Acta Crystallographica Section A
Foundations of Crystallography
ISSN 0108-7673

Received 10 September 2003
Accepted 1 October 2003

Numerically stable algorithms for the computation of reduced unit cells

R. W. Grosse-Kunstleve,* N. K. Sauter and P. D. Adams

Lawrence Berkeley National Laboratory, One Cyclotron Road, Bldg 4R0230, Berkeley, CA 94720-8235, USA. Correspondence e-mail: rwgrosse-kunstleve@lbl.gov

The computation of reduced unit cells is an important building block for a number of crystallographic applications, but unfortunately it is very easy to demonstrate that the conventional implementation of cell reduction algorithms is not numerically stable. A numerically stable implementation of the Niggli-reduction algorithm of Křivý & Gruber [*Acta Cryst.* (1976), A**32**, 297–298] is presented. The stability is achieved by consistently using a tolerance in all floating-point comparisons. The tolerance must be greater than the accumulated rounding errors. A second stable algorithm is also presented, the *minimum reduction*, that does not require using a tolerance. It produces a cell with minimum lengths and all angles acute or obtuse. The algorithm is a simplified and modified version of the Buerger-reduction algorithm of Gruber [*Acta Cryst.* (1973), A**29**, 433–440]. Both algorithms have been enhanced to generate a change-of-basis matrix along with the parameters of the reduced cell.

1. Introduction

A given three-dimensional periodic lattice may be described by an infinite number of certain combinations of non-collinear basis vectors. The ambiguities in the choice of basis vectors have implications for a number of applications, such as auto-indexing of diffraction images or the comparison of crystal structures in a database. Niggli (1928), based on the work of Eisenstein (1851), defines a set of algebraic conditions that lead to a unique choice of basis vectors for a given lattice. These conditions are adopted in *International Tables for Crystallography*, Volume A (Hahn, 1983).

Buerger (1957) introduced an iterative numerical procedure for the computation of reduced cells. Buerger's algorithm does not necessarily result in the unique Niggli-reduced cell and the term Buerger-reduced cell is therefore found in the literature with reference to the conditions encoded in Buerger's procedure. Gruber (1973) gives an alternative algorithm for the computation of Buerger-reduced cells, combined with a thorough mathematical analysis of the system of all Buerger-reduced cells in relation to the 28 unique types of Niggli-reduced cells. Křivý & Gruber (1976) present a modified algorithm that leads directly to the unique Niggli-reduced cell. Zuo *et al.* (1995) give a similar algorithm that aims to minimize the number of iterations.

The definition of Niggli-reduced cells is based on exact algebraic expressions. However, to the best of our knowledge, all existing cell-reduction algorithms are, in practice, implemented using finite-precision floating-point algebra. It is immediately clear that this presents a conflict because it may not even be possible to faithfully represent the original unit-cell parameters if they are the result of expressions involving irrational numbers (such as $2^{1/2}$) as they arise in common change-of-basis transformations. This invalidates using exact conditions from the outset. It is also very easy to demonstrate that conventional implementations are unstable. Practical application of the algorithms of Gruber (1973), Křivý & Gruber (1976) and Zuo *et al.* (1995) reveals that rounding errors owing to floating-point arithmetic can lead to infinite loops given typical experimentally observed cell parameters. To the best of our knowledge, a proper treatment of rounding errors in this situation is not covered in the literature. However, such a treatment is essential in the context of highly automated crystallographic procedures seeking to minimize the need for human intervention. It is also essential that the basis transformation leading to a particular reduced cell is available for subsequent use, for example to transform atomic coordinates, symmetry operations or orientation matrices. The algorithms that we describe below are based on the work of Gruber (1973) and Křivý & Gruber (1976) and address both requirements.

To avoid misunderstandings, we wish to emphasize the distinction between rounding errors and experimental uncertainties, particularly the treatment of experimental uncertainties in the determination of the Bravais type. The treatment of the experimental uncertainties is not covered here. Papers on this subject have been published by Clegg (1981), Le Page (1982), Zimmermann & Burzlaff (1985) and Andrews & Bernstein (1988).

