IUCr Computing Commission

Commission on Crystallographic Computing

International Union of Crystallography
http://www.iucr.org/iucr-top/comm/ccom/

Newsletter No. 2, July 2003
http://www.iucr.org/iucr-top/comm/ccom/newsletters/

Table of Contents

(This Issue's Editor: Lachlan Cranswick)

(Editor's warning – unless you want to kill 57 pages worth of forest – DO NOT press the "print" button. For hardcopies – you may like to only print out the articles of personal interest.)

CompComm Chairman's Message

Before you is the second release of a number of very interesting contributions to our newsletter brought together by our editor Lachlan Cranswick. Of particular interest for me is the article on the 'Beevers-Lipson' strips. Our group in Utrecht owns two of those boxes that were already shelved for display in our 'laboratory museum' by the time I entered as a graduate student in the mid-sixties. By that time those strips were already superseded by three generations of 'electronic' computers. It might be good idea to have an article of that type in future releases of the newsletter as well.

Preparations for the Florence 2005 meeting are underway. There are many proposals for computing related sessions under consideration. Prior to this meeting a crystallographic computing school is being planned. The site will be an old monastery close to the old town Siena, in the middle of the Chianti region. The emphasis of the school will be on current programming techniques and algorithms and should be of particular interest for young scientists (both from the macro and small-molecule background) interested in developing the next generation of crystallographic software.

Ton Spek, Chairman or the IUCr Computing Commission, (a.l.spek@chem.uu.nl)

From the Editor of Newsletter No. 2

On the following pages is the second edition of the IUCr Computing Commission newsletter. In the age of the time limited, thanks must go to the authors who have contributed articles. Authors in the "programming" section of the newsletter of the have been most generous in responding to editorial requests to include samples of crystallographic source code where possible, such that crystallographers and crystallographic programmers can have a better chance of appreciating that which is going on "under the hood".

An overall aim for future editions (as well as hopefully having "theme" issues edited by other guest editors), is for software authors to not only submit articles announcing new features or summarizing their software, but also elaborate on the algorithms with examples of source code, that made such updates feasible.

As knowing the origins of scientific techniques can be interesting and informative, articles of historical interest showing the development of crystallographic computing are also encouraged. This edition contains an article by Bob Gould explaining the use and history of 'Beevers-Lipson' strips. Unless the "Furies and Muses of the time limited" intervene, in the next issue it is hoped to have an invited article on one of the first uses of electronic computational machines; that being the use of Hollerith punched cards as Beevers-Lipson strips via a public utility company's billing system (in the mid to late 1940's). Other historical articles of a similar vein are most welcome.

Lachlan Cranswick (Lachlan.cranswick@nrc.gc.ca)

The IUCr Commission on Crystallographic Computing - Triennium 2003-2005

Chairman: Prof. Dr. Anthony L. Spek
Director of National Single Crystal Service Facility,
Utrecht University,
H.R. Kruytgebouw, N-801,
Padualaan 8, 3584 CH Utrecht,
the Netherlands.
Tel: +31-30-2532538
Fax: +31-30-2533940
E-mail: a.l.spek@chem.uu.nl
WWW: http://www.cryst.chem.uu.nl/spea.html

Professor I. David Brown
Brockhouse Institute for Materials Research,
McMaster University,
Hamilton, Ontario, Canada
Tel: 1-(905)-525-9140 ext 24710
Fax: 1-(905)-521-2773
E-mail: idbrown@mcmaster.ca
WWW: http://www.physics.mcmaster.ca/people/faculty/Brown_ID.html

Lachlan M. D. Cranswick
Neutron Program for Materials Research (NPMR),
National Research Council (NRC),
Building 459, Station 18, Chalk River Laboratories,
Chalk River, Ontario, Canada, K0J 1J0
Tel: (613) 584-8811 ext: 3719
Fax: (613) 584-4040
E-mail: lachlan.cranswick@nrc.gc.ca
WWW: http://lachlan.bluehaze.com.au/

Dr Vincent Favre-Nicolin
CEA Grenoble
DRFMC/SP2M/Nano-structures et Rayonnement Synchrotron
17, rue des Martyrs 38054 Grenoble Cedex 9
38054 Grenoble Cedex 9 – France
Tel: (+33) 4 38 78 95 40
Fax: (+33) 4 38 78 51 97
E-mail: vincefn@users.sourceforge.net
WWW: http://objcryst.sourceforge.net/

Dr Ralf Grosse-Kunstleve
Lawrence Berkeley Lab
1 Cyclotron Road,
BLDG 4R0230,
Berkeley, CA 94720-8235, USA.
Tel: 510-486-5713
Fax: 510-486-5909
E-mail: rwgk@yahoo.com
WWW: http://cci.lbl.gov/

Prof Alessandro Gualtieri
Università di Modena e Reggio Emilia,
Dipartimento di Scienze della Terra,
Via S.Eufemia, 19,
41100 Modena, Italy
Tel: +39-059-2055810
Fax: +39-059-2055887
E-mail: alex@unimore.it
WWW: http://www.terra.unimo.it/mineralogia/gualtieri.html

Prof Ethan A Merritt
Department of Biological Structure
University of Washington
Box 357420, HSB G-514
Seattle, Washington, USA
Tel: 206 543 1861
Fax: 206 543 1524
E-mail: merritt@u.washington.edu
WWW: http://www.bmsc.washington.edu/people/merritt/

Dr. Simon Parsons
School of Chemistry
Joseph Black Building,
West Mains Road,
Edinburgh, Scotland EH9 3JJ, UK
Tel: +44 131 650 5804
Fax: +44 131 650 4743
E-mail: s.parsons@ed.ac.uk
WWW: http://www.chem.ed.ac.uk/staff/parsons.html

Dr. Bev Vincent
RigakuMSC
9009 New Trails Dr,
The Woodlands, Texas 77381-5209, USA
Tel: 281-363-1033
Fax: 281-364-3628
E-mail: brv@RigakuMSC.com
WWW: http://www.rigakumsc.com/

Consultants

Dr David Watkin
Chemical Crystallography,
Oxford University,
9 Parks Road,
Oxford, OX1 3PD, UK.
Tel: +44 (0) 1865 272600
Fax: +44 (0) 1865 272699
E-mail: david.watkin@chemistry.oxford.ac.uk
WWW: http://www.chem.ox.ac.uk/researchguide/djwatkin.html

Dr Harry Powell
MRC Laboratory of Molecular Biology,
Hills Road, Cambridge, CB2 2QH, UK.
Tel: +44 (0) 1223 248011
Fax: +44 (0) 1223 213556
E-mail: harry@mrc-lmb.cam.ac.uk
WWW: http://www.mrc-lmb.cam.ac.uk/harry/

The Clipper C++ libraries for X-ray crystallography

Kevin Cowtan,
Department of Chemistry, University of York, Heslington, York, YO10 5DD, England.
E-mail: cowtan@yorvic.york.ac.uk - WWW: http://www.yorvic.york.ac.uk/~cowtan/

Introduction

The aim of the Clipper project (http://www.yorvic.york.ac.uk/~cowtan/clipper/clipper.html) is to create an object-oriented software library to facilitate the rapid development of new crystallographic applications to perform complex tasks using the simplest possible code. It is envisaged that this will allow many existing methods to be updated to use more modern statistical approaches or automated for use in high-throughput problems, in addition to allowing more complex problems to be tackled.

Motivation

Over the course of several years of crystallographic application development, the author has been involved in writing a range of crystallographic software applications. For historical reasons much of this work was undertaken in Fortran, using the basic libraries provided by the CCP4 software suite. During this work it became increasingly obvious that most of the time required for developing new computational approaches was not spent on the scientific process of evolving the new techniques, but on the technical task of writing source code. Furthermore, this code usually often involved repeated implementation of similar calculations with minor variations to adapt them to a particular task.

A number of workers in the field were also reaching the same conclusion; i.e. that productivity and progress could be greatly improved if the appropriate software tools were available in a form which offered true re-usability for problems not envisaged by the original author.

Object oriented programming, either using compiled languages such as C++, or scripting languages such as Python, provides a basis for much greater re-use of software components, increasing productivity for the developer. This is achieved in part through the mechanism of inheritance and polymorphism: More complex object may inherit properties from simpler ones; and where traditional software libraries allow new code to use old routines, polymorphic objects allow existing library routines to make use of new code. New object-oriented software frameworks for crystallography are under development in the US, the UK, and elsewhere, and initial indications suggest that existing applications can be duplicated with as little as 10% of the code required for the original implementation.

Object orientation also provide other benefits: Object encapsulation isolates the developer from the internal details of the construction of any object, and thus if the objects are well designed, the developer can work in the language of the problem rather than the language of the computer. This leads to programs which are both easier to write and easier to understand.

An additional benefit to moving away from Fortran is access to developers and libraries. Much existing crystallographic software is written in Fortran, and yet few universities are now training Fortran programmers. Fortran development tools are increasingly lagging behind their more modern counterparts, and most modern libraries for modern numerical techniques such as genetic algorithms, neural networks, and automatic differentiation are written in other languages, in particular C, C++ and Java.

In addition to technical considerations concerning the development tools, there are two significant forces for change acting in the sphere of crystallographic software development. The first is the move towards statistical and Bayesian methods, which has been driven by the need to better describe what can be known about ever larger structures given the weak and incomplete data available. The second is the increasing development of automated procedures, driven by the demands of high throughput genomics projects.

This has provided a motivation for many crystallographic applications to be modified or replaced to better satisfy these demands. If major modifications are required, then adoption of the latest tools at the same time is clearly beneficial.

Overall structure

The Clipper libraries may be conceptually divided into three sections, as follows:

1. Crystallographic infrastructure. This has the low-level tools required for crystallographic computations. These include coordinate types, operators, and crystallographic entities such as cell and spacegroup objects.

2. Data objects. General purpose data objects are provided for storing and manipulating data commonly used in crystallographic calculations. These include reflection data, crystallographic and non-crystallographic maps, and atomic models. The general purpose objects are not suitable for every purpose, but are sufficiently flexible for most.

3. Optional packages. These provide additional functionality which is required for specific types of calculation, and components which are dependent on external libraries. These include import/export methods for various file formats, and functional objects for performing common crystallographic computations.

The division of the objects between these sections, and the dependencies of the more complex objects on the more basic types is shown in Figure 1. Some of the objects in each of these sections will be described in further detail.

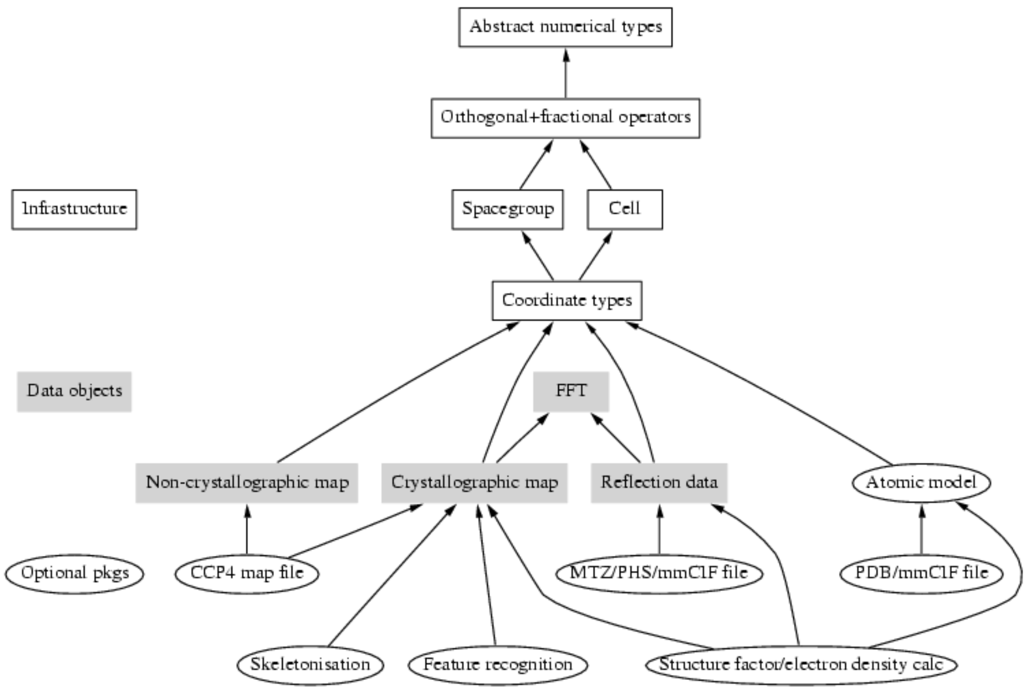


Figure 1: Organisation of objects and their dependencies in the Clipper libraries.

Crystallographic Infrastructure

The crystallographic infrastructure provides the foundations of the Clipper libraries. It provides a set of basic numerical types, and specializations of those types for crystallographic purposes.

. Abstract numerical types. These include a 3-vector, 3x3-matrix, and rotation-translation operator. The latter combines a vector and matrix. All of these types are templates, and so can be instantiated for integer or floating point members of differing precisions. General and symmetric matrices are also provided. All the common numeric operators (addition, multiplication, etc.) may be applied to these types with the expected results.

Concrete coordinate and operator types. These include orthogonal, fractional, and grid coordinates in real space; orthogonal and fractional coordinates and Miller indices in reciprocal space, orthogonal and fractional rotation-translation operators, orthogonal and fractional anisotropic U-values, grid samplings and so on.
Each of these types is distinct, thus it is never possible to mistakenly assign a fractional coordinate to an orthogonal coordinate, or apply an orthogonal operator to a fractional coordinate. Methods for transformation between the various types are provided, using the appropriate cell or grid information. In this way the novice crystallographer is guided through the complexities of crystallographic coordinate frames, and the expert is saved from a range of trivial errors. This is all achieved without significant code duplication through derivation of all these types from the abstract numerical types.

Unit cell object. The cell object includes all the operators required for conversion between spaces and coordinate frames, and the metrics required for distance computation in each space. It also computes real and reciprocal cell volumes.

Spacegroup object. The spacegroup object manages the symmetry operators associated with a spacegroup, as well as reciprocal and real space asymmetric units, Hall and Herman-Maugin spacegroup symbols and so on (Hall, 1981), (Rossmann & Arnold, 2001). The functionality is less comprehensive than that provided in CCTBX (Grosse-Kunstleve et al. 2002), focusing mainly on problems after the origin has been fixed, however the functionality provided is very highly optimised.

Other computational infrastructure is provided, including a message library and common mathematical functions. Additional crystallographic infrastructure includes rotation representations and conversions, Ramachandran plots, and atomic scattering factors and Agarwal coefficients (Agarwal & Isaacs, 1977).

Data objects

The general-purpose data objects are designed to provide to allow common tasks to be performed quickly with only a few lines of code. However there are occasional problems for which the general objects are not suitable, either because of unusual requirements of the problem or because the access pattern is inefficient for the general purpose object.

The reflection data object. Reciprocal space data is stored in `HKL_data` objects, where each data object stores a complete list of a data of a particular data type. Types may include several values: For example, a structure factor magnitude and its standard deviation are a type, as are a calculated magnitude and phase, or a set of four Hendrickson-Lattman coefficients. New types may be added for an particular application.

One property of the types is that they know how to transform themselves under a symmetry operation. This allows the implementation of ‘magic symmetry’, by which the data can appear to fill a sphere in reciprocal space, when in fact only a unique asymmetric unit is stored.

It is common to use several data lists at a time, so lists are used in conjunction with another object, `HKL_info`, which handles the list of reflection indices (h,k,l) and associated information.

The crystallographic map data object. Crystallographic maps which obey a cell repeat and spacegroup symmetry are stored in the `Xmap` object, where each object stores a complete map of data of any simple (integer, real or complex) type. Like the reflection data objects maps implement 'magic symmetry'. Thus the map appears to be infinite in every direction, when in fact only a unique set of points forming an asymmetric unit are stored. Additional objects are provided for interpolation and statistical analysis of maps.

The non-crystallographic map data object. Non-crystallographic maps are stored in the `Nxmap` object, which may also hold data of any simple (integer, real or complex) type. Non-crystallographic maps do not have symmetry or repeat, and are used for storing features not related to a crystallographic frame, such as non-crystallographic symmetry masks and molecular replacement models. The object also stores an operator (or skew matrix) relating the frame of the map to an arbitrary orthogonal coordinate frame. A reference to a non-crystallographic map, with an appropriate transformation, may be placed in a crystallographic frame.

The coordinate model objects. The main object for manipulation of coordinate models is provided by the external MMDB library. This is an extremely sophisticated package for model manipulation and database analysis. MMDB atom selections may be used directly to construct atom lists for use by Clipper methods.

An alternative package, 'MiniMol', is also provided for students and programmers new to C++ who may not need the full power of MMDB. These objects, although technically part of the core distribution, are provided as optional packages so that the developer may chose the most appropriate version.

Optional packages

Optional packages are divided into two types:

Interfaces to external libraries, including packages to access various file formats. These include interfaces to CCP4 for MTZ and map file access, Xtalview PHS file input and output (McRee, 1999), mmCIF reflection file input, and communication with CCTBX.

Crystallographic function objects. These include the calculation of structure factors and electron density maps from atomic models (isotropic and anisotropic), filtering of maps, fragment searching, and 'resolution functions'; a generic implementation of arbitrary statistical functions of position in reciprocal space (Cowtan, 2002).

Some of the objects in the optional packages are defined as templates, and so are not limited to use with the pre-defined data objects, but may also be used with new data objects defined by the developer, assuming the appropriate interfaces are provided.

Data organisation

In order to handle data from multiple sources (and in particular multiple crystals for phasing, multi-crystal averaging or refinement), the relationships between the various data objects must be stored. The Clipper libraries provide an optional mechanism for performing this data organisation by providing 'container' versions of each data object (A container is an object which can contain other objects). Any container object may contain any number of other objects of any type.

The data organisation can therefore be drawn as a tree-structure, with the top-level container as the root. For example if a crystal undergoes a change of cell parameters between room- and low-temperature data collection, it might be represented by a single spacegroup object, containing two cell objects, each of which may contain a set of structure factors. By this means the container objects may be used to construct a complete data model of the experiments and their results.

The data hierarchy can optionally be used to synchronise the fundamental properties of common objects. For example, spacegroup and cell objects are considered to be 'donors' of spacegroup and cell information, whereas data objects such as reflection lists and maps, are considered to be recipients. If a recipient need a piece of information for its initialisation and that information is not supplied, then it will examine its ancestors in the tree to see if any of them provide the necessary information. This approach may be use to avoid most problems of inconsistency between objects, as well as simplifying the code.

Rapid Application Development

The effectiveness of the Clipper libraries as a tool for rapid application development has been demonstrated by example applications at a number of levels of complexity.

At the simplest level performing an FFT, converting between Hendrickson-Lattman coefficients and phase+weight representation, and adding or subtracting Fourier coefficients are single instructions. Other tasks such as reading or writing maps or reflection data, symmetry expanding data, scaling datasets, applying a standard or user-defined filter to a map, performing a structure factor calculation, and skeletonising a map take less then 5 lines of code. Complete applications combining these operations with parsing of the input parameters and input and output or data are typically less than thirty lines of code. The resulting program source code is often comparable in size to the commands and input required to perform the same task with pre-existing CCP4 utility programs.

More complex tasks, such as the likelihood estimation of σ_a (Read, 1986), or the simulation of noisy data for a known structure with similar statistical properties to data from genuine phasing experiments have been performed in less than 100 lines of code. Development times for such applications ranges from a day to a week. The FFT-based anisotropic structure factor calculation implemented within the libraries is itself of a comparable level of complexity, requiring only 25 lines of code.

```
void sfcalc( HKL_data<datatypes::F_phi<float> >& fphidata, const Atom_list& atoms )
{
  // prepare target map
  const HKL_info& hkls = fphidata.base_hkl_info();
  const Grid_sampling grid( hkls.spacegroup(), hkls.cell(), hkls.resolution() );
  Xmap<float> xmap( hkls.spacegroup(), hkls.cell(), grid );

  // work out how big a box we need to calc density over for each atom
  Grid_range gd( hkls.cell(), grid, 3.0 );
  Xmap<float>::Map_reference_coord i0, iu, iv, iw;
  // loop over atoms
  for ( int i = 0; i < atoms.size(); i++ ) if ( !atoms[i].is_null() ) {
    AtomShapeFn sf( atoms[i] );  // get atom shape fn
    Coord_frac uvw = atoms[i].coord_orth().coord_frac( hkls.cell() );
    Coord_grid g0 = uvw.coord_grid( grid ) + gd.min();
    Coord_grid g1 = uvw.coord_grid( grid ) + gd.max();
    i0 = Xmap<float>::Map_reference_coord( xmap, g0 );
    // sum all map contributions from this atoms
    for ( iu = i0; iu.coord().u() <= g1.u(); iu.next_u() )
      for ( iv = iu; iv.coord().v() <= g1.v(); iv.next_v() )
        for ( iw = iv; iw.coord().w() <= g1.w(); iw.next_w() )
          xmap[iw] += sf.rho( iw.coord_orth() );
  }

  // now correct for multiplicity of points on special positions
  Xmap<float>::Map_reference_index ix;
  for ( ix = xmap.first(); !ix.last(); ix.next() )
    xmap[ix] *= xmap.multiplicity( ix.coord() );

  // calc structure factors from map by fft
  xmap.fft_to( fphidata );
}
```

Figure 2: *Twenty-five lines of code for the structure factor calculation using the Clipper libraries. (It's actually 32 if you count the blank lines and comments too)*

One large scale project has been undertaken so far: The automatic tracing of protein chains in medium resolution (3 Angstroms) electron density maps. The time for implementation from the initial idea to a working prototype was about a month, and resulted in about 800 lines of code.

In all but the last of these cases, an existing implementation using the CCP4 libraries and the Fortran language was available for comparison. In each case the new implementation required roughly an order of magnitude fewer lines of code than the original Fortran implementation. This has not been achieved solely by moving functionality into the libraries, since the libraries themselves are less than a third of the size of the CCP4 libraries.

Not only does reducing the complexity of the code required for a particular task reduce the development time (although not necessarily by the same factor, since development of the algorithm is still be required), but it significantly reduces the problems of debugging and support of existing code. Finally, if existing tasks can be implemented more simply, then more complex tasks may be addressed than were previously tractable.

Ultimately it is intended that a scripting interface to the libraries will be provided. This will allow scripting of applications in languages such a Python and TCL, which should further accelerate the development process by eliminating lengthy compilation and debugging steps.

Availability

The Clipper libraries will be distributed with the next version of the CCP4 suite of software, and will be available under the GNU Lesser General Public License (LGPL). The current development version and documentation are available at the following URL:
http://www.ysbl.york.ac.uk/~cowtan/clipper/clipper.html

Conclusion

The Clipper libraries provide a versatile tool for the development of a range of crystallographic applications. The libraries are largely complete and have been extensively documented. They are being used at a number of sites to solve a range of crystallographic problems. Initial applications developed using the libraries show a dramatic decrease in code complexity.

References

Agarwal, R. C.; Isaacs, N. W. (1977) Proc. Natl. Acad. Sci (USA), 74, 2835-2839.
Cowtan K. (2002) J. Appl. Cryst. 35, 655-663.
Hall, S. R. (1981) Acta Cryst. A37, 517-525.
McRee, D. E. (1999) Journal Structural Biology, 125, 156-165.
Read, R. J. (1986) Acta Cryst. A42, 140-149.
Rossmann, M. G.; Arnold, E., editors, (2001) International Tables for Crystallography, Kluwer Academic Publishers, The Netherlands.
Grosse-Kunstleve, R. W.; Sauter, N. K.; Moriarty, N. W., Adams P. D. (2002) J. Appl. Cryst. (2002). 35, 126-136.

cctbx news: Fast triplet generator for direct methods, Gallery of direct-space asymmetric units, et. al.

