Acta Crystallographica Section A
Foundations and Advances
ISSN 2053-2733

Received 19 June 2014
Accepted 8 October 2014

The anatomy of a comprehensive constrained, restrained refinement program for the modern computing environment – *Olex2* dissected

Luc J. Bourhis,[a]* Oleg V. Dolomanov,[b] Richard J. Gildea,[c] Judith A. K. Howard[d] and Horst Puschmann[b]

[a]Bruker AXS–SAS, 4 Allée Lorentz, 77447 Marne-la-Vallée cedex 2, France, [b]OlexSys Ltd, Department of Chemistry, Durham University, South Road, Durham, DH1 3LE, England, [c]Diamond Light Source Ltd, Diamond House, Harwell Oxford, Didcot, Oxfordshire, OX11 0DE, England, and [d]Department of Chemistry, Durham University, South Road, Durham, DH1 3LE, England. Correspondence e-mail: luc_j_bourhis@mac.com

This paper describes the mathematical basis for *olex2.refine*, the new refinement engine which is integrated within the *Olex2* program. Precise and clear equations are provided for every computation performed by this engine, including structure factors and their derivatives, constraints, restraints and twinning; a general overview is also given of the different components of the engine and their relation to each other. A framework for adding multiple general constraints with dependencies on common physical parameters is described. Several new restraints on atomic displacement parameters are also presented.

1. Introduction

During the last four decades, small-molecule crystallographers have used a myriad of stand-alone routines and various comprehensive software packages, most notably those written by Sheldrick (1997, 2008) and that designed by Bob Carruthers and John Rollett, then maintained and enhanced by David Watkin (Betteridge *et al.*, 2003). These latter two packages have dominated the citations in all publications containing crystal structure analyses for many years now.

Therefore it might be claimed that the analysis of single-crystal X-ray (and neutron) diffraction data has reached a certain level of maturity and that there is no need for further program development. Nonetheless, there is still considerable activity in this area by various groups around the world and there has been a new release, *SHELX2013*, from Sheldrick recently. A retro-fit of new ideas to these older programs is not always possible, except perhaps by the authors themselves.

When we first embarked on the project herein reported[1] we were clear about one point – namely that we wanted to create a new and flexible refinement engine, working in a collaborative environment and to be based on trusted and mature pre-existing code. This new refinement engine would be open source and therefore available for verification and modification by others. Modern programming paradigms, unknown when the aforementioned packages were created in the 1960s, would form the basis of our development. It is hoped that such an architecture will allow extensions to the code without breaking the program or endangering the underlying functionality. It is this that we describe below in further detail and which now forms the underlying code for *olex2.refine* (http://www.olex2.org).

We decided to base the computational core of *olex2.refine* on a pre-existing project, the Computational Crystallography Toolbox (*cctbx*), that provides the foundation for the macromolecular refinement facilities in the *PHENIX* suite of Adams *et al.* (2010). We made that choice because it provided the solid and versatile tools (Grosse-Kunstleve & Adams, 2003) that we needed to develop our project. The most important of them is a comprehensive and robust toolbox to handle crystal symmetries (*sgtbx*) that supports every space group in any setting, even rather unusual settings such as tripled cells. Another key *cctbx* toolbox is the *eltbx*, which is concerned with scattering-factor computations for any atom or ion and any wavelength that could be encountered in practice. The *cctbx* also featured many of the restraints we needed, in the module *cctbx.restraints*.

For our project, we contributed to it a Small-Molecule Toolbox (*smtbx*) which features, in particular, the constraint framework presented in detail in §3 and Appendix *C*, and the computation of structure factors presented in Appendix *A*. We also contributed several new restraints to *cctbx*, restraints which are discussed in §4 and Appendix *D*. Finally, both the *cctbx* and the *smtbx* use the Scientific Toolbox (*scitbx*) for arrays, matrices, special functions and other purely mathematical matters. We have contributed a new least-squares toolbox (*lstbx*) to the *scitbx*, which provides flexible tools to deal with generic linear and non-linear problems, with or

[1] EPSRC grant 'Age Concern: Crystallographic Software for the Future' (EP/C536274/1).

without an unknown overall scale factor. It implements the method based on normal equations presented in Appendix *B*.

The program *Olex2* relies either on *SHELX* or on the *smtbx* to refine structures. If using the latter, *Olex2* converts its internal model of a structure into the objects used by the *smtbx* and the *cctbx* to represent the unit cell, reflections, crystal symmetry, constraints, restraints *etc.* Once the *smtbx* has produced the desired results they are sent back to *Olex2* which converts them to its own representations, so as to perform diverse post-processing and eventually display the results to the user of the program. Fig. 1 summarizes the dependencies between the various modules and programs that have just been described.

2. Least-squares refinement

A small-molecule structure refinement typically minimizes the weighted least-squares (LS) function

$$L = \underbrace{\sum_h w_h \left(Y_{o,h} - K Y_{c,h}\right)^2}_{L_{\text{data}}} + \underbrace{\sum_{\text{restraint } i} w_i \left(T_{o,i} - T_{c,i}\right)^2}_{L_{\text{restraints}}}, \qquad (1)$$

where Y_o (respectively, Y_c) denotes either the measured amplitude F_o (respectively, the modulus of the calculated complex structure factor $|F_c|$) or the measured intensity $I_o = F_o^2$ (respectively, the calculated intensity $I_c = |F_c|^2$), whereas K is an overall, unknown scale factor that places Y_c on the same scale as Y_o. The first sum over h runs over a set of m non-equivalent reflections that have been observed. Each observation is given an appropriate weight, w_h, based on the reliability of the measurement. These may be pure statistical weights, $w = 1/\sigma^2(Y_o)$, where σ is the estimated standard deviation of Y_o, though more complex weighting schemes are usually used. In the most general case, w_h is a function of $Y_{o,h}$, $\sigma(Y_{o,h})$, $Y_{c,h}$ and h itself, whereas the most common weighting schemes exhibit only a dependence on the first three.

These X-ray observations can be supplemented with the use of 'observations of restraint', as suggested by Waser (1963), where additional information such as target values for bond lengths, angles *etc.* is included in the minimization. This is the origin of the second term of the sum, where T_o is the target value for our restraint, and T_c is the value of the target function calculated using the current model (see, for example, Giacovazzo *et al.*, 2011; Watkin, 2008). This term $L_{\text{restraints}}$ may of course be absent. Conversely, the data term could be absent in a refinement against geometrical terms only [*cf.* the program *DLS-76* by Baerlocher *et al.* (1978) for example].

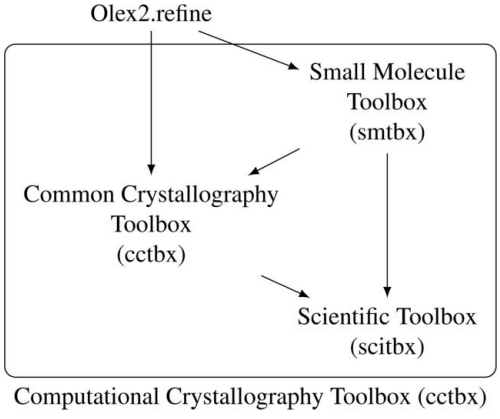

Figure 1
A high-level depiction of the software modules involved in *olex2.refine*: an arrow is drawn from each component pointing towards another component it uses.

In equation (1), the crystallographic parameters [*i.e.* atomic positions, atomic displacement parameters (ADPs) and chemical occupancy in routine refinements] that each component of Y_c and T_c depends upon will be collectively denoted as the vector $y = (y_1, \ldots, y_p)$ and we can therefore denote those dependencies with the compact notations $Y_c(y)$ and $T_c(y)$. The parameters that are actually refined may be a different set $x_1, \ldots, x_n$, collectively denoted as a vector x. We assume that the dependency of y upon x is known analytically. Since the scale factor K is unknown, our problem is the minimization of L with respect to all of $K, x_1, \ldots, x_n$. We will denote that dependency as $L[y(x), K]$, or more tersely when we do not need to remember the crystallographic parameter vector y, as $L(x, K)$. These notations reflect the important fact that we treat the scale factor K separately, as will be explained later.

For a small-molecule structure with a high data-to-parameter ratio, such unconstrained minimization as defined by the first term of equation (1), when $y \equiv x$, may well be sufficient. However, as the structure becomes larger, or the data-to-parameter ratio worsens, unconstrained minimization may not be well behaved, or result in some questionable parameter values. It has become customary to rely on the use of two techniques to solve such issues:

Restraints: by having a non-zero second term in equation (1) – with the use of appropriate weighting of the restraints – the minimization is gently pushed towards giving a chemically sensible and hopefully correct structure.

Constraints: by having a parameter vector x shorter than y, therefore explicitly taking into account exact relationships between some of the parameters y_i.

Whether the refinement uses restraints or constraints or both, at each refinement cycle, we first find the value of K that minimizes $L(x, K)$ keeping the parameter vector x fixed: we will denote it by $\tilde{K}(x)$, where we use a notation that is explicit in its dependency on the value of the refined parameters at the beginning of the cycle. We then search for the shift vector s of the parameter vector x that brings $L[x, \tilde{K}(x)]$ closer to the minimum. It is solution of the so-called normal equations,

$$(B_{\text{data}} + B_{\text{restraints}})s = -(g_{\text{data}} + g_{\text{restraints}}), \qquad (2)$$

where B_{data} and $B_{\text{restraints}}$ are the so-called normal matrices associated with the two terms sharing the same labels in equation (1), respectively, and where the right-hand side g_{data} and $g_{\text{restraints}}$ are equivalently associated with those two terms [see the derivation of equation (72) in Appendix *B*].

The rest of the paper is organized as follows. §3 deals with the computation of the derivatives with respect to the refined parameters $x_1, \ldots, x_n$ from the derivatives with respect to the

crystallographic parameters $y_1, \ldots, y_p$. The computation of the latter derivatives is expounded in Appendix *A* along with the formulae for the complex structure factors F_c. Appendix *C* details the computation of the value and derivatives of constrained parameters for each constraint featured by *olex2.refine*. §4 is devoted to the general principles of the computation of $B_{\text{restraints}}$ and $g_{\text{restraints}}$. It is completed by Appendix *D*, which gives the formulae for each restraint featured by *olex2.refine*. §5 deals with the refinement of twinned structures. §6 details the computation of standard uncertainties (s.u.'s), emphasizing the influence of constraints. §7 gives a very quick overview of the output of the results of a refinement. Appendix *B* presents the method used to find the optimal scale factor *K*, and then the optimal value of all the other refined parameters.

3. Constraints

3.1. Synopsis

The difference between restraints and constraints may be conceptualized mainly in two manners that we will first illustrate with a simple example: a geometrically constrained acetylenic hydrogen, $X{\equiv}C{-}H$. The position of the hydrogen must be such that the distance CH is equal to some ideal bond length d and such that the angle $X\widehat{C}H$ is equal to 180°. Expressed in such an implicit manner, those restrictions could be used to construct restraints by adding to the function to minimize a term $w_1(\mathrm{CH} - d)^2 + w_2(\cos X\widehat{C}H - 1)^2$. By doing so, the number of refined parameters would not be changed but the number of observations would increase by three.

