



CCP4 NEWSLETTER ON PROTEIN CRYSTALLOGRAPHY

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

Number 45

Winter 2007

Contents

News

1. **CCP4 Workshops - where we went in 2006**

Martyn Winn, Peter Briggs, Charles Ballard

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

Software

2. **CCP4 in BIOXHIT: Managing and Visualising Project Data**

Peter Briggs, Wanjuan Yang

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

3. **RAPPER: Real Space Automated Conformer Generation**

Nicholas Furnham, Tom L. Blundell

Department of Biochemistry, Cambridge University, 80 Tennis Court Road, Cambridge, CB2 1GA, UK

4. **Diffraction Image: a new CCP4 library**

Francois Remacle, Graeme Winter

CCLRC Daresbury Laboratory Warrington WA4 4AD, United Kingdom

Methodology

5. **A self-rotation puzzle**

Zhenbo Cao, Neil W. Isaacs

WestCHEM, Department of Chemistry, University of Glasgow, Glasgow G12 8QQ, UK

6. **On determination of T matrix in TLS modelling**

P. Afonine, A. Urzhumtsev

Lawrence Berkeley National Laboratory, One Cyclotron road, BDLG64 R1021, Berkeley, CA 94720 USA

University Louis Pasteur, Strasbourg - IGBMC, 1 rue L. Fries, 67404 Illkirch - IBMC, 15 rue Descartes, 67084 Strasbourg

Editor: Francois Remacle

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

NOTE: The CCP4 Newsletter is not a formal publication and permission to refer to or quote from the articles reproduced here must be referred to the authors.

Contributions are invited for the next issue of the newsletter, and should be sent to Francois Remacle by e-mail at <mailto:fr45@ccp4.ac.uk> by next season. HTML is preferred but other editable formats are also acceptable.

CCP4 Workshops - where we went in 2006

Martyn Winn, Peter Briggs and Charles Ballard

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

CCP4 continues to place conferences and workshops high on our list of ways in which to make contact with the international user community. A listing of upcoming events, as well as resources from past workshops, can be found on our [Courses](#) page. Here we report on a few of the events from 2006. As well as giving us a chance to share our photographs, we hope it encourages you to visit us at a conference or register for a CCP4 workshop.

ACA Honolulu, Hawaii

The 2006 Annual Meeting of the American Crystallographic Association took place in Honolulu, Hawaii at the end of July, and once more CCP4 was present with a stand in the commercial exhibition manned by Ronan Keegan and Peter Briggs.

The stand itself was initially decorated in local Hawaiian style due to a delay in the arrival of our exhibition materials. Once the materials appeared however we were able to revert to a more familiar format, and our giveaways (this year including CCP4 T-shirts) were quickly snapped up.

and after:

Before:



As is always the case at the ACA, we met a mixture of new and familiar faces whom we were happy to help with various questions about the CCP4 software. Of particular interest in 2006 were the two CCP4-related graphics programs CCP4mg and Coot, however there were also many questions about the 6.0 release of the suite and new developments such as MrBUMP.

We'd like to thank all the people who came along and talked to us during the conference and provided us with feedback on the suite - we hope that it was equally interesting and useful for you. We'd also like to thank the ACA organisers for their help, and last but not least Maeri Howard for her behind-the-scenes work in getting everything arranged, printed and shipped.

ECM Leuven, Belgium

Hot on the heels of the ACA, the 23rd European Crystallographic Meeting took place in Leuven, Belgium in the first half of August. CCP4 again had a stand, manned at various times by Francois Remacle, Ronan Keegan, Graeme Winter, Wendy Yang and Martyn Winn. Despite being a little hidden, most found us in the end, and we were pleased to meet many new people, as well as old friends.



As at the ACA, demonstrations of Coot and CCP4mg were the most popular items this year, but we were also able to provide advice on a variety of queries, including model building, refinement and installation.



Beijing workshop

The first dedicated CCP4 workshop to be held in China took place at the Institute of Biophysics, Beijing in October 2006. The five tutors (Eleanor Dodson, Gwyndaf Evans, Roberto Steiner, Martyn Winn and Francois Remacle) gave talks on various aspects of protein crystallography and CCP4, and supervised hands-on sessions in the afternoon. There were around 80 students participating, many of whom had travelled the long distance from Shanghai.



Many of the talks and tutorials are available at <http://www.ccp4.ac.uk/courses/china06/>. The tutorials were written especially for the workshop, and will be included in the next CCP4 release. As well as going through prepared exercises, many students had on-going crystallography projects, and the tutors were able to provide advice on real problems. Finally, as a side effect of our visit to the IBP, we updated much of the local software to the latest versions.

We'd like to thank our hosts Prof. Tao Jiang, Prof. Zihe Rao and Mark Bartlam, as well as all the participants, for making our stay enjoyable and productive. We hope to be back!

Osaka workshop

Since the previous CCP4 tutorial in Japan, back in 2000, there have been many advances in the suite. This year, 27 Nov - 1 Dec, CCP4 held a five day workshop at the IPR, Osaka University. The high powered team of Garib Murshudov (York), Phil Evans (LMB-MRC), Kevin Cowtan, Paul Emsley (both York), Serge Cohen (NKL), Raj Pannu (Leiden), Roberto Steiner (Imperial), Nakagawa Atsushi (Osaka), Kosada Takashi (PDBj) and Charles Ballard (Daresbury) covered the range of the structural solution process from data collection to deposition for 70 students, with a mixture of lectures, tutorials and problem solving.



CCP4 would like to extend it's thanks to the local organisation and support team, in particular Fujita-san and Prof. Nakagawa.

UK university visits

In 2006, we started a programme of 1-day university visits, in order to help practising crystallographers get up-to-speed with the latest software, in particular CCP4 6.0. (Those with a long memory may remember a similar series of visits in 2001.) We visited Manchester, Birkbeck, Edinburgh, Glasgow and Nottingham, and plan to make similar visits in 2007. Activities include a health check on local installations, talks on recent additions to CCP4 and 1:1 sessions on specific problems. We are happy to customise visits as appropriate.

CCP4 in BIOXHIT: Managing and Visualising Project Data

Peter Briggs and Wanjuan Yang

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

Email: p.j.briggs@dl.ac.uk or w.yang@dl.ac.uk

1 Introduction

BIOXHIT is a 4-year integrated project funded within the 6th Framework Programme of the European Commission. The project started in January 2004 and involves over twenty partner institutions, including all the European synchrotrons as well as leading software developers, with the aim of providing an integrated technology platform for Structural Genomics efforts.

One component of this platform is the development of automated structure determination software pipelines that cover the post-data collection stages of structure solution by protein X-ray crystallography. Similar pipelines are also being developed by groups outside of the BIOXHIT project, for example there are a number of CCP4-related efforts including MrBUMP, HAPPy and XIA2 (amongst others). Regardless of their provenance, all these pipelines need to accurately record and track the data that they produce, both for their own operation and for final deposition of the determined structures (considerations that apply equally to more "manual" procedures for determining structures, for example via CCP4i).

CCP4's contribution to BIOXHIT is principally concerned with developing a system for performing this data management in order to address the needs of software pipelines, by building on the existing data management functionality in CCP4i. This article is intended to give a brief overview of progress so far towards this goal.

2 CCP4 in BIOXHIT: managing and visualising project data

2.1 Background

Users of the CCP4 graphical user interface system CCP4i will already be familiar with the job database that forms a key component of the interface. Users divide their work into projects, consisting of a project directory (a UNIX directory or

a Windows folder, depending on the host platform) that is used to hold data files for the project, plus some additional associated data. Most significant among this data is the job database for the project; this is a list of all the jobs (runs of CCP4i tasks) performed within the project. Each job also has associated data that include the taskname, status, date, title, lists of the input and output files, and the parameter file used to run it.

The job database is very simple; however, coupled with a set of tools that allow querying and manipulation of the data it has proved to be one of the most powerful and useful aspects of CCP4i. The fact that every run of a task is automatically recorded in the job database has also contributed to its success; essentially the data is recorded "for free" by the system and so the cost (in terms of the user's time) of using the system to store data is minimal. Expanding this system seemed to be a good starting point to build a more sophisticated project history tracking system.

With this in mind, some of the key considerations for the new system are:

- Implement a single system for both "manual" and automated structure determination
A single system means that data tracking is standardised regardless of the source, so the same analysis and visualisation tools can be applied.
- Automatically gather as much information as possible from client programs
One of the strengths of the existing CCP4i system is that jobs are recorded automatically whenever the user runs a task, so this is a good characteristic to try and emulate.
- Provide an "open architecture" that can accommodate heterogeneous software components
One of the potential barriers to non-CCP4i software using the CCP4i system is that it requires that the external program have some Tcl component. Adopting an open architecture reduces this barrier by removing the dependence on a particular programming language.
- Allow flexibility in how the data is stored
If the way that the data is stored is hidden from the user and the programs that use it, then it is easier to update the database implementation in future (for example if a more sophisticated system such as MySQL is required). This should not be a consideration for the final user of the system however.

2.2 Building the new system

The current implementation of the database in CCP4i is a good start for the new system but suffers from a number of drawbacks, specifically:

- The database is implemented as a simple flat-file (the "database.def" format) which puts limitations on how easily data items can be related to each other, and also becomes less efficient for large amounts of data.
- The database handling functions are embedded within CCP4i and so are not easily accessible from programs that are external to CCP4i.

To address these issues and try to satisfy some of the requirements, a new system is shown schematically in figure 1, and consists of three principal components:

- The "project database handler" is a small server program that provides the functionality for interacting with the database (and thus separates it from CCP4i, which then becomes just another client application).
- The project database itself, which is only accessed via the handler.
- The "project data visualiser" which provides tools to view and interact with the data in the database.

These components are described in more detail in the following sections: the database handler (also called dbccp4i) is described in section 2.3; the database in section 2.4; and the visualiser (dbviewer) in section 2.5.

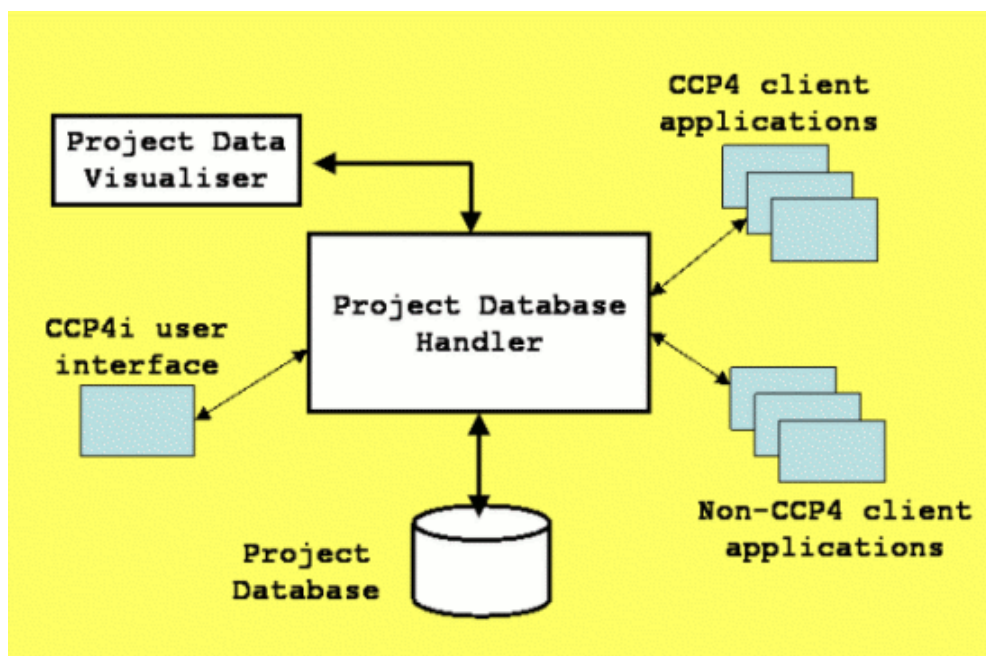


Figure 1:
Schematic
diagram showing
the relationship
between the
database
handler, the
database and the
visualiser.

2.3 The Project Database Handler: dbccp4i

The Project Database Handler "dbccp4i" is the key component of the system. It is a brokering application that mediates interactions between the project database and external applications: essentially, it is a small server program that should run quietly in the background, acting as a single point of access to the database for external programs like CCP4i (these external programs are referred to generally as client applications).

For most users, the initial version of CCP4i that uses the handler will not appear to be very different from the current version that doesn't; the job database will look the same and the available operations to view, manipulate and delete data will continue to work in the same way.

The real difference will be in making the user's projects and databases available to different programs simultaneously, which is something that is not properly

implemented at present. Initially this will mean that you will be able to have several CCP4is running simultaneously and they will be able to access and manipulate the same project database safely. At the same time the new visualiser application also being developed as part of this project will be able to present different views of the project history data independently of CCP4i (see section 2.5, below).

In the longer term, using the handler will allow programs that are not part of CCP4i or CCP4 to access the database. This would mean that you could perform your structure determination using a mixture of "manual" CCP4i tasks, automated pipeline programs and graphical model building tools, and have all the data history appear in the same project. This should make it easier to monitor a structure determination as it progresses, and help with preparing the data required for deposition at the end.

Also as the database content is expanded in future to include more crystallographic data, it will become easier for programs to share this data between them and thus reduce errors in transferring information through the structure solution process.

Software developers who are interested in more details of using the handler can refer to the "Technical Details" section (section 3) later in this article.

2.4 Database for Project Data & Tracking

The project database handler outlined in the previous section provides the core functionality that will allow multiple applications to access a single database simultaneously. Another facet of this work is to define exactly what form that database takes, and what data it will be able to store.

In the current development the system has focused on "project tracking" (also referred to as "project history") based on the existing CCP4i "projects and jobs" model of the data, which was described briefly in section 2.1. This system has already been seen to provide simple but robust tracking of the steps performed (and the associated data and parameter files) through manual structure solution projects via CCP4i. Reusing the existing "database.def" def-file format for storing the tracking database information in the first instance also ensures backwards compatibility with CCP4i.

Although some modifications to the "database.def" format are required to expand the tracking database to accommodate non-CCP4i programs (for example the name of the program - the "user agent", e.g. an automated pipeline - that writes changes to the database), overall it should provide a sound basis for future developments. For example, combined with the existing data harvesting mechanisms in CCP4 (see for example the CCP4 documentation on harvesting at www.ccp4.ac.uk/dist/html/harvesting.html) it should be possible to make improved tools to facilitate deposition of data to the Protein Databank (some work has been done with the EBI within BIOXHIT to investigate this).

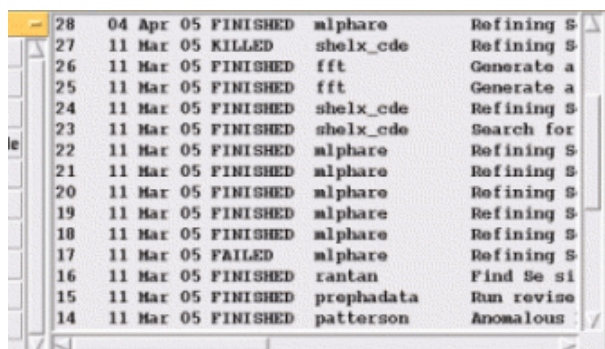
It is important to note that initially, crystallographic data is not being stored directly in the database (only indirectly, for example an MTZ file containing reflection data may be in the project directory and associated with a particular job in the database). However one of the long-term aims of the project is to construct a "rich database" that can combine crystallographic data with the tracking data. To this end some exploratory work (looking at the content of the "rich database" expressed as an SQL schema using an SQLite implementation as an alternative to the def-file format) has been performed, and will form the basis of future developments. See section 3.3 (Development of the "rich database") for more details of these developments.

2.5 Visualisation Tools: dbviewer

Visualisation tools are a general class of software that provide different views of data in order to aid understanding or help to provide new insight. Different visualisers can present the same data in different ways (for example, a PDB file can be viewed in a text editor or in a molecular graphics program) and so can be useful when trying to make sense of large amounts of diverse data.