Ralf W. Grosse-Kunstleve, Buddy Wong and Paul D. Adams,
Lawrence Berkeley National Laboratory, One Cyclotron Road, BLDG 4R0230, Berkeley, CA 94720-8235, USA. E-mail: RWGrosse-Kunstleve@lbl.gov ; WWW: http://cci.lbl.gov/ and http://cctbx.sourceforge.net/

Introduction

The Computational Crystallography Toolbox (cctbx - http://cctbx.sourceforge.net/) is an open-source library of reusable crystallographic algorithms. In this article we give an overview of recent developments.

Fast triplet generator for direct methods

For almost a year the cctbx has included an experimental triplet generator in the cctbx.dmtbx (direct methods toolbox) submodule. This experimental code has now been replaced by a much faster version. In addition the interface was refactored to make it more accessible. Like most components of the cctbx, the triplet generator is scriptable from Python. Here is a simple stand-alone script that generates a list of unique Miller indices, searches for triplet phase relations and shows some statistics:

```
from cctbx import dmtbx
from cctbx import miller
from cctbx import crystal
from cctbx.array_family import flex
import time

crystal_symmetry = crystal.symmetry(
  unit_cell=(10,10,15,90,90,120),
  space_group_symbol="P63/mmc")
miller_set = miller.build_set(
  crystal_symmetry=crystal_symmetry,
  anomalous_flag=False,
  d_min=1.0)
triplets = dmtbx.triplet_generator(miller_set)
print "triplets per reflection: min,max,mean: %d, %d, %.2f" % (
  flex.min(triplets.n_relations()),
  flex.max(triplets.n_relations()),
  flex.mean(triplets.n_relations().as_double()))
print "total number of triplets:", flex.sum(triplets.n_relations())
print "cpu time used: %.2f" % time.clock()
```

(The flex module provides multi-dimensional arrays and associated functions.) The output of the script on a Redhat 8.0 Linux system (2.7 GHz):

```
triplets per reflection: min,max,mean: 1476, 4560, 2283.31
total number of triplets: 666727
cpu time used: 0.43 seconds
```

In a structure solution procedure the triplet search has to be performed only once at startup. The absolute runtime for the triplet search is so short that it is negligible in the context of the complete procedure. The runtime for the largest problem that we have worked on so far (a substructure with 160 sites) is 12.6 seconds on the same platform, generating almost 15 million triplets.

The dmtbx.triplet_generator() call above creates an object containing all the triplets. This object has associated functions, called *methods* in Python terminology, or *member functions* in C++ terminology. Above we have used the n_relations() method already. Another method is apply_tangent_formula() and it is used like this:

```
amplitudes = flex.double(miller_set.size(), 1) # fake data
phases = flex.double(miller_set.size(), 0) # fake phases
new_phases = triplets.apply_tangent_formula(amplitudes, phases)
```

This changes all phases, using all phases. Other tangent refinement protocols are supported. For example:

```
# keep the first 40% of the phases fixed
selection_fixed = flex.bool(int(miller_set.size()*0.4), True)
selection_fixed.resize(miller_set.size(), False)
new_phases = triplets.apply_tangent_formula(amplitudes, phases,
  selection_fixed, use_fixed_only=True, reuse_results=True)
```

The full interface documentation is available online (C++ interfaces -> dmtbx) at the cctbx site.

Gallery of direct-space asymmetric units

We have created a reference file defining direct-space asymmetric units for the 230 crystallographic space groups, following the conventions of the International Tables for Crystallography, Volume A (ITVA). However, ITVA only defines the volumes of the asymmetric units. In contrast our reference file includes additional conditions for the facets, edges and vertices. For example, consider space group P2 (No. 3). ITVA lists the following conditions:

```
Asymmetric unit   0 <= x <= 1; 0 <= y <= 1; 0 <= z <= 1/2
```

Clearly, points with coordinates 0,y,z and 1,y,z are redundant due to the periodicity of the crystal lattice. This can be accounted for by modifying the conditions in the following way:

```
Asymmetric unit   0 <= x < 1; 0 <= y < 1; 0 <= z <= 1/2
```

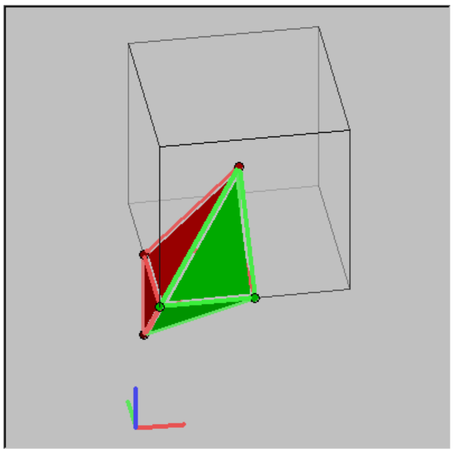
This correction might seem subtle at first, but avoids difficult problems when working with grids over the unit cell as many crystallographic algorithms do. Unfortunately there is still more to be corrected. There are two-fold axes parallel to the facets with z=0 and z=1/2. The two-fold axes generate redundancies in these facets. To account for this we need *facet-specific* conditions. In words:

> If z is exactly equal to zero or 1/2, x must be less than or equal to 1/2.

Our reference file includes additional conditions that make all 230 asymmetric units completely free of redundancies. For the cubic space groups we follow the conventions established by Koch & Fischer (1974), Acta Cryst. A 30, 490-496 (although in general only for the facets but not for the edges and vertices to keep the expressions simple).

Associated with our reference file is a set of tools, constituting about 1600 lines of Python code. These tools are used to generate our online gallery of direct-space asymmetric units, available at http://cci.lbl.gov/asu_gallery/ . For example:

Space group: P 21 3 (No. 198)



[Index] [Previous] [Next] [Grid view]

Surface area: green = inside the asymmetric unit, red = outside
Basis vectors: a = red, b = green, c = blue

```
Number of vertices: 6
  1/2, 1/2, 1/2
  1/2, 1/2, 0
  0, 1/2, 0
  0, 1/2, -1/2
  0, 0, 0
  1/2, 0, 0
```

```
Number of facets: 7
  x>=0 [y<=0]
  x<=1/2
  y<=1/2 [z>0 & x<=1/2 [z<1/2]]
  x-z<=1/2 [x+y<=1/2]
  x-z>=0 [x-y>=0]
  y+z>=0
  y-z>=0
```

[Guide to notation]

The green-and-red coloring of the facets reflects the additional conditions.

All the information shown at the web pages is automatically derived from the primary information in the reference file. The primary information used to generate the web page above is encoded in this Python function:

```
def asu_198(): # P 21 3
  return (direct_space_asu('P 2ac 2ab 3')
    & x0(-y0)
    & x2
    & y2(+z0 & x2(+z2))
    & cut((-1,0,1), r1/2)(m2)
    & cut((1,0,-1), 0)(p0)
    & cut((0,1,1), 0)
    & cut((0,1,-1), 0)
  )
```

There is a one-to-one correspondence between this function and the notation shown in the box starting with Number of facets above. The notation used in the reference file was easier to work with during development and lends itself to easy algorithmic manipulation. The notation shown on the web is easier to understand for humans (follow the Guide to notation link to learn more).

That we are able to easily provide the best notation for a given purpose *without introducing difficult to manage redundancies* is due to the rich set of abstraction facilities supported by the Python language. The first thing to notice is that the Python function with the primary information is just that, a pure Python function. We did not have to invent a new syntax to parameterize the asymmetric units. When the Python function above is called it creates a new *object*, more specifically an *instance* of the direct_space_asu *class*. *Operator overloading* is used to define the facets of the asymmetric unit. We have chosen to use the & operator for this purpose. The working of this can be better understood by considering that the code above is equivalent to:

```
def asu_198(): # P 21 3
  a = x0(-y0)
  b = x2
  c = y2(+z0 & x2(+z2))
  d = cut((-1,0,1), r1/2)(m2)
  e = cut((1,0,-1), 0)(p0)
  f = cut((0,1,1), 0)
  g = cut((0,1,-1), 0)
  return direct_space_asu('P 2ac 2ab 3') & a & b & c & d & e & f & g
```

The assignments create references to objects representing the facets. The direct_space_asu instance has a rule, in the form of the method __and__, that defines what the & operator does. Here is the *complete* code to make this work:

```
class direct_space_asu:

  def __init__(self, hall_symbol):
    self.hall_symbol = hall_symbol
    self.facets = [] # an empty list

  def __and__(self, obj):
    self.facets.append(obj) # adds the facet to the list
    return self
```

We can chain the & operators because the __and__ method returns self, which is "the direct_space_asu instance itself" (equivalent to *this in C++).

Each of the facets of an asymmetric unit is defined by a *cut plane* dividing space into "inside" and "outside." Successive cuts lead to a bounded volume. The additional conditions mentioned before are simply cuts that only apply to a certain facet instead of globally. An intersection of two cuts forms an edge. Some asymmetric units require additional *edge-specific conditions*. I.e. a facet is defined by potentially nested expressions involving one or more cuts which are sometimes combined by boolean operators (see the function above). Giving a full account of this "cut algebra" is beyond the scope of this article, but we will show the two techniques that are at the heart of the implementation.

The first technique that we use is analogous to expression templates that C++ programmers might be familiar with. However being implemented in Python it should have a different name, say *expression objects*, and it is, of course, much more approachable. Here is the *complete* code that makes boolean expressions of cuts work:

```
class cut_expr_ops:
  def __and__(self, other): return cut_expression("&", self, other)
  def __or__(self, other): return cut_expression("|", self, other)

class cut_expression(cut_expr_ops):

  def __init__(self, op, lhs, rhs):
    self.op = op
    self.lhs = lhs
    self.rhs = rhs

class cut(cut_expr_ops):

  def __init__(self, n, c):
    self.n = n
    self.c = c
```

Note that this requires nothing but Python to work, not even the cctbx. For the meaning of n and c refer to http://cci.lbl.gov/asu_gallery/facet_notation.html .

Before we explain how the code above works let's look at some illustrative examples. Let all the letters below be references to cut instances. The code above enables arbitrarily complex, nested expressions:

```
a & b
a | b
a & (b | c)
a & (b & (c | d))
```

The result of each of these expressions is an *object representing the expression*. When Python evaluates the expressions it leads to the cut_expression class recursively keeping track of the boolean operations, i.e. the operator symbol (& or |) and the operands on both sides. The parentheses are resolved for us by Python, so we don't have to keep track of them ourselves.

For people who have worked with language translators before (e.g. yacc and lex): our expression object is equivalent to an *abstract syntax tree*. The difference to the conventional approach is that we don't have to write our own parser. Instead we use operator overloading to leverage Python itself as a parser for our domain-specific syntax. The only restriction is that our syntax must be compatible with the Python syntax. We view this as a useful restriction because it enforces a consistent and familiar syntax.

The code above uses inheritance. For readers not familiar with inheritance, the code could be rewritten as follows:

```
class cut_expression:

  def __init__(self, op, lhs, rhs):
    self.op = op
    self.lhs = lhs
    self.rhs = rhs

  def __and__(self, other): return cut_expression("&", self, other)
  def __or__(self, other): return cut_expression("|", self, other)

class cut:

  def __init__(self, n, c):
    self.n = n
    self.c = c

  def __and__(self, other): return cut_expression("&", self, other)
  def __or__(self, other): return cut_expression("|", self, other)
```

This is doing exactly the same job, but is poor style because it unnecessarily introduces explicit duplication of the same code, and potentially multiple places to fix the same bug.

We can easily evaluate the expression objects in different ways by adding pairs of methods, one to the cut class to do the actual work, and one in the cut_expression class to enable recursive expressions. For example, the is_inside() method to test if a given point is in the asymmetric unit is implemented as follows:

```
class cut_expr_ops:
  def __and__(self, other): return cut_expression("&", self, other)
  def __or__(self, other): return cut_expression("|", self, other)

class cut_expression(cut_expr_ops):

  def __init__(self, op, lhs, rhs):
    self.op = op
    self.lhs = lhs
    self.rhs = rhs

  def is_inside(self, point):
    if (self.op == "&"):
      return self.lhs.is_inside(point) and self.rhs.is_inside(point)
    else:
      return self.lhs.is_inside(point) or self.rhs.is_inside(point)

class cut(cut_expr_ops):

  def __init__(self, n, c):
    self.n = n
    self.c = c

  def is_inside(self, point):
    # do the actual work, return True or False
```

We have implemented similar pairs of methods for rewriting the cut expressions in different formats, or for change-of-basis transformations. Alternatively it is possible to evaluate the expression objects by introspection without adding additional methods. This approach is used, e.g. in the implementation of the computation of the polygon edges and vertices shown in the pictures on the web.

A second fundamental technique is used to implement the recursive nesting of cut expressions. Python's __call__ method (equivalent to operator() in C++) allows objects to act like functions. We chose this as the syntax for defining additional facet- and edge-specific conditions. For example:

```
x0 = cut((1,0,0), 0)) # equivalent to x >= 0
y2 = cut((0,-1,0), r1/2)) # equivalent to y <= 1/2
z4 = cut((0,-1,0), r1/4)) # equivalent to z <= 1/4
x0(y2)
x0(y2(z4))
```

The expression x0(y2) means that a point is inside the asymmetric unit only if x>=0. If x is exactly zero, the point is inside the asymmetric unit only if y<=1/2. The expression x0(y2(z4)) is similar, but if x is exactly zero and y is exactly 1/2 the point is inside the asymmetric unit only if z<=1/4.

This is the corresponding implementation:

```
class cut(cut_expr_ops):

  def __init__(self, n, c, cut_expr=None):
    self.n = n
    self.c = c
    self.cut_expr = cut_expr

  def __call__(self, expr):
    return cut(self.n, self.c, cut_expr=expr)
```

The reader is invited to study the implementation of the is_inside() method to see how the nested cut expressions are used. The full source code is in the directory cctbx/cctbx/sgtbx/direct_space_asu in the cctbx source code distribution (see below).

Macintosh OS 10 support

As of May 2003 the cctbx is fully functional under Mac OS 10. However, there are a few Mac-specific issues. Firstly, Python 2.3 must be available, and it must be installed as root (the Mac is the only platform with this requirement). Secondly, the only C++ compiler that currently works for us under Mac OS 10 is standard gcc 3.3 as available at http://gcc.gnu.org/ . Unfortunately compiler bugs prevent the use of optimization. Therefore the binaries are relatively slow compared to other platforms. We hope that this will improve as new gcc versions are released. Currently Apple's compiler cannot be used to install the cctbx. More detailed Mac-specific information is available at http://cci.lbl.gov/cctbx_build/mac_os_x_notes.html . We will continuously update this web page to reflect the latest developments.

Automatic builds, source code & binary bundles

We have developed a system for automatically generating self-contained cctbx source code bundles and self-extracting binary bundles for a variety of platforms. The bundles are available at http://cci.lbl.gov/cctbx_build/ , complete with build logs and results of automatic regression tests. The cctbx online documentation is also continuously updated at http://cctbx.sourceforge.net/current_cvs/ . It is highly recommended to use the current bundles instead of the old cctbx 1.0 release from November 2001. We take great care that the bundles are consistent and in working order on all platforms. The C++ interfaces have changed significantly since the 1.0 release but are very stable since the fall of 2002.

All bundles are marked with a time stamp and we are archiving all builds that worked on all platforms. Practically this is like a full traditional release, only every other day. This enables us to use a novel release policy. Instead of posting announcements *after* making significant changes we will post announcements *before* starting developments that break backwards compatibility. For example, we are planning to use Python 2.3 features shortly after Python 2.3 final becomes available (scheduled for August). We will specifically set aside a release as "the last release that worked with Python 2.2" and create a source code branch for bug fixing only.

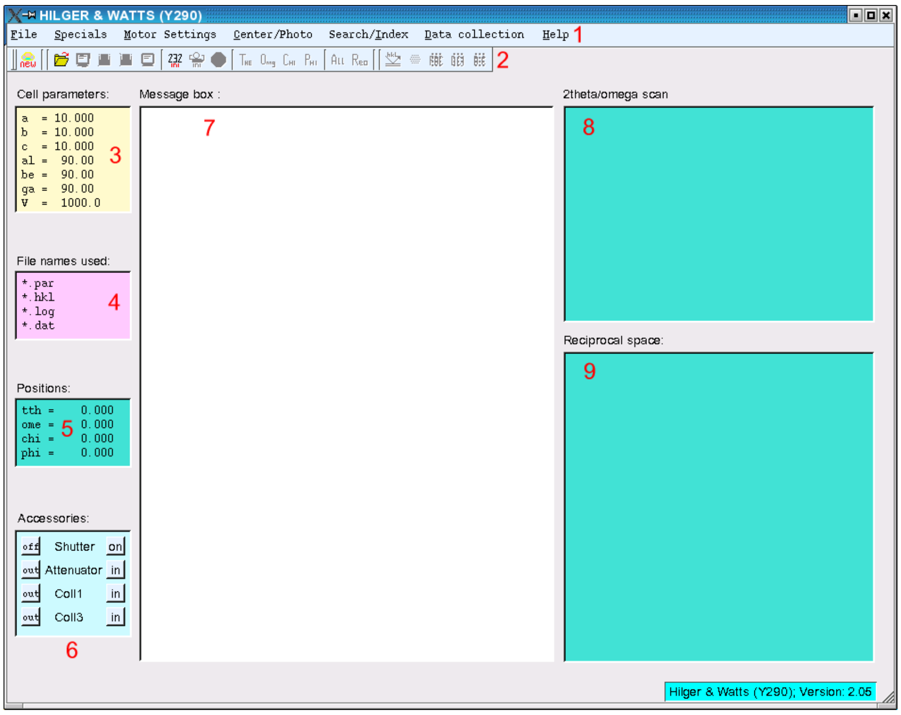
Our work was funded in part by the US Department of Energy under Contract No. DE-AC03-76SF00098. We gratefully acknowledge the financial support of NIH/NIGMS.

Control Program for Single Crystal Diffractometer: The Use of the Qt C++ Class Library

Jürgen Kopf,
Institute of Inorganic and Applied Chemistry, University of Hamburg, Martin-Luther-King-Platz 6, D-20146 Hamburg, Germany. E-mail: kopf@xray.chemie.uni-hamburg.de -
WWW: http://aclinux1.chemie.uni-hamburg.de/~xray/

Abstract

In this paper I describe a new "open source" control program for single crystal diffractometer. This program has been developed to control a very reliable Hilger & Watts (Y290) four circle diffractometer, first installed in 1971. A second version was developed for a Syntex $P2_1$ diffractometer. The new control software is completely written in C++ using the Qt class library, maintained and distributed by the Norwegian software company **Trolltech** [1].



```
a  = 10.000
b  = 10.000
c  = 10.000
al =  90.00
be =  90.00
ga =  90.00
V  =  1000.0

*.par
*.hkl
*.log
*.dat

tth =     0.000
ome =     0.000
chi =     0.000
phi =     0.000

Hilger & Watts (Y290); Version: 2.05
```

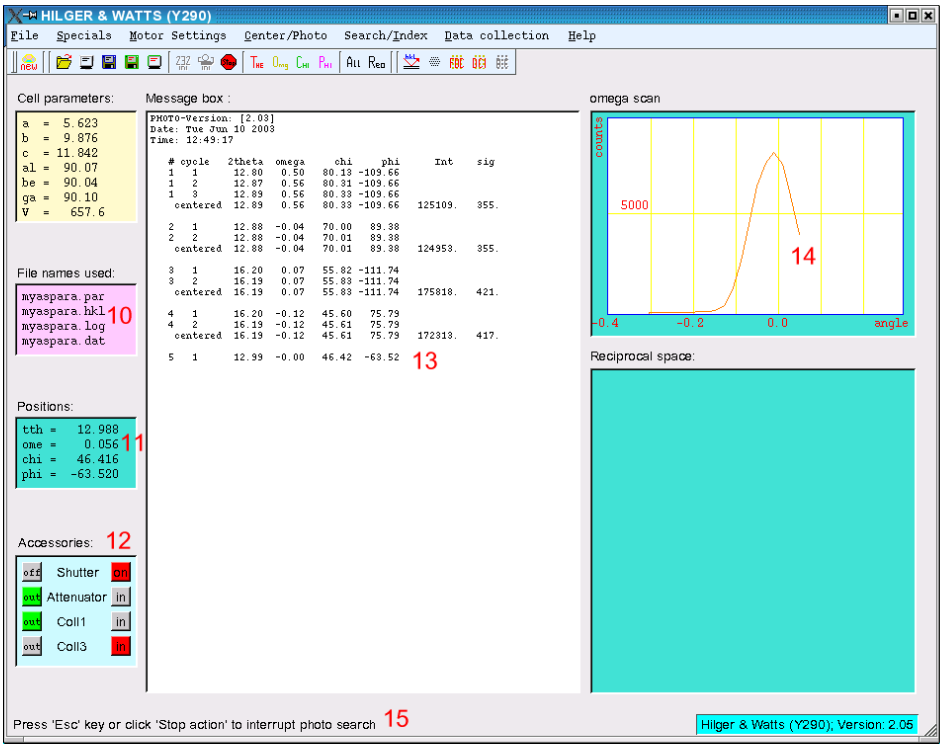
Fig. 1: *Screenshot of program* ***Y290*** *just after the start of the program. Red numbers are inserted into all figures for explanation of important features of the program.*

Introduction

Over the past decade I have been developing single crystal diffractometer control software that uses easy to learn pull-down menus, dialog-boxes and file-selector-boxes. The program **Y290** was developed to control a Hilger & Watts (Y290) four circle diffractometer via an interface, using 68008 microcomputers to control the four stepper motors. The new interface is connected to the PC via a normal serial line (RS-232). The original software, first developed in 1990, was completely written in **Fortran77** for an ATARI Mega ST2 [2].

By isolating all relevant modifications required to drive a diffractometer with different slew directions, it was possible in 1995 to rewrite the same control program for a Syntex $P2_1$ diffractometer [3]. This program **P21** used the so-called "SIEMENS-Box" for driving the four DC motors also via a serial line.

The new diffractometer control software is now completely written in C++ using the Qt class library. In the last issue of the *IUCr Computing Commission Newsletter* was a nice article on "Cross-platform C++ GUI developement using Qt" by Barry R. Smith [4]. I agree with Barry Smith that the Qt API belongs to the best in the world and I also agree that **Trolltech's** documentation and support is among the best I've ever seen for any software.



```
a  =  5.623
b  =  9.876
c  = 11.842
al =  90.07
be =  90.04
ga =  90.10
V  =   657.6
```

```
PHOTO-Version: [2.03]
Date: Tue Jun 10 2003
Time: 12:49:17

   # cycle   2theta  omega     chi     phi       Int     sig
   1   1      12.80   0.50   80.13 -109.66
   1   2      12.87   0.56   80.31 -109.66
   1   3      12.89   0.56   80.33 -109.66
    centered  12.89   0.56   80.33 -109.66   125109.   355.

   2   1      12.88  -0.04   70.00   89.38
   2   2      12.88  -0.04   70.01   89.38
    centered  12.88  -0.04   70.01   89.38   124953.   355.

   3   1      16.20   0.07   55.82 -111.74
   3   2      16.19   0.07   55.83 -111.74
    centered  16.19   0.07   55.83 -111.74   175818.   421.

   4   1      16.20  -0.12   45.60   75.79
   4   2      16.19  -0.12   45.61   75.79
    centered  16.19  -0.12   45.61   75.79   172313.   417.

   5   1      12.99  -0.00   46.42  -63.52
```

```
myaspara.par
myaspara.hkl
myaspara.log
myaspara.dat
```

```
tth =    12.988
ome =     0.056
chi =    46.416
phi =   -63.520
```

Press 'Esc' key or click 'Stop action' to interrupt photo search

Fig. 2: *Screenshot of **Y290** during centering of reflection #5. The centering is done in up to five cycles of omega-, 2theta- and chi-scans in that order. The actual omega-profile is shown in the draw-widget.*

Qt is a multi-platform C++ application framework that lets developers write single-source applications that run – natively – on Windows, Linux/Unix and Mac OS X. It is released in two different versions: The **Qt Free Edition**, which can be used free of charge for developing free software ("open source"), and the **Qt Professional Edition**. Since the new programs **Y290** and **P21** use the Qt free edition on Linux/X11, the source code is also free software (see: reference to the GNU General Public License in the listing of file: **main.cpp**).

