Geometry optimization in delocalized internal coordinates: An efficient quadratically scaling algorithm for large molecules

Jon Baker, Don Kinghorn, and Peter Pulay

Citation: The Journal of Chemical Physics **110**, 4986 (1999); doi: 10.1063/1.478397
View online: http://dx.doi.org/10.1063/1.478397
View Table of Contents: http://scitation.aip.org/content/aip/journal/jcp/110/11?ver=pdfcov
Published by the AIP Publishing

JOURNAL OF CHEMICAL PHYSICS VOLUME 110, NUMBER 11 15 MARCH 1999

Geometry optimization in delocalized internal coordinates: An efficient quadratically scaling algorithm for large molecules

Jon Baker,[a)] Don Kinghorn,[a,b)] and Peter Pulay[a)]
Department of Chemistry and Biochemistry, The University of Arkansas, Fayetteville, Arkansas 72701

(Received 29 September 1998; accepted 11 December 1998)

Using a Z-matrix-like approach for generating new Cartesian coordinates from a new geometry defined in terms of delocalized internal coordinates, we eliminate the costly O(N^3) iterative back-transformation required in standard geometry optimizations using delocalized (or natural/redundant) internals, replacing it with a procedure which is only O(N). By replacing the gradient transformation with an iterative solution of a set of linear equations, we also reduce this step from O(N^3) to roughly O(N^2). This allows a very efficient method for geometry optimization of large molecules in internal coordinates. Several optimizations on systems containing up to 500 atoms are presented, comparing the performance of the new algorithm with its predecessor, and demonstrating the practical utility and efficiency of our approach. © *1999 American Institute of Physics.* [S0021-9606(99)30211-7]

I. INTRODUCTION

Geometry optimization using natural,[1] redundant,[2] or delocalized[3] internal coordinates is now well established. Optimizations using these coordinates are normally far more efficient, in terms of the number of cycles required to reach convergence, than the corresponding optimization using Cartesian coordinates, even if good quality Hessian (second derivative) data are available.[3] A reduction in the number of optimization cycles by an order of magnitude or more compared to Cartesians can frequently be achieved for large molecules.

There are some disadvantages in using internal coordinates, however. Gradients and possibly Hessians, which are originally calculated in Cartesian coordinates, have to be transformed into the corresponding internal coordinate quantities, and this involves construction and inversion of the transformation matrix (the Wilson B-Matrix[4]) together with a number of matrix multiplications. Perhaps the single most time consuming component of an optimization cycle in internal coordinates is the iterative back-transformation whereby a new geometry in internal coordinates is converted back into Cartesians in order to calculate the energy and gradient for the next optimization cycle. This step is roughly cubic, O(N^3), in terms of CPU time, where N is the number of primitive internal coordinates, and becomes dominant as the system size increases.

Until quite recently, the majority of optimizations carried out using redundant internal coordinates were done at the *ab initio* level. Here it mattered little how much time was spent in the optimization algorithm as virtually all the CPU cycles were taken in calculating the energy and gradient. The same holds true, but to a lesser degree, with semi-empirical methods. The situation is different for molecular mechanics and for the increasingly popular hybrid QM/MM methods, in which a small part of the system is calculated using *ab initio* techniques and the rest via molecular mechanics. For these methods, energy and gradient computation is rapid, and optimization in internal coordinates may take longer in total than in Cartesians, despite the potentially large reduction in optimization cycles, because optimization is now dominated by the time needed to do the coordinate and gradient transformations, especially the iterative back-transformation.

