Abstract
An efficient computer-based procedure for randomly accessing reflection data is described.

Introduction
Crystallographic calculations such as data merging and direct methods need to use reflection data in a random order. Most existing procedures for doing this tend to be either slow or inefficient in their use of computer memory, or both.

The most commonly used procedure stores data in a three-dimensional array of subscripted variables based on the Miller indices h, k, l. For instance, the net intensity would be stored in an array INET(IH, IK, IL). Despite the simplicity of this approach it has a number of deficiencies.

Firstly, the time required to access a variable stored in a three-dimensional array is much more than that required to locate the same variable in a one-dimensional array. The actual address of the variable in a 3D array must be computed at execution time according to the expression IH + MAXH*IK + MAXH*MAXK*IL. Secondly, a 3D array defines a ‘rectilinear block’ of memory in which the actual ‘ellipsoid’ of data occupies just over one half of the volume. This results in a serious waste of the computer memory.

Thirdly, to allow for varying requirements of different structures and calculations it is often necessary to specify the limits of a 3D array using variable DIMENSION statements. Apart from the lack of universality of this type of statement, there are also constraints on how and where this type of specification can be made. Fourthly, the necessity to allow for positive and negative h, k, l indices in a 3D array, and the desirability of omitting systematically absent data for centred space groups, requires that the 3D array subscripts take a form such as IH = h/n + |h|max, IK = k/n + |k|max, IL = l/n + |l|max. n is an integer allowing for systematic absence conditions. This requirement adds additional overheads to locating data in a 3D array.

Lastly, and most importantly, the 3D-array approach usually requires an entire ellipsoid (or at least semi-ellipsoid) of data to be stored, independent of the space-group symmetry.

Efficient random access procedure
A procedure for the efficient access to crystallographic data can be based on the use of a single-dimensional array. The importance of a single-dimensional array approach has been stressed by Stewart (1975). Data is stored in the array and sorted so that h1 is the most rapidly varying index and h3 is the least. The indices h1, h2, h3 are the Miller indices h, k, l ordered so that |h1|max ≥ |h2|max ≥ |h3|max. In effect the sorted data is packed into contiguous ‘rows’ of fixed h2h3 and varying h1. The sorted data is accessed via three tables whose coordinates are based on the indices h2 and h3. These are referred to as h2h3 tables. A diagrammatic representation of these three tables is shown in Fig. 1.

The h2h3 pointer table contains the address in memory of each data point with h1 = h1. h2 is the index in the stored h2h3 row that is closest to the value zero. For primitive space groups h2 will always be zero and for nonprimitive space groups it will be some value in the range + 2 to −2. These values are stored in the h2h3 index table. Data for any reflections h1h2h3 may be located in the single-dimensional array by using h2 and h3 to obtain the address of the reflection h2h3h1 and adding the index h1 (or as discussed below, adding a fraction of h1). In order to ensure that the sought reflection h1h2h3 is contained in the row [−nh2h3 ... h1h2h3h1 + nh2h3] the maximum value of |h1| for each row (in this case the value n) is stored in a third |h1|max table and tested before an address is sought.

For nonprimitive space groups the systematically absent reflections are not stored so that the memory is not used for null data. This necessitates the use of the h2h3 index table and the addition of h1/m to the address of h2h3h1. m is the integer in the general hkl systematic absence condition.

An important advantage of this approach is that only a unique segment of data (i.e. no equivalents) need be stored for triclinic, monoclinic and orthorhombic space groups without a deterioration in access times. This is because the h2h3 tables are constructed so that symmetrically equivalent h2h3 always refer to the same unique segment of data. For example, with orthorhombic space groups the four equivalent coordinates +h2, +h3, −h2, +h3; +h2, −h3; −h2, −h3 and −h2, −h3 in each of the h2h3 tables contain identical information. In this way symmetry operations involving equivalent reflections related by a simple sign change in h, k or l are handled automatically during the access to data. This means, however, that for higher-symmetry space groups a minimum of one octant of data must be stored in the single-dimensional array. This redundancy is considered an acceptable trade-off for the more efficient access speeds provided by this approach.

Basic algorithmic procedures

1. Data and pointer table preparation
Step 1. Construct the hkl-to-h1h2h3 decode parameters I1, I2, I3 so that if h = IH (1), k = IH (2), l = IH (3) then h1 = IH (11), h2 = IH (12), h3 = IH (I3).
Step 2. Construct the h1h2h3-to-hkl decode parameters J1, J2, J3 so that if h1 = JH (1), h2 = JH (2), h3 = JH (3) then h = JH (J1), k = JH (J2), l = JH (J3).
Step 3. Generate from the space-group symmetry possible sign combinations of h1, h2, h3 and store as S1(i), S2(i), S3(i), i=1 to NRZ, where NRZ is the number of rotation matrices with an ‘identical pattern of zeros’.

0021-8898/81/030214-02501.00 © 1981 International Union of Crystallography
Step 4. Enter the data file and (a) generate the equivalent \( h, k, l \) and translate into \( h_1, h_2, h_3 \); (b) arrange equivalent \( h_1, h_2, h_3 \) into segments of reflections which are related by simple sign change defined by \( S_1, S_2, S_3 \); (c) select an \( h_1 h_2 h_3 \) segment as the unique data set and extend the remaining equivalent \( h_1, h_2, h_3 \) residues in a common segment to a minimum of one octant if necessary; (d) store reflections as a packed word of the general form
\[
\{ h_3 + p, h_2 + q, h_1 + r, \text{[data]} \}
\]
where \( p, q, r \) are the packing constants determined both by the maximum values of \( h_1, h_2, h_3 \) and the permissible signs, and by whether the \([\text{data}]\) is to be packed or stored separately (this last consideration is structure and problem dependent).

Step 5. Initialize the \( h_2 h_3 \) pointer tables ready for their preparation in step 6(c).

Step 6. Sort on packed \( \{ h_3 + p, h_2 + q, h_1 + r \} \) words and (a) store only the \([\text{data}]\) information (the indices \( h_1 h_2 h_3 \) are implied by their position in memory); (b) fill in 'missing data' so that each row of the stored segment of data varies monotonically; (c) update pointer tables with values of \( h_1 \text{[max]} \), \( h_0 \text{[first]} \), the address of \( h_1 \text{[first]} \) in the sorted 1D array ensuring that these values are inserted for all symmetrically equivalent positions in the \( h_2 h_3 \) plane;

Step 7. Finally scan the pointer tables to ensure that all Friedel-related values are present.

(II) Access the data

Step 1. Translate indices \( h, k, l \) of sought reflection to \( h_1, h_2, h_3 \) using 11, 12, 13.

Step 2. Use \( h_2 \) and \( h_3 \) to obtain from the \( h_2 h_3 \) tables (a) the \( |h_1| \) maximum, and test that \( |h_1| \) is less than this value; (b) the \( h_1 \text{[first]} \) pointer and \( h_1 \text{[index]} \) index, and point to \( h_1 h_2 h_3 \) as \( h_1 \text{[first]} \) pointer + \([h_1 - h_0 \text{[index]}]/m\), where \( m \) is the general systematic absence multiplicity. For the primitive space groups this pointer simplifies to \( h_1 \text{[first]} \) pointer + \( h_1 \).

Reference