Recent developments in the PHENIX software for automated crystallographic structure determination

Paul D. Adams,[a*] Kreshna Gopal,[b] Ralf W. Grosse-Kunstleve,[a] Li-Wei Hung,[c] Thomas R. Ioerger,[b] Airlie J. McCoy,[d] Nigel W. Moriarty,[a] Reetal K. Pai,[b] Randy J. Read,[d] Tod D. Romo,[e] James C. Sacchettini,[e] Nicholas K. Sauter,[a] Laurent C. Storoni[d] and Thomas C. Terwilliger[f]

[a]Lawrence Berkeley National Laboratory, One Cyclotron Road, BLDG 4R0230, Berkeley, CA 94720, USA, [b]Department of Computer Science, Texas A&M University, 301 H.R. Bright Building, 3112 TAMU, College Station, TX 77843, USA, [c]Biophysics Group, Mail Stop D454, Los Alamos National Laboratory, Los Alamos, NM 87545, USA, [d]Department of Haematology, University of Cambridge, Cambridge Institute for Medical Research, Wellcome Trust/MRC Building, Hills Road, Cambridge, CB2 2XY, UK, [e]Department of Biochemistry and Biophysics, Texas A&M University, 103 Biochemistry/Biophysics Building, 2128 TAMU, College Station, TX 77843, USA, and [f]Los Alamos National Laboratory, Mailstop M888, Los Alamos, NM 87545, USA. E-mail: PDAdams@lbl.gov

A new software system called PHENIX (**P**ython-based **H**ierarchical **EN**vironment for **I**ntegrated **X**tallography) is being developed for the automation of crystallographic structure solution. This will provide the necessary algorithms to proceed from reduced intensity data to a refined molecular model, and facilitate structure solution for both the novice and expert crystallographer. Here we review the features of PHENIX and briefly describe the recent advances in infrastructure and algorithms.

Keywords: PHENIX; Python; object-oriented programming; strategies.

1. Introduction

For structural genomics (Burley *et al*., 1999; Sali, 1998) to be possible, structures will need to be solved significantly faster than is currently routinely achievable. This high-throughput structure determination will require automation to reduce the obstacles related to human intervention. Manual interpretation of complex numerical data has a significant subjective component (Mowbray *et al*., 1999) that can lead to delays in reaching the final structure. Thus, the automation of structure solution is essential as it has the potential to produce minimally biased models more efficiently. We are developing the PHENIX software to address these needs (Adams *et al*., 2002).

2. PHENIX design and implementation

2.1. Hybrid programming

PHENIX is based on a tight integration between reusable software components written both in a compiled language and a flexible scripting language. Prior experience implementing the Crystallography & NMR System (Brunger *et al.*, 1998) has shown that this promotes highly efficient software development. High-level algorithms such as complex refinement protocols or phasing procedures are most rapidly developed in a scripting language. By contrast, numerically intensive core algorithms such as the computation of structure factors or discrete Fourier transforms must be implemented in a compiled language for performance reasons.

PHENIX uses Python (http://python.org/) as the scripting language and C++ as the compiled language, see Grosse-Kunstleve *et al.*, 2002 and Grosse-Kunstleve & Adams, 2003 for further details about these choices. This relies on the Boost.Python Library (Abrahams & Grosse-Kunstleve, 2003) for conveniently integrating C++ and Python. It is used to directly connect C++ classes and functions to Python without obscuring the C++ interface. The PHENIX software is developed and tested on the commonly available computing platforms: Redhat Linux, HP Tru64, SGI Irix version 6.5 and Windows 2000. The Macintosh OS-X platform may also be supported in the future as the necessary tools become available.

2.2. Data objects and tools

In order to build a complex, integrated system like PHENIX, certain basic data objects must be available. We have implemented many of the important objects required for crystallographic computations:

- Structure factor objects, which hold reciprocal space data. Data containers have been implemented in C++ and are made available in Python using the Boost.Python library. This design permits the reuse of the "objects" by future developers within either a compiled C++ program, or an interpreted Python script.
- Map objects, which hold real space data such as electron density maps. Data containers have been implemented in C++, and are available from the Python level.
- Molecular objects, which hold the coordinates and topology of a structure. Data containers have been implemented in Python for speed of testing and development. Coordinate files from the Protein Data Bank can be read into PHENIX and the appropriate connectivity between atoms determined using prior topology information.
- Space group symmetry objects, which comprehensively describe crystallographic symmetry and provide methods for manipulation of these symmetries.