Efforts are being made to reduce the impact of these transformations. For example, it is possible to take advantage of the relative sparsity of the B-matrix (which has at most 12 entries per column if primitive internal coordinates are used) and use sparse-matrix methods to solve a series of linear equations instead of carrying out a full matrix inversion. We have been using the sparsity of the B-matrix in geometry optimization for some time (cf the discussion in Ref. 5), and have recently developed a complementary sparse matrix inversion method which permits the extension of internal coordinate geometry optimization to large organic molecules.[6] Other groups are also working on similar approaches.[7]

One way of avoiding the time consuming iterative back-transformation and still carry out the optimization in internal coordinates is to use a Z-matrix. Because it is constructed using *individual* primitive internals, and not the linear combinations of primitives that make up each delocalized internal coordinate, Cartesian coordinates can be generated from Z-matrix internal coordinates by simple trigonometry in a one-step noniterative process *without* having to construct and invert a B-matrix. Unfortunately, constructing a good Z-matrix is still somewhat of an art form, and optimizations in Z-matrix coordinates are generally much less efficient than those in redundant internals; for rigid, symmetrical cyclic systems Z-matrix optimizations are often worse than Cartesians.[8]

In this article we present a hybrid delocalized internal

[a)]Author emails: baker@uafchem1.uark.edu; kinghorn@u.arizona.edu; pulay@comp.uark.edu
[b)]Current address: Department of Chemistry, University of Arizona, 1306 E. University, Tucson, AZ 85721.

Z-matrix approach in which a Z-matrix is constructed and used to convert the new geometry in internal coordinates into Cartesians, while determining the actual optimization step in delocalized internals. In this way, a full optimization in highly efficient delocalized internals is carried out while at the same time eliminating the time consuming back-transformation, replacing it with a simple trigonometric Z-matrix to Cartesian conversion. We also replace the gradient transformation step by the solution of a system of linear equations, similar to, but somewhat more efficient than, our previous scheme.[6]

In the following section (Sec. II), we summarize the theoretical background to the general internal coordinate scheme used in this work, with emphasis on the back transformation. In Sec. III we carry out several optimizations on a number of systems, including fairly large (up to 500+ atoms) alanine polypeptides, using the semi-empirical AM1 method[9] and the original SYBYL force field,[10] comparing the performance of delocalized internal coordinates using the new hybrid Z-matrix back-transformation with the original algorithm (using the full iterative back-transformation) and with Cartesians. Section IV comprises a summary and conclusions.

II. THEORETICAL BACKGROUND

The background to optimization in redundant internal coordinates is now well established, and is contained in the 1992 paper of Pulay and Fogarasi[2] and the later work of Baker *et al.*[3] The geometry of the system is described using a set of n, in general redundant, primitive internal coordinates $\mathbf{q}=\{q_1,q_2,\ldots,q_n\}^T$. Typically $\mathbf{q}$ will consist of all stretches, all bends, and all proper torsions that can be derived based on the atomic connectivity. Small displacements $\Delta\mathbf{q}$ in $\mathbf{q}$ are related to the corresponding Cartesian displacements, $\Delta\mathbf{x}$, by means of the well-known B-matrix[4]

$$\Delta\mathbf{q}=\mathbf{B^q}\Delta\mathbf{x}. \tag{1}$$

Diagonalization of the $n\times n$ symmetric matrix $\mathbf{G}=\mathbf{B^q}(\mathbf{B^q})^\mathbf{T}$ results in two sets of eigenvectors; a set of $m=3N\text{-}6$ (where N is the number of atoms) *nonredundant* eigenvectors with eigenvalues $\lambda>0$ and a set of $n\text{-}m$ *redundant* eigenvectors with eigenvalues $\lambda=0$. The eigenvalue equation for $\mathbf{G}$ can be written

$$\mathbf{G}(\mathbf{UR})=(\mathbf{U}\;\;\mathbf{R})\begin{pmatrix}\mathbf{\Lambda} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{pmatrix}, \tag{2}$$

where $\mathbf{U}$ is the set of nonredundant eigenvectors of $\mathbf{G}$ and $\mathbf{R}$ is the redundant set. In this way there is a natural separation of the redundant and nonredundant subspaces of our original coordinate space; the former can be disregarded when calculating the optimization step.