It is, however, traditional to do the opposite, by reducing the number of parameters by three and keeping the number of observations unchanged. This is achieved by using the implicit form of the constraints to express the position of H as a function of the positions of the two carbon atoms. Denoting by x the triplet of coordinates of the atoms,

$$x_{\mathrm{H}} = x_{\mathrm{C}} + d\frac{x_{\mathrm{C}} - x_X}{||x_{\mathrm{C}} - x_X||}, \tag{3}$$

where $||.||$ denotes the Euclidean norm. By using this formula, the observable Y_c of the fit (either $|F_c|^2$ or $|F_c|$) that used to depend on x_{C}, x_X and x_{H} is replaced by a function $\tilde{Y}_c$ of x_{C} and x_X but not of x_{H}. We will call this transformation a reparametrization: $\tilde{Y}_c(x_{\mathrm{C}}, x_X) = Y_c[x_{\mathrm{C}}, x_X, x_{\mathrm{H}}(x_{\mathrm{C}}, x_X)]$ and we will say that $x_{\mathrm{H}}(x_{\mathrm{C}}, x_X)$ is a reparametrization of x_{H}, whose arguments are x_{C} and x_X.

The derivatives for the remaining variables are obtained with the chain rule

$$\begin{aligned}\frac{\partial\tilde{Y}_c}{\partial x_{\mathrm{C}}} &= \frac{\partial Y_c}{\partial x_{\mathrm{C}}} + \frac{\partial Y_c}{\partial x_{\mathrm{H}}}\frac{\partial x_{\mathrm{H}}}{\partial x_{\mathrm{C}}},\\ \frac{\partial\tilde{Y}_c}{\partial x_X} &= \frac{\partial Y_c}{\partial x_X} + \frac{\partial Y_c}{\partial x_{\mathrm{H}}}\frac{\partial x_{\mathrm{H}}}{\partial x_X}.\end{aligned} \tag{4}$$

We use the following compact notations for derivatives: for a column vector

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix},$$

$\partial F/\partial x$ will always denote the row vector

$$\left(\frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial x_2}, \frac{\partial F}{\partial x_3}\right).$$

Given another column vector

$$y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix},$$

$\partial y/\partial x$ will always denote the matrix

$$\left(\frac{\partial y_i}{\partial x_j}\right)_{1\le i,j\le 3}$$

where i (respectively, j) indexes the rows (respectively, the columns). The identity matrix will be denoted by **1**.

It should be noted that it is customary to work within the 'riding' approximation,

$$\frac{\partial x_{\mathrm{H}}}{\partial x_X} = 0, \quad \frac{\partial x_{\mathrm{H}}}{\partial x_{\mathrm{C}}} = \mathbf{1}, \tag{5}$$

which results in the much simplified chain rule

$$\begin{aligned}\frac{\partial\tilde{Y}_c}{\partial x_{\mathrm{C}}} &= \frac{\partial Y_c}{\partial x_{\mathrm{C}}} + \frac{\partial Y_c}{\partial x_{\mathrm{H}}},\\ \frac{\partial\tilde{Y}_c}{\partial x_X} &= \frac{\partial Y_c}{\partial x_X},\end{aligned} \tag{6}$$

implemented in all refinement programs, including *Olex2*.

It is not always the case that all three coordinates of an atom are removed from the refinement by constraints. For example, an atom in the plane of the mirror z, y, x whose matrix reads

$$M = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \tag{7}$$

has its coordinates $x = (x_1, x_2, x_3)$ constrained by the relation $Mx = x$, which can be reduced to $x_1 = x_3$ only. The corresponding reparametrization may be written

$$x_1 = u_1, \quad x_2 = u_2, \quad x_3 = u_1, \tag{8}$$

the observable Y_c becoming now a function of the vector of newly introduced refinable parameters $u = (u_1, u_2)$. There is a general algorithm implemented in the *cctbx* that for any special position returns a matrix Z so that the reparametrization takes the form

$$x = Zu + z, \tag{9}$$

where Z is a 3×2 or 3×1 matrix and z is a 3-vector. This algorithm first determines the space-group symmetries that leave the site invariant. The resulting system of linear equations, which read $Mx = x$ in our example, is then reduced to a triangular form from which the matrix Z and the vector z are then readily obtained. This algorithm does not therefore try to determine which components of u, if any, are also components

of x. This is why we have presented the reparametrization for our example in this general form instead of keeping x_1 and x_2 as refinable parameters, which would sound more intuitive in the first place. This matrix Z is the matrix of constraints for the position of that atom.

The anisotropic displacement tensor U is subject to symmetry constraints as well. In our example, it must satisfy

$$MUM^T = U, \tag{10}$$

where M^T denotes the transpose of the matrix M (see *e.g.* Giacovazzo *et al.*, 2011). Any number of such matrix equations can always be rewritten as a system of equations whose most general form reads

$$P\begin{pmatrix} U_{11} \\ U_{22} \\ U_{33} \\ U_{12} \\ U_{13} \\ U_{23} \end{pmatrix} = 0, \tag{11}$$

where P has 6 columns and $6n$ rows, where n is the number of symmetry elements other than the identity involved in the special position [indeed equation (10), being trivial for $M = \mathbf{1}$, can safely be discarded]. In our example,

$$P = \begin{pmatrix} -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \end{pmatrix} \tag{12}$$

and equation (11) reduces to

$$U_{11} = U_{33} \text{ and } U_{12} = U_{23}, \tag{13}$$

leading to the constraint matrix

$$\begin{pmatrix} U_{11} \\ U_{22} \\ U_{33} \\ U_{12} \\ U_{13} \\ U_{23} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}. \tag{14}$$

Then one would refine (v_1, v_2, v_3, v_4) instead of the U_{ij}. The *cctbx* provides an algorithm that computes this matrix P for any special positions, and then reduces it to triangular form in order to determine the constraint matrix for the ADP.

In other cases, a reparametrization will make some crystallographic parameters disappear while introducing new refinable parameters. A typical example is that of a tetrahedral $X{-}CH_3$, as the geometrical constraints leave one degree of freedom, a rotation about the axis $X{-}C$. Thus the reparametrization expresses the coordinates of the hydrogen atoms H_0, H_1 and H_2 as functions of the coordinates of the carbon atoms, and of an angle φ modelling that rotation,

$$x_{\mathrm{H}n} = x_\mathrm{C} + d\left\{\sin\alpha\left[\cos\left(\varphi + n\frac{2\pi}{3}\right)e_1 + \sin\left(\varphi + n\frac{2\pi}{3}\right)e_2\right] - \cos\alpha e_0\right\}, \tag{15}$$

where $\alpha \simeq 109.5^\circ$ and (e_0, e_1, e_2) is an orthonormal basis of column vectors with e_0 in the direction of the bond $X \rightarrow \mathrm{C}$. The riding approximation in this case consists of neglecting the derivatives of those base vectors, leading to

$$\begin{aligned} \frac{\partial \tilde{Y}_\mathrm{c}}{\partial x_\mathrm{C}} &= \frac{\partial Y_\mathrm{c}}{\partial x_\mathrm{C}} + \frac{\partial Y_\mathrm{c}}{\partial x_\mathrm{H}}, \\ \frac{\partial \tilde{Y}_\mathrm{c}}{\partial x_X} &= \frac{\partial Y_\mathrm{c}}{\partial x_X}, \\ \frac{\partial \tilde{Y}_\mathrm{c}}{\partial \varphi} &= \frac{\partial Y_\mathrm{c}}{\partial x_\mathrm{H}} d\sin\alpha\left[-\sin\left(\varphi + n\frac{2\pi}{3}\right)e_1 + \cos\left(\varphi + n\frac{2\pi}{3}\right)e_2\right]. \end{aligned} \tag{16}$$

Thus a new derivative with respect to the new refinable parameter φ is introduced by this reparametrization.

The last important concept is that of the chaining or composition of reparametrizations, that we will illustrate with a combination of the examples above. This example is not particularly common but it is a simple illustration of the concept we want to introduce. In the CH_3 case, the atoms C and X could be on the special position studied in the next-to-last example. One type of disorder could be modelled by first applying the reparametrization (15) and then reparametrizing x_C and x_X using equation (8), introducing parameters (u, v) for the former and (u', v') for the latter. The derivatives would then be obtained by the chain rule, *e.g.*

$$\frac{\partial x_{\mathrm{H}_0}}{\partial u} = \frac{\partial x_{\mathrm{H}_0}}{\partial x_\mathrm{C}}\frac{\partial x_\mathrm{C}}{\partial u} = 2 \tag{17}$$

in the riding approximation. This composition of reparametrization may be represented as a graph: each parameter (some of them are triplets of coordinates, others are scalars), $x_{\mathrm{H}_0}, x_{\mathrm{H}_1}, x_{\mathrm{H}_2}, x_\mathrm{C}, x_X, \varphi, u, v, u'$ and v', constitutes a node of that graph, whereas edges are drawn from each node to its arguments, *i.e.* the nodes it depends upon, as shown in Fig. 2. In this example, x_C has only one argument, u, whereas x_{H_0} has three

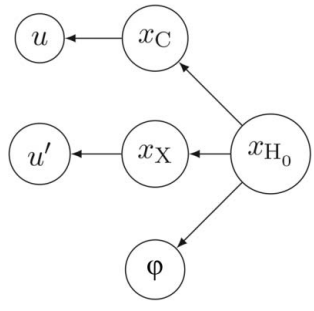

Figure 2
Example of a dependency graph for the chain of reparametrization discussed in §3. Only the part for hydrogen atom H_0 is shown.

arguments, x_C, x_X and φ. The *smtbx* implements reparametrizations by explicitly building such a graph.

As models become more complex, *e.g.* hydrogen atoms riding on the atoms of a rigid body whose rotation centre is tied to an atom whose coordinates are refined, the reparametrization graph becomes deeper. We decided not to put arbitrary limits on that graph. Indeed, we could have made a closed list of reparametrizations and of reparametrization combinations that our framework would accept but instead we decided to write our code so that it could correctly handle the computation of parameter values and of partial derivatives for arbitrary reparametrizations, combined in arbitrary ways. This framework is therefore open as new types of reparametrizations can be added, without the need to change the basic infrastructure in any way, and without the risk of breaking existing reparametrizations. This has proven very useful to the authors as this enabled them to incrementally add the wealth of constraints now available, some of which are unique to *olex2.refine*, as discussed in Appendix *C*. Furthermore, it enables third parties to develop their own constraints without the need for the involvement of the original authors beyond documenting how the reparametrization framework works.

A crystallographic refinement may involve many such reparametrizations. By piecing them all together, we obtain one reparametrization that expresses all refinable crystallographic parameters as a smaller vector of independent parameters that shall then be refined. Our framework safeguards that piecewise construction in several ways. First, at most one reparametrization may be applied to any given parameter. An attempt to add a reparametrization to a parameter that is already subject to one would be rejected by our framework as an error. Then if a cycle were found in the dependency graph, the framework would also reject the parametrization. This would happen when at least one parameter, through a series of reparametrization combinations, depends upon itself. Thus our framework safeguards against incorrect user inputs, and also against bugs in our own code that automatically builds constraints.

3.2. Constraint matrix

There is a wealth of algorithms designed to minimize least squares, but crystallographic software has only implemented a few of them. The two most popular methods have historically been the full matrix[3] (all small-molecule programs, including *Olex2*, as explained in Appendix *B*) and conjugate gradient LS (CGLS, see *e.g.* Björck, 1996) which *SHELXL* offers as an option along with full matrix. The macromolecular community later introduced the limited-memory Broyden–Fletcher–Goldfarb–Shanno method (LBFGS, Nocedal, 1980), *phenix.refine* (Afonine *et al.*, 2012) and a sparse Gauss–Newton algorithm, *REFMAC* (Murshudov *et al.*, 2011, §4 and references therein). All methods mentioned above have in common the fact that they require only the computation of the value and the first-order derivatives of the calculated F or F^2, that we have denoted as Y_c in this paper. Since no higher-order derivatives are used, the implementation of constrained least squares is greatly simplified.

