

# GRINCH User's Manual

*TR88-044*

*Revised-January 1988*

*Helga Thorvaldsdottir*

The University of North Carolina at Chapel Hill  
Department of Computer Science  
CB#3175, Sitterson Hall  
Chapel Hill, NC 27599-3175



*Originally published by Douglas B. Schiff, July 8, 1984*  
*UNC is an Equal Opportunity/Affirmative Action Institution.*

## Table Of Contents

1	Introduction	1
2	Introduction To GRINCH	2
2.1	Overview Of The System	2
2.1.1	Purpose of GRINCH	2
2.1.2	Features of GRINCH	3
2.2	Hardware/Software Environment	5
3	How to Use GRINCH	6
3.1	Preparing to Use GRINCH	6
3.2	Tutorial	6
3.2.1	Command Conventions	7
3.2.2	Executing GRINCH	7
3.2.3	Saving and Quitting	9
3.2.4	Dynamic Controls	10
3.2.5	Commands for Viewing	11
3.2.6	Commands for Interpreting	13
3.2.7	Building the Model	17
3.2.8	Miscellaneous Commands	19
4	Reference Manual	20
4.1	Notation	20
4.2	Commands	21
4.2.1	Viewing	21
4.2.2	Interpreting	24
4.2.3	Program Control	30
4.2.4	Escape Menu Commands	34
4.2.5	Miscellaneous	35
5	Creating a Ridge-Line Graph	36
5.1	Data and Information Needed	36
5.1.1	The Map File	36
5.1.2	The Map Information File	36
5.2	Executing mkskel	39
5.2.1	Command Options	39
5.2.2	Mkskel Output Files	40

# Chapter 1

## Introduction

This manual is an introduction to, and reference manual for, the GRINCH computer graphics system for interpreting electron density maps. It is intended for both first time and experienced users.

Chapter 2 describes the system. It presents an overview of GRINCH's functions and features and describes the computer environment of the system.

Chapter 3 is a tutorial for the first time user. It provides instructions for the most commonly used commands and should be worked through sequentially while running the program.

Chapter 4 is a reference manual. It provides detailed definitions and explanations of all GRINCH commands. Commands are grouped by function and listed alphabetically within groups.

Chapter 5 is a tutorial for creating GRINCH ridge-line graphs from electron density data.

The index provides reference for all commands.

## Chapter 2

### Introduction To GRINCH

GRINCH (Graphical Ridge Line Interpretation - Chapel Hill) is a tool to help the user interpret electron density maps of crystallized proteins. It was originally developed at the University of North Carolina at Chapel Hill in 1982 by Thomas Williams, as part of his work towards the Ph.D. in computer science, under the supervision of Dr. Frederick P. Brooks, Jr. Since then, visiting biochemists and crystallographers have done extensive work using the system. Their feedback has been used to expand and refine GRINCH. As it stands, it is the work of Williams, Tom Hern, Mike Pique, Dore Rosenblum, Doug Schiff, Brian VanDuzee, Lee Westover, Jih-Fang Wang, Sam Black and Helga Thorvaldsdottir.

#### 2.1 Overview Of The System

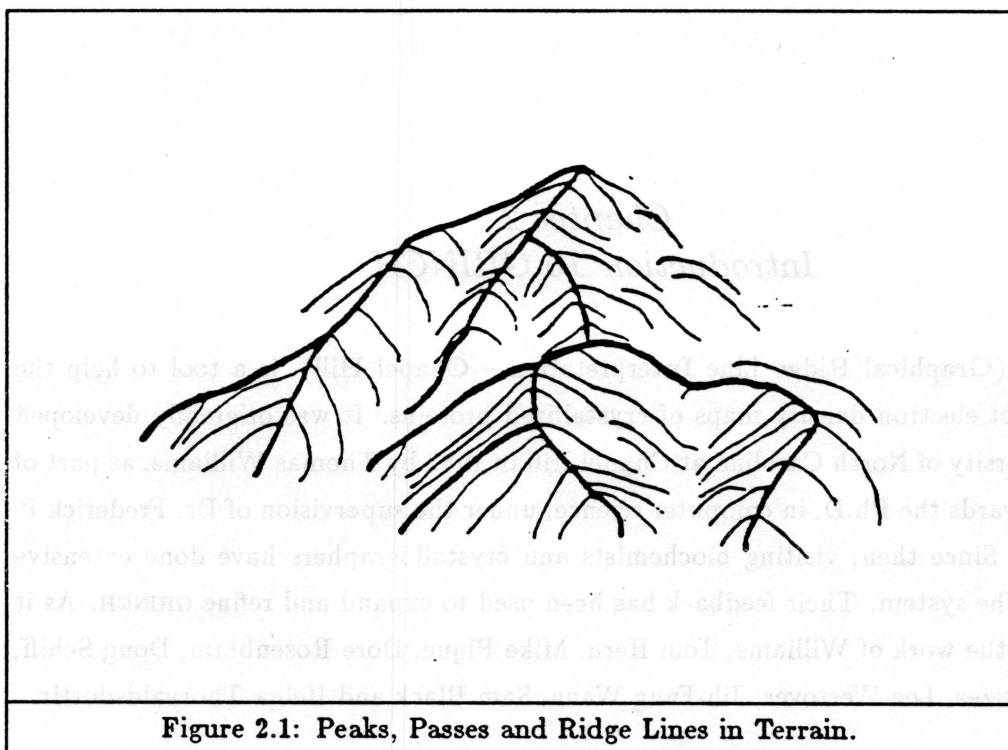
##### 2.1.1 Purpose of GRINCH

GRINCH was designed to supplement or replace the manual *mini-map* method of interpreting electron density maps. It provides essentially the same functions but with far greater power.

The principal function is the identification of the structures of a protein molecule. The GRINCH user is able to locate efficiently and to identify the main chain, as well as secondary structures, of a protein molecule. The user can then continue the process to identify sidechains and carbonyl oxygens.

In addition, GRINCH provides functions to build a three-dimensional model of the protein molecule. Amino acid residues can be selected from a menu and associated with the sidechains and carbonyls identified in the density data. This model, its residue types, atom types, and atom coordinates are available as output from the program.

GRINCH is one component in a larger analytic context. GRINCH provides no functions for modifying the model; bonds cannot be rotated, and angles cannot be adjusted. After identifying atoms which account for the electron density, the user must rely upon other systems to *fit* those atoms to the electron density; e.g. GRIP-75 [Tsernoglou 1977.] [Britton 1977.] [Brooks 1977.] or FRODO [Jones 1979.].



### 2.1.2 Features of GRINCH

**Representation of the Density Map.** The key element of the GRINCH system is its ridge-line representation of electron density maps. Ridge lines connecting local density maxima are analogous to ridges in a geographic terrain.

In describing terrain, local maxima are called *peaks* and saddle points are called *passes*. *Ridges* are the paths of steepest ascent from passes to peaks (Figure 2.1). That is, they run from peak to pass to peak. Each point on a ridge is a local maximum with respect to points in the direction perpendicular to the ridge. Precisely the same concepts describe ridge lines in electron density.

Ridge lines can be also be thought of as lines running through the centers of the tubes formed by a contour-line representation of electron density. For a detailed discussion of ridge lines see Williams' dissertation [Williams 1982].

Representing electron density with ridge lines offers three advantages over contour-line representation:

First, ridge lines approximate the stick-figure model of a molecule. Therefore, ridge lines representing the density of a protein structure often resemble the user's intuitive notion of the shape of that structure. This similarity aids identification.

Second, far fewer lines are required to represent density with ridge lines than with contour lines. This is important for two reasons. The number of lines that can be displayed on a graphics device is limited; the number of lines a user can successfully comprehend is also limited. Consequently, one may view a larger portion of a map and have a more global view of a ridge-line representation than of a contour-line representation.

Third, because there are relatively few ridge lines, each edge of a ridge line can be associated with a specific level of density (that of its mid-point), and ridge lines above a certain density level can be selectively displayed. With GRINCH, a user can dynamically change the level of density above which lines are displayed. This dynamic level-changing is a powerful aid in interpreting electron density and is generally available only with considerable delay for contour-line representations.

**Display Features.** GRINCH provides a full array of display functions, controlling both the objects to be displayed and the user's view of these objects. The user can display the ridge-line map and/or the constructed model. The user may also select various features for display, such as edges identified as belonging to the mainchain, or edges that have not yet been interpreted. The scale and center point of the viewing window can be changed interactively (short delay). The objects can be rotated along two or three axes (depending on the configuration) dynamically (no delay) with a joystick, dials or the keyboard.

**Interpretation Functions.** GRINCH uses different colors to identify features of a ridge-line map. With a picking device (tablet, puck, or mouse) the user may select ridge-line edges and color them according to type, e.g. green for mainchain, purple for sidechain. The edges may be colored singly, or GRINCH can locate contiguous paths of edges of a single type to speed interpretation.