Implementation of these objects makes extensive use of the Computational Crystallography Toolbox (*cctbx*; Grosse-Kunstleve & Adams, 2003; Grosse-Kunstleve *et al.*, 2002). This is a library of *reusable* core crystallographic software components for macromolecular structure determination that have been designed for integration into large, modular, layered software systems. The PHENIX data objects are constructed using the *cctbx array family*, which provides a generalized representation of vector data, of arbitrary numbers of dimensions. The *cctbx* source code is freely available under an Open Source license for both non-profit and commercial use at http://cctbx.sourceforge.net/.

2.3. Algorithms

A number of algorithms for structure solution have been implemented. A new substructure searching procedure, called HYSS (HYbrid Substructure Search), makes use of Patterson and direct methods to locate anomalously scattering or heavy atoms for experimental phasing. This algorithm incorporates criteria that automatically determine when a correct solution is likely to have been found. Once the substructure has been determined there are

interfaces to the SOLVE program that enable rapid configuration of jobs for experimental phasing using MAD/SAD and MIR/SIR methods (Terwilliger & Berendzen, 1999). In the future, efficient new algorithms for phasing that take account of correlations in the errors between derivatives and wavelengths will be used (Terwilliger & Berendzen, 1996, Terwilliger & Berendzen, 1997, Read 1991). Initial phases can also be obtained using molecular replacement incorporating maximum likelihood targets (Read, 2001). The use of these new targets increases the success rate of this method using search models of lower structural similarity. The phases obtained from experimental phasing or molecular replacement are optimised by the application of maximum likelihood density modification algorithms, currently implemented in the RESOLVE program, to produce minimally biased electron density maps (Terwilliger, 2002; Terwilliger, 2001). Electron density maps are automatically interpreted using template matching (Terwilliger, 2003a; Terwilliger 2003b; Terwilliger, 2001) as implemented in RESOLVE, and pattern recognition methods as implemented in TEXTAL (Ioerger & Sacchettini, 2002, Holton *et al.*, 2000). In the near future algorithms for structure refinement will be implemented. The automated map interpretation algorithms will be iterated with maximum likelihood refinement targets (Pannu & Read, 1996; Pannu *et al.*, 1998) and simulated annealing optimisation algorithms (Brunger, Kuriyan & Karplus, 1987; Adams *et al.*, 1997, Adams *et al.*, 1999). We expect that the combination of tools will permit automated structure completion even when diffraction data are only available up to a modest resolution limit (approximately 3Å).

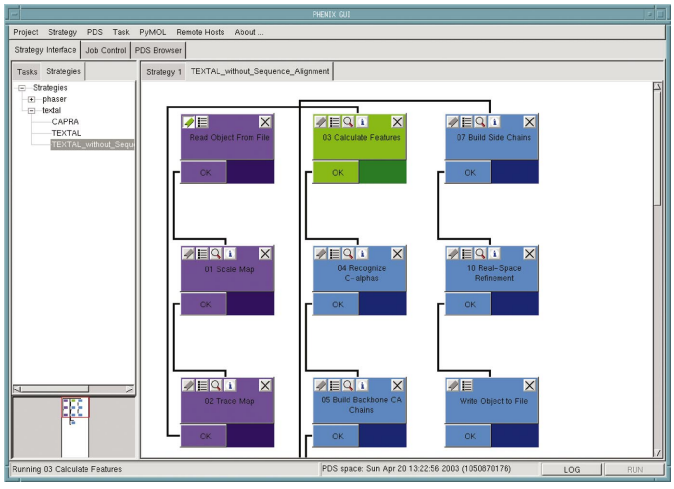

Figure 1

The PHENIX graphical environment. A strategy for automated model building using TEXTAL has been loaded and executed. The currently executing task is shown in green, while tasks already executed are shown in purple. The job control menu permits previous jobs to be revisited and modified for rerunning. The PDS browser allows the user to see what data objects were generated in the course of a strategy run. In the future this will be expanded to allow visualisation of the data objects using the appropriate tools.

