Journal of Mathematical Chemistry 25 (1999) 85–92

Another way to implement the Powell formula for updating Hessian matrices related to transition structures

Josep Maria Anglada [a], Emili Besalú [b], Josep Maria Bofill [c] and Jaime Rubio [d]

[a] *C.I.D.-C.S.I.C., Jordi Girona Salgado 18-26, E-08034 Barcelona, Catalunya, Spain*
E-mail: anglada@qteorl.cid.csic.es

[b] *Institut de Química Computacional, Universitat de Girona, Campus de Montilivi, E-17071 Girona, Catalunya, Spain*
E-mail: emili@stark.udg.es

[c] *Departament de Química Orgànica, Universitat de Barcelona, Martí i Franquès 1, E-08028 Barcelona, Catalunya, Spain*
E-mail: jmbofill@canigo.qo.ub.es

[d] *Departament de Química Física, Universitat de Barcelona, Martí i Franquès 1, E-08028 Barcelona, Catalunya, Spain*
E-mail: jaime@hal6001.qf.ub.es

Received 27 July 1998

A way to update the Hessian matrix according to the Powell formula is given. With this formula one does not need to store the full Hessian matrix at any iteration. A method to find transition structures, which is a combination of the quasi-Newton–Raphson augmented Hessian algorithm with the proposed Powell update scheme, is also given. The diagonalization of the augmented Hessian matrix is carried out by Lanczos-like methods. In this way, during all the optimization process, one avoids to store full matrices.

1. Introduction

The optimization of the geometrical parameters based on the location of stationary points on a potential energy function is important in the prediction of molecular structures. The potential energy functions are non-linear and continuously differentiable. An optimization algorithm is efficient when it presents a fast convergence using small memory. Newton and quasi-Newton–Raphson methods are attractive since they are rapidly convergent to a local stationary point. An important point in this type of algorithms is the required storage needs. For an optimization problem of n variables, the storage of the Hessian (analytic or approximated) is O(n^2) memory locations and the solution of the Newton equation O(n^3) additions and multiplications [26].

For the popular update Hessian formulae, namely Davidon–Fletcher–Powell (DFP) and Broyden–Fletcher–Goldfarg–Shano (BFGS) [15], there already exist expressions which avoid storing the full Hessian matrix [8,9,14,16,21,22]. However, these updates are only efficient to optimize the minimum but for saddle points (transi-

tion states) they are not the best updates [2,7]. For the optimization of saddle points it has been proposed to use the Powell, Murtagh–Sargent–Powell (MSP) and related formulae to update the Hessian matrix at each iteration [5–7,15,24,27] and has been demonstrated that they are very successful for this type of problems [7]. In this paper we present two ways to update the Hessian matrix according to the Powell expression [15,24]. Using one of these formulae, one only needs to store two vectors of length n at each iteration. Employing the other formula only small matrices of dimension $(2k)^2$ are needed to store, where k is the current number of iterations of the quasi-Newton–Raphson procedure. We hope that these expressions will be efficient to locate saddle points with limited memory.

2. The Powell representation formula

The most general rank-two update Hessian matrix formula is [11]

$$\begin{aligned}\mathbf{B}_{i+1} &= \mathbf{B}_i + \mathbf{E}_i = \mathbf{B}_i + \mathbf{j}_i\mathbf{u}_i^{\mathrm{T}} + \mathbf{u}_i\mathbf{j}_i^{\mathrm{T}} - \left(\mathbf{d}_i^{\mathrm{T}}\mathbf{j}_i\right)\mathbf{u}_i\mathbf{u}_i^{\mathrm{T}} \\ &= \mathbf{B}_i + [\mathbf{j}_i \quad \mathbf{u}_i]\begin{bmatrix} 0 & 1 \\ 1 & -\left(\mathbf{d}_i^{\mathrm{T}}\mathbf{j}_i\right)\end{bmatrix}[\mathbf{j}_i \quad \mathbf{u}_i]^{\mathrm{T}}, \quad i = 0, 1, \ldots, \end{aligned} \tag{1}$$