Indeed, after transforming every constraint on the model into a reparametrization (as explained and exemplified in the previous section) and piecing all those reparametrizations together, we obtain a global reparametrization of the vector of crystallographic parameters $y = (y_1, \ldots, y_p)$ as a function of the vector $x = (x_1, \ldots, x_n)$ of the parameters that are actually refined. It should be noted that any component y_k of y that is not reparametrized – *e.g.* a coordinate of the pivot atom on which another atom rides, and of course any parameter that is not involved in any constraint – is also a component x_l of x, *i.e.*

$$\frac{\partial y_k}{\partial x_j} = \begin{cases} 1 & \text{if } j = l, \\ 0 & \text{if } j \neq l. \end{cases}$$

From the known analytical expression of $Y_c(x)$, one computes the derivatives of $\partial Y_c/\partial y_i$ (the reader is referred to Appendix *A* for a detailed presentation of those computations). The minimization algorithm then only needs the derivatives of

$$\tilde{Y}_c(x) = Y_c[y(x)] \tag{18}$$

with respect to each x_i, which are easily obtained by a simple application of the chain rule

$$\frac{\partial \tilde{Y}_c}{\partial x} = \frac{\partial Y_c}{\partial y}\,\frac{\partial y}{\partial x}. \tag{19}$$

The matrix $\partial y/\partial x$ is known as the constraint matrix in crystallographic circles.[4] In standard mathematical nomenclatures, it is called the Jacobian of the transformation $x \mapsto y$ and we will therefore denote it J.

The computation of J takes advantage of the fact that it is a very sparse matrix. This stems from the fact that any given crystallographic parameter y_i depends on very few refined parameters x_j, as amply illustrated in the previous section. We take advantage of this property to drastically reduce the memory cost of J and the cost of computing each of its non-zero coefficients. More precisely, both of these costs scale as the number of non-zero elements in J instead of $O(n^2)$ if J were treated as a dense matrix.

4. Restrained least-squares refinement

In this section we will give an overview of the computations involved in restrained refinement. The mathematical formulae for each restraint objective and their derivatives can be found in Appendix *D*.

Since each restraint target $T_{c,i}$ only depends on a very small subset of the y_j (*e.g.*, in the case of a bond length, only the six coordinates of the two bonded atoms would play a role), the matrix of derivatives with respect to the crystallographic parameters

[3] This method is also known as the Gauss–Newton algorithm.

[4] The earliest appearance of that concept in crystallographic literature, though not by that name, brought to the attention of the authors is by Larson (1980).

$$D_{\text{restraints}} = \frac{\partial T_{\text{c}}}{\partial y} = \left(\frac{\partial T_{\text{c},i}}{\partial y_j}\right)_{i,j}, \tag{20}$$

which is known as the design matrix for the restraints, is very sparse. We therefore use sparse-matrix techniques to efficiently store and perform computations with D, by only storing non-zero elements, and never performing any multiplication that involves an element of D known to be zero. We then introduce the matrix of derivatives with respect to the refined parameters,

$$\tilde{D}_{\text{restraints}} = \frac{\partial T_{\text{c}}}{\partial x} = D_{\text{restraints}} \frac{\partial y}{\partial x}, \tag{21}$$

which is computed by forming the product of D and the constraint matrix. Thus the restraints are initially built up without any knowledge of the constraint matrix. This greatly simplifies their implementation and it simplifies their use in a refinement program that does not use constraints. This organization of the computation does not incur any inefficiency as the product (21) is a cheap operation since both matrices are sparse, and it therefore scales as the number of non-zero elements, which is typically much smaller than the number of parameters n.

The two terms in the normal equations (2) coming from the restraints then read as the matrix product and matrix-vector product,

$$B_{\text{restraints}} = \tilde{D}^{T}_{\text{restraints}} W \tilde{D}_{\text{restraints}} \tag{22}$$

$$g_{\text{restraints}} = \tilde{D}^{T}_{\text{restraints}} \Delta T, \tag{23}$$

where D^T denotes the transpose of D and where W is the diagonal matrix featuring the restraint weights, and where

$$\Delta T_i = T_{\text{o},i} - T_{\text{c},i} \tag{24}$$

is the residual for the ith restraint. We again take advantage of the sparsity of $\tilde{D}_{\text{restraints}}$ to efficiently implement those products.

It would be desirable to place the weights of the restraints on the same scale as the typical residual, such that a restraint will have a similar strength for the same weight in different structures. Rollett (1970) suggests the normalization factor

$$w_{\text{restraints}} = \frac{1}{m-n} \sum_h w_h (Y_{\text{o},h} - K Y_{\text{c},h})^2. \tag{25}$$

This is better known as the square of the *goodness of fit*, χ^2. This normalizing factor also allows the restraints to have greater influence when the fit of the model to the data is poor (and the goodness of fit is greater than unity), whilst their influence lessens as the fit improves (Sheldrick, 1997).

4.1. Implementation

The choice of the minimization algorithm has a significant impact on the organization of the computation of restraints. Indeed, a generic minimizer such as the LBFGS minimizer used in *phenix.refine* requires at each iteration only the function value L and the derivatives $\partial L/\partial x$. The derivatives $\partial L_{\text{data}}/\partial x$ and $\partial L_{\text{restraints}}/\partial x$ can be calculated separately before combining their sum to obtain $\partial L/\partial x$. In contrast, for full-matrix least squares we need the matrices of partial derivatives $\partial F_{\text{c}}/\partial x$ and $\partial T_{\text{c}}/\partial x$. Therefore, depending on the optimization method used, we must be able to compute both $\partial L/\partial x$ and $\partial T_{\text{c}}/\partial x$ where by the chain rule

$$\begin{aligned}\frac{\partial L_{\text{restraints}}}{\partial x} &= \frac{\partial L}{\partial T_{\text{c}}} \frac{\partial T_{\text{c}}}{\partial x} \\ &= 2w(T_{\text{o}} - T_{\text{c}}) \frac{\partial T_{\text{c}}}{\partial x}.\end{aligned} \tag{26}$$

The restraints framework was designed in such a way that it could be easily extended by adding further restraints. Each restraint must provide the array of partial derivatives of the restraint with respect to the crystallographic parameters (one row of the matrix $\partial T_{\text{c}}/\partial y$), the restraint delta, $T_{\text{o}} - T_{\text{c}}$, and the weight, w, of the restraint.

5. Twinning

Like all other refinement programs, we have adopted the model of twins proposed by Jameson (1982) and Pratt *et al.* (1971). The sample is modelled as d domains, each sizeable enough a single crystal to give rise to observable Bragg peaks. If peaks from different domains fall very close in reciprocal space, the integration software analysing frames will be able to compute only the sum of the intensities of these superposed peaks. Data for the rth reflection will therefore consist of an intensity $F^2_{\text{o},r}$ and of a list of s_r Miller indices $h_{r,i_1}, h_{r,i_2}, \ldots, h_{r,i_{s_r}}$ where $h_{r,i}$ is the triplet of Miller indices of the Bragg peak originating from diffraction by domain i. The model of the structure then predicts an intensity $I_{\text{c},r}$ for $F^2_{\text{o},r}$ that reads

$$I_{\text{c},r} = \sum_{l=1}^{s_r} \alpha_{i_l} |F_{\text{c}}(h_{r,i_l})|^2, \tag{27}$$

where α_i is the fraction of the sample volume occupied by twin domain i. The least squares to minimize are then, adapting equation (1) for a twinned structure,

$$L = \sum_{r=1}^{m} w_r \left(F^2_{\text{o},r} - K I_{\text{c},r}\right)^2, \tag{28}$$

where the weight w_r is usually

$$w_r = w[F^2_{\text{o},r}, \sigma(F^2_{\text{o},r}), I_{\text{c},r}], \tag{29}$$

where w would be the same function discussed in the context of equation (1). The minimization of L with respect to the model parameters embodied in F_{c} and with respect to the α's is therefore subject to the constraint

$$1 = \sum_{i=1}^{d} \alpha_i. \tag{30}$$

There is a special case that is common and therefore important, where there are exactly d superposed peaks for each reflection, *i.e.*, $s_r = d$ for every r, and where there is a 3×3 matrix R, the twin law, that generates the Miller indices for each domain $i > 1$ as

$$h_{r,i} = h_{r,i-1} R; \tag{31}$$

in this case, the input consists solely of a list of $F_o(h)^2$ (as for an untwinned refinement but with $h = h_{r,1}$) and of R. Such twins belong to the taxonomies pseudo-merohedral or merohedral. *olex2.refine* is able to refine a twinned structure input in this manner.

SHELXL users would handle the general case by using a reflection file in the HKLF5 format. Such a file is created by the integration software and this approach can deal with the most complex situations. By contrast, *CRYSTALS* always requires a list of twin laws.

olex2.refine can handle a more general case: when general and merohedral twinning are simultaneously present. For example, one may have four domains 1, 2, 3 and 4. Domains 1 and 3 are related by a twin law R, and so are domains 2 and 4, but the relative orientation of domains 1 and 2 does not correspond to any twin law. Thus the measured frames exhibit two lattices of Bragg peaks, with some overlaps, and the integration software will then output a list of (F_o^2, h_1), (F_o^2, h_2) and (F_o^2, h_1, h_2). The refinement engine needs to expand this to a list of $(F_o^2, h_1, h_3 = h_1R)$, $(F_o^2, h_2, h_4 = h_2R)$ and $(F_o^2, h_1, h_2, h_3 = h_1R, h_4 = h_2R)$. *olex2.refine* performs this task on the fly, while passing Miller triplets to the code computing structure factors and their derivatives.

The theory and phenomenology of twinning extends far beyond our exposition, but from the narrow point of view of refinement our presentation is sufficient, since, in this paper, we do not concern ourselves with the computation of the twin law or with the indexing of superposed lattices.

6. Standard uncertainties

6.1. Variance matrix

In this section, we will discuss the rigorous computation of s.u.'s of and correlation between the parameters of a constrained model. As pointed out in Appendix B in the comment about equation (73), the variance matrix for the refined parameters x is

$$\mathrm{Var}\,(x) = B^{-1}, \tag{32}$$

where

$$B_{ij} = \frac{\partial r}{\partial x_i} \cdot \frac{\partial r}{\partial x_j} \tag{33}$$

is the normal matrix for the constrained least-squares minimization. However, we are interested in the variance matrix $\mathrm{Var}\,(y)$ for the crystallographic parameters of the model, not in $\mathrm{Var}\,(x)$. For small variations, we have the linear relation

$$\delta y \approx J \delta x \tag{34}$$

and therefore, using the well known heuristic definition of the variance matrix of y as the mean value of $\delta y \delta y^T$ in the linear approximation around the minimum implicitly assumed throughout crystallographic refinement,

$$\mathrm{Var}\,(y) = J\,\mathrm{Var}\,(x)\,J^T. \tag{35}$$

We would like to stress an important consequence of this formula: constrained parameters generally have non-zero s.u.'s. This is the case, for example, for the coordinates of riding atoms. For most constraints, the s.u.'s of hydrogen coordinates are equal to the s.u.'s of the atom they ride on but *e.g.* for a rotating $-CH_3$, they differ because the s.u. of the azimuthal angle increases the s.u. of the hydrogen coordinates that come from riding only.

