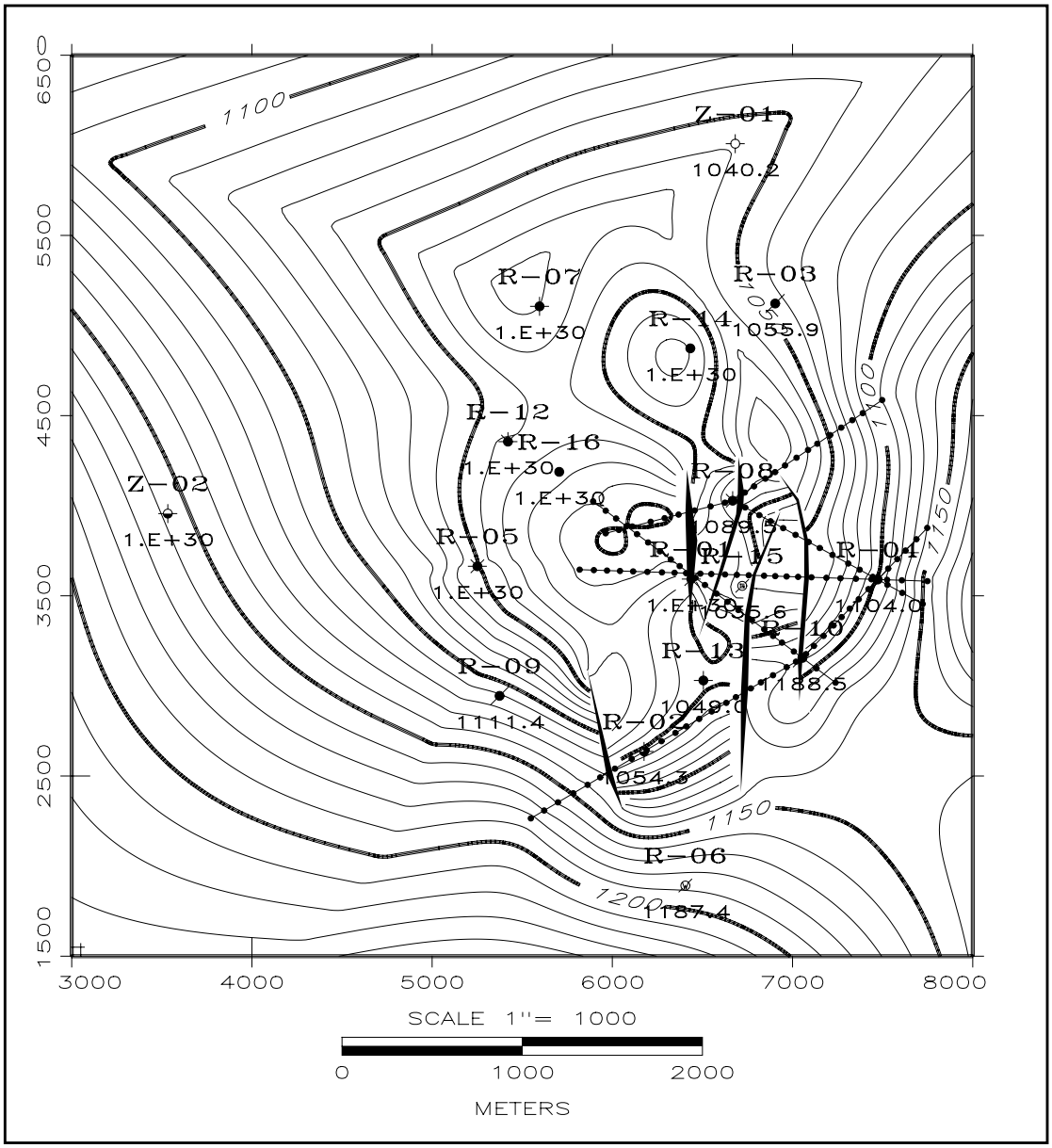


Figure 6.1 Example of a map set displaying seven subsets

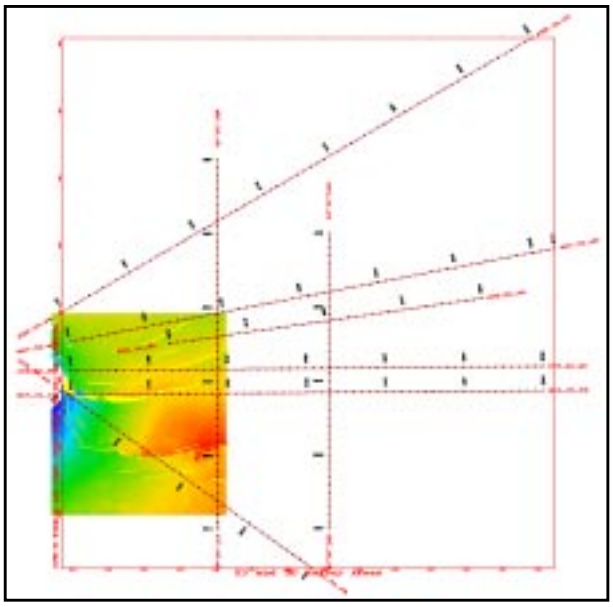
LECTURE for Topic 9
Introduction to Display and Modeling Environments

Overview

Here, we will discuss the concept of mapping environments and their attributes. We will show the difference in environment attributes required to perform **display** functions, versus the attribute requirements to perform **modeling** operations. We will discuss the reason for having multiple environments, and, in a later chapter, we will have exercises to create environments, store them, and edit them.

Just to get started, we will make the following simple, but true, statement:

“Before you can display anything, like the map below, you have to define a **display** environment. Before you can create a grid, you have to define a **modeling** environment”.



Granted, a default mapping and modeling environment is provided as environment #1, however, it may not cover the exact area nor grid lattice that you really want.

The material in this section is designed to provide **formal definitions** of environments, their attributes, overall purpose, and management in a fairly concise lecture format. You will learn most about environments by working with your data. However, it is anticipated that you may refer to this section frequently until you have gained a full understanding of the use of environments.

Definition of Mapping Environment Components

An **Environment** is a **named collection** of information associated with the creation of maps or surface models in a **CPS-3** project.

Attributes which are associated with environments are:

- **Name** and optional **Description**
- Usage **Classification**
  - Both **Modeling** and **Display**, or
  - **Display** only
- **Volume of Interest** - minimum and maximum values for x, y, and z, defining the mapping area and its vertical extents.
- **Geographic Coordinate System** specification
  - Geodetic datum (ellipsoid plus shifts)
  - Map projection: transformation method and associated parameters
  - Rotation: origin and angle
- Definition of **horizontal and vertical units**
- Definition of **horizontal and vertical scale factors**
- Definition of **vertical property code**

Attributes which are associated with **Modeling** operations are:

- Definition of a **grid geometry**
  - name, description, units
  - lattice origin, offset
  - lattice spacing
  - rotation angle

Display Environment

A **Display environment** is used only for display purposes. It does not contain the definition of a grid geometry. Data displayed through the currently active **Display environment** will undergo the following procedures:

- transformed to the active **Geographic Coordinate System**
- scaled to the active scale
- converted to the active horizontal units
- clipped in x and y by the active **Volume of Interest**

Modeling Environment

A **Modeling environment** is used to create a grid or perform other modeling operations. Think of a modeling environment as simply a display environment to which a **grid geometry** has also been added. An environment which does not contain a **binset** definition cannot be used for modeling, only display. Grids created in modeling operations will assume the grid origin and spacing of that defined in the currently active **Modeling environment**.

The Relationship between CPS-3 Modeling Environments and GeoFrame Binsets (Grid Libraries)

The new concept of **CPS-3** mapping environments is tied to the **GeoFrame** data base, at least in terms of binsets or grid libraries. Every **Modeling environment** in **CPS-3** must point to a specific **binset** in GeoFrame. **Display environments**, as you will discover, do not require grid lattice definitions.

Just as all sets must be associated with a particular coordinate system in **GeoFrame**, the grid spacing in all **CPS-3** surfaces must now be defined in terms of a specified **Grid library** (**binset)**. All binsets are stored in GeoFrame. A **binset** is a formal definition of a grid lattice and has several attributes, including x-spacing, y-spacing, rotation, x-offset, and y-offset, but not all of the parameters are associated with a **CPS-3 Modeling environment**. At present, however, all we need to know is that we must either define a new binset (the most common option), or select one from an existing list, in order to specify x and y spacings for surfaces.

Later in this course, exercises will be provided to demonstrate the mechanics of how data can be stored in either **CPS-3** or **GeoFrame**. Also, we will demonstrate how to create, delete, and edit mapping environments.

More Notes on Binsets

1. **Binsets** existed in **GeoFrame** before **CPS-3** began offering **Display** and **Modeling environments**. All three concepts are similar, in that they define a collection of mapping parameters which control high-level display operations and/or the granularity of a geological model. The related concept of **Grid Library**, which is effectively synonymous with **binset** in **GeoFrame**, was invented years ago to guarantee that grids from different applications would overlay. Nodes from all grids created under the same grid library would, by definition, overlay.

2. Once a **binset** has been created, it cannot be modified through the **CPS-3** menus. Even if you could, you would **NOT** want to do this because, in a sense, **binsets** are shared by all users in the **GeoFrame** project, and modification would cause problems for all child-objects.

3. Be careful if you **delete binsets** because every surface you delete, which was created with the **binset** to be deleted, will become unusable in **GeoFrame**. Use the following procedure to find out which **GeoFrame** surfaces were created with a particular **binset**:

Procedure: Determining a Particular Binset

1. From the **GeoFrame Data Manager**, go to **Grid Libraries**. (This opens the **Project Grid Library Data Manager**.)
2. Highlight the desired **binset** and click the **Get information...** icon. (You will see a list of all Surfaces containing grids which were created with the selected **binset**.)

4. In our example, we picked x and y-intervals which divided evenly into our x and y ranges. When the division is uneven, the system automatically **increases** the x-maximum and/or y-maximum values so that the division will be exact.

5. The **coordinate system** defined for the **binset** always becomes associated with the **environment for which we are defining a binset**. Thus, any previous coordinate system in the active environment will be overwritten by the one in a selected **binset**. This is true for the **binset** we created, and would also be true had we used a method of creating our **Modeling environment** which involved the picking of an existing **binset** from the list.

Making Use of Environments

To perform any function in, an appropriate environment must be selected.

- **Listing Available Environments**

  Environments are selected, created, and edited using the **Select Environment** icon  which opens the following dialog box.

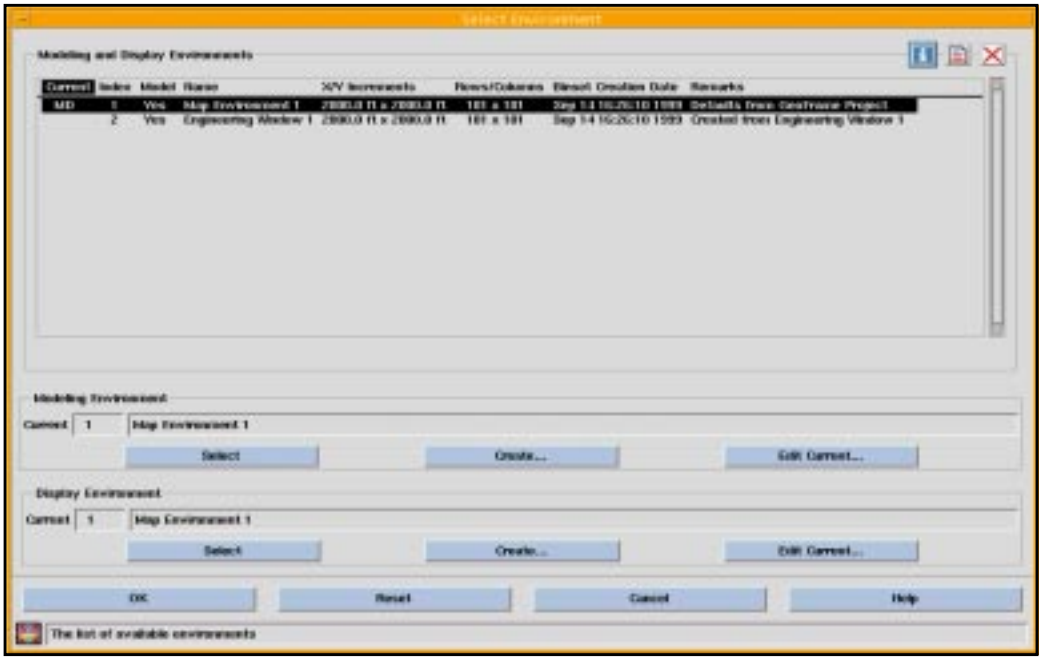


Figure 9.1 Select Environment dialog box

- **Selecting An Environment for Display or Modeling or Both**

  Select an environment by clicking on it with **MB1** in order to highlight it in the Select Environment dialog box, and then clicking **Select** under the **Modeling** or **Display** panel, or both, as appropriate. If the highlighted environment has no **binset** defined, it will show **No** under the **Model** column, and cannot be used for modeling.

- **Creating A New Environment**

  Click **Create** under the **Modeling** or **Display** panel, as appropriate.

- **Editing An Environment**

  Highlight the desired environment, and click **Edit Current** under the **Modeling** or **Display** panel, as appropriate, or simply click the **Edit** icon. 

- **Reviewing Environment Attributes**

  Highlight the desired environment, then click the **Review** icon. 

- **Deleting Environments**

  Highlight the desired environment, then click the **Delete** icon. 

In the many exercises which follow, you will become familiar with the use of these functions.

- **Notes**
  - If during the creation or editing of an a **Modeling environment**, you select an existing **binset**, then the **binset's** x,y box and geographic coordinate system replaces the x,y box and coordinate system of the **Modeling environment** you are creating or editing.
  - **Environments** you define are stored in your current **session file**, **<login id>.1cps**, which is located in the **CPS** partition of the project. ***Refer to a later section regarding the storing and retrieval of session files.***
  - Because of the closer integration with **GeoFrame**, and the institution of environments, **CPS-3** procedures have become more sensitive to **units** and **domains**. For example, an attempt to compute borehole intersections with a surface set, which is stored in milliseconds from a borehole set which was loaded from GeoFrame in seconds, will result in an error message.
  - Just as procedure sensitivity to environments may cause error messages in some cases, it will also provide benefits. For example, during many operations, input data from sets with differing coordinate systems will be ***automatically converted*** to match the coordinate system of the output set.

Multiple Environments

The most apparent characteristic of a display or modeling environment is its x,y extent - the area of interest, or, as it used to be called, the “engineering window”.

From the sample map at the beginning of this section, it can be seen why more than one environment is needed in typical projects. The **Modeling environment,** as its name suggests, is used for defining that area where grid nodes are computed. The **Display environment**, on the other hand, has more to do with defining the extent of the graphic entities making up a basemap. Continuing with the example map, a basemap might be required which included all available seismic surveys including the 2D, but the area required for gridding and volumetric calculation may cover only the smaller 3D survey.

In this case, we would define a small **Display/Modeling** environment covering the 3D survey area, and a larger **Display** environment covering the entire 2D extents, as shown below.

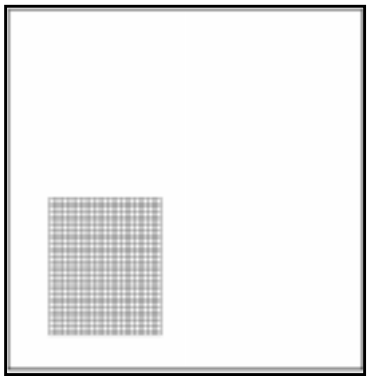


Figure 9.2 Example of CPS-3 ability to define 2D and 3D data on same basemap

Thus, with this version of **CPS-3**, it is possible to perform graphic operations covering one geographic area, while creating grids in another area at the same time.

Setting Up for Horizontal and Vertical Scaling and Limiting

Besides establishing a traditional x,y map scale, **Display environments** let you define the attributes needed to control **vertical** scaling, which is an issue during 3D visualization provided by the **Assembly** tab in **Modeling Office**, as well as the display of ***profiles***, or ***cross sections***.

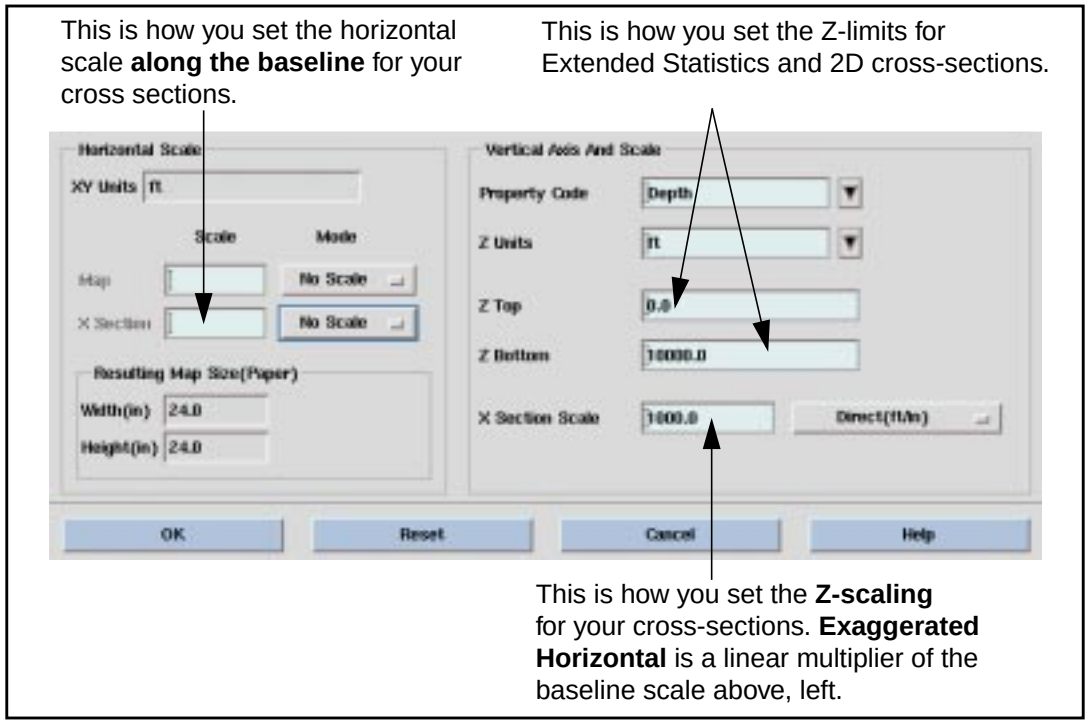


Figure 9.3 Setting vertical and horizontal scaling and limits

Storing and Retrieving Environment Definitions

The current **Environment Selector** and **Environment Editor** are sufficient for the management of environments in most cases. However, because your environments are actually stored in your **session file**, they are somewhat at risk when it becomes necessary to remove the session file during remedial project cleanup activities. Certain temporary problems can be cleared up in **CPS-3** by removing the **session file**, and allowing the system to create a new version. Unfortunately, you lose your **environment definitions** because they are stored there. For this reason, the recommendation is to make use of the **save** and **read** commands to periodically backup your current environment definitions. These commands are activated in the **Command** line box at the bottom of the **Status Window**. Here is how it works.

- **Make a permanent copy of your environments periodically**

  Whenever you create a new environment, **save** your session file as follows:

  Type **`save <file name>`** in the command line.

  This causes your entire session file, which contains the latest values of all parameters you have set, plus your environment definitions, to be written to the file name **<file name>.1cps**. This file will reside in your **CPS** partition.

- **Reinstating lost environments**

  If your session file is lost, you can reload all your environments in ***append*** mode to your current session file as follows:

  Type **`read <file name>`** in the command line.

  This will cause all parameters and environments in the file **<file name>.1cps** to be appended to your current session file.

Tip: *When an environment with an associated **binset** is loaded, the system searches the **GeoFrame** data base for the first **binset** which matches the attributes of the one in the session file, and links it to the environment being loaded, regardless of the name of the original **binset**. If a matching **binset** is not found, the environment is not loaded. For this reason, it is not recommended to make use of sessions files from other projects.*

Rotated Grids

By virtue of the attributes available in the definition of **Modeling environments**, a new facility is now available in **CPS-3** — the ability to create **rotated grids.** The rotation attribute in the **coordinate system** specifications allows the definition of grids like the one shown in the map below, which is based on the survey azimuth.

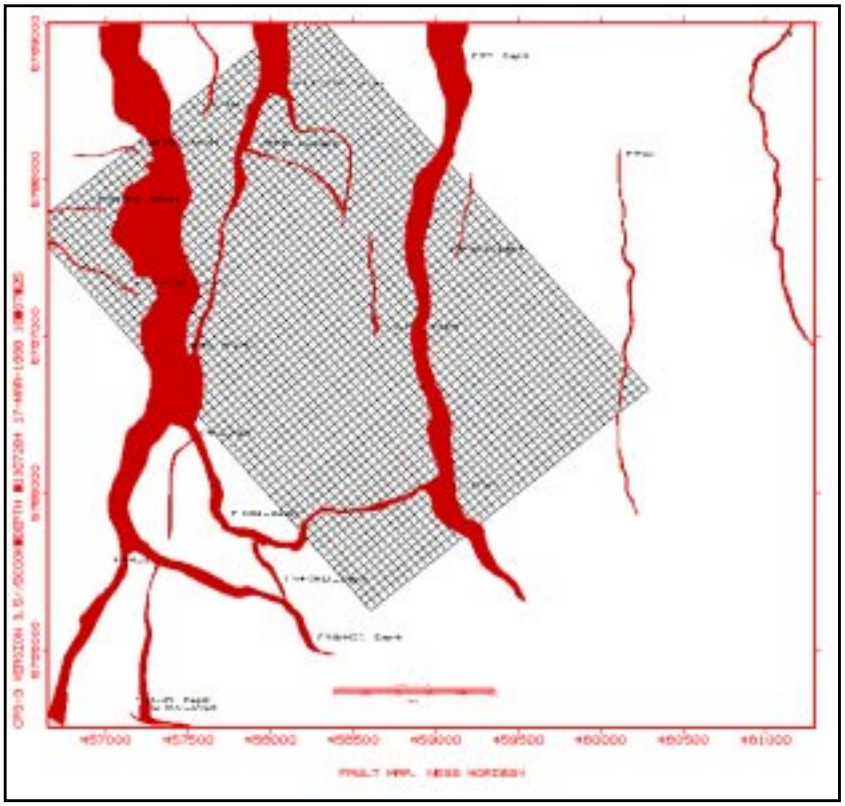


Figure 9.4 Example of a fault map with rotated grid

Association of Environments with Sets

Whenever any of these set types are created, attributes from the **unrotated** active **Modeling environment** are stored in the parameter block of the set: **Data**, **Fault**, **Polygon**. Attributes from the **rotated** active **Modeling environment** are stored in the parameter block of the set: **Surface**, **Map**.

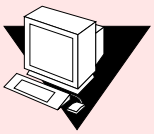
Specifying the Characteristics of Display and Modeling Environments

As you create these two types of environments, you will notice that there is a wide degree of flexibility in the specification of such parameters as X and Y limits, and even grid spacing.

For example, when specifying the X and Y limits of a Display environment, you can use the existing X and Y limits of any selected set. You can also type the X and Y limits in directly in cartesian coordinates, or you can specify them in decimal degrees.

When creating Modeling environments, you have the same flexibility. For example, you have the ability to create a Modeling environment by simply pointing to an existing binset in GeoFrame. If you use this procedure, please make note of the following:

Note: *If you happen to choose a method to create your **Modeling environment** which involves the **selection** of a particular **binset** from the **GeoFrame** list of **binsets**, you should know that **binsets** contain not only a grid lattice definition, but specifications for **x, y limits** and a **coordinate system,** both of which will **supersede** those values you may have already entered for your new environment in the **CPS-3** dialogs.*



LECTURE for Topic 10
Locating Seismic Interpretation Components for CPS-3 Mapping.

Overview

In GeoFrame 4.0, the components of seismic interpretation are effectively stored in the GeoFrame data base and are accessible to CPS-3 in a variety of methods. Some discussion of these new data relationships is needed, however, before the exercises become intuitive. Here is a summary:

3D seismic interpretation is stored as **Grids** in the data base and is visible directly in the CPS-3 Set Selector dialog.

2D interpretation is stored as “Line/Scatter” **Data** and must be loaded into the CPS-3 dsl by the GeoFrame Link.

Fault Cuts are not directly visible from CPS-3 and are loaded into the CPS-3 dsl with the GeoFrame Link.

Fault Contacts are not directly visible from CPS-3 and are loaded into the CPS-3 dsl with the GeoFrame Link.

Fault Polygons are directly visible in the CPS-3 Set Selectors as **Fault** sets.

Seismic Attributes are stored as **Grids** in GeoFrame and are directly visible in the CPS-3 Set Selector dialog.

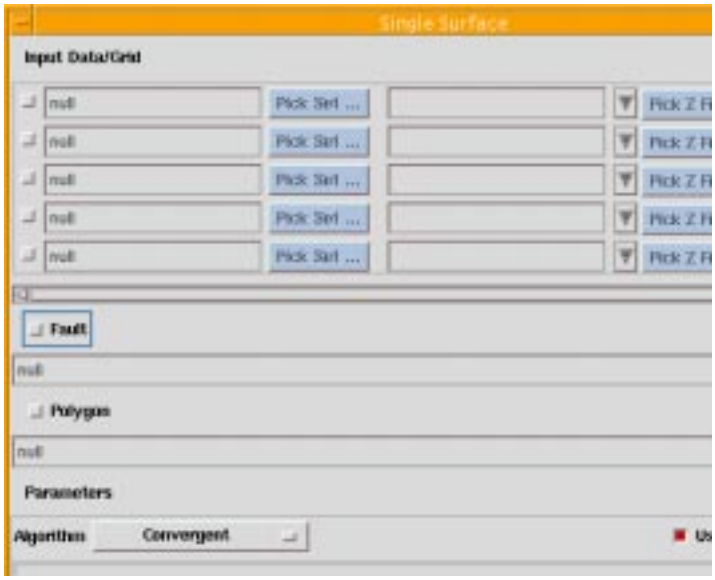
2D locations are not directly visible in CPS-3 and are loaded into the CPS-3 with the GeoFrame Link.

3D locations are visible in CPS-3 and can be posted in the **Display** menu

Cartography in IESX can be imported into CPS-3 using **File/Import/IESX_Culture**

Gridding 3D interpretation

Now, when gridding a 3D horizon, CPS-3 must be able to accept grids in GeoFrame as input to the gridding algorithms. The Gridding menus have been changed accordingly. Note the **Single Surface gridding** dialog now allows the selection of either Data or Grid input:



How do I distinguish an interpretation grid from other grids?

One of the most common objects stored in GeoFrame are grids. The problem arises on how to identify the primary interpretation grids from other derivative grids which can be computed by dozens of other applications.

In any GeoFrame dialog which allows the display of component attributes, you can examine the "**Source Code**" and "**Property Code**" attributes of a grid. If the Source is "Charisma" or "IESX", and the Property Code is "Time", then this is a pretty good indication that you are looking at the primary time interpretation for a particular horizon. A Property Code of "Depth" would indicate primary interpretation in depth. Other, derivative grids will have different Property Codes, for example, "Integrated_Reflection_Strength". A different Source Code also indicates that the grid is not primary interpretation, for example, "Surface Manager" means that the grid was derived from some other grid.

Interpretation Models and CPS-3

A new concept in GF4.0 is the Interpretation Model. The interpretation model is a name given to some collection of interpretation objects - 2D line interpretation, 3D grid interpretation, fault cuts, fault contacts, and fault boundaries, which are related in some way, for example, all being done by the same interpreter. The interpretation model may consist of several horizons and each horizon can consist of multiple patches.

In many applications, the Interpretation Model can be used as a filter when selecting data. For example, during Horizon Modeling in Framework Modeling, you may choose to select input data only from a particular interpretation model. At this time, CPS-3 does not interact with the Interpretation Model concept. All data is seen in the set selection dialog, regardless of its model.

