Coot News

Bernhard Lohkamp, Paul Emsley, Kevin Cowtan

11 July 2005

Coot[1] is a molecular graphics application for electron density-based building, with a particular emphasis on protein model-building. We describe here recent additions to the software.

1 SHELX support

Over recent months[2] Shelx support has been added to Coot. SHELX can output its data in "old style" STAR CIF format. This presented some difficulties because the mmCIF parser built into mmdb[3] (and used by Coot) doesn't parse this type of cif data.

Therefore, it is necessary to convert SHELX cif data [LIST 6] `.fcf` file to an mmCIF-compatible format. Coot detects from the file-name that it needs conversion. This is currently done by using the scripting language to write out an awk file which is excuted using input from the the SHELX `.fcf` file and output to an mmCIF data file. The output file-name is the same as the input file-name with the addition of `".cif"`. This new file can be read directly into Coot and can be used on subsequent occassions obviating the need for the conversion script.

The SHELX coordinates `.ins` file presented more of a problem. Each record is now compared to an extensive (but not officially complete) list of SHELX keywords and only when there is no match is this record considered an atom description. Most of the non-atom keywords are copied without interpretation but some are interpreted, such as the LATT, CELL, SYMM, FVAR cards. In future therefore, it will be possible to modify a SHELX model with operations such as water addition and side-chain re-modelling, modelling of alternate conformations, and creation of an `.ins` file without human intervention from which a new SHELX refinement can be started automatically.

As has been mentioned, Coot has been designed particularly with proteins in mind. As such, Coot currently does not draw bonds between the main molecule and symmetry related copies. However, this may be an inpropriate restriction for some (particularly inorganic) SHELX models with inversion centres.

2 Flexible Ligands

Ligands in protein binding sites are often not in their minimum energy configuration. Thus, when searching for ligands in electron density Coot is more likely to find the correct position and orientation if a selection of conformations is searched. To modify the conformation, Coot uses the ligand geometry torsion description in the mmCIF-based restraints file that one would use for REFMAC refinement.

In the restraints file each rotatable bond is independently described by an initial torsion angle, a standard deviation and a period, and from these a probability distribution can be

[1]Emsley & Cowtan (2004) *Acta Cryst. D* **60**: 2126-2132 Part 12 Sp. Iss. 1 DEC 2004.

[2]and with the advice of George Sheldrick, Gábor Bunkóczi and Judit Debreczeni.

[3]Krissinel E.B. *et al.* (2004) Acta Cryst. D **60**: 2250-2255 Part 12 Sp. Iss. 1 DEC 2004.

constructed. However this is a naïve description - the torsion angles are generally not independent.

For each rotatable bond Coot makes a random selection of a torsion angle from the probability distribution. A model constructed in this manner can therefore lead to high-energy (and therefore unlikely) conformations. Therefore, an extra step has been added to the generation of ligand conformers which does "energy" minimization using non-bonded distance restraints in the target function[4]. However, although this makes the model more realistic it considerably reduces the speed at which new conformers are generated.

3 Difference Map Variance Analysis

The difference map variance analysis serves several purposes

- checks for sphericity of the electron density at water positions
- finds waters with the wrong temperature factor or occupancy
- (if the difference map is an anomalous difference map) finds "anomalously diffracting" water molecules. Such waters are of course candidates to be ions since water molecules are not generally known for their anomalously diffracting properties.

The difference map is sampled around each water postion at 3 different radii at 14 points on the surface of cannonical spheres of radius 0.4, 0.8 and 1.2Å.

The variance for each radius are totalled for each atom and these totals analysed for outstandingly high variance sum. A clickable list of buttons is created for each deviant water.

4 Coot for Windows

Coot has been compiled for Windows platforms. Initially the program was ported to Cygwin and later to a native port using MinGW [5]. The native build for Windows (WinCoot) has almost the full functionality as the Unix-based original. WinCoot uses the Python scripting option of Coot. Most of the Coot Guile scripts have been translated to Python to be used in WinCoot, however these can also be used within unix-based Python-enabled Coot.