2. Prerequisites

2.1. Gruber notation

Gruber (1973) and Křivý & Gruber (1976) use the following notation in the description of the reduction algorithms:

$$\begin{aligned} A &= \mathbf{a}\cdot\mathbf{a} = a^2 \\ B &= \mathbf{b}\cdot\mathbf{b} = b^2 \\ C &= \mathbf{c}\cdot\mathbf{c} = c^2 \\ \xi &= 2\mathbf{b}\cdot\mathbf{c} = 2bc\cos(\alpha) \\ \eta &= 2\mathbf{a}\cdot\mathbf{c} = 2ac\cos(\beta) \\ \zeta &= 2\mathbf{a}\cdot\mathbf{b} = 2ab\cos(\gamma). \end{aligned} \tag{1}$$

a, **b**, **c** are the basis vectors of the unit cell, a, b, c, α, β, γ are the usual unit-cell parameters. The parameters A, B, C, ξ, η, ζ are closely related to the metrical matrix (also known as the metric tensor) **G**:

$$\mathbf{G} = \begin{pmatrix} A & \zeta/2 & \eta/2 \\ \zeta/2 & B & \xi/2 \\ \eta/2 & \xi/2 & C \end{pmatrix}. \tag{2}$$

Including the factors of 2 in the definition of ξ, η, ζ simplifies the expressions in the reduction algorithm.

We will refer to the individual steps of the Gruber (1973) algorithm by the labels used in the original paper as steps N1 through N3 and B1 through B5. Similarly, we will refer to the individual steps of the Křivý–Gruber (1976) algorithm by the labels used in the original paper as steps A1 through A8.

2.2. Definition of 'reasonable' and 'degenerate' unit-cell parameters

We have to consider that all reduction algorithms involve additions of real-valued parameters. Therefore, the use of floating-point arithmetic dictates that these parameters must be within a certain dynamic range. For example, if the double-precision (64-bit) values 1.0 and 10^{-20} are added, the result is 1.0 exactly. To avoid this type of problem, we define a set of unit-cell parameters as *degenerate* if:

(i) The minimum of the cell lengths divided by the maximum of the cell lengths is smaller than a certain factor $\varepsilon_{\text{lengths}}$.

(ii) The unit-cell volume divided by the minimum of the cell lengths is smaller than a certain factor $\varepsilon_{\text{volume}}$.

A set of unit-cell parameters is *reasonable* if it is not degenerate. Based on numerical tests, we use $\varepsilon_{\text{lengths}} = 10^{-10}$ and $\varepsilon_{\text{volume}} = 10^{-5}$.

2.3. Source-code availability

The source code required for reproducing all results reported below is available as open source under a license that grants permission to copy, use or modify the code for any purpose. The code is integrated into the Computational Crystallography Toolbox (Grosse-Kunstleve *et al.*, 2002). In the following, we include pointers to the relevant files in the Computational Crystallography Toolbox. The files may be obtained at http://cctbx.sourceforge.net/ or as complete self-contained bundles at http://cci.lbl.gov/cctbx_build/.

3. Treatment of rounding errors in the Křivý–Gruber algorithm

It is important to note that we have two goals:

Goal 1: The Křivý–Gruber reduction procedure must be numerically stable given reasonable unit-cell parameters.

Goal 2: It must be possible to *test numerically* that the result of the reduction procedure meets the conditions for Niggli-reduced cells according to the definition of Gruber (1973), or according to the alternative but equivalent definition in *International Tables for Crystallography*, Volume A (Hahn, 1983).

The Křivý–Gruber algorithm relies on the outcome of comparisons of algebraic expressions of the real-valued parameters A, B, C, ξ, η, ζ. To make the iterative reduction algorithm numerically stable, we use a tolerance ε in all comparisons. This tolerance accounts for the uncertainties implicitly introduced by the use of finite-precision algebra. Our experience (see below) shows that it is essential that the tolerance is larger than the accumulated rounding errors and that the same tolerance be used for a complete pass through the reduction procedure. To achieve goal 2, the same tolerance must be used in the evaluation of the Niggli conditions.

Finding the optimal (smallest) value for the tolerance is very difficult to achieve in a portable way. Fortunately, for all practical purposes the optimal value is not required. Our approach is to compute ε as

$$\varepsilon = \varepsilon_{\text{relative}} V^{1/3}, \tag{3}$$

where V is the volume of the unit cell. The term $V^{1/3}$ is an approximation to the lengths of the basis vectors (it is exact for cubic unit cells). $\varepsilon_{\text{relative}}$ is an upper estimate of the accumulated rounding errors. As we show below, a practical value is $\varepsilon_{\text{relative}} = 10^{-5}$.

