The Structure Determination Language of the Crystallography and NMR System

Axel T. Brunger[1,2,*], Paul D. Adams[2], Warren L. DeLano[3], Piet Gros[4], Ralf W. Grosse-Kunstleve[1,2], Jian-Sheng Jiang[5], Navraj S. Pannu[6], Randy J. Read[7], Luke M. Rice[2], Thomas Simonson[8]

[1] The Howard Hughes Medical Institute and [2] Department of Molecular Biophysics and Biochemistry, Yale University, New Haven, CT 06511, phone: 203-432-6143, FAX: 203-432-6946, email: brunger@laplace.csb.yale.edu

[3] Graduate Group in Biophysics, Box 0448, University of California, San Francisco, CA 94143, FAX: 415-225-3734

[4] Crystal and Structural Chemistry, Bijvoet Center for Biomolecular Research, Utrecht University, Padualaan 8, 3584 CH Utrecht, The Netherlands, FAX: +31 30 2533940

[5] Protein Data Bank, Biology Department, Brookhaven National Laboratory, Upton, NY 11973-5000, USA, FAX: 516-344-5751

[6] Department of Mathematical Sciences, University of Alberta, Edmonton, Alberta T6G 2G1, Canada, FAX: (403) 492-7521

[7] Department of Medical Microbiology and Immunology, University of Alberta, Edmonton, Alberta T6G 2H7, Canada, FAX: (403) 492–7521

[8] Laboratoire de Biologie Structurale (C.N.R.S.), I.G.B.M.C., 1 rue Laurent Fries, 67404 Illkirch (C.U. de Strasbourg), France, FAX: +33 3 88 65 32 01

[*] corresponding author

Introduction

We have developed a new and advanced software system, termed Crystallography and NMR System (CNS), for crystallographic and NMR structure determination (Brunger *et al.*, 1998). The goals of CNS are: (1) to create a flexible computational framework for exploration of new approaches to structure determination, (2) to provide tools for structure solution of difficult or large structures, (3) to develop models for analyzing structural and dynamical properties of macromolecules, and (4) to integrate all sources of information into all stages of the structure determination process.

To meet these goals, algorithms were moved from the source code into a symbolic structure determination language which represents a new concept in computational crystallography. The high–level CNS computing language allows definition of symbolic target functions, data structures, procedures, and modules. The CNS program acts as an interpreter for the high–level CNS language and includes hard–wired functions for efficient processing of computing–intensive tasks. Methods and algorithms are therefore more clearly defined, and easier to adapt to new and challenging problems. The result is a multi–level system which provides maximum flexibility to the user (Fig. 1). The CNS language provides a common framework for nearly all computational procedures of structure determination. A comprehensive set of crystallographic procedures for phasing, density modification, and refinement has been implemented in this language. Task–oriented input files written in the CNS language, which can also be accessed through an HTML graphical interface (Graham, 1995), are available to carry out these procedures.

CNS Language

One of the key features of the CNS language is symbolic data structure manipulation, e.g.

(1)

```
xray
 do (pa=-2*(amplitude(fp)^2 + amplitude(fh)^2 - amplitude(fph)^2 )
  *amplitude(fp)*real(fh)/(3*v^2 + 4*(amplitude(fph)^2+sph^2)*v) )
    (acentric)
end
```

which is equivalent to the following mathematical expression for all acentric indices $\vec{h}$,

$$p_{\rm a}(\vec{h}) = 2\frac{-(|\mathbf{f}_{\rm p}(\vec{h})|^2 + |\mathbf{f}_{\rm h}(\vec{h})|^2 - |\mathbf{f}_{\rm ph}(\vec{h})|^2)|\mathbf{f}_{\rm p}(\vec{h})|\frac{\mathbf{f}_{\rm h}(\vec{h})+\mathbf{f}_{\rm h}(\vec{h})^*}{2}}{3v(\vec{h})^2 + 4(|\mathbf{f}_{\rm ph}(\vec{h})|^2 + s_{\rm ph}(\vec{h})^2) * v(\vec{h})} \qquad (2)$$

where $\mathbf{f}_{\rm p}$ ("fp" in Eq. 1) is the "native" structure factor array, $\mathbf{f}_{\rm ph}$ ("fph" in Eq. 1) is the derivative structure factor array, $s_{\rm ph}$ ("sph" in Eq. 1) the corresponding experimental σ, v is the expectation value for the lack–of–closure (including lack–of–isomorphism and errors in the heavy atom model), and $\mathbf{f}_{\rm h}$ ("fh" in Eq. 1) is the calculated heavy atom structure factor array. This expression computes the $A_{\rm iso}$ coefficient of the phase probability distribution for single-isomorphous replacement described by Hendrickson & Lattman (1970) and Blundell & Johnson (1976).

The expression in Eq. 1 is computed for the specified subset of reflections "(acentric)". This expression means that only the selected (in this case all acentric) reflections are used. More sophisticated selections are possible, e.g.,

(3)

```
( amplitude(fp) > 2 * sh and amplitude (fph) > 2 * sph and d >= 3 )
```

selects all reflections with Bragg spacing d greater than 3 Å for which both native (fp) and derivative (fph) amplitudes are greater than two times their corresponding σ values ("sh" and "sph", respectively). Extensive use of this structure factor selection facility is made for cross–validating statistical properties, such as R–values (Brunger, 1992), σ_A values (Kleywegt & Brunger, 1996; Read, 1997), and maximum likelihood functions (Pannu & Read, 1996; Adams *et al.*, 1997).

Similar operations exist for electron density maps, e.g.,

(4)

```
xray
   do (map=0) ( map < 0.1 )
end
```

is an example of a truncation operation: all map values less than 0.1 are set to 0. Atoms can be selected based on a number of atomic properties and descriptors, e.g.,

(5)

```
do (b=10)  ( residue 1:40 and
             ( name ca or name n or name c or name o ) )
```

sets the B–factors of all polypeptide backbone atoms of residues 1 through 40 to 10 Å^2.

Operations exist between data structures, e.g., real, reciprocal space arrays, and atom properties. For example, Fourier transformations between real and reciprocal space can be accomplished by the following CNS commands

(6)

```
xray
   mapresolution infinity 3.
   fft grid 0.3333 end
   do (map=ft(f_cal)) ( acentric )
end
```

which computes a map on a 1 Å grid by Fourier transformation of the "f_cal" array for all acentric reflections.

Atoms can be associated with calculated structure factors, e.g.,

(7)

```
associate f_cal ( residue 1:50 )
```

This statement will associate the reciprocal space array "f_cal" with the atoms belonging to residues 1 through 50. These structure factor associations are used in the symbolic target functions described below.