Destinations of Interpretation Components When Imported into CPS-3

When imported from **GeoFrame**, **IESX**, or **Charisma**, the following interpretation components are stored as indicated:

- Geological **horizon markers** (**A**) are stored as **CPS-3 Data** sets (**.dcps**)
- 2D **horizon** interpretations (**B**) are stored as **CPS-3 Data** sets (**.dcps)**
- 3D **horizon** interpretations (**B**) are stored as **CPS-3 Grid** sets (**.scps**)
- 2D/3D **fault segment** interpretations (**C**) are stored as **CPS-3 Data** sets (**.dcps**)
- Seismic **fault contacts** (**D**) are stored as **CPS-3 Data** sets (**.dcps**)
- Seismic **fault polygons** (**E**) are stored as **CPS-3 Fault** sets (**.fcps**)

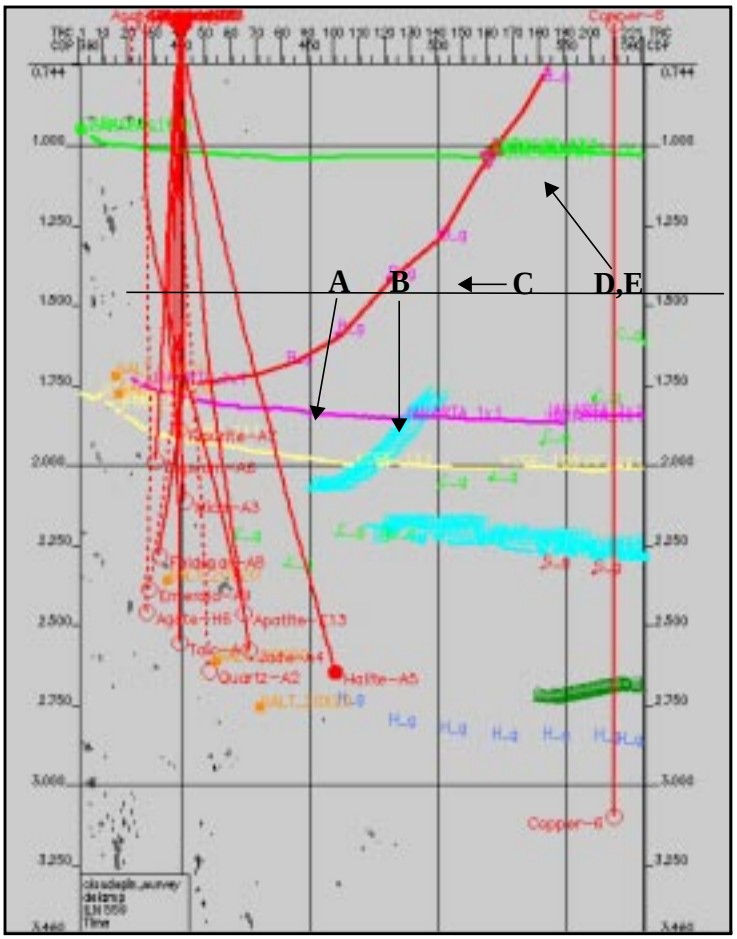


Figure 10.1 Display of interpretation components

LECTURE for Topic 11
Gridding Fundamentals

Overview

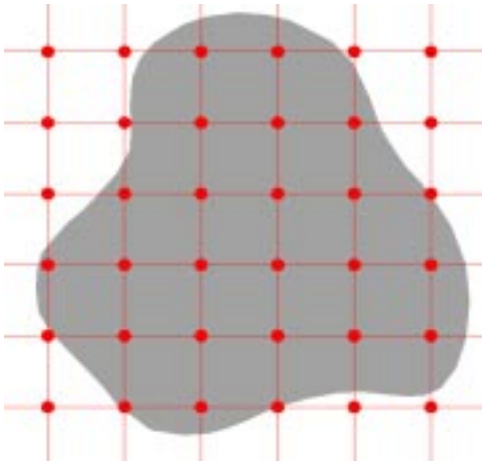


Figure 11.1 Data coverage polygon and grid lattice

The purpose of this chapter is to introduce you to the most important aspects of gridding. You will learn how to select the most critical gridding parameters which affect the quality and appearance of your surface. There are many algorithms from which to choose, and many parameters which can affect your map. However, for the typical mapping task, very good maps can be generated using only a few of these parameters. An **Advanced Topics** class is available for those who wish to study the inner workings of the gridding algorithms more deeply, but the schedule for this course does not allow for all algorithms and their parameters to be covered in depth. In this course, we will try to take the pragmatic approach and give you the tools to get the best map possible in the shortest time.

It is recommended to read the chapter in the on-line **User's Guide** which covers gridding. It gives many details which may be overlooked in this course.

What is Gridding?

Gridding is the process of transforming randomly located or other data into a regularly spaced lattice of values representing the z-dimension of the x, y, and z data. After the transformation, the data points are redundant and unnecessary, since the model of the surface is now embodied totally within the grid and the associated fault traces, if any. Even though defined by a finite number of points, the grid is meant to be thought of as continuous surface.

Grid Terminology

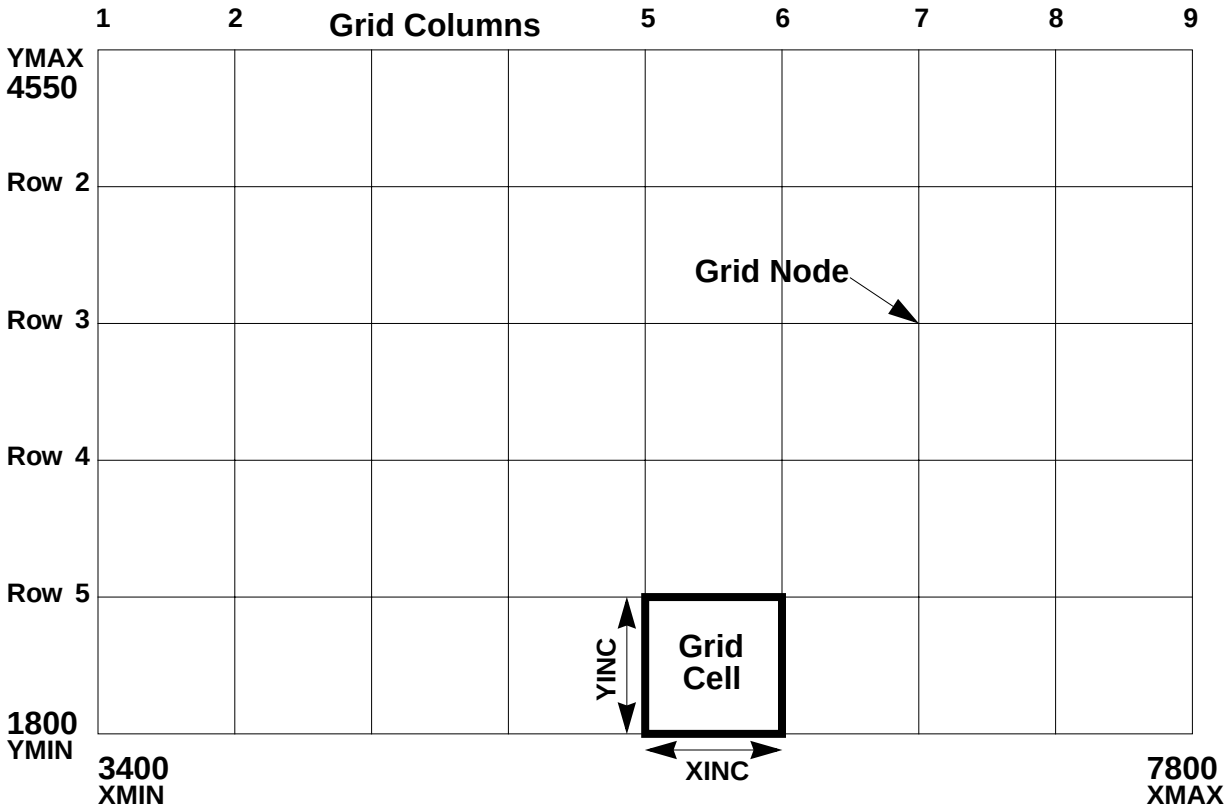


Figure 11.1 Map grid and components

The above grid covers a range in **X** of **3400** to **7800**, and in **Y** of **1800** to **4550**.

This grid is a 6 by 9 grid, meaning 6 rows by 9 columns. The **grid cell size**, the number of **rows** and **columns** and the grid **range** are related as follows:

- **X**-interval = Range in **X** divided by (Number of Columns -1)
- **Y**-interval = Range in **Y** divided by (Number of Rows -1)

CPS-3 may increase **XMIN** or **YMAX** to ensure that the range in x and y is evenly divisible by **xinc** and **yinc**.

Judging the Quality of the Model

After the gridding has taken place, the quality of the model can be determined by inspection of some manifestation of the grid, for example, contours. If the contours indicate that the values of the grid near the data points are **consistent with the data point values**, then the first criteria has been passed.

Next, examine the contours in the areas of the grid between the data points.

- **Are the contours reasonable?**
- **Are they what you would expect?**
- **Are they relatively smooth, considering the data?**
- **Do they continue the trends established by the data?**

If the answer to these questions is **yes**, then the second criteria has probably been fulfilled. Remember that after gridding, your **model** is represented by the **grid and fault traces** - the data has become redundant.

Gridding Algorithms

As you will see, **CPS-3** has many gridding algorithms, but each of them is designed to solve the **basic gridding problem** below:

*Given the **data** points and **fault traces** above, create a model of regularly-spaced grid points which **honor** the data and exhibit a **smooth transition** of the surface between data points and the edge of the map.*

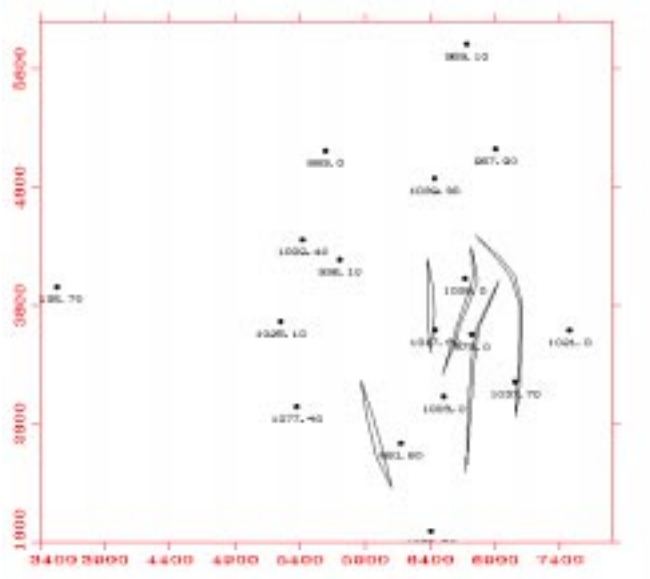


Figure 11.1 Common gridding problem showing data and fault traces

How Do I Prepare for Gridding?

At first, we will assume that the only data to be gridded is the well data shown above. The data for this class includes both 2D and 3D seismic interpretation as well, but for the moment we will disregard it and focus on the wells.

Following is a checklist to help you most effectively grid your data:

- Inspect your **data** points
- Inspect your **fault traces**
- Decide on the **modeling area**
- Decide on **grid spacing**
- Decide which **algorithm** to use
- Decide how to set **gridding parameters**

Data Inspection and Selection of Modeling Area

Inspecting our data shown in the previous figure, let us assume for a minute that we want to create a grid from only this well data and the faults shown. We will assume that we want to include the far western well in the grid and so the modeling area will cover the area shown. Looking at the fault traces, we note that the horizontal separation is narrow for some of the faults, but significant for others. We also note fairly thin fault blocks between several of the faults.

Determining the Grid Cell Size

In its most basic form, the method for determining the proper grid spacing can be stated as follows:

Find the closest two data points whose difference must be distinguishable in the grid and let the grid spacing be 1/2 the distance between them.

This method is almost guaranteed to yield a grid having the desired criteria, that is, a contour map which shows the difference between the two chosen points. However, there may be **other limiting conditions** which you should consider before settling on this grid spacing. For example, it could be that such spacing generates a grid whose number of rows and columns is so large that the gridding takes an **inordinate** amount of time. Is is also possible that such a spacing is so small that it produces a grid which contains a very large amount of high-frequency **noise**. While the stated method of choosing a grid interval is a very good way to start, there are even more reasons why you should look at **other characteristics** of your data as shown below.

Looking at our simple example again in the figure below, we can get a better feel for the scale of our data and the grid spacing by introducing a graphic lattice on the display. The lattice shown is 500 meters on a side. Judging from the spacing of the well points, 500m is not a good grid spacing since there are several points closer than this whose difference we will want to see.

Also note the size of the grid cell relative to the size of the thin **fault blocks**. These blocks will hardly be defined with such large spacing. Sometimes, the ultimate criteria for defining a grid size is the horizontal separation in the **fault zone**. If it is desired to define the fault zones for volumetric purposes, then the chosen grid size must provide enough nodes in each fault zone so that it can be modeled along with the fault blocks.

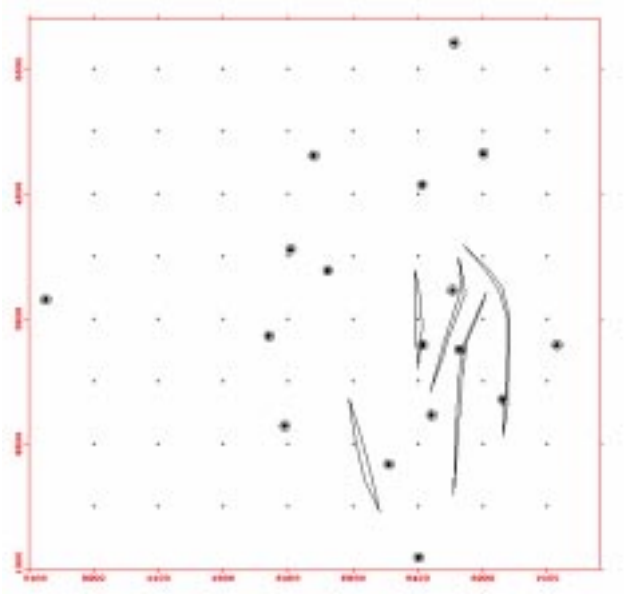


Figure 11.1 Data points and fault traces with grid lattice

If we were to use the “distance between the two closest points” criteria for selecting the interval in this data set, we might choose the two points, each of which borders on a north/south-oriented fault trace toward the southeast center of the map. As we have just noted, however, the width of the fault blocks in that area is even smaller than the distance between these points. Clearly, the width of these fault blocks should be the **defining feature** to be resolved in the final grid. We would therefore choose a grid interval of approximately 100m in this example, which will allow us to define the fault blocks, but not all of the fault zones.

Considering only the well data and the fault patterns, we have determined a reasonable grid interval based on close inspection of the features in our data. But in your project data, you may also have 2D or 3D seismic **interpretation** to go along with the well data. This same sort of analysis should be applied to the seismic data.

As you can see, there is no magic formula for determining a gridding interval, other than deciding on the size of the smallest feature you want to resolve in the resulting surface.

How Do I Choose A Gridding Algorithm?

If your gridding task falls into one of the following **special** categories, you should use the algorithms which are especially recommended for those tasks. Otherwise, you should use the **Convergent Gridding algorithm**.

- create a stratigraphic display of rock types
- create fault plane maps whose features are highly linear
- create a grid honoring very dense grid-like data
- gridding isopachs with partial penetration data
- create a grid having a constant value
- create a mathematically-correct trend grid
- create a mathematically-exact evaluation of a polynomial in x and y
- create a grid of distance or density statistics from a data set

Those gridding tasks mentioned above are special cases and do not represent the typical task of creating a structural model from well data, fault traces, and seismic interpretation, which is the most common modeling exercise. The operations above and the algorithms used for the creation of their surfaces are important to learn about, but in the exercises for this chapter, we will focus on the **Convergent Gridding algorithm**, which is recommended for day-to-day structural modeling.

List of CPS-3 Gridding Algorithms

Convergent

A very stable, fast, general purpose algorithm for computing a smooth, but accurate fit to almost all types of data. This algorithm provides trend-like extrapolation, and being the "flagship" of the **CPS-3** gridding algorithms, it should be the first one considered for almost any structural data.

Below is a schematic showing the refinement of a surface being created by successive iterations of the **Convergent Gridding algorithm**. After establishing an initial trend with a coarse grid, each successive step reduces both the grid cell size, and the radius of influence for each control point, until the surface is locally "tied" to the data.

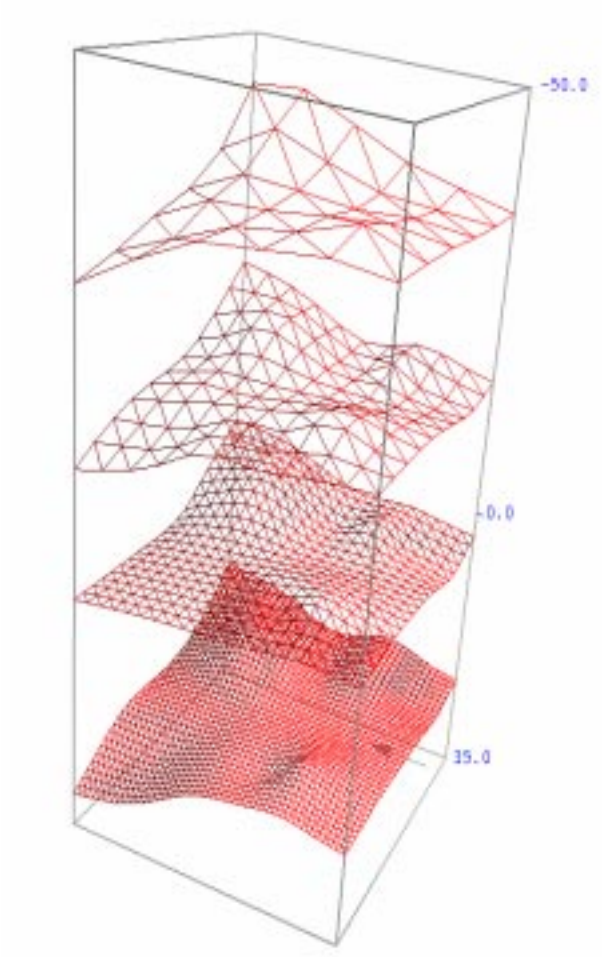


Figure 11.1 Internal surface refinement by the Convergent Gridding algorithm

Contour to Grid

Contour to Grid is a derivative of the **Convergent Gridding algorithm** with the parameters optimized to honor digitized contour data. When generating a grid using digitized contour data as your input, this algorithm will provide the best fit to the contour data.

Least Squares

Least Squares is a general purpose algorithm used for computing a best-fit to scattered data points. It retains the regional trend surrounding grid nodes, while effectively smoothing out some local variation. Although not a good extrapolator, this algorithm is sometimes used for gridding fault surfaces which have little curvature. Before the **Convergent Gridding** algorithm was developed, **Least Squares** was the vanguard gridding algorithm in **CPS-3**.

Moving Average

This is a simple, general purpose algorithm used for computing an average fit to scattered data. **Moving Average** is mostly used for a quick-look or for gridding noisy or statistical data. Use this algorithm when you do not want the surface to contain values which fall beyond the range of the data.

SNAP

SNAP is the building block algorithm incorporated into **Convergent Gridding**. It can be used by itself to grid dense data (3D Seismic, Bathymetry), or for fitting data to an existing grid.

Isopach

Isopach is a specialized version of **Least Squares** or **Convergent Gridding** which treats zero values as surface limits. All positive, non zero-values are honored while zero-values are used to define the zero line for the Isochore.

Trend

Trend is a general purpose algorithm used for computing data trends. You specify the order of the trend (1st, 2nd, etc.).

Polynomial

Polynomial is a special purpose algorithm used for computing fixed value grids as polynomial functions of x and y.

Step

Step is a special purpose algorithm for use in producing lithology, soil or variable hydrocarbon contact maps. Grid node values are set to the **value** of the **nearest** control point.

Distance

Distance is a special purpose algorithm used to quantify the spatial distribution of the data points. Grid node values equal the **distance** to the **nearest** control point.

Density

Density is a special purpose algorithm used for modeling data distribution. Grid values are set to the **number** of data points **falling within** the **Search Limit Radius** (**SLM**) centered on the node.

How Do I Set Gridding Parameters?

In this course, we will focus on the **Convergent Gridding algorithm**. For the purpose of this course, and possibly for most of the data you will be gridding in the near future, there are only a few parameters which you will routinely consider changing.

- **Final grid interval**
- **Initial grid interval** (**Computed***)*
- **Number of Nodes To Snap To** (**16**)

There are many other parameters which can be manipulated in the **Convergent** algorithm, but for the scope of this class, we will highlight these three as being by far the most important.

In the first gridding attempt of any new surface, the recommendation is to determine the **Final grid interval**, just as we discussed earlier in this chapter, and to take the **default values** (shown in bold italics above) for the other two. The only time you **may not** want to take the default value for the **Initial grid interval** is when you know that you have very dense data. If this is the case, you may want to reduce this number to only twice the **Final grid interval**, and also reduce the **Number of Nodes to Snap To** to **2 - 4**.

If, after the first attempt, you are not satisfied with the grid, use the following guidelines for changing one or several of the three parameters above.

Common Gridding Problems and Their Solutions

The following table contains some of the common gridding problems encountered and their solutions.

Problem	Solution
Grid does not honor control points	Make **Final grid interval** smaller
Grid takes forever to compute	Make **Final grid interval** larger or Initial grid interval smaller
Not enough extrapolation beyond data points	Make **Initial grid interval** bigger
Holes in the final grid	Make **Initial grid interval** bigger
Grid too noisy	Make **Final grid** size larger or do a single smoothing operation
Grid does not show features inherent in the data	Make **Final grid** interval smaller

How Do Fault Traces Affect Gridding?

Many surfaces to be gridded are known to be faulted, and the intersections have already been located within the horizon as “fault traces” or “fault boundaries”. Other than for display considerations, it does not matter how a fault boundary is geometrically defined in **CPS-3**. For example, a non-vertical fault boundary could be digitized in any of these ways:

- a series of straight-line segments
- an upthrown polyline and a downthrown polyline
- a closed polygon around the entire fault zone

Regardless of the gridding algorithm chosen, fault boundaries are utilized in the same manner by all algorithms as explained below.

The spacing of the nodes in the figure below represents the 50m x 50m grid spacing which we decided to use for this training data. This figure shows a zoomed-in area of our data, and, for purposes of this faulting discussion, we will assume that the only data we are using for gridding is the well data.

This figure shows how data is selected to be used or not used during the computation of a grid node value, depending upon its spatial relationship to the node being computed and the fault patterns.

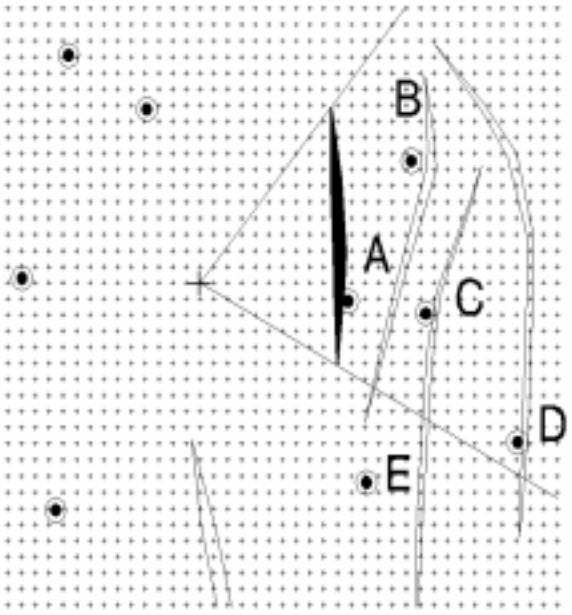


Figure 11.1 Representation of 50m x 50m grid node spacing

Any algorithm in **CPS-3** which is computing the value of a grid node, such as the one marked "+" in the figure above, will treat each fault line segment as a "barrier" during the interpolation process. Data points which can potentially contribute to the value of a node, must first pass a **visibility** test to see if they will be used or discarded. If any point is in the "shadow" of any fault line segment, then it **will not** be used in the computation of the node. For example, in the figure above, points **A**, **B**, **C**, and **D** are in the shadow of at least one of the faults, and will therefore not be used in the computation of that node's value. None of the other points are in the shadow of any other faults, and so they will be used for the computation.

The idea is that only data points which are on the **same side** of a fault as the **node** being computed will be used to compute that node's value. Nodes on the other side of a fault or on another fault block are not used.

Note that this same **visibility** criteria is also used by the **CPS-3** grid-based interpolator during the computation of other manifestations of a surface, such as contouring, volumetrics, and refinement. That is, all grid-based procedures in **CPS-3** are cognizant of where the faults are located and will modify their results based on the faults, as long as they are specified on input.

Gridding Decisions - 2D/3D Seismic Examples

We have discussed how to choose a grid cell size with simplistic well data, and now we provide some guidelines for other geometries. For each of the types of data distributions shown below, we will give recommendations for the **Final grid spacing**, the **Initial grid spacing**, and **Number of Nodes to Snap To**.

2D Seismic

In general, for 2D seismic:

- **Final interval** - roughly the same as the shot point spacing, but be careful not to make it so small as to create noise along the lines
- **Initial interval** - as large as 1/2 the distance between the two furthest-apart “contiguous” lines
- **Number of Nodes To Snap To - 16**

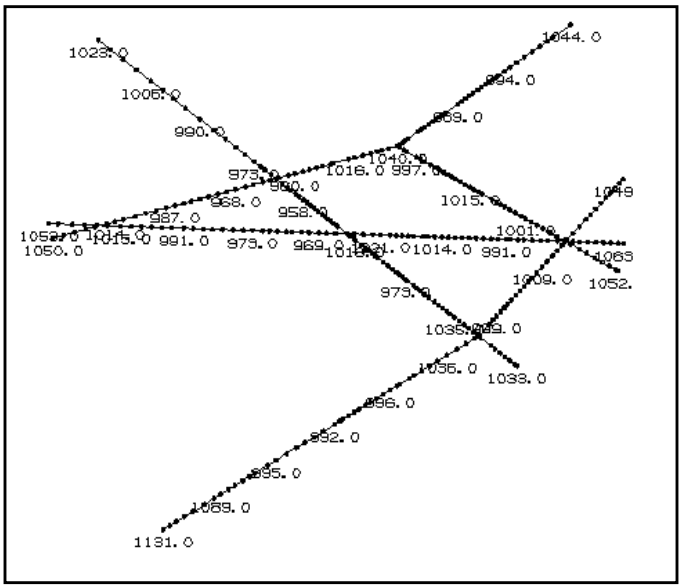


Figure 11.1 2D seismic display

Line-Decimated 3D

The characteristics of line-decimated data is similar to 2D seismic - very dense points along lines which are far apart, relative to the points. This type of data results when the interpreter interprets every 5th, 10th...etc. line.

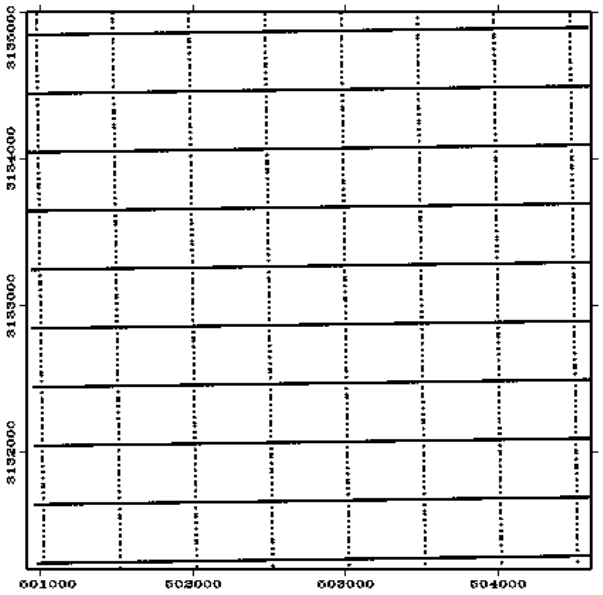


Figure 11.1 Example of Line-Decimated 3D data

For **Line-Decimated 3D**:

- **Final interval** - 1/2 to 1/4 the distance between lines
- **Initial interval** - distance between lines
- **Number of Nodes to Snap To - 4**

Dense 3D

The characteristic of this data is its homogeneous density. In the example below the homogenous pattern is prevalent, but broken by large void areas in the data. Here, the interpreter appears to have interpreted every line or every other line, except in the void areas. Some of the void areas represent faults, others may represent unclear seismic information.

