Chapter 18.7. The TNT refinement package

D. E. TRONRUD AND L. F. TEN EYCK

18.7.1. 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 $e\AA^{-1}$, 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.

18.7.2. 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 to 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.

18.7.3. 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.

18.7.3.1. 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

\[
\text{RESOLUTION 1.9}
\]

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.

### 18.7.3.2. 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 calculation of derivatives of the part of the residual dependent upon the diffraction data with respect to the atomic parameters. The quickest means of performing these calculations requires the use of space-group-optimized fast Fourier transforms (FFTs). The initial implementation of *TNT* used FFTs to calculate structure factors, but the much slower direct summation method to calculate the derivatives. Within a few years, Agarwal’s method (Agarwal, 1978; Agarwal et al., 1981) was incorporated into *TNT* and from then on all crystallographic calculations were performed with FFTs.

The FFT programs of Ten Eyck (1973, 1977) made very efficient use of computer memory. Another means of saving memory was to recognize that the code for calculating stereochemical restraints did not need to be in the memory when the crystallographic calculations were being performed and vice versa. There were two ways to save memory using this information. One could create a series of ‘overlays’ or one could break the calculation into a series of separate programs. The means for defining an overlay structure were never standardized and could not be ported from one type of computer to another and were, therefore, never attempted in *TNT*. For this reason, and a number of others mentioned here, *TNT* is not a single program but a collection of programs, each with a well defined and specialized purpose.

### 18.7.3.3. The source code should not require customization for each project

The need to state this goal seems remarkable in these modern times, but the truth is that most computer programs in the 1970s required specific customizations before they could be used. The simplest modifications were the definitions of the maximum number of atoms, residues, atom types etc. accepted by the program. These modifications are still required in Fortran77 programs because that language does not allow the dynamic allocation of memory. However, in most programs today the limits are set high enough that the standard configuration does not present a problem.

The most difficult modification required for programs like *PROLSO* was to adapt the calculations to the space group in hand. Their authors usually included code for the space groups they were particularly interested in, leaving all others to be implemented by the user. Writing code for a new space group was often a daunting task for someone who was not an expert programmer and had no tools for testing the modifications.

It is too burdensome to require the user to understand sufficiently the internal workings of a complex calculation that they can code and debug central subroutines of a refinement program. In its initial implementation, *TNT* avoided this problem, to an extent, by performing the space-group-specific calculations in separate programs. At least the user did not need to modify an existing program. All that was required was the construction of a program that read the proper format file, performed the calculation and wrote its answer in the proper format. The user was required to supply both a program that could calculate structure factors from the model and another program that could calculate the derivative of the diffraction component of the residual function with respect to the atomic parameters of the model.

While a structure-factor program could usually be located, either by finding an existing program or by expanding the model to a lower-symmetry space group for which a program did exist, the requirement of creating a derivative program proved too great a burden. The derivation of the space-group-specific calculation, its implementation and debugging proved too difficult for almost everyone, and this design was quickly abandoned. Instead, an implementation of Agarwal’s (1978) algorithm was created. In this method, the derivatives are calculated with a series of convolutions with an $F_o - F_c$ map. The calculation of the map is the only space-group-specific part of the calculation, and this was done with a separate program for calculating Fourier syntheses. Such programs were as easy to come by as structure-factor calculation programs and could be replaced by a lower-symmetry program if required.

While it is easier to find or write a program that only calculates a Fourier transform and much easier to debug one than to debug a modification to a larger and more complex program, it is still difficult. The lack of availability of programs for the space group of a crystal often prevented the use of *TNT*. Over time, programs for more space groups were written and distributed with *TNT*. Eventually, a method was developed by one of *TNT*’s authors in which FFTs could be calculated using a single program as efficiently as the original space-group-specific programs. Once this program existed, there was no longer the need for isolated structure-factor and Fourier synthesis programs. These calcula-