GRINCH also provides functions for building a model of the protein molecule. The user selects from a menu of amino acids and indicates the part of the interpreted map that corresponds to that residue. Using a least squares approximation, a model of the residue is *fitted* to the designated density and is connected to existing adjacent residues. An additional feature allows the user to compare a known sequence of amino acids with a sequence built on the system. The feedback provided by GRINCH is discussed in detail below.

The user can also add *landmarks* to the original map. These can be any configuration of line segments to mark locations in the density map, e.g. metal atoms or helices.

## 2.2 Hardware/Software Environment

At UNC, GRINCH is run on a VAX-11/780 with the 4.2 BSD UNIX operating system and on a MASSCOMP MC5500 with RTU UNIX. GRINCH is written almost entirely in the C language, except for some display functions that are dependent on specific display devices.

**Displays.** At this time, the complete GRINCH system can operate on four different display devices. One is the Adage-Ikonas RDS-3000 graphics system, a high speed raster display. The display routines are written in GIA2 [ Gary's Ikonas Assembler 1982.], a locally developed compiler for the AMD 2900 based microprocessor in the Ikonas.

The second display is the Evans and Sutherland Color Picture System 300, a color line-drawing display. Routines for this device are written in the graphics language provided with the display.

The other two displays are components of standalone, color-graphics workstations; the IRIS 3100 and the MASSCOMP MC5500.

**Input Devices.** The GRINCH system on the PS300 uses the display's input devices.

- The terminal keyboard is used to begin execution and to give some commands.
- Three dials on the PS300 dialbox are used to control rotation of the display.
- A stylus and data tablet are used to select items from the menus and to designate edges in the display.
- A dial is used to control dynamically the minimum density level of edges displayed.

When run on other graphics displays, different input devices are used to provide the same functions.

**GRINCH Ports.** In addition to the configurations supported at UNC, GRINCH has been ported to three different host environments:

- a VAX running VMS using the E & S PS300 as the display.
- an IBM mainframe running MVS using the E & S PS300 as the display.
- an IRIS graphics workstation running UNIX

## Chapter 3

### How to Use GRINCH

This chapter provides an introduction to GRINCH for the new user. It includes step-by-step instructions for executing the program, for controlling the system, and for producing output. Unless otherwise noted, the procedures described are applicable for all GRINCH sites regardless of the host computer system.

#### 3.1 Preparing to Use GRINCH

To proceed with the tutorial you must complete the following preparations.

**The Host System.** You must have a basic knowledge of the host system. On a system running UNIX, you must know how to *log on*, how to access the GRINCH program, and how to access files of data.

**The Graphics System.** You must know which graphics device you can use for display and which analog-to-digital input devices (joysticks, dials, etc.) you can use for control.

**Preparation of Data.** You must have a ridge-line graph, as briefly described in Chapter 2, for loading into the GRINCH program. Chapter 5 explains how to make a ridge-line graph from electron density data.

#### 3.2 Tutorial

This tutorial is intended to be worked through sequentially while actually running GRINCH. Many GRINCH commands are introduced and explained here; detailed descriptions of all commands are found in Chapter 4, the reference manual.

Most instructions described below are machine and operating-system independent. A few commands, such as the start-up procedure, are not. The presumed system context for this tutorial is the UNIX operating system. Information on different systems should be obtained from local computer personnel and from other versions of this manual.



### 3.2.1 Command Conventions

- All GRINCH commands use prefix syntax; the command is given first followed by any arguments.
- Commands may be either picked from a menu or entered from the keyboard unless otherwise noted. Commands entered from the keyboard must be followed by hitting the *return* key.
- Throughout the manual commands are printed in **boldface**.

### 3.2.2 Executing GRINCH

**Running Interp.** Begin by logging on to your host computer. The command to execute GRINCH is **interp**. The command is optionally, but usually, followed by specifying the graphics display device to be used, preceded by a hyphen (e.g. -mc500, -ps, -vg, -ik). For example, if you are using an Evans and Sutherland PS300, type

```
interp -ps
```

If the file *interp* is not in your current working directory, you may have to give its full pathname. Here at UNC you would type

```
/unc/grip/rl/interp -ps
```

You should now have GRINCH running. Messages will appear both on the terminal screen and on the graphics display indicating the initialization procedure being run. The main menu and a cursor will next appear on the graphics display (Figure 3.1). The cursor is a small square when it appears in the graph area of the display. It changes to a diamond shape in the the menu area. When you make a *pick* it changes briefly to a triangle, indicating that the pick has been registered. In addition, when a command is picked, the cursor changes color to that of the command as it appears in the menu and retains that color until the command is completed. This is discussed more fully below.

When initialization is complete, a '#' symbol will appear on the terminal screen. This is the prompt for the GRINCH command interpreter indicating that the program is ready for the next command. (Note: Had you not included **-ps** when you started the program, you could now give the command **displayon ps** to get the same result. See the reference manual for details.)

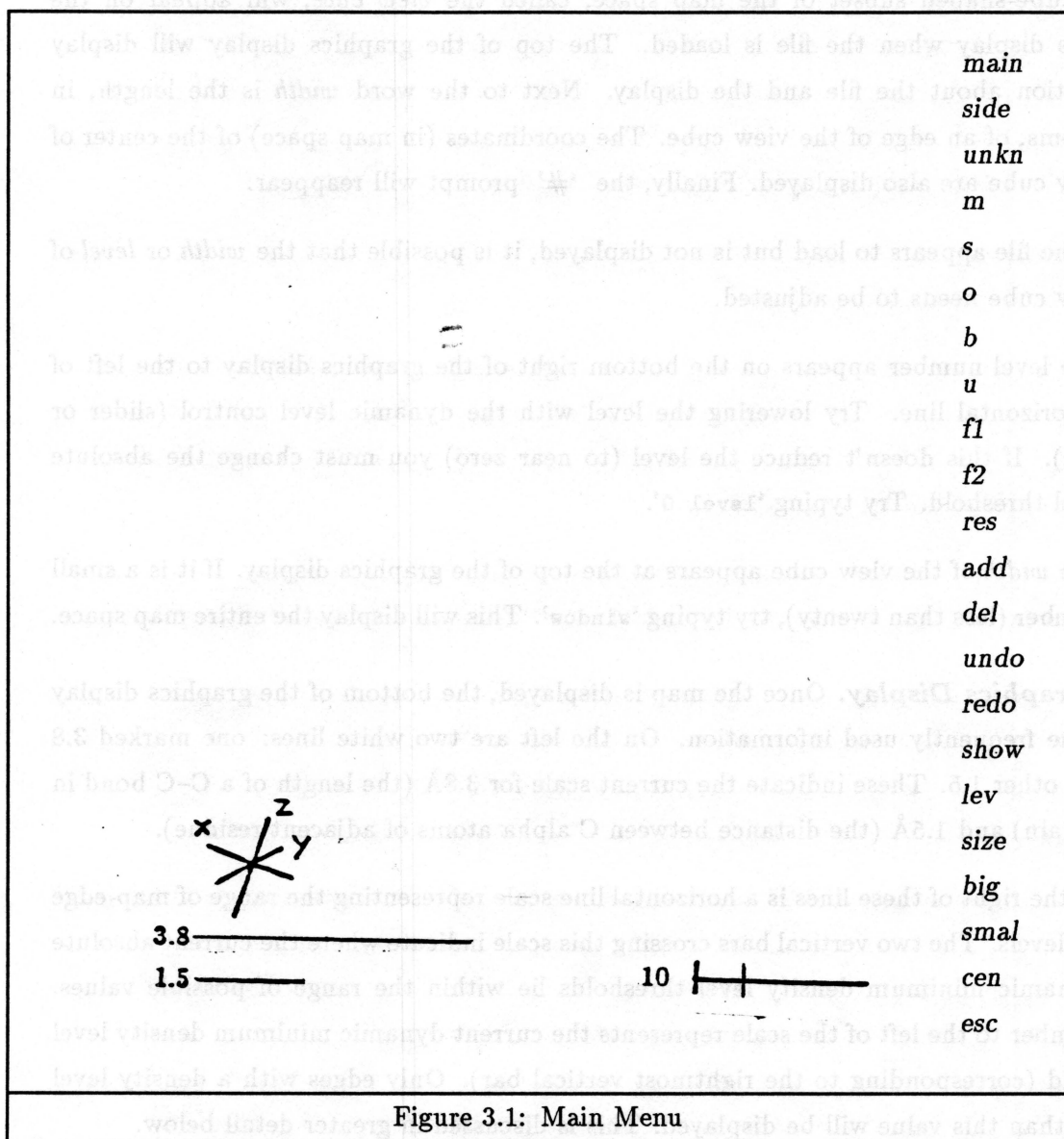


Figure 3.1: Main Menu

**Loading a File.** You are now ready to load a ridge-line graph. If the name of the file containing the graph is *protein.g* (.g for graph), type the command

```
load protein.g
```

If the file is not in your current working directory you will need to give its full pathname. If the error message 'protein.g can't be read' appears, check the name and/or pathname of the file. The time it takes to load varies considerably with the size of the file, as well as other system related factors. A wait of a minute or so is not uncommon. (Note: The file to be loaded can also be specified as the last argument when starting up the program. For example, you could have typed `interp -ps protein.g` to achieve the same result.)