6.2. Derived parameters

We will now discuss the s.u.'s of derived parameters. Such a parameter is a function f of a set of atomic parameters p_i and its variance can be derived using the same heuristic as above in a linear approximation where $\delta f \simeq \sum_i (\partial f/\partial p_i)\delta p_i$ by computing Var (f) as the mean of $(\delta f)^2$. This leads to

$$\sigma^2(f) = \sum_{i,j} \left(\frac{\partial f}{\partial p_i}\right) \left(\frac{\partial f}{\partial p_j}\right) \mathrm{cov}(p_i, p_j). \tag{36}$$

An early occurrence of this formula can be found in Sands (1966).

As a result of this formula and of the consequence of equation (35) stated after it, any bond length, any bond angle and any dihedral angle involving a riding atom has a non-zero s.u. unless that geometrical quantity is fixed by the constraint. For example, in the case of $(R_1, R_2, R_3)-C-H$, for which the constraints do not fix the angles $\widehat{R_iCH}$, those angles have a non-zero s.u. This correct behaviour unfortunately triggers many alerts when a structure refined with *Olex2* is checked with *PLATON*.

PLATON plays a very important role in detecting common flaws in the crystallographic workflow. However, since *PLATON* does not have access either to the covariance matrix or the constraint matrix, it has to make guesses that result in estimation of e.s.d.'s that may be out by up to a factor 2. This is the key problem we encounter with the way *olex2.refine* reports e.s.d.'s. Specifically, most structures refined with *olex2.refine* fail *checkCIF* test PLAT732 (an alert C), for the reason explained in the documentation of this alert (in Note 2):[5] *PLATON* computes the s.u. of the said angle using the s.u. of the atomic coordinates as if they were independent parameters since it does not have access to the variance–covariance matrix that describes their correlations. In this case, those correlations are very significant since the position of H completely depends on the position of $R1$, $R2$ and $R3$. Hence the s.u. of this angle as reported by *Olex2* differs from the *PLATON* estimate. Thus all failures of test PLAT732 for angles or distances involving constrained hydrogen atoms are spurious. As a result, if a referee were to require that all alert C's are to be addressed in a CIF submitted for publication in *Acta Crystallographica*, we advise authors to explain away PLAT732 for riding hydrogen atoms by quoting Note 2 in the documentation for that test. *olex2.refine* will actually automatically prepare the CIF file with such explanations.

6.2.1. Incorporating s.u.'s of unit-cell parameters. Derived parameters such as bond lengths and angles are a function of both the least-squares atomic parameters and the unit-cell

[5] This note seems to be a recent addition and the authors would like to thank Ton Spek for this helpful consideration.

parameters. As such, the s.u. of a derived parameter is likewise a function of both the atomic and unit-cell parameters as well as their respective s.u.'s. If the s.u.'s in atomic parameters are considered to be totally uncorrelated with the s.u.'s in the cell parameters, *i.e.* their covariance is zero, then the s.u. in a derived parameter can be considered as comprising two independent sources of uncertainties:

$$\sigma^2(f) = \sigma^2_{\rm cell}(f) + \sigma^2_{xyz}(f), \tag{37}$$

where $\sigma_{xyz}(f)$ is the part coming from the uncertainties in the least-square estimates of the positional parameters, and $\sigma_{\rm cell}(f)$ comes from the uncertainties in the unit-cell parameters,

$$\sigma^2_{\rm cell}(f) = \sum_{i,j} \frac{\partial f}{\partial i}\,\frac{\partial f}{\partial j}\,{\rm cov}\,(i,j), \tag{38}$$

where $i, j = \{a, b, c, \alpha, \beta, \gamma\}$.

This necessitates the calculation of the derivatives of the function with respect to the unit-cell parameters. In order to do so, it is easier to calculate separately the derivatives of the function with respect to the elements of the metrical matrix, and also the derivatives of the metrical matrix with respect to the cell parameters, and then to use the chain rule

$$\frac{\partial f}{\partial i} = \frac{\partial f}{\partial g_{jk}}\,\frac{\partial g_{jk}}{\partial i},\ \ i = a, b, c, \alpha, \beta, \gamma. \tag{39}$$

Indeed $\partial f/\partial g_{jk}$ must be evaluated for every function, whereas $\partial g_{jk}/\partial i$ is constant for a given unit cell.

Now we consider the application of equation (36) to determine the s.u. in the length of the vector u, in fractional coordinates. The length, D, of the vector u is given by

$$D = (u^T G u)^{1/2}, \tag{40}$$

where G is the metrical matrix (see *e.g.* Giacovazzo *et al.*, 2011).

The derivatives of the distance, D, with respect to the elements of the metrical matrix, G, are given by

$$\frac{\partial D}{\partial g_{ii}} = \frac{1}{2}\,\frac{u_i^2}{D} \tag{41}$$

and (given the metrical matrix is symmetric)

$$\frac{\partial D}{\partial g_{ij}} = \frac{u_i u_j}{D},\ \ \text{for all } i < j. \tag{42}$$

Similarly, for the angle between two vectors in fractional coordinates, u and v, where the angle is defined as

$$\theta = \arccos \frac{u^T G v}{||u^T G u||\ ||v^T G v||} \tag{43}$$

or

$$\theta = \arccos \frac{r_A \cdot r_B}{||r_A||\ ||r_B||}, \tag{44}$$

where r_A and r_B are the Cartesian equivalents of u and v, the derivative of the angle, θ, with respect to the elements of the metrical matrix, G, is given by

$$\frac{\partial \theta}{\partial g_{ii}} = \frac{1}{2\sin\theta}\left(\frac{u_i^2\cos\theta}{||r_A||^2} - \frac{2u_i v_i}{||r_A||\ ||r_B||} + \frac{v_i^2\cos\theta}{||r_B||^2}\right) \tag{45}$$

and

$$\frac{\partial \theta}{\partial g_{ij}} = \frac{1}{2\sin\theta}\left(\frac{u_i u_j\cos\theta}{||r_A||^2} - \frac{u_i v_j + u_j v_i}{||r_A||\ ||r_B||} + \frac{v_i v_j\cos\theta}{||r_B||^2}\right),$$
$$\text{for all } i < j. \tag{46}$$

The derivatives of the metrical matrix with respect to the unit-cell parameters,

$$C = (a, b, c, \alpha, \beta, \gamma), \tag{47}$$

needed in order to apply equation (39) are given below:

$$\begin{aligned}
\frac{\partial g_{11}}{\partial C} &= (2a, 0, 0, 0, 0, 0)\\
\frac{\partial g_{22}}{\partial C} &= (0, 2b, 0, 0, 0, 0)\\
\frac{\partial g_{33}}{\partial C} &= (0, 0, 2c, 0, 0, 0)\\
\frac{\partial g_{12}}{\partial C} &= (b\cos\gamma, a\cos\gamma, 0, 0, 0, -ab\sin\gamma)\\
\frac{\partial g_{13}}{\partial C} &= (c\cos\beta, 0, a\cos\beta, 0, -ac\sin\beta, 0)\\
\frac{\partial g_{23}}{\partial C} &= (0, c\cos\alpha, b\cos\alpha, -ac\sin\beta, 0, 0).
\end{aligned} \tag{48}$$

6.3. Symmetry

The variance–covariance matrix that is obtained from the inversion of the least-squares normal matrix contains the variance and covariance of all the refined parameters. Frequently, it is necessary to compute functions that involve parameters that are related by some symmetry operator of the space group to the original parameters. Sands (1966) suggests that the symmetry should be applied to the variance–covariance matrix to obtain a new variance–covariance matrix for the symmetry-generated atoms. Alternatively, and it is this method that is used here, the original variance–covariance matrix can be used if the derivatives in equation (36) are mapped back to the original parameters.

Let the function f depend on the Cartesian site $y_{\rm c}$ that is generated by the symmetry operator $R_{\rm c}$ from the original Cartesian site $x_{\rm c}$, *i.e.*

$$y_{\rm c} = R_{\rm c} x_{\rm c}. \tag{49}$$

Then the gradient with respect to the original site can be obtained by

$$\frac{\partial f(y_{\rm c})}{\partial x_{\rm c}} = \frac{\partial f(y_{\rm c})}{\partial y_{\rm c}}\,R_{\rm c}. \tag{50}$$

The variance–covariance matrix that is used in this case should be the one that is transformed to Cartesian coordinates. The variance–covariance matrix for Cartesian coordinates can be obtained from that for fractional coordinates by the transformation

$$V_c = \mathcal{O} V_f \mathcal{O}^T, \tag{51}$$

where the transformation matrix $\mathcal{O}$ needed to transform the entire variance–covariance matrix in one operation would be block diagonal, with the 3×3 orthogonalization matrix O repeated at the appropriate positions along the diagonal. This transformation can be computed efficiently using sparse-matrix techniques.

7. Refinement results

olex2.refine uses CIF (Hall *et al.*, 1991) as its main output. The CIF contains information regarding the space group, the data indicators such as merging indices, the refinement indicators such as the *R* factors, goodness of fit, residual electron density, refinement convergence indicators and tabulated structure information, including tables of atomic parameters, bonds and angles. *olex2.refine* produces a table describing the restraints using the CIF restraints dictionary. Moreover, the *olex2.refine* CIF always contains a verbal description of the refinement model – hydrogen-atom treatment, constraints, restraints and their targets. Optionally, the refinement model file (*SHELX* RES file) and the reflections can also be included in the final CIF for deposition. It should be noted that the constraints and restraints unique to *olex2.refine, i.e.* not featured by *SHELX*, are saved in REM sections (using an XML format). This has the advantage that they can be read back by *Olex2* while providing a '.res file' that can also be refined with *SHELX*, albeit with potentially a different model. As part of this work a CIF-handling toolbox (Gildea *et al.*, 2011) was added to *cctbx*.

8. Conclusion

We have presented herein the full mathematical derivations of the concepts used within *olex2.refine*. This new refinement engine is feature-wise on a par with the established software in the field such as *SHELXL* and *CRYSTALS*. It actually provides a richer wealth of constraints than those classic suites. *olex2.refine* is immediately useful to the practising crystallographer since it is presently available from the program *Olex2* by default. It can also be used independently as a library of components to write short scripts as well as more complex programs, dealing with any aspect of small-molecule refinement. Indeed, *olex2.refine* is largely based on the Small-Molecule Toolbox (*smtbx*) that is part of the Computational Crystallography Toolbox (*cctbx*), which is usable as a library for the Python programming language, thus providing great expressiveness, conciseness and ease of coding combined with an immense wealth of tools covering all of crystallography. We have explained herein how the *smtbx* added constrained least-squares refinement from scratch to the *cctbx* and how it added many restraints as well, thus opening new fields to the *cctbx*.

APPENDIX *A*
Computation of structure factors and their gradient

The formulae discussed in this appendix have been known for nearly a century and have been implemented in all known refinement programs. However, it seems to the authors that, during the last decades, the computation for a given Miller index h of $F_c(h)$ and its gradient $\nabla F_c(h)$ with respect to crystallographic parameters has very rarely been presented in all the minute details necessary to implement those formulae in a program, justifying this appendix in our humble opinion. We will compare our algorithm to Rollett (1965) point by point as we present it, but first we would like to highlight a technical difference. Rollett computes the real and imaginary parts separately, whereas our computation is performed with complex numbers, the rationale being that the composition of the different terms of the structure factor can be performed with mere complex multiplications. For example, the incorporation of anomalous scattering in equation (53) corresponds to equations (20) and (21) in Rollett.

