MAIN 95 Users Guide

by Dušan Turk

December 16, 1995

MAIN development started at the end of 1985 at Chemical Institute Boris Kidrič in Ljubljana, Slovenia. In 1988 its development continued at Max-Planck-Institute for Biochemistry in Martinsried, Germany. During 1993 development was mainly transferred from Martinsried to Jozef Stefan Institute in Ljubljana, Slovenia.

MAIN is written by Dušan Turk.

MAIN is a computer program distributed by Computer Programs and Services doo Ljubljana according to the agreement with Garching Innovation Gmbh (Technologien aus der Max-Planck-Gesselschaft).

MAIN documentation was prepared using LaTeX.

to MAIN fans and other (?) users

Contents

1 Greetings 1

2 Introduction 5

What is where in the "Users' guide" 5

3 About MAIN, its structure and syntax rules 7

MAIN philosophy 8

Programming in steps 8

Data transferability - compatibility with other programs 9

Data overview 9

Error recovery and safety 9

Rules of syntax 10

Sentence 10

Command_word 11

Integer_number 11

Atom_number 11

Real_number 12
String 12
Variables 12
Selections and keys 12
Short description of MAIN utilities 13
MAIN data structures 14
Atoms 15
Residues 16
Connectivity tables 16
Points 16
Topology library 16
Force field parameters 17
Maps 17
Reflections 17
Miscellaneous 18
Operations 19
INPUT OUTPUT with open auxiliary programs 19
Map INPUT OUTPUT programs and routines 20
Lwplot 21
Technical data 22

4 Errors and bugs 23

Syntax_errors 23

Data_errors 24

Bugs 24

5 Molecular and map images 25

Tryptophan images demo 26

Protein images 30

Combi images 33

Color and light hints 33

Background images 34

Making plots 34

6 Model building 37

Menu driven model building 37

From scratch 37

Dials driven geometry changes 39

SECONDARY structures 40

ENERGY calculations and MINIMIZATIONS 40

Use of anchors 42

Command sentence driven model building 42

Creation of an extended chain 43

Changing a residue 44

Generation of SEQUENCE and SEGMENT names 44

Attaching residues 45

7 Electron density modifications 47

Map concepts 48

Molecular envelopes 48

Crystal cell generation 48

Maps: enevelopes, density filled regions 49

Solvent flattening 51

The standard procedure 51

The fast solvent flattening procedure 53

Solvent flattening procedure based on a skeletonized map 56

Electron density averaging 56

Averaging of electron density: 2 molecules 58

Averaging density between 2 different crystal forms 62

Local histogram matching 67

8 Electron density map calculations 69

Preparations steps 69

Get structure factors 70

Prepare your working set structure factors 72

From structure factors to a map 74

Interactive map calculations 75

9 Crystallographic refinement 77

Preparing and organizing data for refinement 78

Positional refinement 79

Applying NCS constraints 80

Refinement using NCS in a single crystal form 80

Refinement using NCS between two crystal forms 81

Partial model refinement 86

B-value refinement 90

Overall B-value refinement 91

Interactive positional refinement 91

Weighting the density energy term 92

Refinement against a 2Fobs-Fcalc map 93

10 Topology and parameter files 95

Reading MAIN topology and parameter files 95

Inspecting and saving topology library records - write 96

Inspecting topology library records - show 98

Inspecting and saving force field parameters - write and show 99

Using available MAIN topology files 100

Frequent topology and parameter assignment problems 103

Wrong residue or atom names - rename 103

Missing atoms 103

Using available X-PLOR topology files 104

Converting XPLOR topology files to MAIN form 104

Modifying and creating new topology library entries 106

Modifying an existing residue: norleucine 106

11 A real case 109

12 Generation of solvent molecules 115

WATER.COM 116

13 Preparing coordinate file for deposition in PDB 123

The closest solvent molecule 123

Attaching solvents to a chain 125

Renaming atoms and residues, creation of unusual bonds 126

14 Adapting an atom coordinate file 129

15 Utility macros 133

Protein bonds 133

Show unit cell 134

Ramachandran plot 134

16 Installing MAIN 137

Under UNIX 137

Tailoring your own documentation scripts 139

Under VMS 140

Platforms and array sizes 142

17 License policy 143

A Site Academic License Agreement for MAIN 144

A site liicense agreement for MAIN 145

18 Frequently Asked Questions 147

Abelardo 147

choose a molecule (molecule is in this combination a network of 148

select a residue range 148

to learn more set the level of MAIN information higher 149

energy calculations and minimizations in the dialog mode 149

The dials window cannot be read in stereo. 150

Transparent maps 150

What algorithm are you using for energy minimization? 150

From Ren Bin in Sweden - averaging 150

Rick, surface point file and end of selection 153

Stefan on averaging 154

Saving structure factors 154

Writing out the final maps 154

Segment name E 155

RMS fit convergence 155

19 Glossary of MAIN terms 157

`1.51.5`

Chapter 1

Greetings

Dear reader,

This manual will explain both how to use MAIN as it comes to you, and also how to tailor it to your needs an dstyle, but it will not teach you crystallography although it will help you to gain some experience. MAIN is a very powerful program and a programing language which provides tools in the form of macros to diagnose many of your crystallographical problems. As you learn MAIN, try to develop the habit that before doing a task, you think first how you would do it (which steps should be taken) and then about the syntax and command macros. This will help you a lot to understand how MAIN is performing and what it is capable of doing.

Accept MAIN as a friend, with its own personality, affections, needs and logic, and it will treat you well and you will be happy with it.

Manual is organized in 3 parts: "User's Guide", "Dialog Mode Accessories" and "Command Reference Manual". You should read the "User's Guide" first. It is organized in chapters, that cover a particular topic. Each chapter is self-sufficient and dedicated to master the particular task described therein. The power of MAIN lies in its flexibility and generality. with the MAIN

interface as a programing language, the more chapters you will master, the more flexible it will become under your fingers as you enter successive fields of macromolecular crystallography. So, please, run as many examples and demos as you can.

The part "Dialog Mode Accessories" should be read next. This tiny piece of text explains how to use the mouse, keyboard and dials and what you can do with them, when running MAIN in its dialog mode. Remember that using keyboard is the most intuitive and flexible way of communicating with a computer. You can browse through the descriptions of the menu items (strikes) by clicking them, when the documentation mode is turned on. Pressing 'h' in the image window toggles ON or OFF the DOCUMENTATION mode.

The third part "Command Reference Manual" gives you complete power over MAIN on the command language level. By knowing the MAIN language you are not only able to change arguments and variable values but also to write your own tools. '?' anyweher on the command line in a place of an unknown command word gives you the list of possible continuing commands. The word 'HELP' typead in a place of a command word can bring you information from the "Command Reference Manual" files to the screen while running MAIN.

Quite a substational part of MAIN runs through command file macros, so it is very important for you to be able to modify them by using a text editor. You will then be able to use the menus to your own best advantage.

The demos presented at this year ACA and ECM meetings in Montreal and Lund are included in the show.tar.Z. You may wich to untar and run them by clicking the items GO and NEXT on the menu page 1.

The whole MAIN documetation is available in form of directly printable '.txt' and LaTeX '.tex' files.

Do not hesitate to contact me in a case of any trouble. There is an anonymous FTP account available at stef.ijs.si (193.2.5.152) where you can any time retrieve the latest MAIN documen-

tation, utilities, data files and a DEMO version. Licensed executable files will be put there on a request. MAIN URL is at "html://www-bmb.ijs.si/MAIN/".

dusan turk

My address:

```
Dusan Turk
Department of Biochemistry and Molecular Biology
 JOZEF STEFAN INSTITUTE
Jamova 39, 61000 Ljubljana, Slovenia


phone +386 61 177 3626
fax:          273 594
E-mail:  dusan.turk@ijs.si
```

I wish to thank all MAIN fans and other users who helped me to decide what to do. Without doubt the most enthusiastic MAIN fan, Hans Brandstetter, deserves to be explicitly mentioned.

Special thanks to Chris Cardin,Tim Mathers and Rick Engh who critically read the whole manuscript and sent me a lot of corrections and proposals that are still not completely included in the manual.

The split of the before single and complete manual into several parts and organisation of the Users Guide according to topics was encouraged (enforced) by many MAIN users.

Lynn Ten Eyck is gratefully acknowledged for providing his FFT routines.

Chapter 2

Introduction

MAIN is a programming language governing transformation of data forms common to macromolecular crystallography. Data forms are 3-dimensional molecular models and their constituents, reflections, structure factors and electron density maps. Below you can learn how to use MAIN to perform various tasks in the field of macromolecular crystallography.

Throughout the text in manuals the MAIN command words are written in capital letters, titles or hyperlinks are "bold" or highlighted and MAIN input and output are in a typwriter character style.

What is where in the "Users' guide"

Chapter "MAIN structure and syntax rules" gives a general view of MAIN command language and philosophy. It lists data structures and operations between them and it describes the structure of MAIN command line(s).

Chapter "Errors and bugs" explains how to deal with MAIN error messages and find the origin of your errors. It presents bug reports and ways of getting around them including with their dates of discovery and removal.

Chapter "Molecular and map images" gives introduction to molecular and map models 3-dimensional presentations.

Chapter "Model building" explains how to build molecular models from scratch and modify their topology and geometry. Geometry of models can be changed manually (by using dials

bound to atom movements, chain rotations etc... and by calling secondary structure library), by exploiting energy calculations and combining the two approaches. Use of anchors and pair distance restraints is recommended.

Chapter "Electron density modifications" shows how and when to use solvent flattening, electron density averaging, electron density combination and partial structure refinement to improve electron density maps you want to improve.

Chapter "Crystallographic refinement" shows how to refine structures with crystallographic constraints.

Chapter "Topology and parameter files" explains how topology library and force field parameter files can be used, inspected, modified and created. There are also hints what to do when your model does not match with the topology and parameter library definitions.

Chapter "A real case" describes a how to prepare files for model building, map calculations and crystallographic refinement of a molecular model. All neccessary command files you may want to taylor to your needs are described and presented.

Chapter "Generation of solvent molecules" presents the procedure that generates new solvent molecules according to peaks in a Fobs-Fcalc electron density maps, checks their consistency within their molecular model environment (hydrogen bonds and distances) and within a 2Fobs-Fcalc map and removes the inconsistent ones.

Chapter "Preparing coordinate file for deposition in PDB" provides descriptions and macros that can save you quite some time for generation of coordinate files for the Protein Data Bank (PDB) entries.

Chapter "Adapting an atom coordinate file" describes how to adapt a coordinate file to MAIN environment (unmatched residue and atoms names, ...)

Chapter "Utility macros" explains what do the available 'main/utils/' macros do and how to use them.

Chapter "Installing MAIN" describes how to install MAIN on your computer.

Chapter "License policy" presents the MAIN license policy and license agreements.

Chapter "Frequently Asked Questions" presents a selection of user's questions together with their answers.

Chapter "Glossary of MAIN terms"

Chapter 3

About MAIN, its structure and syntax rules

MAIN is a programming language governing transformation of data forms common to macromolecular crystallography. Data forms are 3 dimensional molecular models and their constituents, reflections, structure factors and electron density maps.

The lowest level of MAIN is its command language. Commands can be grouped into macros. Macros can be called from MAIN directly, through other macros or through the MENU items, when running MAIN in its dialog mode.

The development of MAIN is based on the current requirements of the problems with which I have been more or less directly involved. The program is a network of simple operations that can be applied to atomic, electron density and reflection data. Most tools built into the program are available in variety of other programs as well; however, the way they can be combined is unique and enables a skilled user to deal quickly and efficiently with complex tasks. These aims were achieved by evolving a command language which includes commands for the very elementary operations and which further allows combinations of them in longer sentences and command files. Some available command files also enable an inexperienced user to run the program as a black box.

MAIN philosophy

Programming in steps

To understand why and how to run MAIN the following "illustration" might help: There is an electric cooker with two plates, one hot and the other cold, and a pot of cold water. The problem in the following two scenarios is how to heat the water in the pot. In the first scenario, the pot stands on the table. A mathematician solves the problem by taking the pot and putting it on the hot plate. A physicist also takes the pot and puts it on the hot plate. In the second scenario, the pot of the cold water stands on the cold plate. A mathematician takes the pot from the cold plate, puts it on the table and than acts as in the first case. A physicist takes the pot from the cold plate and puts it directly on the hot plate.

When dealing with molecular model data, there are often problems very similar to the one described above. The tools are computer programs, which are capable of a limited range of operations. When a program capable of one specific operation such as bringing "the pot from the table to the hot plate" exists, we can apply it and the problem is solved.

However, quite often there are "the second case" problems, when there is no program that would bring "the pot from the cold plate to the hot one". There are two alternatives available: either to write a program that brings "the pot from the cold plate to the hot one", or to write a program that brings "the pot from the the cold plate to the table". The decision depends upon the complexity of the operations and skill of the person. In simple cases a text editing program is used to bring "the pot from the cold plate to the table".

The advantage of the "single step" programs is in their simplicity; however, the strategy of writing "single step" programs has its limitations, since the number of the possible connections between various points ("plates" and "tables") increases faster than the number of the points.

The use of a collection of a variety of "single step" programs also has another disadvantage, namely that with an increasing number of programs, the number of steps that connects them increases, increasing also the possibility for error.

MAIN answer to the above problems is the reduction of the programming steps to elementary operations and the control of their combined use inside a single program with a command language user interface, designed according to the programming rule "divide and rule". I understand the development of the single program that can do everything, as a process which may approach its aim, but can never reach it. This is the first premise of MAIN development.

Data transferability - compatibility with other programs

With an increasing total volume of knowledge also comes increasing diversity. Computers enable us to connect, combine and explore knowledge from different sources. This can be achieved when programs can access the data and store the results. The requirement is, however, that the data and programs are compatible. Therefore a program, which can access and store any form of data, is highly desirable. To make the program compatible with as many data forms and sources as possible is the second premise of MAIN development.

Data overview

An individual human being can only possess an overview of a limited set of data. A column of 20 numbers can be read, however a column of 100 or more numbers is already too much. There are two basic approaches presenting large amounts of data; either to use numerical analysis and reduce the data to some reasonable set of numbers, or to use graphical presentation. Graphical presentation by itself has also some limitations, since we are able to distinguish only between a limited set of geometric elements and colors. To create a flexible tool that could handle large quantities of data and present the extracted information in a variety of different forms is the third premise of MAIN development, which follows from the first two.

Error recovery and safety

Every program and computer crashes from time to time, users' mistakes can not be prevented. It is nothing more anoying than repeating a session which includes quite some time and labor.

In order to keep you 'cool', MAIN when running, keeps a complete track of the whole working session in the 'input.cop' file. You can edit this file and extract a session or its part(s) and use it as a command file.

You should follow some safety rules in order to allow MAIN to give you full error recovery:

- never save your results into your input files,
- never delete or modify your 'input.cop' file before you have saved the results,
- do not modify your startup files before you have saved the results

- try to avoid READ-ing data from the system input (terminal), becasue the READ data are not included in the 'input.cop' file.
- regulary saving of your intermediate results (menu item 'SAVE' and the 'save_file.cmds') and not to long sessions, help you to rerun only shorter, labor intensive but CPU notthat extensive sessions.

Rules of syntax

The program interprets command sentences that are typed in from the keyboard, read from a file (command file) or written by the program itself when running in the DIALOG mode.

Each command sentence must begin with one of the command words.

The exclamation mark '!' is used as a comment sign. The program will not try to interpret characters following the exclamation mark. A hyphen (VMS like) or backslash (UNIX like) continue the command sentence onto the next line.

Typing '?' or gives you the list of possible command words.

Typing 'HELP' following or preceeding a command word retrieves information form the command reference manual. This is still not fully operational through the whole program.

The dimension of the command sentence is limited to 2000 characters and 400 command sentence constituents.

All command sentences read on the top input level are stored in the file INPUT.COP that is occupying the FORTRAN unit 80. When a command is read from a file (command file) on a disk, then the program does not copy its content to the file INPUT.COP but only the call. Up to 10 levels call for command files are allowed by the program.

FORTRAN units higher than 80 should not be OPENED since they may be used by the program, however you can REWIND and CLOSE them.

Sentence

MAIN command sentence is a combination of command groups and subgroups. Each command (sub)group starts with one or more command words and can include arguments (integer and

real numbers, atom indices and strings). Command words and arguments must be separated by spaces. Each command sentence begins with a command word. A typed question mark '?' on the place, where a command word is expected, gives a list of possible command words.

Command_word

Command words can be written in small or capital letters. When a command word is expected then MAIN compares character by character of the found string with the list of possible command words. Maximal 8 characters are checked. When a command word is uniquely identified, the remaining up to 8 characters are checked as well, when the string contains these (unnecessary) characters. QUIT for example can be abbreviated to a single letter 'Q', but not to 'QQ'.

In the manual command words are written in capital letters.

Integer_number

Integer can be any integer number (1, -56, 3765, 0). The explicit plus '+' sign is not understand. When an integer value makes no sense or may ever result in an error, MAIN rejects also correctly typed integer numbers. An integer number can be also a result of an arithmetic expression where a list of integers is combined with the operators '+', '-', '*' and '/' .

In the manual integer numbers are denoted as 'inte', 'inte1', 'int2' ...

Atom_number

Atoms can be assigned by their absolute or relative indices (NUMBER). Absolute index is the atom position in the current atom list. Relative index is related to the position in the history list. History list is created by picking atoms from the image (IMAGE HIST PICK atom-num). The last PICKED atom is always the first in the HISTORY list. Absolute atom NUMBERS are integers. Relative numbers start with a dollar sign '$' in front of the integer index.

In the manual atom numbers are denoted as 'atom-num' or 'int'.

Real_number

Any real number is accepted. The use of exponents is not allowed. Also a real number can be also a result of an arithmetic expression where a list of reals is combined with the operators '+', '-', '*' and '/' .

String

String can be any combination of up to 80 characters. When spaces are suppose to be included in the string, the string should be delimited with two double quotes ("), one at the beginning and the other at the end.

Strings can be added together with a '+' sign.

Variables

Each argument (integer or real number and string) can be parsed to the MAIN command sentence interpreter via a variable. Each variable has a name and value. The variable name can be any up to 10 characters long string. The names are case sensitive. The variable value can be an integer or real number or a string. MAIN recognizes the variable type from the argument. When the argument in a SET VARIABLE sentence is an integer number then the variable is an integer variable, when it is a real number, then the variable becomes a real variable, and when it is neither an integer nor a real number, then the variable becomes a string variable. Once a variable name has been exploited, it is not possible anymore to redefine its type (real, integer and string).

Selections and keys

Each atom has in each selection a flag that tells if that atom is selected or not. The command sentence does something with the selected atoms. Selections are defined with the SELECT ... END command subsentence, which on the basis of logical operations merges various atomic properties into a single selection. The SELECT subsentence can be included in almost any command sentence that is related to atoms.

Therefore molecular data need not to be rearanged, grouped or ungrouped, when trying to access a certain part of them. All you have to do is to apply a corresponding selection.

Short description of MAIN utilities

MAIN is the very distinct program for macromolecular crystallography that enables a user to interactively build molecular models into electron density maps and, within the same interactive session, immediatelly refine the built models with crystallographic restraints and recalculate electron density maps for further inspection.

The program is a tool for numerical and visual analysis and modification of molecular data. With MAIN it is possible to construct, modify and analyse 3-D molecular models of large and small molecular systems up to several thousands of atoms. It can be used in model building and analyzing studies, in crystallographic refinement (including positional refinement, R-factor calculation,and placing solvent molecules at the final stages of refinement) and as an electron density maps editor (rotations, averaging, solvent flattening in real and reciprocal space, Fourier transformations ...),... MAIN includes a unit cell generating routine that builds the cell from an asymmetric unit. The asymmetric unit can be placed anywhere in space. The routine is general for any space group, since the equivalent points are calculated applying crystal symmetry operations, and are not generated by applying rules that are different for each symmetry group and position of an asymmetric unit.

MAIN is designed to allow rapid and easy access to any molecular model data using a single program, thus avoiding the confusion and incompatibilities which arise when using a variety of different programs. Each action undertaken by the user that changes the molecular model data is stored in a file (input.cop), allowing easy restoration after a computer or program crash or a user's mistake.

MAIN is driven by command sentences and from a MENU. It has its own language resembling VMS syntax. All files that are read or written by the program are ASCII files, and therefore transferable to any computer. MAIN can be used together with molecular mechanics programs used for energy calculations and X-ray refinement (CHARMM, X-PLOR, GROMOS, EREF), and those for quantum mechanical calculations (GAUSSIANnn, AMPAC). MAIN utilizes X-PLOR force fields for energy minimization. X-PLOR, PROTEIN and CCP4 (ASCII format) electron density maps can be displayed, analysed, modified and transomed. Electron density from any number of local symmetry elements and any number of different crystal forms can be manipulated (averaged, combined, ...).

Molecular models can be built from scratch using the available topology libraries. Topology libraries can be created from coordinates of molecular models. The geometry of molecular models can be modified using command sentences, which apply translations and rotations,

including deorthogonalization, scaling and crystal symmetry operations. Molecular models can be superimposed and compared on the display, and some numerical analysis can also be performed.

The interactively driven geometry changes of a molecular model distinguish MAIN from most other programs. Translations and rotations can be combined with bond rotations (torsions). There can be up to 30 different transformations active, which can be combined in any hierarchical order. Several chains of atoms can be rotated about their bonds simultaneously. Interatomic distances, angles and dihedrals can be interactively displayed while changing the geometry of a model. Manual intervention with a molecular model can be combined with energy minimization procedures, which includes besides the usual bond, angle, dihedral and improper angle, Van der Waals and electrostatic energy terms, also crystallographic restraints, electron density map correlation terms, distance and dihedral angle constraints. Beside the usual ball and stick models, Connolly surfaces and electron density maps can be displayed. Electron density maps are defined in the whole space (based on a P1 unit cell).

A Ramachandran plot of a protein can be displayed and used to monitor the ϕ and ψ angles of a residue as its conformation is changed. Correlation of a molecular model with electron density maps, electron density histograms, temperature factor diagrams, molecular surface distributions, consistency of force field parametrisation regarding discrepancies between ideal and actual geometry, hydrogen bonds etc... can be numerically and graphically (excluding force fiels analyses and map histograms) presented.

To be able to use MAIN, a brief overview of its data structures and the possibilities to change and convert them is necessary.

MAIN data structures

What I call a data structure is a group of data fields within the program related by an index to a term that describes an object. An object might be either a synonym for an atom, residue, electron density map, covalent bond ... or a numerical term like number or matrix. Each data structure may consist of different elements, for example an atom has position (coordinates), name, color etc..

Atoms

Each atom has its position, name, atomic number, color, temperature factor, crystallographic weight, class, partial electric charge and index.

The position is stored either by X, Y, and Z coordinates, that is by default in the Cartesian coordinate system, or by internal coordinates which relate position each atom to another via interatomic distances, angles and dihedral angles. Cartesian coordinates and distances are in Å, and angles in degrees. (In the MAIN syntax a synonym for coordinates is COORDINATES, when a relative positional criterion is chosen in a select sentence, then there are other descriptors in use like AROUND, CENTER, DISTANCE, PLANE ...)

An atom name (called ATOM NAME) is a string up to 4 characters long. According to its place in the periodic table the atomic number is found. (The atomic number is not used in MAIN syntax). An atomic name starting with a letter representing periodic table element symbols H, C, N, O, P or S followed by any character represents a hydrogen, carbon, nitrogen, oxygen, phosphorus and sulphur with the atomic numbers 1, 6, 7, 8, 15, and 16 respectively. Since there are many nomenclatures in use, MAIN has some exceptions from the "periodic table" rule. The CA is the Cα atom of an amino acid residue, and not a calcium atom. A calcium atom must be declared with CAL. Dummy atoms start with X and have atomic number 0.

The temperature factor (called TEMPERATURE) is the crystallographic B value of an atom in Å^2.

The weight (called WEIGHT) is the crystallographic weight factor.

The charge (called CHARGE) is the partial atomic charge in electrons. It is used by energy and electrostatic potential calculations.

Class (called CLASS) is the synonym for atom type in X-PLOR language. According to the atom class the parameters (bond length, angles and force constants ...) from the force field are found.

The index (called ATOM NUMBER) is the consecutive atomic position in the program. Atoms can be accessed directly in image and select sentences.

Each atom has a color (called COLOR) as an integer value. There are default values.

Residues

Each residue has a name up to 4 characters long (ALA) (called RESIDUE NAME), a sequence name up to 5 characters long (A189) (called SEQUENCE). Each residue belongs to a certain segment (called SEGMENT NAME). The segment is a kind of synonym for a molecule. Usually a group of residues have the same segment name and can thereby easily be recognized and selected. A segment name can be up to 4 characters long string. Residues can be selected also by their index. A residue or segment index is defined similarly as for atoms, namely as a consecutive residue or segment number in the program. They are addressed as RESIDUE NUMBER or SEGMENT NUMBER.

Connectivity tables

There are 4 connectivity lists available: In the table of covalent bonds (called CTABLE) each atom has a record in which all attached neighbors are stored. In the internal coordinates table (called ZTABLE) each atom has a record in which related atoms by distance, angle and dihedral angle are stored. The hydrogen bonds table (called HBOND) is a list of atom pairs that form hydrogen bonds donors and acceptors. The pair table (called PAIR) is a list of atoms forming a pair.

