CCP4 NEWSLETTER ON PROTEIN CRYSTALLOGRAPHY

An informal Newsletter associated with the BBSRC Collaborative Computational Project No. 4 on Protein Crystallography.

Number 36 **February 1999**

Contents

1. **News from CCP4**
   Peter Briggs, Martyn Winn, Sue Bailey, and Alun Ashton
2. **Report on the 1999 CCP4 Study Weekend in Sheffield: Data Collection and Processing**
   Johan Turkenburg
3. **Report from the Joint CCP4/EBI Software Developers and Data Harvesting Workshop**
   Kim Henrick and Eleanor Dodson
4. **Report from the CCP4/EU 1998 Advanced Methods Refinement Workshop at York**
   Eleanor Dodson
5. **News of the CCP4I Documentation and Tutorials**
   Maria Turkenburg
6. **MMIFL: Dealing with Diversity in Scientific Image Formats**
   J Bernard Heymann
7. **Recent improvements to the `dm' package**
   Kevin Cowtan
8. **Datasets to Maps: A Wrapper for SHELXS and the CCP4 Program Suite**
   Paul Emsley
9. **DynDom: A Program to Determine Domains, Hinges Axes, and Hinge Bending Residues**
   Steven Hayward
10. **Improved Software for Laue Data Processing**
    Steffi Arzt
11. **Crystallography Under Linux: A Review of Several Currently Available Programs**
    Mary A. Canady and John Tate
12. **News of MOSFLM 6.0**
    Harry Powell
13. **Writing CCP4 Scripts in Perl**
    E. Courcelle and J.P. Samama

Editor: Peter Briggs

Daresbury Laboratory, Daresbury,
Warrington, WA4 4AD, UK

CCP4 News: February 1999

Peter Briggs, Martyn Winn, Sue Bailey, and Alun Ashton

News of CCP4 Release 3.5

By the time you read this, Version 3.5 of the CCP4 suite should be available. In addition to a number of bug fixes and enhancements to existing programs accumulated since Version 3.4, this release includes the following new programs:

- **DETWIN**: detwins merohedrally twinned data (see Andrew Leslie's article in newsletter 35)
- **DYNDOM**: program to determine domains, hinge axes and hinge bending residues in proteins where two confirmations are available (see Steven Hayward's article in this issue).
- **FINDNCS**: detect NCS operations automatically from heavy atom sites (from Guoguang Lu).
- **MTZMADMOD**: jiffy program for converting between F+/F- and F/D.
- **TOPP**: an automatic topological and atomic comparison program for protein structures (from Guoguang Lu).
- **WATNCS**: pick waters which follow NCS and sort out to NCS asymmetric unit (from Guoguang Lu).

There are also new versions of: **DM** (2.0.3) (see Kevin Cowtan's article on the new DM in this issue), **SCALA** (2.5.5), **REFMAC** (4.0.2 with anisotropic refinement - see Garib *et al's* article in newsletter 35), **AREAIMOL** and **CONTACT**.

Other new developments include:

- options for the automatic installation of the X-windows programs along with the main suite for SGIs, Linux and OSF1
- an option for 64-bit compilation for SGIs under Irix 6.*and a new IRIX-64 installation option
- new library routines for inserting HTML tags into logfiles, including java applets for viewing graphs (incorporated into output of DM and SCALA)
- MTZ file format expanded to include information on datasets in the file header
- a new example dataset incorporating NCS (non-crystallographic symmetry)
- extended documentation for library subroutines, and the addition of more ``general help documentation.''

Details of all the changes will be found in the CHANGES file in the top-level directory ($CCP4), and in $CCP4/html/CHANGESinV3_5.html. As always, problems will be reported on the Problems Page, and bugs should be sent to us at ccp4@dl.ac.uk.

News of the CCP4 Graphical User Interface

Work is still on-going with CCP4I, the graphical user interface for CCP4 programs which is being developed by Liz Potterton at York.

An official release of CCP4I is still some months away, but in the meantime Liz would like to invite interested sites to test an "alpha-release" version of the interface which is available now (see http://www.yorvic.york.ac.uk/~lizp/ccp4i_installation.html). All feedback (positive and negative) is extremely valuable and will be much appreciated.

Demonstrations of the interface will be given at a number of conferences and workshops through the year (including the ACA meeting in Buffalo NY and the Como Workshop, both in May, and the IUCr Congress in August), and in addition Liz is also preparing to demonstrate CCP4I at individual sites around the UK. Please contact her by e-mail (lizp@yorvic.york.ac.uk) if you are interested in hosting such a demonstration.

When any issues raised by the alpha test sites have been adressed and when the interface has been brought into line with the new release 3.5 of CCP4, a beta test version will be announced on the CCP4 bulletin board. It is expected that this version will be sound and will cover the range of most commonly used CCP4 programs, so it should be useful to many people and we encourage you to give it a try.

In the meantime, background information about the interface can be found in the previous newsletter article, whilst Maria Turkenburg's article on the associated documentation and tutorials appears in this issue.

Workshops

CCP4 participated in several meetings during the latter half of 1998.

A one day workshop was held on the Saturday before the start of last August's **ECM meeting in Prague**. The morning consisted of a series of introductory talks on the role of CCP4, the nature of the program suite, an overview of the standard file formats used, and brief descriptions of the most important programs. In the afternoon, a number of question and answer sessions were held in parallel covering broad areas such as data processing and model refinement, so that participants could explore details of the structure solution process. The day was very informal, allowing close contact between users of CCP4 and experts in the field.

The following month, a joint CCP4-EBI workshop was held at the **EBI (European Bioinformatics Institute, Hinxton, Cambridge)** on 16th to 19th September of last year. The workshop was aimed primarily at software developers and covered a number of different areas: there was an introduction to the concept of automatic data harvesting during structure determination for deposition information (see Kim Henrick's article on harvesting in newsletter 35). There was also some instruction on the use of the CCP4 libraries and an introduction to the new CCP4 working coordinate file format, based on mmCIF (see 'News' from newsletter 35). This was a lively meeting with discussions extending into the areas of validation and deposition, and much debate was provoked amongst participants representing the depositors and those in charge of the deposition sites. A report on the workshop and the issues raised appears in this newsletter.

1998 ended with a joint CCP4-EU funded **Advanced Methods Refinement Workshop** held at York University on 12th to 17th of December. This brought together students from across the EU and gave them the opportunity to hear program developers and other refinement experts lecture on general principles, and at the same time receive hands-on supervision whilst working with their own data. Eleanor Dodson gives a short report on the workshop in this newsletter.

1999 was seen in with the **CCP4 Study Weekend on Data Collection and Processing**, held in Sheffield on the 8th/9th of January 1999. The meeting was a great success, being attended by nearly 450 people and including a number of prestigious international speakers. Johan Turkenburg, one of the organisers, gives a report on the weekend in this issue.

For the coming year there are plans for demonstrations of the CCP4 graphical user interface at a number of conferences and workshops (see above), and you can see us at our stand at the **XVIIIth International Union of Crystallography Congress in Glasgow** this August (see http://www.chem.gla.ac.uk/iucr99/ for details of the IUCr meeting). We also hope to be running another one-day workshop there, along the lines of the one in Prague - look for announcements on the bulletin board, or contact ccp4@dl.ac.uk.

Acknowledgements, Contributions and Contact Details

CCP4 would like to thank all those who provided articles for this issue of the newsletter.

Contributions for future issues are welcome, please contact Peter Briggs (e-mail:p.j.briggs@dl.ac.uk) in the first instance. The next issue should be due out in August/September 1999.

The general contact details for CCP4 are:

CCP4
Daresbury Laboratory,
Daresbury,
Warrington
WA4 4AD, U.K.
ccp4@dl.ac.uk

Report on the 1999 CCP4 Study Weekend in Sheffield on Data Collection and Processing.

Johan Turkenburg

Venue:	Sheffield University
Dates:	8-9 January 1999
Programme Organisers:	Leo Brady and Johan Turkenburg

Data collection and processing is obviously a crucial step in the determination of any X-ray crystal structure, as it provides the observations that are used throughout the process of structure solution and refinement. Mistakes made at this stage will haunt the unfortunate crystallographer, and may well block the road to crystallographic success. Computer programs can be run time and time again, but restraints on the availability of crystals and (synchrotron) data collection time mean that the data may only be collected once. In recognition of these facts of a crystallographer's life, 'Data collection and Processing' is a recurring theme for the CCP4 study weekends, and was chosen as the subject in 1987 and 1993. Since then third generation synchrotron sources have become available. The development of new optics systems for home sources, the more widespread availability of image plate detectors and more recently CCD detectors and the now universal application of cryo techniques, has changed the face of data collection. Together with advances in data processing and other crystallographic software, and the increased speed of computing this has led to a considerably higher throughput in macromolecular crystallography. It was therefore felt that it was high time to revisit this subject in 1999.

The meeting was held at the University of Sheffield. Thursday night a large number of delegates found their way to the bar in one of the Halls. A great deal of crystallography and other material was discussed while enjoying some beverages. Friday at 11:00 the meeting started in earnest. The programme promised a total of 18 speakers, covering all aspects of data collection and processing. Michael Rossmann gave a general introduction to data processing, and retold some of the history of the software used nowadays. He then went on to describe programmes more recently developed in Purdue in collaboration with ADSC. This has resulted in autoindexing routines which are extensively described in the literature and now also form part of the MOSFLM suite, as described in a later talk by Harry Powell.

Elspeth Garman talked about cryocrystallography and data collection in general. With her vast experience in this area she had examples of almost everything you should think about when preparing for an X-ray experiment, and of lots of more and less common mistakes. Surely no-one is going to breathe on their precious, frozen crystal ever again. She also discussed the potential of cooling crystals to liquid helium temperatures, and the benefits this might bring. This subject was returned to in the general discussion on Saturday afternoon.

After coffee Peter Lindley described the use of a third generation synchrotron source, and more specifically what is available at the ESRF. Colin Nave then explained how to match the X-ray source, optics and detectors to the protein crystal under consideration. Although there is often a limited choice in practice, it is obviously very important to be aware of the issues involved, especially when trying to decide which synchrotron to apply to for beam time. CCD detectors are now becoming standard equipment at most synchrotrons. They are, however, considerably more expensive than image plate detectors, often a major consideration for home laboratories. Steven Muchmore from Abbott Laboratories talked about his experience with CCD detectors on a laboratory source, and the pros and cons of such a setup. An exciting new development in the field of X-ray sources is the microfocus tube developed by Ulli Arndt and more recently Bede Scientific. Anne Bloomer has been testing such a source, and presented results as well as the potential for future development.

Over the last five or so years most laboratories have invested in mirror optics for their protein setups, often in the form of Yale mirrors. The next development is already upon us, with the introduction of multilayer optics for home sources. After tea Joe Ferrara from MSC presented results of extensive tests carried out with Yale mirrors and various incarnations of multilayer optics. The subject then changed from hardware to software, and Harry Powell described the implementation of the Rossmann indexing software in MOSFLM, as well as the algorithms involved. Andrew Leslie covered the next step in data processing, the integration of the diffraction data and described how the sigmas are determined for individual reflections. Zbyszek Otwinowski closed the day on a philosophical note, stressing that scaling, merging and statistics can be very case specific and do not follow hard and fast rules. Critical thinking throughout the process is of the essence.

Saturday morning most of the delegates returned after what some regard as the most important part of the CCP4 Study Weekend: the extended discussion time during dinner and in the bar. Zbyszek Dauter gave a very clear talk on data collection strategies. Reciprocal space suddenly seemed almost real. That diffraction is something we should consider in three dimensions was also stressed by Jim Pflugrath. Fine sliced data collection and matching data processing were illustrated with pizza slices and bits of pepperoni. Surely we all felt that this is a more tasteful approach to data collection! Martin Walsh talked about MAD data collection at the APS. The speed is certainly awesome, and in some cases structures can be determined in a matter of days.

After the coffee break Dominique Bourgeois illustrated the need for specialised processing software when processing time-resolved data. Algorithms developed here can also be used to alleviate overlap problems in conventional data sets. X-rays are of course not the only radiation which can be used to study structure. Johnny Grimes gave some examples of how EM data can complement the X-ray information, and how it can provide initial, low resolution phases. For many years twins have been a common phenomenon in small molecule studies. In recent years protein crystallographers have become more aware of their existence, and Naveen Chandra described various kinds of twinning as well as how to recognise them.

After another excellent lunch Randy Read showed how structure factor statistics may be used to detect outliers in non-redundant data. In such cases there are often at most two observations for a reflection, and deciding whether one of these is an outlier can usefully rely on information from other sources, like known distributions of structure factors. In the 'hot news' slot of the programme, Anastassis Perrakis described a beamline for microcrystals at the ESRF. The design of components like the goniometer and the retractable backstop should make dealing with such crystals more feasable. It is even envisaged that in an effort to overcome problems associated with the finite lifetime of frozen crystals in very intense beams, a number of crystals sitting in one cryo loop can be used one by one for data collection.

The meeting was closed by Phil Evans, who summed up the requirements for proper data collection and processing, and led the final discussion. Elspeth Garman introduced the issue of crystal lifetime, and the possible advantages of using liquid helium. No consensus was reached as to which processes are responsible for crystal damage. There was also some discussion on the need for the design of better backstops to reduce background and the usefulness of fine sliced data collection. Phil remarked that synchrotrons are not just a place to quickly collect data, but also to carry out proper data collection experiments.

The full proceedings will be published as a supplement to Acta Crystallographica section D, sometime towards the end of the year.

As one of the programme organisers I would like to thank the conference office and audio-visual department of Sheffield University, Pat Broadhurst (Daresbury) and the CCP4 staff, especially Alun Ashton and Sue Bailey.

Report from the Joint CCP4/EBI Software Developers and Data Harvesting Workshop

Kim Henrick and Eleanor Dodson

This is a report on the Protein Crystallographic Software Developers Workshop for the use of Standard Library Routines with CCP4 and EBI Data Harvesting which was held on September 16th to 19th 1998 at the European Bioinformatics Institute Hinxton, Cambridge CB10 1SD, UK

The list of participants can be found at http:/www.dl.ac.uk/CCP/CCP4/newsletter36/03_participants.html. This is a slightly edited version of the report originally produced by Kim and Eleanor.

Introduction

The workshop was held to discuss Data Harvesting of the detail required to characterise macromolecular structure determination and description in a consistent manner (see the background notes from a previous newsletter article at http://www.dl.ac.uk/CCP/CCP4/newsletter35/dataharvest.html). The introduction to the CCP4 subroutine libraries and data format files was presented as an example of standards for a mechanism to allow for the free flow of data between the major structure determination packages, including CNS, O, and CCP4, as well as for deposition. CCP4 also presented their current ideas for using mmCIF as a working coordinate file format to be used in the process of a structure determination.

The workshop outlined the current deposition practice, mandatory and voluntary requirements for deposition, the use of mmCIF tokens to define this information, and the demonstration of a future deposition tool. Discussions were also held on methods for the implementation of data harvesting specific to particular software packages, that were realistic and achievable.

1. Harvesting

Kim Henrick gave a general introduction to the harvest concept (see some of the slides at http://www2.ebi.ac.uk/msd/Harvest/notes/index.htm). There was total acceptance that programs should produce the required information in a recognisable style, and it was agreed that every deposition should have associated in-house tags that identified the **Project_Name** and **Data_Set_Name**, where the Project_Name is the working equivalent to what will become a PDB IDcode (or in mmCIF terms the **_entry.id**) and the Data_Set_Name is the particular data set (either X-ray diffraction structure factors or NMR experimentally determined data) that is associated with the deposited refined coordinates (**_diffrn.id** in mmCIF). Each run then of a particular program would then produce a snapshot harvest file containing the relevant information known to that program as mmCIF tags and value pairs. In addition each file requires a date stamp, **_audit.creation_date** , using the CIF definition, CIF-datetime

The final set of these files would then be deposited and the central archive sites would merge and process this information automatically.

It was acknowledged that all software packages need to encourage users to consistently input the same in-house identifiers to each program by a common set of key words for Project_Name and Data_Set_Name. Data harvesting throughout the course of a structure determination requires a minimal discipline from the researcher.

Programmers are required to follow mmCIF definitions for the items selected as important for deposition. Paula Fiztgerald present the plan for extending the mmCIF Dictionary with new data items as set out in the URL http://ndbserver.rutgers.edu/mmcif/dict-templates/ex.html (for Europe http://ndbserver.ebi.ac.uk:5700/mmcif/dict-templates/ex.html) which outlines CIF extensions.

The CCP4 prototype system uses an automatic routine to place harvest files from each program run into a fixed directory structure and file naming convention based on: **$HARVEST/Deposit_Files/Project_Name/Data_Set_Name.Program_name_function** where **$HARVEST** is an environment variable and **function** is the particular run number or run path using a particular software package.

This file naming and placement proposal was discussed and thought to be difficult to maintain by the nature of the problems found when solving a macromolecular structure. In particular some steps may be carried out at different sites and for each stage program steps are often re-run and the stages are repeated in an iterative cycle. Data sets are commonly discarded as experience is gained and better crystals that give better data are grown. Therefore the above two program instructions are only required for the final run of each program. However, it is not always known at the time of a particular run that it will be the final one, and in the case of a preliminary publication the results are not definitive.

The data harvesting concept however is flexible enough to allow each program developer to not only select the data items generated by their own software but also to implement a method that is practical for that programs use.

The CNS developers, Paul Adams and Ralf Grosse-Kunstleve with John Westbrook have a procedure to generate deposition information when a user has finished refinement. The CNS method will request the Project_Name and Data_Set_Name when this macro is run and include sufficient information for the deposition centres to determine the exact chemical nature of all compounds represented in the coordinate set. The CNS mmCIF pdbsubmission macro will be extended to save coordinates, reflections, topology, parameters, and MTF information as a deposit.mmcif file. The macro will be added to with heavy atom details at a later date.