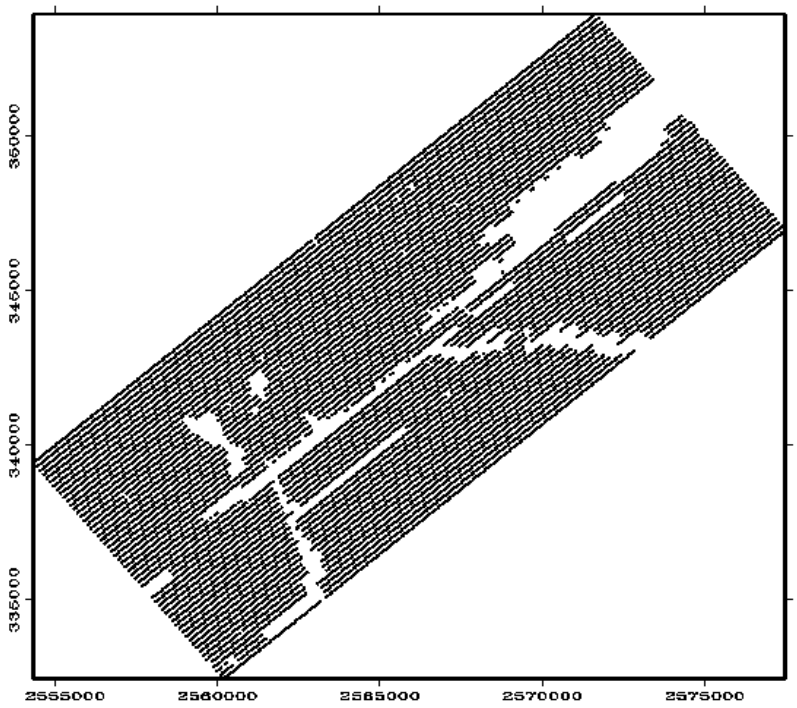


Figure 11.1 An example of dense 3D interpretation

For dense 3D, the first thing to decide is if you really need all the data points which may be present. In some cases, 25m or 50m cdp spacing may lead to grids in **CPS-3** which exceed the reasonable limits. At present “reasonable limits” are determined accordingly:

- 150x150 grid - a modest grid
- 300x300 grid - an average-to-large grid
- 500x500 grid - a very large grid which is manageable, but may cause some delay in processing and require lots of storage if many horizons are involved
- 1000x1000 grid - unless you have an extremely fast server and lots of swap space and lots of storage, grids of this size are not manageable at this time.

For this type of data, the interpretation is typically at a density which is **equal to** or **greater** than the **Final interval** you might choose. For this reason, the proper algorithm to select for gridding is **SNAP**, not **Convergent**. Choosing **SNAP** and setting the parameters as indicated below will produce either a literal copy of your interpretation (without any interpolation taking place), or a sub-sampling of it, depending upon whether the data density is, respectively, **equal to**, or **greater than** the selected **Final interval**.

- **Final interval:** decide based on how much you want to invest in gridding time, grid maintenance time, and storage space.
- **Initial interval:** same as **Final interval**
- **Number of Nodes to Snap To**: **1**

Importance of Fault Zone Definition During Gridding

If your goal involves accurate volume computations between horizons with non-vertical faults, it is important to get a reasonable thickness grid definition in the fault zones. The reason for this is that unless the fault wedge zones for the two horizons are defined in their grids, their resulting isochore grid will not be suitable for accurate volume calculations. If, for example, in the figure below, the Top surface were defined in the grid only in its upthrown and downthrown blocks, and not in the fault wedge zone from A to B, then the thickness grid cannot be defined in the interval from A to B. Similarly, if the Base horizon is blank in its fault zone, the thickness grid will also be blank from C to D. In those locations where the initial thickness grid is blank, no volume will be computed

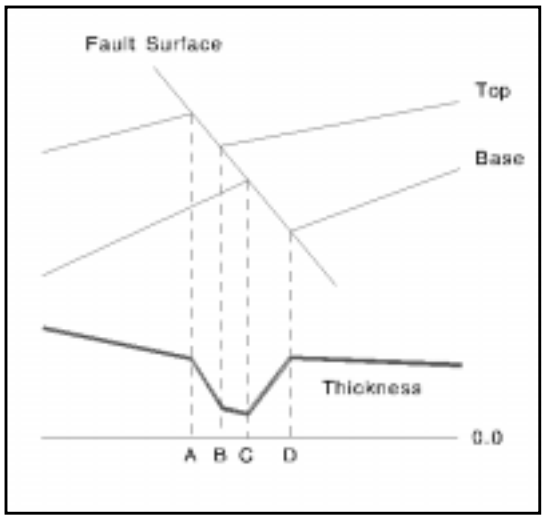


Figure 11.1 Non-vertical fault zone displaying fault wedge zones

As an integral part of the horizon being mapped, the fault zone should receive as much attention as the fault blocks.

Techniques for Filling in Fault Zones

In order to fully define these fault zones, there are several techniques available. A summary is presented below.

Blanking and Regridding

In this technique, you first grid the horizon's fault blocks as well as possible, while using the fault boundaries during gridding. This ensures the integrity of shape for each block. When this initial grid is complete, blank out this grid inside all fault polygons. This cleans the fault zones of any spurious definitions caused by data which happened to fall in the fault zones. Next, copy the blanked grid to a data set, then use that data set to recreate a grid of the horizon with the convergent algorithm WITHOUT using the faults. The theory of this technique is that the best possible data to define the fault zones are the grid nodes right at their edges.

Using Fault Polygon Z-values

If you have quality z-values on your fault boundaries, then this is excellent data to use to define the fault zone. Simply make sure that the gridding algorithm can see and will use these z-values by setting the appropriate switches when specifying gridding parameters. This technique requires only one step, but also requires some quality control for the fault boundaries and their z-values, and assumes there is no conflicting data which might happen to fall in the fault zones.

Using Existing Fault Surfaces

This technique is not an easy one if you have more than a few faults. It does, however, lend itself to being incorporated into a macro. The reason for this is that it requires several operations per fault. Here is the outline of the steps for one fault:

- **blank** the existing fault surface outside of its associated fault boundary polygon
- perform the **surface operation** which replaces nodes in one grid (the horizon grid) with nodes from another (the blanked fault surface). In particular, the surface operation required is the 10th Multiple Surface Logical Operation, "**A in Union**", otherwise defined as

  "C=A, but if A or B is Null, then C = Valid(A,B) (-255)

Contour Visibility in Fault Zones

It is sometimes difficult to tell if grid nodes have been defined in fault zones. When horizontal separation is small, only a few nodes can fit across the heave. In this case, contours may not be generated in the fault zone, even though the nodes are defined. (See the example below.)

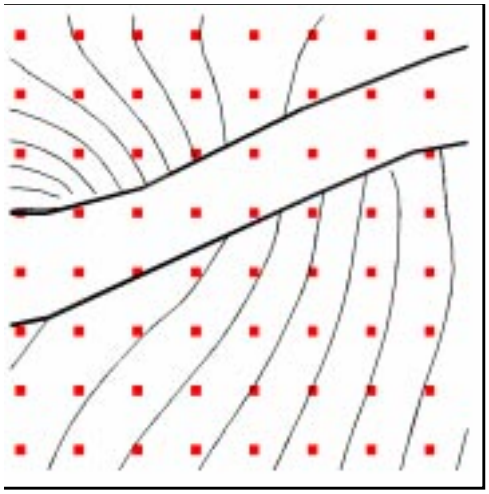
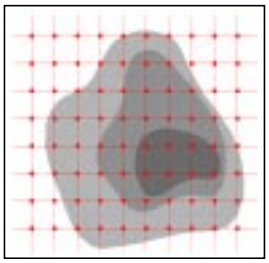


Figure 11.1 Grid nodes across a fault zone

The algorithm’s visibility of grid nodes is sometimes restricted by the upthrown and downthrown traces as it tries to compute inside the fault zone. Not enough nodes are visible to generate reasonable contours. The best way to determine if nodes are defined here or not is to turn off the System switch “Show Graphic Entities When Z is Null”, and display the grid nodes themselves. Null grid nodes will not appear on the display.

LECTURE for Topic 12
Decisions for Gridding

Overview



The purpose of this chapter is to apply the information we learned in the chapter about **Gridding Fundamentals** to specific data sets. We'll discuss the following:

1. Assuming that we will use both well data and seismic data in gridding, we'll go through the decision-making process to settle on a **Final gridding increment** for the sample data sets.

2. We will present some easy-to-remember guidelines for choosing the most important Convergent/Snap gridding parameters - **Starting Grid Interval** and **Number of Nodes to Snap.**

3. We will also discuss the topic of defining the fault zones in a horizon, and when it is necessary.

4. Finally, we'll talk about when it is convenient to grid the fault surfaces themselves, and describe a **macro** which uses a specific technique for gridding fault surfaces.

Selecting the Grid Spacing

The Rule of Thumb

As you may recall from **Chapter 11 - Gridding Fundamentals**, the grid spacing required to replicate all features in a data set is sometimes determined two methods:

- **half the distance between the two closest points**
- **half the distance between the two closest points whose difference you wish to distinguish**

This rule of thumb works fine most of the time for **well** data. However, for very dense seismic interpretation, we may end up with a grid which is too fine. Let us now examine our data in detail to help us pick the appropriate spacing.

Well data

Let’s look only at our well data for a minute. In the map below, several of the wells are fairly close together. The closest are about 50 feet apart.

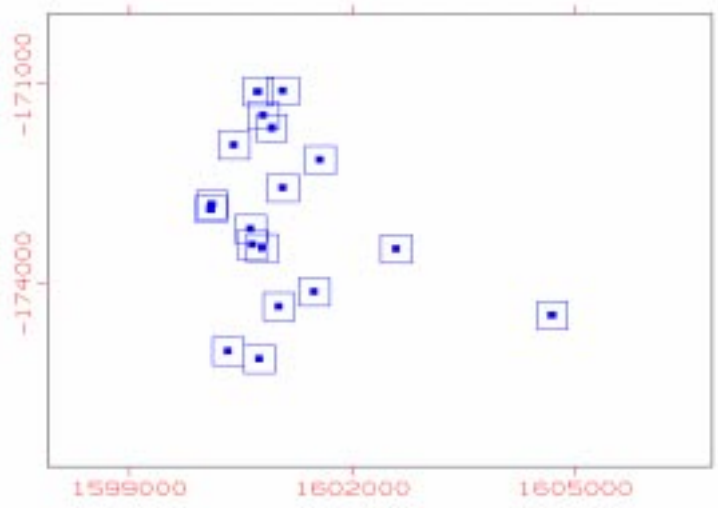


Figure 12.1 Well location map

Things to Consider When Choosing a Grid Interval

In some cases, wells can be so close that choosing a gridding interval, based on their distance, can lead to a grid which contains an inordinate number of nodes. If we choose an **xinc** based on the two closest wells, according to the rule of thumb, we will have a grid with a spacing of 25 feet, 490 columns, and 600 rows.

If this were all the data we had, and our concern was to reflect all data in the grid as accurately as possible, then we might use the 25 foot spacing, especially if the z-values in the two closest wells indicated a substantial difference in slope from the rest of the map. This would guarantee that each data point fell in its own grid cell, and the difference in z-values between the two points would be preserved in the grid and the resulting contours.

If, as is often the case, the z-values of the two wells are not very different, then, for all practical purposes, we only need one of them in the gridding operation. This means that we could ignore their separation as a criteria for the grid spacing and use a larger grid separation.

Here is another way to look at it. There are only two wells which are as close as 50 feet. If there were many wells in the data set, then one might be willing to accept a bit of averaging in the grid around these two wells, if it meant the difference between a modest grid and a very large grid.

Looking at the rest of the wells, it appears that the next smallest separation is approximately 100 or 200 feet. Let us assume that the difference in z-values of the two closest wells is not significant. Then, this inspection of only the well data tells us that we might use a grid interval of about 100 feet. This will provide a surface which is relatively modest in size, but which will resolve any important features inherent in this well data.

Note: *It is appropriate to look closely at the well data when determining a grid spacing, since traditionally, well data is considered to be of a higher quality than the seismic interpretation.*

Let us now look at the seismic data to see if it tells us anything different about the grid spacing.

Seismic Data

As seen below, seismic data comes in a variety of densities and geometries. The one characteristic of all these different interpreted horizons, however, is based on the fact that both cdp and shot point spacing is 55 feet in the **Cloudspin** project. This defines the closest points in the seismic data. Let us see if this affects our chosen grid interval.

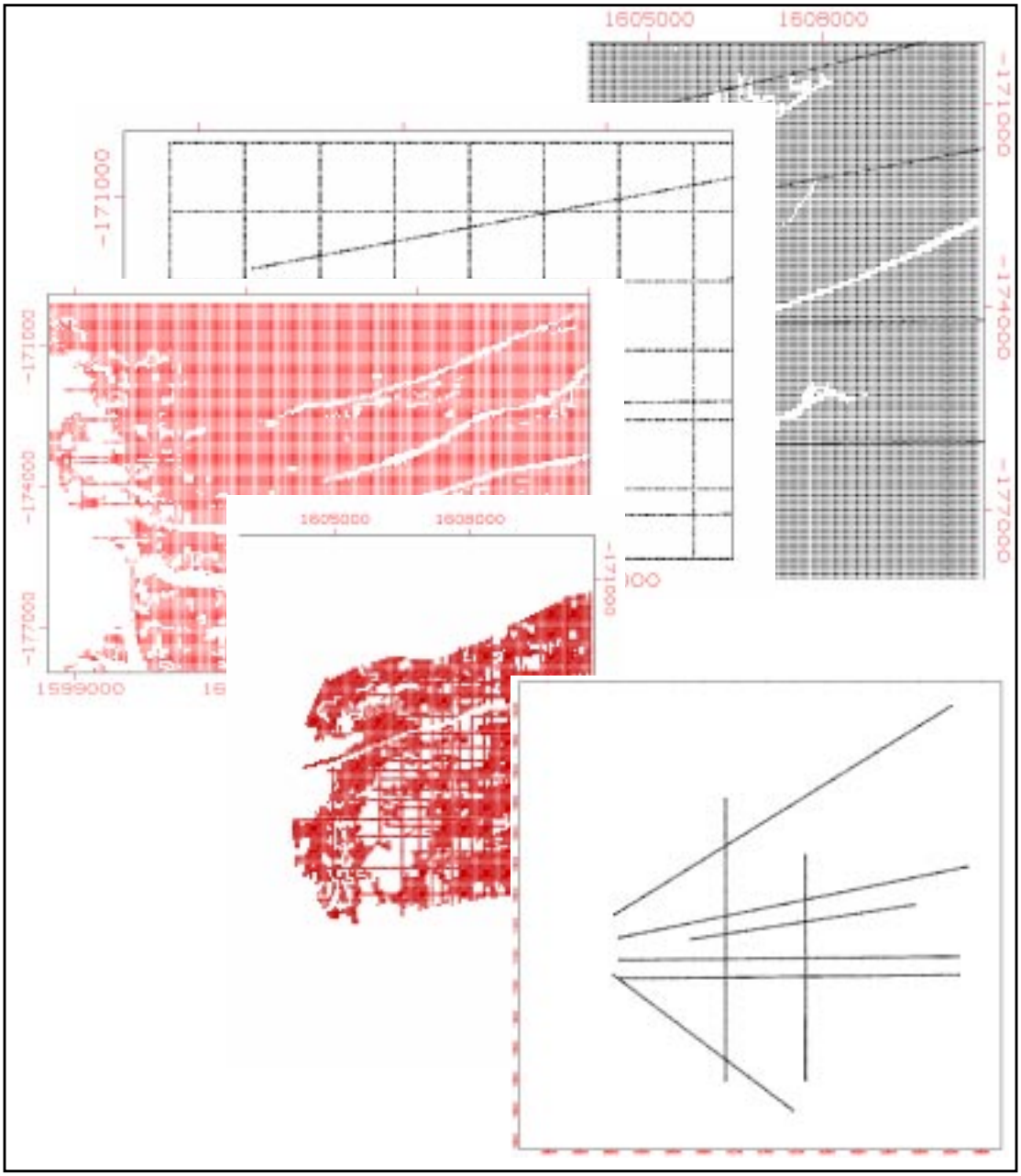


Figure 12.2 Seismic data in a variety of densities and geometries

Varieties of Seismic data geometries in Cloudspin

1. Seed Interpretation

Seed interpretation, or line-decimated interpretation, is interpretation which has been done at a subsampling of the cdp or inline spacing - for example, every 20th inline and crossline, as seen in the figure below. This is a good way to get a horizon interpreted quickly - pick every 20 inline/crossline and let **ASAP** fill in the rest!

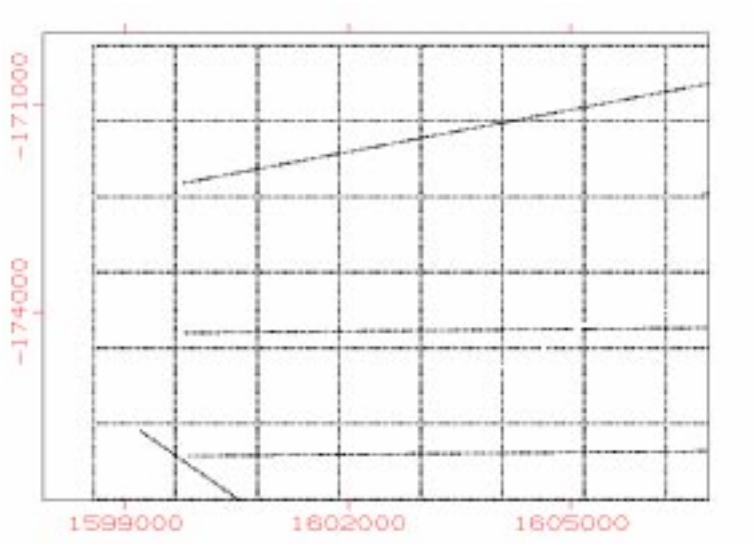


Figure 12.3 Seed interpretation subsampling every 20th inline/crossline

Cloudspin interpreted horizons having **INTRP** in their names fall into this category. This means that the lines are about 1100 feet apart. This particular geometric distribution of data points is characterized by lines which are relatively far apart with respect to the density of points along each line.This type of data distribution sometimes requires extra work in the normally simple gridding operations, especially when a grid spacing is chosen which is close to the distance between points on the lines.

The reason for the extra work is to overcome the tendency of any gridding algorithm to tie the grid closely to the many points along the line, while giving an average solution at nodes between the lines. The following figure shows why the extra step is sometimes needed. It shows a grid computed from **seed interpretation** data where only a simple convergent step was applied. Note the bull’s-eyes along the lines.

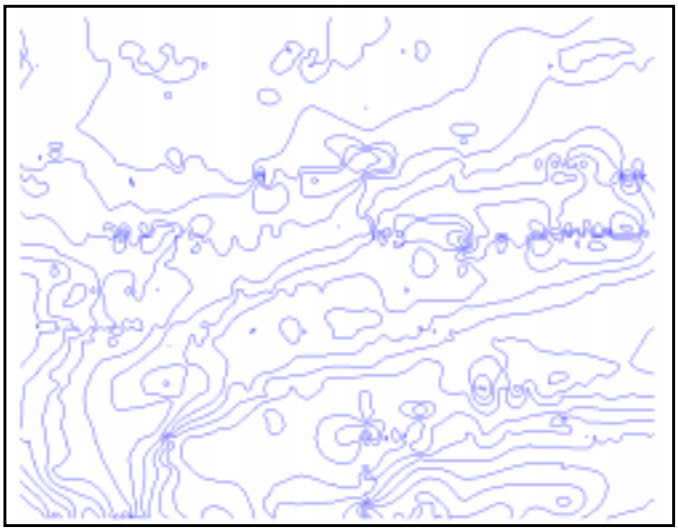


Figure 12.4 Seed interpretation grid displaying noise and bull's-eyes along data

With this type of data, it is sometimes necessary to increase the final grid size in order to reduce the noise and bull's-eyes along the data line. If you have several horizons which must maintain the same x,y extent and grid spacing, this can be a problem, only if this type of data happens to be available for one of the horizons. In this case, the recommendation is to grid the data at the smallest final grid interval which minimizes the bull's-eyes, but then refine the grid down to the smaller required size.

2. ASAP Interpretation

Cloudspin horizons having **ASAP** in their name fall in this category. This type of geometry infers that every cdp in the survey area is potentially defined, as is almost the case in the portion of our **Jakarta** horizon shown in the following figure.

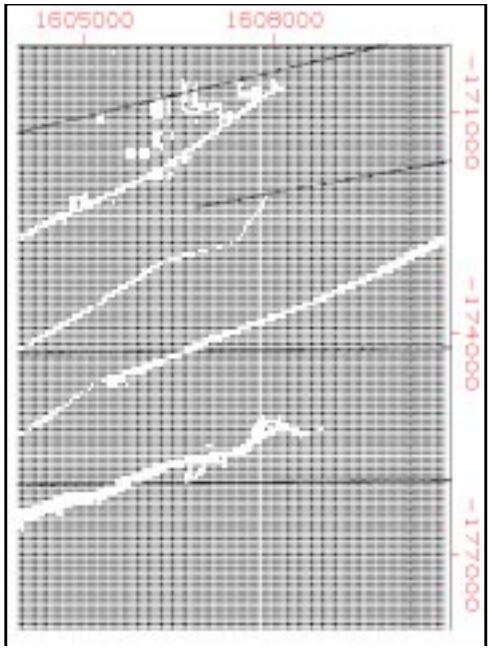


Figure 12.5 Horizon interpretation using ASAP

The **IESX** automatic picking procedure called **ASAP** was used in the above figure to finish the horizon interpretation between the **20x20** lines picked by the interpreter. Dense data like this is usually very simple to grid. However, holes in the interpretation can cause gridding problems. For example, in an **ASAP** version of the **Kobe** horizon below, we see patches of missing interpretation, especially below the Hobart fault in the lower part of the map.

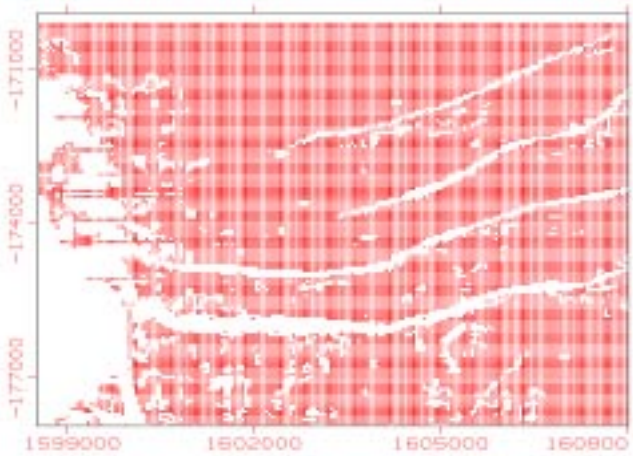


Figure 12.6 Example of holes in the interpretation

When using the **Convergent gridding algorithm**, the key is to make the **Starting interval** large enough to that holes in the grid are avoided. For the **Least Squares algorithm**, make the **Search Radius** larger when unwanted holes in the grid are identified.

3. Combinations

The Paris horizon below contains both **20x20** and **ASAP** interpretation geometry, but it should not be any more difficult to grid than the others. All data distribution types shown here can easily be gridded with the **Snap** and **Convergent Gridding algorithms**, as we will demonstrate.

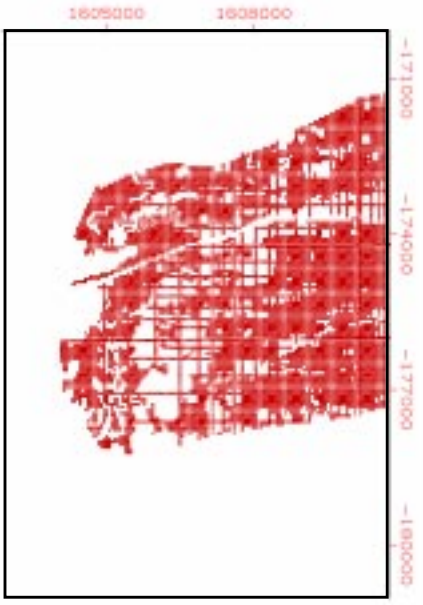


Figure 12.7 Horizon displaying 20x20 and ASAP interpretation geometry

4. 2D Only

This type of data is similar to the seed distribution type, in that, data exists in very dense points along lines which are sparsely distributed. As with the seed data, the potential exists for bull's eyes along the lines if a too small **xinc** is used initially. Either post-refinement of a somewhat larger final grid size, as mentioned above, or smoothing of a required final grid size can help here.

5. Other Seismic Considerations

There are many options available in seismic interpretation packages, and some can affect the way in which data is gridded in the mapping stages. For example, in the **Cloudspin** project, not all horizons were interpreted with the **automatic smoother** activated. We will see one or more of our horizons exhibit high-frequency noise in the initial contours. A small bit of smoothing in **CPS-3** can remove this noise with little affect to data tying.

Conclusion of the data inspection

We chose a grid spacing of 100 feet based on the well data, but note that for the densely-interpreted seismic horizons, many of the data points are 55 feet apart. The choice of a **50 foot** grid spacing would give us a grid having 225 columns and 280 rows. This is not an extremely large grid, and **50 feet** would be an appropriate spacing if this reservoir were modeled with the intent of computing accurate oil in place.

Simple Guidelines for Choosing SNAP/CONVERGENT parameters for Seismic data

1. If the **seismic** interpretation data is **dense** enough that interpretation exists in every grid cell of the selected gridding lattice in all locations of the map where grid definition is desired, then use SNAP with Number of Nodes = 1

2. If the interpretation contains **holes** or does not exist in areas where the grid must be defined, then use the CONVERGENT algorithm with Number of Nodes = 16, and Starting Grid Interval = half the diameter of the largest hole in the data.

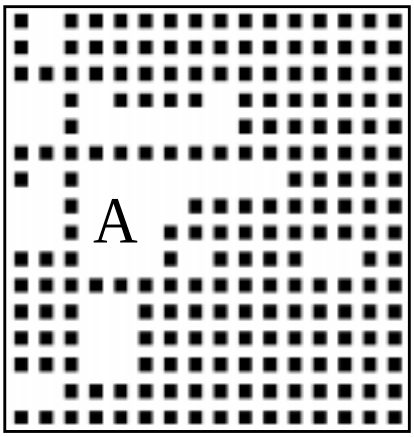


Figure 12.8 Incompletely defined seismic interpretation

For example, if the figure above represents 3D seismic interpretation, the Starting Grid Interval should be at least several times the distance between the data points in order for the grid to fill in the holes in the data.

Defining the Fault Zone in a Horizon - Yes or No

As you prepare to create grids for your horizons, you should think ahead to the end of your particular workflow. If your only concern is to create a structure map, it is common practice to blank out the fault zones on the contour map. You can do this by **blanking the grid** inside of the fault zone, or you may choose to do it graphically, by simply **color-filling the fault zone** during display.

As you will see in the exercises, defining the fault zones in a horizon ***does not require the presence of a surfaces for the faults***, which is discussed in a different context below.