Points

Each point has an index that relates the point to an atom, position in Cartesian coordinates (X, Y, Z), an index that specifies the point kind (called SURFACE, VOLUME, DENSITY, POTENTIAL) and the magnitude of the property (called SURFACE, VOLUME, DENSITY, POTENTIAL). There are 2 kinds of SURFACE points: ACCESSIBLE and REENTRANT. The ACCESSIBLE points are the exposed areas of atoms that can be directly in contact with solvent atoms and the REENTRANT SURFACE points are the ones that can still be accessed by solvent, but are no longer on the surface of atoms. The REENTRANT area is the area where solvent enters the protein surface.

Topology library

The TOPOLOGY library consist of residues. Each topology RESIDUE has a name. Each residue includes list of ATOMS, covalent BONDS and DIHEDRAL and IMPROPER angles.

Each atom has its NAME, CHARGE, CLASS and usually also Cartesian and internal coordinates which serve to create and modify the residues. The improper and dihedral angles lists are used to prepare the lists of improper and dihedral energy terms for energy calculations.

Force field parameters

These are lists of BOND, ANGLE, DIHEDRAL and IMPROPER angles, Van der Waals parameters. According to the atom class the force field constants are ordered to each particular term. Besides these there are available also distance (PAIR) and DIHEDRAL angle constraints plus electron DENSITY energy term.

Maps

A map is a 3-dimensional array of grid points, where each of them has a a value. According to the value, they are treated as empty, density or mask points. Grid points with their values inside the density interval are density points, points with values below the density interval are empty points, and ones above are mask points. Each map has size, its starting coordinates and its cell constants. Translational symmetry (P1) of the lattice is always correctly generated. That means that when a whole unit cell is defined in a map, the program can expand the density through the whole space. A complete unit cell map can be Fast Fourier transformed into a reciprocal space map, which can be afterwards modified and back transformed.

Reflections

Reflections are defined within resolution limits and according to reciprocal space zero plane limits set by a READ REFLECTION INIT sentence.

Each reflection has besides its HKL values fields for a real Fobserved value and a complex Fcalculate and Fworkset values. Values between these fields can be added, scaled and multiplied in all possible combinations. This makes possible to calculate various kinds of electron density maps from 2Fo-Fc, Patterson and various correlation maps that can be utilised in the electron density modification procedures (fast solvent flattenig, histogram matching, molecular model packing functions ...).

Reflections are treated as defined when their Fobserved value is greater than zero. Fcalculate is automatically filled after a Fourier transformation of a map has been performed. Fworkset

is used to fill a map for its subsequent back Fourier transformation.

Miscellaneous

Symmetry operations and cell constants

Only one list of symmetry operations is present at a time in the program. A symmetry operation consist of a rotational (3x3) matrix and translational vector. Both are in fractional coordinates. The cell constants can be different for each map, however, for operations on atoms, only the last one read is valid.

MAIN variables and constants

During run time MAIN creates and updates a variety of variables like natom, nsegm (number of atoms and segments), π etc. The user can create additional ones and change some of the existing ones. Their purpose is to enable noninteractive transfer of data to command sentences and files. The use of MAIN variables is useful in connection with the IF command sentence. There are eight 3×3 matrices available to the user. Some of them are used by the program as well (one for image processing, six for the intermediate results, eight for the stereo pair of matrix 1 and by RMS fit calculations).

Selections and Keys

When a selection is created, each atom is assigned a flag which indicates whether the atom is included in the selection or not. Selections can be also stored permanently as KEYS. In almost any command sentence in which atoms are involved, a selection may be used. When a selection is not explicitly required all atoms are selected. In cases like (DELETE ATOM), however, no default selection is allowed, since it can result in unwanted data loss.

Files

All the files MAIN can read or write are ASCII, so that they can be edited. This makes the connections between MAIN and other programs easier. The exception are the PROTEIN and

Lyn Ten Eyck (FFT) formats of a density map. Maps in PROTEIN format can be read and in FFT can be written.

Operations

The basic input and output operations are READ and WRITE, where files of atomic, topology, map, points, symmetry operations ... data can be read or written. Values in existing data arrays can be modified with a SET command sentence and examined with a SHOW command sentence. IMAGE commands define the image parameters and send images to the graphic display. CALCULATE is used to calculate a connectivity lists like BOND, HBONDS and PAIRS by comparing the atomic positions or electrostatic POTENTIAL in points from the atomic charges. MAKE converts data between different forms, e.g. from atoms to maps, from atoms to points, from points to atoms, from maps to atoms, from connectivity table to segments, from map to map ... POINT generates points. RMS and ANALYSE are use to compare various sets of the same property (like COORDINATES, CHARGES ...) Commands like ROTATE, TRANSLATE, SET CHAIN, SET DISTANCE, SET ANGLE, SET DIHEDRAL, OBJECT and MINIMIZE modify the coordinates of atoms or points. MINIMIZE does so by an energy minimization procedure, while the others change the coordinates by rotation and translation. An OBJECT statement is designed to be almost exclusively activated from the menu. IMAGE commands control visualisation of molecules and maps.

Interactive access to the command reference manual through the MAIN syntax will gradually cover complete code. MENU commands (strikes) documentation is already fully linked.

INPUT OUTPUT with open auxiliary programs

Commands related to INPUT and OUTPUT are READ, WRITE, SAVE and SHOW. READ and WRITE relate to the working data forms and formats.

Output from SHOW can be directed into a file, however, which can not be used for MAIN input any more. It is more or less a formated presentation of MAIN status variables and some data arrays.

SAVE is used to save some MAIN data into a form of MAIN command language. The resulting files can be used as MAIN macros.

Map INPUT OUTPUT programs and routines

Several programs have been written for conversion of density maps (PROMAP, PROMAP_R, XPLMAP, XPLMAP_R, MAPFFT, MAPFFT_R, MAPPRO, MAPPRO_R, MAPFFT, MAPFFT_R, DN6MAP).

The coding of program names is based on a 3 letter code. PRO is abreviation for PROTEIN, XPL for X-PLOR, FFT is used for the format required by program P1SF that does the fast Fourier transform of the electron density, DN6 stands for the FRODO DN6 files. MAP stands for the 'native' MAIN format of the map. There are two MAIN type of maps and with corresponding formats: CHARACTER (character*1) and NUMBERED (real*4). MAIN density maps are ASCII files with maximum 80 characters per record. The header of the map with cell constants is the same for CHARACTER*1 and REAL*4 maps. The "_R" denotes that the file conversion is done from or to the REAL*4 type MAIN map, without "_R" it is assumed that the map is a CHARACTER*1 type.

MAIN can read maps directly in its 3 native formats, called CHARACTER (character*1), NUMBERED (real*4) and ALPHA (hexadecimal character*1 map), PROTEIN, XPLOR and CCP4 (ASCII) format maps. The CHARACTER format is the default one so it doesn't have to be specified. A user should be aware that it is possible to read REAL*4 maps with CHARACTER format. No error occurs and warning is written, but the map is going to be interpreted wrongly. PROTEIN, XPLOR and CCP4 maps are converted to real*4 maps.

MAIN can write maps in FFT, XPLOR, AMoRe and it's three native formats. Each map can be written in any format (CHARACTER, NUMBER, ALPHA, XPLOR and FFT). When there is no format specified, defaults are taken. The default format for a character map is a CHARACTER and for a REAL*4 map is NUMBER. By writing the map identifier (integer number) must be specified. The current MAIN version can store up to 10 maps at once.

There are differences in file structure between UNIX and VMS as regards electron density map transferability. CHARACTER and NUMBER forms are not available on every UNIX machine. CONVEX FORTRAN up to now only supports a FIXED record length ASCII files; the SGI, HP and ESV do not allow usage of CHARACTER and NUMBER formats. The ALPHA format was designed to transfer files between VMS and UNIX for their visual inspection. The XPLOR may be used as well, though it produces larger files.

PROMAP, PROMAP_R, MAPOUT, MAPOUT_R and DN6MAP only work under VMS. The PROMAPs can read only PROTEIN release 2 electron density maps. MAPOUTs can write only PROTEIN release 2 map, that are not readable by PROTEIN any more, however MAPAGE is

capable to generate DN6 files for FRODO users.

The syntax looks as follows:

Character format reading:

```
MAIN> read file x.map map
MAIN> read file x.map map char
```

Real*4 format reading:

```
MAIN> read file x.rmap map numb
```

PROTEIN format reading:

```
MAIN> read file x.diz map protein
```

Write file statements:

```
MAIN> write file x.map map 1
MAIN> write file x.map map 1 number
MAIN> write file x.map map 1 character
MAIN> write file x.map map 1 fft
```

Lwplot

Lwplot reads a MAIN plot file (form used for ORTEP, the same as FRODO). The program was written and modified by a series of people. The advantages of the lwplot distributed with MAIN are that a specified window size also cuts the image at its borders and that each time when it is run, a backup of each user answer is written to the FTN79 that can be later on moved/renamed and edited for further tries.

Lately lwplot works also with colors.

Technical data

The source code containes more than 100 000 lines of source code, about 70 000 lines are graphic independent part, and 30 000 lines are the graphic part. The computer language used is FORTRAN, except the X-windows interface that is written in C. MAIN is written in almost standard FORTRAN77. The two exceptions are listed below:

- variable and routine names are longer than 6 characters (this is tolerated by most of FORTRAN compilers),
- the INCLUDE auxiliary statements are not part of the standard and do, therefore, differ from computer to computer.

The non-graphic part is easily transferable to any computer of appropriate capabilities (a VAX and ALPHA/VMS and CONVEX version are available). The graphic interface is written to communicate with Evans $ Sutherland V workstations (the PS300 series version has not change since 1991 and is available though not supported any more), Hewlett Packard workstations (series 700) using both the X-windows environment and PHIGS. The Silicon Graphics version runs in X-window environment and utilizes GL.

Chapter 4

Errors and bugs

One of the most important things, when running a program, is to understand the error messages. MAIN is programmed so that it shouldn't crash whatever a user makes. (Crashing can unfortunately not be completely disabled.) There are two kinds of errors; syntax errors, which are relatively easy to find, and data errors arising from the missing or incompatible data. The later sometimes result in a zero divide or similar.

Each error message starts with descriptive name of subroutine as for example "READ_COOR>". Most of error messages include the string %ERR or %ERROR, so that they could be easier localized in the output file by using a SEARCH or 'grep' utility. When looking for mistakes don't disregard %WARNINGS. The most important is the first error message.

Syntax_errors

When MAIN can not interpret a command sentence, one or a series of error messages appears on the default output. The most common error messages are

```
> COMMAND WORD NOT FOUND
> COMMAND WORD MISMATCH
> FILE file_name.dat NOT FOUND
> SELECTION WENT WRONG
> AN INTEGER NUMBER NOT FOUND OR NOT APPROPRIATE
```

and so on. I hope that they are clear enough. After the error message comes the list of unprocessed strings. These are strings that were not interpreted by MAIN. The first unprocessed string or the one before it are the places where a syntax error occurs.

There are more kinds of the SELECTION errors:

- a syntax mistake by a tag processing
- brackets don't match
- logical processing is not possible (operator .NOT. acts on a tag or expression in brackets, but can not stand in front of .AND. or .OR..
- selection requires the command word END, so that MAIN knows, where to stop searching for tags.

Data_errors

MAIN is quite well protected against syntax errors. The data errors are harder to find and analyse. The messages are diverse or may even not exist. The program may do nothing, but also not crash.

Sometimes may help to increase the LEVEL of messages by a SET LEVEL command. It works for sure when there is a corrupted record in the input file, because reading of a file was interrupted exactly there.

Bugs

There is always one more bug in a program and MAIN is, as the other programs, also not free of them. When you find one, please let me know about it.

Chapter 5

Molecular and map images

Molecular images in MAIN are 3-dimensional rotatable geometrical objects that represent molecular models. Objects can be drawn as lines, points, text, sticks and balls. Each molecular image consist of atoms that can be picked and thereby manipulated via menu items.

Graphical presentation of maps are either lines or polygones (AREAS) delimiting a specified map contour level.

Section "Tryptophan images demo" presents the elementary images based on a tryptophan atom coordinates. Section "Protein images" shows how to display the most commonly used protein model presentations. Section "Combi images" gives you some ideas what you might be able to do with MAIN. Section "Color and light hints" explains how to deal will image objects' color and lights. Section "Background images" explains to you have you can keep images on screen even though, MAIN does not remember them any more. Section "Making plots" brings the essentials about MAIN plot file creation.

Each object name (created by MAIN) starts with a character, that describes its kind (V for vectors or lines, C for text characters, A for areas or polygons, M for markers as are also points). Each object name has exactly the same color as its 3-dimensional presentation.

User access to the pickable atom list is achieved through access to the KEY 'image'. Each new IMAGE commands makes an logical '.or.' to the already existing KEY 'image'. You can, however, edit the KEY any time, either by 'KEY image SELECT ... END' commands or via the SELECT menu block.

IMPORTANT: You can display only objects that exist. If there are no atoms you can not see

them on a display, if atoms are not connected via covalent bonds, you can not expect to see their covalent bond network... If an image object exists a string with its color is displayed on the right edge of the image window. If an object has no color (color equal to the background) you can not see it. The

```
> image list
```

command prints all molecular image names, their status (D-displayed, R-removed or E-ereased) and their color (index). When you have created a molecular image object and you don't see it, then your SCALE or CENTER might be wrong. The

```
> image center calculate select ... end
```

command is usually successful, although the CENTER of atom selection (arithmetic average of all selected atom coordinates) may sometimes be placed into an empty region between atoms. This usually happens when you are displaying several molecules which are quite separated.

You can INCLUDE and REMOVE each object from the display list by activating INCL_REM image menu mode and clicking the objects.

Run the mol_images examples from your directory by starting MAIN with MAINPS followed by the "¡" or "@" command preceding the file name. After processing the command file type IMAGE DIALOGUE to enter the DIALOGUE mode, where mouse and dials further control the program.

The section "Simple images" should enable you to get the mostly required types of images with the tryptophan molecule. "Protein images" are a further step gradually introducing use of SELECT ... END sentence. "Combi images" look prettier although their purpose can not be immediately obvious. A few hints regarding possible color manipulations are presented in the section "Colors and lights".

If you want more, you should become familiar with the chapter "IMAGE" in the "Reference manual".

Tryptophan images demo

Copy the file '>mol_images/simple/.main' to your directory and start main. The '.main' file is automatically envoked when you start main. It reads the coordinates of a tryptophan molecule

and creates all neccessary data structures you can display by clicking the menu items. It calculates its covalent bonds on default distance criteria, point surface and a molecular envelope. It initializes the image display and expect your respond:

The atom coordinate file is read and covaleny bonds are calculated by checking all possible interatomic distance. When an interatomic distance is below a threshold value, then a bond is created.

```
> read file >doc/mol_images/simple/trp.pdb atom pdb
> calculate bond
```

Connolly surface points generation requires atomic radii, so they are SET first to default vdW values. You can inspect atomic radii using the SHOW command. DENSITY 10. means 10 points per square AA of the surface.

```
> set rad select all end vdw
> show radi select atom number 1 end
> poin surface all sele all end dens 10. init
```

To display a molecular envelope (a map) cell parameters, a map with its GRID sizes and its box (10 grid points AROUND the atoms) should be first created (INITIALIZE -9999). The actual envelope points are all points closer than 1.4AA from any atom center (ATOM MASK 1.4).

```
> read cell
> 20. 20. 20. 90. 90. 90.
```

```
> make map 1 from 0 init -9999 grid 30 30 30 \
>         around 10 sele all end real
> make map 1 atom mask 1.4 sele all end
```

Here the DEMO_IMAGE bock is loaded to give you interactive control over the image composition. The items are explained below, click them and look at the command syntax that creates images.

```
> <>doc/mol_images/simple/load_image_demo
```

IMAGE is INITIALIZED, CENTER set to the ATOM number 3 and the image SCALE is set to 0.4. Some image is brought to the screen, so that you do not stare into nothing. Dial and mouse control of the image is enforced for this demo with the IMAGE FORWARD command. (Do not use the FORWARD command in your everyday work, use IMAGE DIAL instead.)

```
> image init cent atom 3 scale 0.4
> image set bond
> image forward
> return
```

File ">doc/mol_images/simple/load_image_demo.com" creates the DEMO_IMAGE menu block, its color, a documenation link to a file that does not exist, and defines all its items.

```
> menu block DEMO_IMAGE create
> menu block DEMO_IMAGE color 190
> menu block DEMO_IMAGE document "does_not_exist"
```

Each item is a single command line up to 80 characters long and composed of a string within two double quotes (following the word TEXT). You can store this commands in a file or type them if you wish. When it was not possible to achive the desired effect within a sinle command line then they were stored in macros, and a menu items became a macro call.

The 'QUIT' command quits the MAIN, it does notonly exit the dialog mode. IMAGE ERASE deletes al molecular images, but does not effect map presentations.

```
> menu block DEMO_IMAGE item QUIT     add text "quit"
> menu block DEMO_IMAGE item ERASE    add text "image erase"
```

These items define colors that will be later on used. COLO_SET (IMAGE SET) tells the program to take individual atomic colors, while the IMAGE COLOR items (BLUE, YELLOW, GREEN and RED), address color directly by their MAIN index. Only the last COLOR definition is applied in later IMAGE commands, that actually display something on the screen.

```
> menu block DEMO_IMAGE item COLO_SET add text "image set"
> menu block DEMO_IMAGE item BLUE     add text "image color 240"
> menu block DEMO_IMAGE item YELLOW   add text "image color 160"
> menu block DEMO_IMAGE item GREEN   add text "image color  200"
> menu block DEMO_IMAGE item RED   add text "image color  116"
```

Atoms can be denoted in space by 3-D crosses (ATO_CROS), atom names or spheres (ATO_BALL). The balls image is in a macro because the atomic radii applied are set to their default values each time the item is clicked.

```
> menu block DEMO_IMAGE item ATO_CROS add text "image atom cross"
> menu block DEMO_IMAGE item ATO_NAME add text "image atom name"
> menu block DEMO_IMAGE item ATO_BALL \
>      add text "<>doc/mol_images/simple/atom_ball.cmds"
```

Covalent bonds can be presented either as lines (BON_LINE), interatomic distances (BON_DIST) or sticks (BON_STICK).

```
> menu block DEMO_IMAGE item BON_LINE add text "image bond"
> menu block DEMO_IMAGE item BON_DIST add text "image bond dist"
> menu block DEMO_IMAGE item BON_STIC add text "image bond stick"
```

The ball and stick presentation (BAL_STIC) reduses the atomic radii, so that the bond sticks can be seen.

```
> menu block DEMO_IMAGE item BAL_STIC \
>      add text "<>doc/mol_images/simple/ball_stick.cmds"
```

The surface points can be presented in various color modes (SURFACE) and any of their combinations (SURF_MIX).

```
> menu block DEMO_IMAGE item SURFACE  add text "image point surf all"
> menu block DEMO_IMAGE item SURF_MIX \
>      add text "<>doc/mol_images/simple/surface_mix.cmds"
```

Map image control is separated from from the molecule images. It always starts with IMAGE MAP command words. It has its own erase command (MAP_ERAS), line syntaxes (MAP_LINE, MAP_Y) and polygonal presentation (MAP_AREA). Maps can be essentailly contoured along any of the 3 possible directions (X, Y or Z), here only Y is chosen.

```
> menu block DEMO_IMAGE item MAP_ERAS add text "image map erase"
> menu block DEMO_IMAGE item MAP_LINE \
```

```
>      add text "image map 1 dens 0.9 15 15 15 area off"
> menu block DEMO_IMAGE item MAP_AREA \
>      add text "image map 1 dens 0.9 15 15 15 area on"
> menu block DEMO_IMAGE item MAP_Y    \
>      add text "image map 1 dens 0.9 15 15 15 y-ax area off"
```

At the end the MENU PAGE 1 is initialized so that only this menu appears on it.

```
> menu page 1 init
> menu page 1 add DEMO_IMAGE
> return
```

Protein images

This demo is an extension of the previous one to proteins and a slightly more complicated setup of menu structure, but with a for user better overview.

Go to the directory 'main/doc/mol_images/protein' and type 'mainps' or copy the file '.main' to your directory and then use the menu blocks.

The '.main' file reads in the atomic coordinates and the Connolly surface file. If the surface file is not available yet (you are doing it for the first time) uncomment the surface generation lines.

```
> read file >doc/mol_images/protein/throm_ppac_fin.xpl atom xpl
> set rad sele all end vdw
> key noh sele .not atom name H* end
> set rad sele all end vdw
> !poin surf sele segm name TRO* .a noh end sele segm name TRO* .a noh end \
> !   dens 5.0 init
> !write file >doc/mol_images/protein/thro.surf poin surf
> read file >doc/mol_images/protein/thro.surf poin init surf
> <>utils/protein.bond
> <>doc/mol_images/protein/load_image_demo.com
> ima init
> ima set bond cent calc sele segm name PPAC end
> image forward
> return
```

There are four different menu blocks created (DEMO_CONTR - green, DEMO_COLOR - red, DEMO_SELECT - white and DEMO_IMAGE -light blue).

Menu block DEMO_CONTR is capable of erasing the entire image and it can also 'QUIT' MAIN.

```
> menu block DEMO_CONTR create
> menu block DEMO_CONTR color 190
> menu block DEMO_CONTR document >doc/mol_images/protein/demo_image.txt
> menu block DEMO_CONTR item QUIT     add text "quit"
> menu block DEMO_CONTR item ERASE    add text "image erase"
```

Menu block DEMO_COLOR switches between the atomic color set (COLO_SET), which applies atomic individual colors, and different exlicitly defined colors (BLUE, YELLOW, GREEN, RED).

```
> menu block DEMO_COLOR create
> menu block DEMO_COLOR color 115
> menu block DEMO_COLOR document >doc/mol_images/protein/demo_image.txt
> menu block DEMO_COLOR item COLO_SET add text "image set"
> menu block DEMO_COLOR item BLUE     add text "image color 240"
> menu block DEMO_COLOR item YELLOW   add text "image color 160"
> menu block DEMO_COLOR item GREEN   add text "image color  200"
> menu block DEMO_COLOR item RED   add text "image color  115"
```

Menu block DEMO_SELECT selects atoms according to their segment names (A_CHAIN, B_CHAIN PPAC), residue groups (SID_ACID, BASE) or simply ALL.

```
> menu block DEMO_SELECT create
> menu block DEMO_SELECT color 1
> menu block DEMO_SELECT document >doc/mol_images/protein/demo_image.txt
> menu block DEMO_SELECT item A_CHAIN \
>      add text "key active select segm name TRO1 .a noh end"
> menu block DEMO_SELECT item B_CHAIN \
>      add text "key active select segm name TRO2 .a noh end"
> menu block DEMO_SELECT item PPAC \
>      add text "key active select segm name PPAC .a noh end"
> menu block DEMO_SELECT item SID_ACID  add text \
>       "key active select resi  name ASP GLU .a .not atom name H C N O end"
```

```
> menu block DEMO_SELECT item BASE \
>      add text "key active select resi name ARG LYS .a noh end"
> menu block DEMO_SELECT item NO_HYDRO \
>      add text "key active select noh end"
> menu block DEMO_SELECT item ALL \
>      add text "key active select all end"
```

Menu block DEMO_IMAGE combines colors and selections setups as specified by the DEMO_COLOR and DEMO_SELECT items into the images brought to screen. CA atom trace (CA_TRACE), main chain trace (CHAIN_TR), precalculated Connolly surface (SURFACE). The NICE image is a defined macro, which displays protein and its surface points, by highlighting the bound inhibitor (using ball and stick presentation).

```
> menu block DEMO_IMAGE create
> menu block DEMO_IMAGE color 220
> menu block DEMO_IMAGE document >doc/mol_images/protein/demo_image.txt
> menu block DEMO_IMAGE item BONDS \
>      add text "image select active end bond"
> menu block DEMO_IMAGE item CA_TRACE \
>      add text "<>doc/mol_images/protein/CA_trace.cmds"
> menu block DEMO_IMAGE item CHAIN_TR \
>      add text "image select atom name C N CA .a active end bond"
> menu block DEMO_IMAGE item SURFACE \
>      add text "image select active end poin surf all"
> menu block DEMO_IMAGE item NICE \
>      add text "<>doc/mol_images/protein/nice.cmds"
```

Finally the menu page 1 is initialized with only the DEMO_ menu blocks offered on it.

```
> menu page 1 init
> menu page 1 add DEMO_CONTR
> menu page 1 add DEMO_COLOR
> menu page 1 add DEMO_SELECT
> menu page 1 add DEMO_IMAGE
> return
```

Combi images

Free your imagination and donate your own creations. Here are some command macros:

```
>doc/mol_images/combi/churches.com
>doc/mol_images/combi/balls.com
>doc/mol_images/combi/hbonds.com
```

Color and light hints

MAIN internaly works with color maps (Video Lookup Tables or VLT in short) although they appear to a graphics hardware as floating point Red Green Blue (RGB) values. So when you try to color an image object, you refer to it by an index in the color map. Default color 0 is black, 1 is white, colors between 64 and 255 go around in a circle, when you rotate a dial, from blue(64), violet(90), red(110), orange(145), yellow(160), green(200), light blue(220), dark blue(245) and 255 and as next to 64 and reverse.