Windows-based computers are used extensively and crystallographic programs are increasingly being made available for Windows platforms. Furthermore the majority of mobile computers seem to run a version of the Windows OS. Therefore there is some pressure from the community to have Coot run on Windows computers. Windows-based crystallographic model-building programs give the opportunity to the crystallographer to build structures whilst travelling, *e.g.* from collecting data at the synchrotron. Coot for Windows (WinCoot) complements the CCP4 Suite[6], which has been available for Windows for some time. This enables crystallographers to use programs from data processing to model-building on their Windows PCs.

The initial version of WinCoot was compiled with the Cygwin Unix emulation under Windows. However the program turned out to perform slowly, due to the emulation and use of `cygwin.dll`[7]. Therefore Coot was compiled natively on Windows using MinGW, this process is be described in more detail below. The Cygwin Coot port is not supported and updates are no longer available.

[4]torsion angles are not included in the target function.

[5]`http://www.mingw.org`

[6]'"The CCP4 Suite: Programs for Protein Crystallography" Acta Cryst. D (1994) **50**, 760-763.

[7]rather than directly using the Windows system `msvcrt.dll`.

4.1 Compilation of Dependency Libraries

Coot requires a number of different libraries to be compiled and installed prior to compilation of Coot. All these libraries and WinCoot have been compiled on a Windows 2000 computer running MinGW 3.1.0 with Minimal SYStem (msys) 1.0.10. The GNU compiler suite version 3.2.3 was deployed for compiling and linking. The crystallographic libraries, mmdb (1.0.6), SSMlib (0.0-pre1), mccp4 (0.7.1) and clipper[8] as well as the general scientific libraries, FFTW (2.1.3) and the GNU Scientific Library (1.5) were compiled with no or very little adjustments for the WIN32 system.

Coot supports two scripting languages, guile and Python, of which the guile support is more extensive. Guile 1.6 or higher is required for Coot, however we have been unable to compile this on our Windows system. Therefore Python was chosen as the scripting language for WinCoot. Currently Python 2.3.4 is used and binaries including the developer files were obtained by download from the main Python web site, `www.python.org`. Various graphic libraries are used by Coot, namely Glib, GTk+, GTkglarea, GTk-Canvas and its dependency, imlib. Glib, GTk+ and related libraries for Windows are available for download from the developers of the GIMP package, including developer files. However the (available) libraries used are based on the Glib/GTk+-2 package and not 1.2 as required for Coot on Unix based systems. Further details and implications are described further in the next section. Currently Glib version 2.4.7 and GTk+/GDK 2.4.14 are used. (and lots of others, like pango, freetype, gmodule, gdk-pixbuf, libart, libpng). Using these libraries GTkglarea 1.2.3 compiled without any problems on MinGW. GTk-Canvas (and imlib) requires an X-Windows system, which is not available on Windows without using an emulation like cygwin. Therefore GNOME-Canvas, which readily compiles under MinGW, was employed instead of GTk-Canvas. This and the use of GTk+/GDK-2.4, which includes GDK-pixbuf, made the requirement of imlib redundant.

4.2 Compilation of Coot on Windows

Various adjustments of the Coot source code had to be made to compile it successfully on MinGW. The majority of changes are related to the different structure of the OS, Unix and Windows, as well as using updated graphics libraries and the exclusion of an X-Windows system. Some additional changes were made to implement Python scripting. Most of these are of general Coot interest and are described in futher detail.

The majority of changes were due to the different file and directory structure as well as the recognition of storage devices in the two OSs Windows and Unix. Additionally some changes were made to the use of environment variables and related issues. First `$HOME` was changed to `$COOT_HOME`, since Windows either does not use the environment variable `$HOME`, or it interferes with Cygwin. Second WinCoot was adjusted to read the correct paths for possible CCP4 installations and projects. This is necessary since the handling of users is different in Unix and Windows[9].