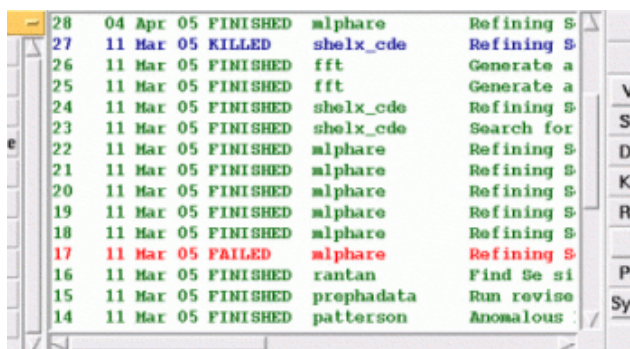
The starting point for this part of the project is the job display in CCP4i's main window, which can be thought of as a simple visualiser for the job database (figure 2). It shows the project history as a list of jobs in chronological order, with some information about each job, and provides tools to perform additional operations to "drill down" into the data (for example, obtaining a list of input and output files associated with a job).

CCP4i already offers some refinements to this view: for example it is possible to colour job records according to user-specified criteria (figure 3), and the "Search and Sort" tool provides another way again to look at the data (figure 4).



28	04 Apr 05	FINISHED	alphare	Refining S
27	11 Mar 05	KILLED	shelx_cde	Refining S
26	11 Mar 05	FINISHED	fft	Generate a
25	11 Mar 05	FINISHED	fft	Generate a
24	11 Mar 05	FINISHED	shelx_cde	Refining S
23	11 Mar 05	FINISHED	shelx_cde	Search for
22	11 Mar 05	FINISHED	alphare	Refining S
21	11 Mar 05	FINISHED	alphare	Refining S
20	11 Mar 05	FINISHED	alphare	Refining S
19	11 Mar 05	FINISHED	alphare	Refining S
18	11 Mar 05	FINISHED	alphare	Refining S
17	11 Mar 05	FAILED	alphare	Refining S
16	11 Mar 05	FINISHED	rantan	Find Se si
15	11 Mar 05	FINISHED	prephadata	Run revise
14	11 Mar 05	FINISHED	patterson	Anomalous

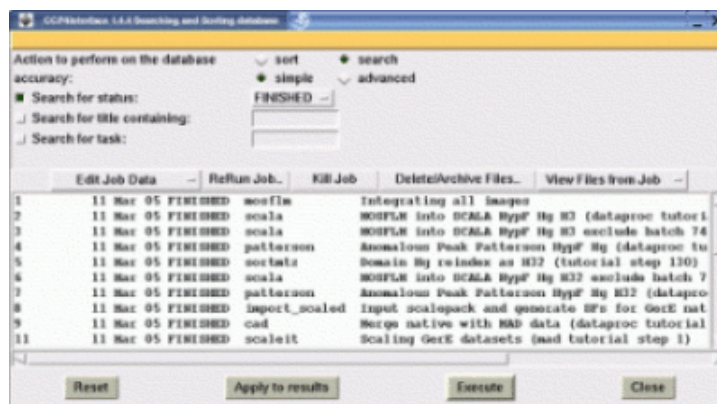
Figure 2 (above): the list of jobs as displayed in the main window of the current version of CCP4i.



28	04 Apr 05	FINISHED	alphare	Refining S
27	11 Mar 05	KILLED	shelx_cde	Refining S
26	11 Mar 05	FINISHED	fft	Generate a
25	11 Mar 05	FINISHED	fft	Generate a
24	11 Mar 05	FINISHED	shelx_cde	Refining S
23	11 Mar 05	FINISHED	shelx_cde	Search for
22	11 Mar 05	FINISHED	alphare	Refining S
21	11 Mar 05	FINISHED	alphare	Refining S
20	11 Mar 05	FINISHED	alphare	Refining S
19	11 Mar 05	FINISHED	alphare	Refining S
18	11 Mar 05	FINISHED	alphare	Refining S
17	11 Mar 05	FAILED	alphare	Refining S
16	11 Mar 05	FINISHED	rantan	Find Se si
15	11 Mar 05	FINISHED	prephadata	Run revise
14	11 Mar 05	FINISHED	patterson	Anomalous

Figure 3 (above): the same list of jobs as in figure 1 with the "custom colours" option turned on, so that jobs with status FINISHED are shown in green; with status FAILED in red; with status KILLED in blue. (Colourising the job listing is an option available under System Administration->Configure Interface.)

Figure 4 (right): The "Search and Sort" window displaying results of a search for all jobs in the current project with status FINISHED. (The "Search and Sort" tool is available from the menu on the right-hand-side of the main CCP4i window.)



While these tools are very useful, the list-based views also have some limitations, for example as the number of jobs increases it can rapidly become difficult to see how the project has progressed. One of the aims of investigating different visualisations of the project history to help the user better understand their progress and current status.

The "dbviewer" is the first version of a new visualisation tool, and has been developed as a Tcl/Tk client application of the database handler. A screenshot is shown in figure 5, which gives a very different view of the same project shown in the other figures - it presents the project history data as a "directed graph", where each job is represented by a "node" in the graph. Jobs are linked together by lines wherever an output file from one job is used as input to another job, to show the flow of data through the various steps.

dbviewer aims to facilitate examination of the project history. Some of its features include:

- Use of different colours to indicate the status of a job (green for completed jobs, red for failed jobs, and yellow for those which are still running)
- The view can be reduced to subsets of jobs via different selection tools, for example: the visualiser can trace all the "antecedents" or "descendents" of a particular job selected by the user. Alternatively the user can manually select a subset of jobs to view. The user can also step back or forward through previous selections.
- "Filters" can be applied to the view, for example to only show jobs that were successfully completed, or which have at least one link to another job.
- Although the visualiser doesn't currently have any functions for modifying the database content, it is able to react to changes in the database in real time. So if a job is added or removed or changes status (for example because CCP4i is running at the same time) then the view will automatically update to reflect this.

Clearly this view and the functions that it offers are useful complements to the existing list-based view in CCP4i.

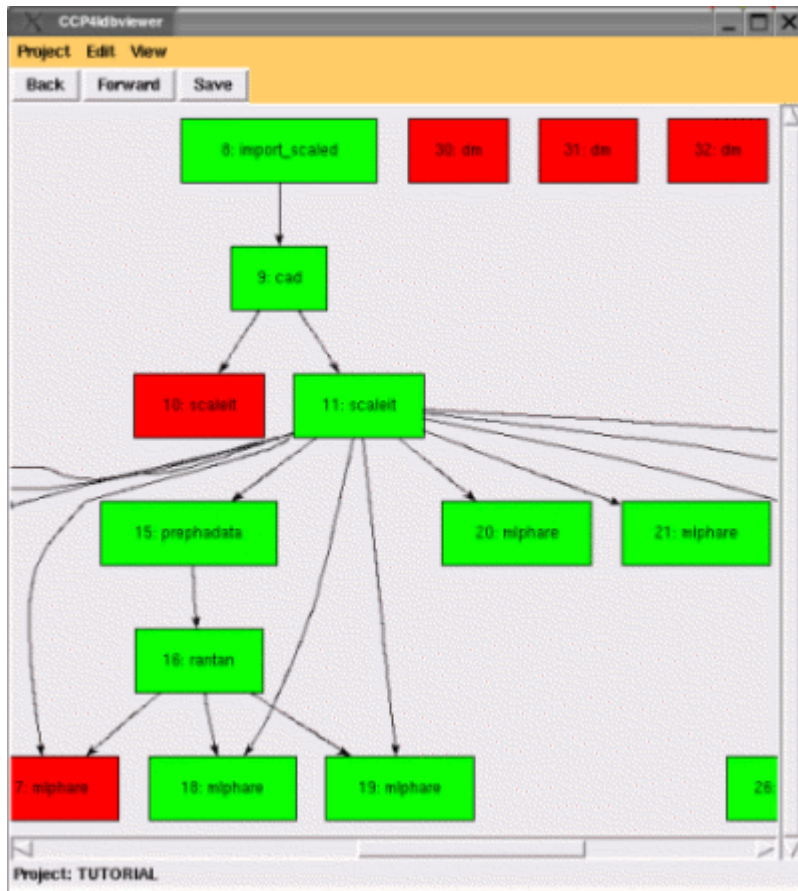


Figure 5: the same project data as shown in figures 2-4 drawn as a directed graph in the dbviewer application. Nodes represent the different jobs with lines between them to indicate the passage of data files.

3 Technical details for software developers

The following sections contain some of the more technical details of the project, which may be of interest to software developers who are considering using some of the components in their own projects. Specifically, they cover how the handler runs (section 3.1), how client applications communicate with the handler to access the database (section 3.2) and some background on the development of the database itself (section 3.3).

For anyone else not interested in these details, this section can safely be skipped.

3.1 How the handler runs

This section aims to clarify the details of how the handler operates in a normal usage environment.

The handler dbccp4i is implemented in Python. Each user runs a single handler process on any machine where they are running client applications that need to access the project database. Only one handler process runs for each user at any one time, and the dbccp4i process automatically manages itself to ensure that this is the case (an example of the [singleton design pattern](#)). The client API (see section 3.2) ensures that a client application always establishes connections with the running handler.

Because all client applications run by the same user access the projects through the same handler, and only the handler reads and writes to the project database, there should be no problems with multiple applications accessing the same resource simultaneously for that user.

Note however that there are still issues in the following situations:

- A user accesses the same project from two different machines (for example a project stored in a directory shared between those machines). In this case there will be two handler processes attempting to access the project simultaneously. The handler is designed to "fail safe" in these cases but the operation is not optimal.
- Different users try to access the same project database at the same time, creating a similar situation to the one outlined above.

Further issues occur because the handler operates in CCP4i's database environment, which could (perhaps rather generously) be characterised as a "distributed database architecture". In this environment there is no central database store, instead different data storage components are stored in different locations on the file system:

- Information about the names and locations of each project for a particular user is stored in that user's `directories.def` file, located in a subdirectory of the user's home area. This file only stores references to projects.
- All the data for a particular project is stored within the project directory for that project, and is independent of any other project.

This distributed setup is extremely flexible and robust, but also means that there are issues for example when one user is interested in sharing a project with another user.

3.2 How client applications communicate with the handler

Communication between the client applications and the handler process are via a simple language expressed in pseudo-XML markup. Requests to the handler from the clients, and responses to those requests, are transmitted using sockets.

One of the functions of the pseudo-XML is to make the communications independent of the programming languages used to implement the handler and the clients. However in practice, dealing directly with the details of socket communications and the pseudo-XML is a potentially significant burden to place on the developer of a client program, and could represent a large barrier to using the system.

To mitigate this, "client API" libraries have been developed that hide the details of the socket and communication protocols from the client. At present client APIs are being developed in Tcl and Python; examples 1 and 2 below show code fragments that illustrate using the APIs within a client application:

```

# Start the handler
if { ![DbStartHandler] } {
    puts "Unable to start the handler"
    exit 1
}
# Connect to the handler
if { ![DbHandlerConnect] } {
    puts "Unable to connect to the
handler"
    exit 1
}
# List the available projects
puts "Projects: [ListProjects]"
...

```

Example 1: Tcl script fragment to illustrate using the client API to connect to the handler and print a list of the user's projects.

```

# Start and connect to the handler
try:
    dbClientAPI.DbStartHandler()
    conn =
dbClientAPI.handlerconnection()
    conn.DbRegister(user, 'dummy', True)
except exceptions.Exception, e:
    print "Connection failed with
exception:"
    print str(e)
    sys.exit(1)
# List the available projects
print "Projects:
"+str(conn.ListProjects())
...

```

Example 2: Python script fragment illustrating the same operations as example 1.

The client APIs are intended to perform additional translations to render (for example) lists of data received from the handler, into the appropriate form for the programming language in question, and remove the burden of understanding and implementing the communication protocols from the application developer.

3.3 Developement of the "rich database"

As described previously in section 2.4, the initial version of the tracking database uses the existing CCP4i job database schema along with the def-file based storage. This provides a simple project history/tracking database system with a relatively straightforward practical implementation, which offers a good baseline for futher development.

However the ultimate aim is to provide a system that can accommodate not only the existing tracking data but also combine this with crystallographic data and other data, to produce a so-called "rich database". From discussions with pipeline developers such a database was envisaged as comprising three conceptual components:

- Knowledge Base: consists of the common crystallographic data items used in the software pipeline that are shared between different applications.
- Operational Database: contains arbitrary application-specific data and representations. Examples of these include CCP4i parameter files or serialised Python objects used in an automated pipeline. The essential difference between this and the knowledge base is that data in the operational database are not intended to be shared between applications.
- Tracking Database: stores the history of the data generation in the knowledge and operational databases. Tracking information should include

"data flow" (data items passed between application runs) and "logical flow" (for example, the sequence of application runs that are linked by some kind of logic).

Significant effort was initially expended attempting to create an SQL schema which would describe the content of the knowledge database. However it became clear that making a schema that was generally applicable to even a small number of pipelines (in our case only two!) was a far more difficult undertaking than originally expected. Progress was further hampered in a practical way by not being able to give the developers a "live" implementation of the system to experiment with.

Given this experience, our attention in this area has since focused on implementing a much smaller schema in order to get the technology working first. We then plan to start with a much smaller "seed" schema that can be grown by adding new data items. We are still optimistic that a useful knowledge database can be created, however this will need real usage cases in order to be successful. In this way we hope to avoid the problems that impeded our initial effort in this area.

Efforts to identify and implement an alternative database backend to the CCP4i def-file format have been more fruitful. MySQL was initially considered for this purpose, and although it has many attractive features it was ultimately rejected at this stage for a number of reasons:

- MySQL potentially places a significant burden on the end user in terms of installation, maintenance and usage. This is a particular consideration for CCP4 users who are not typically expert in these areas, and who are used to a very straightforward installation and usage pattern for CCP4, requiring very little setup.
- The centralised nature of MySQL would be difficult to integrate with the distributed database used in CCP4i (see section 3.2).

Instead an embedded SQLite database has been implemented in the handler as a testbed. This allows us to use SQL to express the database schema, while avoiding many of the drawbacks outlined above, and so far the back-end has been road-tested using a simple SQL version of the current CCP4i database. However in future we to use this to implement the knowledge database.

4 Current Status and Availability

We are currently preparing to make an initial public release version of the handler, the client API libraries for Tcl and Python, and the visualiser. These will all be compatible with the current CCP4i database, and can be run stand-alone.

Once this version is ready it will be announced via the appropriate email lists (for example the CCP4 developers bulletin board, see <http://www.ccp4.ac.uk/dev/dev-bb.php>). Alternatively if you would like us to

contact you directly when the public release is available then please get in touch with us by email. We are particularly interested in:

- Users of CCP4i who would like to try out the visualiser and provide feedback, and
- Software developers who are interested in using the handler and client APIs to communicate with the tracking database.

Finally, the handler, database and visualiser are among a number of publically-available deliverables being developed for the BIOXHIT project by CCP5. As these CCP4 deliverables are completed, they are being made available via the following URL:

- http://www.ccp4.ac.uk/projects/bioxhit_public/

and the initial version of the three components will be added to this page when they are released.

5 Future Plans & Developments

We are currently working on integrating dbCCP4i into CCP4i, to replace the current CCP4i database handling code with calls to the Tcl client API. A first version of this updated CCP4i should be ready in a few weeks, at which point a developmental version will be made publically available for people to try.

Unfortunately it is not clear whether this will be sufficiently tested in order to be considered ready for release with the next version of the CCP4 suite; however it is possible that a version of dbccp4i and the visualiser dbviewer may be included in CCP4 6.1, and subsequent releases of the suite should include CCP4i working with dbccp4i.