Molecular images can be colored from atomic colors

```
> image set
```

or explicitly by a color index.

```
> image color 90
```

After an image is displayed, you can enetr the COLOR mode (click the COLOR in the image window) and then the object for which you want to adjust the color: A dial will be attached to this object and while you turn it, the image objects color will go around in the color interval index between 64 and 255. After a RE_DRAW this colors will be replaced by the initial values, so when you want to keep them you have to either grab their index (IMAGE LIST) or not press RE_DRAW or any function that modifies molecular model geometry. This does not apply to map images, they keep the color you have adjusted with a dial regardless of that what you do with the map.

Each atom has its color index defined by default on the basis of its atomic number. You can change atom colors with SET COLOR commands:

```
> set color sele atom name C .a segm name MOLB end col 200
> set color sele resi name ARG LYS .a segm name MOLB end col 110
```

Light affect only polygonal (AREA) images. Light is only reflected on AREA and not on lines (vectors), points and text objects. It is possible to manipulate at least ten lights (computer hardware dependent) simultaneously. A light can be ambient, directional, positional or a spot light. For more details see IMAGE LIGHT commands and SET DIAL ... LIGHT manipulation of light parameters in the "Reference manual".

Background images

Background images are images prevented from deletion when commands as 'IMAGE ERASE' and redraw ('IMAGE OVER FALSE') are invoked. A typed

```
> image from
```

command sets the image erase counter to the current number of image objects. A following IMAGE ERASE command will only clear the images from the MAIN memory, but will keep them on the screen. This is the only way to keep GRAPH images as Ramachandran plot on screen while doing molecular modeling, because MAIN does not remember them at all. The typed

```
> image from 0
```

command set the image erase counter to 0 and thereby enables deletion of every image object. You can off course set the counter to any valid integer number.

I recommend you to build this command in your startup files to show the unit cell and initial model chain trace. See also the chapter "A real case".

Making plots

The simplest way to prepare a plot of an image is to enter dialog mode and click on the menu item PLOT. The file name to which the image object are to be plotted is INQUIRED from

the standard input window ("PLOT FILE NAME:"). The resulting file can be afterwards converted to a PostScript file by using the program 'lwplot' written and modified by a number of people. For MAIN users backup of typed answers, definition of window size and color use was added to it.

Plot invokes redrawing of the entire image, therefore the background object and image objects displayed through using GRAPH command sentences can not be redrawn.

No polygons can be plotted (balls, sticks and map areas).

GRAPH objects can be plotted with some adittional effort. Since there is no GARPH redraw option, you should do the redrawing explicitlly by yourself. Once the desired GRAPH objects scaling is set OPEN a plot file and redraw the GRAPH objects by repeating the drawing commands (the best way to do it is to copy commands from the input.cop file):

```
> open unit 79 overwrite file YOUR_PLOT_FILE_NAME
> image unit 79 plot on initialize
```

here insert the GRAPH COMMANDS

```
> image plot off
> close unit 79
```

and close the file.

Any area objects (atom balls, bond stick and map areas) are not saved to the plot file.

This MAIN plot file is later on processed with the program 'lwplot', which you can find in the directory 'main/progs' and should be in your 'PATH' already. 'Lwplot' is a simple program, so just try to run it. When executed it creates a backup file with all your answers to questions on a FORTRAN unit 79 (fort79 or similr). This file you can edit and use as input for later 'lwplot' runs. Do not forget to rename ('mv') it, otherwise it will be overwritten by the next run. With a few iterations it is usually possible to adjust scales and window sizes. The output of 'lwplot' is a PostScript file ready to be previewed or printed.

Bellow is such an input file: first comes the output file name, then some declarations of size and fonts, then the input MAIN plot file and then you have to answer which objects to see with which line style and thickness. It works with RGB colors from the screen, too.

```
omit_map.ps
```

```
omit map at 1.85
     !stereo
     3.80     !window
     6.00     !ch-siz
     0.00     !ch-offset
     8.00     !scale
     4.00     !stereo-sep
omit_map.plot
   0     ! vl_1
     0.00
   0     ! vl_2
     2.00
   0     ! h_at
     0.25
   0     !  m_1
     0.05
```

Chapter 6

Model building

User interaction with a molecular model through menu is certainly a distinguished part of MAIN. So do not try to follow what you have learn when using other programs, explore MAIN possibilities first. Combine energy MINIMIZATION with manually driven geometry changes and their control (achieved through HISTORY), SECONDARY structure elements and SELECT. Do not forget about ANCHORS.

Run the "¡>cmds/load_depp_page" to load additional menu blocks.

Menu driven model building

From scratch

The simplest way to enable MAIN to tackle model building is to call a command procedure as

```
> <>utils/get_top_par_19_csd
```

or some similar get_top_par file. Such a procedure reads a selfsuficint set of topology libraries and energy parameter files. The "19" are protein libraries and "11" are DNA. The assigned value of the DEF_ALL variable points to a command file macro name which builds the energy term lists.

Afterwards start using MENU functions.

When available command procedures do not cover your needs, topology libraries and parameter files should be read individually. As examples available get_top_par procedures should be examined. When desired files are not available, they should be created from scratch. For instructions see section "Creation of topology libraries" - when it will be written.

Find a menu page with the red molecular BUILDER menu (1,9). It works in 3 modes (START, ATTACH or CHANGE residue). The current mode is indicated by a filled area. Only the "RESID ??" item really acts, all the others just set flags or variable names.

The items (SEGMENT, SEQUENCE and RESID ??) require a string from the terminal window, where MAIN was started. This may cause some problems when using crystal eyes stereo, so you are advised to switch to a mono image when calling these functions.

When picking SEGMENT you are required to type in a new SEGMENT name that will be applied when you will start creating new segments. Similar yields for SEQUENCE, a new SEQUENCE root is applied when you create new residues. If you want to continue counting ATTACHED residues with the same sequence root as to the residue from which the ATTACHED chain started, syou have to specify the same sequence root it explicitly. "RESID ??" requires a string of RESIDUE names.

When the molecule builder is in CHANGE mode only a single RESIDUE name is accepted, where in other two modes any combination of RESIDUE names and their repetitions is valid. However, it is not possible to place 10 solvent MOLECULES at the same positions with "10 H2O". MOLECULES can also not be attached to any other residue.

STARTING a new chain

For example we want to build alanine chain 10 residues long followed by a tryptophan and 3 cysteines. The chain starts to grow from the current position of the image center, so it should be set properly by dials or mouse. The BUILD menu should be in START mode, a SEGMENT name and SEQUENCE root are supposed to be defined. Afterwards picking item "RESID ??" activates a chain generation.

The "RESID ??" prompt should be answered with desired residue sequence "10 ALA TRP 3 CYS". The desired sequence appears as an extended chain growing from the middle of the image screen.

CHANGING a residue

You can change a residue by setting the CHANGE mode and picking an atom of the residue you want to change. This should be followed by a click on item "RESIDUE ??" which prompts for a new residue. A ¡return> is followed by a residue change and an IMAGE update.

APPENDING a new chain

We may APPEND a new chain of peptides to the C-terminal of a residue by picking an atom of the selected residue, setting the ATTACH mode and responding to the "RESID ??" prompt with appropriate residue names.

N- and C- termini

For attaching the N-terminal atoms setthe builder mode to APPEND click an atom of the N-terminal residue, and APPEND to it the NTR residue. 3 new (HT) hydrogen atoms will apper. Click the old H atom and delete it (by clicking the DELE_ATO in the RENAME menu block). Now click an HT atom and then some other atom from the N-terminal residue and merge the two residues inot a single one (click the MERG_RES from the RENAME menu block). Appending NTR (or NTRP for prolines) deletes the bond between the C atom of the terminal residue and N atom of the next one. So restore it (click the two atoms and then MAKE_BON item from the MAKE_DELE menu bolck).

Similarly the C- terminal carboxylic group is created. First rename the C-terminal carbonyl oxygen into OT1 (click the atom, clock REN_ATOM in the RENAME menu block and then type OT1). Switch on the APPEND model building mode and append CTR residue to the C-terminal residue. Now click the OT2 atom and merge the CTR residue into the C-terminal residue (click the MERG_RES from the RENAME menu block).

Dials driven geometry changes

The geometry MENU as whole is stored on page 2 in dark green color, the essentials are present on the page 9. RT_ANGLE, RT_CHAIN, MOV_DIST, MOV_ATOM, ROTATE, TRANSLATE, INV_CHIR and CONNECT define OBJECTS while OB_START, OB_ACCEP and OB_REJEC either return atoms to initial positions, by keeping active OBJECTS definitions (OB_START),

or by rejecting definitions and returning to the previous stage (OB_REJEC), or accept changes (OB_ACCEP). INV_CHIR converts a chirality of an atom by swapping positions of its two selected side-chains. CONNECT brings together two molecules into a reasonable geometry and joins them with a covalent bond.

A common rule is first to pick atom(s) and afterwards activate a function. The advatage of this logic is double: it enables you to correct the picked atoms list without a need to start all over again and the function does need no aprovement. You only hit the function once when everything is ready for its' action.

A dial set is not always automatically saved. When using RT_CHAIN you may wish to save the current dials definition (DIAL_SAV) in order to be able to switch among different dials sets without losing the chain rotation dials.

SECONDARY structures

The SECONDARY structure elements MENU is placed on the 3rd MENU page and show in violet. Secondary structure elements can be applied in block (HEL_ALPH, HEL_3-10, COLLAGEN, POLI_PRO, BET_ANTI, BET_PARA, EXTENDED) by acting on the ACTIV selection or as turns (TURN_I, TURN_II TURN_III and their 'primes', GAMMA and GAMM_IN-verse truns) acting on the next four or three residues.

See also reference manual for instructions about SELECT MENU.

Hiting UN_DO (depp page 9) retrieves the previous coordinate set in a case of an undesired change.

ENERGY calculations and MINIMIZATIONS

Energy can be calculated for each energy term turned on assuming that all neccessary force filed PARAMETERS have been read. The bonding (BOND, ANGLE, DIHEDRAL and IMPROPER) and nonbonding (VDW and ELEC) terms are defined as for example in X-PLOR. In addition to these energy can be calculated also for PAIR list, DENSITY anticorrelation and DIHEDRAL constraints.

The squares of energy turns on are filled, while the others appear in the ENERGY menu as empty. Clicking the items turns each individual term on or off.

```
> ener all off bond on angle on dihe on impr on
```

The atoms included in the first selection interact with the atoms included in the second. When only one selection is specified it is assumed that the second one is equal to the first. When ENERGY callculation is initiated from the DIALOG mode by picking the MENU item SHOW_ENE the "active" key atoms interact with the "passive" key atoms.

```
> energy select all end
> energy select active end sele passive end
```

The simplest way to define the selections is to hit the depp page 9 items ACT_NEIGH, ACT_SEGM or ACT2RES. The active atoms will be for a short time indicated by white marks on the screen.

To calculate ELECTROSTATIC and VDW interactions partial atomic CHARGES and atom CLASSES have to be assigned to each atom individually. Atom CLASSES are further used to DEFINE lists for energy the bondidng energy terms: BOND, ANGLE, DIHEDRAL and IMPROPER. See for example the ">utils/def_top_par_19.com" file, which is the default macro invoked through the DEFINE and DEFINE_A items. You can redirect these calls by redifining the variable DEF_ALL. Try

```
> show vari DEF_ALL
```

to check the variable status.

```
> defin clas charge by topo
> defin bond angle by auto
> defin dihe improper by topo
```

DENSITY energy term is a negative correlation of a "real" atom with the density of the choosen map. See also "Use of anchors" below and the "Load_xray_maps" in the Dialog mode accessories.

```
> ener density map 1 density scale 30.
```

are starting values.

PAIR constraints require previously DEFINED pair list. Two atoms in a pair are pooled together or apart to a specified distance with a force (stored in the variable "force_pair") acting as a harmonic oscilator (analytically identical to the BOND energy term). PAIRs defined with the menu items MAKE_PAR and ANCHOR are assumed to be pooled together to a zero distance, while clicking the PAIR_D_? inquires from you to type in the desired distance explicitly.

The last option is very useful when you try to model a complex of a small inhibitor with a protein as described in a paper. The structure of the complex is, not availabe to you, however in the paper some distances are often tabulated.

INI_PAIR initializes the pair list. All pairs are deleted.

Use of anchors

Anchor is a dummy atom connected by a PAIR with its original atom. Between an atom and its anchor is acting a force pooling them together, assuming that the PAIR energy term has been turned on. Anchors can be freely moved in space by the MOV_ATOM or TRANSLAT menu item. Their positions may be optimized in an electron density map (MINI_ANC). Properly set anchors can help you to get atoms to the exact positions you want to have them. They may also be used to mark initial chain tracing and can pull a chain to a desired position in an electron density map.

Anchors are supposed to be generated through menu options (see Dialog mode accessoriess) unless you have a strong reason to define them manually. (In such a case generate ANCHORS from the menu and see later the input.cop file for the syntax.)

The SELECT menu offers you more control over the key, however the simplest way to define anchors is to initialize the history list (HIS_ERAS) and then clik the atoms for which you want to define anchors and then hit ANCHORS on the depp page 9, which will provide anchors for the whole history list.

Command sentence driven model building

The most evident way is to SET the message LEVEL to 3 and afterwards observe the terminal window, where each picked menu item command sentences are echoed or look into the 'input.cop' file, where the command sentences are stored.

Commands neccessary for model building Manual model building combines usage of commands within the two modules BUILD and INTERNAL with command sentence CALCULATE COORDINATES.

Only MAIN topology files include data required for building models. Each topology residue includes besides ATOM NAMES, CLASSES, covalent BONDS and DIHEDRAL and IMPROPER energy term lists of the residue also X, Y and Z coordiante of each particular atom and INTERNAL coordiantes.

The X, Y and Z atom cooridnates define geometry of the starting residue in a chain, INTERNAL coordinates enable generation of further (ATTACHED) residues.

For more detailed description of topology files see section "Creation of topology libraries".

Creation of an extended chain

Following sequence of coordinates generates a SEGMENT or MOLECULE from the RESIDUE names and links it together according to the INTERNAL COORDINATES information from topology library

```
> build
> start
> append residues 10 ALA
> exit
```

In the cases, when the starting residue is not a standard amino acid as is ACE (CH3-CO), the Z-TABLE should be modified by manulally changing the Z-TABLE connectivities of the second residue, since ACE includes only 3 atoms

```
>inter
> set ztab 4 all 2 1 3 0
>exit
```

Z-TABLE of the atom number 4, which is the first atom of the following residue, herewith obtains references to the atoms 2, 1 and 3. The position of atom number 4 is defined by a DISTANCE to atom number 1, ANGLE formed by atoms 4, 2 and 1, and a DIHEDRAL angle to 4, 2, 1, 3.

The CALCULATE COORDINATES further converts internal coordinates to CARTESIAN. The APPEND flag is neccessary to prevent modification of already defined X, Y and Z coordinates.

```
> calc coor appe
```

Changing a residue

A residue can be changed or mutated by entering the BUILD module, SELECTING the residue to be changed, and finally converting it to a TRP. Again the CARTESIAN coordinates are calculated from INTERNAL coordinates.

```
> build
> select residue number 11 end
> change residue TRP
> exit
> calc coor sele residue number 11 end append
```

Positions (cartesian coordinates) of atoms with equal names in old and new residue remain constants. Only the positions of the nonmatching atoms are calculated.

This sometimes leads to rediculous geometries, however, it also helps to prevent orientation of the changed side chain as close as possible to its parent. When newly created geometry appears to be unacceptable, the residue conversion should be done in two steps. First by replacing the parent by a small residue as ALA and then convert the ALA to the desired residue. (Geometry can however be improved also by using an energy MINIMIZATION procedure.)

Generation of SEQUENCE and SEGMENT names

When STARTING a new chain it makes sense to give the new chain a new SEGMENT NAME and its SEQUENCE root. SEQUENCE root is a string to which numbers are beeing appended, starting with 1. The SEQUENCE root AB results in sequence of AB1, AB2, ... AB10, ... AFTER specifies the residue after which the new chain will be INSERTED. 'AFTER 0' indicates the position in the front of any already present residue in the program. This is essentially the only way (besides using a text editor to move segments around) to insert a segemt in front of all other atoms.

```
> build
> sequence A
> segm MOL0
> after 0
> start
> insert resi ALA PRO GLU MET
> exit
```

Attaching residues

ATTACHING residues requires a SELECTED residue to which the newly generated chain will be attached. A new SEQUENCE root may be specified. SEGMENT name is not used. A number of repetitions may preceed a RESIDUE name. "3 CYS" is equivalent to "CYS CYS CYS".

```
> build
> attach
> sequence AX
> after sele numb res 4 end
> insert resi 3 CYS
> exit
> calc coor appe
```

Chapter 7

Electron density modifications

Electron density can be manipulated by any mean that will yield a more interpretable electron density map, finally resulting in a completly resolved and refined structure of a macromolecule. Most widely used and verified methods are electron density averaging and solvent flattening (Bricogne, Wang). These methods improve phases by including extra information from two sources, the local symmetry within an asymetric unit and the demarkation of areas occupied by protein and solvent. These then alter electron density maps, from which the whole crystal cell is built and structure factors are calculated, beginning a new cycle. The procedure is repeated until the apparent R-factor of the electron density map converges The aim is to produce an interpretable electron density map.

In last years number of programs dealing with electon density modifications and phase improvements is increasing (see for example papers in Acta Cryst. 1993 D49), so MAIN certainly is not the only choice, however, it offers a relatively simple command language structure and does not limit a user to a particular method or their combination or to a paticular asymetric unit or space group or number of crystal forms. MAIN allows imediate verification of procedures through its graphical facilities - thus making life in this field relatively easy.

MAIN offers a map manipulation language, which allows you to do much more as can be possibly described here. If you want to master more than the usual essentials, you are advised to read the 'MAKE MAP' commands from the "Reference manual".

Section "Solvent flattening" describes how to run the standard reals space procedure (Wang,) and the fast one, which uses the Parseval's correlation theorem to calculate the molecular envelope in reciprocal space. Section "Electron density averaging" brings you essentialls of

how to setup an electron density averaging procedure. Two procedures in two subsections are described. Subsection "Averaging of electron density: 2 molecules" explains a procedure within a single unit cell and is easy expandable to any number of equivelent density regions. Subsection "Averaging density between 2 different crystal forms" explains how to average electron density of two equal molecules in two different crystal forms. Section "Electron density combination" explains how and why combine different origin electron density maps in order to construct better maps. Its extension partial model refinement is described in chapter "Crystallographic refinement" in section "Partial model refinement".

Besides these there are some auxiliary sections, which help you to deal with certain concepts and tools connected with MAIN map manipulations. Section "Making envelopes" shows how envelopes can be constructed with MAIN. Section "Local histogram matching" introduces s short reciprocal space based histogram correlation procedure, which can be built in into any kind of map calculation. Section "Map concepts" gives insight into map structure and map manipulation command language.

Map concepts

Molecular envelopes

In the case of proper averaging one molecular envelope (mask) suffices for all molecules, while for improper averaging the procedure should distinguish between different molecular areas. For this reason the concept of labeled masks was introduced by Bricogne. In MAIN this is solved by storing each molecule's mask in a separate map. Since on each map only a single operation can be performed at a time, there are no principle differences between improper and proper averaging procedures.

Crystal cell generation

The usual way to generate a complete crystal cell from an asymmetric unit was to write a procedure that applies the building rules. Building rules tell which grid point in a cell is equivalent to which one from the asymmetric unit. This approach has several drawbacks which can turn averaging (and solvent flattening as well) to a complicated procedure full of errors. First, for each different space group, different rules should be applied and second, in almost any space group it is possible to choose different definitions of an asymmetric unit. The third complica-

tion sometimes arises from differing numbers of grid points in a unit cell. These routines are, however, not available for every possible case and have to be, when necessary, programmed. Also when they are available their correctness should be verified for each single case. It happened quite often that there were whole empty layers left. At this point real problems may begin specially for an unskilled programmer. Therefore in MAIN another approach is applied: A crystal cell is generated from an asymmetric unit by applying crystal symmetry operations. The consequence of this generality is that there are no limitations on placement of an asymmetric unit and the routines work generally for any space group. Besides the asymetric unit may consist of several independent parts, each one stored in a separate map. The MAIN approach has also another advantage: since only empty points can be modified, there is no danger of having multiple density in certain regions.

Maps: enevelopes, density filled regions

A map is a 3-dimensional array of grid points, each with a value. According to the value, they are treated as empty, density or mask points. The grid points with values inside the density interval are the density points, the ones with values below the density interval are empty points, and the ones above are mask points. In the case of character*1 maps 0 is an empty point and 255 is a masked point. In the case of real*4 maps the empty points have values below -9999.0 and masked above 9999.0. The region inbetween comprises the density points.

Each map has size, starting coordinates and cell constants (cell constants are needed for transforming maps from differents cells). Grid points lying in different unit cells with the same fractional coordinates are identical. That means that, when a whole unit cell is defined in a map, the program can expand the density through the whole space.

There are 9 elementary operations types that can be done with maps:

- Creation of a map,
- Creation and extension of a masked or an empty region,
- Rotations and translations of a map,
- Building the crystal unit cell from an asymmetric unit or its parts,
- Setting values to selected grid points,
- Scaling a single map with a constant,

- Rescaling a map to 1 sigma and shifting its average to 0.0,
- Adding and merging two maps and
- Creation of electron density from atomic positions.

The smallest map contains only a header where cell constants and crystal symmetry operations and number of grid points along the cell axes are stored. MAIN can read PROTEIN, X-PLOR and ASCII CCP4 as well as its own native formats of electron density maps. The maps can be written in Lyn Ten Eyck, X-PLOR and MAIN native formats. The native formats are ASCII files with record length 80 so that they can be edited and changed with a text editor. MAIN map files will be changed to unformatted, since some UNIX systems (Hewlett Packard and Silicon Graphics) cause problems when dealing with fixed length ASCII files. CONVEX FORTRAN causes however no problems. Besides a variety of smaller conversion programs were written to enable conversion of maps between PROTEIN, X-PLOR, P1SF, FRODO and native MAIN formats.

Operations on a map grid point can be applied when the point is empty or masked. (The exception is the SET command that can set a value to a grid point in any specified range.)

A new map can be created from an already existing one by taking its cell constants and size or from scratch. The map size, origin, and number of grid points per cell length and cell constants can be taken from an already existing map. The grid points are initialized to a specified value. The map origin and size can also be defined from an atom selection so that selected atoms plus some boundary grid points lie inside it.

Masked or empty points can be defined in several ways:

- By setting all grid points with their values in specified range to the mask value.
- By a distance criteria from a selection of atomic center positions
- By conversion of grid points to real space points and then to atoms and further to mask grid points.
- By converting unmasked grid points that lie between masked points to masked points.

A map can be transformed (rotated, translated or copied) into the mask points of another map by linear interpolation. This is done so that the position of a mask point is transformed into the space of the map with density. The masked point density value is then obtained by interpolation from the eight surrounding grid points.

Electron density created from atoms can be added to density of a grid point or it can replace the value of an empty grid point. Empty points of a map can be filled with values of the density points of another map by applying crystal symmetry operations. This routine is independent of crystal symmetry. In order to find the position of the grid point into which the density value should be copied rotation matrices and translation vectors are applied.

Two maps at a time can be added or merged together. Adding differs from merging in the way, that only two equipositioned grid points having a density valu can be added together, while merging two maps allows scaling of both map contributions prior to their adding together. When a grid point has no density value its value taken from the other map.

Solvent flattening

It is assumed that the areas of the electron density map where protein lies have areas of greater positive electron density than the solvent areas. The solvent occupied areas inside the crystal have no rigid structure, so they can be 'flattened' (all grids points in the solvent region obtain a single density value - zero).

Solvent flattening, when successful, defines clear boundaries between the solvent and protein occupied volumes and improves the electron density.

The standard procedure

The following group group of commands performs a solvent flattening procedure following Wang (Wang et. al., 1985), where summ of density at all grid points FROM map 1, which density is above a certain threshold value, in a ceratin radius around the central point weighted by distance is stored into the equivalent central point in map 2. Map 2 is further analyzed - a histogram of density points in the 80 density intervals is prepared. (Number of intervals can vary from 1 to the maximal number of atoms your MAIN can accept). This histogram serves later on to preparve the envelope from which a newly generated unit cell is later used for a new structure factor calculation. The procedure is faster then the original programs. It can be significantly accelerated if points in an asymmetric unit are solvent flattened only. It is also possible to limit the algorithm to certain regions of map 2 by applying masks that do not include the whole map.

```
>examples/solv_flat/solv_flat.com:
```

First read an electron density map.

```
> read file molmap.asc map ccp4
```

Create a new map in a region, that may be an asymmetric unit only, and INITIALIZE its values to mask.

```
> make map 2 from 1 init 9999
```

Calculated the weighted densities FROM map 1 and store them in the MAP 2. In this case the density is CUT at 0.0, chosen RADIUS is 8.0AA, and weigting with distance is chosen.

```
> make map 2 from 1 cut 0.0 radi 8.0 weight
```

ANALYZE the just created smeared density in 80 intervals. Especially at the beginning it make sense to write the analyses results.

```
> analyze init
> analyze map 2 step 80
> !write anal
```

Cut 0.33 - one third away - and copy the original points into the proposed nonsolvent region. (*)

```
> make map 2 analyzes range 0.33
> make map 2 from 1 copy
```

Generate a complete unit cell map. BOX 0 0 0 99 99 99 assumes that cell starts at 0,0,0 and includes 100 grids along each cell axes. Fill it with the solvent flattened density and write it to a file in the FFT form for the following FT.

```
> make map 3 from 1 init -9999 cell
> make map 3 from 2 cell
> make map 3 set -10000 -9900 0.0
> write file solv_flat.map_fft map 3 fft
```

```
> quit
```

(*) Before COPYING the density, it may make sense to store the score file and maybe fill the cavities and so generate a more compact envelope:

```
> write file score.rmap map 2 number
> make map 3 from 2 init 9999 copy
> make map 3 from 2 fill 4
> make map 2 set -10000 10000 9999.
> make map 2 from 3 copy
```

The fast solvent flattening procedure

The difference between the standard solvent flattening procedure described by Wang and the herein described fast procedure is in the scrore map calculation. The standard procedure calculates the weighted sum of positive electron density within a sphere. The fast procedure is based on Parseval's theorem which allows to calculate the scoring map (basis of a molecular envelope) in reciprocal space. Fourier coefficients of the weighting scheme P1 map and Fourier coefficients of the positive parts of the map to be flattened are multiplied and then backtransformed. The resulting map is a score map: convolution of the weighting scheme and the posiive parts of the map.

