Acta Cryst. (1998). D54, 905–921

Crystallography & NMR System: A New Software Suite for Macromolecular Structure Determination

Axel T. Brünger,[a,b]* Paul D. Adams,[b] G. Marius Clore,[c] Warren L. DeLano,[d] Piet Gros,[e] Ralf W. Grosse-Kunstleve,[a,b] Jian-Sheng Jiang,[f] John Kuszewski,[c] Michael Nilges,[g] Navraj S. Pannu,[h] Randy J. Read,[i] Luke M. Rice,[b] Thomas Simonson[j] and Gregory L. Warren[b]

[a]*The Howard Hughes Medical Institute, Yale University, New Haven, CT 06511, USA,* [b]*Department of Molecular Biophysics and Biochemistry, Yale University, New Haven, CT 06511, USA,* [c]*Laboratory of Chemical Physics, Building 5, National Institute of Diabetes and Digestive and Kidney Diseases, National Institutes of Health, Bethesda, MD 20892-0520, USA,* [d]*Graduate Group in Biophysics, Box 0448, University of California, San Francisco, CA 94143, USA,* [e]*Crystal and Structural Chemistry, Bijvoet Center for Biomolecular Research, Utrecht University, Padualaan 8, 3584 CH Utrecht, The Netherlands,* [f]*Protein Data Bank, Biology Department, Brookhaven National Laboratory, Upton, NY 11973-5000, USA,* [g]*European Molecular Biology Laboratory, Meyerhofstrasse 1, D-69117 Heidelberg, Germany,* [h]*Department of Mathematical Sciences, University of Alberta, Edmonton, Alberta T6G 2G1, Canada,* [i]*Department of Medical Microbiology and Immunology, University of Alberta, Edmonton, Alberta T6G 2H7, Canada, and* [j]*Laboratoire de Biologie Structurale (CNRS), IGBMC, 1 rue Laurent Fries, 67404 Illkirch (CU de Strasbourg), France.*
E-mail: brunger@laplace.csb.yale.edu

(Received 22 January 1998; accepted 23 February 1998)

Abstract

A new software suite, called *Crystallography & NMR System* (*CNS*), has been developed for macromolecular structure determination by X-ray crystallography or solution nuclear magnetic resonance (NMR) spectroscopy. In contrast to existing structure-determination programs the architecture of *CNS* is highly flexible, allowing for extension to other structure-determination methods, such as electron microscopy and solid-state NMR spectroscopy. *CNS* has a hierarchical structure: a high-level hypertext markup language (HTML) user interface, task-oriented user input files, module files, a symbolic structure-determination language (*CNS* language), and low-level source code. Each layer is accessible to the user. The novice user may just use the HTML interface, while the more advanced user may use any of the other layers. The source code will be distributed, thus source-code modification is possible. The *CNS* language is sufficiently powerful and flexible that many new algorithms can be easily implemented in the *CNS* language without changes to the source code. The *CNS* language allows the user to perform operations on data structures, such as structure factors, electron-density maps, and atomic properties. The power of the *CNS* language has been demonstrated by the implementation of a comprehensive set of crystallographic procedures for phasing, density modification and refinement. User-friendly task-oriented input files are available for nearly all aspects of macromolecular structure determination by X-ray crystallography and solution NMR.

1. Introduction

During the past four decades, macromolecular X-ray crystallography has undergone dramatic development. Advances in molecular biology, crystallization screens, data collection, phasing methods, computer graphics and refinement have produced a nearly exponential growth of the number of X-ray crystal structures solved. Equally dramatic development has occurred in structure determination by solution NMR. Both methods continue to develop: X-ray crystallographers work on larger macromolecular complexes than ever before and NMR spectroscopists are studying macromolecules that were previously thought only to be accessible by X-ray crystallography (for recent reviews see Wagner, 1997; Clore & Gronenborn, 1997). Larger and more challenging problems often require new computational methods to analyze the diffraction or NMR data.

The computer software available for analyzing and interpreting the experimental data is a heterogeneous mixture of often incompatible programs. With regard to the details of the algorithms they use, most programs are poorly documented. Thus, structural biologists are often not provided with enough information to fully understand the operation of the computer programs they are using. At the same time, the complexity of some programs has grown so dramatically that even the most dedicated researcher must spend months in order to understand their finer details.

We have developed a new and advanced software system, called *Crystallography & NMR System* (*CNS*), for crystallographic and NMR structure determination. 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 and NMR. This high-level language allows definition of symbolic target functions, data structures, procedures and modules. The FORTRAN77 code of the *CNS* package 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. User-friendly 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.

Among the new and unique features of *CNS* for crystallographic applications are: automated Patterson-correlation-based heavy-atom searching (RWGK &

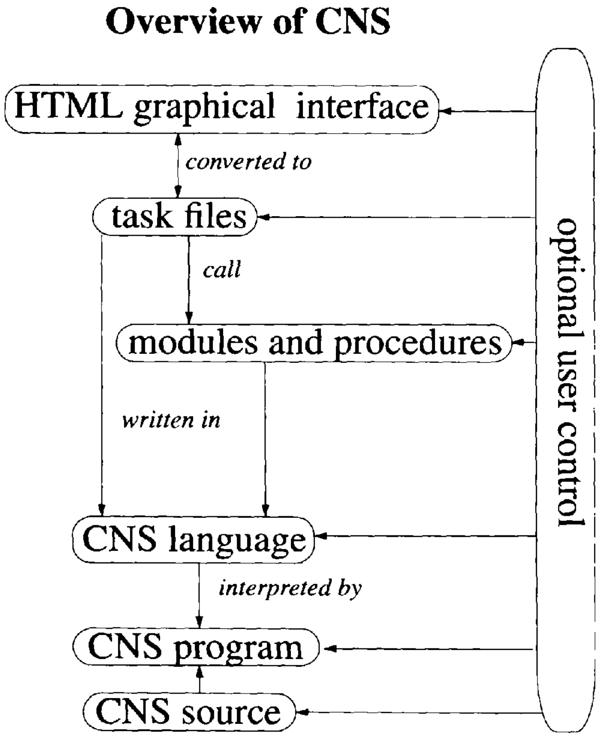


Fig. 1. *CNS* consists of five layers which are under user control. The high-level HTML graphical interface interacts with the task-oriented input files. The task files make use of the *CNS* language and the modules. The modules contain *CNS* language statements. The *CNS* language is interpreted by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

CNS Capabilities

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

NMR Structure Calculation
NOE-derived distance restraints
NOE-intensity restraints
1-bond and 3-bond J-coupling data
α, β carbon and proton chemical shifts
residual dipolar coupling restraints
diffusion anisotropy restraints
dihedral angle restraints
hydrogen-bond distance restraints
simulated annealing structure calculation
refinement

Other
correlated dihedral angle probability conformational database
Protein Data Bank deposition file generation
mmCIF file creation

Fig. 2. Procedures and features available in *CNS* for structure determination by X-ray crystallography and solution NMR.

ATB, unpublished work), a maximum-likelihood implementation of the Phillips & Hodgson (1980) method for multi-wavelength anomalous dispersion (MAD) phasing and refinement (Burling *et al.*, 1996), and combined simulated-annealing/maximum-likelihood model refinement (Adams *et al.*, 1997; Brünger *et al.*, 1997). For NMR structure calculation, one- and three-bond J-coupling (Garrett *et al.*, 1994), carbon and proton chemical shift (Kuszewski, Qin *et al.*, 1995; Kuszewski, Gronenborn *et al.*, 1995; Kuszewski *et al.*, 1996*a*), residual dipolar coupling (Tjandra, Garrett *et al.*, 1997), and rotational diffusion anisotropy (Tjandra, Omichinski *et al.*, 1997; Clore, Groenborn & Tjandra, 1998) data can all be used in addition to nuclear Overhauser effect (NOE) data and torsion-angle restraints (Clore *et al.*, 1985; Nilges, Clore *et al.*, 1988*a*,*b*; Nilges, Gronenborn *et al.*, 1988). The iterative NOE assignment method ARIA (Nilges *et al.*, 1997) will also be implemented in *CNS*. A multi-dimensional database of dihedral angle preferences in proteins and nucleic acids is represented as a pseudo-energy term (Kuszewski *et al.*, 1996*b*, 1997). Powerful optimization methods are available, including simulated-annealing refinement in Cartesian (Brünger *et al.*, 1986, 1987; Clore, Brünger *et al.*, 1986; Brünger, 1988) and in torsion-angle space (Rice & Brünger, 1994). The statistical method of cross-validation is used to monitor the quality of the atomic model. Cross-validated properties include R values (Brünger, 1992), σ_A values (Kleywegt & Brünger, 1996; Read, 1997), NOE intensities, NOE-derived distances (Brünger *et al.*, 1993), and coupling constants (AMJJ & Bonvin, unpublished work).

2. Software description

2.1. *Overview*

CNS consists of five different layers (Fig. 1). The top layer is an HTML graphical interface. It provides easy access to *task*-oriented input files for crystallographic and NMR procedures (Fig. 2) using HTML 'form' pages (Fig. 3*a*). The user can edit fields in the form, and then automatically generate the modified task file. Furthermore, the user can use the HTML interface with 'personal' task files provided they are syntactically correct. The HTML form page is automatically generated from the task file. There is a one-to-one correspondence between HTML form input fields which can be changed and the parameter definitions in the task file (Fig. 3*b*).