The $3N\text{-}6$ nonredundant vectors in $\mathbf{U}$ can be considered as defining a transformation matrix to a set of *delocalized internal coordinates*, $\mathbf{s}$, each delocalized internal being potentially a linear combination of *all* the primitives in the underlying coordinate space, $\mathbf{q}$:[3]

$$\mathbf{s}=\mathbf{U^Tq}. \tag{3}$$

They are automatically symmetry adapted, and in symmetrical molecules only the totally symmetric ones need to be retained, although this is seldom important for large systems.

Displacements in delocalized internal coordinates are related to Cartesian displacement coordinates by

$$\Delta\mathbf{s}=\mathbf{U^TB^q}\Delta\mathbf{x}=\mathbf{B}\Delta\mathbf{x}, \tag{4}$$

where

$$\mathbf{B}=\mathbf{U^TB^q} \tag{5}$$

is the B-matrix for the delocalized internal coordinates. The left Morse–Penrose inverse of the transposed B-matrix is then constructed

$$(\mathbf{B^T})^{-1}=(\mathbf{BB^T})^{-1}\mathbf{B} \tag{6}$$

and used to transform the gradient vector $\mathbf{g}$ from Cartesian to internal coordinates in the usual way[11]

$$\mathbf{g}^{\text{int}}=(\mathbf{B^T})^{-1}\mathbf{g}^{\text{cart}}. \tag{7}$$

Having transformed the gradient from Cartesians to internals [Eq. (7)] and calculated a new step and a new geometry in internal coordinates, $\mathbf{s}$, we need to transform $\mathbf{s}$ back into Cartesian coordinates, $\mathbf{x}$, in order to calculate a new energy and gradient for the next optimization step. This is the back-transformation step, which—due to the nonlinearity of the transformation between internal and Cartesian coordinates—is usually accomplished via the iterative formula

$$\mathbf{x}(k+1)=\mathbf{x}(k)+(\mathbf{B^T})^{-1}(k)[\mathbf{s}-\mathbf{s}(k)]. \tag{8}$$

Here $\mathbf{s}$ is the (known) new geometry in internal coordinates and the iterative procedure is typically started ($k=1$) using the *old* Cartesian geometry (and the old B-matrix). The procedure is terminated when the internal coordinates $\mathbf{s}(k)$ generated on the kth iteration from the Cartesian set $\mathbf{x}(k)$ are identical to $\mathbf{s}$ within a given tolerance. Although this iterative back transformation normally converges extremely rapidly [differences between $\mathbf{s}$ and $\mathbf{s}(k)$ of $<10^{-10}$ in 3–5 cycles], it formally requires the construction and inversion of a new B-matrix at *each* iterative cycle. A better scheme, which works unless the displacements are very large, is to keep the old $(\mathbf{B}^T)^{-1}$-matrix throughout; this increases the number of cycles but each cycle is now much less expensive as no matrix inverse is needed. In either case, the back transformation is costly and it is often the dominant step in the optimization of large systems.

As mentioned in Sec. I, one way of avoiding the iterative back-transformation is to carry out the optimization in Z-matrix coordinates. In a Z-matrix, each new atom after the third is defined with respect to atoms previously defined in the Z-matrix using, for example, one stretch, one bend, and one torsion. The first three atoms in the Z-matrix are unique, with the first atom at the origin, the second atom lying on the Z-axis (connected to the first by a single stretch), and the third lying in the *XZ*-plane (connected to either the first or second atom via a stretch and defining a bend with the unconnected atom). The geometry is thus built up in a treelike manner, and it is relatively straightforward to generate Cartesian coordinates from a given Z-matrix. The time required to do this is O(N), i.e., it increases linearly with system size, an important consideration for large molecules.

What we are proposing in this work is to generate an intermediate Z-matrix *solely* for the purpose of transforming the geometry in internals to Cartesians. The actual optimization will still be carried out in delocalized internals, the Z-matrix serving only to eliminate the expensive iterative back-transformation. The basic strategy is to calculate, from the changes in the delocalized internal coordinates, the corresponding changes in the primitive internals. A subset of the latter can then be used as input to the Z-matrix geometry builder to generate the new Cartesians in an O(N) step. This procedure avoids the O(N^3) construction of the inverse B-matrix [Eq. (6)] and several matrix multiplications.