There are no predefined reciprocal or real–space arrays in CNS. Dynamic memory allocation allows one to carry out operations on arbitrarily large data sets with many individual entries (e.g., derivative diffraction data) without the need for re–compiling the source code. The various reciprocal structure factor arrays must therefore be declared and their type specified prior to invoking them. For example, a reciprocal space array with real values, such as observed amplitudes,

is declared by the following expression,

(8)

```
declare name=fobs type=real domain=reciprocal end
```

Reciprocal space arrays can be grouped. For example, Hendrickson & Lattman (1970) coefficients are represented as a group of four reciprocal structure factor arrays

(9)

```
group type=hl object=pa object=pb object=pc object=pd end
```

where "pa", "pb", "pc", and "pd" refer to the individual arrays. This group statement indicates to CNS that the specified arrays need to be transformed together when reflection indices are changed, e.g., during expansion of the diffraction data to spacegroup P1.

Symbols and Parameters

The CNS language supports two types of data elements which may be used to store and retrieve information. *Symbols* are typed variables such as numbers, character strings of restricted length, and logicals. *Parameters* are untyped data elements of arbitrary length that may contain collections of CNS commands, numbers, strings, or symbols.

Symbols are denoted by a dollar sign ($) and parameters by an ampersand (&). Symbols and parameters may contain a single data element, or they may be a *compound* data structure of arbitrary complexity. The hierarchy of these data structures is denoted using a period (.). Figures 2a and b demonstrate how

crystal lattice information can be stored in compound symbols and parameters, respectively. The information stored in symbols or parameters can be retrieved by simply referring to them within a CNS command: the symbol or parameter name is substituted by its content. Symbol substitution of portions of the compound names (e.g., "&crystal_lattice.unit_cell.$para") allows one to carry out conditional and iterative operations on such data structures, such as matrix multiplication.

Statistical Functions

The CNS language contains a number of statistical operations, such as bin–wise averages and summations. The resolution bins are defined by a central facility in CNS. At present, equal–volume reciprocal bins of specified width are possible. It is planned to provide other bin–wise partitioning schemes in the future.

Figure 3 shows how σ_A, σ_Δ, and D (Read, 1986, 1990) are computed from the observed structure factors ("fobs") and the calculated model structure factors ("fcalc") using the CNS statistical operations. The first five operations are performed for the reflections in the test set while the last three operations expand the results to all reflections. The "norm" function computes normalized structure factor amplitudes for the specified arguments. The "sigacv" function evaluates σ_A from the normalized structure factors. The "save" function computes the statistical average

$$\mathrm{save}(f) = \frac{\sum_{hkl} f_{hkl} \frac{w}{\epsilon}}{\sum_{hkl} w} \tag{10}$$

where w is 1 and 2 for centric and acentric reflections respectively, and ϵ is the statistical weight. The averages are computed bin–wise and the result for a particular bin is stored in all selected reflections belonging to the bin.

Symbolic Target Function

One of the key innovative features of CNS is the ability to symbolically define target functions and their first derivatives for crystallographic searches and refinement. This allows one to conveniently implement new crystallographic methodologies as they are being developed.

The power of symbolic target functions is illustrated by two examples. In the first example, a target function is defined for simultaneous heavy atom parameter refinement of three derivatives. The sites for each of the three derivatives can be disjoint or identical depending on the particular situation. For simplicity, the Blow & Crick (1959) approach is used, although maximum likelihood targets are also possible (see below). The heavy atom sites are refined against the following target

$$\sum_{hkl} \frac{(|\mathbf{F}_{h_1} + \mathbf{F}_p| - \mathbf{F}_{ph_1})^2}{2v_1} + \frac{(|\mathbf{F}_{h_2} + \mathbf{F}_p| - \mathbf{F}_{ph_2})^2}{2v_2} + \frac{(|\mathbf{F}_{h_3} + \mathbf{F}_p| - \mathbf{F}_{ph_3})^2}{2v_3}. \qquad (11)$$

$\mathbf{F}_{h_1}$, $\mathbf{F}_{h_2}$, $\mathbf{F}_{h_3}$ are complex structure factors corresponding to the three sets of heavy atoms sites, $\mathbf{F}_p$ represents the structure factors of the native crystal, and $|\mathbf{F}_{ph_1}|$, $|\mathbf{F}_{ph_2}|$, $|\mathbf{F}_{ph_3}|$ are the structure factor amplitudes of the derivatives, and v_1, v_2, and v_3 are the variances of the three lack of closure expressions. The corresponding target expression and its first derivatives with respect to the calculated structure factors are shown in Fig. 4a. The derivatives of the target function with respect to each of the three associated structure factor arrays are specified with the "dtarget" expressions. The "tselection" statement specifies the selected subset of reflections to be used in the target function (e.g., excluding outliers) and the "cvselection" statement specifies a subset of reflections to be used for cross–validation (Brunger, 1992) (i.e., the subset is not used during refinement but only as a monitor for the progress of refinement).

The second example is the refinement of a perfectly twinned crystal with overlapping reflections from two independent crystal lattices. Refinement of the

model is carried out against the following residual

$$\sum_{hkl} |\mathbf{F}_{\rm obs}| - \sqrt{|\mathbf{F}_{\rm calc1}|^2 + |\mathbf{F}_{\rm calc2}|^2} \tag{12}$$

The symbolic definition of this target is shown in Fig. 4b. The twinning operation itself is imposed as a relationship between the two sets of selected atoms (not shown). This example assumes that the two calculated structure factor arrays ("fcalc1" and "fcalc2") that correspond to the two lattices have been appropriately scaled with respect to the observed structure factors and the twinning fractions have been incorporated into the scale factors. However, a more sophisticated target function could be defined which incorporates scaling.

