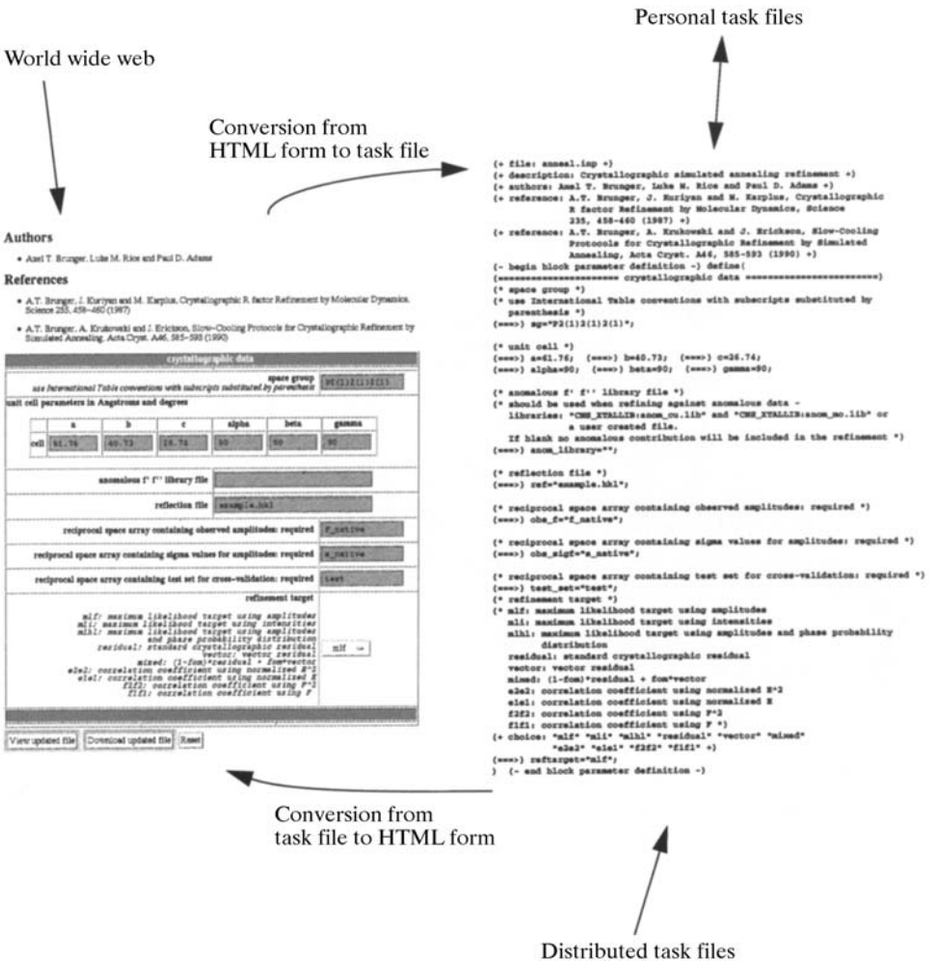

Fig. 25.2.3.10. Use of the CNS HTML form page interface, emphasizing the correspondence between input fields in the form page and parameters in the task file.

validated σ_A estimates, 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 for 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 & Brünger, 1994), noncrystallographic symmetry constraints and restraints (Weis *et al.*, 1990), and 'flat' solvent models (Jiang & Brünger, 1994). Available target functions include the maximum-likelihood functions MLF, MLI and MLHL (Pannu & Read, 1996*a*; 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 & Brunger, 1996; Read, 1997) and the weight between the 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 that 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.

25.2.3.10. *Conclusions*

CNS is a general system for structure determination by X-ray crystallography and solution NMR. It covers the whole spectrum of methods used 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. 25.2.3.10). 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 more than a hundred 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 enables one to define symbolically any target function for a broad range of applications, from heavy-atom phasing or molecular-replacement searches to atomic resolution refinement.

25.2.4. The *TNT* refinement package

(D. E. Tronrud and L. F. Ten Eyck)

25.2.4.1. *Scope and function of the package*

TNT (Tronrud *et al.*, 1987) is a computer program package that optimizes the parameters of a molecular model given a set of observations and indicates the location of errors that it cannot correct. Its authors presume the principal set of observations to be the structure factors observed in a single-crystal diffraction experiment. To complement such a data set, which for most macromolecules has limitations, stereochemical restraints such as standard bond lengths and angles are also used as observations.

A molecule is parameterized as a set of atoms, each with a position in space, an isotropic B factor and an occupancy. The

complete model also includes an overall scale factor, which converts the arbitrary units of the measured structure factors to e Å^{-3}, and a two-parameter model of the electron density of the bulk solvent.

Because a *TNT* model of a macromolecule does not allow anisotropic *B* factors, *TNT* cannot be used to finish the refinement of any structure that diffracts to high enough resolution to justify the use of these parameters. If one has a crystal that diffracts to 1.4 Å or better, the final model should probably include these parameters and *TNT* cannot be used. One may still use *TNT* in the early stages of such a refinement because one usually begins with only isotropic *B*'s.