The changes in the primitive internals, $\Delta\mathbf{q}$, are formally given by

$$\Delta\mathbf{q}=(\mathbf{U}\ \ \mathbf{R})(\Delta\mathbf{s})^T \\ (\Delta\mathbf{r})^T, \tag{9a}$$

where $\Delta\mathbf{r}$ is the change in the redundant coordinates. As the latter cannot change if the redundancy condition is exact, $\Delta\mathbf{r}$ vanishes, and so to first order

$$\Delta\mathbf{q}=\mathbf{U}(\Delta\mathbf{s})^T. \tag{9b}$$

Equation (9b) is not exact because the redundancy condition depends on the geometry, and in practice the two subspaces **R** and **U** will mix somewhat as the optimization progresses.[3] Therefore Eq. (9b) must be invoked iteratively until the changes in the primitives satisfy $\Delta\mathbf{s}=\mathbf{U^T}\Delta\mathbf{q}$ to the desired precision.

With Eq. (9b) defining the *change* in the primitives during an optimization step, the new primitive values after taking the step are given by $\mathbf{q}+\Delta\mathbf{q}$. Now given a complete set of primitives, it is relatively straightforward to select a subset of these primitives suitable for forming a Z-matrix. New Cartesians can then be derived directly from this Z-matrix, avoiding the iterative back-transformation altogether. We use a construction scheme that builds up the Z-matrix from the primitive torsions, using a spanning three algorithm to eliminate redundant bonds.[12] Once the *form* of the Z-matrix (the particular stretches, bends, and torsions that comprise it) has been determined, in subsequent optimization cycles it is simply a matter of plugging in the appropriate values from the primitive list.

As mentioned above, the primitive coordinates obtained from a *single* application of Eq. (9b) are not exact. However, in a similar manner to the original back-transformation, we can iterate to the exact solution

$$\mathbf{q}(k+1)=\mathbf{q}(k)+\mathbf{U}[\mathbf{s}^T-\mathbf{s}(k)^T]. \tag{10}$$

From the current estimate for **q**, we obtain Cartesian coordinates **x** using the Z-matrix back-transformation. The values **q** at the Cartesian geometry **x** are found, and used to derive new values for the internal coordinates $\mathbf{s}=\mathbf{U^T}\mathbf{q}$. These are then compared with the known **s**, and the difference used to obtain an improved estimate **q** using Eq. (10). A new set of Cartesians are then derived from the new **q**. As is the case with the original back transformation [Eq. (8)], convergence is usually rapid, with differences between **s** and $\mathbf{s}(k)$ of $<10^{-10}$ in 3–7 cycles.

Using the above Z-matrix back-transformation replaces the original O(N^3) back-transformation with a procedure which is O(N). There is no B-matrix construction and, in particular, no matrix inverse. We have tested our new Z-matrix back-transformation on a number of medium sized systems (containing up to 100 atoms) using semi-empirical wave functions with complete success. As an example, the CPU time required for the back-transformation step in the optimization of jawsamycin (84 atoms; see Ref. 3) in delocalized internal coordinates is reduced by over two orders of magnitude using our new Z-matrix scheme compared to the original full back-transformation of Eq. (8).

Elimination of the O(N^3) back-transformation is a vital step in applying redundant internal coordinates to the geometry optimization of large systems (containing hundreds of atoms) using semi-empirical or QM/MM force fields. Note that we cannot avoid the equivalent of at least one B-matrix construction (and inversion) per cycle in order to transform the Cartesian gradient to internal coordinates [Eq. (7)], however, this is replaced by the iterative solution of a system of linear equations which scales as O(N^2) asymptotically.