The current version of dbccp4i is missing some functionality that we know users would like to see, including functions to:

- Export and import projects
- Merge and split projects
- Synchronise two copies of a project that have branched
- Move jobs between projects

Aside from this, other longer-term plans include:

- Extending the def file tracking database in order to generalise the data items that are stored; this should make the database less CCP4i-centric,
- Work with external developers to enable their programs to interact with the database via the client API libraries,
- Revisit the knowledge base component of the database, by developing a small "seed" version implemented in SQLite that can be grown more easily with input from collaborators,

- Continue to develop the functionality of the visualiser based on user feedback.

We would also welcome any questions or feedback on the plans or on the work so far.

6 Useful Links

- The BIOXHIT Project website is at www.bioxhit.org
- The CCP4 deliverables from the project are available from http://www.ccp4.ac.uk/projects/bioxhit_public/
- The directed graphs in the visualiser are rendered using a custom Tcl library built on top of the Graphviz "dot" program; see www.graphviz.org.
- The website for the SQLite project can be found at www.sqlite.org. The Python API for SQLite can be obtained via <http://initd.org/tracker/pysqlite>. The MySQL website is at <http://www.mysql.com>.

7 Acknowledgements

PJB and WY's work are funded from the BIOXHIT project, which is funded by the European Commission within its FP6 Programme, under the thematic area "Life sciences, genomics and biotechnology for health", contract number LHSG-CT-2003-503420. Additional funds are provided from CCLRC Daresbury Laboratory via the CCP4 project.

PJB and WY would also like to acknowledge the various contributors who have helped by providing many useful suggestions via discussions, emails and testing - in particular Graeme Winter, Steven Ness, Daniel Rolfe and Charles Ballard.

Peter Briggs & Wanjuan Yang, January 15th 2007

RAPPER: Real Space Automated Conformer Generation

Nicholas Furnham[‡] and Tom L. Blundell

Department of Biochemistry, Cambridge University
80 Tennis Court Road, Cambridge, CB2 1GA, UK

[‡] Corresponding Author: nick@cryst.bioc.cam.ac.uk

Overview

Generating protein conformers by discrete sampling of likely conformers within a given set of restraints provides an attractive method for answering a number of important questions in structural biology [de Bakker, et al., 2006]. This approach, now incorporated in RAPPER, allows the generation of solutions consistent with a combination of restraints derived from a wide variety of sources, both experimental and theoretical. RAPPER can be applied to a number of problems including *ab initio* loop building [de Bakker, et al., 2003], comparative modelling [Furnham, et al., 2007] and C α -trace modelling [DePristo, et al., 2003]. It can also be used to build and refine conformers using X-ray crystallographic data [DePristo, et al., 2004, DePristo, et al., 2005, Furnham, et al., 2006], and this is now being integrated into the CCP4 software suite. The analysis tool RAMPAGE [Lovell, et al., 2003] is also incorporated into RAPPER in order to allow examination of ϕ / ψ values using the most up-to-date propensities to generate a Ramachandran diagram.

Methodology

At the heart of RAPPER lie two principal engines: restraint generation and fast and efficient sampling. The latter uses idealised geometry as the model representation of a protein (as formulated by Engh and Huber [Engh and Huber, 1991]) to generate conformers of main chain and, optionally, side chain heavy atoms. By using idealised geometry the number of degrees of freedom can be reduced to the major degrees of freedom present in a protein (ϕ , ψ , ω and χ). RAPPER samples ϕ / ψ space using fine grained ($5^\circ \times 5^\circ$) residue-specific propensity-weighted maps derived from the Top500 database of protein structures [Lovell, et al., 2003]. The ω dihedral angle is sampled as either the *cis* or *trans* state with the prevalence observed in the PDB [Berman, et al., 2000]. Special consideration is given to pre-proline residues which have a higher propensity to occur in the *cis* state [Jabs, et al., 1999]. In addition to this, side chain modelling is performed by sampling a rotamer library [Shetty, et al., 2003]. Although any library can be used, the principal library employed by RAPPER is the 'penultimate' library, which is derived in a similar way to the ϕ / ψ maps [Lovell, et al., 2003].

The sampling algorithm itself is based on a 'branch and bound population search' procedure. Essentially this comprises a sequential directed search from N to C termini, though there is no fundamental limitation to this and it equally could be C to N. At each residue a population of conformers (children) that extend the previous set of conformers (parents) that satisfy all the enforced restraints are searched. At each level in the search, pruning of an otherwise ever expanding search tree is achieved by maintaining only the conformers that satisfy the current restraints. The result is an exponential decrease in the number of conformers. In practice the size of the population is fixed at 100 and the number of iterations permitted to generate the child population is 100,000. This is summarised in Figure 1.

Sampling occurs within a given set of restraints. These can be both problem-specific and general. Common to all modelling is the hard-sphere excluded-volume restraint. This enforces a minimum distance between atoms, the van der Waals radii for which are taken from PROBE [Lovell, et al., 2000]. The process of checking for these excluded volumes can be very computationally expensive. This problem is addressed in RAPPER by an efficient grid-based checking system, which holds only a small substructure of atoms at each grid point that might overlap with an atom lying near that point. The number of atoms at the grid point that are required to be checked is dependent only on the resolution of the grid and the distribution of the fixed atoms and is independent of the size of the protein. In addition the rate at which atoms are fixed to the grid as they are being built is optimised to a single common ancestor at most twenty residues preceding the residue being built. Thus only a small fraction of the diverse population local (in sequence space) to the residue being built needs to be checked; the rest are reduced to a single shared path of the search tree. Problem-specific restraints include positional restraints to ensure that the sampled conformers fall within a specific position in space; this is utilised in tracing a structure from C α -atom positions. Other restraints derived for loop modelling include gap closure, C-terminal anchor dihedral angle minimisation and strict C-anchor positioning. For X-ray crystallography electron density can be used to derive restraints by ensuring that atoms from the sampled conformers lie within electron density cut-off by a minimum σ threshold.

Applications

The generalised nature of the sampling algorithm permits the application of RAPPER to a number of problems in structural biology. In the first instance it was applied to generating loop regions [de Bakker, et al., 2003]. Further development permitted the regeneration of the entire protein from C α -atoms [DePristo, et al., 2003]. Most recently the combination of these early developments with restraints from electron density has assisted X-ray structure determination.

Although there has been significant progress in developing model building and refinement techniques for structure determination by X-ray crystallography, it still remains a challenge to produce reliable high quality models of proteins from medium and low resolution crystals. This is primarily due to the difficulty in adequately exploring the large and complex energy landscape and determining the set of conformers that best describes the experimental data. In order to address this problem we have explored the use of the real space knowledge-based conformational sampling in RAPPER coupled with reciprocal space refinement by either molecular dynamics / simulated annealing (MD/SA) or maximum likelihood.

We have demonstrated that the combined approach of RAPPER and MD/SA greatly improves the quality and power of refinement when compared to MD/SA alone at medium resolution (2.0-3.0Å) [DePristo, et al., 2004, DePristo, et al., 2005]. This is shown through the re-determination of two previously solved structures, providing a controlled situation to test the procedure, as well as a more realistic challenge of a blind test structure determination. In comparing the combined approach to MD/SA alone we show that RAPPER is able, by discrete sampling, to cross deep wells within the X-ray potential energy landscape from which MD/SA alone cannot escape, particularly those

involving bulky side chains. Limiting the degrees of freedom by sampling only good ϕ/ψ characteristics or rotameric side chains and the enforcement of prior values, such as Engh-Huber bond angles and lengths gained from knowledge of protein structure, avoids exploring unlikely and fruitless areas of conformational space. It is these areas that, at lower resolutions, are likely to accommodate conformations with poor ϕ/ψ and rotameric side chains. This approach helps to overcome the limited specificity, modal bias and phase error of lower resolution maps.

The approach has been further developed to push the boundaries of *de novo* model building and refinement to lower resolutions ($>3\text{\AA}$). This has been demonstrated through the application of RAPPER, along with refinement by maximum likelihood, to the structure of Lif1-Lig4, a 90KDa complex of two protomers involved in the final ligation step of DNA non-homologous end joining. The native dataset from crystals of this complex was collected to $3.9\text{\AA}$ resolution, and experimental phases were collected to $\sim 4.5\text{\AA}$ resolution [Dore, et al., 2006].

The advantage of using RAPPER to model into low resolution data is that it enables the testing of weak hypotheses, assumptions and often speculations about structures suggested by the electron density map. Sequence registry indicated by features in the density as well as information inferred from homologous sequences and structures can be tested. Efficient exploration of conformational space allows the uncertainty of the dataset to be taken into account; this is important for low resolution data [Furnham, et al., 2006].

In all applications RAPPER is able to generate an ensemble of solutions that more or less equally fit the data. Such a crystallographic ensemble can represent one of two possibilities. Firstly, there may be several individual structures that are equally compatible with the data when considered one at a time; these can be thought of as describing the uncertainty of the experimental data. Secondly it can represent spatial heterogeneity and dynamics: a set of structures that, taken over time and the different unit cells, are compatible with the data. Obviously such ensembles can reflect aspects of both factors and at present we are unable to distinguish between the two, but in any case the collection epitomizes the inappropriateness of specification of a model as a single species.

RAMPAGE

The use of the fine grained ϕ/ψ residue specific propensity-weighted maps derived from the Top500 database of protein structures in RAPPER allows for the geometrical evaluation of a protein model to be undertaken. To this end RAMAPGE [Lovell, et al., 2003] has been developed in collaboration with the Richardson Group. This generates a Ramachandran diagram plotting ϕ verses ψ backbone conformational angles for each residue in the model. These can be classified into regions of favoured, allowed and disallowed regions. The contouring of these regions is based on density-dependent smoothing for 81,234 non-Gly, non-Pro, and non-prePro residues with $B < 30$ from 500 high-resolution proteins. Delineation between large empty areas and regions that are allowed but disfavoured can be discerned. One such region is the γ -turn conformation near $+75^\circ, -60^\circ$, counted as forbidden by common structure-validation programs; however, it occurs in well ordered parts of good structures and is

overrepresented near functional sites. Regions are also defined for Pro, pre-Pro, and Gly (see Figure 2 and 3). These plots provide crystallographers with a n up to date tool for locating and fixing local problems during the process fitting electron density and refinement as well as a general assessment of model quality and the clarify if a structure generally meets current standards of good practice in the field [Lovell, et al., 2003].

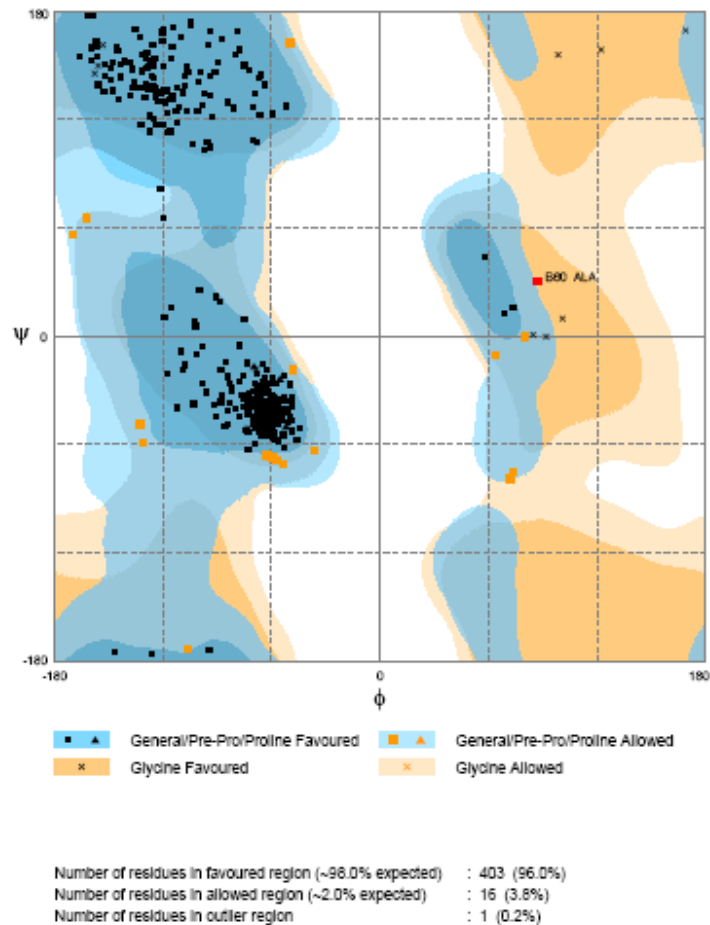


Figure 2: Ramachandran Plot Generated by RAMPAGE. The ϕ / ψ distribution, as analysed by RAMPAGE, for the structure of the complex Xrcc4 and a fragment of Ligase IV (PDB code 1IK9).

model between structural elements generated by automated building programs such as Buccaneer [Cowtan, 2006]. Once a complete or near complete model has been generated RAPPER, in combination with a refinement programme such as REFAMC, can be used to identify automatically, and rebuild poor fitting regions before refining the structure. In a more user interactive mode of operation, where weak or difficult to interpret density exists (both at reasonable resolution and low resolution), RAPPER can be a powerful tool in providing sets of conformers consistent with the experimental data and taking into account any prior knowledge about the structure. To aid the entire process it is hoped that RAPPER will be able to be used though COOT [Emsley and Cowtan, 2004]. Further information can be found at the RAPPER website <http://mordred.bioc.cam.ac.uk/~rapper/>.

References

1. Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., and Bourne, P.E. (2000). The Protein Data Bank. *Nucleic Acids Res* 28, 235-242.
2. Cowtan, K. (2006). The Buccaneer software for automated model building. 1. Tracing protein chains. *Acta Crystallogr D Biol Crystallogr* 62, 1002-1011.
3. de Bakker, P.I., DePristo, M.A., Burke, D.F., and Blundell, T.L. (2003). Ab initio construction of polypeptide fragments: Accuracy of loop decoy discrimination by an all-atom statistical potential and the AMBER force field with the Generalized Born solvation model. *Proteins* 51, 21-40.
4. de Bakker, P.I., Furnham, N., Blundell, T.L., and Depristo, M.A. (2006). Conformer generation under restraints. *Curr Opin Struct Biol*.
5. DePristo, M.A., de Bakker, P.I., and Blundell, T.L. (2004). Heterogeneity and inaccuracy in protein structures solved by x-ray crystallography. *Structure (Camb)* 12, 831-838.
6. DePristo, M.A., de Bakker, P.I., Johnson, R.J., and Blundell, T.L. (2005). Crystallographic refinement by knowledge-based exploration of complex energy landscapes. *Structure (Camb)* 13, 1311-1319.
7. DePristo, M.A., De Bakker, P.I., Shetty, R.P., and Blundell, T.L. (2003). Discrete restraint-based protein modeling and the C α -trace problem. *Protein Sci* 12, 2032-2046.
8. Dore, A.S., Furnham, N., Davies, O.R., Sibanda, B.L., Chirgadze, D.Y., Jackson, S.P., Pellegrini, L., and Blundell, T.L. (2006). Structure of an Xrcc4-DNA ligase IV yeast ortholog complex reveals a novel BRCT interaction mode. *DNA Repair (Amst)* 5, 362-368.

9. Emsley, P., and Cowtan, K. (2004). Coot: model -building tools for molecular graphics. *Acta Crystallographica Section D -Biological Crystallography* **60**, 2126-2132.
10. Engh, R.A., and Huber, R. (1991). Accurate Bond and Angle Parameters for X - Ray Protein-Structure Refinement. *Acta Crystallographica Section A* **47**, 392-400.
11. Furnham, N., de Bakker, P.I., Gore, S., Burke, D.F., and Blundell, T.L. (2007). Comparative Modelling by Restraint -based Conformational Sampling
12. Furnham, N., Dore, A.S., Chirgad ze, D.Y., de Bakker, P.I., Depristo, M.A., and Blundell, T.L. (2006). Knowledge -based real-space explorations for low -resolution structure determination. *Structure* **14**, 1313-1320.
13. Jabs, A., Weiss, M.S., and Hilgenfeld, R. (1999). Non -proline cis peptide bonds in proteins. *J Mol Biol* **286**, 291-304.
14. Lovell, S.C., Davis, I.W., Adrendall, W.B., de Bakker, P.I.W., Word, J.M., Prisant, M.G., Richardson, J.S., and Richardson, D.C. (2003). Structure validation by C alpha geometry: phi,psi and C beta deviation. *Proteins-Structure Function and Genetics* **50**, 437-450.
15. Lovell, S.C., Word, J.M., Richardson, J.S., and Richardson, D.C. (2000). The penultimate rotamer library. *Proteins* **40**, 389-408.
16. Shetty, R.P., De Bakker, P.I., DePristo, M.A., and Blundell, T.L. (2003). Advantages of fine-grained side chain conformer libraries. *Protein Eng* **16**, 963-969.

