CCP4 NEWSLETTER ON PROTEIN CRYSTALLOGRAPHY

An informal Newsletter associated with the BBSRC Collaborative Computational ProjectNo. 4 on Protein Crystallography.

Number 44 **Summer 2006**

Contents

Editor: Francois Remacle

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

Contributions are invited for the next issue of the newsletter, and should be sent to Francois Remacle by e-mail at fr54@ccp4.ac.uk by next season. HTML is preferred but other editable formats are also acceptable.

What's New at CCP4

Maeri Howard, CCP4, CCLRC Daresbury Laboratory, Warrington WA4 4AD

1. Staff changes

CCP4 continues to expand the number of people funded to work on associated projects. The most recent addition is Nick Furnham, located at Cambridge, who will be working on a restraint based conformational search engine called RAPPER. Nick is already busy on the conference scene as he has been asked to give a 30 min presentation entitled "Explorations in Conformational Space: Revealing Inaccuracy and Heterogeneity in Crystal Structures." at the ACA in Hawaii (Abstract can be found at http://www.hwi.buffalo.edu/ACA/ACA06/abstracts/S0102.html) .

CCP4 also bids goodbye to Dan Rolfe, who is leaving the group after eighteen months. Dan is not going far (in fact, he is going two buildings away!) as he will now be working with Dr. Marisa Martin-Fernandez, tracking fluorescents and molecules. We wish Dan the best of luck at his new post.

2. CCP4 Version 6.0

CCP4 staff had a very busy winter, making the next major release for the suite. More information on this can be found in the article *"What's New in CCP4 6.0."*

3. Workshops and Conferences

CCP4 continues to place conferences and workshops high on our list of ways in which to make contact with the user community and 2006 looks to be our busiest year ever. A listing of upcoming conferences can be found at http://www.ccp4.ac.uk/ccp4course.php with more detailed information found below.

Study Weekend 2006 continues to go from strength to strength as each year we have more and more attendees – allowing us to confirm its important place in the PX calendar of events. The topic this year "Crystallography of Complexes" was held the 5^{th} – 7^{th} of January at the University of Leeds and saw over 500 students attend the talks. Almost all of the talks from Study Weekend will be published later this year in a special edition of Acta D.

This year CCP4 also distributed a **questionnaire** to those that attended the Study Weekend. The purpose of the questionnaire was to get information from the users of the suite to help us establish a user profile as well as to get general feedback on the suite. We had 335 users return the questionnaire (a return rate of 62%) which we feel was a healthy return rate and one that allowed us to have good data to work with. The full report on the questionnaire can be found here.

Study Weekend 2007 will be held at the University of Reading 05 – 06 January 2007. The University of Reading is a familiar venue as Study Weekend 2005 was held there. The scientific organisers will be Garib Murshudov (York) and Frank von Delft (Oxford) and the topic will be **Molecular Replacement**. Registration will begin 18 September via the CCP4 website and the announcement will include important changes in the way that payment is made for attendance at Study Weekend.

BCA and the University of Lancaster Francois Remacle and Norman Stein attended the 24th annual British Crystallographic Association spring meeting at Lancaster University on 4th and 5th April 2006. They presented posters on Developments in CCP4 Version 6.0 and CCP4 Automation, which attracted a good level of interest. There was time to attend many of the invited talks, notably the plenary session in which Larry De Lucas (University of Alabama) described his experiences growing protein crystals under microgravity, while aboard the US space shuttle.

ACA 2006 - Honolulu, Hawaii: CCP4 will again be in attendance at the ACA . Due to the volume of other conferences/workshops going on over the year, we have decided to give our workshop a rest this year. But please stop by stand 601 to speak to either Peter Briggs or Ronan Keegan, who will be representing CCP4 at the meeting in Hawaii, see: http://www.xray.chem.ufl.edu/aca2006/index.html

ECM - Leuven, Belgium: CCP4 was lucky enough to secure one of the last spaces available at the ECM conference venue so we will be there! We can be found at Stand 23 from the 6th – 11th of August http://www.ecm23.be/GeneralInformation.html - Martyn Winn, Ronan Keegan, Francois Remacle and Wendy Yang will be on the stand to answer any CCP4 questions.

CCP4 Roadshows are a new addition to our promotions schedule. Since February of this year, CCP4 have visited five different labs located around the UK for one day workshops on the suite. The format for the visits have been very positively received and include talks in the morning on various aspects of the suite and then hands-on sessions to deal with specific questions later in the afternoon. The last of the visits took place at the end of June at the University of Nottingham. We look forward to resuming the roadshows after the summer holidays and will be posting the schedule of visit at our website.

Financially supported workshops: CCP4 continues to support a number of workshops throughout the UK, Europe and North America by providing funds towards the cost of the workshop. Some of the workshops that have received funding this year include: Gordon Conference, Refmac Workshop in Barcelona, South West Structural Biology Meeting and the Protein Structure Workshop at Galashiels.

Future workshops include our first ever Chinese workshop to be held in **Beijing** at the Chinese Academy of Sciences from the 27th – 29th of October. The three day workshop will cover topics such as Molecular Replacement as well as Refinement.

We will also be having a five day workshop from 27 November to 1 December in **Osaka, Japan**, covering all aspects of the structural solution process. Speakers include Kevin Cowtan, Phil Evans, Garib Murshudov, Paul Emsley, Roberto Steiner and Eugene Krissinel

4. Digitisation of CCP4 proceedings: All Study Weekend proceedings, pre ActaD, are now available digitally! CCP4 Study Weekend Conference Proceedings are now all publicly available online at CCLRC's ePubs institutional repository. http://epubs.cclrc.ac.uk/

You can pull up all 13 internally-published conference proceedings (plus a few other papers relating to CCP4) by simply searching for "CCP4". From these results you can drill down for more information on the specific paper and print the document you require.

What's New In CCP4 6.0

Peter Briggs

CCP4, CSE Department, CCLRC Daresbury Laboratory, Warrington WA4 4AD

1 Introduction

CCP4 version 6.0 is the most recent major release of the CCP4 software suite and was made in February 2006. A subsequent patch release 6.0.1 was made in June to fix a number bugs, however there were no major changes to functionality.

The full list of changes to the suite since the previous major version (5.0) can be found in the CHANGES file distributed with the core suite. As the CHANGES file focuses mainly on technical changes, the aim of this article is to give a quick tour of the new and updated features of the latest release in a more user-friendly manner.

The article has the following outline:

- Section 2 introduces the new download and installation mechanisms
- Section 3 comprises the bulk of the material and looks at updated sofware and new programs in the core CCP4 suite, the changes to CCP4i, and the new packages PHASER/CCTBX, CCP4MG, Coot and CHOOCH.
- Sections 4, 5 and 6 overview respectively the availability of, the bug fixes to and recent efforts promoting the suite to the PX community
- Section 7 describes some of the developments beyond release 6.0, and
- Section 8 gives the acknowledgements.

2 New download and install mechanisms for version 6.0

The current release sees the rollout of a new integrated download and install mechanism for the software, which is intended to vastly simplify the otherwise complication task of selecting packages for download via a simple interactive web interface. It also provides mechanisms for easy installation of the packages once they have been transferred to the user's system, using installers appropriate to the operating system in question - see the article on the new system elsewhere in this newsletter.

Using these mechanisms means that it is possible to have CCP4 and the other packages described below working on a machine in just a few minutes. Alternatively, old-fashioned style FTP download is also still provided. All the downloads can be accessed via the CCP4 Downloads Pages.

3 Updates to the software suite

3.1 The components of CCP4 release 6.0

For release 6.0 it was decided to break the distribution of the suite up into the following "packages":

- Core CCP4
- Phaser and CCTBX
- Chooch
- CCP4mg
- Coot

The breakdown is for practical purposes to do with availability, dependencies and rate of change of the various pieces of software - inclusion or exclusion from the "core" was a pragmatic decision and not an indication of the "quality" or "importance" of a particular package. In this context the "core CCP4 package" essentially refers to the the minimal CCP4 suite containing the CCP4 libraries plus the standard programs, the user interface CCP4i, the Clipper software, and some general "third-party" libraries.

Each of the new and updated packages are described briefly in the sections below.

3.2 Updated software in the core CCP4

3.2.1 PDBCUR for coordinate file manipulations

PDBCUR is a utility program for manipulating coordinate files, similar to PDBSET but already with a large number of useful complementary functions (for example selecting one model from a file which contains many models).

Additional functions new for CCP4 6.0 include:

- Produce a summary of the file PDB contents (for example, the number of models, the number of chains and their composition) (SUMMARISE keyword)
- Remove all hydrogen atoms from the model (DELHYDROGEN) *
- Select the most probable alternate conformation found in the file, defined as that with the highest occupancy (MOSTPROB). *
- Remove all atoms with occupancy below a specific threshold value (CUTOCC) *
- Move a subset of atoms in the model (TRANSLATE)

The options marked with * are also available via the CCP4i interface, in the *Edit PDB File* task in the **Coordinate Utilities** module (note that you must select the *pdbcur* option in order to access the new functions).

3.2.2 MATTHEWS_COEF for data analysis

MATTHEWS_COEF is traditionally used to estimate the number of molecules in the unit cell based on solvent content. In CCP4 6.0 it has been updated to additionally output the Kantardjieff and Rupp resolution-based probabilities for each possible number of molecules (*Protein Science* **12** 1865 -1871 (2003)), providing an additional tool for estimating the contents of the cell.

The corresponding CCP4i task *Cell Content Analysis* in the **Molecular Replacement** module has also been updated accordingly.

3.2.3 PDB_EXTRACT for structure deposition

The RCSB PDB_EXTRACT program has been upgraded since CCP4 5.0, to version 1.7 in release 6.0 and more recently version 2.0 in 6.0.1. PDB_EXTRACT can automatically extract data from program log files and other output as the structure determination progresses, and thus help to ease the burden of the final deposition process (see the article in the previous newsletter on this and other RCSB PDB Tools). PDB_EXTRACT is also accessible from within CCP4i via the *Data Harvesting Management Tool* in the **Validation and Deposition** module.

For more information about PDB_EXTRACT and other RCSB PDB Tools, see http://sw-tools.pdb.org/.

3.2.4 Other significant updates

The core CCP4 has a number of other updates to existing programs, most significantly:

- REFMAC 5.2.0019
- MOLREP 9.0.09
- SFCHECK 7.0.18
- MOSFLM 6.2.6 (was 6.2.5 in CCP4 6.0)

3.3 New Programs in the core CCP4

There are a number of new programs in the core CCP4 package in 6.0:

- SUPERPOSE: secondary-structure-based coordinate alignment
- BP3: heavy atom phasing & refinement
- CHAINSAW: molecular replacement model preparation utility
- PIRATE and the Clipper utilities: statistical phase improvement and related software
- PDB_MERGE: utility for combining the content of two PDB files

Each of these is described in more detail in the following sections.

3.3.1 SUPERPOSE

SUPERPOSE is a secondary-structure based structural alignment program. It is an alternative to the least-squares fitting method used in LSQKAB - SUPERPOSE uses the secondary structure features to perform the initial fit, followed by alignment of the protein backbone C• atoms.

The program takes two coordinate files as input and outputs the transformation matrix required to map one (the "moving" coordinate set) onto the other (the "fixed" set). It also gives a per-residue listing of the quality of the match, and identifies the secondary structure.

SUPERPOSE can be run from within CCP4i by selecting the *...Secondary Structure Matching* option in the *Superpose molecules task* (in the **Coordinate Utilities** module). It is also used as the structure alignment engine within both CCP4mg (under ***Applications -> Superpose proteins***) and Coot (under ***Calculate -> SSM Superpose...***).

3.3.2 BP3

BP3 is a heavy atom refinement and phasing program which uses a multivariate likelihood approach that has been reported to perform well in tests compared with other substructure refinement programs (see the references in the CRANK/BP3 article in an earlier newsletter). It can be applied to various types of experiment including single- and multiple isomorphous replacement (SIR and MIR), single- and multiple-wavelength anomalous dispersion (SAD and MAD) and various combinations thereof (i.e. SIRAS and MIRAS experiments).

Another advantage BP3 offers over CCP4's workhorse substructure refinement and phasing program MLPHARE, is that where MLPHARE forces the user to treat MAD phasing experiments with a "pseudo-MIR" approach (effectively selecting one wavelength as the "native" and then treating the other wavelengths as "derivatives"), BP3 treats each anomalous dataset equally and so has a better treatment of the phase errors.

BP3 can be run from CCP4i via the dedicated *Bp3 - Phasing task* in the **Experimental Phasing** module, or as part of the automated CRANK task in the same module (CRANK is described further on in this article).

More information on BP3 can also be found at the BP3 webpage.

3.3.3 CHAINSAW

CHAINSAW is a molecular replacement preparation utility that mutates a template PDB file using a previously-generated sequence alignment provided by the user. CHAINSAW preserves parts of the template which are conserved in the sequence alignment (and which are therefore more likely to be conserved in the target structure), and prunes back other parts which are not conserved (and which are therefore less likely to reflect the target).

CHAINSAW offers a number of different pruning methods (back to gamma atom, back to beta atom, or back to last "common" atom) and preserves more atoms than for example in a polyalanine model. It can accept sequence alignments in a variety of different formats (including ClustalW, PIR and Blast among others), which has the advantage that the user can optimise the alignment using their favourite methods before feeding it into CHAINSAW.

CHAINSAW can be run from CCP4i from the *Create Search Model* task in the **Molecular Replacement** module.

3.3.4 PIRATE and Clipper

PIRATE is a statistical phase improvement program - crudely, it is a replacement for the DM program. PIRATE uses phases from a previously-solved known structure (called the reference structure) as input to the phase improvement process, and doesn't require any a priori knowledgement of the solvent content of the target. PIRATE is run from the command line using the `cpirate` executable, or else through the *Run Pirate task* in the **Density Improvement** module of CCP4i.

In tests conducted by the program's author PIRATE performed better than DM in all but one case. It should be noted however that the program's performance depends critically on the quality of the Hendrickson-Lattmann (H-L) coefficients output from the phasing program used to generate the initial phases (as these give an indication of the error in the input phases). The current version has been "tuned" to work best with H-L coefficients output from PHASER but is known to also work well with the output from SOLVE. Using the output from MLPHARE will generally work less well, and tests with BP3 have not been carried out yet.

PIRATE can also make automatic use of non-crystallographic symmetry (NCS), if it is given a file of heavy atom coordinates.

For typical structures without any unusual features the choice of reference structure is not critical provided that it is relatively large - therefore the CCP4 distribution includes the structure factors for 1AJR in .na4 format (so-called "ascii-MTZ" - to convert to MTZ proper use the NA4TOMTZ program, or run the *Convert to MTZ...* task in the **Reflection Data Utilities** module). For structures with more unusual features (for example metalloproteins or RNA-complexes) it would be worth trying different reference structures which more accurately reflect the features of the target - in this case PIRATE includes a option to evaluate multiple reference structures (check *Use evaluation mode* in the PIRATE interface, or specify the `-evaluate` option if running `cpirate`).

The program MAKEREFERENCE is used to generate the reference input for PIRATE (use the `cmakereference` executable, or the *Make Pirate Reference* task in the **Density Improvement** module of CCP4i). MAKEREFERENCE requires both coordinates and structure factors for the reference structure, and outputs new coordinates and structure factors for input into PIRATE. If the computer has an internet connection then this data can be automatically downloaded from the PDB archive at the EBI.

The PIRATE program is built on top of a set of crystallographic software libraries called **Clipper**. In addition to PIRATE and MAKEREFERENCE, CCP4 6.0 and later also contain a number of other Clipper-based utilities that perform small tasks such as

generating maps and comparing phases. The utilities can be accessed from CCP4i via tasks in the **Clipper Utilities** module, and the CCP4i documentation gives a list of the available programs.

Information about PIRATE, Clipper and the related software can be found in Kevin Cowtan's webpages.

3.3.5 PDB_MERGE

PDB_MERGE is a useful jiffy for combining models from different sources, for example: constructing a complex from its components, constructing a new model of a protein using domains from several other models, or adding in separately generated symmetry mates. PDB_MERGE has just two modes (MERGE/NOMERGE), which control whether chains are merged at the same time as merging the data in the files.

3.4 Updates to CCP4i

CCP4i is the CCP4 graphical user interface system. The current version (released in CCP4 6.0.1) is 1.4.4.1, and the major changes have already been reported in a previous newsletter article. These developments are summarised in the following sections.

3.4.1 New core tools in CCP4i

The new core tools are illustrated in figure 1 and include:

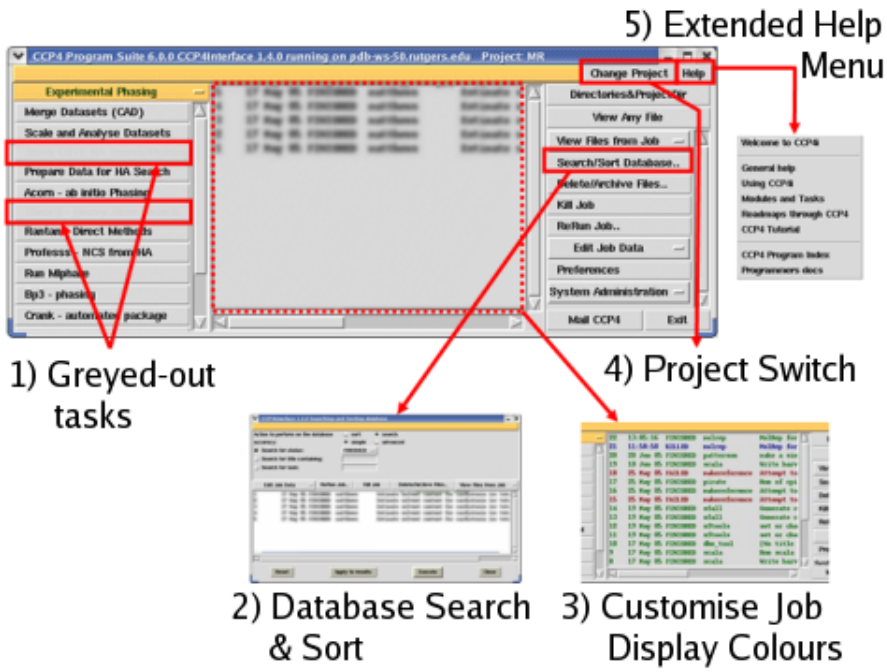


Figure 1: a map of the new core CCP4i tools and features

1. Greying out task buttons for certain tasks where the underlying programs are not found on the user's path
2. A tool to allow searching and sorting of the jobs in the project database
3. A function to allow the user to choose custom colours for the jobs displayed in the project database, to enhance quick comprehension of the content
4. A button to allow quick switching between different projects directly from the main window
5. Top-level help split into a menu with a number of subtopics, to help find relevant documentation more quickly

3.4.2 New interfaces

There are two major new interfaces:

- **CRANK**

  CRANK is a suite of programs for automated macromolecular structure solution. Currently Crank supports SAD, SIR and SIRAS experiments and makes use of various new and existing programs, including BP3, SHELX and various CCP4 programs.