Bart Hazes presented some ideas on how to use harvested data for the purposes of the experimentalists rather than just for the deposition centres. This entails making harvesting attractive to the user by using the harvest file information as a source of in-house archive information. As the harvest files contain the experimental parameters and results of each stage of a structure determination, this information can be used in other ways. A GUI was suggested to read harvest files and to prepare experimental data for later reference, visualisation and additions of non-automatic deposition data items.

Randy Read suggested adding a history of parentage to harvest files to manage versions wherein additional data items would track the stages and versions that lead to each file.

Once an experiment was completed either in-house of the deposition centre could then trace the history of files and offer a mechanism to ``weed'' out or select in only those pieces of information that were considered by the user to be the final versions.

Herb Bernstein gave a brief outline of the proposal to embed binary information in a CIF. This has harvest application in that images collected at an X-Ray source can carry Project_Name and Data_Set_Name tags along with other diffrn category data items can be passed into the first step of structure determination and carried automatically along via file headers throughout the whole structure solution process.

(See the URLs http://ndbserver.rutgers.edu/mmcif/cbf/ (or for Europe http://ndbserver.ebi.ac.uk:5700/mmcif/cbf/ for details of CIF-CBF; see also http://www.esrf.fr/computing/Forum/imgCIF/cbf_definition.html for CBF_definitions and http://www.bernstein-plus-sons.com/software/CBF/ for CBF_software.)

Herb Bernstein also announced that there is now a RASMOL that will work directly on mmCIF input files. This is encouraging news that more tools are being coming available to process mmCIF files.

Follow-up Note:

Raimond Ravelli has proposed that ESRF beam line software can be modified to generate a first mmCIF harvest file containing a set of **_diffrn** and **_exptl_crystal** data items, i.e.

```
_diffrn_radiation.collimation
_diffrn_radiation.monochromator
_diffrn_radiation.wavelength_id
_diffrn_radiation.polarisn_norm
_diffrn_detector.detector
_diffrn_detector.details
_diffrn_detector.dtime
_diffrn_detector.type
_diffrn_radiation_wavelength.wavelength
_diffrn.ambient_temp
_diffrn.crystal_support
_diffrn.crystal_treatment
_diffrn.details
_exptl_crystal.size_max
_exptl_crystal.size_mid
_exptl_crystal.size_min
_exptl_crystal_grow_comp.details
```

Gerard Kleywegt has proposed harvesting for O/OOPS, to generate files containing:-

```
loop_
_struct_mon_prot.comp_id
_struct_mon_prot.auth_seq_id
_struct_mon_prot.asym_id
_struct_mon_prot.alt_id
_struct_mon_prot.RSR_all
_struct_mon_prot.RSR_main
_struct_mon_prot.RSR_side
_struct_mon_prot.RSCC_all
```

Clemens Vonrhein and John Irwin are modifying the SHARP program to harvest heavy atom sites and phasing information.

2. Deposition Tools

Helen Berman presented an overview of the NDB's WWW deposition tool, **mmCIFIT**, and the philosophy behind its function. Here a mechanism for the capture of accurate information was demonstrated. The information content of the NDB was divided between, primary data collected from the depositor; derived information calculated by the NDB; and, qualitative information, shared between annotater and depositor. Some of the problems inherent in producing a smooth running deposition system were discussed, in the handling of multiple file formats, strict definition for the meaning of items, and maintaining a high through-put together with data uniformity. John Westbrook introduced the entire integrated data processing system which has syntax and semantics defined by mmCIF, where dictionaries play the key role in driving the software. The current validation features were also presented.

They divide the task into:-

```
 Pre-deposition checks
  a) Format and nomenclature
  b) validation checks
   Geometry:               Procheck
   Structure factors:      SFCHECK
```

Complete mmCIF files will be generated after processing is complete. Currently structure factors are expected in mmCIF format only. The system allows for pre-deposition checks before a deposition starts. The possibility of validation reports being made available to referees before an entry is released was raised, but is not yet allowed.

John Westbrook demonstrated the mmCIFIT deposition tool and a discussion followed concerning mandatory data items. The NDB viewed the sub-tokens for the PDB records COMPND and SOURCE to be over complex and in need of simplification. The NDB intend to also simplify the possible HEADER record classes for macromolecular entries. There was a discussion on the JRNL record as mandatory. There was in-sufficient time to complete a discussion of mandatory items as the discussion of input via a web tool of PDB equivalents to REMARK 200 and REMARK 280 revealed how complicated modern crystallography data collection could be. To capture all details that would fully define a diffraction experiment that could involve different crystals, and diffraction experiments and relating this to mmCIF identifiers that described crystallisation, diffraction, data reduction, and final structure factor refln sets was shown to be very complex.

Eldon Ulrich then present the BMRB deposition tools and schema for handling NMR experimental data. The BMRB system is a STAR based system that is currently adapted to work with the PDB's AutoDep2.1 procedure. The STAR nmr format was devised to record the multiple sets of samples and experiments that are undertaken by NMR to determine a macromolecular structure and reveal how inadequate the current PDB REMARK 210 was to realistically record details of these procedures. The **ZOO** project, a Desktop Experiment Management Environment that can be customized and placed on the desks of many scientists to manage their experimental studies, was outlined (see http://www.cs.wisc.edu/ZOO/). This is an in-house database system that can be used to manage all aspects of the determination of an NMR macromolecular structure. Coupled with the **DEVise** system, an Environment for Data Exploration and Visualization (see http://www.cs.wisc.edu/devise/), a data exploration system that allows users to easily develop, browse, and share visual presentations of large tabular datasets from several sources, the BMRB is working towards developing an intuitive set of querying and visualization primitives that can be combined to develop a set of visual presentations that integrate data from a wide range of application domains.

Peter Keller presented a short overview of the EBI's development of a deposition tool (see part of this presentation at http://www2.ebi.ac.uk/msd/Harvest/peter/index.htm). This will be based on the experience gained by the PDB in processing real data and will follow the outline of the current AutoDep2.1 process. The internal organisation will use commercial data-base management systems (ORACLE). The schema, processing flow, administration, version control and web interface will be held in the database schema SQL. The system will also generate web pages unique to individual depositions as required to request missing mandatory information.

3. CCP4 Library Routines

These sessions addressed the programmers present at the workshop. Phil Evans described the philosophy behind the CCP4 suite. CCP4 is a set of programs written by individuals, linked by a set of common formats. He devoted some time to describing the reflection data file, with the concept of labeled and typed columns. A detailed description of libraries is on the web at URL, see http://www.dl.ac.uk/CCP/CCP4/dist/html/INDEX.html (CCP4 software documentation), together with library documentation. An extended set of templates for software developers are also available on the web at http://www.dl.ac.uk/CCP/CCP4/dev/templates/templates.html (CCP4 template fortran files).

Eleanor Dodson gave an overview of the CCP4 symmetry subroutine library, Phil Evans introduced the maplib routines and Martyn Winn discussed in detail the reflection format file and software routines for the MTZ library. Liz Potterton gave a presentation on a new CCP4 GUI (see a previous newsletter article at http://www.yorvic.york.ac.uk/~lizp/newsletter.html).

Finally Alun Ashton gave an introduction to how to compile and run CCP4 style programs and introduced the keyparse routines.

4. mmCIF Coordinate Files

CCP4 devoted a session to the detailed discussions of mmCIF questions as related to coordinate files for a working format during the stages of structure determination (see http://www.dl.ac.uk/CCP/CCP4/usercif.html (CCP4 CIF notes) and http://www.dl.ac.uk/CCP/CCP4/dev/cif/newformat.html (CCP4 CIF coordinates)).

CCP4 plans to begin using a subset of mmCIF definitions as an atom coordinate format. The advantages of this are:-

```
The mmCIF atom format is much more appropriate for describing the model;
It is more flexible, and easily extensible.
It moves towards the IUCr standard.
```

The everyday use of this format raises several problems that require restrictions to be put on the mmCIF categories and data items to make them usable within existing programs without major rewriting.

Martyn Winn outlined CCP4 practice/theory.

Within CCP4 programs the atom format (currently PDB) is used to transfer information which is not always strictly atom coordinates. BONES skeletonisation, atomic vectors, and peaks taken from electron density are examples of this.

Martyn is preparing a set of library subroutines to replace RWBROOK which will read and write the cCIF stuff. He uses Peter Keller's libccif toolbox for the low level parsing (see LIBCCIF at http://www.ebi.ac.uk/~keller/ccif).

Initially these will read/write the following categories.

Entity	**describes contents of molecule**
Entity_poly_seq	**gives sequence of polymers within molecule.**
Cell	**cell dimensions**
Atom_sites	**conversion from fractional to orthogonal**
Symmetry	**symmetry information (space group etc)**
Symmetry_equiv	**symmetry operators**
Struct_asym	**content of asymmetric unit (eg: 2 molecules + 1 Zn)**
Struct_conn	**any special connectivities (eg disulphides, covalently bound sugars etc)**
atom_site	**description of atomic position; name, plus xyz**

A set of mmCIF coordinate handling routines have been written and tested. These routines transfer header information from input to output files in a similar way to the MTZLIB procedures. A further set of routines get and put coordinates as individual **atom_site** ``row'' by ``row''.

The CCP4 format will impose some restrictions to standard mmCIF **loop_** structures. The **atom_site** records will be formatted. Each **atom_site** should occupy a single row, so that simple unix and awk utilities could still work. There will be a standard order within the loops. Many character data values will be restricted to char(8) (or less, rather than the mmCIF 80 character limit).

Martyn identified various problems, including.

(1) Syntax of atom selection
(2) Speed - much slower than ASCII reads
(3) **FOO.label_seq_id** only legitimate for atoms within a polymer. All others would have to be ``numbered'' by **FOO.auth_seq_id** [Note: Paula Fitzgerald detailed that as this meant HOHs and non-polymer ligands had no ``residue'' number and felt the mmCIF committee had not initially intended this to be the case.]
(4) esd instead of su

Gerard Kleywegt commented on the many practical problems of moving away from PDB atom format to mmCIF atom format. The program O for example would be very difficult to cope with extended chain_id fields.

Many programs manipulate PDB files, and these programs are easy to write, without the in-convenience of having to link to a subroutine library that handles the reading of mmCIF files. This objection is however not serious as any number of research sites load third party software and for example link perl packages into their perl setup. If a simple easy to use mmCIF coordinate subroutine library was readily available, then to write a simple program would not be difficult and would simply require an extra -lcif link at compile time. Gerard's comment that PDB format atom lines are predictable (i.e. one knows where to look for a value) and are unix friendly for the use of shell scripts is a real problem to shift to mmCIF format for in-house structure solution.

mmCIF tools and/or mmCIF format restrictions are required to gain acceptance. These tools should allow equivalents to grep and sort that do not inconvenience users.

Secondly mmCIF can allow for ``special" characters that are also unix un-friendly. The libccif routines do allow for formatting and aligning columns, even when a value is represented as unknown or not applicable with the ``." and ``?" characters. These values may be confusing to unix commands.

It was generally agreed that a coordinate mmCIF file should have as the first column **_atom_site.group_PDB**, i.e. the word "ATOM", and all columns for an atom be placed on a single line, extending the current mmCIF line limit from 80 characters to un-limited size. Peter Keller's libccif routines do allow for formatting and alignment of columns. This would make the files more unix friendly.

The community at large needs to be aware that one package alone shifting to using mmCIF for coordinates will require a conversion program for mmCIF_to_PDB with no lossage as one alternates between refinement and model building for example.

One unanimous request from the software developers presented to the deposition sites was to re-instate the ``prime" character in atom names that is currently replaced with an ``*", especially for nucleic acid and sugar atom names.

Liz Potterton gave a presentation on the problems and a possible solution to interactive or command file commands that are required to select atom/residue and their ranges as input to working crystallographic programs (see http://www.yorvic.york.ac.uk/~lizp/atom_selection.html).

5. External Reference Files (e.g. Refinement Dictionaries)

It is desirable to have a standard format for the ERFs which describe the expected geometric properties of common macromolecular ``monomers", i.e. the amino-acids, nucleotides, or ligands contained within a structure.

Helen Berman and John Westbrook described the development of standard files for nucleic acids. Their method was outlined as,

```
 Select structures from the CSD
 Get average values for valence geometry
 Get an average coordinate set.
 Save history of source CSD files that contribute to the model}
```

One highlight found for the nucleic acid models is that there is a real difference in the distance and angle values found for sugars in different conformations.

A La Mode is an NDB environment for building models of ligand and monomer molecular components. It is based on queries to CSD, and allows simple analysis of the multiple models (see *A La Mode* at http://ndbserver.rutgers.edu/alamode/index.html, or at for Europe http://ndbserver.ebi.ac.uk:5700/alamode/index.html).

Kim Henrick present a brief view (see http://www2.ebi.ac.uk/msd/Harvest/notes/sld034.htm for some of the slides) of the new CCP4/REFMAC mmCIF refinement data item additions to the mmCIF dictionary. The additions (see "CCP4 proposed extensions" at http://www2.ebi.ac.uk/msd/Harvest/ccp4-cif/cif_mm.dic/Groups/index.html) involve both additions to existing categories in the **chem_comp_group** and the **chem_link_group** and new categories. The new categories, **ccp4_lib_group** and **ccp4_chem_mod_group** describe respectively atom type properties (equivalent to CNS protein-allhdg.top files) and functional mmCIF data items that follow existing CNS and CHARMM chemical patches to MODIFY, DELETE, ADD existing **chem_comp** entries.

The CCP4/REFMAC proposal is also to transfer chemical information instructions from command files used in refinement procedures that generate the molecular description restraints to the header of the mmCIF coordinate file. Additions to the **entity_group** and **struct_group** categories allow for a complete description of all linkages and modifications to be held in the self defining mmCIF header.

Alexei Vagin has tools, SMILES2DICT and MAKEDICT, to create ERFs from coordinates, or SMILES strings. Additional tools use the ERFs, plus coordinates to generate lists of restraints for refinement.

Choices have to be made in whether all possible chemical species that are described by a current three letter code are held in the reference file or methods to generate them via a **chem_mod** instruction set. For example Alanine, ALA, how are D-ALA, L-ALA, Nterminal-D-ALA and Cterminal-D-ALA to be described, when they are components of a polymer chain and when they are the free amino acid complexed to a protein.

The CCP4 proposal is to hold some of this information in an extended **_entity_poly_seq** loop structure that will also cope with branched chain polymers;

Proposal for _entity_poly_seq

```
loop_
_entity_poly_seq.mon_id
_entity_poly_seq.ccp4_auth_seq_id
_entity_poly_seq.entity_id
_entity_poly_seq.ccp4_back_connect_type
_entity_poly_seq.ccp4_num_mon_back
_entity_poly_seq.ccp4_mod_id
ASP 1 AA . n/a NH3
VAL 2 AA TRANS 1 .
PRO 3 AA CIS   2 .
ALA 4 AA TRANS 3 D-AMINO-ACID
CYS 5 AA TRANS 4 COO
```

Linkages between different entities are listed in the standard category **_struct_conn**, however linkages common within all instances of a particular entity would be listed in the new definitions;

Proposal for _entity_link

```
loop_
_entity_link.ccp4_link_id
_entity_link.ccp4_entity_id
_entity_link.ccp4_comp_id_1
_entity_link.ccp4_auth_seq_id_1
_entity_link.ccp4_comp_id_2
_entity_link.ccp4_auth_seq_id_2

SS AA CYS  7 CYS  96
```

Instances of other modifications to non-polymer constituents would be held in the structure;

Proposal for _struct_asym_mod

```
loop_
_ccp4_struct_asym_mod.mod_id
_ccp4_struct_asym_mod.auth_seq_id
_ccp4_struct_asym_mod.comp_id
_ccp4_struct_asym_mod.new_comp_id
_ccp4_struct_asym_mod.label_asym_id

DEL-01      263  MAN  .       Bb
```

6. Validation

A session was held on validation and to what extent the deposition centre carries out validation tests and to whom do they make the results available. Tom Taylor gave an overview of the contributing groups that are part of the European validation, CRITQUAL network (see the EC comprehensive validation package at http://biotech.ebi.ac.uk:8400/). These groups have devised a series of geometrical and structure factor tests that may be used to give confidence levels for how well a crystal structure results both fits the observed density and how the geometry agrees with targets. The Uppsala group has selected 25 structures that have structure factors deposited and have undertaken to refine (using CNS) and model build (using O) this set of structures, (see Validation using Structure Factors at http://alpha2.bmc.uu.se/~tom/keith1.html). A careful record of a manual examination of the density maps is then made, detailing for example possible disorder, side chain re-placement, differences in deposited water structure and possible sequence mis-matches. The other members of the CRITQUAL will then be asked to use their methods to give overall and per-residue quality indicators and then all the results will be correlated, to see how well the geometric quality indicators pinpoint problem regions. The group plan to publish their conclusions by the end of this year and at this stage the deposition centres may then apply the recommendations. The CRITQUAL group also includes an NMR component that will recommend a test method to use NOE's and restraints.

There followed a detailed discussion on what can be validated, and how any validation result or confidence level can be used and mis-used. Eleanor Dodson expressed the view that validation is part of structure solution and refinement, and as new knowledge is established, it can initially be used to validate errors, but in the long run it will be used quite properly during the process. This was met with some comment that as more geometrical tests between found and target values are used and felt required then the refinement programs will simply put restraints to match these and one is then in the position of aiming for what could be a less likely correct structure. This is particularly true for Xray structure determinations at resolutions at about 1.3 Ang . Zbyszek Otwinowski pointed out that for refinement of structures at this resolution one may make the choice to weight geometry input high and meet the target requirements or weight structure factor input high and then tend to get the more likely correct structure at the expense of target geometry.

There was universal agreement that deposition of experimental data is important and should be obligatory. There was a heated discussion over how to detect errors, and on basically the problem of what is experimental data and how biased are derived data from experiment data might be, including intensities.

It was felt that upon deposition gross errors (e.g. Ramachandran plots, poor electron density fit) be summarised and presented to the depositor for annotation. It was recognised that the end users of the macromolecular database can have a multitude of views on the data and that reliability indicators are required across all entries and per residue and per atom.