A cube-shaped subset of the map space, called the *view cube*, will appear on the graphics display when the file is loaded. The top of the graphics display will display information about the file and the display. Next to the word *width* is the length, in Ångstroms, of an edge of the view cube. The coordinates (in map space) of the center of the view cube are also displayed. Finally, the '#' prompt will reappear.

If the file appears to load but is not displayed, it is possible that the *width* or *level* of the view cube needs to be adjusted.

1. The level number appears on the bottom right of the graphics display to the left of a horizontal line. Try lowering the level with the dynamic level control (slider or dial). If this doesn't reduce the level (to near zero) you must change the absolute level threshold. Try typing 'level 0'.
2. The *width* of the view cube appears at the top of the graphics display. If it is a small number (less than twenty), try typing 'window'. This will display the entire map space.

**The Graphics Display.** Once the map is displayed, the bottom of the graphics display has some frequently used information. On the left are two white lines: one marked 3.8 and the other 1.5. These indicate the current scale for 3.8Å (the length of a C-C bond in a sidechain) and 1.5Å (the distance between C alpha atoms of adjacent residue).

To the right of these lines is a horizontal line scale representing the range of map-edge density levels. The two vertical bars crossing this scale indicate where the current absolute and dynamic minimum density level thresholds lie within the range of possible values. The number to the left of the scale represents the current dynamic minimum density level threshold (corresponding to the rightmost vertical bar). Only edges with a density level greater than this value will be displayed. This is discussed in greater detail below.

At the bottom left is a set of coordinate axes. This *gnomon* indicates the orientation of the map's coordinate system.

### 3.2.3 Saving and Quitting

This section explains the procedures for ending a GRINCH session. Of course you've just begun, but this will enable you to stop at any point in the tutorial and resume at another time.

**Saving.** After you have made changes in the map and/or model, you can write the data to a file with the `save` command. We recommend that users number consecutively the different versions of their files and that they use the `.g` suffix for these GRINCH-loadable files. If the name of the initial file was `protein.g`, type

```
save protein1.g
```

If the file named already exists, it will be written over; if not, a new file will be created. This file is in a form suitable for loading by GRINCH. It stores complete information about the map and model as well as the current width, center, and level being displayed. Hence, reloading the file will restore the same view (aside from rotations) that was being displayed when it was saved.

**Quitting.** To end a GRINCH session, type the command:

```
quit
```

If you have not made any changes in the map or model since the last `save` command, the program will terminate. If you have made changes and have not saved them, GRINCH will ask for confirmation. The confirmation is entered at the keyboard. Typing `yes` will terminate the program, and any changes made to the file since the last `save` will be lost. Typing `no` will cancel the `quit` command and return the prompt. At this point, you can save the file or continue to work. (Note: This command can also be picked from the escape menu. This is discussed below.)

The same confirmation is required if you try to load a new file when the current file has been changed and not saved. In this case, typing `yes` results in the new file being loaded, and the current file being discarded (along with your changes!).

### 3.2.4 Dynamic Controls

If you have `quit`, run `interp` again and reload your ridge-line map. With the map on the display, try rotating it using the joystick or dials. You can rotate it about three different axes. Now move the dynamic level control — a slider or dial. It adjusts the effective minimum density level threshold. As the number increases, fewer edges will be displayed; edges that remain have density values above the threshold.

Also, try moving the cursor using the puck, mouse, or stylus. It will appear as a small square when in the display area and as a diamond when near the menu. Notice that it moves smoothly in the area of the map display, but that it moves discretely (making small jumps) when in the area of the menu. This discrete jumping avoids ambiguous menu picks.

### 3.2.5 Commands for Viewing

**Size Commands.** The first series of commands, **big**, **small**, and **size**, change the width of the view cube. Move the cursor to the command **big** on the menu and pick (push on the stylus). The cursor should change shape to a triangle, and then back, indicating the pick was registered. The word *bigger* should appear on the terminal screen. The effect of this command is to *zoom in* on the picture. It actually decreases the width of the view cube by 20%. Notice that the new width appears at the top of the graphics display. Picking the **small** command *zooms out*, increasing the width by 25%. Therefore, **big** followed by **small**, or vice versa, results in no net change. These changes can take as long as thirty seconds.

Now pick the **size** command. The size menu should appear (Figure 3.2). Picking a number from this menu will change the width of the view cube to that number of Ångstroms. This is useful for making more extreme changes in the scale than those provided by **big** and **small**.

5
10
15
20
25
30
35
40
45
50
55
60
70
80
90
full
esc

Figure 3.2: Size Menu

0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
31
esc

Figure 3.3: Level Menu

The **full** command, which appears on the **size** menu, changes the center and width of the view cube so that the entire map is displayed as large as possible.

Each of these viewing commands, as well as all commands discussed below, can be entered from the keyboard as well as picked from a menu. For example, you can type

**width 12.0**

to change to a view cube with a width of 12Å. Note that the number must have a decimal point. See the reference manual for details on keyboard entry of all commands.

**Center Command.** You can change the location of the view cube with the **center** command. First, pick **cen** on the menu. Next, move the cursor to a map edge and make a pick (you can only pick edges, not endpoints). The cursor will change to a triangle when the pick has been registered, then return to a square. The center of the *view cube* will

now be moved to the midpoint of the edge picked. Alternatively, x, y, and z coordinates can be entered from the keyboard. (e.g. type `cen 21.4 16.3 9.1`)

This command is useful for moving laterally through a map while, for example, identifying the mainchain. It is also used for moving to known locations in the map space.

**Level Command.** When a ridge-line graph is calculated, each edge is assigned a level number, on an arbitrary scale between 0 and 31, corresponding to its associated electron density. The `level` command changes the absolute minimum density level threshold. Pick `lev` from the menu. A new menu showing numbers from 0 to 31 is displayed (Figure 3.3). Pick one of the numbers from this menu. Edges with a lower density level will not be displayed. The dynamic level control can increase this threshold, but cannot decrease it.

This command is useful for getting better response when looking at large sections of the map since increasing the absolute level threshold reduces the number of edges involved in calculations. The dynamic level control (slider or dial) affects only the display, not the calculations.

### 3.2.6 Commands for Interpreting

Begin this section by changing the width of the view cube to 15Å. Experience has shown this to be a reasonable scale for interpreting map edges. (Make sure there are enough edges visible for interpreting. If not, change the center to a new location.)

In GRINCH, colors are used to identify interpreted map edges:

green	for	mainchain
purple	for	sidechain
red	for	oxygen
brown	for	disulfide bridges, hydrogen bonds, etc.
white	for	unknown (uninterpreted)
yellow	for	flags or markers
light blue	for	flags or markers

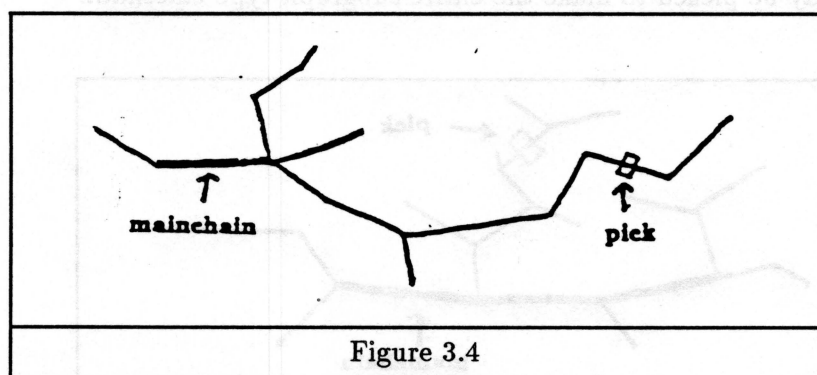
**Single Edges.** Pick `m` from the menu. The cursor should turn green. Now pick a white edge (try to pick one from a chain of connected white edges). The single edge picked turns green and the cursor remains green. Until a different command is picked you may continue to color white edges green without picking `m` again.

Each of the following commands works in the same way, allowing multiple picks and coloring a single edge at a time:

<b>m</b>	for	mainchain
<b>s</b>	for	sidechain
<b>o</b>	for	oxygen
<b>b</b>	for	disulfide bridges, hydrogen bonds, etc.
<b>u</b>	for	unknown (uninterpreted)
<b>f1</b>	for	flag 1
<b>f2</b>	for	flag 2

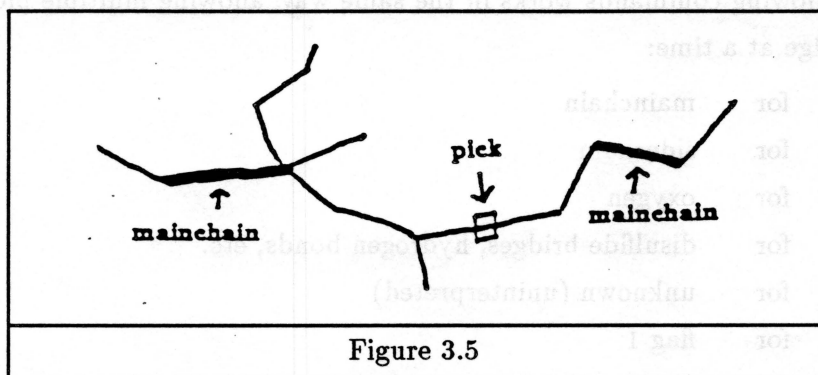
**Multiple Edges.** To speed interpretation, GRINCH provides an alternate method for identifying longer chains of mainchain, sidechain, and unknown map edges.