A major advantage of the symbolic definition of the target function and its derivatives is that any arbitrary function of structure factor arrays can be used. This means that the scope of possible targets is not limited to least–squares targets. Symbolic definition of numerical integration over unknown variables (such as phase angles) is also possible. Thus, even complicated maximum likelihood target functions (Bricogne, 1984; Otwinowski, 1991; Pannu & Read, 1996; Pannu *et al.*, 1998) can be defined using the CNS language. This is particularly valuable at the prototype stage. For greater efficiency, the standard maximum likelihood targets are provided through CNS source code which can be accessed as functions in the CNS language. For example, the maximum likelihood target function MLF (Pannu & Read, 1996) and its derivative with respect to the calculated structure factors are defined as follows

(13)

```
target = (mlf(fobs,sigma,(fcalc+fbulk),d,sigma_delta))

dtarget = (dmlf(fobs,sigma,(fcalc+fbulk),d,sigma_delta))
```

where "mlf()" and "dmlf()" refer to internal maximum likelihood functions, "fobs" and " sigma" are the observed structure factor amplitudes and corresponding σ values, "fcalc" is the (complex) calculated structure factor array, "fbulk" is the structure factor array for a bulk solvent model, "d" and "sigma_delta" are the cross–validated D and σ_Δ functions (Read, 1990; Kleywegt & Brunger, 1996; Read, 1997) which are precomputed prior to invoking the MLF target function using the test set of reflections. The availability of internal FORTRAN subroutines for the most computing–intense target functions and the symbolic definitions involving structure factor arrays allows for maximal flexibility and efficiency. Other examples of available maximum likelihood target functions include MLI (intensity–based maximum likelihood refinement), MLHL (crystallographic model refinement with prior phase information (Pannu *et al.*, 1998), and maximum likelihood heavy atom parameter refinement for multiple–isomorphous replacement (Otwinowski, 1991) and MAD phasing (Hendrickson, 1991; Burling *et al.*, 1996). Work is in progress to define target functions that include correlations between different heavy–atom derivatives (Read, 1994).

Modules and Procedures

Modules exist as separate files and contain collections of CNS commands related to a particular task. In contrast, *procedures* can be defined and invoked from within any file. Modules and procedures share a similar parameter passing mechanism for both input and output. Modules and procedures make it possible to write programs in the CNS language in a manner similar to that of a computing language such as Fortran or C. CNS modules and procedures have defined sets of input (and output) parameters that are passed into them (or returned) when

they are invoked. This enables long collections of CNS language statements to be modularized for greater clarity of the underlying algorithm.

Parameters passed into a module or procedure inherit the scope of the calling task file or module, and thus they exhibit a behavior analogous to most computing languages. Symbols defined within a module or procedure are purely local variables.

The following example shows how the unit cell parameters defined above (Fig. 2b) are passed into a module named "compute_unit_cell_volume" (Fig. 5) which computes the volume of the unit cell from the crystal lattice parameters using well–established formulae (Stout & Jensen, 1989),

(14)

```
@compute_unit_cell_volume ( cell = &crystal_lattice.unit_cell;
                            volume = $cell_volume; ) .
```

The parameter "volume" is equated to the symbol "$cell_volume" upon invocation in order to return the result (the unit cell volume) from this module. Note that the use of compound parameters to define the crystal lattice parameters (Fig. 2b) provides a convenient way to pass all required information into the module by referring to the base name of the compound parameter ("&crystal_lattice.unit_cell") instead of having to specify each individual data element.

Figure 6a shows another example of a CNS module: the module named "phase_distribution" computes phase probability distributions using the Hendrickson & Lattman formalism (Hendrickson & Lattman, 1970; Hendrickson, 1979; Blundell & Johnson, 1976). An example for invoking the module is shown in Fig. 6b. This module could be called from task files that need access to iso-

morphous phase probability distributions. It would be straightforward to change the module in order to compute different expressions for the phase probability distributions.

A large number of additional modules are available for crystallographic phasing and refinement. CNS library modules include spacegroup information, Gaussian atomic form factors, anomalous scattering components, and molecular parameter and topology databases.

Task Files

Task files consist of CNS language statements and module invocations. The CNS language permits the design and execution of nearly any numerical task in X–ray crystallographic structure determination using a minimal set of "hard–wired" functions and routines. A list of the currently available crystallographic procedures and features is shown in Fig. 7.

Each task file is divided into two main sections: the initial parameter definition and the main body of the task file. The definition section contains definitions of all CNS parameters which are used in the main body of the task file. Modification of the main body of the file is not required, but may be done by experienced users in order to experiment with new algorithms. The definition section also contains the directives that specify specific HTML features, e.g., text comments (indicated by {* ... *}), user–modifiable fields (indicated by {===>}), and choice boxes (indicated by {+ choice: ... + }). Figure 8 shows a portion of the "define" section of a typical CNS refinement task file.

The task files produce a number of output files (e.g., coordinate, reflection, graphing, and analysis files). Comprehensive information about input pa-

rameters and results of the task are provided in these output files. In this way, the majority of the information required to reproduce the structure determination is kept with the results. Analysis data is often provided in simple columns and rows of numbers. These data files can be used for graphing, for example, by using commonly available spreadsheet programs. An HTML graphical output feature for CNS which makes use of these analysis files is planned. In addition, list files are often produced that contain a synopsis of the calculation.

HTML–Interface

The HTML graphical interface makes use of the HTML form language to create a high–level menu–driven environment for CNS (Fig. 9a). Two compact and relatively simple Common Gateway Interface (CGI) conversion scripts are available that transform a task file into a form page, and the edited form page back into a task file (Fig. 9b). These conversion scripts are written in the PERL language.

A comprehensive collection of task files are available for crystallographic phasing and refinement (Fig. 7). New task files can be created or existing ones modified in order to address problems that are not currently met by the distributed collection of task files. The HTML graphical interface thus provides a common interface for distributed and "personal" CNS task files (Fig. 9b).

Example: Combined Maximum Likelihood and Simulated Annealing Refinement

CNS has a comprehensive task file for simulated annealing refinement of crystal structures using Cartesian (Brunger *et al.*, 1987; Brunger, 1988) or torsion

angle molecular dynamics (Rice & Brunger, 1994). This task file automatically computes cross–validated σ_A estimates, determines the weighting scheme between the X–ray refinement target function and the geometric energy function (Brunger *et al.*, 1989), refines a flat bulk solvent model (Jiang & Brunger, 1994) and an overall anisotropic B–value of the model by least–squares minimization, and subsequently refines the atomic positions by simulated annealing. Options are available for specification of alternate conformations, multiple conformers (Burling & Brunger, 1994), non–crystallographic symmetry constraints and restraints (Weis *et al.*, 1990), and "flat" solvent models (Jiang & Brunger, 1994). Available target functions include the maximum likelihood functions MLF, MLI, and MLHL (Pannu & Read, 1996; Adams *et al.*, 1997; Pannu *et al.*, 1998). The user can choose between slow cooling (Brunger *et al.*, 1990) and constant temperature simulated annealing, and the respective rate of cooling and length of the annealing scheme. For a review of simulated annealing in X–ray crystallography, see Brunger *et al* (1997).