There were conflicting views on what to use as indicators of individual parameter reliability, with Phil Evans and Zbyszek Otwinowski favoring B-values while Garib Murshudov suggested that B-value is an estimation of displacement parameters but not reliability and that B-values are not reliable anyway. There is correlation between B and SU of atomic parameters but it is not 1:1. The use of real space correlation as reliability was questioned as it depends on the map calculated and that this indicator in principle is not an unbiased estimator.

There was some agreement on refinement programs being made to give better statistics and use refinement algorithms which can give standard uncertainties (SU's).

The only conclusion that the deposition centres therefore can make is to collect, store and *derive everything* and give choices to users of a search and retrieval system that will allow individuals to select an appropriate reliability indicator. Meaningful selection criteria can be readily built into a relational database system and a different users preferred, or coached, view point can be accommodated.

The deposition site can accept and store input quality indexes such as **_struct_mon_prot.RSR_all** . Deposited structure factors that contain data items that generate the phases used in the laboratory final map can be used to check and annotate density fit. Finally a data base global consistent quality index can be derived using deposited amplitudes and a standard tool such as SFCHECK.

Geometrical checks can be carried out in a similar manner, entry specific and global standards.

Unfortunately the workshop did not discuss to the same extent NMR validation. There was disagreement whether NMR and Xray structures should both have a common ``quality index'' that was applied for all entries. Janet Thornton suggested that this was essential.

7. STRUCT_BIOL and MOL_ID

A presentation (see http://www2.ebi.ac.uk/msd/Harvest/ejd/index.htm for some of the slides) was given on the EBI's view on how a macromolecular database was to be organised in a relational database and how this structure could be populated. Apart from coordinates, validation data items, and experimental tables, the EBI database is primarily a hierarchy of tables based on consideration of an ENTRY as consisting of:

```
one or more  MacroMolecular Assemblies
    each of which contains one or more  Intact Biological Macromolecules
        each of which contains one or more  Chains
            each of which contains one or more  Residues
                each of which contains one or more  Atoms
```

At each level properties such as BoundMolecules, Features, and derived values are associated tables. In many cases the first three levels are actually identical. The first level represents the molecule(s) found by the experiment and can be generated automatically and presented to the depositor for checking. (see PQS automatic file server at http://pqs.ebi.ac.uk/)

The second level is represented in the current PDB format guide by the SOURCE and COMPND sub-token **MOL_ID**. In the mmCIF dictionary there is currently no direct mapping for the PDB's MOL_ID. Problems with MOL_ID in the PDB arise as it is not used consistently. MOL_ID should be strictly defined and used as a classification of the overall found structure into, where possible, discrete polymer sub-component molecules. The sub-component molecules are biological sub-divisions rather than structure sub-divisions. The simplest example is a FAB/protein antigen complex that is the actual molecular assembly studied and presented in the resulting coordinates. However the complex in a biological sense consists of 2 MOL_ID's, the FAB chains and the protein antigen. The EBI's view is that further subdivision of the FAB molecule into the Heavy and Light chains is a structural sub-division and not a biological sub-division.

The current mmCIF definitions for **_struct_biol.id** and **_struct_biol_gen** allow for author choice in the number of and the content of biol.id's. This does not exactly match the EBI's view that sub-biological structures are not the same as dividing a structure such as a homo-tetramer into sets of chains to derive or compare structural components such as different chain-chain interfaces.

The following proposal summarizes some of the ideas discussed at the workshop for expressing higher level chemical and biological features of structures using the mmCIF ENTITY and **_struct_biol** categories, John Westbrook has now formalised a proposed mmCIF match to the PDB MOL_ID.

John has pointed out a further benefit of the addition of the parent_biol_id, in that sub-biological entities related to the full biological assembly can be enumerated. These sub-biological entities appear to be the closest reasonable match to the spirit of the PDB MOL_ID. In most cases, MOL_ID's are associated with realizable biological building blocks rather than individual mmCIF entities.

These are:
_entity.msd_parent_entity_id where the value of _entity.msd_parent_entity_id identifies the parent entity for cases in which an entity is assembled from a collection entities. Complex entities are assembled using the connectivity information in category **entity_link**.

_struct_biol.msd_parent_biol_id where the value of _struct_biol.msd_parent_biol_id identifies the parent structure for cases in which an biological assembly is generated from a collection biological subunits.

Proposed Example

```
loop_
_entity.id
_entity.type
_entity.src_method
_entity.msd_parent_entity_id
_entity.details
PP     POLYMER   MAN .  'DNA strand'
```

```
DNA     POLYMER   NAT .  'Protein strand'
LIG     THING     SYN .  '15 2,3-dihydroxy-1,4-dithiobutane'
SOL     SOLVENT   .   .  'water solvent structure'

loop_
_struct_asym.id
_struct_asym.entity_id
_struct_asym.details
C    DNA   'DNA 20-mer first chain of duplex'
D    DNA   'DNA 20-mer second chain of duplex'
A    PP    'Protein transcription factor first chain of homo-dimer'
B    PP    'Protein transcription factor second chain of homo-dimer'

loop_
_struct_biol.id
_struct_biol.details
_struct_biol.msd_parent_biol_id
 PPDD1  'Protein DNA/Complex'  .
 DD1    'DNA duplex'           PPDD1
 PP1    'Protein dimer'        PPDD1

loop_
_struct_biol_gen.biol_id
_struct_biol_gen.asym_id
_struct_biol_gen.symmetry
PPDD1    A   1_555
PPDD1    B   1_555
PPDD1    C   1_555
PPDD1    D   1_555
PP1      A   1_555
PP1      B   1_555
DD1      C   1_555
DD1      D   1_555
```

Report from the CCP4/EU 1998 Advanced Methods Refinement Workshop

Eleanor Dodson

Venue: University of York, UK

Dates: Sunday December 12th - Friday December 17th, 1998

Organisers: Eleanor Dodson and Keith Wilson.

Background

The 3D analysis of macromolecules is now an established tool for molecular biologists. The facilities for collecting excellent data from protein crystals are improving rapidly, and in response, new refinement methods are being developed. There are more laboratories exploiting Xray techniques, and many more structures being solved, often by beginners in the discipline. It is important that knowledge of the new techniques is disseminated through the community quickly and accurately. Small hands-on training workshops where the program developers both lecture on the general principles, and supervise students working with their own data are educational for both sides.

To address this need a meeting was planned with funds coming jointly from CCP4 and from an EU Advanced Methods network (CT94-0690) (which covered Italy, Spain, Portugal, France, Germany, Benelux, and the UK). Seven UK students funded by CCP4 and 18 European students and helpers funded by the EU were accepted. There were well over 50 applicants, some from outside the EU. In future we will consider providing one CCP4 funded grant to give an Indian or Eastern European student a place.

Lecturers describing programs were: Paul Adams (CNS), Dale Tronrud (TNT), Isabel Uson (SHELXL), Garib Murshudov (REFMAC), Anastassis Perrakis and Victor Lamzin (Arp_Warp).

In addition we tried to provide a complete background to the method; the lecture program is available on the web at http://www.yorvic.york.ac.uk/~ccp4/workprog.html.

Details of the course

David Watkin again introduced the fundamental mathematics and principles used to fit a model to the experimental X-ray data.

Johan Turkenburg described the nature of the experimental data, how to optimise it, and a few nightmare scenarios on ways to abuse it also.

The next two days were heavily biased towards practical requirements of getting programs running. Dale Tronrud gave an overview of macro-molecular refinement and discussed how to use TNT. Garib Murshudov introduced the use of REFMAC, Paul Adams introduced CNS, and Isabel Uson SHELXL.

Setting up the tutorials was greatly simplified by the availability of the CCP4 interface. Liz Potterton explained how it could be used to transfer data from one format to another, and the students took to it very easily. The Derwent College classroom is now equipped with 20 working SGI O2s and after some helpful tweaking by the York University Computing Service staff, these provided a superb resource.

By the Tuesday evening everyone had calculations running on their own problems.

In addition more general lectures emphasising essential skills for those carrying out refinement were given. Kevin Cowtan gave an excellent and well-illustrated lecture on maps, map bias, and their expected features. Tom Oldfield went over the geometric and conformational properties of proteins, and showed how these can be used as aids for validation. These are especially valuable when used during map interpretation.

The main topic for Wednesday was the use of Arp_Warp. Arp is useful for finding solvent molecules, or unmodelled parts of the chain. Tasos Perrakis is developing automated chain tracing algorithms which work well with structures where the data extends well past 2A. The scripts provided attempt to automate the whole procedure of rebuilding after refinement, and can be used in conjunction with REFMAC or CNS.

The final session for the students addressed questions of validation against Rfactors (Ian Tickle) and the importance of deposition and of data-bases. Fortunately the departmental seminar on the Friday was given by Geoff Barton and Kim Henrick from the EBI, and they helped broaden the perspectives of the students.

In addition these workshops give an excellent opportunity for the lecturers to describe their current research interests and to gain valuable insights from each other. Isabel Uson described the progress towards ab initio phasing; Dale Tronrud his ideas on joint refinement of related structures, Randy Read the important contribution of experimental phasing to the process of refinement, Garib Murshudov the progress towards parameterising thermal motion within crystals, Paul Adams future developments to CNS, and Kevin Cowtan the results from calculating eigen values and vectors for individual parameters of a least-squares refinement.

On the Thursday the students requested a Q&A session, where they put the lecturers in the dock. This was well organised by Fabricia Fusetti, and gave a chance for various issues to be raised. One of the most persistent is that of designing restraints for ligands. The programmers present were all very conscious of the need for simple procedures to be available.

The York Christmas party was combined with the end-of-course disco and some warm close-bonded interactions were observed.

Problems of the course

It is always difficult to select the participants, and doubtless some terrible injustices were done. There is a huge demand for in-depth teaching, but it does effectively take out at least a week of the lecturer's lives, and can only be done at occasional intervals. Being somewhat disorganised Eleanor failed to hand out a questionnaire so she has no real idea what the student complaints were! However they seemed happy, and not to be-grudge working so close to Christmas.

Acknowledgements

The lecturers were wonderful, and we owe them all a great debt of gratitude. The York Computing Service staff were extremely supportive and again helped enormously to keep the show on the road. The students all worked extremely hard, and squeezed as much as they could from the time available. We are grateful to CCP4 and the EU for funding.

News of the CCP4I Documentation and Tutorials

Maria Turkenburg (mgwt@yorvic.york.ac.uk)
January 1999

CCP4I: Graphical User Interface

Documentation/Tutorials

CCP4I Documentation

With the official release of the CCP4 Interface (CCP4I, for short), there will be a comprehensive package of documentation, written in HTML, and included with the CCP4 release at a level similar to the CCP4 Program Documentation. The documentation as it stands today, is already available in York, where it is being written by me, Maria Turkenburg, and where the Interface itself is being written by Liz Potterton (see News of Progress on the CCP4 Graphical User Interface - CCP4 Newsletter July 1998).

Contents

After an introductory page, the documentation can be viewed with or without frames, to suit everybody's browser and personal preferences. In the frames version, there is a navigation bar on the left, containing a list of all available documents. This list is also shown when you try to enter the frames version in a frames incapable browser. Furthermore, there are navigation buttons, which get you through the whole set of documents one by one.

The first section of the documentation contains General information on the runnings of the Interface, explaining the philosophy and its features. This section is ready for release. I will keep adapting it as new features are added to the Interface.

While testing the Interface Modules and Tasks, I write the documentation for the next sections (describing the Interface Modules and Interface Tasks), adding to and adapting the pieces Liz writes. There is specific information on the Task Window Layout, highlighting the parts the user is most-advised to look at. In addition, these sections have a more detailed description of the Interface and the programs it is running. I have included cross-references to the CCP4 Program Documentation and any other appropriate information I could find, in the form of clickable links.

Certain aspects of the documentation, written together with Eleanor Dodson, have grown into separate entities, directly related to specific problems in crystallographic structure solution. They are: reindexing, twinning, handling of FreeR sets in conjunction with UNIQUEIFY, and choosing the right scaling program (the last one straight from a tutorial by Ian Tickle). There has been a call to make them available in advance of the official release of the Interface. I have created a Specialist Help section in the Interface Documentation, which is now available through my lab web pages.

Self-contained tutorial material written in HTML by other people, has been included in a section called External Special Help. This contains the complete Isomorphous Replacement tutorial from the Bath School (by Ian Tickle) and links to Kevin Cowtan's dm Workshop, Structure Factor Tutorial and Fourier Book. I am now working on including the Molecular Replacement tutorial from the Bath School (also by Ian Tickle), too.

The last section of the documentation contains the Tutorials (see below) and Documentation for Programmers.

Colours

The colours I have chosen, are based on the Interface and CCP4 colours. Starting from the main colour of the Interface as a background for the documents, and with the official CCP4 weblogo included in every page, it needed another colour to balance the scheme. It turned out that the information on colour wheels and colour theory I found on the web, was a great help. That is where the reds come in. A rosy red for the titles, and a dark brown-red for visited links.

Windows

I have put those parts of the Documentation which I consider to be separate, *i.e.* the External Special Help, the tutorials, the documentation for programmers, and the CCP4 program documentation, in separate browser windows. And, to be able to read the tutorials while carrying out some of the actions, it may be necessary to open extra browser windows. This might not suit everybody, but I have tried to make it as painless as possible.

CCP4I Tutorials

A guided tour of the Interface, solving and refining RNAse SA on the way!

The tutorials are intended to be stand-alone entities, representing the various stages of a crystallographic structure solution, and can be worked through separately. At the beginning of every tutorial is a section describing the directories and files needed for that particular tutorial. Pre-prepared files for various stages will be available from $CEXAM.

In these tutorials, actions you are required to perform will be displayed in **bold CCP4bluegrey letters**.

The first tutorial is concerned with setting up and getting organised, and deals with the 'Utility' features of the Interface, *i.e.* the Database, the FileViewer *etc*.

Future plan

In conjunction with CCP4 Program developers, more documentation will be developed on the output of the programs. I will need lots of help with this!

MMIFL

(Macromolecule Structure File Formats in IFL)

Dealing with Diversity in Scientific Image Formats

J Bernard Heymann

Maurice E. Müller Institute for Microscopic Structural Biology
Biozentrum, University of Basel
Klingelbergstrasse 70
4056 Basel, Switzerland

heymann@ubaclu.unibas.ch

Almost every software package for crystallography and image processing defines its own image file format, creating unending headaches for the user. Standardization is a useful approach to lessen the administrative problems associated with format conversion. However, this only remains a practical solution as long as the field remains isolated. In contrast, modern science drives towards large-scale integration of data obtained from diverse sources, requiring ever more conversion between different file formats.

The traditional solution to the conversion problem is a giant program, converting an input file format into a common one, before writing the desired format. While this is a viable approach widely used, it means having only this one program able to deal with all the different formats. In addition, this leads to the proliferation of copies of a data set that needs to be kept in different formats. Furthermore, the programmer writing data processing code must then decide which file format he likes, and often he invents a new format as well.

An effort to rationalize the integration of scientific data from different sources, especially microscopic image data, developed into a database prototype project called BioImage. One of the issues that had to be resolved, was which file format or formats would be adopted. It was clear that some sort of conversion utility would be needed for a variety of file formats, including the crystallographic formats, CCP4 and MRC, as well as other formats developed for single particle analysis by electron microscopy (such as SPIDER) and confocal light microscopy (such as BioRad). To write yet another conversion program in the conventional style seemed to be a poor solution.

Because the BioImage project also involves the development of 3D visualization in collaboration with SGI (Silicon Graphics Inc.) [1], it was logical to look at the way file format handling is done on SGI computers. It turned out that SGI's Image Format Library (IFL) provides a convenient way of specifying file formats once for multiple uses, even in existing programs not specifically written for some particular formats. The library allows specification of a large number of image properties or attributes, providing a rich set of options to deal with existing and new file formats. This is therefore a good basis for more specialized formats such as those for molecular structures and crystallography.

The ease with which new formats can be specified in IFL was and is exploited to provide extensions covering the relevant crystallographic and microscopic image formats. The focus of these extensions was originally macromolecular 2D and 3D data sets, and therefore the set of extensions is called the "Macromolecular IFL" or MMIFL. In particular, an effort was made to provide a mechanism to preserve crystallographic and other information contained in the headers of the new file formats. Here I report on the philosophy of design and progress of MMIFL.

IFL: The Image Format Library

SGI's IFL (part of the ImageVision Library) provides an elegant way to deal with a proliferation of file formats without the need to recompile the source code of applications (Figure 1). The base classes of IFL deal with extracting parameters common to all image formats, thus defining a single foundation for handling all image formats. The particular features of image file formats are specified in subclasses of the generic file classes, each compiled in a DSO (Dynamically Shared Object) and placed in the appropriate library directory (/usr/lib for old 32-bit objects and /usr/lib32 for new 32-bit objects). Each file format must then be "registered" by placing a reference to it in a special file: /usr/lib/ifl/ifl_database (Figure 2). The formats already registered in this file can be listed using the program "imgformats".

