ELSEVIER

Sequential and parallel molecular mechanics calculations

David N.J. White

Department of Chemistry, University of Glasgow, Glasgow, Scotland

This article describes a gradient algorithm for the computational optimization of model molecular structures, and discusses the various compromises inherent in the practical expression of the algorithm in a Fortran computer program (VULCAN) for both sequential and parallel computers. Details are given of some previously undiscussed properties of gradient algorithms; various acceleration techniques are compared; and some traps for the unwary are highlighted.

INTRODUCTION

We have had a long involvement with the development of molecular mechanics calculations,[1,2] molecular mechanics force fields,[3,4] and the development of computer programs to implement both of the foregoing.[5,6] Until fairly recently our production molecular mechanics code was written in Occam and ran on a parallel computer composed of an array of 64 first-generation (T800) floating point transputers.[7] We realized that this was not a particularly portable program but were prepared to sacrifice this property for the increased speed of the Occam implementation; a factor of two when compared with a Fortran variant. The very late delivery and effective commercial failure of the second-generation (T9000) transputers forced us to look elsewhere when the time came to upgrade our T800-based parallel computer.

In common with a number of other universities and commercial enterprises we determined that it was no longer cost effective to design and build proprietary nodes for a parallel computer, given the low prices and high performance (even when compared with the latest and faster of "hot" chips) of mass-produced PC/workstation motherboards. The exercise then becomes one of designing and building an appropriate interconnection network for the host and nodes of a parallel computer built around these motherboards. We chose to use 486/Pentium/P6 motherboards for our parallel computer because they offered the best price/performance ratio; and also because software, particularly operating systems and compilers, for IBM-compatible PCs is an order of magnitude less expensive than that for most of the alternatives. Our molecular mechanics programs were then translated into Fortran and the parallel version based around a library of simple message-passing subroutines. The message-passing subroutines have exact analogs on all commercially available message-passing parallel computers, making our programs highly portable. The design of our "Parallel PC" machine[8] and its associated message-passing subroutine library, COMFORT,[9] have been described in detail elsewhere, but a brief description of the hardware follows for the sake of completeness.

As currently implemented the Parallel PC consists of a host 486/PC connected to each of eight node 486/PCs (all minus keyboard and monitor) by a dedicated 20-Mbits/sec Inmos serial link. The nodes are also completely interconnected (i.e., each to every other) by means of these high-speed serial links. In addition, the host and nodes are linked by a local ethernet, but this is too slow to be used for anything other than diagnostic and debugging purposes. An important property of the interprocessor serial link system is that it supports broadcasting in hardware; which means, for example, that the host can load programs or data onto all eight nodes simultaneously by concurrently transmitting the same information down all eight of its serial links. Our machine is physically compact because the nodes are stacked next to the host, but there is no reason why the interprocessor links and software could not be added to a cluster of PCs normally used for word processing or other purposes. This particular machine architecture is not really expandable beyond 32 processors because of the "complete connect" interprocessor link scheme; but it would be possible to have more sparsely interconnected clusters of 32 completely interconnected nodes. The programming language used is Microsoft Fortran Powerstation, which has a 32-bit flat memory space (i.e., no 64-KB memory segmentation as with 16-bit compilers for the PC) allowing direct access to all of the memory on a PC.

After the initial construction of our molecular mechanics program on the new machine it was decided that the time was opportune to review the implementation in detail, with a view to improving on the algorithms and methodology then in use. The next section gives a brief introduction to molecular mechanics calculations and discusses the various

Address reprint requests to: David N.J. White, Department of Chemistry, University of Glasgow, Glasgow G12 8QQ, Scotland.
Received 20 February 1996, accepted 24 February 1996.

655 Avenue of the Americas, New York, NY 10010

0263-7855/96/$15.00
PII S0263-7855(96)00042-2

factors that led to our particular implementation of the methodology.

MOLECULAR MECHANICS CALCULATIONS

The central problem of molecular mechanics calculations is essentially one of multidimensional function minimization. The function to be minimized is the molecular potential energy, or steric energy, expressed in terms of the deviations of internal coordinates and nonbonded distances from arbitrary reference or "strain-free" values.

In the first instance, therefore, a set of molecular coordinates, either internal (i.e., bond lengths, valency angles, and torsion angles) or Cartesian must be obtained in order that the steric energy may be evaluated and subsequently minimized by systematic variation of these coordinates. The coordinates would typically be generated by means of a molecular graphics program.[10] If the global, or absolute, minimum energy conformation of the molecule is being sought, then the initial, or trial, coordinates must correspond to a point in the potential energy hypersurface that lies within the potential well whose bottom represents the global energy minimum. The problem of separating the global energy minimum from a multitude of local energy minima is a formidable problem in itself[11,12] but will not be discussed further at this point; this article is exclusively concerned with efficient means of locating the bottom of potential energy wells whether they correspond to global or local energy minima.

The steric energy

The steric energy, V_s, is usually calculated in terms of internal coordinates and nonbonded distances, which are readily calculated from Cartesian coordinates, so that, in its simplest form,

$$V_s = \sum_l V_l + \sum_\theta V_\theta + \sum_\omega V_\omega + \sum_r V_r + \sum_q V_q + \sum_\chi V_\chi$$

$$V_l = \tfrac{1}{2}k_l(l - l_0)^2 \qquad V_\theta = \tfrac{1}{2}k_\theta(\theta - \theta_0)^2$$

$$V_\omega = \tfrac{1}{2}k_\omega(1 + s\cos n\omega) \qquad V_r = -Ar^{-6} + Br^{-12}$$

$$V_q = e_i e_j / Dr \qquad V_\chi = \tfrac{1}{2}k_\chi(\chi - \pi)^2$$

where l, θ, ω, r, q, and χ are the bond lengths, valency angles, torsion angles, 1 ... 4 and higher interatomic distances, Coulombic interactions, and improper torsion angles, respectively. k_l and k_θ are the force constants, and l_o, θ_o the reference values, for bond stretching and angle bending; k_ω is the barrier to free rotation; $s = 1$ for a staggered and -1 for an eclipsed torsional potential energy minimum of periodicity n; A and B are constants specific to each pair of nonbonded atom types; e_i and e_j are partial atomic charges; D is the dieletric constant; k_χ is the force constant for out-of-plane bending; while the magnitude of V_χ is a function of the amount by which χ departs from π radians.