where $\mathbf{B}_i$ is the approximated Hessian matrix at the iteration i and the vectors $\mathbf{j}_i = \mathbf{y}_i - \mathbf{B}_i\mathbf{d}_i$ and $\mathbf{u}_i = \mathbf{M}_i\mathbf{d}_i/(\mathbf{d}_i^{\mathrm{T}}\mathbf{M}_i\mathbf{d}_i)$. The $\mathbf{M}_i$ represents a symmetric and positive-definite matrix [7,17]. Finally, the correction vectors are $\mathbf{y}_i = \mathbf{g}_{i+1} - \mathbf{g}_i$ and $\mathbf{d}_i = \mathbf{x}_{i+1} - \mathbf{x}_i$. The vectors $\mathbf{g}_i$ and $\mathbf{x}_i$ are the gradient and the current approximation to the solution at the iteration i, respectively. We remember that, in Powell's update formula, the set of matrices $\mathbf{M}_0 = \mathbf{M}_1 = \cdots = \mathbf{M}_i = \mathbf{M}_{i+1} = \cdots = \mathbf{I}$, where $\mathbf{I}$ is the unit matrix; consequently, $\mathbf{u}_i = \mathbf{d}_i/(\mathbf{d}_i^{\mathrm{T}}\mathbf{d}_i)$.

2.1. The direct representation of Powell's update formula (DRP)

Expression (1) should we written in a compact way as

$$\mathbf{B}_{k+1} = \mathbf{B}_0 + \sum_{i=0}^{k}\mathbf{E}_i = \mathbf{B}_0 + \sum_{i=0}^{k}\left[\mathbf{j}_i\mathbf{u}_i^{\mathrm{T}} + \mathbf{u}_i\mathbf{j}_i^{\mathrm{T}} - \left(\mathbf{d}_i^{\mathrm{T}}\mathbf{j}_i\right)\mathbf{u}_i\mathbf{u}_i^{\mathrm{T}}\right], \quad k = 0, 1, \ldots. \tag{2}$$

This formula takes explicitly the information of all previous iterations. To evaluate the vector $\mathbf{j}_i$ one needs to compute the vector $\mathbf{B}_i\mathbf{d}_i$ and this is carried out using the following formula:

$$\mathbf{B}_i\mathbf{d}_i = \mathbf{B}_0\mathbf{d}_i + \sum_{l=0}^{i-1}\left[\mathbf{j}_l\mathbf{u}_l^{\mathrm{T}}\mathbf{d}_i + \mathbf{u}_l\left(\mathbf{j}_l^{\mathrm{T}}\mathbf{d}_i - \mathbf{d}_l^{\mathrm{T}}\mathbf{j}_l\mathbf{u}_l^{\mathrm{T}}\mathbf{d}_i\right)\right], \quad i = 1, \ldots, k. \tag{3}$$

In a normal quasi-Newton–Raphson iteration k, the matrix $\mathbf{B}_{k+1}$ is updated by the matrix $\mathbf{B}_0$ and the set of the pair vectors $\{\mathbf{j}_i, \mathbf{u}_i\}_{i=0}^{k}$ and the set of scalars $\{\mathbf{d}_i^{\mathrm{T}}\mathbf{j}_i\}_{i=0}^{k}$.

However, as it will be seen below, since we are interested in limited memory problems, the more important is the computation of the vector $\mathbf{B}_{k+1}\mathbf{v}$, for some n-dimensional $\mathbf{v}$ vector. This vector is evaluated using a similar expression to the one given in equation (3) replacing the $\mathbf{d}_i$ vector by the $\mathbf{v}$ vector, that is,

$$\mathbf{B}_{k+1}\mathbf{v} = \mathbf{B}_0\mathbf{v} + \sum_{i=0}^{k} \left[\mathbf{j}_i\mathbf{u}_i^{\mathrm{T}}\mathbf{v} + \mathbf{u}_i\left(\mathbf{j}_i^{\mathrm{T}}\mathbf{v} - \mathbf{j}_i^{\mathrm{T}}\mathbf{d}_i\mathbf{u}_i^{\mathrm{T}}\mathbf{v}\right)\right], \quad k = 0, 1, \ldots. \tag{4}$$

The algorithm for evaluating the vector $\mathbf{B}_k\mathbf{d}_k$ is:

1. Compute $\mathbf{B}_0\mathbf{d}_k$ and store it in a temporal vector, say $\mathbf{b}$. Using the pair of vectors $\{\mathbf{j}_i, \mathbf{u}_i\}_{i=0}^{k-1}$ and the scalars $\{\mathbf{d}_i^{\mathrm{T}}\mathbf{j}_i\}_{i=0}^{k-1}$ do:
   Loop from $i = 0$ to $k - 1$.
   2. Compute the scalar products $\mathbf{u}_i^{\mathrm{T}}\mathbf{d}_k$ and $\mathbf{j}_i^{\mathrm{T}}\mathbf{d}_k$.
   3. Compute $\mathbf{b} \leftarrow \mathbf{b} + \mathbf{j}_i\mathbf{u}_i^{\mathrm{T}}\mathbf{d}_k + \mathbf{u}_i(\mathbf{j}_i^{\mathrm{T}}\mathbf{d}_k - \mathbf{j}_i^{\mathrm{T}}\mathbf{d}_i\mathbf{u}_i^{\mathrm{T}}\mathbf{d}_k)$.
   End of loop.

A similar algorithm is needed to compute the $\mathbf{B}_{k+1}\mathbf{v}$ vector. The computation of this vector requires about $4kn$ multiplications. Note that the DRP formula at each iteration stores two vectors of length n; consequently, if $\mathbf{B}_0$ is selected to be diagonal, then at the iteration k the amount of memory needed using this formula is $2nk + n$. Finally, we note that if the $\mathbf{M}_k$ matrix is taken as $\mathbf{M}_k = a_k\mathbf{B}_{k+1} + b_k\mathbf{B}_k$ for some selected scalars a_k and b_k and it is substituted in equation (1), using straightforward algebra one gets the BFGS formula [2]. With this consideration, the previous algorithm is transformed to the one proposed by Mahidhara and Lasdon [22] for the direct representation of the BFGS update formula.

2.2. The compact representation of Powell's update formula (CRP)

We define the set of matrices

$$\mathbf{R}_k = [\mathbf{r}_0, \ldots, \mathbf{r}_{k-1}] = \mathbf{Y}_k - \mathbf{B}_0\mathbf{D}_k = [\mathbf{y}_0, \ldots, \mathbf{y}_{k-1}] - \mathbf{B}_0[\mathbf{d}_0, \ldots, \mathbf{d}_{k-1}], \tag{5}$$

where $\mathbf{r}_i = \mathbf{y}_i - \mathbf{B}_0\mathbf{d}_i$ and the matrix

$$\mathbf{U}_k = [\mathbf{u}_0, \ldots, \mathbf{u}_{k-1}]. \tag{6}$$

With the above definitions, the approximated Hessian matrix $\mathbf{B}_k$ is given by

$$\mathbf{B}_k = \mathbf{B}_0 + [\mathbf{R}_k \quad \mathbf{U}_k] \begin{bmatrix} \mathbf{O}_k & \mathbf{N}_k^{\mathrm{T}} \\ \mathbf{N}_k & -\mathbf{P}_k \end{bmatrix} \begin{bmatrix} \mathbf{R}_k^{\mathrm{T}} \\ \mathbf{U}_k^{\mathrm{T}} \end{bmatrix}, \quad k = 1, \ldots, \tag{7}$$

where the matrices $\mathbf{O}_k$, $\mathbf{N}_k$ and $\mathbf{P}_k$ are defined and updated in the following way:

$$\begin{cases} \mathbf{O}_1 = 0, \\ \mathbf{O}_k = \begin{bmatrix} \mathbf{O}_{k-1} & \mathbf{0}_{k-1} \\ \mathbf{0}_{k-1}^{\mathrm{T}} & 0 \end{bmatrix}, \end{cases} \tag{8}$$

where $\mathbf{0}_k$ is the k-dimensional zeroth vector,

$$\begin{cases} \mathbf{N}_1 = 1, \\ \mathbf{N}_k = \begin{bmatrix} \mathbf{N}_{k-1} & \mathbf{0}_{k-1} \\ -\mathbf{p}_{k-1}^{\mathrm{T}}\mathbf{N}_{k-1} & 1 \end{bmatrix}, \end{cases} \tag{9}$$

where the vector $\mathbf{p}_k = \mathbf{U}_k^{\mathrm{T}}\mathbf{d}_k$, and, finally,