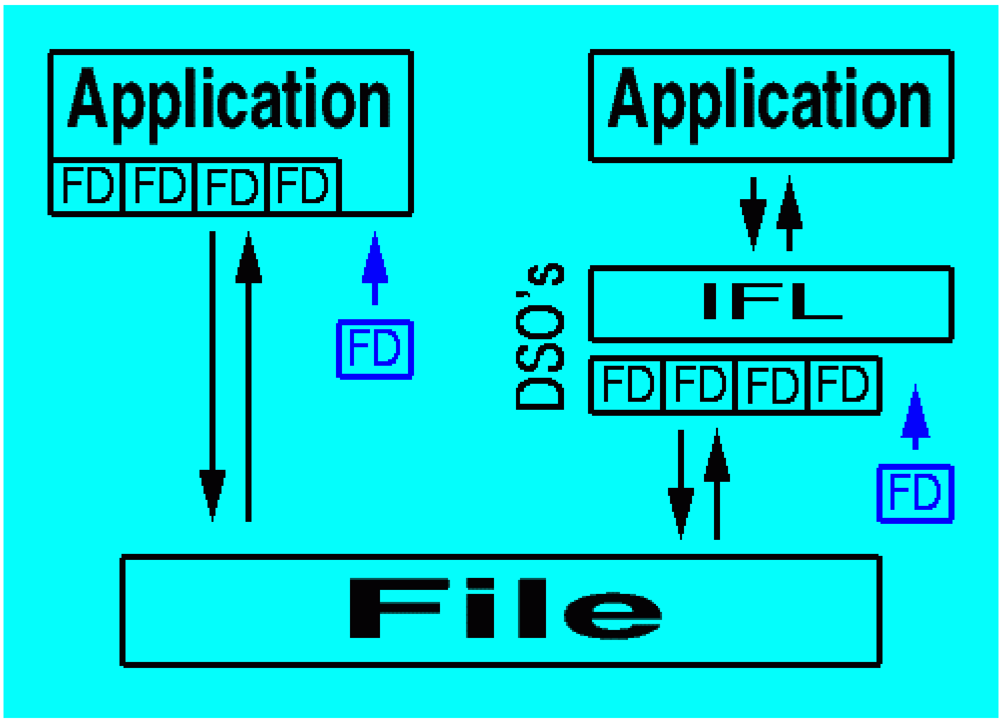
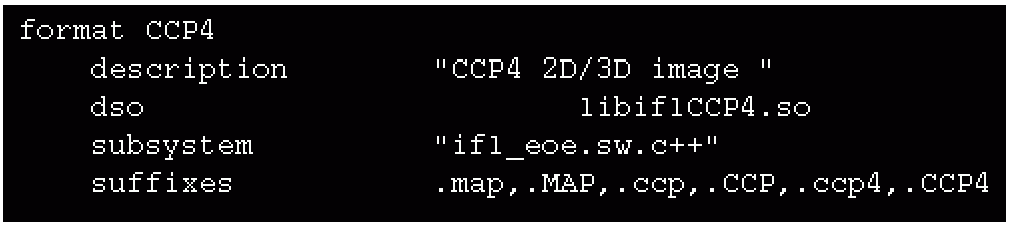


Figure 1: Traditional applications (left) directly include file format definitions (FD's) within the code, thus limiting support for file formats to those the programmer included. Applications using the Image Format Library (right) have access to any format definition defined in the library (implemented as Dynamically Shared Objects on SGI computers). Addition of new format definitions (FD in blue) requires recompilation of the conventional application (left), but only the compilation and addition of a single DSO within IFL (right).



```
format CCP4
    description       "CCP4 2D/3D image "
    dso                       libiflCCP4.so
    subsystem         "ifl_eoe.sw.c++"
    suffixes          .map,.MAP,.ccp,.CCP,.ccp4,.CCP4
```

Figure 2: The registration of the CCP4 format in the file, /usr/lib/ifl/ifl_database, gives the description returned by the program "imgformats", the name of the DSO, and the suffixes identifying the file format.

MMIFL: Philosophy and design

The common design of crystallographic and other scientific image file formats consists of a relatively simple header of constant or varying size, followed by the 2D or 3D data. Some of the information in the header coincides with attributes defined within IFL, in particular the size, the data type (called the mode in crystallography jargon), and statistical information such as the minimum and maximum. The rest of the information need to be accessed via tag-based extensions defined in the file format.

It was decided to create one source code header file, iflMacroMol.h, for the whole group of macromolecule structure file formats. The file formats implemented in MMIFL version 0.5b are shown in Table 1. This source code header file specifies the tags required to obtain information from any of the header formats included. Each image header format is referenced as a structure which can be accessed from a C program using the tag "HEADER". Information such as the space group and unit cell parameters are associated with the tags "SPACEGROUP" and "CELL".

The extensions in MMIFL is written to conform as closely with the format specifications as possible to allow the use of converted data sets in the original software packages. Where information for particular fields in the headers is lacking, suitable defaults tailored for each file format are introduced. Conversions from little-endian and VAX-type floating point numbers are intended to be done automatically, although the detection of byte order is not trivial for all file formats. As MMIFL progresses, these issues will be addressed to provide a clean and simple interface for file format handling.

Table 1: Scientific image file formats of MMIFL

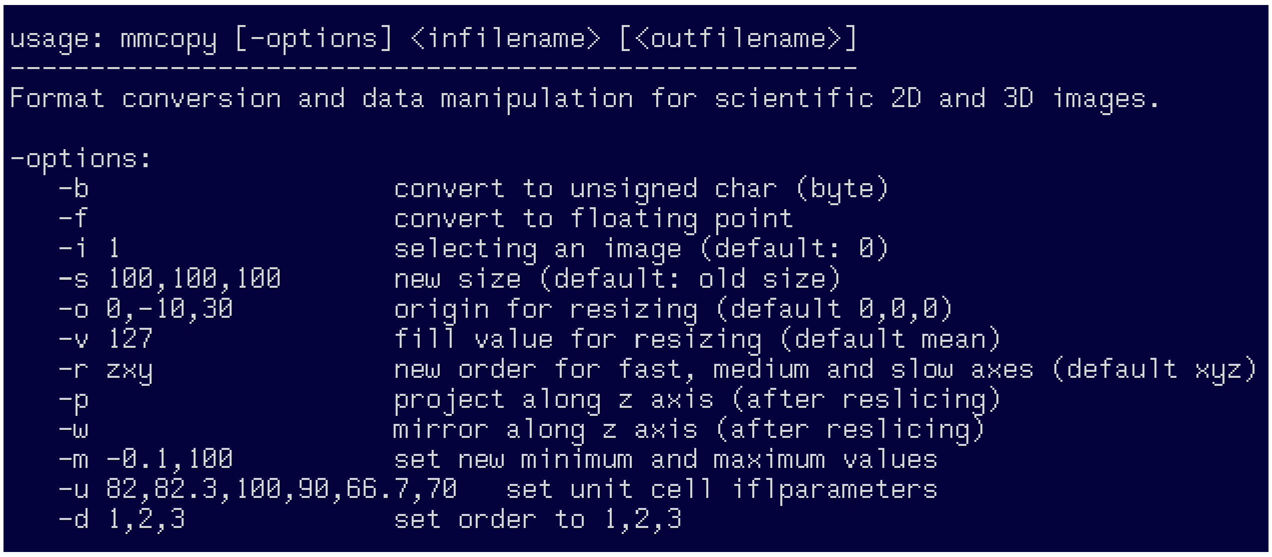
Format	**Field(s)**	**Description**
CCP4	XRD, EC	Widely used format in X-ray crystallography
MRC	EC, EMIP	Widely used format in electron crystallography
GRD	EMIP	Basel in-house format for 3D reconstructions
BRIX	XRD, EC, M	Map format for the program "O"
MFF	XRD, EC, M	Map format for the program "What If"
EM	EMIP	Single particle analysis package from the MPI, Martinsried
SPIDER	EMIP	Single particle analysis
IMAGIC	EMIP	Single particle analysis
BIORAD	CLM	Format for confocal data acquisition from BioRad
DI	AFM	Format for AFM dat acquisition software from Digital Instruments

XRD: X-ray diffraction
EC: Electron crystallography
EMIP: Image processing for electron microscopy
CLM: Confocal light microscopy
AFM: Atomic force microscopy
M: Model building

MMCOPY: Yet another conversion program: But with a few twists

IFL and MMIFL offers a set of file format definitions available to the programmer, in particular in MMIFL mechanisms to use crystallographic and other scientific information in the file headers. To illustrate its utility and provide a useful application at the same time, a simple conversion program was written called "mmcopy". The intention with this program was to be able to convert between different formats seemlessly without losing important header information. Crystallographic parameters are read if present, and written if the target file format includes them. If no such parameters are derived from the input file, defaults are generated to be written into a target file. Different file formats may support different data types, and mmcopy attempts to convert the input format into an appropriate data type for the output format.

Because of the facility with which different formats can be accessed using MMIFL through mmcopy, several manipulation options were added (Figure 3). These include explicit format changes to byte or floating point, selection from multiple images in the input file, resizing, reslicing, and changing header information. Many of these options arose from a need to prepare files for input to other programs, such as the BRIX format for O and the MRC format for image processing with the MRC package.



```
usage: mmcopy [-options] <infilename> [<outfilename>]
-----------------------------------------------------
Format conversion and data manipulation for scientific 2D and 3D images.

-options:
   -b                     convert to unsigned char (byte)
   -f                     convert to floating point
   -i 1                   selecting an image (default: 0)
   -s 100,100,100         new size (default: old size)
   -o 0,-10,30            origin for resizing (default 0,0,0)
   -v 127                 fill value for resizing (default mean)
   -r zxy                 new order for fast, medium and slow axes (default xyz)
   -p                     project along z axis (after reslicing)
   -w                     mirror along z axis (after reslicing)
   -m -0.1,100            set new minimum and maximum values
   -u 82,82.3,100,90,66.7,70   set unit cell iflparameters
   -d 1,2,3               set order to 1,2,3
```

Figure 3: Typing "mmcopy" alone gives brief instructions for usage.

How I use MMIFL currently

Direct access to the formats defined by MMIFL facilitated my work in electron crystallography and visualization for the BioImage project. Because a standard program on SGI computers such as imgview uses IFL, it is now a simple matter to directly look at a map (Figure 4 left). In addition, a VRML isosurfacing node developed for the BioImage project also uses IFL, allowing rendering of a map in 3D with the Cosmoplayer plug-in for Netscape (Figure 4 right). The BioImage project requires direct and automatic access to image files, and the program mmcopy has been integrated with the database prototype for converting and downloading image data. Support for interactive isosurfacing in VRML is also used to provide 3D visualization within the context of the database.

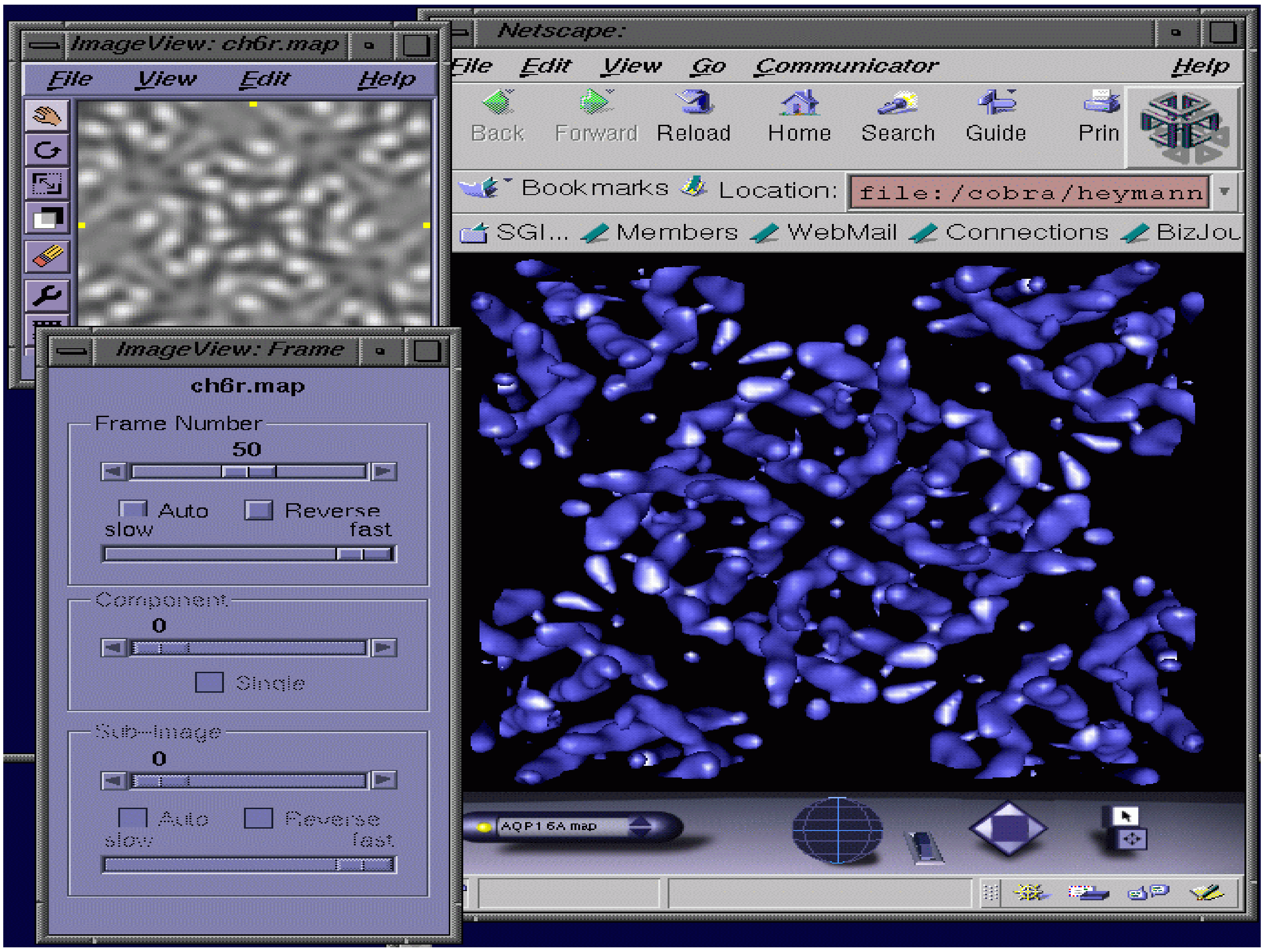


Figure 4: Specification of the CCP4 map format as part of MMIFL allows the display of maps with the program "imgview" (left) and an isosurface rendering in a VRML world (right). In imgview every frame (i.e., slice in the z-direction) can be viewed in a 3D data set. The isosurfacing in the VRML world uses a new experimental node developed in the BioImage project and is available from SGI's developers program. The map shown is that of the red blood cell water channel, aquaporin 1, obtained by electron crystallography at 6 Å resolution [2].

The Future of MMIFL

It is clear IFL offer a mechanism to deal with image file formats in an almost transparent way. However, its limited availability for SGI and Windows computers is a serious hurdle to widespread acceptance. Several avenues will be explored to overcome these restrictions. In the meantime, MMIFL will be further extended on SGI computers.

Availability

The **MMIFL** library was developed within the BioImage Project to facilitate integration of scientific and non-scientific visualization and manipulationtools into a biological image database. The extensions are limited to SGI computers and were tested on the O2 and Octane and require IRIX 6.2 and IFL 1.1.1 or later versions. Programs and utilities using IFL (such as imgview and imgcopy) can be found in the ImageVision package.

The current package is available as a gzipped and tarred file at:

http://www.mih.unibas.ch/Bioimage/mmifl_0.5.tar.gz

Documentation on MMIFL is available at:

http://www.mih.unibas.ch/Bioimage/iflMacroMol.html

Another link with a mailing list can be found at SGI's developer program site:

http://www-devprg.sgi.de/devtools/tools/MMIFL/

Acknowledgments

I wish to thank J.J.Pittet from SGI for initial support and discussions on IFL and A. Engel for his support and discussions. Work on MMIFL and the BioImage project was supported by the European Union through grant PL 960472.

References

1. Pittet, J. J., Henn, C., Engel, A. and Heymann, J. B. (1999) Visualizing 3D Data obtained from Microscopy on the Internet. *J. Struct. Biol.* (in press).
2. Walz, T., Hirai, T., Murata, K., Heymann, J. B., Mitsuoka, K., Fujiyoshi, Y., Smith, B. L., Agre, P. and Engel, A. (1997) The 6 Å three-dimensional structure of aquaporin-1. *Nature* **387**, 624-627.

Recent improvements to the `dm' package

Kevin Cowtan

The CCP4 `dm' package is widely used for phase improvement calculations, largely because of its automation and ease-of-use. Over the past year a number of improvements have been made, in terms of the underlying techniques, as well as the user interface. The most significant of these improvements are described here.

1. Phase error estimation

In the past most of the attention in density modification methods has been devoted to the modifications which are applied to the electron density map. However this is only one part, and possibly not the most critical part, of the process. Once the map has been modified it is back-transformed to give rise to a set of modified magnitudes and phases. These phases will subsequently be combined with the initial phase probability distribution in order produce an updated phase probability distribution.

For this phase combination step to take place, the modified phases must be replaced by a probability distribution. The two distributions may then be multiplied, to provide a new (hopefully sharper) distribution. To convert the modified phase to a phase probability distribution, the error in the modified phase must first be estimated. This is a problem which has already been tackled for phases from incomplete models by the sigma-a method (Read, 1986), by which the error in the phases is estimated from the difference between the observed and modified magnitudes.

For the combination of phase probability distributions to be valid, they must contain information from independent sources. In density modification calculations however, phase information from the experimental phasing is combined with phase information from the modified map, which itself is calculated from the experimental phasing. Clearly the two sources of information are not independent, thus the existing phases are reinforced and the phase errors underestimated. The result is that the FOMs from density modification calculations are seriously overestimated. The resulting bias cripples subsequent density modification procedures, and can lead to actual errors in the model if used for phased refinement.

The solution to this problem is to obtain the density modified phases from a source which is independent from the initial phases.

1.1. Reflection Omit

The problem of phase bias was initially tackled in `dm' using the reflection omit method (Cowtan, 1996). Since experimental phasing is calculated on a reflection-by-reflection basis, the phase estimates from individual reflections are independent. Therefore if the density modified phase for a particular reflection depends only on the experimental phases from other reflections, then the density modified phase will be independent of the initial phase from that reflection.

The reflection omit scheme is therefore constructed in the following manner. A set of reflections are set to zero. A map is then calculated. Density modification is applied to the map, and it is back-transformed. The modified phases are then stored only for those reflections which were not used in the initial map. This process is repeated, omitting a different set of reflections, until modified phases are available for all the phases.

This method is successful in reducing the phase bias, but it has two disadvantages:

1. Each density modification step must be performed many times, making the calculation very slow, and often impractical for averaging calculations.
2. Omitting large groups of reflections introduces noise into the map.