Pick **main** from the menu. The cursor again turns green. Now pick an unknown edge that is connected to a chain of white edges which connect to mainchain (Figure 3.4). GRINCH will search adjacent chains of white edges and will find the shortest path to existing mainchain. The edge picked, and all edges in that shortest path, are made mainchain edges and colored green.



Alternatively, if an edge is picked between two sections of mainchain (Figure 3.5), the shortest path of unknown edges to each will be found and colored. This, in effect, makes one longer chain of the two original chains.

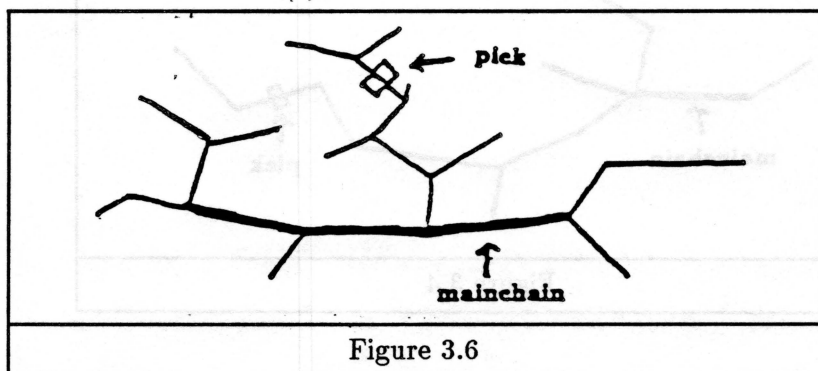




There are certain patterns GRINCH knows to disallow, such as loops or forks in the mainchain, and an error will be displayed should these be found. If no path is found, the single edge selected is made *mainchain*.

The **main** command allows edges to be picked until a new command is specified. The same is true for the **side** and **unkn** commands described below.

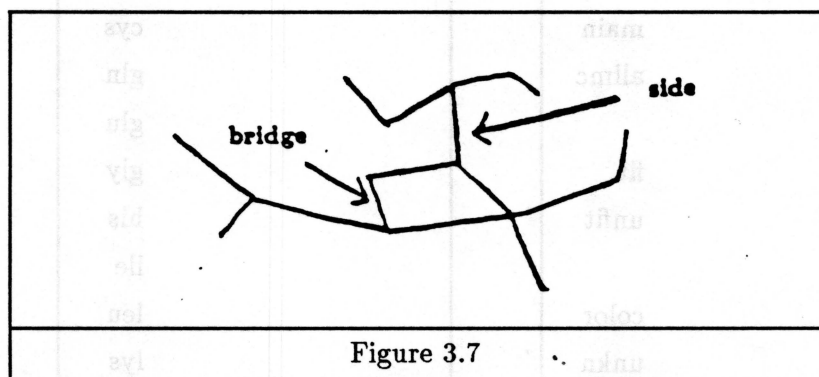
The **side** command has a similar function and is generally used for coloring small subgraphs of white edges attached to mainchain at a single point (Figure 3.6). An edge of this subgraph may be picked to make the entire subgraph type *sidechain*.



If the subgraph selected is attached to mainchain at two points, an error is returned. One of the purposes of the **b** (bridge) command is to get around this limitation. Identify as type *bridge* all but one of the edges connected to mainchain. This will enable the **side** command to color the rest of the subgraph.

This same technique is used to identify proline as the sidechain. For proline, the alpha-carbon beta-carbon edge should be made type *bridge* (Figure 3.7). This is also

necessary for building the model with the `residue` command, which will return an error if the sidechain selected is connected to mainchain at more than one point. This is described in detail in Chapter 4.



The `unkn` command has fewer restrictions than do `side` and `main`. Picking `unkn` and a map edge of a given type, say *sidechain*, changes all map edges of that type linked to the picked edge by similar edges, to type *unknown*. Hence, you could unintentionally interpret an entire connected mainchain with one pick. This is a good reason for having an undo command.

**Undo, Redo, and History.** Most GRINCH commands are *undoable* and the system enables the user to back up through a sequence of commands. Type `h` at the keyboard. This `history` command returns a list of up to your last twenty *undoable* commands.

Now pick `undo` from the menu. It should undo the last *undoable* command you gave. Pick `undo` again. It should undo the previous *undoable* command.

Now pick `redo`. It should redo the last command undone. Pick `redo` again. It will redo your previous `undo`. Pick `redo` one more time. You should get an error. **Redo** only redoes consecutive `undo` commands.

**Selective Display of Edges.** At this point, you should have some interpreted (colored) edges on the display. If not, color some edges before executing the next command. Pick `show` from the menu. The show menu will be displayed (Figure 3.8). Now pick any combination of the edge classifications listed in the menu, completing the command by picking `end`. After the command is executed, only edges fitting into one or more of the categories chosen will be displayed. This remains in effect until the show command is executed again.

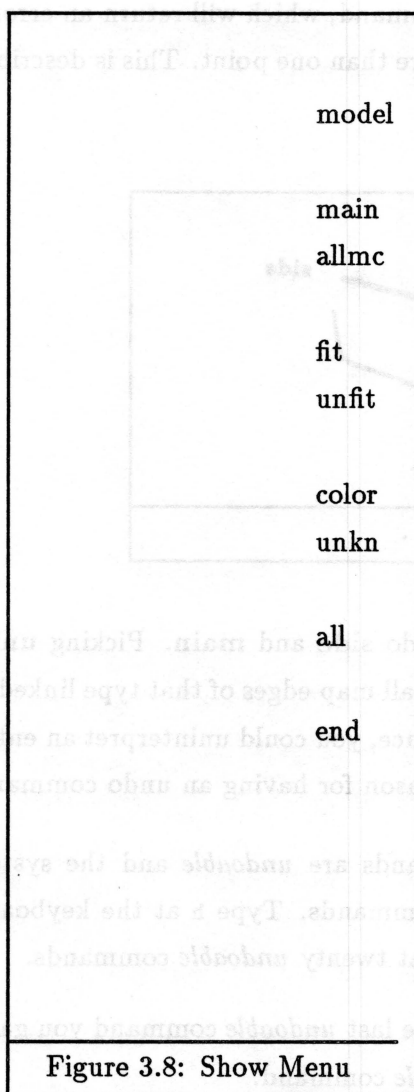


Figure 3.8: Show Menu

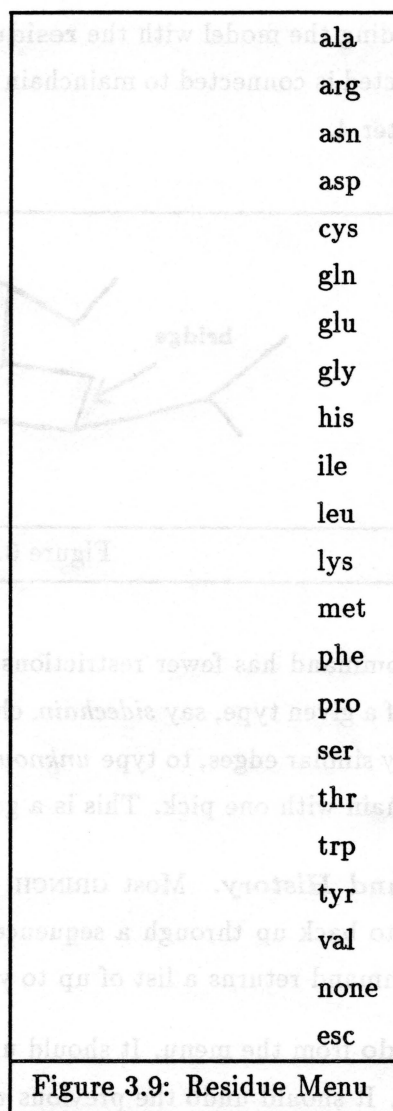


Figure 3.9: Residue Menu

Try displaying just *mainchain* edges with the sequence of picks

**show main end**

(Note that at this time there are no *model* or *fit* edges. These will appear in the next section.) Detailed explanations of the choices are given in the reference manual.