The task files make use of a large variety of *CNS modules* for crystallographic and NMR structure determination. The task and module files all make use of the *CNS language*, which is plain ASCII text readable by the user. It allows structured statements and various types of symbol substitutions. Symbolic operations on a variety of data structures can be performed and specific elements selected with a general selection syntax. Data structures that can be manipulated include reciprocal-space arrays [*e.g.* structure factors and Hendrickson–Lattman (Hendrickson & Lattman, 1970) phase-probability distributions], real-space arrays (*e.g.* electron-density maps and masks), and atomic property arrays. It is planned to provide similar operations on NMR data in the future. Transformations or associations from one data structure type (*e.g.* atomic coordinates) to another type (*e.g.* structure-factor array) can be performed. The *CNS language* is interpreted by the *CNS* program which is written in FORTRAN77.

CNS has been tested on a large number of UNIX platforms, including Hewlett Packard HP 735, Silicon Graphics, CRAY Research, Dec Alpha Unix/OSF, and PCs running LINUX.

2.2. *Source code*

The source code of the *CNS* program is written in FORTRAN77 for UNIX-based operating systems. A few extensions to standard Fortran are used: management of dimension statements in common blocks and structured loop statements, such as 'DO WHILE'. A pre-processor is used to convert these extensions into standard FORTRAN77 code. The source code consists of a highly modular set of subroutines and functions. The main data structures reside in separate common blocks. Dynamic memory allocation is accomplished by use of the C-function 'malloc'. Installation and compilation of the program has been automated by the use of the UNIX 'make' facility.

CNS has version control, *i.e.* the consistency of the version numbers of task and module files is checked against the version of the executing *CNS* program.

2.3. *CNS language*

The *CNS* symbolic structure-determination language resides above the source code. The *CNS* language has some elements which are similar to certain script languages, such as structured control and symbol substitution. One of the key features of the *CNS* language is symbolic data structure manipulation, *e.g.*

```
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
```

(1)

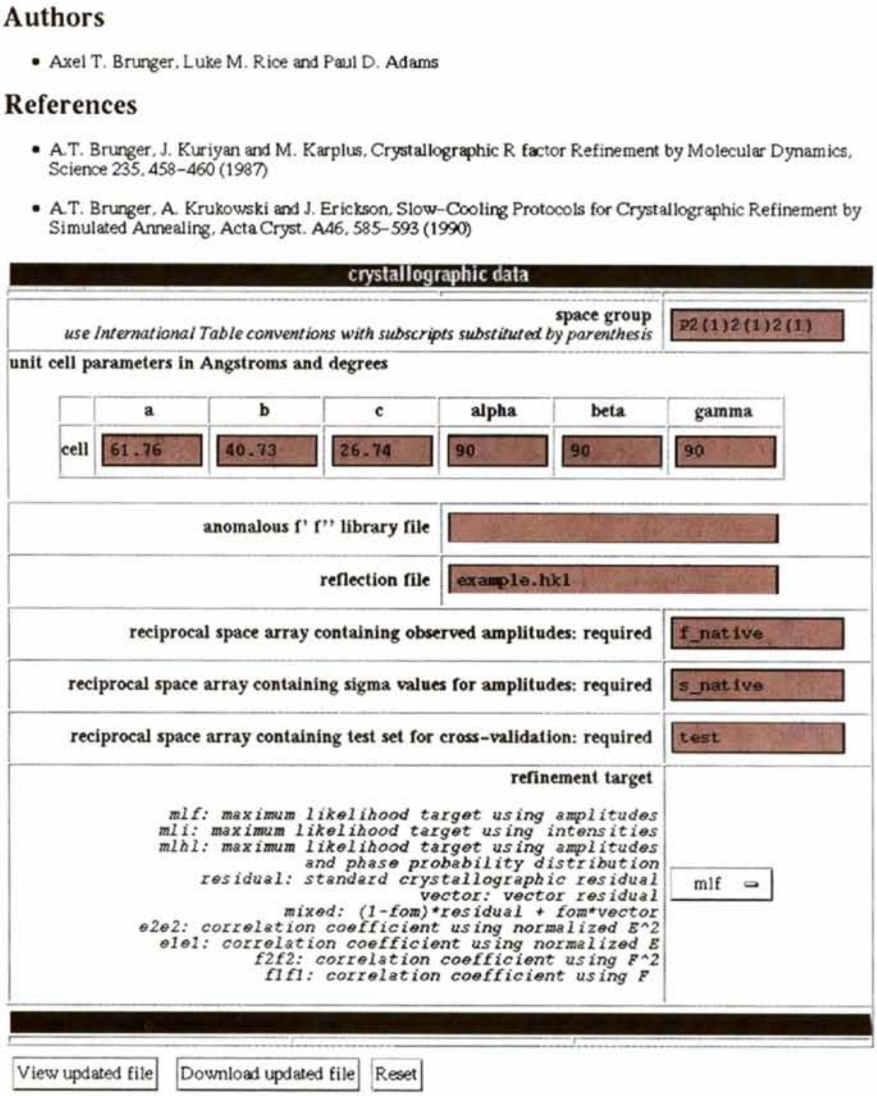
which is equivalent to the following mathematical expression for all acentric indices **h**,

$$p_{\rm a}(\mathbf{h}) = -2\frac{[|\mathbf{f}_{\rm p}(\mathbf{h})|^2 + |\mathbf{f}_{\rm h}(\mathbf{h})|^2 - |\mathbf{f}_{\rm ph}(\mathbf{h})|^2]|\mathbf{f}_{\rm p}(\mathbf{h})|\frac{[\mathbf{f}_{\rm h}(\mathbf{h})+\mathbf{f}_{\rm h}(\mathbf{h})^*]}{2}}{3\nu(\mathbf{h})^2 + 4[|\mathbf{f}_{\rm ph}(\mathbf{h})|^2 + s_{\rm ph}(\mathbf{h})^2] * \nu(\mathbf{h})} \quad (2)$$

where $\mathbf{f}_{\rm p}$ ['fp' in (1)] is the 'native' structure-factor array, $\mathbf{f}_{\rm ph}$ ['fph' in (1)] is the derivative structure-factor array, $s_{\rm ph}$ ['sph' in (1)] the corresponding experimental σ, ν 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 (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 (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.*

$$\texttt{(amplitude(fp)} > 2 * \texttt{sh and amplitude (fph)} > 2 * \texttt{sph and d} > = 3\texttt{)} \quad (3)$$

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 (Brünger, 1992), σ_A values (Kleywegt & Brünger,



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

```
    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
```

(*a*)

Fig. 3. (*a*) Example of a *CNS* HTML form page. This particular example corresponds to the task file in Fig. 6.

1996; Read, 1997), and maximum-likelihood functions (Pannu & Read, 1996; Adams *et al.*, 1997).

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

```
xray
 do (map = 0) (map<0.1)          (4)
end
```