A1. Structure factor of one atom

Since the structure factor of the entire unit cell is the sum over the contribution of each scatterer, we will focus on one such contribution only. We therefore consider a scatterer with fractional coordinates x, a thermal displacement tensor U in fractional coordinates and a chemical occupancy s (*i.e.* this occupancy does not take crystallographic symmetries into account). Its contribution $F_{uc}(h)$ to the entire unit cell is obtained from its structure factor $F(h)$ by the sum

$$F_{uc}(h) = \sum_{(R|t)\in\mathcal{S}} F^{(R|t)}(h). \tag{52}$$

$\mathcal{S}$ denotes the subset of symmetry operators $(R|t)$ of the rotational part R and translational part t that generate the orbit of the position x under the application of the entire set of symmetries in the space group of the structure, whereas $F^{(R|t)}(h)$ is the transform of $F(h)$ under the symmetry $(R|t)$. For a spherical atomic model with an elastic scattering factor $f(h^2)$ that does not depend on the direction of h and an inelastic scattering factor $f' + if''$ that does not depend on h (a property that holds for X-ray diffraction), F reads

$$F(h) = s[f(h^2) + f' + if''] \underbrace{\exp(-hUh^T)\exp(i2\pi hx)}_{G(h)}. \tag{53}$$

We will consistently denote a triplet h of Miller indices as a row vector, whose transpose h^T is therefore a column vector, and a triplet x of fractional coordinates as a column vector. In this context, h^2 denotes the Euclidean square of h, that therefore involves the reciprocal metric matrix M^* as $h^2 = hM^*h^T$.

If the thermal displacement is isotropic, the term $\exp(-hUh^T)$ is replaced by $\exp(-2\pi^2 h^2 u_{iso})$ and it is then factored out of $G(h)$.

We are therefore left with computing the sum of the transforms of $G(h)$ under the symmetries in $\mathcal{S}$, the other terms

forming a factor multiplying this sum. By the definition of a Fourier transform, that transform reads

$$G^{(R|t)}(h) = G(hR)\exp(i2\pi ht). \tag{54}$$

Let us consider the case of non-primitive unit cells first. The sum can be partitioned as follows:

$$\sum_{(R|t)\in\mathcal{S}} G^{(R|t)}(h) = \sum_{(R|t)\in\mathcal{S}'} \sum_{\tau\in\mathcal{T}} G^{(R|t+\tau)}(h),$$

where $\mathcal{T}$ is the set of all centring translations, including zero, and $\mathcal{S}'$ is the set of 'primitive' symmetries. Then with equation (54) we can factorize the sum as

$$\sum_{(R|t)\in\mathcal{S}} G^{(R|t)}(h) = \left[\sum_{\tau\in\mathcal{T}} \exp(i2\pi h\tau)\right]\left[\sum_{(R|t)\in\mathcal{S}'} G(hR)\exp(i2\pi ht)\right]. \tag{55}$$

The first term of the right-hand side does not depend upon the scatterer and it can therefore be precomputed. Commonly, Miller indices satisfy the centring conditions $h\tau = 0 \bmod 1$. In this case, this term is equal to the number of centring translations. However, this assumption can be invalid in the refinement of some pseudo-merohedral twins since Miller indices satisfying a centring condition may be transformed by the twin law in Miller indices that do not satisfy it.

It should be noted that Rollett already advocated skipping the computation of the terms in $\sum_{(R|t)\in\mathcal{S}} G^{(R|t)}(h)$ corresponding to non-zero centring translations and then multiplying the result by 2. However, our framework can handle any group $\mathcal{T}$, not only those with $+1/2$ translations. It would work for tripled cells as well, for example.

Thus we are left with the computation of the second term of the right-hand side of equation (55). This is equivalent to treating the case of a primitive unit cell. Three cases are to be considered.

(i) The space group is non-centric. Then there is no further simplification.

(ii) The space group is centric but the inversion $(\bar{1}|t_{\bar{1}})$ may not be located at the origin. Then the sum over the symmetries may be split into the sum over the set $\mathcal{S}^+$ of proper symmetries and the sum over the set of improper symmetries. Since every improper symmetry may be written as $(\bar{1}|t_{\bar{1}})(R|t)$ where $(R|t)$ is a proper symmetry, we have

$$\sum_{(R|t)\in\mathcal{S}} G^{(R|t)} = \underbrace{\sum_{(R|t)\in\mathcal{O}^+} G^{(R|t)}}_{\Sigma} + \sum_{(R|t)\in\mathcal{S}^+} G^{(\bar{1}|t_{\bar{1}})(R|t)}.$$

However, the product involving the inversion simplifies as $(\bar{1}|t_{\bar{1}})(R|t) = (-R| - t + t_{\bar{1}})$ and therefore using equation (54)

$$G^{(\bar{1}|t_{\bar{1}})(R|t)} = G(-hR)\exp(-i2\pi ht)\exp(i2\pi ht_{\bar{1}}),$$

but then from the very definition of $G(h)$ in equation (54)

$$G(-hR) = G(hR)^*, \tag{56}$$

and therefore

$$\sum_{(R|t)\in\mathcal{S}} G^{(R|t)} = \Sigma + \Sigma^*\exp(i2\pi ht_{\bar{1}}). \tag{57}$$

(iii) If $ht_{\bar{1}} = 0$, which holds true for every reflection if the space group is origin centric ($t_{\bar{1}} = 0$), the previous case resolves to the real part of Σ only,

$$\sum_{(R|t)\in\mathcal{S}} G^{(R|t)} = 2\sum_{(R|t)\in\mathcal{S}^+} \exp[-hRU(hR)^T]\cos(hRx + t). \tag{58}$$

The total sum over symmetries is therefore real, and is its derivative, the former involving a cosine and the latter involving a sine for the derivatives with respect to x.

In order to avoid redundant computations, our code distinguishes these three cases. One piece of code handles the case of centrosymmetric space groups using only real numbers to compute Σ (an equivalent optimization was presented in Rollett). Another piece of code handles the other two cases, first computing Σ using complex number algebra and then using equation (57) if the space group is centric and $ht_{\bar{1}} \neq 0$ (an optimization not found in Rollett). Another optimization we applied is to precompute the terms hR and $\exp(i2\pi t)$ appearing in equation (54) as well as the test for the condition $ht_{\bar{1}} = 0$ before the loop over all scatterers in the asymmetric unit (an optimization already advocated by Rollett). In all three cases, the computation of $\cos(hRx + t)$ and $\sin(hRx + t)$ is the most costly operation. This is mitigated by computing the structure factor and its derivatives together, as the cosine and sine then only need to be computed once for each reflection and scatterer. This optimization is the reason why we have written our own new code into the *smtbx* instead of reusing the original code in the *cctbx*. Indeed, in the latter, structure factors are computed separately from the derivatives, resulting in two separate loops over the reflections and the scatterers, and in sines and cosines being computed twice. This is rather well suited to the optimization algorithm used in *phenix.refine* (LBFGS) because it does not need the derivatives at every step. But it does not fit well with full-matrix least squares as is prevalent in small-molecule crystallography.

Our code also provides two options to compute the sines and cosines: either by using the trigonometric functions of the standard C++ library, or by using a *cctbx* function that interpolates between tabulated values of sines and cosines. The latter is much faster but less precise. *Olex2* uses the former as *CRYSTALS* and *SHELXL* use the standard FORTRAN sin and cos functions. It should be noted that Rollett used an approximation of trigonometric functions by Chebyshev polynomials, which used to be employed by *CRYSTALS*.

A2. Derivatives of $|F|^2$ and $|F|$

Having computed the unit-cell structure factor F_c and its derivatives, we need to compute the derivatives of $|F_c|^2$ and perhaps as well $|F_c| = (|F_c|^2)^{1/2}$ if performing a refinement against F. Since $|F_c|^2 = F_cF_c^*$, where z^* denotes the complex conjugate of the complex number z, for any crystallographic parameter ξ,

$$\frac{\partial|F_c|^2}{\partial\xi} = F_c^* \frac{\partial F_c}{\partial\xi} + F_c \frac{\partial F_c^*}{\partial\xi},$$

but since the parameter ξ is always real in practice,

$$\frac{\partial F_c^*}{\partial\xi} = \left(\frac{\partial F_c}{\partial\xi}\right)^* \tag{59}$$

and therefore

$$\frac{\partial|F_c|^2}{\partial\xi} = 2\mathrm{Re}\left(F_c^* \frac{\partial F_c}{\partial\xi}\right) \tag{60}$$

where Re denotes the real part. Then,

$$\frac{\partial|F_c|}{\partial\xi} = \frac{1}{|F_c|}\,\mathrm{Re}\left(F_c^* \frac{\partial F_c}{\partial\xi}\right). \tag{61}$$

The *smtbx* provides code to compute derivatives for both of these observables. However, *Olex2* currently only offers the choice to refine against F^2.

A3. Extinction correction

olex2.refine models primary and secondary extinction with the same empirical correction used in *SHELXL*,

$$F_c' = F_c\left(1 + 0.001x\,\frac{F_c^2\lambda^3}{\sin 2\theta}\right)^{-1/4}, \tag{62}$$

where the so-called extinction parameter x is added to the set of refined parameters. As noted in *SHELXL* documentation, it is close to the work of Becker & Coppens (1974) but not identical.

APPENDIX *B*
Minimization of least squares with an overall scale factor

In this appendix, we are concerned with the minimization of $L_{\mathrm{data}}(x, K)$ as defined in equation (1). We will drop the label 'data' throughout this appendix for clarity. We will denote by (x^*, K^*) the values of those parameters at which $L(x, K)$ reaches the minimum we are interested in.

It is well known that the overall scale factor tends to be highly correlated with the thermal displacements. As a result, a starting value of K far from K^* tends to destabilize the fit and at the very least will increase the number of iterations necessary to converge to the minimum. It is, however, easy to compute a reasonable approximation of K^*. Indeed, as a function of K only, keeping all the other parameters x fixed, $L(x, K)$ is a second-order polynomial. An analytical formula therefore exists for the value of K that minimizes that polynomial. Using the geometrical interpretation of least squares is the fastest manner to demonstrate this formula and leads to the most compact formula. We therefore introduce the scalar product of two sets of observables,

$$Y \cdot Y' = \sum_h w(h) Y(h) Y'(h), \tag{63}$$

and the associated norm $||Y||$,

$$||Y||^2 = \sum_h w(h) Y(h)^2, \tag{64}$$

as well as the residual vector

$$r(x, K) = Y_o - K Y_c(x). \tag{65}$$

With those notations, $L(x, K)$, which reads

$$L(x, K) = ||r(x, K)||^2, \tag{66}$$

reaches a minimum at

$$\tilde{K} = \frac{Y_c \cdot Y_o}{||Y_c||^2}, \tag{67}$$

while keeping x fixed, and the value at the minimum reads

$$L(x, \tilde{K}) = ||Y_o||^2 - \tilde{K}^2 ||Y_c(x)||^2. \tag{68}$$

We could then simply use $\tilde{K}$ as a starting value for a combined refinement of x and K, *i.e.* the minimization of $L(x, K)$ with the independent vector (x, K) of parameters.

However, we chose to minimize $L(x, \tilde{K})$ instead, which is a function of x only since $\tilde{K}$ is completely determined by x. This is a special case of a technique called separable least squares (Nielsen, 2000, and references therein) that converges to the same minimum as previously, but with fewer iterations for the same cost per iteration. In order to carry out that minimization, we first need to compute the first-order derivatives,

$$\frac{\partial L(x, \tilde{K})}{\partial x_j} = \frac{\partial L}{\partial x_j}(x, \tilde{K}),\ 1 \le j \le n, \tag{69}$$

where the chain rule would normally mandate a second term $(\partial L/\partial K)\,(x, \tilde{K})(\partial\tilde{K}/\partial x_j)$, but in this case it is zero because of the definition of $\tilde{K}$. In this expression,

$$\frac{\partial L}{\partial x_j} = 2r \cdot \frac{\partial r}{\partial x_j}. \tag{70}$$

Thus, the first-order derivative is exactly the same as it would have been with an independent parameter K. We then need the second-order derivatives

$$\frac{\partial^2 L(x, \tilde{K})}{\partial x_i \partial x_j} \approx \frac{\partial r(x, \tilde{K})}{\partial x_i} \cdot \frac{\partial r(x, \tilde{K})}{\partial x_j},\ 1 \le i, j \le n, \tag{71}$$

where we have neglected the term $[\partial^2 r(x, \tilde{K})/\partial x_i \partial x_j] \cdot r(x, \tilde{K})$ because, for a well behaved fit, the residual r and its curvature are small enough that this term can safely be neglected.