$$\begin{cases} \mathbf{P}_1 = \mathbf{d}_0^{\mathrm{T}}\mathbf{r}_0, \\ \mathbf{P}_k = \begin{bmatrix} \mathbf{P}_{k-1} & \mathbf{N}_{k-1}\mathbf{n}_{k-1} - \mathbf{P}_{k-1}\mathbf{p}_{k-1} \\ \mathbf{n}_{k-1}^{\mathrm{T}}\mathbf{N}_{k-1}^{\mathrm{T}} - \mathbf{p}_{k-1}^{\mathrm{T}}\mathbf{P}_{k-1}^{\mathrm{T}} & p_{k-1} \end{bmatrix}, \end{cases} \tag{10}$$

where the vector $\mathbf{n}_k = \mathbf{R}_k^{\mathrm{T}}\mathbf{d}_k$. The scalar p_k is defined as

$$p_k = \mathbf{d}_k^{\mathrm{T}}\mathbf{r}_k - \begin{bmatrix} \mathbf{n}_k^{\mathrm{T}} & \mathbf{p}_k^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{O}_k & \mathbf{N}_k^{\mathrm{T}} \\ \mathbf{N}_k & -\mathbf{P}_k \end{bmatrix} \begin{bmatrix} \mathbf{n}_k \\ \mathbf{p}_k \end{bmatrix}. \tag{11}$$

The proof of equation (7) will be done by induction. For $k = 1$, equation (7) gives equation (1) for $i = 0$. Let us assume that equation (7) is true for some k. Then, we substitute equation (7) into equation (1) for $i = k$, and after some manipulation we get

$$\begin{aligned} \mathbf{B}_{k+1} = \mathbf{B}_0 &+ \begin{bmatrix} \mathbf{R}_k & \mathbf{U}_k \end{bmatrix} \begin{bmatrix} \mathbf{O}_k & \mathbf{N}_k^{\mathrm{T}} \\ \mathbf{N}_k & -\mathbf{P}_k \end{bmatrix} \begin{bmatrix} \mathbf{R}_k^{\mathrm{T}} \\ \mathbf{U}_k^{\mathrm{T}} \end{bmatrix} \\ &+ \left(\mathbf{y}_k - \mathbf{B}_0\mathbf{d}_k - \begin{bmatrix} \mathbf{R}_k & \mathbf{U}_k \end{bmatrix} \begin{bmatrix} \mathbf{O}_k & \mathbf{N}_k^{\mathrm{T}} \\ \mathbf{N}_k & -\mathbf{P}_k \end{bmatrix} \begin{bmatrix} \mathbf{R}_k^{\mathrm{T}} \\ \mathbf{U}_k^{\mathrm{T}} \end{bmatrix} \mathbf{d}_k \right) \mathbf{u}_k^{\mathrm{T}} \\ &+ \mathbf{u}_k \left(\mathbf{y}_k - \mathbf{B}_0\mathbf{d}_k - \begin{bmatrix} \mathbf{R}_k & \mathbf{U}_k \end{bmatrix} \begin{bmatrix} \mathbf{O}_k & \mathbf{N}_k^{\mathrm{T}} \\ \mathbf{N}_k & -\mathbf{P}_k \end{bmatrix} \begin{bmatrix} \mathbf{R}_k^{\mathrm{T}} \\ \mathbf{U}_k^{\mathrm{T}} \end{bmatrix} \mathbf{d}_k \right)^{\mathrm{T}} \\ &- \left(\mathbf{d}_k^{\mathrm{T}}\mathbf{y}_k - \mathbf{d}_k^{\mathrm{T}}\mathbf{B}_0\mathbf{d}_k - \mathbf{d}_k^{\mathrm{T}}\begin{bmatrix} \mathbf{R}_k & \mathbf{U}_k \end{bmatrix} \begin{bmatrix} \mathbf{O}_k & \mathbf{N}_k^{\mathrm{T}} \\ \mathbf{N}_k & -\mathbf{P}_k \end{bmatrix} \begin{bmatrix} \mathbf{R}_k^{\mathrm{T}} \\ \mathbf{U}_k^{\mathrm{T}} \end{bmatrix} \mathbf{d}_k \right) \mathbf{u}_k\mathbf{u}_k^{\mathrm{T}}. \end{aligned} \tag{12}$$