The source code of both programs contains functions (subroutines) for

- •taking rotation photos with a **Polaroid** camera,
- •fast photo rotation search,
- •axes photos about all three cell parameters (this is important for detecting "twinning problems"),
- •fast "random" reflection search,
- •carefully centering of all reflections found,
- •auto-indexing routines (**Sparks** algorithm and/or **Duisenberg's** *DirAx* algorithm currently not fully implemented),
- •least-squares refinement for cell parameters and orientation matrix,
- •a Bravais-routine following the famous **Le Page** algorithm,
- •flexible and fast precession simulation,
- •a fast data collection routine for high order reflections with following reflection centering for the 25 reflections with highest intensity and
- •flexible data collection.

All functions use dialog-, alert- and file-selector-boxes and the scan profiles are shown online during the centering of reflections. All functions can be interrupted at any time.

C++ program Y290

The source code of program **Y290** consists of one file containing the event-loop: **main.cpp** and currently (version: 2.05) **49** additional files, all ending with: **.cpp** and corresponding header-files, ending with: **.h**. A "makefile" is important for compiling and linking all source-files to an executable module. Program **Y290** uses shared libraries and currently has a size of **773 KB**. It is also possible to compile a statically linked version, but this has a larger size and takes more time to start.

C++ program P21

The source code of program P21 consists of one main file with the event-loop and additional (version: 2.07) **51** files, all ending with: **.cpp** and corresponding header-files, ending with: **.h.** Program **P21** currently has a size of **725 KB**. Most of these files are completely identical to the **Y290** version except for the motor functions.

The development and testing of this program was done via the internet. The real diffractometer Syntex $P2_1$ with the "SIEMENS-Box" is located at the University of Augsburg about 800 km away from Hamburg. This is not a problem except that, if something goes wrong with the diffractometer and/or the mounted crystal, the operator in Augsburg has to help.

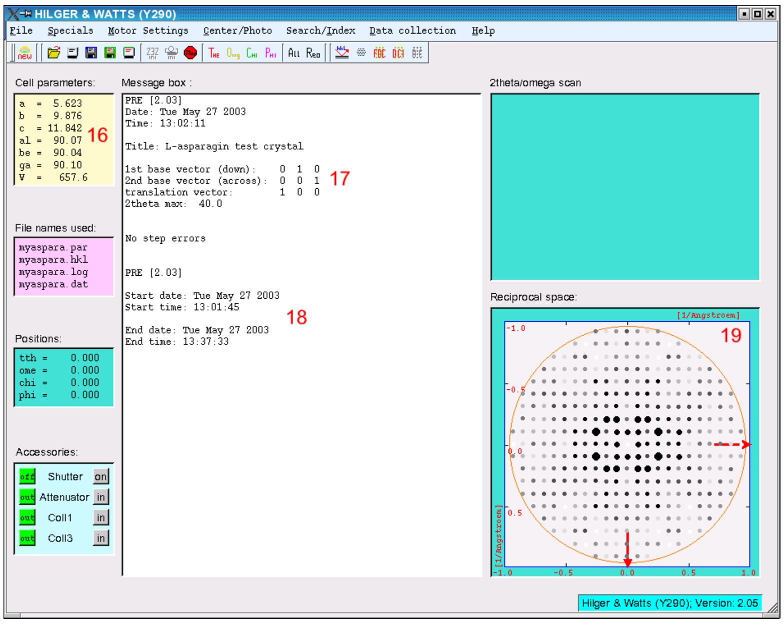
Since the Syntex $P2_1$ possesses a new **Seifert ID 3003** X-ray generator that can be controlled via a second serial line program P21 contains a special function for controlling the generator. This feature can be used during the data collection when very strong reflections have to be measured, be-cause the Syntex $P2_1$ does not possess an attenuator. Therefore, strong reflections are all re-measured at the end of the data collection with lower intensity of the primary beam.

Explanation of important features marked with red numbers in all figures

In Fig. 1 the startscreen of program **Y290** is shown. Fig. 2 is a snapshot during the centering of one reflection, Fig. 3 shows a measured 1kl precession simulation, Fig. 4 shows the data collection and, finally, Fig. 5 shows an **OpenGL** simulation of the Hilger & Watts diffractometer, in case that there is no real diffractometer available. In all figures red numbers are inserted and detailed explanations of important features of the program **Y290** are given here:

<u>Red numbers in Figure 1:</u>

1)**<u>Menubar:</u>** A nice feature of Qt is that the different menus can be easily enabled or disabled. With this feature a sophisticated user interface can be realized which is much easier to use than the classical command-line input. For example: A photo rotation search is only useful after the user has taken a rotation photo and has given a meaningful input of reflections to search. Another example: A data collection can only be measured after indexing the crystal, refinement of the orientation matrix and correct determination of the Bravais lattice. **<u>Important:</u>** All menus have meaningful keystroke short cuts. Therefore, its is very easy and effective to run the software without the mouse.



```
Cell parameters:
a  =  5.623
b  =  9.876
c  = 11.842   16
al =  90.07
be =  90.04
ga =  90.10
V  =   657.6

File names used:
myaspara.par
myaspara.hkl
myaspara.log
myaspara.dat

Positions:
tth =    0.000
ome =    0.000
chi =    0.000
phi =    0.000

Accessories:
off  Shutter     on
out  Attenuator  in
out  Coll1       in
out  Coll3       in
```

```
Message box :
PRE [2.03]
Date: Tue May 27 2003
Time: 13:02:11

Title: L-asparagin test crystal

1st base vector (down):      0  1  0
2nd base vector (across):    0  0  1   17
translation vector:          1  0  0
2theta max:  40.0


No step errors


PRE [2.03]

Start date: Tue May 27 2003
Start time: 13:01:45
                                 18
End date: Tue May 27 2003
End time: 13:37:33
```

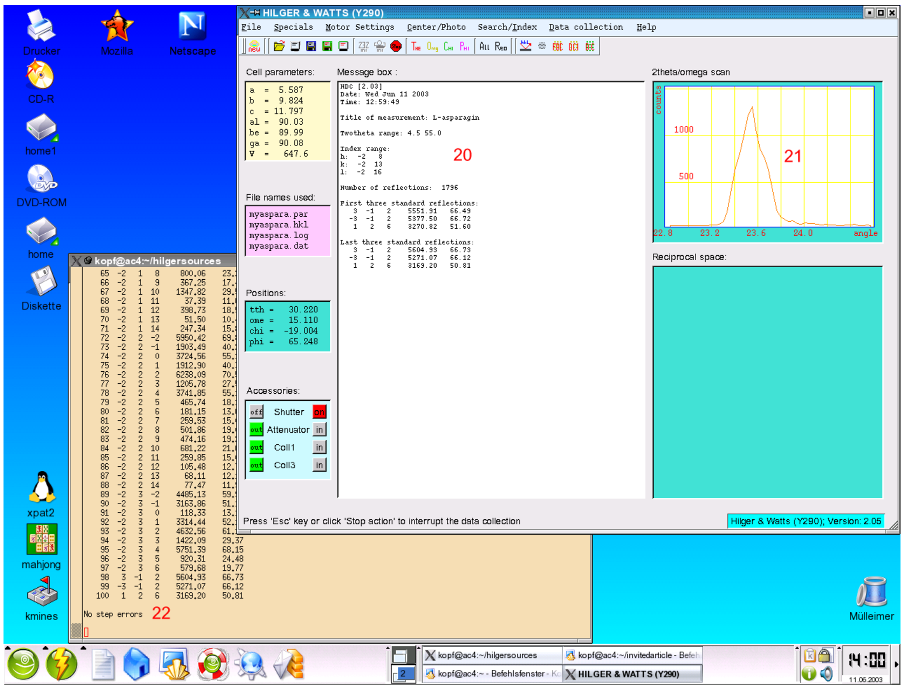
Fig. 3: *Screenshot of a* ***1kl****-precession simulation. Note the orthorhombic* ***mm*** *symmetry of the reciprocal plane of the L-asparagin test-crystal.*

2)**<u>Toolbar:</u>** Some icons are still missing. Only three icons are **enabled** at the beginning: new, open and ini-232. This means the user can start a new project with the compound dialog, open an existing file, or he can, finally, initialize the serial line (RS-232) and, if this was successful, he can then initialize the Hilger & Watts diffractometer.

3)**<u>Cell-parameter-widget:</u>** The actual cell parameters are shown here. In this case the default values are shown.

4)**<u>File-name-widget:</u>** This widget shows the file names that are in use. No file names and/or no project are given so far.

5)**<u>Position-widget:</u>** In this widget all four diffractometer angles are shown. One nice feature of the new Hilger & Watts interface [5] is that the positions can be asked via the serial interface at any time during positioning of the four circles.



```
Cell parameters:
a  =  5.587
b  =  9.824
c  = 11.797
al =  90.03
be =  89.99
ga =  90.08
V  =   647.6

File names used:
myaspara.par
myaspara.hkl
myaspara.log
myaspara.dat

Positions:
tth =   30.220
ome =   15.110
chi =  -19.004
phi =   65.248

Accessories:
off  Shutter     on
out  Attenuator  in
out  Coll1       in
out  Coll3       in
```

```
Message box :
NDC [2.03]
Date: Wed Jun 11 2003
Time: 12:59:49

Title of measurement: L-asparagin

Twotheta range: 4.5 55.0

Index range:                    20
h:  -2   8
k:  -2  13
l:  -2  16

Number of reflections:  1796

First three standard reflections:
   3  -1   2    5551.91   66.49
  -3  -1   2    5377.50   66.72
   1   2   6    3270.82   51.60

Last three standard reflections:
   3  -1   2    5604.93   66.73
  -3  -1   2    5271.07   66.12
   1   2   6    3169.20   50.81
```

Press 'Esc' key or click 'Stop action' to interrupt the data collection

```
kopf@ac4:~/hilgersources
    65  -2   1   8     800.06    23.
    66  -2   1   9     367.25    17.
    67  -2   1  10    1347.82    29.
    68  -2   1  11      37.39    11.
    69  -2   1  12     398.73    18.
    70  -2   1  13      51.50    10.
    71  -2   1  14     247.34    15.
    72  -2   2  -2    5950.42    69.
    73  -2   2  -1    1903.49    40.
    74  -2   2   0    3724.56    55.
    75  -2   2   1    1912.90    40.
    76  -2   2   2    6238.09    70.
    77  -2   2   3    1205.78    27.
    78  -2   2   4    3741.85    55.
    79  -2   2   5     465.74    18.
    80  -2   2   6     181.15    13.
    81  -2   2   7     259.53    15.
    82  -2   2   8     501.86    19.
    83  -2   2   9     474.16    19.
    84  -2   2  10     681.22    21.
    85  -2   2  11     259.85    15.
    86  -2   2  12     105.48    12.
    87  -2   2  13      68.11    12.
    88  -2   2  14      77.47    11.
    89  -2   3  -2    4485.13    59.
    90  -2   3  -1    3163.86    51.
    91  -2   3   0     118.33    13.
    92  -2   3   1    3314.44    52.
    93  -2   3   2    4632.56    61.
    94  -2   3   3    1422.09    29.37
    95  -2   3   4    5751.39    68.15
    96  -2   3   5     920.31    24.48
    97  -2   3   6     579.68    19.77
    98   3  -1   2    5604.93    66.73
    99  -3  -1   2    5271.07    66.12
   100   1   2   6    3169.20    50.81

No step errors   22
```

Fig. 4: *Screenshot of program* ***Y290*** *during the data collection. In this figure the whole screen is shown.*

6)**<u>Accessory-widget:</u>** Here the user can open or close the shutter or he can insert or remove the attenuator of the Hiler & Watts by clicking the corresponding buttons. Also the state of the accessories are shown here, if the shutter is opened by the program.

7)**<u>Message-box-widget:</u>** This widget is a (readonly) Qt multi-line editor widget and functions as a message box.

8)**<u>Draw-widget:</u>** This widget shows the actual profiles during centering and data collection. The scaling of the diagram (abscissa and ordinate) is done automatically.

9)**<u>Precession-widget:</u>** This widget shows the measured reciprocal plane if a precession simulation was started.

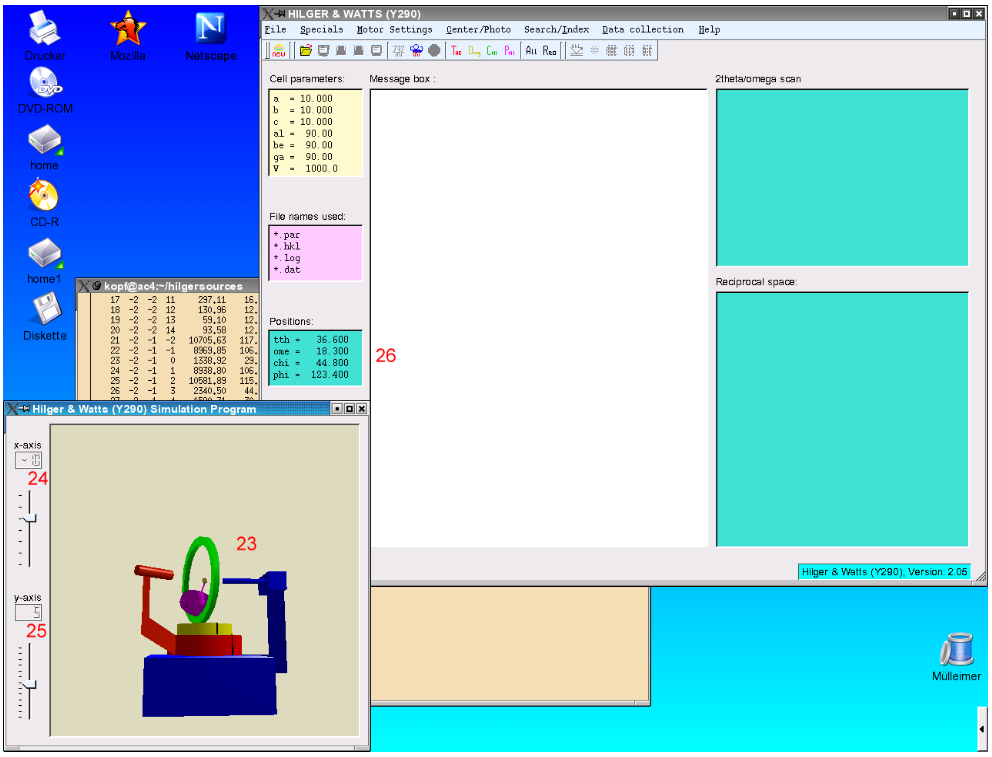
<u>Red numbers in Figure 2:</u>

10)The file-name widget is now initialized to project: **myaspara**. In this Fig. 2 a photo reflection search and centering is done. The search and the centering can be interrupted at any time. Compare red number 15.

11)The positions of all four circles for reflection #5 are given. The position of the omega circle is: **0.056** at this moment (compare with the abscissa of the draw widget).

12)The accessory widget shows that the shutter is open and the 3mm secondary beam collimator is inserted. Inserting these collimators gives sharper profiles, especially for the chi scans. This is also a nice feature of the new Hilger & Watts interface [5].

13)The centering information of the photo reflection search is shown in the message box. Up to this moment, four reflections are found and centered. In the centering functions there are up to five cycles of omega-, 2theta- and chi-scans in that order. If the precision is exact enough the final intensity and the corresponding standard deviation is measured.



```
a  = 10.000
b  = 10.000
c  = 10.000
al =  90.00
be =  90.00
ga =  90.00
V  =  1000.0
```

```
*.par
*.hkl
*.log
*.dat
```

```
tth =   36.600
ome =   18.300
chi =   44.800
phi =  123.400
```

```
17  -2  -2  11     297.11    16.
18  -2  -2  12     130.96    12.
19  -2  -2  13      59.10    12.
20  -2  -2  14      93.58    12.
21  -2  -1  -2   10705.63   117.
22  -2  -1  -1    8969.85   106.
23  -2  -1   0    1338.92    29.
24  -2  -1   1    8938.80   106.
25  -2  -1   2   10581.89   115.
26  -2  -1   3    2340.50    44.
```

Fig. 5: ***OpenGL*** *simulation of the Hilger & Watts. An extra window opens up in case that there is no real diffractometer available.*

14)The actual omega profile of reflection #5 is shown. The first cycle is finished and the omega scan of the second centering cycle is on the way. These online profiles can be very helpful in detecting "twinning" problems.

15)The photo reflection search with centering can be interrupted at any time. Therefore, in the **status bar** a permanent message is given to the user that he has to press the 'Esc' key or can click the 'Stop' icon in the toolbar in cases he wants to interrupt the photo reflection search.

Red numbers in Figure 3:

16)Now the L-asparagin test-crystal is indexed and an orientation matrix exists. Therefore, in the cell parameter widget the actual cell parameters are given. In this situation the user can measure different reflections or he can start a precession simulation. It is also possible to start a fast data collection or to give the input for the data collection. All corresponding icons are set active.
17)In this case the user has started a **precession** simulation. In a small and simple dialog the user has to give **two base vectors** and a **translation vector** and finally a maximal 2theta value. The first base vector always indicates down while the second base vector indicates to the right. This information is repeated in the message-box.
18)The precession simulation started at 13:01:45 and ended at 13:37:33. That means, the whole precession simulation took about 35 minutes to measure.
19)In the precession widget the measured **1kl** plane is shown. Notice the orthorhombic **mm** symmetry of the reciprocal plane. Notice also the extinguished **1 0 0** reflection in the middle of the plane. L-asparagin crystallizes in space group: $P2_12_12_1$.

Listing 1: File **main.cpp** with copyright notice, reference to the GNU General Public License and the main event-loop

```
/*************************************************************************/
/* File: main.cpp                                                        */
/* Copyright (C)   1997-2003   J. Kopf                                   */
/* University of Hamburg.                                                */
/* Revision date:  Mar/11/2003                                           */
/* Linux-Version:  2.05                                                  */
/* e-mail:         kopf@xray.chemie.uni-hamburg.de                       */
/*                                                                       */
/* This program is free software; you can redistribute it and/or modify */
/* it under the terms of the GNU General Public License as published by */
/* the Free Software Foundation; either version 2 of the License, or    */
/* (at your option) any later version.                                   */
/*                                                                       */
/*************************************************************************/
/*                                                                       */
/* This is the main program for the HILGER & WATTS (Y290)                */
/* diffractometer program.                                               */
/*                                                                       */
/* Content of this file:                                                 */
/* ---------------------                                                 */
/* int main(int argc, char* argv[])                                      */
/*                                                                       */
/*************************************************************************/

#include "y290widget.h"
#include "datatypes.h"
#include "y290.h"

int main(int argc, char* argv[])
{
  QApplication::setColorSpec(QApplication::CustomColor);
  QApplication y290app(argc, argv);

  Y290Widget* y290widget = new Y290Widget();
  y290widget->setGeometry(50, 20, 940, 570);

  y290app.setMainWidget(y290widget);

  y290widget->show();

  return y290app.exec();
}
```

<u>Red numbers in Figure 4:</u>

20)Actual **data collection** information is given in the message-box. These are: a) title of measurement, b) 2theta range, c) index range, d) number of reflections to be measured and e) index and intensity information about the first three and the last three measured reflections.
21)The online 2theta/omega reflection profile of the last reflection (in this case the third standard reflection) is shown.
22)In the terminal window all measured reflections are listed. Every 97 reflections the three standard reflections are measured. After that the stepper motors are checked ('no step errors'). In fact, the new Hilger & Watts stepper motor system [5] does not have any encoder system.

<u>Red numbers in Figure 5:</u>

23)The **simulated** Hilger & Watts diffractometer is shown in an extra window. The base of the diffractometer is given in blue, while the four circles have four different colors: 2theta in red, omega in yellow, chi in green and phi in violet.
24)With this slider bar the whole diffractometer can be rotated about the x-axis. In this screenshot the diffractometer was rotated by **-10** degrees.
25)The whole diffractometer can also be rotated about the y-axis (**5** degrees in this case).
26)The final positions of all four simulated circles are shown in the position widget.

Example source code

In order to convince you that it is easy to realize a sophisticated user interface with menus and icons I would like to give some source code examples. In Listing 2 a code section of file **y290widget.cpp** is given that creates and activates the toolbar and the menubar. Numbers are added at the begin of each line for explanations of some features.

