Reprinted from

THEO
CHEM

Journal of Molecular Structure (Theochem) 398–399 (1997) 111–119

Parallelization strategies and experiences with the dynamically defined reaction path (DDRP) method[1]

L.L. Stachó[a], Gy. Dömötör[b], M.I. Bán[b,*], T. Csendes[c]

[a]Bolyai Institute for Mathematics, Attila József University, Aradi Vértanúk tere 1, H-6720 Szeged, Hungary
[b]Institute of Physical Chemistry, Attila József University, P.O. Box 105, H-6701 Szeged, Hungary
[c]Institute of Informatics, Attila József University, P.O. Box 652, H-6701 Szeged, Hungary

ELSEVIER

ELSEVIER Journal of Molecular Structure (Theochem) 398–399 (1997) 111–119

THEO CHEM

Parallelization strategies and experiences with the dynamically defined reaction path (DDRP) method[1]

L.L. Stachó[a], Gy. Dömötör[b], M.I. Bán[b,*], T. Csendes[c]

[a]*Bolyai Institute for Mathematics, Attila József University, Aradi Vértanúk tere 1, H-6720 Szeged, Hungary*
[b]*Institute of Physical Chemistry, Attila József University, P.O. Box 105, H-6701 Szeged, Hungary*
[c]*Institute of Informatics, Attila József University, P.O. Box 652, H-6701 Szeged, Hungary*

Abstract

The possibilities and advantages of parallel realizations of the "Dynamically Defined Reaction Path" (DDRP) method on large (Single Instruction Multiple Data – SIMD) and small (Multiple Instruction Multiple Data – MIMD) computer architectures, together with some runtime estimates and simulations, are discussed, preceded by a short theoretical introduction referring to the basic mathematical concepts, a description of the algorithm and a numerical realization of the general DDRP procedure. The main difficulty in getting optimal runtimes when using the method of small steps was found to be in the storage strategy of SCF data. © 1997 Elsevier Science B.V.

Keywords: DDRP method; Parallel computation; Path-following

1. Introduction

As a consequence of theoretical works [1] and using the extensive results of optimization methods [2] and of SCF convergence acceleration and iteration improving procedures [3] for calculating stationary points of potential energy hypersurfaces (PES) of chemical reactions and to determine reaction paths (RPs), several local and global methods [4] have been developed during the past two decades. The latest versions of semiempirical [5,6] and ab initio [7,8] quantum chemical program packages available commercially or in the public domain contain one or more of such methods as standard. Most of these programs can also be used effectively in parallel computations [9]. In a series of papers [10–18] new variants of curve variational path-following methods – the Dynamically Defined Reaction Path (DDRP) methods[2] – have been introduced. These methods belong to the class of global procedures and share some common features with the "chain method" of Liotard [19]. However, his method originally gave rise to the so-called "meandering phenomenon" which made it difficult (and sometimes impossible) to determine the RP. Later Liotard introduced a variation of the modified "simulated annealing" procedure [6,20] into the chain method, thus effectively accepting the idea of homogenization [10,11,13]. Schlegel also seems to be

* Corresponding author.
[1] Presented at WATOC '96, Jerusalem, Israel, 7–12 July 1996.

[2] We use the term "dynamically" in the following sense: the dynamics of the (negative) gradient field of the potential function determines RPs as tangent curves to the gradient field connecting stationary points.

familiar with the same technique when employing lately a so called "redistribution" procedure [21]. The popular global curve variational Elber–Karplus method [22] is based upon the minimization of the energy average (with respect to the arc length) along the curve joining the two minima corresponding to the reactant and the product. Unfortunately, it can easily be shown that this method is mathematically false. In a future paper we shall discuss our mathematical arguments and shall suggest a possible correction which can also be considered as a DDRP variant.

The most important aim of the present paper is to show that by high level parallelization and by using adequate parallel computation techniques, the DDRP method [10] can be as effective as any other path-following method. It is to be emphasised that in complicated topological situations (e.g. a higher number of stationary points, winding RPs, local minima, etc.) the DDRP methods are very stable and reliable even when the use of other methods (especially local sequential searches and most global procedures) is generally problematic (or often almost absolutely hopeless) on the theoretical level as well. A widespread objection to the application of DDRP methods is the high amount of computation required in comparison with sequential methods which work well generally, at least in topologically simple cases (e.g. two minima with a saddle point on the PES), although not in cases where the IRC goes on through several saddle points, with orthogonal bifurcations (see, for example, the IRC and PES of the mathematical function simulating the conformational change of the catechol molecule [10,12]). Nevertheless, a time-consuming search like ours is worth applying primarily if, for example, sequential methods do not fulfil the purpose. As we have shown in Ref. [16], when using a suitable parametrization, all the existing curve variational methods can be regarded as DDRP variants. The present paper will show by theoretical runtime estimates and through relevant numerical experiences that, with appropriately designed software and parallel hardware architectures, DDRP methods can be competitive to other methods, even when using just a few parallel processors. The algorithm as discussed below cannot be considered as a tentative optimal realization of the DDRP methods; therefore it is not aimed at direct calculations in chemical practice. For practical parallelization purposes, specially designed energy functions are needed. Here we are using MNDO functions that have not particularly good convergence properties; nevertheless, even when employing improved versions of semiempirical methods [5,6] convergence problems cannot be completely avoided. It was found, however, that adoption of the simple MNDO method [23] in a parallel realization of the DDRP procedure – when using highly sophisticated algorithms – was quite suitable even for modelling complicated practical situations. Finally, it is to be noted that parallelised versions of energy calculating programs are already available [7–9]. Unfortunately, for technical reasons such kinds of parallelizations could not be applied to date in our calculations. Nevertheless, should we have a suitably large number of parallel processors such programs could also be used with the effect of increasing the speed of our DDRP computations as well.