1.2. The perturbation-gamma

Abrahams (1997) suggested an alternative approach to the problem. If the dependence of the modified map on the initial map is linear, that dependence may be removed by subtracting the initial map, scaled by some factor gamma, from the modified map. The dependence between the initial and modified maps may be calculated by a simple theoretical argument for solvent flattening and averaging calculations.

Histogram matching and other density modifications present more difficulties. Firstly, histogram matching is a non-linear modification, and so it is not necessarily clear that the approach will work at all. Secondly, attempts to calculate a theoretical value for gamma fail consistently.

An alternative approach has been developed in `dm' version 2.0. In this approach, two initial maps are calculated, one with the current data, and one with the current data plus a small `noise' signal. Density modification is applied to both of the maps, and the resulting map coefficients are compared. The level of the noise signal in the modified map as a fraction of the initial noise signal provides an accurate estimate of the linear dependence between the initial and modified maps, and thus of gamma. The noise-free map may then be corrected using this value of gamma. An additional benefit of this approach is that gamma may be estimated for subsets of reflections, or as a function of resolution.

This method has been applied to histogram matching. In practice the linear approximation is sufficient and the resulting phases virtually unbiased. Before bias correction it was thought that histogram matching was a less powerful (but complementary) approach to solvent flattening. Once proper bias correction is introduced, it becomes obvious that histogram matching is actually a far more powerful phase constraint than solvent flattening, but had previously appeared weaker because it was more subject to bias.

The perturbation-gamma correction is much faster than the reflection omit approach, and introduces no noise into the resulting maps. As a result it is now the default method for all density modification calculations.

After a single cycle of density modification, the phase errors estimates are found to be as good as the initial data. However each cycle of density modification introduces correlation between phases, so after further cycles of density modification the phase error estimates deteriorate. However, phases from 3, 5 or 7 cycles of density modification may still be used for phased refinement in `refmac' by careful use of the `refmac' phase blurring parameter.

2. Multi-resolution modification

Multi-resolution modification was first employed in version 1.8 of the `dm' package to exploit the fact that electron density histograms have been predicted over a wide range of resolutions. Solvent flattening and histogram matching constraints are therefore applied at two different resolutions: A low resolution map is initially calculated from a set of reflections truncated to the lower resolution. This map is modified by solvent flattening and histogram matching using the electron density histogram at that resolution. The resulting map coefficients (which extend to higher resolution) are averaged with the initial map coefficients. The new map coefficients are then used to calculate a higher resolution map. This map is modified to produce a map which is more consistent with the density constraints at both resolutions. This technique has provided a small but significant additional improvement over existing methods over a wide range of test cases, and is now widely used for real problems as well.

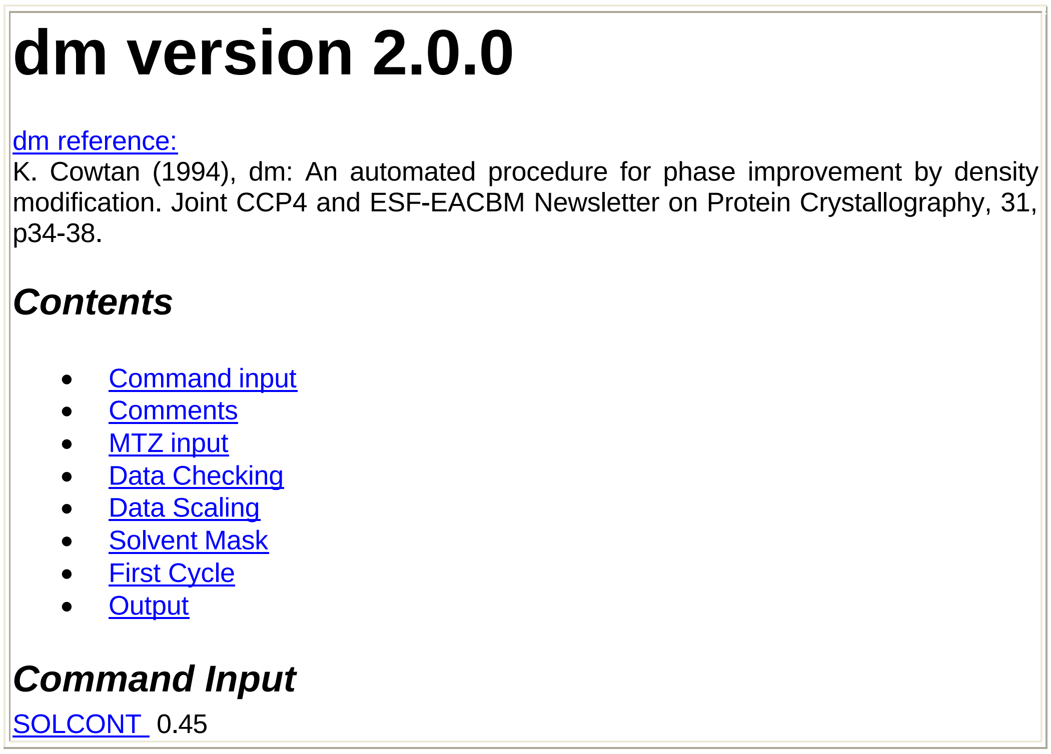
3. Mask refinement

`dm' contains facilities for the automatic calculation of both solvent and averaging masks. Earlier versions allowed the user to provide their own mask, which was fixed for the whole calculation, or to allow the program to calculate its own mask. In the later case, the solvent mask would be refined at each stage, but the averaging mask would remain fixed.

In version 2.0, complete control is provided over mask calculation. Input masks may be provided for the initial stages, and then recalculated as the calculation proceeds. Alternatively, auto-masks may be calculated as often or as rarely as required.

4. HTML log files

From version 2.0 the `dm' log-file is output in HTML format, to be read in a standard web browser. A contents section at the top of the log-file provides links to important parts of the calculation. The command file is linked directly back into the documentation. Extensive commentary is provided on the user command input to help detect errors and improper use of the program. A section from the logfile is shown in Figure 1.



dm version 2.0.0

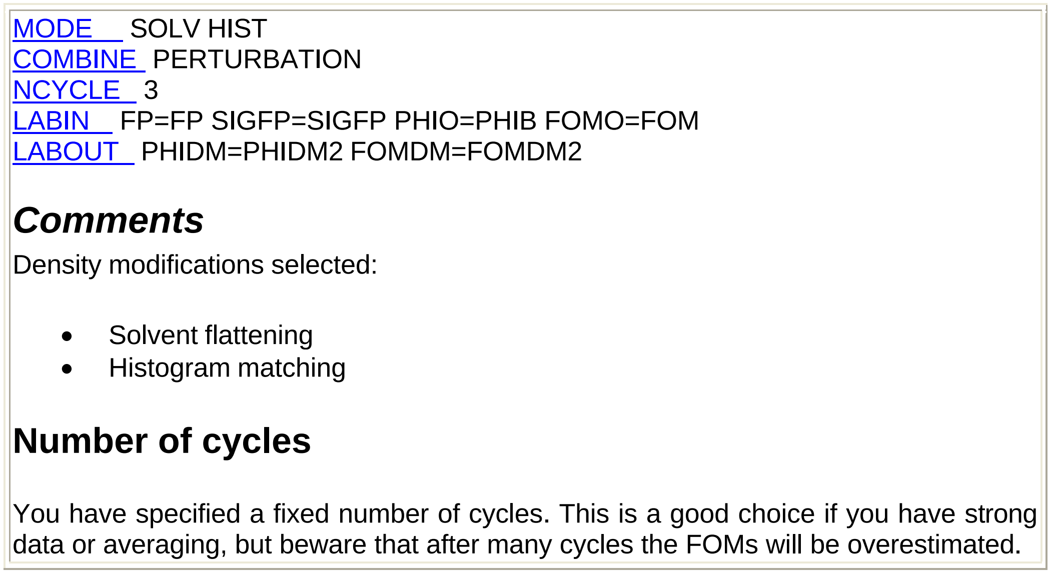
dm reference:
K. Cowtan (1994), dm: An automated procedure for phase improvement by density modification. Joint CCP4 and ESF-EACBM Newsletter on Protein Crystallography, 31, p34-38.

Contents

- Command input
- Comments
- MTZ input
- Data Checking
- Data Scaling
- Solvent Mask
- First Cycle
- Output

Command Input

SOLCONT 0.45



```
MODE     SOLV HIST
COMBINE  PERTURBATION
NCYCLE   3
LABIN    FP=FP SIGFP=SIGFP PHIO=PHIB FOMO=FOM
LABOUT   PHIDM=PHIDM2 FOMDM=FOMDM2
```

Comments

Density modifications selected:

- Solvent flattening
- Histogram matching

Number of cycles

You have specified a fixed number of cycles. This is a good choice if you have strong data or averaging, but beware that after many cycles the FOMs will be overestimated.

Figure 1

One innovation is the use of a Java applet, JLogGraph, to display log-graphs in-line in the HTML document. This saves starting up the Loggraph utility, although this may still be used for advanced formatting or printing. An (inactive) image of a JLogGraph is shown in Figure 2.

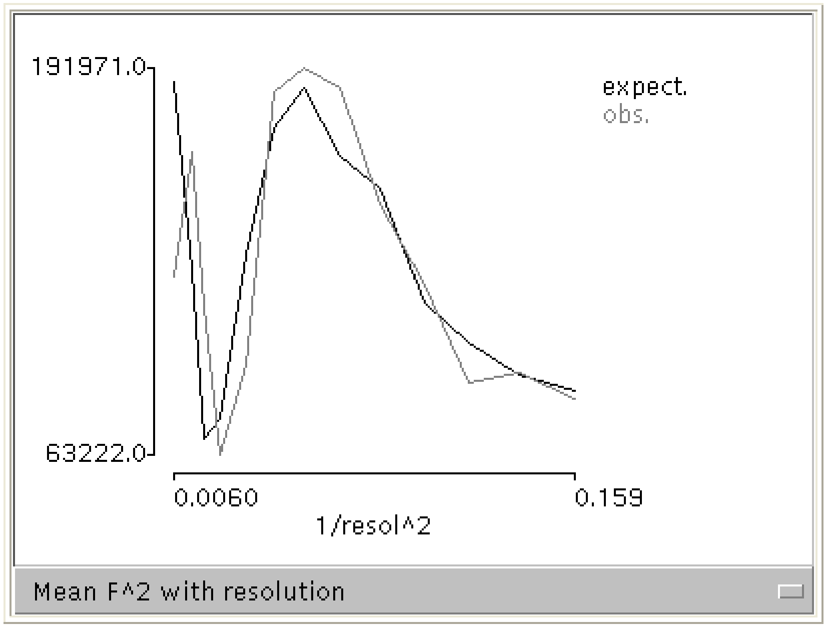


Figure 2

The HTML formatting has been implemented through a set of library routines, so that similar features may easily be added to other programs.

References

- Abrahams J. P. (1997) Bias reduction in phase refinement by modified interference functions: Introducing the gamma correction. *Acta Cryst*, **D53**, 371-376
- Cowtan K. D., Main P. (1996) Phase combination and cross validation in iterated density modification calculations. *Acta Cryst.*, **D52**, 43-48
- Read R.J. (1986) Improved Fourier coefficients for maps using phases from partial structures with errors. *Acta Cryst.*, **A42**, 140-149

Datasets to Maps: A Wrapper for SHELXS and the CCP4 Program Suite

Paul Emsley

Abstract

Chart [1] is a wrapper for SHELXS and the CCP4 Program Suite that creates (density modified) MIR(AS) or MAD [2] maps suitable for density fitting. [3] The starting point is MIR(AS) data sets. Chart terminates with what it considers to be an optimal density modified map. Chart has a minimalist GUI, which should be suitable for novice crystallographers and hopefully can be easily integrated with ccp4i. [4] Chart incorporates a ``divide and conquer'' procedure to optimize various parameters, such as resolution limits or best solvent content.

1. Introduction

The procedure of creating an initial map from MIR(AS) or MAD data is beset with traps for the novice crystallographer (and others not so novice). This program attempts to remove the drudge and pitfalls of heavy atom structure solution. I am aiming for a 90% success rate (*i.e.* Chart finds an acceptable map for at least 90% of soluble cases). Chart may create a final map that is worse than an experienced SHARP user could create. But it will create it much quicker.

Chart first uses the Patterson overlap method implemented in SHELXS to find initial heavy atom positions, running over various resolution ranges (typically more than one would do ``by hand''). Acceptable SHELXS solutions are then exhaustively refined with MLPHARE. Atoms that refine well are combined. A cross-phased difference Fourier is synthesised and peaks checked to search for other derivatives. The solutions are then combined and the phases passed to DM. DM parameters are optimised for the best map. [5]

One needs to specify:

- the Heavy Atom for the derivatives
- the number of residues in the protein
- the file-names for the post-truncate MTZ files
- and whether the structure solution should be MAD or MIR(AS). Chart seems to work for SIRAS data too.

2. *Screenshot*

The Chart main widget is shown in figure 1.

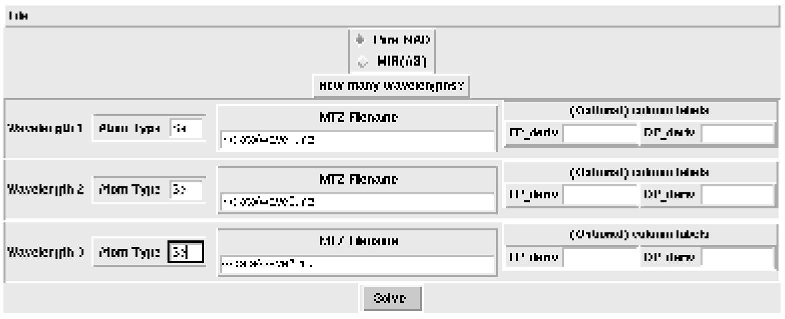


Figure 1: Chart main widget

3. *Execution*

As you would expect for a program that solves structures, Chart uses many programs of the CCP4 Program Suite. Most heavily used is MLPHARE.

Chart creates conventional csh ``com'' and ``log'' files for programs in the CCP4 suite, so that it makes it easy to see what is going on with the refinements and so that you can tweak the refinement after Chart has finished (*e.g.* by adding, for example, new derivative sites or reducing the solvent content). Chart may not infallibly get all parameters correct.

3.1 Implementation

Chart comes in two parts (at the time of writing). chart.scsh is code for the Scheme shell, scsh, and chart.stk is code for the GUI written for STk [6] (Scheme with Tk extensions). Hopefully soon I will be able to integrate the two by using guile, a Scheme interpreter with both scsh shell and Tk extensions. You will of course need both STk and scsh if you wish to use the interpreters, but binary distributions [7] for various sytems will be provided on the web site, for those wishing to avoid these compilation/installation steps.

The current implementation of chart runs CCP4 programs single-threadedly in the foreground. However, Chart can be changed to submit multiple jobs to remote processors [8]. Chart is intensely parallelizable. Basically speed is proportional to the number of nodes or processors. I imagine that one could solve typical 50kD proteins in less than 5 minutes with 10 or so nodes *i.e.* processors).

3.2 Execution time

The execution time is dominated by the time taken to run the CCP4 programs. The time spent in the shell is trivial in comparison.

Molecule	**Size**	**Space Group**	**Notes**	**Time**
Shikimate Kinase	19KD	$P4_12_12$	Dimer in AU	1hr 50 minutes
MIP-1 alpha	7.8kD	$P3_121$		38 minutes

Table 1: Sample execution times. Calculations performed on single processor, single node 333MHz Pentium II processor, running RedHat Linux 5.0, using CCP4 Program Suite 3.4 compiled with GNU Fortran 0.5.24-19980804 g77.

4. MAD

I have not tested the program with MAD data, although I hope to soon. Some changes will be needed of course, but largely the procedure remains unchanged. I need to include an interface to rantan and revise.

5. Future

- Add MAD analysis, rantan, revise.
- Integrate the Chart GUI into the main program
- Add dimer-searching [note: perhaps Kevin Cowtan has done this already with new DM?]
- Add map analysis program (score map according to ``the variance of variance of the map density'').
- Add skeletonisation map scoring.
- Test parallelization.
- Integrate into ccp4i
- Change to guile Scheme interpeter.

6. Where to get it

Chart is copyright Paul Emsley and is released under the GNU GPL. It will be available for free.

At the time of writing, Chart has not been released to the general public. It may have been by the time you read this however, but I don't want to release a first version that is very broken. See the web site for more details and updates:

http://www.chem.gla.ac.uk/~paule/chart

Footnotes:

[1] The name ``Chart'' has a rather contrived expanded form: Characterising Heavy Atoms and Refining Them, it is also the latinization of Χαρτεσ (Xartes), the Greek for ``maps''.

[2] Not at the moment.

[3] In the style described in the article in CCP4 newsletter 35 by L.M. Urzhumtseva & A.G. Urzhumtsev (see http://www.dl.ac.uk/CCP/CCP4/newsletter35/tcltk_software.html).

[4] The CCP4 GUI, work in progress.

[5] Will be, that is.

[6] STk is free software and scsh is available for free to academics but companies pay (although I believe this licence is changing).

[7] I use GNU/Linux to develop Chart, which probably means that that system will be best supported.

[8] These remote machines must have the same directory (typically $CCP4_SCR) at the same mount points (but that's a natural consequence, if one uses automount).

DynDom: A Program to Determine Domains, Hinges Axes, and Hinge Bending Residues

Steven Hayward(steve@chem.rug.nl) Biophysical Chemistry, Department of Chemistry, University of Groningen, Nijenborgh 4, 9747 AG Groningen, The Netherlands.

Introduction

DynDom is a program that determines protein domains, hinge axes and amino acid residues involved in the hinge bending.

You can use DynDom if you have two conformations of the same protein. These may be two X-ray structures, or structures generated from NMR distant contraints, or structures derived from simulation techniques such as molecular dynamics or normal mode analysis.