Now, making some rearrangements in equation (12) and using relationships (5) and (6), we obtain

$$\begin{aligned} \mathbf{B}_{k+1} &= \mathbf{B}_0 + \begin{bmatrix} \mathbf{R}_k & \mathbf{r}_k & \mathbf{U}_k & \mathbf{u}_k \end{bmatrix} \\ &\quad \times \begin{bmatrix} \mathbf{O}_k & \mathbf{0}_k & \mathbf{N}_k^{\mathrm{T}} & -\mathbf{N}_k^{\mathrm{T}}\mathbf{p}_k \\ \mathbf{0}_k^{\mathrm{T}} & 0 & \mathbf{0}_k^{\mathrm{T}} & 1 \\ \mathbf{N}_k & \mathbf{0}_k & -\mathbf{P}_k & -(\mathbf{N}_k\mathbf{n}_k - \mathbf{P}_k\mathbf{p}_k) \\ -\mathbf{p}_k^{\mathrm{T}}\mathbf{N}_k & 1 & -(\mathbf{N}_k\mathbf{n}_k - \mathbf{P}_k\mathbf{p}_k)^{\mathrm{T}} & -p_k \end{bmatrix} \begin{bmatrix} \mathbf{R}_k^{\mathrm{T}} \\ \mathbf{r}_k^{\mathrm{T}} \\ \mathbf{U}_k^{\mathrm{T}} \\ \mathbf{u}_k^{\mathrm{T}} \end{bmatrix} \\ &= \mathbf{B}_0 + \begin{bmatrix} \mathbf{R}_{k+1} & \mathbf{U}_{k+1} \end{bmatrix} \begin{bmatrix} \mathbf{O}_{k+1} & \mathbf{N}_{k+1}^{\mathrm{T}} \\ \mathbf{N}_{k+1} & -\mathbf{P}_{k+1} \end{bmatrix} \begin{bmatrix} \mathbf{R}_{k+1}^{\mathrm{T}} \\ \mathbf{U}_{k+1} \end{bmatrix}, \end{aligned} \tag{13}$$

which proves relations (7)–(11) for all k. As in DRP, the computation of the vector $\mathbf{B}_k\mathbf{v}$ using equation (7) needs about $4kn$ multiplications. A similar equation for the BFGS formula has been proposed by Byrd et al. [9]. Note that using the DRP formula one only needs to store vectors of length n and small matrices of dimension $(2k)^2$, where k is the current number of iterations.

3. A saddle point search algorithm with limited memory

Rather than to compute the quasi-Newton–Raphson step in the standard way, we compute an approximation to it by the so-called Augmented Hessian (AH) procedure [3]. The AH technique is based on a rational function approximation to the Newton–Raphson quadratic model [1,3,19]. It has the advantage that it introduces some type of restricted step converting the algorithm into a quite powerful method [1]. To locate and optimize a transition state with the AH at any iteration, say k, one should diagonalize up to the second eigenpair the following secular equation [1,3,19]:

$$\mathbf{B}_k^{\mathrm{a}}\mathbf{v}_\nu^{(k)} = \begin{bmatrix} 0 & \mathbf{g}_k^{\mathrm{T}} \\ \mathbf{g}_k & \mathbf{B}_k \end{bmatrix} \mathbf{v}_\nu^{(k)} = \begin{bmatrix} 0 & \mathbf{g}_k^{\mathrm{T}} \\ \mathbf{g}_k & \mathbf{B}_k \end{bmatrix} \begin{pmatrix} v_{1,\nu}^{(k)} \\ \mathbf{v}_\nu'^{(k)} \end{pmatrix} = \lambda_\nu^{(k)}\mathbf{v}_\nu^{(k)} \quad \forall \nu = 1, \ldots, n+1, \tag{14}$$

where $(\mathbf{v}_\nu'^{(k)})^{\mathrm{T}} = (v_{2,\nu}^{(k)}, \ldots, v_{n+1,\nu}^{(k)})$ and the correction vector is given by