The fast procedure is not numerically identical to the standard one because of Foruier series truncation error, however, it is sufficiently accurate to be used. The major advantage of the fast procedure is that it is incomparable faster than the standard one. It can be performed interactively at a workstation display. Its speed is not sphere radius dependent. See also SOLV_FLAT menu item in MAP_ATOM menu block.

With a help of a single variable (HIST_TEMP) it is possible to apply additional map modification (a local histogram matching) which is essentially the probability of finding an atom at map positions.

```
>examples/solv_flat/solv_flat_fast.com:
```

Read an MIR map, symmetry operations and a reflection file.

```
> read file mir_clear_3.5_heavy.xmap map xplor
```

```
> read file >symm/c2.symm symm
> read file ../refl/catl.fobs refl init limit -99 -99 0 \
>           friedel reso 3.3 100.0
```

Set variables MAP_DENS (origin of density) and MAP_SOLV (solvent falttened map).

```
> set vari MAP_DENS = 1
> set vari MAP_SOLV = 2
> set vari SOLV_CUT = 0.4
> set vari SOLV_RAD = 8.0
> set vari HOLE_SIZE = 6
> set vari HIST_TEMP = 0.0
```

Initiate a unit cell with the density set to EMPTY and a density of an atom that is placed at coordinate system origin and which density is distance dependent (1. - d / SOLV_RAD). Set the empty points to zero, Fourier transform the map and save its Fourier coeficients in the FWORKSET array.

```
> make map MAP_SOLV from MAP_DENS init -9999  cell real
> make map MAP_SOLV atom dist SOLV_RAD
> make map MAP_SOLV set -100000 0 0.0
> fourier map MAP_SOLV
> reflect set ampl phase fwork = fcalc * 1.0
```

Copy the MIR density into the new MAP_SOLV, set all negative density values to zero and Fourier transform it. R-value calculation is performed in order to follow convergence of a further procedure.

```
> make map MAP_SOLV from 2 init 9999  cell real
> make map MAP_SOLV from MAP_DENS copy
> make map MAP_SOLV set -99999 0 0.0
> fourier map MAP_SOLV
> reflect shells 10 r-values
```

Multiply FWORKSET (sphere model) and FCALCULATE (truncated MIR density) Fourier coeficients, convert the MAP_SOLV map to a convex map, fill it with the multiplied Fourier coefficients and Fourier BACK transform it. (The DEFINE option and application of all symmetry operators in the reflection fill-map command is OK too.)

```
> reflect set ampl phase fwork = fwork * 1.0  *  fcalc * 1.0
> make map MAP_SOLV conv complex
> make map MAP_SOLV zero
> refl fill-map MAP_SOLV symm 1 1
> four map MAP_SOLV back
> make map MAP_SOLV rescale
```

Perform the score map analyses (map occupancy in 256 steps), convert the selected parts to a MASK region and copy density into it. Set the density of the empty region to, Fourier transform it and caluclate R-values.

```
> anal init
> anal map MAP_SOLV
> make map MAP_SOLV anal SOLV_CUT
```

Remove small lonely clouds of the mask and fill the holes (in several iterations):

```
> make map MAP_SOLV + 1 from MAP_SOLV init 9999 copy
> make map MAP_SOLV from MAP_SOLV + 1 remove HOLE_SIZE
> make map MAP_SOLV + 1 from MAP_SOLV init 9999 copy
> make map MAP_SOLV from MAP_SOLV + 1 remove HOLE_SIZE
```

```
> make map MAP_SOLV + 1 from MAP_SOLV init 9999 copy
> make map MAP_SOLV from MAP_SOLV + 1 fill HOLE_SIZE
> make map MAP_SOLV + 1 from MAP_SOLV init 9999 copy
> make map MAP_SOLV from MAP_SOLV + 1 fill HOLE_SIZE
```

```
> make map MAP_SOLV from MAP_DENS copy
> make map MAP_SOLV set -99999 -9000 0.0
> fourier map MAP_SOLV
> reflect shells 10 r-values
```

Save the FCALCULATE values in the FWORKSET array and calculate an Fobs map

```
> reflect set ampl phase fwork = fcalc * 1.0
> reflect set ampl fwork = fobs * 1.0
```

or 2 Fobs - Fcalc map.

```
> reflect set ampl scale fcalc fwork = fobs * 2.0 - fcalc * 1.0
```

At this point you can call the MAIN atomic probability function

```
> if HIST_TEMP .gt. 2.0 <>utils/hist_match
```

```
> make map MAP_SOLV zero
> refl fill-map MAP_SOLV symm defined
> fourier back map MAP_SOLV
> make map MAP_SOLV rescale
```

Here you can either save or display the solvent flattened map or calculated the next cycle by redefining the variables (see also >cmds/solv_flat.cmds file):

```
> set vari MAP_DENS = 2
> set vari MAP_SOLV = 3
```

Solvent flattening procedure based on a skeletonized map

See menu items SKELETON and SOLV_MAS in menu block MAP_ATOM. In a while a better description will appear also here.

Electron density averaging

Electron density averaging requires clear boundaries between protein and solvent volumes, it should be almost inevitably preceded by a solvent flattening in the case of multiple isomorphous replacement phase evaluation.

There are two types of non-crystallographic or local symmetry: proper (also called spherical) and improper symmetry. Molecules of an asymetric unit are related by proper symmetry when they can be superimposed upon each other by a single rotation about a local symmetry axis, while in the case of improper local symmetry, superposition of the molecules require other operations (rotation usually combined with translation). Therefore, procedures for proper and improper symmetry averaging differ. For proper symmetry averaging, equivalent areas do not have to be separated, while for improper averaging it is necessary to distinguish between them.

Besides improving the phases (and the electron density maps) of the starting resolution range, it is possible to evaluate phases of higher resolution reflections by gradually increasing the resolution range. The procedure is called phase extension. The larger the number of molecules in an asymmetric unit, the better the results which can be obtained with phase extension, with however the condition that the initial phases are sufficiently correct for the procedure to converge properly (Bricogne, 1974, Podjarny, 1990).

To perform real space electron density averaging some initial set of phases, equivalent areas and geometric transformations (rotation and translation parameters) are required.

In a molecular replacement procedure, the equivalent areas are defined from the initial model placement. Transformations between them can be easily constructed by superimposing the molecular models (as demonstrated in the cases of cathepsin B and riboflavin synthase).

In the case of a single or multiple isomorphous replacement procedure, it is possible to construct the transformation parameters from the heavy atom positions when they fulfill the local symmetry conditions. When they do not, it is necessary to construct the rotational parameters by an autocorrelation of a Patterson map and then to find the center of rotation and corresponding translational components by autocorrelating electron density (as demonstrated in the case of carbamoylsarcosine hydrolase). To recognize the boundaries of equivalent areas, it is recommended first to average the initial electron density map, whereby the uncorrelated areas are supposed to smear out, and in this averaged density to then find the borders (envelope) of the equivalent areas interactively at a graphical display by going through all the layers of an asymmetric unit (2-dimensional construction using a program such as X-CONTOUR (Buchberger, 1990)) or by preparing a 3-dimensional map representation of a whole asymmetric unit and determining the equivalent parts from its 3-dimensional skeletonized image (this can be done with MAIN).

The theoretical basis for electron density averaging or real space averaging was established in late 1960s and early 1970s (Rossman, 1972). Bricogne has written a review (Bricogne, 1974) of the method, its application and limitations. There it was shown that the averaging in direct space is equivalent to the procedure in reciprocal space and that averaging in direct space has advantages over reciprocal space procedures. The reason seems to lie in the greater computational inaccuracies in reciprocal space calculations. In 1976 he published a description of his program (Bricogne, 1976). Besides Bricogne's program, other attempts have also been made, but none of them (Johnson in Rayment *et al.,* 1978; Nordman, 1980) is used so widely and in so many variations as Bricogne's. Recently, Lawrence (Lawrence, 1991) reviewed the method and its applications to *de novo* determined structures.

Averaging of electron density: 2 molecules

This procedure describes and performs averaging of electron density of two locally equivalent molecules that together form an asymmetric unit of a single crystal form. The procedure is intentionally not optimized, so it is easy to adapt it to any number of locally equivalent molecules. It is also assumed that your envelope as well as translational and rotational parameters can be obtained from molecular models.

The file do_all.com controls the run of the whole procedure. It first reads all neccessary data and set variables (file names, total number of avraging cycles (CYCLE_N), reads the data (cell constants, symmetry operations, molecular model and reflection file).

```
> set lev 2
> set vari FILE_CELL cb-inh-p21.cell
> set vari FILE_SYMM >symm/p21.symm
> set vari FILE_REFL cb-inh-p21.fobs
> set vari FILE_ATOM hucathb_10t_2.xpl
> set vari CYCLE_N = 4

> read file FILE_CELL cell
> read file FILE_SYMM symm
> read file FILE_REFL reflect init resol 3. 8. limit -99 -99 0
> read file FILE_ATOM atom pdb
```

The further procedure refers to MOL1 and MOL2 as segment names so be cautious when renaming your segment names. Segment name can contain only up to four characters (MOL1 is equal to MOL10)!

```
> rename segm MOL1 sele segm name CBA1 CBB1 end
> rename segm MOL2 sele segm name CBA2 CBB2 end
```

The rms_fit.com file determines the local symmetry transformations by optimizing the superposition of the two molecules and writes results (rotation matrix and translation vector in a form of MAIN command file that is later on applied within the averaging loop).

```
> <rms_fit
```

Here the unit cell is created.

```
> set vari MAP_CELL = 1
> set vari GRID_A = 128   !    herm fft direction - must be even
> set vari GRID_B = 48
> set vari GRID_C = 128
> make map MAP_CELL from 0 init -9999 \
>      grid GRID_A GRID_B GRID_C  cell real
```

On the unit cell basis molecular envelopes for each of two molecular models are created within the file make_masks.com.

```
> <make_masks
```

The unit cell is first filled with molecular model density and then (calc_a_map) retransformed into the first 2Fobs - Fcalc electron density map.

```
> make map MAP_CELL \
>      atom dens select segm name MOL1 MOL2 end \
>      atom symmetry
> <calc_a_map
```

The variable CYCLE is the electron density averaging cycle counter. The aver_loop.com procedure is completed when the CYCLE reached the total number of cycles (CYCLE_N).

```
> set vari CYCLE = 0
> <aver_loop
```

At the end the final map and structure factors are stored to a file.

```
> write over file aver_dens.xmap map MAP_CELL xpl
> write over file aver_fcalc.refl reflect fobs fcalc polar defined

> quit
```

The rms_fit.com and make_masks.com only generate the neccessary data based on molecular models for the subsequent averaging. When molecular models are not available, it is neccesary to determine geometrical supperposition of localy equivalent regions by methods that optimize

electron density superposition, and to create molecular envelopes on procedures based on solvent flattening and map skeletonisation (see description above and description of menu itmes SKELETON, SOLV_FLAT).

The 'aver_loop.com' is easily applicable to any kind of electron density procedure, It is relatively easy to expand the procedure to a larger number of molecules in asymmetric unit.

```
> set vari CYCLE = CYCLE + 1
> sh vari CYCLE
> if CYCLE .gt. CYCLE_N return

> set vari MAP_DENS = MAP_CELL + 1
```

Rotating and adding density maps together. Remember that each molecular envelope should be an average of electron density of all of them. So call all 'mol_1_to_2.com' files followed by add_a_map.com' and store the averaged map on a disk. If you number of maps is small you may think of keeping them in memory all teh time, howvere, the procedure should be then slightly modified. (See below to the two crystal forms averaging.)

```
> set vari FILE_MASK mol1_mask.xmap
> set vari FILE_DENS mol1_dens.xmap
> read file FILE_MASK map xpl over MAP_DENS
> make map MAP_DENS from MAP_CELL copy
> <mol_1_to_2
> <add_a_map
> make map MAP_DENS rescale
> write over file FILE_DENS map MAP_DENS xplor

> set vari FILE_MASK mol2_mask.xmap
> set vari FILE_DENS mol2_dens.xmap
> read file FILE_MASK map xpl over MAP_DENS
> make map MAP_DENS from MAP_CELL copy
> <mol_2_to_1
> <add_a_map
> make map MAP_DENS rescale
> write over file FILE_DENS map MAP_DENS xplor
```

Unit cell density generation should call each locally averaged map one after another. The number of points which are incorporated into the unit cell should be more or less equal for all

envelopes. If that is not the case, then you have substantial envelope overlap and you should return to the 'make_mask.com' procedure.

```
> make map MAP_CELL zero
> make map MAP_CELL set -99999 99999 -9999.
```

Repeat this part until all averaged maps are processed.

```
> set vari FILE_DENS = mol1_dens.xmap
> <aver_map_to_cell
>
> set vari FILE_DENS = mol2_dens.xmap
> <aver_map_to_cell
```

Calculate a new map and jump to the top ('rewind file').

```
> <calc_a_map
> rewind file
```

The file 'add_a_map.com' read current molecular envelope, gets its rotated and translated density and adds it to the base map which is, when the density summ is complete, averaged.

```
> read file FILE_MASK map xpl over MAP_DENS + 1
> make map MAP_DENS + 1 from MAP_CELL \
>     rota matr MAT_ROT \
>     tran vect XTRAN YTRAN ZTRAN \
>     center coor 0 0 0
> make map MAP_DENS from MAP_DENS + 1 add
> return
```

The file 'aver_map_to_cell' incorporates an averaged density map into the unoccupied grid points of the unit cell by applying all symmetry operations.

```
> read file FILE_MASK map xpl over MAP_DENS + 1
> make map MAP_DENS + 1 from MAP_CELL \
>     rota matr MAT_ROT \
>     tran vect XTRAN YTRAN ZTRAN \
```

```
>     center coor 0 0 0
> make map MAP_DENS from MAP_DENS + 1 add
> return
```

The file 'calc_a_map' calculates a new structure factor set, its r-value and then the map which eneters the next cycle of averaging:

```
> make map MAP_CELL set -10000 -9900 0.0
> fourier map MAP_CELL
> reflect shell 12  r-values
> refl set ampl phase fwork = fcalc * 1
> refl set ampl scale fcalc   fwork = fobs * 2 - fcalc * 1
> make map MAP_CELL zero
> make map MAP_CELL convert complex
> reflect fill-map MAP_CELL defined
> fourier map MAP_CELL back
> make map MAP_CELL rescale
> return
```

Averaging density between 2 different crystal forms

All command files you can fine in the directory 'main/doc/dens_mod/aver_dens/aver_2_crys/'. This procedure differs from the above described single crystal form averaging in the following:

Since MAIN is not capable to deal with several refelction sets and symmetry groups simoultaneously these have to be read each time when they are referenced. To make file manipulation easier (and error free) 'get_data_hex.com' and 'get_data_ortho.com' files were created. They are only seting values of string variable to apprpriate file names.

```
> set vari FILE_CELL MY_PROJECT:p6522/model/cell_1.dat
> set vari FILE_SYMM >symm/p65_2_2.symm
> set vari FILE_REFL MY_PROJECT:p6522/model/p622all.fob
> set vari FILE_ATOM MY_PROJECT:p6522/model/model_26.xpl
> set vari FILE_MASK mask_hex.xmap
> set vari FILE_MAP cell_hex.xmap
> return
```

The 'rms_fit.com' calculates the rotation and translational parameters of the procedure and saves the parameters into appropriate files referenced in the averageing part below.

```
> <get_data_ortho
> read file FILE_ATOM coor pdb
> key MOL1 sele all end
>
> <get_data_hex
> read file FILE_ATOM coor pdb
> key MOL2 sele .not MOL1 end
>
> key ca sele atom name CA .a segm name CB_1 end
>
> rms coor all \
>     sele ca .a  MOL1 end \
>     sele ca .a  MOL2 end
> save over file mol_ortho_to_hex.com rms
>
> rms coor all init 180. 0. 0. 10. 0. 0. \
>     sele ca .a  MOL2 end \
>     sele ca .a  MOL1 end
> save over file mol_hex_to_ortho.com rms
>
> return
```

The 'make_masks.com' file is replaced by two command files 'make_mask_hex.com' and 'make_mask_orth where each of them defined molecular envelopes and also generates the firts molecular model based 2Fobs-Fcalc maps and writes them to files.

```
> <get_data_hex

> read file FILE_CELL cell
> read file FILE_SYMM symm
> read file FILE_REFL reflect init resol 3.3 100. limit 0 0 0 friedel
> read file FILE_ATOM atom pdb
> set weight sele atom name H* end 0.0
> !set temp sele all end 30.0
> key KEY_MASK sele all end

> set vari MAP_CELL = 1
> set vari MAP_MASK = 2
> set vari HOLE_SIZE = 8
> set vari ATOM_RAD = 3.5
> set vari GRID_ARO = 10
```

```
> make map MAP_CELL from 0 init -9999 grid  80 80 162 cell real
> make map MAP_MASK from MAP_CELL init -9999  \
>      around GRID_ARO sele KEY_MASK end  real
> make map MAP_MASK atom mask ATOM_RAD select KEY_MASK end
```

Fill the holes in envelope that have HOLE_SIZE grid points in diameter

```
> make map MAP_MASK + 1 from MAP_MASK init 9999 copy
> make map MAP_MASK from MAP_MASK + 1 fill HOLE_SIZE
```

Generates symmetry equivelent atoms and subtracts their volume from the original model.

```
> symm sele all end around sele all end dist 6. cut
> make map MAP_MASK set 9000 100000 100.
> make map MAP_MASK atom mask ATOM_RAD / 2.  sele segm name #* end
> make map MAP_MASK set 9000 100000 -9999.
> make map MAP_MASK set 0 1000 9999.
> write over file FILE_MASK map MAP_MASK xplor
```

Finaly the electron density map is calculated. You can also shift this part into the 'do_aver.com' file as in the above case.

```
> make map MAP_CELL zero
> make map MAP_CELL set -1 1 -9999.
> make map MAP_CELL atom dens \
>                   atom symm select segm name MOL2 end
> make map MAP_CELL rescale
> <calc_a_map
> write over file FILE_MAP map MAP_CELL xplor
>
> return
```

The 'do_aver.com' reads this files, initiates and performs the averaging.

```
> read file cell_hex.xmap map xpl over 1
> read file cell_ortho.xmap map xpl
```

```
> set vari RESOL_HEX = 3.3
> set vari RESOL_ORT = 3.3
> set vari CYCLE_N = 6
> set vari CYCLE = 0
> set vari MAT_ROT = 3
> set lev 2
> <aver_loop


> write over file cell_hex_aver.xmap map 1 xpl
> write over file cell_ortho_aver.xmap map 2 xpl
> write over file dens_hex_aver.xmap map 3 xpl
> write over file dens_ortho_aver.xmap map 4 xpl


> set lev 0
> return
```

The 'aver_loop.com' starts with the loop couning and exits when sufficient number of cycleshas been performed.

```
> set vari CYCLE = CYCLE + 1
> sh vari CYCLE
> if CYCLE .gt. CYCLE_N return
```

Rotating and adding density maps together:

```
> read file mask_hex.xmap map xpl over 3
> make map 3 from 1 copy
>
> read file mask_hex.xmap map xpl over 4
> <mol_hex_to_ortho
> make map 4 from 2  \
>     rota matr 3 \
>     tran vect XTRAN YTRAN ZTRAN \
>     center coor 0 0 0
> !wr over file dens_ort_to_hex.xmap map 4 xpl
> make map 3 from 4 add
> make map 3 rescale
> !wr over file dens_aver_hex.xmap map 3 xpl
```

```
> read file mask_ortho.xmap map xpl  over 4
> make map 4 from 2 copy
> read file mask_ortho.xmap map xpl
> <mol_ortho_to_hex
> make map 5 from 1  \
>     rota matr MAT_ROT \
>     tran vect XTRAN YTRAN ZTRAN \
>     center coor 0 0 0
> !wr over file dens_hex_to_ort.xmap map 5 xpl
> make map 4 from 5 add
> make map 4 rescale
> !wr over file dens_aver_ortho.xmap map 4 xpl
```

The new unit cell generation and a new 2Fobs-Fcalc map calculation for each crystal form independently:

```
> <get_data_hex
> read file FILE_CELL cell
> read file FILE_SYMM symm
> read file FILE_REFL reflect init resol RESOL_HEX 100. limit 0 0 0 friedel
> set vari MAP_CELL = 1
> make map MAP_CELL zero
> make map MAP_CELL set -1 1 -9999.
> make map MAP_CELL from 3 cell
> <calc_a_map
```

```
> <get_data_ortho
> read file FILE_CELL cell
> read file FILE_SYMM symm
> read file FILE_REFL reflect init resol RESOL_ORT 100. limit 0 0 0 friedel
> set vari MAP_CELL = 2
> make map MAP_CELL zero
> make map MAP_CELL set -1 1 -9999.
> make map MAP_CELL from 4 cell
> <calc_a_map
> rewind file
```

Local histogram matching

Local histogram matching in contrary to global history matching, where histogram of the whole map is matched, calculates match of an density map with an atom locally for each point of the whole unit cell map.

The following procedure ">utils/dens_comb.com" calculates a histogram of a single atom (sulphur) placed at the coordinate system origin in a P1 cell with its' b-value equal to HIST_TEMP, and structure factors for its translational function throughout the unit cell. These structure factors are later Fourier backtransformed into a density function. Essentially probability to find such an atom at a position in a unit cell is calculated.

This procedure is intended to be called from other procedures as a subprocedue, which, after a structure factor calculation and its storage in the "fwork" array, multiplies it with the structure factors of the histogram. Product can be afterwards (after the return) transformed into a map that serves for inspection or further calculations.

This file is made to be called from a solvent flattening procedure and utilizes the MAP_SOLV for its structure factor calculation. In your own cases change the variable name or assign accordingly the MAP_SOLV and HIST_TEMP values.

main/utils/dens_comb.com

```
> read atom
> *
> SGX


> set weight sele numb atom natom end 1.0
> set temp sele numb atom natom end HIST_TEMP
> make map MAP_SOLV zero
> make map MAP_SOLV conv real
> make map MAP_SOLV atom dens sele numb atom natom end
> delete atom sele numb atom natom end
> four map MAP_SOLV
> reflect set ampl phase fwork = fwork * 1.0  *  fcalc * 1.0
> return
```

Chapter 8

Electron density map calculations

This chapter guides you through essential steps of electron density map calculations:

- generation of electron density map from a model,
- Fast Fourier Transformation of the unit cell, and extraction of structure factors (Fcalc),
- transformation of structure factor amplitudes and/or phases based on simple arithmetic operations between Fcalc, Fobs and Fwork
- expansion of working set amplitudes and phases (Fwork) in P1 reciprocal space unit cell,
- calculation of electron density map with back Fourier transformation and if desired,
- normalization of map electron density (rescaling).

Section "Preparation steps" describes how and which data structures you should prepare and organize before you start a map calculation. Section "Get structure factors" explains how to create or use an initial map for a further Fourier transformation. Section "Prepare your working set structure factors" explains how to manipulate structure factors before calculating a map. Section "From structure factors to a map" describes how to get an electron density map from the working set structure factors.

Preparations steps

First read the cell constants and symmetry operators.

```
> read cell
> *
>    86.51    34.15    85.53     90.0   103.1  90.0
> read file >symm/p21.symm symm
```

The cell constants are neccessary for generation and organisation of reciprocal space data (REFLECTIONS). Reflection data field is index with an READ REFLECTION INITIALIZE command, where RESOLUTION range and LIMITS of reciprocal space oktants are also specified.

MAIN only uses a hemisphere of reciprocal space data for Fourier transformations, although you can generate the whole sphere. Reading Friedel pairs does not hurt MAIN. They are automatically merged while reading.

X-plor reflection files can be read directly.

```
> read file ../p21-b/cb-inh-p21.fobs refl resol 2.0 8.0 \
>         limits -99 0 0 init
```

Get structure factors

Structure factors (Fcalc) are retreived from a map automatically after its Fourier transformation has completed. For this purpose you can either read a map or generate it. If you decide to read in a map then you should know that the most suitable form of a map is exact unit cell.

```
> set vari MAP_ONE = 1
> read file my_map.xmap map xplor over MAP_ONE
```

If the map is not an exact unit cell you should generate it.

```
> make map 2 from 1 init -9999 cell
> make map 2 from 1 cell
> set vari MAP_ONE = 2
```

You should be aware that reading a map replaces the current MAIN cell constants. These should not differ from the above specified ones, otherwise structure factor manipulation can be effected.