Importance of Fault Zone Definition

In horizons with non-vertical faults, it is very important to get a reasonable thickness grid definition in the fault zone if your goal involves accurate volume computations. The reason for this is that unless the fault wedge zones for the two horizons are defined in the grid, their resulting isochore grid will not be suitable for accurate volume calculations. If, for example, in the figure below, the Top surface were defined in the grid only in its upthrown and downthrown blocks, and not in the fault wedge zone from A to B, then the thickness grid cannot be defined from A to B. Similarly, if the Base horizon is blank in its fault zone, the thickness grid will also be blank from C to D. In those locations where the initial thickness grid is blank, no volume will be computed

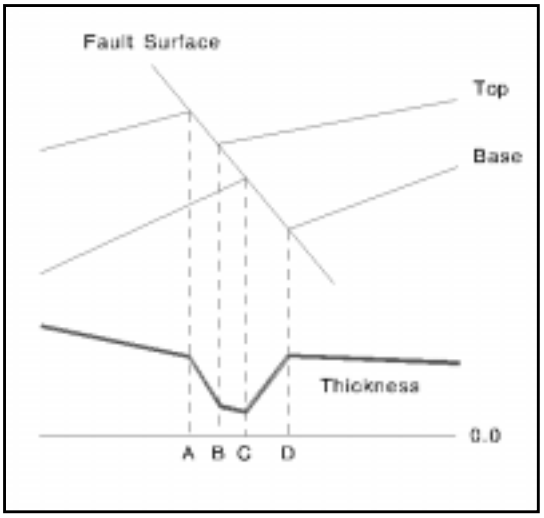


Figure 12.9 Non-vertical fault zone displaying fault wedge zones

Contour Visibility in Fault Zones

It is sometimes difficult to tell if grid nodes have been defined in fault zones. When horizontal separation is small, only a few nodes can fit across the heave. In this case, contours may not be generated in the fault zone, even though the nodes are defined. (See the example below.)

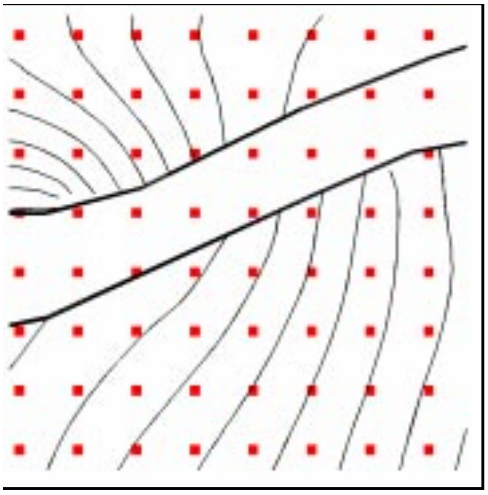


Figure 12.10 Grid nodes across a fault zone

The algorithm's visibility of grid nodes is sometimes restricted by the upthrown and downthrown traces as it tries to compute inside the fault zone. Not enough nodes are visible to generate reasonable contours.

Tip: *Even though display functions like contouring may not be able to visually render every portion of a gridded surface, it is still important that the nodes be defined properly for structural reasons.*

When Are Fault Surfaces Needed?

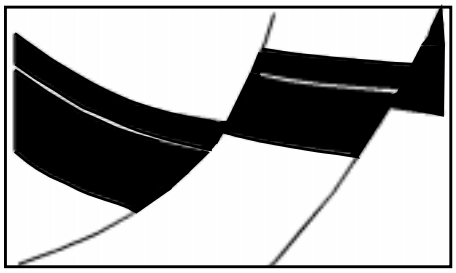


Figure 12.11 Normal faults in profile view

Then answer to this question depends on how much detail you want in the structural envelope you create for volumetrics. If your structural envelope does not involve sealing faults, then you probably don't need to grid your fault surfaces. If the envelope does involve one or more sealing faults, but having very modest **horizontal** and **vertical separation**, you may still decide not to grid the fault surfaces, and simply treat the faults as ***vertical faults***. However, in the case of large throw or large heaves on the sealing faults, you will probably get more accurate results in your **volumetrics** if you include the fault surface in the structural envelope.

A Predefined Technique for Fault Surface Gridding

Here, we'll discuss a **technique for gridding fault data** which may not be necessary for all fault data sets, but which is a very useful alternative to **Convergent** gridding. The origin of this gridding technique is the realization that most fault surfaces have strong linear components in the direction of dip. With the typically sparse data provided by fault segments, as interpreted in the seismic, strong linear trends are not always honored by the **Convergent** algorithm. The gridding technique consists of the following:

- Create an initial 2nd order **Trend** grid from the fault data points.
- At all data point locations, compute the **difference** between the **Trend** surface and the z-value in the data point.
- Subtract the two values, creating a new z-field called **Error**.
- Create a grid of the **Error** and **add** it to the initial **Trend** grid, giving the final grid which contains a strong fault-like trend downdip, but also ties to the observed data.

This technique is contained in the macro called **k_grid_fault.mac**

LECTURE for Topic 16
Fault Surfaces

Overview

In this chapter, we will turn our attention to another step in the workflow whose culmination will be the computation of oil in place in a reservoir

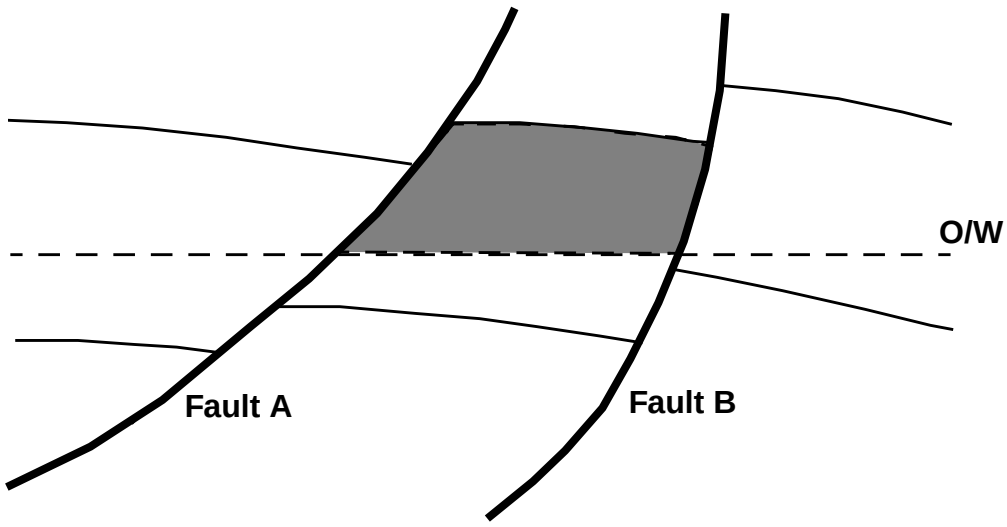
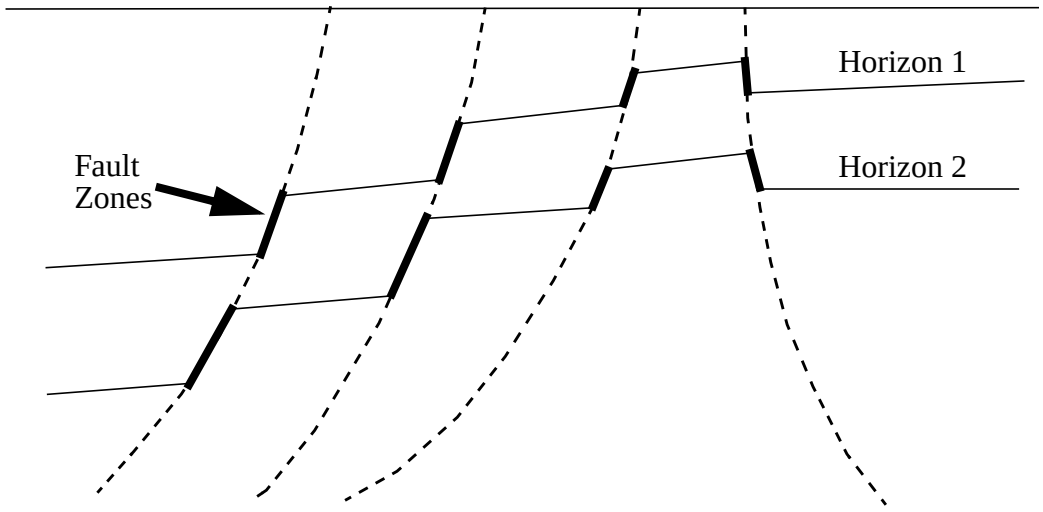


Figure 16.1 Sealing Faults

In those cases where a reservoir is bounded by sealing faults, it will be necessary to make grids of the fault surfaces, unless they are actually vertical.

We have already discussed the importance of defining the fault zones when volumetrics is the workflow focus, and we should have already accommodated this requirement when gridding the top and base structures as depicted below.



We are careful when creating grids to be used in volumetrics to define all fault zones in each horizon. There are many techniques for doing this as discussed in a previous chapter, and not all of them require an actual fault surface. The definition, during gridding, of all fault zones should be done for volumetrics whether there are any sealing faults or not. It simply guarantees that the top and base envelope will be continuous and that the resulting isochore will not have any holes in it due to undefined fault zones

Creating Fault Surfaces

In this chapter, we continue with modeling requirements in the case of sealing faults, where one or more fault surfaces must be integrated into the top or base envelope, or both.

The technique is to simply define the fault surface as well as possible with whatever data is available. This data could be

- scatter points
- fault cuts (segments) from seismic interpretation
- digitized contours
- external grid

There are many algorithms to choose from when gridding faults, just as there are when gridding any data. One of the differences between typical horizon interpretation data and fault interpretation data is that the fault data tends to be sparser and of a lower quality. For this reason, it is not always possible to obtain a good model of the fault on the first iteration.

Try the Convergent algorithm first, or any other algorithm you choose. If you like the resulting grid and it honors the data, then that's probably as far as you need to go.

In the GeoQuest modeling system called Framework 3D, faults are gridded many at a time, and a specific algorithm is used there which has a high probability of creating a good grid on the first try. This algorithm can be "duplicated" by several CPS-3 procedures and embodied in a macro. There is another algorithm which has already been put into a macro which uses even another technique. Both of these are discussed below.

Both of these techniques work well for. Be aware the output grid may not tie to all of the data points, but it works well for fault input data which is either very noisy or very sparse. Below is a description of it.

Predefined Techniques for Fault Surface Gridding

Trend Method

This is a technique for gridding fault data which may not be necessary for all fault data sets, but which is a very useful alternative to **Convergent** gridding. The origin of this gridding technique is the realization that most fault surfaces have strong linear components in the direction of dip. With typically sparse data, strong linear trends are not always honored by the **Convergent** algorithm, nor ingrained in the fault cut data set.

This particular technique is directly available in the CPS-3 set of System macros. It's called "GridFault". Here is an outline of how it works:

- Create an initial 2nd order **Trend** grid from the fault data points.
- At all data point locations, compute the **difference** between the **Trend** surface and the z-value in the data point.
- Subtract the two values, creating a new z-field called **Error**.
- Create a grid of the **Error** and **add** it to the initial **Trend** grid, giving the final grid which contains a strong fault-like trend downdip, but also ties to the observed data.
- Display the **contours** for the fault grid
- Display the original **data points** for the grid.

Slope Method

In FW3D, a new fault gridding algorithm for GF4.0 has been installed which, in most cases, gives better results than the ***Trend Method***. It may not be in the CPS-3 set of System Macros, depending of the version of your software, but it works like this:

- Use the data you have to create a fault surface using the Convergent gridding algorithm for a grid which is 4 times a coarse as the desired final grid size.
- Use the Control Point operations to compute slopes at all control points so that you have an augmented control point set containing X,Y,Z,dZ/dX, dZ/dY.
- Use the augmented control point set to create the fault grid at the final grid size.

LECTURE for Topic 18
Computing a Volumetric Envelope

Overview

The first step in computing oil in place within an interval is to define the top and the base of the oil-bearing rock. If the top and base horizons were flat surfaces, cleanly defined across the entire area, then the gross rock thickness could be computed by simply subtracting the two surfaces. However, we must consider the common case where both of these surfaces intersect, or at least onlap or baselap an unconformity or other bounding strata. In addition, we will also consider the common condition where a fault surface seals the envelope along one of its boundaries as well. These surface-to-surface interactions mean that we must perform other mapping operations to create an envelope which surrounds only the oil-bearing rock in this interval.

Since we must also account for the presence of water and gas within the structural envelope, one of the last steps will be the integration of the gas/oil and oil/water contacts into the final envelope.

Here, we will outline all the steps required to compute a volumetric envelope. These steps should also work for any reservoir that you encounter outside of class.

Recommended Sequence for Computing an Isochore for Volumetrics

The following steps should be taken when computing an oil-only isochore between two horizons. This procedure is designed to preserve the true location of the zero line in the isochore. Refer to following section, **Location of the Zero-Line in Isochores**. Z-units are assumed to be in **depth**.

Procedure: Derive top structural envelope

1. Identify the top of the interval for which the isochore is to be computed.
2. If the top is intersected by any unconformity, merge the two grids, retaining the **deeper** portions of both.
3. If the top intersects any other sealing features, merge the two appropriately.
4. Merge the result of **step 2** with the gas/oil contact, if any, retaining the **deeper** portions of both.

Procedure: Derive the bottom structural envelope

1. Identify the base of the interval for which the isochore is to be computed.
2. If the base intersects any **lower** sealing surface, including faults, merge the two, retaining the **shallower** portions of both.
3. Merge the result of **step 6** with the oil/water contact, if any, retaining the **shallower** portions of both.
4. Subtract the top envelope, the result of **step 3**, from the base envelope, the result of **step 6**, giving the final isochore.

Tip: *One important goal of these operations is to end up with a top envelope and a base envelope which **cleanly intersect,** but do not overlap (are not coincident) along their **edges**.*

Location of the Zero Line in Isochores

Isochores, created for the purpose of contouring or volumetrics, should preserve the location of the zero line in the grid by containing negative values on the other side of the zero line. This does not have to be artificially introduced, since it can be a natural by-product of the process of creating the structural envelopes. If the isochores are left **clipped to zero**, as can happen when surfaces are prematurely truncated, it effectively ***moves*** the zero contour line, ***shrinking*** the perimeter of the positive isochore. The contouring and volumetrics algorithms will both do a much better job when isochores are **not clipped**. You should, however, clip isochores to zero when using them for surface operations, such as adding them to a top or base structure.

A visual symptom of isochores which have been clipped to zero are ragged contours along the zero line. See example below which exhibits a wobbly zero contour in the north.

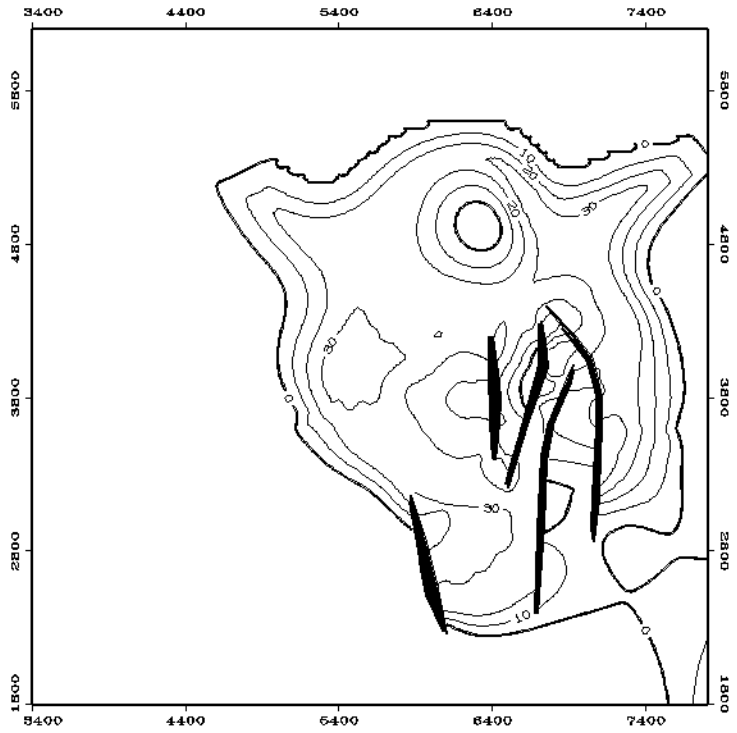


Figure 18.1 Isochore clipped to zero displaying ragged contour along zero line

In this example, the clipping was not explicitly performed. It was the result of premature merging or truncation of surfaces, as is explained below.

How Did the Top and Base Envelope Become Coincident?

The profile below shows a top envelope and a base envelope which are the origin of the isochore contoured above. There is a section along the profile on the left where both surfaces are coincident. This is what is causing the flat zero area in the isochore, and the loss of the true location of the intersection. The coincidence in the envelopes was caused when the two original horizons were truncated along an unconformity at some step along the way. While these envelopes are geologically correct, they are not formed to gain the best results from volumetric calculations.

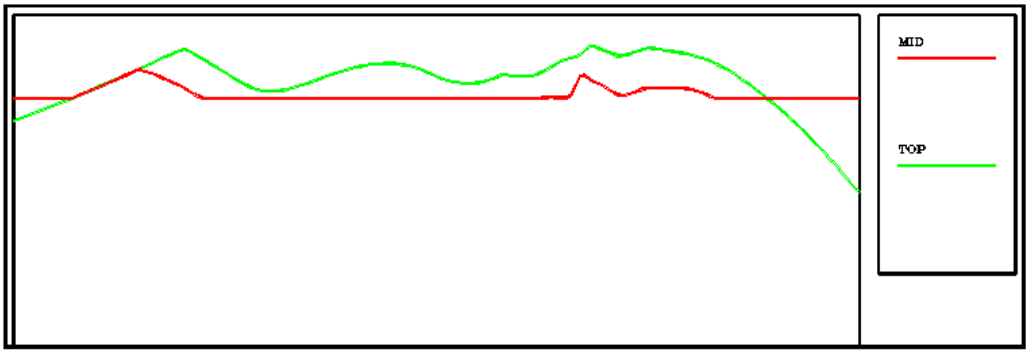


Figure 18.2 Top envelope and base envelope of isochore in Figure 18.1

If the preceding **Preferred Sequence of Operations** had been followed, this loss of volume would not have happened, because the base envelope would not have been merged with the upper unconformity. Only the top envelope interacts with the upper unconformity.

Accounting for Non-vertical Fault Discontinuities in the Volumetric Isochore

For the most accurate results involving faulted horizons, an extra step is needed for the preparation of the fault boundary sets used during the volume calculations. Thinning of the isochore is the reason for this extra step, and is shown in the following figure. **A** is the location of the upthrown side of the fault boundary for the Top. **B** is the location of the downthrown side. **C** and **D** are the corresponding locations for the Base.

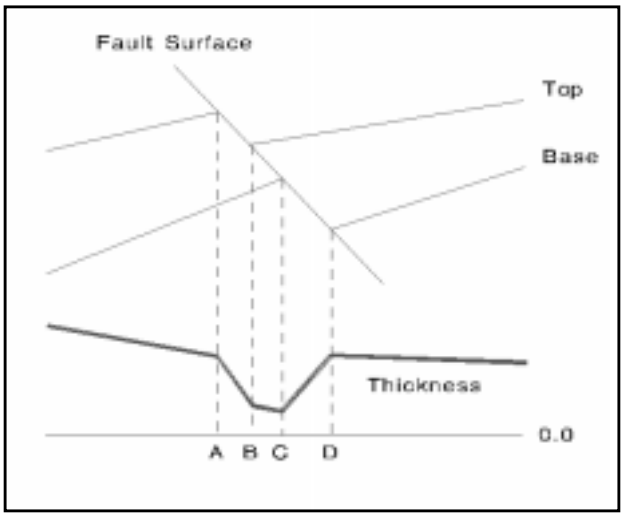


Figure 18.3 Non vertical zone displaying wedge zones

As you can see, the thickness grid resulting from the subtraction of two faulted structure grids, is, itself, faulted, or at least discontinuous in z in the fault zones. It requires the boundary sets of both of the structure grids to separate one discontinuous zone from another, and to allow the algorithm to compute the most accurate results. For this reason, we will prepare fault sets to use during the volume calculations for both intervals. Each prepared fault set will be the combination of the fault traces for the top and bottom of the associated envelope. We will do this below with a simple **Copy/Merge** operation for each isochore.

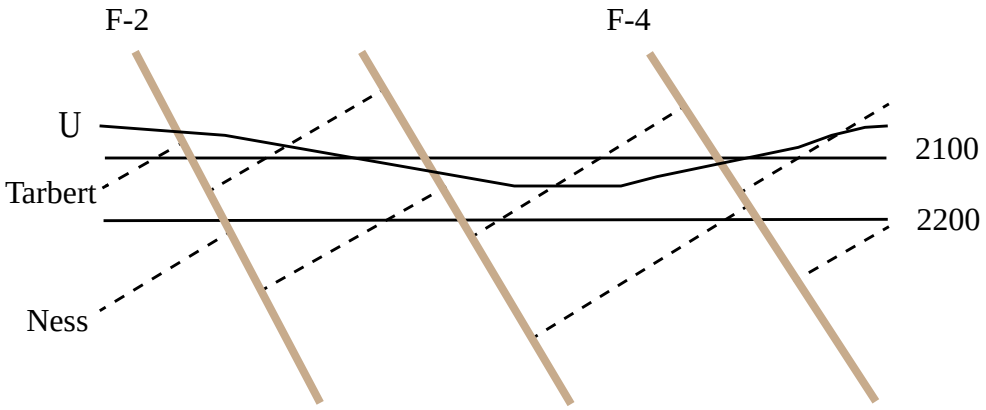
It should be noted that if the fault zones are large, fault surfaces should normally be used as part of the structural envelope, if available, and if the faults are sealing.

Example of Creating a Structural Envelope

We use an example from the GullFaks field. Below is the diagram of a typical section through a reservoir. Our job at the moment is to identify those structural components which contribute to the TOP of the reservoir. First, we recognize that the F-2 fault will become the Western edge of the BASE of the envelope, and so this gives us a convenient starting place for identifying the top of the envelope.

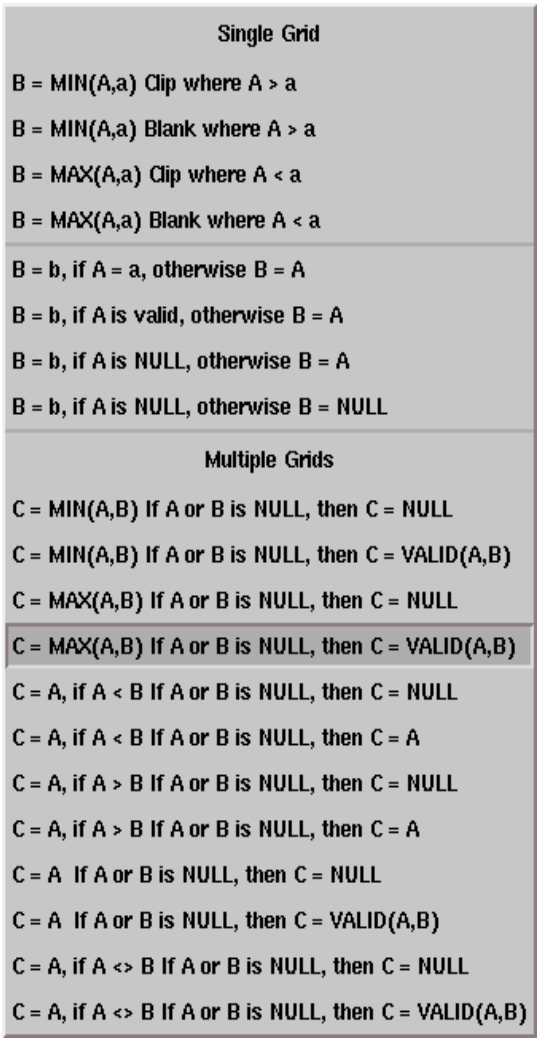
Starting just to the right of the sealing F-2 fault, at the Tarbert horizon, we will examine each intersection which occurs, noting which of the two intersecting grids is the stratigraphicly LOWEST at the right of the intersection. That one which is lower then becomes the upper boundary. For example the first intersection with the Tarbert is along the 2100 g/o contact. Moving to the right, the g/o contact then intersects the Unconformity, which now becomes the boundary. Continuing on in this manner we can see that, excluding the faults, the top of the reservoir is formed by sections of the Tarbert, the 2100 o/w contact, the Unconformity, and the F-4 fault.

Therefore, it is these four grids which we must merge to form the top.



Merging takes place two grids at a time, using a CPS-3 logical operation which will take the stratigraphicly lower value of the two grids at each node location. When one grid does not exist, output takes the other value.

Note that in all cases, the concept of "**maximum**", "**minimum**", "**highest**", and "**lowest**" in CPS-3 are to be taken ALGEBRAICALLY, not GEOLOGICALLY. This means that you must consider the ***Z-units of the grids*** which you are manipulating. If your units are in elevation, rather than depth, then your choice of operation from the table below will be different.

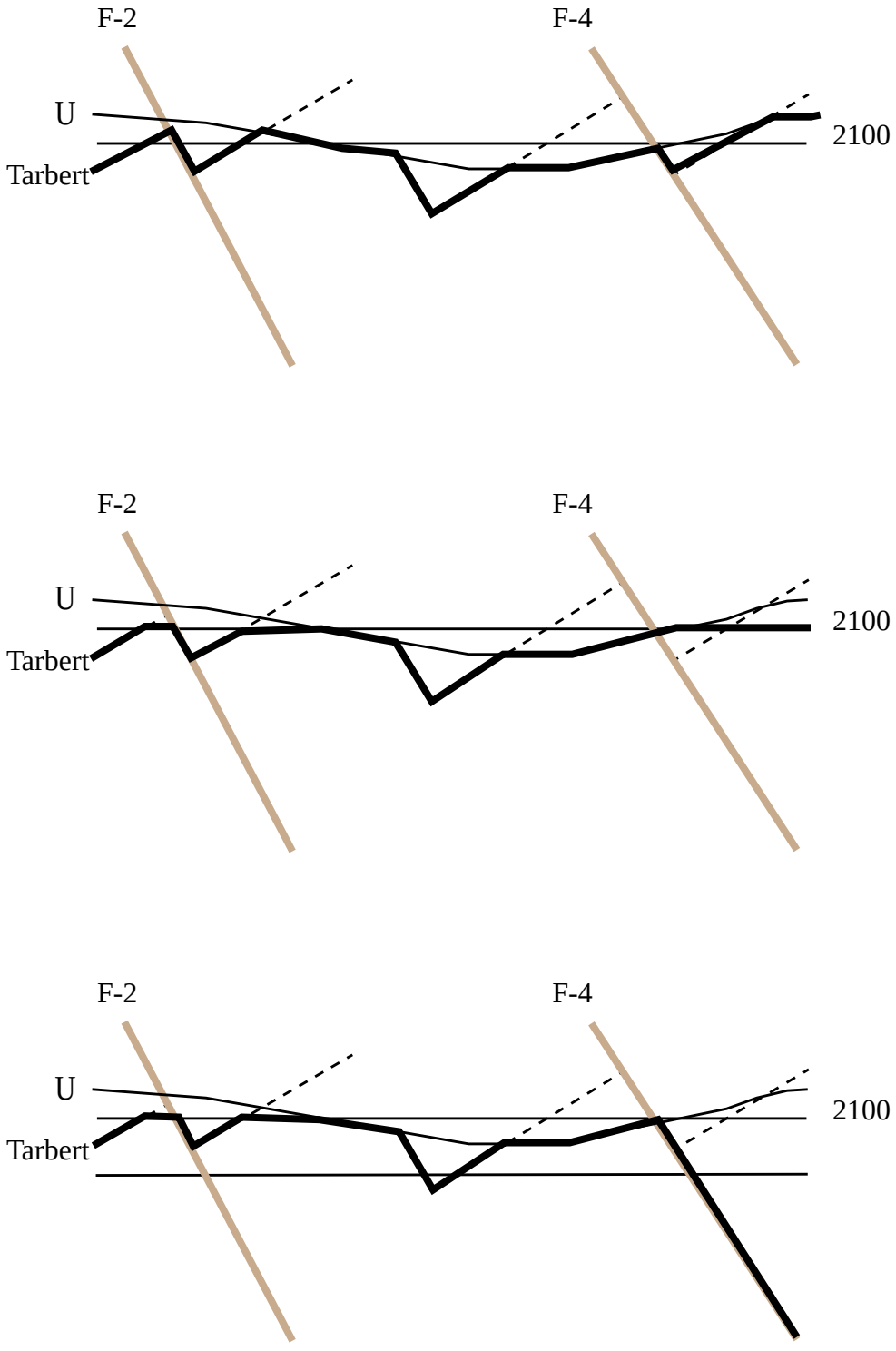


The operation highlighted in the dialog above is the one we want for merging the top envelope components, since we are in depth. When units are in depth, the **lower** component will have the **higher depth** values.