The use of GTk+-2 required some syntax changes to the source code. Since Coot uses some GTk+ structures which are deprecated in GTk+-2 and known to be buggy (*e.g.* GtkCList), adjustments were made to maintain functionality of the widgets. Experimentally some of the deprecated structures were translated into new functionality, but implementation is not yet fully complete. Furthermore code for the scrolling functions for changing the map contour level had to be re-written due to the different signal of the mouse scroll wheel in Windows. The change from GTk-Canvas to GNOME-Canvas was straight-forward since GNOME-Canvas is based on GTk-Canvas. All calls for a `gtk_canvas_function()` are changed to the corresponding `gnome_canvas_function()`. WinCoot was compiled with the gcc compiler option `-mms-bitfields`, which is necessary for GTk+ to function.

[8] `http://www.ysbl.york.ac.uk/~cowtan/clipper/clipper.html`

[9] designed in the days before XML, perhaps?

4.3 Windows installer and availability

WinCoot is not straight-forward to compile and for ease of use, is therefore available as compiled binaries. WinCoot can be downloaded as a self extracting exe file (see Section 5).

The WinCoot installer was built with HM NIS Edit, then compiled and compressed with NSIS (Nullsoft Scriptable Install System). It contains all necessary libraries (DLLs) including all required Python files. Therefore no additional installations or programs are required. The installer will create shortcuts to WinCoot for every user, ready to run the program. WinCoot is launched with a Windows batch file (`runwincoot.bat`) which means that no setting or changes of environment variables are necessary, thus avoiding possible conflicts with other programs and editing of Windows system files.

The current version of WinCoot corresponds to Coot version 0.0.25. Availability and integration of WinCoot into the CCP4 package for Windows is planned.

4.4 Running WinCoot

WinCoot was successfully tested on Windows XP, Windows 2000, Windows NT and Windows 98. The `runwincoot.bat` batch file sets all necessary environment variables and then executes the `coot.exe` file. It is possible to run Coot from the Windows Command Shell, Cygwin or MinGW shells, but some manual adjustments to the environment variables or location of the batch file might have to be made in that case.

On multi-user Windows PCs the batch file can be customised, so that each user or project has its own. This will allow individually associated coot setup files (`.coot.py` in `$COOT_HOME`) and backup directories (`$COOT_BACKUP_DIR`). If CCP4 for Windows is installed on the computer REFMAC5 can be run *via* WinCoot (version 0.0.31 or better) and CCP4 project directories are recognised.

Overall WinCoot has nearly the full functionality of the corresponding Coot version. However currently some minor restrictions apply due to deprecated GTk+ functions and/or missing functions in MinGW, *e.g.* filtering of files in file selection widgets. These problems have been recognised and addressed in ongoing development of WinCoot.

4.5 Python Scripting

On initiation Coot reads the `coot.py` module created by SWIG[10]. Then a personal coot setup file, `.coot.py`, is loaded if it exists in the home directory (or for WinCoot in `$COOT_HOME`). This file can (as can the corresponding guile file) contain global parameters and settings, including parameters to run REFMAC5. Then a variety of python modules are loaded from `$COOT_PYTHON_DIR`. If a python state file, `0-coot.state.py` exists in the directory from which `coot` was started, it can be loaded via a GTk dialog (*Calculate → Run Script...*).

Currently not all the scripting functions that have been written in guile scheme are available as Python functions. The available modules are `hello.py` (welcome notes), `gap.py` (for loop fitting), `mutate.py` (for mutation of residue range), `fitting.py` (animated protein fit of whole protein), `refmac.py` (execute REFMAC5 from within Coot) and some Coot utility functions in `coot-utils.py`. Further modules are under construction, although some use GTk functions and therefore will require an additional library in form of PyGTk.

[10]`http://www.swig.org`

5 Getting Coot

Coot is Free Software and has a variety of distribution points:

Coot Main Page (with mailing list info, FAQ, links, *etc*):
`http://www.chem.gla.ac.uk/~emsley/coot`

WinCoot:
`http://www.chem.gla.ac.uk/~bernhard/coot/wincoot.html`

Mac Coot by William Scott:
`http://www.chemistry.ucsc.edu/~wgscott/xtal/coot/`