An efficient method for the coordinate transformation problem of massively three-dimensional networks

Károly Németh
Laboratoire de Chimie théorique, UMR CNRS No. 7565, Université Henri Poincaré, B.P. 239, F-54506 Vandœuvre-lès-Nancy, France and Centre Charles Hermite, B.P. 239, F-54506 Vandœuvre-lès-Nancy, France

Olivier Coulaud
INRIA, B.P. 101, F-54602 Villers-lès-Nancy, France

Gérald Monard and János G. Ángyán
Laboratoire de Chimie théorique, UMR CNRS No. 7565, Université Henri Poincaré, B.P. 239, F-54506 Vandœuvre-lès-Nancy, France

(Received 7 December 2000; accepted 19 March 2001)

A new and efficient algorithm is presented for the coordinate transformation problem of massively three-dimensional networks formed, e.g., by the atoms of crystal fragments or molecular clusters. The new algorithm is based on a divide-and-conquer technique to perform iterative coordinate transformation, applicable even for three-dimensional networks, with linear scaling memory and near linear scaling CPU time requirements. The new algorithm proved to be very fast in the coordinate transformation problems and geometry optimization of diamond fragments, water clusters, globular proteins, and proteins in solvent. © 2001 American Institute of Physics. [DOI: 10.1063/1.1370534]

I. INTRODUCTION

Many important developments have been made recently in the numerical treatment of the coordinate transformation problem of molecular geometry optimization. These developments have been motivated essentially by the rapid evolution of linear scaling electronic structure calculation techniques recently reviewed by Goedecker and Scuseria. Billeter, Turner and Thiel developed a linear scaling technique for this latter case, using the AINV algorithm, developed by Benzi, Meyer, and Tuma. It achieves linear scaling for quasi one-dimensional molecules, but scales only quadratically for 3D networks. Challacombe proposed a linear scaling technique for this latter case, using the AINV algorithm, developed by Benzi, Meyer, and Tuma.

Here, we suggest to combine Cholesky factorization with a special divide-and-conquer technique to address the challenge of coordinate transformation of massively 3D networks. The paper is built up as follows. First we briefly summarize our previously described algorithm particularly adapted to treat coordinate transformations of quasi one-dimensional (1D) systems. Then we show, how to divide 3D networks into 1D subsystems and solve the 3D problem by combining results from the solutions of the 1D subsystems. We illustrate the performance of our method on large diamond fragments, water clusters and globular proteins. Finally we summarize the new results presented in the paper.

II. ALGORITHM FOR QUASI 1D SYSTEMS

In geometry optimizations performed in internal coordinates, Cartesian energy gradients, \( \mathbf{g}_c \) must be transformed into internal coordinate representation, \( \mathbf{g}_i \). Then, the displacements of the \( \mathbf{q} \) internal coordinates, \( \Delta \mathbf{q} \) is constructed...
from the internal gradients. These displacements must be transformed back into Cartesian ones \((\Delta x)\) and used to update the Cartesian coordinates \((x)\) of the molecule.

The gradient transformation is formulated as

\[
g_i = g_c A.
\]  

(1)

Here, and in the following developments all vectors denote row vectors. The matrix \(A\) is the Morse–Penrose pseudoinverse of the vibrational \(B\) matrix.\(^{14,15}\) The transformation of coordinate displacements from internal to Cartesian representation happens via the iterative back-transformation

\[
x^{(k+1)} = x^{(k)} + (q - q^{(k)})A^t.
\]  

(2)