To achieve the highest degree of consistency for all floating-point comparisons, we have replaced the exact comparisons in the Křivý–Gruber algorithm with the following:

exact	implementation
$x < y$	$x < y - \varepsilon$
$x > y$	$y < x - \varepsilon$
$x \leq y$	$\text{not}\, y < x - \varepsilon$
$x \geq y$	$\text{not}\, x < y - \varepsilon$
$x = y$	$\text{not}\,(x < y - \varepsilon \,\text{or}\, y < x - \varepsilon).$

Note that our implementation reduces to the conventional implementation if $\varepsilon = 0$.

In the following, all comparisons are understood to be using the tolerance ε, except for the comparison used to evaluate the sign function in steps A5, A6 and A7 of the Křivý–Gruber algorithm. Using the exact less-than comparison is the correct approach at this point because the values involved are first compared (using the tolerance) with the parameters A or B, which must be strictly greater than zero because the unit cell is degenerate otherwise.

To further maximize the numerical stability of our implementation, we treat the expression $\xi\eta\zeta > 0$ in step A3 of the Křivý–Gruber algorithm in a special way. Each parameter is tested individually, *e.g.* we test $\xi < 0$, $0 < \xi$, counting the number of positive and negative values. These integer counts are then used to decide if $\xi\eta\zeta > 0$. We also completely avoid floating-point multiplications. That is, all evaluations in the reduction procedure are implemented using only additions and subtractions.

Our implementation of the Křivý–Gruber algorithm can be found in the directory cctbx/cctbx/uctbx (see §2.3). The average time for one full pass through the reduction algorithm is about 2.6 ms on a 2.8 GHz Intel Linux computer.

4. Construction of change-of-basis matrices

Each of the eight steps in the Křivý–Gruber algorithm consists of testing for a condition, followed by an action. The actions are given by Křivý & Gruber as reassignments of the parameters A, B, C, ξ, η, ζ. A mathematically equivalent definition is given by the tensor transformation

$$\mathbf{G}' = \mathbf{C}^T\mathbf{G}\mathbf{C} \tag{4}$$

with a 3×3 transformation matrix $\mathbf{C}$ acting on the metrical matrix $\mathbf{G}$. $\mathbf{C}^T$ is the transpose of $\mathbf{C}$. There is an obvious one-to-one correspondence between the actions of the Gruber (1973) algorithm and the actions of the Křivý–Gruber algorithm. The following transformation matrices are labeled according to the actions defined in the original papers:

$$\text{N1, A1}: \mathbf{C} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \tag{5}$$

$$\text{N2, A2}: \mathbf{C} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix} \tag{6}$$

$$\text{N3 true branch, A3}: \mathbf{C} = \begin{pmatrix} i & 0 & 0 \\ 0 & j & 0 \\ 0 & 0 & k \end{pmatrix}, \tag{7}$$

$i = -1$ if $\xi < 0$ and 1 otherwise. j and k are evaluated in the same way using η and ζ, respectively.

$$\text{N3 false branch, A4}: \mathbf{C} = \begin{pmatrix} i & 0 & 0 \\ 0 & j & 0 \\ 0 & 0 & k \end{pmatrix}, \tag{8}$$

i, j, k are integer variables initialized with the values 1.

Let p be a reference that is initially undefined.

If $\zeta > 0$, $i = -1$ else if not $\zeta < 0$, p is redefined as a reference to i.

If $\eta > 0$, $j = -1$ else if not $\eta < 0$, p is redefined as a reference to j.

If $\zeta > 0$, $k = -1$ else if not $\zeta < 0$, p is redefined as a reference to k.

If $ijk < 0$, the variable referred to by p is set to -1.