2. Algorithm

First we have to recall the common ideas forming the basis of DDRP methods (which all are indeed various numerical realizations of the fundamental principles laid down in Ref. [10]). By a RP we mean a curve, in the configuration space parameterised on the interval [0,1], connecting (local) minima of the Born–Oppenheimer energy function U of the chemical system in question whose tangent vectors are parallel to ∇U, the gradient of U. In most applications, for RP, the intrinsic reaction coordinate (IRC) [1] is investigated. In such cases the configuration space is the set of mass-weighted coordinates of the atoms involved in the reaction, i.e. $\mathbb{R}^\nu$, where $\nu=3\kappa$ if we consider κ atoms.

Let us take an (arbitrary) initial curve $c_0:[0,1]\rightarrow\mathbb{R}^\nu$. By the results in Refs. [10–12], under the not too restrictive hypothesis about the energy function, $U:\mathbb{R}^\nu\rightarrow\mathbb{R}$, the curves

$$C_t:=\{y_s(t):\ 0\le s\le 1\}\quad (t\ge 0) \tag{1}$$

converge uniformly to some RP as $t\rightarrow\infty$, where for every fixed $s\in[0,1]$, the function $t\mapsto y_s(t)$ is the solution of

$$\frac{\mathrm{d}}{\mathrm{d}t}y_s(t)=-\nabla U(y_s(t)),\quad y_s(0)=c_0(s). \tag{2}$$

Moreover, if the energy function U and the given initial curve $C_0(=\{c_0(s) : 0 \leq s \leq 1\}$ are analytic, we will have (slow) local uniform convergence even for all the derivatives (tangent vectors, curvatures etc.) of the curves C_t $(t \longrightarrow \infty)$.

3. The numerical realization DDRP-2

As a first realization of the DDRP procedure a program called DDRP-I [24] was developed on the basis of the semiempirical MNDO method, for determining the RP of simple collinear chemical reactions [14,15,17]. The program has been generalised for requested cases as follows. We represent the given initial curve $c_0 : [0,1] \longrightarrow \mathbb{R}^\nu$, with arc-length proportional parametrization, by $N+1$ homogeneously distributed points

$$p_0^{(0)} = c_0(0), p_1^{(0)} = c_0(1/N), \ldots, p_{N-1}^{(0)} = c_0((N-1)/N),$$
$$p_N^{(0)} = c_0(1). \tag{3}$$

Choose and fix two parameters:

η (> 0) for step differences in t, and
δ (> 0) for the shifting-length limit.

First we apply a shift S^η, by the vector field $-\eta \nabla U$, to the curve representation

$$C^{(n)} : = \{p_0^{(n)}, \ldots, p_N^{(n)}\} \tag{4}$$

calculating

$$S^\eta p : = p - \eta \nabla U(p) \quad (p = p_0^{(n)}, \ldots, p_N^{(n)}). \tag{5}$$

Notice that the point $(S^\eta)^n p_k^{(0)}$ (where $(S^\eta)^n$ means n-fold iteration of the operation S^η) is an Euler approximation for the point $y_{k/(N+1)}(n\eta)$. However, from a numerical point of view, the set $(S^\eta)^n C^{(0)}$ is a poor representation of $C_{n\eta}$ for two reasons. If the gradient happens to be large, even $(S^\eta)^n p_k^{(0)}$ can be far from $y_{k/(N+1)}(n\eta)$. Conversely, the points $y_{k/(N+1)}(n\eta)$ will cluster around the stationary points of U for large n. Therefore even if all the $(S^\eta)^n p_k^{(0)}$ are good approximations for the respective points $y_{k/(N+1)}(n\eta)$, the set $(S^\eta)^n C^{(0)}$ will furnish almost no information about the whole curve $C_{n\eta}$.

