Acta Crystallographica Section D
Biological Crystallography
ISSN 0907-4449

Coot: model-building tools for molecular graphics

Paul Emsley* and Kevin Cowtan

York Structural Biology Laboratory, University of York, Heslington, York YO10 5YW, England

Correspondence e-mail: emsley@ysbl.york.ac.uk

Received 26 February 2004
Accepted 4 August 2004

CCP4mg is a project that aims to provide a general-purpose tool for structural biologists, providing tools for X-ray structure solution, structure comparison and analysis, and publication-quality graphics. The map-fitting tools are available as a stand-alone package, distributed as '*Coot*'.

1. Introduction

Molecular graphics still plays an important role in the determination of protein structures using X-ray crystallographic data, despite on-going efforts to automate model building. Functions such as side-chain placement, loop, ligand and fragment fitting, structure comparison, analysis and validation are routinely performed using molecular graphics. Lower resolution ($d_{\rm min}$ worse than 2.5 Å) data in particular need interactive fitting.

The introduction of *FRODO* (Jones, 1978) and then *O* (Jones *et al.*, 1991) to the field of protein crystallography was in each case revolutionary, each in their time breaking new ground in demonstrating what was possible with the current hardware. These tools allowed protein crystallographers to enjoy what is widely held to be the most thrilling part of their work: giving birth (as it were) to a new protein structure. The *CCP*4 program suite (Collaborative Computational Project, Number 4, 1994) is an integrated collection of software for macromolecular crystallography, with a scope ranging from data processing to structure refinement and validation. Until recently, molecular graphics had not been part of the suite. With the recent computational and graphical performance of relatively cheap hardware, the time had arrived for *CCP*4 to provide graphical functionality for knowledge-based (semi-)-automatic building using powerful modern languages in a flexible extendible package. CCP4mg (Potterton *et al.*, 2004) is an initiative by *CCP*4 to provide libraries and a molecular graphics application that is a popular system for representation, modelling, structure determination, analysis and validation. The aim is to provide a system that is easy to use and a platform for developers who wish to integrate macromolecular computation with a molecular-graphics interface. There are several modules to such graphical functionality; the protein model-building/map-fitting tools described here are only a part. These tools are available as a stand-alone software package, *Coot*.

A map-fitting program has to provide certain functionality, which is not required by a molecular-display program. These functions include symmetry coordinates, electron-density map contouring and the ability to move the coordinates in various ways, such as model idealization or according to side-chain rotamer probabilities.

doi:10.1107/S0907444904019158

The map-fitting and model-building functions described here have a functionality broadly similar to that of programs such as *O*, *Xtalview* from *Xfit* (McRee, 1999) or *QUANTA* (Accelrys, San Diego, CA, USA). However, in the spirit of the *CCP*4 program suite, it is possible for others to read and modify the program.

Coot attempts (generally speaking) to provide more transparency, ease of use, better extendability, (semi-)automated model-building methods and convenient integration with programs of the *CCP*4 suite.

2. Program functions

Coot has been substantially built around two major libraries: mmdb (Krissinel *et al.*, 2004), a library for the handling of macromolecular coordinates, and Clipper (Cowtan, 2002, 2003), a library for crystallographic objects and computation thereof. The various functions of *Coot* are split into 'standalone' classes in the sense that an attempt has been made to minimize the dependence of the classes on anything other than the above libraries. With portability in mind, special effort was made not to introduce GUI dependences into the interface to *Coot*'s library of tools.

Coot is event-driven; functions are only run as a result of user action (typically moving or clicking the mouse).

2.1. Symmetry

Coordinate symmetry is recomputed and redisplayed at every recentre event. For each molecule for which the user wishes to display symmetry, symmetry atoms are displayed within a particular distance criterion of the display centre. By using a set of pre-computed guide points that mark the extents of the molecule and applying the symmetry operators and cell shifts to these guide points, a set of operator indexes and cell shifts are generated that may contain symmetry-related atoms close to the screen centre (where 'close' is defined by a user-settable parameter). For each of these sets, all atoms in the molecule are transformed and a check is made for each to see if it is within the symmetry display radius of the position at the centre of the screen. Thus, symmetry is kept current and relevant to the current display centre.