During simulated annealing refinement the model can be significantly improved. Therefore, it becomes important to recalculate the cross–validated σ_A error estimates (Kleywegt & Brunger, 1996; Read, 1997), and the weight between X–ray diffraction target function and the geometric energy function in the course of the refinement (Adams *et al.*, 1997). This is important for the maximum likelihood target functions which depend on the cross–validated σ_A error estimates. In the simulated annealing task file, the recalculation of σ_A values and subsequently the weight for the crystallographic energy term are carried out after initial energy minimization, and also after molecular dynamics simulated annealing.

Conclusions

CNS is a general system for structure determination by X-ray crystallography and solution NMR. It covers the whole spectrum of methods to solve X–ray or solution NMR structures. The multi–layer architecture allows use of the system with different levels of expertise. The HTML–interface allows the novice to perform standard tasks. The interface provides a convenient means of editing complicated task files, even for the expert (Fig. 9b). This graphical interface makes it less likely that an important parameter will be overlooked when editing the file. In addition, the graphical interface can be used with any task file, not just the standard distributed ones. HTML-based documentation and graphical output is planned in the future.

Most operations within a crystallographic algorithm are defined through modules and task files. This allows for the development of new algorithms and for existing algorithms to be precisely defined and easily modified without the need for source code modifications.

The hierarchical structure of CNS allows extensive testing at each level. For example, once the source code and CNS basic commands have been tested, testing of the modules and task files is performed. A test suite consisting of hundreds of test cases is frequently evaluated during CNS development in order to detect and correct programming errors. Furthermore, this suite is run on several hardware platforms, in order to detect any machine–specific errors. This testing scheme makes CNS highly reliable.

Algorithms can be readily understood by inspecting the modules or task files. This self–documenting feature of the modules provides a powerful teaching tool. Users can easily interpret an algorithm and compare it with published methods in the literature. To our knowledge, CNS is the only system that pro-

vides the ability to symbolically define any target function for a broad range of applications ranging from heavy–atom phasing, molecular replacement searches, to atomic resolution refinement.

Acknowledgment

Support by the Howard Hughes Medical Institute and the National Science Foundation to A.T.B. (DBI–9514819 and ASC 93–181159), the Natural Sciences and Engineering Research Council of Canada to N.S.P., the Howard Hughes Medical Institute and the Medical Research Council of Canada to R.J.R. (MT11000), the Netherlands Foundation for Chemical Research (SON–NWO) to P.G., and the Howard Hughes Medical Institute to L.M.R. is gratefully acknowledged.

References

Adams, P.D., Pannu, N.S.,Read, R.J. & Brunger, A.T. (1997). Cross-validated maximum likelihood enhances crystallographic simulated annealing refinement. *Proc. Natl. Acad. Sci. USA* **94**, 5018–5023.

Blow, D.W. & Crick, F.H.C. (1959). The treatment of errors in the isomorphous replacement method *Acta Cryst.* **12**, 794—802.

Blundell, T.L. & Johnson, L.N. (1976). Protein Crystallography, Academic Press, London, pp. 375–377.

Bricogne,G. (1984). Maximum Entropy and the Foundation of Direct Methods. *Acta Cryst.* **A40**, 410–445.

Brunger, A.T. (1988). Crystallographic refinement by simulated annealing: Application to a 2.8 Å resolution structure of aspartate aminotransferase, *J. Mol. Biol.* **203**, 803–816.

Brunger, A.T. (1992). The free R value: a novel statistical quantity for assessing the accuracy of crystal structures, *Nature* **355**, 472–474.

Brunger, A.T., Kuriyan, J., & Karplus, M. (1987). Crystallographic R factor refinement by molecular dynamics, *Science* **235**, 458–460.

Brunger, A.T., Karplus, M., & Petsko, G.A. (1989). Crystallographic refinement by simulated annealing: Application to a 1.5 Å resolution structure of crambin, *Acta Cryst. A* **45**, 50–61.

Brunger, A.T, Krukowski, A., & Erickson, J. (1990). Slow–cooling protocols for crystallographic refinement by simulated annealing, *Acta Cryst. A* **46**, 585–593.

Brunger, A.T., Adams, P.D., & Rice, L.M. (1997). New applications of simulated

annealing in X-ray crystallography and solution NMR. *Structure* **5**, 325–336.

Brunger, A.T., Adams, P.D., Clore, G.M., Gros, P., Grosse–Kunstleve, R.W., Jiang, J.-S., Kuszewski, J., Nilges, M., Pannu, N.S., Read, R.J., Rice, L.M., Simonson, T., Warren, G.L. (1998). Crystallography and NMR system (CNS): A new software system for macromolecular structure determination, *Acta Cryst. D*, in press.

Burling, F.T. & Brunger, A.T. (1994). Thermal motion and conformational disorder in protein crystal structures: Comparison of multi–conformer and time–averaging models, *Israel Journal of Chemistry* **34** 165–175.

Burling, F.T., Weis, W.I., Flaherty, K.M., & Brunger, A.T. (1996). Direct observation of protein solvation and discrete disorder with experimental crystallographic phases, *Science* **271**, 72–77.

Graham, I.S. (1995). The HTML Sourcebook, John Wiley and Sons.

Hendrickson, W.A., & Lattman, E.E. (1970). Representation of phase probability distributions for simplified combination of independent phase information. *Acta Cryst. B* **26**, 136–143.

Hendrickson, W.A. (1979). Phase information form anomalous–scattering measurements. *Acta Cryst. A* **35**, 245–247.

Hendrickson, W.A. (1991). Determination of macromolecular structures from anomalous diffraction of synchrotron radiation, *Science* **254**, 51–58.

Jiang, J.-S. & Brunger, A.T. (1994). Protein hydration observed by x-ray diffraction: solvation properties of penicillopepsin and neuraminidase crystal structures, *J. Mol. Biol* **243**, 100–115.

Kleywegt, G.J. & Brunger, A.T. (1996). Checking your imagination: applications

of the free R value, *Structure* **4**, 897–904.

Otwinowski, Z. (1991). *in:* Proc. CCP4 study weekend 25-26 January 1991 (W. Wolf, P.R. Evans, A.G.W. Leslie, eds.), SERC Daresbury laboratory, 80–85.

Pannu, N.S. & Read, R.J. (1996). Improved structure refinement through maximum likelihood, *Acta Cryst. A* **52**, 659–668.

Pannu, N.S., Murshudov, G.N., Dodson, E.J., & Read, R.J. (1998). Incorporation of prior phase information strengthens maximum likelihood structural refinement. *Acta Cryst. D*, in press.