(In the context of the Křivý–Gruber algorithm, p is always defined if $ijk < 0$.)

$$\text{A5}: \mathbf{C} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -\text{sign}(\xi) \\ 0 & 0 & 1 \end{pmatrix}, \quad \text{B2}: \mathbf{C} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -\text{entire}((\xi + B)/2B) \\ 0 & 0 & 1 \end{pmatrix} \tag{9}$$

$$\text{A6}: \mathbf{C} = \begin{pmatrix} 1 & 0 & -\text{sign}(\eta) \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \text{B3}: \mathbf{C} = \begin{pmatrix} 1 & 0 & -\text{entire}((\eta + A)/2A) \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{10}$$

$$\text{A7}: \mathbf{C} = \begin{pmatrix} 1 & -\text{sign}(\zeta) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \text{B4}: \mathbf{C} = \begin{pmatrix} 1 & -\text{entire}((\zeta + A)/2A) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{11}$$

Following the definition of Gruber (1973) and Křivý & Gruber, sign(x) is $+1$ if $x > 0$, -1 if $x < 0$. entire(x) is the greatest integer that is not greater than x.

$$\text{A8}: \mathbf{C} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix},$$
$$\text{B5}: \mathbf{C} = \begin{pmatrix} 1 & 0 & -\text{entire}((\xi + \eta + \zeta + A + B)/2(\zeta + A + B)) \\ 0 & 1 & -\text{entire}((\xi + \eta + \zeta + A + B)/2(\zeta + A + B)) \\ 0 & 0 & 1 \end{pmatrix}. \tag{12}$$

The individual change-of-basis matrices for each action taken are multiplied to yield the final change-of-basis matrix that transforms the original unit-cell parameters to the parameters of the Niggli-reduced cell. The multiplication order is such that the matrix for the first action taken is the left-most matrix, the matrix for the last action taken is the right-most matrix. All matrices $\mathbf{C}$ shown above have determinant 1. Each matrix could also be replaced by the matrix product

$$\mathbf{C}\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \tag{13}$$

since any lattice exhibits a center of inversion at the origin. However, the choices above ensure that the final matrix has determinant 1. This is a very useful property because otherwise a determinant of -1 would result in the transformation of a right-handed system into a left-handed system (and *vice versa*). With the definitions above, the final change-of-basis matrix is directly suitable for transforming symmetry operations, atomic coordinates, orientation matrices or other crystallographic parameters.

5. Analysis of the numerical behavior of the Křivý–Gruber algorithm

To ensure that our implementation of the Křivý–Gruber algorithm meets both goals stated in §3, we have written a comprehensive set of tests that systematically exercises the reduction algorithm with all possible types of unit cells. The three major tests are:

Test 1: A two-deep loop over seven point groups (one for each crystal system) and the conventional lattice centering types P, A, B, C, I, R, F. An arbitrary unit cell compatible with the point-group symmetry is transformed from the centered setting to a primitive setting and then passed to the reduction algorithm. Using the conditions defined in §9.3.2 of *International Tables for Crystallography*, Volume A (Hahn, 1983), it is asserted that the result passes a numerical evaluation of the Niggli conditions.

Test 2: A six-deep loop over selected values for the unit-cell parameters: $a, b, c = \{10, 20, 30\}, \alpha, \beta, \gamma = \{10, 30, 45, 60, 90, 120, 150, 170^\circ\}$. Only 3456 of the 13824 combinations yield a metrical matrix with a strictly positive determinant (corresponding to the square of the unit-cell volume). For each of these 3456 valid parameter combinations, it is asserted that the result of the reduction algorithm passes a numerical evaluation of the Niggli conditions.

Test 3: Based on Table 1 of Gruber (1973), we construct a given number (typically 100) of random Niggli-reduced cells for each of the 28 unique Niggli types. Each Niggli-reduced cell is transformed using a random integer matrix with determinant 1 (any 3×3 integer matrix with determinant 1 or -1 is a valid basis transformation). It is asserted that the reduction algorithm recovers the original Niggli-reduced cell.

Running these tests on a variety of hardware architectures (see http://cci.lbl.gov/cctbx_build/) confirms that our implementation of the Křivý–Gruber algorithm is numerically stable if the value 10^{-5} is used for the tolerance $\varepsilon_{\text{relative}}$. That is, the algorithm does not enter infinite loops (goal 1 in §3) and the results consistently pass a numerical evaluation of the Niggli conditions (goal 2). This proves that our implementation is free of errors and fully consistent with Table 1 of Gruber (1973). This result can be reproduced by running the `cctbx/cctbx/regression/tst_krivy_gruber.py` script (see §2.3).