To correct these two types of errors, we use two further operations at each step. When the shifts are too large (larger than δ), first we cut them to length δ. In other words, instead of taking simply $S^\eta C^{(n)}$, we calculate the points

$$C^\delta S^\eta p = p - \max\{\tau \leq \eta : \tau \|\nabla U(p)\| \leq \delta\} \nabla U(p)$$
$$(p = p_0^{(n)}, \ldots, p_N^{(n)}). \tag{6}$$

Then to obtain the next RP approximation $C^{(n+1)} = \{p_k^{(n+1)} : k = 0, \ldots, N\}$, with the points shifted in the former way, we homogenise the set $\{C^\delta S^\eta p_k^{(n)} : k = 0, \ldots, N\}$ as follows. To any polygon $Q : = \{q_0, \ldots, q_N\}$ of $N+1$ points in $\mathbb{R}^\nu$, let us calculate its length $\lambda(Q) : = \sum_{k=1}^N dist(q_{k-1}, q_k)$ and let

$$\begin{aligned} q_k^* &= [\text{the point on } Q^{(n+1)} \text{ with} \\ &\quad \text{arc}-\text{distance } k\lambda(Q^{(n+1)})/N \text{ from } q_0^{(n+1)}] \\ &\quad (k = 0, 1, \ldots, N), \\ HQ &: = \{q_0^*, \ldots, q_N^*\}. \end{aligned} \tag{7}$$

We call HQ the *homogenization* of the polygon Q. Thus

$$\begin{aligned} C^{(n+1)} &= \{p_k^{(n+1)} : k = 0, \ldots, N\} \\ &= H\{C^\delta S^\eta p_k^{(n)} : k = 0, \ldots, N\} \end{aligned} \tag{8}$$

By the results in Ref. [11], the polygons $C^{(n)}$ in the configuration space $(n = 0, 1, 2, \ldots)$ represent a sequence of curves converging uniformly to some RP. Obviously, the accuracy of the method improves with increasing number of points $N+1$ and with decreasing values of the governing parameters η and δ. However, in order to obtain suitable results within reasonable time limits, it is always very important to find good compromises. At the beginning of path-following (far from the true IRC) only slow Euler steps are safe for convergence. For example, Liotard's chain method [19,20] uses steps orthogonal to the approximating curves, resulting in meandering. When we are close enough to the true IRC, other kinds of steps can be more efficient; however, this does not influence the discussion of parallelizability.

4. SCF iteration

The main obstacle for chemical applications of the above method is the fact that it requires the possibility of evaluating of the Born–Oppenheimer energy function for any configuration. It is a well known experience that no available program, semi-empirical or ab initio, can suit this requirement without occasional technical manipulations. Moreover, it cannot even be stated that such programs work automatically perfectly in some neighbourhood of the stationary points of the energy function. The reason for this lack is that the cores of such programs are iteration procedures which fail to converge in many cases.

In our investigations we used the semiempirical MNDO method [23] which is also based on an SCF iteration procedure. Our task is now to force the SCF iteration in MNDO to converge in the course of the DDRP calculations (to the SCF with the lowest energy).

The following techniques seem to be suitable for solving this problem:

4.1. Forcing SCF convergence by modifying the algorithm

Example: The level shifting procedure [25] has been used even in ab initio methods. The convergence is safe but mostly slow. Furthermore the automatic choice of the governing parameters (the so-called dynamical level shifting [26]) is not yet completely solved.

4.2. Semi-safe convergence accelerating procedures

Example: Modified SCF iteration for MNDO [27]. This procedure converges rapidly for many, but not for all, configurations. In the treatment of the CH_3O_2 molecule [28] we applied this modification.

4.3. Method of small steps

In general, if a large number of similar iterative calculations is required, then the following approach, an idea which is also applied in recent program realizations of sequential path following methods, can be very useful. Apply the SCF function, found for some configuration a, as a starting SCF approximation function for a configuration b near to p. Then the iteration procedure will converge again, requiring only a small number of iterations if a and b are sufficiently close to each other. This latter property ensures that, with a proper software design, the parallelised DDRP does not in practice require excessive calculation time. To be more precise, the above idea works if the following hypothesis is fulfilled:

$$\text{HYPOTHESIS}: \exists\delta>0\ \forall a,b \text{ config. } \|a-b\|\le\delta$$
$$\Rightarrow [\text{SCF iteration}]_b(\phi_a)\rightarrow\phi_b \tag{9}$$

where ϕ_a and ϕ_b denote the SCF functions for the energy at configurations a and b, respectively. In our experience, this hypothesis holds for MNDO, although there is no general theoretical argument in the literature for its validity. There is another theoretical point that should be mentioned here. Even if SCF convergence is successfully forced by small steps, one must check the SCF function obtained to see whether it corresponds to the lowest energy value. After making several small steps ending at the starting configuration, by using plain MNDO [23], it is quite possible to get significantly different energy values, for example for the same CH_3O_2 species [28]. MNDO, if it converges, necessarily furnishes the lowest energy eigenvalue. In our experience, this paradox could be avoided by applying the modified iteration (Section 4.2) combined with the method of small steps. We advise the use of the following variants of the method of small steps:

START from a configuration a where ϕ_a is known (e.g. if the SCF iteration has been converged).
Given any configuration $x\in\mathbb{R}^\nu$,

1. we insert a and x into a sequence

$$a=x_0,x_1,\ldots,x_m=x\quad \|x_k-x_{k-1}\|\le\delta \tag{10}$$

and calculate

$$\phi_{x_k}=\lim[\text{SCF iteration}]_{x_k}(\phi_{x_{k-1}}). \tag{11}$$

2. If δ is unknown, we proceed as follows:

First try $[\text{SCF iteration}]_x(\phi_a)$ directly.
No convergence $\Rightarrow$ try $[\text{SCF iteration}]_b(\phi_a)$ with the midpoint $b:=(x+a)/2$.
No convergence again $\Rightarrow$ try $[\text{SCF iteration}]_c(\phi_a)$ with the midpoint $c:=(b+a)/2$.
⋮
Convergence is achieved in finitely many steps, e.g. at point f. Replace a with f and repeat the procedure.