In practice more complex expressions are used for angle bending, to allow for anharmonicity; for bond torsion, to simulate the complex barriers to free rotation in nonsymmetric situations; and for Coulombic interactions to allow the use of a distance-dependent dielectric constant. Therefore,

$$V_\theta = \tfrac{1}{2}k_\theta\{\Delta\theta^2 - k'_\theta(|\Delta\theta^3| - k''_\theta|\Delta\theta^5|)\} \qquad \Delta\theta = \theta - \theta_0$$

$$V_\omega = \tfrac{1}{2}k_\omega(1 + s\cos n\omega) + \tfrac{1}{2}k'_\omega(1 + s\cos\omega)$$

$$V_q = e_i e_j / r^2$$

The cubic term in the angle-bending potential function allows for anharmonicity when $\Delta\theta$ is large (a not uncommon situation given that k_θ is relatively small) and the $\Delta\theta^5$ term removes an unwanted maximum in the quadratic plus cubic potential. The torsional potential function is now a short Fourier series, where the onefold term is necessary to reproduce phenomena such as the gauche/trans energy difference in n-butane and the energy difference between cis and trans amides. The Coulombic potential function incorporating a distance-dependent dielectric constant seems to work as well as, if not better than, the classical expression and has the added advantage of computational economy (there is no need to calculate odd powers of r and therefore no need to take computationally extravagant square roots).

Energy minimization

After the steric energy has been evaluated then the coordinates must be iteratively adjusted in such a manner as to minimize this energy. The majority of molecular mechanics programs accept their primary input as Cartesian coordinates and calculate internal coordinates, as necessary, in order to evaluate the steric energy. Energy minimization in Cartesian hyperspace is straightforward as the Cartesian coordinates are always independent. However, if a gradient energy minimization algorithm is used then obtaining an expression for the first and second partial derivatives of the steric energy, which is expressed as a function of internal coordinates, with respect to the Cartesian coordinates is not particularly straightforward.

Conversely, if one begins with internal coordinates then the calculation of nonbonded distances from internal coordinates is not as straightforward as the conversion from Cartesians to internals; and the energy minimization of cyclic molecules is problematical because the equations of constraint between the internal coordinates necessary to ensure ring closure are not easily formulated. This latter problem is usually overcome by treating the molecule as linear, which obviates the need for constraints, but including a large fictitious potential between the chain ends in order to ensure that the ring remains closed. Expressions for the first and second partial differential coefficients of the steric energy with respect to the internal coordinates, bearing in mind the effect of the nonbonded potential, for use with gradient minimizers are about as laborious to obtain as the corresponding partial derivatives in Cartesian coordinates.

A further consideration is that gradient molecular mechanics calculations and molecular dynamics algorithms are frequently implemented in the same computer program, as there is a considerable amount of code common to both procedures if the minimization is executed in Cartesian hyperspace. Molecular dynamics calculations are based on Newton's laws of motion, which are valid only in Cartesian coordinates so that the use of internals is not a choice in this case. On balance, therefore, it is more economical with computer code and probably more elegant from a mathematical point of view to minimize in Cartesian coordinates,

although at least one widely used (minimization only) program uses internals.[13]

Having made the decision to minimize in Cartesian coordinates the next step is the choice of minimization algorithm. There are two major classes of minimization procedures: search methods and gradient methods. The latter class may be further subdivided into gradient methods that use only first derivatives and those that use first and second derivatives.

The choice of minimization routine for use in a molecular mechanics program depends on the criteria used to specify the minimum and, more practically, on the tests for convergence incorporated in the same. In the past criteria such as an energy decrement of 10^{-2}–10^{-6} kcal mol^{-1} from the penultimate iteration have been used as indictors of convergence on a minimum, sometimes with disastrous results.[14] For example, it is obvious that fairly large but balanced changes in molecular geometry will leave the energy unchanged, so that energy decrements are a dangerous and uncertain gauge of convergence. It is instructive to examine the criteria specified by Murray[15] in order to test whether the position vector **x** (components x_i, $i = 1,3N$, where N is the number of atoms in the molecule) specifying an approximation to a strong local minimum, is sufficiently close to the position vector **x*** representing the strong local minimum for the minimization to be terminated.

Test	Result
1. Test the final rate of convergence by constructing a table of $f(\mathbf{x}_{n-1}) - f(\mathbf{x}_n)$, where n is the iteration number	(i) Rate of convergence is superlinear (ii) Rate of convergence linear but fast (iii) Rate of convergence linear and slow
2. Test the magnitude of the Euclidean norm $\|f'(\mathbf{x})\|_e$	(i) $\|f'(\mathbf{x})\|_e < 10 \times 2^{-t}$ (ii) $10 \times 2^{-t} \leq \|f'(\mathbf{x})\|_e < 2^{-t/2}$ (iii) $\|f'(\mathbf{x})\|_e > 2^{-t/2}$ where t is the machine word length in bits
3. Test whether or not the Hessian matrix is positive definite and determine a bound on κ, its condition number	(i) Positive definite, $\kappa < \|f'(\mathbf{x})\|_e^{-1/2}$ (ii) Positive definite, $\kappa < 0.1\|f'(\mathbf{x})\|_e^{-1}$ (iii) Indefinite, or $\kappa \geq 0.1\|f'(\mathbf{x})\|_e^{-1}$

Where $f'(\mathbf{x})$ is the vector of first partial derivatives $\partial V_s/\partial x_i$ and the Hessian is the matrix of second partial derivatives $\partial^2 V_s/\partial x_i \partial x_j$. If after applying the tests the results are always (i) then it is almost certain that **x** is a good approximation to the solution; and the converse is likely to be true if the results are always (iii). Murray then gives a list of optimization methods most likely to provide estimates that will satisfy these tests; the higher in the list the better the method.

1. Second-derivative methods
2. Quasi-Newton methods
3. Conjugate gradient methods
4. Discrete quasi-Newton methods
5. Conjugate direction methods
6. Direct search methods

It is our experience that only second-derivative methods are uniformly reliable for molecular mechanics calculations across a wide spectrum of molecular species[16]; although the (first derivatives only) conjugate gradient method is quite often used,[17] particularly for large molecules, because of the computational expense of evaluating second derivatives. Although search methods appear at the bottom of the list they do have some advantages over the gradient methods, which dominate the top of the list. Direct search methods guarantee convergence on a local energy minimum; whereas gradient methods minimize and maximize with equal facility (the outcome depends on whether the calculation is started closer to a minimum or maximum), requiring computational tests to distinguish one from the other. The simplex direct search method forms the basis of at least one molecular mechanics program[18] because it has the useful facility, under some circumstances, of being able to tunnel through small potential barriers and locate the lowest energy well in a group of closely proximate minima (not global energy minimization, but a step [sic] in the right direction).