Listing 2: Code section of file **y290widget.cpp** that creates and activates the toolbar and the menubar at the start of the program.

```
 1: // create the toolbar
 2:   qtools = new QToolBar(this, "file operations");
 3:
 4:   finew = new QToolButton(QIconSet(QPixmap(pm1)),
 5:     "new project", "This button defines a new project",
 6:     this, SLOT(filenew()), qtools);
 7:   finew->setEnabled(true);
 8:
 9:   qtools->addSeparator();
10:   qtools->addSeparator();
11:
12:   fiopen = new QToolButton(QIconSet(QPixmap(pm2)),
13:     "read file", "Click this botton to read a: *.par data-file",
14:     this, SLOT(fileread()), qtools);
15:     ......
16:
17: // set icons active/inactive
18:   fiopen->setEnabled(true);
19:   fiinfo->setEnabled(false);
20:   ......
21:
22:   qtools->addSeparator();
23:
24:   sprs232 = new QToolButton(QIconSet(QPixmap(pm7)),
25:     "ini RS232", "Click this botton to initialize the RS232",
26:     this, SLOT(initrs232()), qtools);
27:   spiny = new QToolButton(QIconSet(QPixmap(pm8)),
28:     "ini Y290", "Click this botton to initialize the Y290",
29:     this, SLOT(initemufs()), qtools);
30:   spsto = new QToolButton(QIconSet(QPixmap(pm9)),
    "Stop action", "Click this botton to stop any action",
    this, SLOT(stopaction()), qtools);
  qtools->addSeparator();

// set icons active/inactive
  sprs232->setEnabled(true);
  spiny->setEnabled(false);
  spsto->setEnabled(false);

// create the file menu
  file = new QPopupMenu();
  filnew = file->insertItem(pm1, "New...", this,
                            SLOT(filenew()), CTRL + Key_N);
  file->insertSeparator();
  filgo = file->insertItem("Go", this,
                            SLOT(filego()), CTRL + Key_O);
  file->insertSeparator();
  filrea = file->insertItem(pm2, "Read...", this,
                            SLOT(fileread()), CTRL + Key_R);
  filinf = file->insertItem(pm3, "Info", this,
                            SLOT(fileinfo()), CTRL + Key_I);
  ......

  file->setItemEnabled(filnew, true);
  file->setItemEnabled(filgo, false);
  file->setItemEnabled(filrea, true);
  file->setItemEnabled(filinf, false);
  file->setItemEnabled(filwri, false);
  file->setItemEnabled(filwas, false);
  file->setItemEnabled(filedi, false);

// create the special menu
  spec = new QPopupMenu();
  spers232 = spec->insertItem(pm7, "Initialize RS232", this,
                              SLOT(initrs232()), ALT + Key_R);
  spey290  = spec->insertItem(pm8, "Initialize Y290", this,
                              SLOT(initemufs()), ALT + Key_Y);
  speshu   = spec->insertItem("Specials...", this,
                              SLOT(getspecials()), Key_F4);
  spec->insertSeparator();
  spesto   = spec->insertItem(pm9, "Stop action", this,
                              SLOT(stopaction()), Key_F1);

  spec->setItemEnabled(spers232, true);
  spec->setItemEnabled(spey290,false);
  spec->setItemEnabled(speshu, false);
  spec->setItemEnabled(spesto, false);
  ......

// create a submenu for the search/index menu
  QPopupMenu* subindex = new QPopupMenu;
  subindex->insertItem("Sparks-Algorithm", this, SLOT(index()));
  subindex->insertItem("DirAx-Algorithm", this, SLOT(dirax()));

// create the search/index menu
  ind = new QPopupMenu();
  indrsd = ind->insertItem("Reflection search dialog...", this,
                           SLOT(searchinput()), CTRL + Key_F);
  indsea = ind->insertItem("Reflection search", this,
                           SLOT(searchrefl()), CTRL + Key_S);
  ind->insertSeparator();
  indind = ind->insertItem("Index", subindex, Key_F1);
  indlsq = ind->insertItem("Lsq...", this,
                           SLOT(lsqrefine()), Key_F2);
  indlep = ind->insertItem("Bravais...", this,
                           SLOT(bravais()), Key_F3);
  ind->insertSeparator();
  indexi = ind->insertItem("Expand input...", this,
                           SLOT(gethklinput()), CTRL + Key_X);
  indexc = ind->insertItem("Expand center", this,
                           SLOT(expand()), CTRL + Key_G);

  ind->setItemEnabled(indrsd, false);
```

```
104:   ind->setItemEnabled(indsea, false);
105:   ind->setItemEnabled(indind, false);
106:   ind->setItemEnabled(indlsq, false);
107:   ind->setItemEnabled(indlep, false);
108:   ind->setItemEnabled(indexi, false);
109:   ind->setItemEnabled(indexc, false);
110:   ......
111:
112: // finally create the menubar
113:   QMenuBar* menu = new QMenuBar(this);
114:   menu->setFont(QFont("Courier", 10));
115:   menu->insertItem("&File  ", file);
116:   menu->insertItem("&Specials  ", spec);
117:   menu->insertItem("&Motor Settings  ", set);
118:   menu->insertItem("&Center/Photo  ", cen);
119:   menu->insertItem("Search/&Index  ", ind);
120:   menu->insertItem("&Data collection  ", col);
121:   menu->insertSeparator();
122:   menu->insertItem("&Help", help);
```

On line 2, the toolbar is created on the *heap* with `qtools` as a pointer to a `QToolBar` object, which is defined in the *header* file **y290widget.h**. Since the object `qtools` is a pointer, you use the `->` operand to access its members (line 9 and 10). On line 4, 12, 24, 27 and 30 buttons are created for this toolbar. The arguments given to the `QToolButton` constructor are a `QIconSet` object which holds a `QPixmap(pm1)`, the text that will appear if you hold the mouse pointer over the icon for a few seconds, the text that will appear in the status bar, the object `this` holding the slot you want the icons connected to, the actual slot `filenew()` and, finally the parent widget. On line 7, 18 and 36 the corresponding icons are activated, while on line 19, 37 and 38 the icons are set inactive. This means that the user can only create a new project, read a parameter file or initialize the serial line RS-232 after the start of program **Y290**. At this point the user usually initializes the serial line and, if this was successful, icon `sprs232` is disabled and icon `spiny` is enabled. Now the user can initialize the diffractometer. After that, both icons are disabled.

To create pull-down menus, you need to use two Qt classes `QMenuBar` and `QPopupMenu`. While the latter represents the individual menus, the `QMenuBar` represents the entire bar of menus. To create a bar with a few menus, you first need to create one or more `QPopupMenu` objects and add entries to them. Then, you have to create one `QMenuBar` object and add the `QPopupMenu` objects to it with the `insertItem()` member function.

On line 41, 63 and 86 `QPopupMenu` objects are created on the *heap* with default arguments. Note that on line 42, 45, 48 and 50 the menus **new**, **go** (not jet implemented), **read** and **info** are inserted. You do this through the `QPopupMenu::insertItem()` member function, which takes in this case a `QPixmap(pm1)`object, the menu text, the object holding the slot, the actual slot and, finally the keystroke short cuts as arguments. Similar as in the toolbar on line 54 through 60 the filnew and filrea menus are enabled, while all other menus are set inactive.

Finally, on line 113 the menubar is created on the *heap*. Note that line 114 formats the text of the menubar. This is done through the `setFont()` function, which takes a `QFont` object as an argument. On line 115 through 120 the pull-down menus are added with the `insertItem()` function. With the help of the ampersand character & you can choose a character of your menu that will be shown "underlined". If you press `Alt+F` on the keyboard, for example, you can reach the File menu without using the mouse pointer.

Conclusions

I hope I have convinced you that it is easy to write sophisticated, menu-driven user interfaces with the Qt Class Library. The new versions of **Y290** and **P21** have been in use for over three years, reliably and stable without any problems. Of course, both programs are not finished programs and a lot of improvements are still possible. Also complete documentation has to be written. Internationalization is easy to realize with the Qt Class Library and has to be done, at least for the German users.

If you are interested in this development and/or have an old, mechanically reliable diffractometer, please, contact me via e-mail. I am especially interested in old Syntex/Nicolet (P21 or P3) diffractometers with the so-called "SIEMENS-boxes". It should also be possible to control a Bruker P4 diffractometer that uses GGCS (General Goniometer Control System) commands as an interface between host computer (PC) and the goniometer hardware. Also Huber four circle diffractometers connected to SMC 9000 controller are possible candidates for driving the motors with this software. However, it was not possible for the author to test any of these two controllers under realistic conditions.

Availability

As mentioned earlier, programs **Y290** and **P21** are both free software that can be used without any limitations. The source code and the Linux installation files ("makefile" and parameter-files) can be obtained from the author via e-mail. The adaptations to the new Qt version 3.1.1 are under developement.

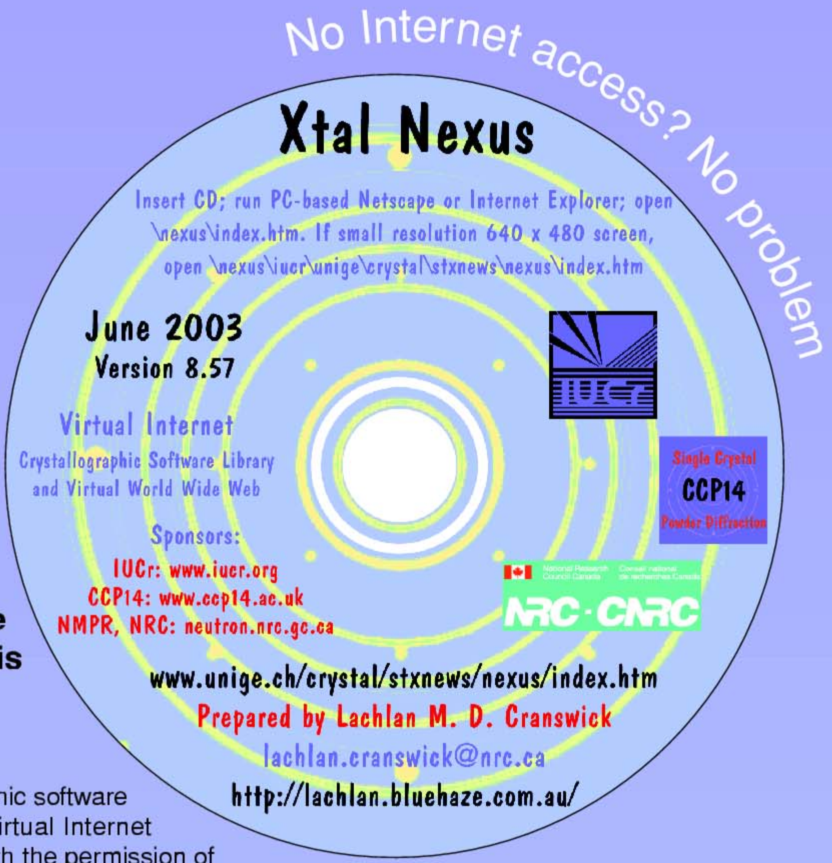
Acknowledgements

I would like to thank Dr. Dirk Abeln for helping with the ATARI version [6] of program **Y290**. He also developed the absorption data collection version of that program which is not yet implemented under Linux.

References

[1] http://www.trolltech.com/
[2] Abeln D., Kopf J., Abstract PS-02.06.01, IUCr XVI, Beijing, China, 1993.
[3] Kopf J., Abeln D., Abstract P21-10, ECM-16, Lund, Sweden, 1995.
[4] Smith B. R., *IUCr Computing Commission Newsletter,* **1**, 63-69, 2003. http://www.iucr.org/iucr-top/comm/ccom/newsletters/2003jan/
[5] Lange J., Burzlaff H., *J. Appl. Cryst.* **24**, 190-194, 1991.
[6] Abeln D., *Dissertation,*University of Hamburg, Germany.

JUNE 2003 EDITION
OUT NOW!


No Internet access? No problem
Xtal Nexus
Insert CD; run PC-based Netscape or Internet Explorer; open \nexus\index.htm. If small resolution 640 x 480 screen, open \nexus\iucr\unige\crystal\stxnews\nexus\index.htm
June 2003
Version 8.57
Virtual Internet
Crystallographic Software Library and Virtual World Wide Web
Sponsors:
IUCr: www.iucr.org
CCP14: www.ccp14.ac.uk
NMPR, NRC: neutron.nrc.gc.ca
IUCr
Single Crystal
CCP14
Powder Diffraction
NRC·CNRC
www.unige.ch/crystal/stxnews/nexus/index.htm
Prepared by Lachlan M. D. Cranswick
lachlan.cranswick@nrc.ca
http://lachlan.bluehaze.com.au/

If you're working in the developing world and have limited or no access to the Internet, then the free Xtal Nexus CD-ROM is here to help.

Packed with a host of crystallographic software packages and resources, these virtual Internet CD-ROMs have been produced with the permission of software and website custodians. The organizer of the Nexus project, the IUCr Committee on Electronic Publishing, Dissemination and Storage of Information, invites academics and students using single-crystal, powder diffraction and other crystallographic analysis methods in developing countries that are isolated from the Internet to register their interest. The CD-ROMs are free of charge and available while stocks last.

The CD-ROM is sponsored by the IUCr, the EPSRC-funded Collaborative Computational Project No. 14 (CCP14) for Single Crystal and Powder Diffraction, and the Neutron Program for Materials Research of the National Research Council of Canada.

Contents

Multiple software packages and suites relevant to single-crystal, powder diffraction, crystallographic structure validation and crystallographic teaching, including

- single-crystal structure solution and refinement suites for PC
- crystallographic structure visualization programs
- powder diffraction pattern visualization, peak profiling and indexing software
- powder diffraction structure solution
- classic and modern crystallographic program source code
- Rietveld refinement packages
- structure validation and checking software
- powder pattern simulation and journal-quality structure plotting and photorealistic rendering programs
- educational websites dealing with crystallographic teaching
- webpage-based tutorials on how to use many of the programs within the CD-ROM
- part of the IUCr website including IUCr Commissions.

Send your request for a free Xtal Nexus CD-ROM (preferably via e-mail or by conventional mail) to

Lachlan M. D. Cranswick
NPMR, NRC
Building 459, Station 18
Chalk River Laboratories, Chalk River
Ontario, Canada, K0J 1J0

E-mail:	lachlan.cranswick@nrc.gc.ca
Tel:	+1 (613) 584-8811 ext. 3719
Fax:	+1 (613) 584-4040
WWW:	http://neutron.nrc.ca/

The CD-ROMs, and regular updates, will be sent by airmail.

Crystallographers join the Crystallography Open Database

COD

www.crystallography.net

Deposit your crystal data in the Public Domain

Thanks !

Computing before computers - The Beevers-Lipson Strips

Robert O. Gould
*Structural Biochemistry, The University of Edinburgh, Swann Bldg, Kings Buildings, Mayfield Road, Edinburgh, EH8 9XD, Scotland. Tel +44 (0)131 650 7058: E-mail: gould@ed.ac.uk ;
WWW: http://www.bch.ed.ac.uk/ and http://www.chem.ed.ac.uk/staff/gould.html*

Sometime in the early 80s, I gave my 12-year old son a little help with something he was doing on a BBC-micro, and got the response, half in wonder, half in annoyance, “But Dads aren’t supposed to know anything about computers!” (Un)fortunately, he had a crystallographer for a father, and we had got into computers even before the children did! The reason isn’t hard to seek. Long before most scientists undertook calculations which were beyond the slide-rule, we were faced with the necessity of doing very long, tedious calculations, which were just waiting for the digital computer. Probably the main one of these was Fourier synthesis, required for relating the measured diffraction pattern to the crystal structure. Today, when Fourier transforms on several thousand data items for large, three dimensional unit cells at a resolution of about 0.3Å require so little time we can’t even gulp down a cup of coffee while they proceed, it is difficult to imagine the size of the task confronting early crystallographers, where this operation was almost unthinkable. After all, even a two dimensional summation of

$$\Sigma \{ \mathrm{A} \cos 2\pi\,(hx+ky) + \mathrm{B} \sin 2\pi\,(hx+ky) \}$$

for 500 data items into 2000 points requires 10^6 summations, let alone any work required to calculate the trigonometric functions! The great contribution of Beevers and Lipson in 1936 was twofold: to simplify the calculations greatly by factorising the trigonometric expressions to reduce the two dimensional calculations to many fewer one dimensional ones, and to provide a convenient technique for carrying out the summations – the strips themselves. The factorisation technique is still at the heart of many computer programs.

The strips themselves were produced for many years by Arnold Beevers in Edinburgh, and were sold in pairs of boxes as shown in the following illustration, one box for sines and one for cosines. As you can see, the strips themselves (4000 of them in the original version) were handsomely presented for convenient use. Each compartment corresponded to a value of h. The original version had strips representing $F\cos 2\pi hx$ and $F\sin 2\pi hx$ corresponding to amplitudes from -99 to 99 and from h = 1 to 20. The strips were placed in order in each compartment, and one of the most important rules was to replace them after use in *exactly* that order!

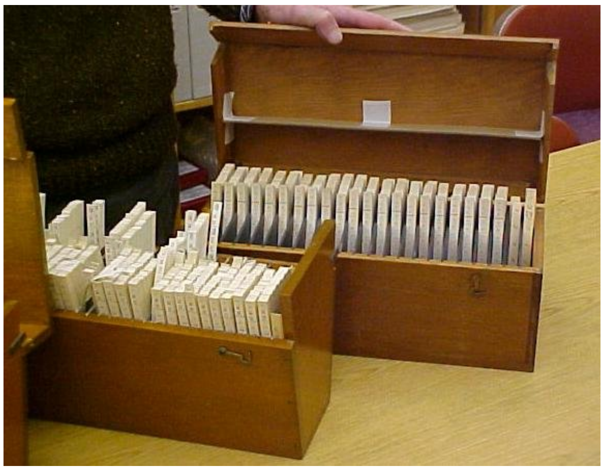


Figure 1: *Boxes of Beevers-Lipson Strips in the laboratory of Hans-Rudolf Wenk, Department of Geology and Geophysics at the University of California Berkeley, USA.*

Figure 2: *Henry Lipson (left) and Arnold Beevers (right) with the author (middle) in 1988.*

In their final form, each cosine strip gave the values of Fcos 2*πhx* for 2*πx* from 0 to 90°, i.e. for *x* from 0 to 0.25. One side (the even side) gave this in intervals of 6° in 2*πx, i.e.* 0.0133 in *x*. The other side (the odd side) gave the intervening values to give an overall resolution of 3° or 0.0067 in *x*. They were produced on foolscap cards using a classical duplicator, which somehow (usually) managed to match front and back sides well. Each card contained 36 strips. They were then carefully guillotined and placed in the appropriate slot. Producing a new edition was a mammoth undertaking. Over 500 sets were sold between 1948 and 1970.

At one time, the boxes were one of the main signs of a Crystallography Laboratory. Their fame was sufficiently great that a photograph of Arnold and Henry in bathing attire labeled "Beevers and Lipson stripped" was a great hit. They did not survive long into the computer age, although at first Ph.D. supervisors tried to get students to do one simple structure using the strips in order to understand the technique better. Predictably, that arrangement did not last long, and the boxes have become rare collectors' items!

The two sides (CE) and (CO) of a sample strip are shown below for : *F* = 19 and *h* = 3. The figure at the right is the checksum, *i.e.* the sum of the values on the strip.

3 CE 19 | 19 18 15 13 6 0 $\bar{6}$ $\overline{13}$ $\overline{15}$ $\overline{18}$ $\overline{19}$ $\overline{18}$ $\overline{15}$ $\overline{13}$ $\bar{6}$ 0 | ($\overline{52}$)

3 CO 19 | 18 16 14 10 3 $\bar{3}$ $\overline{10}$ $\overline{14}$ $\overline{16}$ $\overline{18}$ $\overline{18}$ $\overline{16}$ $\overline{14}$ $\overline{10}$ $\bar{3}$ | ($\overline{61}$)

A simple example of a one-dimensional sum showing how the strips were used is that for iron pyrites, projected onto its a-axis. The space group is *Pa* $\bar{3}$ with *a* = 5.40Å. The structure is centrosymmetric, so only cosine terms need be considered. Only even orders are present for *h*00, and the iron atom at the origin makes it probable that all signs are positive. Hence, using observed amplitudes, the required strips are placed beneath one another, and the columns summed :

2 CE 67	67	65	61	54	45	34	21	7	$\bar{7}$	$\overline{21}$	$\overline{34}$	$\overline{45}$	$\overline{54}$	$\overline{61}$	$\overline{65}$	$\overline{67}$	(0)
4 CE 4	4	4	3	1	0	$\bar{2}$	$\bar{3}$	$\bar{4}$	$\bar{4}$	$\bar{4}$	$\bar{3}$	$\bar{2}$	0	1	3	4	($\bar{2}$)
6 CE 17	17	14	5	$\bar{5}$	$\overline{14}$	$\overline{17}$	$\overline{14}$	$\bar{5}$	5	14	17	14	5	$\bar{5}$	$\overline{14}$	$\overline{17}$	(0)
8 CE 48	48	32	$\bar{5}$	$\overline{38}$	$\overline{47}$	$\overline{24}$	15	44	44	15	$\overline{24}$	$\overline{47}$	$\overline{38}$	$\bar{5}$	32	48	(50)
10 CE 28	28	14	$\overline{14}$	$\overline{28}$	$\overline{14}$	14	28	14	$\overline{14}$	$\overline{28}$	$\overline{14}$	14	28	14	$\overline{14}$	$\overline{28}$	(0)
Sums:	164	129	50	$\overline{16}$	$\overline{30}$	5	47	56	24	$\overline{24}$	$\overline{58}$	$\overline{66}$	$\overline{59}$	$\overline{56}$	$\overline{58}$	$\overline{60}$	(48)

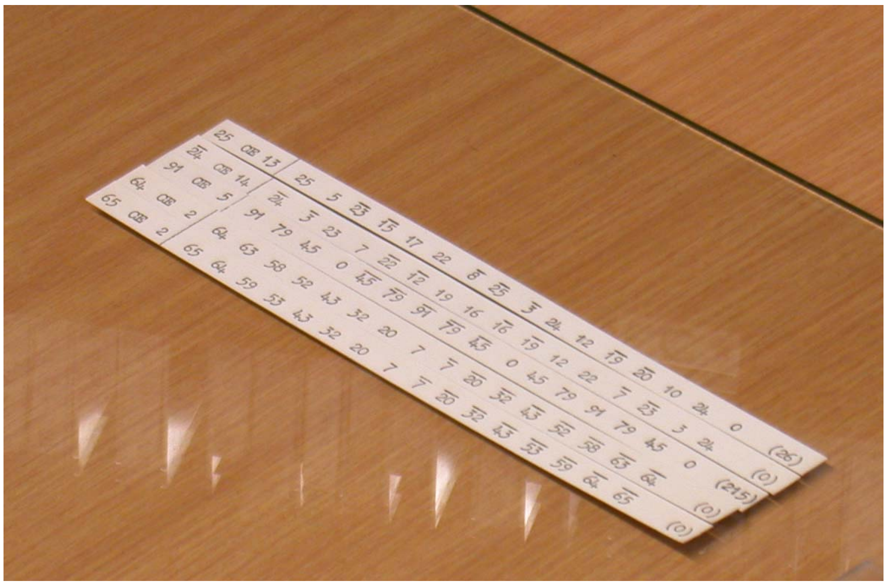


Figure 3: *Image of four Beevers-Lipson Strips. Photographed by Keith Waters at the Chemical Crystallography Laboratory in Oxford, England (October 2002).*

The checksum in the lower right is calculated vertically and horizontally, and the fact that both sums give 48 gives some confidence that the additions are correct! The peak corresponding to the sulfur position is near 41° or *x* = 0.12, which actually corresponds to a position *x,x,x* in the three dimensional structure.

In most cases, it is necessary to extend the sums beyond *x*=0.25. This is done by taking account of properties of the sine and cosine functions. For example odd values of *h,* cos 2π*hx* =
– cos2π*h(½–x*), while for even *h,* cos 2π*hx* = cos2π*h(½–x*). Thus, if the terms with *h* odd and *h* even are added separately, the function between 90°and 180°is obtained by subtracting the odd sum from the even sum and reversing its direction. Similar arrangements enabled both sine and cosine strips to be summed over the range of the unit cell.

The great value of the strips was in doing the much more useful two-dimensional calculations. Considering again the centrosymmetric case, a factorisation is possible:

$$\Sigma\{\ 2F\ (\cos 2\pi(hx+ky)\ \} = \Sigma\ \{\ 2F\ (\cos 2\pi hx \cos 2\pi ky - \sin 2\pi hx \sin 2\pi ky)\}$$

While that may not, at first glance, seem a great advantage, it has reduced the two-dimensional sum to one-dimensional ones. In some cases, the simplification goes much further. For the (rather rare) plane group *p*2*mm*, for example, the fact that the points $x, y;\ \bar{x}, \bar{y};\ x, \bar{y}$ and $\bar{x}, y$ are all equivalent reduces the sum to: Σ { 4F (cos2π*hx* cos2π*ky*)} for all data.

To use the method efficiently, data are sorted so that all reflections with the same value of |*h*| are grouped together, and the one dimensional sum Σ { 4F (cos2π*ky*)} is carried out as above. The resulting row vector is then multiplied by the column vector cos2π*hx* to obtain the values to be contributed to each point *x*, *y* on the grid being considered. A similar operation is then carried out for other values of |*h*|. Most plane groups are a little more complicated than this. For example, the much more common group *p*2*gg*, gives Σ { 4F (cos2π*hx* cos2π*ky*)} when *h*+*k*= 2*n*, but Σ { –4F (sin2π*hx* sin2π*ky*)} otherwise.

Even these two dimensional sums were very time-consuming, and were not undertaken lightly; considerable thought was given to whether the trial structure really needed to be checked with a Fourier synthesis, or could be stretched a little further before the next Fourier was needed. Two things are essential. One is a thorough understanding of the symmetry of the projection and the implications of that symmetry on the Fourier summations. The other was a means of adding the columns rapidly. Some workers did use hand calculators, but most found it quicker to do the sums mentally. Some years ago, a BBC documentary showed Rosalind Franklin with a box of strips laboriously cranking them into a mechanical calculator. This was almost certainly apocryphal. She would not have wasted time on that!