2.4. Strategies and the graphical user interface

Many crystallographic software packages either provide the user with a collection of tools for analysis of the data, or with a monolithic "black box" application that performs all the relevant tasks in an automated fashion. Neither situation is optimal for the user. The first case is appropriate for the expert user who can use the tools in the correct sequence and also make decisions based on the results at each stage. The novice user can often make good use of a "black box" system. However, when problems are encountered (i.e. the automation fails), it is often very difficult to identify the problem or implement a solution.

The *strategy* concept in PHENIX is used to avoid these problems. This provides a way to construct complex networks of tasks to perform a higher-level function. For example, the steps required to go from initial data to a first electron density map in a SAD experiment can be broken down into well-defined tasks, which can be reused in other procedures. Instead of requiring the user to run these tasks in the correct order they are connected together by the software developer, and can thus be run in an automated way. However, because the connection between tasks is dynamic they can be reconfigured or modified, and new tasks introduced as necessary if problems occur. This provides the flexibility of user input and control, while still permitting complete automation when decision-making algorithms are incorporated into the environment.

It should be emphasized that the tasks and their connection into strategies rely on the use of plain text task files written using the Python scripting language. This implementation was chosen so as to not restrict the use of PHENIX to a graphical user interface. This enables the algorithms to be used easily in a non-graphical environment. The PHENIX GUI permits strategies to be visualized and manipulated (Figure 1). These manipulations include loading a strategy distributed with PHENIX, customizing and saving it for future recall. Customisation of the task input parameters is achieved via the graphical interface. Insertion of new tasks is performed by choosing from the tree menu on the left of the main window. A job control menu permits the status of user jobs to be monitored and revisited. The GUI also provides a means for the user to view results of calculations. PyMOL (DeLano, 2002), a molecular graphics system written in C and Python, allows easy viewing of structures and maps via close integration with PHENIX. Graphical data is transferred from PHENIX to PyMOL through a socket connection. Native PHENIX display windows can also represent simple results such as x-y graphs.

2.5. The Project Data Storage (PDS)

One of the major problems facing the crystallographer is the organization, tracking and archiving of data. These problems are significantly compounded by the move to high-throughput crystallography, which leaves no time for user control of data management. Therefore, in PHENIX we have introduced the concept of the Project Data Storage (PDS). This is a data management system that oversees the information generated for each structure determination (or "Project"), and contains a complete history of each structure solution, along with all of the generated structural information. A PDS browser has been implemented that allows users to see which data objects have been generated during the course of calculations on the data. In the future this browser will be extended to allow direct visualization of the data objects.

In order to make full use of the computing resources typically available to researchers we have developed and implemented a distributed computing model for PHENIX. This permits the remote execution of computationally intensive tasks (for example, a job can be set up on a user's desktop PC, but sent to a high-performance computing platform for execution). An interface to the popular ssh program is used to start remote jobs so as to maintain user password security. Once a job has been started it can also be monitored and controlled remotely by multiple instances of the graphical user interface running on different machines. This distributed computing model relies on a PHENIX PDS server that coordinates the information flow for each user project.

3. Conclusions

The development of PHENIX is a collaborative project whose primary goal is the creation of a comprehensive, integrated system for automated crystallographic structure determination. The PHENIX infrastructure is also designed to be open and easily shared with other researchers. Source code will be distributed to academic groups, and the use of the Python scripting language will facilitate interfacing with the system and its use in different contexts. Other developers will be able to implement their algorithms to the PHENIX environment thus providing easy access to a large number of crystallographic and computational tools. The high-level graphical programming environment in PHENIX is designed to let researchers easily link crystallographic tasks together, thus creating complex algorithms without having to resort to low-level programming. The graphical interface and the underlying Python scripting language also provide a framework suitable for implementing decision-making algorithms that will be critical for robust and reliable automation. User testing of the alpha release of version 1.0 of PHENIX is underway. We anticipate a general release of the software in the fall of 2003.