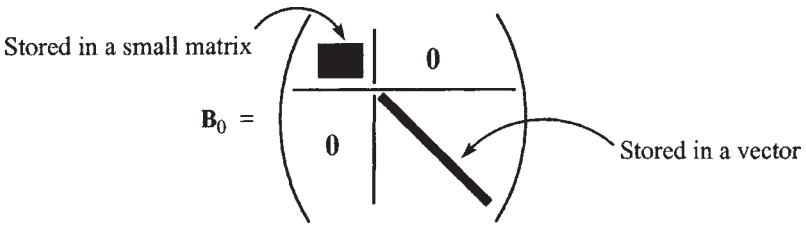
$$\mathbf{d}_k = \frac{1}{v_{1,2}^{(k)}}\mathbf{v}_2'^{(k)}. \tag{15}$$

Since we never store the full matrix, the two first eigenpairs of equation (14) are obtained by a Lanczos-type method [25] like the one proposed by Davidson [10]. On the basis of these ideas, a sketch of the algorithm is:

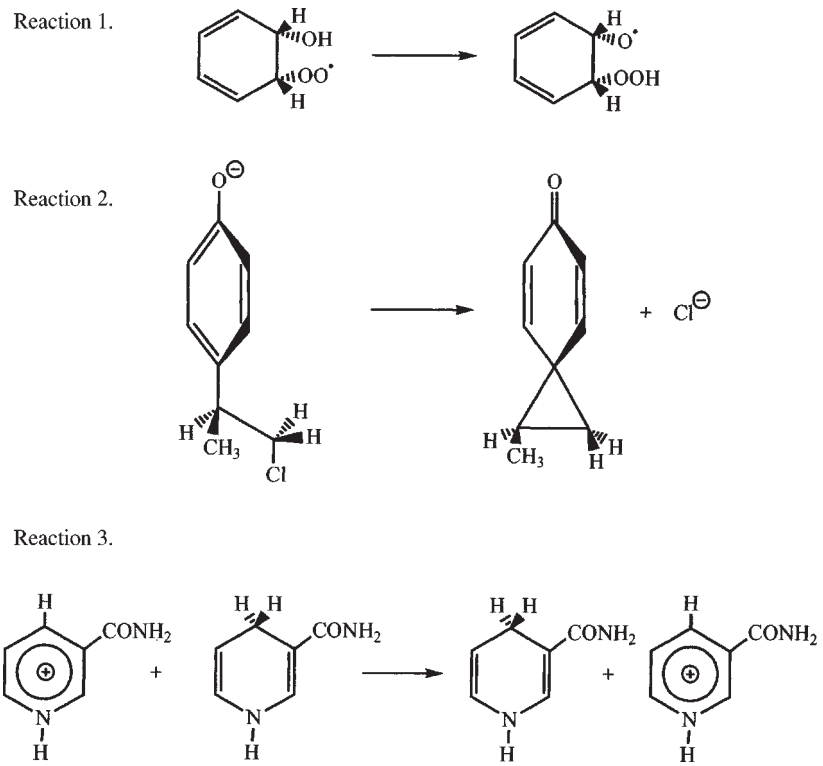
1. Give $\mathbf{B}_0$, and the vectors $\{\mathbf{j}_i, \mathbf{u}_i\}_{i=0}^{k-1}$ and the scalars $\{\mathbf{d}_i^{\mathrm{T}}\mathbf{j}_i\}_{i=0}^{k-1}$ (option DRP) or the matrices $\mathbf{R}_k$, $\mathbf{U}_k$, $\mathbf{O}_k$, $\mathbf{N}_k$ and $\mathbf{P}_k$ (option CRP). It is assumed that $\mathbf{B}_0$ is not complete, but it possesses the necessary correct structure in order to locate the desired transition structure.
2. Obtain the first two eigenpairs of the eigenvalue equation (14) by a Lanczos-type algorithm. At each Lanczos-type iteration, the vector $\mathbf{B}_k\mathbf{v}_\nu^{(k)}$ ($\nu = 1$ or 2) is evaluated by using the algorithm described in section 2.1 (option DRP) or using equation (7) (option CRP).
3. Using equation (15) compute $\mathbf{d}_k$ and the new $\mathbf{x}_{k+1}$, energy $E(\mathbf{x}_{k+1})$ and gradient $\mathbf{g}_{k+1}$. Test for convergence: if $\|\mathbf{g}_{k+1}\| < \varepsilon$, stop ($\|\cdot\|$ defines the Euclidean norm).
4. If option DRP is used, then compute the vectors $\mathbf{u}_k$ and $\mathbf{B}_k\mathbf{d}_k$, the latter using the algorithm described in section 2.1, and from this the vector $\mathbf{j}_k$ and the scalar $\mathbf{d}_k^{\mathrm{T}}\mathbf{j}_k$ and store it. If option CRP is used, then update the matrices $\mathbf{R}_k \rightarrow \mathbf{R}_{k+1}$, $\mathbf{U}_k \rightarrow \mathbf{U}_{k+1}$, $\mathbf{O}_k \rightarrow \mathbf{O}_{k+1}$, $\mathbf{N}_k \rightarrow \mathbf{N}_{k+1}$ and $\mathbf{P}_k \rightarrow \mathbf{P}_{k+1}$ using the relationships (5), (6), (8), (9) and (10), respectively.