2.2. Electron density

Because *Coot* is based on the Clipper libraries, it is easy to generate maps by reading a file of structure factors that contain phase information (typically an MTZ file). Density is not limited to any particular part of the unit cell; the relevant symmetry-related density is generated (and then contoured) automatically using Clipper functionality. The electron-density maps can be simply recontoured (provoked by script or keyboard or mouse events) at a different level using a predetermined increment. Every map (displayed or undisplayed) is regenerated and contoured: this process is not optimally fast but simplifies the user interface.

2.3. Interface to *REFMAC*

On reading an MTZ file one can optionally assign parameters for running *REFMAC* (Murshudov *et al.*, 1997). *REFMAC* is a program of the *CCP*4 suite for maximum-likelihood-based macromolecular refinement. After a period of interactive model building, the user can choose to use *REFMAC* to refine the current coordinates (in combination with MTZ parameters). *Coot* blocks until *REFMAC* has terminated and then automatically reads the newly generated (refined) coordinates and MTZ file, from which a map is generated (and displayed).

2.4. Rigid-body refinement

Clipper library functions provide easy access to the map gradients. For a selected coordinate set, the map gradients at the atom centres are averaged. A shift is applied (to all the selected atoms) that is some simple fraction of the average gradient. The rotational component of the rigid-body refinement is generated in the following manner: the rotations to be calculated (α_x, α_y and α_z) are (small) rotations around the coordinate axes, the centre of rotation (V) being the centre of the rotating atoms. Let V_{p_i} be the projection on to the XY plane of the vector between the position of atom i and V, the unit vector being $\widehat{V_{p_i}}$. The dot product of the gradient with $\widehat{V_{p_i}}$ provides $d_{V_{p_i}}$.

The required angle is $\arctan(d_{V_{p_i}}/|V|)$. These angles are available for each atom and they are averaged to obtain three perpendicular rotations: α_x, α_y and α_z. These angle transformations are applied to the coordinates. The application of transformations continues until the average shift length is less than 0.0005 Å.

This is a reasonable approach for much of a protein's structure, but could behave badly where there is a combination of relatively heavy and light atoms, such as sulfates or methionines. This problem could be countered by weighting the atom-density score by the atomic weight.

2.5. Rotamers

The rotamer library used in *Coot* is the backbone-independent library of Dunbrack & Cohen (1997). It is formed from a reasonably large sample set (850 chains), is reasonably up to date (May 2002) and provides a more accurate estimation of the population of rare rotamers.

The *Coot* function 'Auto-fit Rotamer' takes a set of most likely rotamers for a particular side chain and generates coordinates for each rotamer. Each test rotamer is then rigid-body refined and the final position is scored according to the fit to the density (the residue's backbone atoms are included in the set of refined atoms). The best fit rotamer is chosen and replaces the previous coordinates.

2.6. Regularization and refinement

Molecular-graphics model building requires the ability to regularize ('idealize') the coordinates of the model. In order to do so, the ideal values of the geometry of the macromolecule should be known. These ideal values can come in various forms. The interface in *Coot* reads the mmCIF dictionaries of *REFMAC*, which define idea values and estimated standard deviations for bond lengths, angles, torsions planes and chiral centres. *Coot* uses the Polak–Ribiere variant of the BFGS (Broyden–Fletcher–Goldfarb–Shanno) conjugate-gradient multi-variable function minimizer to optimize the coordinates.

The analytical gradient derivations are described in Appendix *A*.

2.6.1. Fitting to the map. As described above, the map gradients are provided by a Clipper function. These map gradients (at the positions of the atom centres) are simply multiplied by a (user-changeable) scaling factor and added to the geometric terms to define the target function (this is called 'Refinement' in *Coot*).

2.7. Finding ligands

A map can be masked by a set of coordinates (typically those of the currently determined atoms of the protein model). This approach leaves a map that has positive density at places where there are no atoms to represent that density (similar, in fact, to an $F_o - F_c$ map). This masked map is searched for 'clusters' of density above a particular level. The clustering of the grid points of the asymmetric unit into potential ligand sites is performed conveniently using a recursive neighbour search of the map. The clusters are sorted according to size and electron-density value. Eigenvalues and eigenvectors are calculated for each cluster of grid points.