This work was funded by NIH/NIGMS under grant number 1P01GM063210, with initial funding to PDA from the Department of Energy under contract No. DE-AC03-76SF00098.

References

Abrahams D. and Grosse-Kunstleve R. W. (2003). C/C++ Users Journal **21**, 29-36.
Adams P. D., Grosse-Kunstleve R. W., Hung L.-W., Ioerger T. R., McCoy A. J., Moriarty N. W., Read R. J., Sacchettini J. C., Sauter N. K. and Terwilliger T. C. (2002). *Acta Cryst*. **D58**, 1948-1954.
Adams P. D., Pannu N. S., Read R. J. and Brunger A. T. (1997). *Proc. Nat. Acad. Sci. USA*. **94**, 5018-5023.
Adams P. D., Pannu N. S., Read R. J. and Brunger A. T. (1999). *Acta Cryst*. **D55**, 181-190.
Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., Bourne, P. E. (2000). *Nucleic Acids Research*, **28**, 235-242.
Brunger A. T., Kuriyan J. and Karplus M. (1987). *Science* **235**, 458–460.
Brunger, A. T., Adams, P. D., Clore, G. M., Gros, P., Grosse-Kunstleve, R. W., Jiang, J.-S., Kuszewski, J., Nilges, N., Pannu, N. S., Read, R. J., Rice, L. M., Simonson, T. & Warren, G. L. (1998). *Acta Cryst*. **D54**, 905-921.
Burley S. K., Almo S. C., Bonanno J. B., Capel M., Chance M. R., Gaasterland T., Lin D., Sali A., Studier F. W. and Swaminathan S. (1999). *Nat. Genet*. **23**, 151-157.
DeLano, W. L. (2002). The PyMOL Molecular Graphics System (http://www.pymol.org)
Grosse-Kunstleve R. W. and Adams P. D. (2003). *IUCr Computing Commission Newsletter* **1**.
Grosse-Kunstleve, R. W., Sauter, N. K., Moriarty, N. W. & Adams, P. D. (2002). *J. Appl. Cryst*. **35**, 126-136.
Holton T., Ioerger T. R., Christopher J. A. and Sacchettini J. C. (2000). *Acta Cryst*. **D56**, 722-734.
Ioerger T. R. and Sacchettini J. C. (2002). *Acta Cryst.*, **D58**, 2043-2054.
Mowbray S. L., Helgstrand C., Sigrell J. A., Cameron A. D. and Jones T. A. (1999). *Acta Cryst*. **D55**, 1309-1319.
Pannu N. S. and Read R. J. (1996). *Acta Cryst*. **A52**, 659-668.
Pannu N. S., Murshudov G.M., Dodson E.J. and Read R.J. (1998). *Acta Cryst*. **D54**, 1285-1294.
Read, R. J. **In**: Isomorphous Replacement and Anomalous Scattering: Proceedings of the CCP4 Study Weekend, W. Wolf, P. R. Evans and A. G. W. Leslie (eds.) Science and Engineering Research Council, Daresbury Laboratory. pp. 69-79 (1991).
Read, R. J. (2001). *Acta Cryst*. **D57**, 1373-1382.
Sali A. (1998). *Nature Struct. Biol.* **5**, 1029-1032.
Terwilliger, T. C. (2001). *Acta Cryst*. **D57**, 1755-1762.
Terwilliger T. C. (2002). *Acta Cryst*. **D58**, 2082-2086.
Terwilliger T. C. (2003a). *Acta Cryst*. **D59**, 38-44.
Terwilliger T. C. (2003b). *Acta Cryst*. **D59**, 45-49.
Terwilliger, T. C. and Berendzen, J. (1996). *Acta Cryst.* **D52**, 749-757.
Terwilliger, T. C. and Berendzen, J. (1997). *Acta Cryst.*, **D53**, 571-579.
Terwilliger T.C. and Berendzen J. (1999). *Acta Cryst*. **D55**, 849-861.

 Received 30 April 2003 • Accepted 15 October 2003