  CRANK starts from scaled and merged data and allows the automatic solution of macromolecular structures up to the point of density modification. However it also has a "transculent box" design, intended to help teach novice users about the various programs used in crystallography.

  The CRANK interface is part of CCP4i's **Experimental Phasing** module. A more detailed article about CRANK appeared in a previous newsletter; alternatively, visit the CRANK webpage for further information.

- **Shelx C/D/E**

  The SHELX_CDE task interfaces to the SHELX programs, specifically SHELX C, D and E. The task takes either SCALEPACK format reflection files (note that the SCALA task can now output SCALEPACK-style files that are suitable for input into SHELX and SOLVE) or MTZ files containing intensities (preferred) or structure factor amplitudes as input.

  The task can be used to run the programs in a "pipeline" fashion from data preparation through heavy atom site location to density modification and hand determination, and generates useful plots from the output of each program.

  Like CRANK, the SHELX_CDE interface is also part of CCP4i's **Experimental Phasing** module. Note that the SHELX programs themselves must be obtained directly from the SHELX website.

3.5 The other new packages in CCP4 6.0

3.5.1 PHASER and CCTBX

PHASER is a maximum likelihood-based phasing program. The current version distributed by CCP4 (PHASER 1.3) has methods for molecular replacement, however functions for experimental phasing are also under development. One of PHASER's strengths is that its scoring function gives a more accurate estimate of the quality of its molecular replacement solutions; another is that it can search for solutions using ensembles made up of many possible search models, with each model's contribution to the ensemble weighted by estimates of its similarity to the target. It also allows searching for multiple molecules in multiple spacegroups.

PHASER's functionality can be accessed in a variety of ways - it allows each step in the MR process to be run independently, alternatively there is an "automatic" mode which will run through the whole process without user intervention. PHASER can also be run via a CCP4i task interface in the **Molecular Replacement** module.

PHASER depends upon the CCTBX crystallographic software libraries, and these must also be installed before building PHASER from source (they are not required for binary installations). For more information about CCTBX see the CCTBX pages on SourceForge. For more information about PHASER, see the PHASER webpages.

3.5.2 CCP4mg: presentation graphics in CCP4

CCP4mg is the official CCP4 molecular graphics package, and has focused so far on structure analysis and high-quality presentation graphics (including movies). CCP4mg is highly compatible with the CCP4 environment, and has an interface with a similar look-and-feel to CCP4i.

CCP4mg is built around the idea of *data objects* (the raw data that is loaded into the program, such as coordinates or reflections) and *display objects* (the representations of that data in the graphics window). A single data object can have many associated display objects, which can represent all or just some of the data (for example just the active site, a single chain or the whole molecule) in a variety of different ways (for example as ball-and-stick, ribbons, spheres or surfaces) and can reflect different properties of the data (for example secondary structure elements, solvent accessibility or electrostatic potential). Multiple display objects are easily generated and combined to quickly build complex representations.

Some examples taken from the CCP4mg website are shown in figures 2 thtough 4 below. These and other images showing the capabilities of the program can be seen in the CCP4mg Gallery, and in the examples in the 10-minute tutorials.

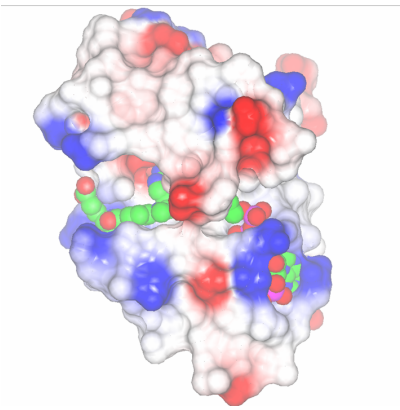


Figure 2: Displaying a surface coloured by electrostatic potential

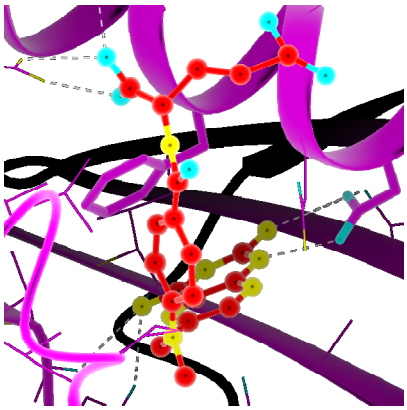


Figure 3: Active site in 1DFR, showing the local environment

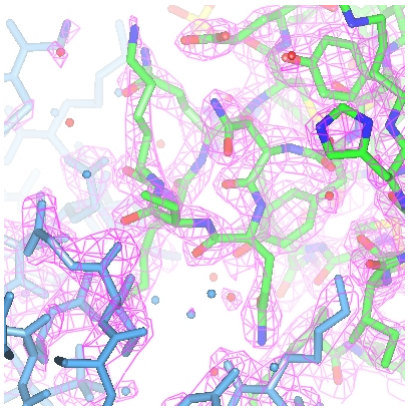


Figure 4: Displaying an electron density map

** Images taken from the CCP4mg website.*

CCP4mg has support for generating molecular surfaces (including transparent surfaces), and has various display modes for situations like protein-RNA complexes and ligand binding to DNA. The program can also display maps (for example to show electron density), either directly from a CCP4 map file or generated "on the fly" from an MTZ file containing structure factor amplitudes and phases. In addition it has tools for model superposition (via SUPERPOSE), generating symmetry and packing diagrams, and for examining the local environment for part of a protein model.

There is also functionality for adding captions and legends, for customising colours, and for rendering images at high-resolutions suitable for publication.

For more information see the CCP4mg website.

3.5.3 Coot

Coot is a platform for semi-automated model building and validation tools, and as such its functionality complements that in CCP4mg. The validation tools in Coot (including "dynamic" Ramachrandran plots as shown in figure 5, and geometry and B-factor graphs) allow the user to quickly focus in on areas of the model that are a poorly fitted, and once problems are identified there are a number of tools to help fix them (for example, rotamer fitting as shown in figure 6, residue mutation, loop fitting and real-space refinement amongst others).

Coot also offers tools to search for waters and for other "blobs" of unmodelled density, and to then model them for example by placing atoms in the density, or by allowing the user to pull ligands from the REFMAC monomer library and "drop" them into the model. Coot also has the option to run rounds of REFMAC refinement from within the program, as well as a large number of other tools for model completion not described here.

For further information about Coot visit the Coot webpages.

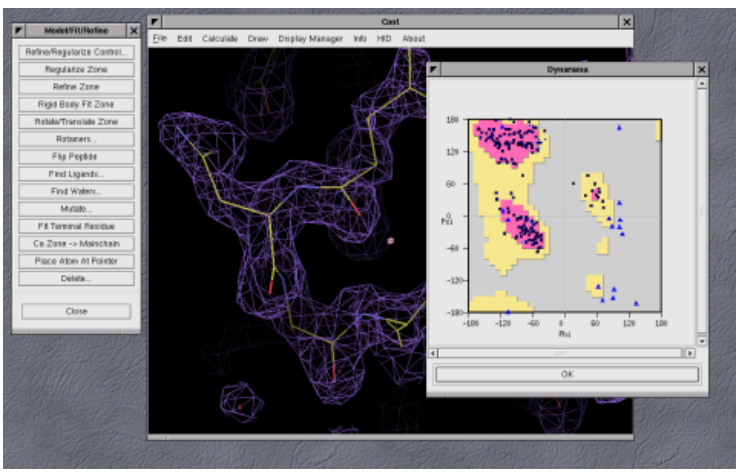


Figure 5: Example of Coot's dynamic Ramachandran plots

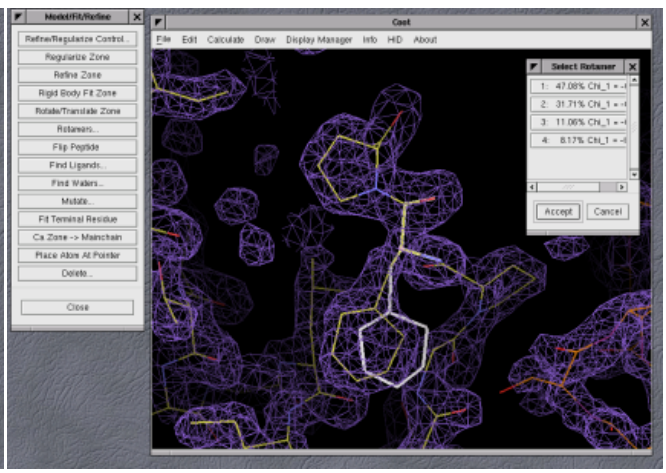


Figure 6: Example of rotamer fitting in Coot

** Images taken from the Coot website.*

3.5.4 CHOOCH

Having accurate values for the anomalous scattering factors f' and f'' is essential for the success of anomalous phasing methods. CHOOCH is able to determine the values of

these parameters from the raw fluorescence spectra. Currently there is no CCP4i interface to run CHOOCH, however this will be addressed in a future release.

An article on CHOOCH (in Word format) appeared in a previous newsletter, or alternatively additional information is available at the CHOOCH webpages.

4 Availability

Version 6.0 of the CCP4 software suite is available for download and use free of charge for academic and non-profit use, upon the completion of a valid licence agreement. The academic licence is included with the suite, or else can be found via our licence pages, however please note that the licence for version 5.0 is not valid for version 6.0. Organisations wishing to use the software for commercial purposes should contact CCP4 to obtain a commercial licence - this information is also available via our licence pages.

Whenever possible we have endevoured to provide both source code and precompiled binaries for all the programs and packages on all the platforms that we have access to. This includes various flavours of Linux and some older UNIXes, as well as Microsoft Windows and Mac OS-X. Our intention is to give users of the software as many options as possible for installation.

The current major exceptions are CCP4mg and Coot, both of which are only available from the CCP4 website as binary packages. This is for practical reasons only: building these programs from source code is not a trivial task and depends on the system setup and on the level of experience of the installer. For those who do wish to take up the challenge, the source code can be obtained from the appropriate websites: http://www.ysbl.york.ac.uk/~ccp4mg/download/ for CCP4mg and http://www.ysbl.york.ac.uk/~emsley/coot/ for Coot.

5 Fixing bugs

As part of the life cycle of any software release, bugs are continually being discovered and fixed. Fixes to problems are usually posted on the Problems Pages, and typically take the form of a "workaround" (suggested steps to avoid the bug) or as fixes to the program source code. In this case the fixes are normally distributed as "patches" (a *patch* is a file containing instructions on how to modify another file, which can be read by the UNIX `patch` program). It is always a good idea to check the Problems Pages before reporting problems.

CCP4 6.0 and later includes a new utility called *patch_ccp4.sh*, which on UNIX systems will automatically check for available patches to the release, download and apply them. If your installation was from source code then you still need to recompile any updated programs, however this will also apply fixes to things like CCP4i which do not need to be (re)compiled.

Depending of the lifespan of the release and the number of fixes accumulated since the last release, patch releases such as 6.0.1 are occasionally produced. These updates should not add any new functionality. It is likely that at some stage there will be a 6.0.2 patch release of CCP4 6.0.

6 Promoting the current release

Since the release of CCP4 6.0, CCP4 staff and associated developers have been visiting groups at various labs in the UK (and in one case further afield) in order to demonstrate the new features of the software and talk to scientists about using CCP4. Feedback from these visits has been very positive, so if you feel that your group might benefit from a visit by CCP4 staff and developers then please contact us at ccp4@dl.ac.uk.

CCP4 staff are also attending a number of international conferences, with exhibition stands at the ECM and ACA conferences amongst others, and will be happy to demonstrate the software and to answer questions about it. Please see the CCP4 courses pages for the most up to date information.

7 Beyond 6.0: other programs and projects

Beyond CCP4 6.0 there are a number of other current and forthcoming programs and projects associated with CCP4. Many of these are linked from either the CCP4 Projects Page or the Prerelease Page.

Some of the programs that are now currently available outside of the CCP4 6.0 include:

- **MrBump:** Molecular replacement with Bulk Model Preparation

  MrBump is an automated molecular replacement pipeline, with emphasis on generating a variety of search models. It uses various "helper" applications (e.g. CHAINSAW), bioinformatics tools (e.g. FASTA) and on-line databases (e.g. the PDB) to generate models, and then uses MOLREP, PHASER and REFMAC for the MR steps.

  In favourable cases MrBump has given a "one-button" MR solution; in less favourable cases it can still be used to generate search models for further investigation.

  MrBump has a CCP4i task interface but can also be run standalone. It can be obtained via http://www.ccp4.ac.uk/MrBUMP/ and will be included in the next major release of CCP4.

- **PISA:** Protein Interfaces Surfaces and Assemblies

  PISA is an interactive tool for the exploration of macromolecular (proteins and DNA/RNA) interfaces, prediction of probable quaternary structures (assemblies) and database searches of structurally similar interfaces and assemblies.

  PISA is currently available as a webservice hosted by the MSD-EBI: http://www.ebi.ac.uk/msd-srv/prot_int/pistart.html. It is also planned that a future release of CCP4 will feature a standalone version of PISA.

- **iMOSFLM**

iMOSFLM is the new interface to the MOSFLM data processing program. It is intended to replace the current dated X11-based interface with a more intuitive and user-friendly graphical interface that will also work on Microsoft Windows.

The GUI is now available for public download from http://alf1.mrc-lmb.cam.ac.uk/~geoff/mosflm/.

- **POINTLESS**

  POINTLESS does a number of things: it can be used to determine the Laue group and spacegroup from a set of unmerged reflections; it can score possible indexing schemes for a merged or unmerged dataset against a merged dataset; and it can be used to reindex or change the spacegroup of a reflection file.

  POINTLESS can be obtained from the CCP4 Prerelease Page.

- **Buccaneer**

  BUCCANEER is a new Clipper application that performs statistical chain tracing by identifying connected alpha-carbon positions using a likelihood-based density target. It is essentially a model building program, and follows on very naturally from the output of experimental phasing steps and in particular from PIRATE. Tests by the program's author suggest that it is complementary to ARP/wARP, since BUCCANEER is relatively insensitive to resolution but very sensitive to the quality of the phases and the estimates of their errors.

  BUCCANEER can be obtained via http://www.ysbl.york.ac.uk/~cowtan/.

8 Acknowledgements

The CCP4 project is a collaborative effort and continues to thrive through generous contributions of time, energy and software from members of the UK and internation PX communities. Unfortunately time and space do not permit the acknowledgement here of all these valuable contributions, however a list of acknowledgements is included in the current release, and acknowledgements for the specific developments described in this article are given below:

- PHASER is developed by Randy Read's group (Wellcome Trust, Cambridge). CCTBX is developed by the Computational Crystallography Initiative (CCI) at Lawrence Berkeley, California.
- SUPERPOSE, PISA and PDBCUR are developed by Eugene Krissinell (MSD-EBI). The new functionality added to PDBCUR for CCP4 6.0 was written by Martyn Winn.
- MATTHEWS_COEF was originally written by Misha Isupov; the Kantardjieff and Rupp analysis was added by Charles Ballard.
- PDB_EXTRACT is developed by Huanwang Yang (RCSB PDB).
- BP3 is developed by Navraj Pannu (Leiden University). CRANK has been developed by Navraj and Steven Ness.
- CHAINSAW is developed by Norman Stein (Daresbury Laboratory).
- PIRATE, BUCCANEER and Clipper are developed by Kevin Cowtan at (York University).

- PDB_MERGE is developed by Martyn Winn (Daresbury Laboratory).
- CCP4i is maintained and developed by the CCP4 group at Daresbury. The Database Search/Sort, Project Switching and Job Colourisation functions were developed by Francois Remacle. The SHELX_CDE interface was written by Peter Briggs. The SHELX programs themselves are developed by George Sheldrick and co-workers.
- CCP4mg is developed by Liz Potterton and Stuart McNicholas at (York University).
- Coot is developed by Paul Emsley (York University).
- CHOOCH is developed by Gwyndaf Evans (DIAMOND Light Source).
- The new download and installation mechanism was developed by the CCP4 group at Daresbury, with Francois Remacle as the programming lead.
- MrBump is developed by Ronan Keegan and Martyn Winn, originally under the auspices of the e-HTPX e-science project and now as part of CCP4.
- iMOSFLM is developed by Geoff Battye (MRC-LMB Cambridge).
- POINTLESS is developed by Phil Evans (MRC-LMB Cambridge).

The images for CCP4mg and Coot were taken without permission (and thus with apologies) from the relevant websites. PJB would also like to apologise to anyone who has been missed from these acknowledgements.

The CCP4 suite is maintained, developed and released by the CCP4 group in the Computational Science and Engineering Department at CCLRC Daresbury Laboratory, and comprises Charles Ballard, Peter Briggs, Maeri Howard, Ronan Keegan, Francois Remacle, Dan Rolfe, Norman Stein and Martyn Winn.

The CCP4 project is supported by the BBSRC, by income from commercial distribution of the software, and by CCLRC Daresbury Laboratory. CCP4 would also like to thank the many people past and present who support the project, both with their time and with their contributions to the software suite itself - without which the project would not be able to exist.

Peter Briggs, July 5th 2006

The New Download and Installation Mechanisms in CCP4 6.0

Francois Remacle, Peter Briggs

CCP4, Daresbury Laboratory, Warrington WA4 4AD, UK

Introduction

With many recent developments going on, the size of CCP4 program suite has grown continuously. With the latest version there has been large scale software added into it. Altogether internal dependencies, operating system dependent availability has lead to a complex package to deal with.

In the previous releases, users were forced to take responsibility for finding what they needed and understanding the instructions on how to install/build it on their particular system. Very often this meant turning to network or computer system technicians to ask them to do it on their behalf.

It is in this context that the packages selection from the new system was introduced, to make things easier for end users without requiring them to have any deep knowledge of their system, and by showing them directly what is only available for their particular machine.

Outline

We will describe in the three following parts, what the new download system looks like, how the new installation mechanism works, and give some information about how it might be enhanced in the future

Download Mechanism

Along with the test releases of version 6.0 of CCP4, a new download and installation mechanism has been developed and tested. There was a need to develop a new mechanism to give users a quick and easy way to find what they want for the platform they are using. The idea was to provide users with as much automatic aid as possible in order for them to act as supervisors that can overrule decisions if they want, but would have very few things to do otherwise.

After exploring different ways of presenting the relevant information we finally chose the presentation style you can see in figure 1.