Similarly, the eigenvalues and eigenvectors of the search ligands (there can of course be just one search ligand) are computed (the parameters being the positions of the atom centres). The eigenvalues of the ligands are compared with the eigenvalues of each of the electron-density clusters and if they are sufficiently similar the ligand is placed into the cluster by matching the centre of the test ligand and the centre of the cluster. The ligand is oriented in each of the four different orientations that provide coinciding eigenvectors and then rigid-body refined and scored. The score is simply the sum of the electron density at the atom centres. The score at each site for each different ligand is compared and the best fit (highest score with sufficient fraction of atoms in positive density after the rigid-body refinement) is chosen. This last check ensures that oversized ligands are not fitted into small clusters.

2.7.1. Flexible ligands. Instead of having a series of different ligand compounds, the search ligands can be generated from a single ligand that has rotatable bonds. The ligand dictionary provides a description of the geometry of the ligand including torsions. These torsions are randomly sampled for a number of trials (by default 1000) to provide coordinates that can be checked against the potential ligand sites as described above. An enhancement would be to allow the determination of the number of trials to depend on the number of torsions.

2.7.2. Finding water molecules. The electron density is clustered as described for ligands. For clusters that have a volume below a certain upper limit (4.2 Å^3, which stops water molecules being placed in multi-atom ligand sites) a starting position is determined from the mean position of the grid coordinates of the cluster. This position is then optimized by refining the position to the local maximum as determined by cubic interpolation of the map. A map sphericity test is then applied; the variance of the cubic interpolated electron density at points 0.3, 0.6 and 0.9 Å from the local maximum in positive and negative offsets along the *x*, *y* and *z* axes are determined. The variances are summed and must be lower than a user-changeable cutoff (default 0.07 e^2 Å^{-6}). The successful positions are then compared with the coordinates of the protein's O and N atoms. If the distance is between user-changeable criteria (default 2.4–3.4 Å) then the position is accepted as a solvent O atom and (optionally) added to the protein model.

2.8. Add terminal residue

Given the selection of a terminal residue (which also could merely be the start of a gap of unplaced residues), two residue-type independent randomly selected φ/ψ pairs are made from Clipper's Ramachandran distribution of φ/ψ pairs. These angles are used to generate positions of C, C$^\alpha$, O and N main-chain atoms for the neighbouring two residues using the peptide geometry. This set of atoms then undergo rigid-body refinement to optimize the fit to the map. The score of the fit and the positions of the atoms are recorded. This procedure is then repeated a number of times (by default 100). The main-chain atoms of the neighbouring residue's best-fit atoms are then offered as a position of the neighbouring residue (the atoms of the next neighbouring residue are discarded).

2.9. Skeletonization and C$^\alpha$ building

Coot uses a Clipper map to generate and store the skeleton. This approach is convenient because, like electron-density maps, the skeleton can be displayed 'on the fly' anywhere in the crystal (*i.e.* it is not limited to a precalculated region). The Clipper skeletonization algorithm is similar to that employed in *DM* from *CCP*4 (Cowtan, 1994). A skeleton 'bond' (bone) is drawn between neighbouring map grid points if both parts are marked as skeleton points.

The skeleton can be further trimmed by recursive tip removal (a tip being a grid point with one or zero neighbours). This process removes side chains and, potentially, parts of the termini, but provides an easy means of identifying the fold and non-crystallographic symmetry.

Like some validation (Kleywegt, 1997) and other attempts at automated model building (Morris *et al.*, 2002; Oldfield & Hubbard, 1994; Oldfield, 2001), a likelihood distribution for the pseudo-torsion angle $C^\alpha(n)-C^\alpha(n+1)-C^\alpha(n+2)-C^\alpha(n+3)$ *versus* the angle $C^\alpha(n+1)-C^\alpha(n+2)-C^\alpha(n+3)$ has been generated from high-resolution structures in the PDB (Berman *et al.*, 2000) (Fig. 1). Once at least three C$^\alpha$ atoms have been placed, this is used as prior knowledge in the placement of the next C$^\alpha$ position in the following manner. Skeleton points between 2.4 and 3.4 Å from the current C$^\alpha$ position (which has an associated nearby skeleton point) are selected. These skeleton points are tested for direct connectivity to the current skeleton point. Skeleton points that are