The application of DynDom provides a view of the conformational change that is easily understood. The conformational change may be quite complicated in detail, but by using DynDom you can visualize it as involving the movement of domains as quasi-rigid bodies.

DynDom allows you to visualize the domain motion in terms of the rotation of one domain relative to another. The program's output is designed for display with RasMol, although it is easy to alter for display with other molecular graphics software. Figure 1 depicts the domain motion that occurs upon substrate binding in citrate synthase as determined by DynDom.

Methodology

DynDom determines domains by looking for clusters of rotation vectors that describe the rotational aspect of the conformational change. These rotation vectors can also be viewed using RasMol as seen in Figure2 for bacteriophage T4 lysozyme. Clusters form dynamic domains which are then analysed for their interdomain motions to give hinges axes. Finally the residues involved in the hinge bending are deterimined.

Usage

DynDom is easy to use requiring the setting of basically only three parameters. Default values exist which have proved to give good results in a number of cases. It is fast, usually taking no more than a few seconds on a Silicon Graphics O2. More information can be found at the DynDom home page where also you can see some results from the application of DynDom to a number of cases. Although the name implies that it can only be applied to domain proteins, it can be applied equally well to smaller regions of proteins such as loops.

References

Main References:

S.Hayward, H.J.C.Berendsen,"*Systematic Analysis of Domain Motions in Proteins from Conformational Change; New Results on Citrate Synthase and T4 Lysozyme*" **Proteins**, 30, 144, 1998.

S.Hayward, A.Kitao, H.J.C.Berendsen,"*Model-Free Methods of Analyzing Domain Motions in Proteins from Simulation: A Comparison of Normal Mode Analysis and Molecular Dynamics Simulation of Lysozyme*"**Proteins**, 27, 425, 1997.

Application:

B.L.de Groot, S.Hayward, D.van Aalten, A.Amadei, H.J.C.Berendsen, "*Domain Motions in Bacteriophage T4 Lysozyme; A Comparison between Molecular Dynamics and Crystallographic Data* " **Proteins**, 31, 116, 1998.

Improved Software for Laue Data Processing

Steffi Arzt

A new program from the Daresbury Laue Software Suite for Laue data processing has been accepted as an associated program with CCP4 and is now available from the CCP4 site.

The program LSCALE scales and normalizes raw integrated X-ray and neutron Laue intensity data to yield fully corrected structure amplitudes. It unites and improves the fuctions of the older programs.

LAUENORM and LAUESCALE formerly used for Laue data processing. The wavelength normalisation curve is calculated with Chebyshev polynomials. It can be derived internally from the Laue data using symmetry equivalent data recorded at different wavelengths or as an alternative by scaling the Laue data to a reference e.g. monochromatic set of data. The wavelength normalisation can be carried out for up to five wavelength ranges. The Chebyshev polynomials handle even undulator data very well.

The program also enables a wavelength and position dependent absorption correction to be calculated by two-dimensional Chebyshev-polynomials.

Harmonic multiples data may be deconvoluted again making use of symmetry equivalent data recorded at different wavelengths. When there is a measured single reflection in the equation system it will be used for deconvolution, but put in the output file with its old value.

The program provides flexibility in the control of the refinement procedure. Data input is via a control file. Refinable parameters can be refined all together in the least squares refinements or one after the other in any order. The refined scaling parameters are saved for re-use as required. On completion of the calculations the program allows a number of diagnostic tables or graphs to be output and examined and, if required, to be saved in hard copy form on an individual plot basis.

LSCALE accepts .mtz or .ge1/.ge2-files produced by LAUEGEN , the Laue intensity integration program (or modified into the required format from other sources). The scaled data may be output as .mtz and/or SHELX files with unmerged or fully merged data.

For more details see S.Arzt, J.W. Campbell, M.M. Harding, Q. Hao& J.R.Helliwell, J. Appl. Cryst. (1999) 32, in the press.

The program documentation can be found on the CCP4 web site at http://www.dl.ac.uk/CCP/CCP4/main.html.
For a copy of the program please contact: ccp4@dl.ac.uk. If you have any problems or comments, write to Steffi Arzt (arzt@esrf.fr) or John W. Campbell (j.w.campbell@dl.ac.uk).

Crystallography Under Linux: A Review of Several Currently Available Programs

Mary A. Canady and John Tate#*
**Department of Molecular Biology, The Scripps Research Institute, La Jolla, CA 92037, mcanady@scripps.edu*
#San Diego Supercomputer Center, University of California, San Diego, La Jolla, CA 92093, jtate@sdsc.edu

Introduction

With the recent release of O for linux [1], the very exciting possibility of fully functional, affordable PC or PowerPC based crystallographic workstations for both graphics and computing applications seems to be a reality. Besides offering the crystallographic community more computing power for their research dollar, the linux workstations also offer word processing [2,3] and graphics applications [4] that are affordable and easy to use. In addition, these linux boxes can be rebooted into the Win95 and MacOS operating systems when needed. The linux community has thrived because the software developers are both proud and excited about their products, sentiments that are mirrored in the developers of such programs as O, Xtalview, X-PLOR and CCP4. Through resources such as the CCP4 Bulletin Board and the various mailing lists, crystallographers have long reaped the benefits of the "open source" idealogy, in which bugs can be reported and fixed routinely. Now, we have the added benefit of an operating system that functions in the same way.

The purpose of this review is to both stimulate an interest in linux [5] and to serve as a quick guide for the installation and use of 7 programs we have chosen to review: O, XtalView, Gl_Render, MolScript (and BobScript), Raster3D, X-PLOR and CCP4. Our emphasis is on the graphical aspect of crystallographic computing, but we have included short descriptions of X-PLOR and CCP4 for completeness (unfortunately we have not had time to install and use other programs that run well on linux such as SHELX and the HKL Package, but have made a list of crystallographic programs not reviewed here below). Indeed, the non-graphical programs are well established on the linux operating system. For the most part, the authors have successfully ported the graphical programs to linux and so differences between how the program runs on linux and how it runs on an SGI will be minor. As stated above, the enthusiasm of the authors seems to ensure that bugs will be worked out. Indeed, the increasing popularity of programs distributed for linux seems to also be an assurance of their stability. XtalView for linux downloads increased 3-fold in 1998 compared to 1997, surpassing the number of SGI downloads.

Installing Linux

One of the major headaches with a constantly developing platform such as linux, is the frequency with which libraries and components change. There are currently two main flavors of linux, based on two different sets of system libraries (the so-called libc5 and libc6 (or glibc2) implementations). Programs compiled on one system may not run on another system. Irritating though this is, this is not an unusual situation for crystallographers, who are well used to heterogenous computing environments: exactly these sorts of difficulties will be familiar to anyone using IRIX systems. However, with careful installation and administration practices, a linux workstation can be as stable and practical as the more expensive unix systems.

While some crystallographers may find installing linux and becoming a part-time system administrator daunting, the current linux distributions (*e.g.*, RedHat [6]) include step-by-step installation documentation, and vendors such as RedHat offer free email installation support (providing the software was purchased rather than downloaded). Most distributions offer installer software which lead a user through the process, and can cope with the some of the more commonly encountered problems. Installation is probably the most daunting task for a new user, but also potentially one of the most informative: most linux users would probably agree that they learned a few things during their installation of linux that they carried over into their everyday unix usage. Users looking for the minimum amount of installation time and the most user support should probably choose the RedHat distribution, since it is one of the most commonly used distributions, ensuring a wealth of pre-compiled software and a very large user base across the internet. One of the major strengths of linux is that wide user base, and one can easily tap into the very considerable experience of general users across the world. Usenet news groups are an excellent starting point: it is almost certain that someone, somewhere has encountered the same problems as you, and equally likely that someone else has a solution.

Probably the most stable and most widely used platform for linux is an Intel-processor based PC. Linux has been ported to practically every possible platform, but the PC platform has been around the longest and is therefore the most bug-free and well supported. RedHat does offer and support versions of their distribution for sun and DEC alpha, it is unlikely that most crystallographers will want to convert machines running OSF, for example, into linux machines. More usefully, an increasingly important branch of the linux community is making linux a reality on Mac/PowerPC machines. With the advent of the new Apple G3, it is eminently possibly to use macs for high power computing applications, while even lower-end macs can be turned into fully functional X-windows terminals without too much effort. However, while support for the PowerPC distributions (linuxppc [7]) is building, we still suggest that Pentium-based machines be utilized for crystallography, in order to ensure the most functionality and support. Despite this warning, we are currently running a PowerPC mac under linuxppc and have found it extremely stable with exceptional graphics.

Software Installation

Several of the available linux distributions have the added advantage of using software management systems. One of the most common is the RedHat Package Manager (RPM [8]), in which all of the files associated with a particular program are bundled into a single archive, known as rpm's. Other distributions also use RPM, while others, such as Debian [9] have similar package-based systems. The software contained in an rpm file can be in either source or binary format, and most software distributed in this way will have been pre-compiled for a variety of platforms - it is often possible to completely install and configure a complex software package with a single command. For a binary package (containing pre-compiled executables and all associated files) the installation process also checks the linux systems for components which are required by the new package. Before any files are placed on your system, RPM checks for the necessary system libraries and that the version numbers are compatible with what you are installing. If required components are not located, the package will not be installed and error messages will tell the user exactly which components are still required. Finally, once a package is installed, a complete record of every file and where it was placed, is maintained by the system. Just as installation is a one line command, deinstallation of every file associated with a particular software package can be removed with a single command.

Although using rpm's greatly simplifies the installation of large software packages, is can also mask some of the significant problems with the installation procedure. Installing new versions of libraries, for example, can generate numerous error messages, about broken dependencies and requirements. Some may be safely ignored, while others mean that you are about to severely damage some other component of your system. Many linux system administrators are learning that trade on-the-fly, and simply do not have the time to learn all the intricacies of a very complete system: bitter experience has shown that even using a system like RPM (or perhaps because of using it) serious damage can be done to the operating system itself. That said, by careful use of rpm's (using options which query or which produce copious logging messages) the chances of removing or overwriting something important can be minimized. For help with using rpm, view the rpm man page or type:

```
rpm --help
```

The particularly useful rpm options are the query options, which will list the dependencies of a package, the files it contains, and any information that came with the package:

```
rpm -ql gimp
rpm -qi gimp
```

Would give the files installed using the gimp rpm and any information distributed with the rpm, respectively. As an alternative to using the pre-packaged software, more "traditional" methods of installation are usually possible, so called 'tgz' files (a .tar file, gzipped), which contain the executables or the source code and the Makefiles to compile them. This allows a more controlled, but potentially more difficult, installation.

We have found that taking a few simple steps during the installation of new software can prevent conflicts later. During the installation, keep notes on what files were created or linked, and try to summarize what files, libraries, links, and additions to the .cshrc (or equivalent) were needed to get the program running. Always make a copy of files which are edited to make a new addition work, such as your .cshrc file. Nothing is more frustrating than getting a program to work only to find that another does not with the current .cshrc file. Adding the offending lines into a short script that is executed when the "new" program is envoked is a way around this. For example, installing the correct fonts for gimp can often be a headache, and I have precluded the possible problems the fonts may cause with other programs by envoking a short cshell script when the program is run (This is just an example, please do not email me with a solution to this problem):

```
#!/bin/csh
xset +fp /usr/lib/X11/fonts/Type1
xset fp rehash
gimp
```

It may be a good idea to take a day or two to install all of the current versions of the software. This "luxury" has only been made possible very recently, since a large number of crystallography programs are currently available for linux. The advantage here is that possible conflicts will be more evident, and each program can be tested after every program has been installed. In addition, down time should be limited to the day (or two) that the programs are installed, without intermittent problems spread over weeks. This type of planned installation should also inspire you to take notes on the dependencies of each program, so that as new versions of software are available you will be prepared to safely install them.

O

O [10,11] is a program that needs little introduction to the CCP4 readership. It is a protein modelling program tailored for the crystallographer written by Alwyn Jones and Morten Kjeldgaard. Its developers have not only facilitated the model building procedure, but have greatly simplified it with their chain tracing tools (bones), assorted features for "smart" as-you-go geometry validation, and the accompanying macros and x-utils programs [12] which extend its functionality. One of the delays in porting O to linux has been the lack of a suitable replacement for the proprietary OpenGL implementation which is standard on systems such as SGI. Relatively recently, a free implementation of the OpenGL [13] library, called Mesa [14], reached a point where it was stable and fast enough to allow work on O to begin. Clearly the graphics capability of the machine on which you run it will greatly affect the quality of your O for linux experience (see Kjeldgaard's O for Linux performance comments page [15]). As for any other program, the faster the PC, the higher-end your graphics card, and the more memory you have, the more pleasant it is likely to be. It is beyond the scope of this review to indicate which graphics cards are the best or what the minimum requirement is, but certainly a 2D graphics card with some 3D-acceleration will greatly improve the performance of any 3D-graphics program. The O for linux author/porter, MortenKjeldgaard, has tested the program on a 120 MHz machine with 2Mb video ram, and says it runs slowly, but I have had excellent results with 166MHz, 4Mb video RAM, 32Mb RAM, and 59 kD (a 540 residue protein). An advantage of linux is that you can add virtual memory easily (at time of installation) by partitioning as much as you like of your hard drive as "swap space", although more physical memory is always preferable (and very inexpensive these days).

Installing O was relatively straightforward. Kjeldgaard offers rpms for O installation on Intel (i386) and PowerPC (ppc) architectures running RedHat or SuSE, as well as .tgz files for Intel machines[1]. In addition, he provides the proper Mesa and Glut (GL Utility Toolkit) libraries needed for Intel machines. Mesa and Glut rpms for the PowerPC architecture are available as well[16]. The current version of O for Linux will run "naturally" with the Mesa 2.6 library, and optionally with the Mesa 3.0 library provided the proper links are made by the system administrator[1]. If you would like to run MolScript as well, you should use the Mesa 3.0 library and make the links. This linking shouldn't be necessary in the future O for Linux releases (but check the documentation!). The O installation files are distributed via a web interface which creates a database of O for Linux users and facilitates downloading.

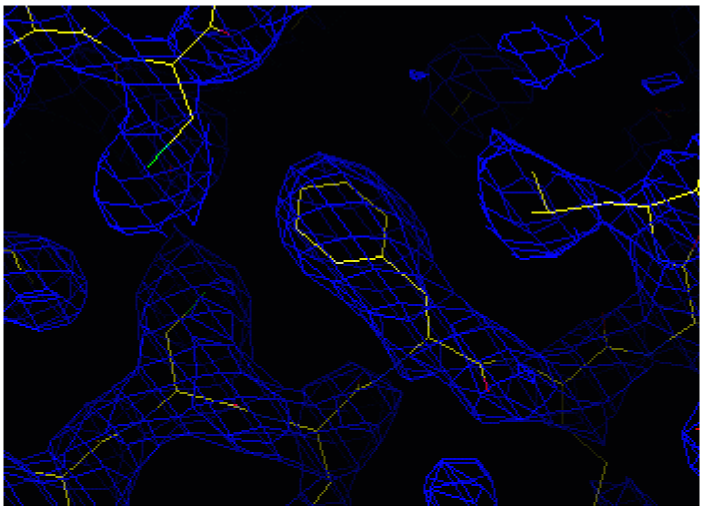


Figure 1. A 2Fo-Fc map displayed using O for Linux, using Mesa 2.6

The performance and appearance of O for Linux are both pleasantly surprising. I had been monitoring Alwyn Jones' O for Linux progress page for quite some time, and the predictions seemed gloomy for a high performance O under Linux. Presumably, hard work on Kjeldgaard's part has been a key factor in this advancement, as it seems he has been working towards this end for at least a year. It seems as though everything works as well as it would on an SGI--the pull-down menus, the fake dials, the molecule movement. Map

(Kjeldgaard reports that using the Mesa 3.0 libraries smoother lines can be obtained). A larger version of this image, showing the full window, can be found here. This figure, along with all other figures, were snapshot from a standard Intel-based PC running linux.

drawing and contouring (using the quick map functions) was done quickly as well. Kjeldgaard claims that the map contouring may still have some problems.

Another consideration, especially for those of us who would like to run O at home, is the limits that a small monitor place on running graphics applications. Most every window manager for Linux will allow the user or system administrator to declare a virtual terminal size which is larger than what can be seen at any one time--the monitor is basically a "window" to this terminal which can be moved by placing the mouse cursor at the edges of the monitor screen. In addition, 2-9 of these virtual terminals are available to the user under most window managers, which allows a virtual desktop that is much larger than with most operating systems. While these two features indeed make the O experience more enjoyable, I found that they were not essential in order to view molecule along with 2-3 menus.

In concluding, there is not much to say about O for Linux, except that it is spectacular! Of course, we were not able to do a rigorous testing of all of O's features, and anyone considering serious modelling should be prepared for possible bugs. Kjeldgaard welcomes and encourages bug reports, and has fixed several in his newest release, O-6.3A-2.

XtalView

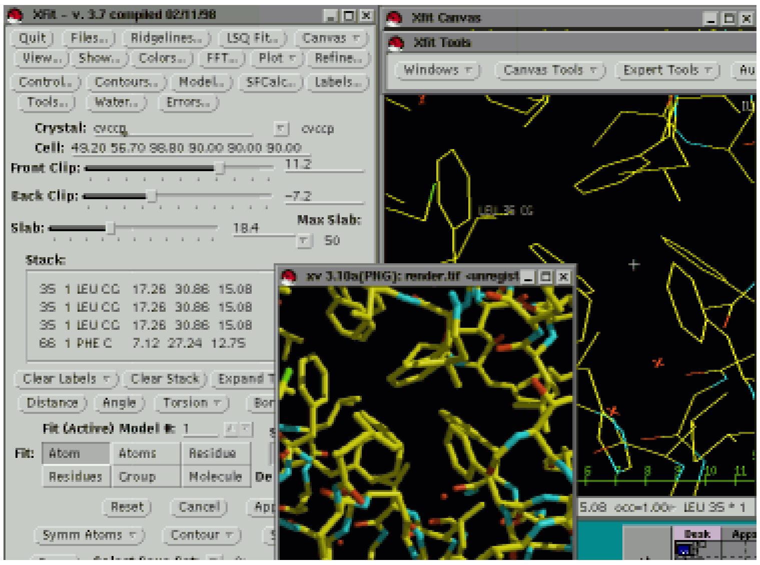
XtalView [17,18] is a program suite that has been running on linux for quite some time. It offers a complete package for crystallography, from MIR and MAD structure determination to model building to refinement. There are several subprograms in Xtalview, and probably the most useful are xtalmgr, which manages and starts XtalView programs concerned with data manipulation and structure determination, and xfit, which is used to build and display molecular models using maps that are generated within the program. Documentation and tutorials for XtalView can be found in several places [17,18,19,20], including a new version of McRee's book Practical Protein Crystallography coming in May, with tutorials and examples, some of which will be available on-line [20]. A complete manual for the program suite is being planned. The program depends heavily on a GUI (Graphical User Interface), and new users can get up and running with the current documentation.