\(q\) contains the required new values of the internal coordinates, while \(q^{(k)}\) holds their value in the \(k\)th iteration step. The upper index \(t\) denotes matrix transposition. The iterative back-transformation is a Newton algorithm, which is applied because of the nonlinear nature of the relationship between the Cartesian and internal coordinates. The \(B\) matrix consists of the derivatives of the internal coordinates with respect to Cartesian ones

\[
B_{ik} = \frac{\partial q_k}{\partial x_i}, \tag{3}
\]

where \(q_k\) is the \(k\)th internal coordinate and \(x_i\) is the \(i\)th Cartesian coordinate. \(B\) is always sparse provided that local, e.g., primitive, internal coordinates are used. \(A\) is constructed traditionally as the right-hand-side pseudoinverse of \(B\), i.e., \(A(BB^t) = B^t\). However, as it has been pointed out by several authors, the left- and right-hand side pseudoinverses of \(B\) are equivalent, and it is less elaborate to work with the left-hand-side one.\(^{3,6,9}\) We consider \(A\), the left-hand-side pseudoinverse, given by

\[
G_c A = B^t. \tag{4}
\]

with

\[
G_c = B^t B. \tag{5}
\]

\(G_c\) is usually singular, with 6 zero eigenvalues, therefore, its inverse must be constructed in a generalized sense, by inverting only the nonzero eigenvalues of the spectral resolution. \(B\) and \(A^t\) are general rectangular, \(N_c \times N_e\) matrices, while \(G_c\) is of the size \(N_e \times N_e\). \(N_c\) denotes the number of Cartesian coordinates: \(N_c = 3N\), where \(N\) is the number of atoms. \(N_e\) denotes the number of internal coordinates, which is usually much larger than \(N_c\).

\(G_c\) is positive semidefinite by construction. The singularity of \(G_c\) can be eliminated by a small spectral shift

\[
G_c = G_c + \epsilon I_c. \tag{6}
\]

where \(I_c\) is the \(N_c \times N_c\) identity. As we have pointed out, the generalized inverse of \(G_c\) can be approximated by the inverse of \(G_c\).\(^9\) According to our experience, this approximation works very well in the coordinate transformation problem. Multiplication of vectors by the inverse of \(G_c\) can be carried out efficiently via the \(U_c\) Cholesky factors of \(G_c\), which are defined by

\[
G_c = U_c^t U_c. \tag{7}
\]

Using a Newton-type algorithm, the gradient transformation problem can be solved in an iterative fashion

\[
g_i^{(k+1)} = g_i^{(k)} + [g_c - g_i^{(k)}]B U_c^{-1} U_c^{-t} B^t,
\]

(8)

with \(g_i^{(0)} = 0\). This iteration converges to a solution of

\[
g_c = g_B. \tag{9}
\]

which is equivalent to Eq. (1). It is important to note that there is, in principle, an infinite number of solutions to this equation, since any vector of the zero subspace of the \(G_c = BB^t\) matrix can be added to a solution of Eq. (9), \(g_c\), and the sum still remains a solution of the same equation.

The explicit evaluation of the inverse Cholesky factors is not necessary, since we are interested only in the results of multiplications by these inverses. Such multiplications can be carried out very efficiently by using forward and backward substitutions, thus avoiding the explicit evaluation of the nonsparse inverses of the Cholesky factors.

The iterative back-transformation can be formulated similarly

\[
x^{(k+1)} = x^{(k)} + (q - q^{(k)})B U_c^{-1} U_c^{-t}. \tag{10}
\]

As already mentioned, \(B\) is very sparse when local internal coordinates are used. This ensures also high sparsity in \(G_c\). However, even if \(G_c\) is very sparse there is no guarantee that the corresponding \(U_c\) Cholesky factors become also sparse. For quasi one-dimensional systems \(G_c\) can be brought to an approximately banded structure, or to a tree-type structure by symmetric permutations. For chain type molecules, with no spatial connection between topologically far parts of the chain, this permutation can be best done by minimum degree ordering. If there are, in addition, some side chains the optimal ordering can be carried out by nested dissection technique. By these permutations the profile of the \(G_c\) matrix gets minimized.\(^16\) All nonzero elements of the corresponding Cholesky factors appear inside the profile. However, there are systems in which even reordering of \(G_c\) by symmetric permutations cannot help to make the profile as simple as in the above mentioned cases. Such systems are formed, e.g., by massively three-dimensional connectivity networks which appear in crystals, in clusters of molecules or in globular proteins surrounded by solvent. At this point, it is important to emphasize that the structure of \(G_c\) reflects the connectivity of the atoms by the internal coordinates, and the sparsity of \(G_c\) is not a sufficient condition for a linearly scaling Cholesky factorization. Considering a series of \(G_c\) matrices constructed for massively 3D networks of increasing size, but having similar topological structure, the number of nonzero elements of the corresponding Cholesky factors will increase quadratically with the number of atoms. As a consequence, memory and CPU requirements for the Cholesky factorization scale quadratically with system size. This phenomenon is very well known in the theory of Cholesky factorization,\(^16\) and has also been recognized in works of Scuseria\(^{11}\) and Challacombe.\(^12\)