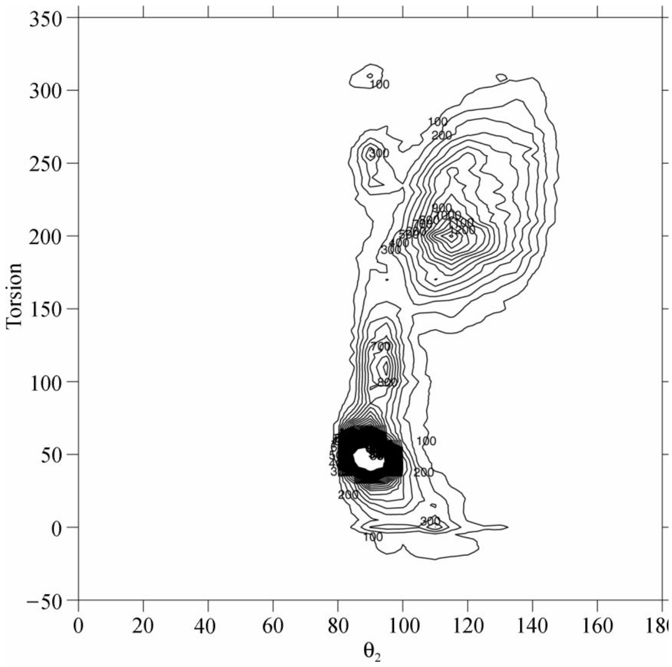

Figure 1
C$^{\alpha}$ pseudo-torsion angle *versus* opening angle for proteins in the PDB used in the likelihood assignment of potential C$^{\alpha}$ positions.

directly connected are assigned a score of 100; those that are unconnected are assigned a score of 0.1. For each selected skeleton point, a test point is then generated 3.8 Å from the current C$^{\alpha}$ position in the direction of skeleton point. A C$^{\alpha}$ pseudo-torsion angle and angle pair are generated from the position of the test point, the current C$^{\alpha}$ position and the two previously assigned C$^{\alpha}$ positions. This pseudo-torsion angle and angle pair are used to generate a score by looking up the value in the internal representation of Fig. 1 using cubic interpolation. This value is combined with the skeleton-based score for this particular test point. This procedure is then repeated in a 'look-ahead' manner, assuming that the test point is a member of the set of four C$^{\alpha}$ positions generating the C$^{\alpha}$ pseudo-torsion/angle pair. The most likely solution for the look-ahead is combined with the score for the current test point. The test points are then sorted by combined score and interactively offered as potential positions for the next C$^{\alpha}$ atom, the positions with the best score being offered first.

Occasionally (usually as a result of a positional error in the current C$^{\alpha}$ position), 3.8 Å is the wrong distance to the next correct C$^{\alpha}$ position; thus the user is allowed to change the length to something other than 3.8 Å.

The depth of the look-ahead in the current implementation is at level 1 but could trivially be extended (in tests, a level 2 look-ahead was better but took too long to be considered pleasantly interactive).

This algorithm has room for improvement: for example, by considering the value of the density at the test point and along the C$^{\alpha}$ pseudo-bond, one-third and two-thirds of the way along the bond (corresponding to positions that are close to the peptide C and N atoms; Oldfield, 2003).

C$^{\alpha}$ coordinates are converted to main-chain coordinates in a manner similar to that previously described (Jones & Thirup, 1986; Esnouf, 1997).

APPENDIX *A*
Regularization and refinement derivatives

The function that we are trying to minimize is S, where

$$S = S_{\rm bond} + S_{\rm angle} + S_{\rm torsion} + S_{\rm plane}.$$

Let us take these four parts in turn.

A1. Bonds

$$S_{\rm bond} = \sum_{i=1}^{N_{\rm bonds}} (b_i - b_{0_i})^2,$$

where b_{0_i} is the ideal length (from the dictionary) of the ith bond, $\mathbf{b}_i$ is the bond vector and b_i is its length.

$$\frac{\partial S_i}{\partial x_m} = \frac{\partial S_i}{\partial b_i}\frac{\partial b_i}{\partial x_m} = [2(b_i - b_{0_i})]\frac{\partial b_i}{\partial x_m},$$

$$b_i = [(x_m - x_k)^2 + (y_m - y_k)^2 + (z_m - z_k)^2]^{1/2}.$$

Therefore

$$\frac{\partial b_i}{\partial x_m} = \left(\frac{1}{2}\frac{1}{b_i}\right)2(x_m - x_k) = \frac{(x_m - x_k)}{b_i}$$

and

$$\frac{\partial S_i}{\partial x_m} = 2[b_i - b_{0_1}]\frac{(x_m - x_k)}{b_i}.$$

A2. Angles

We are trying to minimize $S_{\rm angle}$, where (for simplicity, the weights have been omitted)