Diffraction Image: a new CCP4 library

*Francois Remacle, Graeme Winter, CCLRC Daresbury Laboratory Warrington WA4 4AD,
United Kingdom*

What is this library?

The Diffraction Image library started as part of XIA automation project. Since it appeared that it could be useful to have it as a separate entity, it was decided to include this library with the other CCP4 core libraries and therefore make it available publicly. XIA2 now relies a couple of simple programs using this library.

The purpose of the library is to have a single way of handling diffraction image that can be originally of various different formats. Therefore, even if the library contains several different files implementing similar methods for different format of images it has been designed so that you only need to use a single object. All the work of identifying what type of image is done internally.

Currently, the following format are supported by this library:

- ADSC
- MAR: MAR345, MAR300, MARCCD
- RIGAKU: R-AXIS IV, SATURN
- CBF

The library also contains a PeakList object that can be populated with the peaks (spots) found on particular images, there are then some operations that can be performed on the peaks.

The Diffraction Image Library comes with Tcl and Python interfaces so that the library can be used with both of these scripting languages as well.

How to use it and what can you do with it?

Firstly, below is the description of the classes from the Diffraction Image library, there are only two classes / objects availables which make things a lot simple to use.

There are different ways for initialising a diffraction Image Object. Which one is more appropriate mainly depends on what you want to do.

- You can create an empty object and load an image in the object via the
`load`
`method`.
- You can create an object giving the filename as a parameter to have everything already loaded (i.e. header and image).

- You can create an object and only load the header via the

`loadHeader`

method. This is useful if you just need to access information from the header.

To illustrate this let us use some source code

```
void printHeader(char* filename)
{
    DiffractionImage* diff=new DiffractionImage();
    diff->loadHeader(filename);
    printf("Image type : %s\n",diff->getFormat());
    printf("Collection date : %s\n",diff->getDate());
    printf("Exposure time : %f s\n",diff->getExposureTime());
    printf("Detector S/N : %s\n",diff->getSerialNo());
    printf("Wavelength : %f Ang\n",diff->getWavelength());
    printf("Beam center : (%f mm,%f mm)\n",diff->getBeamX(),diff->getBeamY());
    printf("Distance to detector : %f mm\n",diff->getDistance());
    printf("Image Size : (%d px, %d px)\n",diff->getWidth(),diff->getHeight());
    printf("Pixel Size : (%f mm, %f mm)\n",diff->getPixelX(),diff->getPixelY());
    printf("Oscillation range : %f -> %f deg\n",diff->getPhiStart(),diff->getPhiEnd());
    printf("Two Theta value: %f deg\n",diff->getTwoTheta());
}
```

We used the third option because in this method we just want to output the header information. But the lines

```
DiffractionImage* diff=new DiffractionImage();
diff->loadHeader(filename);
```

Could have been replaced by

```
DiffractionImage* diff=new DiffractionImage(filename);
```

or

```
DiffractionImage* diff=new DiffractionImage();
diff->load(filename);
```

There is not one more suitable than another in general, it mainly depends on what you intend to do, especially whether or not you need to load the image in memory which can slow down a treatment of large number of images.

Examples of programs

Currently the library provides also two simple programs called "diffdump" and "printpeaks" that are respectively used to display all the "standard" information of a Diffraction Image and printing the list of peaks found on the image. These are also simple examples of how to use the object from the library.

diffdump usage:

```
"diffdump <filename>"
```

Example of running diffdump:

```
>diffdump adsc_example_001.img
Image type : adsc
Collection date : Sun Sep 26 14:01:35 2004
Exposure time : 5.000000 s
Detector S/N : 445
Wavelength : 0.979660 Ang
Beam center : (105.099998 mm,101.050003 mm)
Distance to detector : 170.000000 mm
Image Size : (2048 px, 2048 px)
Pixel Size : (0.102400 mm, 0.102400 mm)
Oscillation range : 290.000000 -> 291.000000 deg
Two Theta value: 0.000000 deg
```

printpeaks usage:

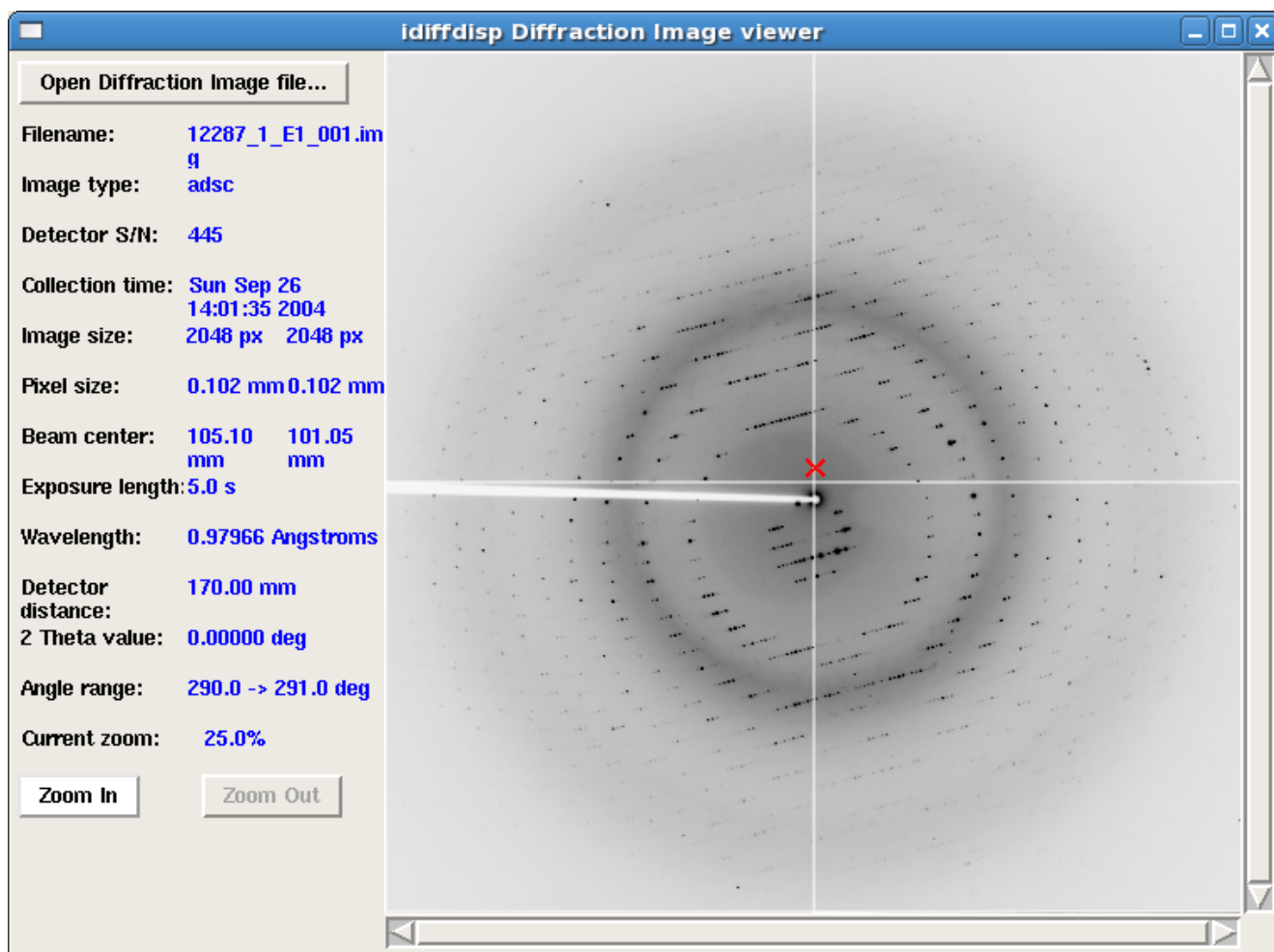
```
"printpeaks [-th <intensity_threshold>] <filename>"
```

Example of running printpeaks

```
>printpeaks -th 5 12287_1_E1_001.img
Searching for peaks...
Done!
 1212 peaks found

Peak 1 = x:17.642262 y:5.422019 phi:290.500000 intensity:411.251343
Peak 2 = x:14.642123 y:-38.075809 phi:290.500000 intensity:327.479645
Peak 3 = x:11.323599 y:11.402020 phi:290.500000 intensity:326.635406
Peak 4 = x:16.016949 y:11.563182 phi:290.500000 intensity:312.215332
Peak 5 = x:-14.611582 y:45.403568 phi:290.500000 intensity:297.010834
Peak 6 = x:4.550874 y:-46.884823 phi:290.500000 intensity:285.352112
Peak 7 = x:4.951574 y:-0.162381 phi:290.500000 intensity:284.619934
Peak 8 = x:14.114961 y:-26.515749 phi:290.500000 intensity:265.874084
Peak 9 = x:31.641386 y:-19.859549 phi:290.500000 intensity:258.448975
Peak 10 = x:-17.301394 y:-37.574718 phi:290.500000 intensity:235.708313
... ..
Peak 1203 = x:6.917424 y:89.008492 phi:290.500000 intensity:5.065027
Peak 1204 = x:-57.386951 y:78.550629 phi:290.500000 intensity:5.062943
Peak 1205 = x:92.328209 y:19.058695 phi:290.500000 intensity:5.055988
Peak 1206 = x:-80.527000 y:-22.509737 phi:290.500000 intensity:5.051531
Peak 1207 = x:-69.974136 y:51.322342 phi:290.500000 intensity:5.051506
Peak 1208 = x:-67.926010 y:-52.717159 phi:290.500000 intensity:5.049273
Peak 1209 = x:-42.943699 y:-5.723580 phi:290.500000 intensity:5.041045
Peak 1210 = x:-5.873695 y:57.763443 phi:290.500000 intensity:5.033473
Peak 1211 = x:41.841999 y:-63.973579 phi:290.500000 intensity:5.027031
Peak 1212 = x:-43.768307 y:96.585030 phi:290.500000 intensity:5.017936
```

In addition to these a tcl image viewer called idiffdisp is being developed and will be later accessible through ccp4i and be used by XIA2 it uses the tcl interface of the library to access all methods from the library. It currently displays the header information on the left part of the window and the image on the right part, it also enables different zoom level (25%, 50%, 100%, 200%). Below you can see a screenshot of the current development state of idiffdisp.



Acknowledgments

- Rigaku for supplying demonstration code for Raxis IV images.
- Many different suppliers of example images of various detectors.
- Paul J. Ellis and Herbert J. Bernstein authors of the CBF library on which this library relies to support CBF format.

Appendix: Library API

- DiffractionImage

Method	Description
DiffractionImage()	Default constructor of a Diffraction Image.
DiffractionImage(string filename)	Create and load everything available from the image file called "filename".
int getWidth()	Return the width of image in pixels.
int getHeight()	Return the height of image in pixels.
float getBeamX()	Return the x position of the beam center in millimeters.
float getBeamY()	Return the y position of the beam center in millimeters.

float getWavelength()	Return the wavelength of X-Ray source in Angstroms.
float getDistance()	Return the crystal to detector distance in millimeters.
float getTwoTheta()	Return the two-theta angle used in data collection
float getPhiStart()	Return the phi-angle at the beginning of the oscillation
float getPhiEnd()	Return the phi-angle at the end of the oscillation
float getPixelX()	Return the width of a pixel in millimeters.
float getPixelY()	Return the height of a pixel in millimeters.
float getExposureTime()	Return the exposure time in seconds.
char* getFormat()	Return "mar", "adsc", "raxis", etc... according to the image format.
char* getDate()	Return the date of the image collection using a formatted string (dd-mm-YYYY).
char* getSerialNo()	Return the detector serial number if available.
unsigned short * getImage()	Return a pointer to the image.
void setBeamX(float x)	Sets the x-position of the beam center in millimeters.
void setBeamY(float y)	Sets the y-position of the beam center in millimeters.
void setWavelength(float wave)	Sets the wavelength in Angstroms.
void setDistance(float dist)	Sets the crystal to detector distance in millimeters.
void setTwoTheta(float twotheta)	Sets the two-theta angle used in data collection.
void setPhiStart(float phistart)	Sets the phi-angle at the beginning of the oscillation
void setPhiEnd(float phiend)	Sets the phi-angle at the end of the oscillation
void setPixelX(float pixelwidth)	Sets the pixel size in millimeters
void setPixelY(float pixelheight)	Sets the pixel size in millimeters
void setExposureTime(float time)	Sets the exposure time in milliseconds
void load(string fileName)	Load in an image from the file "filename".
void loadHeader(string fileName)	Load header information only from the file "filename".
float* assymetry(void)	Give an estimate of how much symmetry there is in the image background.
float* radial(int bins)	Generate a radial profile of the image
int corner(void)	Determine whether the image goes to the corner or whether it is an inscribed circle.
int* histogram(void)	Calculate a contrast histogram.
float gain(void)	<i>Calculate an estimation of the gain (yet to be implemented).</i>

- PeakList

Method	Description
PeakList()	Default constructor of a empty PeakList.
PeakList(DiffractionImage* diffractionimage)	Construct a PeakList from the given DiffractionImage object.
int length()	Gives the length of the list.
void clear()	Clears the list.
void add(float x, float y, float intensity)	Add a peak the image coordinates (x,y) and with the specified intensity.
void remove(int offset)	Remove the Peak at the specified position in the list.
void add(Peak peak)	Add the specified peak to the list.
void find(DiffractionImage* diffractionimage)	Search for the peaks in the specified DiffractionImage object.
void find(DiffractionImage* diffractionimage, int maxPeaks)	Search for the peaks in the specified DiffractionImage object with a maximum of maxPeaks results.
void find(DiffractionImage* diffractionimage, int maxPeaks, float intensityThreshold)	Search for the peaks in the specified DiffractionImage object with a maximum of maxPeaks results having an intensity of at least intensityTreshold.
void reciprocal(float dist, float wave)	Convert the peak coordinates from dectetor coordinates to reciprocal space coordinates.
void detector(float dist, float wave)	Convert the peak coordinates from reciprocal space coordinates to detector coordinates.
Peak pop(int offset)	Gets the peak from the specified position and removes it from the list.
Peak * getPeaks()	Return the content of the PeakList as a vector of Peak.
int circle(int iter, float width, float& x, float& y, float& r)	Find a circle in the spot using a stochastic process and return the most promising candidate. Width determines how close to the arc a spot must be to be accepted. Iter is the number of attempts in fitting a circle.
int remove(float width, float x, float y, float r)	Removes the spots that were found to be on the circle fitted with the circle() method described above.

- Peak is a structure having the following 7 fields,
 - o **float x** : x coordinate of the peak on the image
 - o **float y** : y coordinate of the peak on the image
 - o **float theta** : Theta value for the image where the spot is found
 - o **float phi** : phi value for the image where the spot is found
 - o **float kappa** : kappa value for the image where the spot is found
 - o **float[3] p** : Array containing the reciprocal spaces coordinates of the Peak
 - o **float intensity** : Intensity of the Peak

A self-rotation puzzle

Zhenbo Cao^{1,2,*}, Neil W. Isaacs¹

¹WestCHEM, Department of Chemistry, University of Glasgow, Glasgow G12 8QQ, UK

²Department of Biochemistry and Molecular Biology, University of Glasgow, Glasgow
G12 8QQ, UK

*corresponding author, E-mail: Zhenbo@chem.gla.ac.uk

Introduction

The peroxiredoxins (Prxs) are a ubiquitous family of antioxidant enzymes that regulate intracellular levels of H₂O₂ where they are implicated in both tissue protection against oxidative stress and H₂O₂-mediated signalling pathways (Wood *et al.* 2003). In recent years, their key role in antioxidant defence has been emphasised by their high abundance in both bacterial and mammalian cells. Peroxiredoxin III (Watabe *et al.* 1994) is a typical member of the 2-Cys PrxIII subclass with catalytic cysteines at its N(Cys47) and C(Cys168) termini and with a dimer as the functional unit. Electron Microscopy (EM) studies (Gourlay *et al.* 2003; Wood *et al.* 2003) have shown that PrxIII exists as an oligomeric ring. We have determined the crystal structure of bovine mitochondrial PrxIII C168S mutant at 3.3Å resolution (Cao *et al.* 2005).