Another important consideration is that while the rate of convergence (measured by the inverse of the number of iterations/function evaluations required to locate the minimum to a given degree of accuracy, starting from the same point) and ability to locate minima accurately decrease as the table is traversed from top to bottom, the radius of convergence (measured by the distance between **x*** and the worst case starting point that will still converge on the minimum) increases as the table is traversed in the same direction. Some second-derivative procedures will not converge at all unless **x** is very close to **x*** and generating a good enough starting point usually entails preminimization with a large radius of convergence procedure (often a direct search or first-derivative gradient method).[19]

In the light of the foregoing discussion therefore, we use the second-derivative Newton–Raphson iteration:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha F^{+} \nabla V_s(\mathbf{x})$$

where **x** is the vector of atomic coordinates, k is the iteration number, α is the step length, F is the Hessian matrix $\partial^2 V_s/\partial x_i \partial x_j$, F^{+} is the generalized inverse of F, and $\nabla V_s(x)$ is the vector of first partial derivatives $\partial V_s/\partial x_i$. Use of the generalized inverse provides a convenient method of removing overall molecular rotation and translation from the singular, $3N \times 3N$ square Hessian.[20] If **x** is not a sufficiently good approximation to **x*** then the correction vector $F^{+}\nabla V_s(\mathbf{x})$ often has the "correct" direction but the "wrong" Euclidian norm, so that the iteration becomes oscillatory or even

divergent. This behavior may be remedied by setting $0 < \alpha < 1.0$ during the initial stages of the calculation.

The full matrix Newton–Raphson (FMNR) procedure outlined above has two practical drawbacks: it has an extremely small radius of convergence, so that the use of a preminimizer is always required; and the evaluation, followed by inversion, of a Hessian much larger than 1 000 × 1 000 is too time consuming and uncertain (because of potential numerical instability) to be practicable. The FMNR is therefore useful only for models of molecules comprising up to a maximum of 200–300 atoms. Beyond this upper limit for FMNR a variety of approximations such as truncated Newton methods,[21] adopted basis Newton methods,[22] and block diagonal Newton–Raphson (BDNR) methods[23] have been employed. We have chosen to use the latter.

The BDNR method approximates the Hessian by

$$F = \partial^2 V_s/\partial x_i \partial x_j \quad \begin{matrix} i = 3m+1, 3m+3 \\ j = 3m+1, 3m+3 \end{matrix} \} m = 0, N-1$$

so that the Hessian corresponds to a series of 3 × 3 submatrices along the leading diagonal, one relating to each atom in the molecule, with all other elements set to zero. Because each atom is treated in isolation the submatrices (which are individually no longer singular) can be inverted separately. As the level of approximation to the Hessian increases, for example on going from FMNR → BDNR → pure diagonal NR → steepest descents then the radius of convergence increases while the rate of convergence decreases. The BDNR method is close to an optimum compromise in this respect; its radius of convergence is sufficiently large that it will still converge when starting from rather poor initial structures, while its rate of convergence is sufficient that an inordinately large number of iterations is not required, even for large molecules.

The partial derivatives

The partial derivatives $\partial V_s/\partial x_i$ and $\partial^2 V_s/\partial x_i \partial x_j$ can be calculated either numerically or analytically, and both methods are used in commonly available molecular mechanics programs. Some programs even calculate the first partial derivatives analytically and the second partial derivatives numerically by finite differences from the first. Calculating the derivatives numerically would appear to be straightforward but slow; while by comparison, analytical calculation would appear to be fast at the expense of some tedious differential calculus. We have tried both methods and found the gap to be not as large as it at first appears.

Numerical derivatives can be calculated from a small number of function evaluations provided that the correct strategy is employed. The first partial derivatives of V_s with respect to x_i are calculated by central differences:

$$f'(x_i) = \{f(x_i + \delta x) - f(x_i - \delta x)\}/2\delta x$$

Notice that while a function evaluation could be saved here by using either forward or reverse differences, we have found that the ensuring poor estimate of $f'(x_i)$ slows convergence appreciably; and in any case both $f(x_i + \delta x)$ and $f(x_i - \delta x)$ are required to calculate the second derivatives. It is possible to evaluate the second derivatives by using central differences twice to give

$$f''(x_i, x_j) = \{f(x_i + \delta x, x_j + \delta x) - f(x_i - \delta x, x_j + \delta x) - f(x_i + \delta x, x_j - \delta x) + f(x_i - \delta x, x_j - \delta x)\}/4\delta x^2$$

but this requires an extra four function evaluations per second derivative. A much more elegant solution involves calculation by a combination of forward and reverse differences[24]; and while this leads to a poorer estimate of the second derivatives than repeated central differences the convergence of BDNR is much less sensitive to the quality of the second derivatives than the first derivatives. If we use forward differences to obtain

$$f'(x_i) = \{f(x_i + \delta x) - f(x_i)\}/\delta x$$

then a subsequent application of reverse differences gives $\partial^2 V_s/\partial x_i^2$ as

$$f''(x_i,x_i) = \{f(x_i + \delta x) + f(x_i - \delta x) - 2f(x_i)\}/\delta x^2$$

while a further application of forward differences gives $\partial^2 V_s/\partial x_i \partial x_j$ as

$$f''(x_i, x_j) = \{f(x_i + \delta x, x_j + \delta x) - f(x_i, x_j + \delta x) - f(x_i + \delta x, x_j) + f(x_i, x_j)\}/\delta x^2$$

This requires only one function evaluation, $f(x_i, x_j) = f(x_i) = f(x_j)$, to calculate each $\partial^2 V_s/\partial x_i^2$ and one function evaluation, $f(x_i + \delta x, x_j + \delta x)$, to calculate each $\partial^2 V_s/\partial x_i \partial x_j$ as we can reuse the $f(x_i + \delta x)$ and $f(x_i - \delta x)$ values after all of the first derivatives have been calculated by central differences. This means that numerical calculation of the first and second partial derivatives takes two and one function evaluations per derivative, respectively, compared with one and one for analytical partial derivatives. As there are many more second, as compared to first, partial derivatives the difference in compute time between numerical and analytical calculation of partial derivatives is small. Overall, the price of using numerical derivatives is a small increase in compute time and an even smaller decrease in the rate of convergence.