Let us now build the normal equations,

$$Bs = -g, \tag{72}$$

for the shift, s, of the parameter vector x. The matrix B is known as the normal matrix. Those equations are simply the Newton equations, featuring the Hessian matrix, $B_{ij} = \partial^2 L(x, \tilde{K})/\partial x_i \partial x_j$, computed using the approximation of equation (71) and the gradient g of $L(x, \tilde{K})$. It should be noted that the solution, s, of equation (72) is then also the solution of the linear least-squares problem

$$\min_s \left|\left| r(x, \tilde{K}) + \frac{\partial r(x, \tilde{K})}{\partial x} s \right|\right|^2. \tag{73}$$

The inverse of the normal matrix B when the least-squares minimum has been reached by $L(x, \tilde{K})$ can therefore be viewed as the variance matrix of x, which should be contrasted to the more classic combined refinement of x and K, for which the variance matrix is the sub-matrix of the inverse of the normal matrix obtained by removing the column and the row corresponding to K.

Using the definition (65) of r and the expression (67) of $\tilde{K}$, we have

$$\frac{\partial r(x,\tilde{K})}{\partial x_i} = -\left(\tilde{K}\frac{\partial Y_c}{\partial x_i} + \frac{\partial \tilde{K}}{\partial x_i} Y_c\right), \tag{74}$$

$$\frac{\partial \tilde{K}}{\partial x_i} = \frac{1}{||Y_c||^2}\frac{\partial Y_c}{\partial x_i}\cdot\left(Y_o - 2\tilde{K}Y_c\right) \tag{75}$$

and therefore the normal matrix B and the right-hand side of the normal equations g read

$$\begin{aligned} B_{ij} &= \tilde{K}^2\frac{\partial Y_c}{\partial x_i}\cdot\frac{\partial Y_c}{\partial x_j} + \tilde{K}\left(\frac{\partial \tilde{K}}{\partial x_j}Y_c\cdot\frac{\partial Y_c}{\partial x_i} + \frac{\partial \tilde{K}}{\partial x_i}Y_c\cdot\frac{\partial Y_c}{\partial x_j}\right) \\ &\quad + \frac{\partial \tilde{K}}{\partial x_i}\frac{\partial \tilde{K}}{\partial x_j}||Y_c||, \\ g_i &= -(Y_o - \tilde{K}Y_c)\cdot\left(\tilde{K}\frac{\partial Y_c}{\partial x_i} + \frac{\partial \tilde{K}}{\partial x_i}Y_c\right). \end{aligned} \tag{76}$$

It should be noted that the first term of B would appear in exactly the same form if we had kept the scale factor K as an independent parameter. Moreover, terms very similar to the second and third terms of B would need to be computed in that case, as the normal matrix B would then have one more column and one more row which would feature those similar terms. Thus, the computation cost of our unorthodox method is the same as that of the more traditional refinement of the overall scale factor along with the other crystallographic parameters. In any case, for all methods based on the normal equations, the computation time is dominated by the first term of B, as it scales as $O(mn^2)$, where m is the number of reflections and n the number of parameters.

It should also be noted that we never construct and store the whole of the so-called design matrix $[\partial Y_c(h)/\partial x_i]_{h,i}$ that appears in those equations. Instead, we compute the vector of derivatives $[\partial Y_c(h)/\partial x_i]_i$ for a few reflections h and then we immediately accumulate them into the normal matrix B and into the right-hand side g, as opposed to constructing the whole design matrix and then forming the products $\partial Y_c/\partial x_i \cdot \partial Y_c/\partial x_j$ appearing in equation (76), for example. It used to be the only efficient way to proceed a few decades ago when least-squares refinement was being developed, but the fantastic increase of computer memory has made that point largely irrelevant.

The traditional argument is to show that the normal matrix is typically one to two orders of magnitude smaller than the design matrix. Indeed, the latter has mn elements, whereas the former has approximately $n^2/2$ elements for large values of n. Since in a typical small-molecule crystal structure determination the data-to-parameter ratio m/n is typically in the range 10–30, the normal matrix is therefore 20 to 60 times smaller than the design matrix. With the common use of constraints, particularly with respect to those on the parameters of hydrogen atoms, this contrast gets even more striking as n shall then be replaced with the number of parameters actually refined, which can be as small as $n/2$ for typical organic structures. This results in another factor 2 in the comparison of the respective sizes of the design and of the normal matrix, and it makes the latter typically 40 to 120 times smaller than the former.

However, in the extreme case of a structure with about 1000 atoms in the asymmetric unit, the design matrix stored in double precision only requires 40 to 120 Mb of memory, assuming constraints reducing the number of refinable parameters by a factor 2, as opposed to around 2 Mb for the normal matrix. They would therefore both easily fit in the main memory of a modern PC which comes with many Gigabytes of RAM.

The counter-argument goes even further, as the normal matrix is actually not necessary either to solve the least-square problem or to compute estimated standard deviations for geometric quantities, which are necessary to publish a structure. Indeed, there are well known mathematical techniques based only on the design matrix to solve these two problems [*cf.* for example, Björck (1996), sections 2.8.3 for s.u. computations and 2.4.3 for the solution of the least squares].

However, the best of those design-matrix techniques, the QR decomposition, involves about twice as many floating-point operations as the method based on the normal equations. That is the actual motivation for our choosing the latter and therefore adhering to a long crystallographic tradition.

It should be noted that for singular or nearly singular problems, one could solve the LS problem by using singular-value filtering on the design matrix, which is the most robust technique. On the other hand, if one computes the normal matrix, therefore squaring the singularities, solving the LS problem with eigenvalue filtering is of little practical interest.

APPENDIX *C*
Constraints

olex2.refine provides constraints to influence the position(s) or the ADP('s) of an atom or a group of atoms, as well as their occupancy factors. This section will enumerate the constraints and provide the details of their implementation.

C1. Occupancy constraints

The *smtbx* provides a generic tool to express any scalar parameter v as an affine function of other scalar parameters $u_1, u_2, \ldots, u_s$,

$$v = \sum_{i=1}^{s} a_i u_i + b, \tag{77}$$

where the number of arguments s as well as the coefficients a_i and b can be freely chosen. The common case of two atoms

whose occupancies o and o' shall add up to 1 is easily handled by the case $s = 1$, as $o' = 1 - o$.

Higher values of s are useful to model a site partially occupied by different types of atoms when the requirement of full occupancy is complemented by other constraints. The example used in the *SHELXL* manual to illustrate the restraint command SUMP is such a case: a site is occupied by ions Na^+, Ca^{2+}, Al^{3+} and K^+ with an average charge of +2, leading to the restrictions on their occupancies:

$$o_{Na^+} + o_{Ca^{2+}} + o_{Al^{3+}} + o_{K+} = 1, \tag{78}$$

$$o_{Na^+} + 2o_{Ca^{2+}} + 3o_{Al^{3+}} + o_{K+} = +2. \tag{79}$$

Instead of enforcing those relations as restraints, as advocated in the *SHELX* documentation, we can solve those equations to get the explicit dependencies

$$o_{Na^+} = o_{Al^{3+}} - o_{K+}, \tag{80}$$

$$o_{Ca^{2+}} = 1 - 2o_{Al^{3+}}, \tag{81}$$

which can then readily be expressed as special cases of equation (77).

C2. Shared site and shared *U* constraint

Two scatterers share the same site or the same ADP tensor. These constraints are pretty straightforward to implement within the *smtbx* framework. Values of dependent parameters and the corresponding rows of the Jacobian are set equal to the values for the reference atom.

C3. Riding atoms/group *U* constraint

This constraint defined a group as riding on the given, pivot atom, *i.e.* the coordinates of the groups are given the same shifts as the pivot atom (*SHELXL* AFIX 3). Moreover the distances from the atom to the atoms of the group can be refined (*SHELXL* AFIX 4), which is applicable to groups like XY_n for $1 \leq n \leq 4$ where X is B, C, N *etc.* and Y is H, F, Cl *etc.* In general the transformation of the triplet of coordinates r is written down like this:

$$r' = s(r - r_p) + r_p, \tag{82}$$

where s is the distance scaling component ($s = 1$ if not refined) and r_p is the pivot atom centre. In the case when s is refined, the corresponding derivative of the transformation is

$$\frac{\partial r'}{\partial s} = r - r_p, \tag{83}$$

whereas

$$\frac{\partial r'}{\partial r} = s\mathbf{1}. \tag{84}$$

C4. Rotated *U* constraint

This constraint relates the ADPs of one atom to the ADPs of another atom as rotated by a fixed or refined angle around a given direction. The direction can be defined as a static vector, best line or best plane normal. Considering atoms A and B, the relation between their ADP tensor U_A and U_B reads

$$U_B = RU_AR^T, \tag{85}$$

where R is the rotation matrix

$$R = \begin{pmatrix} txx + c_\alpha & tyx + zs_\alpha & tzx - ys_\alpha \\ txy - zs_\alpha & tyy + c_\alpha & tzy + xs_\alpha \\ txz + ys_\alpha & tyz - xs_\alpha & tzz + c_\alpha \end{pmatrix} \tag{86}$$

with $c_\alpha = \cos\alpha$, $s_\alpha = \sin\alpha$ and $t = 1 - c_\alpha$, and where the rotation direction is represented by a normal (x, y, z) and where α is the rotation angle around this direction. This transform is trivially rewritten as a linear transform of $(U_{A,11}, U_{A,22}, U_{A,33}, U_{A,12}, U_{A,13}, U_{A,23})$ into $(U_{B,11}, U_{B,22}, U_{B,33}, U_{B,12}, U_{B,13}, U_{B,23})$, whose matrix is therefore the Jacobian needed for the chain rule [equation (19)]. If the rotation angle is refined, then the derivative of this transformation by the angle is

$$\frac{\partial U_B}{\partial \alpha} = RUR_\alpha'^T + R_\alpha' UR^T, \tag{87}$$

where R_α' is the derivative of R by α:

$$R_\alpha' = \begin{pmatrix} xxs_\alpha - s_\alpha & yxs_\alpha + zc_\alpha & zxs_\alpha - yc_\alpha \\ xys_\alpha - zc_\alpha & yys_\alpha - s_\alpha & zys_\alpha + xc_\alpha \\ xzs_\alpha + yc_\alpha & yzs_\alpha - xc_\alpha & zzs_\alpha - s_\alpha \end{pmatrix}. \tag{88}$$

C5. Pivoted rotation of a rigid body

This constraint is suitable for groups which can be rotated around a given direction (*SHELXL* AFIX 7). For groups XY_n with $1 \leq n \leq 4$ and X being B, C, N *etc.* and Y being H, F, Cl *etc.*, the distances from X to Y can also be refined (*SHELXL* AFIX 8). The coordinate transformation is

$$r' = sR(r - r_p) + r_p, \tag{89}$$

and its derivatives are

$$\frac{\partial r'}{\partial \alpha} = sR_\alpha'(r - r_p), \tag{90}$$

$$\frac{\partial r'}{\partial s} = R(r - r_p), \tag{91}$$

where r_p is the centre of the pivot atom, s is the distance scale component ($s = 1$ if not refined), R is the rotation matrix [equation (86)] and R' is the derivative of the rotation matrix by the angle [equation (88)]. If neither the angle nor distances are refined, this constraint reduces to simple riding.