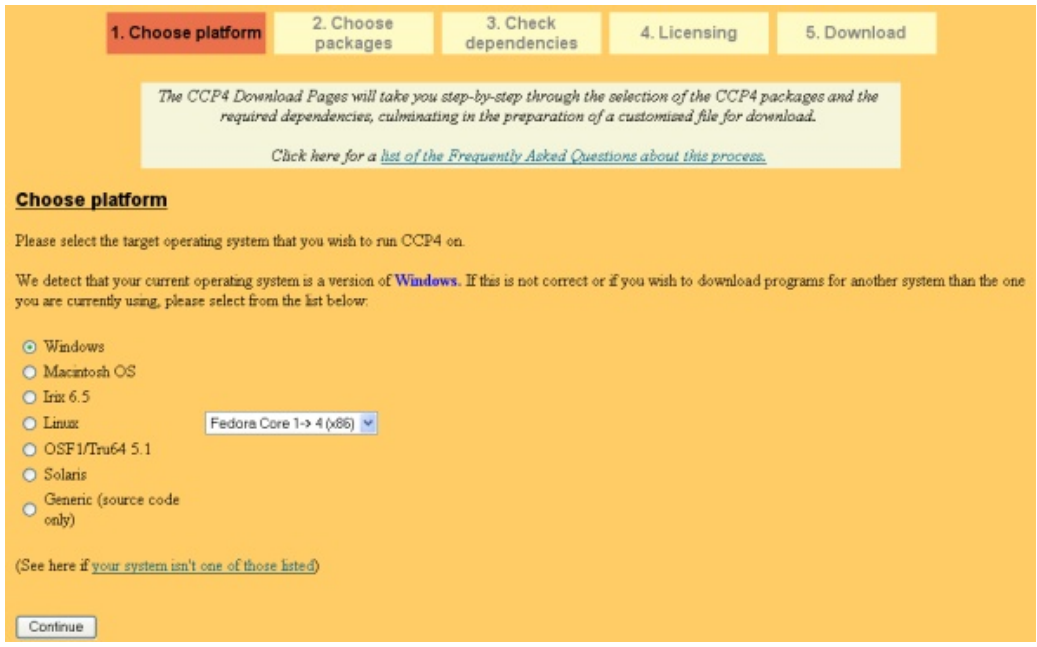
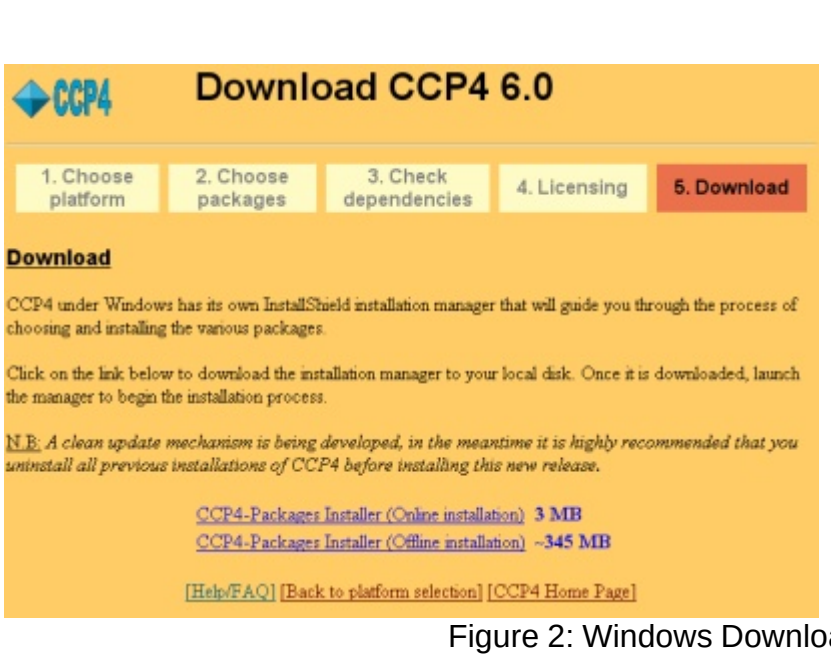


Figure 1: Download Home page

Even though there was the clear intention of making the process identical on each platform. Windows and Macintosh users are treated differently because those systems have specific installation mechanism. Therefore they are directed immediately to a download step where they can pick up the easy "click and go" installers.



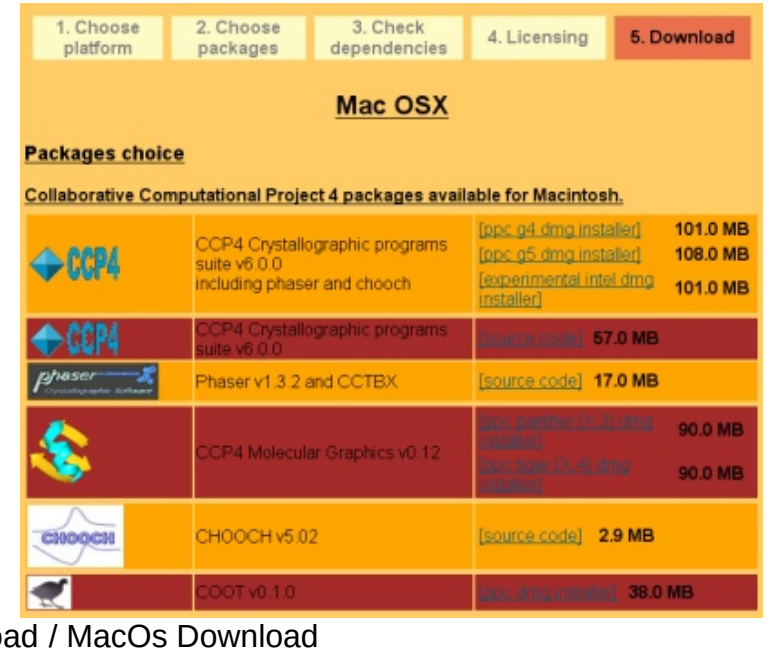


Figure 2: Windows Download / MacOs Download

The other platforms, as shown in figure 3, presents users with a choice of software available for the platform they have chosen. Sometimes only executables are available, some times only source code, sometimes both. This depends on the choice of platform. Users can then select what to download or simply ask for everything.

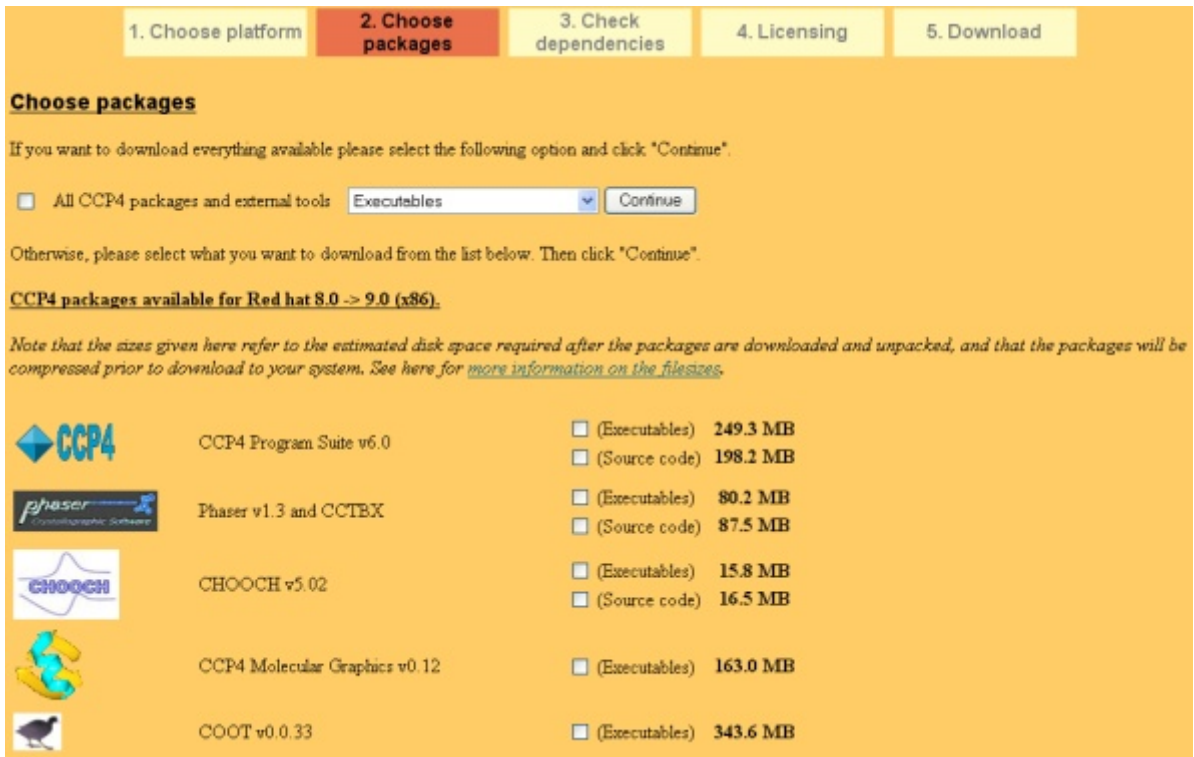


Figure 3: Package Selection

As some of the software depends on other packages to work properly, the selection will be checked for users to ensure they do not forget these (cfr figure 4 left). However, any dependencies will only appear as a suggestion, this means that users are the ones who take the final decision. The next step, as shown in the right part of figure 4, will then provide users with a summary of their choice with the compressed size (how big the size of the download file will be) and uncompressed size (how much space it will take on your hard disk when unpacked). By agreeing to the license user will then be directed to download step.

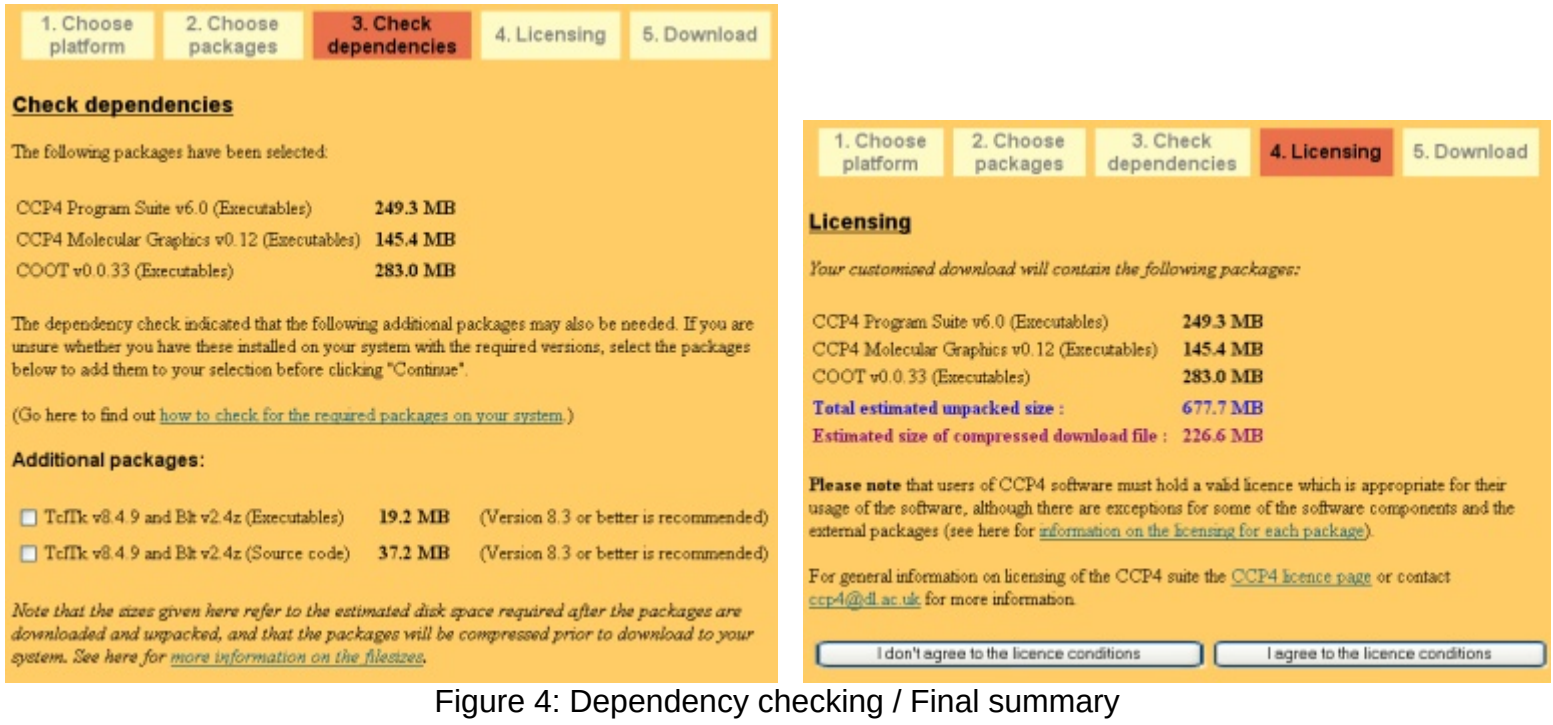


Figure 4: Dependency checking / Final summary

The server will give back a single "customised" file for download that contains only the specifically requested packages and tools. It can take the server several minutes to package up a particular combination of components "on the fly". However, this is only done once for each combination because the result is cached in order for subsequent requests for the exact same set of components to be downloaded immediately (N.B: it sometimes happens that archive are corrupted when being packaged up, either due to very old web browser either due to a concurrent packaging). Before a release, we will pre-generate the most popular combinations (according to our website statistics) to make downloads quicker.

Installation Mechanism

As stated above there is a difference in installations between Windows/Macintosh and the other platforms. The reason is that both Macintosh systems and Windows have their own easy installers, as shown in figure 5, where an installation wizard guides you step by step.

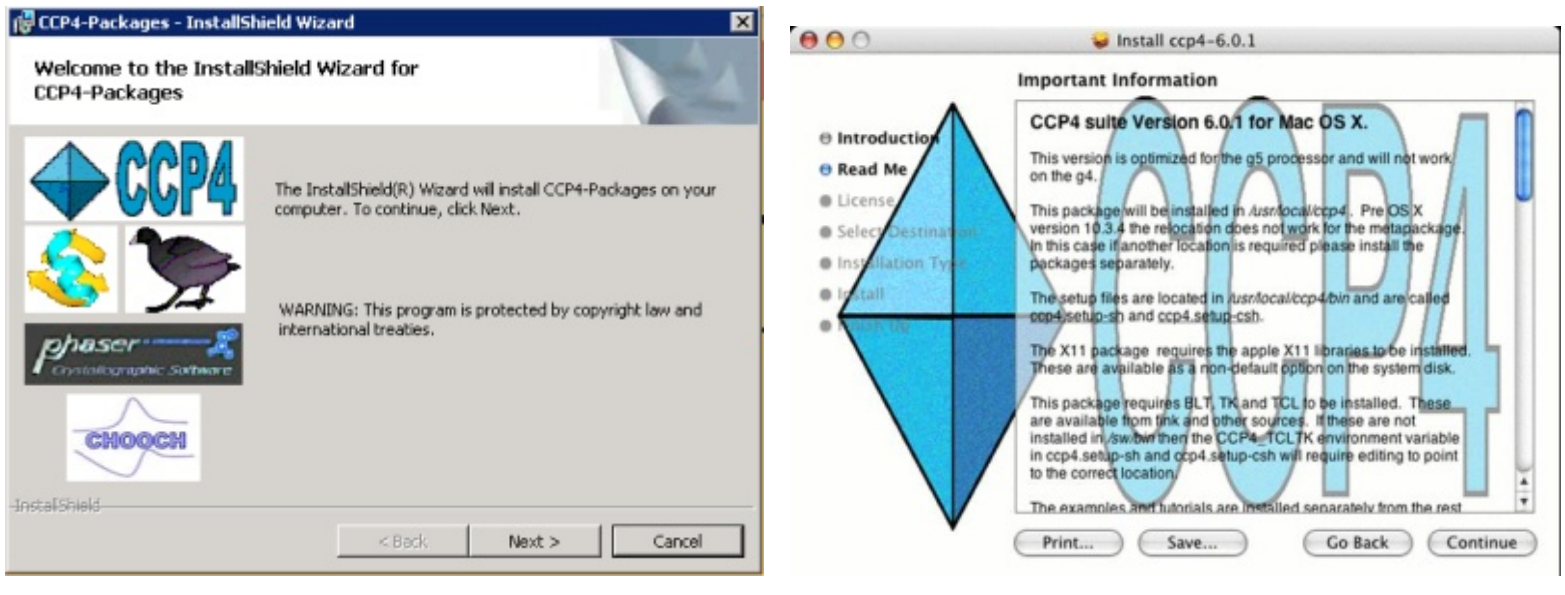


Figure 5: Windows Installer / MacOs Installer

People who are less familiar with UNIX software installation can be intimidated by large software packages like CCP4, particularly when there are diverse components which each have different installation procedures. The new download mechanism helps them to select the packages that they want but there is still the problem of how to actually install the software and make it useable once it is downloaded. To cope with this problem, as part of the download process the server also secretly generates an installation script which is customised to the exact choice of packages requested by the user.

The philosophy of the installation script is similar to the download mechanism: it hides much of the detail from the user and presents only the essential choices that need to be made, in a friendly fashion. The user is given an initial choice of either a "fully automatic" default installation, or a "custom" installation where they have more control. In the first case (see figure 6), the process is reduced to just two questions after which the script should run to completion without further user intervention, simply displaying information about its actions as it goes along.