At the other extreme of resolution, *TNT* begins to break down with data sets limited to only about 3.5 Å data. This breakdown occurs for two reasons. First, at 3.5 Å resolution, the maps can no longer resolve β-sheet strands or α-helices. The refinement of a model against data of such low resolution requires strong restraints on dihedral angles and hydrogen bonds – tasks for which *TNT* is not well suited. Second, the errors in an initial model constructed with only 3.5 Å data are usually of such a magnitude and quality that the function minimizer in *TNT* cannot correct them.

25.2.4.2. *Historical context*

The design of *TNT* began in the late 1970s, and the first publishable models were generated by *TNT* in 1981 (Holmes & Matthews, 1981). Its design was greatly influenced by observations of the strength and weaknesses of programs then available.

The first refinement of a protein model was performed by Jensen and co-workers at the University of Washington (Watenpaugh *et al.*, 1973). This structure refinement was atypical because of the availability of high-resolution data. The techniques of pre-least-squares small-molecule refinement were simply applied to this much larger model. Since many of the calculations were performed manually, no comprehensive software package was created for distribution.

It was quickly realized that for macromolecular refinement to become common, the calculations had to be fully automated and ideal stereochemistry had to be enforced. In the late 1970s, four programs became available, all of which automated the refinement calculations, but each implemented the enforcement of stereochemistry in different ways. They were *PROLSQ* (Hendrickson & Konnert, 1980), *EREF* (Jack & Levitt, 1978), *CORELS* (Sussman *et al.*, 1977) and *FFTSF* (Agarwal, 1978). *PROLSQ* was, ultimately, the most popular.

At one end of the spectrum lay *FFTSF*. This program optimized its models to the diffraction data while completely ignoring ideal geometry. Following a number of iterations of optimizing the fit of the model to the structure factors, the geometry was idealized by running a separate program. At the other extreme was *CORELS*. It optimized its models to the diffraction data while allowing no deviations from ideal stereochemistry. The model was allowed to change only through the rotation of single bonds and the movement of rigid groups. Both approaches were frustrating to a certain extent. With *FFTSF* it was a struggle to find a model that agreed with all observations at once. With *CORELS* it was difficult to get the model to fit the density, because small and, apparently, insignificant deviations from ideality often added up after many residues to large and significant displacements, and these were forbidden. Neither approach to stereochemistry seemed very convenient, although *CORELS* was used for early-stage refinement for many years because of its exceptional radius of convergence.

Both *PROLSQ* and *EREF* enforced ideal stereochemistry and agreement with the diffraction data simultaneously. This strategy proved very convenient and generated models that satisfied their users. The two programs differed significantly in the form in which they required the ideal values be entered. *PROLSQ* required that the ideal values for both bond lengths and bond angles be entered as distances, *e.g.* an angle was defined by the distance between the two extreme atoms. *EREF* required that the standard value for an angle simply be entered as the number of degrees. Since *EREF* stored its library of standard values in the same terms as those with which people were familiar, it was much easier to enter the values.

These two programs differed in another way as well. *PROLSQ* stored ideal values for the stereochemistry of each type of residue (*e.g.* alanine, glycine *etc.*), while *EREF* parameterized the library in terms of atom types. For example, the angle formed by three atoms, the first a keto oxygen, the second a carbonyl carbon and the third an amide nitrogen, would have a particular ideal value regardless of where these three atoms occurred. In this matter, *PROLSQ* was more similar to the thought patterns of crystallographers.

25.2.4.3. *Design principles*

TNT was designed with three fundamental principles in mind. Each principle has a number of consequences that shaped the ultimate form of the package.

25.2.4.3.1. *Refinement should be simple to run*

The user should not be burdened with the choice of input parameters that they may not be qualified to choose. They also should not be forced to construct an input file that is obscure and difficult to understand. It is hard now to remember what most computer programs were like in the 1970s. Usually, the input to the program was a block of numbers and flags where the meaning of each item was defined by its line and column numbers. This block not only contained information the programmer could never anticipate, like the cell constants, but defined how the computer's memory should be allocated and obscure parameters that could only be estimated after careful reading of research papers.

TNT was one of the first programs in crystallography to have its input introduced with keywords and to allow input statements to come in any order. As an example of the difference, consider the resolution limits. Usually, a crystallographic program would have a line in its input similar to

```
99.0, 1.9,
```

One had to recognize this line amongst many as the line containing the resolution limits. (In many programs, a value of 99 was used to indicate that no lower-resolution limit was to be applied.) In *TNT* the same data would be entered as

```
RESOLUTION 1.9
```

The keyword identifies the data as the resolution limit(s). If the statement contained two numbers, they were considered the upper and lower limits of the diffraction data.

The preceding example also shows how default values can be implemented by a program much more safely with keyword-based input. In the previous scheme, if a value was ever to be changed by the user, its place had to be allocated in the input block. This often left numbers floating in the block which were almost never changed, and because they were so infrequently referred to, they were usually unrecognized by the user. It was quite possible for one of these numbers to be accidentally changed and the error unnoticed for quite some time. When the data are introduced with keywords, a data item is not mentioned if the default value is suitable.

25.2.4.3.2. *Refinement should run quickly and use as little memory as possible*

The most time-consuming calculations in refinement are the calculation of structure factors from atomic coordinates and the