C6. Free rotation of a rigid body

This constraint is the most generic case of the rigid-body refinement (*SHELXL* AFIX 6). In this case there are six refined parameters – the three Euler angles and three positional components; the ADPs are normally refined independently. The coordinate transformation in this case happens similarly to §C5; however a different rotation matrix is used:

$$R = \begin{pmatrix} c_\beta c_\gamma & -c_\beta s_\gamma & s_\beta \\ s_\alpha s_\beta c_\gamma + c_\alpha s_\gamma & -s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & -s_\alpha c_\beta \\ -c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma & c_\alpha s_\beta s_\gamma + s_\alpha c_\gamma & c_\alpha c_\beta \end{pmatrix} \quad (92)$$

and its derivatives by the angles are

$$\frac{\partial R}{\partial \alpha} = \begin{pmatrix} 0 & 0 & 0 \\ c_\alpha s_\beta c_\gamma - s_\alpha s_\gamma & -c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & -c_\alpha c_\beta \\ s_\alpha s_\beta c_\gamma + c_\alpha s_\gamma & -s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & -s_\alpha c_\beta \end{pmatrix} \quad (93)$$

$$\frac{\partial R}{\partial \beta} = \begin{pmatrix} -s_\beta c_\gamma & s_\beta s_\gamma & c_\beta \\ s_\alpha c_\beta c_\gamma & -s_\alpha c_\beta s_\gamma & s_\alpha s_\beta \\ -c_\alpha c_\beta c_\gamma & c_\alpha c_\beta s_\gamma & -c_\alpha s_\beta \end{pmatrix} \quad (94)$$

$$\frac{\partial R}{\partial \gamma} = \begin{pmatrix} -c_\beta s_\gamma & -c_\beta c_\gamma & 0 \\ -s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & -s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma & 0 \\ c_\alpha s_\beta s_\gamma + s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma - s_\alpha s_\gamma & 0 \end{pmatrix}. \quad (95)$$

For the special cases of spherical counter-ions or groups like Cp and Ph the size component of this constraint can also be refined (*SHELXL* AFIX 9).

C7. Non-crystallographic symmetry constraint

This constraint is applicable to structures containing fragments which should be exactly superposable onto each other but which appear at different positions with different orientations. The same equations as for the free rotation of a rigid body described in §C6 are used in this case, but the coordinates of one of the group are refined along with the rotation angles and shift [*i.e.* equation (82) is used where all r's are refined, contrary to the rigid-body case where those coordinates are fixed input, usually taken from a collection of known chemical groups]. This allows more observations to be used in the refinement of the atomic coordinates and ADPs.

APPENDIX *D*
Restraints and their derivatives

Possible restraints on the stereochemistry or geometry of atomic positions include restraints on bond distances, angles and dihedral angles, chiral volume and planarity. These restraints are used extensively in macromolecular crystallography, and hence were already implemented within the *cctbx* as part of the macromolecular refinement program *phenix.refine*. With the exception of the bond-distance restraint, these restraints were not able to accept symmetry-equivalent atoms. Since this is more frequently required in small-molecule crystallography, these restraints have now been extended to allow for symmetry. We have also implemented other restraints commonly used in small-molecule structure refinement, such as a bond similarity restraint, and restraints on anisotropic displacement parameters including restraints based on Hirshfeld's 'rigid-bond' test (Hirshfeld, 1976), similarity restraints and isotropic ADP restraints.

D1. Restraints involving symmetry

Given a restraint, $f(x)$, involving a site x which is outside the asymmetric unit and which is related to the site y within the asymmetric unit by some symmetry transformation $(R|t)$, *i.e.* $x = Ry + t$, the gradient is transformed as

$$\frac{\partial f(x)}{\partial y} = \left.\frac{\partial f}{\partial x}\right|_{x=Ry+t} R. \quad (96)$$

D2. Bond similarity restraint

The distances between two or more atom pairs are restrained to be equal by minimizing the weighted variance of the distances, where the least-squares residual, L, is defined as the population variance biased estimator

$$L = \langle r - \langle r \rangle \rangle^2, \quad (97)$$

where $r = (r_1, \ldots, r_n)$ are the Euclidean distances between the atoms of each pair. The mean is defined as

$$\langle \xi \rangle = \sum_i \omega_i \xi_i, \quad (98)$$

where

$$\omega_i = \frac{w_i}{\sum_j w_j}, \quad (99)$$

and where w_i is therefore the weight for the ith pair.

The computation of the derivatives is easier with the alternate form of this variance,

$$L = \langle r^2 \rangle - \langle r \rangle^2 = \sum_i \omega_i r_i^2 - \left(\sum_i \omega_i r_i \right)^2. \quad (100)$$

The derivative of L with respect to a distance r_i is then

$$\frac{\partial L}{\partial r_i} = 2\omega_i (r_i - \langle r \rangle). \quad (101)$$

Then the derivatives of $r_i = (x_a^2 - x_b^2)^{1/2}$, where x_a and x_b are the Cartesian coordinates of the two atoms in that pair, are given by

$$\frac{\partial r_i}{\partial x_a} = \frac{x_a - x_b}{r_i}, \quad (102)$$

and the same formula with $b \leftrightarrow a$. Therefore, using the chain rule, the derivatives of the residual with respect to x_a are

$$\frac{\partial L}{\partial x_a} = \frac{2\omega_i (r_j - \langle r \rangle)(x_a - x_b)}{r_j}, \quad (103)$$

and the same formula with $b \leftrightarrow a$.

D3. Planarity restraint

In this section, we will discuss a restraint enforcing a group of atom sites to be coplanar. We will first consider the case of four sites, with Cartesian coordinate vectors x_i, x_j, x_k, x_l. They are coplanar if and only if the associated tetrahedron has a zero volume. This volume reads

$$V_{ijkl} = \frac{a \cdot (b \times c)}{6} \quad (104)$$

and any circular permutation of (a, b, c), where a, b, c are any triplet of distinct edges of the tetrahedron. Choices that lead to elegant formulae are

$$\begin{aligned} a &= x_i - x_j, \\ b &= x_j - x_k, \\ c &= x_k - x_l, \end{aligned} \tag{105}$$

and any circular permutations of (i, j, k, l).

Therefore a straightforward restraint enforcing coplanarity is the square of this volume, with a weight w,

$$L_{ijkl} = wV_{ijkl}^2. \tag{106}$$

Its derivative with respect to x_i is then easily obtained from equations (104) and (105):

$$\frac{\partial L_{ijkl}}{\partial x_i} = \frac{wV_{ijkl}}{3}\left[(x_j - x_k) \times (x_k - x_l)\right]^T. \tag{107}$$

The derivatives with respect to x_j, x_k, x_l are then obtained by circular permutations of (i, j, k, l), taking advantage of the behaviour of equations (104) and (105) with respect to permutations stated above.

We will now consider p sites with Cartesian coordinate vectors $x_1, x_2, \ldots, x_p$ that we want to restrain to be coplanar, with $p > 4$. We form the list $\mathcal{T}$ of tetrahedron $(x_i, x_{i+1}, x_{i+2}, x_{i+3})$ where $1 \leq i \leq p-3$. If the volume of the ith and $(i+1)$th tetrahedron are, respectively, zero, then $x_i, x_{i+1}, x_{i+2}, x_{i+3}, x_{i+4}$ are coplanar as the first four and the last four are, respectively, coplanar and those two quadruplets share a common triplet. Thus if all tetrahedra in $\mathcal{T}$ have a zero volume, then all sites are coplanar, which leads to building a restraint as the sum of the restraint for each tetrahedron,

$$L_{\rm flat} = \sum_{i=1}^{p-3} L_{i,i+1,i+2,i+3}. \tag{108}$$

This method is identical to the FLAT restraint in *SHELXL*, perhaps apart from differing implementation details.

The minimum number of degrees of freedom necessary to model p points in a plane is $2p + 3$: $2p$ for the coordinates in the plane, plus two angles for the orientation of the plane, plus one for the distance of the origin to the plane. Since this number of degrees of freedom is exactly the unconstrained and unrestrained number of degrees of freedom $3p$ minus the number of restraints in the sum of equation (108), this restraint $L_{\rm flat}$ is therefore optimal.

D4. Restraints on atomic displacement parameters

We will discuss three types of restraints on anisotropic displacement parameters (Rollett, 1970), similar to restraints implemented in *SHELXL* and *REFMAC* (Murshudov *et al.*, 1999).

D4.1. Rigid-bond restraint. In a 'rigid-bond' restraint the components of the anisotropic displacement parameters of two atoms in the direction of the vector connecting those two atoms are restrained to be equal. This corresponds to Hirshfeld's 'rigid-bond' test (Hirshfeld, 1976) for testing whether anisotropic displacement parameters are physically reasonable (Sheldrick, 1997) and is in general appropriate for bonded and 1,3-separated pairs of atoms and should hold true for most covalently bonded systems.

Since the mean-square displacement of an atom of thermal displacement tensor U along a normalized vector u is $u^T U u$, the restraint residual for two atoms A and B at respective positions x_A and x_B and with respective thermal displacement tensors U_A and U_B reads

$$r = \frac{u^T (U_A - U_B) u}{||u||}, \tag{109}$$

where

$$u = x_A - x_B, \tag{110}$$

and the restraint term is then

$$L(r_A, U_A, r_B, U_B) = wr^2. \tag{111}$$

Those formulae are valid whether using Cartesian or fractional coordinates, providing the adapted formula is used to compute $||u||$, *i.e.*

$$||u|| = \left(\sum_i u_i^2\right)^{1/2} \tag{112}$$

in Cartesian coordinates and

$$||u|| = \left(\sum_i G_{ij} u_i u_j\right)^{1/2} \tag{113}$$

in fractional coordinates, where G is the metric matrix.

The computation of the derivatives of L with respect to the components of U_A and U_B proceeds in two steps. First, since $u^T U u = \sum_i U_{ii} u_i^2 + 2\sum_{i<j} U_{ij} u_i u_j$, where U is either U_A or U_B,

$$\frac{\partial r}{\partial U_{A,ij}} = \begin{cases} \dfrac{u_i^2}{||u||^2} & \text{if } i = j, \\ \dfrac{2u_i u_j}{||u||^2} & \text{otherwise} \end{cases} \tag{114}$$

$$\frac{\partial r}{\partial U_{B,ij}} = \begin{cases} -\dfrac{u_i^2}{||u||^2} & \text{if } i = j, \\ -\dfrac{2u_i u_j}{||u||^2} & \text{otherwise.} \end{cases} \tag{115}$$

The chain rule then gives, where U is either U_A or U_B,

$$\frac{\partial L}{\partial U_{ij}} = 2wr \frac{\partial r}{\partial U_{A,ij}}. \tag{116}$$

The derivatives of $L(r_A, U_A, r_B, U_B)$ with respect to the positions r_A and r_B are ignored.