### 3.2.7 Building the Model

To do this section of the tutorial, you'll need some interpreted edges. If you haven't already done so, color some possible stretches of mainchain, as well as some sidechains and oxygens (Figure 3.10). These identifications need only be conceivable, not convincing. (In fact, you'll be able to build a model from mainchain alone if sidechain and/or oxygen edges are missing from the map.)

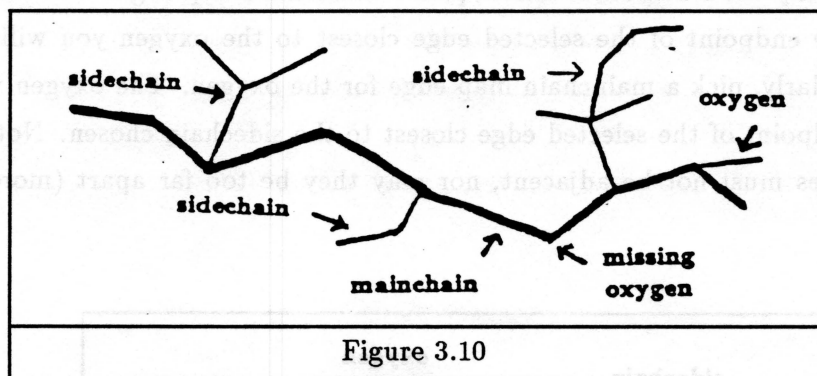


Figure 3.10

**Residue Command.** You should have in mind a place where you're going to build the first residue of your model. Pick `res` from the menu. The residue menu will appear (Figure 3.9). Next, pick the residue you're going to build.

You must now make two picks of map edges. These identify the interpreted edges of the map to which the model of the residue will be *fitted*. The first pick identifies the sidechain and can be any edge of the sidechain subgraph you've interpreted. (The specific edge of the subgraph selected will not affect the placement of the residue model.)

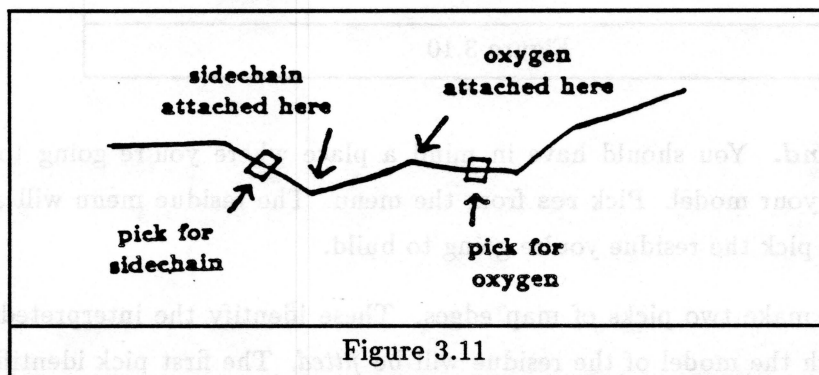
The second pick indicates the oxygen placement. Typically this is a single red edge. There is a restriction that the sidechain and oxygen selected must not attach to the mainchain at the same node, nor may they be too far apart (more than  $3.8\text{\AA}$ ). In either case, an error will be returned.

A few seconds after the last pick, the main menu should reappear. Following this, a blue model of the residue selected will appear in the identified location. If you are dissatisfied with the selection or placement, there are two ways to undo it. You can use the `undo` command, or you can replace the residue you built with residue `none`. To do the latter, you must follow the same sequence of steps as above (e.g. pick `res`, pick `none`, and pick two map edges. They need not be the same two edges picked when building the residue, but they must be associated with the residue.)

**Missing Sidechain or Oxygen.** It is not uncommon for there to be no sidechain and/or oxygen edges along part of a mainchain of a ridge-line map. GRINCH will build a residue with either or both missing if appropriate picks of mainchain map edges are used to indicate where the sidechain and oxygen should connect to the mainchain.

Try this on any sequence of mainchain map edges where no residue has been fitted. (There must be at least three consecutive green edges.) Pick `res` followed by a residue

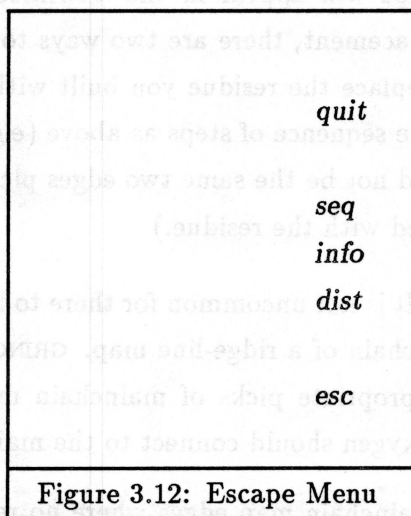
type. Then, to identify the sidechain location, pick a mainchain map edge. The sidechain will be built at the endpoint of the selected edge closest to the oxygen you will select (Figure 3.11). Similarly, pick a mainchain map edge for the oxygen. The oxygen will be attached at the endpoint of the selected edge closest to the sidechain chosen. Note that the two picked edges must not be adjacent, nor may they be too far apart (more than 3.8Å).



Try building some adjacent residue models. Notice that GRINCH connects one to the other. You might also try the `show` command with the *fit*, *unfit*, and *model* choices, now that some residues have been fitted.

### 3.2.8 Miscellaneous Commands

There is one more menu of commands which is displayed by picking `esc` (escape) from the main menu (Figure 3.12). These commands are discussed in the reference manual.



## Chapter 4 Reference Manual

### 4.1 Notation

The following notation is used in this chapter:

- Commands are printed in **boldface**.
- Commands may be either picked from the menu or entered from the keyboard unless otherwise noted.
- Alternate spellings for a command are listed in parentheses. Any of these may be keyed in, though only one will be found on the menus.
- Arguments are printed in *italics* and may be one of the following types:
  - integer - a number without a decimal point picked from a menu or entered from the keyboard
  - float - a number with a decimal point picked from a menu or entered from the keyboard
  - edge or map\_edge - an edge picked from the display or keyed as 'e' followed by the edge number, e.g. e1234
  - residuetype - one of twenty amino acids, picked from the menu or typed using the corresponding number (1 to 20; 0 for none).
  - filename - any acceptable system filename entered from the keyboard
  - resnum - any residue from the model; selected from the display by picking any associated edge, or keyed as 'r' followed by the residue number, e.g. r90
- Arguments in brackets ( [ ] ) indicate a choice of those listed.

- Arguments followed by a plus (+) indicate that one or more may be selected.

The commands are divided into four categories: viewing, interpreting, program control, and miscellaneous. Within each category, the listing is alphabetical. The index provides an alphabetical listing of all the commands.

## 4.2 Commands

### 4.2.1 Viewing

The volume of the map space that is visible is always a cube and is referred to as the *view cube*. The width of the view cube is the length of an edge of the cube, in Ångstroms. That is, a width of 10.0 means that  $10.0 \times 10.0 \times 10.0$  cubic Ångstroms of map space are visible. The width and center point of the view cube appear on the graphics display after any change.

#### **bigger**

( **big** )

The edges displayed are made bigger (zoom in) by decreasing the width of the view cube by 20%. The new width, in Ångstroms, appears on the graphics display.

#### **center edge**

( **cen** )

The center of the view cube is set to the midpoint of the selected edge. Use this to move to different parts of the map space. The coordinates of the new center point are displayed on the graphics display.

#### **center float float float**

( **cen** )

The three numbers represent the x, y, and z coordinates, in Ångstroms, of a point in the map space. The center of the view cube is set to this point. The coordinates of the new center point are displayed on the graphics display.

#### **level integer**

( **lev** )

The minimum absolute density level threshold is set to the specified number.

Each edge in a ridge-line map has an associated electron density value between 0 and 31, inclusive. This value is determined by *mkskel*, the program that calculates the

ridge-line graph from the electron density data. Mkskel scales the original range of densities to this smaller range.

Only map edges with a density value higher than the absolute threshold are included in the calculation of the display. The dynamic level control (slider or dial) affects only those edges included in this calculation. Hence, this control can make the effective threshold greater than, but not less than, the absolute threshold. The current absolute and dynamic thresholds are displayed at the bottom right of the graphics screen.

(Note: This command exists only for performance reasons. For example, changes in the view cube will occur more rapidly with a higher absolute threshold because a smaller number of map edges will be included in the calculation of the display.)

**show** [ *all model main allmc fit unfit unknown* ]+ **end**

Only edges fitting into one or more of the specified categories (within the current view cube) are displayed. This remains in effect until the next **show** command. The arguments are defined as follows:

<i>all</i>	all map and model edges
<i>model</i>	all model (blue) edges
<i>main</i>	mainchain map edges
<i>allmc</i>	mainchain map and model edges
<i>fit</i>	map edges to which a residue model has been fit; mainchain, sidechain, and oxygen edges associated with residues
<i>unfit</i>	map edges to which no residue model has been fit
<i>unknown</i>	map edges which have not been interpreted (white edges)
<i>color</i>	map edges which have been interpreted (green, purple, red, brown, light blue and yellow map edges)

Any combination of arguments can be selected, and the results are additive. The list of arguments must be followed by the command, **end** or **endshow**.