What is the puzzle?

Crystals production, data collection and processing were described previously (Cao *et al.* 2005). The crystals belong to the monoclinic space group C2 with the Matthews coefficient (Matthews 1968) suggesting 10 ($V_m=2.8 \text{ Å}^3/\text{Da}$) or 12 ($V_m=2.3 \text{ Å}^3/\text{Da}$) monomeric subunits in the crystal asymmetric unit. The usual statistical indicators (CCP4 1994) gave no indication of crystal twinning. Since most (6 out of 8) known typical 2-Cys Prxs structures are decamers, a self-rotation function was calculated using the program MOLREP to locate the expected NCS two-fold and five-fold axes. The

results are shown in Figure 1 as stereographic projections of polar angles. The absence of substantial peaks on the $\chi=72^\circ$ section indicate there is no 5-fold symmetry in the crystal structure (Fig 1a). However, 3 peaks on the $\chi=60^\circ$ section indicate three 6-fold symmetry axes, each of which is perpendicular to 6 two-fold axes shown as lines of 6 peaks in the $\chi=180^\circ$ section (Fig 1b). In each line the peaks occur at an angle of about 30 degrees to each other and are perpendicular to a 6-fold peak, which also coincident with a 2-fold peak. Taken together, these facts suggest a dodecameric rather than a decameric structure. However, as the Matthews coefficient suggested only one dodecameric ring in the asymmetric unit, the indication of three different 6-fold axes of rotation really puzzled us.

Initial attempts to solve the structure by molecular replacement by MOLREP (CCP4 1994), AMoRe (Navaza 1994) and PHASER (Storoni *et al.* 2004) with the default parameters using thioredoxin peroxidase B (TPxB) (PDB Code 1qmv) monomer, dimer and decamer as the starting model were all tried without any successful solutions. A solution was finally obtained by the molecular replacement program PHASER (Storoni *et al.* 2004) using the dimer of TPxB as a search model (Cao *et al.* 2005).

What is the answer to the puzzle?

The structure shows that stable PrxIII dimers are formed across a non-crystallographic 2-fold axis that extends the central β -sheet. The 6-fold NCS-related dimers are assembled into a dodecameric ring structure with outer and inner diameters of 150 and 70 Å respectively (*cf* 130 and 60 Å for the decameric Prxs) (Fig 2a and 2b).

The surprising feature of the crystal structure of PrxIII C168S is its presence as a 2-ring catenane comprising two interlocking dodecameric toroids (Fig. 2c), which are arranged such that half of one ring is related to half of the other ring by the crystallographic 2-fold axis. The planes of the rings are not at right angles, but are inclined at an angle of 55° , which allows a larger contact surface between the rings. This structure provides the solution to the self-rotation puzzle. The two 6-fold peaks ($\chi=60^\circ$) at $\phi=66^\circ$, $\phi=104^\circ$ and $\phi=66^\circ$, $\phi=-104^\circ$, with coincident 3-fold ($\chi=120^\circ$) and 2-fold ($\chi=180^\circ$) axes, are the two 6-

fold axes perpendicular to the planes of the rings. The two lines of six 2-fold peaks ($\phi = 90, \varphi = +165; \phi = 64, \varphi = +150; \phi = 38, \varphi = +130; \phi = 25, \varphi = +75; \phi = 38, \varphi = +23; \phi = 64, \varphi = 0$) represent the 2-fold axes in the plane of the ring. Because the ring is composed of six homodimers, there are 12 2-fold axes in the plane, giving a 30 degree angle between adjacent axes. What is the large 6-fold, peak at $\phi = 64, \varphi = 0$? This is actually the tail of a peak arising from an improper rotation of 55 degrees (Fig 3a), which is the angle between the planes of the two rings (Fig 3b). Finally, the 2-fold peak at $\phi = 26, \varphi = 180$ is perpendicular to this and relates the two rings as shown in Fig 3b where the axis is in the plane of the page and runs vertically through the centre of the model. The two peaks at $\phi = 90, \varphi = 90$ and $\phi = 90, \varphi = -90$ are the crystallographic 2-fold symmetry axes of the spacegroup C2.

Other catenanes

There are three previous examples of protein catenanes cited in the literature and two of them are rather specialised cases. One is a totally artificially -produced peptide catenane based on a small segment of a dimeric mutant of the p53^{tet} protein generated *in vitro* using chemical techniques (Yan and Dawson 2001). Another one is a viral capsid assembly of 420 subunits where the subunits are topologically linked by covalent (isopeptide) bonds creating a form of protein 'chain mail' which is highly resistant to degradation (Wikoff *et al.* 2000). The third example is the crystal structure of RecR from *Deinococcus radiodurans*, which is involved in homologous recombinational DNA repair in procaryotes (Lee *et al.* 2004) (PDB ID: 1VDD). Four RecR monomers form a ring-shaped tetramer of 222 symmetry with a central hole of 30-35 Å diameter. In the crystal, two tetramers are concatenated (Fig 4).

How is the catenane formed?

We have no data indicating how the 2-ring catenane structure is formed but a model described previously (Cao *et al.* 2005) is shown in Fig 5. Briefly, dimeric units can interact in two different modes that are not mutually exclusive. One mode produces the dimer-dimer contacts, primarily hydrophobic, associated with ring generation in this and other Prx structures. The other mode gives polar contacts that could potentially initiate

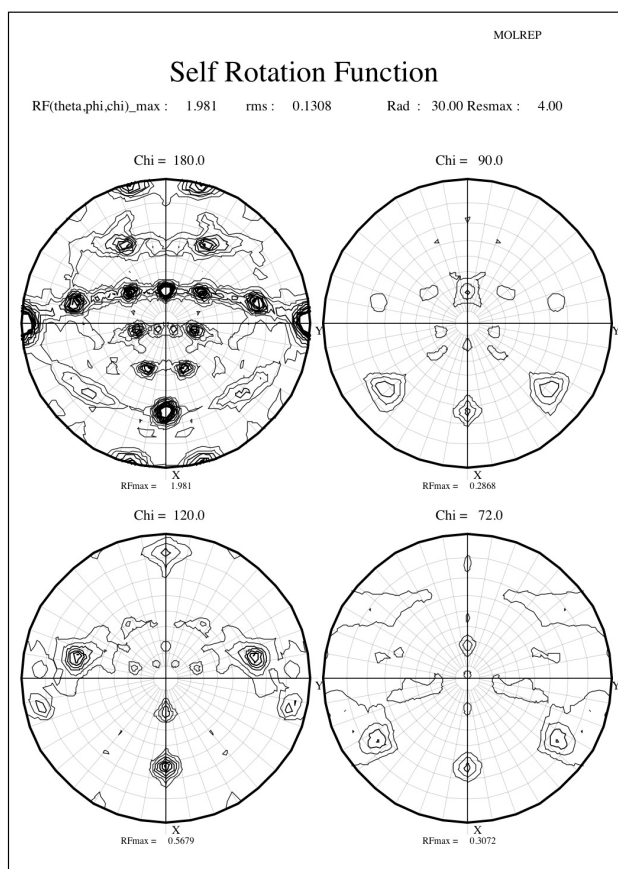
catenane formation at any stage during single toroid assembly by allowing two rings to form simultaneously around each other.

At present it is unclear whether the catenane structure has any physiological relevance, but it provides interesting new insights into protein topology and mechanisms of subunit assembly.

References:

- Cao, Z., A. W. Roszak, et al. (2005). "Bovine mitochondrial peroxiredoxin III forms a two - ring catenane." *Structure (Camb)* **13**(11): 1661-4.
- CCP4 (1994). "The CCP4 Suite - Programs for Protein Crystallography." *Acta Crystallographica Section D-Biological Crystallography* **50**: 760-763.
- Gourlay, L. J., D. Bhella, et al. (2003). "Structure -function analysis of recombinant substrate protein 22 kDa (SP-22). A mitochondrial 2-CYS peroxiredoxin organized as a decameric toroid." *J Biol Chem* **278**(35): 32631-7.
- Lee, B. I., K. H. Kim, et al. (2004). "Ring -shaped architecture of RecR: implications for its role in homologous recombinational DNA repair." *Embo J* **23**(10): 2029-38.
- Matthews, B. W. (1968). "Solvent content of protein crystals." *J Mol Biol* **33**(2): 491-7.
- Navaza, J. (1994). "Amore - an Automated Package for Molecular Replacement." *Acta Crystallographica Section A* **50**: 157-163.
- Storoni, L. C., A. J. McCoy, et al. (2004). "Likelihood -enhanced fast rotation functions." *Acta Crystallogr D Biol Crystallogr* **60**(Pt 3): 432-8.
- Watabe, S., H. Kohno, et al. (1994). "Purification and characterization of a substrate protein for mitochondrial ATP -dependent protease in bovine adrenal cortex." *J Biochem (Tokyo)* **115**(4): 648-54.
- Wikoff, W. R., L. Liljas, et al. (2000). "Topologically linked protein rings in the bacteriophage HK97 capsid." *Science* **289**(5487): 2129-33.
- Wood, Z. A., E. Schroder, et al. (2003). "Structure, mechanism and regulation of peroxiredoxins." *Trends Biochem Sci* **28**(1): 32-40.
- Yan, L. Z. and P. E. Dawson (2001). "Design and Synthesis of a Protein Catenane." *Angew Chem Int Ed Engl* **40**(19): 3625-3627.

a)



b)

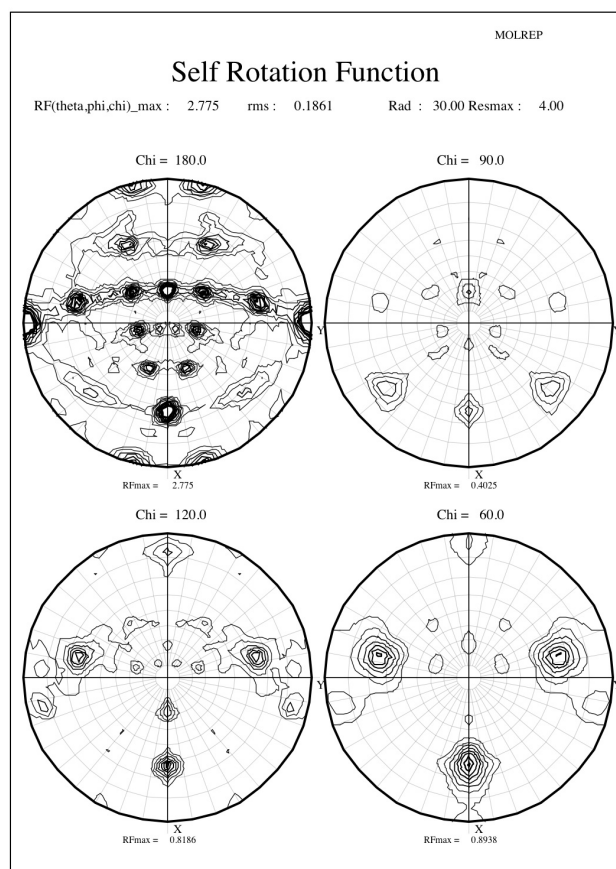


Figure 1 Self-rotation functions calculated by MOLREP with the chi angles shown.

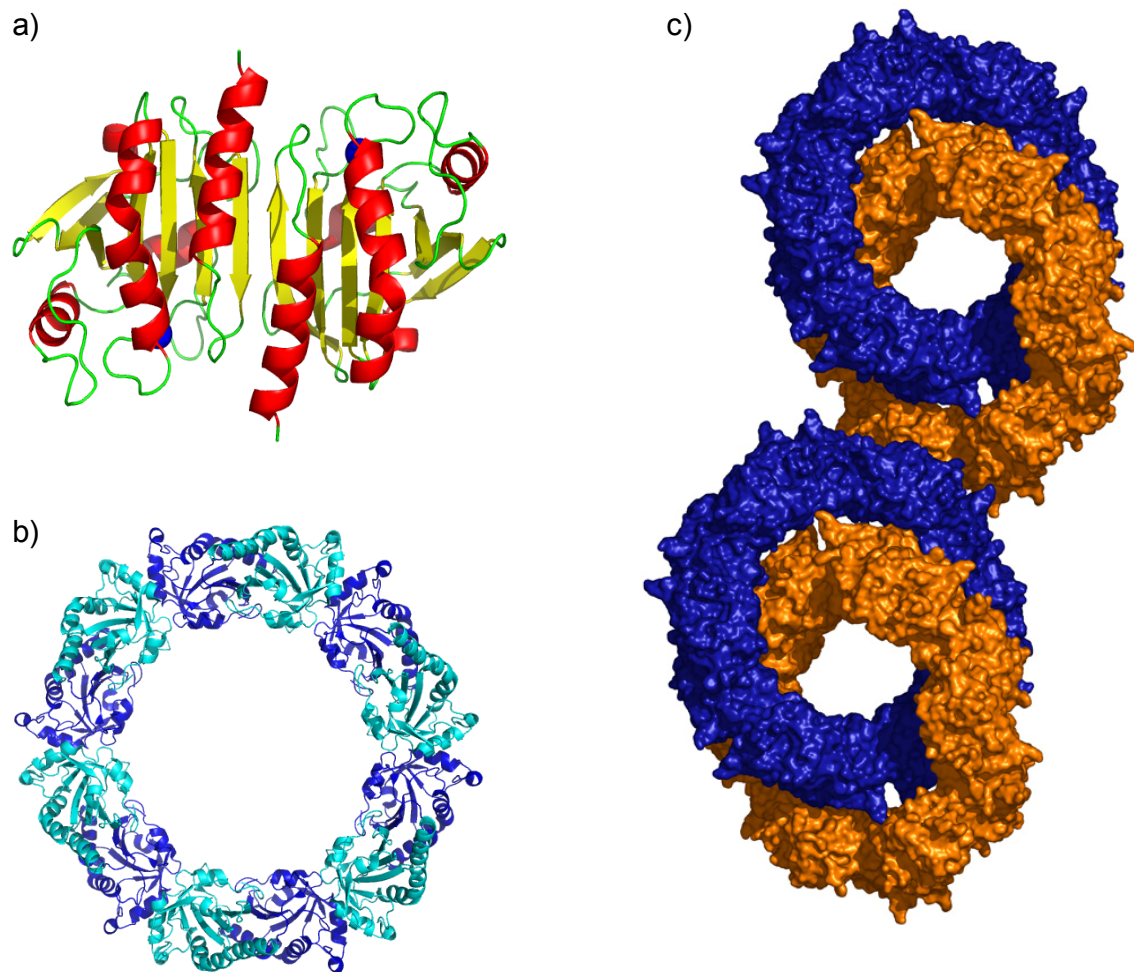


Figure 2 Crystal structure of PrxIII C168S

- a) PrxIII dimer. The active site Cys47 is highlighted by blue ball presentation.
- b) PrxIII dodecamer
- c) Two dodecamer rings (gold and blue) of PrxIII form the interlocked 2 -ring protein catenane structure in the unit cell.