D4.2. ADP similarity restraint. The anisotropic displacement parameters of two atoms are restrained to have the same U_{ij} components. Since this is only a rough approximation to reality, this restraint should be given a smaller weight in the least-squares minimization than for a rigid-bond restraint and is suitable for use in larger structures with a poor data-to-parameter ratio. Applied correctly, this restraint permits a gradual increase and change in direction of the anisotropic displacement parameters along a side chain. This is equivalent

to a *SHELXL* SIMU restraint (Sheldrick, 1997). The weighted least-squares residual is defined as

$$L = w\sum_{i=1}^{3}\sum_{j=1}^{3}(U_{A,ij} - U_{B,ij})^2, \tag{117}$$

which, denoting $\Delta U = U_A - U_B$ the matrix of deltas, is the trace of $\Delta U \Delta U^T$, making it clear that it is invariant under any rotation R, since it transforms ΔU into $R\Delta U R^T$.

Since U is symmetric, *i.e.* $U_{ij} = U_{ji}$, equation (117) can be rewritten as

$$L = w\left[\sum_i (U_{A,ii} - U_{B,ii})^2 + 2\sum_{i<j}(U_{A,ij} - U_{B,ij})^2\right]. \tag{118}$$

Therefore the gradients of the residual with respect to diagonal elements are then

$$\frac{\partial L}{\partial U_{A,ii}} = 2w(U_{A,ii} - U_{B,ii}), \tag{119}$$

whereas the gradients with respect to the off-diagonal elements are

$$\frac{\partial L}{\partial U_{A,ij}} = 4w(U_{A,ij} - U_{B,ij}), \tag{120}$$

and then the same equation with $B \leftrightarrow A$.

D4.3. Isotropic ADP restraint. Here we minimize the difference between the ADP's U, and the isotropic equivalent, defined in Cartesian coordinates by

$$U_{\rm eq} = U_{\rm iso}\mathbf{1}, \tag{121}$$

and

$$U_{\rm iso} = \frac{1}{3}{\rm Tr}U. \tag{122}$$

Again, this is an approximate restraint and as such should have a comparatively small weight. A common use for this restraint would be for solvent water, where the two restraints discussed previously would be inappropriate (Sheldrick, 1997). As in equation (117), we define a restraint term invariant under rotations, as $L = {\rm Tr}(U - U_{\rm eq})(U - U_{\rm eq})^T$, or explicitly,

$$L = w\left[\sum_i (U_{ii} - U_{{\rm eq},ii})^2 + 2\sum_{i<j}(U_{ij} - U_{{\rm eq},ij})^2\right] \tag{123}$$

which simplifies in Cartesian coordinates into

$$L = w\left[\sum_i (U_{ii} - U_{\rm iso})^2 + 2\sum_{i<j} U_{ij}^2\right]. \tag{124}$$

By mere inspection, we can see that the derivatives of the residual with respect to the off-diagonal elements are in Cartesian coordinates:

$$\frac{\partial L}{\partial U_{ij}} = \begin{cases} 2w(U_{ii} - U_{\rm iso}) & \text{if } i = j, \\ 4wU_{ij} & \text{otherwise.} \end{cases} \tag{125}$$

D4.4. ADP $U_{\rm eq}$ similarity. The ADP of two atoms A and B may be restrained to have the same $U_{\rm eq}$. The restraint term for atom A reads

$$L_A = w\left(U_{A,\rm eq} - \frac{U_{A,\rm eq} + U_{B,\rm eq}}{2}\right)^2 \tag{126}$$

or in Cartesian coordinates

$$L_A = w\left(\sum_{i=1}^{3}\frac{U_{A,ii} - U_{B,ii}}{6}\right)^2. \tag{127}$$

This restraint is therefore invariant under rotation.

The derivatives with respect to the diagonal elements of U_A are then

$$\frac{\partial L_A}{\partial U_{A,ii}} = \frac{2w}{6}\sum_{i=1}^{3}\frac{U_{A,ii} - U_{B,ii}}{6}, \tag{128}$$

and the same formula with $B \leftrightarrow A$.

D4.5. Fixed ADP $U_{\rm eq}$. When the $U_{\rm eq}$ of an atom is restrained to a fixed value, P, the residual for this restraint is

$$L_A = w(U_{\rm eq} - P) = w\left(\frac{1}{3}\sum_{i=1}^{3}U_{ii} - P\right), \tag{129}$$

and the derivatives of the residual with respect to diagonal values of U are

$$\frac{\partial L}{\partial U_{ii}} = \frac{1}{3}. \tag{130}$$

D4.6. ADP volume similarity. The volumes of the thermal ellipsoids of two atoms A and B may be restrained to be equal. The restraint term for atom A reads

$$L_A = w\left(V_A - \frac{V_A + V_B}{2}\right)^2 = w\left(\frac{V_A - V_B}{2}\right)^2. \tag{131}$$

The harmonic anisotropic displacement of an atom is described by a multivariate normal distribution of covariant matrix U (see *e.g.* Trueblood *et al.*, 1996), which is the anisotropic displacement tensor of that atom. Therefore, the probability distribution function is

$$p(x) = \frac{1}{[(2\pi)^3 \det U]^{1/2}}\exp(-\tfrac{1}{2}x^T U^{-1}x).$$

The surface of constant probability p is therefore an ellipsoid of equation

$$x^T A x = 1,$$

where

$$A = \frac{U^{-1}}{-2\log\{p[(2\pi)^3 \det U]^{1/2}\}}.$$

In order to arrive at a simple formula, we fix the probability p such that the denominator is equal to 1, *i.e.*

$$p = \frac{1}{[(2\pi)^3 e \det U]^{1/2}}. \tag{132}$$

The volume of an ellipsoid with such an equation is given by $(4/3)\pi(\det A^{-1})^{1/2}$, *i.e.* with the chosen value of p,

$$V = \frac{4\pi}{3}(\det U)^{1/2}, \tag{133}$$

whose derivatives read

$$\begin{pmatrix} \frac{\partial V}{\partial U_{11}} \\ \frac{\partial V}{\partial U_{22}} \\ \frac{\partial V}{\partial U_{33}} \\ \frac{\partial V}{\partial U_{12}} \\ \frac{\partial V}{\partial U_{13}} \\ \frac{\partial V}{\partial U_{23}} \end{pmatrix} = \frac{4\pi}{6(\det U)^{1/2}} \begin{pmatrix} U_{22}U_{33} - U_{23}^2 \\ U_{11}U_{33} - U_{13}^2 \\ U_{11}U_{22} - U_{12}^2 \\ 2(U_{13}U_{23} - U_{12}U_{33}) \\ 2(U_{12}U_{23} - U_{13}U_{22}) \\ 2(U_{12}U_{13} - U_{23}U_{11}) \end{pmatrix}. \tag{134}$$

We would like to emphasize once more that our choice of ellipsoid whose volume we wish to restrain lacks physical or mathematical motivations. It is only driven by simplicity. It is interesting to contrast the thermal ellipsoid volume restraint with the U_{eq} similarity restraint. Indeed, the former restrains the sum of the eigenvalues of U whereas the latter restrains the product of those eigenvalues. They are therefore complementary.

This work has been funded by the EPSRC grant 'Age Concern: Crystallographic Software for the Future' (EP/C536274/1) from the British Government. We wish to thank David Watkin with whom we shared this grant for his immense contribution to our understanding of crystallographic refinement and for suggesting many improvements and corrections to this paper. We also wish to thank all the contributors to the *cctbx* library which served as a foundation and model for our work, especially its main author Ralf W. Grosse-Kunstleve whose support has been invaluable. We also wish to thank Professor George Sheldrick for many fruitful discussions. Finally, we wish to thank one of the referees for pointing out that we should handle Miller indices that do not satisfy centring conditions, a remark that led us to change this article and our code accordingly.

References

Adams, P. D. *et al.* (2010). *Acta Cryst.* D**66**, 213–221.
Afonine, P. V., Grosse-Kunstleve, R. W., Echols, N., Headd, J. J., Moriarty, N. W., Mustyakimov, M., Terwilliger, T. C., Urzhumtsev, A., Zwart, P. H. & Adams, P. D. (2012). *Acta Cryst.* D**68**, 352–367.
Baerlocher, C., Hepp, A. & Meier, W. M. (1978). *DLS-76, a FORTRAN program for the simulation of crystal structures by geometric refinement.* Institut für Kristallographie und Petrographie, ETH, Zürich, Switzerland.
Becker, P. J. & Coppens, P. (1974). *Acta Cryst.* A**30**, 129–147.
Betteridge, P. W., Carruthers, J. R., Cooper, R. I., Prout, K. & Watkin, D. J. (2003). *J. Appl. Cryst.* **36**, 1487.
Björck, Å. (1996). *Numerical Methods for Least Squares Problems*. Philadelphia: Society for Industrial and Applied Mathematics.
Giacovazzo, C., Luis Monaco, H., Artioli, G., Viterbo, D., Milanesio, M., Gilli, G., Gilli, P., Zanotti, G., Ferraris, G. & Catti, M. (2011). *Fundamentals of Crystallography*, 3rd ed., edited by C. Giacovazzo. Oxford University Press.
Gildea, R. J., Bourhis, L. J., Dolomanov, O. V., Grosse-Kunstleve, R. W., Puschmann, H., Adams, P. D. & Howard, J. A. K. (2011). *J. Appl. Cryst.* **44**, 1259–1263.
Grosse-Kunstleve, R. W. & Adams, P. D. (2003). *Comput. Commun. Newsl.* **1**, 28–38.
Hall, S. R., Allen, F. H. & Brown, I. D. (1991). *Acta Cryst.* A**47**, 655–685.
Hirshfeld, F. L. (1976). *Acta Cryst.* A**32**, 239–244.
Jameson, G. B. (1982). *Acta Cryst.* A**38**, 817–820.
Larson, A. C. (1980). *Computing in Crystallography*, edited by R. Diamond, S. Ramaseshan & K. Venkatesan, pp. 11.01–11.07. Bangalore: Indian Academy of Sciences.
Murshudov, G. N., Skubák, P., Lebedev, A. A., Pannu, N. S., Steiner, R. A., Nicholls, R. A., Winn, M. D., Long, F. & Vagin, A. A. (2011). *Acta Cryst.* D**67**, 355–367.
Murshudov, G. N., Vagin, A. A., Lebedev, A., Wilson, K. S. & Dodson, E. J. (1999). *Acta Cryst.* D**55**, 247–255.
Nielsen, H. B. (2000). *Separable Nonlinear Least Squares.* IMM, Department of Mathematical Modelling, Report No. IMM-Rep-2000-01. Technical University of Denmark, Lyngby, Denmark.
Nocedal, J. (1980). *Math. Comput.* **35**, 773–782.
Pratt, C. S., Coyle, B. A. & Ibers, J. A. (1971). *J. Chem. Soc. A*, pp. 2146–2151.
Rollett, J. S. (1965). Editor. *Computing Methods in Crystallography*. Oxford: Pergamon Press.
Rollett, J. S. (1970). *Crystallographic Computing*, edited by F. R. Ahmed, pp. 167–181. Copenhagen: Munksgaard.
Sands, D. E. (1966). *Acta Cryst.* **21**, 868–872.
Sheldrick, G. M. (1997). *The SHELX-97 Manual*. Department of Structural Chemistry, University of Göttingen, Göttingen, Germany.
Sheldrick, G. M. (2008). *Acta Cryst.* A**64**, 112–122.
Trueblood, K. N., Bürgi, H.-B., Burzlaff, H., Dunitz, J. D., Gramaccioli, C. M., Schulz, H. H., Shmueli, U. & Abrahams, S. C. (1996). *Acta Cryst.* A**52**, 770–781.
Waser, J. (1963). *Acta Cryst.* **16**, 1091–1094.
Watkin, D. (2008). *J. Appl. Cryst.* **41**, 491–522.