Read, R.J. (1986). Improved Fourier coefficients for maps using phases from partial structures with errors. *Acta Crystallogr. A* **42**, 140–149.

Read, R.J. (1990). Structure–factor probabilities for related structures. *Acta Cryst. A* **46**, 900-912.

Read, R.J. (1994), Maximum likelihood refinement of heavy atoms. Lecture notes for workshop on Isomorphous Replacement Methods in Macromolecular Crystallography, Americal Crystallographic Association Annual Meeting, 1994, Atlanta, GA, USA.

Read, R.J. (1997). Model phases: probabilities and bias. *Meth. Enzymol.* **278**, 110-128.

Rice, L.M. & Brunger, A.T. (1994). Torsion angle dynamics: reduced variable conformational sampling enhances crystallographic structure refinement, *Proteins: Structure, Function, and Genetics*, **19**, 277–290.

Stout, G.H. & Jensen, L.H. (1989). X–ray structure determination. Wiley Interscience, New York, pp. 33.

Weis, W.I, Brunger, A.T., Skehel, J.J., Wiley, D.C. (1990). Refinement of the

Influenza Virus Haemagglutinin by Simulated Annealing, *J. Mol. Biol.* **212**, 737–761.

Figure Captions

Figure 1: CNS consists of five layers which are under user control. The high–level HTML graphical interface interacts with the task–oriented input files. The task files make use of the CNS language and the modules. The modules contain CNS language statements. The CNS language is interpreted by the CNS FORTRAN77 program. The program performs the data manipulations, data operations, and "hard–wired" algorithms.

Figure 2: Examples of compound symbols, compound parameters. (a) The "evaluate" statement is used to define typed symbols (strings, numbers, and logicals). Symbol names are indicated in bold. (b) The "define" statement is used to define untyped parameters. Each parameter entry is terminated by a semi–colon. The compound base name "crystal_lattice" has a number of sub–levels such as "space_group" and the "unit_cell" parameters. "unit_cell" is itself base to a number of sub–levels, such as "a" and "alpha". Parameter names are indicated in bold.

Figure 3: Example for statistical operations provided by the CNS language. "norm", "sigacv", and "save", and "sum" are functions that are computed internally by the CNS program. Bin–wise operations are indicated in italics ("sigacv", "save", and "sum"). The result for a particular bin is stored in all elements belonging to the bin. The σ_A ("sigmaA") parameters are computed in bin–wise resolution shells. The σ_Δ ("sigmaD") and D parameters are then computed from σ_A and bin–wise averages involving $|\mathbf{F}_o|^2$ and $|\mathbf{F}_c|^2$. The bin–wise results are expanded to all reflections by the last three statements. The last three operations expand the bin–wise values to all reflections. "test" is an array that is one for all reflections in the test set and zero otherwise. "sum" is a bin–wise operation on all reflections with the same partitioning as used for the test set.

Figure 4: Examples for symbolic definition of a refinement target function and its derivatives with respect to the calculated structure factor arrays. (a) Simultaneous refinement of heavy atom sites of three derivatives. The target function is defined by the "target" expression. "f_h_1", "f_h_2", and "f_h_3" (indicated in bold) are complex structure factors corresponding to three sets of heavy atoms that are specified using atom selections (Eq. 7). The target function and its derivatives with respect to the three structure factor arrays are defined symbolically using the structure factor amplitudes of the native crystal "f_p", those of the derivatives "f_ph_1", "f_ph_2", "f_ph_3", the complex structure factors of the heavy atom models "f_h_1", "f_h_2", "f_h_3", and the corresponding lack–of–closure variances "v_1", "v_2", and "v_3". The summation over the selected stucture factors ("tselection") is performed implicitly. (b) Refinement of two independent models against perfectly twinned data. "fcalc1" and "fcalc2" are complex structure factors for the models that are related by a twinning operation (indicated in bold). The target function and its derivatives with respect to the two structure factor arrays are explicitly defined.

Figure 5: Use of compound parameters within a module. This module computes the unit cell volume (Stout & Jensen, 1989) from the unit cell geometry. Input and output parameter base names are indicated in bold. Local symbols, such as $cabg.1 are defined through "evaluate" statements. The result is stored in the parameter "&volume" which is passed to the invoking task file or module.

Figure 6: Example of a CNS module (a) and the corresponding module invocation (b). Input and output parameters are indicated in bold. The module invocation is performed by specifying the "@" character followed by the name of the module file and the module parameter substitutions. The ampersand (&) indicates that the particular symbol (e.g., "&fp") is substituted with the specified value in the invocation statement (e.g., "fobs" in the case of "&fp" in (b)). The module parameter substitution is performed literally and any string of characters between the equal sign and the semicolon will be substituted.

Figure 7: Procedures and features available in CNS for structure determination by X–ray crystallography.

Figure 8: Example of a typical CNS task file: a section of the top portion of the simulated annealing refinement protocol which contains the definition of various parameters that are needed in the main body of the task file. Each parameter is indicated by a name, an equal sign, and an arbitrary sequence of characters terminated by a semicolon (e.g., "a=61.76;"). The top portion of the task files also contain commands for the HTML–interface embedded in comment fields (indicated by braces "{ ... }"). The commands that can be modified by the user in the HTML form are indicated in bold.

Figure 9: (a) Example of a CNS HTML form page. This particular example corresponds to the task file in Fig. 8. (b) Use of the CNS HTML form page interface, emphasizing the correspondence between input fields in the form page and parameters in the task file.