Luckily, it does not matter which order in which these operations occur, the result will be the same. Let’s merge these components in the following order:

- Merge the **Tarbert** with the **Unconformity**
- Merge the result with the **G/O contact**
- Merge the result with the **F_4** fault

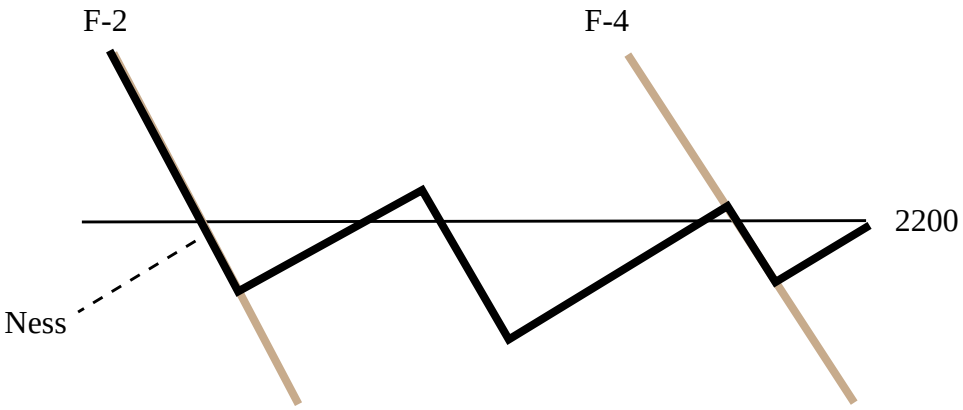
  See each result below


F-2
F-4
U
Tarbert
2100
F-2
F-4
U
Tarbert
2100
F-2
F-4
U
Tarbert
2100

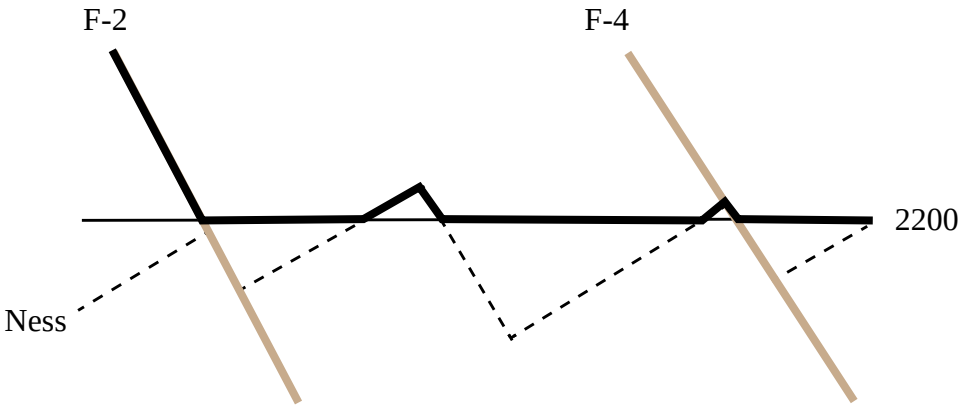
Now for the bottom of the envelope. We can see that its components are the F_2 fault, the Ness horizon, and the O/W contact.

In the same manner we can merge these components, using a different logical operation, one that takes the minimum (higher values in depth) from the two grids at each step. We'll merge in this order:

- Merge **F_2** fault with **Ness**, giving the result below

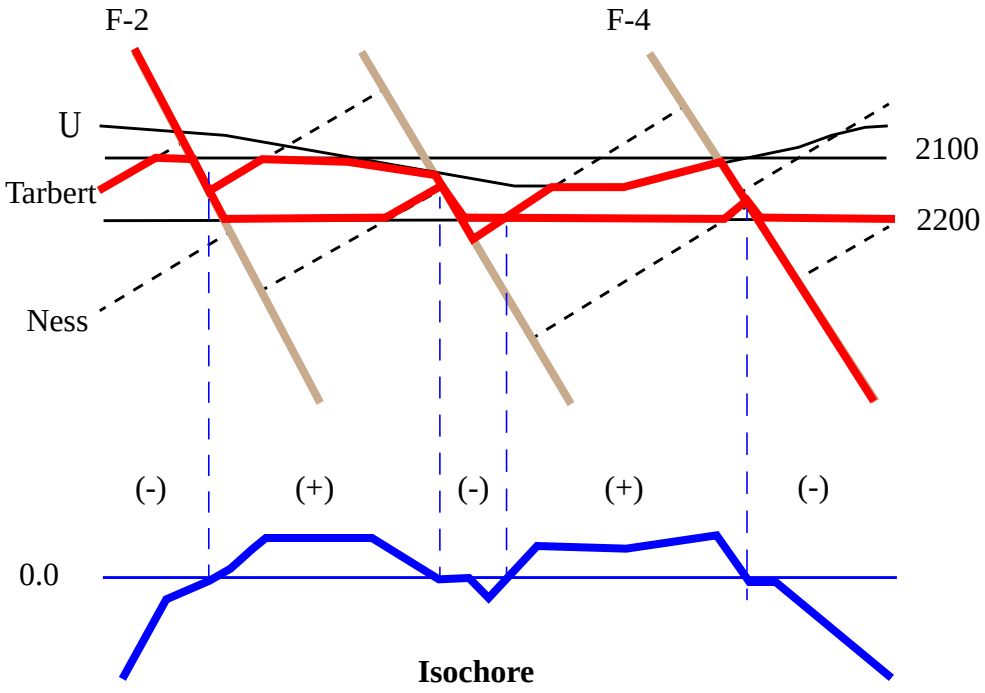


- Merge result with **O/W contact**.



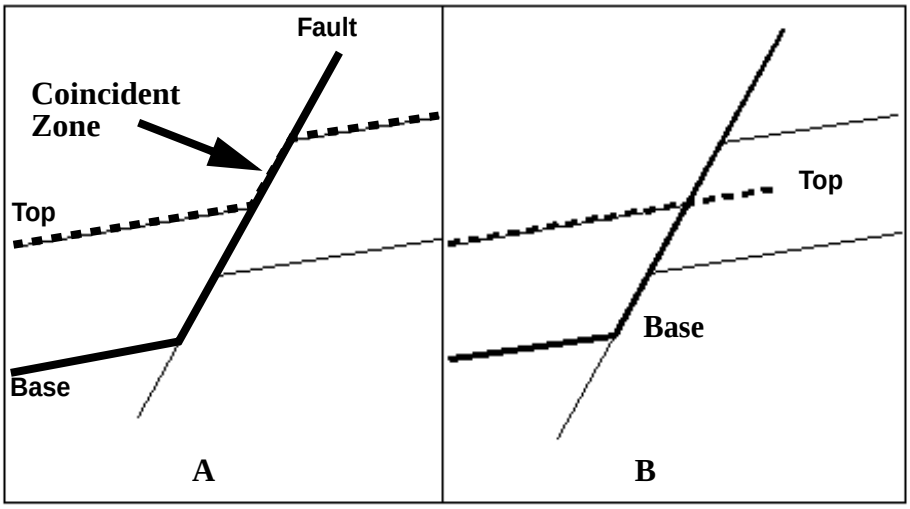
The next step, of course, is to make sure that the top envelope and the base envelope cross at the edges, so that their subtraction will result in positive isochore, where needed, as well as negative isochore where needed.

Below, we have superimposed the top and base envelopes.



We see that top and base envelopes cross well, but we see some characteristics which may not be what we want. We note the small negative component in the isochore in the middle of the map, but realize that this will not cause any problems and can be disregarded. However, the top and the base of the isochore have three areas where they are **coincident.** We would like to avoid this, if possible. Therefore, when the isochore map is created, we must examine the extent of these resulting flat zero areas. They may be small and insignificant, but it is also possible that the isochore may need some repair.

Let’s see how these coincident areas occurred and what we can do about them.



In the case of a bounding fault grid which forms one side of the reservoir, a potential problem exists in the isochore. If the Fault is merged with the Base to form the bottom envelope, and the Top is left by itself to form the top envelope as in Case A, then when they are subtracted, there will be a small zone where the top envelope and bottom envelope overlap, causing a flat zero area in the isochore grid, where, instead, there should be an abrupt change to negative values. In many cases, there areas are very small and insignificant, but, in others, must be addressed. On way to address this problem is to edit the top envelope so that it does not coincide with the base envelope, but cuts cleanly across it, as in Case B.

LECTURE for Topic 20
Applying Reservoir Properties to the Gross_Isochore for Oil in Place

Overview

In this chapter we will learn how to gain access to the necessary **reservoir property** data from **GeoFrame** so that we can compute oil in place according to this simple formula:

Oil in Place = **Gross Isochore Volume* Net/Gross * Porosity * Saturation**

We'll discuss the origins of the Zone properties, as stored in GeoFrame, and demonstrate the calculation of each of the **property grids**, as well as the series of **volumetric grids**:

- Net Isochore
- Net Pore Volume
- Net Pay

Finally, we'll demonstrate the use of the volumetric procedure and discuss the report which it produces.

Origin of property data used by CPS-3

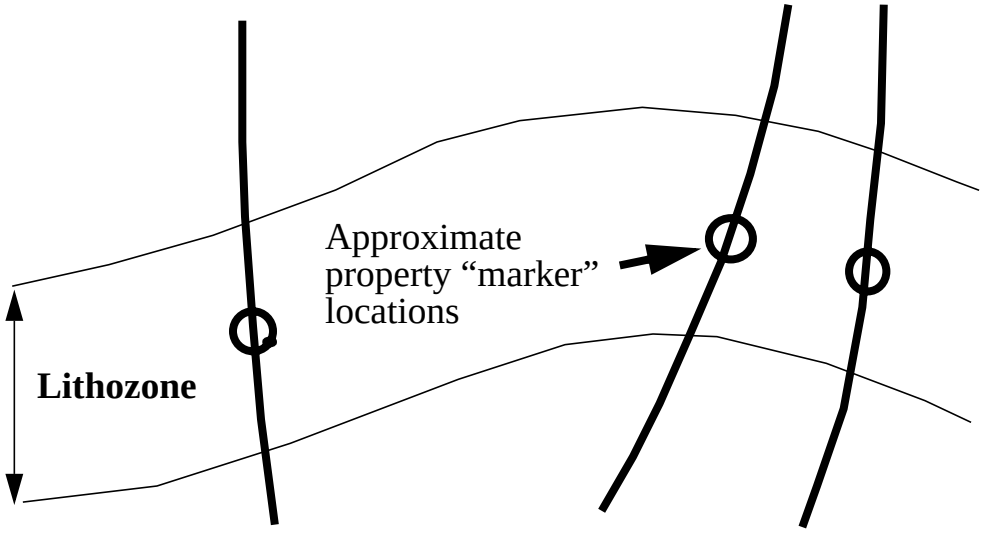
The origin of the property data used in volumetrics are the well logs. Reservoir engineers or petrophysicists determine accurate values for many of the reservoir properties such as **porosity** and **saturation**. In GeoFrame, the tools required to do this are found on the Geology and Petrophysics catalogs. Applications such as **BoreView, PetroViewPlus, WellPix, Stratlog, and ResSum** all help to define these property values.

Ultimately, average property values are computed by **ResSum** for properties in each **lithozone**. There can be many "versions" of these average calculations, and so they are grouped and categorized by **Zone Version**. When we compute volumes in CPS-3 for a particular interval, or lithozone, we need to make sure and extract the property values from the proper Zone Version. Zone Versions allow different interpretations to exist for the same lithozone. For each property calculation within a specific Zone Version, ResSum provides accumulations of the value based on different geometries - for example, according to True Vertical Thickness (TVT), Measured Depth (MD), and others. Where available, TVT or **TVD** computations should be chosen for volumetrics.

Let's assume that we wish to compute Oil in Place between two horizons named Jakarta and Kobe. A typical GeoFrame workflow for computing the required properties in ResSum is:

- **Load** the appropriate ***well logs***
- **Create** the geological ***markers*** for the Jakarta and the Kobe, either in **WellPix** or by loading them with the **General Ascii Loader**
- In **Well Pix**, **define** a ***lithozone*** between the Jakarta and the Kobe and establish a ***Zone Version***.
- In **ResSum**, **calculate** the ***ratios*** and property ***averages*** for the layer.

When imported into CPS-3 via the GeoFrame link, each property value for a particular Zone Version will take on the characteristics similar to a **set of markers**; that is, a set of scatter points based on the well paths. The X and Y values for these property values do not sit at the top or base of the lithozone, but, rather, in the middle of it as seen below, depending on the particular computational geometry chosen (TVT, MD, ...).



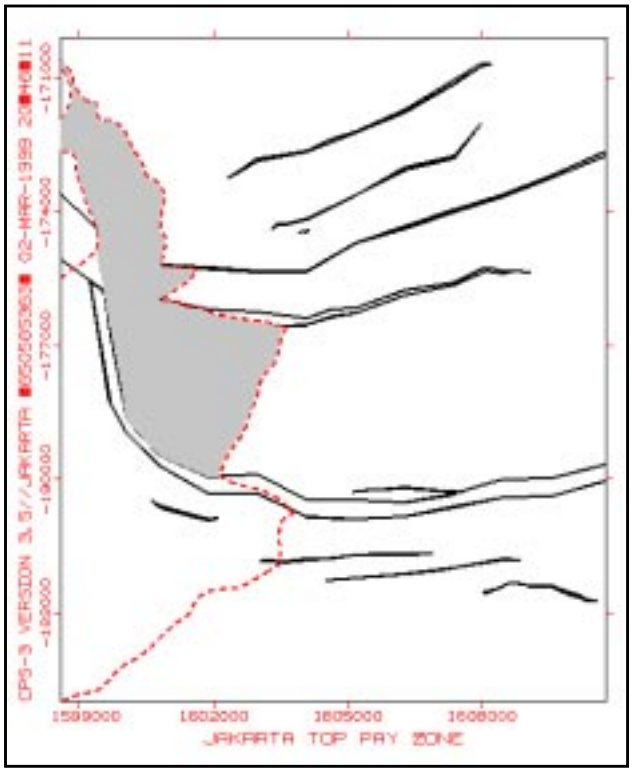
Zone Property Values and Grid Calculations

Typically, the following ResSum properties are the ones used in the volumetric equation for CPS-3:

Net Thickness, Gross Thickness, Net Porosity, and Net Pay Water Saturation.

Each is retrieved as a **scatter set** from the GeoFrame data base with the **GFLink**. and then gridded. These property grids can then be applied to the gross isochore with CPS-3 surface arithmetic operations.

Quality and Characteristics of Property Grids



In the example shown, note that the gross isochore (shaded area) covers only a portion of our **Area of Interest** (**AOI**). When we look at the quality of each property grid we create, we will not be concerned with areas which are not within this pay zone. Within the pay zone, however, there are certain criteria which must be met:

- **Porosity** and **Saturation** values **must remain between 0. and 1.0.** This means that we may have to change parameters, or even gridding algorithms if too much slope is introduced into the grids by the algorithm.
- The grids **must be completely defined** within the pay zone. No holes in the grid are allowed, otherwise, no volume can be calculated there.
- As with other grids, the grid values **must honor** the values in the wells.
- The grids should be relatively smooth between the well data points, and should **not contain sharp discontinuities** which are not associated with the existing fault boundaries, if used.

Typical Difficulties When Gridding Properties

1. Faulting can affect property values during deposition, as well as after deposition. In some reservoirs, the **fault boundaries** for either the **top** of zone or **base** of zone are used during property gridding to help provide discrete ranges of the property value in certain fault blocks, where appropriate.

2. Good property data for all lithozones is typically hard to come by and data from the wells is usually **sparse**. This type of data can be challenging to model adequately. We see that in the example below, the data points for which we have property values barely reaches into the reservoir area.

If seismic attributes exist for which it can be shown that there is a correlation with property data, then interval property grids can be improved with various techniques including the use of the application **Log Property Mapping (LPM)** which can extend the accurate extent of property grids beyond the lateral limits imposed by the well data.

Gridding Guidelines

With sparse data, we will probably get the best results using the Convergent algorithm. When gridding property data, it is almost invariable that extrapolation will be required.

The only other issue to worry about is the value for the Starting Grid Interval when using the Convergent algorithm. The rules of thumb we discussed in the earlier gridding chapter can be summarized here for sparse data:

Pick the larger of

- the largest empty distance between control points
- the largest empty distance between a control point and the edge of the pay zone.

Remember, this number is not critical to determine the very first time. If it is too small (pay zone not fully defined, or holes in the grid), then we'll simply re-grid and make it bigger the next time.

Continuing with the OIP Equation

Once the individual **property grids** exist, the following series of grids are normally computed individually with Single and Multiple Surface Operations.

- Net-to-Gross = **Net Thickness** / **Gross Thickness**
- Net Isochore = Net-to-Gross * ***Gross_Isochore***
- Net Pore Volume = Net Isochore * **Net Porosity**
- Net Pay = Net Pore Volume * (1.0 - **Net Pay Water Saturation**)

Items in **boldface** are the individual property grids. The ***Gross_Isochore*** is the structural envelope thickness, which is simply the difference of the top and base structural envelopes. The final Net Pay grid becomes the input to the Volumetrics operation, along with any associated fault polygons and the lease polygons.

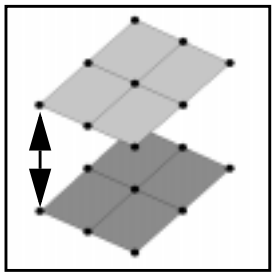
All operations involve only ***surface arithmetic*** when applying the properties. When we created the structural envelope, we used primarily ***surface logical operations.***

Using the Formula Processor for a Shortcut

Instead of computing the grids above independently, it is possible to compute the **Net Pay** grid from the ***Gross_Isochore*** and the individual **property grids** in a single operation. In the exercises for this chapter, we'll show you how to solve the OIP equation at the beginning of the chapter in one step from the dialog under OPERATIONS/FORMULA.

Note on Arithmetic and Logical Operations

In all the logical operations which we have performed to shape the envelope, and all the surface arithmetic we have performed to compute the Net_Pay grid, we have not used the **fault boundaries** (traces) at all. The reason for this is that the CPS-3 logical operations and surface arithmetic functions perform their calculations node-by-node, **vertically**, on corresponding grid nodes of the input grids.



Computing Oil in Place with Volumetrics

We use the CPS-3 Volumetrics procedure to calculate oil in place. This procedure has the ability to perform **numerical integration between a surface and a base plane**. It accumulates volume grid cell by grid cell and has the ability to differentiate the volumes on either side of a fault or lease polyline when a cell is dissected by the line. Refer to the on-line documentation for this procedure for an summary of its operation.

If the **volumetric input surface** were a gross isochore, the results would simply be volume of rock. However, having applied all of the required rock properties to the thickness grid, the **volumetric input surface** is no longer in a simple thickness domain, and the result on the volumetrics report will be oil in place. We have an option to report the results in barrels or other units, if desired. A separate report for each lease polygon is generated.

The inputs to the **Volumetric** procedure are:

- single **Net_Pay** grid which we have just calculated.
- set of combined top and base **fault boundaries**
- lease **polygons**

The output from the **Volumetric** procedure is a report such as the example shown below.

```
---------------------- A Polygon   4:  DeanWhistler

  Panel  1:  Rows      1 to    111.  Columns     32       106

FULL AREA of Polygon (or Window):  2662.993555999999899

        Integrated Results ABOVE the Horizontal Reference Plane (CUT):
           VOLUME:                      18.4334160451391966
           FLAT AREA:                  1671.350426209423631
           SURFACE AREA:               1671.362227900795688

        Integrated Results BELOW the Horizontal Reference Plane (FILL):
           VOLUME:                       1.3749645819102270
           FLAT AREA:                   99.3336770884893099
           SURFACE AREA:                99.3351902546194623

        NET   Results (ABOVE - BELOW):
           VOLUME:                      17.0584514632289697
           FLAT AREA:                  1572.016749120934264
           SURFACE AREA:               1572.027037646176041

        TOTAL Results (ABOVE + BELOW):
           VOLUME:                      19.8083806270494236
           FLAT AREA:                  1770.684103297912998
           SURFACE AREA:               1770.697418155415107

Ratio of INTEGRATED AREA to FULL AREA:    0.6649224138407616
```

Volume is the integrated volume between the surface and the base plane

Flat Area is the flat area of that portion of the surface ABOVE or BELOW the base plane. The area of the **polygon** (lease) is also a flat area.

Surface area is the curvilinear area of that portion of the surface ABOVE or BELOW the base plane and should always be equal or greater than the Flat Area.

For a **Net_Pay** volumetric grid which is based on an isochore, and whose values of interest should be positive, we are only interested in numbers above zero, and so we need only to look at the **Integrated Results ABOVE the Horizontal Reference Plane** in the report. The other sections are useful when the volumetric grid is a structural model and Civil Engineering issues are to be studied, such as cut and fill for highway design.

LECTURE A for Topic 21
Overview of Model Editor

Overview

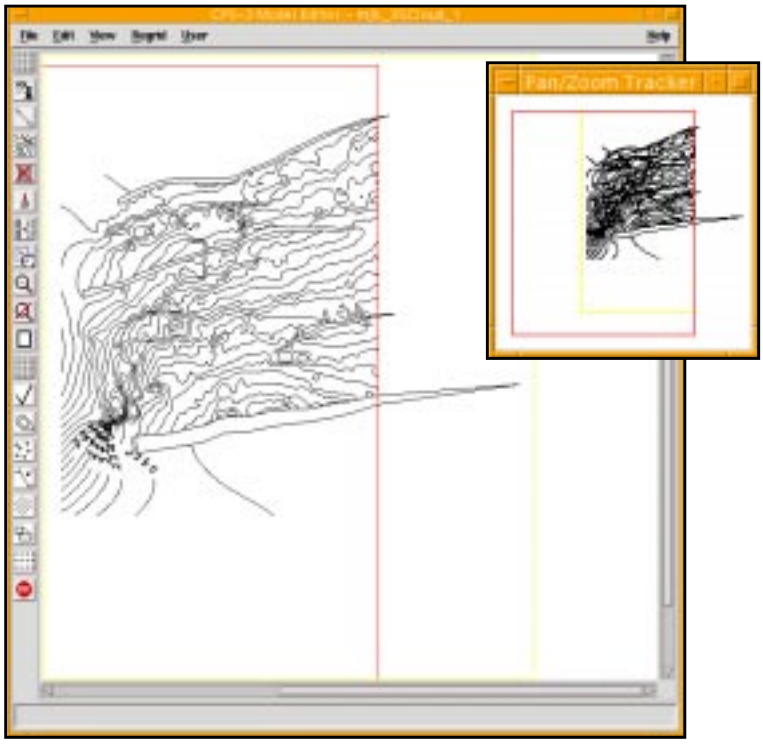


Figure 21.1 CPS-3 Model Editor displaying its Pan/Zoom feature

The **CPS3 Model Editor** is a powerful tool that lets you edit gridded surface models, along with data, faults, polylines, and cultural features. Surfaces are edited though **manipulation of contour lines** and/or data values and fault locations. After the edits are complete, the surface is then regridded using the changed contours and/or data.

You may also change a grid by simply editing the node values directly in a variety of ways.

The **Model Editor** gives you greater control during modeling, and allows you to focus in specific areas for modification. It lets you integrate your knowledge and interpretation into the modeling process.

Aside from editing the gridded model itself, the **Model Editor** has many provisions for making changes to your **Data** sets, **Fault** sets, and **Polygon** sets.

Starting the Model Editor

There are four locations at which the **Model Editor** may be launched. There are also two modes in which launching occurs: **independent mode** and **open mode**.

Independent Mode

When launched in **independent mode**, the **Model Editor** makes no assumptions regarding the sets which you may wish to edit. After it comes up, you load each set you want by clicking **File > Load** as below.

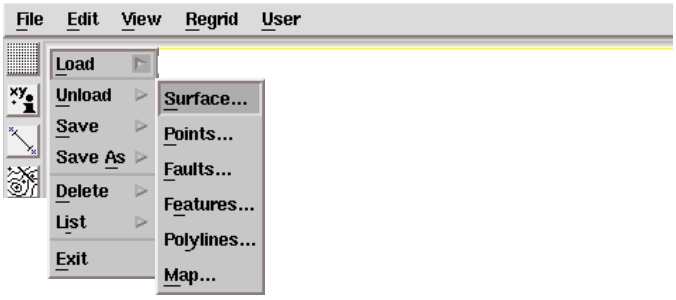


Figure 21.1 CPS-3 Model Editor displaying its load features

The **Model Editor** is launched in **independent mode** from the following two two locations:

- In the **GeoFrame Visualization Catalog**, click on **Model Editor**.
  - At the bottom, designate on which monitor you want the **Model Editor** to be launched.
  - < **machine_name**>:**0.0** will launch it on the **left** monitor, :**0.1** will launch to the **right**.
  - Click **OK**.
- In the **CPS-3 Main Module**, click **Tools > Model Editor**.

Open Set Mode

When launched in **open set mode**, the **Model Editor** assumes that you want it to load those sets which are currently open in your **CPS-3** session. Look at your **CPS-3 Status Information** window and it will show you the currently open sets at the top of the window. The following dialog box appears in open set mode, and you can change any of the set names before the **Model Editor** launches and loads them.

Figure 21.1 Link to Model Editor dialog box

The **Model Editor** is launched in **open set mode** in the following two methods:

- In the **CPS-3 Main Module**, click **Operations > Surface/Model Editor**.
- Click on the **Model Editor** icon. 

Model Editor functions

The **Model Editor** has many features within its icons and menus, and there are many ways to actually modify a grid, based on personal preference. Here, we refer you to the on-line **CPS-3 User's Guide** which contains menu-by-menu and icon-by-icon documentation of every feature included in the **Model Editor**.

We recommend that you read this documentation if you intend to use the **Model Editor** to any degree.

In this chapter, however, we want to go through examples of some of the most common editing tasks using the **Model Editor**, rather than recite the function of each menu and icon. which is well done in the document above.

Typical Editor session

- Launch **Model Editor**.
  - — Load the surface set, along with relevant data and faults, if any.
  - — Generate surface contours lines.
- Identify the area need to be edited, **zoom in** if necessary.
- Edit the contours, data points, faults, polylines, etc., as needed.
- Set an edit window to enclose all your modifications in this area.
- Regrid the area defined by the edit window.
  - — If the regridded results, showing as contours in a different color, are NOT satisfactory, **Undo** regrid.
  - — Go back to editing, and regrid again.
- Recalculate contours to match the contours for the regridded node values, if the regridding reached the desired results.
- Repeat Step 2 - 6, as needed, until you are satisfied with the entire surface.
- **Save** your edits.

Tips Regarding Grid Editing

- It is better to edit your data and regrid the entire surface in the **CPS-3 Main Module**, if possible, to make the corrections you desire. In this way you are able to recreate the grid at any time, as long as the original data is available.
- If your surface requires editing in large areas, do not try to do it in the **Model Editor**. The **Model Editor** was not designed to do regional edits, but is most useful in small local edits to simply clean up a grid.
- 3. Edit each area independently, zooming into the smallest area possible, and setting the smallest regridding area possible.
- 4. Save your grid often; use the **Save As** to retain intermediate versions so that you do not lose your work.
- 5. Try to finish all your data and fault edits before moving on to your surface edits.

Overview of the CPS-3 Map Editor

The **CPS-3 Map Editor** lets you perform simple graphic editing on saved map sets. Map sets are saved during sessions in the **CPS-3 Main Module** and contain graphic objects and their attributes. Map sets are identifiable by their **UNIX** file extension, **.mcps**. The **Map Editor** is not a substitute for a full-featured **CAD** program. It does, however, provide a number of useful editing features.

- Add, move, copy, and delete text and symbols.
- Modify graphical attributes, such as font, size, color, rotation angle, justification, etc., for text and symbols.
- Modify graphical attributes for lines and polygons (polylines), such as line color, style, thickness.
- View and edit map subsets.
- Combine up to five map sets into a composite map.

