commonly placed in the model in plausible positions according to molecular geometry, but this can be misleading to people using the coordinate set. If the atoms are included in the model, the atomic displacement parameters generally become very large, and this may be an acceptable flag for dynamic disorder. The hazard with this procedure is that including these atoms in the model provides additional parameters to conceal any error signal in the data that might relate to problems elsewhere in the model.

At high resolution, it is sometimes possible to model the correlated motion of atoms in rigid groups by a single tensor that describes translation, libration and screw. This is rarely done for macromolecules at present, but may be an extremely accurate way to model the behaviour of the molecules. The recent development of efficient anisotropic refinement methods for macromolecules by Murshudov *et al.* (1999) will undoubtedly produce a great deal more information about the modelling of dynamic disorder and anisotropy in macromolecular structures.

Macromolecular crystals contain between 30 and 70% solvent, mostly amorphous. The diffraction is not accurately modelled unless this solvent is included (Tronrud, 1997). The bulk solvent is generally modelled as a continuum of electron density with a high atomic displacement parameter. The high displacement parameter blurs the edges, so that the contribution of the bulk solvent to the scattering is primarily at low resolution. Nevertheless, it is important to include this in the model for two reasons. First, unless the bulk solvent is modelled, the low-resolution structure factors cannot be used in the refinement. This has the unfortunate effect of rendering the refinement of *all* of the atomic displacement parameters ill-determined. Second, omission or inaccurate phasing of the low-resolution reflections tends to produce long-wavelength variations in the electron-density maps, rendering them more difficult to interpret. In some regions, the maps can become overconnected, and in others they can become fragmented.

18.1.8. Optimization methods

Optimization methods for small molecules are straightforward, but macromolecules present special problems due to their sheer size. The large number of parameters vastly increases the volume of the parameter space that must be searched for feasible solutions and also increases the storage requirements for the optimization process. The combination of a large number of parameters and a large number of observations means that the computations at each cycle of the optimization process are expensive.

Optimization methods can be roughly classified according to the order of derivative information used in the algorithm. Methods that use no derivatives find an optimum through a search strategy; examples are Monte Carlo methods and some forms of simulated annealing. First-order methods compute gradients, and hence can always move in a direction that should reduce the objective function. Second-order methods compute curvature, which allows them to predict not only which direction will reduce the objective function, but how that direction will change as the optimization proceeds. The zero-order methods are generally very slow in high-dimensional spaces because the volume that must be searched becomes huge. First-order methods can be fast and compact, but cannot determine whether or not the solution is a true minimum. Second-order methods can detect null subspaces and singularities in the solution, but the computational cost grows as the cube of the number of parameters (or worse), and the storage requirements grow as the square of the number of parameters – undesirable properties where the number of parameters is of the order of 10^4.

Historically, the most successful optimization methods for macromolecular structures have been first-order methods. This is beginning to change as multi-gigabyte memories are becoming more common on computers and processor speeds are in the gigahertz range. At this time, there are no widely used refinement programs that run effectively on multiprocessor systems, although there are no theoretical barriers to writing such a program.

18.1.8.1. *Solving the refinement equations*

Methods for solving the refinement equations are described in *IT* C Chapters 8.1 to 8.5 and in many texts. Prince (1994) provides an excellent starting point. There are two commonly used approaches to finding the set of parameters that minimizes equation (18.1.4.1). The first is to treat each observation separately and rewrite each term of (18.1.4.1) as

$$w_i[y_i - f_i(\mathbf{x})] = w_i \sum_{j=1}^{N} \left(\frac{\partial f_i(\mathbf{x})}{\partial x_j}\right)(x_j^0 - x_j), \qquad (18.1.8.1)$$

where the summation is over the N parameters of the model. This is simply the first-order expansion of $f_i(\mathbf{x})$ and expresses the hypothesis that the calculated values should match the observed values. The system of simultaneous *observational equations* can be solved for the parameter shifts provided that there are at least as many observations as there are parameters to be determined. When the number of observational equations exceeds the number of parameters, the least-squares solution is that which minimizes (18.1.4.1). This is the method generally used for refining small-molecule crystal structures, and increasingly for macromolecular structures at atomic resolution.

18.1.8.2. *Normal equations*

In matrix form, the observational equations are written as

$$\mathbf{A}\Delta = \mathbf{r},$$

where $\mathbf{A}$ is the M by N matrix of derivatives, Δ is the parameter shifts and $\mathbf{r}$ is the vector of residuals given on the left-hand sides of equation (18.1.8.1). The *normal equations* are formed by multiplying both sides of the equation by $\mathbf{A}^T$. This produces an N by N square system, the solution to which is the desired least-squares solution for the parameter shifts.

$$\mathbf{A}^T\mathbf{A}\Delta = \mathbf{A}^T\mathbf{r} \text{ or } \mathbf{M}\Delta = \mathbf{b},$$

$$m_{ij} = \sum_{k=1}^{M} w_k \left(\frac{\partial f_k(\mathbf{x})}{\partial x_i}\right)\left(\frac{\partial f_k(\mathbf{x})}{\partial x_j}\right),$$

$$b_i = \sum_{k=1}^{M} w_k[y_k - f_k(\mathbf{x})]\left(\frac{\partial f_k(\mathbf{x})}{\partial x_i}\right).$$

Similar equations are obtained by expanding (18.1.4.1) as a second-order Taylor series about the minimum $\mathbf{x}_0$ and differentiating.

$$\Phi(\mathbf{x}-\mathbf{x}_0) \approx \Phi(\mathbf{x}_0) + \left\langle \left(\frac{\partial\Phi}{\partial x_i}\right)_{\mathbf{x}_0} \middle| (\mathbf{x}-\mathbf{x}_0) \right\rangle + \frac{1}{2}\left\langle (\mathbf{x}-\mathbf{x}_0) \middle| \left(\frac{\partial^2\Phi}{\partial x_i \partial x_j}\right)_{\mathbf{x}_0} \middle| (\mathbf{x}-\mathbf{x}_0) \right\rangle,$$

$$\left|\left(\frac{\partial\Phi}{\partial\mathbf{x}}\right)\right\rangle \approx \left|\left(\frac{\partial^2\Phi}{\partial x_i \partial x_j}\right)_{\mathbf{x}_0}\right| (\mathbf{x}-\mathbf{x}_0)\right\rangle.$$

The second-order approximation is equivalent to assuming that the matrix of second derivatives does not change and hence can be computed at $\mathbf{x}$ instead of at $\mathbf{x}_0$.

references