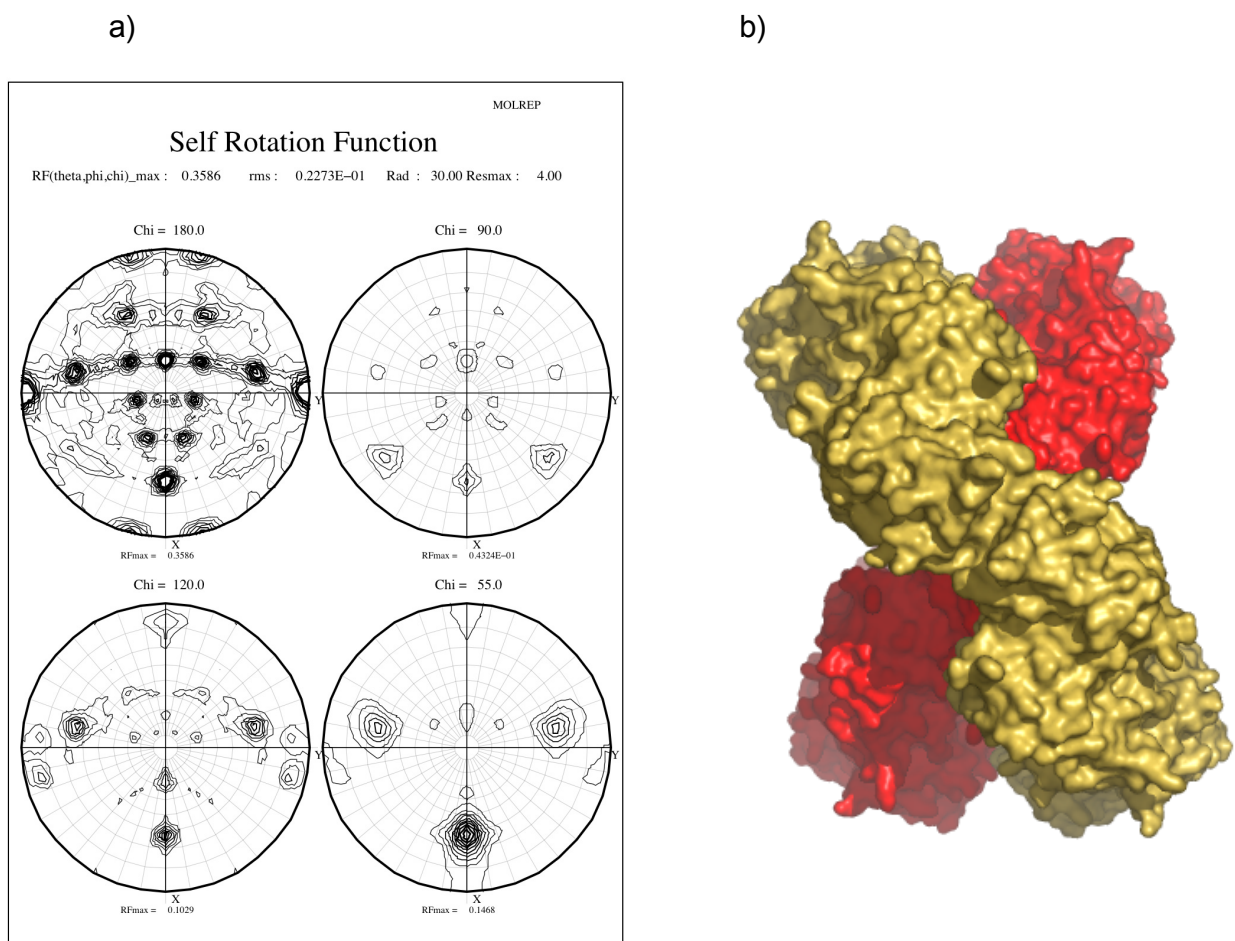


Figure 3

- a) The self-rotation function calculated by MOLREP with chi angle at 55 degrees.
- b) The side view of the 2-ring catenane structure. The figure was take from (Cao *et al.*

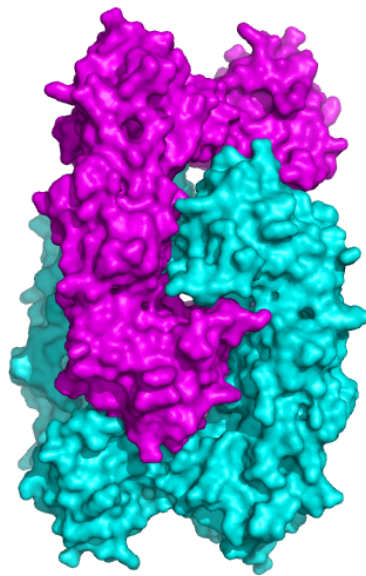


Figure 4 The crystal structure of *Deinococcus radiodurans* RecR octamer. Two tetramers (pink and cyan) are concatenated to form an octameric structure in the crystal.

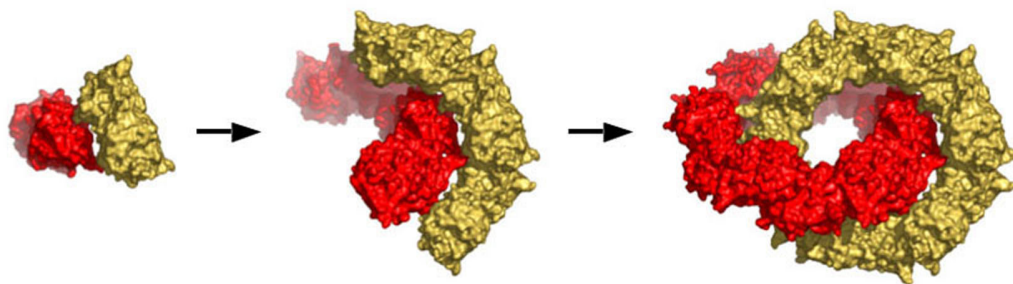


Figure 5 Proposed mechanism of assembly of the 2 -ring catenane structure.

Polar contacts between dimers (shown in red and gold), potentially occurring at any stage during single toroid formation, provide the basis for initiating the generation of a second topologically -linked ring leading to the overall 2-ring catenane structure. The figure was take from (Cao *et al.* 2005).

On determination of T matrix in TLS modelling

P.Afonine[§] and A.Urzhumtsev[€]

[§]*Lawrence Berkeley National Laboratory, One Cyclotron Road, BLDG 64R0121, Berkeley, CA 94720 USA.*

[€]*University Louis Pasteur, Strasbourg - IGBMC, 1 rue L.Fries, 67404 Illkirch - IBMC, 15 rue R.Descartes, 67084, Strasbourg.*

e-mail : PAfonine@lbl.gov, sacha@igbmc.u-strasbg.fr

1. Introduction

Uncertainty in an atomic position is characterised, in a harmonic approximation, by a quadratic function of the vector of atomic displacement. In mathematics, such a scalar function of a vector is called quadratic form. In a chosen coordinate system, each vector is presented by a vector-column $\mathbf{u}$ of its coordinates and a quadratic form is characterised by a quadratic symmetric matrix U . When moving from one coordinate system to another, a matrix of a quadratic form changes following the rule

$$U_{new} = Q^t U Q \quad (1.1)$$

where Q is the matrix of transition to the new basis and Q^t stands for transposed matrix Q . To obtain Q , one expresses the vectors of the new basis in the initial basis and takes corresponding coefficients as columns of Q .

By definition, matrices of quadratic forms describing atomic uncertainties are positively definite, or, in singular cases, positive semidefinite. We remind that a quadratic matrix U is positive definite if for any vector $\mathbf{u}$ the scalar product $(\mathbf{u}, U\mathbf{u}) > 0$; in the text below we shall note this as $U > 0$. For a positive semidefinite (alias non-negatively definite) matrix U such that $(\mathbf{u}, U\mathbf{u}) \geq 0$ for any $\mathbf{u}$ we shall use notation $U \geq 0$.

TLS modelling is an efficient way to treat uncertainties in atomic positions in X-ray crystallography considering an atomic group displacing as a rigid body (to simplify the presentation, in what follows we consider a model composed from a single rigid group). In such a model an uncertainty of each atom $m=1, \dots, n$ is described by a quadratic form whose matrix $U_{TLS,m}$ is a sum of three components (Schomaker & Trueblood, 1968)

$$U_{TLS,m} = T + A_m L A_m^t + (A_m S + S^t A_m^t) \quad (1.2)$$

Here T stands for the common component due to a translation (vibration) of the group as a rigid body, and two other terms,

$$U_{LS,m} = A_m L A_m^t + (A_m S + S^t A_m^t) \quad (1.3)$$

different for different atoms, describe uncertainties due to rigid body libration. Of course, the total uncertainty of atom m may have a contribution from other sources

$$U_m = U_{TLS,m} + U_{add,m} \quad (1.4)$$

that we do not discuss here. By the definition, U_m , $U_{TLS,m}$, T , $U_{LS,m}$, $U_{add,m}$ are positive semidefinite.

A knowledge of matrices T , L and S allows calculating $U_{TLS,m}$ for each atom of the group. The inverse problem of finding T , L and S arises when one analyses the collective vibration of molecular fragments given the set of individual U_m . This problem is commonly solved by least-squares fit

$$\sum_{m=1}^n \sum_{j,k=1,3} \left(U_{TLS,m}^{(jk)}(T, L, S) - U_m^{(jk)} \right)^2 \rightarrow \min_{t,l,s} \quad (1.5)$$

where variables are independent elements $t^{(j,k)}$, $l^{(j,k)}$, $s^{(j,k)}$ of T , L and S (20 elements per each rigid group; Schomaker & Trueblood, 1968). As a rule, refinement starts from zero values that means that initially all atomic uncertainties supposed to be individual with no common terms. When the model is composed from several rigid groups, (1.5) includes also a sum over the groups.

In fact this approach is not good since by minimising (1.5) we are looking for $U_{TLS,m}$ that are ‘in average equal to U_m ’ (sometimes ‘smaller’, sometimes ‘larger’) and the difference $U_m - U_{TLS,m}$ (commonly referenced as a local or residual B-factor) is not guaranteed to be positive definite as it must be.

Below we propose an algorithm to directly estimate the matrix T starting from a set of U_m and without involving a least-squares fit (1.5). This algorithm is accompanied by comments making it more understandable for a reader not very familiar with linear algebra. In what follows, capital italic (M) stand for matrices of quadratic forms and small italic greeks (μ) stand for their eigenvalues. Small bold ($\mathbf{u}$) stand for vector-columns composed from three vector coordinates in the chosen basis.

2. Determination of a common component for isotropic and anisotropic cases

2.1. Problem and definitions

Let U_m , $m=1, \dots, n$, be symmetric 3×3 matrices representing positive definite forms U_m in some previously chosen *orthonormal* basis common for all of them. We want to present all U_m as

$$U_m = U_{res,m} + T_{max} \quad (2.1)$$

where T_{max} is a positive definite matrix such that all residual $U_{res,m}$ are positive semidefinite and that for any positive semidefinite matrix subtracting it from the set of residual $U_{res,m}$ breaks this condition of non-negativity at least for one of them. In what follows we shall call such a matrix T_{max} ‘maximal’.

2.2. Isotropic models

In the case of an isotropic model, all U_m are proportional to the unit matrix,

$$U_m = \begin{pmatrix} u_m & 0 & 0 \\ 0 & u_m & 0 \\ 0 & 0 & u_m \end{pmatrix} = u_m \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = u_m I, \quad u_m > 0, \quad m=1, \dots, n. \quad (2.2)$$

Matrix T is also diagonal, and L and S are zero.

For isotropic models the solution T_{max} is unique and presented by the matrix $T_{max} = t_{max} I$ with the element

$$t_{\max} = \min_{m=1,\dots,n} \{u_m\} \quad (2.3)$$

everywhere at its diagonal. Obviously, if we are back from the purely algebraic problem formulated in 2.1 to the original crystallographic one, (2.2-2.3) is equivalent to a search for the minimal value of the isotropic B factor for the group of atoms.

2.3. Anisotropic models

When working with anisotropic atomic displacement, that is the case of TLS modelling, the matrices U_m are no more diagonal and the solution is no more unique (in fact, one may show that it is not unique even in the case of matrices in two-dimensional space) and imposing different extra conditions leads to different solution.

In our algorithm we replace a simultaneous search of the maximal matrix $T_{\max}$ for all matrices by a consecutive search for two matrices at a time. To do so, we find first the maximal matrix T_{1-2} , say, for U_1 and U_2 . We apply then the same procedure for T_{1-2} and U_3 getting T_{1-3} , etc., finishing with $T_{\max}=T_{1-n}$. Determination of the maximal matrix $T_{\max}$ for two matrices is done using the algorithm described below.

3. Determination of a maximal matrix for two positive definite matrices – general case

3.1. General comments

Fig. 1 schematizes the search algorithm for the maximal matrix $T_{\max}$ for two positive definite quadratic 3×3 matrices. More formally, let $M > 0$ and $N > 0$ be two positive definite matrices presenting in some Cartesian coordinate system two quadratic forms $\mathbf{M}$ and $\mathbf{N}$. We search for a matrix $T_{\max} > 0$ such that $M - T_{\max} \geq 0$, $N - T_{\max} \geq 0$, and for any other matrix $\tilde{T} \geq 0$ at least one of two conditions $M - T_{\max} - \tilde{T} \geq 0$, $N - T_{\max} - \tilde{T} \geq 0$ is broken. The text below illustrates and comments the scheme. Numbers in parentheses at the beginning of paragraph corresponds to the numbers in yellow rectangulars at the scheme.

Let $\tau_1 \geq \tau_2 \geq \tau_3 > 0$ stand for the eigenvalues of $T_{\max}$. As we mentioned above, an extra condition should be imposed to define which of possible maximal $T_{\max}$ we are looking for. Let's choose $T_{\max}$ such that its minimal eigenvalue is as large as possible, $\tau_3 \rightarrow \max$, and then from all possible matrices with this τ_3 to choose the one for which its second eigenvalue is also maximal, $\tau_2 \rightarrow \max$.

It is easy to show (see Appendix B, P1) that each $T_{\max} > 0$ may be presented as a sum

$$T_{\max} = T_3 + T_{\text{res}} \quad (3.1)$$

where $T_3 > 0$ has equal positive eigenvalues (and is diagonal in any orthonormal basis, Appendix B, P2) and $T_{\text{res}} \geq 0$ has at least one eigenvalue equal to 0. In its turn, T_{res} also can be presented as a sum

$$T_{\text{res}} = T_2 + T_1 \quad (3.2)$$

where $T_2 \geq 0$ has two equal positive and one zero eigenvalues, and $T_1 \geq 0$ has one positive and two zero eigenvalues (in a singular situation any of T_2 and T_1 may be absent). By this reason, we search for $T_{\max}$ presenting it as

$$T_{max} = T_3 + T_2 + T_1 \quad (3.3)$$

with the properties indicated above.

3.2. Initialisation and search for the maximal matrix T_3 with three equal eigenvalues

- (1) In what follows let $\mu_1 \geq \mu_2 \geq \mu_3 > 0$ and $\nu_1 \geq \nu_2 \geq \nu_3 > 0$ be eigenvalues of M and N , respectively, and $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ and $\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3$ be the corresponding normalised eigenvectors. Let's suppose also that the minimal eigenvalues of M are larger than those of N , that means $\mu_3 > \nu_3$; or if $\mu_3 = \nu_3$ then $\mu_2 > \nu_2$; finally if $\mu_3 = \nu_3$ and $\mu_2 = \nu_2$ then $\mu_1 > \nu_1$. Otherwise we simply exchange the name of the matrices.
- (2) If $\mu_3 \geq \nu_1$ then the situation is singular, we directly get the final answer $T_{max} = N$ (see Appendix B, P3) and interrupt the procedure.
- (3) In a non singular situation, $\mu_3 < \nu_1$,

$$T_3 = \begin{pmatrix} \nu_3 & 0 & 0 \\ 0 & \nu_3 & 0 \\ 0 & 0 & \nu_3 \end{pmatrix} \quad (3.4)$$

(see Appendix B, P4). Residual matrices are

$$\tilde{M} = M - T_3, \quad \tilde{N} = N - T_3 \quad (3.5)$$

respectively. Note that their eigenvectors are the same as for M and N and that for the eigenvector $\mathbf{g}_3$ corresponding to ν_3 we have a condition

$$\tilde{N}\mathbf{g}_3 = (N - T_3)\mathbf{g}_3 = \nu_3\mathbf{g}_3 - \nu_3\mathbf{g}_3 = 0 \quad (3.6)$$

Eigenvalues of the new matrices $\tilde{M}$ and $\tilde{N}$ are

$$\tilde{\mu}_1 = \mu_1 - \nu_3 \geq \tilde{\mu}_2 = \mu_2 - \nu_3 \geq \tilde{\mu}_3 = \mu_3 - \nu_3 \geq 0 \text{ and } \tilde{\nu}_1 = \nu_1 - \nu_3 \geq \tilde{\nu}_2 = \nu_2 - \nu_3 \geq \tilde{\nu}_3 = 0, \quad (3.7)$$

It follows from (3.6) that $T_{res}\mathbf{g}_3 = 0$ (Appendix B, P5) meaning that $\mathbf{g}_3$ defines a one-dimensional subspace in which no more common displacement described by M and N may be found.