When you generate structure factors from a molecular model, you should read in the model and create an empty unit cell map before you start generating the density. Generate a unit CELL map with the specified number of GRID points along each crystallographic axes. A grid size should be at least one third of resolution. Take care that the number of grids along rotational or screw axes is dividable with the fold of the symmetry and the number of grids along X (a) should be even in order to allow the real FFT to run.

```
> set vari MAP_ONE = 1
> make map MAP_ONE from 0 init -9999 grid 128 54 128 real cell
```

Read in the coordinates.

```
> read file ../ca030/hucathb_13_temp.xpl_save coor xpl
```

Before you actually generate density in a unit cell it is wise to be sure that all atoms were also accordingly recognized and that all required scattering constants are available.

```
> write select element 900 1000 end coor xpl
```

If this write command produced any output you should check this atomic records and at least set these atoms to dummy, because they will not be considered anyway, you may only get a bounch of error messages.

```
> set weight select element 900 1000 end 0.0
```

If you are not sure if MAIN already includes the correct and complete set of scattering factors related to your molecular model atoms, write and check them:

```
> write scatter
  C 2.31  20.85 1.0 10.21 1.59 0.57 0.86 51.65 0.216 0.0 ! 6 1
```

A line of output records contains the element identifier name and the nine gaussian coefficients atomic structure factor description (from international tables). The numbers after the exclamation mark are the atomic number of the scatterer and a record counter. You can change the gaussians by a READ-ing SCATTERING FACTORS. Element names can be modified or

added with help of READ ELEMENTS command. There is a room for 110 different elements and them corresponding scattering factors.

Now model density can be genareted. Here it is assumed that all symmetry operators will be exploited, you can however limit their application by explicitly refering to one of them with symmetry operator consecutive number following the SYMMETRY command word.

```
> make map MAP_ONE atom symm -
>            atom dens -
>            sele all end
```

If you wish you can sink your model density so, that its average density becomes zero by a MAKE MAP RESCALE command (a kind of solvent correction) or only set the empty points to zero and perform a Fast Fourier Transformation (FFT) of the map.

```
> make map MAP_ONE rescale
> make map MAP_ONE set -10000 -100 0.
> fourier map MAP_ONE
```

The FFT automaticaly extracts structure factors into the FCALCULATE array.

Prepare your working set structure factors

This subsection explains several possibilities how to merge PHASE and AMPLITUDE information from FCALCULATE and FOBSERVED into FWORKSET in order to BACK transform an electron density map.

In order to combine amplitudes of FCALC with FOBS these two arrays need to be sacled to each other. Scaling constants are obtained with an R-VALUE calculation. MAIN calculates scale constants for each resolution shell independently. This resolution shell independent scaling allows to use all measured reflections including the low resolution terms. Simultaneously an overall reciprocal space B-value, is also calculated. R-VALUE and with it scaling constants and overall B-value depend upon the number of resolution shell divisions applied. This then of course effects also refinement of your model. Only defined reflections with FOBS and FCALC > 0 and reflections for which I/sigma is more than the specified threshold value (2.0 is the default) are accepted into the R-VALUE calculation.

```
> reflection shells 10 sigma 2.0 r-values
```

Here you may wish to write the Fobs, Fcalc and the phase information of defined reflections (defined reflections are those with Fobs > 0)

```
> write over file cb-inh-p21.phase refl fobs fcalc phase defined
```

Now the desired AMPLITUDE and PHASE information should be transferred into the FWORK-SET array that will later fill the complex map MAP_ONE that will be transformed into electron density with a Fourier back-transformation.

Here it is decided what kind of map will be calculated. For a 2Fobs - Fcalc map calculation the Fcalc should be scaled to Fobs in using scaling constants from the last R-VALUE calculation. First the PHASES and AMPLITUDES are copied from FCALC to FWORK. Essentially only the PHASE information should be brought to FWORK, but since FWORKSET is an array of complex numbers, their phase can not exist without a certain nonzero amplitude. SCALE FCALC means that FCALC should be scaled in accordance with the last R-VALUE calculation.

```
> reflect set ampl phase fwork = fcalc * 1.
> reflect set ampl scale fcalc   fwork = fobs * 2. - fcalc * 1.
```

For an Fobs - Fcalc map calculation the scales of FOBS and FCALC amplitudes need to be set to one. Also here SCALE FCALC means that FCALC should be scaled in accordance with the last R-VALUE calculation.

```
> reflect set ampl phase fwork = fcalc * 1.
> reflect set ampl scale fcalc     fwork = fobs * 1. - fcalc * 1.
```

A Patterson Fobs map requires only a square of FOBS with zero phase.

```
> refl set ampl phase fwork = fobs ^ 2
```

An Fcalc map with all reflection (P1 cell) needs only the FCALC information.

```
> refl set ampl phase fwork = fcalc * 1.
```

In addition to these you can also scale Fcalc resolution dependent with your own TEMPERATURE factor and a resolution shell independant constant.

```
> refl set temp 20. const 1.0 ampl phase scale fcalc   fwork = fcalc * 1.
```

For a translation function (or a correlation calculation) you need to transform the first FCALC set into FWORK, then perform another FCALC calculation and then multiply FCALC with FWORK and store results in FWORK:

```
> refl set ampl phase fwork = fcalc * 1.0 * fwork * 1.0
```

From structure factors to a map

When FWORKSET contains the desired structure factors they have to be inserted into a map for a further Fourier transformation. Usually all SYMMETRY operations (default) and DEFINED reflections are inserted. At this stage RESOLUTION can be modified. By default the whole specified RESOLUTION range is taken, you can however change it. For any case first clear the map and convert it into a complex map.

```
> make map MAP_ONE zero
> make map  MAP_ONE convert complex
> reflect fill-map  MAP_ONE defined resolution 3. 8.
```

Now perform Fourier back-transformation of the complex map MAP_ONE and RESCALE the map to sigma = 1 and average = 0. If the map shift is not zero, this is a good indication that something went wrong.

```
> four map  MAP_ONE back
> make map  MAP_ONE rescale
```

Finally you can write the map to a file or look at it on the screen.

```
> write over file test.xmap map MAP_ONE xplor
> quit
```

Interactive map calculations

In this case electron density maps will be calculated within an interactive model building session and the molecular model will be crystallographically refined. Menu items RE_PHASE, 2FO-FC_MAP, FO-FC_MAP, REFINE are used. The description of these menu items macros is presented in chapter "A real case" and in the part "Dialog mode accessories".

You should pay attention to different map variables used in these commands. nameply MAP_2FOFC, and MAP_FOFC are used instead the above MAP_ONE. The sequence of RE_PHASE, MAP_2FOF and MAP_FOFC items performs it all, assuming however, that the macros are available and properly adjusted.

```
<re_phase.cmds
<calc_2fofc_map.cmds
<calc_fofc_map.cmds
```

Chapter 9

Crystallographic refinement

MAIN calculates positional and B-value derivatives out of the Fobs-Fcalc electron density map by so called difference Fourier refinement method.

Crystallographic R-value coming out of MAIN is a result of a fitting procedure, where Fcalc are scaled to Fobs independently for each resolution shell. The default number of shells is 10. The overall reciprocal space temperature factor is calculated independently for each r-value calculation during positional refinement procedure, and it is set to zero during B-value refinement procedure.

The conventional (model Fcalc against Fobs) refinement happens with the MINIMIZE commands. There are, however, also command files (re_phase.cmds, calc_2fofc_map.cmds, dens_comb.cmds, refine.cmds, refine_b.cmds) that are called from the dialog mode by clicking items. The text here should enable you to adjust this files to your needs.

Within MINIMIZE command you can instead of using MAIN standard setup of structure factor difference map calculations, call your own MACROS which do it. This allows to combine density modification procedures with (partial) model refinement as well as refinement of molecular in several crystal forms simultaneously.

A common start is presented in section "Preparing and organizing data for refinement". Section "Positional refinement" explains you how the setup a positional refinement (MINIMIZATION). Section "Applying NCS constraints" explains how to apply Non-Crystallographic-Symmetry model constraints for equivelent atoms within a single and also among several crystal forms. Section "Partial model refinement" explains how to setup a procedure for refining a partial (incomplete) molecular model by combining its density with an MIR density to get combined

structure factors (Fcalc). In section "B-value refinement" you can learn how to perform individual and overall B-value refinement. Section "Weighting the density energy term" explains how to adjust the density weights (crystallographic weight) by keeping a reasonable model geometry. Section "Interactive positional refinement" explains how to use and adjust the refinement macros invoked by clicking menu items in dialog mode. Section "Refinement against a 2Fobs-Fcalc map" encourages you to refine your entire model against a 2Fobs-Fcalc map in the first and middle stages of your structure refinement and model building.

Preparing and organizing data for refinement

First read force-field and topology parameters, coordinates and then a connectivity table (or use macro "¡>utils/protein.bond" to calculate covalent bonds using distance criteria) .

```
> <>utils/get_top_par_19_csd
> read file amodl_8.xpl coor xpl
> read file amodl.ctab ctable
```

The DEF_ALL is a string variable defined by pointing for example to the ">utils/def_top_par_19.com" file. You can replace it or modify the command file. This command procedure usually works, but for cases with unusual residues additional commands are sometimes neccessary.

```
> <DEF_ALL
```

Here we set the side chain charges to zero. This is not desired when you want to enforce formation of salt bridges.

```
> set charge sele .not atom name C CA N H O .a \
>    resi name ASP GLU LYS ARG end 0.0
```

Cell, symmetry operationas and diffraction data reading

```
> read cell
> *
> 120.5 120.5 63.15 90. 90. 120.
```

```
> read file >symm/r32.symm symm
> read file ../pla2.fobs refl init limit 0 0 -99 friedel reso 2.6 8.0
```

If you are not sure if the program has read intire data set in the specified resolution range play with H, K and L lower LIMITS and maybe turn the FRIEDEL symmetry on.

Here we define 2 maps (unit cell) for possible further calculations. Crystallographic refinement needs one map only. The second map (MAP_2FOFC) is here to be applied in electron density fitting. In the case you do refinement without a graphic display, you can omit creattion of the MAP_2FOFC map.

```
> make map 1 from 0 init -9999 grid 160 160 90 cell real
> make map 2 from 1 init -9999
```

```
> set vari MAP_2FOFC = 2
> set vari MAP_FOFC = 1
```

Remember: Number of grid points along each axis shuld be dividable with the number of the screw (hexagonal screw along Z requires the last grid number to be dividable with 6), and number of grid points along X (a) should be even. Take care that the highest prime number does not acceed 19 and that the grid size is at least 1/3 of the resolution you are currently working along each cell axis direction.

Positional refinement

Here we set crystallographic weight DENSITY SCALE and assign the MAP_FOFC as the one where the DENSITY gradients will be calculated.

```
> ener diel 2.0 dens on dens map MAP_FOFC dens scal 150.
```

MAIN uses the difference Fourier method to calculate positional derivatives. In order to avoid to frequent map generation procedures (which is the most time consuming part) I recommand to rephase the system (recalculate the difference density map) after several cycles of minimization (5, 10 or even 40 may be OK).

The generated electron density maps are rescaled to average value of 0.0 in ordr to introduce a solvent correction. The RESCALE OFF turns the map rescaling before the Fourier transformation off.

GAIN function was introduced to help the minimizer to continue with optimization after a new difference map calculation, which actually brakes the continuity of the energy function. The GAIN coefficient is an ENERGY term (product of R-VALUE difference between the two consecutive rephase steps). (The GAIN function is a preliminary solution - but it works.)

```
> mini position phase 5 sele segm name AMDL end step 80 \
>     gain 60000.0 writ 1
```

Applying NCS constraints

Non-Crystallographic-Constraints can be applied between equivalent molecular parts of in principle unlimited number of equivalent regions within a single crystal as well as among several crystal forms.

Refinement using NCS in a single crystal form

The equivalent groups first need to be DEFINED. The simplest case is when we have two molecules (MOL1 and MOL2) in the same crystal. We will impose stronger NCS constraints on main chain atoms (FORCE 20.) and slightly weeker on the side chain atoms (FORCE 5.)

```
> define init constrain ncs
> define constraint ncs sele segm name MOL1 .a atom name CA N C O H end
> define constraint ncs sele segm name MOL2 .a atom name CA N C O H end
> define constraint ncs force 20.
```

```
> define constraint ncs next
> define constraint ncs \
>             sele segm name MOL1 .a .not atom name CA N C O H end
> define constraint ncs \
>             sele segm name MOL2 .a .not atom name CA N C O H end
> define constraint ncs force 5.
```

First it makes sense to verify the definitions. Since equivalent atoms are the consecutive ones in the both selections the NCS keys should contain at least the same number of atoms:

```
> show key

 SHOW> KEY:   9 int_ncs__1 SELECTED  254 OF TOTAL 6176
 SHOW> KEY:  10 int_ncs__2 SELECTED  254 OF TOTAL 6176
```

The next more sever test (good for debugging) is to create PAIR lists and see, whether the equivalen atoms match by their atomi, residue and sequence names or not.

```
> make pair init select int_ncs__1 end select int_ncs__2 end
> write pair
```

You can never be sure that the default initial guess will work well so calculate it before the minimizer starts to work:

```
> define constraint ncs auto_guess

 RMS_FIT> PARAMETERS X-ROT: 191.373 Y-ROT": 111.292 X-ROT"":  35.276
     XYZ-TRAN:  88.258   6.726  15.660 RMS-FIT   .4520
```

The 'RMS-FIT' value should look reasonable. Values above 3. are very likely to mean that the procedure did not converge properly or that the selections are not properly defined.

After these test are successfully completed, you only need to turn the NCS term on and let the MINIMIZER to do the job as described above in the section "Positional refinement".

```
> energy ncs on
```

Refinement using NCS between two crystal forms

When refining a molecular model in two or more crystal forms means that each model should interact with its own refelction set. Converted to MAIN language it means, that each molecule should interact with its own crystal form difference Fourier map during a refinement procedure

and that all energy interactions between the molecular models, besides the NCS term, should be excluded.

Here only the differences will be described in detail, complete example files listed below are only briefly presented.

The NCS term DEFINITIONS between a single or several crystal forms look exactly the same, so there is no need to repeat them.

Lets us first define a KEY selection for each crystal form for reasons of easier handling. Molecule one is in hexagonal and molecule two in an orthorombic cell.

```
> key MOL_HEX sele segment name MOL1 end
> key MOL_ORT sele segment name MOL2 end
```

With the DEFINE CONSTRAINT INTERACTIONS we prevent atoms from cross interactions between the models in the two different crystal forms. These however requires un update of the bonding energy term lists (BOND, ANGLE, DIHEDRAL and IMPROPER terms).

```
> define init constr inter
> define constr inter sele MOL_HEX end sele  MOL_HEX end
> define constr inter sele MOL_ORT end sele  MOL_ORT end
> <DEF_ALL
```

Hereby we limit the interactions of each model with its difference density. The maps should, however, be already created before.

```
> set vari MAP_HEX = 1
> set vari MAP_ORT = 2
> define init constr map
> define constr map MAP_HEX sele MOL_HEX end
> define constr map MAP_ORT sele MOL_ORT end
```

Now we define scaling factors for each map and can start the minimizer. Since MAIN can not deal with two reflection sets simultaneously, we nedd to call a MACRO which will calculate for its crystal form its difference Fourier map.

```
> energy dens map MAP_HEX dens scale 180.
```

```
> energy dens map MAP_ORT dens scale 220.
>
> minimize sele all end step 100 writ 5 phase 15 gain 90000 \
>      macro phase_ncs.com
```

The 'do_ncs.com', 'get_data_hex.com', 'get_data_ortho.com' and 'phase_ncs.com' files can do the whole work, besides saving the results.

The 'do_ncs.com' file:

```
> <>utils/get_top_par_19_csd
> set vari MAP_HEX = 1
> set vari MAP_ORT = 2
> set vari RESOL_HEX = 3.3
> set vari RESOL_ORT = 3.3

> <get_data_hex
> read file FILE_CELL cell
> read file FILE_SYMM symm
> read file FILE_ATOM atom pdb

> make map MAP_HEX from 0 init -9999 grid  AGRID BGRID CGRID cell real

> key MOL_HEX sele all end

> <get_data_ortho
> read file FILE_CELL cell
> read file FILE_SYMM symm
> read file FILE_ATOM atom pdb

> make map MAP_ORT from 0 init -9999 grid  AGRID BGRID CGRID cell real

> key MOL_ORT sele .not MOL_HEX end

> set weight sele atom name H* end 0.0
> !set temp sele all end 30.0

> define init constr inter \
>   sele MOL_HEX end sele  MOL_HEX end
```

```
> define constr inter \
>   sele MOL_ORT end sele  MOL_ORT end

>  define init constr map
>  define constr map MAP_HEX sele MOL_HEX end
>  define constr map MAP_ORT sele MOL_ORT end

> define init constr ncs
> define constr ncs sele atom name CA N C O H .a numb segm 2 end
> define constr ncs sele atom name CA N C O H .a numb segm 4 end
> define constr ncs force 10.
>  <>utils/protein.bond
>  <DEF_ALL
>  energy dens map MAP_HEX dens scale 180.
>  energy dens map MAP_ORT dens scale 220.
> ener ncs on dens on
>  minimize sele all end step 100 writ 5 phase steps 15 gain 90000 \
>       phase macro phase_ncs.com
> return
```

The file 'get_data_hex.com' assignes file names to variables and sets the unit cell GRID sizes. The 'get_data_ortho.com' file is equal in form only with different file names so it is not shown here.

```
> set vari FILE_CELL PROJECT:p6522/model/cell_1.dat
> set vari FILE_SYMM >symm/p65_2_2.symm
> set vari FILE_REFL PROJECT:p6522/model/P622all.fob
> set vari FILE_ATOM PROJECT:p6522/model/model_26.xpl
> set vari AGRID = 80
> set vari BGRID = 80
> set vari CGRID = 162
> return
```

Here the two difference Fourier maps are being calculated, first the hexagonal one and then the ortorombic one. Reflection file, cell constants and symmetry operators are read for ech calculation separately, since MAIN is currently only able to deal with a single space group at a time. The 'phase_ncs.com' file:

```
> <get_data_hex
```

```
> read file FILE_CELL cell
> read file FILE_SYMM symm
> read file FILE_REFL reflect init resol RESOL_HEX 100. limit 0 0 0 friedel
> set vari MAP_CELL = MAP_HEX
> make map MAP_CELL zero
> make map MAP_CELL set -1 1 -9999.
>
> make map MAP_CELL atom dens \
>                   atom symm select MOL_HEX end
> make map MAP_CELL rescale
> <calc_a_fofc_map
>
> <get_data_ortho
> read file FILE_CELL cell
> read file FILE_SYMM symm
> read file FILE_REFL reflect init resol RESOL_ORT 100. limit 0 0 0 friedel
> set vari MAP_CELL = MAP_ORT
> make map MAP_CELL zero
> make map MAP_CELL set -1 1 -9999.
>
> make map MAP_CELL atom dens \
>                   atom symm select MOL_ORT end
> make map MAP_CELL rescale
> <calc_a_fofc_map
> return
```

The 'calc_a_fofc_map.com' finally calculates the 'Fobs-Fcalc' map.

```
> make map MAP_CELL set -10000 -9900 0.0
> fourier map MAP_CELL
> reflect shell 12  r-values
> refl set ampl phase fwork = fcalc * 1
> refl set ampl scale fcalc   fwork = fobs * 1 - fcalc * 1
> make map MAP_CELL zero
> make map MAP_CELL convert complex
> reflect fill-map MAP_CELL defined
> fourier map MAP_CELL back
> !make map MAP_CELL rescale
> return
```

Partial model refinement

A refinement procedure works well when (positional) derivaties are as coorect as possible. They depend upon correctnes of structure factor amplitudes and phases. Wrong phases result in meaningless positional gradients, which are in MAIN calculated from difference (Fobs-Fcalc) density maps. When model is incomplete the model structure factors are quite far away from their final form regarding both, amplitudes and phases.

The idea here is that we try to fullfil the partial model density with density information obained from other sources. These can be an MIR map, a solvent flattened map, an averaged map, a 2Fobs-Fcalc map, ... actually any kind of map which contains additional density information about the structure and is not included in the current partial structure model available. From this map the structure factors are calculated that quide the current atom models with help of a minimizer to their final place. This procedure is essentially a density modification method, but since it happens through MINIMIZE command it was placed under refinement. After refinement is completed you should look into your map MAP_FO, since it may contain substantial differences (improvements) when compared with other kind of maps (obtained by phase combination or density modification procedures).

The following procedure tries to reduce structure factor errors by combining partial model density map with MIR density map into a map used for structure factor calculation. From these Fcalc-s a difference (Fobs-Fcalc) map is calculated and later on used for model positional derivatives calculation within MIMIZATION procedure.

The follwoing procedure (dens_comb.cmds) requires 3 unit cell maps to be held in the memory (MAP_ORIG -the MIR map), a work space map (MAP_FOFC) and the (MAP_FO). It sets the variables for the whole run (dens_comb.com).

First define the energy term lists and set the crystallographic weights and some (maybe) undefined b-values.

```
> <DEF_ALL
> define impr by topo init
> set charge sele resi name ARG LYS ASP GLU .a .not -
>                  atom name C N CA O end   0.0
> set weigh sele atom name H* end 0.0
> set temp sele all end 28.0
```

Copy the density of the MAP_ORIG map (MIR) into the MAP_FO and specify the number of

shells used for the r-value calculation (scalng of Fcalc against Fobs).

```
> make map MAP_FO zero
> make map MAP_FO set -1 1 9999.
> make map MAP_FO from MAP_ORIG copy
> refl shel 15
```

Set the variables to control the density combination run: density energy scale factor (DENS_SCAL), number of minimization steps (MIN_STEP), number of minimization cycles after which a maps will be calculated, radius of the molecular model envelope (RAD_MASK), radius of the model map which will be considered to replace the current density, additional weight for the Fcalc map (WEIGHT_FC). The WEIGT_FC is an empirical factor. It should scale the model Fcalc map so that it does not disapear in the MIR map noise as well that this map would dominate over the MIR map. My criterion is to calculate the combined map so that the contours of the model fcalc map nicely fullfil the MIR density. This contour lavel is later on used in the procedure. The useful interval might be between 60 to 140% of the contour level. My advise is: try and see.

```
> set vari DENS_SCAL = 800.
> set vari MIN_STEP = 120
> set vari RE_PHASE = 15
> set vari RAD_MASK = 4.5
> set vari RAD_COMB = 2.8
> set vari WEIGHT_FC = 1.1
```

The key "model_part" includes atoms model that will be refined and included into density combination procedure, while the key "model_mask" includes actually all the atoms places at the location where the model might be. Thereby we expand the model region and include a solvent flattening cycle too.

```
> key model_part select segm name DM_* end
> key model_mask select segm name D* end
```

```
> ener dens map MAP_FO dens scal DENS_SCAL
> set vari DONE = 0
> minimize sele model_part end step MIN_STEP writ 1 \
>       phase steps RE_PHASE gain 90000 \
>       phase macro dens_comb.com
```

```
> return
```

The 'dens_comb.com' macro combines the MIR and partial model maps and calculated the difference map for the MINIMIZER.

First the model Fcalc map is calculated. Fcalc-sare scaled to Fobs in order to reduce the difference between b-values of the two maps.

```
> <re_phase.cmds
> make map MAP_FCALC zero
> make map MAP_FCALC conv complex
> refl set ampl phase fwork = fcalc * 1.
> refl set ampl phase scale fcalc fwork = fcalc * 1.
> refl fill-map MAP_FCALC resol 3.0 100.  defined
> four map MAP_FCALC back
> make map MAP_FCALC rescale
```

Create the model mask to take only the closest parts of the model Fcalc density.

```
> make map MAP_MASK from MAP_ORIG init -9999 around 9 sele model_mask end
> make map MAP_MASK atom mask RAD_COMB sele model_part .a weight 0.9  1.1 end
> make map MAP_MASK from MAP_FCALC copy
> make map MAP_MASK from MAP_FO rescale
> make map MAP_MASK scale WEIGHT_FC
```

Create the molecular envelope (mask) to take the whole (MIR) map of the molecule into the density combination and solvent flattening.

```
> make map MAP_MASK2 from MAP_ORIG init -9999 around 9 sele model_mask end
> make map MAP_MASK2 atom mask RAD_MASK sele model_mask end
> make map MAP_MASK2 + 1 from MAP_MASK2 init 9999 copy
> make map MAP_MASK2 from MAP_MASK2 + 1 fill 6
> make map MAP_MASK2 + 1 from MAP_MASK2 init 9999 copy
> make map MAP_MASK2 from MAP_MASK2 + 1 fill 6
> make map MAP_MASK2 from MAP_FO copy
```

In the case you may wish to replace the MIR density with your fcalc density uncoment the follwoing 3 lines and comment the last 2 lines reletd the density merging:

```
> !make map MAP_MASK2 atom mask RAD_COMB \
> !            sele model_part .a weight 0.9 1.1 end
> !make map MAP_MASK2 from MAP_MASK copy
> make map MAP_MASK2 from MAP_MASK merge higher ! weight 0.5 0.5
> make map MAP_MASK2 rescale
```

Calculate new structure factors (Fcalc) and the r-value to get the scaling factors and control of konvergence.

```
> make map MAP_FOFC zero
> make map MAP_FOFC conv real
> make map MAP_FOFC set -1 1 -9999.
> make map MAP_FOFC from MAP_MASK2 cell
> make map MAP_FOFC set -10000 -1000 0.0
> four map MAP_FOFC
> reflection  r-values
```

And complete with difference (Fobs -Fcalc) map calculation, prepare an Fobs map with current phases for the next density combination cycle and return from macro back into minimizer. The files 'calc_fofc_map.cmds' and 'calc_fo_map.cmds' are equivelent to the ones described elsewhere, the first calculates a Fobs-Fcalc difference map and the second calculates an Fobs map with the phases from the current structure factors.

```
> <calc_fofc_map.cmds
> <calc_fo_map.cmds

> return
```

The file re_phase.cmds calculates your model phases and the current model r-value.

```
> make map MAP_FOFC conv real
> make map MAP_FOFC zero
> make map MAP_FOFC set -10 10 -9999.
> make map MAP_FOFC atom symm  \
>            atom dens \
>            sele model_part end
> make map MAP_FOFC rescale
> make map MAP_FOFC set -100000 -100 0.
```

```
> fourier map MAP_FOFC
> reflection r-values
> return
```

B-value refinement

For B-value optimizations GROUPS have to be DEFINED. The smallest group is an atom and the largest are all atoms. Between these two extrems any number of groups composed from any combination of atoms is valid. Currently GROUPS can be defined through SINGLE, ATOM, RESIDUE and SEGMENT command words. The GROUP SINGLE command defines single atoms group - each selected atom is a separate group. The GROUP ATOM defines all selected atoms as a group. The GROUP RESIDUE command DEFINES each selected residue as a group and the GROUP SEGMENT does the same on the segment level.

```
> define init group
> define group atom select all end
> define group single select all end
```

The reciprocal space B value is zero during B-value refinement. It can not be modified.