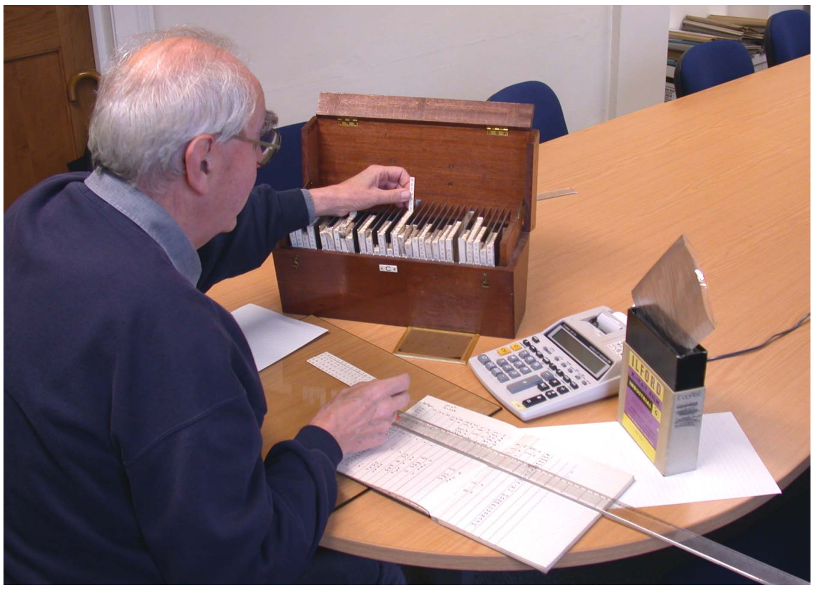


Figure 4: *Photo of Professor Keith Prout at Oxford University demonstrating the setup and use of Beevers-Lipson Strips. Included in the picture is a modern version of that "apocryphal" calculator. Photographed by Keith Waters at the Chemical Crystallography Laboratory in Oxford, England (October 2002).*

Obviously, the method can be extended to three dimensions, and this was sometimes done, particularly for Patterson functions, which are always centrosymmetric, and in principle only have to be carried out once! In the pre-computer age, most structures were refined in two or three projections. The limitation does not apply to computer programs, and the factorization technique has been the backbone of many computer programs for years.

ICSD-for-WWW, a crystallographic database on the WWW

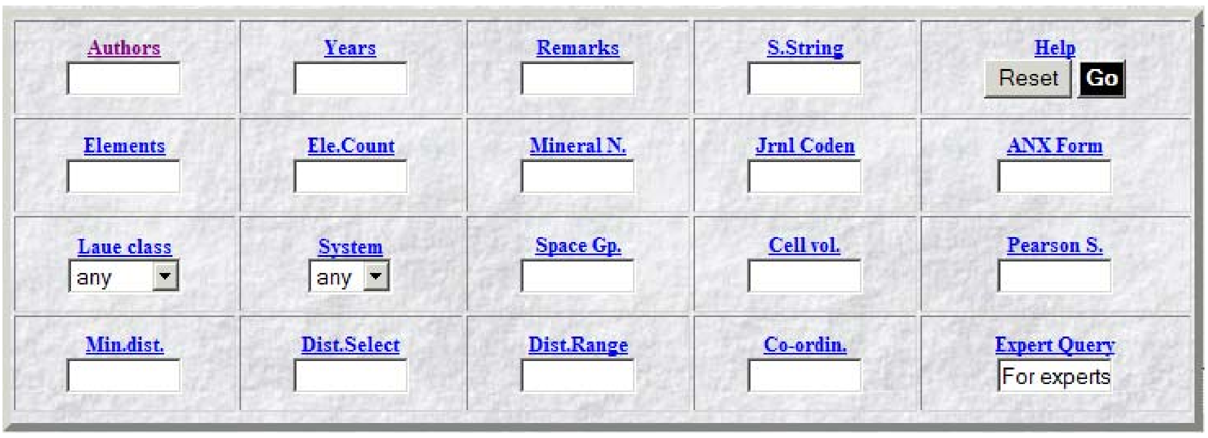
Alan Hewat[1] and Peter Hewat[2],
(1) Diffraction Group Leader, Institut Laue-Langevin (ILL), B.P. 156X Grenoble Cedex 9, FRANCE. Tel: (33) 476.20.72.13; Fax: (33) 476.20.76.48 E-mail: hewat@ill.f rWWW: http://www.ill.fr/dif/ and (2) Institut National de Recherche en Informatique et en Automatique (INRIA) ZIRST - 655 Avenue de l'Europe, 38330 Montbonnot Saint Martin, France. Tel: (33) 6.2381.6283; E-mail: hewat@free.fr WWW: http://hewat.free.fr/

Introduction

Since 1997 the ILL Grenoble has provided a WWW interface to the ICSD (Inorganic Crystal Structure Database), making it simpler to use. ICSD is a product of the Fachinformationszentrum (FIZ) Karlsruhe, a non-profit organisation of the ILL's German Associate offering more than 200 electronic databases in almost every field of science and technology. FIZ already provided internet access to ICSD and other databases via their STN International service, but this was limited to a simple command-line interface. ICSD also existed at that time as a PC programme, with a rather primitive DOS interface.

ICSD now contains over 70,000 entries describing inorganic structures. The rate of adding new entries has greatly accelerated with the growth of electronic publishing, since journals can now submit the contents of their new issue directly to the database. A new database is issued twice a year, and the latest issue contained over 5000 new entries and corrected another 3000 entries. ICSD is not simply a collection of CIF coordinate files, but contains many other data fields containing information about the conditions of the measurement, the results of various tests for possible errors in the structure, links to electronic versions of the original papers and much more. All entries are checked and re-checked, not only for trivial syntactic errors, but more importantly for reasonable symmetry, bond lengths, coordination, temperature factors, valence sums etc.

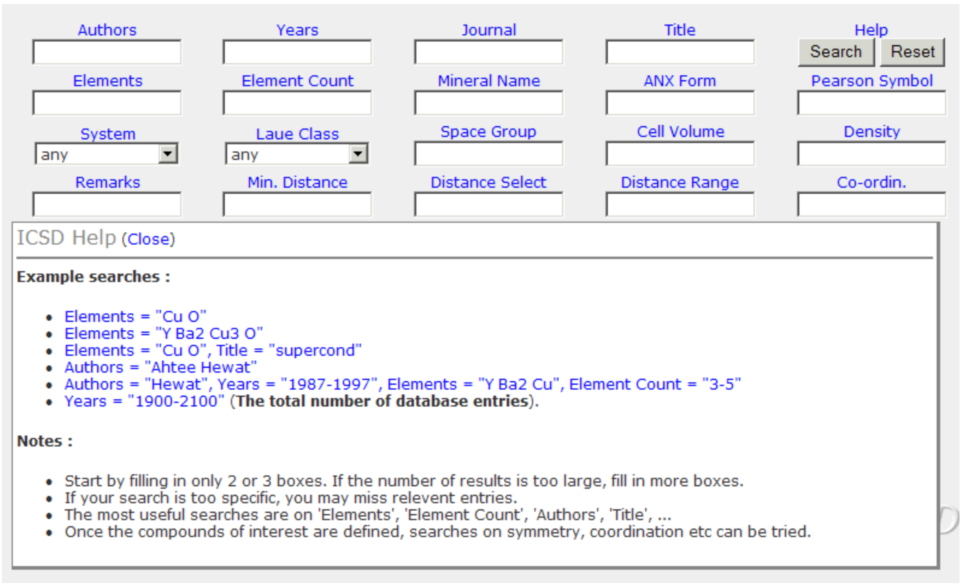
ICSD-for-WWW made it possible for the first time for non-expert users to make complex searches for inorganic crystallographic structures over the internet, without the need to install any special software. Even crystallographers had found it difficult to remember the many commands needed to retrieve information from the old CRYSTIN fortran programme written by R. Hundt & R. Sievers. The WWW-interface generated the CRYSTIN commands from data such as author names and chemical formulae entered into 20 query boxes. Quite complex queries, involving symmetry, bond-lengths and coordination was possible.



Authors	Years	Remarks	S.String	Help Reset Go
Elements	Ele.Count	Mineral N.	Jrnl Coden	ANX Form
Laue class any	System any	Space Gp.	Cell vol.	Pearson S.
Min.dist.	Dist.Select	Dist.Range	Co-ordin.	Expert Query For experts

For some years now, several countries have purchased national licenses for ICSD-for-WWW, allowing any scientist within that country unlimited access to the full database over the WWW. For example the Daresbury laboratory provides a national server for the whole of the UK, which services hundreds of users every week. Similar access is available in the Netherlands, Switzerland, Spain, Taiwan etc. Where national licences are not available, laboratory-wide licences have been purchased by, for example, ILL-ESRF, NIST, Oak Ridge, Los Alamos, Sandia, CalTech etc.

In 2001 it was decided to replace the original database with a modern SQL database, and a completely new version of ICSD-for-WWW was written using PHP-mySQL. The new interface was designed to look familiar, but was improved in many ways. The old interface was written in simple HTML-Javascript, even though much of it was generated on-line by a perl CGI. The new interface is generated using the PHP Hypertext Preprocessor, which allows perl-like PHP instructions to be embedded within HTML. PHP is often associated with the popular mySQL database engine which provides complex relational database queries and runs on all popular computer platforms from linux to Windows and Macintosh.



ICSD Help (Close)

Example searches :

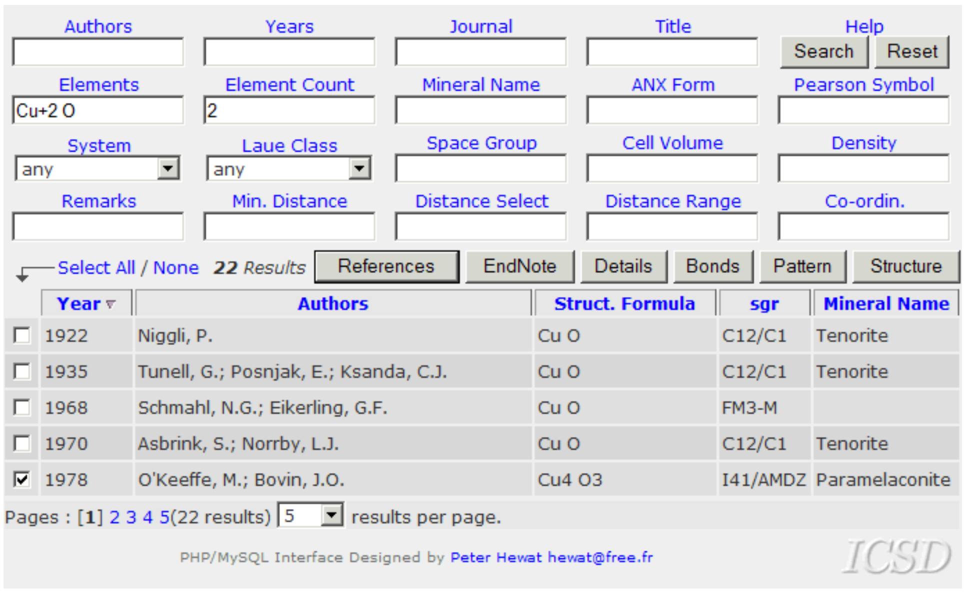
- Elements = "Cu O"
- Elements = "Y Ba2 Cu3 O"
- Elements = "Cu O", Title = "supercond"
- Authors = "Ahtee Hewat"
- Authors = "Hewat", Years = "1987-1997", Elements = "Y Ba2 Cu", Element Count = "3-5"
- Years = "1900-2100" (**The total number of database entries**).

Notes :

- Start by filling in only 2 or 3 boxes. If the number of results is too large, fill in more boxes.
- If your search is too specific, you may miss relevent entries.
- The most useful searches are on 'Elements', 'Element Count', 'Authors', 'Title', ...
- Once the compounds of interest are defined, searches on symmetry, coordination etc can be tried.

Although the new interface looks similar, it is much more powerful, and even easier to use. The titles to the query boxes again contain examples of the query required, but they are now much simpler and drop down beneath the query form when required. To learn to use the interface the user can simply click on these titles, click on an example query, and immediately see the result of that query. If necessary, additional queries can be added and the search repeated.

The user can then order the selected entries according to year, author name, formula, space group or mineral name, select those that appear of interest, and list these references. Direct links to PDF files of the original publication are provided in some cases, notably for IUCr journals. These references can also be exported to the EndNote personal reference database, which will automatically format them appropriate for citing in a particular journal. The *Details* button can be used to compare several entries, while the *Pattern* button produces postscript or PDF plots with [hkl] labels. Finally, the *Structure* button produces interactive 3D structure drawings in VRML or PDB/Chime format. Again entries can be compared by super-imposing diffraction patterns or structure drawings.



Authors	Years	Journal	Title	Help
				Search Reset
Elements	**Element Count**	**Mineral Name**	**ANX Form**	**Pearson Symbol**
Cu+2 O	2			
System	**Laue Class**	**Space Group**	**Cell Volume**	**Density**
any	any			
Remarks	**Min. Distance**	**Distance Select**	**Distance Range**	**Co-ordin.**

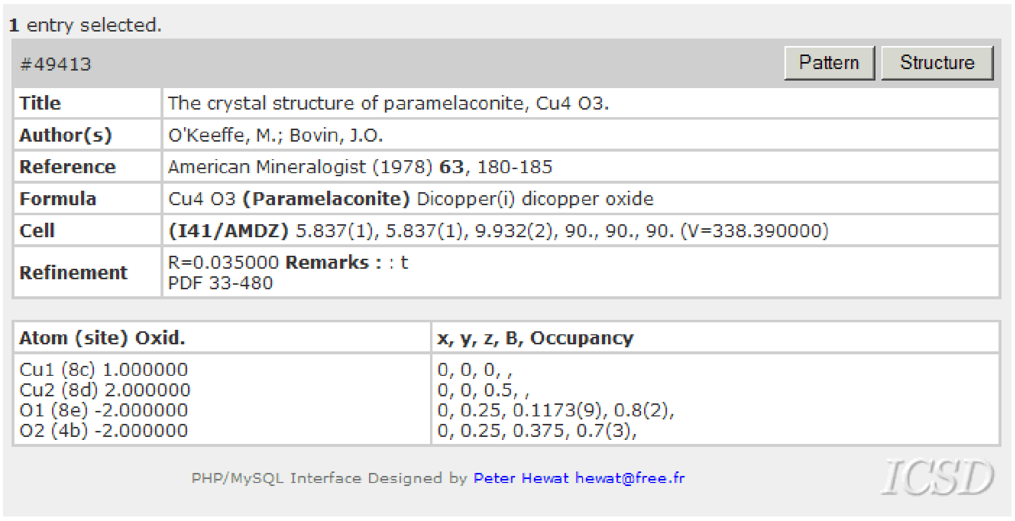
Select All / None *22 Results* References EndNote Details Bonds Pattern Structure

	Year	Authors	Struct. Formula	sgr	Mineral Name
☐	1922	Niggli, P.	Cu O	C12/C1	Tenorite
☐	1935	Tunell, G.; Posnjak, E.; Ksanda, C.J.	Cu O	C12/C1	Tenorite
☐	1968	Schmahl, N.G.; Eikerling, G.F.	Cu O	FM3-M	
☐	1970	Asbrink, S.; Norrby, L.J.	Cu O	C12/C1	Tenorite
☑	1978	O'Keeffe, M.; Bovin, J.O.	Cu4 O3	I41/AMDZ	Paramelaconite

Pages : [1] 2 3 4 5(22 results) 5 results per page.

PHP/MySQL Interface Designed by Peter Hewat hewat@free.fr

In the above example, we found 22 entries for copper oxide containing Cu^{2+}. Most are simply CuO of course, but ICSD also found the interesting compound Cu_4O_3 by O'Keefe and Bovin (1978). The formal copper valence of this compound is 1.5 ! The details below confirm that indeed *Paramelaconite* appears to be a mixed valence compound containing Cu^{1+} on site (8c) and Cu^{2+} on (8d).



1 entry selected.

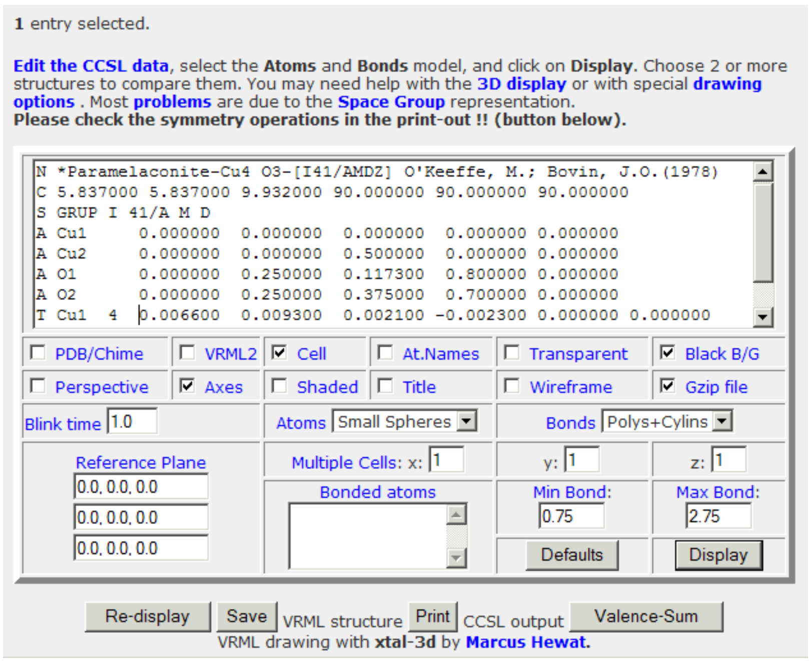
#49413 Pattern Structure

Title	The crystal structure of paramelaconite, Cu4 O3.
Author(s)	O'Keeffe, M.; Bovin, J.O.
Reference	American Mineralogist (1978) **63**, 180-185
Formula	Cu4 O3 **(Paramelaconite)** Dicopper(i) dicopper oxide
Cell	**(I41/AMDZ)** 5.837(1), 5.837(1), 9.932(2), 90., 90., 90. (V=338.390000)
Refinement	R=0.035000 **Remarks :** : t PDF 33-480

Atom (site) Oxid.	x, y, z, B, Occupancy
Cu1 (8c) 1.000000	0, 0, 0, ,
Cu2 (8d) 2.000000	0, 0, 0.5, ,
O1 (8e) -2.000000	0, 0.25, 0.1173(9), 0.8(2),
O2 (4b) -2.000000	0, 0.25, 0.375, 0.7(3),

PHP/MySQL Interface Designed by Peter Hewat hewat@free.fr

To investigate further, we click on the *Structure* button. The drawing of the crystal structure is completely automatic – we do not need to specify which atoms are bonded (although we can if we wish using the *Bonded Atoms* box). We have various other options to draw the structure, but generally we need only click on the *Display* button to obtain a 3D VRML drawing of *Paramelaconite* produced using xtal-3d.



1 entry selected.

Edit the CCSL data, select the **Atoms** and **Bonds** model, and click on **Display**. Choose 2 or more structures to compare them. You may need help with the **3D display** or with special **drawing options** . Most **problems** are due to the **Space Group** representation.
Please check the symmetry operations in the print-out !! (button below).

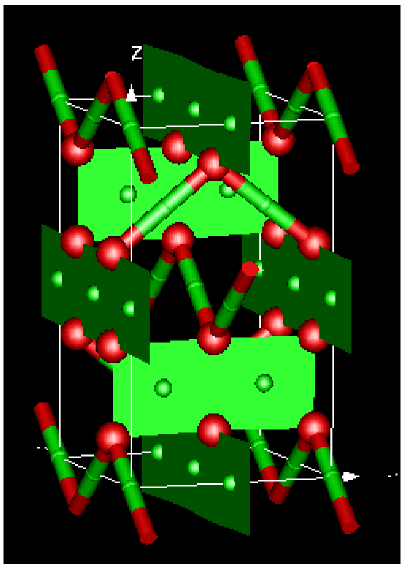
```
N *Paramelaconite-Cu4 O3-[I41/AMDZ] O'Keeffe, M.; Bovin, J.O.(1978)
C 5.837000 5.837000 9.932000 90.000000 90.000000 90.000000
S GRUP I 41/A M D
A Cu1      0.000000  0.000000  0.000000  0.000000 0.000000
A Cu2      0.000000  0.000000  0.500000  0.000000 0.000000
A O1       0.000000  0.250000  0.117300  0.800000 0.000000
A O2       0.000000  0.250000  0.375000  0.700000 0.000000
T Cu1   4  0.006600  0.009300  0.002100 -0.002300 0.000000 0.000000
```

☐ PDB/Chime ☐ VRML2 ☑ Cell ☐ At.Names ☐ Transparent ☑ Black B/G
☐ Perspective ☑ Axes ☐ Shaded ☐ Title ☐ Wireframe ☑ Gzip file

Blink time 1.0 | Atoms Small Spheres | Bonds Polys+Cylins

Reference Plane: 0.0, 0.0, 0.0 / 0.0, 0.0, 0.0 / 0.0, 0.0, 0.0 | Multiple Cells: x: 1 y: 1 z: 1
Bonded atoms | Min Bond: 0.75 | Max Bond: 2.75

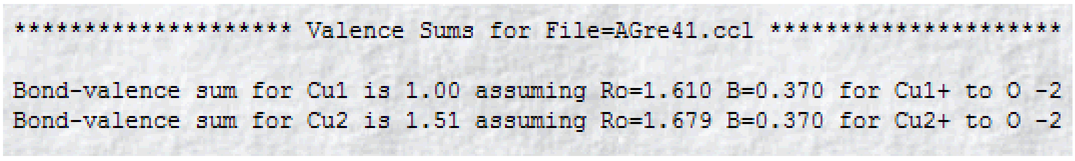
VRML drawing with **xtal-3d** by **Marcus Hewat.**



ICSD shows us that there are indeed two very differently coordinated copper atoms (green) – one at the centre of a square of 4 oxygen atoms (red) and the other with only 2 oxygen neighbours. If we were to select the option to label the atoms we would see that Cu1 had two oxygen neighbours and Cu2 four.

But we can do better ! If we click the *Valence-Sum* button, ICSD displays the Brown-Shannon valence sum for the various Cu-O bonds. We see that the valence sum for Cu1 is indeed 1, while for Cu2 it is 1.51, somewhat less than 2.0. It is not unusual that the valence sum is not a simple integer for all atoms, since coordination is usually a compromise between the relative sizes of the different types of atoms and some bonds are strained. Xtal-3d, which did this calculation, had to automatically decide whether to use the bond-valence parameters Ro and B for Cu^{1+} or for Cu^{2+}; it did this by simply trying both and then finding that Cu2 was closer to Cu^{2+}.