$$S_{\rm angle} = \sum_{i=1}^{N_{\rm angles}} (\theta_i - \theta_{0_i})^2.$$

Angle θ is contributed to by atoms k, l and m:

$$\cos\theta = (\underline{a} \cdot \underline{b})/(ab),$$

where $\underline{a}$ is the bond of atoms k and l $[(x_k - x_l), (y_k - y_l), (z_k - z_l)]$ and $\underline{b}$ is the bond of atoms l and m $[(x_m - x_l), (y_m - y_l), (z_m - z_l)]$. Note that the vectors point away from the middle atom l.

Therefore,

$$\theta = a\cos(P), \tag{1}$$

where

$$P = (\underline{a} \cdot \underline{b})/(ab).$$

Using the chain rule,

$$\frac{\partial\theta}{\partial x_k}=\frac{\partial\theta}{\partial P}\frac{\partial P}{\partial x_k}. \qquad (2)$$

Given that we are only interested in θ in the range $0 \rightarrow \pi$,

$$\frac{\partial\theta}{\partial P}=-\frac{1}{\sin\theta}. \qquad (3)$$

Again using the chain rule,

$$\frac{\partial P}{\partial x_k}=Q\frac{\partial R}{\partial x_k}+R\frac{\partial Q}{\partial x_k}, \qquad (4)$$

where

$$Q=\underline{a}\cdot\underline{b}, \qquad (5)$$

$$R=1/(ab). \qquad (6)$$

A3. Angles: the middle atom

A middle atom is somewhat more tricky than an end atom because the derivatives of ab and $\underline{a}\cdot\underline{b}$ are not so trivial. Let us change the indexing so that we are actually talking about the middle atom, l.

Differentiating (6) gives

$$\frac{\partial R}{\partial x_l}=-\frac{1}{(ab)^2}b\frac{\partial a}{\partial x_l}-\frac{1}{(ab)^2}a\frac{\partial b}{\partial x_l}. \qquad (7)$$

$\partial a/\partial x_l$ here is exactly the same as for bonds,

$$\frac{\partial a}{\partial x_l}=\frac{x_l-x_k}{a}.$$

Similarly,

$$\frac{\partial b}{\partial x_l}=\frac{x_l-x_m}{a}.$$

Therefore, substituting these equations into (7) gives

$$\frac{\partial R}{\partial x_l}=-\frac{x_l-x_k}{a^3b}-\frac{x_l-x_m}{ab^3}.$$

Turning to Q, recall (5); therefore

$$Q=\big[(x_k-x_l)(x_m-x_l)+(y_k-y_l)(y_m-y_l)+(z_k-z_l)(z_m-z_l)\big]$$

and hence

$$\frac{\partial Q}{\partial x_l}=-(x_k-x_l)-(x_m-x_l).$$

Substituting all the above into (4) gives

$$\frac{\partial P}{\partial x_l}=(\underline{a}\cdot\underline{b})\Big(-\frac{x_l-x_k}{a^3b}-\frac{x_l-x_m}{ab^3}\Big)+\frac{-(x_k-x_l)-(x_m-x_l)}{ab}.$$

Combining this expression and (3) into (2) we obtain

$$\frac{\partial\theta}{\partial x_l}=\frac{1}{\sin\theta}\frac{\partial P}{\partial x_l}.$$

A4. Angles: an end atom (atoms *k* or *m*)

This case is simpler because there are no cross-terms in $\partial R/\partial x_k$ and $\partial Q/\partial x_k$.

$$\frac{\partial R}{\partial x_k}=\frac{(x_k-x_l)}{ab}.$$

and

$$\frac{\partial Q}{\partial x_k}=(x_m-x_l),$$

and so

$$\frac{\partial\theta}{\partial x_k}=-\frac{1}{\sin\theta}\left[\frac{(x_l-x_k)}{a^2}\cos\theta+\frac{x_m-x_l}{ab}\right]. \qquad (8)$$

A5. Torsion angles

The torsion angle is the angle between **a** × **b** and **b** × **c** (Fig. 2) and this can be written as

$$\arctan\big\{(\mathbf{a}\cdot\hat{\mathbf{b}}\times\mathbf{c})/\big[-\mathbf{a}\cdot\mathbf{c}+(\mathbf{a}\cdot\hat{\mathbf{b}})(\hat{\mathbf{b}}\cdot\mathbf{c})\big]\big\}, \qquad (9)$$

where $\hat{\mathbf{b}}$ is a unit vector in the direction of $\mathbf{b}$, $\hat{\mathbf{b}}=\mathbf{b}/b$.