Set weights (SCALE, ANGLE, BOND, GROUP) and MINIMAL and MAXIMAL atomic B-value for refinement with an ENERGY command and then start MINIMIZATION.

The B-VALUE SCALE pulls individual atomic B-VALUES against the difference map (Fobs - Fcalc) values, while the BOND, ANGLE and GROUP constraints try to make them equal. BOND and ANGLE forces act between the groups. They pull group B-values of covalently attached atoms with the specified harmonic constraint against their average. GROUP force acts within each groups. It pulls atomic B-values towards the group average. When GROUP constraint is a negative value, then GROUP constraint is 'hard': B-values and their derivatives for atoms within each group are averaged. When it is positive, then the constraints are soft: atomic B-values within a group are pulled against the average group value with a harmonic force. By default GROUP restraints are hard (-1).

An equilibrated assignment of B-VALUE scales is essential for a succesfull refinement. Values as BOND 1.0 for example result in an equal B-value for all atoms. If you have already refined a structure before then first check the B-VALUES sigmas with a SHOW TEMPERATURE command.

```
> ener b-val scal 80. b-val bond 0.003 b-val angle 0.002 \
>      b-val group -1. b-val min 5. b-val max 80.
```

Command below submits 10 cycles of B-VALUE refinement. Energies will be WRITTEN each iteration step.

```
> mini b-val   sele segm name AMDL end step 10 writ 1
> sh temp sele weight 0.9 1.1 end

> write over file amodl_result.xpl sele segm name AMDL end atom xpl
```

Pay attention also to the "a_real_case" example.

Overall B-value refinement

The parameter you have to deal with in addition to the above specified ones is the number of reflection shells. This defines whether you try to refine the overall B value or you want to make the fine tuning of the atomic B values.

The number of reflection shells is supposed to be set to 1 to get around of the bias introduced with the shell independent scaling.

```
> reflect shell 1
```

Another parameter is the number of groups within the system. You are however not obliged to use only one or more large groups, you can still work with individual atomic B-values combined with any combination of BOND, ANGLE and GROUP constraints.

!WARNING: Overall B-value refinement will probably not work if !atomic B-values inside each group before the beginning of the !procedure are not equal.

Interactive positional refinement

When inserting new residues or rebuilding them it is not enough only to recalculate the density (RE_PHASE - re_phase.cmds). In fact, this only slightly effects the phases. The model should

be refined as the whole against the Fobs in order to spread the newly introduces geometrical change over the whole molecule (and thereby phases).

A file "refine.cmds" is called by clicking the REFINE menu item.

First the topology and energy lists are updated (for the quite common case when we forgett to do it manually) after new or deleted resdues ...). The L chirality is in addition enforced by the DEFINE IMPR BY TOPOLOGY with no CHECK option.

```
> set temp sele .not atom name H* end 30.
> set weigh sele atom name H* end 0.0
> <DEF_ALL
> define impr by topo init
```

First use only the energy terms that you realy may need and set a scale factor for the density energy.

```
> ener all off bond on angle on dihe on impr on vdw on elec on
> ener  dens on dens map MAP_FOFC dens scal 240.

> mini phase 3 sele segm name CB* end sele segm name CB* end  gain 90000. \
              step 20 writ 1
> mini writ 10 step 80 phase 0
> ener dens map MAP_2FOFC
> return
```

Weighting the density energy term

You can check the model geometry deviations from ideality with:

```
> anal resi init
> anal param bond sele all end
> wr anal
```

If the deviations are too high, reduce the weight to a smaller number. Check the angle term as well.

```
> energy density scale 100.
```

Refinement against a 2Fobs-Fcalc map

When your model crystalographic R-value is still high in high 20-ties or higher, a refinement of your complete model against a 2Fobs-Fcalc map or a similar (MIR, phased combined or density modified) can significantly improve your R-value in a relatively short time. So try it.

```
> ener all off bond on angle on dihe on impr on vdw on elec on
> ener  dens on dens map MAP_2FOFC dens scal 240.
> mini phase 0 sele segm name CB* end sele segm name CB* end  step 500 writ 10
> return
```

Chapter 10

Topology and parameter files

In this chapter you can learn how to use available topology and force field parameter files and how to create your own ones.

Topology files contain atom type (CLASS) and residue(s) definitions, while parameter files include force field terms.

Using MAIN topology files enable you to build molecular model from scratch or modify atomic composition of your molecular model and later perform ENERGY calculations

X-PLOR topology and parameters can be used only for ENERGY calculations. You can not use them to build molecular models from scratch or modify their topology. You can however any time convert X-PLOR files into MAIN files. See the section "Converting XPLOR topology files to MAIN form".

Reading MAIN topology and parameter files

The simplest way is to call a ">utils/get_top_par..." command macro (with "@" or "¡") which reads appropriate topology and parameter files and specifies the macro name (the character variable DEF_ALL) invoked by clicking DEFINE or DEFINE_A to redefine atoms types (CLASSES), atomic CHARGES, BOND, ANGLE, DIHEDRAL and IMPROPER angle lists. If everything is fine with your case and you do not want to know how topology assignemnt works, you can stop here.

The ">utils/get_top_par_19_csd.com" macro reads the Engh & Huber topology and parameter

files for amino acid residues including necessary C- and N- termini definitions plus a solvent (H2O) molecule definition, and defines the value of the variable DEF_ALL.

```
> read file >top/top_19_csd.main top main
> read file >top/par_19_csd.main par init
> set vari DEF_ALL = >utils/def_top_par_19.com
> return
```

Inspecting and saving topology library records - write

Now you can see what the program has read in by typing

```
> write topology
```

WRITE TOPOLOGY writes the whole topology library. You can limit the written information to a residue by specifying it:

```
> write topology residue ALA
```

The output now is shorter, it lists all known CLASSes and then only the information related to the alanine residue topology. Each residue record starts with the word "residue" followed by the residue abbreviation code "ALA" (up to 4 characters long). Currently each residue can consist only of a single "group".

```
residue ALA
group
atom N clas=NH1 charge=-.350 coor .000 .000 .000
atom CA clas=CH1E charge=.100 coor .539 -1.353 -.060
atom H clas=H charge=.250 coor .617 .761 .021
atom C clas=C charge=.550 coor 2.063 -1.335 -.081
atom O clas=O charge=-.550 coor 2.682 -.272 -.051
atom CB clas=CH3E charge=.000 coor .003 -2.077 -1.287
```

The "atom" records include ATOM NAME (up to 4 characters are read). Each atom name must be unique within a residue in order to allow unique atom differentiation. Number of atom

records per residue is not limited, however, the total sum of all atoms present in the topology library is limited.

The "clas" identifier is atom type assigned to the atom. It is used to correctly assign forcefield parameters to its bonding and nonbonding terms.

Atomic partial "charge" can be any real number (take care when inventing your own.)

The "coor-dinates" data are used only when MAIN starts to build a new segment. These numbers are actual coordinates of atoms at the time of creation of this topology library, specified relatively to the first atom in the residue.

```
bond N -C bond CA N bond H N bond C CA bond O C
bond CB CA
```

The "bond" records list all the covalent bonds within the residue. MAIN topology file in contrary to the X-PLOR convention allows assignment of bonds also to the atoms of a previous residue based on a convetion that the '-' sign relates to the previous residue. The bond records are usually used only for model building, while the energy bond lists are created from the connectivity table automatically.

```
dihe -C N CA C dihe -N -CA -C N
dihe -CA -C N CA

impr CA N C CB impr -C -CA N -O impr N -C CA H
```

The "dihedral" and "improper" angle records list DIHEDRAL and IMPROPER angle definitions needed for energy calculations. Again, as by bonds, the '-' sign means that the related atoms are to be found in a previous residue. So it is possible to avoid use of XPLOR "patches" for the peptide chain when generating all bonding lists.

```
inte N -C 1.329 -CA 116.214 -O -180.000
inte CA N 1.458 -C 121.609 -CA 180.000
inte H N .980 -C 119.139 -CA .000
inte C CA 1.525 N 111.063 -C 180.000
inte O C 1.231 CA 120.839 N -.059
inte CB CA 1.522 N 110.147 -C 57.525
```

The "internal" coordinate records include distance of the record atom to its neighbor, angle to the neighbor of the neighbor and an dihedral angle to one more atom behind. These records are used to attached residues to a previously defined residue.

```
end
```

Each residue definition ends with a record consisting of "end".

Inspecting topology library records - show

A different way of inspecting a topology library is by using SHOW TOPO commands. For the SHOW commands the rule, what can be written can be also READ, does not apply.

```
> show topo resi *
```

The SHOW TOPO alone gives you the number of entries in the topology library, togethre with RESIDUE * lists all short names of available residues. The '*' means any residue name will match, 'H*' means any residue starting with 'H' will match and 'ALA' mean only ALA residue name matches.

Complete information about a topology library residue can be obtained with the following command

```
> show topology resi ALA atom * all bond * dihe * impr * inte *
```

where the '*' charcter stands for any atom. The selection of matching atoms can be reduced by supplying characters from atom names. Any combination of items (BOND, DIHEDRAl, IMPROPER, ATOM, INTERNAL) is acceptable. When only instepcting a topology residue description you may prefer the SHOW TOPOLOGY commands, since outpu is formated. The following output is result of the above command:

```
SHOW> TOTAL NUMBER OF TOPOLOGY RESIDUES KEPT   28
    1 RESIDUE ALA  ATOMS:   6 KIND: r
ATOMS
N    NH1    -.3500    .000E+00    .000E+00    .000E+00
```

```
CA   CH1E    .1000    .539E+00   -.135E+01   -.600E-01
H    H       .2500    .617E+00    .761E+00    .210E-01
C    C       .5500    .206E+01   -.134E+01   -.810E-01
O    O      -.5500    .268E+01   -.272E+00   -.510E-01
CB   CH3E    .0000    .300E-02   -.208E+01   -.129E+01
 SUM OF CHARGES .0
BONDS: 6
N    -C
CA   N
H    N
C    CA
O    C
CB   CA
 DIHEDRALS: 3
-C   N    CA   C
-N   -CA  -C   N
-CA  -C   N    CA
 IMPROPERS: 3
CA   N    C    CB
-C   -CA  N    -O
N    -C   CA   H
INTERNALS
 N    -C   -CA  -O    1.3290  116.21 -180.00
 CA   N    -C   -CA   1.4580  121.61  180.00
 H    N    -C   -CA    .9800  119.14     .00
 C    CA   N    -C    1.5250  111.06  180.00
 O    C    CA   N     1.2310  120.84    -.06
 CB   CA   N    -C    1.5220  110.15   57.53
```

The command SHOW TOPOLOGY CLASS lists you all know atom types.

Inspecting and saving force field parameters - write and show

Complete current force field parameters can be saved to a file using a WRITE command:

```
> write parameters
```

```
bond    C       O    1480.000    1.231
```

```
 angle    CH2E    C5W     CR1E  863.744  126.900
 dihedral    CH1E    C       N       CH1E  1250.00    2   180.00
 improper    C       X       X       C       75.00    0      .00
 nonbonded    OH1      .159    2.851     .159    2.851
```

This is only a selection of written parameters. Each record starts with an identifier (BOND, ANGLE, DIHEDRAL, IMPROPER or NONBONDED) that is followed by the required number of atom CLASSES forming a term, force constant and equilibrium geometry. DIHEDRAL and IMPROPER records have an insertion, after the force constant periodicity of the angle is specified. 'X' atom class means any atom class - this only applies for IMPROPER and DIHEDRAL records.

The NONBONDED terms specify van der Waals interaction energies. The first number is interaction energy at the optimal distance (second number) for two equal atoms. From these two numbers interaction constants for VdW interaction energies are calculated for each possible pair of specified atom CLASSES. NONBONDED interaction differentiate between 1-4 interactions and the more distant ones, therefore also two pairs of numbers have to be specified.

```
> show param
 SHOW> BONDS  71   ANGLES 139   DIHEDRALS  43   IMPROPERS  64
```

gives you number of force field records. With

```
> show param bond C O
 bond C O 1480.000 1.231
```

you can retrieve a single force field term. No wild characters for atom CLASS matches are accepted.

Using available MAIN topology files

After topology library and parameter files have been read, their information needs to be merged into description of a molecular model. Once this is done, you can start ENERGY calculations including POSITIONAL MINIMIZATION of your model.

The simplest way is to call the DEF_ALL command file (defined in get_top_par_19_csd.com) by clicking a menu item DEFINE or DEFINE_A. No error messages should appear, besides the

ones related to N- and C-terminal OT and HT atoms. When it si not so you must find out the reasons and act. You can not expect that the system will behave under MINIMIZATION nicely if it has some undefined force constants or even a couple of atom CLASSES undefined.

First the ">utils/def_top_par_19.com" will be described, and then we shall check some quite frequent problems.

```
define class charge by topo sele .not t_anchor .or atom name X* end
```

The first DEFINE command assigns each atom (besides anchors and dummy atoms) its CLASS from topology library entries. For each selected atom program tries to find a match by its RESIDUE and ATOM NAME. Only not found residue entries are reported.

```
set class sele atom name S* .and. by bond atom name S* end S
```

SET CLASS selects only those sulphur atoms included in a disulfide bond arrangement and SETS their CLASS explicitly. So disulfide bridges can be formed implicitly (by a distance criterion) and no explicit knowledge about exact disulfide bond arrangement is required. WARNING: This is a more comfort way, but if you forget to calculate disulfide bonds, VdW forces may pull SG atoms apart and the next time, when you will try to calculate the disilfides they will not be created.

```
defin class charge sele by resi atom name HT* end by topo resi NTER
defin class charge sele by resi atom name OT* end by topo resi CTER
```

It is assumed that the hydrogen and oxygen atoms at both chain termini have 'T' as the second character of their atom names. These make possible to implicitly assign NTER-minal and CTER-minal atom definitions with no need to create different segments or chains or even read chains separately as in XPLOR.

After atom CLASS assignment has been successfully completed, you may proceed to start creating bonding energy lists. If there are still problems see the section "Frequent topology and parameter assignment problems" below.

```
define bond angle init select .not t_anchor end by auto
```

The BOND and ANGLE lists are created BY AUTOMATIC procedure which builds the lists by using covalent bond connections present in the connectivity table (CTABLE). (What you see is what you get, which means if you will break a covalent bond and redefine the lists, it will not be taken into an energy calculation neither as BOND or as a part of an ANGLE term.)

```
define dihe by topo init select .not t_anchor end
define impr by topo init select .not t_anchor end check
```

DIHEDRAL and IMPROPER angles definitions are usually part of the topology library residue entries. Mathematically there is no difference between the two terms. DIHEDRALS are usually conformational angles of 4 atoms (1-2-3-4) describing rotation about the middle covalent bond (2-3). IMPROPERS are defined by 4 atoms as well with the difference that the 4 atoms are not necessarily covalently attached in in the pattern 1-2-3-4.

IMPROPERS usually define planarity or chirality of a group. Command word CHECK effects only chiral impropers in the way that it preserves the current chirality. This allows building L or D amino-acids with no need for applying a patch DtoL or vice versa, similar as in XPLOR. If you want to enforce chirality of L amino-acids then remove the CHECK.

Additional improper or dihedral angle definitions that are not included in the residue name as for example the C- and N- termini needs to be explicitly defined as follows:

```
define impr dihe by topo select by resi atom name HT* end resi NTER
define impr dihe by topo select by resi atom name OT* end resi CTER
return
```

The difference bewteen these commands and the higher placed general topology library calls is that here RESIDUE NAMES are not taken from the actually selected atoms, but are explicitly enforced: each selected residue is checked against the specified NTER and CTER DIHEDALS and IMPROPERS lists, and for each match in all four atom names additional DIHEDALS and IMPROPERS is created. When an N-terminal residue is a proline use NTPR insetad of NTER.

XPLOR uses the word "nil" to patch a single residue, in MAIN you simply don't need to specify a SYMBOL or NEIGHBOR one character identifier(s). You select a residue and explicitly call a particular topology library entry as shown above for the N- and C- terminal improper and dihedral angle definitions.

Any additional entry in the any of the bonding energy lists can be also explicitly created or deleted. See the section "Modifying and creating new topology lists".

Frequent topology and parameter assignment problems

Wrong residue or atom names - rename

In cases when a residue name from your structure is not found in topology library try to find out if your residue only needs a different name by comparing atoms from the residue with the appropriate topology library residue entry. If the two describe the same structure, you only have to RENAME your residue. The MAIN Engh & Huber topology library has for example no HIS residue, but a HISH:

```
> rename residue HISH select resi name HIS end
```

If you are not lucky enough continue with the section "Modifying and creating new topology library entries".

When an atom does not find a match in equivalent topology library residue you should RENAME it. A usual case is the CD1 atom of isoleucine:

```
> rename atom CD select residue name ILE .and. atom name CD1 end
```

Missing atoms

The most common things are problems with N- and C- termini, however, this is presented in the chapter "Model building". Here you need to know only that you have to create other missing atoms and finally fulfill the model with its termini atoms.

These commands need to be explicitly typed:

```
MAIN> build
BUILD> select ... end
BUILD> fill
BUILD> exit
MAIN> calculate coordinates append
```

This procedure finds all the missing atoms and builds them into the model by keeping the known atoms where they were. WARNING: The atoms that are in your model, but do not match your topology library residue atom list, are deleted. Therefore you are advised to do other checks first. This happens to N- and C- terminal 'HT*' and 'OT*' atoms as well.

Using available X-PLOR topology files

XPLOR topology files are not in the form that allows MAIN to build and modify them, You can apply them for energy calculations only. XPLOR topology libary residue has no links or definitions of lists that refer to some other residue. Therefore it uses patches (PRESIDUES) to define the connectivities. The patches have in front if each atom name a symbol ('-', '+', '1', '2') which refers to a selection. You can use them in MAIN too, for example the peptide bond patch can be invoked through the "NEIGHBOR - +" command, which applies the - and + for each pair of consecutive residues through the whole selection.

```
> define dihed impr select sequ 1 99 end residue PEPT neighbor - +
```

A more general case is when you want to apply patches to any pair of residues:

```
> define dihedral residue DISU symbol 1 select sequence 1 end \
>         symbol 2 select sequence 122 end
```

Once you can do energy calculations with an XPLOR residue description it is straight forward to create MAIN topoology library residue(s). See section "Converting XPLOR topology files to MAIN form"

Converting XPLOR topology files to MAIN form

As you may have alreay notice above, there are several differences between the contents of MAIN and XPLOR topology library residue entries, namely MAIN ATOM description includes COORDINATES, MAIN INTERNAl coordinate description has defined referenced distances and angles and referenced atoms in MAIN could be found in a previous residues (using the '-'atom name prefix).

If you only READ XPLOR topology libaries and WRITE them in the MAIN format, all these for MAIN important data will be missing or undefined. In order to allow MAIN to fulfill the libraries you must provide the missing data.

So read in a coordinate file of the residue, CALCULATE its covalent bonds, MAKE the internal coordinate table (Z-matrix table - ZTABLE), CALCULATE INTERNAL cordinates and DEFINE all necessary energy parameters for it (atomic CLASS, CHARGE and all DIHEDRALS

and IMPROPERS). While creating new residues MAIN accesses these data fields and transforms them into a topology library description.

Now let us define a new alanine residue (ALA) based on XPLOR topology library description:

First read a coordinate file with the ALA residue. The ALA residue should not be the first one in the chain if you want to be able to attached it to some other residues, create the tables and CALCULATE internal coorinates:

```
> read file ala.pdb coordinates pdb
> calculate bond
> make ztable from ctable
> calculate internal
```

Then read the XPLOR files and DEFINE atom CLASSES, CHARGES and energy lists. (No explicit SELECTION is specified so the default SELECT ALL END is taken.)

```
> read file xplor/toppar/toph19x.pro topo xpl init
> read file xplor/toppar/param19x.pro para xpl init
> defin class charge by topo
> defin bond angle init by auto
> defin dihe improper init by topo
> defin dihe impr by topo resi PEPT sele all end neigh - +
```

Now perform at leats the minimum check of correctness of the above definitions by an energy calculation:

```
> energy select all end
```

If the energy term values appear OK create the topology library residue and write it to a file. Topology libary is initialized because of two reasons. First, because we usually only want to write the newly created residues and second, because a residue, which is on the list but not the last one, can not be created twice. (A way around this is to change the RESIDUE NAME by a RENAME command.)

```
> topo init
> topo append sele resi name ALA end
> write file top_ala.main topology
```

Following the just described procedure not only one residue, but also whole XPLOR libraries could be converted.

Modifying and creating new topology library entries

This is an advanced topic, so do not start here. Here it is assumed that you have basic knowledge about the ENERGY calculations, that you can perform them, and off course, that you know how topology library entries and force field parameters are organized. You should also be able to perform interactive model manipulation and building.

You are strongly discuraged to create a topology library entry while working with your macromolecule. Start MAIN and only read in the molecule with the yet undefined topology residue together with the topology and parameter libraries you want to expand.

Modifying an existing residue: norleucine

So let us first create a norleucine, a residue similar in topology to methionine, which has the sulphur atom replaced with a carbon. So load the menus and topology and parameter files, initialize display and enter the dial mode:

```
> <>cmds/load_depp_page
> <>utils/get_top_par_19_csd
> image init dial
```

Set the START flag, choose a name for the new segment and a sequence root (click START, SEGMENT and SEQUENCE in any order - page 9) and then create a two residue segment by clicking RESIDUE?? and responding with ALA MET. It is important to start with a residue before MET because only so it will be possible to build chains with the newly created toology library residue entry (as already explained above). Define all energy lists and atom CLASSES and CHARGES by clicking DEFINE. Display all atom NAMES and CLASSES to have their visual control:

```
> image set atom name atom class
> image dial
```

Go to page 1, click the SG atom, rename the atom to CD (click REN_ATOM and answer with SD) rename the residue from MET to NOR (click REN_RESI and respond with NOR), go to page 2 to RE_DRAW the image and then to page 4 to and modify the atom clas to CH2E (click INQ_CLASS and respond with CH2E). In the case that you have invented new atom CLASS you should rad it in too as descrived below for the invented force field parameter(s) and finally insert these new CLASS records on the top of newly created topology file with a text editor.

Now the data for the new residue are ready so click two norleucine atoms, select this residue into the key 'active' (go to page 9 and click ACT_2RES) and finally create the new entry (go to page 4 and click on MAKE_TOP). Check it with

```
> write topo resi NOR
```

Pay attention to this record which is shows that the bond length CG-CD and CD-CE are unusually long for the sp3 carbon bond length.

```
 inte CG CD 1.803 CB 112.956 CA -160.979
 inte CE CD 1.791 CD 100.936 CB -99.220
```

But now the topology is ready to be used so you can redefine the energy term lists, and optimize geometry (go to page 9 and click DEFINE). The error message

```
 DEF_ANGL> WARNING: UNDEFINED ANGL   13*CH2E*   14*CH2E*   15*CH3E*
```

will tell you that a parameter for angle CH2E-CH2E-CH3E is missing. Before doing any further steps the missing parameters need to be invented: So look inot force field parameter lists and find the most similar one:

```
> wr param
```

I have chosen the angle CH1E-CH2E-CH3E. This is not entirely correct but it will work good enough I suppose. Now read it MAIN:

```
> read param
 angle    CH2E    CH2E    CH3E  440.686  113.800
 end_of_file
```

WARNING: If you try to exit reading with ctrl-D MAIN get lost. Type in the string end_of_file to get the MIAN prompt back. Now you can press DEFINE and no error messages will appear so the norleucine geometry can be optimized now. At this point it makes sense to use only the bonding terms (BOND, ANGLES, DIHEDRALS and IMPROPERS) so turn the VDW and ELECTROSTATIC term before off, and then MINIMIZE it (click ENE_VDW and ENE_ELEC and then MINIMIZE).