Following Ref. 6, we reformulate the gradient transformation [Eq. (5)] as

$$(\mathbf{BB^T})\mathbf{g}^{\text{int}}=\mathbf{B}\mathbf{g}^{\text{cart}}. \tag{11}$$

Unlike the original B-matrix in *primitive* internals, the B-matrix in terms of delocalized internals is not at all sparse; however, it has a significantly smaller dimension (at most $3N$-6, less if there is symmetry) and, certainly in the early stages of the optimization, the G-matrix ($\mathbf{BB^T}$) is *diagonally dominant*. [Indeed, on the first optimization cycle, it *is* diagonal, its diagonal elements being the positive eigenvalues, λ, of Eq. (2).] It does not stay diagonal, because the vector spaces **U** and **R** mix as the optimization progresses, but it should stay diagonally dominant. This is an important difference relative to the use of natural or redundant primitive coordinates[6] where the G-matrix is typically only weakly or not at all diagonally dominant.

Splitting off the diagonal **D** of $\mathbf{BB^T}$, we can rewrite Eq. (9) as

$$\mathbf{D}\mathbf{g}^{\text{int}}=\mathbf{B}\mathbf{g}^{\text{cart}}+(\mathbf{D}-\mathbf{BB^T})\mathbf{g}^{\text{int}} \tag{12}$$

and convert it into an iterative scheme

$$\mathbf{g}^{\text{int}}(k+1)=\mathbf{D}^{-1}[\mathbf{B}\mathbf{g}^{\text{cart}}+(\mathbf{D}-\mathbf{BB^T})\mathbf{g}^{\text{int}}(k)]. \tag{13}$$

Equation (13) is solved by the conjugate gradient method,[6] and given the diagonally dominant nature of $\mathbf{BB^T}$, convergence should be fairly rapid.

The O(N) back-transformation combined with a roughly O(N^2) transformation of Cartesian gradients to internals allows the *efficient* use of internal coordinates in large semi-empirical and QM/MM optimizations. In the following sec-

TABLE I. Timings (CPU; s) on an IBM RS6000/390 workstation for various steps in the internal-coordinate large-molecule optimization algorithm.

	B-matrix[b] construction		Gradient[c] transformation		Optimization[d]	Back[c]-transformation		Total time	
Cycle[a]	old	new	old	new	step	old	new	old	new
Jawsamycin ($C_{32}H_{43}N_3O_6$, 84 atoms)									
1	34.01	12.61	2.01	0.03	0.85	16.54	0.12	53.41	13.61
2	2.09	0.27	2.03	0.10	0.02	16.54	0.14	20.68	0.53
3	2.12	0.27	2.02	0.12	0.02	16.54	0.17	20.70	0.58
30	2.12	0.27	2.01	0.55	0.02	12.37	0.23	16.52	1.07
Taxol ($C_{47}H_{51}NO_{14}$, 113 atoms)									
1	110.06	23.09	4.46	0.05	1.53	36.19	0.23	152.24	24.90
2	4.57	1.45	4.41	0.22	0.04	36.15	0.25	45.17	1.96
3	4.58	1.45	4.41	0.22	0.04	36.15	0.25	45.18	1.96
68	4.59	1.44	4.46	1.36	0.04	27.10	0.98	36.19	3.82
$(\text{Alanine})_{20}$ ($C_{60}H_{102}N_{20}O_{21}$, 203 atoms)									
1	826.07	97.25	99.91	0.29	20.52	492.97	0.42	1439.47	118.48
2	23.71	9.30	99.96	1.07	0.18	626.28	0.51	750.13	11.06
3	23.80	9.27	100.01	1.42	0.18	620.58	0.51	744.57	11.38
63	23.80	9.29	100.37	2.50	0.18	372.89	0.58	497.24	12.55

[a]Shown are timings for the first three optimization cycles and the final cycle. Convergence criteria are: maximum gradient component $<0.000\,05E_h/a_0$; energy change $<10^{-7}E_h$.
[b]Timings for B-matrix construction on cycle 1 include once only construction of **U**.
[c]The fast gradient and back-transformation were converged to an accuracy of better than 5×10^{-9}.
[d]Timings for optimization step on cycle 1 include once only construction of $\mathbf{H}^{-1}$.

tion we present a simple quasi-Newton-like algorithm for geometry optimization using delocalized internal coordinates and use it to optimize the structures of a number of larger molecules including alanine polypeptides containing over 500 atoms.