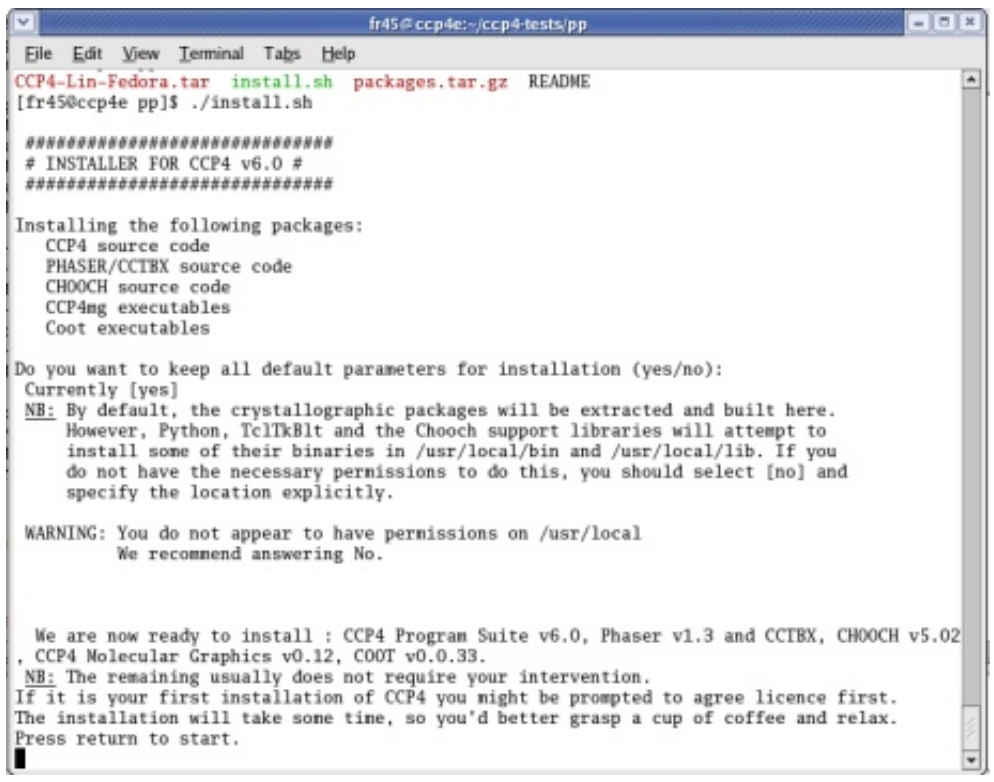


Figure 6: Simple Case

In the case of the custom installation (see figure 7), the user is asked more fine-grained questions. They can either take the script's defaults or else change as few or as many options as they wish.

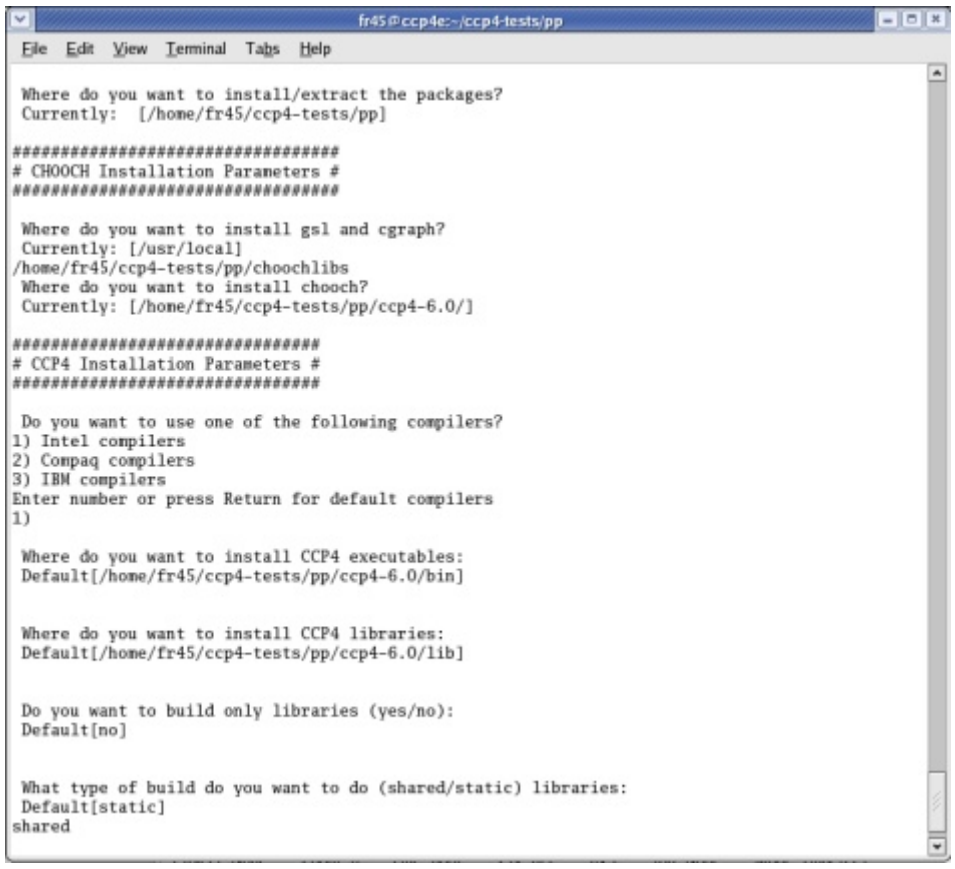


Figure 7: Advanced Case

Hopefully everything is then installed and prepared for you to setup your Unix system environment easily. In the event that something goes wrong, the installation script keeps track of specific error messages from the process, and writes these to a special file. This error file is checked at the end to see if anything went wrong. If this is the case then the error log provides a quick way to locate the error, instead of having to go through loads of compiler output. This also makes error reporting easier - the user can provide the CCP4 staff with the log files from the installation, and this helps us to identify and address the problem for the user.

Future

Currently there are a number of different ideas being investigated to further improve the installation mechanism.

- A clean update mechanism for Windows: this should enable user to update their installation, rather than deleting the old version to install the new one.
- A "windows-like" installer for executables on other platforms (i.e Solaris, Mac and Linux): A prototype already exists for Linux, and operate in a similar way to the installer for Windows: you are taken step by step through the installation process, and at the end your enviroment is automatically setup. However there are some limitations:
    1. it uses a different method from the traditional ccp4.setup file to set the environment, by modifying the user's login files on the system - some users don't like this.
    2. The installer can only set the environment for sh shells (not csh).
    3. A minor limitation is that it should create shortcuts on the desktop and on the main menus. However this does not work for all types of Linux.

Figure 8: Prototype of a installshield installer wizard for Linux

- Development of a similar mechanism to download and install, updates, patches into existing installation.

Acknowledgements

Most of the development of the download pages has been done by Francois Remacle with the guidance and participation of the other CCP4 staff based in Daresbury.

We would like to thank all the CCP4 developers and beta-testers who helped us test the pages to identify the best design, solve problems and improve the pages to the state they are now in.

Finally we also would like to thank all CCP4 users for using the suite and helping us improve it through their feedback and comments.

BioCrystallographica, a package for doing Crystallography with *Mathematica*

Nicolas Ambert*, Julien Vanwinsberghe*, Philippe Dumas.

Equipe de Cristallographie
UPR9002 du CNRS conventionnée avec l'ULP
IBMC, 15 rue René Descartes 67084 Strasbourg cedex FRANCE

** N.A & J.V contributed equally to this work*

Corresponding author: P. Dumas
e-mail: p.dumas@ibmc.u-strasbg.fr

Abstract

We have made a package for doing macromolecular crystallography with the software *Mathematica* from Wolfram Research. This package contains in its present state all necessary functions for handling files commonly used and performing all basic calculations required in daily work. Our goal was obviously not to offer a stand-alone alternative to well known (and huge) crystallographic packages like CCP4 or CNS, but to open the way for using a programming language with very rich symbolic capabilities. To take benefit of existing softwares, we have made possible to call various CCP4 routines from within the package and to get the results immediately available in the package environment. Such possibilities could be easily extended to CNS, or other packages. In its present state, *BioCrystallographica* is not really intended for a common usage, but rather to allow those involved in the field to analyze their results and develop new methods. The *Mathematica* programming language, as other languages of that kind, requires a significant practice before being used efficiently, particularly for abandoning procedural programming methods in favor of functional ones. This investment, however, is worth the effort in view of the extreme density of the code and of the depth of the questions that can be addressed.

1. Introduction

There now exists a large number of crystallographic softwares, most often organized in more or less general packages. CCP4 (CCP4, 1994) and CNS (Brünger et al., 1998) are particularly well known examples. Having a look to Jean Cavarelli's web site in Strasbourg provides an almost exhaustive list of available softwares. All of them rely on different kinds of programming languages, from the venerable Fortran to the 'up-to-date' Java and Python, passing by the not so young C++. To our knowledge, no serious attempt of using a programming language with symbolic capabilities like *Mathematica* has been made. Several reasons can be put forward to explain this situation. Probably, the most important ones are (i) that such a programming language is considered as slow and, (ii) that it is seen as expensive. However, we think that such a tool offers invaluable possibilities that cannot be found in any existing crystallographic softwares. Indeed, although packages like CCP4 and CNS are extremely versatile by allowing to combine at will individual programs, they do not allow to perform

immediately an original methodological test requiring a *not yet programmed function*. On the contrary, the environment offered by a programming language like *Mathematica* authorizes to make such a test very easily.

2. Essential features and working environment of *Mathematica*

Mathematica is organized in two parts: the so-called 'front-end' corresponding to the program-user interface, and the 'kernel' receiving the commands from the front-end, doing the calculations and returning the result to the front-end. Although it is still possible to interact directly with the kernel in an 'old-fashioned' way (*i.e.* at the command-line level, *e.g.* in a unix window), it is obviously much more efficient and user-friendly to do so through the front-end. This allows to obtain directly graphical results immediately after the corresponding lines of code, and also to interact very efficiently with an exhaustive on-line help. The front-end interface is organized by the user into individual cells that represent a logical unit of code. This organization is flexible because it may be modified at will. The input code is interpreted cell by cell if desired, which also provides a high degree of flexibility during development.

It should be realized that the essential characteristics of a software like *Mathematica* resides in its symbolic capabilities. Therefore, it differs from softwares devoted to numerical calculations by the fact that, by default, it does not assume that a general symbol has a precise meaning. For example, defining a simple function like $f(x) = a\ x^2 + b\ x + c$ without mentioning anything about the parameters ***a, b, c*** and the variable ***x*** lets open the possibility that these may correspond to usual real numbers (with finite machine precision), to numbers corresponding to symbols with infinite precision (*e.g.* π), to complex numbers, to matrices, or even to yet undefined symbols that will allow such algebraic calculation. Obviously, such a burden may have a price in terms of speed, but may also yield results unattainable with classical 'purely numerical' softwares. However, when restricting the input data to numbers with finite precision, the calculations become comparable in execution time to that obtained with more specialized softwares. For example, obtaining the eigen values of a large numerical matrix is in now way particularly slow.

3. Illustration of the symbolic capabilities

Suppose we are interested in expressing the general form of a matrix representing a rotation of angle θ around an axis defined by a unit vector **u.** Of course, such simple objects are immediately available in the set of predefined tools. However, we will show as an example how to define and use it symbolically. For avoiding lengthy developments beyond the scope of this note, several technical aspects will not be detailed, or will even be left in the shadow, when the context is sufficiently clear or when this does not compromise understanding. Everyone interested in details should search in the on-line help for additional explanations (and accept to spend some learning time).

We can first make use of the intrinsic definition of a rotation. Rotating a vector **V** by θ around a unit vector **u** defining the axis of rotation may be expressed as :

$$\begin{aligned} \mathbf{W} &= (\mathbf{u}\cdot\mathbf{V})\,\mathbf{u} + \cos\theta\,[\mathbf{V} - (\mathbf{u}\cdot\mathbf{V})\,\mathbf{u}] + \sin\theta\,(\mathbf{u}\times\mathbf{V}) \\ &= (1-\cos\theta)(\mathbf{u}\cdot\mathbf{V})\,\mathbf{u} + \cos\theta\,\mathbf{V} + \sin\theta\,(\mathbf{u}\times\mathbf{V}) \end{aligned} \qquad (1)$$

Such an expression has an obvious geometric meaning in terms of the component of **V** along the axis of rotation **u** (invariant under the rotation), and of the rotated component of **V** perpendicular to **u**. Equation (1) has an immediate translation in the *Mathematica* language:

```
RotVec[u_List,θ_][V_List]:=(1-Cos[θ])*(u.V)*u + Cos[θ]*V + Sin[θ]*Cross[u,V]
```
(2)

This defines a function Rotvec depending on the two parameters **u** and θ and taking **V** as the variable. Note that apart for idiosyncratic specificities inherent to any programming language, the latter definition (2) is easily decipherable. The nice thing is that (2) can be used symbolically with **u** defined as *{a, b, g}* and **V** as {x, y, z} without giving explicit values to the variables. As detailed a little bit more in the following, the braces { } are the hallmark of any kind of list in *Mathematica* and a vector may be considered as a 1D list. One may thus ask for the result of :

```
W = RotVec[{a, b, g},q][{x, y, z}]
```
(3)

which will appear as a list of three linear forms in x, y and z. By applying the following syntax (which will not be explained in details here, but see **§ 6**),

```
m = Map[Coefficient[#,{x,y,z}]&,W]
```
(4)

the coefficients of x, y and z are obtained, which after an additional line of code for factorization (not shown here), yields the sought after rotation matrix m :

$$\begin{pmatrix} a^2(1-Cos[q])+Cos[q] & ab(1-Cos[q])-g\,Sin[q] & ag(1-Cos[q])+b\,Sin[q] \\ ab(1-Cos[q])+g\,Sin[q] & b^2(1-Cos[q])+Cos[q] & bg(1-Cos[q])-a\,Sin[q] \\ ag(1-Cos[q])-b\,Sin[q] & bg(1-Cos[q])+a\,Sin[q] & g^2(1-Cos[q])+Cos[q] \end{pmatrix} \quad (5)$$

Note that the latter expression was not hand-written, but is a direct output of *Mathematica*. In order to achieve giving a glimpse on the symbolic capabilities, we will simply show the result of the function Eigenvalues applied to the latter matrix m:

```
Eigenvalues[m] /.(α^2+β^2+γ^2)→1
```
(6)

In (6), the syntax /.(α^2+β^2+γ^2)→1 is a 'replacement rule' forcing **u** to be a unit vector. As output of (6), one readily obtains the list of eigen values:

```
{1, Cos[q]-iSin[q], Cos[q]+iSin[q]}
```
(7)

This is a useful reminder that e^{-iq} and e^{iq}, as well as 1, are eigen values of a matrix of rotation by an angle q. Asking for the corresponding eigen vectors with:

```
FullSimplify[Eigenvectors[m]] /.(α^2+β^2+γ^2)→1
```
(8)

one obtains as output:

$$\left\{\left\{\frac{a}{g}, \frac{b}{g}, 1\right\}, \left\{-\frac{b^2+g^2}{ib+ag}, -\frac{ia+bg}{a^2+b^2}, 1\right\}, \left\{-\frac{b^2+g^2}{-ib+ag}, \frac{ia-bg}{a^2+b^2}, 1\right\}\right\} \quad (9)$$

containing, first the expected vector *{a/g, b/g, 1}* parallel to **u** ={a, b, g}, as well as those corresponding to the complex eigen values $e^{-i\eta}$ and $e^{i\eta}$. In (9), `FullSimplify` is a call to a simplification procedure of wide use for algebraic expressions.

4. Description of the basic set of utilities available in the package

We will only give a brief list of essential utilities.

4.1 Reading (and making) files

PDB coordinate and reflection files
Denzo and CCP4 MTZ reflection files (MTZ files can be read and written)
CNS and CCP4 electron density maps (CCP4 maps can be read and written)

4.2 Atomic model representation

A simple graphics representation of an atomic model is available. For now, this is only a very crude option for the sake of verification of the coordinates in use.

4.3 Crystallographic data bases (from CCP4)

All symmetry operations.
Limits of asymmetric units and alternative origins for each space group.
All atomic scattering factors.

4.4 Elementary calculations

Orthogonalization and deorthogonalization matrices.
Metric tensor in reciprocal space, calculation of resolution.
Sorting out centric and general reflections.

4.5 Phase and map calculation

Phase and map calculations can be made either internally (*i.e.* as a *Mathematica* procedure), or by launching from within *Mathematica* in a 'user-transparent' way a CCP4 calculation (*i.e*. without any need of typing a DOS or Unix line of command or of using the CCP4 GUI).

4.6 Plotting and graphics representation

Mathematica provides a great deal about plotting data and graphics representation. There exists many possibilities for exporting and importing graphics. This aspect is not secondary because working in such an environment makes graphics representation immediately available for anything. Map section contouring, for example, is extremely efficient and fast. In the present state, however, it is not possible to make plots with oblique axis as requested for non orthorhombic space groups. It is intended to remove this limitation in the next release of the package.

4.7 On-line help

It is an important feature that all tools in the package are referenced in the on-line help, exactly as for any 'official' *Mathematica* tool. It should also be emphasized that this on-line help is not only an exhaustive list of the syntactic rules, but that it also allows a dynamic test of them. For example, when searching for how a call to CCP4 is made for calculating an electron density map, the user has the possibility, *within the help itself*, of running real examples and changing parameters.

5. Organization of data

One major tool in *Mathematica* is the 'list'. As previously mentioned, a 'list' is simply notated by braces enclosing different elements separated with commas (*e.g.* `FirstPrimes = {1,2,3,5,7};` one particular element, for example 5 at the fourth position, is retrieved as `FirstPrimes[[4]]`). It can be viewed as representing any collection of any objects structured in any possible way. A list can be as simple as a 1D vector of numbers (atomic coordinates are obviously structured as a list {x,y,z}), but can collect symbols, *e.g.* $\{\partial/\partial x, \partial/\partial y, \partial/\partial z\}$, or even symbolic rules. It can be structured to any depth, regularly as a m×n matrix, or without any regularity. It is of such use that many powerful tools exist to manipulate them. Simple examples will illustrate its usage.

5.1 Definition of 'parallel lists'

What we call 'parallel lists' is of extremely common usage in the package. Two lists are said 'parallel' if each element in one list corresponds to the element at the same position in the other list. As a consequence, 'parallel lists' must have the same length. The importance of this notion is made very clear in the following with the organization of atomic coordinates or of reflections. For example atomic coordinates and the corresponding atomic names are stored in 'parallel lists'. Importantly, the substructure of each list may be different; one only requires that the two lists be identically structured at the first level.

One very important function allowing to make a single list from two parallel lists or, alternatively, to split a single list into parallel lists is TRANSPOSE . For example, with the two parallel lists:

```
hkl = {{h1,k1,l1},{h2,k2,l2},{h3,k3,l3},…};

F  =  {F1, F2, F3,…};
```

one obtains a single list:

```
hklF = {{{h1,k1,l1},F1},{{h2,k2,l2},F2},{{h3,k3,l3},F3},…};
```

with the syntax:

```
hklF = Transpose[{hkl,F}];
```

Conversely, hklF may be splitted into hkl and F with the syntax:

```
hkl = Transpose[hklF][[1]];
F   = Transpose[hklF][[2]];
```

Note that the semicolon at the end of each definition tells *Mathematica* not typing the result on the screen (which may correspond to huge outputs with long lists).