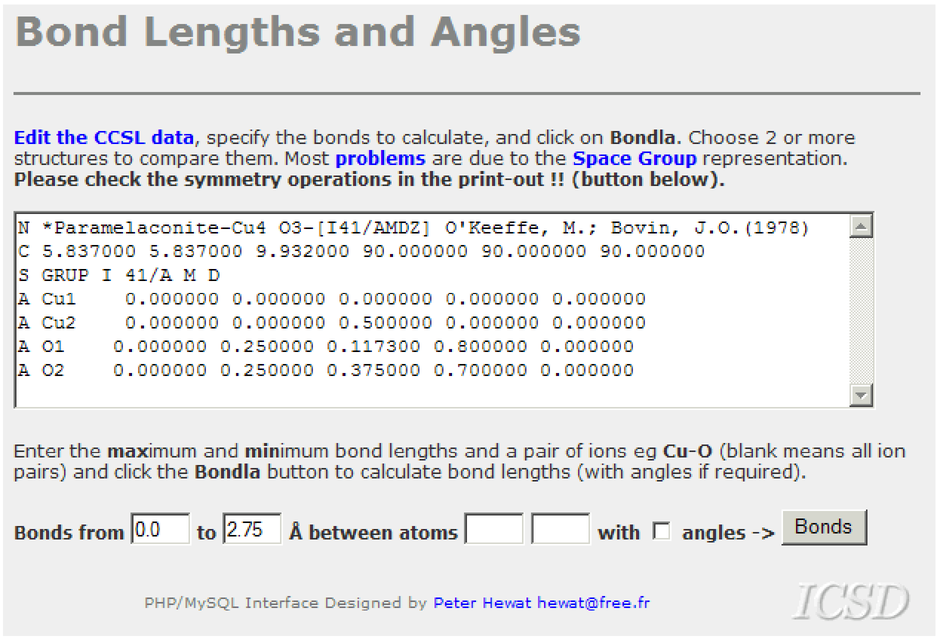
Instead of using xtal-3d, we can generate PDB/Chime format, which displays as a simple space-filling model. In either case we must install the appropriate plug-ins to display on Windows or other computers.



```
******************** Valence Sums for File=AGre41.ccl *********************

Bond-valence sum for Cu1 is 1.00 assuming Ro=1.610 B=0.370 for Cu1+ to O -2
Bond-valence sum for Cu2 is 1.51 assuming Ro=1.679 B=0.370 for Cu2+ to O -2
```

This simple example shows that any scientist, not necessarily a crystallographer, would be able to use ICSD to find compounds containing copper in *mixed-valence* states, and to understand the structure of these compounds with a few clicks of the mouse-button. ICSD is designed not only for crystallographers ! If we have chosen the example of copper, it is because such mixed valence compounds are found as high temperature superconductors, as colossal magneto-resistive magnetites and as many other new materials with very interesting properties. The structures of these materials are often difficult to understand by many non-crystallographers working with them.

But returning to the crystallographers, we can also use ICSD to calculate the bond-lengths (with bond-angles and standard errors if required). Clicking on the *Bonds* button brings up a similar form containing the cell dimensions, space group and atom coordinates. Incidentally, the format used for both structure drawing and bond-length calculations is that of the *Cambridge Crystallographic Subroutine Library*, which is used for these calculations. These forms can of course be edited to see the effect on the structure of adding or removing specific atoms, or displacing them. The list of Cu-O bonds again shows that Cu1 has only two oxygens at 1.87Å, while Cu2 has two at 1.97Å and another two at 1.92Å.



Bond Lengths and Angles

Edit the CCSL data, specify the bonds to calculate, and click on **Bondla**. Choose 2 or more structures to compare them. Most **problems** are due to the **Space Group** representation.
Please check the symmetry operations in the print-out !! (button below).

```
N *Paramelaconite-Cu4 O3-[I41/AMDZ] O'Keeffe, M.; Bovin, J.O.(1978)
C 5.837000 5.837000 9.932000 90.000000 90.000000 90.000000
S GRUP I 41/A M D
A Cu1     0.000000 0.000000 0.000000 0.000000 0.000000
A Cu2     0.000000 0.000000 0.500000 0.000000 0.000000
A O1     0.000000 0.250000 0.117300 0.800000 0.000000
A O2     0.000000 0.250000 0.375000 0.700000 0.000000
```

Enter the **max**imum and **min**imum bond lengths and a pair of ions eg **Cu-O** (blank means all ion pairs) and click the **Bondla** button to calculate bond lengths (with angles if required).

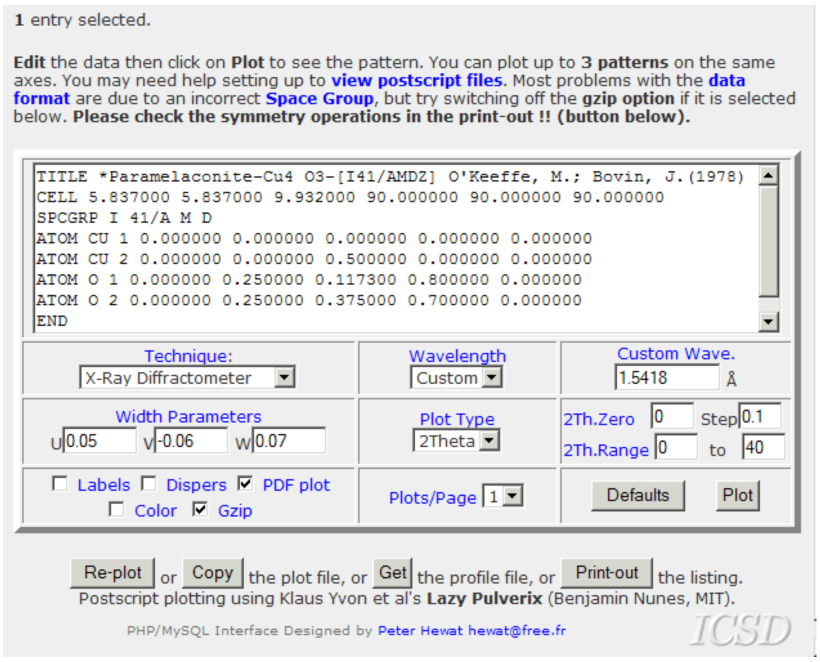
Bonds from 0.0 **to** 2.75 **Å between atoms** [ ] [ ] **with** ☐ **angles ->** Bonds

PHP/MySQL Interface Designed by Peter Hewat hewat@free.fr

Paramelaconite-Cu4 O3-[I41/AMDZ] O'Keeffe, M.; Bovin, J.O.

	Atom1	Atom2	Atom3	Value	Error
Bond	Cu1	O1		1.8673	
Bond	Cu1	O1		1.8673	
Bond	Cu2	O1		1.9663	
Bond	Cu2	O1		1.9663	
Bond	Cu2	O2		1.9159	
Bond	Cu2	O2		1.9159	
Bond	O1	O2		2.5595	

Finally, for the crystallographer, and especially for the Rietveld refiner, ICSD can calculate the powder diffraction pattern for both X-rays and neutrons, or even compare powder patterns for different structures, labelling the [hkl] peaks if required.



1 entry selected.

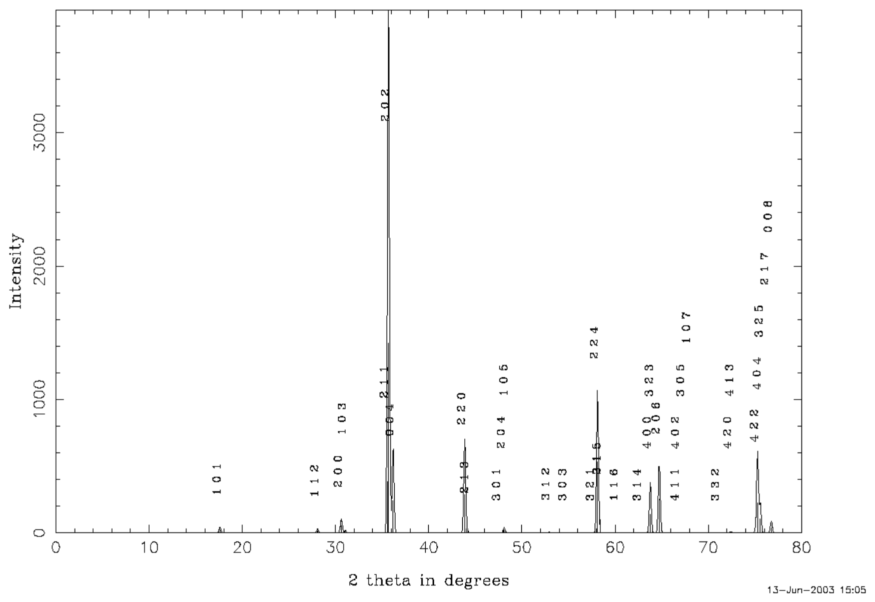
Edit the data then click on **Plot** to see the pattern. You can plot up to **3 patterns** on the same axes. You may need help setting up to **view postscript files**. Most problems with the **data format** are due to an incorrect **Space Group**, but try switching off the **gzip option** if it is selected below. **Please check the symmetry operations in the print-out !! (button below).**

```
TITLE *Paramelaconite-Cu4 O3-[I41/AMDZ] O'Keeffe, M.; Bovin, J.(1978)
CELL 5.837000 5.837000 9.932000 90.000000 90.000000 90.000000
SPCGRP I 41/A M D
ATOM CU 1 0.000000 0.000000 0.000000 0.000000 0.000000
ATOM CU 2 0.000000 0.000000 0.500000 0.000000 0.000000
ATOM O 1 0.000000 0.250000 0.117300 0.800000 0.000000
ATOM O 2 0.000000 0.250000 0.375000 0.700000 0.000000
END
```

Technique: X-Ray Diffractometer
Wavelength: Custom
Custom Wave. 1.5418 Å
Width Parameters U 0.05 V -0.06 W 0.07
Plot Type 2Theta
2Th.Zero 0 Step 0.1
2Th.Range 0 to 40
☐ Labels ☐ Dispers ☑ PDF plot
☐ Color ☑ Gzip
Plots/Page 1
Defaults Plot

Re-plot or Copy the plot file, or Get the profile file, or Print-out the listing.
Postscript plotting using Klaus Yvon et al's **Lazy Pulverix** (Benjamin Nunes, MIT).
PHP/MySQL Interface Designed by Peter Hewat hewat@free.fr
ICSD

*Paramelaconite–Cu4 O3–[I41/AMDZ] O'Keeffe, M.; Bovin, J.(1978)



How easy/hard is to convert raw data into a web database?

Armel Le Bail,
Université du Maine, Laboratoire des Fluorures, CNRS UMR 6010, Avenue O. Messiaen, 72085 Le Mans 9, France - Email : alb@cristal.org ; WWW: http://www.cristal.org/

As a jury member, I recently heard a PhD candidate explaining that, in spite of his efforts, he could not find any solid compound during the exploration of a given chemical system. And I remembered that "I" had explored (not really me, I am a too bad chemist) the same system without success ten years ago... This is certainly a frequent situation in a laboratory where different people repeat infinitely the same unsuccessful stories. Why ? Because we more generally publish only positive results, and because failures are kept in confidential laboratory notes. Europe administrators would like to organize research in order to avoid redundancy and remakes in an increasing number of different nations. We would dream of such an efficiency at the world scale, but it is even not realized between you and your nearest colleague. Any solution ? Yes : knowledge databases, either intranet or internet depending on your willing to share.

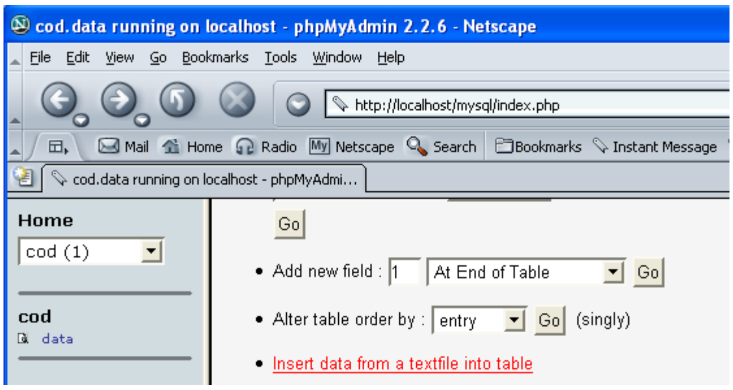
Laboratories have the obligation to produce regularly big reports for their funding agencies or institutions. Invariably these reports present publication lists. Most laboratories have complete lists of publication since their origin. These lists are used internally by staff members for verifying if their current idea has not already been realized by a colleague. Well, you may say that these publications are indexed by some information company (ISI, Science Direct, etc) and that, having access, you just have to make there your bibliography. Not really true because these information companies have started to add the abstracts of articles only ten years ago in their databases, so that for older references, you will only have access to a search by words in titles, and this is too limited if you have a precise or imprecise chemical formula in mind. OK, databases specialized in chemistry and crystallography exist which could give an answer to the question "does that compound exist and was it fully characterized already ?" (but remember that these databases will not answer to all questions of the kind : "do new compounds will appear in these synthesis conditions ?"). So, laboratories, institutions, nations, federations, world(s), have large interest in offering more easy access to knowledge databases.

You may think that this is quite a hard job and can be made only by large scale companies with tens or hundreds of secretaries copying manually titles and abstracts of articles in the paper version of edited journals, writing apart the atomic coordinates, cell parameters, etc. This was true ten years ago, and that old way generated huge quantities of typos. But nowadays, the knowledge is digitized at its source : researchers do not produce paper documents anymore, they produce files which can be printed or transferred to the antipodes in a few seconds without any typo. This concerns not only reports, text ready for publication, but also raw data. No one should have to retype, copy-paste is enough.

Let us take a concrete example and consider how many crystal structures are determined per day in the whole world : close to 100. Quite a small number which would need 50 secretaries in the ancient non-digitized world for building a database. How many people would be really necessary today if things were done in the right way ? I will let you find the answer alone.

Individually, or at the laboratory scale, we can change the crystallography world to an extent which you do not imagine. First, think to the small library in your laboratory where are stored probably a few journals in paper form (*J. Solid State Chem.*, *J. Appl. Cryst.*, *Acta Cryst.*, *Powder Diffraction*, etc) as well as the full collection of all the documents ever published by the lab staff. About that list of lab publications, you remember one about antiphase domains in $KAlF_4$, and would like to verify some detail in the original text. Well, a sample of the publication is there on a shelf, but how to find it fast ? The lab secretary has made a numbered list, printed, but there are hundreds of references in that list (thousands in big labs). If the list exists as a text file, it can be edited, and if the editor software allows you to search for a word, then already this is a minimal database. Typing "KAlF4" or "antiphase", you have chances to find the reference, its number in the list and to have quickly at hands the expected paper. But this will be possible only if there is a computer in the lab library, and no one using it. Much better if you can obtain the document as a PDF without having to move from your office. OK, I know, many scientists working at large scale facilities or in rich labs have access to all they can want already. Not exactly my case... Not your case too ? Then let us continue. It was even less my case five years ago, and so I decided to make a small database of my lab publications. It was promptly installed on the Internet [1], built up from a simple script in PERL language which was searching in an ASCII file (a table) containing the different parts (fields) of the bibliographic references (authors, title, journal name, year, issue, pages) separated by a group of 3 characters (";"), nothing difficult. Since that day, I could obtain fast the reference number of a paper on the shelf in the lab library. A tutorial explaining how to build such a small database was installed online as well [2]. But nowadays most authors obtain a PDF of their manuscript after publication (or can build it from their own manuscript) and are generally allowed by the publisher to make it available on an intranet. So that this previous database could be improved by the possibility to provide online the document corresponding to the query. Many people are already living in that ideal world for a researcher, having almost immediate access to everything. Unfortunately, many more researchers have no access at all, either in developing countries or in small labs with limited budgets, or they have access only to a small part because of the fragmented nature of the knowledge (for instance, the crystal structure atomic coordinates are in 5 distinct databases of which 3 cannot be accessed unless paying fees [3], and even the Powder Diffraction File, an invaluable tool for phase identification, is now fragmented).

The next desire in a crystallography lab could be to have not only the database of the lab published papers references together with the online access to the digitized text, but also to be able to perform more specialized searches (on the chemical formula, the cell parameters, cell volume, etc). Informations attain the official crystallography databases with some delay. When people in different teams in a lab are working on similar subjects, they can synthesize the same compound and undertake to determine its structure twice, almost simultaneously. This never happens in your lab ? It does happened in mine. So, an intranet database of the latest lab productions would be of the highest interest in the four-circle diffractometer room, together with the access to CSD, ICSD, CRYSTMET, PDB and NDB. Can you build easily, and at low cost, a lab database of crystal structures with similar performances (at least able to answer to the question : is that crystal structure already determined by someone in my lab ?) as those of the above big products ? The answer is now yes since the birth of the Crystallography Open Database [4] based on open source software Apache/MySQL/PHP (available in the EasyPHP package [5]). You can get it for free, you can use its structure for the building of your own intranet (or internet if you wish) lab crystallography database. Moreover, the COD international advisory board expects that doing so, you will also allow free access to your lab crystal data. That way the COD would grow as a major free service to the international community of crystallographers. Remember crystal data books ? The minimum information for building a useful small crystallography database is to add the chemical formula sum, the cell parameters and the space group to each structure paper of your lab list of publication references. A little more work and you build a CIF by adding atomic coordinates (the COD provides a REF2CIF software). The Apache server can run either in standalone or in server mode. The MySQL part provides administration tools allowing to modify the structure with ease (add or remove tables or fields) and to add new entries simply from textfiles (fig. 1).


cod.data running on localhost - phpMyAdmin 2.2.6 - Netscape

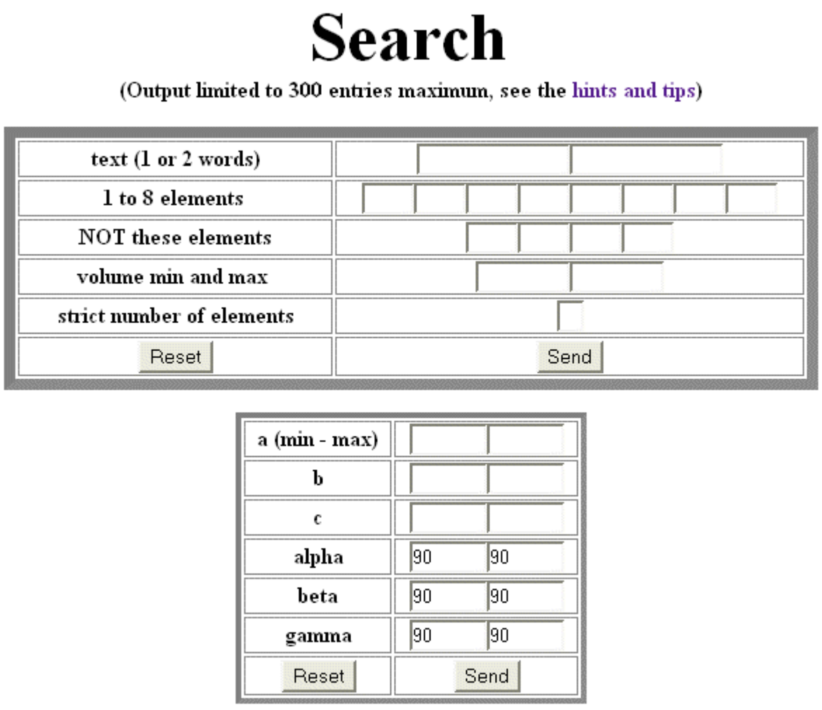
http://localhost/mysql/index.php

- Add new field : 1 At End of Table Go
- Alter table order by : entry Go (singly)
- Insert data from a textfile into table

Fig. 1: *Very small part of the possibilities offered by the phpMyAdmin tool.*

The PHP language is highly efficient, more intuitive and understandable than the PERL language (some say maliciously that PHP means People Hate PERL). The querying system in fig. 2 is done inside of a 150 lines PHP script. The user can combine in any logical way: text, cell volume, chemical elements and strict number of elements. Adding in the script that the request will include a volume search is made by a simple line of PHP code :

```
$requete=$requete . "vol BETWEEN $vmin and $vmax ";
```



Search

(Output limited to 300 entries maximum, see the hints and tips)

text (1 or 2 words)	
1 to 8 elements	
NOT these elements	
volume min and max	
strict number of elements	
Reset	Send

a (min - max)		
b		
c		
alpha	90	90
beta	90	90
gamma	90	90
Reset	Send	

Fig. 2 - The simple COD search form.

Currently, the COD contains 12.000 entries corresponding either to CIF or REF files. The COD is only a 4 months baby so that it may be certainly considerably improved. However, the average user’s simple needs should not be forgotten. A light and user-friendly search form will be preferred to an overloaded one. More demanding users will find satisfaction later, hopefully.

Finally, a database of unfruitful chemical synthesis attempts is a next step on the "to do list", maybe, quite interesting as well ;-).

[1] Fluoride Lab publications : http://sdpd.univ-lemans.fr/perl/pubfluo.html
[2] How to install a small database online (1998) : http://sdpd.univ-lemans.fr/perl/howto.html
[3] Crystallographic Databases, special issue of Acta Crystallographica, Ed : F. Allen (2002)
[4] Crystallography Open Database (2003) : http://www.crystallography.net/
[5] EasyPHP : http://www.easyphp.org/

Automated Data Collection & Integration; Recent Changes to Mosflm

Harold R. Powell,
MRC-LMB, Cambridge, UK, CB2 2QH. E-mail: harry@mrc-lmb.cam.ac.uk ;
WWW: http://www.mrc-lmb.cam.ac.uk/harry/

The program is available from http://www.mrc-lmb.cam.ac.uk/harry/mosflm/, where further information may also be found.

Mosflm [1] is a well-established program for integrating film and area detector images, and is optimized for processing images collected from macromolecular crystals with relatively large unit cells. It uses the method of two-dimensional integration, so it is ideally suited to processing wide-phi sliced data (i.e. most reflxections are fully recorded on single images). However, it is also capable of accurately measuring diffraction data which has been collected using fine-phi slicing (i.e. essentially all reflections are present as partially recorded spots over several images).

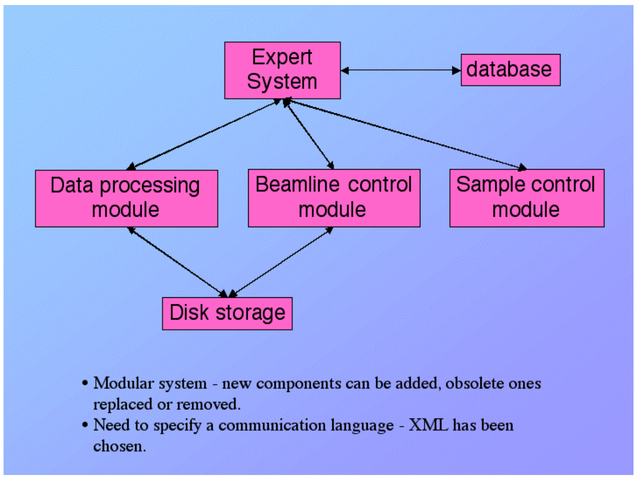
In order to process data as efficiently as possible, many of the integration parameters are optimized by the program without user intervention. It is important to stress that these are not (in the main) default values; they are determined by Mosflm following analysis of the images. Examples of these are the measurement boxes, spot profiles and tests for convergence of refinement. All of the automatically determined values may be over-ridden by the user, and while this may be necessary occasionally, it is rarely done even by experienced users. Much careful thought has gone into implementing the necessary analyses.

As data collection becomes faster at synchrotron sources, the need for automating the processes involved becomes more important. A full data collection experiment can be completed in well under an hour in many cases at a third generation synchrotron. Robotised transfer of crystals onto the goniometer allows a large number of crystals to be examined with minimal user intervention.

While it may be reasonable to expect a user to devise an appropriate experimental protocol if they are collecting five or six datasets a day, if they are collecting 20 or 30 this becomes tedious and prone to error. One option is to have a standard method of data collection and hope that it works sufficiently well for all cases. This can be improved by making small changes to fit the individual cases better. However, by far the best option is to treat each crystal as an individual and optimise all data collection and processing parameters to suit.

Recently, the authors of Mosflm have been involved with beamline personnel at the ESRF (Grenoble) and the SRS (Daresbury) in a project to automate data collection and subsequent processing, with a view to aiding high-throughput protein crystallography. This collaboration is known as the DNA project, for reasons which it is unnecessary to discuss here. Further details of the various contributors and progress can be found at *http://www.dna.ac.uk/* ; those interested in taking part should first consult the documents contained therein. The central part of the software in DNA is an Expert System which makes decisions about data collection and data processing, based on information provided by users and other programs with which it communicates.

One prerequisite of the DNA project is the need to streamline and further automate parts of Mosflm. We have decided to add functionality so that decisions can be made by the program in a way that is robust and as error-free as possible. In other words, the decisions that Mosflm makes should be similar to those made by an expert user. In particular, it is the early part of the process which needs to be addressed, e.g., which autoindexing solution to choose, or how to determine an appropriate value for the mosaic spread before integration. Other questions also arise, e.g. how do we determine the correct exposure time for a given crystal, or the effective resolution limit of the data? This requires a somewhat more involved analysis, and the results are not yet included in Mosflm; however, they have been addressed by Sasha Popov of EMBL in Hamburg, and his approach is likely to be used in future versions [2].