**size float****( width )**

Displays the size menu, from which a number can be selected. The width of the view cube is set to the chosen number of Ångstroms. This command is useful for changes in scale greater than the changes produced by **big** and **small**. The new width appears on the graphics display.

**size full****( width )**

The width and center of the view cube are set so that the entire map and model are in the view cube and are displayed as large as possible. The new center and width appear on the graphics display.

This command is only available as a menu pick. See **window** for the corresponding keyboard command.

**smaller****( smal, small )**

The edges displayed are made smaller (zoom out) by increasing the width of the view cube by 25%. The new width, in Ångstroms, appears on the graphics display.

**window**

The width and center of the view cube are set so that the entire map and model are in the view cube and are displayed as large as possible. The new center and width appear on the graphics display.

This command is only available as a keyboard command. See **size full** for the corresponding menu pick.

### 4.2.2 Interpreting

**bridge** *map\_edge* +

( **B**, **b**, **br** )

The selected edges are identified as type bridge and colored brown. Multiple picks are possible.

This command is intended for identifying such structures as disulfide bridges and hydrogen bonds. In addition, GRINCH will not allow certain configurations of edges, such as a sidechain touching a mainchain in more than one point (in a proline ring, for example). In this situation, all but one of the sidechain edges connected to mainchain should be made type bridge.

**flag1** *map\_edge* +

( **F1**, **f1** )

The selected edges are colored light blue. Multiple picks are possible. This command is intended for identifying edges of the map as markers. For example, edges added by the user might be marked for later scrutiny, or edges along the boundary of a molecule could be identified. This is useful for orientation in global views of the map.

**flag2** *map\_edge* +

( **F2**, **f2** )

The selected edges are colored yellow (See **flag1**).

**m** *map\_edge* +

The selected edges are made type mainchain and are colored green. Multiple picks are possible.

**main** *map\_edge* +

( **mc**, **mainchain** )

Searches for the shortest path of visible, unknown (white) edges from the edge selected to a visible mainchain edge. If a path is found, all the edges in the path are identified as type mainchain and colored green. If not, only the selected edge becomes mainchain. Multiple picks are possible.

Interpreting mainchain with this command proceeds much more rapidly than by selecting a single edge at a time. It can be used in two ways. A single existing chain of mainchain edges can be extended, or an edge between two separate chains of mainchain edges can be selected and the path between the two will be found and interpreted. Note that the command will not create branches or loops in the chain.

o *map\_edge +*

The selected edges are made type oxygen and are colored red. Multiple picks are possible.

**residue** *residue\_type map\_edge1 map\_edge2*

( *res* )

A model of the selected residue is built and *fitted* to the part of the map indicated by the edges selected. In the most common case there is a subgraph of sidechain map edges connected to the mainchain at one node, and an oxygen edge connected at a different node. *Map\_edge1* indicates the sidechain and can be any edge in the subgraph of sidechain edges. *Map\_edge2* indicates the oxygen edge.

If there are no sidechain edges, a mainchain map edge may be chosen for *map\_edge1*. In this case, the sidechain is inferred to be attached to the endpoint of *map\_edge1* closest to *map\_edge2*.

If no oxygen edge exists, a mainchain edge may be chosen for *map\_edge2*. The oxygen is inferred to be attached to the endpoint of *map\_edge2* closest to *map\_edge1*.

If neither sidechain nor oxygen edges exist, the two mainchain edges selected must have at least one edge between them, and may be no farther than 3.8Å apart.

**sequence** *integer edge*

( *seq* )

Assigns sequence number *integer* to the residue that contains *edge*. It attempts to work along the chain assigning consecutive residue numbers in both directions. See **model save**.

**sequence** *modeledge1 modeledge2*

( *seq* )

Matches a sequence of residues which have been *fitted* to the map, to the known residue sequence. The two residue arguments, *modeledge1* and *modeledge2*, must be

connected by a sequence of residues. Either *modeledge1* or *modeledge2* may be the amino end of the specified sequence.

The computer initially constructs a trial matrix,  $T$ , with one column for every residue in the chain specified by *modeledge1* and *modeledge2*, and one row for each residue type (Figure 4.1). An element of the  $i^{\text{th}}$  row and  $j^{\text{th}}$  column of  $T$  gives an estimate of the probability that the  $j^{\text{th}}$  residue fitted really is of the type in row  $i$ . The probability ratings are based upon structural comparisons between the fitted residue and each residue type. Values range from a *worst* rating of 1 to a *best* rating of 10.

Matrix  $T$  is presented through the *vi* text editor [Joy 1980.] so that the user may alter the matrix for the specific environment of the residues that were fitted. Matrix entries can be changed, using commands in *vi*, to increase or decrease the probability values for specific residues, or the initial values may be used. When the matrix contains the desired values, exit the editor with the command `:wq` or the command **ZZ**.

The image shows a faint, mirrored representation of a matrix table. The table is oriented upside down and contains numerical values for various residue types. The residue types listed on the left side of the table (when viewed normally) are W, Y, V, H, R, K, and M. The numerical values represent probability ratings from 1 to 10.

W	1	1	1	1	1	1
Y	1	8	1	2	2	1
V	1	10	1	2	2	1
H	1	8	1	2	2	1
R	1	2	1	7	7	1
K	2	2	2	10	10	1
M	1	2	2	1	1	1

The testing matrix is used to match the specified sequence to the known sequence. The known sequence must be in a file named *sequence* and program execution begins. Three-letter or one-letter abbreviations may be used, with letters separated by blanks or newlines; all letters must be upper case.

G	4	1	2	4	4	
A	10	1	3	5	5	
S	8	1	8	3	3	
T	2	1	10	1	1	
V	2	1	8	1	1	
P	3	1	5	1	1	
I	1	1	5	1	1	
L	1	4	2	2	2	
N	1	3	3	2	2	
D	2	3	3	2	2	
Q	1	3	3	2	2	
E	2	3	3	2	2	
C	3	1	7	1	1	
M	1	5	2	4	4	
K	2	2	2	10	10	
R	1	2	1	7	7	
H	1	8	1	2	2	
V	1	10	1	2	2	
Y	1	8	1	2	2	
W	1	6	1	1	1	

Figure 4.1: Sequence Command Trial Matrix

The resulting matrix is used to match the specified sequence to the known sequence. The known sequence must be in a file named *sequence* when program execution begins. Three-letter or one-letter abbreviations may be used, with entries separated by blanks or newlines; all letters must be upper case.

43	S	H	T	R	K		7
36	A	F	T	I	A		1
33	T	H	S	K	A		9R
30	A	A	M	K	R		14R
28	A	I	T	F	A		5R
25	A	M	K	R	T		13R
23	A	K	S	H	T		5
22	F	T	I	A	K		2
19	T	I	A	K	S		3
19	K	R	T	A	S		11R

Figure 4.2: Top Ten Rated Registrations

GRINCH tries each possible registration of the specified sequence with the known sequence, in both directions. A sum of the appropriate matrix elements is computed and the user is presented with the top ten rated registrations (Figure 4.2). The number on the left of each row is the *score* of the registration. The number on the right is the position, within the known sequence, of the registration. An *R* indicates that the sequence is reversed.

In the present implementation, no further action is taken by the computer; the user must make any changes required to match his/her fitted sequence with the registration chosen.

#### s map\_edge +

The selected edges are made type sidechain and are colored purple. Multiple picks are possible.

**side** *map\_edge* +  
( *sc*, *sidechain* )

Searches for the largest connected subgraph of visible, unknown(white) map edges connected to the selected edge. If the subgraph touches mainchain in exactly one point, all the edges are made type sidechain and colored purple. If the subgraph touches mainchain in more than one point, no action is taken and an error message is displayed on the terminal screen. If the subgraph does not touch mainchain at all, it is cut into two parts at the selected edge, and only the selected edge and the smaller part are made type sidechain. No action is taken if a path of length greater than 14 Å exists in the set. Multiple picks are possible.

If the subgraph is supposed to connect to the mainchain in two points (e.g. proline), the alpha-carbon beta-carbon edge must be identified as some other type (usually *bridge*) before the **side** command can be used. Furthermore, the **residue** command will return an error if the sidechain selected is attached to mainchain at two points (see above).

**split** *map\_edge*

The edge selected is split into two equal edges and a new node is created at the center. If the picked edge has one or more residues, the new edges get the residues of the closer original node. If the new node is between two nodes with the same residue, the new node gets that residue; otherwise, the new node's residue is undefined.

*Note:* This command is not fully implemented.

This command can only be entered from the keyboard.

**u** *map\_edge* +

The selected edges are made type unknown and are colored white. Multiple picks are possible.

**unknown** *map\_edge* +  
( *unkn* )

All edges in the subgraph containing the selected edge (edges of the same type, connected to it by edges of the same type) are made type unknown and colored white. (Note the lack of restrictions! This command can color an entire connected mainchain white with one pick.) Multiple picks are possible.