5.2 Atomic coordinate organization

Atomic coordinates are naturally organized as a list with the hierarchy :

```
{chain(s) {residues {atoms}}}.
```

If a list of coordinate is obtained from a PDB file, by default it is named `XYZ`, and `XYZ[[2]]` will refer to the second chain, `XYZ[[2,3]]` to its third residue, and `XYZ[[2,3,1]]` to the first atom of the third residue of the second chain. `XYZ[[2,3,1]]` thus corresponds to a list of atomic coordinates of type `{X,Y,Z}`. All other relevant pieces of information for each atom are stored in different 'parallel lists'. For example the list named `QB` stores occupancies `Q` and temperature factors `B`, and `QB[[2,3,1]]` refers to the two-value list `{Q,B}` for the atom with coordinates `XYZ[[2,3,1]]`. The same holds true for the lists `ATOMNAMES` and `ATOMTYPES`. For the sake of consistency and programming generality the same structure is always used, even if one level is not useful in a particular case. For example, for a molecule with two different chains the coordinates are naturally stored as {{chain1},{chain2}}, but for a molecule with one single chain they are still kept as {{chain1}}, not {chain1}.

5.3 Symmetry operation organization

All space groups are stored in a list of sublists, with one sublist for each space group. The necessary information was obtained from the CCP4 package. The information for a specific space group is retrieved as:

```
InfoSpaceGroup["P21"]  (or InfoSpaceGroup[4])
```
(10)

which yields as output the following list:

```
{4, P21, MONOCLINIC, {{{1, 0, 0}, {0, 1, 0}, {0, 0, 1}},
  {{-1, 0, 0}, {0, 1, 0}, {0, 0, -1}}}, {{0, 0, 0}, {0, 1/2, 0}}}
```
(11)

In the fourth position are stored the symmetry matrices and in the fifth position the corresponding translations.

5.4 Reflection organization

Reflections are stored in 'parallel lists'. One of these lists, commonly named `hkl`, stores Miller indeces as a list of individual `{h,k,l}`. Intensities, amplitudes, sigmas, phases, etc… are stored at will of the user in 'parallel lists'. Note that the user is free of grouping whatever he wants in a single list. For example, if F's and sigmas are grouped to form `FSigma`, `FSigma` is made of elements `{F(h,k,l),SigmaF(h,k,l)}`. The two lists `hkl` and `FSigma` are obviously parallel.

5.5 Electron density maps

Electron density maps in 3D are stored in lists of sections, each section being a list of rows, and each row a list of values. They can be readily obtained as such by a call to CCP4 from within *Mathematica*:

```
DensityMap = CalcMapCCP4[hkl,F,PHI,UnitCell,SpaceGroup,(Options)]
```
(12)

where `hkl` is a list of Miller indeces, and `F` and `PHI` the parallel lists of structure factor amplitudes and phases. `UnitCell` is a list of the form `{a, b, c, a, b, g}` and `SpaceGroup` a character string of the form `"P21"`. All options for CCP4 map calculation can be given in a list of `Options`.

6. Examples of 'elementary' manipulations

A few examples will be worked out in some details to illustrate the usage of the stored data, as well as some basic features of the philosophy of *Mathematica* programming, and the density of the code. As already stated before, several technical aspects will not be developed, or will be left in the shadow, when the context is sufficiently clear or when this does not compromise understanding.

6.1 Manipulating electron density maps

Manipulating an electron density map as a whole is a trivial operation in *Mathematica*. For example, applying a mask mask onto a density map map for solvent flattening is obtained as:

```
FlatMap =  mask*map ;                                    (13)
```

Predefined interpolation tools allow to determine functional approximations of a map very efficiently. This allows, for example, to resample a map at will on any 'skewed' mesh for molecular replacement. Such tools seem promising, also in the scope of molecular replacement, for applying smooth distortions on a map by using normal-mode-like methods as already described (Delarue & Dumas, 2004; Suhre & Sanejouand, 2004). We think that there exist considerable possibilities in this area because of the great efficiency of many preexisting tools in *Mathematica.* This will not be detailed further here.

6.2 Calculating scattering factors

The scattering factor of a particular atom type is obtained as `InfoFormFactor["atomtype"]`, `"atomtype"` representing a character string. For example, `InfoFormFactor["Ca"]` returns the information for calcium:

```
{Ca,{{8.6266,10.4421},{7.3873,0.6599},{1.5899,85.7484},{1.0211,178.437},{
1.3751,0.}},{0.341,1.286,0.203,0.306}}
```

where the parameters for the usual four-gaussian approximation (International Tables, Vol 4, 1974) are returned as the second element of the latter list, that is to say:

```
GaussCa = InfoFormFactor["Ca"][[2]];                     (14)
```

which contains the following numerica values:

```
{{8.6266,10.4421},{7.3873,0.6599},{1.5899,85.7484},{1.0211,178.437},{1.3751,0.}}
```

Summing up the five functions $g(Q_i, B_i, s) = Q_i e^{-B_i s^2}$ (with $s = \sin\theta / \lambda$), corresponding to the five elements of form `{Qi, Bi}` in the latter list `GaussCa` (note that the fifth Gaussian function is in fact a constant), can be performed with the self-explanatory usual 'procedural' syntax:

```
Sum[g[GaussCa[[i,1]], GaussCa[[i,2]], s], {i, 1, 5}]     (15)
```

Although it appears quite cryptic at first sight, it is much more preferable to use the following functional method. First, one 'maps' the function g onto the list GaussCa, that is onto each element of GaussCa:

```
Map[g[#[[1]],#[[2]],s]&, GaussCa]                    (16)
```

which yields as result the following list:

```
{Q1*Exp[-B1 s^2], Q2*Exp[-B2 s^2], Q3*Exp[-B3 s^2], …}    (17)
```

with Q1=8.6266, B1=10.4421, Q2=7.3873, B2=0.6599, etc… In (16) the syntax #[[1]] and #[[2]] stand for the arguments of a function (here, the function g) by referring to each element of the list it is mapped onto. In the present case, each element of GaussCa is a sublist of the kind {Qi_i, Bi}, and #[[1]] thus refers to Qi, and #[[2]] to Bi. Note also the character '&' that is the mark of the end of the definition of what is called a 'pure function' in *Mathematica*. The usage of # for referring to variables, and of '&' for marking a pure function cannot be dissociated. This syntax allows using a function without predefining it. In some sense, a pure function may be viewed as a 'disposable' function. For example, the sum of the five functions $g(Q_i, B_i, s) = Q_i e^{-B_i s^2}$ can readily be invoked as follows (note the use of the trace operator Tr for summing up the elements of a 1D list):

```
diffus[GaussX_List][s_]:=Tr[Map[#[[1]]*Exp[-#[[2]]*s^2]&,GaussX]  (18)
```

to define the value of the scattering factor of any element X with parameters stored in GaussX. The form used in (18), *i.e.* the mapping of a pure function onto a list, is equivalent to that used in (15) as far as the result is concerned, but (18) has the advantage of yielding very often the most concise code, and of never requesting explicit handling of the limits of a list. Since *Mathematica* has symbolic capabilities, GaussX may be replaced with a list of symbols like {{Q1,B1},{Q2,B2},{Q3,B3},{Q4,B4},{c,0}} to obtain a purely analytic expression.

6.3 Sorting out centric and general reflections

As another example, we will consider a concise way (and even also a very concise way) of extracting all centric reflections from a list of reflections. A reflection {h,k,l} is centric if there exists a symmetry operation whose matrix m is such that :

```
{h,k,l} + Transpose[m].{h,k,l}  = {0,0,0}                (19)
```

One thus first maps the pure function (# + Transpose[m].#)& onto the full list of reflections hkl and then searches for positions of {0,0,0} in the result; this can be made a function of m and hkl as follows:

```
CenPos[m_List,hkl_List]:=Position[Map[#+Transpose[m].#&,hkl],{0,0,0}]  (20)
```

Then, one can map this new function onto the list Sym of all symmetry matrices for the space group of interest (see **§ 5.3**), and consider the union of the whole set of positions:

```
CenPositions = Apply[Union, Map[CenPos[#, hkl]&, Sym]];
```

From this list of positions in hkl, one can then extract all centric reflections from hkl:

```
Centric = Map[Extract[hkl,#]&, CenPositions];
```

In the package, these three lines of code are grouped into a callable function (named `CentricHKL`), and the list of centric reflections is returned at once as:

```
Centric = CentricHKL[hkl,SpaceGroup];
```

In fact, these three lines of code can even be reduced to a bit cryptic 'one-liner':

```
Union@@Table[Cases[hkl,_?(#+Transpose[Sym[[i]]].#=={0,0,0}&)],{i,2,Length[Sym]}]
```
(21)

This is only shown here to illustrate how concise the code can be and this will not be discussed further. The former three-line version is used in the package because it turns out to be faster. This example is only a very short glimpse at the powerful capabilities of *Mathematica* for sorting and extracting information from any kind of lists.

7. Conclusion

In its present state, *BioCrystallographica* allows performing original methodological tests because all basic tools for doing crystallography are available and embedded in a very powerful environment. We hope a significant number of persons among those interested in methods will participate in developing it. We consider as an interesting goal the extension of links between *BioCrystallographica* and other crystallographic softwares as this is already done with CCP4. Clearly, reinventing the wheel and reprogramming already available tools has no interest, but making them available in an original powerful environment is of great interest. For the time being the link to external programs (like CCP4) has been tested under Windows. Extending this functionality to Unix-, or Linux-based platforms should be straightforward.

The package is freely available for academic users upon request. We have made a web site to illustrate our work with *Mathematica* in crystallography. For the time being, it shows essentially teaching applications. We plan to enhance the functionalities in a near future.

N. Ambert (2003) and J. Vanwinsberghe (2006) made each a four-month student work.

References

COLLABORATIVE COMPUTATIONAL PROJECT, NUMBER 4. 'The CCP4 Suite: Programs for Protein Crystallography'. (1994) *Acta Cryst*. **D50**, 760-763.

Brünger, A.T., Adams, P.D., Clore, G.M., DeLabo, 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). *Acta Cryst.* **D54**, 905-921.

Delarue, M. & Dumas, P. ‘On the use of low-frequency normal modes to enforce collective movements in refining macromolecular structural models’. (2004) *PNAS* **101**, 6957-6962.

International Tables for X-ray crystallography, Vol **4** (1974) The Kynoch press, Birmingham, pp. 99-101.

Suhre, K & Sanejouand Y.H. ‘On the potential of normal mode analysis for solving difficult molecular replacement problems’, (2004) *Acta Cryst.* ***D*****60**, 796-799.

The "Buccaneer" protein model building software.

Kevin Cowtan

York Structural Biology Laboratory, University of York, York YO10 5YW, UK

Introduction

"Buccaneer" is a new software tool to trace protein structures in electron density maps by identifying connected alpha-carbon positions using a likelihood-based density target. At this stage it does not do refinement or rebuilding. Until these are implemented it will not be directly comparable to existing automated model building packages. However, it is quite fast, and can work at low resolutions given good phases.

Method

The underlying principle of "Buccaneer" is the repeated application of an electron density likelihood function to identify probable oriented Ca positions in a noisy electron density map. The same likelihood is applied in several ways: to find candidate positions, using a six dimensional search, to grow a chain by adding new residues either side of an existing Ca position, and to refine Ca positions.

The density likelihood function

The density likelihood function is used to recognise characteristic features corresponding to a Ca position in a sphere of density whose radius is 4.0Å. This approach is related to that of CAPRA (Ioerger and Sacchettini, 2002), however in this case oriented electron density features are used. The density at a candidate position and orientation are scored in terms of a target function based on the mean electron density averaged over many Ca positions in an electron density map from a known structure. However, the score is also weighted according to how conserved the density is at any position relative to the Ca in the known structure, with strongly conserved density given a high weight (i.e. it must match the target density well), and poorly conserved density a low weight (i.e. it doesn't matter whether it matches). This information is combined in a likelihood function using the method of Cowtan (2001).

The likelihood function can therefore be described in terms of an expected density and a weighting. These are visualised in figures 1-3. Figure 1 shows a set of superimposed Ca positions from a known reference structure, along with the 4.0Å sphere of interest. Figure 2 shows the mean density corresponding to those atoms, and Figure 3 the variance density, contoured to highlight the most conserved regions. Note that the most conserved regions are not necessarily those of highest density: regions of low density can also provide important information in locating Ca positions.

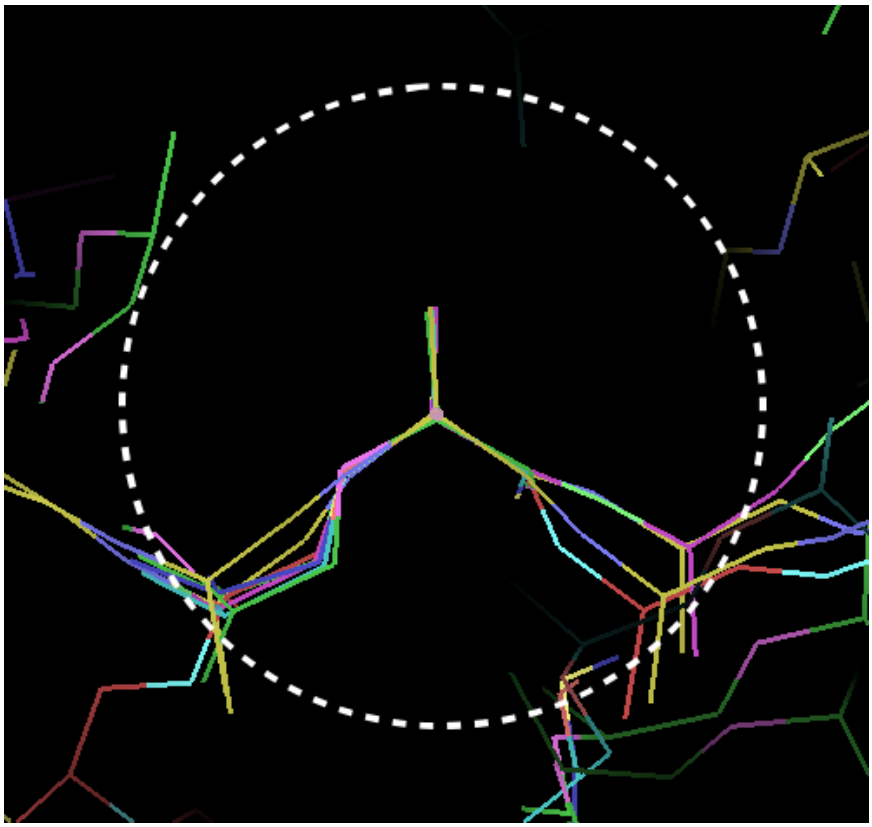


Figure 1: Superimposed Ca positions from a known reference structure.

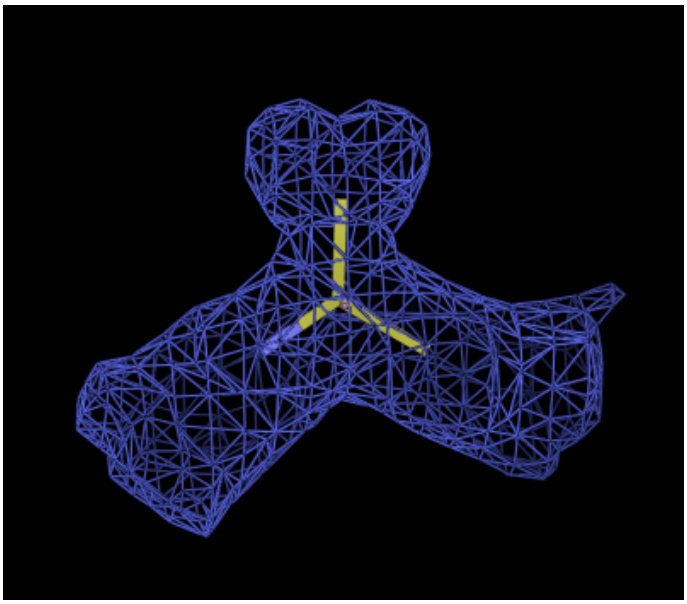


Figure 2: Mean density calculated over many Ca groups.

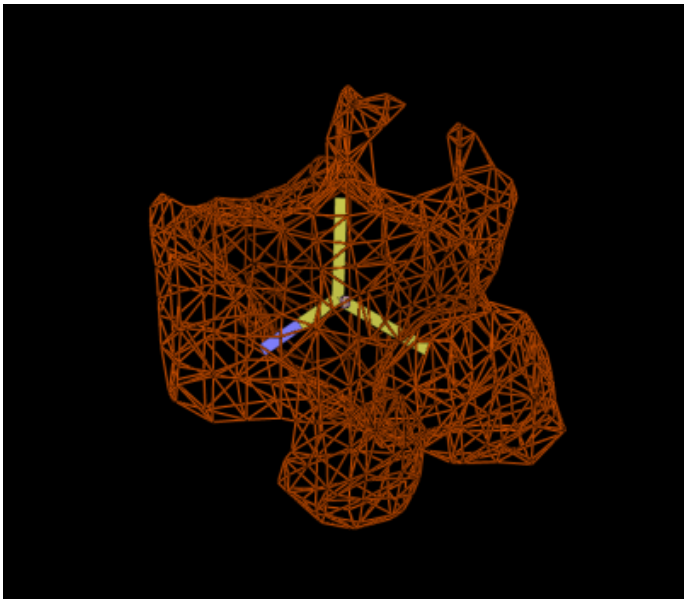


Figure 3: Variance density calculated over many Ca groups.

It is important that the search target function should represent the features to be identified in the target electron density map as well as possible. As a result, the electron density target function is generated from scratch for every calculation. A known, reference structure, for which an atomic model is available, is used. A simulation process is used to create an electron density map for the reference map whose features match the target map as well as possible. The resolution, the atomic motion, and the noise level in the reference map must therefore be comparable to those of the target map to be solved. This simulation process involves a number of scaling steps, and addition of a noise term to the reference data based on the figures-of-merit of the target data. The calculation is therefore dependent on realistic error estimates (i.e. figures-of-merit) being available for the structure to be solved.

Since the electron density features corresponding to a Ca position in the simulated reference map are known, these may therefore be used to identify probable Ca positions in the target map. This inference is illustrated in Figure 4.