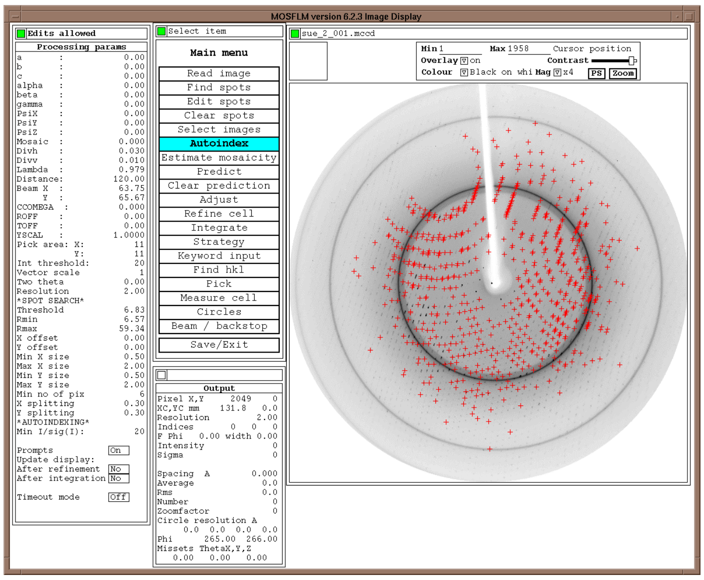


- Modular system - new components can be added, obsolete ones replaced or removed.
- Need to specify a communication language - XML has been chosen.

Fig 1: *Hierarchy of the DNA project showing that it is composed of separate modules which communicate with each other.*

It makes sense to outline the facilities beyond data integration which pre-exist in Mosflm; it includes routines for spot finding, autoindexing, post-refinement of crystal and detector parameters and for data collection strategy.

The spot finding routine is used to provide the autoindexing with a list of diffraction peaks for use in determining the unit cell dimensions of the crystal. Autoindexing, as the name implies, indexes the spot list automatically. Post-refinement minimizes a residual involving the recorded intensities of partially recorded reflections on adjacent images, and has been discussed in detail elsewhere [3]. It is called "post"-refinement because it can only be done after the integration of some images. Although this requires initial parameters to be reasonably close to their final values, because it effectively decouples the cell refinement from the crystal to detector distance; this is not the case if a residual based on spot positions were to be used. For the lower resolution data (where there may be a strong correlation between the apparent cell dimensions and the detector distance) typical of protein datasets, it provides a reliable means of obtaining accurate cell parameters. In Mosflm, it is performed before the main integration run through the data, usually using one or two segments of images widely separated in phi.



```
Edits allowed
Processing params
a         :           0.00
b         :           0.00
c         :           0.00
alpha     :           0.00
beta      :           0.00
gamma     :           0.00
PsiX      :           0.00
PsiY      :           0.00
PsiZ      :           0.00
Mosaic    :          0.000
Divh      :          0.030
Divv      :          0.010
Lambda    :          0.979
Distance:           120.00
Beam X    :          63.75
     Y    :          65.67
CCOMEGA   :          0.000
ROFF      :           0.00
TOFF      :           0.00
YSCAL     :         1.0000
Pick area: X:           11
           Y:           11
Int threshold:          20
Vector scale             1
Two theta             0.00
Resolution            2.00
*SPOT SEARCH*
Threshold             6.83
Rmin                  6.57
Rmax                 59.34
X offset              0.00
Y offset              0.00
Min X size            0.50
Max X size            2.00
Min Y size            0.50
Max Y size            2.00
Min no of pix            6
X splitting           0.30
Y splitting           0.30
*AUTOINDEXING*
Min I/sig(I):           20

Prompts             On
Update display:
After refinement    No
After integration   No

Timeout mode        Off
```

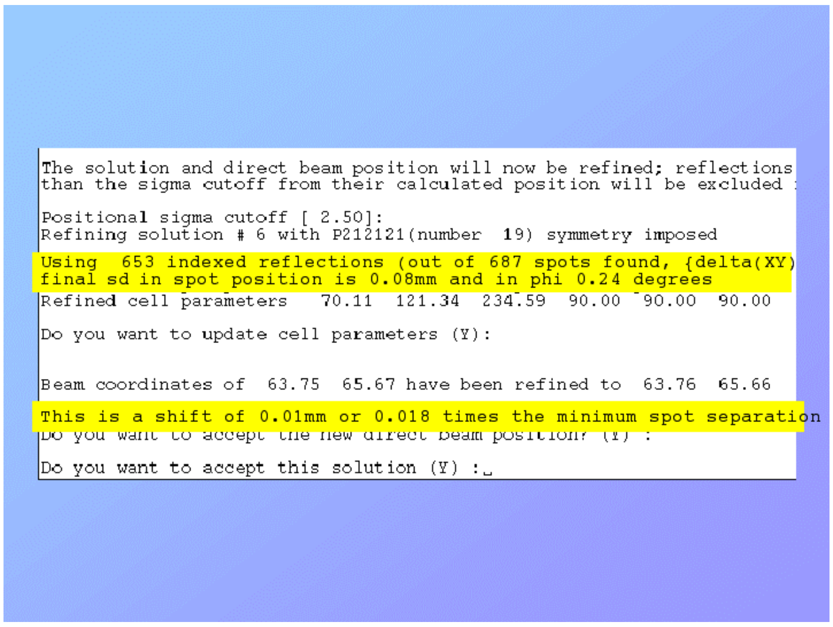
```
Output
Pixel X,Y      2049     0
XC,YC mm      131.8   0.0
Resolution           2.00
Indices         0   0   0
F Phi   0.00 width 0.00
Intensity               0
Sigma                   0

Spacing  A          0.000
Average               0.0
Rms                   0.0
Number                  0
Zoomfactor              0
Circle resolution A
    0.0  0.0  0.0  0.0
Phi       265.00  266.00
Missets ThetaX,Y,Z
   0.00   0.00   0.00
```

```
sue_2_001.mccd
Min 1          Max 1958       Cursor position
Overlay on                    Contrast
Colour Black on whi  Mag x4
```

Fig 2: *Spots found on an image which have I/sig(I) greater than the autoindexing threshold (in this case, I/sig(I) > 20).*

For many years, Mosflm has included code to calculate the most efficient data collection strategy once the crystal symmetry, cell dimensions and orientation are known. Options are available for optimizing data collections with either overall or anomalous completeness (e.g. for MAD or SAD experiments), as well as for collecting data in more than one segment. This latter can be extremely useful in two instances in particular. First, if a partial dataset has been collected from another crystal (or crystals) in the batch, only the missing data need be collected and therefore this strategy determined. Second, it is often possible to collect a complete dataset with two segments which require a smaller total oscillation range than would be required for a single segment. This may be useful if there is insufficient time to collect a full single segment dataset.

Of these existing functions provided by Mosflm, it is the autoindexing step which requires most user intervention. Although the indexing itself is highly automated, choosing a solution with the correct symmetry is more difficult. Following indexing, it is also important to make a reasonable estimate of the crystal's mosaic spread; a feature which performs this task has been added recently to Mosflm.



```
The solution and direct beam position will now be refined; reflections
than the sigma cutoff from their calculated position will be excluded

Positional sigma cutoff [ 2.50]:
Refining solution # 6 with P212121(number  19) symmetry imposed

Using  653 indexed reflections (out of 687 spots found, {delta(XY)
final sd in spot position is 0.08mm and in phi 0.24 degrees
Refined cell parameters   70.11  121.34  234.59  90.00  90.00  90.00

Do you want to update cell parameters (Y):


Beam coordinates of  63.75  65.67 have been refined to  63.76  65.66

This is a shift of 0.01mm or 0.018 times the minimum spot separation
Do you want to accept the new direct beam position? (Y) :

Do you want to accept this solution (Y) :
```

Fig 3: *Highlighting parts of the autoindexing output which show useful criteria for assessing the quality of a chosen solution.*

It seems reasonable to start by asking "how does an expert crystallographer approach these problems?". The following scenario might be proposed;

The first step is normally to expose a single image (either a still or a small oscillation) and visually inspect these for the presence of diffraction spots and check whether they were split or smeared out, or otherwise imperfect. Individual lunes of spots should be seen.

An expert would then probably autoindex this image and choose the appropriate lattice symmetry from those suggested by the program. Overlaying a prediction of the spot positions would indicate that a suitable solution had been chosen. It would also be sensible at this point to estimate the mosaicity of the crystal, by predicting spots using different values for the mosaicity, and deciding when all spots were being successfully predicted. These steps will be discussed further here.

This might be repeated for a second image 90 degrees away in phi, and checked by using both images, to make sure there was three dimensional consistency in the results.

Estimating a value for the mosaicity, should be straightforward - an expert would manually increase the value, and overlay predictions until measurement boxes covered all and there were none in places with no spots.

Parts of this are actually rather difficult to automate; while it is trivial for a user to detect split spots or separate lunes, neither is an easy task for the programmer to implement. In the first instance, therefore, it is necessary to simplify the process and decide how to determine crystal quality by measurable quantities.

In the DNA project, some parts are performed by Mosflm and some by the other components. Here, only those parts implemented in Mosflm will be examined.

An initial criterion for crystal quality is the failure or success of autoindexing. The usual reasons for the failure of autoindexing are;

- Incorrect physical parameters (beam centre, wavelength, distance)
- Insufficient spots (implies weak diffraction)
- Multiple spots or lattices
- Excessive mosaic spread
- Algorithm failure

Of these, the last (algorithm failure) is the least common these days. Direct space autoindexing routines based on FFTs are used in the most popular integration programs because they are so reliable. A harder problem is encountered when we come to choose the right symmetry. All modern indexing programs give some kind of figure of merit for solutions with more than triclinic symmetry; while it is easy for an experienced crystallographer to make a suitable choice to start off with, it is much harder to quantify exactly what they are doing.

It can be instructive to have a look at how picking a solution would be done by an expert user.

In this example using Mosflm, spots are found on a single image and then those with I/s(I) > 20 are used for indexing (the red crosses on the image mark the spots which fulfil this criterion).

Mosflm provides a list of the 44 possible characteristic lattices, sorted into order of decreasing penalty, each with the possible chiral space groups for that symmetry. Only the solutions with penalties < 200 are reported in the GUI; all are shown in the launch window and the logfile output. At this stage the crystal symmetry is not imposed, and the deviations from the ideal values can be a useful indication as to the correct choice.

The important point to emphasize here is that, while it is easy for a user to pick a sensible answer, if the process is to be automated, it may be necessary to quantify the results and impose a threshold beyond which the result would not be reasonable.

On examining the top part of the output, it can be seen that the penalties may be divided into groups (0 - 9, 63 - 77 and 131 and above). To the experienced user, a penalty below 10 represents a particularly good solution, while one above about 25 would normally be unreasonable, so a cut-off can be applied. In most cases, if a unit cell has metric orthorhombic symmetry, then the true symmetry is also orthorhombic (but beware; this is not always true!). For data collection purposes, it is sensible to ensure that the dataset is as complete as possible before other considerations (e.g. multiplicity) are taken into account, so the the highest symmetry solution would be chosen from those in the low penalty group, and then to proceed on that basis until or unless another path were to be indicated.

The most obvious change which can be detected by a program is a sudden change in successive penalties. However, the ratio of successive penalties is actually a better diagnostic, and this can be used to make the selection of the low penalty group relatively reliable. Mosflm preselects only those solutions with a penalty below 100; as good solutions are most likely to have penalties < 25, this provides a wide margin for error.

Having done this, Mosflm determines if there is an obvious large step between solutions by determining the maximum value of the following quotient;

$$\frac{penalty(N+1) - penalty(N)}{penalty(N) + 1}$$

All good solutions should appear in the list below solution N. In practice, it is remarkable how often this simple method works; the "correct" high symmetry solution is chosen ~98% of the time.

Following the choice of a solution, the correct metric symmetry is imposed and the solution refined. This provides a number of diagnostics which allow the user (or the DNA Expert System) to determine whether this is truly a realistic solution or not. There are no plans at present to include tests for the quality of these indicators in Mosflm.

(1) The proportion of spots used by the autoindexing which are also used in the refinement, and have not been rejected because they are too far from the predicted positions. Normally, around 90% of the available spots should be used at least - often many more.

(2) The final SD in spot positions. For a well set up beam line with a good crystal, this is normally around half the pixel size - e.g. for an ADSC Quantum 4 detector, values around 0.04mm should indicate high confidence of success. Values of approximately twice this are probably still acceptable, but even larger values usually indicate problems either with the crystal itself or the solution which has been chosen.

(3) The SD in phi should be around one-quarter of the oscillation angle for a crystal with low mosaicity. Much larger values indicate a problem!

(4) The direct beam position should not be refined through large distances; a change of more than about a quarter of the minimum spot separation for any cell is an indication that the initial position or the solution was incorrect. If the shift is greater than half the spot separation, the solution is almost certainly wrong.

Of course, if fewer symmetry constraints are applied, the residual will be lower and the proportion of spots used in refinement will be larger.

Having chosen the solution from autoindexing, the expert will then make an estimate of the mosaic spread of the crystal. This would normally be done by picking a value, predicting the spot positions and checking to see whether all predictions cover real spots, and vice-versa. Usually, the user will reach a compromise where nearly all spots are predicted or there are few predictions where there are no apparent spots. Changing the upper displayed limit for spot intensities can display spots which would otherwise be hidden in the background.

It is important to give any integration program a reasonably good estimate for the mosaic spread so that integration boxes can be placed around all existing spots and areas of background noise are not integrated.

The approach taken in Mosflm to estimate the mosaicity is to integrate the image with different values of the mosaic spread. The working hypothesis is that the total intensity (for all predicted reflections) will reach a maximum at the correct value. If we plot mosaicity against the normalized total intensity for a real case, we find that the slope rises steeply then tails off gradually before becoming more or less flat. If the value of mosaicity is increased further, it is likely that the curve will start to fall, as new spots overlap each other and are rejected from the integration. In the best cases, where the mosaic spread is low and there is little diffuse scatter, the result is a graph with two more-or-less straight lines, and the intercept represents the mosaicity. In most cases, the drop-off is more gradual; the solution is to take the value corresponding to 80% of the extra intensity over the value for mosaicity of 0º.

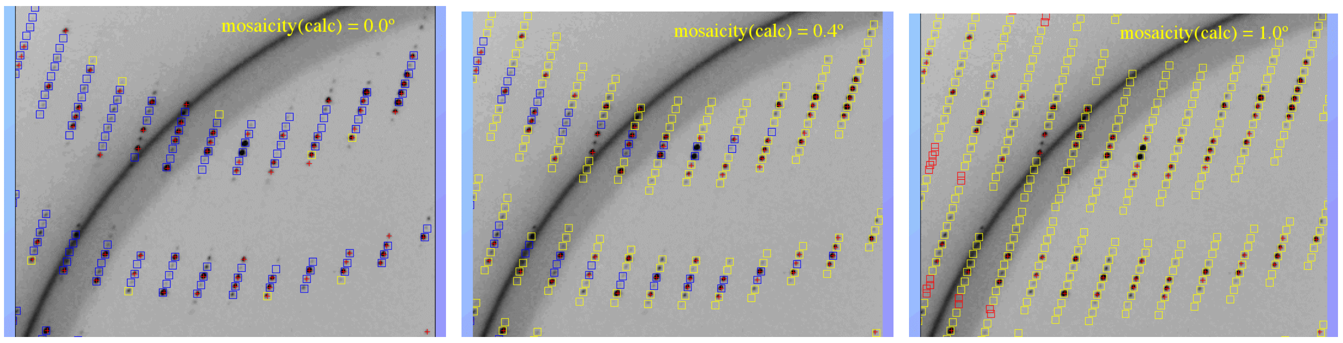


Fig 4a-c: *Predicted measurement boxes for three different values of the mosaic spread a) 0.0°, b) 0.4° and c) 1.0°; illustrating the increasing number of boxes, the increasing proportion of partially recorded spots (yellow - fulls are blue), and the eventual appearance of overlapped spots.*

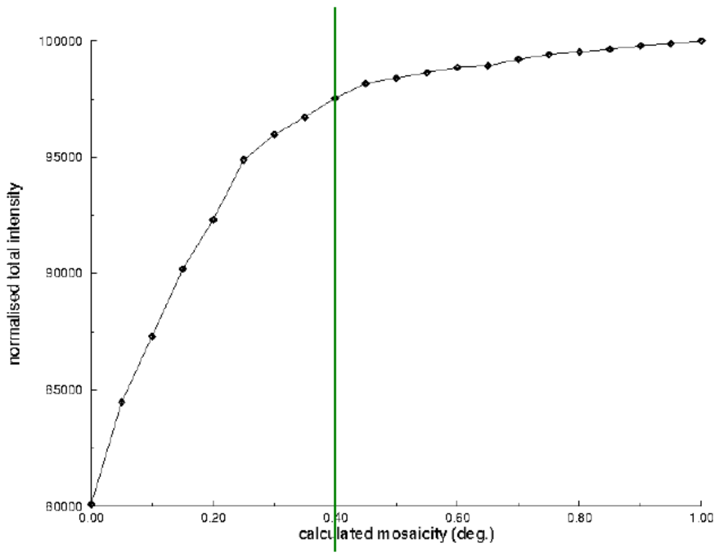


Fig 5: *Plot of normalized intensity against mosaicity with the value estimated by Mosflm indicated.*

References

[1] Leslie, A.G.W., (1992), Joint CCP4 + ESF-EAMCB Newsletter on Protein Crystallography, No. 26.
[2] http://www-hasylab.desy.de/science/annual_reports/2002_report/part2/intern/8482.pdf
[3] Vriend, G., Rossmann, M.G., Arnold, E., Luo, M., Griffith, J.P. and Moffat, K. (1986), J. Appl. Cryst., 19, 134 – 139.

Overview of powder-indexing program algorithms (history and strengths and weaknesses)

Robin Shirley,
School of Human Sciences, University of Surrey, Guildford, Surrey GU2 7XH, U.K. E-mail: R.Shirley@surrey.ac.uk ; WWW: http://www.surrey.ac.uk/Psychology/staff/RShirley.htm and http://www.ccp14.ac.uk/tutorial/crys/

Abstract

An attempt has been made within the space available to give an overview of powder indexing, including its history and the algorithms and programs that have been used. The main methods in use are briefly described, with a summary of their strengths and weaknesses (mostly in table form). Some recently released software is also discussed, plus notes on anticipated developments.

Introduction

Powder indexing is the *ab initio* determination of previously unknown (at least for that specimen) unit cell constants from powder diffraction data. It is sometimes called "autoindexing", a less specific and arguably less helpful name, since the term "indexing" is used in other contexts, and powder-indexing methods should not be treated as fully automatic but acknowledged to require some human judgement.

Powder indexing is a gateway technology, in the sense that the unit cell must be determined before one can proceed to the powerful techniques now available for solving and refining structures from powder data. Only when the cell is known can indices be assigned to their appropriate profile components and the corresponding diffracted intensity located correctly in reciprocal space. So, unless the pattern from a powder phase can be indexed, further structural work is blocked.

This process is called "indexing" because, leaving aside any instrumental constants, determining the cell constants (in a continuous parameter space with 6 mathematical dimensions) is equivalent to assigning the appropriate triple of Miller indices to each of the N_{obs} observed powder lines (in a $3N_{obs}$-dimensional integer-valued index space). Hence one way to classify indexing programs and algorithms is whether they operate primarily in parameter space or in index space. The two spaces are related by the basic indexing equation, in which quadratic forms Q (of d^*) have been used to produce a linear relation in the powder constants Q_{A-F}:

$$Q_{hkl} = h^2Q_A + K^2Q_B + l^2Q_C + klQ_D + lhQ_E + hkQ_F$$

where $Q = d^{*2} = 1/d^2$, $Q_A = a^{*2}$, $Q_D = b^*c^*\cos\alpha^*$, and the remaining terms obtained by cyclic permutation of reciprocal sides and angles. Q should not be confused with q (= d*) as used in energy-dispersive studies. Q-values are typically less than 0.01 A^{-2}, and so for convenience are often scaled up by a factor of 10000 to give working "Q-units" (QU.).

Some history

The first powder-indexing method was published almost at the dawn of powder diffraction (Runge, 1917). Runge's remarkable paper was symmetry-general and greatly ahead of its time, using what later became known as Ito's method or zone-indexing. This is based on the decomposition of the general 6-constant indexing equation into three 3-constant "zone" equations of the form:

$$Q_{hk0} = h^2Q_A + k^2Q_B + hkQ_F$$

each of which describes a central powder zone - a plane passing through the origin of reciprocal space and containing two of the principal lattice vectors (in this case the [001] or hk0 zone, containing $\mathbf{a}^*$ and $\mathbf{b}^*$). If quartets of related low-angle powder lines near the origin such as 100, 010, -110 and 110 can be identified, the three associated powder constants Q_A, Q_B and Q_F can then be determined through simple relations such as $Q_F = 0.5(Q_{110} - Q_{-110})$.

For nearly fifty years, indexing remained an exclusively manual operation, using a number of mainly graphical methods for high-symmetry cases (i.e. tetragonal and above). Hesse (1948) and Lipson (1949) attempted to extend this to orthorhombic cases by inferring cell constants from recurrences in $\sin^2\theta$ difference tables. Unfortunately in inexperienced hands the Hesse-Lipson method would cheerfully produce a supposed "orthorhombic" cell for just about any powder pattern. Since two decades would pass before the introduction of figures of merit brought some discipline into the evaluation of trial cells, its main effect was to flood the (especially mineralogical) literature with fictitious "orthorhombic" cells, and lead to an understandable attitude of caution regarding all powder-derived cells unless they were high-symmetry.

The general opinion that low-symmetry, and particularly triclinic, cases were unfeasible persisted until Runge's method was rediscovered by Ito (1949, 1950). Ito's method was more effective at raising hopes than for practical indexing since, in most crystal systems above triclinic, some at least of the reflections on which it relied would be systematically absent. This restriction was lifted when the method was generalised by de Wolff (1957-1963) to become zone indexing, and implemented from 1967 onwards by Jan Visser in his widely-used ITO series of programs (Visser, 1969).

Powder indexing was an obvious problem to delegate to an electronic computer, and the first serious indexing programs date to the early days of mainframe computers - in one case for a machine that used vacuum tubes rather than transistors (Whalley, c.1965)! However, the pioneers soon learned that matters were not as easy as they had hoped. Manual methods were sufficiently laborious that most users were glad to accept the first moderately plausible solution they found. Computers could certainly remove this drudgery, but at the cost of throwing up far too many potential solutions, of which at least all but one must be wrong.

Hesse/Lipson-type programs (e.g. Hoff & Kitchingman, 1966) and simplistic Ito-style approaches (e.g. Roof, 1968) came and went. Apart from Visser's ITO, there were to be only two other survivors from this pre-1970 era: an early version of Per-Eric Werner's semi-exhaustive TREOR index-space program (Werner, 1964) and an ancestor of the heuristic grid-search approach that would later be implemented as the SEARCH module of the POWDER49 system (Shirley, 1975) and in Crysfire as Mmap (Shirley, 2002). Many programs still in use date to the mid-1970s - for a comparative review, see Shirley (1978).

A surprisingly hard problem

Why should powder indexing be hard? At first sight, it looks trivial by comparison with structure determination and refinement from powder data, which both involve many more unknown parameters, but experience has shown otherwise. Generating a powder diffraction pattern from a known unit cell resembles what in cryptography is called a one-way or trapdoor function, since in the forward direction it is completely determinate and easy to compute, while its inverse, the reconstruction of the correct cell from the resulting powder pattern, is ambiguous, incompletely determinate and hugely more laborious.