It is an interesting mathematical task to find the most economical version of step 2. From the viewpoint of time consumption the SCF iteration and storage strategies are crucial. This fact has indeed already been noticed by several groups designing chemical program packages. For example, the built-in sequential search for IRC in GAUSSIAN 94 [8] uses also the data of a previous step as an input for the next energy calculation. Convergence problems cannot completely be avoided by any of the available energy calculation methods. Although the procedure used in our numerical experiments is a badly converging one, it can still show how to treat the worst complicated case with a sophisticated method like the DDRP method. In our numerical experiences, forcing fixed numbers of SCF iterations may waste time at an average rate of 30–40%. However, at the same time, the numerical energy function calculated in this way will be analytic with very advantageous numerical properties, while different numbers of SCF iterations may cause inconvenient discontinuities.

5. Parallelization

The extremely good parallelizability of the DDRP methods is based upon the fact that the cut–shifting operations $C^{\delta}S^{\eta}p_k^{(n)}$ can be carried out independently for $k=0,\ldots,N$ (at each step n). Thus if we have $N+1$ parallel working processors, the non-homogenised polygon $C^{\delta}S^{\eta}C^{(n)}$ can be calculated from $C^{(n)}$ in the same time as the slowest calculation $C^{\delta}S^{\eta}p_k^{(n)}$ is carried out. The consecutive homogenization (in order to obtain $C^{(n+1)}=HC^{\delta}S^{\eta}C^{(n)}$) requires negligible runtime compared to SCF iterations. In practice (e.g. for molecules up to about 10 atoms) $N=200$ gives a realistic picture and, indeed, vector computers with many more than 200 parallel working processors are already in existence.

First we discuss how to implement the DDRP procedure in a large (Single Instruction Multiple Data – SIMD) vector computer. SIMD may be on the way out, but it is still worth noting that the DDRP methods would be just ideal for SIMD machines. Our discussion, of course, is mainly based on working with Multiple Instructions Multiple Data (MIMD) architectures, so we also discuss the implementation possibilities in smaller MIMD architectures along with some runtime estimates. Our experimental algorithms are designed only for scientific considerations and not intended for commercial uses. The main technical difficulty in both cases is the storage strategy of SCF data, in order to get optimal runtime with the method of small steps.

6. Numerical gradient

To calculate the shift operations $p \mapsto S^{\eta}p$ instead of the energy value $U(p)$ itself, we are primarily interested in the gradient $\nabla U(p)$. In our previous considerations we mainly concentrated on the number of SCF iterations required to obtain energy values $U(p)$. Recently there has been a growing interest in analytical gradient formulae whose calculation seems to be proportional to the number of SCF iterations for a single energy value.

Here we would like to note that there is a numerically very convenient way to calculate a good approximation for $\nabla U(p)$ when we know the corresponding SCF ϕ_p. Instead of calculating SCFs of the form[3] $\phi_{p+\varepsilon e_i}$ and then corresponding energy values $U(p+\varepsilon e_i)$ $(i=1,\ldots,\nu)$ to get the numerical gradient $((U(p+\varepsilon e_i)-U(p))/\varepsilon:\ i=1,\ldots,\nu)\approx\nabla U(p)$, we can proceed as follows. Given a point $p\in\mathbb{R}^{\nu}$

1. calculate $U(p)$ and store the corresponding SCF ϕ_p;
2. for $i=0,1,\ldots,\nu$ calculate $U(p+\varepsilon e_i)$ by starting the SCF iteration from ϕ_p and perform a *single* iteration step;
3. compute the components $(U(p+\varepsilon e_i)-U(p))/\varepsilon$ $(i=1,\ldots,\nu)$ of the numerical gradient.

The calculation of $S^{\delta}p_{iK+j-1}^{(n)}$ by the jth processor $PROC[j]$ can be started with the SCF $\phi_{p_{iK+j-1}^{(n)}}$. In this manner less iteration steps will be necessary to reach convergence.

7. Large vector architecture

We consider the following model. In our machine K processors $PROC[1],\ldots,PROC[K]$ work independently. Parallel instructions are available for them.

[3] Here $e_1,\ldots,e_{\nu}$ are the unit vectors of the configuration space $\mathbb{R}^{\nu}$. Furthermore we write $e_0=0$.