4. Comparative performance of Powell's formulae

The transition structure optimizations were carried out with the AM1 [12] semi-empirical Hamiltonian implemented in the program package MOPAC [23]. The appropriate wavefunction (i.e., RHF or UHF) was taken in each case. Since both DRP or CRP need the same number of operations to compute the $\mathbf{B}_k\mathbf{v}$ vector, the present calculations were carried out employing the DRP option. The guess $\mathbf{B}_0$ matrix has the following structure:



where the large diagonal part is stored in a vector and the main part related to the desired transition vector and non-positive-defined, the so-called active part [13], is stored in a small matrix. In scheme 1 we propose three reactions whose transition



Scheme 1.

Table 1
Comparative performance between the DRP algorithm and the standard algorithm to optimize transition structures.

Reaction[a]	nv[b]	$\|\mathbf{g}_0\|$[c]	DRP[d]	Standard method[d,e]
1	42	0.4	4	4
2	57	0.4	11	2
3	92	3.9	14 (48)[f]	8

[a] See scheme 1.
[b] Number of variables in internal coordinates (bond distances, bond angles and dihedrals).
[c] Initial gradient norm in internal coordinates defined as $\|\mathbf{g}_0\| = (\mathbf{g}^T\mathbf{g}/nv)^{1/2}$. The units are kcal/(mol Å) for bond distances and kcal/(mol rad) for bond angles and dihedrals.
[d] Total number of iterations to reach the convergence. The convergence criteria are $\|\mathbf{g}\| \leqslant 0.1$, $\max_i |(\mathbf{g})_i| \leqslant 5.0$, $\|\mathbf{d}\| = (\mathbf{d}^T\mathbf{d}/nv)^{1/2} \leqslant 0.008$, $\max_i |(\mathbf{d})_i| \leqslant 0.04$.
[e] In the standard method the full matrix **B** is stored and the optimization is carried out according to the algorithm described in [5].
[f] The number of iterations needed to achieve the convergence using a different $\mathbf{B}_0$ guess matrix are given in parenthesis. See text for more details.

states have been optimized using the above algorithm. In table 1 we show the number of iterations used in each case. Reactions 1 [20] and 2 [18] are intramolecular rearrangements. The thermochemistry of reaction 1 was studied recently by Lay et al. [20] at AM1 level. For reaction 1, the active part of the guess $\mathbf{B}_0$ matrix contains only the elements related to the parameters of the bond breaking/bond formation –O···H···O– process. On the other hand, for reaction 2, the active part of the $\mathbf{B}_0$ matrix is defined with the geometrical parameters Cl–C, C–C–Cl and C–C–C bond angle of the cyclopropane ring. The third example consists in the location of the *cis* transition state for the reaction of the protonated nicotinamide with 1,4-dihydronicotinamide studied at AM1 level by Bodor et al. [4] and by Wu et al. [28] at *ab initio* level. Despite that this reaction occurs through C_2 symmetry, it was studied within C_1 symmetry. Using this symmetry point group the system has 92 geometrical variables. The active part of the $\mathbf{B}_0$ matrix is defined by the geometrical parameters corresponding to the bond breaking/bond formation –C···H···C– process. According to the results presented in table 1, the degradation of the method, with respect to the standard method of [5] with full **B** matrix, increases with the number of variables. However, if in example 3 the active part of the initial $\mathbf{B}_0$ is redefined in such a way that it takes into account the interaction between –C···H···C– and the two C=O bonds, then the number of iterations decreases to 14. This interaction in this transition state is very important as discussed by Wu et al. [28]. This result reveals that the active part of the $\mathbf{B}_0$ matrix should be accurately selected in order to obtain a good convergence using the proposed algorithm.