Like structure solution but more strikingly so, success with *ab initio* indexing tends to be an all-or-nothing matter, which may come out relatively easily using one or more programs, but can also be deeply hard and accomplished only after much labour by an experienced specialist, if at all. It will usually not be obvious in advance which is the case for a particular dataset.

Although pattern generation involves at most 6 lattice parameters and so just 6 degrees of freedom, this would only be true of indexing if we had available the complete, infinite, calculated powder pattern, with no accidental line overlaps and at infinite resolution. Because the pattern is necessarily truncated at the limiting sphere (if not sooner due to line overlaps and intensity fall-off), the diffraction order of any line can only be determined relatively, not absolutely, at least until the complete structure is known. Including a possible common integer factor due to diffraction order (typically less than 4 and usually 1 or 2) for each of the three axes, the total number of degrees of freedom for the indexing process rises to 9 (or 10 if one includes a calibration term such as the zero of 2θ).

But in practice only the first 20-30 observed lines contain much indexing information, since at higher angles the calculated line positions become too sensitive to small changes in cell constant to yield stable indexings and hence to place much constraint on possible unit cells. Thus indexing turns out to be less over-determined than structure solution and refinement, where information about many hundreds of Bragg intensities may be available in the observed profile.

This relatively modest degree of over-determination is one of the reasons why indexing is mathematically so strikingly a multi-solution process, yielding hundreds and often thousands of potential solutions (and hence more or less plausible trial cells), of which of course only one can correspond to the physical lattice.

Figures of merit - distinguishing between plausible cells

A plausible trial cell is one whose calculated pattern is in reasonably good agreement with the observed one. This agreement can be assessed by a figure of merit (FOM) such as M_{20} (de Wolff, 1968, 1972). FOM of this type (M_{20}, M1, etc.) are based on the sum of discrepancies between observed and calculated Q-values. The Smith & Snyder (1979) F_N criterion has broadly similar properties, but is optimised more for evaluating datasets than for trial solutions.

Most indexing programs, including the three most widely used - ITO (Jan Visser), TREOR (Per-Eric Werner) and DICVOL (Daniel Louër & A. Boultif) - use some combination of "M_{20}" (simplified from de Wolff's original definition) and the number of lines indexed (out of the first 20 observed), to rank the solutions they have found into descending order of plausibility.

Unfortunately all such discrepancy-based FOM discriminate poorly between trial solutions when the observed pattern contains non-model features such as impurity lines, which inject large noise terms into the difference summations and thus degrade the FOM for good solutions, so that they no longer stand out. Attempts to avoid this situation by excluding some lines as "unindexed" merely introduces arbitrary discontinuities and increases the potential number of spurious solutions. These properties make it hard (many would say hopeless) to determine unit cells for specimens with substantial amounts of unidentified impurities - especially for mixed phases, where each additional solid phase consumes a further 9 degrees of freedom.

This limitation can be reduced and perhaps avoided by using FOM based on joint probability, such as PM (Ishida & Watanabe, 1967, 1972, 1981) as used in the FJZN6 and LZON programs within Crysfire. Two new and better-behaved functions which are relatively transparent to non-model features will be described at ECM-21 in Durban this August.

Methods and algorithms: a) Strategies

1) **Zone indexing** This uses the Runge-Ito-de Wolff-Visser strategy of decomposition into central zones as described above. Trial zones that share a common row are then combined to produce trial lattices. Advantages: can be very fast and relatively insensitive to impurity lines (except at low angles). Limitations: not exhaustive, intolerant of errors, especially at low angles.

2) **SIW heuristic** This requires only a single well-established zone and then exploits the existence of an arbitrary choice of 001 from the first-level lines to reduce the problem to 2 dimensions (α^* and β^*), which can then be searched exhaustively. Advantages: final searches are exhaustive, any search method can be used, powerful for dominant zones, can be made less sensitive to errors and impurities (at the expense of longer run times). Limitations: depends on the accuracy of the basis zone and the line used as 001; depending on the method used, the 2-dimensional exhaustive searches can be moderately time-consuming.

3) **Profile-based** With increasing processor power, attempts have been made to work with whole profiles rather than line positions, with obvious potential gains in generality, but costs in speed. Advantages: avoids explicit extraction of line positions, relatively insensitive to impurities and measurement errors. Limitations: though not exhaustive, can still be quite slow - an effective compromise is to regenerate suitably idealised profiles from line-position data, as used in recent versions of McMaille, but overnight runs may still be needed on a fast processor.

4) **Scan/covariance** An unusual approach used only in Bergmann's EFLECH/INDEX system, which extracts both line positions and a covariance matrix from the original profile, then makes use of the latter during indexing. It performed well during a recent indexing round-robin.

5) **Index heuristics** This category contains programs which use complex systems of highly-tuned rules, wholly or mainly in index space, to infer index information from apparent relations between observed line positions. Apart from being powerful in its own right, this different approach offers an independent check of results obtained by the various parameter-space methods. Advantages: relatively insensitive to impurities, a complementary approach. Limitations: not exhaustive, only moderately fast.

6) **Index permutation** An obvious exhaustive approach in index space (but used only by Taupin's POWDER [=TAUP] is to permute indices systematically (within reasonable limits) for sufficient of the observed lines to calculate trial cells. To be feasible in lower symmetry, this must be accompanied by prompt pruning of failing search-trees, but even so it is not going to be fast. Advantages: exhaustive and reasonably fast in higher symmetry. Limitations: very slow in low symmetry, vulnerable to impurities.

Methods and algorithms: b) Search methods

1) **Successive dichotomy** This is the name used for binary search applied to indexing: which is the optimum method for continuous, approximately uniform solution spaces, for which an exclusion criterion exists. Unlike the other methods, it uses hard-edged inequalities to determine whether a solution can exist within each search domain, and descends by parameter-dichotomy, discarding failing sub-domains, until each surviving sub-domain is too small to contain more than one solution. Advantages: insensitive to absences, optimally fast for exhaustive searches in parameter space (even without using crystallographic heuristics, which these programs do sometimes include). Limitations: vulnerable to impurities and liable to be very slow in low symmetry.

2) **Grid search** A repetitive step-and-repeat search across some or all of parameter space, inherently much slower than binary search (dichotomy), but able to use figure-of-merit-based criteria instead of exact inequalities. Advantages: flexible, exhaustive within its specified search ranges, can be made insensitive to errors and impurities. Limitations: slow for more than two search dimensions.

3) **Global optimisation methods** In general, these are not well suited to indexing's solution space, which can contain many narrow local minima. Thus they need efficient refinement procedures to broaden their radius of convergence. Some specific examples are discussed below. Advantages: flexible, insensitive to absences, can be made relatively insensitive to errors and (to some extent) to impurities. Limitations: not exhaustive, sensitive to dominant zones, a search in low symmetry can involve numerous restarts and much processor time.

a) **Genetic algorithms** Includes varieties of methods that use iterative separation and recombination of solution features, with only the more successful surviving to enter the next cycle. These work best when the different solution features ("genes"), such as the powder constants, are relatively separable and can thus be treated as approximately independent in their effects. This is valid for powder constants if the cell is orthogonal, but less so with oblique angles. More sophisticated schemes might take advantage of separation into zones, etc. Advantages: scope for introducing crystallographic knowledge. Limitations: as for other global methods.

b) **Monte Carlo** Successive sets of trial cell parameters are generated randomly and refined and evaluated as fast as possible. Advantages: simplicity, scope for using parallel processing. Limitations: requires much optimisation and processor speed.

c) **Simulated annealing** & d) **Diffusion equation** These do not seem yet to have been used, but experience elsewhere suggests that (c) and especially (d) might be well suited to the localised minima in indexing's solution-space.

Table 1: Classification of indexing methods

Method (strategy/search)	Space	Exhaustive	Status	Are program(s) available?
Zone indexing	Parameter	No	Mature	Yes (2+)
SIW heuristic	Parameter	Semi	Mature/developing	Yes (3+)
Profile-based	Parameter	No	Developing	Yes (1+)
Scan/covariance	Par. (both)	To monocl.	Developing	Yes (1)
Index heuristics	Index	No	Mature	Yes (2+)
Index permutation	Index	Yes	Mature	Yes (1)
Successive dichotomy	Parameter	Yes	Mature/developing	Yes (3+)
Grid search	Parameter	Yes	Mature/developing	Yes (3+)
Genetic algorithms	Parameter	No	Developing	Yes (1+)
Monte Carlo	Parameter	No	Developing	Yes (2)
Simulated annealing	Parameter	No	Not yet tried	No
Diffusion equation	Parameter	No	Not yet tried	No

(The list is not claimed to be complete. Programs may need to be classified both by strategy and search method. Only symmetry-general methods have been included)

Table 2: Methods vs programs

Space	Method (strategy/search)	Programs using this method
Parameter	Zone indexing	ITO12, FJZN6, [for zone-searches: LZON]
	SIW heuristic	LZON, LOSH, (Powder49), Mmap, (Hmap)
	Profile-based	McMaille, (GAIN: Harris et al)
	Successive dichotomy	DICVOL91, [in part: LZON, LOSH], (X-Cell)
	Grid search	SCANIX, (Powder49), Mmap, (Hmap)
	Genetic algorithms	AUTOX, (MRIA), (GAIN: Harris et al)
	Monte Carlo	McMaille, (SVD-Index)
	Simulated annealing	(not yet implemented)
	Diffusion equation	(not yet implemented)
Par. (both)	Scan/covariance	EFLECH/INDEX
Index	Index heuristics	TREOR90, TMO [=KOHL], (N-TREOR)
	Index permutation	POWDER [=TAUP]

(Underlined if supported by Crysfire. In round brackets if part of a commercial suite, or otherwise not yet generally available. These classifications are approximate and, where applicable, programs are listed both under their strategy and their search method.)

Recently released programs (2002-03)

McMaille (Armel Le Bail) - freely available.

A Monte Carlo program that has been in development for less than a year, during which it has passed through several versions with a great increase in speed as it moved from full profile data to idealised profiles generated by applying simplified line profiles to extracted line positions. It also relies for its speed on fast full-profile refinement algorithms inherited from Le Bail's structure solution and refinement software. It can still require an overnight run in fully automatic "black box mode", but the fact that it can work effectively at all shows how risky it can be to make negative predictions - less than a year ago at the Geneva Congress I predicted that it would be many years before computers became fast enough for whole-profile-based indexing to become feasible!

SVD-index (Alan Coelho) - within the TOPAS suite from Bruker AXS (Coelho, 2003).

A commercial Monte Carlo program that again leans heavily on existing fast refinement routines to obtain the necessary speed and radius of convergence - in this case a singular-value decomposition algorithm used elsewhere within the TOPAS suite. SVD-Index was presented last summer at the Geneva IUCr Congress and, though at that time it had been tested mainly against simulated data, its speed seemed reasonable (especially when it got lucky in a relatively early trial) and the success rate looked encouraging. It is claimed to be insensitive to missing low angle lines, impurities and zero errors.

X-Cell (Marcus Neumann) - within the Materials Studio suite from Accelrys (Neumann, 2003).

Another commercial program presented at the Geneva Congress. Like DICVOL, X-Cell is based on successive dichotomy, though with a number of variations and additions to enhance its performance, such as exploiting the presence of systematic absences. It is described as "virtually exhaustive" and is claimed to perform well in the face of missing lines, impurities and errors. Presumably, in so far as it is exhaustive, it may need long run times in low symmetry, especially in the presence of significant amounts of impurity.

In the pipeline (Summer 2003)

Denver X-Ray Conference (8 August 2003)

Daniel Louër will be presenting a talk "Indexing powder diffraction patterns with the dichotomy method: new developments" - a possible successor to DICVOL?

ECM-21, Durban (28 August 2003)

For the first time at any major meeting there will be a special microsymposium on powder indexing, the scheduled speakers being Robin Shirley (on joint-probability methods, including the new figures of merit Pr and Ir), Victor Zlokazov (on AUTOX, etc), Armel Le Bail (on McMaille) and Frank Stowasser (on SVD-Index).

References

(Only those explicitly cited in the text are shown here - for additional references on the various programs, see the Crysfire manual and the CCP14 website: www.ccp14.ac.uk).

Coelho, A. A. (2003). "Indexing of powder diffraction patterns by iterative use of singular value decomposition", *J. Appl. Cryst.*, 36, 86-95.

Hesse, R. (1948). "Indexing Powder Photographs of Tetragonal, Hexagonal and Orthorhombic Crystals", *Acta Cryst.*, 1, 200-207.

Hoff, W. D. & Kitchingman, W. J. (1966). "Computer indexing of x-ray powder patterns from crystals of unknown structures", *J. Sci. Instrum.*, 43, 952-953.

Ishida, T & Watanabe, Y. (1967). "Probability Computer Method of Determining the Lattice Parameters from Powder Diffraction Data", *J. Phys. Soc. Japan*, 23, 556-565.

Ishida, T & Watanabe, Y. (1971). "Analysis of Powder Diffraction Patterns of Monoclinic and Triclinic Crystals", *J. Appl. Cryst.*, 4, 311-316.

Ito, T. (1949). "A General Powder X-Ray Photography", *Nature*, 164, 755-756.

Ito, T. (1950). *X-ray Studies on Polymorphism*, Maruzen, Tokyo, 187-228.

Neumann, M. A. (2003). "X-Cell: a novel indexing algorithm for routine tasks and difficult cases", *J. Appl. Cryst.*, 36, 356-365.

Roof, R. B. (1968). *INDX: A Computer Program to Aid in the Indexing of X-Ray Powder Patterns of Crystal Structures of Unknown Symmetry*, Los Alamos Laboratory, University of California, Report LA-3920.

Shirley, R. (1975). "Recent advances in determining unknown unit cells from powder diffraction data", *Acta Cryst.*, A31, S197.

Shirley, R. (1978). "Indexing powder diagrams", in *Crystallographic Computing*, ed. Schenk, H., Olthof-Hazekamp, R., van Koningsveld, H. & Bassi, G. C., Delft University Press: Holland, 221-234.

Shirley, R. (2002). *The Crysfire 2002 System for Automatic Powder Indexing: User's Manua*l, The Lattice Press: Guildford, U.K. (http://www.ccp14.ac.uk/tutorial/crys/)

Smith, G. S. & Snyder, R. L. (1979). "F_N: A Criterion for Rating Powder Diffraction Patterns and Evaluating the Reliability of Powder-Pattern Indexing", *J. Appl. Cryst.*, 12, 60-65.

Ishida, T & Watanabe, Y. (1982). "A criterion method for indexing unknown powder diffraction patterns", *Z. Krist.*, 160, 19-32.

Lipson, H. (1949). "Indexing Powder Photographs of Orthorhombic Crystals", *Acta Cryst.*, 2, 43-45.

Runge, C. (1917). "Die Bestimmung eines Kristallsystems durch Rontgenstrahlen", *Physik. Z.*, 18, 509-515.

Visser, J. W. (1969). "A Fully Automatic Program for Finding the Unit Cell from Powder Data", *J. Appl. Cryst.*, 2, 89-95.

Whalley, D. (c.1965). Exhaustive ("blockbuster") indexing program for the Ferranti Pegasus computer, Statistics & Chemistry Departments, University College London - unpublished report.

Wolff, P. M. de (1957). "On the Determination of Unit-Cell Dimensions from Powder Diffraction Patterns", *Acta Cryst.*, 10, 590-595.

Wolff, P. M. de (1958). "Detection of simultaneous zone relations among powder diffraction lines", *Acta Cryst.*, 11, 664-665.

Wolff, P. M. de (1961). "Reliability of Unit Cells Derived from Powder Diffraction Patterns", *Acta Cryst.,* 14, 579-582.

Wolff, P. M. de (1963). "Indexing of Powder Diffraction Patterns", *Adv. X-Ray Anal.*, 6, 1-17.

Wolff, P. M. de (1968). "A Simplified Criterion for the Reliability of a Powder Pattern Indexing", *J. Appl. Cryst.*, 1, 108-113.

Wolff, P. M. de (1972). "The definition of the indexing figure of merit M_{20}", *J. Appl. Cryst.*, 5, 243.

Future Symposia relevant to Crystallographic Computing

During ACA 2003, 26th to 31st July, Cincinnati, USA: abstracts for *"Future strategies for successful crystallographic computing"*, 08:30 – 12:30pm, Room 6, Monday 28th July, 2003.

One half-day session organized by Ross Angel (rangel@vt.edu ; Virginia Tech Crystallography Lab) and David Watkin (david.watkin@chemistry.oxford.ac.uk ; Oxford)

08:30 - 09:00 AM SP.02.01 (W0139) *David Watkin.* **Crystallographic Software - A Bleak Future?** *et al: Richard Cooper.*

09:00 - 09:30 AM SP.02.02 (W0097) *Lachlan M. D. Cranswick.* **Government Funded Central Initiatives for Encouraging a Diversity of Freely Available Crystallographic Software; and the Threat of Crystallographic Software Patents.**

09:30 - 09:45 AM SP.02.03 (W0362) *Carroll Johnson*. **Hacker Vulnerability: A Major New Complication in Crystallographic Computing.**

09:45 - 10:00 AM Discussion.

10:00 - 10:30 AM Coffee Break.

10:30 - 10:45 AM SP.02.04 (W0174) *Brian Toby*. **Crystallographic Software from the NIST Center for Neutron Research.**

10:45 - 11:00 AM SP.02.05 (W0438) *J. W. Pflugrath.* **There is No Such Thing as Free Software.**

11:00 - 11:15 AM SP.02.06 (W0194) *Susan Byram*. **Crystallographic Computing Strategies at Bruker Nonius.**

11:15 - 11:30 AM SP.02.07 (W0136) *Mathias Meyer*. **The CrysAlis Software Suite for Area and Point Detector Measurements - Open Source Option and User Modifications.**

11:30 - 11:45 AM SP.02.08 (W0008) *Paul Adams*. **Developing Modern Crystallographic Libraries and Applications: PHENIX and the Computational Crystallography Toolbox**. *et al: Ralf Grosse-Kunstleve, Nigel Moriarty, Nicholas Sauter.*

More details of the ACA meeting and registration and abstract submission at
http://www.che.uc.edu/aca/ and http://www.hwi.buffalo.edu/ACA/ACA03/abstracts/

ECA-21 2003, 24th to 29th August 2003, Durban, South Africa

A number of sessions relevant to Crystallographic Computing are occurring at the ECA-21 meeting in Durban South Africa.

For more details and registration information, refer to the conference homepage at http://www.sacrs.org.za/ecm21/ and http://www.sacrs.org.za/ecm21/sci_timetable.html

- SIG 3 – 1: *New developments in the structure determination of quasicrystals. Chair*: G Chapuis, (gervais.chapuis@ic.unil.ch); Co-chair: W Steurer (w.steurer@kristall.erdw.ethz.ch) (Monday 25 August, 09:15 - 11:15)

- Plenary Lecture 6: presented by Bill David (wifd@isise.rl.ac.uk): *Structure determination from powders - crossing the 100-atom threshold.* (Monday 25 August, 11:45 - 12:45)

- SIG8 - 5: *Modern programming languages and programming techniques. Can they lead to higher reliability and programmer productivity? Chair L. Cranswick (l.m.d.cranswick@dl.ac.uk); Co-chair: J. Dillen JLMD@land.sun.ac.za)* (Tuesday 26 August, 13:45 - 15:45)

- SIG9 – 6: *Diffraction image processing and data quality. Chair: E Dodson (ccp4@yorvic.york.ac.uk); Co-chair: S Parsons (s.parsons@ed.ac.uk)* (Wednesday 27 August, 09:15 - 11:15)

- SIG9 – 7: *Algorithms of the future; Chair: D Watkin. (david.watkin@chem.ox.ac.uk); Co-chair: A Urzhumtsev (sacha@lcm3b.uhp-nancy.fr)* (Wednesday 27 August, 13:45 - 15:45)

- SIG9 – 8: *Indexing Powder Diffraction Patterns: An opportunity for new heuristic and global optimisation methods Chair: R Shirley (r.shirley@surrey.ac.uk); Co-chair: C Rademeyer (cor@spl.setpoint.co.za)* (Thursday 28 August, 09:15 - 11:15)

- SIG6 – 4: *Automatic structure determination: Challenges for the Future. Chair: A.L. Spek (spea@chem.uu.nl); Co-chair: D Billing (dave@hobbes.gh.wits.ac.za)* (Thursday 28 August, 16:15 - 18:15)

Free one day Single Crystal and Powder Diffraction Crystallographic Software Workshop as part of ECA 2003 in Durban, South Africa

With a main theme of "Crystallographic Software Wizardry in Single Crystal and Powder Diffraction", and thanks to the conference organisers, there will be a free one day single crystal and powder diffraction software workshop on Sunday 24th August 2003 (at the start of the ECA conference). The emphasis will be on practical 20 minute "power user" examples of features within the software that casual users may not appreciate. The workshop will take place in the same facility as the conference, that being the International Conference Centre in Durban, South Africa.

The workshop webpage, with links to the presenters and the software being presented is at:

http://www.ccp14.ac.uk/projects/ecm21-durban2003/

The morning will consist of the powder diffraction session. This will include: use of the ICDD database; powder indexing of large volume cells (including proteins); powder indexing of impure samples using whole profile methods; structure solution using direct methods and EXPO; structure solution using real space methods and the FOX software; Rietveld refinement of complex inorganic materials using Fullprof; Rietveld Structure Refinement of protein powder diffraction data using GSAS

The afternoon will then be on single crystal methods. Topics include: Using CCDs for visually finding tricky cells, supercells and incommensurate cells; Advanced absorption correction options using Platon and Euhedral; Data processing of Bruker and Nonius CCD data using the WinGX Single Crystal suite, incorporating Sortav; SnB (Shake-n-Bake) direct methods software; Dirdif - fragment searching to solve structures that direct methods won't; Crystals to CCDC Mogul geometry validation; and non-trivial applications of Platon, Addsym and intra/inter-molecular validation

Speakers who are presently listed at time of writing include (for Powder Diffraction): Brian O'Connor, Robin Shirley, Armel Le Bail, Vincent Favre-Nicolin, Mauro Bortolotti, Luca Lutterotti, Juan Rodriguez-Carvajal and Jon Wright. (Single Crystal): Martin Lutz, Louis Farrugia, Charles Weeks, Bob Gould, David Watkin and Ton Spek.

While the workshop is free, places are limited. People attending ECM-21 and who wish to also go to the software workshop are requested to E-mail their interest in attending to confcall@yebo.co.za **and** also Cc a copy to Lachlan Cranswick at l.m.d.cranswick@dl.ac.uk.

Lachlan M. D. Cranswick
NPMR, NRC,
Building 459, Station 18,
Chalk River Laboratories,
Chalk River, Ontario,
Canada, K0J 1J0
Tel: (613) 584-8811 ext 3719
Fax: (613) 584-4040
E-mail: lachlan.cranswick@nrc.gc.ca
WWW: http://neutron.nrc.ca/ and http://www.ccp14.ac.uk/

Call for Contributions to the Next CompComm Newsletter

The third issue of the Compcomm Newsletter is expected to appear around January of 2004 with the primary theme still to be determined. If no-one is else is co-opted, it will be edited by Lachlan Cranswick.

Contributions would be greatly appreciated on matters of interest to the crystallographic computing community, e.g. meeting reports, future meetings, developments in software, algorithms, coding, programming languages, techniques and news of general interest.

Please send articles and suggestions directly to the editor.

Lachlan M. D. Cranswick
NPMR, NRC,
Building 459, Station 18,
Chalk River Laboratories,
Chalk River, Ontario,
Canada, K0J 1J0
E-mail: lachlan.cranswick@nrc.gc.ca