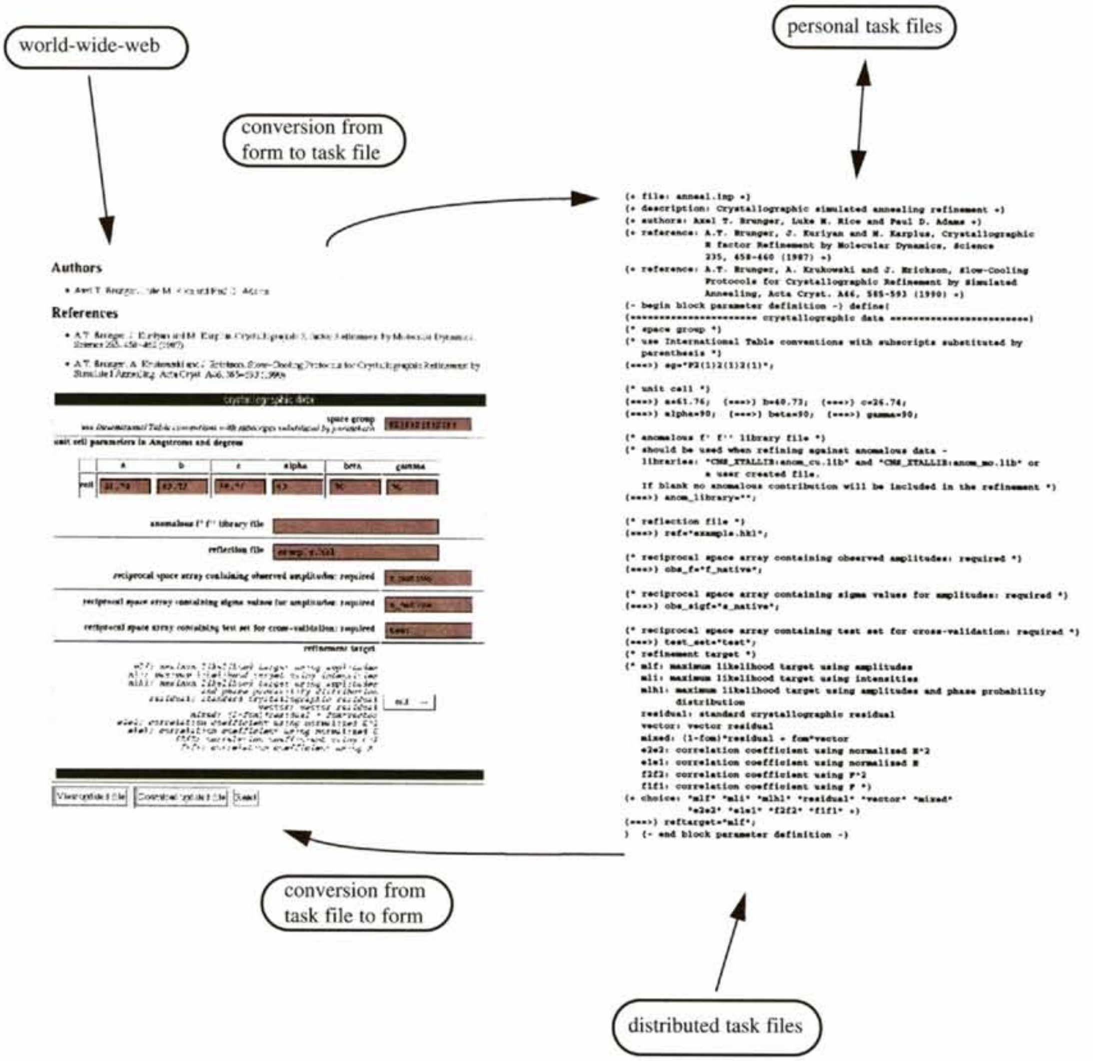
is an example of a truncation operation: all map values less than 0.1 are set to 0. Atoms can be selected based on a number of atomic properties and descriptors, *e.g.*

```
do (b = 10) (residue 1 : 40 and
   (name ca or name n or name c or name o))     (5)
```

sets the B factors of all polypeptide backbone atoms of residues 1–40 to 10 Å^2. Operations exist between arrays, *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

```
xray
 mapresolution infinity 3.
 fft grid 0.3333 end              (6)
 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 reflec-


world-wide-web
conversion from form to task file
personal task files
conversion from task file to form
distributed task files

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 [illegible]
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 [illegible]

reciprocal space array containing observed amplitudes: required [illegible]

reciprocal space array containing sigma values for amplitudes: required [illegible]

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

refinement target [illegible]

```
{+ 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 *}
{===>} refl="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 -}
```

(*b*)

Fig. 3. *cont.* (*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.

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

$$\texttt{associate f\_cal (residue 1 : 50)}. \qquad (7)$$

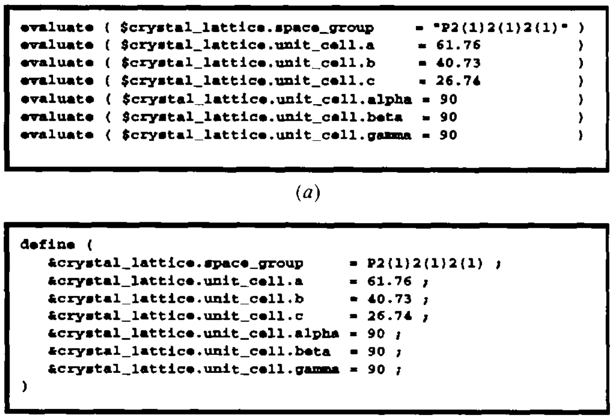
This statement will associate the reciprocal-space array '`f_cal`' with the atoms belonging to residues 1–50. These structure-factor associations are used in the symbolic target functions described below. There are no predefined reciprocal or real-space arrays in *CNS*. Dynamic memory allocation allows one to carry out operations on arbitrarily large data sets with many individual entries (*e.g.* heavy-atom-derivative diffraction data) without the need for re-compilation of the source code. The various reciprocal structure-factor arrays must, therefore, be declared and their type specified prior to invocation. For example, a reciprocal-space array with real values, such as observed amplitudes, is declared by the following expression,

$$\begin{aligned}&\texttt{declare name} = \texttt{fobs type} = \texttt{real}\\ &\quad\texttt{domain} = \texttt{reciprocal end}\end{aligned} \qquad . \qquad (8)$$

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

$$\begin{aligned}&\texttt{group type} = \texttt{hl object} = \texttt{pa}\\ &\quad\texttt{object} = \texttt{pb object} = \texttt{pc}\\ &\quad\texttt{object} = \texttt{pd end}\end{aligned} \qquad (9)$$

where 'pa', 'pb', 'pc', and 'pd' refer to the 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 space group *P*1.



```
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. 4. Examples of compound symbols, compound parameters, and use of compound parameters. (*a*) The '`evaluate`' statement is used to define typed symbols (strings, numbers, and logicals). (*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'. (*c*) Use of compound parameters within a module. This module computes the unit-cell volume (Stout & Jensen, 1989) from the unit-cell geometry. 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.

2.4. *Symbols and parameters*

CNS 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 logical variables. *Parameters* are untyped data elements of arbitrary length that may contain collections of *CNS* commands, numbers, strings, or symbols.

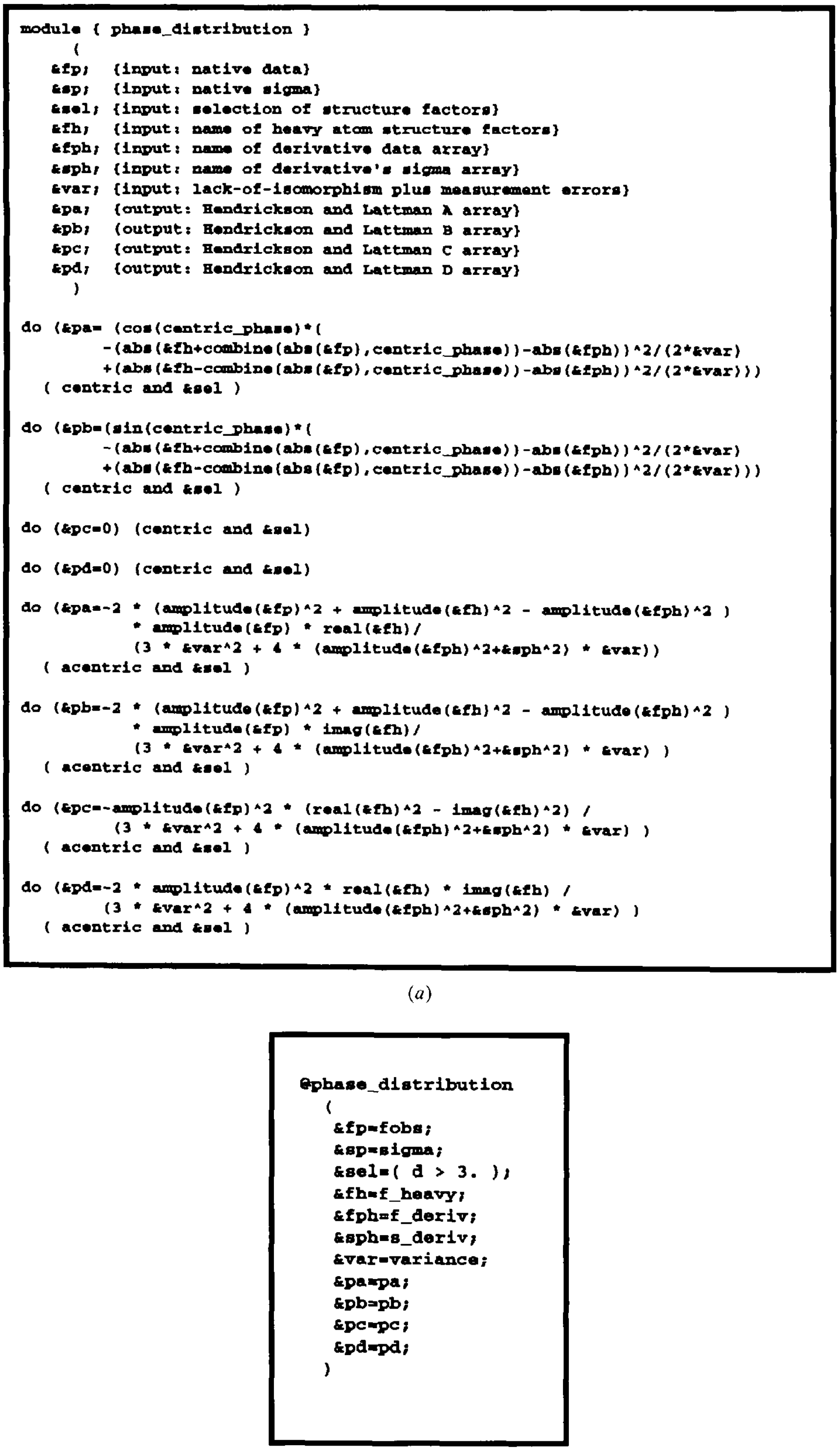
Symbols are denoted by a dollar sign ($) and parameters by an ampersand (&). Symbols and parameters may contain a single data element, or they may represent a *compound* data structure of arbitrary complexity. The hierarchy of these data structures is denoted using a period (.). Figs. 4(*a*) and 4(*b*) demonstrate how crystal lattice information can be stored in compound symbols and parameters, respectively. The information stored in symbols or parameters can be retrieved by simply referring to them within a *CNS* command: the symbol or parameter name is substituted by its content. Symbol substitution of portions of the compound names (*e.g.* '`&crystal_lattice.unit_cell.$para`') allows one to carry out conditional and iterative operations on such data structures, such as matrix multiplication.