To calculate analytical partial derivatives $\partial V_s/\partial x_i$ and $\partial^2 V_s/\partial x_i \partial x_j$ are expanded by the chain rule to give

$$\frac{\partial V_s}{\partial x_i} = \frac{\partial V_s}{\partial \phi} \cdot \frac{\partial \phi}{\partial x_i}$$

and

$$\frac{\partial^2 V_s}{\partial x_i \partial x_j} = \frac{\partial^2 V_s}{\partial \phi^2} \cdot \frac{\partial \phi}{\partial x_i} \cdot \frac{\partial \phi}{\partial x_j} + \frac{\partial V_s}{\partial \phi} \cdot \frac{\partial^2 \phi}{\partial x_i \partial x_j}$$

where ϕ represents the various geometric parameters l, θ, ω, r, and χ that describe the molecule. Expressions for first and second partial derivatives of V_s with respect to ϕ are easily derived, while the first and second partial derivatives of ϕ with respect to x_i, x_j are only a little more complex if $\phi = l$ or r. However, the derivatives of ϕ with respect to x_i, x_j are not quite so straightforward if $\phi = \theta$, ω, or χ. In these cases it is easier to derive expressions for derivatives of $\cos \phi$ and make use of the relationships

$$\frac{\partial \phi}{\partial x_i} = -\frac{1}{\sin \phi} \cdot \frac{\partial(\cos \phi)}{\partial x_i}$$

and

$$\frac{\partial^2 \phi}{\partial x_i \partial x_j} = -\frac{\cos \phi}{\sin^3 \phi} \cdot \frac{\partial(\cos \phi)}{\partial x_i} \cdot \frac{\partial(\cos \phi)}{\partial x_j} - \frac{1}{\sin \phi} \cdot \frac{\partial^2(\cos \phi)}{\partial x_i \partial x_j}$$

While analytical partial derivatives are more time consuming to formulate than numerical derivatives even a small reduction in compute time is worthwhile given the intensive use and long lifetimes of molecular mechanics programs.

SEQUENTIAL IMPLEMENTATION

The forerunners of VULCAN include Fortran incarnations on sequential microcomputers,[25] minicomputers,[26] and mainframes[27]; as well as an Occam version for parallel computers.[7] The papers documenting the minicomputer[26] and parallel versions[7] describe the workings of the respective programs in some detail and the broad outline of these algorithms has carried over to VULCAN, so that they will not be detailed here. However, a brief outline of VULCAN and the facilities that it offers are included for completeness.

VULCAN is an unconstrained/constrained BDNR minimizer using analytical derivatives. Constraints can be used to set any geometric parameter (atom position, molecule position, bond length, valency angle, torsion angle) to any reasonable value and to drive torsion angles for mapping conformational interconversions. VULCAN does not require the explicit input of lists of bonds, angles, torsion angles, nonbonded contacts, and other geometric variables; these are calculated/inferred from the molecular coordinates, tables of van der Waals radii, etc. There is no limit, other than available memory, to the number of atoms and molecules that can be handled. VULCAN can be invoked with an interactive command line interface, a GUI, and in batch mode; various termination and acceleration procedures are available for use at the discretion of the user. Details of the force-field used are available in the literature.[7,28] The rest of this section details the ways in which VULCAN improves on its predecessors—and perhaps more importantly details some "improvements" which were not particularly effective.

Our original program only accepted input in the form of files generated by our COMMET[29] molecular modelling program, a decision that led to a considerable amount of text editing over the years in converting "foreign" files to our own format. This task was eased by the availability of format conversion programs such as BABEL,[30] which handles a wide range of formats, but unfortunately performs only a vestigial conversion in a number of instances so that text editing is still required. We have therefore written an input routine for VULCAN that handles a (growing) number of common file formats in all of their variations. These formats include COMMET, CHEMMOD, BROOKHAVEN, CSSR(DTMM), ALCHEMY, BALL & STICK, SYBYL, and CIF. The output format can be different from the input format so that the energy minimization program can be used for file conversion as well as energy minimization.

Molecular mechanics calculations are frequently performed on molecular coordinates that result from X-ray crystal structure analyses. If hydrogen atom coordinates are included in the set of molecular coordinates then one always finds that X–H bonds lengths are systematically too short, although the bond vector is pointing in the correct direction.[31] These short bond lengths result in unnecessarily high steric energies at the beginning of an energy minimization. We originally included code in VULCAN to correct the lengths of such bonds simply by elongating the bond vector the necessary amount, before energy minimization, in the expectation that starting from a lower steric energy would allow the BDNR algorithm to converge in fewer iterations. In practice this does not happen; BDNR always fixes bad interactions in the first one or two iterations and the number of iterations required for convergence on the default criterion [$\|\nabla V_s(\mathbf{x})\| < 0.1$ kcal mol^{-1} Å^{-1}] is the same whether the X–H bond lengths are "fixed" or not.

We also tried letting hydrogen atoms "ride" on the medium weight atoms to which they are bonded during the early cycles of a calculation.[32] In other words the coordinate shifts calculated for the medium weight atoms are also applied to the attached hydrogen atoms; only later in the calculation are the hydrogen atoms allowed to move independently. Once again we found that the decrease in program run time to convergence was essentially insignificant. Notice that in both cases ("fixing" and "riding") the picture might have been somewhat different had we employed a slack convergence criterion such as a steric energy decrement from the penultimate iteration.

The precursors of VULCAN maintained interaction lists of lengths, angles, torsion angles, and nonbonded distances for each atom in a multiply redundant format, designed for speed of access. This methodology served its purpose well until we began to perform energy minimization calculations on proteins, when the N-square symmetric byte array used to store nonbonded interactions began to consume significant amounts of memory. An arbitrary element $(i,j) = (j,i)$ of the array contained a code number indicating whether the ith and jth atom were bonded, 1 . . . 3, 1 . . . 4, or 1 . . . 5 and greater with respect to each other. The array was used only for locating nonbonded (i.e., 1 . . . 4 and higher) atom pairs, so that most of the information contained in the array was redundant. Accordingly VULCAN uses an N-square symmetric bit array for the same purpose, where bit(i,j) is set to indicate a 1 . . . 4 or higher nonbonded contact and reset otherwise.

VULCAN uses analytical first and second partial derivatives of the steric energy with respect to the atomic coordinates whereas all previous versions used numerical derivatives as detailed in the previous section. VULCAN runs about 15% faster overall using analytical rather than numerical derivatives, regardless of molecular size. If an FMNR or truncated Newton, rather than BDNR, minimizer were used the advantage of analytical over numerical derivatives would decline asymptotically with increasing molecular size (i.e., as the ratio of the number of second partial derivatives to the number of first partial derivatives increases).

During energy minimization VULCAN moves each atom to its new position as soon as the coordinate corrections have been calculated; in this way each atom "sees" the most up-to-date potential field when the time comes for its coordinate corrections to be calculated. This speeds up convergence at the expense of being able to make use of relationships such as $\partial V_{rij}/\partial x_i = -\partial V_{rij}/\partial x_j$ during calculation of the partial derivatives. Relationships between partial derivatives are used to reduce the amount of computation necessary in the "calculate the entire correction vector and then move all atoms at once" approach. We have found that the reduction in computation time due to the faster convergence

of the "one atom at a time" algorithm outweighs any gains that might be had by utilizing relationships between partial derivatives in the "all atoms at once" methodology.