It follows from the above discussion that the previously described algorithm\(^9\) should be further improved to handle the coordinate transformation problem in a linear scaling fashion for 3D networks.
III. ALGORITHM FOR 3D NETWORKS

A. Conceptual formulation

In order to achieve linear scaling, at least in the memory requirements, we need to reduce the complexity of the $G_c$ matrices to be factorized. This is equivalent to a reduction of the connectivities inside the molecule, in other words, we have to remove some connectivities (internal coordinates) and do the transformations only on the rest. Since all internal coordinates are important for the optimization of the molecule, in a forthcoming step of the coordinate transformation procedure we cut another set of internal coordinates and include previously cut ones into the transformation. Thus, during the coordinate transformation procedure we let to relax components of the internal gradients to a solution of Eq. (1) in a sequence, and do similarly for the iterative back-transformation. This kind of methodology to solve large systems of equations is called in general divide-and-conquer technique. Following these lines we may write for the gradient transformation:

$$g_l^{(k+1)} = g_l^{(k)} + [G_c^{-1}B]u_r^{(k)}u_r^{t},$$

and similarly for the coordinate transformation:

$$x^{(k+1)} = x^{(k)} + [q - q^{(k)}]B_r^{t}u_r^{(k)}u_r^{t}.$$  

In these equations $B$ contains contributions from all internal coordinates, $B_r$ has no contributions from the removed internals, $u_r$ is constructed by the factorization of $G_r + \varepsilon I$ with $G_r = B_r^{t}B_r$, $B_r$ and $u_r$ change from iteration to iteration.

B. Prism distributions

In order to determine $B_r$, we have to decide which internal coordinates be active and which be inactive at different steps of the coordinate transformation. Such a choice could be based on a topological analysis of the molecule. We want to avoid performing a general topological analysis, but still to have a generally applicable algorithm. A simple alternative is to divide the molecule into a system of nonoverlapping prisms, and then consider only those internal coordinates, which are not cut by the surfaces of these prisms. The choice of prisms has the advantage that the molecule is allowed to make the maximum possible relaxation along the whole length of the prisms, i.e., along the full size of the molecule. This feature would not be available in case of choosing cubes. For different iteration steps of the coordinate transformation different distributions of prisms may be used. In the construction of these prism distributions there are two main points which must be considered. First, all internal coordinates must appear at least once unremoved. The second point is that maximum available relaxation should be made possible in all coordinate directions. A minimum of five prism distributions can satisfy these conditions. Currently we use six different prism distributions, in order to allow sufficient relaxation in all coordinate directions. These prism distributions consist of uniform prisms. For each coordinate direction two differently positioned distributions of prisms are used. The relative positions of all these prism distributions are set such that they together satisfy the first condition. Furthermore, limited interaction between the prisms may also be allowed. This interaction is switched on by allowing some internal coordinates to be active, if these coordinates connect the tops or bottoms of spatially and topologically neighboring prisms. Thus the isolated prisms become linked and form a quasi one-dimensional system. The different prism distributions are used sequentially in different iteration steps of the coordinate transformation. The cross section of each prism is kept small, independent from the length of the prism, which is determined by the system size. The size of the cross section is determined by the length of the longest connectivity established by local internal coordinates. Usually, at least twice this length is applied to set the size of the smaller edges of the prisms. If some of the internal coordinates do not fit into the prism distributions, the default size of the prisms can automatically be readjusted, before any matrix operations happen.