Tip: *The **Map Editor** performs only graphic editing, and is limited to moving and creating simple objects and modifying their attributes, such as color, font, size, line style, width. The main use of this application is to clean up maps which have already been created. The **Map Editor** is NOT designed for editing contours, data sets, fault sets, polygon sets, or grids. These should be edited in the **CPS-3 Model Editor**. Changes made in the **Map Editor** will only be reflected in the **CPS-3 Map set**.*

Starting the Map Editor

There are two ways to start the **Map Editor**.

- **GeoFrame Application Manager** > **Visualization icon** > **Visualization Catalog**
  - — Click on the **CPS-3** folder
  - — Click once on **Map Editor** to highlight it, and at the bottom, designate the monitor on which to launch it.
  - — Click **OK**.
- You may also launch the **Map Editor** from the **CPS-3 Main Module**, under **Tools**. (The target screen is controlled at the bottom of the **Visualization Catalog**.

Figure 21.1 CPS-3 Map Editor main menu

Pull-down menus

The pull-down menus - **File**, **Activate**, **Mode**, **View** and **Options** are located across the top of the **Map Editor** window. Most of the functions under the pull-down menus are also available as icons in the tool bar. Here are some of the functions:

File

- **Load**: you may load up to 5 map sets for each session;
- **Save**: save the edited map set;
- **Save as**: save the edited map set under a different name;
- **Unload**: unload a map set from the session;
- **Exit**: exit the session.

Activate

- **Map set**: same as the icon **(ACTIVATE:) Map**. Among the loaded map sets (up to 5 sets), only one of them is **active**. If not picked here, the **last one loaded to the session** will be the active map set, which is the one on display.
- **Editable Layers**: same as the icon **(ACTIVATE:) E Layers**. All elements/subsets of the active map set are listed here as separate layers. You may turn ON and OFF any of them to allow selected ones to be edited. The layers you turn off may NOT be edited.
- **Viewable Layers:** same as the icon **(ACTIVATE:) V Layers**. All elements/subsets of the active map set are listed here as separate layers. You may turn off certain layers to hide them temporarily if you do not want them to be on display.

Mode

Under **Mode**, you will see three groups of functions, **Select**, **Add** and **Composite**. Functions under each group are also iconized in the tool bar.

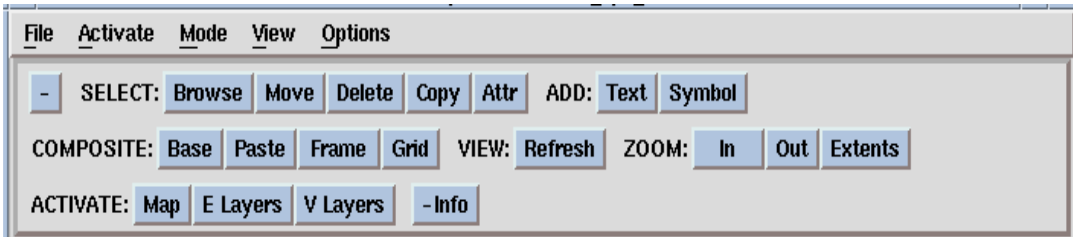


Figure 21.1 CPS-3 Map Editor bar menu

Select Mode

- **Browse**: This function displays the x,y location of the cursor as you click the mouse button.
- **Move**: This function allows you to move an object. Click on the **Move** button, and click on the object that you want to move. You will see a white frame enclosing the object you just picked. Leave the cursor inside the white frame and click the mouse button again to pick the white frame up and drop to the place you want to move to. You can only move one object at a time.
- **Delete**: This function allows you to delete an object. Click on the **Delete** button, and click on the object that you want to delete. You will see a white frame enclosing the object you just picked. Leave the cursor inside the white frame and click the mouse button again to delete the object. You can only delete one object at a time.
- **Copy**: This function allows you to copy an object. Click on the **Copy** button, and click on the object that you want to copy. You will see a white frame enclosing the object you just picked. Leave the cursor inside the white frame and click the mouse button again to pick the white frame up and drop to the place you want to copy to. You can only copy one object at a time.
- **Attribute**: This function allows you to change the graphic attributes of an object. To change attributes, click on the **Attr** button, and click on the object you want to change. You will see a white frame enclosing the object that you just picked. Leave the cursor inside the white frame and click the mouse button again, an attribute dialog box will pop up. If the object you picked is **text**, you can change the text content, size, rotation angle, font, color, quality and justification. If it is a **symbol**, you will be able to select a different symbol, along with size, color and rotation angle. If it is a line, you may change the line style, width, color and smoothness. After you input the new attributes, click on **Apply** button. The changes will display on screen. You can only change attributes on one object at a time.

Note: *Changing of text font will not show up in the **Map Editor** display.*

Add Mode

This feature allows you add text and symbols to a map set.

- **Text**: To add text, click on the **(ADD:) Text** button, displaying a dialog box which allows you to specify text attributes. After setting the text attributes, click on the **Apply** button. Then move the cursor to the **Map Editor** window, a white frame will appear. Locate the white frame at where you want the text to be, click the mouse button again to display it.
- **Symbol**: To add a symbol, click on the **(ADD:) Symbol** button, displaying a dialog box which allows to you pick a symbol from the library and specify its attributes. After setting the symbol attributes, click on the **Apply** button. Then move the cursor to the **Map Editor** window, a white frame will appear. Locate the white frame at where you want the symbol to be, click on mouse button again to display it.

Composite Mode

Composite functions allow you to combine up to five map sets to make a single, composite map.

- **Base**: This provides you with a blank canvas on which you can size and arrange the active map set, leaving space for pasting additional map sets. If **Grid** is turned ON, the canvas will show grid marks. If the **Snap to Grid** function is ON, the size of the map set will be snapped to the nearest grid mark.
- **Paste**: Other than the active map set, all additional map sets loaded into the session (up to four sets) can be pasted onto the canvas. You can resize and rearrange these map sets. The **Grid** and **Snap** function works the same way as in **Base** mode.
- **Frame**: This function allows you to enlarge the border of your composite map set.
- **Grid**: This function allows you to turn the grid mark on the canvas ON and OFF, turn **Snap to Grid** function ON and OFF, and define the size of the grid on the canvas.

View

- **Refresh** - redraws the contents of the screen.
- **Zoom** - provides zoom functions - **In**, **Out**, **Extents**

Options

- **Icon Bar** -toggles to icon bar on and off.
- **Info Window** - toggles the info window on and off
- **Set Background Color** - changes the background color of the canvas
- **Set Line Edit On** - edits a line
- **Set Point Edit On** - edits a single point
- **Set Quick Screen Repair** - sets quick refresh

LECTURE B for Topic 21
Overview of the CPS-3 Map Editor

Overview

The **CPS-3 Map Editor** lets you perform simple graphic editing on saved map sets. Map sets are saved during sessions in the **CPS-3 Main Module** and contain graphic objects and their attributes. Map sets are identifiable by their **UNIX** file extension, **.mcps**. The **Map Editor** is not a substitute for a full-featured **CAD** program. It does, however, provide a number of useful editing features.

- Add, move, copy, and delete text and symbols.
- Modify graphical attributes, such as font, size, color, rotation angle, justification, etc., for text and symbols.
- Modify graphical attributes for lines and polygons (polylines), such as line color, style, thickness.
- View and edit map subsets.
- Combine up to five map sets into a composite map.

Tip: *The **Map Editor** performs only graphic editing, and is limited to moving and creating simple objects and modifying their attributes, such as color, font, size, line style, width. The main use of this application is to clean up maps which have already been created. The **Map Editor** is NOT designed for editing contours, data sets, fault sets, polygon sets, or grids. These should be edited in the **CPS-3 Model Editor**. Changes made in the **Map Editor** will only be reflected in the **CPS-3 Map set**.*

Starting the Map Editor

There are two ways to start the **Map Editor**.

- **GeoFrame Application Manager**>**Visualization icon**>**Visualization Catalog**
  - — Click on the **CPS-3** folder
  - — Click once on **Map Editor** to highlight it, and at the bottom, designate the monitor on which to launch it.
  - — Click **OK**.
- You may also launch the **Map Editor** from the **CPS-3 Main Module**, under **Tools**. (The target screen is controlled at the bottom of the **Visualization Catalog**.

Figure 25.1 CPS-3 Map Editor main menu

Pull-down menus

The pull-down menus - **File**, **Activate**, **Mode**, **View** and **Options** are located across the top of the **Map Editor** window. Most of the functions under the pull-down menus are also available as icons in the tool bar. Here are some of the functions:

File

- **Load**: you may load up to 5 map sets for each session;
- **Save**: save the edited map set;
- **Save as**: save the edited map set under a different name;
- **Unload**: unload a map set from the session;
- **Exit**: exit the session.

Activate

- **Map set**: same as the icon **(ACTIVATE:) Map**. Among the loaded map sets (up to 5 sets), only one of them is **active**. If not picked here, the **last one loaded to the session** will be the active map set, which is the one on display.
- **Editable Layers**: same as the icon **(ACTIVATE:) E Layers**. All elements/subsets of the active map set are listed here as separate layers. You may turn ON and OFF any of them to allow selected ones to be edited. The layers you turn off may NOT be edited.
- **Viewable Layers:** same as the icon **(ACTIVATE:) V Layers**. All elements/subsets of the active map set are listed here as separate layers. You may turn off certain layers to hide them temporarily if you do not want them to be on display.

Mode

Under **Mode**, you will see three groups of functions, **Select**, **Add** and **Composite**. Functions under each group are also iconized in the tool bar.

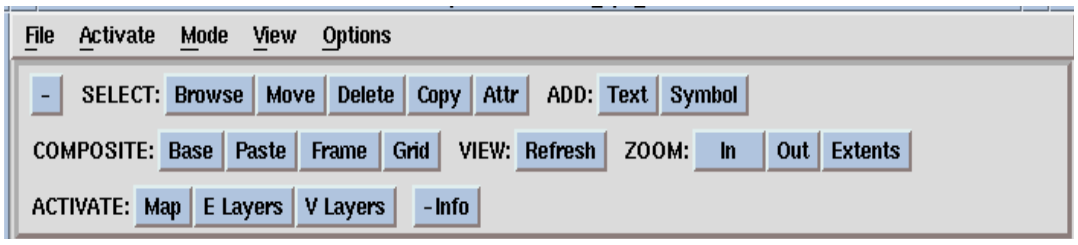


Figure 25.1 CPS-3 Map Editor bar menu

Select Mode

- **Browse**: This function displays the x,y location of the cursor as you click the mouse button.
- **Move**: This function allows you to move an object. Click on the **Move** button, and click on the object that you want to move. You will see a white frame enclosing the object you just picked. Leave the cursor inside the white frame and click the mouse button again to pick the white frame up and drop to the place you want to move to. You can only move one object at a time.
- **Delete**: This function allows you to delete an object. Click on the **Delete** button, and click on the object that you want to delete. You will see a white frame enclosing the object you just picked. Leave the cursor inside the white frame and click the mouse button again to delete the object. You can only delete one object at a time.
- **Copy**: This function allows you to copy an object. Click on the **Copy** button, and click on the object that you want to copy. You will see a white frame enclosing the object you just picked. Leave the cursor inside the white frame and click the mouse button again to pick the white frame up and drop to the place you want to copy to. You can only copy one object at a time.
- **Attribute**: This function allows you to change the graphic attributes of an object. To change attributes, click on the **Attr** button, and click on the object you want to change. You will see a white frame enclosing the object that you just picked. Leave the cursor inside the white frame and click the mouse button again, an attribute dialog box will pop up. If the object you picked is **text**, you can change the text content, size, rotation angle, font, color, quality and justification. If it is a **symbol**, you will be able to select a different symbol, along with size, color and rotation angle. If it is a line, you may change the line style, width, color and smoothness. After you input the new attributes, click on **Apply** button. The changes will display on screen. You can only change attributes on one object at a time.

Note: *Changing of text font will not show up in the **Map Editor** display.*

Add Mode

This feature allows you add text and symbols to a map set.

- **Text**: To add text, click on the **(ADD:) Text** button, displaying a dialog box which allows you to specify text attributes. After setting the text attributes, click on the **Apply** button. Then move the cursor to the **Map Editor** window, a white frame will appear. Locate the white frame at where you want the text to be, click the mouse button again to display it.
- **Symbol**: To add a symbol, click on the **(ADD:) Symbol** button, displaying a dialog box which allows to you pick a symbol from the library and specify its attributes. After setting the symbol attributes, click on the **Apply** button. Then move the cursor to the **Map Editor** window, a white frame will appear. Locate the white frame at where you want the symbol to be, click on mouse button again to display it.

Composite Mode

Composite functions allow you to combine up to five map sets to make a single, composite map.

- **Base**: This provides you with a blank canvas on which you can size and arrange the active map set, leaving space for pasting additional map sets. If **Grid** is turned ON, the canvas will show grid marks. If the **Snap to Grid** function is ON, the size of the map set will be snapped to the nearest grid mark.
- **Paste**: Other than the active map set, all additional map sets loaded into the session (up to four sets) can be pasted onto the canvas. You can resize and rearrange these map sets. The **Grid** and **Snap** function works the same way as in **Base** mode.
- **Frame**: This function allows you to enlarge the border of your composite map set.
- **Grid**: This function allows you to turn the grid mark on the canvas ON and OFF, turn **Snap to Grid** function ON and OFF, and define the size of the grid on the canvas.

View

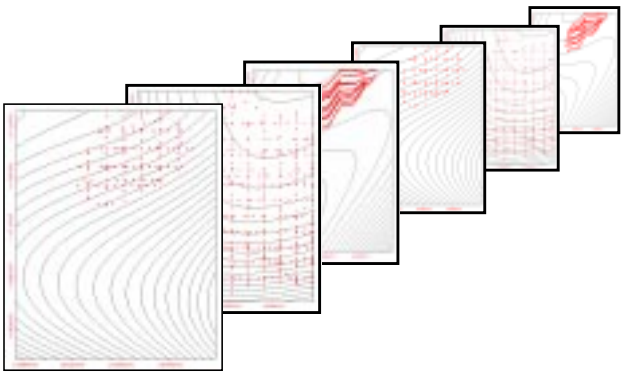
- **Refresh** - redraws the contents of the screen.
- **Zoom** - provides zoom functions - **In**, **Out**, **Extents**

Options

- **Icon Bar** -toggles to icon bar on and off.
- **Info Window** - toggles the info window on and off
- **Set Background Color** - changes the background color of the canvas
- **Set Line Edit On** - edits a line
- **Set Point Edit On** - edits a single point
- **Set Quick Screen Repair** - sets quick refresh

LECTURE for Topic 22
Creating, Using and Editing Macros

Overview



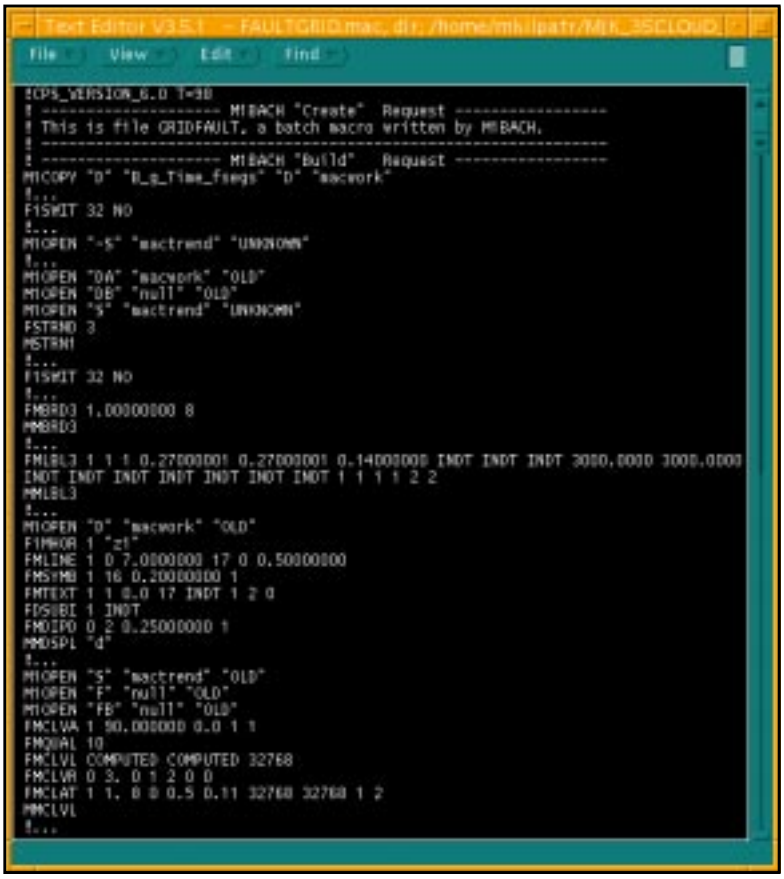
One of the most useful features in a mapping package is the ability to perform a series of predefined steps at different times or under different conditions. **CPS-3** has had the ability to make macros for some time, but macro extensibility has not been generally possible. Recently, however, macros were made much more powerful with the introduction of a flexible **macro command language**, which are an addition to the current macro structure, and allow prompting, variable substitution, and other features to make macros more powerful than before.

There are 3 categories of macros, defined by where they are stored:

1. **System** macros - stored in <install_path>/cps3_run_mac/
2. **Project** macros - stored in the project's CPS-3 dsl.
3. **User** macros - stored in user-specified location

Basic Macro Format

A macro is a simple **ASCII** file that contains a series of **CPS-3** native mapping commands. **Native** commands are documented in the **Reference Manual**, which is available on-line under **Tools** in the **CPS-3 Main Module**. You do **not** need to know native commands in order to use macros. Those who run macros frequently become knowledgeable about native commands, but it is not necessary to begin using them. Being stored in **ASCII** files, macros are easily transported and edited. A portion of an **ASCII** macro file is shown below as seen from the Unix “TextEdit: file editor.



Creating Macros

Macros are built, one command at a time, interactively in the **CPS-3 Main Module**, as you step through the dialog boxes performing the actions which you want to be stored in the macro. You can choose to only “go through the motions” when creating a macro, or you can literally execute the steps as you

create it. By executing the steps as you go, you can catch errors which you would not see if you only “go through the motions”, however, both methods have their uses.

Once created, the macro is placed in the **Project macro** directory, which is the project’s CPS-3 dsl.

Macro Grouping

For all your macros, you have the option to organize them into meaningful **groups**. Read how to do this in the **online CPS-3 User Manual**. You can define group names in a simple file, and assign each macro to a different group. As you access the Macro functions from the Main Module menu, you then have the opportunity to select your macros by group.

Running Macros

After you create a macro from CPS-3, you can run it macro interactively, or “in the background” while you continue other mapping tasks. Macros enable you to automate common, repetitive mapping tasks, as well as mapping tasks that take a long time to complete.

Making a Macro Universally Useful

As created from CPS-3, macros contain the actual set names which were used for input and output as the macro was created. This limits the flexibility of macros, and this is why the **macro command language** was added to the system - to be able to transform the ***fixed-name*** initial macro into a universal tool which itself is able to prompt the user for the names of input and output sets. The **macro command language** also adds other capabilities to macros besides prompting such as executing loops, executing system commands, and calling other macros.

Below, you’ll see how to modify a fixed-name macro into one which prompts for input.

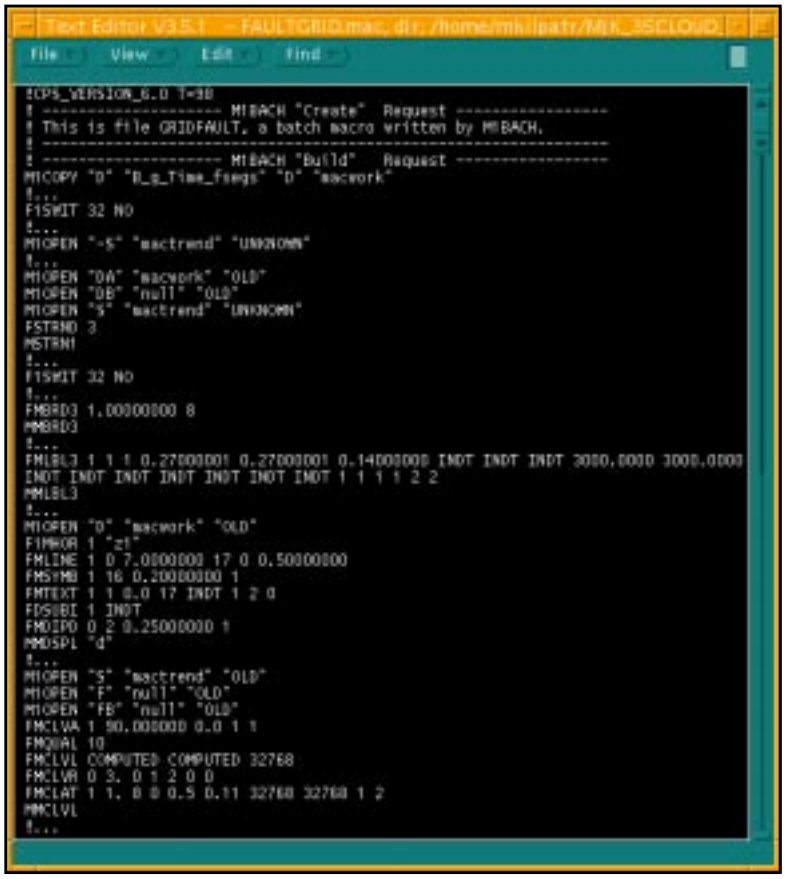
Besides the **Project macros**, this version of CPS-3 supports access to an existing library of macros called **System macros** which are found in the GeoFrame installation directory under the sub-directory .../cps3_run_mac/.

The New Macro Language Abilities

The new language provides you with a variety of abilities within any macro you create with CPS-3. These abilities are listed below.

- Establish **named variables** in a macro for your own purposes. Variables can contain characters, numbers, set names, or colors.
- Perform **arithmetic and logical operations** between variables.
- Control the **flow of the macro** with these operators:
  - **While** statement
  - **If - Then - Else**
- Add **interactive prompting** inside the macro to establish variable values.
- Perform **substitution** operations on set names and parameter values before the macro is run.
- Spawn detached processes, or spawn a process and wait for it to finish.

Note: *For detailed descriptions of each of these language facilities, please refer to the chapter about Macros in the **CPS-3 User's Manual**.*



Basic Facts about Macro Syntax and Organization

- Every line in the macro which starts with an exclamation point is a **comment**.
- Every line in the macro which starts with a six-letter command beginning with **F**, such as **F1SWIT** is a **parameter-setting command**.
- Each value following a parameter-setting command sets the value of a specific parameter. Some commands contain only one parameter, others contain many more parameters.
- Every line in the macro which starts with a six-letter command beginning with **M**, such as **M1OPEN**, is an **operation command**.
- Some operation commands rely on **previous** parameter-setting commands to establish how the command will be executed, such as **MSTRN1**. Others rely on parameters which occur on the same line of the operation, such as **M1OPEN**.
- All parameter-setting commands and operation commands are documented in detail in the **CPS-3 Reference Manual**, which can be found under **Tools** in the **CPS-3 Main Module**.

Typical Prompting Language Added to a Macro

Below, you see some typical **macro commands** relating to prompting components which are added to macros. These commands are in a file call k_prompt.mac in the external training data for CPS-3.

This file could be inserted as it is into a macro if you wanted to prompt for a Data set, and an output Surface set name. As you can see, this file could be used as simple template and you can change the names, or even add others sets to be prompted for. Refer to the on-line **User's Manual** for details.

```
!Prepare macro for prompting
!
! Add these lines to the top of
! any macro to prompt for an input
! data set name and an output
! surface name before execution
!
!.................................begin prompt
declare $DATA1 as set
declare $surf as string
let $DATA1 = "null"
let $surf  = "null"
begin_dlg
prompt $DATA1 "Select data set" "data"
end_dlg
begin_dlg
prompt $surf "Type output surface name"
end_dlg
!.................................end prompt
```

Looking at each command in this example, we see that

"**declare**" defines variables such as set names or strings.

"**let**" assigns initial values to the variables

"**begin_dlg**" and "**end_dlg**" bracket the prompting commands for a single dialog

"**prompt**" activates a single prompt line in the dialog for one variable

In general, variables should be declared as **set**, if they will be used for the selection of existing input sets. Variables declared as **string** are set by the user only by typed response.

Substituting Fixed Set Names with Variable Names

The next phase in making a macro universally useful is to find all places lower in the macro where the fixed-name sets are specified, and change them to the appropriate variable name. For example, if the original unedited macro contained the line

M1OPEN “DA” “mysetname” “UNKNOWN”

which opens a Data set named mysetname, we should replace the fixed set name (including double quotes) with the variable name $DATA1 (including the Dollar sign):

M1OPEN “DA” $DATA 1“UNKNOWN”

After all fixed set names have been replaced as shown with the proper variable name, the macro will have been converted into a universally useful tool which can be passed around for anyone to use in their own project.

Scan the Edited Macro for Errors

If you like you can scan an edited macro for errors before you actually try to use it. Use the MACROS/SCAN features

Current Constraints: Macro Execution and Environments

1. At present, no information is written to a macro during Macro Creation during Environment **selection, creation, or editing**.

2. At present, there is no macro command language to **create** new environments or **edit** existing environments

3. At present, when a macro executes, it assumes that the **currently active display and modeling environments are to be used**. You can change this, as will be shown below, by inserting various environment-related commands in the macro.

Getting Around the Constraints

Future releases may improve upon these constraints, but until then, we must use whatever tools are available to help us manage environments within macros. Below, we document what is available in this release of the software. These capabilities exist because environments are stored in the CPS-3 session files.

To cause a specific session file be used in a macro:

Add the following line in the macro.

READ <**full path to a saved session file name**>

Session files are files of the form <login_id>.1cps.

To add a display environment specification in a macro:

Add the following line in the macro:

```
F1DRAW n
```

where **n** is an environment number in the current session file. The only way to determine **n** is to view all environments in CPS-3 and identify it in the GUI.

To add a modeling environment specification in a macro

Add the following line in the macro:

```
F1MODL n
```

where **n** is an environment number in the current session file.

To save the currently defined modeling environments to a session file when in CPS-3:

Although not specifically applicable **inside** of a macro, this function is run from the **CPS-3 Main Module** and is useful in preparing a session file for a macro to use. In the **CPS-3 Status Information** window, enter the following command at the bottom of the dialog box in the field labeled **CPS-3 Command**. This command can be done at any time, but should probably be done just before you begin to make your macro.

SAVE <**name of file to contain the saved session file**>

Compatibility: Running Pre-GF3.5 Macros

Since many internal formats and parameter storage mechanisms have changed, older macros must be converted to the current macro format before being executed. The rules for pre-GF3.5 macros are as follows:

The macro is read and then converted to a **6.0** macro, having a naming convention of **CPS60_**<**old name**>. During this process, older **native** commands such as **F1WINE**, **F1GINC**, **F1ZTYP**, **F1UNIT**, and **F1VSCL**, which were previously used to establish a primitive mapping environment, are commented out.

After a macro has been converted, it contains a header record at the beginning of the file which looks like

!CPS_VERSION_6.0 T=98

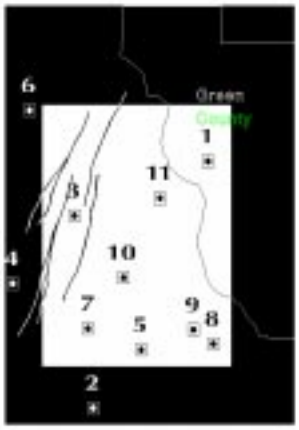
which means that CPS-3 will read and execute the macro without assuming that it needs conversion. All macros created in GF3.5 and later will contain the same header, so that no conversion will be attempted.

Managing Macros - Enhancements for GF4.0

Please refer to the Lecture Document **I40_c05_Newfor40** which describes new capabilities for macros regarding their organization, description, and selection.