It is possible for one or more of the 3 × 3 Hessian submatrices to become singular during the course of a BDNR energy minimization, particularly if the calculation is initiated with a molecule that has a poor starting geometry; similarly, the BDNR occasionally becomes ill conditioned (i.e., diverges) during the course of a calculation even though the starting model looked reasonable. In the past we had simply checked to see if the determinant of the submatrix was close to zero and if so switched to steepest descents by setting the inverse of the submatrix to a constant (usually 1×10^{-3}) times the identity matrix. When VULCAN was being implemented we examined the behavior of the Hessian submatrices during known pathological minimizations (e.g., clonidine in Table 1), in detail. Somewhat to our surprise we never found a case where the submatrix was close to zero; it was either massively positive definite or massively nonpositive definite (corresponding to a local energy maximum in the hypersurface for the atom concerned)—nothing in between. This meant that our "switch to steepest descents if the matrix is singular" code was never being executed. We then changed the code so that there was a switch to steepest descents if the determinant of the submatrix was close to zero (just in case) or negative. The pathological minimizations still converged, but much more slowly. This means that the "reciprocals" of the nonpositive definite matrices that we had inadvertently generated, by means of the textbook "inverse from the adjoint" method for small matrices, still contained much useful information and that their continued use was preferable to any other method of proceeding. Notice that the initially nonpositive definite matrices were recomputed as positive definite after a few iterations of BDNR, when the overall molecular geometry had improved significantly, and remained so until convergence. VULCAN therefore uses the original approach of checking only for a zero or close to zero determinant and applying the steepest descents fixup in the unlikely event that this is required (this checking is necessary only because the adjoint matrix is divided by the determinant to obtain the inverse—resulting in a divide by zero runtime error and automatic termination of the program if this condition were not trapped).

As mentioned previously, the BDNR iteration, as well as all other variants of the NR iteration, are less sensitive to the quality of the second, rather than the first, partial derivatives of the steric energy with respect to the atomic coordinates. Indeed, the second partial derivatives change relatively slowly from one iteration to the next. This suggests the possibility of reusing the Hessian for several iterations before updating it. We have found that a strategy of recalculating the Hessian every fourth iteration works effectively for well-conditioned calculations on small to medium-sized (up to about 200 atoms) molecules. This is the default mode of running VULCAN, but can be altered by the user as required. Table 1 indicates the effectiveness of this approach.

The coordinates of the molecules used in Table 1 were obtained as follows: those of cyclohexane, butylamine, and cyclodeca-1,5-diene were generated by the COMMET[29] molecular modeling program; while those of clonidine [2-(2,6-dichlorophenylamino)-2-imidazoline hydrochloride] and crambin were obtained from X-ray crystal structure analyses.[33,34]

The step length, α, in the VULCAN BDNR iteration is set to a user-specified constant, usually 1.0 (the default) for small to medium-sized molecules, and less than 1.0 for large molecules, in order to ensure convergence. An alternative approach is to try a line search along the correction vector calculated by BDNR to find the point of minimum energy, but we have rejected this as too time consuming. However, an examination of the line search procedure led us to the conclusion that BDNR systematically underestimated the magnitude of the correction vector for a well-conditioned calculation on molecules comprising up to about 150 atoms. Accordingly we have implemented a simple but nonetheless effective acceleration technique whereby the calculated correction vector is multiplied by an acceleration factor of between 1.0 and 1.5, the value increasing gradually with iteration number, for those iterations calculated after $\|\nabla V_s(\mathbf{x})\|$ has attained a value of 0.35 kcal mol^{-1} Å^{-1}. Table 1 shows that this approach leads to a worthwhile increase in the rate of convergence for calculations on small and medium-sized molecules. This acceleration technique is ineffective for large molecules, probably because of the considerable amount of fine structure on the steric energy hypersurface in such cases.

Table 1. The steric energy (in kcal mol^{-1}) and norm of the vector $\nabla V_s(x)$ (in kcal mol$_{-1}$ Å^{-1}) for a range of molecules after 50 iterations of BNDR energy minimization[a]

Molecule		NA/H4	NA/H1	A/H4	A/H1
Clonidine	$\|\nabla V_s(\mathbf{x})\|$	Diverges	0.0722	Diverges	0.0355
	V_s	Diverges	10.8386	Diverges	10.8192
Cyclohexane	$\|\nabla V_s(\mathbf{x})\|$	0.0019	0.0019	0.0004	0.0004
	V_s	2.9890	2.9890	2.9890	2.9890
Butylamine	$\|\nabla V_s(\mathbf{x})\|$	0.0115	0.0111	0.0017	0.0009
	V_s	2.3995	2.3995	2.3995	2.3995
Cyclodeca-1,5-diene	$\|\nabla V_s(\mathbf{x})\|$	0.0861	0.0826	0.0251	0.0216
	V_s	14.2624	14.2616	14.2545	14.2545
Crambin	$\|\nabla V_s(\mathbf{x})\|$	1.2550	3.6460	n/a	n/a
	V_s	560.6088	590.8547	n/a	n/a

[a]The Hessian submatrices were recalculated either every iteration (H1) or every fourth iteration (H4) and similarly some calculations used an accelerated minimizer (A) while others did not (NA). n/a, Not applicable.

PARALLEL IMPLEMENTATION

While remote supercomputers are a suitable vehicle for some computational chemistry codes, this remoteness may preclude their effective use in situations where interactive codes are necessary. Laboratory-scale parallel processing appears to offer a suitable price/performance ratio for the large-scale scientific calculations involved in molecular modelling and other interactive simulation techniques. The argument that sustained the popularity of the minicomputer for so long ("it's better to have exclusive use of a small resource rather than shared use of a large resource") is also relevant here. However, until fairly recently technical obstacles with hardware, lack of parallel programming facilities, and high cost ensured that parallel computers remained a development laboratory curiosity.

The widespread use of parallel computers for scientific calculations was catalyzed by the advent of the floating point transputer in 1987.[35] This low-cost, high-performance microprocessor spawned an entire new industry worldwide; and unleashed a storm of competition in hardware and software for parallel computing that both swallowed its progenitor and gave us the low-cost, and infinitely more powerful, parallel computers of today. It is interesting to note that our first use of a parallel computer for molecular mechanics calculations predates these events by several years.[36]

Parallel computers come in several architectures but the multiple instruction multiple data (MIMD) architecture is probably the most popular. There are two variants of MIMD parallel computers; those which all of the processors communicate via a single shared memory, and those in which the memory is distributed among processors that communicate via multiple point-to-point links.

Shared memory machines are popular because their development environments are quite successful at hiding the underlying parallel architecture from the programmer—albiet at the expense of overall efficiency. However, shared memory machines scale poorly because of contention for access to the shared memory, and are typically more expensive per processor than distributed memory machines. It is rare to see conventional shared memory parallel computers with more than 16 processors.