Inside the prisms, the atoms are numbered such that for the linked system of prisms, which forms a quasi one-dimensional structure, the numbering starts at one end of the 1D system and ends at the other end, scanning continuously the line which defines one-dimensionality. The $G_r$ matrices of these systems, as well as the corresponding Cholesky factors will always be band diagonal.

We have found that there is a big difference in the number of iteration steps when using linked or unlinked prisms. For linked prisms the gradient transformation converges often in 3 to 4 steps, thus some of the prism distributions will not be used in the evaluation of internal gradients. This results in a special solution of Eq. (9), namely no forces will be generated on some internal coordinates. The physical meaning of such a solution is that not all internal coordinates are needed for the optimization of the molecule. However, for the stability of the iterative back-transformation and the geometry optimization it is usually much better to use all internal coordinates. In order to have forces on all internal coordinates, in the first six iteration steps nonlinked prisms are used, thus avoiding too rapid convergence. Then, in the forthcoming iteration steps linked prisms are applied to accelerate convergence. For the iterative back-transformation only linked prisms are applied.

C. Choice of internal coordinates

The above procedure works very well for systems like diamond fragments or water clusters. For water clusters, beside the local primitive internal coordinates, further distance coordinates have to be introduced in order to describe water–water interaction. For globular proteins, in order to achieve rapid convergence in the coordinate transformations, extra internal coordinates need to be introduced beside the primitive ones. In these systems, isolated fragments of the molecules may turn up inside some of the prisms. Thus, the relative position of these fragments is very poorly described. As a result, the coordinate transformation may not converge. A solution is to introduce extra internal coordinates, which bridge the neighboring fragments inside the prisms. Thus we may increase the number of nonzero elements inside the band of $G_r$ without generating any new elements outside the band. Correspondingly, the Cholesky factors remain
band-diagonal and the memory and CPU time requirements of the factorization scale linearly with system size.

We generate the extra internal coordinates by the following means. First, the prisms are divided into approximately uniform boxes each of them containing at least three atoms. The length of these boxes is equal to the width of the prisms. If a box contains less than three atoms, it is merged to one of its neighbors. Then, the largest surface triangle formed by the atoms in the box is determined in all the boxes. The atoms of the triangle are connected by extra stretchings. Then all other atoms of the box are connected to the triangle atoms, each by three extra stretchings. The triangle atoms of neighboring boxes are also linked by three stretchings. After the introduction of these new coordinates rapid convergence can be achieved also in the coordinate transformations of globular proteins. We call these new coordinates skeletal coordinates. Obviously, the problem of appearance of isolated fragments in the prisms could have been treated by various choices of internal coordinates, but an exhaustive analysis of the different possibilities is out of the scope of the present paper.

As an alternative solution to the above problems we have also tried to include the complete set of Cartesian displacement coordinates beside the primitive internal ones. However, it did not help much to decrease the number of iteration steps so far. The reason is probably that the relative position of the isolated fragments can be optimized only slowly in Cartesian coordinates. It is still worth having a look at the formalism of these coordinate transformations since it gives a physical interpretation of the spectral shift we apply on the \( G_k \) matrices. The \( B \) matrix of the Cartesian displacement coordinates is just the \( I \) identity. Thus, after the inclusion of the Cartesian displacements, the modified \( B \) matrix becomes \( B^\ast = B \oplus I \mathbf{c} \) and the corresponding \( G_k \) matrix \( G_k = G_k + I \mathbf{c} \) is always positive definite. The gradient transformation thus becomes

\[
g_k^{(k+1)} = g_k^{(k)} + g_k^{(k)}
\]

\[
[ g_k^{(k)} + g_k^{(k)} ] B^\ast ] G_k^{-1} B_g^t.
\]