Prompted by a referee skeptical about our use of the tolerance ε, we have carefully analyzed the numerical behavior of the Křivý–Gruber algorithm if we set $\varepsilon = 0$, corresponding to the conventional implementation. The exact behavior is of course highly platform specific, depending not only on the floating-point hardware but also on the compiler and optimization settings. Therefore, we present our results as observations based on a large number of random trials executed with double-precision (64-bit) floating-point arithmetic on a Pentium IV processor. In approximately 14000 trials with reasonable unit-cell parameters, as outlined above, the algorithm enters infinite loops approximately 100 times (*i.e.* the failure rate is 0.7%). This result can be reproduced by running the `tst_krivy_gruber.py` script with the `--track_infinite` option. This script prints a trace of the actions executed during the Křivý-Gruber algorithm. In the output, we found evidence of cyclic behavior with period lengths of two, four and six, for example the infinitely repeated cycle of actions with the Křivý–Gruber labels A4, A6, A3, A5, A4, A8 (see §2.1). When we analyzed the cycles, we found in all cases that the set of parameters generated after any of the actions in the cycle does not pass a numerical evaluation of the Niggli conditions if $\varepsilon = 0$. Many sets of parameters do not even pass a numerical evaluation of the main conditions according to §9.3.2 in *International Tables for Crystallography*, Volume A (Hahn, 1983) and many cells have mixed acute and obtuse angles, which presents an obvious violation of the Niggli conditions. Even one such failure is sufficient to prove that the conventional implementation of the Křivý–Gruber algorithm is not stable on current computing hardware architectures. We expect that this is the case for any implementation based on finite-precision algebra.

We determined the best choice for $\varepsilon_{\text{relative}}$ by executing tests 1–3 as outlined above with different values. If we set $\varepsilon_{\text{relative}} = 10^{-10}$, some tests involving Niggli-reduced cells transformed with unimodular matrices (test 3) that contain large elements (*e.g.* 50) fail because the result of the iterative reduction procedure does not match the original Niggli-reduced cell. This means that the accumulated rounding errors due to repeated additions and subtractions lead to a different Niggli cell considering the small $\varepsilon_{\text{relative}}$. This is a violation of our goal 2 stated in §3. All tests pass only after increasing $\varepsilon_{\text{relative}}$ to 10^{-6}. To be certain, we use $\varepsilon_{\text{relative}} = 10^{-5}$.

6. The minimum reduction algorithm

The problems reported in the previous section prompted us to consider the motivation behind the Křivý–Gruber algorithm. Why is it useful to compute the Niggli-reduced cell? In the literature, the Niggli reduction consistently appears in the context of determining the Bravais type, for example by lookup in Table 9.3.1 of *International Tables for Crystallography*, Volume A (Hahn, 1983). However, as Le Page (1982) and Andrews & Bernstein (1988) point out, this approach is bound to fail because of experimental uncertainties. We fully agree with this view and add that our results in the previous section show how rounding errors alone can invalidate the approach of determining the Bravais type *after* determining the Niggli-reduced cell.

Le Page (1982) suggests an alternative, elegant and robust algorithm for the determination of the lattice symmetry, which takes into account experimental uncertainties. The fundamental idea is to exhaustively search for twofold axes by testing all possible integral lattice vectors with elements up to absolute value two. The combination of twofold axes is a unique and unambiguous indication of the Bravais type. Le Page shows that restricting the search to vectors with elements up to absolute value two is sufficient if the basis vectors of the input unit cell correspond to the shortest vectors of the lattice. This is the only requirement. Le Page's procedure is insensitive to the choice of the angular unit-cell parameters. In view of this, it is surprising that Le Page employs a Buerger reduction equivalent to that of Gruber (1973) which is also concerned with the choice of the angular parameters. We have determined empirically that the Buerger reduction according to Gruber exhibits the same numerical problems as the Křivý & Gruber reduction (this test is included in the `tst_krivy_gruber.py` script).