2.5. *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 *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. 4*b*) are passed into a



```
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 )
```

(*a*)

```
@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;
  )
```

(*b*)

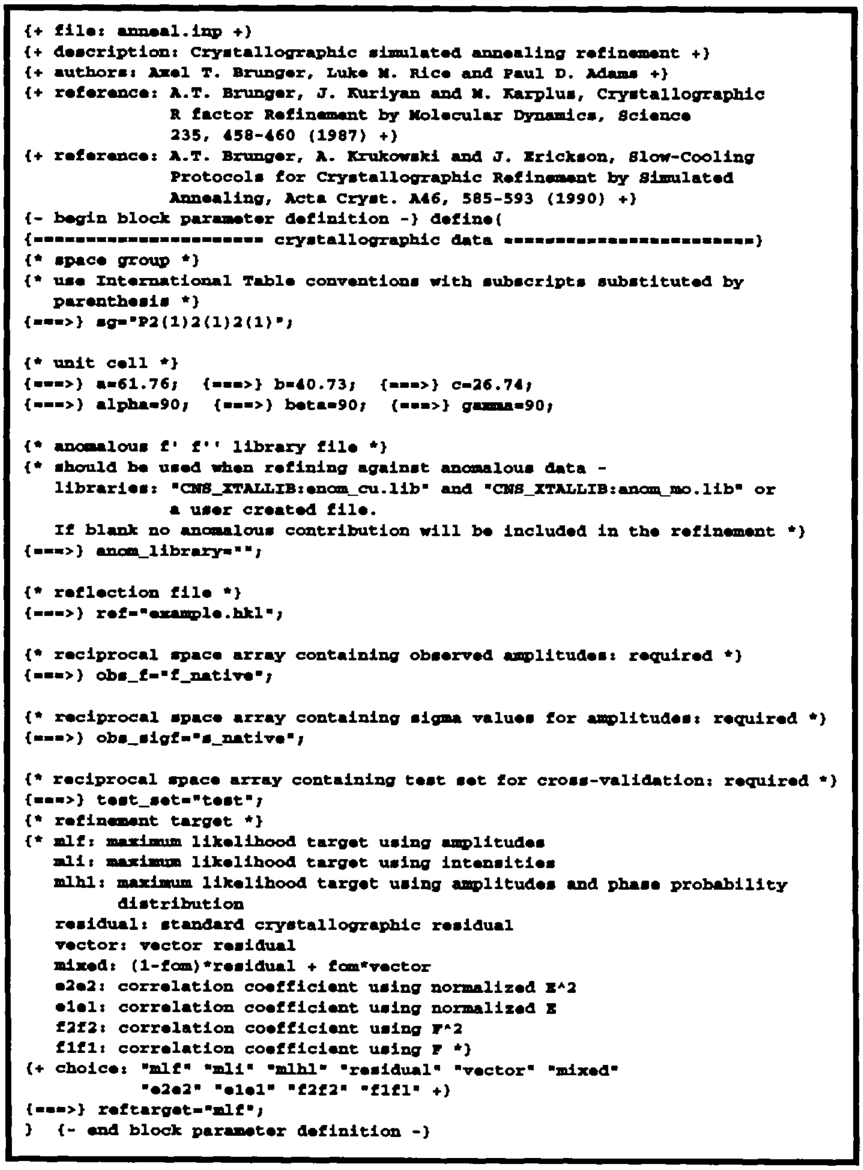
Fig. 5. Example of a *CNS* module (*a*) and the corresponding module invocation (*b*). The module invocation is performed by specifying the '@' character followed by the name of the module file and the module parameter substitutions. The ampersand (&) indicates that the particular symbol (*e.g.* '&fp') 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 equals sign and the semicolon will be substituted.

module named 'compute_unit_cell_volume' (Fig. 4*c*) which computes the volume of the unit cell from the crystal lattice parameters using well established formulae (Stout & Jensen, 1989),

```
@compute_unit_cell_volume
(cell = &crystal_lattice.unit_cell;        (10)
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. 4*b*) 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.

Fig. 5(*a*) shows an 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. 5(*b*). This module could be called from task files that need access to isomorphous phase-probability distributions. It would be straightforward to change the module in order to compute different expressions for the phase-probability distributions.



```
{+ 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. 6. Example of a typical *CNS* task file: 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 equals 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 directives for the HTML interface embedded in comment fields (indicated by braces '{...}').

A large number of additional modules are available for crystallographic phasing and refinement, and for NMR structure calculation.

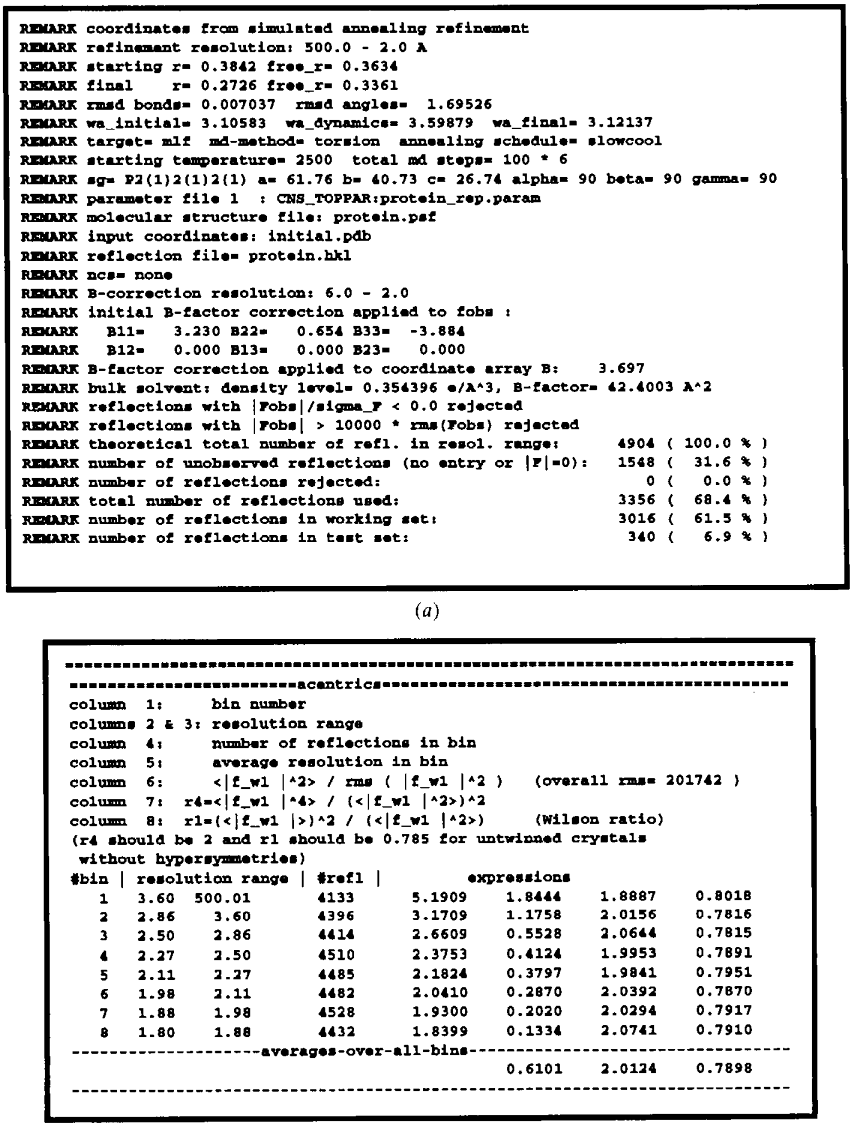
2.6. *Library modules*

CNS library modules include space-group information, Gaussian atomic form factors, anomalous scattering components, NMR random-coil chemical shifts, molecular parameter and topology databases, and a conformational database which contains multi-dimensional probability distributions for preferred rotamers in proteins and nucleic acids (Kuszewski *et al.*, 1997).

2.7. *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 and NMR structure determination using a minimal set of 'hard-wired' functions and routines. A list of the currently available procedures and features is shown in Fig. 2. The list excludes data reduction and three-dimensional graphics which are outside the scope of *CNS*.