Each processor *PROC*[*i*] can have an independent quick memory block *ST*[*i*] which is large enough to store SCF data for the energy calculations at a single point. Let the number of points be $N+1=K{\cdot}M$. We then calculate the cut shift of the polygon $C^{(n)}$ in M parallel steps, each of which consists of the calculations at K points. These parallel steps are followed by the sequential homogenization

$$H:\{C^{\delta}S^{\eta}p_0^{(n)},\ldots,C^{\delta}S^{\eta}p_{M\cdot K}^{(n+1)}\}\to\{p_0^{(n+1)},\ldots,p_{m\cdot K}^{(n+1)}\}$$

If we use the strategy of small steps to force SCF convergence, then the required numerical accuracy can be achieved in a different number of SCF iterations at each point. It should be tested for all *PROC*[*i*] ($i\le K$) whether its SCF iteration is finished (to the desired accuracy). At least one processor always works for longer than others do; all the others resume SCF iteration. Finally, if *PROC*[*i*] has treated configuration $p_j^{(n)}$ during the above parallel steps, the SCF data $\phi_{p_j^{(n)}}$ is sent from *ST*[*i*] to a memory segment MEM[*j*] with the aim that the content of *MEM*[*j*] is loaded into *ST*[*i*] the next time when configuration $p_j^{(n+1)}$ is treated necessarily by *PROC*[*i*] again. Notice that we must be sure that all the SCF iterations lead to the lowest energy value, at least in some neighbourhood of the IRC.

In our experiences with the simulation, fatal losses were not caused in runtime if instead of the strategy of small steps we fixed a finite (not too large) number of SCF data $\phi_1,\ldots,\phi_t$ such that in some reasonably large neighbourhood of the IRC the SCF search was accurate enough after a relatively small number s of iterations. We calculated the energy function U as the minimum energy value was obtained after s iteration steps started from the SCF data $\phi_1,\ldots,\phi_t$. For this strategy, we need $s{\cdot}t$ iteration steps to calculate the energy function U at any configuration. Thus for the system CH_3O_2 [27], one can choose three starting SCF data (thus $t=3$) such that only $s=15$ iteration steps are sufficient to reach convergence. Notice that, in this case, the numerical energy function is the minimum of t rational polynomials. Since the latter is very smooth piecewise, the convergence of the obtained polygons $C^{(n)}$ to the IRC is better than that obtained with the strategy of small steps.

7.1. Runtime estimate

The key observation for runtime estimates is the following: to reach satisfactory convergence, calculate approximately as many shifted curves as the maximal number M^* of steps with length $\max\{\eta\|\nabla U\|,\delta\}$ when locating the stationary points with gradient minimization (GM) starting from the points of the initial curve. The reason for this is that the points of the sequences $p_i^{(0)},p_i^{(1)},p_i^{(2)},\ldots$ are approximately as far from the IRC as those in $p_0^{(0)},p_0^{(1)},p_0^{(2)},\ldots$ or $p_N^{(0)},p_N^{(1)},p_N^{(2)},\ldots$ from the stationary points of U they converge to. Therefore the approximate parallel runtime T_{par} reached with large vector architecture can be estimated as

$$\begin{aligned}T_{\text{par}} &\approx M*\frac{N}{K}T_{\text{shift}},\\ M*T_{\text{shift}} &\approx [\text{max time of GM-search}\\ &\quad\text{from } c_0]\\ &\approx T_{\text{SCF}}\,\text{max\#}\,\{\text{SCF iterations}\\ &\quad\text{in GM-search from } c_0\}\end{aligned} \tag{12}$$

Here T_{SCF} is the CPU time of a single SCF iteration. It may be of some interest to compare this time result with that obtained with sequential RP construction on a conventional machine.

In sequential RP constructions, smaller steps are necessary for the numerical stability of the algorithm than with DDRP methods. Nevertheless, using a good substitution and storage strategy for SCFs, we have that

$$T_{\text{seq}}\approx\mu{\cdot}[\text{length of RP}]T_{\text{SCF}} \tag{13}$$

with some factor μ. Hence we can estimate

$$\begin{aligned}&\text{max\#}\{\text{SCF iterations in GM-search from } c_0\}\\ &\quad\approx\frac{\mu}{\text{const.}}[\text{max path-length of GM-searches}\\ &\quad\text{from } c_0].\end{aligned} \tag{14}$$

The constant in the denominator should be $\gg 1$ for the sake of numerical accuracy in sequential computation.

We can also estimate (from numerical experiences)

$$\begin{aligned}&[\text{max path-length of GM-searches from } c_0]\\ &\quad\le\text{length of RP}.\end{aligned}$$

Therefore

$$\frac{T_{\mathrm{par}}}{T_{\mathrm{seq}}} \leq \frac{N}{K\cdot\mathrm{const.}} \quad (K \leq N,\ \mathrm{const.} \gg 1). \tag{15}$$

In practice the constant in the denominator should be around 5. Therefore, if the number of processors is greater than 1/5 times the number of the points in curves we can expect that parallel running will be superior to the sequential one (disregarding higher hardware costs). It would be an interesting theoretical mathematical problem to give precise estimates for the value of the constant mentioned above.

7.2. Numerical experiences

We have tested the effect of parallelization of the DDRP method on the molecular problems H_3, HCN, CH_3O_2 [28,29]. Although we had no access to any large vector architecture, accurate runtime estimates could be given even for this case because the estimation formulae are mathematically quite precise and even somewhat pessimistic, and hence the rate between parallel and sequential runs can be well approximated in terms of numbers of required SCF iterations (which can be calculated even on simple PCs).