5. Conclusion

We have presented a method to optimize transition structures with limited memory. The method is based on a reformulation of the Powell update formula [15,24]. Coupling this formula with the AH method diagonalized by the Lanczos-type algorithm, one does not need to store any full matrix. The only drawback of the proposed algorithm is in the correct selection of the active part of the initial guess for the approximated Hessian matrix.

Acknowledgements

We are indebted to Professor S. Olivella for they valuable suggestions. This research was supported by the Spanish DGICYT (Grant PB95-0278-C02-01).

References

[1] J.M. Anglada and J.M. Bofill, Int. J. Quantum Chem. 62 (1997) 153.
[2] J.M. Anglada and J.M. Bofill, J. Comput. Chem. 19 (1998) 349.
[3] A. Banerjee, N. Adams, J. Simons and R. Shepard, J. Phys. Chem. 89 (1985) 52.
[4] N. Bodor, M.E. Brewster and J.J. Kaminski, J. Mol. Struc. (Theochem) 206 (1990) 315.
[5] J.M. Bofill, J. Comput. Chem. 15 (1994) 1.
[6] J.M. Bofill, Chem. Phys. Lett. 260 (1996) 359.
[7] J.M. Bofill and M. Comajuan, J. Comput. Chem. 16 (1995) 1326.
[8] A. Buckley and A. LeNir, Math. Programming 27 (1983) 103.
[9] R.H. Byrd, J. Nocedal and R.B. Schnabel, Representations of quasi-Newton matrices and their use in limited memory methods, Technical report NAM-03, Northwestern University (1996).
[10] E.R. Davidson, J. Comp. Phys. 17 (1975) 87.
[11] J.E. Dennis, Jr. and J.J. Moré, SIAM Rev. 19 (1977) 46.
[12] M.J.S. Dewar, E.G. Zoebisch, E.F. Healy and J.J.P. Stewart, J. Am. Chem. Soc. 107 (1985) 3902.
[13] W. Diaz, J.M. Aullo, M. Paulino and O. Tapia, Chem. Phys. 204 (1996) 195.
[14] T.H. Fischer and J. Almlöf, J. Phys. Chem. 96 (1992) 9768.
[15] R. Fletcher, *Practical Methods of Optimization* (Wiley, New York, 1987).
[16] J.C. Gilbert and C. Lemaréchal, Math. Programming 45 (1989) 407.
[17] J. Greenstadt, Math. Comp. 24 (1970) 1.
[18] N.S. Isaacs, *Physical Organic Chemistry* (Longman Scientific, New York, 1987) p. 613.
[19] Yu.G. Khait, A.I. Panin and A.S. Averyanov, Int. J. Quantum Chem. 54 (1995) 329.
[20] T.H. Lay, J.W. Bozzelli and J.H. Seinfeld, J. Phys. Chem. 100 (1996) 6543.
[21] D.C. Liu and J. Nocedal, Math. Programming 45 (1989) 503.
[22] D.Q. Mahidhara and L. Lasdon, A SQP algorithm for large sparse nonlinear programs, Technical report, MSIS Department, University of Texas, Austin, TX (1990).
[23] MOPAC program, local version.
[24] M.J.D. Powell, Math. Programming 1 (1971) 26.
[25] Y. Saad, *Numerical Methods for Large Eigenvalue Problems* (Manchester University Press, Halsted Press, Wiley, 1992).
[26] T. Schlick and M. Overton, J. Comput. Chem. 8 (1987) 1025.
[27] S. Simons, P. Jorgensen, H. Taylor and J. Ozment, J. Phys. Chem. 87 (1983) 2745.
[28] Y.-D. Wu, D.K.W. Lai and K.N. Houk, J. Am. Chem. Soc. 117 (1995) 4100.