Distributed memory parallel computers, on the other hand, can be scaled up to thousands of processors cost effectively because off-the-shelf hardware is commonly used for everything except the interprocessor links. The processors of distributed memory machines invariably communicate by passing messages to and fro across the interprocessor links. This makes it somewhat harder to insulate the programmer from knowledge of the machine architecture but does allow properly constructed programs to make effective use of the available computing power. For these reasons distributed memory parallel computers are much more common than shared memory machines.

A number of computational chemistry codes have been converted to run on message-passing, distributed memory parallel computers.[37,38] These programs are usually written in Fortran and might use either the emerging standard MPI,[39] the public domain PVM,[40] the commercial Express,[41] or a hardware vendor-specific[42] library of message-passing subroutines for interprocessor communication. While the use of MPI guarantees a considerable degree of program portability this is usually at the expense of performance because MPI is typically implemented on top of an existing, machine-specific subroutine library. Fortunately potability is not yet too much of a problem for computational chemistry codes because almost all of them are implemented with a small number of message-passing subroutines, typically five or six. Most of these key subroutines, implementing functions such as send message and receive message, perform virtually identical operations in all of the libraries and differ only in minor syntax and implementation details. The COMFORT message-passing subroutine library[9] is similar in form and function to those discussed above and only a few of its subroutines are used to implement VULCAN; this makes the program highly portable, while delivering maximum performance because the subroutine library is specifically tailored to support the underlying hardware.

To determine the most effective method for parallelizing the BDNR energy minimization algorithm let us first examine the pseudocode for the sequential algorithm:

```
Begin
  Read in atomic coordinates
  Calculate initial geometry
  Set up bond length lists/arrays
  Set up bond angle lists/arrays
  Set up torsion angle lists/arrays
  Set up non-bonded lists/arrays
  Set up coulombic lists/arrays
  Assign force constants
  Calculate initial energy
  Print out initial geometry and energy
  Do until convergence
   Do for each atom
    Calculate ∇V_s(X)
    Calculate F
    Calculate correction vector = -αF^t
∇V_s(x)
    Add correction vector to current
position vector
   End dc
  End do
  Store final coordinates on disk
  Print out final geometry and energy
End
```

Given that that VULCAN requires access to data stored on disk we could either allow each node processor of the parallel machine to be responsible for its own input/output (i/o), the "hostless" approach; or alternatively one of the nodes could be designated as the host, which would then distribute data, as required, to the nodes, the "host/node" paradigm. We chose to use the latter approach mainly because it is simpler and less error prone to maintain a single copy of key data files, and because the host/node methodology is supported on all message-passing parallel computers. In fact, the host/node organization is not merely restricted to data distribution, the host processor is responsible for loading programs onto the nodes (either from the program cache on the node disk or via an interprocessor link if the program has not been previously cached) and initiating node program execution.

Even a cursory examination of the pseudocode reveals an embarrassing number of possibilities for parallel processing. For instance, the various bond length, bond angle, torsion angle, etc., lists and associated arrays of force constants, etc., can be calculated independently and each calculation could be farmed out to a different node processor. This would be a poor strategy for at least three reasons: one, the calculations involved are so trivial that sending the data necessary for the calculations to the nodes would take (much) longer than calculating all of the lists/arrays on the host; two, it is all very well if there are the same number of lists/arrays and node processors but what happens if, say, there are five lists/arrays and eight nodes? (three nodes sit idle while the other five are calculating); and three, the lists/arrays are all of different sizes and would take different times to calculate so that the nodes that finished first would sit idle while the host waited for the rest of the results to be calculated before it could proceed.

The simple example discussed above illustrates a number of important points to bear in mind when parallelizing sequential programs. Give each node the largest computational task it can handle before it requires communication either with the host or another node; in other words, use as coarse a grain of parallelism as possible. Ensure that the parallelization strategy chosen will scale up efficiently from small numbers to large numbers of nodes. Make sure that, insofar as it is possible, each node has an equal-sized computational task on hand at any one time; in other words, make sure that the computational load on the nodes is balanced. Communicate only as a last resort because communication is slow relative to computation on most distributed memory parallel computers; it is frequently faster to perform multiple redundant calculations of data on each node rather than have the host calculate the data and distribute it to the nodes.

The desire for coarse-grained parallelism leads naturally to a focus on where the program spends most of its time. In the case of gradient energy minimization programs this means the contribution to $\nabla V_s(\mathbf{x})$ and the Hessian from the nonbonded interactions. One approach might therefore be to allow the host to deal with program setup and the contribution to $\nabla V_s(\mathbf{x})$ and the Hessian from bond lengths, bond angles, torsion angles, and out-of-plane bending, while calculation of the contribution to $\nabla V_s(\mathbf{x})$ and the Hessian from the nonbonded interactions is equally distributed among the nodes (assumed to be of identical computational power). Programs based on this approach have been described in the literature[38] but suffer from two relatively minor drawbacks. First, the host and node programs are quite different, which means extra coding effort when the program is first developed but is irrelevant thereafter; and second, the load balancing is not optimum because the host will not usually finish its calculation at the same time as the nodes, leading to wasted CPU cycles on either the host or the nodes.

A strategy that avoids these problems consists of running essentially the same code on the host (the host still must set up the nodes, calculate initial data, and prime the nodes with data) and nodes while dividing the calculation equally between them. This can be achieved by making the host and nodes each responsible for a "slice" of the total number of atoms in the molecule, where the slices are as equal in size as possible. This is identical to the approach used in our Occam parallel BDNR minimizer' and other similar Fortran programs.[37] Notice that COMFORT allows for a host and nodes of varying computational power by assigning the host or node a "slice" of atoms whose size is directly proportional to the power of the processor. The load balancing of this overall approach is not perfect, because each host and node processor will generate slightly different-sized interaction lists depending on the bonding pattern of the atoms in their slice. However, the load balancing is sufficiently close to optimum that any attempt to correct it would undoubtedly do more harm than good. Notice that this latter statement is only true if nonbonded pair list generation is performed on an atom-to-atom basis. Some programs designed for use with polypeptides generate pair lists at the amino acid residue level, so that if any two atoms from two different residues are within the cutoff distance for nonbonded interactions then all of the atoms in each residue are considered to interact with all of those of the other (even though some individual distances will be greater than the cutoff distance). In this case the load imbalance would become a problem and a secondary redistribution of nonbonded pairwise interactions between the host and nodes is sensible.[37]

The host and node pseudocode for VULCAN is as follows:

Host:

```
Begin
  Initialize parallel machine
  Load node programs
  Read in atomic coordinates & initial
data
  Broadcast subset of initial data to
nodes
  Calculate initial geometry
  Set up bond length lists/arrays
  Set up bond angle lists/arrays
  Set up torsion angle lists/arrays
  Set up non-bonded lists/arrays
  Set up coulombic lists/arrays
  Assign force constants
  Calculate initial energy
  Print out initial geometry and energy
  Do until convergence
   Broadcast complete set of atomic
coordinates to nodes
   Do for each atom of host slice
    Calculate ∇V_s(x)
    Calculate F
    Calculate correction vector =
    -αF^+∇V_s(x)
    Add correction vector to current
    position vector
   End do
   Get slice of corrected coordinates
   from each node
  End do
  Store final coordinates on disk
  Print out final geometry and energy
End
```

Node:

```
Begin
```

```
Do until convergence
  Get complete set of atomic
  coordinates from host
 If first iteration
  Calculate initial geometry
  Set up bond length lists/arrays
  Set up bond angle lists/arrays
  Set up torsion angle lists/arrays
  Set up non-bonded lists/arrays
  Set up coulombic lists/arrays
  Assign force constants
 End if
  Do for each atom of node slice
  Calculate ∇V_s(x)
  Calculate F
  Calculate correction vector =
  -F'∇V_s(x)
  Add correction vector to current
  position vector
  End do
  Send slice of corrected coordinates
to host
 End do
End
```

The differences between the host parallel and sequential pseudocodes, presented earlier, are small and encompass the initialization of the Parallel PC, loading code onto the node processors, broadcasting a subset of the initial data to the nodes, broadcasting coordinates to the nodes, retrieving slices of corrected coordinates from the nodes, and the fact that the host is responsible for a slice of coordinates rather than the whole set as in the sequential case. These differences are reflected by 60–70 extra lines of Fortran in the host parallel version of VULCAN compared with the purely sequential version (in fact, there is only one version of VULCAN that runs on both sequential, if the program cannot detect any node processors, and parallel computers).

The node code for the Parallel PC is simply a cut-down version of the host code in which all of the initialization, program loading, and Fortran i/o has been removed. Notice that on the first iteration each node calculates the interaction lists and associated data it will need for subsequent iterations; this is faster than having the host calculate these lists and broadcasting them to the nodes. Notice also that parallelizing VULCAN has introduced a subtle difference from the purely sequential code, in that when the coordinate corrections are calculated for each atom they are done so in the light of the most up-to-date potential field defined by the positions of the atoms in a particular node's slice, and do not see the improved potential resulting from new atomic positions calculated by other nodes until the global update of coordinates at the beginning of each subsequent iteration. This has the effect of slowing convergence for the parallel code by a small amount compared with the sequential version, the exact amount of degradation tending to rise with decreasing numbers of atoms in the slice assigned to each node and with decreasing locality of the atoms within these slices. This means that it is possible to alter the rate of convergence slightly by rearranging the order in which atoms are input to VULCAN.

Performance figures for an eight-node Parallel PC, with 1, 2, 4, and 8 nodes active, are given in Table 2. The host processor was a 100-MHz AMD DX4/100 with 16 MB of memory and each node consisted of an 80-MHz AMD DX2/80 with 4 MB of memory at the time of writing. Notice that we have used parallel efficiency rather than speedup as a measure of effectiveness because speedup is not particularly meaningful when the host and/or nodes differ in computational power. The coordinates for ubiquitin were obtained from an X-ray crystal structure analysis.[43] Performance figures are given only for large molecules because the run times of calculations on small to medium-sized molecules, using a single processor, are so short (on the order of seconds to tens of seconds) that it is not worth using the Parallel PC in these cases. Remember, too, that Hessian reuse and step length acceleration will speed up sequential calculations on small and medium-sized molecules but are ineffective with large molecules. Nevertheless, parallel efficiencies are still high down to the 50-atom molecule level; beyond this point the communication time becomes an increasingly large proportion of the overall program run time, thereby depressing parallel efficiency (e.g., parallel efficiency is 50% for a 27-atom molecule and 8 processors).

Table 2. Energy minimization run times and parallel efficiences for 250-iteration calculations on crambin and ubiquitin as a function of the number of processors used for the calculation

Number of processors	Crambin (327 atoms)		Ubiquitin (602 atoms)	
	Time (sec)	Efficiency (%)	Time (sec)	Efficiency (5)
Host	788.73	100.0	1776.19	100.0
Host + 1 node	447.26	97.97	1013.43	97.37
Host + 2 nodes	316.70	95.78	721.72	94.66
Host + 4 nodes	204.38	91.88	456.76	92.59
Host + 6 nodes	155.00	87.73	336.36	91.05
Host + 8 nodes	123.51	86.30	272.45	88.10

The parallel efficiency figures are good and somewhat higher than those typically quoted for similar calculations on other machines.[38] There are good reasons for this; first, the host and all of the nodes are run in a single tasking environment so that there is no task swapping overhead; second, all communications are blocking and unbuffered so that there is no buffer management overhead; third, broadcasting is handled in hardware rather than software; and finally there is no switching or routing overhead. In other words, our parallel PC is as simple as possible without compromising performance or ease of programming. The amount of time spent in communication inevitably increases with the number of nodes, hence the gentle decline in parallel efficiency as more nodes are utilized. This effect can be reduced, but not entirely eliminated, by using faster host-to-node interprocessor links and/or overlapping communication and computation rather than using the synchronous

methodology described above. We are currently investigating several possible methods of implementing asynchronous communication in VULCAN.

CONCLUSION

We have described and justified our molecular mechanics methodology, discussed various procedures for enhancing/accelerating energy minimization procedures, and presented details of practical implementations of these algorithms in both sequential and parallel Fortran programs. For the future we intend to combine both sequential and parallel molecular mechanics and dynamics calculations into a single Fortran program with an improved force field. In addition, our Parallel PC will continue to be upgraded as new 80 × 86 compatible processors and faster interprocessor link chips become available.

REFERENCES

1 White, D.N.J., and Sim, G.A. The santonins: Quantitative conformational analysis. *Tetrahedron.* 1973, **29,** 3933

2 White, D.N.J. Molecular mechanics calculations: Chemical Society specialist periodical reports. *Mol. Struct. Diffr. Methods.* 1978, **6,** 38

3 White, D.N.J. and Guy, M.H.P. The molecular conformation of cyclodi-βalanyl. *J.C.S. Perkin.* 1975, **II,** 43

4 White, D.N.J. and Bovill, M.J. Molecular mechanics calculations on Alkanes and Non-conjugated Alkenes. *J.C.S. Perkin.* 1977, **II,** 1610

5 White, D.N.J. The principles and practice of molecular mechanics calculations. *Computr. Chem.* 1977, **1,** 225

6 White, D.N.J. and Morrow, C. Cyclic tetrapeptides I: The calculated potential energy minima of cyclic tetrapeptides composed of small amino-acid residues. *Computr. Chem.* 1979, , 33