This definition of the torsion angle is used rather than the more common definition, which uses three cross-products, because our version and its derivatives are faster to calculate.

Let us split the expression up into tractable portions; the evaluation of θ in the program will combine these expressions, starting at the end (the most simple).

From the primatives,

$$a_x=P_{2_x}-P_{1_x},\ b_x=P_{3_x}-P_{2_x},\ c_x=P_{4_x}-P_{3_x},$$

$$a_y=P_{2_y}-P_{1_y},\ b_y=P_{3_y}-P_{2_y},\ c_y=P_{4_y}-P_{3_y},$$

$$a_z=P_{2_z}-P_{1_z},\ b_z=P_{3_z}-P_{2_z},\ c_z=P_{4_z}-P_{3_z},$$

$$\theta=\arctan(D),$$

where

$$D=\frac{\mathbf{a}\cdot(\mathbf{b}\times\mathbf{c})/b}{-\mathbf{a}\cdot\mathbf{c}+(\mathbf{a}\cdot\mathbf{b})(\mathbf{b}\cdot\mathbf{c})/b^2}.$$

So

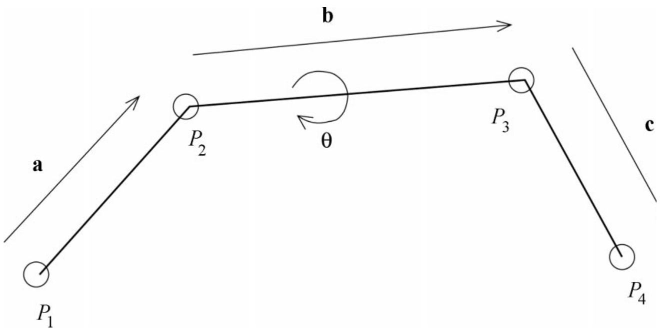

Figure 2
Nomenclature used for torsion angles.

$$\frac{\partial\theta}{\partial x_{P_1}} = \frac{\partial\theta}{\partial D}\frac{\partial D}{\partial x_{P_1}}, \qquad (10)$$

$$\frac{\partial\theta}{\partial x_{P_1}} = \frac{1}{1+D^2}\frac{\partial D}{\partial x_{P_1}}. \qquad (11)$$

Let

$$E = \mathbf{a}\cdot(\mathbf{b}\times\mathbf{c})/b,$$

$$F = [-\mathbf{a}\cdot\mathbf{c} + (\mathbf{a}\cdot\mathbf{b})(\mathbf{b}\cdot\mathbf{c})/b]^{-1},$$

$$F = 1/G, \qquad (12)$$

$$G = -\mathbf{a}\cdot\mathbf{c} + (\mathbf{a}\cdot\mathbf{b})(\mathbf{b}\cdot\mathbf{c})/b^2,$$

$$H = -\mathbf{a}\cdot\mathbf{c},$$

$$J = \mathbf{a}\cdot\mathbf{b},$$

$$K = \mathbf{b}\cdot\mathbf{c},$$

$$L = 1/b^2.$$

Differentiating (12) gives

$$\frac{\partial F}{\partial x_{P_1}} = -\frac{1}{G^2}\frac{\partial G}{\partial x_{P_1}}.$$

Substituting for the derivative in (10),

$$\frac{\partial\theta}{\partial x_{P_1}} = \frac{1}{1+D^2}\left(F\frac{\partial E}{\partial x_{P_1}} + E\frac{\partial F}{\partial x_{P_1}}\right).$$

We also have

$$G = H + JKL.$$

Differentiating this gives

$$\frac{\partial G}{\partial x_{P_1}} = \frac{\partial H}{\partial x_{P_1}} + JL\frac{\partial K}{\partial x_{P_1}} + KL\frac{\partial J}{\partial x_{P_1}} + JK\frac{\partial L}{\partial x_{P_1}}.$$