XtalView was written by Duncan McRee at Scripps, and is maintained in his lab and distributed by the Computational Center for Macromolecular Structures (CCMS). Version 3.2 can be obtained [21], with optionally a newer version of xfit (3.7)[18] which may not be compatible with the older XtalView programs. Installation of XtalView is very straightforward.

XtalView interfaces with X-PLOR, TNT, and CCP4 programs, and the CCP4 graphical interface will soon support a "view in XtalView" feature. In addition, XtalView interfaces with Raster3D in order to quickly produce ball-and-stick renderings of the chosen view (shown in Figure 2, lower middle.) The performance of the graphics in Xfit is good, and seems slightly faster than O for Linux. The only problem is the long windows which do not always fit onto one virtual terminal. This can be fixed by increasing the size of your virtual terminal by inserting a line such as this to the Display subsection of your XF86Config file:

```
Virtual     1100 900
```

McRee has had good results using a PC stereo system similar to CrystalEyes for the SGI but much more reasonably priced, and this should be available to users in the near future.



Crystal: cvccp
Cell: 49.20 56.70 98.80 90.00 90.00 90.00
Front Clip: 11.2
Back Clip: -7.2
Slab: 18.4
Max Slab: 50

Stack:

```
35  1 LEU CG   17.26  30.86  15.08
35  1 LEU CG   17.26  30.86  15.08
35  1 LEU CG   17.26  30.86  15.08
68  1 PHE C     7.12  27.24  12.75
```

Fit (Active) Model #: 1

5.08 occ=1.00 LEU 35 * 1

Figure 2. A screenshot of Xfit, the model building program in XtalView. The programs are mostly menu driven, and straightforward, but there are many time saving features of the program that are not obvious to the new user. A larger version of this screenshot can be seen here, and a view of the xtalmgr window is also available.

MolScript

Per Kraulis' MolScript [22,23] has become the standard tool for the generation of publication-quality figures. With the relatively recent release of version 2.0, it has been improved and enhanced and now sports a slew of different image format for output, from postscript to VRML. One of the most attractive features of molscript2 is the interactive GL interface, which allows a user to preview a scene in a window, without the need to render the image using Raster3D [24] or view it as postscript.

Molscript2 and the accompanying molauto program, are distributed as source code and can be easily compiled under linux in their most basic form. The simplest compilation procedure builds a version of molscript2 which can produce postscript, VRML and Raster3D output formats. In order to produce images directly, molscript2 must be compiled with various external libraries, some of which are likely to be pre-installed on any linux system (e.g. JPEG and TIFF libraries) as well as others which are not (e.g. GLut and Mesa). These libraries themselves can be tricky to compile and install, but most are available in RPM format for the most common flavors of linux. The molscript webpage lists the library requirements for building a version of molscript2 which can output images and has the GL interface, and also gives links to where these libraries [25] (or the source code for them) can be obtained.

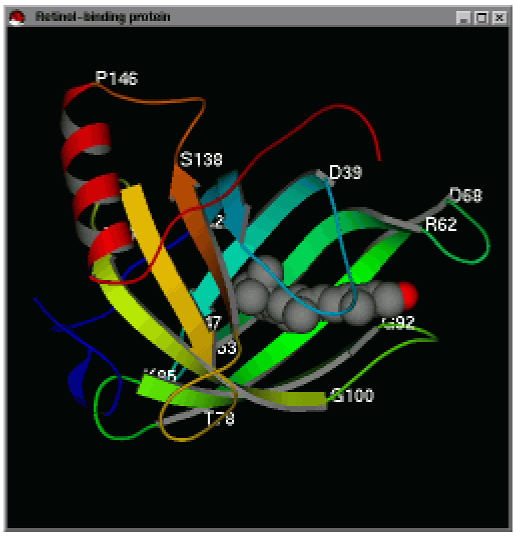


Figure 3. Molscript, with its -gl option, produces real-time movement of nicely rendered molecules under linux. The -gl option will work on linux boxes with the help of the Mesa libraries, but there are mixed reviews on how easy this is to install. A larger version of this image is here.

Although building a version of molscript2 with all of the bells and whistles can be tricky, it is possible to build one which can interactively display a scene using the GL interface. Indeed, the quality of the image and the performance of the interface are surprisingly good, even on relatively low powered linux machines. MolScript with an interactive interface can be built on linux using a Makefile from the molscript website which was contributed by a user. This version of the Makefile uses Mesa version 3.0, and one needs to keep this in mind if both O and MolScript are installed on the same system.   As noted earlier, O should work with this version of Mesa provided the correct links are made. Kjeldgaard has noted that future versions of O will be fully compatible with the Mesa 3.0 libraries, which will obviate the step of making the links.

Raster3D

One of the most common ways to turn the output from molscript into high quality ray-traced images, is using render, from the Raster3D [24,26] package. On SGI machines, render can produce images in SGI's proprietary "rgb" format, or in the more widely used TIFF format. Under linux, the necessary libraries to produce rgb images are not available, but those required for making TIFF images should come pre-installed with most linux distributions. Compared to the complexity of compiling molscript, with its numerous external libraries and version problems, Raster3D can be easily compiled under linux. A Makefile which is suitable for the job is provided with the source code, and builds all of the programs in the Raster3D package without fuss.

BobScript

Although it is based on the older MolScript v1.4 (and therefore lacks the interactive GL interface of the newer molscript2), BobScript [27] is another program which has gained a following amongst crystallographers. A heavily modified version of the original MolScript, BobScript has enhanced colouring capabilities and new drawing functions, but perhaps most significantly, it is capable of incorporating electron density contour maps into a standard molscript plot. The program reads several map file formats directly, including DSN6, brix, X-PLOR and both binary and ASCII CCP4 format maps. Electron density can be built as either the traditional `chicken-wire' mesh, or as solid, surfaces e.g. in cryo-EM maps. The final scene, complete with density, can be output either as postscript or in a form suitable for raster3D.

Although not specifically supported under linux, BobScript requires no unusual libraries or environments to compile or run. Both bobscript and its ancilliary programs can be very easily built from source without modification. Contact the author of bobscript, Robert Esnouf, for details on obtaining the program and licence agreements.   (We do not have his email address at this moment).

GL_Render

The GL_Render [28] program is a nicely constructed front end and interactive viewer for BobScript/Raster3D images written by Lothar Esser. Taking either a BobScript (or MolScript) input file, or the Raster3D format output from either program, it produces a rendered image in a window. As in molscript2, this image can be interactively manipulated (and views can be outputted), but can also be sent for rendering by povray. POV-Ray(the Persistence of Vision ray-tracer) [29] is a freeware general rendering program, widely available and used across the internet. Since it is completely general, it has far more flexibility than the more limited render. With this flexibility comes an extra level of complexity, but if you're willing to learn how to use it, some very nice results can be obtained with POV-Ray.

GL_Render is very useful even if one isn't interested in using POV-Ray. It largely automates the procedure of running and re-running bobscript/molscript, and adds additional functionality by way of extra clipping planes, or the interactive addition of labels to a pre-existing scene. The interactive display rivals that of molscript gl, and could be used for this purpose if molscript gl installation proves difficult.

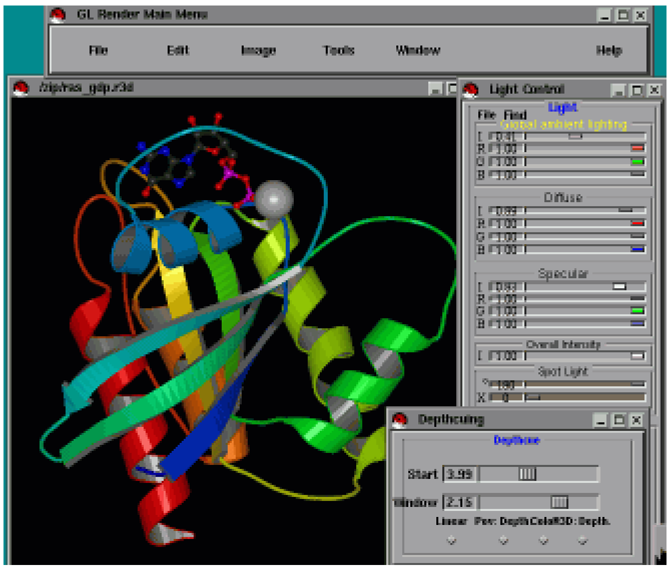


Figure 4. GL_Render interactively displays and renders any object that can be rendered by molscript or bobscript (including maps). Settings such as light and depthcueing can also be interactively changed. A larger version of this screenshot exists here.

The program is distributed only in binary form, but, significantly, one of the four supplied binary formats is linux. The current version is 0.2a, with 0.3a coming shortly. This program runs "out of the box," providing you have an Intel machine running RedHat. Since the program is very new, it is conceivable that versions for other architectures will someday be available. POV-Ray can be obtained as source code from the central povray website [29], but binary versions are available on the net.

CCP4

The CCP4 [30,31] suite is perhaps the most comprehensive collection of crystallography programs available, and as such it can be a complex and potentially difficult job to install it on any system. Linux is listed as one of the supported platforms and therefore the suite *will* compile and run. There are unfortunately certain caveats to this statement. Perhaps most significant are the problems with the actual compilation of the programs. Although previous versions of CCP4 were known to compile using the fort77 compiler, the GNU gcc/g77 compiler is recommended for the latest version. This, like all linux software, is constantly under development, and there are still some known problems with compiling some of the CCP4 programs. To complicate matters further, a newer version of this compiler, known as egcs, is also available and appears to be able to build the majority of the suite. This compiler (or a modification of it) purports to improve the performance of programs pentium computers, so it may be possible to squeeze a little extra out of even old machines using it.

For information and advice on compiling CCP4 under linux, the CCP4 website has a linux page [32]. This site goes into the specifics of compiling the package and gives some comments on the problems which you might reasonably expect to have when building CCP4. With the exception of the X-windows programs, most of the suite should compile and run well under linux. Indeed, given some of the problems which we have had in the past while compiling CCP4 even for the SGI, compilation under linux seemed no more difficult.

X-PLOR/CNS

Like CCP4, X-PLOR [33,34] has been one of the cornerstones of crystallographic software for a long time. A version of X-PLOR 3.851 is available for linux and compiles relatively easily. Along the same lines as X-PLOR, the new program CNSsolve [35,36] is now available and supported for linux. The installation of CNSsolve should be straightforward, since the automatic installation procedure automatically detects the platform on which it is being built and, hopefully, does all the hard work of compilation for you. As for CCP4, there are certain issues surrounding the compilers used to build CNSsolve, but, unusually, there are (currently) no known problems with the installation of the program under linux.

Linux may well be one of the more useful platforms from which to use CNSsolve, since this program makes considerable use of an HTML interface for the construction of scripts. Installation of the HTML interface on a local machine requires the use of a locally accessible webserver, of which linux boasts several. The most widely used webserver on the internet is currently Apache, the free HTTP server based on the NCSA server.

Conclusions

The challenge of porting the major crystallographic programs, including those with cpu intensive graphics, to linux has been met in earnest by the authors of the programs reviewed here. As users, it is our responsibility to faithfully report any problems with the installation or operation of these programs. If a problem seems to be a true bug of the program, and not just a problem with your individual installation or settings, you should contact the program author, but in order to prevent the authors from being overwhelmed with tedious questions perhaps mailing lists should first be used in order to ensure that the problem is a true bug.

It is indeed an exciting time for crystallographic computing, since it seems as though affordable, versatile PC based workstations are finally possible. The reliance on the more expensive Silicon Graphics computers, with the accompanying expense of dial boxes and CrystalEyes, meant a shortage of resources for many crystallography labs. Linux systems, while they may require more time to set up, can be a stably running alternative to the SGI, with the added advantage that work at home can be easily transferrable to work in the lab (zip disks can facilitate this). The individual crystallographer may need to be a bit more in tune with the operating system, in order to troubleshoot possible problems, but the benefit of a fully functioning workstation on one's desk is surely worth this effort. The fact that stereo will probably be available at a reasonable cost and the graphics performance is good makes one believe if there is such a thing as a free lunch! To our knowledge, however, there will probably not be dials for the linux workstations in the near future. Buying a high quality (linux compatible) mouse should help in this area.

There are certainly other crystallographic programs that run under linux (see Klas Andersson's Linux Applications in Crystallography [37] page). Interest in linux for crystallography is increasing, and while we have not covered every program, we hope that this review will help to encourage others to install, use, and review (!) the programs running under linux. We realize that a review such as this is bound to have its shortcomings, and encourage any errors to be reported to the appropriate mailing list.

References

1. O for linux URL: http://origo.imsb.au.dk/~mok/o-linux/

2. Wordperfect for Linux URL: http://linux.corel.com/linux8/index.htm

3. Applixware, fully featured office suite software URL:http://www.applix.com/appware/linux/

4. Gimp, The Gnu Image Manipulation Program URL: http://www.gimp.org

5. Linux master site URL: http://www.linux.org

6. RedHat Software URL: http://www.redhat.com

7. LinuxPPC URL: http://www.linuxppc.org

8. RedHat Package Manager URL: http://www.rpm.org

9. Debian GNU/Linux URL: http://www.debian.org

10. Jones, T.A. and Kjeldgaard, M. (1997) Electron-density map interpretation. *Meth Enzymol* **277,** 173-208.

11. O program URL: http://imsb.au.dk/~mok/o/

12. Uppsala Software Factory (X-Utils) URL: http://alpha2.bmc.uu.se/~gerard/manuals/gerard_manuals.html

13. OpenGL: http://reality.sgi.com/opengl/

14. Mesa : http://www.mesa3d.org/

15. O for linux performance comments URL: http://origo.imsb.au.dk/~mok/o-linux/platforms.php3

16. LinuxPPC Mesa/Glut rpm URL: http://ftp.linuxppc.org/RPMS/MByName.html

17. McRee, D. E. *Practical Protein Crystallography*, (1993). Academic Press, San Diego, CA.

18. XtalView URL:http://www.scripps.edu/pub/dem-web/index.html

19. XtalView Documentation
URL: http://www.sdsc.edu/CCMS/Packages/XTALVIEW/XV1TOC.html

20. Practical Protein Crystallography, Second Edition URL:http://ppcII.scripps.edu

21. XtalView Download
URL: http://www.sdsc.edu/CCMS/Packages/XTALVIEW/xtalview.html

22. Kraulis, P.J. (1991). MOLSCRIPT: a program to produces detailed and schematic plots of protein structure. *Journal of Applied Crystallography* **24**, 946-950.

23. Molscript URL: http://www.avatar.se/molscript

24. Raster3D URL : http://www.bmsc.washington.edu/raster3d/raster3d.html

25. Molscript - additional libraries URL: http://www.avatar.se/molscript/doc/links.html

26. Merritt, E.A. & Bacon, D.J. (1997). Raster3D Photorealistic Molecular Graphics. *Methods in Enzymology* **277**, 505-524.

27. Esnouf, R.M. (1997). An extensively modified version of MolScript that includes greatly enhanced coloring capabilities. *J Mol Graph Model* **15**, 132-4, 112-3.

28. GL_Render (Lothar Esser) URL: ftp://www.hhmi.swmed.edu/pub/gl_render/

29. POV-Ray URL: http://www.povray.org

30. Collaborative Computational Project, N. (1994). The CCP4 Suite: Programs for Protein Crystallography. *Acta Crystallographica* **D50**, 760-763.

31. CCP4 URL:http://www.dl.ac.uk/CCP/CCP4/main.html

32. CCP4 on LINUX URL: http://www.dl.ac.uk/CCP/CCP4/ccp4onLINUX.html

33. Brunger, A.T. (1996). Recent developments for crystallographic refinement of macromolecules. [Review] [81 refs]. *Methods in Molecular Biology* **56**, 245-66.

34. X-PLOR homepage URL: http://xplor.csb.yale.edu/xplor-info/

35. Brunger A.T., Adams P.D., Clore G.M., DeLano W.L., Gros P., Grosse-Kunstleve R.W., Jiang J.S., Kuszewski J., Nilges M., Pannu N.S., Read R.J., Rice L.M., Simonson T., Warren G.L. (1998) Crystallography & NMR system: A new software suite for macromolecular structure determination. *Acta Crystallographica* **D54** 905-921.

36. CNSsolve homepage URL: http://xplor.csb.yale.edu/cns_solve/

37. Linux Applications in Crystallography URL:
http://www.fos.su.se/struc/linux/linuxtal.html

Crystallographic Software for Linux Not Reviewed (our apologies!):

1. SHELX URL: http://shelx.uni-ac.gwdg.de/SHELX/

2. HKL Package URL: http://www.hkl-xray.com/

3. Rasmol URL: ftp://ftp.dcs.ed.ac.uk/pub/rasmol/
(New version--2.6.4 (December 1998)--seems to be easier to compile with 16bit and 32bit graphics)

Other Useful Links:

1. Corel Linux Users Network: http://linux.corel.com/index.htm

2. Google, a good search engine for linux: http://www.google.com

3. Metalab (formerly Sunsite), a warehouse of downloadable software:
http://www.metalab.org

4. Linux HOWTOs: http://metalab.unc.edu/LDP/HOWTO/

5. Slashdot.org, Linux news: http://slashdot.org/

6. SAL, Scientific Applications Under Linux: http://SAL.KachinaTech.COM/Z/2/index.shtml

7 Choosing a Window Manager:http://www.PLiG.org/xwinman/

8. Fortran Compiler rpm (fort77-1.14a-4.i386.rpm):
http://metalab.unc.edu/pub/Linux/distributions/redhat/redhat-5.2/i386/RedHat/RPMS/

News of MOSFLM 6.0

Harry Powell, MRC-LMB, Cambridge.