III. THE ALGORITHM

For test purposes, we have adopted a straightforward quasi-Newton algorithm with a direct BFGS update of the inverse Hessian matrix.[13] The steps involved in each cycle of our internal-coordinate large-molecule optimization algorithm are as follows:

step 01 (once only)

Construct the primitive $\mathbf{B^q}$ *matrix, form* $\mathbf{G}=\mathbf{B^q}(\mathbf{B^q})^\mathbf{T}$, *and solve the eigenvalue equation [Eq. (2)] to give the transformation matrix,* **U**, from primitive to delocalized internal coordinates.

This is potentially *the* most time consuming step, as it involves the equivalent of an $O(N^3)$-matrix diagonalization. However, it is done *once only*, at the initial geometry, and fixes the transformation matrix for the rest of the optimization. Formerly, the full $\mathbf{B^q}$ matrix was constructed, **G** was formed by a full matrix–matrix multiply, and a complete diagonalization was carried out to give both the **U** and **R** vector spaces. In the current algorithm, only the nonzero elements of $\mathbf{B^q}$ are stored, **G** is formed by a sparse matrix multiply and only the highest $3N$-6 roots are found (i.e., only the nonredundant vectors **U**) using the LAPACK routine dspevx.[14]

step 02 (once only)

Estimate the initial inverse Hessian matrix.

This is done by first estimating the diagonal force constants in the *primitive* basis, transforming to delocalized internal coordinates (for details, see Ref. 3) and directly inverting the resultant Hessian matrix. Again this is an $O(N^3)$ step, but again it is done once only. Alternatively a diagonal matrix can be estimated directly in the delocalized internal coordinate space—this makes it harder to estimate a good starting Hessian, but very easy to get the initial Hessian inverse.

step 1

At the current geometry, construct the B-matrix over the delocalized internal coordinates [Eq. (5)].

Formerly this involved a simple matrix–matrix multiply. In the current algorithm, $\mathbf{B^q}$ is never explicitly constructed; instead the nonzero elements of each column of $\mathbf{B^q}$ are multiplied by the appropriate elements of $\mathbf{U^T}$ as they are formed and added directly to **B**.

step 2

Transform the gradient from Cartesian into internal coordinates.

Formerly accomplished via Eq. (7); now done iteratively using Eq. (13).

step 3

On cycle 1 take the quasi-Newton step $\mathbf{\Delta s}=-\mathbf{H}^{-1}\mathbf{g}^{\text{int}}$. *On subsequent optimization cycles, update* $\mathbf{H}^{-1}$ *using the BFGS update and then take the step.*

step 4

Transform the new geometry in delocalized internal coordinates back to Cartesians.

Formerly accomplished via Eq. (8); now done using the

TABLE II. Timings (CPU; s) on an IBM RS6000/390 workstation for various steps in the internal-coordinate large-molecule optimization algorithm, together with a comparison with Cartesian coordinates.

Cycle[a]	B-matrix[b] construction	Gradient[c] transformation	Optimization[d] step	Back[c]-transformation
(Alanine)$_{30}$ ($C_{90}H_{152}N_{30}O_{31}$, 303 atoms)				
1	296	0.96	-	1.10
2	20	3.68	0.36	1.12
3	20	5.87	0.36	1.10
153	20	11.05	0.36	1.10
Cartesian Optimization: converged in 1005 cycles				
(Alanine)$_{40}$ ($C_{120}H_{202}N_{40}O_{41}$, 403 atoms)				
1	670	1.62	-	1.98
2	37	8.18	0.8	1.92
3	37	12.30	0.8	1.89
140	37	20.15	0.8	1.64
Cartesian Optimization: converged in 1180 cycles				
(Alanine)$_{50}$ ($C_{150}H_{252}N_{50}O_{51}$, 503 atoms)				
1	1296	2.74	-	1.99
2	57	13.59	1.3	2.10
3	57	23.34	1.3	2.08
160	57	32.49	1.3	2.12
Cartesian Optimization: converged in 1432 cycles				