(13)

\( g_k^{(k)} \) is allowed to converge to a different vector than \( g_k \). This flexibility must be allowed, since \( B^\ast \) may mix the Cartesian and internal components of \( g_k^{(k)} \). In the light of the above expressions it is obvious that the spectral shift technique [Eq. (6)] is equivalent to the inclusion of the Cartesian coordinates with a small weight. Since the weight factor, \( c \), is chosen to be quite small, the resulting Cartesian component \( g_k^{(k)} \) will practically be zero. Therefore, it is not necessary to evaluate it explicitly.

**D. Projecting out redundancy, translations, and rotations**

According to our experience, if a highly redundant set of internal coordinates is used and the displacements of internal coordinates are small, the iterative back-transformation may not converge. This behavior is due to the redundancy of internal coordinates. The internal coordinate displacements are supposed to lie in a \( N_f \) dimensional space, where \( N_f = 3N - 6 \) for general molecular structures and \( 3N - 5 \) for linear molecules. However, if a redundant set of internal coordinates is used, this property of the displacements is usually not ensured. The space, in which the nonredundant components of the displacements are lying is defined by the columns of the \( B \) or by the rows of the \( A \) matrix. Working with \( B_{r,k} \) matrices implies that the definition of this space varies from iteration to iteration. The required displacement of the internal coordinates, \( \Delta q \), can be decomposed into two parts. One of them, the nonredundant one, \( \Delta q_{\text{nr}} \), is lying in the space spanned by the columns of \( B \), the redundant one, \( \Delta q_{\text{r}} \), is lying out of this space. The iterative back-transformation acts only on the nonredundant component, thus the redundant one cannot be eliminated by the iterations. The convergence of the back-transformation is detected by the fact that the Cartesian coordinates do not change, even if the required change of the internals is not completely satisfied. This means that only the nonredundant component of \( \Delta q \) has been eliminated. However, if different \( B_{r,k} \) matrices are used, the space defining the nonredundant component also changes and oscillations may happen during the back-transformation. We have found that these convergence problems can be avoided, if only the non-redundant component of \( \Delta q \) is used, instead of the total \( \Delta q \) in the iterations, and \( \Delta q_{\text{nr}} \) is determined by the complete \( B \) matrix. Paizs et al. constructed\(^6\) a projector, which separates the redundant and nonredundant components. For building this projector, \( 3N - 6 \) rows of the \( B \) matrix must be selected and their overlap matrix, \( BB^t \) must be built. Then this \( (3N - 6) \times (3N - 6) \) overlap matrix is Cholesky factorized and used in the projection. This technique is very similar to the rank revealing Cholesky factorization.\(^17\) However, as Paizs et al. mention,\(^6\) carrying out the projection may need some experimentation. The instability of such procedures is well known from the theory of rank revealing Cholesky factorization.\(^17\) Also, in many cases the factorization of an overlap-type matrix scales quadratically with the system size, as discussed above. To overcome these limitations, we constructed a different algorithm to project out the nonredundant component of \( \Delta q \). We use the complete \( B \) matrix and the matrices we invert are all small. Our projection scheme is fast, stable and can efficiently be applied also for massively 3D networks.

In our approach the redundancy violating components of \( \Delta q \) are obtained by a procedure, which avoids the use of the full projector. We select sequentially a number of columns (maximum 100) of the \( B \) matrix and store them in \( B_k \). The components of \( \Delta q \) that are perpendicular to the columns of \( B_k \) are obtained by acting on \( \Delta q \) with the complementary projector \( Q_k = I - P_k \), with \( P_k = B_k G_k^{-1} B^t \). After the application of the first partial projector, \( Q_1 \), the resulting redundant component, \( \Delta q_{(1)} = Q_1 \Delta q \) is still contaminated by components, which are parallel to the columns of \( B_2, B_3 \), \ldots, therefore, this step should be repeated until all the columns of the \( B \) matrix have been projected against. At each step, the projected-out components are summed up in a buffer, which is subtracted from \( \Delta q \) only at the end of the procedure:

\[
\Delta q_{\text{nr}} = \Delta q - Q_{N_p} - Q_2 Q_{k-1} \cdots Q_2 Q_1 \Delta q.
\]