### 4.2.3 Program Control

**[add] ascii load filename [ yes no ]**

The named ridge-line graph file (in ascii format, see **ascii save**) is loaded into the program and displayed. It is loaded in the exact form in which it was saved (or created, if never loaded previously).

If **add** is specified, the newly loaded ridge-line graph will be displayed in addition to the one previously loaded (if there is one).

If **add** is not specified, any previously loaded graphs will be deleted. In that case, if the GRINCH map interpretation has been changed since the last **save** command, confirmation of *yes* or *no* must be given. The confirmation is typed at the keyboard. A *yes* indicates that the file will be read and the resident interpretation discarded. A *no* aborts the **load** and returns the prompt for a new command. At this point, the resident interpretation could be saved and the **load** tried again.

This command can only be entered from the keyboard.

**ascii save filename**

The current map, model, and display data are stored in the named file in ascii format. If the file already exists, it is overwritten; if not, a new file is created. See **ascii load**.

This command can only be entered from the keyboard.

**bgload filename**

( bgld )

Bgload adds a background object to the current display list. The object is read from the specified file. Each line of the file specifies the coordinates of the endpoints of one edge, its type, and its density, as shown below. (Fields are separated by white space.)

```
x1 y1 z1  x2 y2 z2  type density
```

The coordinates are signed real numbers and are interpreted as being in the same coordinate system as the current molecule. The type is a character representing the type of the edge and its associated color, with the values shown below.

M	mainchain	green
S	sidechain	purple
O	oxygen	red



B	bridge	brown
U	unknown	white
R	model	blue
F	flag	light blue
G	flag	yellow

The default type, when none is specified, is F. The density is an integer between 0 and 31, inclusive. The default density value, when none is specified, is 31 (the greatest density value). If you save the molecule after loading the background object it will be saved with the molecule.

This command can only be entered from the keyboard.

#### **displayon name**

Turns on the display named. Recognized names are:

ik, ikonas	Adage-Ikonas RDS3000
ps, ps300	Evans Sutherland PS300
vg	Vector General 3033
crt	Terminal Screen
mps	Multi-Picture System
masscomp, mc500	Masscomp 5500

This command can only be entered from the keyboard.

#### **displayoff name**

Turns off the display named. See **displayon**.

This command can only be entered from the keyboard.

#### **esc**

When selected from the main menu, this command displays the escape menu which includes the commands **quit**, **seq**, **info**, and **dist**. When selected from any other menu, the main menu is displayed and the prompt returns.

**h**

The history command displays the last twenty 'undoable' and 'redoable' commands (see **undo**).

This command can only be entered from the keyboard.

**inputon name**

Turns on the input device named. Recognized names are:

tablet	puck and data tablet
viewpointer, vuptr, joystick	3-d viewpointer
slider	slider for dynamic level control
keyboard	terminal keyboard
mouse	Masscomp mouse
psknobs	PS300 control dials
pstablet	PS300 stylus and data tablet

This command can only be entered from the keyboard. Note that **displayon ps** automatically turns on psknobs and pstablet. **Displayon mps** acts analogously. **Displayon mc500** turns on the mouse and keyboard.

**inputoff name**

Turns off the input device named. See **inputon**.

This command can only be entered from the keyboard.

**load filename [ yes no ]**

The named ridge-line graph file (in GRINCH format, see Chapter V) is loaded into the program and displayed. It is loaded in the exact form in which it was saved (or created, if never loaded previously).

If the GRINCH map interpretation has been changed since the last **save** command, confirmation of *yes* or *no* must be given. The confirmation is typed at the keyboard. A *yes* indicates that the file will be read and the resident interpretation discarded. A *no* aborts the **load** and returns the prompt for a new command. At this point, the resident interpretation could be saved and the **load** tried again.

This command can only be entered from the keyboard.

**model delete**

All model edges are deleted from the display and the program. The system does not ask for confirmation, so do not use this command unless you are sure you want to delete all model edges.

This command can only be entered from the keyboard.

**model load [ filename ]**

The named model file is loaded into the program and displayed, in addition to the existing model. See **model save**.

This command can only be entered from the keyboard.

**model save [ filename ]**

The current model only is written into file. If the file already exists, it is overwritten; if not, a new file is created. See **model load**.

This command can only be entered from the keyboard.

**redo**

Reverses the effects of an immediately preceding **undo**, with results as if the **undo** had never been given. If another **redo** command is given, it reverses the **undo** immediately preceding the previous **undo**. An error is returned if there is no immediately preceding **undo**.

**save filename**

The current map, model, and display data are stored in GRINCH format in the named file. If the file already exists, it is overwritten; if not, a new file is created.

This command can only be entered from the keyboard.

**set menu** [ *off on* ]

The *off* option is used to delete the menu from the display (useful for photographs).

The *on* option is used to restore the menu to the display.

This command can only be entered from the keyboard.

**set stereo** [ *on off* ]

Turns stereo on in time division mode, or turns it off.

This command can only be entered from the keyboard.

**undo**

Reverses the effect of the last command, with results as if that command had never been given. If another **undo** is given, it reverses the effect of the command given prior to the one already undone. In this manner, the user can backup up as many as twenty commands. An error message is displayed on the terminal screen if there are no more commands to undo or if the previous command cannot be undone. The commands **load** and **save** cannot be undone.

#### 4.2.4 Escape Menu Commands

These commands are displayed when the command, **esc**, is picked from the main menu. Each command can also be entered from the keyboard.

**add** *map\_edge1 map\_edge2*

An edge of type unknown (white) is added between one endpoint of the first edge picked and one endpoint of the second edge picked. The endpoints are chosen to make the new edge as short as possible.

**delete** *map\_edge*

( **del** )

The map edge selected is deleted from the display and from the graph.

**distance** *edge1 edge2*

( **dist** )

The shortest distance between the endpoints of the first edge and the endpoints of the second edge is displayed on the terminal screen.

**info edge**

All known information about the edge and its endpoints is displayed on the terminal screen. This includes edgetype, edge number, length, coordinates of the endpoints, fit or unfit; if fit, then also displayed are the residue type, residue number, atom types of the endpoints, and whether twistable or not twistable.

**quit [ yes no ]**

To terminate GRINCH, type the command, **quit**. If a file is resident in GRINCH and has been changed since the last **save** command, the program will prompt for confirmation of *yes* or *no*. Typing *yes* indicates that the session will end and that the resident file will be discarded. Typing *no* aborts the **quit** and returns the prompt for a new command.

**4.2.5 Miscellaneous****plot filename**

Saves the displayed image in character 2-D plot format. This is suitable for making hardcopy plots of the image. On UNIX at UNC the command

```
atop < filename | plot -Tghp
```

will plot the image on the graphics lab HP pen plotter.

**print**

Displays, on the terminal screen, detailed information of all nodes and edges in the current display. This includes node number, node density, edge number, edge type, edge density, residue type, node number for each endpoint and length of the edge.

!

This is the same shell escape found in the *vi* editor. Any acceptable shell command will be executed, after which the prompt will return and GRINCH will await the next command.

## Chapter 5

### Creating a Ridge-Line Graph

This chapter describes how to generate a ridge-line graph from an electron density map using the program *mkskel*. *Mkskel* reads a file of electron density data and writes a file suitable for loading into GRINCH. *Mkskel* can also write files of "edges" which are suitable for other (non GRINCH system) programs to display or manipulate.

Throughout this chapter, commands (executable files) appear in **boldface**. Arguments appear in *italics*. Arguments that appear within brackets ( [ ] ) are optional.

#### 5.1 Data and Information Needed

*Mkskel* requires two input files. The map file contains electron density data and the map information file contains information relevant to the processing of the map file.

##### 5.1.1 The Map File

The map file contains the electron density value for each vertex of a three-dimensional grid. This data must be from an orthogonal map; *mkskel* cannot work with triclinic or other non-orthogonal maps.

The density values can be stored in character format or in binary format. If the values are stored as characters they should be separated by one or more blank spaces, tabs or newlines, and the file should contain no extraneous marks (e.g. end-of-plane markers). The first line of such a file might look like:

```
20.1 12.8 -3.2 18.4 3.2 53.4 30.2 85.3 43.2 -12.0
```

The map file must be given a name of the form *root.map* where *root* is generally some abbreviation of the molecule name.

##### 5.1.2 The Map Information File

The map information file contains various mandatory and optional items that specify characteristics of the electron density map, input and output file formats, and other parameters. The map information file must be given the name *root.mi*, where *root* is

the same as for the *.map* file. Figure 5.1 shows the contents of a sample map information file.