Fig. 1

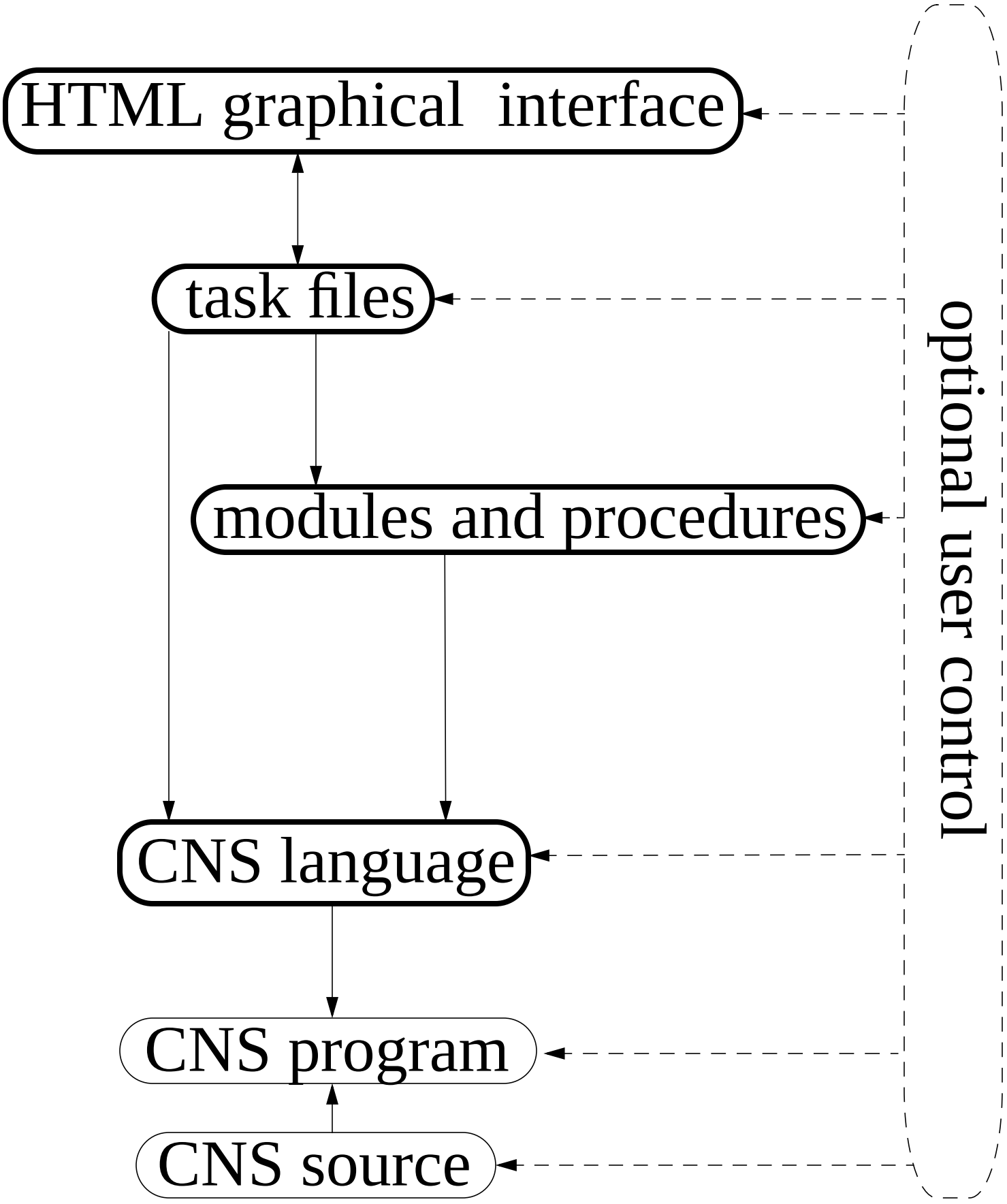

HTML graphical interface
task files
modules and procedures
CNS language
CNS program
CNS source
optional user control

Fig. 2

```
evaluate ( $crystal_lattice.space_group     = "P2(1)2(1)2(1)" )
evaluate ( $crystal_lattice.unit_cell.a     = 61.76           )
evaluate ( $crystal_lattice.unit_cell.b     = 40.73           )
evaluate ( $crystal_lattice.unit_cell.c     = 26.74           )
evaluate ( $crystal_lattice.unit_cell.alpha = 90              )
evaluate ( $crystal_lattice.unit_cell.beta  = 90              )
evaluate ( $crystal_lattice.unit_cell.gamma = 90              )
```

(a)

```
define (
   &crystal_lattice.space_group     = P2(1)2(1)2(1) ;
   &crystal_lattice.unit_cell.a     = 61.76 ;
   &crystal_lattice.unit_cell.b     = 40.73 ;
   &crystal_lattice.unit_cell.c     = 26.74 ;
   &crystal_lattice.unit_cell.alpha = 90 ;
   &crystal_lattice.unit_cell.beta  = 90 ;
   &crystal_lattice.unit_cell.gamma = 90 ;
)
```

(b)

Fig. 3

```
do (eobs=norm(amplitude(fobs)))                                  ( test_set )
do (ecalc=norm(amplitude(fcalc)))                                ( test_set )
do (sigmaA=sigacv(eobs,ecalc))                                   ( test_set )
do (sigmaD=sqrt(save(amplitude(fobs))^2 (1-sigmaA^2)))           ( test_set )
do (D= sigmaA * sqrt(save(amplitude(fobs))^2  /
           save(amplitude(fcalc)^2)))                            ( test_set )

do (sigmaA=sum(sigmaA*test)/ max(1,sum(test)))                   ( all )
do (sigmaD=sum(sigmaD*test)/ max(1,sum(test)))                   ( all )
do (D=sum(D*test)/ max(1,sum(test)))                             ( all )
```

Fig. 4a

```
associate f_h_1 <atom-selection-1>
associate f_h_2 <atom-selection-2>
associate f_h_3 <atom-selection-3>

target=(
         (abs(f_h_1+f_p)-f_ph_1)^2 / (2*v_1)
         (abs(f_h_2+f_p)-f_ph_2)^2 / (2*v_2)
         (abs(f_h_3+f_p)-f_ph_3)^2 / (2*v_3)
       )

dtarget(f_h_1)=
        (
         2*(abs(f_h_1+f_p)-f_ph_1)
           *(f_h_1+f_p)/abs(f_h_1+f_p) / (2*v_1)
         )

dtarget(f_h_2)=
        (
         2*(abs(f_h_2+f_p)-f_ph_2)
           *(f_h_2+f_p)/abs(f_h_2+f_p) / (2*v_2)
         )

dtarget(f_h_3)=
        (
         2*(abs(f_h_3+f_p)-f_ph_3)
           *(f_h_3+f_p)/abs(f_h_3+f_p) / (2*v_3)

tselection=<selection>
cvselection=<selection>
```

Fig. 4b

```
associate fcalc1 <atom-selection1>
associate fcalc2 <atom-selection2>

target=( abs(fobs) - sqrt(abs(fcalc1)^2+abs(fcalc2)^2))^2 )

dtarget(fcalc1)=( 4* (
            abs(fobs-sqrt(abs(fcalc1)^2+abs(fcalc2)^2))
            abs(fcalc1)/(sqrt(abs(fcalc1)^2+abs(fcalc2)^2))

dtarget(fcalc2)=( 4* (
            abs(fobs-sqrt(abs(fcalc1)^2+abs(fcalc2)^2))
            abs(fcalc2)/(sqrt(abs(fcalc1)^2+abs(fcalc2)^2))

tselection=<selection>
cvselection=<selection>
```