LECTURE for Topic 23
Display/Graphic Operations and the Environment

Overview



In this chapter, you will be introduced to many concepts in **CPS-3** associated with the creation, storage, modification, clipping, viewing, and plotting of graphic displays. In earlier chapters, we talked about mapping **environments** and how the X,Y box is the most visible of environment attributes. Here, we will show how graphic objects are classified and how each interacts with the X,Y box of the changing **Display** environments during your **CPS-3** session.

Graphic Display in CPS

Whenever anything is displayed on the screen in **CPS-3**, it is recorded in **screen memory,** which is actually a hidden **CPS-3** map set used to refresh the screen quickly.

If a **permanent** copy of the screen contents is desired, use the **Save currently displayed map** icon  to write screen memory to a permanent **CPS-3** map set.

Saved map sets retain all displayed graphic components and their attributes. In addition, subset markers are written into the map set so that the graphic output of one process can be distinguished from another. These layers of a saved map set can be **deleted**, or **rearranged**, if desired. Use the **Manipulate current map layers** icon  to invoke the **Map Layer Manager**.

We will perform an exercise later in the course to illustrate how map layers are rearranged and deleted.

Honoring the Active Display Environment

Before anything is actually stored in screen memory, the active display environment determines if any type of transformation is required. In particular, the displayed data may be:

- **transformed** to the active **Geographic Coordinate System**
- **scaled** to the active scale
- **converted** to the active horizontal and vertical units
- **clipped** by the active **Volume of Interest** (**VOI**)

It is the last operation, **clipping**, which provides the main focus for this chapter. We will discuss conditions when graphic displays become clipped, or when they are merely hidden.

Note: *The basic 2D clipping rectangle is defined by the X and Y extents of the **Volume of Interest**. However, these can be enlarged by the temporary specification of a **Top**, **Bottom**, **Right**, and **Left margin**, if desired.*

When Are Graphic Objects Clipped?

Two Display Classes

For clipping purposes, objects to be displayed in **CPS-3** fall into one of two classes - **Inside** objects and **Outside** objects.

Inside

Inside objects (see the following) are **clipped** to the **Volume of Interest** during display.

- Data
- Faults
- Polygons
- Surfaces
- Maps (optional)

Any graphic manifestations of the **CPS-3** set types shown above, such as point display, polylines, data value display, or contouring, are also considered to be **Inside** objects, and any graphics generated outside the X,Y box will be discarded.

Upon displaying a **Map** set, you will be given the choice of treating it as an **Inside** or **Outside** object, as desired.

Outside

Outside objects are **NOT** clipped during display. The following are considered **Outside** objects:

- Maps (optional)
- Borders,
- Labels

- Title blocks
- North arrows
- In general, graphic output from all **Basemap** dialog boxes, except **Display Set** and **Lines and Annotation** sections, is considered **Outside** graphic objects.

Clipping During Graphic Display

When **Inside** objects are displayed on the screen, they are clipped exactly to the active **Volume of Interest**, and then stored in **Screen Memory**. This occurs whether the user is zoomed in or not.

When **Outside** objects are displayed on the screen, they are NOT clipped, and are stored in their entirety in **Screen Memory**, even though they may not be visible. Again, this occurs whether or not the user is zoomed in. Even though **Outside** graphics may not fall within the visible area of the display, they can be made visible by creating and activating a **Display** environment whose display volume is large enough to include the data.

Alternatively, clicking the **Reveal all graphics** icon,  will show all graphic elements currently in screen memory.

The **Reveal all graphics** icon is activated when the current display does not show all graphic components stored in **Screen Memory**. An example might be a title block which was placed at some x and y offset from the lower left corner of the map, but which fell outside the current viewing area.

Note: *There was no mention of the current zoom window in the previous discussion concerning clipping. Since there is now only one clipping window, which is defined by the X, Y limits of the currently active* ***Display*** *environment, and the margins,* ***display of data while zoomed in will NOT cause the data to be clipped to the zoom window.*** *This means that when you zoom out again after having displayed some data, you will see all of it inside the X, Y limits, not just a piece of it, as in earlier versions.*

In the exercises which follow, we will demonstrate the concept of graphic clipping, and the difference between **Inside** and **Outside** graphic objects.

The exercises for this chapter will provide the opportunity to demonstrate the principles discussed here.

LECTURE for Topic 25
Technical Note - Formatting Data for the CPS-3 ASCII Data Loader

Overview

The **CPS-3 ASCII Data Loader** is invoked from the **CPS-3 Main Module** menu bar and offers a very wide variety of data loading features.

Click on **File > Import > ASCII** to see that you can load any of the five basic set types in **CPS-3** from **ASCII** files.

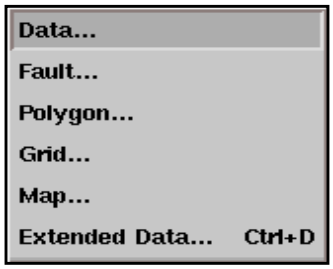


Figure 27.1 Available ASCII file types

The **Extended Data** type at the bottom is an enhanced method of loading **Data** sets when it contains textual information or graphic symbology in the file. This will be explained later.

While the **CPS-3 ASCII Data Loader** is very flexible and can load many different file formats, there are certain conventions, regarding the organization of these files, which are outlined in this chapter.

General Requirements/Options

In general, each physical record in a data file represents **one data point** or **vertex** in **CPS-3**.

All information associated with **one point** must be on the same physical record.

There is a storage limit of **50 z-attributes per data point**. That is, a data point can have up to 50 z-values, each representing a measurement.

You can skip any number of records at the beginning of a file, but you **CANNOT** skip any at the end. Trailer records, which are not data records, must be physically deleted from the end of the file before loading.

Comments, prefaced with a "**!**" in **column 1** of a record, can be embedded in any of the **ASCII** file types, and will be ignored.

The **order** of x, y, z, text, and symbology fields in each record is irrelevant, since the location of each can be identified during input, This is not true, however, when using the **Ordered Input/Output** loading method because the system expects data fields to occur in this order: x, y, z1, z2,...zn.

Each x, y, z, text, and symbology field must be **justified** to the same column in each record.

Defining Subsets During Loading

There are three ways to maintain the integrity of line-oriented data when loading into **CPS-**3 from **ASCII** files. These methods apply to the loading of **Data**, **Fault**, and **Polygon** sets.

- Use subset markers in the file.
- Encode the identifying name in each record in the file.
- If no subset markers or names exist in the file, then a "difference threshold" criteria will be initiated by **CPS-3** while loading the points to help it identify separate lines. The threshold works like this:
  - Let D1 be the distance between the current point being loaded and the previous point loaded.
  - Let D2 be the distance between the previous point loaded and the point before it.
  - If D1/D2 is greater than the threshold, then the current point being loaded is considered to be the beginning of a new line.

ASCII Data Loading Menu

Most of the parameters on the following menu are self-explanatory, such as the **Number of Z fields** parameter. **File Type** and **Format**, however, may need some extra explanation.

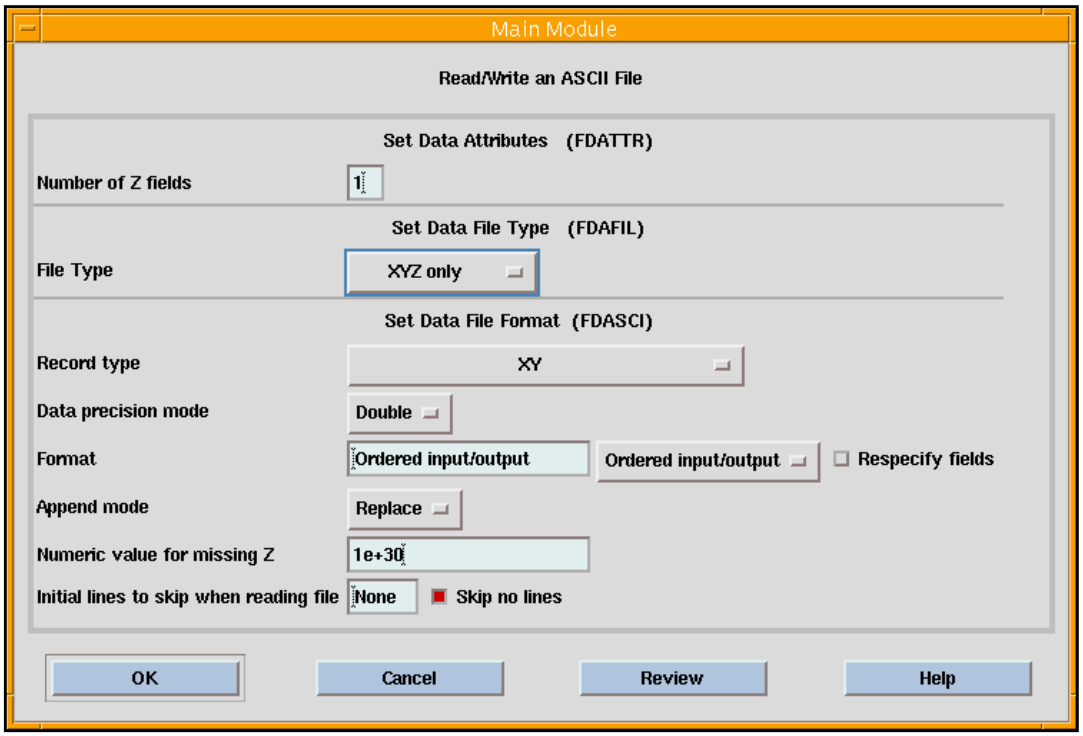


Figure 27.2 Read/Write an ASCII File dialog box

Format

Format actually refers to the manner in which you want to define the contents of the data fields in the **ASCII** file. These options apply to **Data**, **Fault**, **Polygon**, **Surface**, and **Map** set loading.

1. Fortran Format Specification

In earlier versions of **CPS-3,** you could actually specify a **Fortran** format statement to be used by **CPS-3** when loading the different fields in your **ASCII** data file. You can still do this by typing in the format in the menu, or by including the format statement as the first record in the file. However, more convenient methods are available as alternative choices in the menu.

2. Ordered Input/Output

This method causes the system to read data in the following order from each record without the user having to specify a format:

x, y, z1 ....zn

Each field must be separated by a **comma** and/or at least **one space**.

3. Point to Fields

This option provides you with a special menu on which your data file is displayed so that you can pick the column limits for each field you want to read into **CPS-3**.

File Types

There are almost no limits to the way in which **ASCII** data can be organized. However, these predefined formats are simplistic enough to cover just about any kind of data which is loaded into **CPS-3**.

1. X,Y,Z Only

This file format contains only x,y,z values in each data record and is typically read using non-extended mode.

$$x_1 \quad y_1 \quad z_1 \quad \ldots z_{1.n}$$

$$x_2 \quad y_2 \quad z_2 \quad \ldots z_{2.n}$$

$$x_3 \quad y_3 \quad z_3 \quad \ldots z_{3.n}$$

$$\ldots$$

2. X, Y, Z Plus Name Field

This format includes the name (for example, line name or well name) in each record. This type is typically read using non-extended mode. All points having the same name field will be grouped into the same subset with that name. If the name field is a unique well name, then each well becomes its own subset, and its name can be posted during graphic display.

$$x_1 \quad y_1 \quad z_1 \quad \ldots z_{1.n} \quad \text{JOE}$$

$$x_2 \quad y_2 \quad z_2 \quad \ldots z_{2.n} \quad \text{SALLY}$$

$$x_3 \quad y_3 \quad z_3 \quad \ldots z_{2.n} \quad \text{FRED}$$

$$\ldots$$

3. CPS-3 ASCII Format

Unlike other formats, this one incorporates the setting of parameter values at the top of the file and uses subset markers to define subsets. When exported in this format, the parameters associated with the set are automatically recorded by the system. This can be an advantage when transferring sets from one network node to another while still retaining certain set characteristics embodied in the parameters. It is not common practice to load data with these parameters unless reloading one which was exported from **CPS-3**. The main advantage of this format is the use of the subset markers to define individual lines and polylines. As with the other set formats, multiple z-fields may be present.

! comments

! comments

...

native parameter command 1

native parameter command 2

native parameter command 3

...

->subset 1 name

x_1 y_1 z_1 . . .

x_2 y_2 z_2 . . .

x_3 y_3 z_3 . . .

->subset 2 name

x_4 y_4 z_4 . . .

x_5 y_5 z_5 . . .

x_6 y_6 z_6 . . .

...

4. X,Y,Z with Text Fields and Symbology

In this format, multiple text fields exist in each record, as well as symbol codes or other graphic attributes which are to be associated with the individual data point or vertex represented by the record. As with other file formats, multiple z-fields may be present. Data in this format should be loaded with the **Extended Data** loader, so that all fields and symbology can be stored and used.

x_1 y_1 z_1 . . . Text1.1, 1.2 . . . Attribute1.1, 1.2 . . .

x_2 y_2 z_2 . . . Text2.1, 2.2 . . . Attribute2.1, 2.2 . . .

x_3 y_3 z_3 . . . Text3.1, 3.2 . . . Attribute3.1, 3.2 . . .

Extended and Non-Extended Data Sets

When loading **Data** sets, there are two loaders to choose from as mentioned previously - the standard **Data** loader and the **Extended Data** loader.

Load data into **non-extended** data sets if it does not contain, or you do not wish to use, textual or symbology fields. For example, if your **ASCII** data file contains a well name and a well symbol code, but you do not plan to make use of them, load just the x,y,z portion into a **non-extended** data set, ignoring the other fields in the record.

Load data into **extended** data sets if the **ASCII** file contains text, numerics, or symbology fields which you wish to display, or make use of, on a base map.

- Examples of text fields - well name and operator name
- Examples of numeric fields- shot point number and line number
- Examples of symbology - well symbol, well symbol size, and well symbol color

Examples of File Formats

Data Files

Here is an example of an **X,Y,Z Only** data file:

```
6438.0    3593.0    1017.91
6176.0    2636.2     991.80
6903.7    5122.1     967.90
```

Here is an example of **X,Y,Z Only** with multiple z-fields:

```
6438.0    3593.0    1017.91    43.4
6176.0    2636.2     991.80    52.9
6903.7    5122.1     967.90    46.7
```

Here is an example of well data in **X,Y,Z Plus Name Field** format with a well name:

```
6438.0    3593.0    1017.9    R-01
6176.0     636.2     991.80   R-02
6903.7    5122.1     67.90    R-03
```

Here is an example of the same portion of the data file above in which the fields are separated only by commas. This file can be read by selecting the **Ordered Input/Output** format.

```
6438.0,3593.0,1017.9,R-01
6176.0,636.2,991.80,R-02
6903.7,5122.1,967.90,R-03
```

Here is an example of simple 2D seismic data **CPS-3 ASCII** format:

```
! This data contains 2D seismic lines from the Wellington survey
! Use the vertical faults for Block Beta
->RAD-90-001
  5547.700    2264.800    1131.000
  5580.413    2284.136    1124.000
  5613.125    2303.473    1119.000
  5649.281    2324.845    1114.000
  5691.462    2349.779    1108.000
  5744.835    2381.328    1100.000
  5792.182    2409.315    1094.000
  5823.173    2427.634    1089.000
->CON-66-004
  7472.566    3590.510    1021.000
  5251.076    3663.922    1025.153
  6405.266    1892.124    1097.970

...
```

Here is an example of **X,Y,Z with Text Fields and Symbology** with some comments at the top, three z-fields, a well name, operator name, symbol code, and a symbol color associated with each well location.The **Extended Data** loader is best used to load these types of data files.

```
! This file contains example well data,
! for three horizons, Z1, Z2 and Z3,
! including a well name and symbol code
6438.0   3593.0  1017.9  1047.9  1.E+30  R-01  Mobil    2  4
6176.0   2636.2  991.80  1023.8  1054.3  R-02  Mobil    13 5
6903.7   5122.1  967.90  1000.9  1.E+30  R-03  Chevron  18 3
7472.5   3590.5  1021.0  1064.5  1104.0  R-04  Chevron  18 4
5251.0   3663.9  1025.1  1051.5  1.E+30  R-05  Chevron  19 2
6405.2   1892.1  1097.9  1136.9  1187.4  R-06  Chevron  56 3
5596.1   5106.4  993.00  1020.0  1.E+30  R-07  Chevron  42 7
6667.8   4027.9  1039.0  1073.0  1089.5  R-08  Mobil    19 5
5374.2   2944.7  1077.4  1103.4  1111.4  R-09  Chevron  18 5
7054.7   3155.6  1037.7  1070.5  1188.5  R-10  Chevron  18 6
6682.0   6009.0  969.10  1000.1  1.E+30  Z-01  Chevron  12 2
5420.7   4355.6  1009.4  1036.9  1.E+30  R-12  Chevron  17 1
```

```
6505.1   3030.2  1003.0  1034.0  1049.0  R-13  Chevron  42 1
6432.7   4873.2  1039.3  1068.8  1.E+30  R-14  Mobil    16 5
6720.8   3555.2  975.00  1012.6  1035.6  R-15  Mobil    55 4
5705.2   4187.9  996.10  1022.6  1.E+30  R-16  Chevron  16 5
3533.2   3956.2  1135.7  1157.7  1.E+30  Z-02  Chevron  13 1
```

Fault Files

Here is an example of a fault file showing the use of the subset markers to define the separate fault traces for each named fault. It is valid for fault vertices to contain z-values to aid in the gridding process. We would choose a fault attribute of **x,y only** and a file type of **CPS-3 format** for loading this data.

```
->Northwest_vertical
  6101.402      2260.635
  6037.690      2552.553
  5958.145      2881.542
  5867.917      3162.905
  5905.114      2886.946
  5989.938      2520.760
  6101.402      2260.635
->Southern_Green_A
  6398.598      3401.793
  6387.917      3629.874
  6387.917      3667.197
  6377.361      3959.115
  6377.361      4192.600
  6414.432      3969.671
  6430.392      3746.742
  6425.114      3640.556
```

Next is an example showing the same fault traces, but having the fault name encoded on each vertex record, removing the need for the subset markers.We would choose the attribute **x,y only** attribute, and the **X,Y Plus Name Field** file type to load these faults.

```
6101.402    2260.635   Northwest_vertical
6037.690    2552.553   Northwest_vertical
5958.145    2881.542   Northwest_vertical
5867.917    3162.905   Northwest_vertical
5905.114    2886.946   Northwest_vertical
5989.938    2520.760   Northwest_vertical
6101.402    2260.635   Northwest_vertical
6398.598    3401.793   Southern_Green_A
6387.917    3629.874   Southern_Green_A
6387.917    3667.197   Southern_Green_A
6377.361    3959.115   Southern_Green_A
6377.361    4192.600   Southern_Green_A
6414.432    3969.671   Southern_Green_A
6430.392    3746.742   Southern_Green_A
6425.114    3640.556   Southern_Green_A
```

Polygon Files

Here is an example of a polygon lease file with lease names used as the subset names.

```
->Smith_Hawking Lease
  4101.402    2660.635
  4037.690    2852.553
  3958.145    2281.542
  3867.917    3362.905
  3905.114    2786.946
  3989.938    2120.760
->Wiley_Chevron Lease
  6698.598    2401.793
  6787.917    2629.874
  6787.917    2667.197
  6477.361    2959.115
  6477.361    3192.600
```

Data Transformations

As **ASCII** data is imported or exported from **CPS-3**, transforms can be applied to the data to move it, rotate it, scale it, or perform other arithmetic operations. The transformation menu is shown below

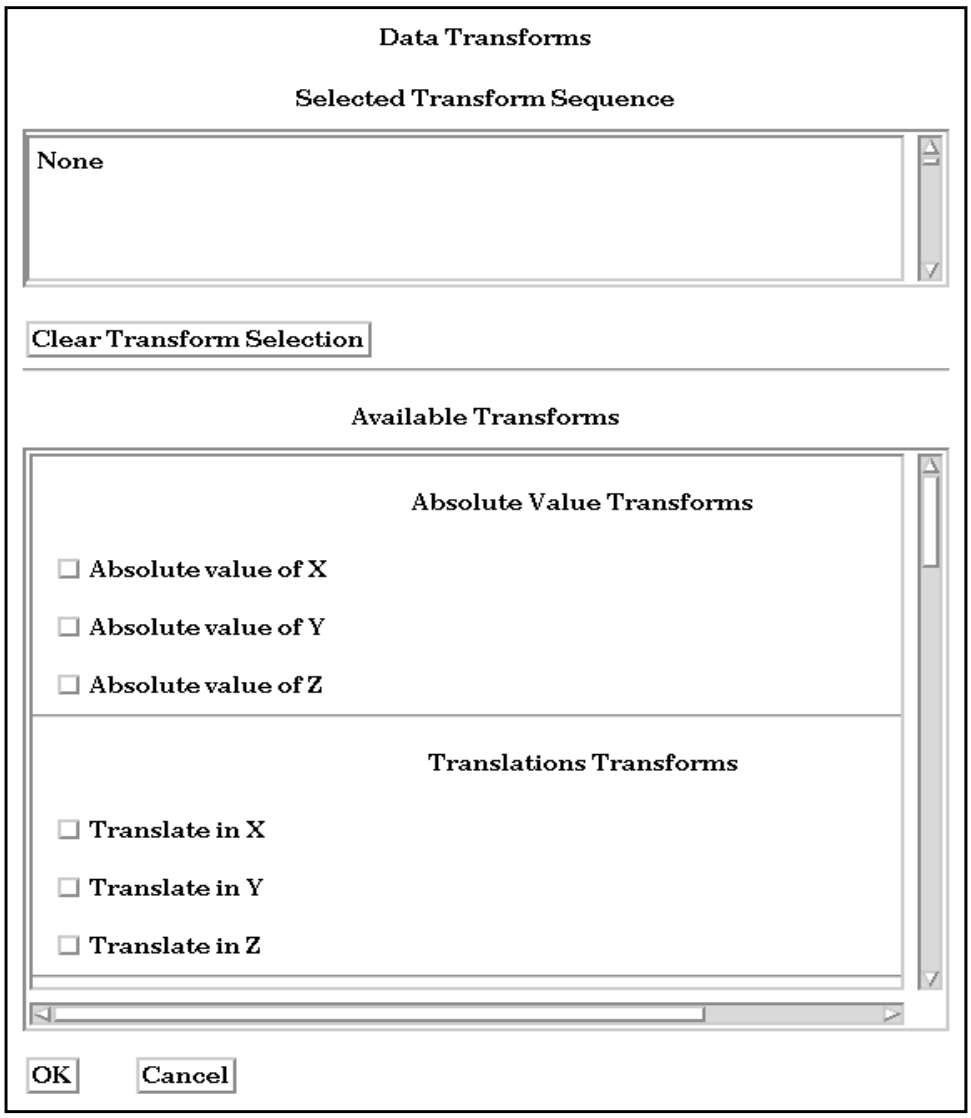


Figure 27.3 Data Transforms dialog box

List of Arithmetic Transforms

Absolute value of X

Absolute value of Y

Absolute value of Z

Translate in X

Translate in Y

Translate in Z

Scale in X, centered about an X-position

Scale in Y, centered about a Y-position

Scale in Z, centered about a Z-position

Rotate parallel to X-axis about a Y-Z position

Rotate parallel to Y-axis about a Z-X position

Rotate parallel to Z-axis about a X-Y position

Min/max limit in X

Min/max limit in Y

Min/max limit in Z

Set min/max limit condition to CLIP

Set min/max limit condition to REJECT (default)

Log of X

Log of Y

Log of Z

Set log to BASE-10 log

Set log to NATURAL log (default)

Reject record if any Z-field is INDT

Reject record if Z-field “n” is INDT

LECTURE for Topic 26
Technical Note - The Convergent Gridding Algorithm Explained

Overview

The **Convergent** gridding algorithm has proven to be a very reliable and predictable general-purpose gridding algorithm. Its innovative methodology has clearly set a standard for the industry. In this chapter, we will give you a brief overview of the mechanics of its operation, so that you can use the information to your advantage.

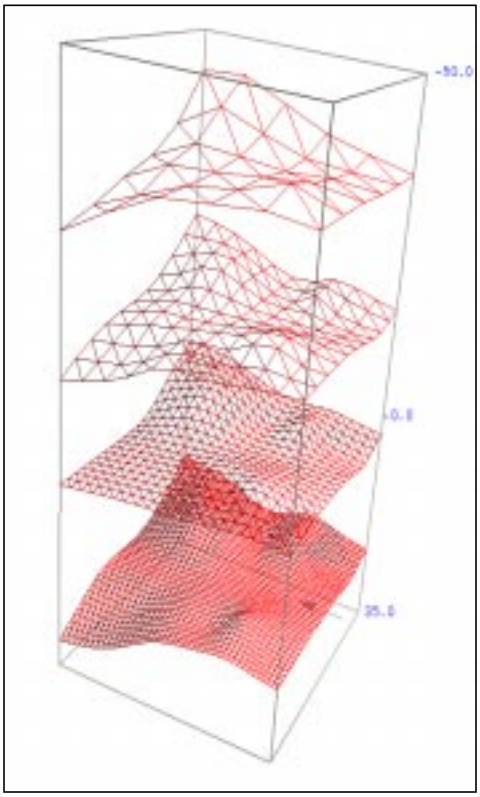


Figure 28.1 Internal surface refinement in Convergent gridding

Iterative Procedure

Convergent gridding gets its name from a process whereby grid node values are converged upon by iteratively assigning control point values to nearby grid nodes. The process starts with a large grid cell size, the **Starting Interval**, in the first iteration, and the process ends when the grid cell size reaches its desired size, the **Final Interval.** The grid is refined to smaller and smaller cell sizes between each iteration. At each iteration, each control point value is assigned to a certain number of the **closest** grid nodes, specified by the parameter **Number of Nodes to snap to**. Thus, the most important parameters associated with the **Convergent** gridding algorithm are:

- **Starting Interval**
- **Final Interval**
- **Starting Number of Nodes to snap to** (**max** = **16**)

At each iteration, the algorithm performs the following operations:

- **assigns** or **interpolates** control point values to nearby nodes. (In the case of nodes already having a values, implement the blending scheme described in the following **Blending Algorithm** section.)
- **smooths** the grid
- **refines** the grid

Through all iterations, the following become **smaller:**

- The grid cell size
- The **Number of Nodes to snap to** parameter (**1** on the last iteration)

As you can see, in the early iterations, each control point contributes to a large geographic area, but in the final iterations, each control point is tied to a very small area. The result of the **Convergent** gridding technique is that areas outside of the data have been modeled with a smooth, trend-like solution, but yet in the middle of the data, the grid ties the data as closely as the final grid size will allow.

In the case of faulted surfaces, the **Convergent** algorithm applies the standard visibility criteria when determining if a particular data point is appropriate to use in the computation of grid node values.

Blending Algorithm

Multiple weighted Z values assigned by several control points to the same grid node are mathematically merged using a sophisticated blending function. The function is based on the **Taylor Series**, which allows the prediction of a shape at any point "**x**" by knowing the behavior of the shape at several points "**a**".

$$f(x) = f(a) + f'(a)(x-a) + \frac{(f''(a))}{2!}(x-a)^2$$

"**X**" is the grid node location, and "**a**" are the data point locations.

The **Convergent** process involves multiple iterations of interpolation, smoothing and refining to achieve a trend like surface in extrapolation and an accurate fit in the presence of data. It starts with course grid interval which controls the extrapolation and ends with a fine grid interval approximating the data density, therefore controlling the accuracy of the model.

Slope and curvature information is calculated on the first iteration and carried from one iteration to the next.

There are certain options available during gridding:

- Gridding inside of a polygon
- Weighting the different control point sets
- Providing dip and strike information along with the z-value in the control points.

We will not explicitly discuss these options in class, but you can read about them in the on-line **User's Guide**.

To help you see how the **Convergent** algorithm computes the final grid, the following schematics of several iterations illustrate the process.