(14)
The nonredundant displacement vector, $\Delta q_{\text{irr}}$, obtained by the above algorithm, is not exact because the partial projectors do not commute exactly. However, we can remark that, as a consequence of the sparsity of the $B$ matrix, wach partial projector $P_k$ projects only to a limited number of internal coordinates. The partial projectors access to different localization regions, and these regions have only a limited overlap. Therefore, the above sequential projection scheme is usually a reasonable approximation to the exact redundancy projector. In any case, the application of the approximate projector turned out to be sufficient to correct $\Delta q$ in the difficult convergence cases of the back-transformation.

Since the matrix $G_k$ may be singular, its inverse must be calculated by singular value decomposition (SVD). However, as its size is kept at a fixed small value, performing the SVD is easy. Since the columns of $B$ are very sparse the above projection can be carried out by linear scaling effort for any molecules, without compromising accuracy.

Similarly, rotations and translations of the whole molecule must be projected out from the displacements of the Cartesian coordinates. This can be done via constructing the vectors of molecular translations and infinitesimal rotations. In our experience, translational and rotational components of the Cartesian displacements turned out to be negligibly small.

**IV. RESULTS AND DISCUSSION**

**A. Computational details**

For all sparse matrix algebraic operations a zero threshold of $10^{-10}$ has been applied, while all matrix elements have been expressed in atomic units. The spectral shift has been chosen to $\epsilon = 10^{-6}$ a.u. The following test systems have been used. Diamond fragments of cubic shape, which have been generated by translating the elementary cell of diamond. Cubic clusters of water molecules. Biomolecular coordinates have been taken from The Protein Data Bank (PDB).18 Crambin (PDB entry 1CBN, 642 atoms), Bovine Pancreatic Trypsin Inhibitor (1BPI, 898 atoms), Urate Oxidase (1UOX, 4684 atoms) and Beta-Lactamase (4BLM, 4023 atoms). Some of these biomolecules have also been studied in spherical water environment. The complete set of primitive internal coordinates has been applied in all calculations. For water clusters, in addition to the primitive coordinates, distance coordinates have been introduced. To all biomolecules skeletal coordinates have also been generated by translating the elementary cell of diamond. Cubic clusters of water molecules. Biomolecular coordinates have been generated by translating the elementary cell of diamond. Cubic clusters of water molecules. Biomolecular coordinates have been generated by translating the elementary cell of diamond.

FIG. 1. Number of nonzero elements in the lower triangle of the Cholesky factors calculated for diamond fragments and water clusters at different prism distributions.

**B. Matrix sparsity**

Figure 1 shows the number of nonzero elements in the Cholesky factors (NZ) of different diamond and water cluster systems. The largest molecules, which have been used in the tests contain about 12 000 atoms. Since there are six different prism distributions for all molecules, Fig. 1 shows all corresponding NZ-s. It can be clearly seen that the number of nonzero elements of the Cholesky factors scales linearly with system size, while some prism distributions result in sparser matrices than the others. Other matrices, used in the transformations are much sparser than the Cholesky factors. Linear scaling of the number of nonzero elements in the Cholesky factors is a result of the quasi one-dimensionality introduced by the divide-and-conquer scheme via the prism distributions.

**C. Number of iteration steps**

The number of iteration steps in the gradient transformation does not change much with the system size for diamond. For the smallest diamond fragments (about 100 atoms) it is 7, for the largest ones (about 12 000 atoms) it is 10. For water clusters this number increases from 14 (smallest cluster, about 600 atoms) to 28 (largest cluster, 12 000 atoms). This number is strongly influenced by the choice of internal
coordinates. It is also worth mentioning that if there are atoms, which remain completely isolated in any of the prisms, the number of iteration steps may strongly increase. A tool to avoid this problem is the introduction of skeletal coordinates. The number of iteration steps in the iterative back-transformation was not larger than 12 for the systems studied in the present work.