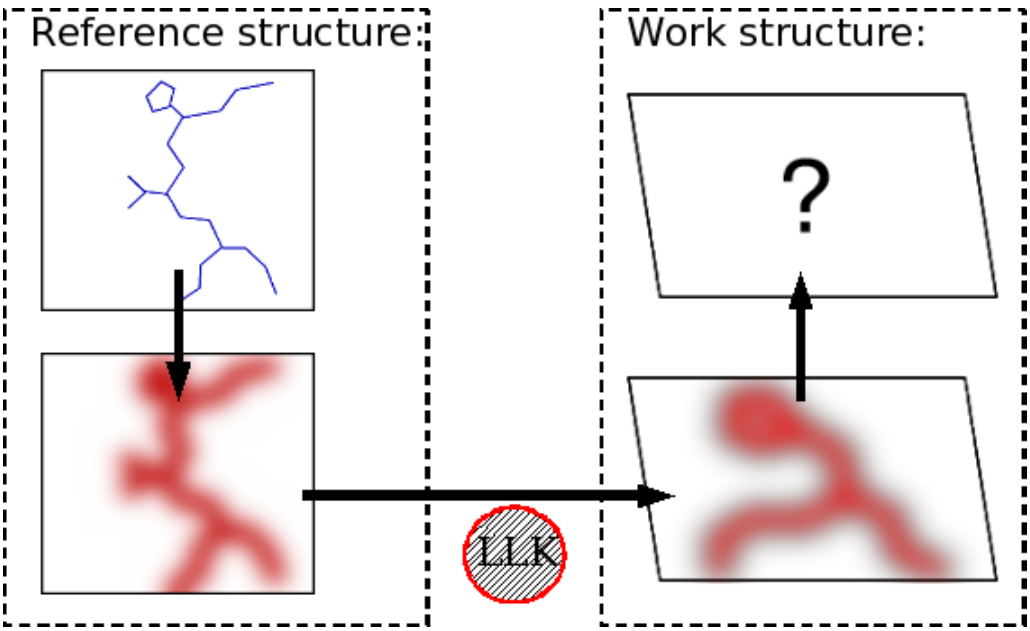


Figure 4: Inference of Ca in the work map based on features of the reference map.

Steps in the calculation.

Finding candidate Ca positions

Candidate Ca positions are located by searching the electron density. A 6-dimensional position and orientation search are conducted using FFTs to perform the positional search, using the "fffear" approach (Cowtan, 1998).

Growing candidate Cas into protein fragments

The candidate Cas are grown by adding residues at either end, in accordance with the restraint that the new resulting Ramachandran angles must be plausible. A two-residue look-ahead approach is used to add each new residue, in a manner similar to the approach of Terwilliger (2002).

Joining Fragments into chains

Overlapping fragments are joined to make longer chains. This can lead to branches in the chain. In this case, the chain is routed in such a way to produce the longest connected chain, in a manner similar to Cohen et al (2004). Note that there will also be chain fragments which cannot be joined, for example forward and backward traces of the same chain segment.

Assigning Sequence

Likelihood comparison between the density of each residue in the work structure and the residues of the reference structure allows sequence to be assigned to longer fragments.

Pruning Fragments Clashing

Clashing fragments are examined and the shorter fragment, or the unsequenced fragment is pruned to eliminate the clash.

Rebuilding

The carboynl oxygens, and side chain atoms for any sequenced residues, are added.

Results

The method has been implemented in a software package, "buccaneer", which may be obtained by download from JCSG data archive after phase improvement in "pirate" (Cowtan, 2000) is shown in figures 5-10.

The software has been tested on 58 test structures, and has been found to be useful over a wide range of resolutions down to 3.6Å. However in its current form it is dependent on good starting phases from phase improvement.

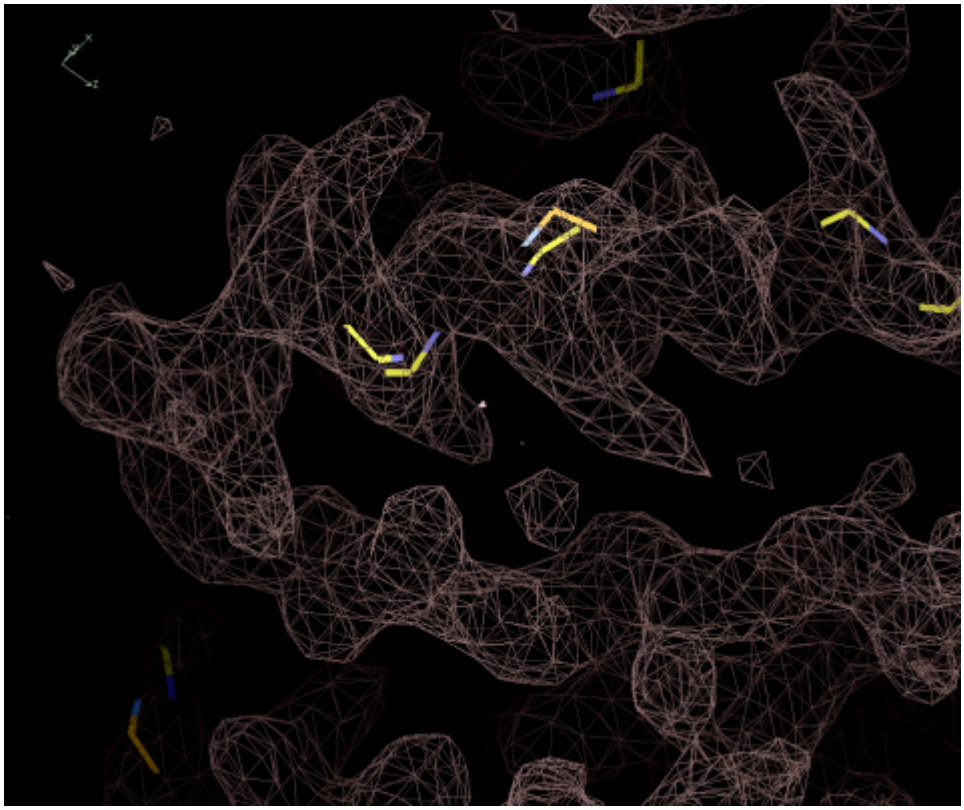


Figure 5: Initial Ca positions from finding. Of the 5 shown, 3 are correct and 2 incorrect.

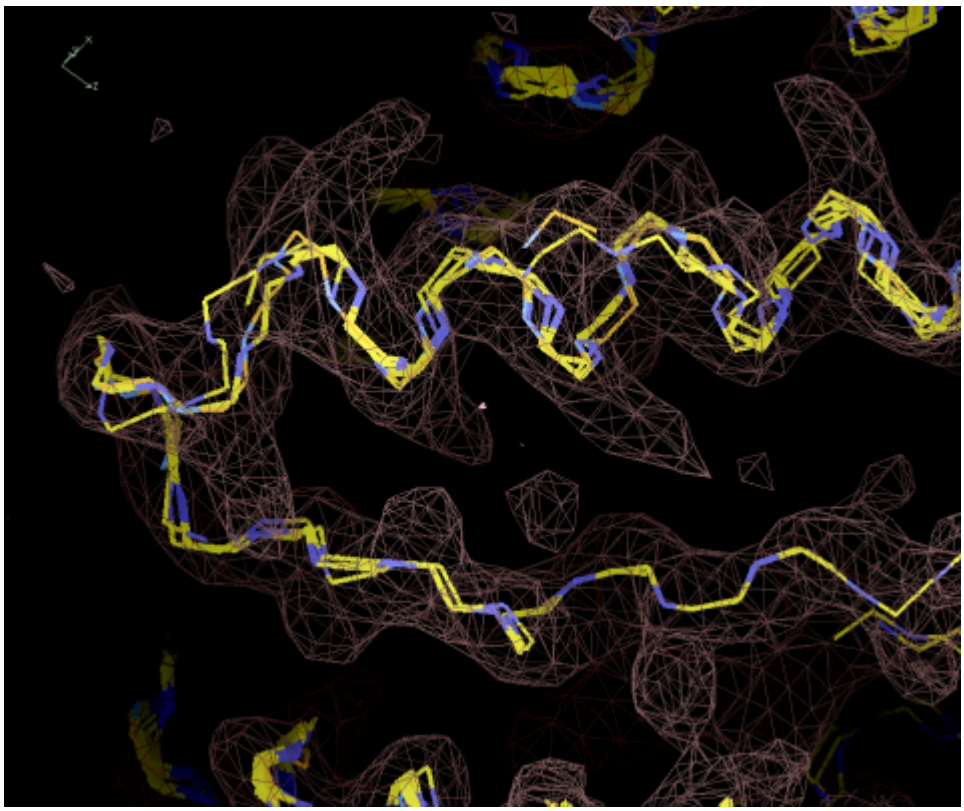


Figure 6: Chain fragments after growing. The same segment is traced multiple times.

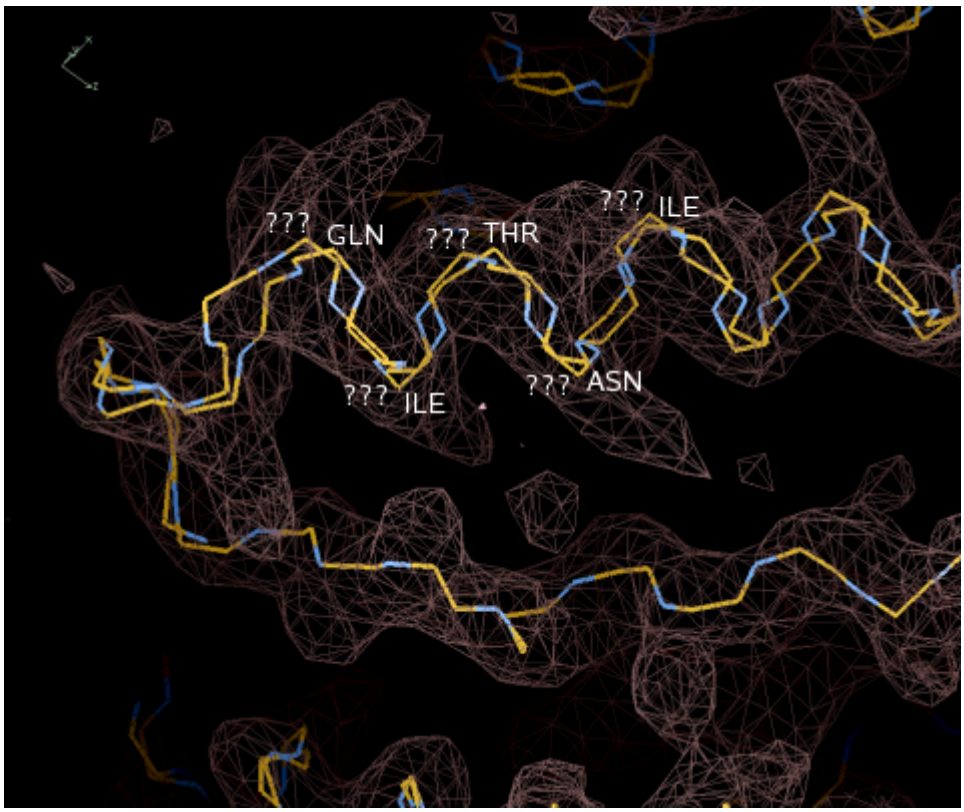


Figure 7: Joined fragments. Note one helix has been traced in both directions. The sequencing step has also been performed, and some residues labeled.

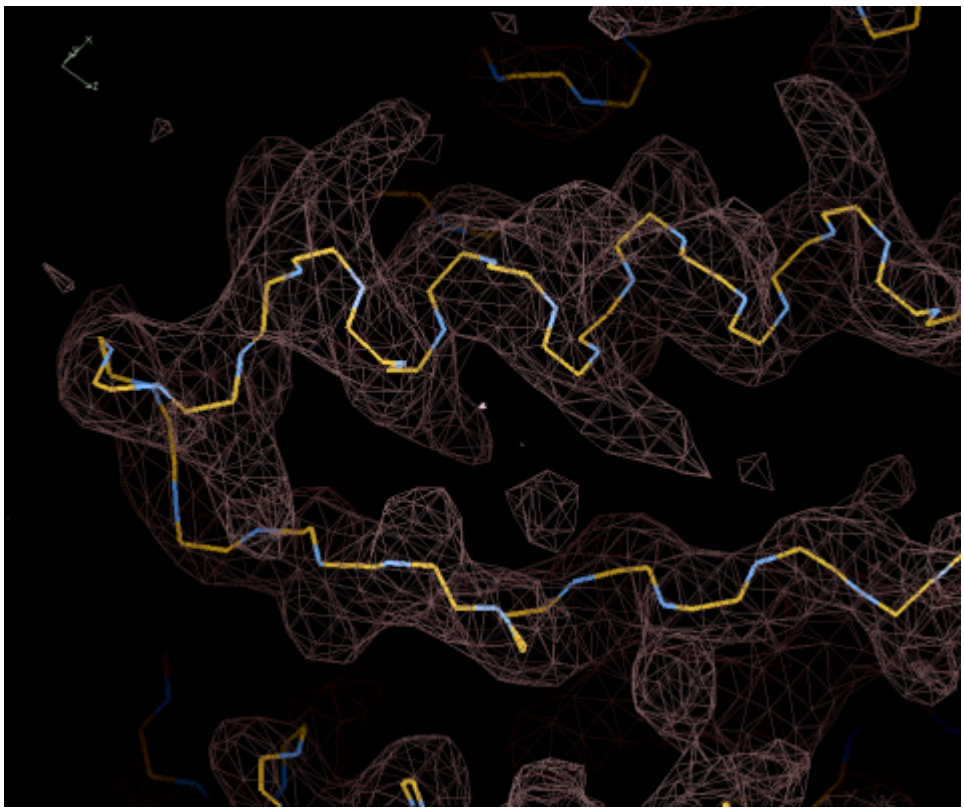


Figure 8: Pruned model. The reversed fragment has been removed on the grounds that it was not sequenced. (It is also shorter.)

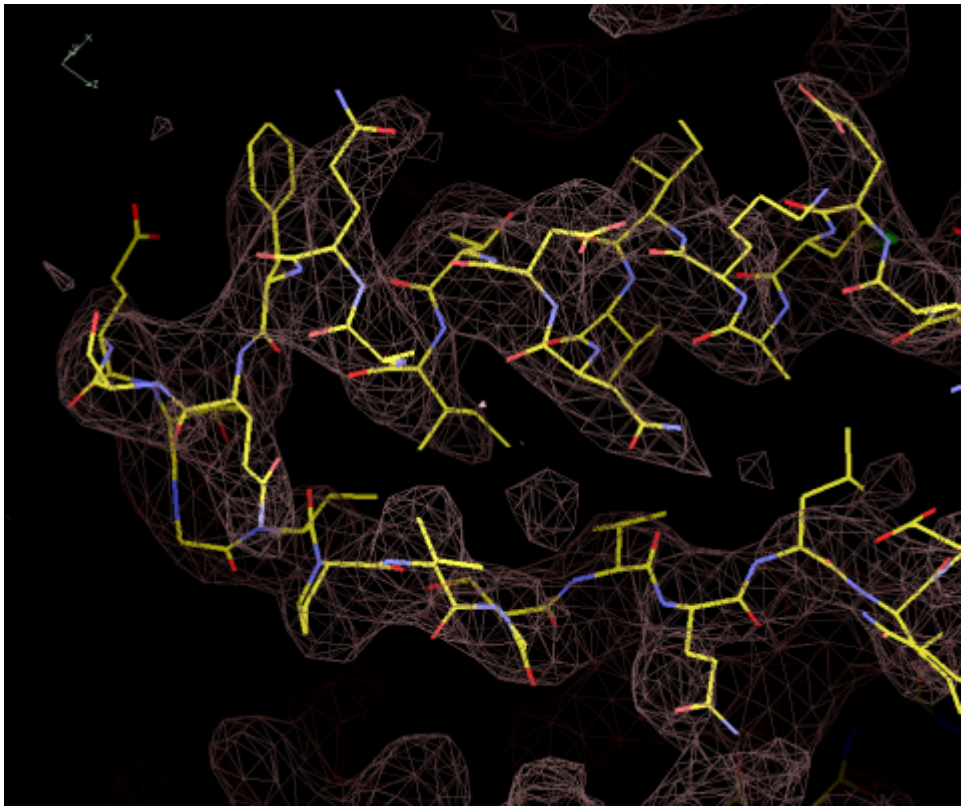


Figure 9: Model after rebuilding, with the side-chain and carboynl oxygens added.

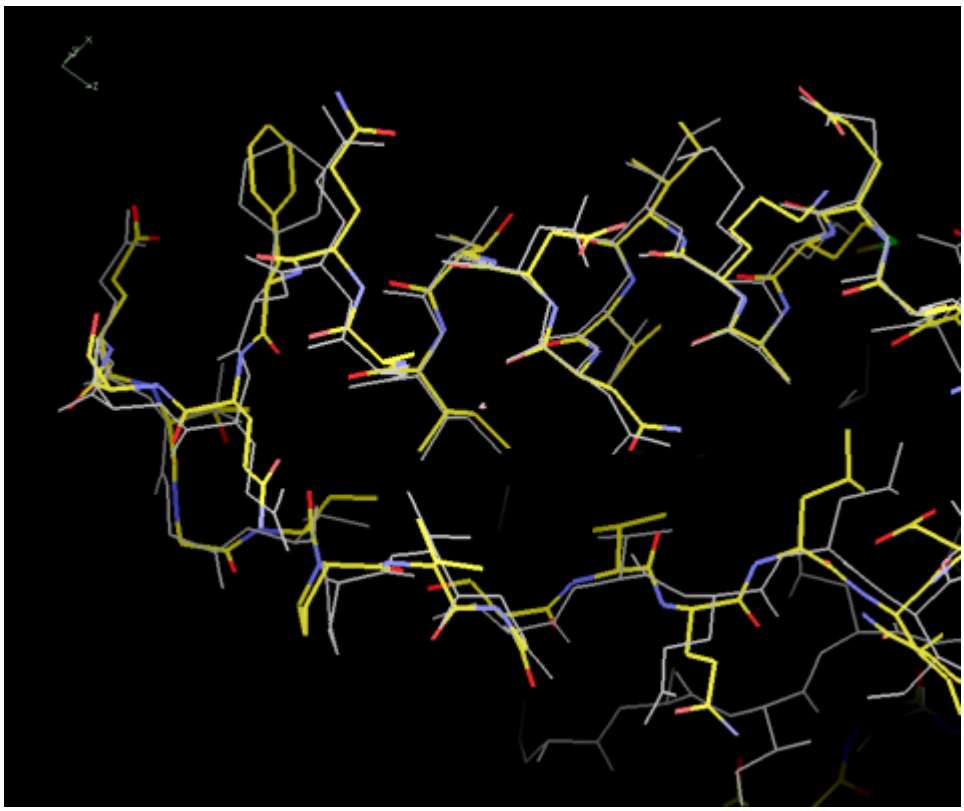


Figure 10: The "buccaneer" model compared to the deposited structure.
The coordinates agree very well given that the resolution is only 3.0Å

Conclusions

The "buccaneer" model building software is extremely simple, relying on the application of a single likelihood function in several different ways to trace and sequence protein main-chains in experimentally phased electron density maps. The method is reasonably fast, taking minutes to an hour, and can give a partial trace even at low resolutions (i.e. worse than 3.0Å). However the method is dependent on the quality of the initial experimental phasing and phase improvement results.