[a]Shown are timings for the first three optimization cycles and the final cycle. Convergence criteria are: maximum gradient component $<0.000\,05E_h/a_0$; energy change $<10^{-7}E_h$.
[b]Timings for B-matrix construction on cycle 1 include once only construction of **U**.
[c]The fast gradient and back-transformation were converged to an accuracy of better than 5×10^{-9}.
[d]The first optimization step is a simple steepest descent and takes minimal CPU time.

new Z-matrix back-transformation, Eq. (10).

step 5

Calculate the energy and gradient at the new geometry and return to step 1.

Steps 1 through 5 are repeated until the maximum gradient component is less than 5×10^{-5} a.u. and the energy change from the previous cycle is less than 10^{-7} hartree. (We use atomic units for compatibility with *ab initio* optimizations.)

Despite the relative simplicity of the actual optimization step (step 3), the algorithm performs very well in practice. Table I shows some sample optimizations on typical organic molecules containing from 80 to 200 atoms run on an IBM RS6000/390 workstation. These were run using the semi-empirical AM1 method.[9] Table I gives timings for the respective steps in the optimization using the original (full) and the new (fast) transformations. Also shown are the number of optimization cycles required for convergence.

As can be seen from Table I, the savings in CPU time using the new algorithm are immense. Taking the largest system, the 20 alanine unit polypeptide fragment, the time required to transform the gradient is reduced from 100 s to around 1.5, and the back-transformation is reduced from 600 s (for the full transformation) to just over half a second. Note that the time for both the fast gradient and fast back-transformation increases as the optimization progresses (as evidenced by the larger timings during the last cycle); this is due to the inevitable mixing of the original **R** and **U** subspaces. For (alanine)$_{20}$ the dominant step in the optimization is now neither the back-transformation nor the transformation of the gradient but the contruction of the B-matrix (step 1 in the algorithm)! Additionally, it can clearly be inferred from Table I that the savings over the original algorithm will increase with system size.

Table II shows optimizations using the original SYBYL force field[10] on alanine polypeptides containing 30, 40, and 50 alanine fragments, respectively, comparing the performance of the new delocalized internal coordinate algorithm with the same BFGS *optimization* algorithm only using Cartesian coordinates. All optimizations were started with a unit matrix in the appropriate optimization space as the initial estimate of the inverse Hessian.

As can be seen, the internal coordinate optimizations converge in far fewer cycles than the corresponding Cartesian optimizations, the average reduction was by a factor of 8. (This would have been even greater if a better Hessian estimate had been used.) The *average additional CPU time* required for the transformations (distributing the CPU time for cycle 1 over all optimization cycles) was approximately 34, 64, and 98 s per cycle for the 30-, 40-, and 50-unit alanine polypeptide, respectively. For these systems, both optimizations converged to essentially the same final energy. Although clearly more efficient in terms of the number of optimization cycles, whether or not the internal coordinate optimizations are more efficient overall (i.e., in terms of the total CPU time) depends on how much CPU time is required to calculate an energy and gradient in the force field; the more complex the force field the more this will favour the internal coordinate optimization.

For our implementation of the SYBYL force field, the CPU time for a single energy+gradient was 17, 40, and 76 s, respectively, for the 30-, 40-, and 50-unit alanine polypeptides. This means that the internal coordinate optimization was actually *more* efficient overall than the corresponding Cartesian optimization by factors of around 2.2, 3.3, and 3.9, respectively. However, our current implementation of SYBYL is not particularly efficient, and furthermore includes *no* cutoff of the pairwise van der Waals repulsion term. If we introduce a cutoff of, say, 10Å (i.e., all interactions between atoms more than 10 Å apart are neglected), then the CPU time for an energy+gradient for $(\text{alanine})_{50}$ falls to less than 3 s, a much more typical time for a force field calculation on this system. This would now favour the Cartesian optimization by a factor of around 2.7, despite the large reduction in the number of optimization cycles.