The H, J, K and L derivatives are

$$H = -\mathbf{a}\cdot\mathbf{c} = -a_x c_x - a_y b_y - a_z c_z,$$

$$\frac{\partial H}{\partial x_{P_1}} = c_x, \quad \frac{\partial H}{\partial x_{P_2}} = -c_x, \quad \frac{\partial H}{\partial x_{P_3}} = a_x, \quad \frac{\partial H}{\partial x_{P_4}} = -a_x,$$
$$\frac{\partial K}{\partial x_{P_1}} = 0, \quad \frac{\partial K}{\partial x_{P_2}} = -c_x, \quad \frac{\partial K}{\partial x_{P_3}} = c_x + b_x,$$
$$\frac{\partial K}{\partial x_{P_4}} = b_x, \quad \frac{\partial J}{\partial x_{P_1}} = -b_x, \quad \frac{\partial J}{\partial x_{P_2}} = b_x - a_x,$$
$$\frac{\partial J}{\partial x_{P_3}} = a_x, \quad \frac{\partial J}{\partial x_{P_4}} = 0.$$

The $\partial b/\partial x$ terms are just like the bond derivatives,

$$\frac{\partial L}{\partial x_{P_1}} = \frac{\partial L}{\partial b}\frac{\partial b}{\partial x_{P_1}},$$

i.e.

$$\frac{\partial L}{\partial x_{P_3}} = -\frac{2}{b^3}\frac{x_{P_3} - x_{P_2}}{b} = -\frac{2(x_{P_3} - x_{P_2})}{b^4}.$$

The derivative with respect to x_{P_2} has the opposite sign.

Notice that **b** involves only atoms P_2 and P_3, so that the derivates of L with respect to the P_1 and P_4 coordinates are zero.

A6. Torsion angles: $\partial E/\partial x$ terms

Recall that

$$E = \mathbf{a}\cdot(\mathbf{b}\times\mathbf{c})/b.$$

Let

$$M = \mathbf{a}\cdot(\mathbf{b}\times\mathbf{c}),$$

i.e.

$$E = M/b.$$

Differentiating gives

$$\frac{\partial E}{\partial x_{P_3}} = -\frac{M}{b^2}\frac{\partial b}{\partial x_{P_3}} + \frac{1}{b}\frac{\partial M}{\partial x_{P_3}},$$

where, as for bonds,

$$\frac{\partial b}{\partial x_{P_3}} = \frac{x_{P_3} - x_{P_2}}{b}.$$

However, note again that the derivative of b is zero for atoms P_1 and P_4, *i.e.* for atoms P_2 and P_3

$$\frac{\partial E}{\partial x_{P_3}} = -\frac{M(x_{P_3} - x_{P_2})}{b^3} + \frac{1}{b}\frac{\partial M}{\partial x_{P_3}},$$

but for atoms P_1 and P_4

$$\frac{\partial E}{\partial x_{P_1}} = \frac{1}{b}\frac{\partial M}{\partial x_{P_1}},$$

$$M = a_x(b_y c_z - b_z c_y) + a_y(b_z c_x - b_x c_z) + a_z(b_x c_y - b_y c_x).$$

So here are the primitives of $M = \mathbf{a}\cdot(\mathbf{b}\times\mathbf{c})$

$$\frac{\partial M}{\partial x_{P_1}} = -(b_y c_z - b_z c_y),$$

$$\frac{\partial M}{\partial x_{P_2}} = (b_y c_z - b_z c_y) + (a_y c_z - a_z c_y),$$

$$\frac{\partial M}{\partial x_{P_3}} = (a_z c_y - a_y c_z) + (b_y a_z - b_z a_y),$$

$$\frac{\partial M}{\partial x_{P_4}} = (a_y b_z - a_z b_y),$$

$$\frac{\partial M}{\partial y_{P_1}} = -(b_z c_x - b_x c_z),$$

$$\frac{\partial M}{\partial y_{P_2}} = (b_z c_x - b_x c_z) + (a_z c_x - a_x c_z),$$

$$\frac{\partial M}{\partial y_{P_3}} = -(a_z c_x - a_x c_z) + (b_z a_x - b_x a_z),$$

$$\frac{\partial M}{\partial y_{P_4}} = -(b_z a_x - b_x a_z),$$

$$\frac{\partial M}{\partial z_{P_1}} = -(b_x c_y - b_y c_x),$$

$$\frac{\partial M}{\partial z_{P_2}} = (b_x c_y - b_y c_x) + (a_x c_y - a_y c_x),$$