7 White, D.N.J., Ruddock, J.N., and Edgington, P.R. Molecular design with transparallel supercomputers. *Mol. Simul.* 1989, **3,** 71–100

8 White, D.N.J. A hardware and software environment for parallel processing with PCs. *Comput. Chem.* 1996, **20,** 381–384

9 Bissland, L. and White, D. N.J. Parallel molecular mechanics calculations. In: *Transputer Applications and Systems '95,* (Cook, M., et al., eds.) IOS Press Oxford, 1995, pp. 473–487

10 White, D.N.J. and Pearson, J.E. A comprehensive molecular modelling system. *J. Mol. Graphics.* 1986, **4,** 134

11 White, D.N.J. and Morrow, C. The global minimum energy conformation of cyclotetraglycyl. *Tetrahedron Lett.* 1977, 3385

12 White, D.N.J. and Kitson, D.H. Computational conformational analysis of cyclohexaglycyl. *J. Mol. Graphics* 1986, **4,** 112

13 Momany, F.A., McGuire, R.F., Burgess, A.W., and Scheraga, H.A. Energy parameters in polypeptides. VII. Geometric parameters, partial atomic charges, nonbonded interactions, hydrogen bonded interactions, and intrinsic torsional potentials for the naturally occurring amino acids. *J. Phys. Chem.* 1975, **79,** 2361–2381

14 Ermer, O. Determination of molecular symmetries by force field calculations and evaluation of symmetric and nonsymmetric conformational transition states avoiding complete point to point mapping. *Tetrahedron,* 1975, **31,** 1849

15 Murray, W. (ed.). *Numerical Methods for Unconstrained Optimization.* Academic Press, London 1972, p 107

16 White, D.N.J. and Ermer, O. Molecular mechanics—A cautionary note. *Chem. Phys. Lett.* 1975, **31,** 111

17 Weiner, P.K. and Kollman, P.A. AMBER: Assisted model building with energy refinement. A general program for modelling molecules and their interactions. *J. Comput. Chem.* 1981, **2,** 287–303

18 Vinter, J.G., Davis, A., and Saunders, M.R. The COSMIC molecular mechanics force field. *J. Comput.-Aided Mol. Design.* 1987, **1,** 31

19 Niketić, S.R. and Rasmussen, K. The consistent force field: A documentation. *Lecture Notes in Chemistry.* (Berthier, G., et al., eds.), Vol. 3. Springer-Verlag, Berlin, 1977

20 White, D.N.J. Further notes on the generalized inverse. *Acta Crystallogr.* 1977, **A33,** 1010

21 Ponder, J.W. and Richards, F.M. An efficient Newton-like method for molecular mechanics energy minimization of large molecules. *J. Comput. Chem.* 1987, **8,** 1016–1024

22 Brooks, B.R., Bruccoleri, R.E., Olafson, B.D., States, D.J., Swaninathan, S., and Karplus, M. CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *J. Comput. Chem.* 1983, **4,** 187–217

23 Allinger, N.L. and Lane, G.A. Conformational transmission: A quantitative approach to rates of benzylidene formation in steroidal 3-ketones. *J. Am. Chem. Soc.* 1974, **96,** 2937–2941

24 Busing, W.R. *WMIN—A Program to Represent Crystals or Molecules by a Potential Energy Model, Adjust Parameters of the Potential, Adjust the Structure for Minimum Energy, or Calculate Its Vibrational Frequencies.* Oak Ridge National Laboratory, Oak Ridge, Tennessee, 1984

25 White, D.N.J., Lindley, M.R., and Tyler, J.K. High performance microcomputer molecular modelling. *Computr. Chem.* 1986, **10,** 193

26 White, D.N.J. and Morrow, C. The Glasgow University Chemical Graphics System I. In: *Proceedings DECUS UK Conference.* Digital Equipment Company, Maynard, Massachusetts, 1978, p 17

27 Kitson, D.H. A Conformational Study of Some Cyclic Peptides. Ph.D. Thesis. Glasgow University, Glasgow, Scotland, 1983, pp 19–48, 212–241

28 White, D.N.J., Ruddock, J.N., and Edgington, P.R. Molecular mechanics. In: *Computer-Aided Molecular Design.* (Richards, W.G., ed.). IBC Technical Services, London, 1989, pp 23–41

29 White, D.N.J. and Ruddock, J.N. COMMET—*A COncurrent Molecular Modelling Environment for Transparallel Computers.* University of Glasgow, Glasgow, Scotland, 1988

30 Walters, P., and Stahl, M. *BABEL—A File Conversion Utility.* University of Arizona, Tucson, Arizona, 1994
31 Dunitz, J.D. *X-Ray Analysis and the Structure of Organic Molecules.* Cornell University Press, London, 1979, p 391
32 Allinger, N.L., Yuh, Y.H., and Lii, J.-H. Molecular mechanics: The MM3 force field for hydrocarbons. *J. Am. Chem. Soc.* 1989, **111,** 8551–8576
33 Cody, V. and DeTitta, G.T. Crystal and molecular structure of clonidine hydrochloride. *J. Cryst. Mol. Struct.* 1980, **9,** 33
34 Hendrickson, W.A. and Teeter, M.M. Structure of the hydrophobic protein crambin determined directly from the anomalous scattering of sulphur. *Nature (London).* 1981, **290,** 107
35 Fuge, T. The floating point transputer—IMS T800. *Electron Prod. Design.* 1987, **8,** 33–36
36 White, D.N.J. A micro vector processor for molecular mechanics calculations. In: *ACS Symposium Series, No. 173: Supercomputers in Chemistry.* American Chemical Society, Washington D.C., 1981, p 193
37 Vincent, J.J. Mertz, K.M., Jr. A highly portable parallel implementation of AMBER4 using the message passing interface standard. *J. Comput. Chem.* 1995, **16,** 1420
38 Swanson, E. and Lybrand, T.P. PVM-AMBER: A parallel implementation of the AMBER molecular mechanics package for workstation clusters. *J. Comput. Chem.* 1995, **16,** 1131
39 *MPI: A Message Passing Interface Standard.* University of Tennessee, Knoxville, Tennessee, 1994
40 Dongarra, J., Geist, G.A., Mancheck, R., and Sunderam, V.S. PVM—Parallel virtual machine. *Comput. Phys.* 1993, **7,** 166
41 Parasoft Corp. *Express—3L Fortran, User's Guide and Reference.* Parasoft Corp., Pasadena, California, 1990
42 For example, Intel Supercomputer Systems Division. *iPSC/2 & iPSC/860 Programmers Reference Manual.* Intel Supercomputer Systems Division, Beaverton, Oregon, 1991
43 Vijay-Kumar, S., Bugg, C.E., Cook, W.J. The structure of ubiquitin refined at 1.8Å. *J. Mol. Biol.* 1987, **194,** 531