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



```
REMARK coordinates from simulated annealing refinement
REMARK refinement resolution: 500.0 - 2.0 A
REMARK starting r= 0.3842 free_r= 0.3634
REMARK final    r= 0.2726 free_r= 0.3361
REMARK rmsd bonds= 0.007037  rmsd angles=  1.69526
REMARK wa_initial= 3.10583  wa_dynamics= 3.59879  wa_final= 3.12137
REMARK target= mlf  md-method= torsion  annealing schedule= slowcool
REMARK starting temperature= 2500  total md steps= 100 * 6
REMARK sg= P2(1)2(1)2(1) a= 61.76 b= 40.73 c= 26.74 alpha= 90 beta= 90 gamma= 90
REMARK parameter file 1  : CNS_TOPPAR:protein_rep.param
REMARK molecular structure file: protein.psf
REMARK input coordinates: initial.pdb
REMARK reflection file= protein.hkl
REMARK ncs= none
REMARK B-correction resolution: 6.0 - 2.0
REMARK initial B-factor correction applied to fobs :
REMARK   B11=   3.230 B22=   0.654 B33=  -3.884
REMARK   B12=   0.000 B13=   0.000 B23=   0.000
REMARK B-factor correction applied to coordinate array B:    3.697
REMARK bulk solvent: density level= 0.354396 e/A^3, B-factor= 42.4003 A^2
REMARK reflections with |Fobs|/sigma_F < 0.0 rejected
REMARK reflections with |Fobs| > 10000 * rms(Fobs) rejected
REMARK theoretical total number of refl. in resol. range:      4904 ( 100.0 % )
REMARK number of unobserved reflections (no entry or |F|=0):   1548 (  31.6 % )
REMARK number of reflections rejected:                            0 (   0.0 % )
REMARK total number of reflections used:                       3356 (  68.4 % )
REMARK number of reflections in working set:                   3016 (  61.5 % )
REMARK number of reflections in test set:                       340 (   6.9 % )
```

(*a*)

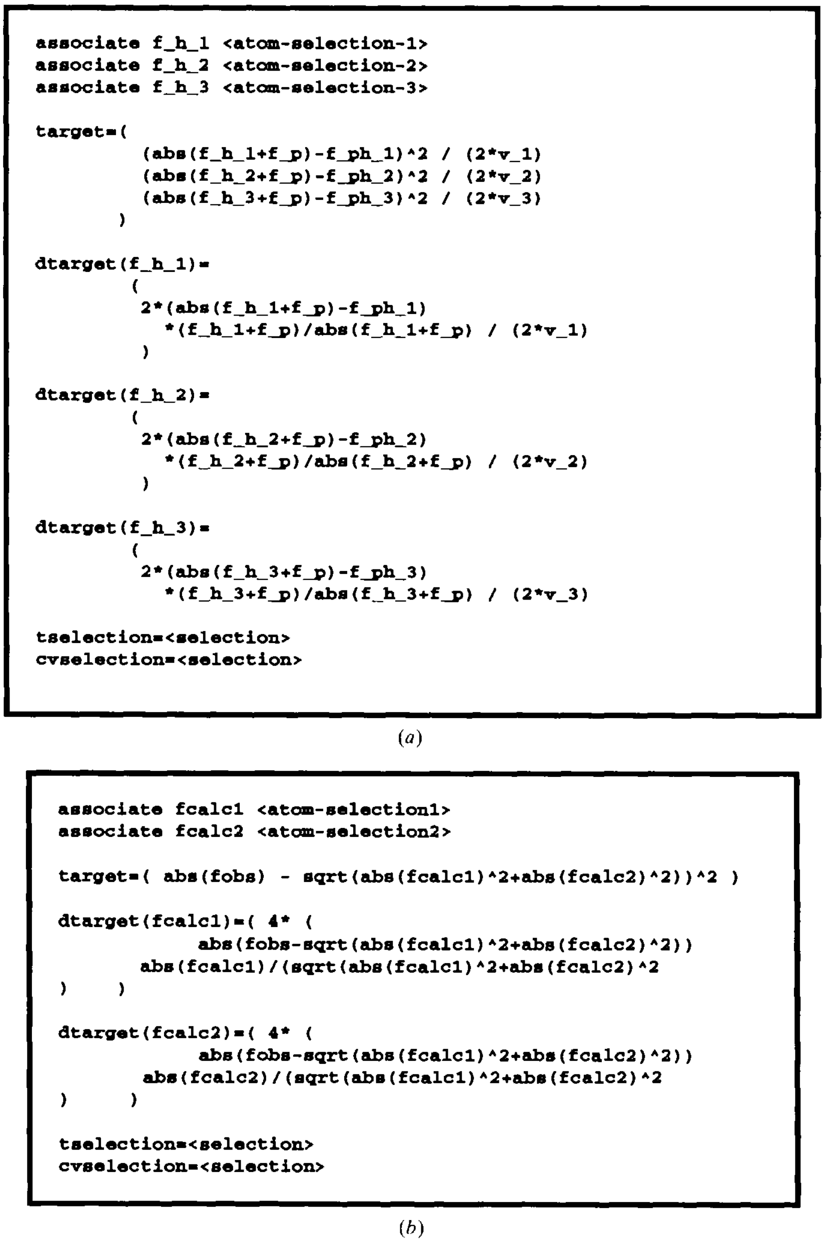
```
============================================================================
========================acentrics===========================================
column  1:     bin number
columns 2 & 3: resolution range
column  4:     number of reflections in bin
column  5:     average resolution in bin
column  6:     <|f_wl |^2> / rms ( |f_wl |^2 )   (overall rms= 201742 )
column  7:  r4=<|f_wl |^4> / (<|f_wl |^2>)^2
column  8:  rl=(<|f_wl |>)^2 / (<|f_wl |^2>)     (Wilson ratio)
(r4 should be 2 and rl should be 0.785 for untwinned crystals
 without hypersymmetries)
#bin | resolution range | #refl |          expressions
   1   3.60  500.01        4133       5.1909     1.8444     1.8887     0.8018
   2   2.86    3.60        4396       3.1709     1.1758     2.0156     0.7816
   3   2.50    2.86        4414       2.6609     0.5528     2.0644     0.7815
   4   2.27    2.50        4510       2.3753     0.4124     1.9953     0.7891
   5   2.11    2.27        4485       2.1824     0.3797     1.9841     0.7951
   6   1.98    2.11        4482       2.0410     0.2870     2.0392     0.7870
   7   1.88    1.98        4528       1.9300     0.2020     2.0294     0.7917
   8   1.80    1.88        4432       1.8399     0.1334     2.0741     0.7910
--------------------averages-over-all-bins----------------------------------
                                                 0.6101     2.0124     0.7898
---------------------------------------------------------------------------
```

(*b*)

Fig. 7. Output from task files: (*a*) header of a coordinate file produced by simulated-annealing refinement in *CNS*. All parameters necessary to reproduce the result are included. (*b*) Analysis of the intensity distribution for a particular diffraction data set.

task file. Modification of the main body of the file is not required, but may be performed by experienced users in order to experiment with new algorithms. The definition section also contains directives that specify HTML features, *e.g.* text comments (indicated by {*... *}), user-modifiable fields (indicated by { = = =>}), and choice boxes (indicated by {+ choice: ... + }). Fig. 6 shows a portion of the 'define' section of a typical *CNS* refinement task file.

2.7.1. *Output from task files.* The task files produce a number of output files (*e.g.* coordinate, reflection, graphing and analysis files). Comprehensive information about input parameters and results of the task are provided in these output files. For example, the PDB REMARK header of coordinate files produced by the simulated-annealing refinement task file is shown in Fig. 7(*a*). 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 (Fig. 7*b*). These data files can be used for graphing, for example by using commonly available spreadsheet programs. An HTML



```
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>
```

(*a*)

```
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>
```

(*b*)

Fig. 8. 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' are complex structure factors corresponding to three sets of heavy atoms that are specified using atom selections (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. The target function and its derivatives with respect to the two structure-factor arrays are explicitly defined.

graphical plotting interface is planned which makes use of these analysis files. In addition, list files are often produced that contain a synopsis of the calculation.

2.8. *HTML interface*

The HTML graphical interface makes use of the HTML form syntax to create a high-level menu-driven environment for *CNS* (Fig. 3*a*). 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. 3*b*). These conversion scripts are written in the PERL language.

A comprehensive collection of task files are available for crystallographic phasing and refinement, and for NMR structure calculation (Fig. 2). 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. 3*b*).

We plan to establish a large number of mirror sites worldwide in order to facilitate easy access to the conversion system. In addition, we will provide instructions describing how to establish the conversion system locally if an HTML server is available.