$$\frac{\partial M}{\partial z_{P_3}} = -(a_x c_y - a_y c_x) + (a_y b_x - a_x b_y),$$

$$\frac{\partial M}{\partial z_{P_4}} = -(a_y b_x - a_x b_y).$$

A7. Combining terms

Combining, we obtain the following expression for the derivative of torsion angle θ in terms of the primitive derivates,

$$\frac{\partial \theta}{\partial x_{P_1}} = \frac{1}{(1+\tan^2\theta)} \frac{\partial D}{\partial x_{P_1}},$$

where

$$\frac{\partial D}{\partial x_{P_1}} = \left[F \frac{\partial E}{\partial x_{P_1}} - \frac{E}{G^2} \left(\frac{\partial H}{\partial x_{P_1}} + JL \frac{\partial K}{\partial x_{P_1}} + KL \frac{\partial J}{\partial x_{P_1}} + JK \frac{\partial L}{\partial x_{P_1}} \right) \right].$$

A8. Planes

$$S_{\rm plane} = \sum_{i=1}^{N_{\rm planes}} \sum_{j=1}^{N_{{\rm atoms}_i}} e_{ij}^2,$$

where e_{ij} is the distance of the ith plane restraint's jth atom from the ith plane restraint's least-squares plane.

Recall the equation of a plane: $ax + by + cz + d = 0$. Firstly, the centres of the sets of atoms, $x_{\rm cen}$, $y_{\rm cen}$, $z_{\rm cen}$, are determined. The plane is moved so that it crosses the origin and therefore $d = 0$ (it is moved back later). The problem then involves three equations, three unknowns and an eigenvalue problem, with the smallest eigenvalue corresponding to the best-fit plane.

The least-squares planes of the plane restraints are recalculated at every iteration.

The authors thank Garib Murshudov, Eleanor J. Dodson, Jack Quine and the many *Coot* testers. KC is supported by The Royal Society (grant No. 003R05674). PE is funded by BBSRC grant No. 87/B17320.

References

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N. & Bourne, P. E. (2000). *Nucleic Acids Res.* **28**, 235–242.

Collaborative Computational Project, Number 4 (1994). *Acta Cryst.* D**50**, 760–766.

Cowtan, K. (1994). *Jnt CCP4/ESF–EACBM Newsl. Protein Crystallogr.* **31**, 34–38.

Cowtan, K. (2002). *Jnt CCP4/ESF–EACBM Newsl. Protein Crystallogr.* **40**.

Cowtan, K. (2003). *Crystallogr. Rev.* **9**, 73–80.

Dunbrack, R. L. Jr & Cohen, F. E. (1997). *Protein Sci.* **6**, 1661–1681.

Esnouf, R. M. (1997). *Acta Cryst.* D**53**, 665–672.

Jones, T. A. (1978). *J. Appl. Cryst.* **11**, 268–272.

Jones, T. A., Cowan, S., Zou, J.-Y. & Kjeldgaard, M. (1991). *Acta Cryst.* A**47**, 110–119.

Jones, T. A. & Thirup, S. (1986). *EMBO J.* **5**, 891–822.

Kleywegt, G. J. (1997). *J. Mol. Biol.* **273**, 371–376.

Krissinel, E. B, Winn, M. D., Ballard, C. C., Ashton, A. W., Patel, P., Potterton, E. A., McNicholas, S. J., Cowtan, K. D. & Emsley, P. (2004). *Acta Cryst.* D**60**, 2250–2255.

McRee, D. E. (1999). *J. Struct. Biol.* **125**, 156–165.

Morris, R. J., Perrakis, A. & Lamzin, V. S. (2002). *Acta Cryst.* D**58**, 968–975.

Murshudov, G. N., Vagin, A. A. & Dodson, E. J. (1997). *Acta Cryst* D**53**, 240–255.

Oldfield, T. J. (2001). *Acta Cryst.* D**57**, 82–94.

Oldfield, T. J. (2003). *Acta Cryst.* D**59**, 483–491.

Oldfield, T. J. & Hubbard, R. E. (1994). *Proteins Struct. Funct. Genet.* **18**, 324–337.

Potterton, L., McNicholas, S., Krissinel, E., Gruber, J., Cowtan, K., Emsley, P., Murshudov, G. N., Cohen, S., Perrakis, A. & Noble, M. (2004). *Acta Cryst.* D**60**, 2288–2294.