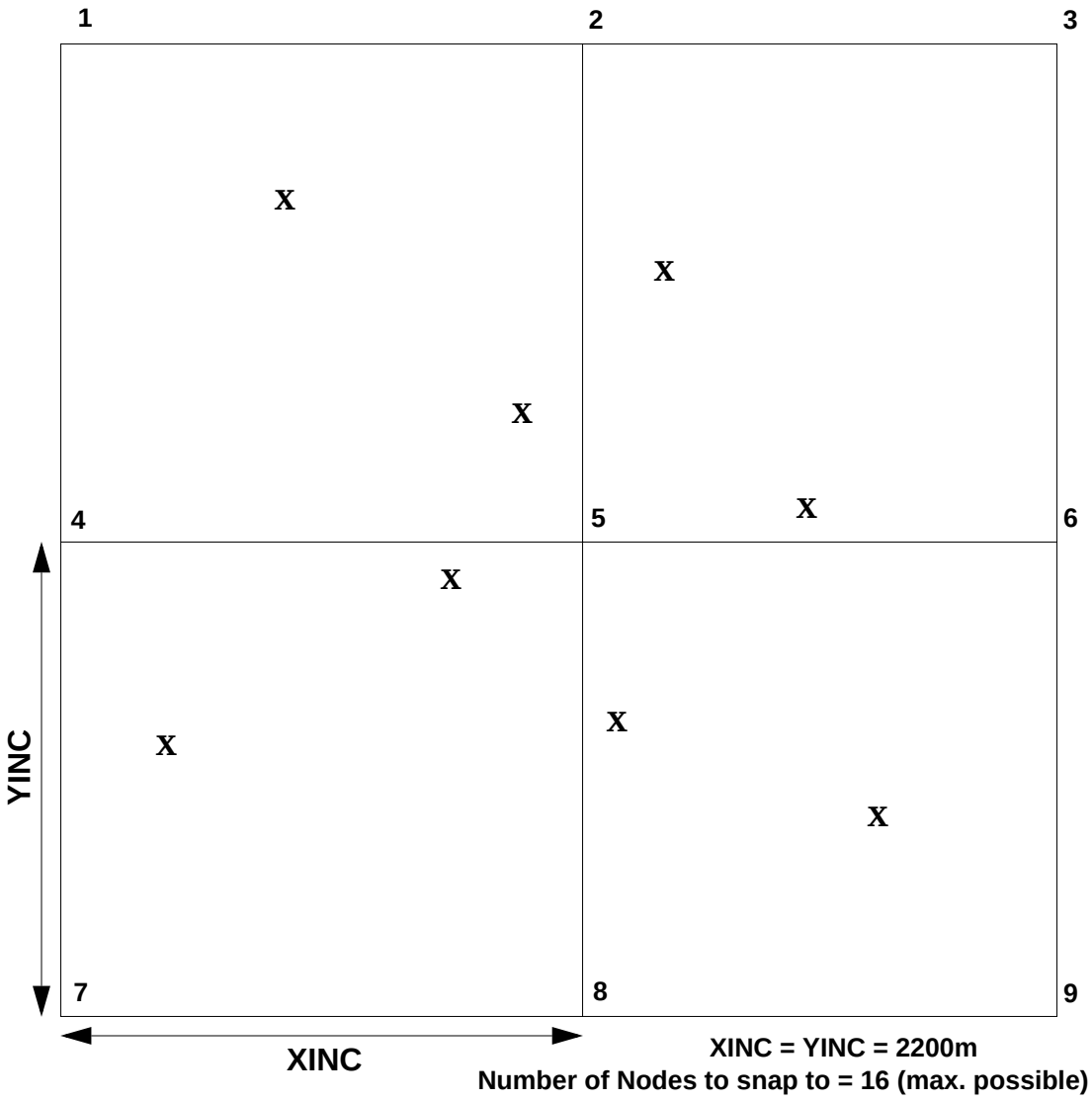


Figure 28.1 Convergent gridding: first iteration

At the beginning of the first iteration, the grid is null, and all control points will contribute, to some degree, to all grid node values. The effect of each data point depends upon its distance to the grid node. At the end of the first iteration, the grid is essentially a weighted average. Note that the amount of overall extrapolation desired around the edge of the data can be controlled by the Initial grid interval.

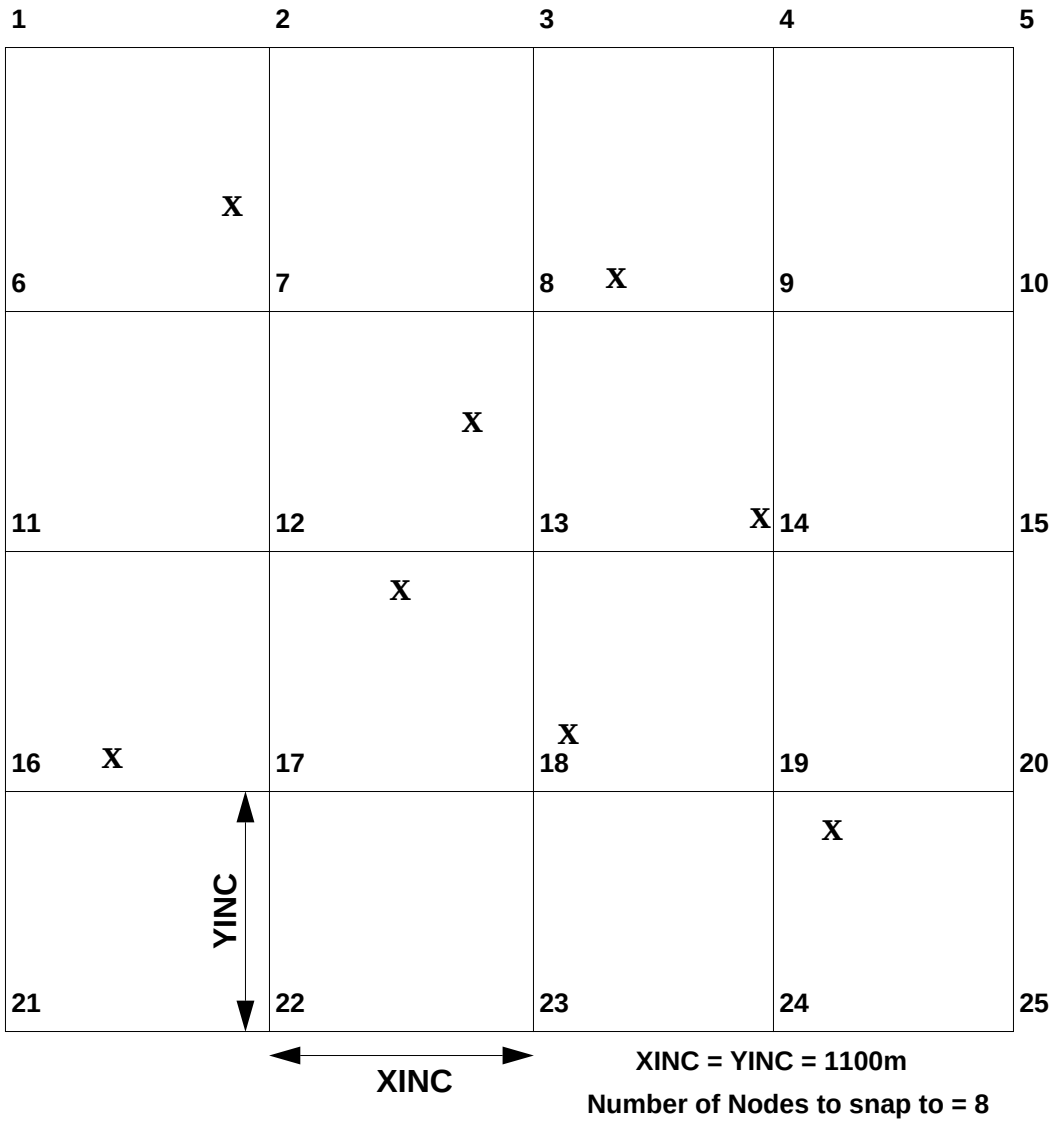


Figure 28.1 Convergent gridding: second iteration

In this iteration, the grid starts with the refined values of the first iteration, and the area affected by each control point has become smaller. For example, the 8 closest nodes for the upper left control point are 1, 2, 6, 7, 11, 12, 8, and 3. Because the grid starts with existing values in this iteration, the assignment of control point values now becomes the projection of control point values using the slopes and curvature computed from the grid.

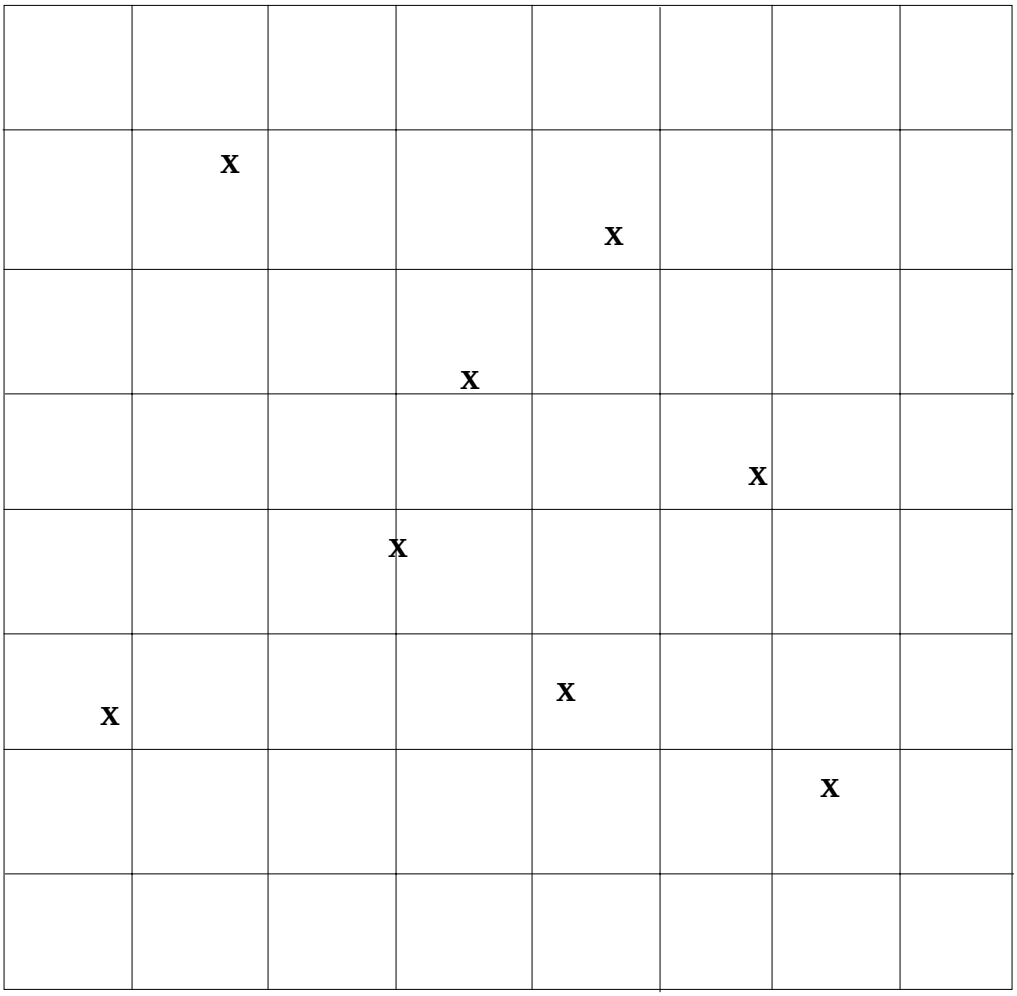


Figure 28.1 Convergent gridding: third iteration

In this iteration, the grid begins with the refined values from the second iteration, and the area affected by each control point has become even smaller. Each control point in this iteration will change only the four nodes of the cell in which it falls, but the overall trend in the grid is retained. The change and readjustment of the four nodes continues as the projection of control point values along the slopes and curvature computed from the grid.

LECTURE for Topic 27
Glossary of Computer Mapping Terms for CPS-3

anisotropic: Having properties that differ according to the direction of measurement. Least-Squares gridding allows for directional bias based on user-defined trends.

Ex. If a region is known to have high porosity trends running northeast-southwest, the user may wish to use anisotropic weighting when gridding.

area of interest (AOI): A rectangular geographic area in which CPS-3 models data, performs data and surface operations and displays 2-D graphics. The AOI boundaries are defined by the minimum and maximum, X and Y engineering coordinates. (See also engineering window.)

audit trail: When activated, the audit trail records all parameters and operations used to create or modify data, grids, faults, polygons and maps. The user can request an audit report for a specific set or all sets in the project.

azimuth: Direction used to specify dip direction or 3-D viewing angle. Measured clockwise from north.

batch execution: The execution of a user-specified list of commands (macro) as a background computer process without user intervention. This mode is useful when immediate results are not necessary or when cup intensive mapping tasks are to be run in offbeat computer load periods. This mode is also used when hardcopy graphics are required.

bathymetric data: x,y,z data sets describing the depth of large bodies of water

biharmonic filter: See smoothing.

blanking: A procedure for setting the value of selected grid nodes to a null (indeterminate) value. Nodes are usually selected by enclosing them in polygons.

bulls-eye: A contour pattern consisting of tightly spaced concentric circular contours which stand out as an anomalous area on the map. This pattern is generally the result of a bad data point, seismic misties, or data that varies greatly, in relation to the contour interval, within a small distance.

cell: The base unit of the grid; bounded by four grid nodes. The dimensions of a cell are defined by the x- and y-increment.

clipping: There are two types of clipping. **1.**Surface clipping--Surface operations can be used to ensure a surface is not greater than/less than a user specified cutoff value. All areas of the surface greater than/less than the cutoff will be set either to the cutoff value or to indeterminate. **2.**Map clipping--CPS-3 allows graphics to be deleted from an existing map set either inside or outside user specified polygons.

color-shade contours: A user specified color palette is used to color the space between contour intervals. A single color from the palette is used to paint the area of the surface falling within a defined contour interval. Adjacent contour intervals are painted with adjacent colors from the color palette.

column: Within a computed grid, all nodes with equal X locations, but different Y locations comprises a single grid column

command: A single word entered at the command line to invoke a mapping function or procedure. (See the CPS-3 Quick Reference Guide.)

conformal limiting: An option used during surface modeling (gridding, surface operations) to control the z minimum and maximum of the created surface. The surface limits can be controlled by either user defined constants or surfaces.

container: A named collection of information in GeoFrame. For example, a surface container holds various versions of gridded models for a specific horizon and the fault traces which are associated with them; a data container holds various versions of data points or scatter points for a particular horizon.A container has no intrinsic data except as a holding tank for other collections of information.

contours: There are two types of contours within CPS-3. **1.**Normal contours--lines on a map joining points of equal z-values (elevation). **2.** Orthogonal contours--These are generally drawn perpendicular to normal contours to indicate direction of flow.

contour interval: The difference between z-values of adjacent normal contours (see contours).

contour-to-grid: See gridding.

control points: A point on a map represented as X,Y,Z1,Z2...Zn where X and Y determine the location of the point on the map and Zn represents the value of surface n at that point.

convergent gridding: See gridding.

coordinate system: A means of spatially locating X and Y data on a flat plane such as a map. It may be a simple XY coordinate system in which the area has been defined by a minimum to maximum X and Y values. If the data is located by another means such as latitude and longitude, the X,Y axes must be defined with those coordinates. Latitude/Longitude may transformed to other coordinate systems via different projections (UTM, Lambert, etc).

cultural data: Any information, such as political boundaries, roads, rivers and lease boundaries used as reference points on a map.

data transformation: Any function that can be performed on the data during it's import/export into CPS-3. System functions include scaling, shifting the origin, rotation, or converting from latitude/longitude to X,Y.

default (default value): A software supplied answer to a question posed by the program. Most default values represent a typical situation. The default value may be changed or left as is.

density gridding: See gridding.

deterministic data: Control points with very precise Z-values, such as well data.

digitize: Method of directly entering new data into CPS-3 via a mouse attached to a digitizing tablet or the screen.

digitizing tablet: A peripheral device for converting 2-D picture (hardcopy) data such as cultural data, contours, or well locations into CPS-3 format.

dip: The amount of slope at a particular control point measured in degrees from horizontal.

distance gridding: See gridding.

edge effect: Term used to describe undesirable gridding (contouring) results beyond the edge of the data limits. This is due to erroneous extrapolation outside the data limits; most often associated with least squares gridding.

engineering units: The units of measure contained within the users' data, usually feet, meters, miles, or kilometers.

engineering window: A three dimensional geographic area in which CPS-3 models data, performs data and surface operations and displays graphics. The boundaries are defined by the minimum and maximum X, Y and Z engineering coordinates. The Z limits are used for 3-D graphic displays only. (see also area of interest).

extrapolation: The process of projecting, extending, or expanding slope and gradient information from known data into an unknown area. CPS-3 uses extrapolation during the gridding process.

fault boundaries: These are equivalent to polygonal fault traces (see below). In IESX, fault boundaries can be digitized directly, or initialized from the fault contacts (see below).

fault contacts: Those points computed in IESX which are the actual or projected intersection of an interpreted fault surface and a specific interpreted horizon along a seismic line. Viewed in plan view, the fault contacts for a particular horizon and fault approximate the fault boundaries which, in IESX, can be derived from the contacts. Fault contacts in IESX can be identified by horizon, by fault, and by upthrown or downthrown side. Typically, the fault contacts are used as a guide to digitize or initialize the fault boundaries (polygons).

fault cuts or fault segments: Seismic interpretation of a fault surface along one or more seismic lines. The collection of fault cuts for a particular fault can be used as data points to create a gridded model of the surface.

fault intersections: These are equivalent to **fault boundaries**, **fault traces**, and **fault polygons**.

fault trace (fault): A line marking any discontinuity (abrupt change in elevation or slope) in a surface model. Typically a fault trace marks the intersection of a fault surface with the modelled surface as viewed from the top. Fault traces are minimally defined by X,Y vertices. A more complete fault definition can include elevation (Z) and vertical throw (T) in the format of (X,Y,Z,T). Non-vertical faults are represented by closed polygons (fault traces) and vertical faults are represented by lines.

field: Within an ASCII data file, a field contains one item of information in a computer record (a record may have one or more fields). In multiple records, a given field should always contain the same type of data. An example is a well record in which one field stores the well name, an X-field contains latitude, a Y field contains longitude and the Z-field contains a formation top.

filter: There are two types of filters within CPS-3. **1.File filter**--The graphical interface allows the user to limit the file list when dealing with files external to CPS-3. Ex. When importing ascii fault files into CPS-3, the user, by default, is allowed to select the file from a list of all existing **.flt** files. If the users file has a different extension (**.dat**), the filter can be changed to **.dat**. The user will then be allowed to select the correct file. **2.Smoothing Filter**--There are two types of filters which may be used during the smoothing process: **a.** The biharmonic filter is a converging filter requiring one or more passes through the surface to converge on the final smoothed surface model. This filter minimizes curvature without affecting local slopes. This is usually the preferred filter and is always used during snap or convergent gridding. **b.** The ring convolution filter is a non-converging filter that passes through the grid a user-specified number of times to create the final smoothed surface model. This filter reduces both slope and curvature. (see also smoothing).

fishnet isometric: A common method of representing a surface in three dimensions. The user specifies an azimuth and elevation from which to view the surface. The surface is represented by a grid lattice where grid nodes are placed at their correct grid elevation revealing the peaks and valleys of the model. There are two such types of 3-D displays. **1.**No Hidden Line Removal (Fishnet)--All segments of the grid lattice are displayed regardless of whether they would be obscured from the users view because they fall behind other features of the surface. **2.**Hidden Line Removal (Isometric)--Grid lattice segments which fall behind portions of the surface being displayed are not displayed.

graphic displays: Two or three-dimensional map view representations of any data, cultural information, faults, surfaces, etc.

grid: A model showing the distribution of a user defined attribute such as depth, porosity or contaminant concentration. Often referred to as a surface. The grid consists of a set of ordered Z (attribute)-values occurring at regular intervals of rows and columns usually calculated from a set of user-defined attributes at irregular X,Y locations. As a verb, grid is the process of creating the model from the data (see also gridding).

grid blanking: See blanking.

grid cell: See cell.

grid column: See column.

grid increment: See increment.

grid node: See node.

grid refinement: See refine.

grid row: See row.

gridding: The basic computer mapping process for transforming user x,y,z data into a regularly distributed x,y,z data set referred to as a grid. There are several algorithms within CPS-3 for creating grids.

contour-to-grid--Modified convergent gridding algorithm used for creating grids from digitized contour data.

convergent--Iterative gridding process which begins with a very coarse grid increment which continually refines and reties data to the grid until the final grid increment is reached. Excellent algorithm for all common data type Minimizes edge effects producing a model more closely resembling a hand drawn contour map.

density--Grid node values are set to the number of valid control points within the search limit radius.

distance--Grid node values are set to the distance between the grid node and nearest control point.

isopach--Specialized gridding for data sets where a zero value indicates the modelled attribute is not present at that location. Commonly used for gridding isopach data. Uses a data preprocessor to project a proper zero line and assign the appropriate negative number to the zero data value before gridding takes place. Can be used with both convergent and least squares gridding techniques.

least squares--One of the oldest methods of computing grids. Can be used for all types of data except digitized contours. Should not be used where extrapolation across large areas is required due to undesirable edge effects.Can use dip and azimuth information at control points if available. Allows for anisotropic weighting of the data.

moving average--Does not consider slope or curvature information when creating the grid. Performs a weighted average on data within the Search Limit Radius to calculate the grid node value. Not recommended for most data types. Produces acceptable results for point-source data or dispersion modelling

polynomial--Grids created from the function:
Z=C1+C2*X+C3*Y+C4*X^2+C5*X*Y+C6*Y^2+C7*X^3+C8*X^2*Y+C9*X*Y^2+C10*Y^3

where C# is a user-specified constant. A constant value grid is created by setting C1 to the constant value and C2-C10 to 0.

gridding (**continued**)...

snap--A single iteration of convergent gridding. Data is tied to the nodes of a new or existing surface. Excellent for quick look gridding (especially 3-D seismic).Also used to tie an existing surface to new or modified data.

step--Grid node values are set to the value of the closest control point.

trend--Used to generate regional trend and residual maps. An approximation of the data model which reveals gross features only and hides local data variations.

horizon: A particular stratigraphic geologic sequence. In most instances this may also be referred to as a grid or surface. FFMS uses the horizon name as a key word for naming the surfaces and faults it calculates.

increment (interval): This is the basic measuring unit of a grid. It corresponds to the X, Y dimensions of a grid cell and determines the resolution of the map or model.

indeterminate value (INDT): A value used to indicate a null condition such as the absence of data at a particular location. When assigned to a grid node, the surface is undefined in that area. The abbreviation used by CPS-3 is INDT. The value is internally stored as 1.0E+30.

interactive execution: The processing mode in which the software performs operations immediately upon command of the user. Graphic output is normally sent to a terminal for immediate viewing. This mode is sometimes referred to as foreground execution. (see also batch execution).

interpolation: A process used during gridding to estimate values that lie between two known values.

inverse interpolation: A process that back-interpolates a z-value from a grid at a given xy location or cursor location as in the browse facility in the model editor.

isochore: A surface which represents the true vertical thickness between two other surfaces. Often the result of subtracting two surfaces.

isometric display: See fishnet isometric

isopach: Literally it is the true stratigraphic thickness between two other surfaces. This surface is always greater than or equal to the isochore thickness. CPS-3 does not calculate true isopach surfaces. However, many people use the term isopach synonymously with isochore, particularly when the bed dip angle is negligible.

isopleth: A grid representing the spatial distribution of some property or attribute.

isotropic: Having equal properties in any direction of measurement. Most gridding assumes isotropic properties. (see also anisotropic).

lattice: The wire-mesh figure created by drawing line segments through each column and row in a grid.

least squares gridding: See gridding

macro: A text file consisting of one or more commands which perform one or more mapping tasks. These can be built interactively using the macro build command within CPS-3 and recording your commands or macros can be built using the macro builder under the CPS-3 Application Manager.

map: The entire graphic image generated by CPS-3. Maps may be displayed on the screen or sent to plotters for hardcopy output.

menu: A list of options presented to the user for selection. Menu choices may invoke mapping functions or lead to other menus.

moving average gridding: See gridding

multiple surface operations: Algebraic or logical operations which can be performed between two surfaces.

native commands: Low-level commands underlying CPS-3 that provide access to the functions of the subroutine library. (see also parameter, procedure)

node: The intersection points of the rows and columns of the grid lattice. These are the locations of the calculated Z-attributes of the grid model.

normal contours: See contours

orthogonal contours: See contours

parameter: User- or system-specified values that a procedure uses to perform a task. For example, the native command FSGRID is a parameter specifying the initial grid interval for gridding. The naming convention for CPS parameters is as follows:

Fcxxxx--where "**F**" signifies this is a parameter, "**c**" identifies the category (system, data, fault polygon, surface, map), and xxxx is a mnemonic descriptor. (See also procedure.)

plotting units: The units of measure used to define the size of graphical items posted on the plot or the size of the plot itself. The default units of measure are inches but can be changed to centimeters.

plotting window: A rectangle which positions the engineering window on the plotting device (screen or plotter). If a map scale is not set, displays will be created to fit within the specified treated as a temporary work file or permanent archive from session to session. It stores data, faults, surfaces, maps. The project file will be obsolete from CPS-3 v4.0 onward.

projection: Used to convert data defining a three-dimensional body, such as a sphere, into a two-dimensional drawing (a flat plane). In cartography, many different projections exist to represent points on the earth in map view. Example: Mercator, Transverse Mercator, Lambert Conformal Conic.

refine: The process of changing the X- and Y-increment of an existing grid to a different (larger/smaller) X- and Y-increment by resampling the existing grid.

report file: During each interactive CPS-3 mapping session a file called **project_name.rep** is created. This file contains a summary of all the commands and parameters used during that session.

response file: A text file that stores all the user responses made during creation of a macro within the Macro Builder Module. The response file is used to edit a macro previously created with the module. Response files have an **.rsp** extension.

ring convolution filter: See filter.

row: Within a computed grid, all nodes with equal Y locations, but different X-locations comprises a single grid row.

search limit (SLM): The length of the radius used to describe a circle which contains all control point data used to calculate a value for a grid node at the center of the circle. A search limit radius is used in least squares and moving average gridding algorithms.

single surface operations: Algebraic or logical operations which can be performed on an existing surface.

smoothing: A procedure performed on a surface (grid) to reduce the surface curvature and produce a surface that is smooth and free of irregularities. There are two types of filters which may be used by the smoothing process. (also see filters).

snap gridding (snapping): See gridding.

statistical data: Control points with relatively imprecise Z-values, such as magnetometer and seismic data. Statistical data is not meant to be honored as strictly as deterministic data.

step gridding: See gridding.

strike: The direction of a line formed by the intersection of a fault with a surface. In the CPS-3 FFMS module, by default, this is measured in degrees positively counter-clockwise from north.

subroutine library: The core of the CPS-3 system. The library is comprised of a set of highly organized subroutines for performing all of the mapping functions available with the software.

surface: See grid.

surface strike: The direction normal to the dip of a surface. Normal contours follow surface strike.

switch: See toggle switch

symbol code: A numeric code used by CPS-3 to post the appropriate symbol at a particular location. Numbers 1-65 represent stroked symbols. The default symbols and codes are shown in Appendix C of the CPS-3 User's manual. Symbol numbers 1001 and greater are machine symbols and are posted much quicker.

throw: The amount of vertical separation across a fault trace.

toggle (switch): To change the status of a parameter or other item from ON to OFF or vice versa. CPS-3 provides a number of global parameters (also called “switches”) that can be turned ON or OFF. Example. **INDT** is a switch which when ON displays all graphic items. When it is OFF, only items with a valid z-value are displayed.

transformation sequence: The order in which X,Y and/or Z values are manipulated during CPS-3 input/output. (See data transformation.)

trend gridding: See gridding.

unconformity: A surface of erosion or non deposition that separates younger strata from older rocks.

vertices: The x,y locations defining a polygon or fault.

viewport: See plotting window.

volumetric: The computations performed on one or more surfaces to compute the volumes within an enclosed area.

X-increment: The length of a grid cell along the x-axis measured in engineering units. The distance between grid columns.

Y-increment: The length of a grid cell along the y-axis measured in engineering units. The distance between grid rows.

Z-value: Value of any attribute to be modeled at a specific location. Typical attributes are depth, porosity and saturation values. Data points, grid nodes and fault vertices can all contain z-values.