Fig. 5

```
module { compute_unit_cell_volume }
(
   &cell;
   &volume;
)


evaluate ( $cabg.1=cos(&cell.alpha) )
evaluate ( $sabg.1=sin(&cell.alpha) )

evaluate ( $cabg.2=cos(&cell.beta) )
evaluate ( $sabg.2=sin(&cell.beta) )

evaluate ( $cabg.3=cos(&cell.gamma) )
evaluate ( $sabg.3=sin(&cell.gamma) )

evaluate ( &volume=&cell.a * &cell.b * &cell.c *
                  sqrt(1+2*$cabg.1*$cabg.2*$cabg.3
                        -$cabg.1^2-$cabg.2^2-$cabg.3^2) )
```

Fig. 6a

```
module { phase_distribution }
     (
   &fp;  {input: native data}
   &sp;  {input: native sigma}
   &sel; {input: selection of structure factors}
   &fh;  {input: name of heavy atom structure factors}
   &fph; {input: name of derivative data array}
   &sph; {input: name of derivative's sigma array}
   &var; {input: lack-of-isomorphism plus measurement errors}
   &pa;  {output: Hendrickson and Lattman A array}
   &pb;  {output: Hendrickson and Lattman B array}
   &pc;  {output: Hendrickson and Lattman C array}
   &pd;  {output: Hendrickson and Lattman D array}
     )

do (&pa= (cos(centric_phase)*(
        -(abs(&fh+combine(abs(&fp),centric_phase))-abs(&fph))^2/(2*&var)
        +(abs(&fh-combine(abs(&fp),centric_phase))-abs(&fph))^2/(2*&var)))
  ( centric and &sel )

do (&pb=(sin(centric_phase)*(
        -(abs(&fh+combine(abs(&fp),centric_phase))-abs(&fph))^2/(2*&var)
        +(abs(&fh-combine(abs(&fp),centric_phase))-abs(&fph))^2/(2*&var)))
  ( centric and &sel )

do (&pc=0) (centric and &sel)

do (&pd=0) (centric and &sel)

do (&pa=-2 * (amplitude(&fp)^2 + amplitude(&fh)^2 - amplitude(&fph)^2 )
           * amplitude(&fp) * real(&fh)/
           (3 * &var^2 + 4 * (amplitude(&fph)^2+&sph^2) * &var))
  ( acentric and &sel )

do (&pb=-2 * (amplitude(&fp)^2 + amplitude(&fh)^2 - amplitude(&fph)^2 )
           * amplitude(&fp) * imag(&fh)/
           (3 * &var^2 + 4 * (amplitude(&fph)^2+&sph^2) * &var) )
  ( acentric and &sel )

do (&pc=-amplitude(&fp)^2 * (real(&fh)^2 - imag(&fh)^2) /
         (3 * &var^2 + 4 * (amplitude(&fph)^2+&sph^2) * &var) )
  ( acentric and &sel )

do (&pd=-2 * amplitude(&fp)^2 * real(&fh) * imag(&fh) /
        (3 * &var^2 + 4 * (amplitude(&fph)^2+&sph^2) * &var) )
  ( acentric and &sel )
```

Fig. 6b

```
@phase_distribution
  (
   &fp=fobs;
   &sp=sigma;
   &sel=( d > 3. );
   &fh=f_heavy;
   &fph=f_deriv;
   &sph=s_deriv;
   &var=variance;
   &pa=pa;
   &pb=pb;
   &pc=pc;
   &pd=pd;
  )
```

Fig. 7

Experimental Phasing
- heavy atom (Patterson) searches
- Patterson refinement
- multiple-isomorphous replacement phasing and site refinement
- multi-wavelength anomalous dispersion phasing and site refinement

Molecular Replacement
- Patterson real-space and direct rotation searches
- Patterson-correlation refinement
- fast FFT-translation search

Density Modification
- creation of envelopes
- solvent-flattening
- density averaging
- histogram matching

Refinement
- maximum likelihood targets
- torsion-angle molecular dynamics
- Cartesian molecular dynamics
- conjugate gradient minimization
- composite annealed omit map

Other
- Protein Data Bank deposition file generation
- mmCIF file creation

Fig. 8

```
{+ file: anneal.inp +}
{+ description: Crystallographic simulated annealing refinement +}
{+ authors: Axel T. Brunger, Luke M. Rice and Paul D. Adams +}
{+ reference: A.T. Brunger, J. Kuriyan and M. Karplus, Crystallographic
              R factor Refinement by Molecular Dynamics, Science
              235, 458-460 (1987) +}
{+ reference: A.T. Brunger, A. Krukowski and J. Erickson, Slow-Cooling
              Protocols for Crystallographic Refinement by Simulated
              Annealing, Acta Cryst. A46, 585-593 (1990) +}
{- begin block parameter definition -} define(
{====================== crystallographic data ======================}
{* space group *}
{* use International Table conventions with subscripts substituted by
   parenthesis *}
{===>} sg="P2(1)2(1)2(1)";

{* unit cell *}
{===>} a=61.76;   {===>} b=40.73;  {===>} c=26.74;
{===>} alpha=90;  {===>} beta=90;  {===>} gamma=90;

{* anomalous f' f'' library file *}
{* should be used when refining against anomalous data -
   libraries: "CNS_XTALLIB:anom_cu.lib" and "CNS_XTALLIB:anom_mo.lib" or
              a user created file.
   If blank no anomalous contribution will be included in the refinement *}
{===>} anom_library="";

{* reflection file *}
{===>} ref="example.hkl";

{* reciprocal space array containing observed amplitudes: required *}
{===>} obs_f="f_native";

{* reciprocal space array containing sigma values for amplitudes: required *}
{===>} obs_sigf="s_native";

{* reciprocal space array containing test set for cross-validation: required *}
{===>} test_set="test";
{* refinement target *}
{* mlf: maximum likelihood target using amplitudes
   mli: maximum likelihood target using intensities
   mlhl: maximum likelihood target using amplitudes and phase probability
         distribution
   residual: standard crystallographic residual
   vector: vector residual
   mixed: (1-fom)*residual + fom*vector
   e2e2: correlation coefficient using normalized E^2
   e1e1: correlation coefficient using normalized E
   f2f2: correlation coefficient using F^2
   f1f1: correlation coefficient using F *}
{+ choice: "mlf" "mli" "mlhl" "residual" "vector" "mixed"
           "e2e2" "e1e1" "f2f2" "f1f1" +}
{===>} reftarget="mlf";
}  {- end block parameter definition -}
```