For the system H_3 we found that $s=30$ SCF iterations were enough to achieve convergence at any points reasonably near to the IRC. Therefore we can design the following parallel program for determining the IRC for CH_3O_2. First we make 30 parallel SCF iteration steps at the points of the initial curve (regardless of the convergence threshold). Then we can reduce the number of parallel SCF iterations to $s=6$ as the DDRP procedure adjusts the curves nearer to the IRC. In our large vector architecture simulation we used $N+1=64$ points and a straight line segment joining the approximating minima for the initial curve. We calculated 300 polygons to achieve IRC; thus if we had an architecture consisting of 64 parallel processors we would need $30+299\times 6=1824$ parallel iteration steps.

8. Small parallel configuration (PVM)

As we have seen, the overwhelmingly largest time consuming cut–shift operations $C^{\delta}S^{\eta}p_i^{(n)}$ can run in a completely independent manner for any approximating polygon $C^{(n)}$. Thus if we have K computers (even if they are not of the same type) we can also proceed as follows. Given an approximating polygon $C^{(n)}=\{p_i^{(n)}:\ i=1,\ldots,N\}$, we always give the first unprocessed configuration data $p_j^{(n)}$ to the first computer which has just finished the calculation of some $C^{\delta}S^{\eta}p_i^{(n)}$ (with $i<j$ necessarily), and this computer starts to calculate $C^{\delta}S^{\eta}p_j^{(n)}$ from the SCF $\phi_{p_i^{(n)}}$ just determined. In this way we achieve the best work load for our computers for the calculation of polygon $C^{\delta}S^{\eta}C^{(n)}$. However, if the rate N/K is not too large, the configurations $p_i^{(n)}$ and $p_j^{(n)}$ can often be quite distant from each other and hence the calculation of $C^{\delta}S^{\eta}p_j^{(n)}$ from the SCF $\phi_{p_i^{(n)}}$ requires too many iterations unless $K \ll N$.

The above *semi-sequential storage strategy* can be realised with very small alterations to the classical sequential code of the DDRP method for a set of computers connected with a master computer under PVM (Parallel Virtual Machine).

We can formulate the PVM version of the semi-sequential storage strategy of the DDRP method as follows. Processors *PROC*[1],...,*PROC*[*K*] have direct access to small disjointed memory segments *S*[1],..., *S*[*K*]. The number of points in the curves is $N+1 \gg K$. The homogenization $\boldsymbol{H}$ is performed after the parallelly computed cut–shifts (this requires negligible CPU time).

We begin with

$$\begin{array}{l} p_0^{(n)} \rightarrow PROC[1] \rightarrow \begin{pmatrix} C^{\delta}S^{\eta}p_0^{(n)} \\ \phi_{p_0^{(n)}} \end{pmatrix} \cdots \\ p_{K-1}^{(n)} \rightarrow PROC[K] \rightarrow \begin{pmatrix} C^{\delta}S^{\eta}p_{K-1}^{(n)} \\ \phi_{p_{K-1}^{(n)}} \end{pmatrix}. \end{array} \tag{16}$$

A sequential code is parallelised under PVM as follows. The computationally most intensive parts of the sequential C code are marked for PVM. These parts are typically cores of iteration cycles which can be independently executed. In our case the cut–shifts of points are marked for parallelization. According to this, the master process passes the data $p_i^{(n)}$ of the first unprocessed task to the first free processor *PROC*[*j*]. *PROC*[*j*] calculates $C^{\delta}S^{\eta}p_i^{(n+1)}$ from $p_i^{(n)}$ starting the SCF iterations from $\phi_{p_k^{(n)}}$ stored in *S*[*j*].

8.1. Runtime estimate

Let us write

1. T_k = [runtime of a single SCF iteration on $PROC[k]$] ($k \le K$),
2. n_k = [number of SCF iterations by $PROC[k]$]
3. T_{com} = [average communication time related to a single SCF iteration].Then we can estimate

$$n_k(T_k+T_{\mathrm{com}}) \approx T_{\mathrm{run}},$$

$$n_1+\cdots+n_K=[\text{total number of SCF iterations}].$$

Therefore

$$\begin{aligned} T_{\mathrm{run}} &\approx \frac{[\text{total number of SCF iterations}]}{(T_1+T_{\mathrm{com}})^{-1}+\cdots+(T_K+T_{\mathrm{com}})^{-1}} \\ &\approx \gamma_K \max \# \\ &\quad \times \frac{[\text{SCF iterations in GM-searches from } c_0]}{(T_1+T_{\mathrm{com}})^{-1}+\cdots+(T_K+T_{\mathrm{com}})^{-1}}. \end{aligned} \qquad (17)$$

The constant $\gamma_K > 1$ is due to the fact that $C^\delta S^\eta p_j^{(n+1)}$ is almost never computed from $\phi_{p_i^{(n)}}$ with $|i-j|=1$ but it is mostly computed with $|i-j| \gg 1$ requiring essentially more SCF iterations than in case of the vector model. Unfortunately, the factors γ_K increase with K.