The method as presented here is incomplete, lacking removal of incorrectly traced features, refinement of the resulting model, or recycling to model completion. However "buccaneer" is already usable as a model building tool in conjunction with the "coot" software for structure completion (Emsley and Cowtan, 2004), and provides a suitable basis for future development.

Acknowledgements

This work has been funded by the Royal Society. Dr Cowtan would like to thank Paul Emsley and Eleanor Dodson for their helpful comments, and the JCSG data archive for the provision of test data.

References

- Cowtan K. (1998) Acta Cryst. D54, 750-756.
  Modified phased translation functions and their application to molecular fragment location.
- Cowtan K. (2000) Acta Cryst. D56, 1612-1621.
  General quadratic functions in real and reciprocal space and their application to likelihood phasing.
- Cowtan K. (2001) Acta Cryst. D57, 1435-1444.
  Fast Fourier feature recognition.
- Ioerger T. R., Sacchettini J. C. (2002) Acta Cryst. D58, 2043-2054.
  Automatic modeling of protein backbones in electron-density maps via prediction of Ca coordinates
- Terwilliger T. C. (2002) Acta Cryst. D59, 34-44.
  Automated main-chain model-building by template-matching and iterative fragment extension.
- Emsley P., Cowtan K. (2004) Acta Cryst. D60, 2126-2132.
  Coot: model-building tools for molecular graphics.
- Cohen S. X., Morris R. J., Fernandez F. J., Ben Jelloul M., Kakaris M., Parthasarathy, Lamzin V. S., Kleywegt G. J., Perrakis A. (2004) Acta Cryst. D60, 2222-2229.
  Towards complete validated models in the next generation of ARP/wARP

Release 0.4 of PIMS

Chris Morris

CCP4, Daresbury Laboratory, Warrington WA4 4AD, UK

Introduction

The PIMS project aims to produce a commercial-quality Laboratory Information Management System (LIMS) to meet the needs of structural biology laboratories.

The project, funded by the BBSRC-SPoRT initiative for 5 years, is for the incremental development and deployment of a LIMS, initially for use by the other BBSRC-funded SPoRT consortia: SSPF and MPSI, but to be freely available to the scientific community as a whole.

Designed with flexibility in mind, PIMS will be suitable for use in all academic environments ranging from the single, hypothesis driven laboratory to multi-site and automated high-throughput settings.

Version 0.4 of Protein Information Management System was released on 24th May 2006.

What's New

PiMS 0.4 includes facilities to manage constructs. Thanks to Johan van Niekerk of the SSPF for this functionality, and also to Jon Diprose of the OPPF, as well as the PiMS developers. It also includes diagrammatic displays of target progress. It will be used by the Scottish Structural Proteomics Facility.

PiMS 0.3 already included facilities for managing targets. It is in use by the Membrane Protein Structure Initiative.

There is an online demonstration available at http://www.pims-lims.org:8080/pims0_4/index.html. To use this demo the username is demo, password is demo. Later on example data, courtesy of SSPF, will be added to it.

Installation

To install PIMS version 0.4 you just need to take the java WebArchive at http://www.pims-lims.org/pims0_4.war and follow the instructions from http://www.pims-lims.org/svn/pims/releases/V0_4_0/pims-servlet/dist/docs/README.txt

PIMS is available under the Lesser GNU Public License (LGPL).

For further information please see

http://www.pims-lims.org

Portable Running of Programs for Automation

G Winter*, R. Keegan†, CCLRC Daresbury Laboratory

July 6, 2006

1 Introduction

Almost any automation in protein crystallography structure solution will involve running existing programs at some stage - the only alternative is to rewrite the substantial body of programs which is available. If we therefore assume that running programs is desirable, the only problem is then the matter of how.

Some systems, for example autoSHARP [Bricogne et Al., 2002] and Elves [Holton & Alber, 2004], rely on a UNIX shell for this task - a reliance which has typically been reasonably portable. However, in recent years the nature of crystallography and the crystallographer has changed - more people are now using Microsoft Windows for "real" computing tasks than ever before.

The nature of crystallographic computing is also changing. It used to be usual to do all of the computing on a central server - now it's more likely to be a personal workstation or a cluster of computers. These environments are both moderately hostile to script based solutions, though can be catered for.

Here, we describe a more elegant solution, where the running of the program is separated from the rest of the system, in such a way as to allow extension to cover a variety of platforms. Implementation of this system in Python[1] is also discussed, along with some experiences of using the system in anger.

2 The Problem

As described in the introduction, developing a system to automate a protein crystallography process often depends on interaction with the operating system to run programs. This interaction can be difficult to port from one platform to another, and can therefore limit the possible users of the system.

*g.winter@dl.ac.uk
†r.m.keegan@dl.ac.uk
[1]http://www.python.org

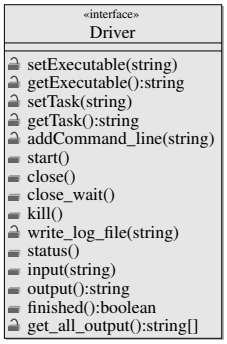


Figure 1: The "Driver" interface.

In general, when running programs for automating a crystallographic process, we are interested in the following:

- Starting the program
- Providing program input
- Reading program output
- Deciding if the program has worked
- Handling input and output files
- Handling cases where things go wrong, for example killing the program

Of course, almost all systems will interact with more than one program, so chaining will be important. This is however out of the scope of this discussion.

Developing portable tools to do this in addition to developing the "interesting stuff" is not a trivial undertaking.

3 Conceptual Solution

Once we have decided on the functionality we want for running programs, it is relatively straightforward to design an interface. Given an interface, the only problem is then to implement or *realize* this interface, to give us something we can use. The interface described above we call "Driver", and is shown as a UML diagram in Figure 1.

Since there are a number of ways we could implement this interface, and a number of environments where this could take place, the next task is to design a mechanism to hide these details from the application. A useful metaphor or pattern[2] here is the factory. We ask the factory to provide

[2]If you want to find out more about this, google "Design Patterns."

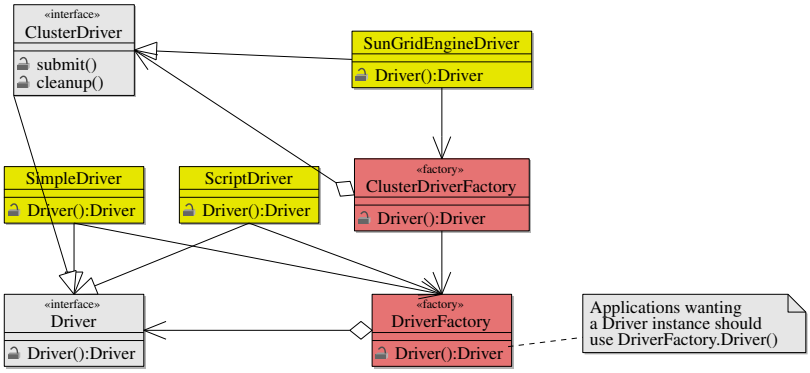


Figure 2: Accessing Driver implementations from a factory.

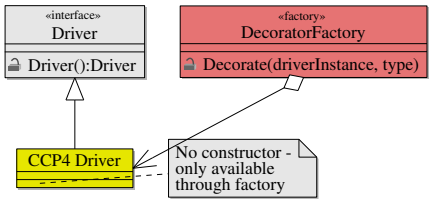


Figure 3: Customizing Driver classes for e.g. CCP4.

something which satisfies the interface Driver, and it provides such a thing. This is illustrated in Figure 2. In this way the factory can be delegated to provide the most useful implementation for the current environment.

In some cases there will be sets of Driver implementations which have a lot in common. For example, batch queuing systems on clusters all work by submitting a script. Writing the script and handling the status of the job is a general problem, while the details of the submission will depend on the exact cluster. We can therefore define a new interface, ClusterDriver, and a new factory to produce them. This factory can then be called by the "general" DriverFactory when a Driver is wanted in a cluster environment.

Although the functionality we want *in general* is described above, there is often a great deal of commonality between programs in a suite. For example, in CCP4 most programs accept the "logical names" HKLIN, HKLOUT. We could implement this every time we want to write a program, but this will be both untidy and inefficient. A more elegant solution is to take the Driver instance and "decorate[3]" this with the things which are nice for running CCP4 programs, in particular methods for assigning HKLIN & HKLOUT files, checking the status and parsing loggraph output. However, unlike standard inheritance, this must be able to decorate anything which implements the Driver interface. This is shown in Figure 3.

So far this has involved no programming - we are simply describing a

[3]Another design pattern

set of requirements and designing a package which would be able to satisfy those requirements. What we have described here could be implemented in any moderately modern language. In the next section we describe the Python implementation.

4 Python Implementation

Implementing this system in Python is in some ways straightforward, and in others rather a challenge. Python includes some very nice tools for interacting with the operating system, and more often than not these are portable across the "usual suspect" platforms of Linux, Mac OS X and Windows. However, the language does not include the concept of a virtual class or interface, as used above. Some interesting "spells" were therefore necessary to give this kind of functionality.

Defining the interface is relatively straightforward - simply write a Python class which looks like what you want, have all of the useful methods raise exceptions like "implement me", and insist that any Driver implementations inherit from this. A simple check of the class hierarchy can test this.

In python a factory is very simple - this is simply a function which returns a fresh instance of a class. Implementing the DriverFactory as seen above therefore only needed one detail - how to specify the default driver type. This was solved by the old-fashioned approach of setting an environment variable XIA2CORE_DRIVERTYPE.

Implementation of the Decorator pattern was less straightforward. Without interfaces, a class implementation (lilke CCP4Driver) must inherit from a concrete implementation of a Driver class. This presented two possibilities - either implemnenting $m \times n$ classes like CCP4ScriptDriver, or investigating dynamic inheritance. The latter was selected in an instant!

Implementing a system using dynamic inheritance looks a lot like a factory function - you have a function that you call which will return a new instance of a class. The only difference here is that you pass in an instance of the class that you wish to decorate, giving the following pseudocode:

```
function CCP4DriverFactory(DriverInstance d) returns CCP4Driver
{
  DriverClass = d.class()

  class CCP4DecoratorClass(DriverClass)
  {
    // implement class based on Driver interface in here
  }

  return new CCP4DriverClass()
}
```

In Python, this allows you to decide at construction time what Driver class we wish to inherit from, and so circumvents the challenges described above.

This has only one side effect - when you wish to inherit from the CCP4-decorated driver, you have to have a similar scheme in the code. However, this ends up being a ten-line boilerplate, which is simply an empty wrapper which can be populated with the "interesting stuff". This is a relatively small price to pay for portability.

5 Usage Examples

5.1 xia2dpa

xia2dpa is a toolkit for data processing and analysis, and builds on the Python implementation of the xia2core described above. This makes use of a large number of CCP4 programs as well as programs from outside. In the new development portability was a key criterion, initiating the development of the new core.

xia2dpa is currently being developed in a bottom-up approach. Those components at the lowest level (program wrappers, inherited from Drivers) are combined to give useful modules. An example is the development of the xia2scan application - for analyzing diffraction images when using a humidifier. This makes use of a couple of "standard" programs (labelit.index and labelit.distl [Sauter et Al., 2004, Zhang et Al, 2006]) and also makes use of a "jiffy" application printheader. All of the programs have wrappers in the xia2dpa wrapper library, and so can be treated as functions directly from Python. In this application, results from the indexing and image analysis are combined to select the best image - from here the best humidity for data collection can be selected.

5.2 MrBUMP

MrBUMP is a new CCP4 project for doing automated molecular replacement (MR). It is implemented in Python and wraps around the commonly used MR programs Phaser and Molrep with particular emphasis on generating search models for use in MR. It takes a brute force approach to the problem and attempts to generate as many suitable models as is practical. It is designed to make use of compute clusters to aid this process. Apart from Phaser and Molrep, MrBUMP makes use of several other CCP4 utility programs for preparing the structure coodinate files of the search models such as Chainsaw, Pdbcur, PDbset and Coord_format.

MrBUMP is designed as a framework allowing for additional model preparation methods and MR programs to be incorporated into it as it is further developed. The ease with which the xia2core Driver allows for

the creation of wrappers to programs is ideal for MrBUMPs purposes. In addition the built-in portability afforded by xia2core means that program wrappers can be developed on a single platform and be guaranteed to work on any other platform. Another particularly attractive feature of the implementation is the Cluster Driver that can allow MrBUMP to take advantage of the wide variety of different clustering systems available such Sun Grid Engine, PBS, and LSF to name but a few, without having to be tailored to handle the idosyncrasies of each of these systems.

Currently, MrBUMP has its own built-in wrappers for each of the programs it uses but re-writing these to make use of xia2core has proven to be a fairly trivial task given the simple way in which interfaces can be designed using xia2core. A developer looking to utilise it only needs a very rudimentary understanding of how the underlying Driver and Decorator classes are implemented before they can make use of them. This "low potential barrier" to use is perhaps the most appealing feature of xia2core from a practical point of view.

6 Discussion & Availability

6.1 License

This software is provided under the CCP4 "Applications" license, and should shortly be distributed as a part of the CCP4 suite, in version 6.1.

6.2 Download

The latest version of the xia2core, including the Python implementation described here, is available from http://www.ccp4.ac.uk/xia/xia2core-0.0.3.tar.gz. For information on more up-to-date versions when available, please contact g.winter@dl.ac.uk.

7 References

- Bricogne, G., Vonrhein, C., Paciorek, W., Flensburg, C., Schiltz, M., Blanc, E., Roversi, P., Morris, R. & Evans, G. (2002). Acta Cryst. A58, C239.
- Holton, J & Alber, T, (2004) PNAS 101, 1537-42.
- Sauter, N. K., Grosse-Kunstleve, R. W. & Adams, P. D. (2004). J. Appl. Cryst. 37, 399-409.
- Zhang, Z., Sauter, N. K., van den Bedem, H., Snell, G. & Deacon, A. M. (2006). J. Appl. Cryst. 39, 112-119.

Exploring Metric Symmetry

P.H. Zwart, R.W. Grosse-Kunstleve & P.D. Adams

Lawrence Berkeley National Laboratory, 1 Cyclotron Road, BLDG 64R0121, Berkeley, California 94720-8118, USA. Email: PHZwart@lbl.gov; www: http://cci.lbl.gov

1. Introduction

Relatively minor perturbations to a crystal structure can in some cases result in apparently large changes in symmetry. Changes in space group or even lattice can be induced by heavy metal or halide soaking (Dauter *et al*, 2001), flash freezing (Skrzypczak-Jankun *et al*, 1996), and Se-Met substitution (Poulsen *et al*, 2001). Relations between various space groups and lattices can provide insight in the underlying structural causes for the symmetry or lattice transformations. Furthermore, these relations can be useful in understanding twinning and how to efficiently solve two different but related crystal structures. Although (pseudo) symmetric properties of a certain combination of unit cell parameters and a space group are immediately obvious (such as a pseudo four-fold axis if *a* is approximately equal to *b* in an orthorhombic space group), other relations (e.g. Lehtio, *et al*, 2005) that are less obvious might be crucial to the understanding and detection of certain idiosyncrasies of experimental data.

We have developed a set of tools that allows straightforward exploration of possible metric symmetry relations given unit cell parameters and a space group. The new `iotbx.explore_metric_symmetry` command produces an overview of the various relations between several possible point groups for a given lattice. Methods for finding relations between a pair of unit cells are also available. The tools described in this newsletter are part of the CCTBX libraries, which are included in the latest (versions July 2006 and up) PHENIX and CCI Apps distributions.

2. Methods

2.1. Determination of the lattice symmetry

The determination of the lattice symmetry is based on ideas by Le Page (1982) and Lebedev *et al.* (2006). Given a reduced cell (e.g. Grosse-Kunstleve *et al.* 2004a), it is sufficient to search for two-fold axes to determine the full symmetry. Subjecting the two-folds to group multiplication produces the higher-order symmetry elements, if present.

Le Page (1982) searches for the two-folds by computing angles between certain vectors in direct space and reciprocal space. This search is relatively expensive. Recently Lebedev *et al.* (2006) introduced the idea of simply enumerating all 3x3 matrices with elements {-1,0,1} and determinant one. As an additional requirement group multiplication based on each matrix individually has to produce matrices exclusively with elements {-1,0,1}. There are only 480 matrices that conform to all requirements. Lebedev *et al.* (2006) argue that this set covers all possible symmetry operations for reduced cells. We were able to confirm this intuitive argument empirically via simple brute-force tests.

Only 81 of the 480 selected matrices correspond to two-folds. These are easily detected by establishing which of the matrices produce the identity matrix when multiplied with themselves (and are not the identity matrix to start out with). To replace the expensive search for two-folds in the original Le Page (1982) algorithm, the 81 two-fold matrices are tabulated along with the axis directions in direct space and reciprocal space. The axis direction in direct space is determined as described by Grosse-Kunstleve (1999). The axis direction in reciprocal space is determined with the same algorithm, but using the transpose of the matrix. The complete implementation of the algorithm for generating the table (essentially just six lines of Python code) can be found in the file

`cctbx_sources/cctbx/cctbx/examples/reduced_cell_two_folds.py`

in the cctbx distributions.

The search for two-folds computes the Le Page (1982) δ for each of the 81 tabulated pairs of axis directions. The corresponding symmetry matrix for each pair is immediately available from the table. In contrast, the original Le Page algorithm requires the evaluation of 2391 pairs of axis directions, and the computation of the symmetry matrices involves expensive trigonometric functions (sin, cos) and change-of-basis calculations.

The matrices with a Le Page δ smaller than a given threshold are sorted, smallest δ first. Successive group multiplication as described in Grosse-Kunstleve et al. (2004b) and Sauter *et al.* (2006) yields the final highest lattice symmetry. The complete search algorithm is implemented in the file

`cctbx_sources/cctbx/sgtbx/lattice_symmetry.cpp`.

Since the algorithm determines the lattice symmetry in a primitive setting (the reduced cell), the resulting symmetry matrices do not in general correspond to one of the usual space group settings for which a Hermann-Mauguin symbol is available. However, all lattice symmetries can be exactly characterized with Hall symbols. In contrast to Hermann-Mauguin symbols, Hall symbols uniquely define the orientation of the symmetry elements with respect to the cell axes. This property is vital for understanding relations between different point groups.