Our first attempt at devising a numerically stable cell-reduction procedure that does not require the use of a tolerance was to remove the conditions that most obviously conflict with the use of finite-precision algebra: the tests for exact equality. The first interesting observation is that the end result of simplifying the Gruber algorithm in this way is almost identical to the similar simplification of the Křivý–Gruber algorithm. The only difference is the use of the *entire* function in the Gruber algorithm compared to the sign function in the Křivý-Gruber algorithm. We prefer the *entire* function because the algorithm converges faster (Gruber, 1973). The second important observation is that the simplified algorithm still converges to a unit cell with basis vectors of minimum length. The resulting unit cell is therefore suitable as an input to Le Page's robust search for the lattice symmetry.

Unfortunately, the simplified Gruber algorithm can still be unstable, even given reasonable unit-cell parameters. Detailed analysis of the cyclic behavior based on a large number of random trials revealed that there are at least two types of cycles with period length two and four, respectively. In each case, the algorithm alternates between the action in the N3 false branch and one or two other actions (B3 and/or B5). In all steps of a cycle, the unit-cell lengths are identical within a very small tolerance after each of the actions involved. To test for this condition, we evaluate the expression

$$(x \times multiplier + (x - y)) - (x \times multiplier) \qquad (14)$$

numerically for each Gruber parameter A, B, C, where x is the most current parameter after executing the action in the N3 false branch, and y is the corresponding parameter after the previous execution of the same action. *multiplier* is a small factor. Based on a large number of tests, we work with *multiplier* = 10. For example, with this choice, using 64-bit floating-point arithmetic, the evaluation of (14) yields exactly 0 if $x = 1$ and $y = 1 + 0.1111 \times 10^{-15}$. The reduction algorithm is terminated if the numerical evaluation of (14) yields 0 for all Gruber parameters A, B, C after two subsequent passes through the action in the N3 false branch. Terminating at this point ensures that all cell angles are obtuse. This is an arbitrary choice that does not affect the suitability of the end result as input to Le Page's algorithm for the determination of the lattice symmetry. We refer to the simplified and modified Gruber algorithm as the minimum reduction because it produces a unit cell with basis vectors of minimum length.

By repeatedly running a large number of random trials (tests 1–3 outlined in §5) on a variety of platforms, we have established that the minimum reduction is stable given reasonable unit-cell parameters. This result can be reproduced by running the `tst_krivy_gruber.py` script mentioned before with the `--Forever` option. This script also tests that the difference between the unit-cell lengths produced by our implementation of the Křivý–Gruber algorithm with $\varepsilon_{\rm relative} = 10^{-5}$ and the parameters produced by the minimum reduction is smaller than 10^{-6} in absolute units, validating both algorithms.

Of course, given degenerate unit-cell parameters, it is still possible that the minimum reduction enters an infinite loop or that the Gruber parameters A, B, C become zero or negative in the course of the algorithm. To prove that this occurs only for degenerate parameters, we have set up an additional test with both reasonable and degenerate parameters. Any set of random parameters that leads to a unit-cell volume greater than zero is processed by the minimum reduction algorithm. If the algorithm exceeds 100 iterations or if any of the parameters A, B, C becomes zero or negative, it is asserted that the parameters are degenerate according to our definition in §2.2. By executing the minimum reduction one million times with random parameters, we have verified that this test is invoked in about 1.9% of the trials and succeeds reliably. This result can be reproduced by running the `cctbx/uctbx/boost_python/tst_uctbx.py` script with the `--hardest` option.

A reference implementation of the minimum reduction is in the file `cctbx/cctbx/uctbx/gruber_1973.py`, class `fast_minimum_reduction`. Owing to the clear and concise syntax of the Python language, this code will be most useful to study the complete algorithm. A C++ implementation of the exact same algorithm is in the file `cctbx/include/cctbx/uctbx/fast_minimum_reduction.h`. This code executes about 50 times faster than the Python implementation but is longer. The average time for one full pass through the C++ implementation of the minimum-reduction algorithm is about 43 µs on a 2.8 GHz Intel Linux computer.

7. Summary

We have shown that the Křivý–Gruber (1976) algorithm, given reasonable unit-cell parameters, is numerically stable if a certain tolerance is considered in the evaluation of the conditions embedded in the algorithm. Using the same tolerance, it is also possible to verify that the result passes a numerical evaluation of the Niggli conditions. We obtained analogous results with the Buerger reduction algorithm of Gruber (1973).