Correct the internal geometry of the residue by recalculating the interatomic distances, angles and dihedrals from the current coordinates and then either click MAKE_TOP in the dialog mode again or type the commad explicitly:

```
> calc internal
> topology append select active end
```

What remains is to save the new residue entry to a file.

```
> write file top_19_nor.main topology resid NOR
```

Do not forget that also an angle parameter term was missing, so create a file for the missing force field parameter term(s) too, and read it next time after the standard parameter files. (The results of this procedure are the files '>top/top_19_nor.main and '>top/par_19_nor.main'.)

Chapter 11

A real case

For the most cases (1 protein molecule per asymmetric - molecular replacement or in the final stages of model building) the command file (read.com) and the (cmds) menu macros present in the >examples/a_real_case will be enough to start working. Off course they should be trimmed to match your case. Mostly it should be enough to modify some file and segment names.

The startup file (read.com) calc the first macro (¡>utils/get_top_par_19_csd) to readin the topology and parameter files (csd is the Engh & Huber force field), atomic coordinates file (amdl_2.xpl) and a corresponding covalent bond list in the form of a connectivity table.

```
> <>utils/get_top_par_19_csd
> read file >show/amdl/amodl_2.xpl coor xpl
> read file >show/amdl/amodl.ctab ctable
```

If you are running this for the first time, then the connectivity table (CTABLE) does not exist yet so replace the READ CTABLE line with

```
> <>utils/protein.bond
```

This macro calculates the covalent bond pattern (CTABLE) of a protein. See chapter "Utility macros" section "Protein bonds".

Next a DEFINE macro should be called, which defines atom types (CLASSES) and CHARGES and generates the lists of BOND, ANGLES, DIHEDRALS and IMPROPERS energy terms. The DEF_ALL is a file variable that is defined in the "get_top_par" procedure. The side chain

charges of charged residues as (ASP, GLU, LYS and ARG) are here then SET to 0.0. Each atom is definied as a single GROUP for a possible later atomic B value refinement.

```
> <DEF_ALL
> set charge sele .not atom name C CA N H O .a \
>     resi name ASP GLU LYS ARG end 0.0
> define init group
```

The cell constants, symmetry group and reflection (FOBS) file are read.

```
> read cell
> *
> 120.5 120.5 63.15 90. 90. 120.
```

```
> read file >symm/r32.symm symm
> read file >show/amdl/pla2.fobs refl init limit 0 0 -99 friedel reso 3.0 8.0
```

Here map arrays in the size of unit cell are organized. Take care to choose the correct (precise enough with regards to the resolution of your data you are currently using) grid size for the later Fourier transfromations.

```
> make map 1 from 0 init -9999 grid 120 120 64 cell real
> make map 2 from 1 init -9999
> copy map 2 3
```

The variables MAP_2FOFC and MAP_FOFC are used by the macros (re_phase.cmds, calc_2fofc_map.cmds, refine.cmds ...). The initial SCALE for the DENSITY ENERGY is set to 200, which pulls your model into the density and also start to introduce severe geometry distortions when conformations are not reasonable.

```
> set vari MAP_2FOFC = 1
> set vari MAP_FOFC = 2
> ener all on dens on dens map MAP_2FOFC dens scal 200.
```

In these two steps the structura factors (phases) are calculated from the current model and on their basis the first 2Fobs - Fcalc map.

```
> <re_phase.cmds
> <calc_2fofc_map.cmds
```

Now it is time to open the graphic interface (image init) and go to the last model position and view (model.view is created by invoking the save_file.cmds by clicking the SAVE menu item).

```
> image init cent calc
> <model.view
```

At this point you may wish to get the Ramachandran plot on the screen. If so remove the exclamation mark.

```
> !<>utils/rama
```

Here we display the unit cells and the chain trace of the current model and put all image objects into the image background.

```
> <>utils/show.cell
> delete atom sele cell_atom end
> ima sele atom name CA C N end col 1 bond
> ima from erase
```

It remains to contour the 2Fobs-Fcalc map around the current center, to load the additional menus, to upgrade the molecular model image and you are ready to go (IMAGE DIAL).

```
> image col 245 map MAP_2FOFC dens 0.8  12 12 12
> <>cmds/load_depp_page
> <re_image.cmds
> return
```

Do not forget to adjust the macros invoked by clicking the menu items RE_PHASE(re_phase.cmds), RE_IMAGE (re_image.cmds), REFINE (refine.cmds and maybe refine_b.cmds), SYMMETRY (symmetry.cmd), SAVE (save_file.cmds) and GET_SKEL (get_skel.cmds). The macros are presented here only briefly. For more details, see the part "Dialog mode accessories".

In the file 're_phase.cmds' (RE_PHASE) choose your own selection of atoms you are using for phasing.

```
> make map MAP_FOFC conv real
> make map MAP_FOFC zero
> make map MAP_FOFC set -10000000 1000000 -9999.
> make map MAP_FOFC atom symm \
>            atom dens \
>            sele segm name AMDL end
> make map MAP_FOFC rescale
> make map MAP_FOFC set -100000 -100 0.
>
> fourier map MAP_FOFC
> reflection r-values
>
> return
```

The file 're_image.cmds' updates your image (RE_IMAGE). Here atoms with zero occupancy (weight) the attached hydrogen atoms are colored green, and the weighted atoms (occupancy one) together with the attached atoms are colored yellow. Change at least the segment names!

```
> image erase
> delet atom sele segm name #* end
> key image sele .not all end
> !image sele segm name AMDL .a .not atom name H* end col 160 bond
> set col sele segm name AMDL .a weight 0.9 1.1 end col 160
> set col sele segm name AMDL .a weight -0.9 0.1 end col 190
> set col sele atom name H* .a by bond weight 0.9 1.1 end col 160
>
> image sele segm name AMDL end set bond
> return
```

The files 'refine.cmds' and 'refine_b.cmds' perform structure refinement, so you may need to change the scaling of the density energy terms. When you are changing topology of molecular model you may wish to redefine the energy lists and set some B values. By increasing the DENSITY term, you should not forget to accordingly increase the GAIN coefficient.

```
> !<DEF_ALL
> !defin impr init by topo
> !set charge sele .not atom name CA N C O H .a  resi name AGR LYS ASP GLU end 0.0
> !set temp sele segm name AMDL .and. temp -0.1 0.1 .aweight 0.9 1.1 end 20.
>
```

```
> ener pair off
> ener  dens on dens map MAP_FOFC dens scal 100.
> mini position phase 19 sele segm name AMDL end step 20 \
>      gain 400000.0 writ 1
> mini phase 0
>
> return
```

Refinement of B-values is presented here separately from the above shown positional refinement procedure, although you may includ it in the 'refine.cmds' file as well. The major reason for doing this lies in CPU time use. For B-value refinement a new difference map and thereby new structure factors are calculated each minimization step, therefore it does not really matter whether the procedure is invoked through menu items or from the command line.

```
> ener b-val scal 50. b-val bond 0.005 b-val angle 0.008
> mini b-val   sele segm name AMDL end step 30 writ 1
> show temp sele weight 0.9 1.1 end
> mini posit
> return
```

Here we SAVE current model coordinates and view (save_file.cmds). When new residues are appended to the structure then connectivity table could be saved as well.

```
> write over file amodl_save.xpl sele segm name AMDL end coor xpl
> save over file model.view view
> return
```

Chapter 12

Generation of solvent molecules

In the final stages of a protein structure refinement well ordered solvent molecules are placed around the protein. Their placement and confirmation of their positions require in a crystallographic refinement procedure much time consuming manual work. In order to avoid it the below described procedure was developed. It

- removes the already existing water molecules placed at too low electron density (2Fobs-Fcalc),
- removes ones with too high temperature factor,
- inserts oxygen atoms at positions of peaks found in the difference electron density map (Fobs-Fcalc),
- removes peaks placed at too low electron density in 2Fobs-Fcalc) maps,
- removes all peaks that are closer than 1.5Å to any already present atom,
- removes all peaks that are further than 5.0Å to any protein atom,
- removes all peaks that are closer than 5.0Å to any dummy protein atom,
- removes all by crystal symmetry related peaks,
- gives a list of possibly wrongly placed atoms,
- removes all peaks that have no hydrogen donor or acceptor atom in their vicinity.

The following command procedure files are required to check old and new insert solvent molecules:

- water.com
- remove_symm.com
- atom_neigh.com

WATER.COM

At the top of the input file variables (file names and parameters) are specified to adjust procedure to each paticular case:

File name variables:

```
MAIN> set vari ATOM_COOR /b/dturk/cathb/p21/hucathb_5t_1.xpl
MAIN> set vari 2FO-FC_MAP /tmp/dturk/dens_2.2.xmap
MAIN> set vari FO-FC_MAP /tmp/dturk/diff_2.2.xmap
MAIN> set vari WATER_OUT water.xpl
MAIN> set vari SYMMETRY /b/stef/dus/main/symm/p21.symm
```

Cutoffs:

```
MAIN> set vari TOO_CLOSE 1.5
MAIN> set vari TOO_FAR 4.5
MAIN> set vari PAIR_CUT 1.5
MAIN> set vari PAIR_LIST 2.1

MAIN> set vari 2FO-FC_CUT 0.8
MAIN> set vari FO-FC_CUT 2.2
MAIN> set vari FO-FC_ION 0.5
MAIN> set vari TEMP_CUT 80.0
```

Grid size +40% to enable proper pick segmentation and the starting sequence number for the

```
MAIN> set vari GRID_SIZE  1.1
MAIN> set vari START_WATER 500
```

Read the coordinate file in X-PLOR format:

```
MAIN> read file ATOM_COOR coordinates  xplor
```

Assign to keys old water molecules, protein nonhydrogen atoms, all nonhydrogen atoms and all dummy nonhydrogen protein atoms:

```
MAIN> key wat select segment name WAT end
MAIN> key protein select .not. segment name WAT .and. .not. atom name H* end
MAIN> key orig select .not. atom name H* end
MAIN> key dummy select weight -0.1 0.1 .and. protein end
MAIN> key old select all end
```

Read both electron density files plus symmetry operations:

```
MAIN> read file 2FO-FC_MAP map xplor
MAIN> read file FO-FC_MAP  map xplor
MAIN> read file SYMMETRY symmetry
```

Remove all water molecules which oxygens are placed at positions with lower electron density then 0.8:

```
MAIN> key dw select map 1 -20.0 0.8 from wat .and. atom name O* end
MAIN> write select dw end coor xpl
```

Remove all water molecules with oxygen temperature factors higher than TEMP_CUT:

```
MAIN> key dw select temp TEMP_CUT 1000.0 .and. atom name O* .and. wat end
MAIN> write select dw end coor xpl
MAIN> delete atom select by resi dw end
```

Writes the solvent molecules with still positive difference electron density to the file ion.list. Their electron number seems to be too small for their position.

```
MAIN> write file ion.list -
MAIN>   select .not. atom name H* .and. map 2 FO-FC_ION 100. -
MAIN>   from wat end coordinates  xplor
```

Finds peaks in the difference density map (2) higher than FO-FC_CUT and write the found peaks to the file PICK.DAT The area of grid points cube (300 300 300) should be larger than the area where the molecules are placed and the molecules should lie inside the difference density map (2). The cut command word cut tells the program not to extrapolate the density into the equivalent regions outside the map.

```
MAIN> make poin map 2 300 300 300 extreme FO-FC_CUT 20.0 cut
MAIN> show point dens
MAIN> write file pick.dat poin dens
```

Make atom from points (peaks) and assign the newly created peaks to the key xxx

```
MAIN> make atom poin
MAIN> key xxx select .not old end
MAIN> show key
```

Delete all the peaks that are placed at positions where the electron density map (1) ,2Fobs-Fcalc, is lower than 2FO-FC_CUT sigma. -300.0 sigma is assumed to be much lower than any reasonable lowest density value.

```
MAIN> delete atom select map 1 -300. 2FO-FC_CUT from xxx end
```

Delete peaks closer than TOO_CLOSE to any "orig" atom, peaks further than TOO_FAR from any "protein" atom and peaks closer than TOO_FAR from any dummy protein atom:

```
MAIN> delete atom select around key orig dist TOO_CLOSE from xxx end
MAIN> delete atom select .not. around key protein -
MAIN>   dist TOO_FAR from xxx .and. xxx end
MAIN> delete atom select around key dummy dist TOO_FAR from xxx end
```

Since the newly creted atoms (peaks) were created as dummy atoms (X) they had to be renamed to oxygen(OH2) of water residues(H2O) in order to allow peak segmentation traced through covalent bonding networks and reduce each network into a singe atom placed at its arithmetic middle point position:

```
MAIN> rena select xxx end atom OH2
```

```
MAIN> calc bond distance GRID_SIZE select xxx end
MAIN> make segm select xxx end
MAIN> make atom merge resi select xxx end
MAIN> show key
```

Rename the water clusters into a single segment WAT2 and their residue names to H2O:

```
MAIN> rena select xxx end resi H2O
MAIN> rena select xxx end seg WAT2
```

Set their temperature factors to 30.0 and crystallographical weights to 1.0:

```
MAIN> set temp select xxx end 30.0
MAIN> set weigh select xxx end 1.0
```

At this stage their position in the Fobs-Fcalc electron density can be optimized by energy minimization. First all energy terms except the density one should be excluded map 1 is chosen for the electron density calculation the density scaling factor (ratio) is set to 10.0:

```
MAIN> energy all off dens on dens map 2 dens ratio 10.0
MAIN> minimize select xxx end
MAIN> minimize select xxx end
```

Initialize the variable ii and sets its value to the current number of segments (the variable is later applied in the command file REMOVE_SYMM.COM where the symmetry equivalent solvent atoms are removed) and define a key nnn with all currently present atoms:

```
MAIN> set vari ii = nsegm
MAIN> key nnn select all end
```

Generate the symmetry related atoms from all solvent oxygen atoms that are closer than TOO_FAR from any atom in nnn. The identity operation is not applied (delete on):

```
MAIN> symmetry select xxx .or. wat .and. atom name O* end -
MAIN>  around select nnn end dist TOO_FAR cut delete on
```

Assign key "symm" to all non "nnn" atoms:

```
MAIN> key symm select .not. nnn end
MAIN> show segment
```

Calculate the pair list between the solvent oxygen atoms and their symmetry related ones. A pair of atoms (peaks) is assumed to be identical when they are closer than PAIR_CUT:

```
MAIN> calc pair select symm end select xxx .or. wat .and. atom name O* end -
MAIN>   range 0.0 PAIR_CUT initalize
```

All atoms generated with the same symmetry operation obtain a unique segment name starting with the character #. The remove_symm.com command file runs iteratively until the pair list becomes empty:

```
MAIN> @remove_symm
MAIN> calc pair select symm .or. xxx end -
MAIN>           select orig .and. .not. atom name H* end -
MAIN>           range 0.0 PAIR_CUT initalize
```

Finaly remove the symmetry related peaks by checking the neighbours as well:

```
MAIN> key out select by pair orig .and. ( xxx .or. symm ) end
MAIN> del atom select by sequence out .and. ( xxx .or. symm ) end
```

Calculates the remaining peaks closer than PAIR_LIST to any protein atom. The positions of the paired protein residues is stored in the file pair.list and is supposed to be checked at the display.

```
MAIN> calc pair select protein end -
MAIN>           select symm .or. xxx .or. wat .and. atom name O* end -
MAIN>        range 0.0 PAIR_LIST initial
MAIN> write file pair.list pair
```

After this all solvent atoms that have no hydrogen donnor or acceptor in their vicinity (3.4Å) are deleted with the command file ATOM_NEIGH.COM.

```
MAIN> @atom_neigh
```

It remains only to write the file with the old and newly generated solvent molecules with their starting sequence name START_WATER to the file WATER_OUT and quit the job:

```
MAIN> rename select xxx end seg WAT
MAIN> write file WATER_OUT select wat .or. xxx end -
MAIN>       coordinates  xplor number START_WATER
MAIN> quit
```

Chapter 13

Preparing coordinate file for deposition in PDB

MAIN can not write all for deposition neccessary records, however it can ofer you a substantial help when be their generation. With the help of following scripts you can:

- rearange solvent molecules so that each appear only in the position symmetrically closest to protein atoms,
- attach solvent molecules to their closest chain,
- rename some (nonstandard) tom and residue names and create nonstandard connectivities (as disulfides and attachments between het-groups).

The closest solvent molecule

```
 The following command files 'water_close.com' and
'water_close_symm.com' generate for each solvent molecule all
symmetry equivalent positions and keep only the position
closest to a protein atom.
```

Besides the coordinates file also cell constants and symmetry operations are required. All hydrogen atoms are deleted. Key 'protein' includes the first 6 segments. The number of 'protein' residues is 512, total number of residues is 1046.

get_closets.com:

```
> read file ../ca030/hucathb_10t_3.xpl atoms xpl
> read file ../ca030/dens_10t_omit_80_cyc.amap map alpha
> read file >symm/p21.symm symmetry
> delete atom select atom name H* end
> key protein select segment number 1 6 end


> set vari nres = 1046
> set vari ires = 512


> <water_close
> delete atom select segment name #* end
> write file cathb_to_pdb.pdb atoms pdb
> quit
```

water_close.com:

```
>  set vari ires = ires + 1
>  if ires .gt. nres return
>  calculate pair select residue number ires end select protein .and. \
>       atom name O* N* end initialize range 0. 10. close
>  show pair
>  show pair 1
>  set vari d_close = RESULT_0
>  set vari iatom = IRESULT_0
>  set vari iorig = IRESULT_0
>
>  delete atom select segment name #* end
>  symm select residue number ires end around select protein end \
>       dist 6. cut   delete on
>  calculate pair select segment name #* end select protein .and. \
>       atom name O* N* end initialize range 0. 10. closest
>
>  show pair
>  if IRESULT_0 .le. 0 rewind file
>  show pair 1
>  if RESULT_0 .ge. d_close rewind file
>  set vari d_close = RESULT_0
```

```
>  set vari iatom = IRESULT_0
>  if iorig .eq. iatom rewind file
>  write select atom number iatom end atoms pdb
>  copy coord select atom number iatom end select residue number ires end
>  rewind file
```

Attaching solvents to a chain

The command file 'list_water.com' creates a PAIR list in which each solvent oxygen atom is coupled to its closest protein atom (key 'protein'). It is assumed that the solvent molecule consist of single atom residues and that they are stored at the end of the atomic data. The variable 'ires' should be set to the number of the last protein residue. Some atoms and residues need to be renamed as well.

attach_solvent.com:

```
> read file cathb_to_pdb.pdb atoms pdb
> key protein select number segment 1 6 end
> set vari nres = 1046
> set vari ires = 512


> calculate pair each_closest select .not. protein end \
>    select protein end range 0. 1000.
> set vari iseg = 0
> <write_by_pair
> quit
```

The command file 'write_by_pair.com' writes all 6 protein segments together with their closets solvent molecules.

write_by_pair.com

```
> set vari iseg = iseg + 1
> if iseg .ge. 7 return
> write append file cathb_to_pdb_2.pdb select number segment iseg \
>        .or. by pair number segment iseg end atoms pdb
> rewind file
```

Renaming atoms and residues, creation of unusual bonds

This is the last step where residue and atom names should be adjusted to PDB rules. You do not have to worry about HET records. Program compares each residue name with the list of standard PDB residue names (Appendix C of PDB Atomic coordinate and bibliographic entry format description, Feb-1992).

Program generates RESIDUE list on the segment basis. The residue lists are followed by the ATOM, ATZERO and HETATOM records. Chain identifier is the first letter from the segment name. Cell constants, symmetry operations, disulfides and other conections are written as well. Be careful by calculation of covalent bonds, since each non-disulfide covalent bond connection will be written to the file.

```
>  read file cathb_to_pdb_2.pdb atoms pdb
>  read file ../ca030/diff_10t_3.amap map alpha
>  read file >symm/p21.symm symmetry
>
>  rename atom O select atom name OT1 end
>  rename atom OTX select atom name OT2 end
>  rename atom CD1 select atom name CD* .and. residue name ILE  end
>
>  rename select residue name H2O end resid HOH
>  rename select residue name ET-O end resid ETO
>  rename select residue name EPO .and. atom name O end atom O1
>  rename select residue name EPO .and. atom name C end atom C1
>  rename select residue name EPO .and. atom name O2 end atom O4
>  rename select residue name EPO .and. atom name C2 end atom C4
>  rename select residue name EPO .and. atom name CA end atom C2
>  rename select residue name EPO .and. atom name CB end atom C3
>  rename select residue name EPO .and. atom name OG end atom OH
>
>  rename select number segment 11 12 end segment F
>  rename select number segment  9 10 end segment E
>  rename select number segment  7  8 end segment D
>  rename select number segment  5  6 end segment C
>  rename select number segment  3  4 end segment B
>  rename select number segment  1  2 end segment A
>
>  calculate bond disulfide
>  calculate bond select atom name C2 SG end dist 2.3
```

```
>  calculate bond select resi name ETO EPO .and. atom name OG C4 end
>
>  write over file cathb_pdb_deposit.pdb select all end atoms standard
>  quit
```

Chapter 14

Adapting an atom coordinate file

```
> read file grdbd.xpl atom xpl
```

and connectivity table calculated

```
>  calc bond resi
>  calc bond neigh sele atom name C N end
>  calc bond neigh sele atom name P O* end
>  calc bond disu
```

Second topology libraries and corresponding parameter files must be read. First amino acid parameters

```
> <>utils/get_top_par_19
```

and then nucleic acid parameters

```
>  <>utils/get_top_par_11
```

Residue and atom name matching is veryfied by an DEFINE CLAS sentence.

```
> define clas by topo sele all end
```

As long as this sentence produces error messages there are either nonmatched residues or different atom names.

When atoms do not match with available topology libraries, they have to be modified. Residues and atom names have to be changed by using RENAME:

```
> rename atom CD select atom name CD1 .and. residue name ILE end
```

If some names have blanks in the first columns and they can not be selected by a select command, the best way here is to modify the original file by using a text editor by shifting blanks around and start again at the top.

When there are some missing atoms, they should be inserted. Be carefull with this step, since the nonmatched atoms to a valid topology residue will be gone. (for example C-terminals might be affected).

Missing atoms (for example missing hydrogens) inserted, according to their internal coordinates representation. First the Z-table is created from atom bonding connectivities. The internal coordinates are afterwards calculated according to their connectivities from atom coordinates. Missing atoms are inserted inside the BUILD module and their cartesian coordinates calculated from internal afterwards.

```
> make ztable
> calc internals
> key old sele all end
> build
BUILD> select all end
BUILD> fill
BUILD> dihed
> exit
> calc coord append
```

Since new atoms have probably not been inserted with optimal geometry, it make sense to optimize their geometry by an quick energy minimization. Firts all energy terms have to be defined: Note that atom class of sulphur atoms included in disulphide bonds has to be modified after their assignment from topology residues. The sam is true for C- and N-terminals. Usually it suffices to call the standard procedure (the file name variable DEF_ALL).

```
> <DEF_ALL
```

First energy calclulation is checked and VDW and ELECTROSTATIC terms turned off:

```
> energy vdw off elec off
> energy sele .not old end sele all end
```

If the energy calclulation has been successful, minimization may follow (maybe it should be repeated several times).

```
> minimize sele .not old end sele all end
```

Here carbon atom colors have been modified in order to differentiate easier between protein from the DNA structure.

```
> set col
COLOR> sele name seg DN* .a tom name C* end col 212
> exit
```

Image has been initialized and bonds displayed according to SET of defined atom colors.

```
> image init
```

```
>  image set bond
>  image cent calc
```

If commands are part of a command file, it is nice to tell the program to RETURN input one level higher.

```
> return
```

Modification of amino acid residues usually results in appropriate geometry, however, nucleic acids have atom name starting from the wrong ring, so that a two step procedure is recommended: CHANGE first a THY to NONU (no nucleic) and then to ADE.

Chapter 15

Utility macros

This chapter lists and explains how to use the utility macros, that is the command files in >main/utils.

Protein bonds

The file ">utils/protein.bond" CALCULATES covalent BONDS between a protein atoms on the basis of interatomic distances. In order to speed up the calculation first only bonds within each single residue are calculated, then the connections between C and N atoms for NEIGHBOR residues (residues along the chain) are calculated. Finally DISULFIDE bonds are calculated with an explicit distance criterion 2.1AA.

```
> calculate bond residue
> calculate bond select atom name C N end neighbor
> calculate bond distance 2.1 disulfide
> return
```

Covalent bonds are stored in a connectivity table (CTABLE), where each atom has an entry. Each atom entry contains a list of all its covalent bond partners.

WARNING: You shoul not apply this procedure when several disulfide bonds are overlaping. This is the case when you want to compare similar molecules, superimposed to each other. Apply your own and calculate the disulfide bridges for each molecule separately.