2.2. Construction of a point group graph

In order to be able to systematically explore various possible point groups for a given set of unit cell parameters, a graph is constructed that encodes group-subgroup relations between various point groups. A graph is constructed in the following manner:

- The lattice symmetry of the Niggli cell is deduced, as described above.
- Starting from the `base point group` (P1 *or* the point group corresponding to a user specified space group), each symmetry elements from the point group of the lattice not present in the base point group is combined with the base point group to generate a supergroup.
- The former procedure is carried out for the newly generated point groups. The procedure is iterated until no new point groups are generated. The relations between the constructed point groups are described in a graph.

The resulting point group graph can be used to visualize relations between various symmetries or could be used to systematically assess the validity of a number of intensity symmetry hypotheses. It is important to note that the set of relations generated in this manner only describe the change in point group upon addition or removal of rotational symmetry, but does *not* cover changes due to addition or removal of translational symmetry.

2.3 Generation of compatible space groups

Only a limited set of space groups are compatible with a given point group of the lattice. This set of space groups compatible with a certain point group is for instance used in molecular replacement when the space group is not known.

To further limit the set of set of compatible space groups, it is assumed that systematic absences dictated by the user supplied space group have to be retained. However, this set of systematic absences is not assumed to be complete.

The list of possible space groups derived for each point group is constructed while ensuring that systematic absent Miller indices in the user supplied space group will also be absent. Non-absent reflections in the user supplied space group are not required to be non-absent in the new space group.

A simple example illustrates the described rules in the selection of possible space groups. If the symmetry of the lattice is `P 2 2 2` and the user-supplied space group is `P 21 21 2`, the possible space groups are `P 21 21 2` and `P 21 21 21`. The space group `P 2 2 2`, 3 settings of `P 2 2 21` and 2 settings of the space group `P 21 21 2` all violate the systematic absences dictated by the user-supplied space group.

2.4. Sub-lattice generation

Although the above methods allow the user to inspect the effect of removal or addition of rotational symmetry in the lattice, space group transitions often involve a change in the reduced cell volume as well.

Generation of all sub-lattices that result in a certain volume change, can be obtained by generating all matrices $\mathbf{M}$ with integer indices that have a determinant equal to the desired change in volume of the reduced cell. The resulting sublattice is than obtained from the original lattice via the relation:

$$\mathbf{L}' = \mathbf{L}\mathbf{M}$$

Generating all matrices with a specified determinant in a naive manner by looping over all matrix elements within in fixed range (say from -5 to 5) is however impractical (approximately 2.4 10^9 iterations). A more practical solution lies in the fact (Billet & Rolley Le-Coz, 1980; Rutherford, 2006) that it is sufficient to only generate matrices in the Hermite normal form:

$$\mathbf{M} = \begin{pmatrix} a & d & e \\ 0 & b & f \\ 0 & 0 & c \end{pmatrix}$$

The elements of $\mathbf{M}$ are all integers. The elements a, b and c are all larger than zero. The restrictions on the values of d, e and f depend on the diagonal element of the row in which the elements occur :

$$d, e \in [-(a+1)/2, -(a+1)/2+1, \cdots, (a+1)/2] \ a \text{ odd}$$
$$d, e \in [-(a/2), -(a/2)+1, \ldots, (a/2)] \ a \text{ even}$$

Similar restrictions for f apply.

Because the determinant (and thus the resulting volume change) of the matrix is equal to abc, all matrices with a determinant Δ can be constructed by generating all triplets (a, b, c) for which $\Delta = abc$ holds. Generation of triples is done using an algorithm known as trial division.

After generating the new lattice as described above, the new unit cell parameters can be compared to a *target* unit cell. The new unit cell is reduced to a Niggli cell. The new Niggli cell cannot be compared directly to the *target* Niggli cell, because small differences in unit cell parameters, can result in large differences in reduced cell parameters (Andrews *et al.*, 1980). Therefore, all unimodular transforms with matrix elements in the range from -1 to 1 are generated. After using these matrices to transform the new Niggli cell, it considered similar to the *target* Niggli cell if user-defined tolerances on length and angular deviations are fulfilled. An example of this procedure is shown in section **3.2**, where two unit cells are compared. The Niggli cell with smallest volume will be used as a building block (denoted `Lego cell` in the output) to try and construct a unit cell similar to the Niggli cell with the larger volume (denoted as `Target cell` in the output).

3. Examples

3.1. Incorrectly processed Insulin

An insulin dataset (kindly provided by C.B. Trame, Berkeley Center for Structural Biology) was purposely incorrectly indexed and scaled.

As a test, the data was indexed, integrated and scaled in P1. The resulting cell is equal to:

```
68.4 68.4 68.3 109.5 109.4 109.5
```

As the symmetry of the lattice is cubic (`Hall:  I 4 2 3 (y+z,x+z,x+y)`), the resulting point group graph contains all point groups between cubic and anorthic (P1) symmetry.

The point group graph generating algorithm is used in Xtriage (Zwart *et al*, 2005) in order to determine *missing rotational symmetry*. A scoring function based on R-values (similar to those used by Labelit (Sauter *et al.*, 2006)) is available to help guide the user to the most likely point group:

```
-------------------------------------------------------------------------------------------
| Point group                    | mean R_used | max R_used | mean R_unused | min R_unused | choice |
-------------------------------------------------------------------------------------------
| Hall:  I 4 (y+z,x+z,x+y)       | 0.309       | 0.436      | 0.280         | 0.042        |        |
| P 1                            | None        | None       | 0.291         | 0.042        |        |
| Hall:  C 2y (2*y,x+y+z,x+y)    | 0.432       | 0.432      | 0.260         | 0.042        |        |
| Hall:  C 2y (x+y,z,x-y)        | 0.436       | 0.436      | 0.295         | 0.042        |        |
| Hall:  C 2y (x+y,-x+y+z,z)     | 0.044       | 0.044      | 0.306         | 0.042        |        |
| Hall:  R 3 (2*x-y,x+y,x+z)     | 0.043       | 0.043      | 0.271         | 0.042        |        |
| R 3 2 :R                       | 0.435       | 0.436      | 0.433         | 0.432        |        |
| Hall:  C 2y (z,x-y,x+y)        | 0.435       | 0.435      | 0.259         | 0.042        |        |
| Hall:  R 3 (x+y,x+z,2*x-y)     | 0.044       | 0.044      | 0.270         | 0.042        |        |
| Hall:  R 3 (x+z,2*x-y,x+y)     | 0.042       | 0.042      | 0.269         | 0.042        |        |
| Hall:  I 4 2 (y+z,x+z,x+y)     | 0.340       | 0.436      | 0.240         | 0.042        |        |
| Hall:  F 2 2 (-x+y+z,2*z,x+y+z)| 0.435       | 0.436      | 0.360         | 0.044        |        |
| Hall:  I 4 (x+z,x+y,y+z)       | 0.307       | 0.435      | 0.280         | 0.042        |        |
| Hall:  R 3 2" (2*x-y,x+y,x+z)  | 0.176       | 0.434      | 0.434         | 0.433        |        |
| Hall:  I 4 2 (x+z,x+y,y+z)     | 0.339       | 0.435      | 0.434         | 0.433        |        |
| Hall:  I 4 2 3 (y+z,x+z,x+y)   | 0.291       | 0.436      | None          | None         |        |
| Hall:  F 2 2 (2*z,x-y+z,x+y+z) | 0.433       | 0.435      | 0.359         | 0.044        |        |
| Hall:  C 2y (x+y,2*y,x+y+z)    | 0.434       | 0.434      | 0.296         | 0.042        |        |
| Hall:  C 2y (z,x+y,-x+y+z)     | 0.034       | 0.034      | 0.281         | 0.042        |        |
| Hall:  R 3 2" (x+y,x+z,2*x-y)  | 0.434       | 0.436      | 0.176         | 0.042        |        |
| Hall:  I 2 2 3 (y+z,x+z,x+y)   | 0.043       | 0.044      | 0.433         | 0.433        | <---   |
| Hall:  C 2y (x-y,x+y,z)        | 0.435       | 0.435      | 0.224         | 0.034        |        |
| Hall:  F 2 2 (x-y+z,x+y+z,2*z) | 0.242       | 0.433      | 0.205         | 0.042        |        |
| Hall:  I 4 (x+y,y+z,x+z)       | 0.304       | 0.435      | 0.280         | 0.042        |        |
| Hall:  R 3 2" (x+z,2*x-y,x+y)  | 0.434       | 0.435      | 0.175         | 0.042        |        |
| Hall:  C 2y (-x+y,z,x+y+z)     | 0.042       | 0.042      | 0.281         | 0.042        |        |
| Hall:  I 4 2 (x+y,y+z,x+z)     | 0.244       | 0.435      | 0.434         | 0.433        |        |
| R 3 :R                         | 0.042       | 0.042      | 0.269         | 0.042        |        |
| Hall:  I 2 2 (y+z,x+z,x+y)     | 0.043       | 0.044      | 0.305         | 0.042        |        |
| Hall:  C 2y (x+y+z,x+y,2*y)    | 0.433       | 0.433      | 0.188         | 0.034        |        |
-------------------------------------------------------------------------------------------


R_used: mean and maximum R value for symmetry operators *used* in this point
        group
R_unused: mean and minimum R value for symmetry operators *not used* in this
          point group
The likely point group of the data is:  Hall:  I 2 2 3 (y+z,x+z,x+y)

Possible space groups in this point groups are:
  Unit cell: (78.9732, 78.9732, 78.9732, 90, 90, 90)
  Space group: I 2 3 (No. 197)

  Unit cell: (78.9732, 78.9732, 78.9732, 90, 90, 90)
  Space group: I 21 3 (No. 199)
```

3.2. Unit cell comparison

The presence of pseudo symmetric properties of a lattice associated with a set of unit cell parameters, is often discovered at the stage of indexing and scaling the data. If however a derivative or crystal from slightly different crystallization conditions is available, the presence of a relation to the native or other crystal form is often not immediately clear. Using the sublattice tools discussed in section **2.4**, relations between two unit cells can easily be determined.

For example, Poulsen *et al.* (2001) observed crystal of the native protein and two Se-Met derivatives with the following cell parameters and space groups:

```
Native :  P 21 21 21    61.8   97.7  148.9  90  90  90
SeMet1 :  P 1 21 1     115.5  149.0  115.6  90 115  90 (pseudo hexagonal)
SeMet2 :  C 2 2 21     123.6  195.4  148.9  90  90  90
```

Comparing Native and SeMet1

Using the command

```
iotbx.explore_metric_symmetry --unit_cell="61.8 97.7 148.9 90 90 90"
   --space_group=P212121 --other_unit_cell="115.5 149.0 115.6 90 115 90"
   --other_space_group=P21 --no_point_group_graph
```

The following output is produced:

```

--------------------
Unit cell comparison
--------------------

The unit cells will be compared. The smallest Niggli cell,
will be used as a (semi-flexible) lego-block to see if it
can construct the larger Niggli cell.

Crystal symmetries in supplied setting
--------------------------------------
Target crystal symmetry:
    Unit cell: (115.5, 149, 115.6, 90, 115, 90)
    Space group: P 1 21 1 (No. 4)
Building block crystal symmetry:
    Unit cell: (61.8, 97.7, 148.9, 90, 90, 90)
    Space group: P 21 21 21 (No. 19)

Crystal symmetries in Niggli setting
------------------------------------
Target crystal symmetry:
    Unit cell: (115.5, 115.6, 149, 90, 90, 115)
    Space group: P 1 1 21 (No. 4)
Building block (lego cell) crystal symmetry:
    Unit cell: (61.8, 97.7, 148.9, 90, 90, 90)
    Space group: P 21 21 21 (No. 19)

Volume ratio between target and lego cell:  2.01

Cartesian basis (column) vectors of lego cell:
  /  61.8   0.0   0.0 \
  |   0.0  97.7   0.0 |
  \   0.0   0.0 148.9 /

Cartesian basis (column) vectors of target cell:
  / 115.5 -48.9   0.0 \
  |   0.0 104.8   0.0 |
  \   0.0   0.0 149.0 /

A total of   20 matrices in the Hermite normal form have been generated.
The volume changes they cause lie between    3 and    2.

Trying all matrices

   1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
   . . . . . * . . . | . . . . . . . . . . |

 Listing all possible solutions

Solution    1
------------------------------------------------------------
Target unit cell :     115.5 115.6 149.0  90.0  90.0 115.0 (Sub lattice)
Lego cell :             61.8  97.7 148.9  90.0  90.0  90.0 (Super lattice)

               /   2    1    0  \
matrix :  M =  |   0    1    0  |
               \   0    0    1  /

Additional Niggli transform:      x,-y,-z
Additional similarity transform:  x,y,z
Resulting unit cell :  118.3 118.3 148.9  90.0  90.0 120.0
Deviations :            -2.5  -2.4   0.1   0.0   0.0  -5.0
Deviations for unit cell lengths are listed in %.
Angular deviations are listed in degrees.
------------------------------------------------------------
```

This result indicates that the two cells are related. A graphical depiction of the unit cell transformations is shown below.

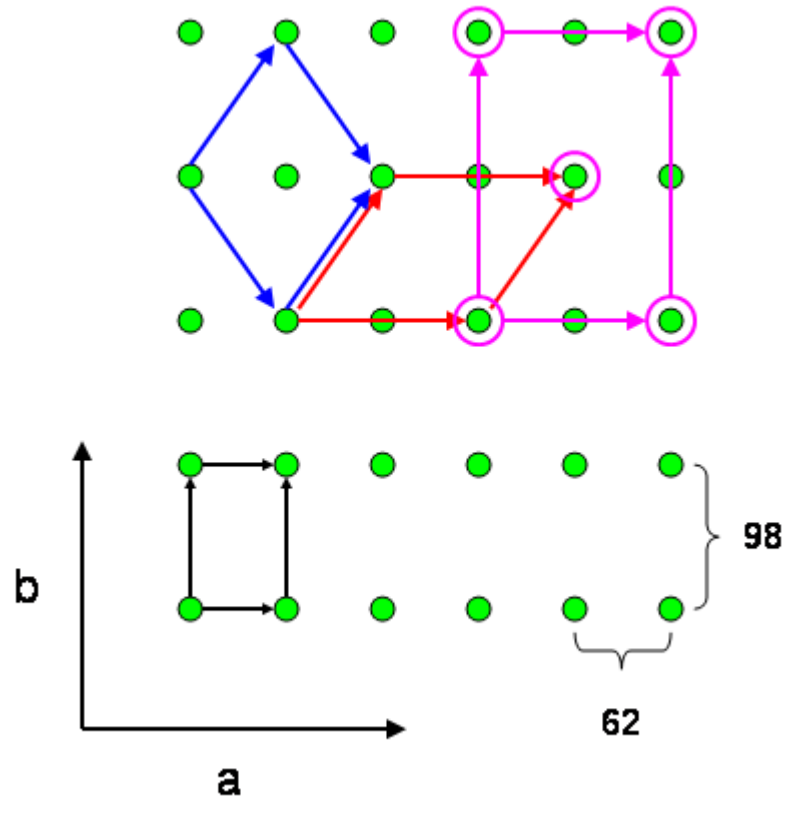


*Figure 1. The action of matrix **M** on the original lattice of the (P212121 native) lego cell is shown. The original orthorhombic cell is shown in black. The monoclinic cell obtained after application of matrix **M** is shown in red. A cell reduction of the red unit cell results in the blue SeMet1 cell. The C-centered cell of the SeMet2 derivative is shown in pink. The view is along the original c axis.*

Comparing Native and SeMet1 to SeMet2

Although it is rather obvious that the Native data is related to SeMet2 (there is a factor 2 in the *a* and *b* axis), it might not be entirely clear where the centring comes from. The presence of the centring operator becomes however clear when comparing the Niggli cells of the two Se-Met derivatives

```
Niggli cell SeMet1:
   Unit cell: (115.5, 115.6, 149, 90, 90, 115)
   Space group: P 1 1 21 (No. 4)

Niggli cell SeMet2:
   Unit cell: (115.6, 115.6, 148.9, 90, 90, 115.4)
   Space group: Hall:  C 2c 2 (x-y,x+y,z)
```

Since the Niggli cells of the derivatives are similar, the introduction of the centring is a direct result of the pseudo symmetric properties of the Niggli cells of the Selenium derivatives. A graphical depiction of the situation is shown in Figure 1.

The relations found between these various crystal forms can be practically relevant when attempting to perform molecular replacement against multiple crystal forms (Di Constanzo *et al.*, 2003) or when performing multiple crystal density modification.

4. Availability

The program `iotbx.explore_metric_symmetry` is *open source* code included in the CCTBX.

5. Acknowledgements

We gratefully acknowledge the financial support of NIH/NIGMS through grants 5P01GM063210 and 5P50GM062412. Our work was supported in part by the US department of Energy under Contract No. DE-AC02-05CH11231.

6. References

- Andrews, L.C., Bernstein, H.J. & Pelletier, G.A. (1980). *Acta Cryst.* **A36**, 248-252.
- Billiet, Y. & Rolley Le Coz, M. (1980). *Acta Cryst.* **A36**, 242-248.
- Dauter, Z., Li, M. & Wlodawer, A. (2001). *Acta Cryst.* **D57**, 239-249.
- Di Constanzo, L., Forneris, F., Geremia, S. & Randaccio, L. (2003). *Acta Cryst.* **D59**, 1435-1439.
- Grosse-Kunstleve, R.W. (1999). *Acta Cryst.* **A55**, 383-395.
- Grosse-Kunstleve, R.W., Sauter, N.K. & Adams, P.D. (2004a). *Acta Cryst.* **A60**, 1-6.
- Grosse-Kunstleve, R.W., Sauter, N.K. & Adams, P.D. (2004b). *Newsletter of the IUCr Commission on Crystallographic Computing* **3**, 22-31.
- Lehtio, L., Fabrichniy, I.,Hansen, T.,Schonheit, P.,Goldman, A. (2005). *Acta Cryst.* **D61**, 350-354.
- Le Page, Y. (1982). *J. Appl. Cryst.* **15**, 255-259.
- Lebedev, A.A., Vagin, A.A. & Murshudov, G.N. (2006). *Acta Cryst.* **D62**, 83-95.
- Poulsen, J.-C.N., Harris, P., Jensen, K.F.& Larsen, S. (2001). *Acta Cryst.* **D57**, 1251-1259.
- Rutherford, J.S. (2006). *Acta Cryst.* **A62**, 93-97.
- Sauter, N.K., Grosse-Kunstleve, R.W. & Adams, P.D. (2006). *J. Appl. Cryst.* **39**, 158-168.
- Skrzypczak-Jankun, E. ,Bianchet, M.A.,Mario Amzel, L. & Funk, M.O. Jnr, (1996). *Acta Cryst*. **D52**, 959-965.
- Zwart, P.H., Grosse-Kunstleve, R.W. & Adams, P.D. (2005), CCP4 Newsletter, **42**, Winter 2005.