Journal of Mathematical Chemistry 13 (1993) 331–342

Nested summation symbols and perturbation theory

Ramon Carbó and Emili Besalú

Institute of Computational Chemistry, University of Girona, Hospital Square 6, 17071 Girona, Spain

Received 29 June 1992

A new mathematical concept, the *nested summation symbol* (NSS) has been developed. This concept is attached to a mathematical linear operator directly related to the summation symbols. After a discussion on its properties, we investigate the potential usefulness of this symbol in the developing of sequential and parallel computational algorithms, constituting a powerful link between mathematical formalism and high level languages programming. A NSS is well suited in order to express some kind of mathematical formulae and to implement them in any computational environment. In this sense, NSSs are directly related to *artificial intelligence* techniques. Nested sums are connected with *generalized nested do loop* (GNDL) structures, a programming concept developed in our laboratory. This paper shows an application of the NSS. The NSS concept has been used to obtain in a compact form the expressions of the general energy and wavefunction corrections associated to the perturbation theory under the Brillouin–Wigner or the Rayleigh–Schrödinger formalisms.

1. Introduction

Through our research on some quantum chemical topics, we have been facing the need to obtain useful mathematical expressions, based on elementary mathematical concepts [1–3]. We asked ourselves about the ability of such a formulation to exhibit the attractiveness of being simple, elegant, general and susceptible of immediate translation to high level programming languages. Also, because mathematical symbols will be ineluctably involved in such a framework, other scientific areas as computational chemistry and physics or applied mathematics in general can be benefited too.

In our intention was, with high priority, implicit the condition that the final working schemes can serve as tools to build up convenient bridges between mathematical general formulae writing and computationally valid general program structures. Thus, programming techniques can also be assisted by means of this process, as well as artifical intelligence [2] algorithms may use partially the results of our outline in order to increase the performances of formulae generation and translation programs.

In this sense, here we treat the quantum chemical problem of finding the energy and wavefunction corrections of a system within a perturbational theory framework. However, this problem transcends in various aspects the quantum chemical scene and can also be classified to belong to a more broad quantum mechanical circle. In many text books [4–7] the development of the theory of perturbations and the associated methodology to obtain the corrections for the energy and the perturbed wavefunction can be found. The resolvent technique [4,6] allows one to express in a formal manner all the corrections of the wavefunction and also those of the energy values. But also it is common to obtain explicit expressions for the first corrections of the energy. In fact, the perturbation technique is cumbersome because, as the order of the correction increases, a large number of summation symbols appearing in the explicit formulae for these corrections are needed [7,8]. NSS [1,2] override this problems as will be shown below.

Some years ago one of us was interested in studying various aspects of general expressions of Rayleigh–Schrödinger perturbation theory [9]. Here general formulae are presented expressing the energy corrections both in the Brillouin–Wigner and Rayleigh–Schrödinger formulations and an algorithm is developed which allows to codify in a high level computer language the obtained formulae.

2. Definition and properties of the NSS

2.1. DEFINITION OF THE NSS

The *nested summation symbol* concept corresponds to an operator attached to an arbitrary number of nested sums. In other words, a NSS represents a set of summation symbols the number of which can be variable. In a general notation one can write a NSS as $\sum_n(\boldsymbol{j} = \boldsymbol{i}, \boldsymbol{f}, \boldsymbol{s}, \boldsymbol{L})$ where the meaning of this convention corresponds to perform all the sums involved in the generation of all the possible values of the index vector $\boldsymbol{j}$ under the fulfillment of the set of logical expressions collected in the components of the vector $\boldsymbol{L}$. The elements of the vector $\boldsymbol{j}$ have the following limits:

$$\{i_k \leqslant j_k \leqslant f_k, \quad \text{if } s_k \geqslant 0\} \text{ or } \{i_k \geqslant j_k \geqslant f_k, \text{ if } s_k \leqslant 0\}; \quad \forall k = 1, n, \tag{1}$$

where the j_k indices can be incremented or decremented respectively in steps of length s_k. The index n is the *dimension of the NSS*, that is: the number of summation symbols embedded in the operator, and thus the dimension of the involved vectors $\boldsymbol{j}, \boldsymbol{i}, \boldsymbol{f}$ and $\boldsymbol{s}$. The set of all the vectors appearing as arguments of the NSS can be named *parameters of the NSS*.