8.2. Numerical experiences

Concerning the PVM realization, we have used an 8-node IBM SP1 machine in a multiuser and multitasking system. In this case the theoretical formulae are strongly dependent on experimental constants. Thus we have to use our specific numerical experiences to estimate these factors. We found the communication time in the range of 0.01–0.2 s. The measured user times can be slightly perturbed by the number of users, yet the figures given in Table 1 and Table 2 well represent the typical behaviour. The speedup for eight processors may be terrible for practical purposes, but nevertheless, our numerical experiments, even with such small numbers of processors, show them to be enough for good comparisons.

Table 1
Runtimes for H_3, $N+1=64$ points, 100 polygons

No. of processors	Runtime (s)
1	1612.13
2	896.57
4	726.85
8	515.32

Table 2
Runtimes for CH_3O_2, $N+1=64$ points, 50 polygons

No. of processors	Runtime (s)
1	3306.32
2	1644.73
4	1055.09
8	889.12

Acknowledgements

The authors wish to thank Sz. Szakacsits for carrying out the numerical calculations. This work was supported by the Hungarian Scientific Research Found (Grant Nos. OTKA T4202 and T020743).

References

[1] K. Fukui, J. Phys. Chem. 74 (1970) 4161; see also the excellent surveys given by e.g. K.P. Lawley (Ed.), Potential Energy Surfaces, Wiley, New York, 1980; D.G. Truhlar (Ed.) Potential Energy Surfaces and Dynamics Calculations, Plenum, New York, 1981; J.N. Murrell, S. Carter, S.C. Farantos, P. Huxley, A.J.C. Varandas, Molecular Potential Energy Functions, Wiley, New York, 1984; D.M. Hirst, Potential Energy Surfaces: Molecular Structure and Reaction Dynamics, Taylor and Francis, London, 1985; P. Jorgensen,.J. Simons, Geometrical Derivatives of Energy Surfaces and Molecular Properties, Reidel, Dordrecht, 1985; P.G. Mezey, Potential Energy Hypersurfaces, Elsevier, Amsterdam, 1987; D. Heidrich, W. Kliesch, W. Quapp, Properties of Chemically Interesting Potential Energy Surfaces, Lecture Notes in Chemistry No. 56, Springer–Verlag, Berlin, 1991.

[2] J.J.P. Stewart, J. Comput. Chem. 10 (1989) 209, 221.; J. Comput. Chem. 13 (1989) 157; M.J.S. Dewar, W. Thiel, J. Am. Chem. Soc. 99 (1977) 4899; M.J.S. Dewar, Y. Yamaguchi, Comp. Chem. 2 (1978) 25.

[3] C.G. Broyden, J. Inst. Math. and Appl. 6 (1970) 222.; D. Goldfarb, Math. of Comput. 24 (1970) 23; D.F. Shanno, Math. of Comput. 24 (1970) 647; G.B. Bacskay, Chem. Phys. 61 (1981) 385; Chem. Phys. 65 (1982) 383; A.V. Mitin, J. Comp. Chem. 9 (1988) 107; M.J.S. Dewar, D.A. Liotard, J. Mol. Struct. (Theochem) 206 (1990) 123.

[4] C.P. Baskin, C.F. Bender, C.W. Bauschlicher, Jr., H.F. Schaefer III, J. Am. Chem. Soc. 96 (1974) 2709; J.W. McIver, A. Komornicki, J. Am. Chem. Soc. 96 (1974) 5798; P. Pechukas, J. Chem. Phys. 64 (1976) 1516; K. Ishida, K. Morokuma, A. Komornicki, J. Chem. Phys. 66 (1977) 2153; K. Müller, L.D. Brown, Theor. Chim. Acta 53 (1979) 75; W.L. Hase, R.J. Duchovic, J. Chem. Phys. 83 (1985) 3448; M.W. Schmidt, M.S. Gordon, M. Dupuis, J. Am. Chem. Soc. 107 (1985) 2585; P.G. Jasien, R. Shepard, Int. J. Quantum Chem. Symp. 22 (1988) 183.; M. Page, J.W. McIver, J. Chem. Phys. 88 (1988) 922; C. Gonzalez, H.B. Schlegel, J. Chem. Phys. 90 (1989) 2154; C. Gonzalez, H.B. Schlegel, J. Phys. Chem. 94 (1990) 5523; C. Gonzalez, H.B. Schlegel, J. Chem. Phys. 95 (1991) 5853; J.E. Sinclair, R. Fletcher, J. Phys. C7 (1974) 864.; C.J. Cerjan, W.H. Miller, J. Chem. Phys. 75 (1981) 2800; S. Bell, J.S. Crighton, R. Fletcher, Chem. Phys. Letters 82 (1981) 122; S. Bell, J. Crighton, J. Chem. Phys. 80 (1984) 2464; L.R. Pratt, J. Chem. Phys. 85 (1986) 5045; R. Elber, M. Karplus, Chem. Phys. Lett. 139 (1987) 375; A. Ulitsky, R. Elber, J. Chem. Phys. 92 (1990) 1510; R. Czerminski, R. Elber, Int. J. Quantum Chem. 24 (1990) 167.