Fig. 9a

Authors

- Axel T. Brunger, Luke M. Rice and Paul D. Adams

References

- A.T. Brunger, J. Kuriyan and M. Karplus, Crystallographic R factor Refinement by Molecular Dynamics, Science 235, 458–460 (1987)
- A.T. Brunger, A. Krukowski and J. Erickson, Slow-Cooling Protocols for Crystallographic Refinement by Simulated Annealing, Acta Cryst. A46, 585–593 (1990)

crystallographic data

space group
use International Table conventions with subscripts substituted by parenthesis: P2(1)2(1)2(1)

unit cell parameters in Angstroms and degrees

	a	b	c	alpha	beta	gamma
cell	61.76	40.73	26.74	90	90	90

anomalous f' f'' library file:

reflection file: example.hkl

reciprocal space array containing observed amplitudes: required: f_native

reciprocal space array containing sigma values for amplitudes: required: s_native

reciprocal space array containing test set for cross-validation: required: test

refinement target

mlf: maximum likelihood target using amplitudes
mli: maximum likelihood target using intensities
mlhl: maximum likelihood target using amplitudes and phase probability distribution
residual: standard crystallographic residual vector: vector residual
*mixed: (1-fom)*residual + fom*vector*
e2e2: correlation coefficient using normalized E^2
e1e1: correlation coefficient using normalized E
f2f2: correlation coefficient using F^2
f1f1: correlation coefficient using F

mlf

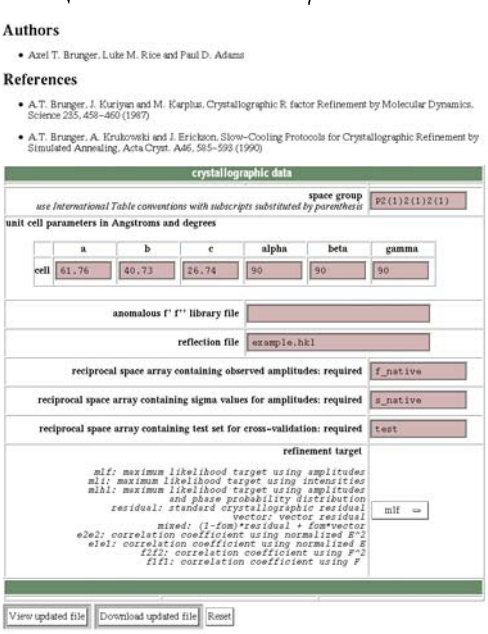
View updated file | Download updated file | Reset

Fig. 9b

world-wide-web

personal task files

conversion from
HTML form to task file



Authors

- Axel T. Brunger, Luke M. Rice and Paul D. Adams

References

- A.T. Brunger, J. Kuriyan and M. Karplus, Crystallographic R factor Refinement by Molecular Dynamics, Science 235, 458–460 (1987)
- A.T. Brunger, A. Krukowski and J. Erickson, Slow-Cooling Protocols for Crystallographic Refinement by Simulated Annealing, Acta Cryst. A46, 585–593 (1990)

crystallographic data

space group: P2(1)2(1)2(1)

use International Table conventions with subscripts substituted by parenthesis

unit cell parameters in Angstroms and degrees

	a	b	c	alpha	beta	gamma
cell	61.76	40.73	26.74	90	90	90

anomalous f' f'' library file:

reflection file: example.hkl

reciprocal space array containing observed amplitudes: required: f_native

reciprocal space array containing sigma values for amplitudes: required: s_native

reciprocal space array containing test set for cross-validation: required: test

refinement target

mlf: maximum likelihood target using amplitudes
mli: maximum likelihood target using intensities
mlhl: maximum likelihood target using amplitudes and phase probability distribution
residual: standard crystallographic residual
vector: vector residual
*mixed: (1-fom)*residual + fom*vector*
e2e2: correlation coefficient using normalized E^2
e1e1: correlation coefficient using normalized E
f2f2: correlation coefficient using F^2
f1f1: correlation coefficient using F

mlf

View updated file | Download updated file | Reset

```
{+ file: anneal.inp +}
{+ description: Crystallographic simulated annealing refinement +}
{+ authors: Axel T. Brunger, Luke M. Rice and Paul D. Adams +}
{+ reference: A.T. Brunger, J. Kuriyan and M. Karplus, Crystallographic
              R factor Refinement by Molecular Dynamics, Science
              235, 458-460 (1987) +}
{+ reference: A.T. Brunger, A. Krukowski and J. Erickson, Slow-Cooling
              Protocols for Crystallographic Refinement by Simulated
              Annealing, Acta Cryst. A46, 585-593 (1990) +}
{- begin block parameter definition -} define(
{====================== crystallographic data ======================}
{* space group *}
{* use International Table conventions with subscripts substituted by
   parenthesis *}
{===>} sg="P2(1)2(1)2(1)";

{* unit cell *}
{===>} a=61.76;  {===>} b=40.73;  {===>} c=26.74;
{===>} alpha=90;  {===>} beta=90;  {===>} gamma=90;

{* anomalous f' f'' library file *}
{* should be used when refining against anomalous data -
   libraries: "CNS_XTALLIB:anom_cu.lib" and "CNS_XTALLIB:anom_mo.lib" or
              a user created file.
   If blank no anomalous contribution will be included in the refinement *}
{===>} anom_library="";

{* reflection file *}
{===>} ref="example.hkl";

{* reciprocal space array containing observed amplitudes: required *}
{===>} obs_f="f_native";

{* reciprocal space array containing sigma values for amplitudes: required *}
{===>} obs_sigf="s_native";

{* reciprocal space array containing test set for cross-validation: required *}
{===>} test_set="test";
{* refinement target *}
{* mlf: maximum likelihood target using amplitudes
   mli: maximum likelihood target using intensities
   mlhl: maximum likelihood target using amplitudes and phase probability
         distribution
   residual: standard crystallographic residual
   vector: vector residual
   mixed: (1-fom)*residual + fom*vector
   e2e2: correlation coefficient using normalized E^2
   e1e1: correlation coefficient using normalized E
   f2f2: correlation coefficient using F^2
   f1f1: correlation coefficient using F *}
{+ choice: "mlf" "mli" "mlhl" "residual" "vector" "mixed"
           "e2e2" "e1e1" "f2f2" "f1f1" +}
{===>} reftarget="mlf";
}  {- end block parameter definition -}
```

conversion from
task file to HTML form

distributed task files