D. CPU times

Figure 2 shows average CPU times required for one step of the gradient transformation of diamond and water cluster systems. The CPU times scale linearly with system size. For comparison, total CPU times of the gradient transformations of diamond fragments are shown in Fig. 3. Data were collected from calculations with prism distributions and from calculations without prisms, using nested dissection ordering. It can be clearly seen that nested dissection requires quadratically scaling CPU times, while the divide-and-conquer technique scales near linearly. The same holds also for the memory requirements. One step of the iterative back-transformation requires a similar amount of CPU time than one step of the gradient transformation.

E. Geometry optimizations

In order to be sure that the newly developed coordinate transformation algorithms work well in the context of geometry optimization, large diamond fragments have been optimized in the complete set of primitive internal coordinates, using the steepest descent method. Figure 4 shows how the maximum forces converge. The smooth convergence can be achieved only after projecting out redundancies from the internal coordinate displacements. Similar optimizations have been carried out on water clusters: Forces and energies decrease smoothly.

F. Biomolecules

Table I contains gradient transformation data for biomolecules. Two methods are compared. The abbreviation ND stands for the method previously developed for quasi-1D systems, using the nested dissection ordering, while PD denotes the divide-and-conquer technique, based on prism distributions, described in the present paper. For the molecules which are not surrounded by water, the ND technique proved to be more efficient. In the PD method a considerable amount of skeletal coordinates should be introduced: Approximately $3N$ for each of the six prism distribution. This means that the total size of the linear system of equations is supplemented approximately by $6 \times 3N$. Due to the quasi-1D nature of these bipolymers, the ND method gives already satisfactory results.

For the molecule 4BLM the ND technique needs quite a large number of iteration steps. After the introduction of additional internal coordinates the number of iteration steps may strongly decrease. For example, after the addition of the skeletal coordinates the number of iteration steps decreases to 2.

For the biomolecules in water environment skeletal coordinates have been used for both ND and PD methods. The introduction of skeletal coordinates was necessary even for the ND technique, in order to connect disjunct parts of the system (water molecules among each other and with respect to the protein). Here, the divide-and-conquer PD technique
proved to be clearly superior, it requires much less memory and CPU time than the ND method.

V. SUMMARY

In this article we describe a divide-and-conquer technique for carrying out the coordinate transformation problem of molecular geometry optimization. The memory requirements of the new technique scale linearly with system size, even for massively three-dimensional networks of atoms. The CPU time required for the coordinate transformations scales near linearly with the system size, when an appropriate choice of internal coordinates is available. To ensure this condition, in addition to the local primitive internal coordinates skeletal coordinates have been introduced. A new projection technique has been developed, to separate the redundant and nonredundant parts of the displacements of the internal coordinates. This technique proved essential in improving the convergence of the geometry optimization. Parallelization of the Cholesky factorization is feasible, when noninteracting prisms are considered. Also the projection of the redundancies can be parallelized, since the SVD problems can be solved separately and the rest consists only of matrix multiplications. The algorithms described in this paper are especially useful when the conformation of biomolecules in solvent is studied, or when the effects of dopings or lattice defects destroy the translational symmetry of three-dimensional networks of atoms.

ACKNOWLEDGMENTS

K.N. is indebted to Professor Jean-Louis Rivail for his hospitality at Université Henri Poincaré, Nancy I, and for his continuous support and interest in our work. The authors thank François Dehez, Christophe Chipot, and Aylin Sungur for providing us with some of the coordinates of the test molecules. K.N. gratefully acknowledges a postdoctoral fellowship to the Center Charles Hermite, Université Henri Poincaré Nancy I.