[5] J.J.P. Stewart, J. Comput.-Aided Mol. Des. 4 (1990) 1; MOPAC 93.00 Manual, Fujitsu Ltd., Tokyo, Japan, 1993, and references therein.

[6] M.J.S. Dewar, J.J.P. Stewart, J.M. Ruiz, AMPAC 5.0; AMPAC 5.0 User's Manual, Semichem, Inc., Summit, Shawnee, KS 66216, USA, 1994, and references therein.

[7] M.W. Schmidt, K.K. Baldridge, J.A. Boatz, S.T. Elbert, M.S. Gordon, J.H. Jensen, S. Koseki, N. Matsunaga, K.A. Nguyen, S. Su, T.L. Windus, M. Dupuis, J.A. Montgomery Jr., J. Comput. Chem. 14 (1993) 1347; GAMESS User's Guide, Department of Chemistry, Iowa State University, Ames, IA 50011, USA; original program assembled by the staff of the NRCC: M. Dupuis, D. Spangler and J.J. Wendoloski.

[8] M.J. Frisch, A. Frisch, J.B. Foresman, GAUSSIAN 94 User's Reference, Gaussian, Inc., Pittsburgh, PA, 1994, and references therein.

[9] T.L. Windus, M.W. Schmidt, M.S. Gordon, in: T.G. Mattson (Ed.), Parallel Computing in Computational Chemistry, ACS Symposium Series 592, Washington, DC, 1995, pp. 16–28; K.K. Baldridge, M.S. Gordon, J.H. Jensen, N. Matsunaga, M.W. Schmidt, T.L. Windus, J.A. Boatz, Th.R. Cundari, ibid, pp. 29–46.

[10] L.L. Stachó, M.I. Bán, Theor. Chim. Acta 83 (1992) 433.

[11] L.L. Stachó, M.I. Bán, J. Math. Chem. 11 (1992) 405.

[12] L.L. Stachó, M.I. Bán, Theor. Chim. Acta 84 (1993) 535.

[13] L.L. Stachó, M.I. Bán, Comp. Chem. 17 (1993) 21.

[14] Gy. Dömötör, M.I. Bán, L.L. Stachó, J. Comput. Chem. 14 (1993) 1491.

[15] M.I. Bán, Gy. Dömötör, L.L. Stachó, J. Mol. Struct. (Theochem) 311 (1994) 29.

[16] L.L. Stachó, M.I. Bán, J. Math. Chem. 17 (1995) 377.

[17] L.L. Stachó, Gy. Dömötör, M.I. Bán, J. Mol. Struct. (Theochem) 337 (1995) 99.

[18] Gy. Dömötör, M.I. Bán, L.L. Stachó, J. Comput. Chem. 17 (1996) 289.

[19] D.A. Liotard, J.-P. Penot, in: J. Della Dora, J. Demongeot, B. Lacolle (Eds.), Numerical Methods in the Study of Crytical Phenomena, Springer, Berlin, 1981, p. 213.

[20] D.A. Liotard, Int. J. Quant. Chem. 43 (1992) 723.

[21] H.B. Schlegel, Oral communication in WATOC'96, Jerusalem, Israel, July 7–12, 1996 and paper in this Special Issue of Theochem, dedicated to the Jerusalem WATOC'96 meeting.

[22] R. Elber, M. Karplus, Chem. Phys. Lett. 139 (1987) 375.

[23] W. Thiel, QCPE program # 353: Molecular Orbital Calculations by the MNDO Method with Geometry Optimization. W. Theil, QCPE 10 (1978) 353.

[24] Gy. Dömötör, L.L. Stachó, M.I. Bán, DDRP-1, QCPE program, QCMP # 149: DDRP-1; QCPE Bulletin 15(4) (1995) 64.

[25] P. Pulay, in: H.F. Schaefer III (Ed.), Modern Theoretical Chemistry, Vol. 4., Plenum, New York, 1977; P. Pulay, Chem. Phys. Lett. 74 (1980) 393; T.P. Hamilton, P. Pulay, J. Chem. Phys. 84 (1986) 5728.

[26] Gy. Dömötör, M.I. Bán, Computers Chem. 13 (1989) 53.

[27] P. Badziag, F. Solms, Computers. Chem. 12 (1988) 233.

[28] Gy. Dömötör, M.I. Bán, L.L. Stachó, Reaction Paths of Larger Chemical Systems by the DDRP-2 Method, Poster presented at WATOC'96, 4th World Cogress of Theoretically Oriented Chemists, 7–12 July 1996, Jerusalem, Israel; WATOC'96 Program and Abstracts, p. 267.

[29] M.I. Bán, Gy. Dömötör, L.L. Stachó, Scrutinies of Simple Chemical Reactions by the path-following method DDRP-2, Poster presented at WATOC'96, 4th World Cogress of Theoretically Oriented Chemists, 7–12 July 1996, Jerusalem, Israel; WATOC'96 Program and Abstracts, p. 266.