3. 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 heavy-atom 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-atom 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. 8(*a*). 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 (Brünger, 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

$$\sum_{hkl} |\mathbf{F}_{\rm obs}| - (|\mathbf{F}_{\rm calc1}|^2 + |\mathbf{F}_{\rm calc2}|^2)^{1/2}. \qquad (12)$$

The symbolic definition of this target is shown in Fig. 8(*b*). 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

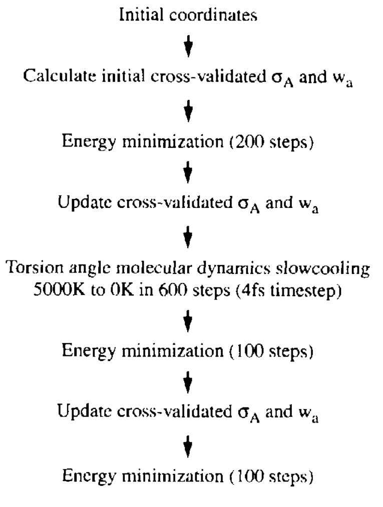

Initial coordinates
Calculate initial cross-validated σ_A and w_a
Energy minimization (200 steps)
Update cross-validated σ_A and w_a
Torsion angle molecular dynamics slowcooling
5000K to 0K in 600 steps (4fs timestep)
Energy minimization (100 steps)
Update cross-validated σ_A and w_a
Energy minimization (100 steps)

Fig. 9. Automated torsion angle dynamics simulated-annealing protocol for maximum-likelihood based refinement. Refinements with the maximum-likelihood target require computation of cross-validated σ_A values. After an initial 200 conjugate-gradient minimization steps, the estimates of σ_A and the weight for the maximum-likelihood target with respect to the chemical restraints (w_a) are updated. Torsion angle molecular dynamics in combination with simulated annealing is then started from a temperature of 5000 K and decreased in 25 K steps to 0 K. A final conjugate-gradient minimization cycle is carried out after an update of the σ_A and w_a values.

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* FORTRAN77 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

$$\begin{aligned}&\texttt{target} = (\texttt{mlf(fobs, sigma, (fcalc} + \texttt{fbulk), d,}\\&\texttt{sigma\_delta))}\\&\texttt{dtarget} = (\texttt{dmlf(fobs, sigma, (fcalc} + \texttt{fbulk), d,}\\&\texttt{sigma\_delta))}\end{aligned} \tag{13}$$

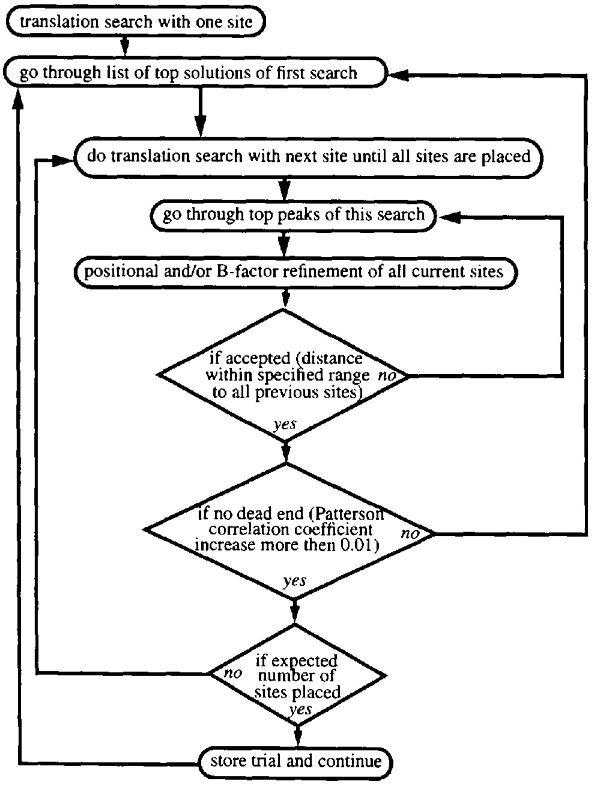
where '`mlf()`' and '`dmlf()`' refer to internal maximum-likelihood functions, '`fobs`' and '`sigma`' are the observed structure-factor amplitudes and corresponding σ values, '`fcalc`' is the (complex) calculated structure-factor array, '`fbulk`' is the structure-factor array for a bulk solvent model, '`d`' and '`sigma_delta`' are the cross-validated D and σ_Δ functions (Read, 1990, 1997; Kleywegt & Brünger, 1996) which are precomputed prior to invoking the MLF target function using the test set of reflections. The availability of internal FORTRAN77 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 (Pannu & Read, 1996)], 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).

4. Selected examples

4.1. *Combined maximum-likelihood and simulated-annealing refinement*

CNS has a comprehensive task file for simulated-annealing refinement of crystal structures using Cartesian (Brünger *et al.*, 1987; Brünger, 1988) or torsion-angle molecular dynamics (Rice & Brünger, 1994). This task file automatically computes a cross-validated σ_A estimate, determines the weighting scheme between the X-ray refinement target function and the geometric energy function (Brünger *et al.*, 1989), refines a flat bulk solvent model (Jiang & Brünger, 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, multi-conformer refinement (Burling & Brünger, 1994), and non-crystallographic symmetry (Weis *et al.*, 1990). 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 (Brünger *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 Brünger *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 & Brünger, 1996; Read, 1997), and the weight between X-ray diffraction target function and


translation search with one site
go through list of top solutions of first search
do translation search with next site until all sites are placed
go through top peaks of this search
positional and/or B-factor refinement of all current sites
if accepted (distance within specified range to all previous sites)
no
yes
if no dead end (Patterson correlation coefficient increase more then 0.01)
no
yes
if expected number of sites placed
no
yes
store trial and continue

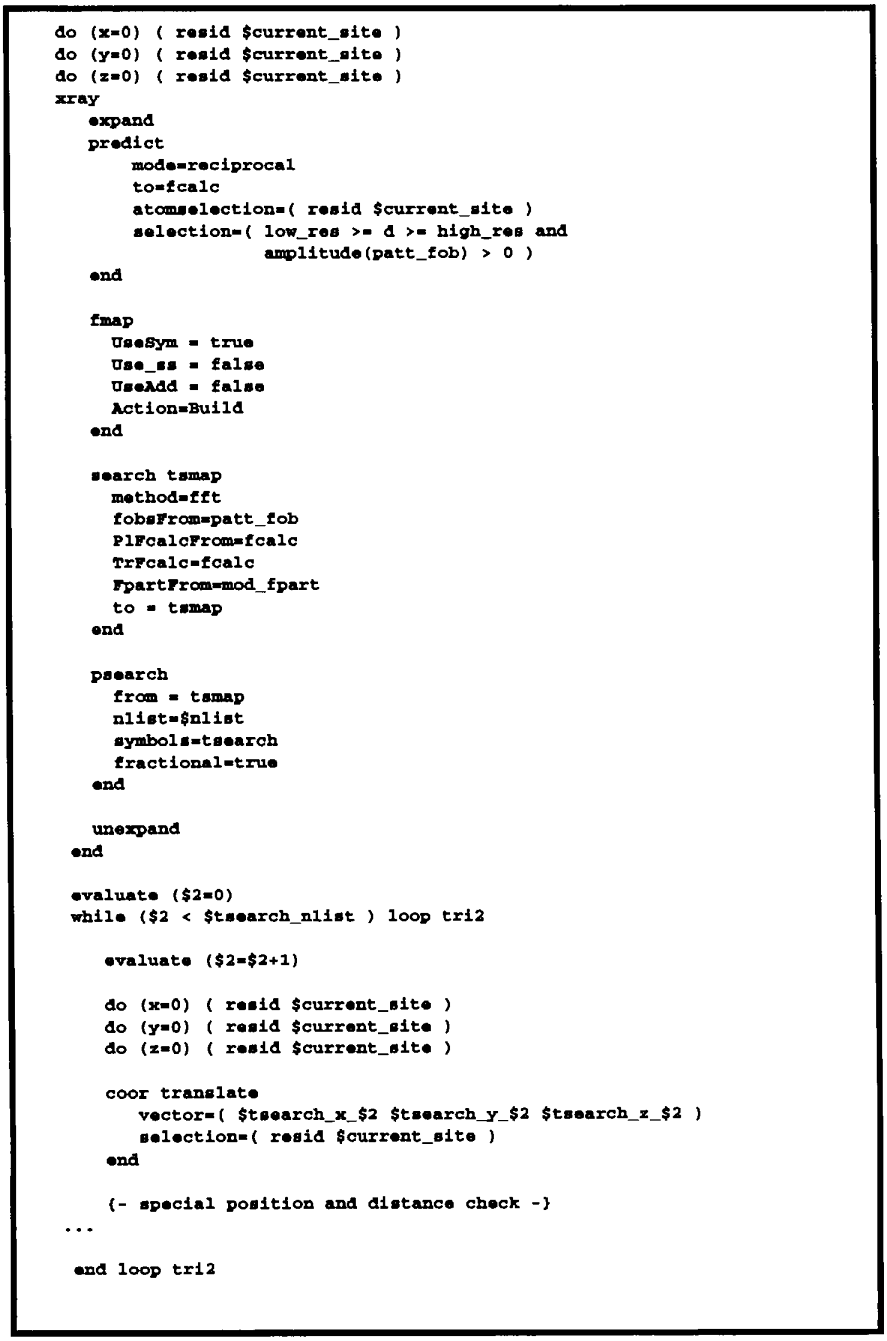
(*a*)

Fig. 10. Heavy-atom search protocol in *CNS*. (*a*) Flow diagram.

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 (Fig. 9).

