tions have disappeared into the heart of TNT, and TNT consists of many fewer programs today than in the past.

18.7.4. Current structure of the package

TNT presents different faces to different users. Some users simply want to run refinement; they see the shell interface. Others want to use the TNT programs in untraditional ways; they see the program interface. A few users want to change the basic calculations of TNT; they see the library interface.

The shell interface is the view of TNT that most people see. It is the most recent structural addition, having been added in release 5E in 1995. At this level, the restraints, weights and parameters of the model are described in the ‘TNT control file’ and the user performs particular calculations by giving commands at the shell prompt. For example, refinement is performed with the ‘tnt’ command and maps are calculated for examination with some graphics program with the ‘make_maps’ command. TNT is supplied with about two dozen shell commands. These commands allow the running of refinement, the conversion of the model to and from TNT’s internal format, and the examination of the model to locate potential problem spots. The TNT Users’ Guide describes the use of TNT at this level.

The program interface consists of the individual TNT programs along with their individual capabilities. TNT consists of the program Shift, which handles all the minimization calculations, a program for each module (restraints that fall into a category), a program for each module (restraints that fall into a category), a program for each module (restraints that fall into a category), and a program for each module (restraints that fall into a category). In addition, TNT presents different faces to different users. Some users want to change the basic calculations of TNT; they see the library interface. A few users want to use the TNT programs in untraditional ways; they see the program interface. A few users want to change the basic calculations of TNT; they see the library interface.

Shift, which handles all the minimization calculations, a program for each module (restraints that fall into a category), a program for each module (restraints that fall into a category), a program for each module (restraints that fall into a category). In addition, TNT presents different faces to different users. Some users want to change the basic calculations of TNT; they see the library interface. A few users want to use the TNT programs in untraditional ways; they see the program interface. A few users want to change the basic calculations of TNT; they see the library interface.

The shell interface is the view of TNT that most people see. It is the most recent structural addition, having been added in release 5E in 1995. At this level, the restraints, weights and parameters of the model are described in the ‘TNT control file’ and the user performs particular calculations by giving commands at the shell prompt. For example, refinement is performed with the ‘tnt’ command and maps are calculated for examination with some graphics program with the ‘make_maps’ command. TNT is supplied with about two dozen shell commands. These commands allow the running of refinement, the conversion of the model to and from TNT’s internal format, and the examination of the model to locate potential problem spots. The TNT Users’ Guide describes the use of TNT at this level.

The program interface consists of the individual TNT programs along with their individual capabilities. TNT consists of the program Shift, which handles all the minimization calculations, a program for each module (restraints that fall into a category), a program for each module (restraints that fall into a category), a program for each module (restraints that fall into a category). In addition, TNT presents different faces to different users. Some users want to change the basic calculations of TNT; they see the library interface. A few users want to use the TNT programs in untraditional ways; they see the program interface. A few users want to change the basic calculations of TNT; they see the library interface.

18.7.5. Innovations first introduced in TNT

TNT was not only designed to be an easy-to-use tool for the refinement of macromolecular models, but also as a tool for testing new ideas in refinement. Since its source code is designed to allow easy reordering of tasks and simple modifications, a number of innovations in refinement made their first appearance in TNT. These features include the following.

18.7.5.1. Identifying and restraining symmetry-related contacts (1982)

Without a search for symmetry-related bad contacts, it was quite common to build atoms into the same density from two different sides of the molecule. A number of models in the Protein Data Bank contain these types of errors because neither the refinement nor the graphics programs available at that time would indicate this type of error.

18.7.5.2. The ability of a single package to perform both individual atom and rigid-body refinement (1982)

Prior to TNT, one often started a refinement with rigid-body refinement using CORELS and then switched to another program. TNT was the first refinement package to allow both styles of refinement. One was not required to learn about two different packages when running TNT.

18.7.5.3. Space-group optimized FFTs for all space groups (1989)

This innovation allowed TNT to run efficiently in all space groups available to macromolecular crystals.

18.7.5.4. Modelling bulk solvent scattering via local scaling (~1989)

With a simple and quick model of the scattering of the bulk solvent in the crystal (Tronrud, 1997), the low-resolution data could be used in refinement for the first time. The inclusion of these data in the calculation of maps greatly improved their appearance.

18.7.5.5. Preconditioned conjugate-gradient minimization (1990)

This method of minimization (Axelsson & Barker, 1984; Tronrud, 1992) allows the direct inclusion of the diagonal elements of the second-derivative matrix and the indirect inclusion of its off-diagonal elements. An additional benefit is that it allows both positional parameters and B factors to be optimized in each cycle. Previously, one was required to hold one class of parameter fixed while the other was optimized. It is much more efficient and simpler for the user to optimize all parameters at once. This method, because it incorporates the diagonal elements directly, produces sets of B factors that agree with the diffraction data better than those from the simple conjugate-gradient method.

18.7.5.6. Restraining stereochemistry of chemical links to symmetry-related molecules (~1992)

It is not uncommon for crystallization enhancers to be found on a special position in the crystal. In addition, cross-linking the molecules in a crystal is often done for various reasons. In both cases, the model contains chemical bonds to a molecule or atoms in another asymmetric unit of the crystal. In order for the stereochemistry of these links to be properly restrained, it must be possible to describe such a link to the refinement program.

18.7.5.7. Knowledge-based B-factor restraints (~1994)

When the resolution of the diffraction data set is less than about 2 Å, the individual B factors of a refined model are observed to vary wildly from atom to atom, even when the atoms are bonded to one another. This pattern is not reasonable if one interprets the B factor as a measure of the vibrational motion of the atom. Traditionally, one applies an additional restraint on the B factors of the model, where the ideal value for the difference in B factor for two bonded atoms is zero.