However, it is clear from the above discussion that—although we may not be there yet—we are certainly close to having optimization algorithms available in internal coordinates that are *more* efficient than Cartesians even for molecular mechanics force fields. For large semi-empirical optimizations and QM/MM computations, the above algorithm should have a significant impact on the total CPU time.

The largest system we have considered here, $(\text{alanine})_{50}$, has just over 500 atoms. This is about the size limit for the described algorithm, principally because of storage requirements. With 500 atoms we have roughly 3000 degrees of freedom and quite easily, say, 5000 primitives. Thus we will need 15 million storage locations just for the **U**-matrix (around 120 MB), not counting storage for the Hessian matrix, transformation matrices, and other scratch storage. Memory is becoming much cheaper—at the time of writing it has almost reached $1 per MB—and modern workstations can easily have a gigabyte or more of RAM memory; even so the memory requirements of the algorithm are excessive.

One obvious way to reduce the storage requirements on **U** is to use natural internal coordinates instead of—as in this work—delocalized internals. Nature internals are linear combinations of typically 10 or less primitives, and in our 500 atom example, this world reduce the storage required for **U** from 120 MB to easily less than 1 MB. Any matrix multiplications involving **U** would also be favorably affected, although the gradient transformation may take longer. The same efficient Z-matrix back-transformation described here could be used essentially unchanged.

We believe that the large-molecule optimization algorithm we have described in this work, in conjunction with natural internal coordinates, would for the first time allow molecular mechanics optimizations carried out in internal coordinates to be *more* efficient, in terms of the *total* CPU time, than optimizations in Cartesian coordinates. Work on such an algorithm is already in progress.

ACKNOWLEDGMENT

This work was supported by the U.S. Air Force Office for scientific research under Grant No. F49620-98-1-0082.

[1] (a) P. Pulay, G. Fogarasi, F. Pang, and J. E. Boggs, J. Am. Chem. Soc. **101**, 2550 (1979); (b) G. Fogarasi, X. Zhou, P. W. Taylor, and P. Pulay, J. Am. Chem. Soc. **114**, 8191 (1992).
[2] P. Pulay and G. Fogarasi, J. Chem. Phys. **96**, 2856 (1992).
[3] J. Baker, A. Kessi, and B. Delley, J. Chem. Phys. **105**, 192 (1996).
[4] E. B. Wilson, J. C. Decius, and P. C. Cross, *Molecular Vibrations* (McGraw-Hill, New York, 1955).
[5] J. Baker and P. Pulay, J. Chem. Phys. **105**, 11100 (1996).
[6] B. Paizs, G. Fogarasi, and P. Pulay, J. Chem. Phys. **109**, 6571 (1998).
[7] Ö. Farkas and H. B. Schlegel, J. Chem. Phys. **109**, 7100 (1998).
[8] J. Baker and W. J. Hehre, J. Comput. Chem. **12**, 606 (1991).
[9] M. J. S. Dewar, E. G. Zoebisch, E. F. Healy, and J. J. P. Stewart, J. Am. Chem. Soc. **107**, 3902 (1985).
[10] M. Clark, R. D. Cramer, and N. van Opdenbosch, J. Comput. Chem. **10**, 982 (1989).
[11] P. Pulay, Mol. Phys. **17**, 197 (1969).
[12] V. Chachra, P. M. Ghare, and J. M. Moore, *Applications of Graph Theory Algorithms* (Elsevier, New York, 1979).
[13] R. Fletcher, *Practical Methods of Optimization, Vol. 1: Unconstrained Optimization* (Wiley, New York, 1980).
[14] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, LAPACK: A Portable Linear Algebra Library for High-Performance Computers, University of Tennessee Technical Report CS-90-105 (May 1990).