<i># sample.mi</i>	
<i>mapinformat</i>	<i>CHAR</i>
<i>rloutformat</i>	<i>EDGES</i>
<i># name of the molecule</i>	
<i>title</i>	<i>Deoxypreposterase ABC</i>
<i>order</i>	<i>XZY</i>
<i>dimension</i>	<i>60 50 72</i>
<i>spacing</i>	<i>1.1 0.9 0.8</i>
<i>origin</i>	<i>-2 0 3</i>

Figure 5.1: Sample Map Information File

Mkskel requires the following mandatory information:

- (1) *order XYZ or XZY or YXZ or YZX or ZXY or ZYX*

The sort order of the density values. Since the coordinates of each value are not included in the file, mkskel must know which dimension of the grid (X, Y, or Z) varies most slowly, which next most slowly, and which most quickly.

For example, listing the values from the first ZY-plane, then the next, and so on, means that X varies most slowly. Within each ZY-plane, listing all the values for a given Z coordinate (along the Y-dimension) means that Z varies more quickly than X, and Y varies most quickly. This is described in the information file as XZY.

- (2) *dimension slowsize middlesize fastsize*

The extent of the map. This is the number of density values along each axis of the map. (One more than the number of spaces!) The order is by sort order, not by X Y Z. The values in the example show 60 points in the X direction, 50 in the Z direction, and 72 in the Y direction.

- (3) *spacing slowspace middlespace fastspace*

The spacing between any two consecutive density values along the corresponding dimension (in Ångstroms), also in sort order. The values in the example show a 1.1Å space between consecutive points along the X dimension, 0.9Å. between points in the Z dimension, and 0.8Å between points in the Y dimension.

(4) *origin sloworigin middleorigin fastorigin*

The coordinates of the origin of the map, i.e. which plane, row and column passes through the origin of the map coordinate system (starting numbering at one). If the first density value is at the origin, the entry in the map information file would be *origin 1 1 1*. The values in the example show the plane with X value -2, the row in that plane with Z value 0, and the column in that row with Y value 3. Values may be negative or even fractional.

The map information file may also contain any of the following optional items:

(5) Comments. Blank lines or any line beginning with “#” are ignored by mkskel.

(6) *title text*

The title is for informational use only and is ignored by mkskel.

(7) *mapinformat type*

The format of the density map. There are several possible choices for *type*: CHAR (or its synonym ASCII or EBCDIC) for character format (default), FLOAT4, FLOAT for four byte floats, FLOAT8, DOUBLE for eight byte floats, INT1, BYTE for one byte integers, INT2 for two byte integers or INT4 for four byte integers.

(8) *rloutformat type*

The format of the output file. There are several possible choices for *type*: CHAR, ASCII, EBCDIC for GRINCH character format, INT, BINARY, BYTE for GRINCH binary format (default) or EDGES, EDGE for gedgeprint format. There will be a separate output file for each unique *rloutformat type* entry in the map information file.

(9) *mapfile name*

The name of the input density map file. The default is *root.map* where *root* is the filename mkskel was invoked with.



(10) *matrix a b c d e f g h i*

A rotation or skewing matrix applied to all output coordinates. *a* through *i* are floating point numbers.

## 5.2 Executing *mkskel*

Execute *mkskel* by entering:

```
mkskel [-minden minden -maxlen maxlen -range lo hi] root
```

The program will read the files *root.map* and *root.mi* and produce an output graph file named *root.g* (*g* for graph). During execution, the terminal will display information about the run, including numbers of edges which could not be created. As many as eight to ten per plane are reasonable.

### 5.2.1 Command Options

The *minden* parameter sets the minimum ridge-line density and the *maxlen* parameter sets the maximum ridge-line edge length. The range parameters clamp density values below or above the specified range to the *lo* or *hi* values, respectively.

The minimum ridge-line density is the smallest density value to be considered for ridge-lines. The appropriate value here varies with the quality of the map and the range of density values. Making the number smaller will increase the number of ridge-lines in the map; making it larger will decrease the number of ridge-lines. The default is currently 10.

Experience has shown that maximizing the number of ridge-lines in the graph is a useful strategy. However, there is a limit to the number of edges that may be attached to a single node of the graph. If the value of the *minden* parameter is too small, the limit of each node is filled by edges connecting the node to points in its immediate vicinity and a degenerate map is created. Such a map is easy to recognize when displayed with GRINCH; it appears as a screen full of points. Should this occur, increase the value of this parameter and rerun *mkskel*.

If the range of useful density data has been mapped to values between 0 and 120, begin with 10 as the value of the parameter (you will probably have to try several different values).

The maximum edge length is the largest allowable length for a single edge connecting two points in the grid of density data. The value specified is a function of the quality and

resolution of the map, and the spacing of the grid, but the function is not yet defined. The default is currently 2.

A typical value for a 3.0Å map with 1.0Å spacing has been 2.8Å. This allows diagonal edges connecting points in grid planes separated by a single grid plane ( 2.0Å). Further study is required to determine the best value for this parameter.

### 5.2.2 Mkskel Output Files

The .g file produced by mkskel is in the correct format for loading into GRINCH. Displaying the ridge-line graph using GRINCH is generally the best way of determining whether any of the input parameters need to be adjusted.

Mkskel can also write ridge-line files of "edges" which are suitable for other (non GRINCH system) programs to display or manipulate. To get an edges file, include the line

*rloutformat EDGES*

in your map information file.

Each line in the character edges file has the following information (fields are constant length):

Edge number (an arbitrary internal edge number)  
 R indicating "ridge-line" (*i.e.*, not a molecular model or background edge)  
 U indicating "unknown edge type" (*i.e.*, not yet sidechain, mainchain, etc)  
 X, Y, Z of node at the first endpoint of this edge, in Ångstroms  
 X, Y, Z of node at the second endpoint of this edge, in Ångstroms  
 density at first endpoint node (in internal GRINCH range 0 to 31)  
 density at second endpoint node  
 length of this edge, in Ångstroms  
 Node number of node at first endpoint (an arbitrary internal node number)  
 Node number of node at second endpoint

For example:

```
1 R U 29.39 16.30 19.69 29.67 15.34 19.76 9 14 1.0 12 11
2 R U 29.39 16.30 19.69 30.25 17.26 18.69 9 3 1.6 12 45
3 R U 29.24 15.98 23.36 28.23 17.36 22.91 4 12 1.8 13 1
4 R U 29.24 19.38 27.20 29.24 20.14 27.19 28 23 0.8 14 15
5 R U 29.24 19.38 27.20 29.98 19.18 26.83 28 31 0.9 14 53
```

Thus, edge #3 is a line from (29.24, 15.98, 23.36) to (28.23, 17.36, 22.91) and the density at node #13 is 4 and at node #1 is 12.

## Index to Commands

add, 30  
bgload (bgld), 30  
bigger (big), 11, 21  
bridge (B, b, br), 13-15, 24, 29  
center (cen), 12, 21  
delete (del), 33,34  
displayon, 7, 31,32  
displayoff, 31  
distance (dist), 31, 34  
esc, 19, 31, 34  
full, 12, 23  
F1 (flag1, f1), 13, 23  
F2 (flag2, f2), 13, 24  
history (h), 16, 32  
info, 31, 35  
inputon, 32  
inputoff, 32  
interp, 7  
level (lev), 9, 13, 21-22  
load, 8-9, 32  
main (M, mainchain, mc), 13-15, 24-25  
model, 33  
O, 13, 25  
plot, 35  
print, 35  
quit, 10, 35  
redo, 16, 33  
residue (res), 16, 18, 25  
save, 10, 30, 33  
sequence (seq), 25-28, 31  
set menu, 34  
set stereo, 34  
show, 16-17, 22  
side (S, sidechain, sc), 13-15, 28-29  
size, 11-12, 23  
smaller (small, smal), 11, 23  
split, 29  
undo, 16, 34  
unknown (unkn), 13, 14, 16, 29  
width, 12, 23  
window, 9, 23  
! (shell escape), 35

## References

- Bishop, G. 1982. *Gary's Ikonas Assembler, Version 2; Differences Between Gia2 and C*, TR82-010 UNC Department of Computer Science, Chapel Hill, NC.
- Britton, E. G. 1977. *A Methodology for the Ergonomic Design of Interactive Computer Graphic Systems, and its Application to Crystallography*, PhD Thesis, University of North Carolina, Chapel Hill.
- Brooks, Jr, F. P. 1977. "The Computer 'Scientist' as Toolsmith: Studies in Interactive Computer Graphics," *Proceedings of IFIP*.
- Jones, T. A. 1979. "A Graphics Model Building and Refinement System for Macromolecules," *Journal of Applied Crystallography*.
- Joy, W. Nov 1980. "An Introduction to Display Editing with VI, revised by M. Horton," *UNIX Programmer's Manual Volume 2c - Supplementary Documents, Seventh Edition, Virtual Vax-11 Version*.
- Tsernoglou, D., G. A. Petsko, J. E. McQueen, and J. Hermans. 1977. "Molecular Graphics: Application to the Structure Determination of a Snake Venom Neurotoxin," *Science*, Vol. 197, 1378-1381.
- Williams, T. 1982. *A Man-Machine Interface for Interpreting Electron Density Functions*, PhD Thesis, University of North Carolina, Chapel Hill.