4.2. *Heavy-atom search*

The following example outlines a new heavy-atom search method that was entirely developed within the



```
do (x=0) ( resid $current_site )
do (y=0) ( resid $current_site )
do (z=0) ( resid $current_site )
xray
   expand
   predict
       mode=reciprocal
       to=fcalc
       atomselection=( resid $current_site )
       selection=( low_res >= d >= high_res and
                   amplitude(patt_fob) > 0 )
   end

   fmap
     UseSym = true
     Use_ss = false
     UseAdd = false
     Action=Build
   end

   search tsmap
     method=fft
     fobsFrom=patt_fob
     P1FcalcFrom=fcalc
     TrFcalc=fcalc
     FpartFrom=mod_fpart
     to = tsmap
   end

   psearch
     from = tsmap
     nlist=$nlist
     symbols=tsearch
     fractional=true
   end

   unexpand
 end

 evaluate ($2=0)
 while ($2 < $tsearch_nlist ) loop tri2

    evaluate ($2=$2+1)

    do (x=0) ( resid $current_site )
    do (y=0) ( resid $current_site )
    do (z=0) ( resid $current_site )

    coor translate
       vector=( $tsearch_x_$2 $tsearch_y_$2 $tsearch_z_$2 )
       selection=( resid $current_site )
    end

    {- special position and distance check -}
...

 end loop tri2
```

(*b*)

Fig. 10. *cont.* (*b*) Portion of the *CNS* task file for heavy-atom searches.

framework of the *CNS* language. The method is based on sequential placement of heavy atoms (Fig. 10*a*).

A portion of the protocol is shown in Fig. 10*b*. The diffraction data are expanded to space group *P*1. Structure factors are computed for the current heavy-atom site and stored in the '`fcalc`' structure-factor array in *P*1. An internal data structure is created by the '`fmap`' statement which describes the asymmetric unit for the subsequent translation search. A fast translation search (Navaza & Vernoslova, 1995) is carried out using the '`fcalc`' structure-factor array, the structure-factor array '`mod_fpart`' with the already placed sites, and a structure-factor array '`patt_fob`' that contains the Fourier transform of the Patterson map. The results of the translation search are stored in a three-dimensional map '`tsmap`'. A list of the top peaks is created and stored in *CNS* language symbols (`$tsearch_x_i`, `$tsearch_y_i`, `$tsearch_z_i` were `i` is the peak number). The diffraction data and all reciprocal-space arrays are reduced to the asymmetric unit of the space group of the crystal. The subsequent loop checks the acceptability of each peak. The acceptance criteria require that the new site be within a specified distance range from any other previously placed site and optionally exclude sites located at special positions.

All trials are refined and sorted using the correlation coefficient between observed and calculated normalized squared structure-factor amplitudes (Fujinaga & Read, 1987; Brünger, 1990). In our experience, the correct solution is characterized by a significant gap between the trials with the top correlation coefficients and the remaining trials. In test calculations, the *CNS* heavy-atom search protocol was successful in finding up to 30 seleno-methionine sites in anomalous difference Patterson maps (RWGK and ATB, unpublished work).

4.3. *NMR structure calculation*

The NMR structure calculation protocols in *CNS* consist of four main sections: data input, annealing protocols, acceptance tests and analysis of all NMR structures. The data input includes NOE-derived distances, NOE intensities, torsion-angle restraints, coupling constants, ^{1}H chemical shifts, ^{13}Cα and ^{13}Cβ secondary shifts, dipolar coupling data, and heteronuclear T_1/T_2 ratios. Many of the these features have been summarized in a recent review (Clore & Gronenborn, 1998).

Distance restraints can be represented as harmonic functions (Clore *et al.*, 1985; Clore, Brünger *et al.*, 1986), quadratic square-well functions (Clore, Nilges *et al.*, 1986), and quadratic asymptotic functions (Nilges *et al.*, 1988*b*). In the case of prochiral centers with unknown stereospecific assignments or in the case of ambiguous NOE assignments, a number of different procedures are available, including center (Clore *et al.*, 1985), $\langle r^{-6}\rangle^{-1/6}$ (Clore, Brünger *et al.*, 1986) averaging, and $\sum\langle r^{-6}\rangle^{-1/6}$ summation (Nilges, 1993). Overlap of NOEs can be properly accounted if they occur because of degeneracy of chemical shifts (Nilges, 1995) or because of symmetry in oligomeric molecules (O'Donoghue *et al.*, 1996; Nilges, 1993).

CNS also has features that permit direct refinement against NOE intensities using either a full-relaxation matrix approach (Nilges *et al.*, 1991) or a quasi-relaxation matrix method which iterates between distances and NOE intensities until convergence has been achieved (GLW and ATB, unpublished work).

Torsion-angle restraints can be represented as either harmonic or quadratic square-well functions (Clore, Nilges *et al.*, 1986). Two forms of coupling-constant restraints are available (Garrett *et al.*, 1994): three-bond couplings which are related to a single torsion angle, and one-bond couplings which are related to two torsion angles (*e.g.* the one-bond Cα—H coupling is related to both φ and ψ backbone torsion angles).

^{1}H chemical-shift restraints include ring current, magnetic susceptibility and electric field effects (Kuszewski, Gronenborn *et al.*, 1995). A multiple ^{1}H chemical-shift function involving sums and differences of the chemical shifts is also available in order to automatically handle chemical shifts involving prochiral protons without the need for making *a priori* stereo-assignments (Kuszewski, Gronenborn *et al.*, 1996*b*).

The dipolar coupling and heteronuclear T_1/T_2 restraints provide long-range structural information in terms of the orientations of particular one-bond vectors to an external axis system (Tjandra, Garrett *et al.*, 1997; Tjandra, Omichinski *et al.*, 1997; Clore, Gronenborn & Tjandra, 1998). In the case of the dipolar couplings, the axis system may be the magnetic susceptibility or molecular alignment tensor. In the case of the T_1/T_2 restraints it is the diffusion tensor. *CNS* permits use of both axially symmetric and the generally fully asymmetric cases. The external axis system is represented by an artificial tetra-atomic molecule consisting of four atoms, representing the *x*, *y* and *z* axes of the tensor (Clore, Gronenborn & Tjandra, 1998). As the orientation of the axis system is not known *a priori* it is allowed to float during the simulated-annealing calculations. The magnitude of the tensor, however, must be specified. This can usually be determined directly from the experimental data, in the absence of any prior structural information, by examining the distribution of dipolar couplings (Clore Gronenborn & Bax, 1998) or T_1/T_2 values (Clore, Gronenborn, Szabo *et al.*, 1998), provided the distribution of one-bond vectors is relatively uniform and isotropic. Alternatively, a grid search can be employed (Clore, Gronenborn & Tjandra, 1998).

Non-bonded interactions may be represented by a Lennard–Jones potential or by simplified repulsive (Nilges, Clore *et al.*, 1988*a*,*b*; Nilges, Gronenborn *et al.*, 1988) or attractive–repulsive functions (Kuszewski *et al.*, 1996*a*).

The starting points for the NMR structure calculation and refinement protocols are randomized extended strands corresponding to each disjoint molecular entity (polypeptide chain or oligonucleotide acid strand) or pre-folded structures. The first section of the protocol consists of reading the various data structures. This is followed by an initialization section for statistical analysis of average properties. A constant high-temperature Cartesian or torsion-angle annealing stage follows (Rice & Brünger, 1994; Stein *et al.*, 1997). This is followed by a slow-cooling stage with either torsion angle or Cartesian dynamics. Finally, an additional Cartesian dynamics cooling stage and a minimization stage follow. A number of trials are performed by starting the simulated-annealing calculation with different randomly selected initial atomic velocities.

Analysis of deviations and violations for the various experimental and chemical restraints is carried out and written to the header sections of the coordinate file corresponding to the particular trial. The acceptability of the trial is tested and analysis of average properties carried out. The whole process begins again using different initial velocities (or coordinates) which in general produces a different result.

5. Parallelization

Parallelization of the *CNS* FORTRAN77 program has been accomplished using a single program multiple data (SPMD) model. The parallel virtual machine (PVM) (Geist *et al.*, 1994) parallel programming environment has been used to provide portability across computing platforms. All parallel communications routines are centralized in one C code module, which also parses the UNIX command line arguments and starts the main *CNS* code. This modularity facilitates easy conversion to different parallel environments such as the message passing interface (MPI) (Gropp *et al.*, 1994). An MPI port is already in place for shared-memory multiprocessor Silicon Graphics platforms and the massively parallel Cray T3E.

The parallelization of the program is carried out at two distinct levels. This is achieved *via* the notion of processors and groups. Groups are entities that function independently at the level of *CNS* task files, whereas processors act cooperatively at the level of the *CNS* source code. Each group can contain several processors. The user specifies the number of groups and the number of processors within each group at program execution time *via* the UNIX command line arguments. The group-based coarse-grained parallelism is available to the user at the *CNS* language level through the definition of symbols containing information about the total number of groups and the group identity of each process. Appropriately written task files use this information to parallelize *CNS* language level loop execution. The processor-based fine-grained parallelism is performed within the *CNS* source code, using the underlying message passing tools. To date, the chemical energy terms (or 'restraints') dealing with covalent and non-bonded interactions, and the Cartesian molecular dynamics integrator have been parallelized. Parallelization of crystallographic tasks has started with the rotation searches used in molecular replacement. Future work will focus on the efficient parallelization of both crystallographic and NMR target functions.