3.3. Additional maximal degenerate matrix

The matrix T_{res} that should be added to T_3 has at least one of its eigenvalues equal to zero and is no more positive but only non-negative definite. We want to exclude from further analysis the direction $\mathbf{g}_3$, following the notes of step (3).

In this Section we consider only a general situation when $\tilde{\mu}_1 \geq \tilde{\mu}_2 \geq \tilde{\mu}_3 > 0$ and $\tilde{\nu}_1 \geq \tilde{\nu}_2 > \tilde{\nu}_3 = 0$. All other situations, considered as singular, are studied separately in Section 4.

- (4) We change the basis in which the matrices are given. The vectors of the new basis are eigenvectors of N (and those of $\tilde{N}$), $\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3$. This basis is orthogonal,

by the properties of eigenvectors, and normal, by the initial choice. The residual matrices of the quadratic forms in the new basis are given by

$$M_G = G^t \tilde{M} G, N_G = G^t \tilde{N} G \quad (3.8)$$

In the new basis, N_G is diagonal,

$$N_G = \begin{pmatrix} \tilde{v}_1 & 0 & 0 \\ 0 & \tilde{v}_2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (3.9)$$

The latter presentation means that in this basis any matrix $T \geq 0$, such that $N_G - T \geq 0$, has zero in the right column and the bottom line (see Appendix B, P6-P7). In particular, this is true for T_{res} .

3.4. Additional single degenerate maximal matrix T_2

The fact that $T_{res} \geq 0$ has $\mathbf{g}_3$ as its eigenvector corresponding to $\tilde{\tau}_3 = 0$ means that in the basis of its own eigenvectors $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3 = \mathbf{g}_3$ this matrix is presented as a diagonal one

$$T_{res} = \begin{pmatrix} \tilde{\tau}_1 & 0 & 0 \\ 0 & \tilde{\tau}_2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (3.10)$$

with $\tilde{\tau}_1 \geq \tilde{\tau}_2 \geq 0$. An analysis similar to the previous one leads to the conclusion that in this basis

$$T_2 = \begin{pmatrix} \tilde{\tau}_2 & 0 & 0 \\ 0 & \tilde{\tau}_2 & 0 \\ 0 & 0 & 0 \end{pmatrix}, T_1 = \begin{pmatrix} \tilde{\tau}_1 - \tilde{\tau}_2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (3.11)$$

One may note that T_2 has exactly the same form in any orthonormal basis with the third vector equal to $\mathbf{g}_3$, in particular, in the basis of vector $\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3$ (demonstration similar to P2 in Appendix B). Therefore we need to search for the matrix in the form

$$T_2 = \begin{pmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad (3.12)$$

where $a \geq 0$ is as large as possible such that $M_G - T_2 \geq 0$ and $N_G - T_2 \geq 0$.

(5a) Condition $N_G - T_2 \geq 0$ gives, similarly to the previous analysis, $t \leq \tilde{v}_2$. Condition $M_G - T_2 \geq 0$ in a non-singular case may be expressed using the function $\alpha(M_G)$ defined in Appendix A. Summarising,

$$a = \min \{ \alpha(\tilde{M}_G), \tilde{v}_2 \} \quad (3.13)$$

After matrix T_2 is defined, new residual matrices are calculated as

$$\tilde{M}_G = M_G - T_2, \tilde{N}_G = N_G - T_2 \quad (3.14)$$

3.5. Additional double degenerate maximal matrix T_1

(6a) In (3.13) we consider as a general situation the case when $a = \alpha(\tilde{M}_G)$. This means that each of two matrices $\tilde{M}_G$ and $\tilde{N}_G$ has at least one zero eigenvalue and the corresponding eigenvectors $\tilde{\mathbf{e}}_3$ ($\tilde{M}_G \tilde{\mathbf{e}}_3 = 0$) and $\mathbf{g}_3$ ($\tilde{N}_G \mathbf{g}_3 = 0$) are non collinear (if they were collinear, then $\tilde{\mathbf{e}}_3 = \lambda \mathbf{g}_3$, $\tilde{M}_G \mathbf{g}_3 = 0$ that would mean $(M_G - T_2) \mathbf{g}_3 = M_G \mathbf{g}_3 - T_2 \mathbf{g}_3 = M_G \mathbf{g}_3 = 0$ contradicting the condition $\tilde{\mu}_3 > 0$). The maximal matrix T_1 has both $\tilde{\mathbf{e}}_3$, $\mathbf{g}_3$ and any their linear combination as its eigenvectors corresponding to zero eigenvalue (see Appendix B, P8).

We change once more the basis. Similarly to step (4), Section 3.3, the vectors of the new basis are defined as

$$\mathbf{p}_3 = \mathbf{g}_3, \mathbf{p}_2 = [\tilde{\mathbf{e}}_3 - (\tilde{\mathbf{e}}_3, \mathbf{p}_3) \mathbf{p}_3] \cdot \|\tilde{\mathbf{e}}_3 - (\tilde{\mathbf{e}}_3, \mathbf{p}_3) \mathbf{p}_3\|^{-1}, \mathbf{p}_1 = \mathbf{p}_2 \wedge \mathbf{p}_3 \quad (3.15)$$

Here $\mathbf{u} \wedge \mathbf{v}$ stands a vector product of $\mathbf{u}$ and $\mathbf{v}$ and $\|\mathbf{u}\|$ stands for the module of the vector $\mathbf{u}$, which in the orthonormal basis can be calculated by the Pythagor theorem. This new basis is orthonormal. The property that $T_1 \mathbf{p}_3 = T_1 \mathbf{p}_2 = T_2 \mathbf{p}_3 = 0$ proves that indeed $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$ is the basis of eigenvectors of T_{res} considered above in (3.11).

(7a) We build the matrix P of transition to the new basis. This matrix is composed in columns from the coordinates of the vectors $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ in the previous basis $\{\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3\}$

$$M_P = P^t \tilde{M}_G P, N_P = P^t \tilde{N}_G P \quad (3.16)$$

In the new basis, T_1 is searched in form

$$T_1 = \begin{pmatrix} b & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (3.17)$$

(8) The value of b for the maximal T_1 to be subtracted from M_P and N_P can be calculated using the function β defined in Appendix A:

$$b = \min\{\beta(M_P), \beta(N_P)\} \quad (3.18)$$

4. Maximal matrix for two positive semidefinite matrices: singular cases

The analysis in Section 3 does not consider singular situations. These cases are studied below completing the algorithm

(5b-7b) After step (3) $\tilde{N}$ is a double degenerate matrix, $\tilde{\nu}_2 = \tilde{\nu}_3 = 0$. Then T_2 is a zero matrix and T_1 may be searched for as (3.17-3.18) directly in the basis $\{\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3\}$ of the eigenvectors of N .

(5c-6b-7b) After step (3) $\tilde{\nu}_2 > 0$ and $\tilde{M}$ is a degenerate matrix, $\tilde{\mu}_3 = 0$, with the same eigenvector $\mathbf{g}_3$ ($\mathbf{e}_3 = \mathbf{g}_3$). Both M_G and N_G have 0 in the right column and the bottom line, and the problem is reduced to a two-dimensional. Condition $\tilde{\mu}_3 = 0$ means that

$\mu_3 = \nu_3$ and then by the initial choice, $\tilde{\mu}_2 \geq \tilde{\nu}_2 > 0$. Then in (3.12-3.13) the value of a for T_2 becomes

$$a = \tilde{\nu}_2 \quad (4.1)$$

and we can search for T_1 as (3.17-3.18) directly in the basis $\{\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3\}$.

(5d-7a) After step (3) $\tilde{\nu}_2 > 0$ and $\tilde{M}$ is a degenerate matrix, $\tilde{\mu}_3 = 0$, with the eigenvector $\mathbf{e}_3 \neq \mathbf{g}_3$. This condition may be formally expressed, for example, by $(\mathbf{g}_3, \tilde{M}\mathbf{g}_3) > 0$. Condition $\tilde{\mu}_3 = 0$ means that $\mu_3 = \nu_3$ and then by the initial choice, $\tilde{\mu}_2 \geq \tilde{\nu}_2 > 0$. The situation is absolutely similar to the case (6a) except that here the zero-value eigenvector is computed for the initial matrix M (the same eigenvectors as for $\tilde{M}$) and not for the updated one, $\tilde{M}_G$, as in (6a). Therefore, we change the basis to

$$\mathbf{p}_3 = \mathbf{g}_3, \mathbf{p}_2 = [\mathbf{e}_3 - (\mathbf{e}_3, \mathbf{p}_3)\mathbf{p}_3] \cdot \|\mathbf{e}_3 - (\mathbf{e}_3, \mathbf{p}_3)\mathbf{p}_3\|^{-1}, \mathbf{p}_1 = \mathbf{p}_2 \wedge \mathbf{p}_3 \quad (4.2)$$

define T_2 as zero matrix and search for T_1 as (3.17-3.18) directly in the new basis.

(6b) If in (3.12-3.13) the value of $a = \tilde{\nu}_2$, then T_1 may be searched for as (3.17-3.18) directly in the basis $\{\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3\}$ of the eigenvectors of N . Also immediately

$$\beta(N_P) = \tilde{\nu}_1 - \tilde{\nu}_2. \quad (4.3)$$

5. Algorithm termination and concluding remarks

(9) Finally, collecting T_1 , T_2 and T_3 together in the initial coordinate system gives

$$T_{max} = T_3 + G(T_2 + PT_1P^t)G^t \quad (5.1)$$

where in some cases either G or P may be unitary matrices and either T_2 or T_1 be zero matrices.

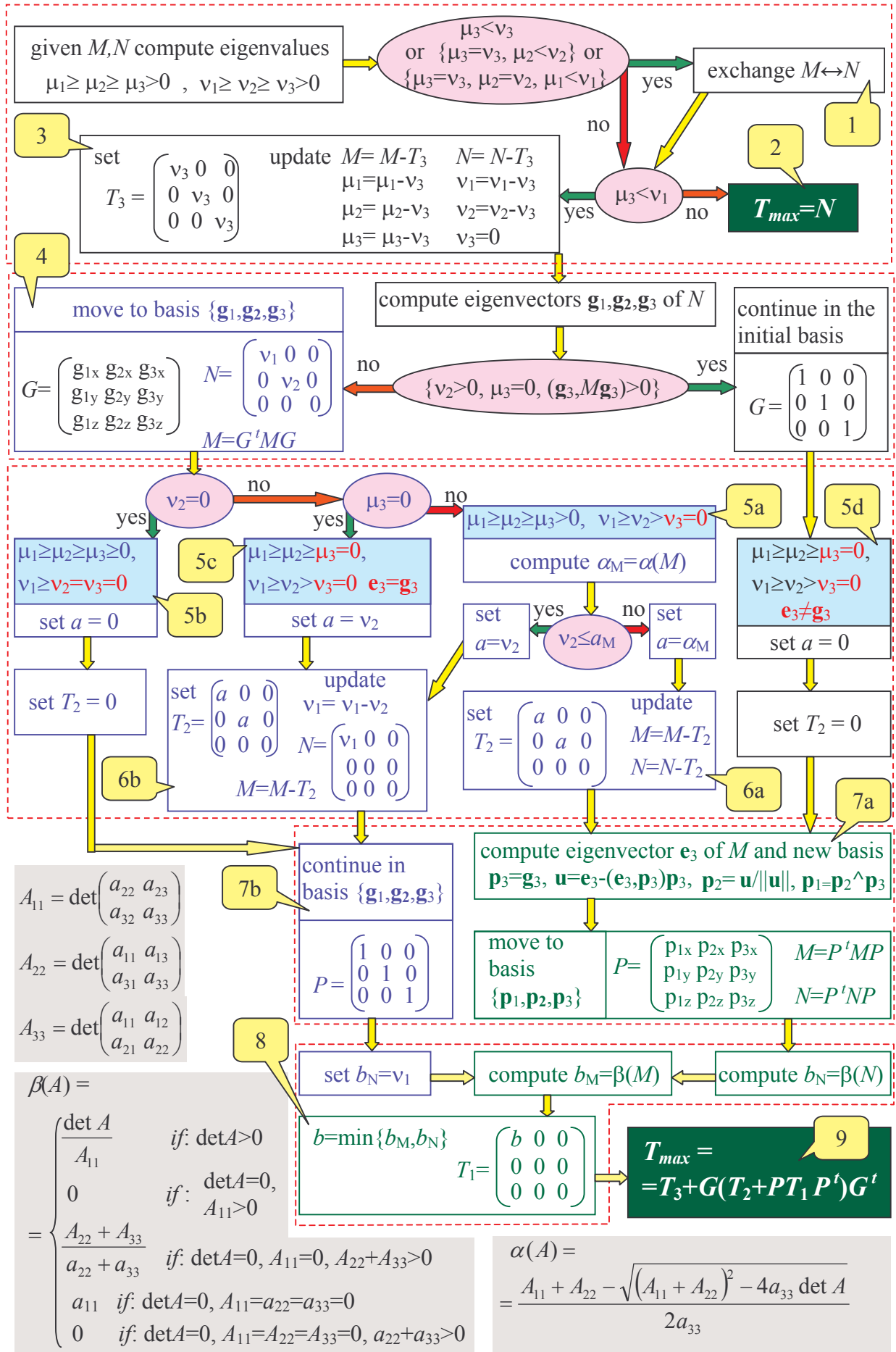
This algorithm was realised in the frame of the program complex *PHENIX* (Adams *et al.*, 2002; www.phenix-online.org) and extensively tested.

The authors thank E.Shnol for numerous advices without which this work would never be done. The authors thank P.Adams and D.Moras for their support. PA gratefully acknowledges the financial support of NIH/NIGMS under grant number P01GM063210 and support by the US Department of Energy under Contract No. DE AC02-05CH11231.

References

- 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. & Terwilliger, T.C. (2002). "PHENIX: building new software for automated crystallographic structure determination". *Acta Cryst.*, **D58**, 1948-1954.
- Schomaker, V. & Trueblood, K.N. (1968). "On the rigid-body motion of molecules in crystals". *Acta Cryst.* **B24**, 63-76.

Fig. 1. Scheme of the determination of the maximal matrix T_{max} for two positive definite 3×3 matrices M and N . Different colours (black, blue, green) of rectangulars correspond to different bases. Ping ellipses indicate the conditions, and light blue highlights the condition with which corresponding rectangular can be reached. Grey rectangular show the formula to calculate the functions α and β . Red broken contours indicate the principals steps of the procedure: initialisation, search for T_3 , change of the basis, search for T_2 , second change of the basis, search for T_1 and final computing of T_{max} . Numbers in yellow rectangulars correspond to the numbers of the steps in the text.



Appendix A. Definition of $\alpha(A)$ and $\beta(A)$

For a given matrix $A \geq 0$ of dimension 3×3 with the elements a_{jk} we define two functions $\alpha(A)$ and $\beta(A)$ of its elements.