We have also provided a proof that the same implementation is not stable on current computing hardware architectures if the tolerance is set to zero, emulating the conventional implementation. We have no reason to believe that this could be different as long as finite-precision algebra is used. On the contrary, we are quite surprised that the obvious conflict between the exact tests for equality included in the Niggli conditions and finite-precision algebra has not been addressed in the literature for almost five decades since Buerger (1957) introduced the first numerical reduction procedure. A possible explanation is that the failure rate is only on the order of 1%. If a human is involved, an algorithm that enters an infinite loop can be terminated manually and the input modified slightly until the algorithm succeeds.

For a highly automated process, a failure rate of 1% is not acceptable. For example, auto-indexing programs might have to process hundreds of cell reductions per diffraction image. The need for human intervention is clearly prohibitive and ignoring reasonable cell parameters only because the reduction algorithm is not stable might lead to misindexed images. This, combined with the insight that the Niggli reduction is

performing more work than is actually required to solve the problem of determining the Bravais type, has prompted us to devise the minimum reduction algorithm which is numerically stable because exact tests for equality are omitted and because it is possible to reliably terminate the algorithm based on equation (14).

We believe that the algorithm with the highest practical value is the minimum reduction. We use this algorithm in a novel auto-indexing procedure (unpublished results). The Computational Crystallography Toolbox includes a web interface (http://cci.lbl.gov/cctbx/lattice_symmetry.html) to the part of the procedure that determines the lattice symmetry based on the work of Le Page (1982) and Grosse-Kunstleve (1999). The Křivý–Gruber algorithm, using 64-bit floating-point algebra and the tolerance $\varepsilon_{\rm relative} = 10^{-5}$, has the advantage of producing results that are identical on different platforms (otherwise test 3 outlined in §5 would fail). Therefore, we use the Křivý–Gruber algorithm to transform atomic coordinates and reflection data to a primitive setting in order to speed up certain computations, in particular the computation of the fast translation function (Navaza & Vernoslova, 1995; Grosse-Kunstleve & Adams, 2003). If the transformations to the primitive setting are identical on different platforms, results of subsequent computations can be compared more easily.

All results reported here are based on the cctbx source code bundle with the version tag 2003_09_25_1908. It is our intention to provide this bundle (and all bundles that worked on all platforms) indefinitely at http://cci.lbl.gov/cctbx_build/.

Our work was funded in part by the US Department of Energy under contract No. DE-AC03-76SF00098 and NIH/NIGMS under grants Nos. 1P50GM62412 and 1P01GM063210.

References

Andrews, L. C. & Bernstein, H. J. (1988). *Acta Cryst.* A**44**, 1009–1018.
Buerger, M. J. (1957). *Z. Kristallogr.* **109**, 42–60.
Clegg, W. (1981). *Acta Cryst.* A**37**, 913–915.
Eisenstein, G. (1851). *J. Math. (Crelle)*, **41**, 141.
Grosse-Kunstleve, R. W. (1999). *Acta Cryst.* A**55**, 383–395.
Grosse-Kunstleve, R. W. & Adams, P. D. (2003). *Acta Cryst.* D**59**, 1966–1973.
Grosse-Kunstleve, R. W., Sauter, N. K., Moriarty, N. W. & Adams, P. D. (2002). *J. Appl. Cryst.* **35**, 126–136.
Gruber, B. (1973). *Acta Cryst.* A**29**, 433–440.
Hahn, T. (1983). *International Tables for Crystallography*, Vol. A. Dordrecht: Kluwer.
Křivý, I. & Gruber, B. (1976). *Acta Cryst.* A**32**, 297–298.
Le Page, Y. (1982). *J. Appl. Cryst.* **15**, 255–259.
Navaza, J. & Vernoslova, E. (1995). *Acta Cryst.* A**51**, 445–449.
Niggli, P. (1928). *Krystallographische und strukturtheoretische Grundbegriffe. Handbuch der Experimentalphysik*, Vol. 7, part 1, pp. 108–176. Leipzig: Akademische Verlagsgesellschaft.
Zimmermann, H. & Burzlaff, H. (1985). *Z. Kristallogr.* **170**, 241–246.
Zuo, L., Muller, J., Philippe, M.-J. &Esling, C. (1995). *Acta Cryst.* A**51**, 943–945.