The recent release of MOSFLM 6.0 contains a number of new features. Two new detector types have been added - the LIPS (Large Image Plate Scanner at ESRF) and SBC1 (Westbrook detector at APS).

The principal improvement is the inclusion of the Fourier auto-indexing routines which have been written by Ingo Steller in the Rossmann group at Purdue University. These are a set of freely available programs written in ANSI C using ANSI FORTRAN FFT routines from the National Center for Atmospheric Research, Colorado; they are also available in the DPS Data Processing Suite for the ADSC detector used at MacCHESS

A few local modifications to the routines have been made which increase their utility; chief among these is that reflections may be selected from multiple images, which is especially useful in the case of weakly diffracting images or where the reciprocal lattice has a principal axis close to the X-ray beam. The incorporation of the routines gives MOSFLM a robust and reliable indexing mode which is similar to that used in Denzo in its effectiveness.

For the first time we have included a protocol in the build for PCs running Linux. The program has been built successfully on a Pentium MMX 233MHz with 128Mb RAM under RedHat release 5.1-2 using the GNU C compiler gcc 2.7.2.3-14 and FORTRAN compiler egcs-g77 1.0.3a-14. MOSFLM builds in around 10 minutes on this system. This method can also be used on a PowerPC machine (e.g. Macintosh) running LinuxPPC Release 4.

Executables for the following platforms/operating systems are available (see below) on the LMB's ftp server.

```
Digital Alphas: Digital UNIX 4.0
SGIs:           Irix 5.3, Irix 6.2, Irix 6.4, Irix 6.5
Intel PC:       RedHat Linux 5.1
PPC Macintosh:  LinuxPPC Release 4
```

Alternatively, these can be built locally (using CCP4 release 3.4 libraries) under the csh or tcsh shell after extracting the directory structure simply by typing "make"; the HOSTTYPE environment variable is checked and Makefiles chosen accordingly. Irix 6.4 and 6.5 are accommodated by setting HOSTTYPE to an appropriate value.

All the builds require that the CLIB environment variable has been set to the directory containing the two libraries libccp4.a and lib_xdlview.a; a customized version of the code for lib_xdlview.a required for a build under Linux has been prepared by Joachim Meyer of EMBL.

MOSFLM can be built either with or without the existing autoindexing code REFIX as supplied by Wolfgang Kabsch. If the source code for REFIX is required, andrew@mrc-lmb.cam.ac.uk should be contacted directly.

The program is available via anonymous ftp from ftp.mrc-lmb.cam.ac.uk/pub/mosflm.

During the recent CCP4 Study Weekend in Sheffield, a number of people made the comment that MOSFLM seems to run more slowly than Denzo when integrating images. This prompted me to do a spot of bench-marking. I took two datasets collected from protein crystals produced in the LMB and ran comparative jobs for both programs on our Digital Alpha 'farm'; this is a bank of processors given over to batch processing single jobs at the same time, so the timings of the programs should not be affected by time-sharing considerations. For each job, the images were copied onto a scratch disk local to the processor being used to minimize the effect of network traffic.

Hardware: Digital Alphaservers, 500MHz processors, 128Mb main memory, 1.5Gb swap. In each case the batch job was submitted 10 times and run independently. The Denzo and MOSFLM command files were written to reflect what a reasonably experienced user of either program might use for *initial* processing of the datasets; no attempt was made to optimize the process, hence the relatively poor agreements between the datasets.

The time taken to convert Denzo .x files to .mtz format has been ignored in the calculations, as it is possible that some users prefer not to use the CCP4 suite for further processing.

Post-processing for both datasets was performed using programs from the CCP4 suite.

Dataset 1

Hen egg-white lysozyme, space group $P4_32_12$, 45 images, Mar 30cm. Resolution range 2 - 60 A, overall R(merg) between datasets on F 0.048, RF_I 0.062, Wted_R 0.053

Dataset 2

Hepatitis B capsid with ligand bound, space group C2, 60 Images, Mar 30cm. Resolution range 6 - 60 A, overall R(merg) between datasets on F 0.095, RF_I 0.129, Wted_R 0.070

		N(obs)	N(ref)	N(merg)	Rmeas	Complete	Mean	Max	Min
HEWL	**Denzo**	42095	27826	8544	0.125	97.4%	2'16.5"	2'14"	2'22"
	MOSFLM	47956	28293	8554	0.117	98.6%	2'40.5"	2'33"	2'43"
HepB	**Denzo**	168104	87028	61308	0.246	100%	6'57.2"	7'15"	6'44"
	MOSFLM	198425	87545	61589	0.256	100%	6'50.2"	6'59"	6'39"

The conclusion from this is that the two programs perform comparably as regards speed, and this should not be a deciding factor in choosing which program to use.

It has been suggested that another explanation for the observation is that MOSFLM might run more slowly on machines with limited RAM due to excessive paging, as it keeps two images in memory simultaneously as opposed to one for Denzo. This seems unlikely to provide a complete answer as the following test indicates. I ran tests on an Intel PC (Pentium MMX, 233MHz) running Linux with different amounts of RAM, all frames on an internal hard disk.

Using the lysozyme dataset above, I obtained the following times for complete integration, again for 10 jobs each;

```
128Mb: mean 9'02.3",  min 8'58",  max 9'05"
32Mb:  mean 10'44.4", min 10'39", max 10'47"
24Mb:  mean 11'19.5", min 11'16", max 11'27"
```

MOSFLM would not run on a PC with only 16Mb of RAM. Obviously it runs more slowly with less memory, but the difference is not as great as the 5 - 10 times slower that had been mentioned at the Sheffield CCP4 workshop.

Harry Powell

Writing CCP4 scripts in PERL

E. Courcelle and J.P. Samama

Institut de Pharmacologie et Biologie Structurale, IPBS, 205 route de Narbonne, 31077 TOULOUSE, Cedex, France

e-mail : manu@ipbs.fr

I. Introduction

Handling and treatment of crystallographic data commonly requires several programs. They are provided by the ccp4 project and by other sources. The treatments include:

- data formats translation programs
- data treatment programs
- normalization

Although possible, interactive work is rather fastidious, inefficient and error prone, as every program requires the reading and writing of several data, log, and command files. It thus seems more appropriate to write scripts, in which all calls, input or output file names, etc., will be coded so that a single script execute all the programs. However, large scripts end up looking alike for any informatic program: according to the way they have been written, they can be more or less difficult to understand, and thus difficult to maintain or modify. We have tried to implement a tool which would permit the users to write large but nevertheless clear and easy-to-maintain scripts.

II. Choosing a scripting language

It is most common, when writing scripts which execute scientific programs, to use traditional unix shells, such as sh, ksh, csh or one of their dialects. This is a good and well-known approach, when the objective is only to chain the execution of several programs. However, it appears that data treatments now make it necessary to use more and more complex scripts, invoking not only CCP4 programs, but also programs from other sources (particularly xplor or cns). An example of those complex scripts could be for instance running a given program in a loop and exiting from the loop only when a given condition is fulfilled. The execution of the program should be considered as a black box, the behaviour of which may be described as:

- reading data
- doing something with the data, according to another input file (keywords, xplor input file, etc).
- writing output files
- telling the user what has been done

The output files of a given program are often the inputs of another one, defining some piping mechanism.
Another case arises when the script should analyze the log file, whether to extract some

results and print a summary, or to take a decision based upon those results. It is thus important to be able to easily extract the information from ascii files. In both cases, the file will be treated inside the script, and will probably not be used later. Thus its name does not have to be chosen by, or even known to the user; however one must define an algorithm for the script to choose unique file names, avoiding any conflict. It is clear that this "black box concept" is most easily implemented using object oriented programming. One thus needs a scripting language with two main characteristics:

- efficient ascii file treatment
- support for object oriented programming

Neither of these two characteristics is correctly implemented in traditional Unix shell languages: without even speaking of object orientation, the users have to use external unix programs, like grep, awk, etc. to extract information from ascii files. However, starting from version 5, the Perl language is nicely adapted to these constraints:

- regular expression matching, as well as other facilities, are implemented as part of the language. Thus, one does not have to call external utilities to analyze ascii files.
- The language is extensible, so that it is possible to write Perl modules. Those modules may support data encapsulation, inheritance, methods... thus it is possible to implement object oriented programming in Perl.
- Efficient memory management, a very useful functionality, associated with data encapsulation, for efficient object programming.

III. The occp4.pm module.

A Perl module was written which allows the user to write Perl scripts in an object oriented way. This module is loaded through a classical require Perl instruction (cf. example)

Conception of an object:

The constructor:

Before the execution of a program, an object, whose class is called occp4, is created (cf example). The parameters of the constructor are:

- The name of the ccp4 program
- The pairs (logical name, file name) that the program will use for its input/output.

As noted previously, in many cases, the files created or read by the programs do not have to survive after the script: they can be treated as temporary files, so the user does not have to worry about the choice of a filename. In those cases, the special name 'CHANA' (acronym of 'CHoose A NAme for me') causes the object to compute a unique temporary name for this file. The algorithm involved is implemented by the occp4 module, in such a way that names are generated in an "orderly" manner. This helps when debugging the script. The temporary files are deleted by the destructor of the object, unless otherwise specified.

The member functions:

The behaviour of the object is then controlled by the member functions; the most important functions are quoted here:

- iofiles, iofildel to specify the input-output pairs of (logicals, file names), after the constructor has been called, or to retrieve the file name, if chosen by the system.
- keywords, keywrep, keywdel to generate and modify a list of pairs of (keywords, values) controlling the behaviour of the program, as described in the documentation.
- input_src, input_string, input_file: these functions allow the user to let the object read its control script not only from the list of (keywords,values), but also from a string or from a file: they are particularly useful when dealing with programs which do not belong to ccp4.
- logfile allows the user to know the name of the (temporary) logfile computed by the object. It is also possible to force the object to use another log file. Forcing all objects to share the same logfile allows the user to maintain and keep an unique logfile for the whole script.
- debug controls the messages printed by the object at the time of key moments in its life (mainly when the program is executed or when the object is destroyed).
- run starts the program. The execution code of the program is returned, so that error handling is very easy to achieve.

The destructor:

The destructor is called by the system, as explained below. It essentially removes all the temporary files handled by the objects, i.e. the files whose names were declared as "CHANA". It is thus very simple to implement pipe-like mechanisms, feeding a program with the output of another program, without having to worry about intermediate file names.

IV. Playing with memory management

Perl provides a sophisticated memory management system. An appropriate knowledge of its logic is important to correctly use the occp4 objects. While the constructor is *explicitly* called when a new object is created, the destructor is *implicitly* called when the object runs beyond its scope.

The use strict instruction

In order to make your script clearer, and to be sure of the scope of your variables (and consequently of your objects), you *should* declare use strict at the beginning of the script (cf example); with this declaration, you are ensured that:

- Every variable must be declared with the my instruction before being used
- Every variable declared with the my instruction is local to the block; this insures that the memory allocated for this variable is given back to the system at the end of the block

How to call the destructor ?

Objects in Perl are so-called "blessed" references (other languages like C refer to pointers). Working only with local variables, as noted above, the following rules may be applied in order to control the destructor call:

- When an object is created (calling its constructor with an instruction like $obj = new occp4(...)):
    - some memory allocation is performed for the object
    - the object is initialized
    - the constructor returns a reference pointing to the created object
    - The reference is stored inside $obj
- It is legal to write instructions like $o1=$obj. Only the reference is copied then, the object itself is *not* duplicated.
- If the scope of $o1 is longer than the scope of $obj, the object is still alive, even when $obj is destroyed. The exact rule is: "the object is alive as long as there is at least one reference pointing towards him". The object will be destroyed, and thus the destructor will be called, when the *last* reference pointing to this object is suppressed, **or** when it will be assigned again. For instance with the instruction $obj = 0. See the third example for a short illustration of this property.

Thus, even if the destructor cannot be explicitly called, as in other object oriented languages like C++, you may control exactly when the destructor is called by the system, just by controlling the affectation or scopes of reference variables.

V. some examples

This section shows 3 short examples of scripts, written in Perl, with the use of the occp4.pm module. Those scripts can be copied and pasted, the numbers shown inside comments refer to the notes below each example script.

Use of two CCP4 objects

This example shows the basics of occp4.pm:

- use instructions and other magical formulas
- How to create two objects
- A file generated by the first object and used as input by the second
- The destruction of those objects

The debug flag is also illustrated here.

```
#!/usr/local/bin/perl

use strict;
require "occp4.pm";                                   # 1


{                                                   # 2
my $obj_fft = new occp4 ('fft',
                         'hklin'=>"some_file",
                         'mapout'=>'CHANA.map');        # 3
$obj_fft->logfile(">>file.log");                        # 4

$obj_fft->keywords('TITLE'=>"some title",
                   'LABI' =>" F1=F1 SIG1=SIGF1 F2=FC PHI=PHIC",
                   'RESO' =>"$low_res $high_res,
                   'SCALE'=>"F1 3.0 0.0 F2 2.0 0.0",
                   'BINMAPOUT'=>' ');                   # 5
die (« something wrong in fft ») if ($obj_fft->run());  # 6

my  $obj_ext = new occp4('extend',
                         'MAPIN' =>$obj_fft->iofiles('MAPOUT'),
                         'MAPOUT'=>outfile.map',
                         'XYZIN' =>"some_file.pdb");    # 7

$obj_ext->debug("RVDF");                                # 8

$obj_ext->keywords('BORDER'=>"5.");
die « something wrong in extend ») if($obj_ext->run());

}                                                      # 9
```

1. Magical formulas
2. Start a new block
3. Alloc memory for an object:
   - The associated program is fft
   - hklin is some file already created
   - mapout is a temporary file, whose name is chosen by the object
4. The logfile is appended to the main script log file
5. The ccp4 keywords are entered
6. fft is run, the script is killed in the case of an error
7. Allocate memory for another object:
   - The associated program is extend
   - the mapin file is the mapout file of the previous fft object
   - the mapout file is a definitive one.
8. Set some debugging flags:
   - R: A message is printed when the program is run
   - V: Every file is verified just before running the program
   - D: A message is printed when the object is destroyed
   - F: The temporary files are *not* removed when the object is destroyed.
9. End of the block:
   - Both objects are destroyed
   - The temporary files (among them the intermediate file created by $obj_fft and read by $obj_ext) are removed.
   - The permanent files (the logfile and the last mapout file) are kept.
   - The temporary file (the logfile) created by $obj_fft is *not* removed, because the 'F' debug flag was set for this object; however, a message is printed about this file.

Use of xplor and explicit destruction of an object

This example shows:

- How to use xplor
- How to destroy the objects explicitly.

```
my $obj_xpl = new occp4 ('xplor.exe');                      # 1
$obj_xpl->input_src("F");                                   # 2
```

```
$obj_xpl->input_file("file.inp");                          # 3
$obj_xpl->logfile(">>file.log");
die "ERROR in xplor" if ($obj_xpl->run());
$obj_xpl=0 ;                                               # 4
```

1. Allocate memory for a new object:
   - the associated program is xplor.exe, which must be present in the path
   - There are no logical/physical file names, because xplor does not use the CCP4 library.
2. Select a file as the source of commands for this object : as xplor does not use the CCP4 library, the keywords/values pairs are not a relevant way of controlling the program. We prefer to write xplor commands to a file. However, the commands could have been also written to a string.
3. The name of the file with the xplor commands in it
4. The $obj_xpl reference is set to 0, and because there are no other reference points to our object, it will be destroyed.

How to write loops

The following example shows:

- The allocation of an object inside a loop
- The loop is exited when some condition is reached
- Differed destruction of the objects

```
my $old_obj=0 ;                                                    # 1
while (1) {                                                        # 2
        my $xyzin ;
        if ($old_obj) {
                $xyzin = $old_obj->iofiles("XYZIN") ;
        } else {
                $xyzin = "some_initial_value" ;
        } ;                                                         # 3
my $obj = new occp4("some_program",                                # 4
                        XYZIN=>$xyzin,
                        XYZOUT=> "CHANA") ;
        keywords etc.
        if (some_condition) {
                break ;                                            # 5
        } else {
                $old_obj = $obj ;                                   # 6
        } ;
} ;                                                                # 7

my $loop_obj = $old_obj;                                              # 8
```

1. Allocate memory for a reference, no object is pointed to yet.
2. start a block, specifying an infinite loop
3. The input file is specified as follows:
   - Some initial value when entering the loop, or
   - A file written at the previous iteration
4. Allocate memory for a new object:
   - The input file name was calculated at the previous line.
   - The output file is temporary.
5. If the condition is verified, exit from the loop
6. If the condition is not verified, do not exit:

  - We set the $old_obj reference so that it points to the last created object. This prevents the destruction of $obj, although we are going to start a new iteration.
  - Unless we are executing the first iteration, the object that was pointed to by $old_obj is now destroyed, because no other reference is pointed towards it.
7. This iteration is terminated, we exit from the block and start a new iteration. However, the object that was pointed to by $obj is still alive, and is now pointed to by $old_obj.
8. The loop is terminated, $old_obj is the object produced by the penultimate iteration.

VI. Module availability

occp4.pm is available on the ipbs ftp server ftp://ftp.ipbs.fr/pub/occp4. Also available at the same url is refmac.pl, a perl rewrite of a script written in ksh by Laurent Maveyraud (refmac.ksh) to perform refmac refinement with bulk solvent correction (in xplor or refmac).