We define $\alpha(A)$ as a value such that $A - T(\alpha) \geq 0$ for $0 \leq \alpha \leq \alpha(A)$, where $T(\alpha) = \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 0 \end{pmatrix}$,

and for $\alpha > \alpha(A)$ this condition breaks. Let A_{jj} be determinants of its principal minors:

$$A_{11} = \det \begin{pmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{pmatrix}, \quad A_{22} = \det \begin{pmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{pmatrix}, \quad A_{33} = \det \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}. \quad (\text{A.1})$$

Then for $A > 0$ this maximal value of α is given by

$$\alpha(A) = \frac{A_{11} + A_{22} - \sqrt{(A_{11} + A_{22})^2 - 4a_{33}\det A}}{2a_{33}} > 0 \quad (\text{A.2})$$

(for a proof see Appendix B, P9).

We define $\beta(A)$ as a value such that $A - T(\beta) \geq 0$ for $0 \leq \beta \leq \beta(A)$, where $T(\beta) = \begin{pmatrix} \beta & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$,

and for $\beta > \beta(A)$ this condition breaks. This value is given by

$$\text{a) } \beta(A) = \frac{\det A}{A_{11}} > 0, \text{ if } \det A > 0 \quad (\text{A.3})$$

$$\text{b) } \beta(A) = 0, \text{ if } \det A = 0 \text{ and } A_{11} > 0 \quad (\text{A.4})$$

$$\text{c) } \beta(A) = \frac{A_{22} + A_{33}}{a_{22} + a_{33}} > 0, \text{ if } \det A = 0 \text{ and } A_{11} = 0 \text{ and } A_{22} + A_{33} > 0 \quad (\text{A.5})$$

$$\text{d) } \beta(A) = a_{11} \geq 0, \text{ if } \det A = 0 \text{ and } A_{11} = a_{22} = a_{33} = 0 \quad (\text{A.6})$$

$$\text{e) } \beta(A) = 0, \text{ if } \det A = 0 \text{ and } A_{11} = A_{22} = A_{33} = 0 \text{ and } a_{22} + a_{33} > 0 \quad (\text{A.7})$$

(for a proof see Appendix B, P10).

Appendix B. Proofs of some propositions

P1. Presentation of T as $T = T_3 + T_2 + T_1$ (Section 3.1)

In the (orthonormal) basis of its own eigenvectors the matrix T is diagonal,

$$T = \begin{pmatrix} \tau_1 & 0 & 0 \\ 0 & \tau_2 & 0 \\ 0 & 0 & \tau_3 \end{pmatrix} \quad (\text{B.1})$$

and can be decomposed into a sum

$$T = \begin{pmatrix} \tau_3 & 0 & 0 \\ 0 & \tau_3 & 0 \\ 0 & 0 & \tau_3 \end{pmatrix} + \begin{pmatrix} \tau_2 - \tau_3 & 0 & 0 \\ 0 & \tau_2 - \tau_3 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} \tau_1 - \tau_2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (\text{B.2})$$

of one positive definite and two positive semidefined matrices, with 1 and 2 zero eigenvalues, respectively.

P2. Diagonal presentation of T_3 (Section 3.1)

In the *orthonormal* basis of its own eigenvectors the matrix T_3 is

$$T_3 = \begin{pmatrix} \tau_3 & 0 & 0 \\ 0 & \tau_3 & 0 \\ 0 & 0 & \tau_3 \end{pmatrix} = \tau_3 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \tau_3 I \quad (\text{B.3})$$

In any other *orthonormal* basis obtained with the transition matrix Q (a rotation or rotoinversion matrix) the matrix of T_3 is

$$T_{3,Q} = Q^T T_3 Q = Q^{-1} T_3 Q = Q^{-1} \tau_3 I Q = \tau_3 Q^{-1} Q = \tau_3 I \quad (\text{B.4})$$

P3. $T_{\max} = N$ if $\mu_3 \geq \nu_1$ (Section 3.2)

Let $\mu_1 \geq \mu_2 \geq \mu_3 \geq \nu_1 \geq \nu_2 \geq \nu_3 > 0$. With the notation used, for any vector $\mathbf{u}$

$$(\mathbf{u}, M\mathbf{u}) \geq \mu_3(\mathbf{u}, \mathbf{u}), \quad \nu_1(\mathbf{u}, \mathbf{u}) \geq (\mathbf{u}, N\mathbf{u}) \geq \nu_3(\mathbf{u}, \mathbf{u}). \quad (\text{B.5})$$

Therefore,

$$(\mathbf{u}, (M-N)\mathbf{u}) \geq (\mathbf{u}, M\mathbf{u}) - (\mathbf{u}, N\mathbf{u}) \geq (\mu_3 - \nu_1)(\mathbf{u}, \mathbf{u}) \geq 0. \quad (\text{B.6})$$

In other words, $M-N \geq 0$. It is clear that nothing more can be subtracted from the second matrix, that proves the proposition.

P4. $T_3 = \nu_3 I$ (Section 3.2).

$\mu_1 \geq \mu_2 \geq \mu_3 > 0$, $\nu_1 \geq \nu_2 \geq \nu_3 > 0$, $\mu_3 \geq \nu_3$. With the notation used, for any vector $\mathbf{u}$

$$(\mathbf{u}, (M - \nu_3 I)\mathbf{u}) = (\mathbf{u}, M\mathbf{u}) - \nu_3(\mathbf{u}, \mathbf{u}) \geq (\mu_3 - \nu_3)(\mathbf{u}, \mathbf{u}) \geq 0, \quad (\text{B.7})$$

$$(\mathbf{u}, (N - \nu_3 I)\mathbf{u}) = (\mathbf{u}, N\mathbf{u}) - \nu_3(\mathbf{u}, \mathbf{u}) \geq (\nu_3 - \nu_3)(\mathbf{u}, \mathbf{u}) = 0. \quad (\text{B.8})$$

At the same time, for any $\nu > \nu_3$ the matrix $N - \nu I$ is not positive semidefinite. For example, if $\mathbf{g}_3$ is eigenvector of N corresponding to ν_3 then

$$(\mathbf{g}_3, (N - \nu I)\mathbf{g}_3) = (\mathbf{g}_3, N\mathbf{g}_3 - \nu\mathbf{g}_3) = (\nu_3 - \nu)(\mathbf{g}_3, \mathbf{g}_3) < 0. \quad (\text{B.9})$$

P5. $T_{\text{res}}\mathbf{g}_3=0$ (Section 3.2).

If $N \geq 0$, $T \geq 0$, $N-T \geq 0$, and $N\mathbf{u}=0$, then

$$0 \leq (\mathbf{u}, (N-T)\mathbf{u}) = (\mathbf{u}, -T\mathbf{u}) = -(\mathbf{u}, T\mathbf{u}) \leq 0 \quad (\text{B.10})$$

that is possible only if $(\mathbf{u}, T\mathbf{u})=0$. Since T is positive semidefinite with eigenvalues $\tau_1 \geq \tau_2 \geq \tau_3 \geq 0$, presenting $\mathbf{u}$ by its coordinates u_1, u_2, u_3 in the basis of eigenvectors of T we obtain that $(\mathbf{u}, T\mathbf{u}) = \tau_1 u_1^2 + \tau_2 u_2^2 + \tau_3 u_3^2 = 0$ that proves $T\mathbf{u}=0$.

P6. $a_{j3} = a_{3j} = 0, j=1,2,3$ if $A \geq 0$ with $a_{33} = 0$ (Section 3.3).

General theory tells that for a matrix $A \geq 0$ the determinants of the principal minors of A are non negative. For a matrix of dimension 3×3 with the elements $a_{jk} = a_{kj}$ this means that

$$A_{11} = \det \begin{pmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{pmatrix} \geq 0, A_{22} = \det \begin{pmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{pmatrix} \geq 0, A_{33} = \det \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \geq 0, \quad (\text{B.11})$$

$$a_{11} \geq 0, a_{22} \geq 0, a_{33} \geq 0$$

If $a_{33}=0$ then $A_{11} \geq 0$ and $A_{22} \geq 0$ give immediately $a_{13}=a_{31}=a_{23}=a_{32}=0$. Similar results may be obtained for $a_{11}=0$ or $a_{22}=0$.

P7. $T \geq 0$ has 0 in the right column and bottom line if $N \geq 0$ has (Section 3.3).

Let n_{jk} and t_{jk} be the elements of $N \geq 0$ and $T \geq 0$, respectively. If $n_{33}=0$ the condition $N-T \geq 0$ leads to $n_{33}-t_{33} \geq 0$ (see P6) that with $t_{33} \geq 0$ gives $t_{33}=0$. Then by (P6)

$$t_{13}=t_{31}=t_{23}=t_{32}=0. \quad (\text{B.12})$$

P8. $\tilde{\mathbf{e}}_3$ and $\mathbf{g}_3$ are two non collinear eigenvectors of T_1 (Section 3.5).

Let $M \geq 0$, $N \geq 0$, $T \geq 0$, $M-T \geq 0$, $N-T \geq 0$ and $M\mathbf{u} = 0$, $N\mathbf{v} = 0$. Then

$$\begin{aligned} 0 &\leq (\mathbf{u}, (M-T)\mathbf{u}) = (\mathbf{u}, M\mathbf{u}) - (\mathbf{u}, T\mathbf{u}) = -(\mathbf{u}, T\mathbf{u}) \leq 0 \\ 0 &\leq (\mathbf{v}, (N-T)\mathbf{v}) = (\mathbf{v}, N\mathbf{v}) - (\mathbf{v}, T\mathbf{v}) = -(\mathbf{v}, T\mathbf{v}) \leq 0 \end{aligned} \quad (\text{B.13})$$

that is possible only if $(\mathbf{u}, T\mathbf{u}) = (\mathbf{v}, T\mathbf{v}) = 0$ and then $T\mathbf{u} = T\mathbf{v} = \mathbf{0}$ (as in P5). That means also that for any linear combination of $\mathbf{u}$ and $\mathbf{v}$ we have $T(\alpha\mathbf{u} + \beta\mathbf{v}) = \mathbf{0}$.

P9. Demonstration of the formula for $\alpha(A)$ (Appendix A).

Let $a_{jk} = a_{kj}$, $j, k=1, 2, 3$, be the elements of the matrix $A \succ 0$. We define the matrix A_α

$$A_\alpha = A - \alpha \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} = A - \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (\text{B.14})$$

For $\alpha \neq 0$ by definition $A_\alpha \succ 0$. When $\alpha \gg 1$, two of diagonal elements of A_α are negative that is impossible for positive definite matrix. The question is to find minimal positive value of α such that the condition $A_\alpha \succ 0$ breaks. This is equivalent to the condition when one of eigenvalues of A_α becomes equal to zero. A product of eigenvalues is equal to the determinant of A_α , i.e.

$$\det A_\alpha = \alpha^2 a_{33} - \alpha [A_{11} + A_{22}] + \det A = P_2(\alpha) \quad (\text{B.15})$$

where A_{kk} are corresponding principal minors of A . By the properties of positive definite matrix, $a_{33} > 0$, $A_{11} + A_{22} > 0$, $\det A > 0$. The quadratic equation $P_2(\alpha) = 0$ has two real roots

$$\frac{A_{11} + A_{22} \pm \sqrt{(A_{11} + A_{22})^2 - 4a_{33}\det A}}{2a_{33}} > 0 \quad (\text{B.16})$$

and that taken with sign '-' is $\alpha(A)$. The presence of solutions, i.e. the fact that

$$(A_{11} + A_{22})^2 - 4a_{33}\det A \geq 0 \quad (\text{B.17})$$

may be justified, for example, by the same argument that for large α the matrix A_α is not positive definite (the condition $\det A_\alpha > 0$ simply means that both eigenvalues are negative). Therefore, due to continuity of the problem there should be values of α when each of the eigenvalues becomes negative (in a singular situation, both at the same time).

It may be noted that, as it has been shown in the text, if $A \geq 0$ with one zero and two positive eigenvalues then $\alpha(A)$ is equal to the minimal positive eigenvalue; finally for $A \geq 0$ with two zero eigenvalues $\alpha(A) = 0$.

P10. Demonstration of the formula for $\beta(A)$ (Appendix A).

Let $a_{jk}=a_{kj}$, $j,k=1,2,3$, be the elements of the matrix $A \geq 0$. We define the matrix A_β

$$A_\beta = A - \beta \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = A - \begin{pmatrix} \beta & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (B.18)$$

For $\beta=0$ by definition $A_\beta \geq 0$. When $\beta > 1$, one of diagonal elements of A_β is negative that is impossible for positive semidefinite matrix. The question is to find minimal non-negative value of β such that the condition $A_\beta \geq 0$ breaks.

When $\det A > 0$, this corresponds to the (positive) root of

$$\det A_\beta = -\beta A_{11} + \det A = P_1(\beta) = 0 \quad (B.19)$$

giving (A.3).

When $\det A = 0$ (one zero eigenvalue exists already) a search for β with $\det A_\beta = 0$ cannot be applied any more and we shall use the characteristic equation for the eigenvalues of A_β

$$\begin{aligned} P_3(\lambda, \beta) &= \\ &= -\lambda^3 + \lambda^2(a_{11}+a_{22}+a_{33}-\beta) - \lambda[A_{11}+A_{22}+A_{33}-\beta(a_{22}+a_{33})] + (\det A - \beta A_{11}) = \\ &= -\lambda^3 + \lambda^2(a_{11}+a_{22}+a_{33}-\beta) - \lambda[A_{11}+A_{22}+A_{33}-\beta(a_{22}+a_{33})] - \beta A_{11} = 0 \end{aligned} \quad (B.20)$$

If $A_{11} > 0$, for any $\beta > 0$ we have $P_3(0, \beta) < 0$ that necessary means an existence of a negative root of the characteristic equation, thus $\beta(A) = 0$ as (A.4).

When $\det A = 0$ and $A_{11} = 0$, the characteristic polynomial becomes

$$\begin{aligned} P_3(\lambda, \beta) &= -\lambda^3 + \lambda^2(a_{11}+a_{22}+a_{33}-\beta) - \lambda[A_{22}+A_{33}-\beta(a_{22}+a_{33})] = \\ &= -\lambda \{ \lambda^2 - \lambda(a_{11}+a_{22}+a_{33}-\beta) + [A_{22}+A_{33}-\beta(a_{22}+a_{33})] \} = -\lambda Q_2(\lambda, \beta) \end{aligned} \quad (B.21)$$

Here

$$Q_2(0, \beta) = A_{22}+A_{33}-\beta(a_{22}+a_{33}), \quad Q_2'(0, \beta) = -a_{11}-a_{22}-a_{33} + \beta \quad (B.22)$$

$$Q_2(0, 0) = A_{22}+A_{33} \geq 0, \quad Q_2'(0, 0) = -a_{11}-a_{22}-a_{33} < 0 \quad (B.23)$$

($Q_2'(0, 0) = 0$ only for zero matrix), that confirms non-negativity of A eigenvalues. If $Q_2(\lambda, \beta) = 0$ has two positive roots for $\beta = 0$ (corresponding condition $A_{22}+A_{33} > 0$), then the smallest of them becomes equal to 0 for

$$\beta(A) = \frac{A_{22} + A_{33}}{a_{22} + a_{33}} \quad (B.24)$$

(here $a_{22}+a_{33} > 0$, otherwise $a_{22}=a_{33}=0$ would give $a_{k2}=a_{2k}=a_{k3}=a_{3k}=0$ by (P6), and $A_{22}=A_{33}=0$). If A has a single positive root and two zeros (the corresponding condition $\det A = A_{11}=A_{22}=A_{33}=0$), then

$$P_3(\lambda, \beta) = -\lambda Q_2(\lambda, \beta) = -\lambda \{ \lambda^2 - \lambda(a_{11}+a_{22}+a_{33}-\beta) - \beta(a_{22}+a_{33}) \} \quad (B.25)$$

If $a_{22}+a_{33} > 0$ then $Q_2(0, \beta) < 0$ for any $\beta > 0$ indicating a negative root that proves (A.6). Finally, $\det A = A_{11}=0$ and $a_{22}=a_{33}=0$ gives (A.7) (as noted above, condition $A_{22}=A_{33}=0$ follows from $a_{22}=a_{33}=0$).