Show unit cell

The procedure '>utils/show.cell' reads in fractional coordinates of cell origins, connects them with covalent bonds and converts them to the orthogonal world. It then displays in blue lattice of several unit cells, designates by 'ATOM NAMES' crystal origin and the axis. WARNING: The unit cell parameters should be read before calling the procedure.

```
> key old sele all end
> read file >utils/5.cell coor gauss
> key cell_atom sele .not. old end
> calc bond sele cell_atom end dist 1.1
> set coor sele cell_atom end ortho
> ima col 245 sele cell_atom end bond
> ima col 245 sele name atom C[%] .and. cell_atom end atom name
> return
```

I recommend you to display unit cells as the first object, immediately after the 'IMAGE INIT' call, delete the unit cell atoms and save the images into the background:

```
> image init
> <>utils/show.cell
> delete atom sele cell_atom end
> image from
```

Ramachandran plot

The command file '>utils/rama.com' reads in the GRAPH of Ramachandran plot and displays it in the GRAPH OBJECT 1. Then a dial set is defined, which allow you to define the size of the plot and position it to the desired place. The dial set is protected from deletion with the 'SET DIAL 1 FROM 1' command.

```
> graph
> object 1
> scale 1.0
> character scale 0.01
> read file >utils/rama.graph
```

```
> exit
>  set dial 1 init
>  set dial 1 graph obj 1 translate
>  set dial 4 graph obj 1 scale
>  set dial 5 graph obj 1 char
>  set dial 1 write
>  set dial 1 from 1
> return
```

he Ramachandarn plot is composed of several image objects, so turn the ones that disturb you off by clicking their image names on the right side of the image window.

Similarly as the dials set, it makes sense to put the plot into the image background in order to prevent its deletion, since as any graph object it will disappear from the image display as soon a redraw command will be invoked somewhere within MAIN. T

```
> image from
```

After this is done, display your molecules and start manipulating them. In order to see where you residue lies click an atom first and then teh HIS_RAMA. A cross will appear at the residue position in the Ramachandran plot. When you rotate about a phi or psi bond the cross will monitor the rotations.

Chapter 16

Installing MAIN

The native environment of MAIN is now UNIX. The section "Under UNIX" describes how to install MAIN under a UNIX operating system. The section "Under VMS" describes how to install MAIN under VMS. You should be aware that the VMS installation is not complete nor completely supported. The section "Tailoring your own documentation scripts" briefly describes the organization of MAIN documentation files and how the PostScript manuals are created. The section "Platforms and array sizes" describes which computer platforms can run MAIN and the corresponding maximum system dimensions.

Under UNIX

The MAIN distribution consists of the following files: doc.tar.Z, show.tar.Z, utils.tar.Z, main.exe_???.Z and .login_UNIX. Compressed manual PostScrip files are available too.

The MAIN distribution files are (mostly) compressed tar files of a directory tree with subdirectories: - "doc" contains the MAIN documentation consisting of txt,tex and example files, - "utils" includes auxiliary command files and data used when running MAIN, code for several symmetry groups, "cmds" files called by MENU functions, sources in "./prog" are auxiliary programs to convert data, in "./top" are topology files and - "show" files are two demonstration examples, - "main.exe_??" are compressed executable files, - ??.ps.Z are compressed manual in PostScript form, ready to be printed, - .login_UNIX is MAIN login ASCII file that needs to be adapted to each installation environment.

In order to install MAIN, first create a directory where MAIN shall reside. Afterwards copy

(ftP) all distribution files to it and install files by running the script install.com:

```
> mkdir main
> cd main
> mkdir dist
> cd dist
```

```
> ftp stef.ijs.si
 Name   : anonymous
 Password: me@....
 ftp> cd dist
 ftp> ascii
 ftp> get README
 ftp> get install.txt
 ftp> get install.com
 ftp> get .login_UNIX
 ftp> bin
 ftp> get mainps_95_DEMO.exe_SG.Z
 ftp> get utils.tar.Z
 ftp> get doc.tar.Z
 ftp> get show.tar.Z
```

The manual PostScript files differ in page size. If you are in the US page size region ftp the '_US' files otherwise get the compressed PostScript files without the '_US' in their names.

```
 ftp> get users_guide_US.ps.Z
 ftp> get dialog_manual_US.ps.Z
 ftp> get ref_manual_US.ps.Z
 ftp> quit
> cd ..
> make -f install.com
```

For a first MAIN installation type:

```
> make -f install.com extract
```

On ESV systems use /bin/tar instead of tar.

After properly extracting the distribution files, create or modify the MAIN ".login_UNIX" file (replace /disk/username with appropriate settings) to define the MAIN environment (MAIN home directory) and aliases. DO not forget to rename the MAIN executable file or correspondingly change the 'mainps' aliases definitions.

```
setenv MAIN "/disk/username/main/"
set path = ($MAIN $MAIN"prog/" $path)
alias main $MAIN"main.exe_SG"
alias mainps $MAIN"mainps_95.exe_SG"
alias psmain $MAIN"mainps_95.exe_SG"
alias offstereo  $MAIN"offstereo"
alias lwplot $MAIN"prog/lwplot.exe_SG"
```

The next step is to fill the "main/top" directory with missing force field parameter files that are part of an X-PLOR distribution. Copy or link xplor files with the names shown below:

- "param19x.pro" to "main/top/par_19x.xpl"
- "param11x.dna" to "main/top/par_11x_dna.xpl"

Then go to the main/prog directory and run the appropriate Makefile (Makefile_SG for Silicon Graphics, Makefile_UALP for DEC/OSF alphas ...). The created executable 'lwplot' transforms MAIN plot files into PostScript files, the 'to_tex' and 'help_to_tex' convert the documentation '.txt' files into LaTeX '.tex' files.

Tailoring your own documentation scripts

The MAIN documentation is stored in the main/doc directory. Original files are editable ASCII text files (*.txt) found in the doc directory. They are converted with help of two executables (to_tex, help_to_tex) and several UNIX 'make' scripts into LaTeX (*.tex) files. The 'to_tex' and 'help_to_tex' are created by running the make command in the 'main/prog' directory.

That is, the simple command

```
> make
```

creates all missing 'tex' files from their 'txt' files,

```
> make latex
```

creates three dvi files via LaTeX (users_guide.dvi, dialog_manual.dvi, ref_manual.dvi) and converts them into corresponding PostScript files. The three major LaTeX files are users_guide.tex, dialog_manual.tex and ref_manual.tex. Their headers assign the main character and page sizes. If you have LaTeX installed on your system and you are not satisfied with the page size or anything else you may create the 'ps' files explicitly with other tools.

```
> make latex_US
```

is an adaptation of 'make latex' which creates 'ps' files in the US letter size on the basis of users_guide_US.tex, dialog_manual_US.tex and ref_manual_US.tex. The 'US.tex' files are created with a different head (latex_head_US.txt) in front of the LaTeX manual structure file (as is users_guide.txt). So all you have to do it to create your 'heads' and correspondingly modify the 'Makefile'.

```
> make clean
```

will delete every file that can be derived from the originals via the to_tex converters, LaTeX and dvips. (If you are using dvi2ps adapt the 'Makefile'.)

The 'xdvi' utility, a previewer of the 'dvi' files usually included in LaTeX installastions, enables you to interactively browse through the 'dvi' files. To use it online with MAIN, add the following aliases to your MAIN '.login' file:

```
alias cool_users_guide  xdvi $MAIN"doc/users_guide"
alias cool_dialog  xdvi $MAIN"doc/dialog_manual"
alias cool_reference  xdvi $MAIN"doc/ref_manual"
```

Under VMS

The login_VMS.com and login_users_gift_VMS.com are VMS login files. The login_users_gift_VMS.com is a good alternative generated by W. Bennett.

To use MAIN on a VMS system create or modify a proper "LOGIN.COM":

```
$ ass disk$huber disk$main
$ ass disk$main:[turk.main] main$d
$ ass disk$main:[turk.main.utils] main$u
$ ass disk$main:[turk.main.prog] main$p
$ ass/tran=conc disk$huber:[turk.main.example.] main$exa
$ ass disk$main:[turk.main.help] main$h
$ ass main$c:sizes. sizes

$ ps9a :== ass disk$main:[turk.main.utils]ps390a.dat psdev

$ main :== $disk$main:[turk.main]main
$ help_me*_main :== help/libr=disk$main:[turk.main]main.hlb
$ lwplot :== $disk$main:[turk.main.prog]lwplot

$ dn6map :== $main$p:dn6map
$ mapout :== $main$p:mapout
$ mapout_r :== $main$p:mapout_r
$ mapfft :== $main$p:mapfft
$ mapfft_r :== $main$p:mapfft_r
$ xplmap :== $main$p:xmap
$ xplmap_r :== $main$p:xmap_r
$ promap :== $main$p:promap
$ promap_r :== $main$p:promap_r
```

A PSDEV file is required for MAINPS to address a desired PS system as follows:

```
LOGDEVNAM=PS330A/PHYDEVTYP=ETHERNET
MODEL=PS330
```

or

```
LOGDEVNAM=PS390C/PHYDEVTYP=ETHERNET
MODEL=PS390
```

WARNING: there should be no blank character at the beginning of a record in a PSDEV file.

Reference manual '[.main.doc.com]*.txt' files are written in a form that allows the conversion of these files into a VMS help library:

```
$ libr/create/help [-]main.hlb help.txt,*.txt
```

Platforms and array sizes

The complete MAIN installation currently runs on: - the Evans & Sutherland ESV series - Hewlett-Packard 700 under X-windows using PHIGS - Silicon Graphics IRIX 4,5,6 machines under X-windows using GL.

MAIN without the graphical interface is available and supported also for VAX VMS and ALPHA/VMS and OSF and CONVEX machines.

For the default sized MAIN, at least 64 Mb of computer memory is strongly recommended, 128Mb preferred. MAIN runs on some 8 bit graphics displays (SGI, HP), but 24 bits of color depth and graphics hardware including Z-buffering algorithms is recommended.

MAIN is written to be generally adaptable for future computers, without for example extensive system specific storage optimization. However, many algorithms implemented in MAIN, such as image rotations, are optimized.

MAIN does not use dynamical memory allocation. Default MAIN can include 30000 atoms, 140 000 points and 6 M real*4 or 24 M character*1 grid point values for map storage. Larger array sizes can be generated by request, to any size structure compatible with computer memory limitations.

Chapter 17

License policy

MAIN development is a research project. The license fees are there to support MAIN development and also living standard of the author(s). The policy is that adding new utilities should not increase the license fees. With growing number of users it is to expect that the license fees will decrease.

There are two kinds of license you can sign for MAIN use: the one restricted to academic use and a general one. They both are a site license agreements and they both include maintenance and user support. The license fees are fixed, which does not mean that you are not allow to donnate more money if you want to. The fee for the academic license is not obligatory, it is a gesture of your good will to reward MAIN creation and user support. The academic license agreement covers a two year maintenace period in order to reduce the paper work.

When requesting a license agreement please send me (by e-mail) your full adress and name and the TCP/IP numbers of your site including its domain name(s). If you want to send a license agreement, please sent it in two signed copies. I will send you one with my signature back and the keep the other for my books.

Each MAIN user gets his own executable file(s), which is bound to a site by its TCP/IP numbers or domain names. The executable file includes the name of the institution and sometimes the name of responsible scientist signing the license agreement. This information is printed when you start the program.

Anyone can freely use complete MAIN documentation, its auxiliary files and MAIN demonstration executable file(s), but remember that freely use does not mean freely distribute and, when it comes to that, a citation of the work is still required. You are encouraged to donnate

your own macros, utilities and a corresponding documentation to MAIN users community as well.

A Site Academic License Agreement for MAIN

```
A Site Academic License Agreement for MAIN


by
CPS (Computer Programs and Services) d.o.o. Ljubljana
Povsetova 33
61000 Ljubljana
Slovenia
```

The following two parties, CPS d.o.o. Ljubljana, as licensor and .....

as licensee, agree to sign the following academic license agreement for computer software MAIN:

This academic license agreement for the MAIN (copyright by CPS d.o.o. Ljubljana) distributed on a distribution media gives right to use the licensed software to the licensee and his contemporary coworkers and guest scientist under the following conditions:

This license agreement is valid when it is signed by both parties and the license fee (1500 DEM) has been paid.

(1) The licensee has no right to give or sell the licensed software in any form to any other person or institution.

(2) The program(s) will be used in academic research institutions located at the licensee research institution only on computers with the first 3 TCP/IP numbers equal to ..... corresponding to the local domain name ..... The fourth number is free. The given license is given to unlimited number of computers and kind of hardware, assuming that an executable file can be provided. If the licensee has access to a supercomputer placed at some other position, this computer should be should explicitly included by its TCP/IP number to this license agreement.

(3) The licensor is obliged to provide software maintenance during period of two years, starting by the date when the license agreement has been signed. Software maintenance includes upgrades of software, bug removal and customer support by the program usage. After this period, licensee may still continue using the software, however the licensor is obliged to continue software maintenance only when an additional license fee prolongs it.

(4) For software usage in commercial activities a commercial license agreement should be signed; commercial activities include, in particular, work under contract from a commercial company.

(5) Program usage will be cited in any publication of scientific results where it was applied. The current reference is: D. Turk: Ph.D. Thesis, 1992, TU Muenchen.

(6) The program bugs will be reported to the author.

(7) The licensor is not liable for any kind of mistake(s) or any possible damage caused by the program(s) usage, including when they are result of a program bug.

(8) Violation of any above specified points terminates the license agreement immediately.

Name, postal and e-mail address of the responsible scientist:

Date, Signature:

For CPS d.o.o. Ljubljana Dusan Turk

A site liicense agreement for MAIN

```
A Site License Agreement for MAIN


by
CPS (Computer Programs and Services) d.o.o. Ljubljana
Povsetova 33
61000 Ljubljana
Slovenia
```

The following two parties, CPS d.o.o. Ljubljana, as licensor and .....

as licensee, agree to the following conditions for use of the computer software MAIN (copyright by CPS d.o.o. Ljubljana) and the accompanying programs:

This license agreement is valid when signed by both parties and the license fee (8000 DEM) has been paid.

(1) The licensee may not sell, give, or otherwise distribute the licensed software in any form to any other person or institution.

(2) The program(s) will be used at locations with the above specified postal address only on

computers with the first 3 TCP/IP numbers of ..... The fourth number is free. The license is otherwise valid for unlimited numbers and types of computers.

(3) The licensor agrees to provide software maintenance during period of one year, starting from the date when the license agreement has been signed. Software maintenance includes upgrades of software, bug removal and customer support for program use. After this period an additional license agreement must be signed for continued use.

(4) The program MAIN will be cited in any publication of scientific results where it was applied. The current reference is: D. Turk: Ph.D. Thesis, 1992, TU Muenchen.

(6) Suspected program bugs will be reported to the author.

(7) The licensor is not liable for any mistake(s) or any possible damage caused by the program(s) usage, including when they are result of a program bug.

Name, postal and e-mail address of the responsible scientist:

Date, Signature:

For CPS d.o.o. Ljubljana Dusan Turk

Chapter 18

Frequently Asked Questions

Abelardo

```
> i) How to display more than one contour level for a given map?
```

you make internal copies of a map:

```
MAIN> copy map 1 2
MAIN> image color 200 map 2 dens 1.3 box 15 15 15
```

this copy just copeis a header and not the data arrray. You can deal with 10 maps simultaneously each at a different contouring level.

try

```
MAIN> show map
```

or

```
MAIN> show map 2
```

```
> ii) Can you explain me how to use the menu on the graphic screen, if
> it is in the manual I could not find it.
```

x-menu and it is in appendix A within DIALOG MODE ACCESSORIES where you will find also the keybord and mouse usage descriptions.

```
> iii) Can you give some example of using the selection menu during
dialoge mo de?
```

First read the description in x-menu and afterwards also the SELECT chapter in command reference manual, and afterwards try to do the following:

Menu is relatively complicated since it allows you to 3 to power of 4 combinations. Try first the simplest:

choose a molecule (molecule is in this combination a network of

covalently attached atoms)

- activate buttons SELE_nn, SEL_MOLE and SEL_LAST (this is the default)
- click an atom on screen -
- activate key active by clicking KEY_ACTI
- clear the key by clicking SEL_CLEA
- press the SEL .or.

and the selected molecule should be marked with white stars

- by clicking SHOW_KEY you can turn the stars off.

select a residue range

- activate buttons SELE_nn, SEL_RESI and SEL_RANG (this is the default)
- click 2 atoms on screen (1st atom = first residue you want to choose) (2nd atom = last residue ..)
- activate key active by clicking KEY_ACTI

- clear the key by clicking SEL_CLEA
- press the SEL .or.

to learn more set the level of MAIN information higher

```
MAIN> set lev 3
```

and then enter the dialog mode and watch the syntax appearing in the MAIN window after clicking menu options. Note that the buttons are flags only - they do not do anything. Action is caused by pressing non buttoned menu keys.

energy calculations and minimizations in the dialog mode

You need to define 2 keys 'active' and 'passive' (KEY_PASI),

- the best way to do it is to define first active (KEY_ACTI)
- and afterwards press KEY_PASI
- and then SEL_COPY

and all atoms included in active will be copied to the key passive in addition each atom of a covalently attached residue to key active will be included in passive as well. Thereby the key active is anchored to the rest of a molecule.

If you have already DEFINED your energy terms, press SHOW_ENE to see the energy terms of the two selections. Key active is interaction with key passive. Active atoms are allowed to move while passive can move only if they are included in the key active as well.

If energy terms are not defined go to page 4 and click DEFINE_A. There may be some warnings comming because some atoms are missing. Take serious the one related to RESIDUE or ATOM not found and the ones related to bond and angle terms.

See what you can do with a RENAME command to adjust your system. very common case:

```
MAIN> rename atom CD select residu name ILE .and. atom name CD1 end
```

MINIMIZE the selections by clicking MINIMIZE (by default it does 50 steps)

The dials window cannot be read in stereo.

I appologize - I am aware of it and I was to lazy to fix it. It will come.

Transparent maps

```
> I found some options in the examples which are not in the
> documentation.  For instance creating continuos surfaces to
> represent maps. Is it possible to create trasparent surfaces?
```

only as maps - not as connoly surfaces

but be aware of the hardware limitations:

```
MAIN> image map 2 dens 1.0 box 10 10 10 area
```

is a code to generate a map surface (or the MAP_AREA menu item).

clicking the REFLECT mode in the image window turns the reflectance mode on then click the Amap_2 for example and rotate dials - transparency is the dial number 4 (the same as scaling).

there is still some problem with it - in some directions transparency still has some problems.

What algorithm are you using for energy minimization?

Konjugated gradient.

From Ren Bin in Sweden - averaging

```
> what's the purpose of RHOGEN?  what is the difference
> between the main map and the protein map?
```

The purpose of this file is replace the FFTFC program that was used before to calculate structure factors from atomic model. RHOGEN.COM generates electron density from atomic model through the whole unit cell and writes the map in the Lyn Ten Eyck format readable by PROTEIN.

```
> In RHOGEN, I need a file called "set\_cell\_box". what
> kind of file is it? How to determine the input values in
> this file?
```

RHOGEN requires layout (0 0 0 to the end of the cell)

```
> In RHOGRN, the origin of the box was set to "0 0
> 0". Is this a general situation?  How to determine it?
```

Yes it should always be 0 0 0.

```
> (4). In the command file you sent to me, the symmetry
> operations repeated 3 times, using matr 2, matr 3 and matr
> 4. It seems that the number of repeat corresponds to the
> number of the crystal symmetry operations. Is this correct?
> In my case, I have space group I4(1).  Should I repeat 7
> times, from "matr 2" to "matr 8"?
```

Yes the number of reeats cooresponds to the number of symmetry operations that are carried out one by one. The case I have sent you has C2 group.

The current version generates electron density from atomic model appyling all crystal symmetry operation:

```
MAIN> make map 1 atom dens select all end symmetry
```

```
> Another problem is how to determine "tran 0 0 0". Is
> this the translation of the crystal symmetry operation? If
> I have a symmetry operation like: " -x+1/2, -y+1/2, z+1/2",
> should I specify it as "tran 0.5 0.5 0.5"? Why did you use
> "0 0 0" in all of the symmetry operations?
```

The 'tran 0 0 0' is an additional translation applied along the crystal axis. It is a MAIN syntax requirenment. It simply means that no additional translation will be applied. Translational components can only be integer numbers and never 0.5. The 0.5 or similar shifts should be included already in the symmetry operators.

```
> (5). After RHOGEN, I will calculate the density map
> *.diy. In the calculation, I need a file called "br3.pro"?
> Is this a fobs file (Fc) calculated from PROTEIN using FFT?
> If not, how to generate it?
```

br3.pro is a PROTEIN reflection file.

```
> (6). In your last message, you suggested that it's better
> to use RMS.COM to generate the rotation matrix that will be
> used in averaging. How to incorporate the
> "set\_tran\_*\_to\_*.com" into the command files of the new
> version? should I do averaging in two steps (like the old
> version): ROTA\_MAPS.COM and AVER\_MAPS.COM ?
```

You can do it as you wish. The previous version stores intermediate results to disk and does averaging within an additional step. The example I have sent you no intermediate results (rotated maps) are beeing stored.

The set_tran*to*.com defines parameters for the

```
> make map nn from mm rotate ...
```

so that you can still use the 'old' syntax for the map rotations.

On the fly calculations of parameters are neccessary if your model is being refined during averaging, because model transformations slightly effect the rotational and translational parameters.

```
> (7). In GEN\_CELL.COM, file "head.rmap" will be used. I have
> edited a head.rmap, but I don't know whether it is correct?
> There are different kinds of head.rmap in the manual.  I
> don't know which one I should follow. Could you send a
> example to me?
```

You should use your own head.rmap. The head.rmap contains the symmetry parameters, cell constants and layout of the map. It is always case specific - it means that I can not create a header for you.

```
> Sorry to ask so many questions. I wish you wouldn't feel bored.
```

Thanks for good wishes.

Rick, surface point file and end of selection

```
> Writing out surface file.  Reading back there are holes
> in it.  If you display immediately after calculation it is
> ok, so apparently not all of the surface is written to the
> file.
```

The default should be all points (but I am never sure). All points mean contact + reentrant. I guess that some default must be wrong. Try:

```
MAIN> image col 160 poin surface all
```

and compare it with

```
MAIN> image col 200 poin surface
```

or

```
MAIN> write poin surface all
```

and

```
MAIN> write poin surface
```

```
> If you have a segment name 'E', you cannot make the
> command: select seg name A .or seg name B .or seg name C
> .or seg name E end because E is confused with end (or
> something like this).
```

END can be abreviated to an E. (First the interpreter searches for end, then logical operations (.and. .or. .not. and brackets) and afterwards starts to interpret tags. You can get around with this:

```
> ... select segm name A B C D E* end
```

which includes atoms of segments A, B, C, D and all segments starting with E.

Stefan on averaging

To be sure that you have set the procedure properly check each step. Inspect all the first time averaged maps (not only the final result and its r-value).

Look at messages coming out of MAIN and try to understand them. Comparing numbers of converted, rotated and set grid points tell you a lot about consistency of your masks, and the procedure in general.

The numbers coming out of MAKE_MAP_CELL> tell you how many of the density points of each mask have been brought to the P1 environment.

Saving structure factors

```
> write over file xx refl fobs complex defined
```

should be OK. PHASE is not a command word there (POLAR is what writes amplitudes (fcalc) and the phases.) Complex is another possibility. Try :

```
> write refl ?
```

to get the ideas about the possible command words.

Writing out the final maps

There is no need to write everything in an X-plor map format.

First: the main number format is under VMS much smaller

```
> write over file average.rmap map 1 number
```

if you are happy with 8 bit precesion you can store a map with:

```
> write .... map 1 char
```

or with ALPHA instead of the CHAR format . CHAR maps are not transferrable to other machines, with the alpha form is computer system independent (but is only MAIN readable).

second: use MAKE syntax to create a smaller map for visual inspection:

```
> make map 2 from 1 around 8 select segment name ... end init 9999
> make map 2 from 1 copy
> write file xx.rmap map 2 xplor/alpha ....
```

the around 8 means create a box 8 grid points outside your selection. If the averaging procedure has converged , then it should be in principle enough to work with a single model only.

Segment name E

```
> still I dont know how to avoid this non unique selection E
>
> rename segm MOL4 sele segm name  D end
> MAIN>
> rename segm MOL5 sele segm name  E end
> FINDCOM> COMMAND IS NOT UNIQUE **E en**
> MAIN>
> rename segm MOL6 sele segm name  F end
> MAIN>
```

The segment name E is a pain of the select sentence. MAIN thinks that it is an abbreviation for END. See in the manual user questions or users corner (at the very end) there should be something about this. A quick suggestion: Use

```
> rename segm MOL5 sele segm name  E* end
```

RMS fit convergence

```
> in the step rms_fit.com there are some things in the
> log file that I don't understand. What is this SPECIAL STEP
> and is the message "CAN NOT CONTINUE" a normal termination
> of the process?
> The special step gives no improvement in about 50% of the cases
> and improvement in the rest as you can see...
```

This was OK. The message only means that the procedure did not numerically converge, however the results could not be improved either. What you should care about is the last value in the line

```
> RMS_FIT> PARAMETERS  -7.145 -66.704 -99.066 -84.176  97.608  48.024  0.3329
                                                                       ^^^^^^
```

This number is the RMS deviation. If it is unreasonable high (above 5 or so) then you should offer the fit procedure some initial values to improve convergence. As I see everything is OK with your case.

```
> rms coor all sele ... end sele ... end init 0. 180. 0. 70. 8. 90.
```

or whatever (x, y' and x" rotations plus 3 translations).

Chapter 19

Glossary of MAIN terms

#connectivity table# table of covalent bond connectivities. Each atom has a record in which all attached atoms are listed

#ctable# see connectivity table

#history list# all clicked atoms, enoted in the image with white labels of atom, residue sequence names, unless you have change the defaults.