6. Distribution

CNS will be made available in its entirety, including source code. It is hoped that this approach will foster fruitful interactions among research groups and contributions to the future development of *CNS*. Plans are currently being made to establish a suitable user support facility for the structural and molecular biology communities.

7. 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. 3*b*). 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 or solution NMR algorithm are defined through modules and task files written in the *CNS* structure determination language. This allows for the development of new algorithms and for existing algorithms to be precisely defined and easily modified without the need for FORTRAN77 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 provides 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.

The authors wish to thank members of their respective groups, the Yale Center for Structural Biology, and many colleagues of the worldwide structural biology community for valuable suggestions and comments during the development of *CNS*. Support by the Howard Hughes Medical Institute and the National Science Foundation to ATB (DBI-9514819 and ASC 93-181159), the AIDS targeted antiviral program of the Office of the Director of the National Institute of Health to GMC, the Natural Sciences and Engineering Research Council of Canada to NSP, the Howard Hughes Medical Institute and the Medical Research Council of Canada to RJR (MT11000), the Netherlands Foundation for Chemical Research (SON) with financial aid from the Netherlands Organization for Scientific Research (NWO) to PG, and the Howard Hughes Medical Institute to LMR is gratefully acknowledged.

References

Adams, P. D., Pannu, N. S., Read, R. J. & Brünger, A. T. (1997). *Proc. Natl Acad. Sci. USA*, **94**, 5018–5023.

Blow, D. W. & Crick, F. H. C. (1959). *Acta Cryst.* **12**, 794–802.

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

Bricogne, G. (1984). *Acta Cryst.* A**40**, 410–445.

Brünger, A. T. (1988). *J. Mol. Biol.* **203**, 803–816.

Brünger, A. T. (1990). *Acta Cryst.* A**46**, 46–57 (1990).

Brünger, A. T. (1992). *Nature (London)*, **355**, 472–474.

Brünger, A. T., Adams, P. D. & Rice, L. M. (1997). *Structure*, **5**, 325–336.

Brünger, A. T., Clore, G. M., Gronenborn, A. M. & Karplus, M. (1986). *Proc. Natl Acad. Sci. USA*, **83**, 3801–3805.

Brünger, A. T., Clore, G. M., Gronenborn, A. M., Saffrich, R. & Nilges, M. (1993). *Science*, **261**, 328–331.

Brünger, A. T., Karplus, M. & Petsko, G. A. (1989). *Acta Cryst.* A**45**, 50–61.

Brünger, A. T, Krukowski, A. & Erickson, J. (1990). *Acta Cryst.* A**46**, 585–593.

Brünger, A. T., Kuriyan, J. & Karplus, M. (1987). *Science*, **235**, 458–460.

Burling, F. T. & Brünger, A. T. (1994). *Isr. J. Chem.* **34** 165–175.

Burling, F. T., Weis, W. I., Flaherty, K. M. & Brünger, A. T. (1996). *Science*, **271**, 72–77.

Clore, G. M., Brünger, A. T., Karplus, M. & Gronenborn, A. M. (1986). *J. Mol. Biol.* **191**, 523–551.

Clore, G. M. & Gronenborn, A. M. (1997). *Nature Struct. Biol.* **4**, 841–844.

Clore, G. M. & Gronenborn, A. M. (1998). *Proc. Natl Acad. Sci. USA*, **95**, 5891–5898.

Clore, G. M., Gronenborn, A. M. & Bax, A. (1998). *J. Magn. Reson.* **133**, 216–221.

Clore, G. M., Gronenborn, A. M., Brünger, A. T. & Karplus, M. (1985). *J. Mol. Biol.* **186**, 435–455.

Clore, G. M., Gronenborn, A. M., Szabo, A. & Tjandra, N. (1998). *J. Am. Chem. Soc.* **120**, 4889–4890.

Clore, G. M., Gronenborn, A. M. & Tjandra, N. (1998). *J. Magn. Reson.* **131**, 159–162.

Clore, G. M., Nilges, M., Sukumaran, D. K., Brünger, A. T., Karplus, M. & Gronenborn, A. M. (1986). *EMBO J.* **5**, 2729–2735.

Fujinaga, M. & Read, R. J. (1987). *J. Appl. Cryst.* **20**, 517–521.

Garrett, D. S., Kuszewski, J., Hancock, T. J., Lodi, P. J., Vuister, G. W., Gronenborn, A. M. & Clore, G. M. (1994). *J. Magn. Reson. Ser. B*, **104**(1), 99–103.

Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R. & Sunderam, V. (1994). *PVM: Parallel Virtual Machine A Users' Guide & Tutorial for Networked Parallel Computing.* Cambridge: MIT Press.

Graham, I. S. (1995). *The HTML Sourcebook.* New York: John Wiley.

Gropp, W., Lusk, E. & Skjellum, A. (1994). *Using MPI: Portable Parallel Programming with the Message-Passing Interface.* Cambridge: MIT Press.

Hendrickson, W. A. (1979). *Acta Cryst.* A**35**, 245–247.

Hendrickson, W. A. (1991). *Science*, **254**, 51–58.

Hendrickson, W. A. & Lattman, E. E. (1970). *Acta Cryst.* B**26**, 136–143.

Jiang, J.-S. & Brünger, A. T. (1994). *J. Mol. Biol.* **243**, 100–115.

Kleywegt, G. J. & Brünger, A. T. (1996). *Structure*, **4**, 897–904.

Kuszewski, J., Gronenborn, A. M. & Clore, G. M. (1995). *J. Magn. Reson. Ser. B*, **107**(3), 293–297.

Kuszewski, J., Gronenborn, A. M. & Clore, G. M. (1996*a*). *J. Magn. Reson. Ser. B*, **112**(1), 79–81.

Kuszewski, J., Gronenborn, A. M. & Clore, G. M. (1996*b*). *Protein Sci.* **5**, 1067–1080.

Kuszewski, J., Gronenborn, A. M. & Clore, G. M. (1997). *J. Magn. Res.* **125**, 171–177.

Kuszewski, J., Qin, J., Gronenborn, A. M. & Clore, G. M. (1995). *J. Magn. Reson. Ser. B*, **106**(1), 92–96.

Navaza, J. & Vernoslova, E. (1995). *Acta Cryst.* A**51**, 445–449.

Nilges, M. (1993). *Proteins*, **17**, 297–309.

Nilges, M. (1995). *J. Mol. Biol.* **245**, 645–660.

Nilges, M., Clore, G. M. & Gronenborn, A. M. (1988*a*). *FEBS Lett.* **229**, 317–324.

Nilges, M. Clore G. M, & Gronenborn, A. M. (1988*b*). *FEBS Lett.* **239**, 129–136.

Nilges, M., Gronenborn, G. M., Brünger, A. T. & Clore, G. M. (1988). *Protein Eng.* **2**, 27–38.

Nilges, M., Habazettl, J., Brünger, A. T. & Holak, T. A. (1991). *J. Mol. Biol.* **219**, 499–510.

Nilges, M., Macias, M. J., O'Donoghue, S. I. & Oschkinat, H. (1997). *J. Mol. Biol.* **269**, 408–422.

O'Donoghue, S. I., King, G. F. & Nilges, M. (1996). *J. Biomol. NMR*, **8**, 193–206.

Otwinowski, Z. (1991). *Proceedings of the CCP4 study weekend 25–26 January 1991*, edited by W. Wolf, P. R. Evans & A. G. W. Leslie, pp. 80–85. Warrington: Daresbury Laboratory.

Pannu, N. S., Murshudov, G. N., Dodson, E. J. & Read, R. J. (1998). *Acta Cryst.* D**54**. In the press.

Pannu, N. S. & Read, R. J. (1996). *Acta Cryst.* A**52**, 659–668.

Phillips, J. C. & Hodgson, K. O. (1980). *Acta Cryst.* A**36**, 856–864 (1980).

Read, R. J. (1990). *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, Am. Crystallogr. Assoc. Ann. Meet. 1994, Atlanta, GA, USA.

Read, R. J. (1997). *Methods Enzymol.* **278**, 110–128.

Rice, L. M. & Brünger, A. T. (1994). *Proteins Struct. Funct. Genet.* **19**, 277–290.

Stein, E. G., Rice, L. M. & Brünger, A. T. (1997). *J. Magn. Reson.* **124**, 154–164.

Stout, G. H. & Jensen, L. H. (1989). *X-ray Structure Determination*, p. 33. New York: Wiley Interscience.

Tjandra, N., Garrett, D. S., Gronenborn, A. M., Bax, A. & Clore, G. M. (1997). *Nature Struct. Biol.* **4**, 443–449.

Tjandra, N., Omichinski, J. G., Gronenborn, A. M., Clore, G. M. & Bax, A, (1997). *Nature Struct. Biol.* **4**, 732–738.

Wagner, G. (1997). *Nature Struct. Biol.* **4**, 841–844.

Weis, W. I, Brünger, A. T., Skehel, J. J. & Wiley, D. C. (1990). *J. Mol. Biol.* **212**, 737–761.