The logical vector $\boldsymbol{L}$ is of the type $\{\delta(L_i)\}$. The delta symbol corresponds to a *logical Kronecker delta* (LKD). A LKD is a generalization of the Kronecker delta symbol. A LKD symbol has a logical argument and this function can obtain two possible values: 1 if the argument is true, or 0 otherwise. In this manner, the indices

of the vector $\boldsymbol{L}$ are 0's or 1's. So, the convention of a NSS stands for the generation of all the possible forms of the index vector $\boldsymbol{j}$ attached to the logical vector $\boldsymbol{L} = \mathbf{1} = (1, 1, \ldots, 1)$.

A NSS possesses a computational implementation we have called a GNDL [1,2,10]. The Fortran code of the algorithm implementing a GNDL can be found described as program 1 in appendix A below. The GNDL algorithm constitutes the link between the mathematical notation of the NSS and the computer codification of this operator.

When some parameters of the NSS are obvious or are unnecessary to be specified, they are omitted when writing the NSS symbol. In this text, when the step vector $\boldsymbol{s}$ is not specified, it must be assumed that its implicit value is the vector $\mathbf{1}$. If the logical vector $\boldsymbol{L}$ is not specified it means that all the possible forms of the vector $\boldsymbol{j}$ must be generated with any restriction.

2.2. MATHEMATICAL NSS PROPERTIES

(a) NSSs can be recognized as *linear operators* with respect to any general expression placed at the right side of the symbol. At the same time, these expressions can also be studied as multivariate functions whose integer valued variables are the index vector elements:

(b) A product of two NSSs is another NSS, or

$$\sum_n(\boldsymbol{j} = \boldsymbol{i}, \boldsymbol{f}, \boldsymbol{s}, \boldsymbol{L}) \sum_m(\boldsymbol{j}' = \boldsymbol{i}', \boldsymbol{f}', \boldsymbol{s}', \boldsymbol{L}') = \sum_{n+m}(\boldsymbol{j} \oplus \boldsymbol{j}' = \boldsymbol{i} \oplus \boldsymbol{i}', \boldsymbol{f} \oplus \boldsymbol{f}', \boldsymbol{s} \oplus \boldsymbol{s}', \boldsymbol{L} \oplus \boldsymbol{L}'), \tag{2}$$

where the new index vectors are constructed using the direct sum of the original vectors appearing in the product.

(c) The symbol $\sum_0(\boldsymbol{j} = \boldsymbol{i}, \boldsymbol{f}, \boldsymbol{s}, \boldsymbol{L})$ can be made by convention equivalent to the unit operator.

(d) The classical summation symbol $\sum_{j=i}^{f}$ is a particular case of the NSS one, it can be written as $\sum_1(k = i, f, 1)$.

(e) Einstein's convention, by which a set of nested sums are omitted from an expression, corresponds to a NSS like: $\sum_n(\boldsymbol{j} = \mathbf{1}, m\mathbf{1})$

3. Perturbation theory

In order to define the notation which we will use from now on, let us consider the application of the perturbation theory to a system which has a perturbed Hamiltonian H composed by an unperturbed one, H^0, plus a perturbation operator λV, where $\lambda \approx 0$,

$$H = H^0 + \lambda V. \tag{3}$$

From here, the goal consists of finding the eigenvalues and the eigenvectors of the perturbed system, which we denote as the sets $\{E_i\}$ and $\{|i\rangle\}$ respectively. That is, the target is focussed into solving the eigenvalue problem,

$$H|i\rangle = E_i|i\rangle . \tag{4}$$

The eigenvalues and eigenvectors of the unperturbed Hamiltonian are assumed to be known,

$$H^0|0;i\rangle = E_i^{(0)}|0;i\rangle \tag{5}$$

and the ket $|0;i\rangle$ stands for the unperturbed eigenfunction of the ith state and $E_i^{(0)}$ is the corresponding energy. Also it is assumed that this system has an energy spectrum with a simple structure.

The perturbed energies for the ith state can be expressed as

$$E_i = \sum_{n=0}^{\infty} \lambda^n E_i^{(n)} \tag{6}$$

and the corresponding wavefunction is

$$|i\rangle = \sum_{n=0}^{\infty} \lambda^n |n;i\rangle , \tag{7}$$

where the index n signals the correction order in expressions (6) and (7).

On the other hand, the nth order energy correction can be written using the form

$$E_i^{(n)} = \langle i;0|V|n-1;i\rangle ; \quad n>0 , \tag{8}$$

provided that the orthogonality condition holds between the unperturbed state wavefunction and the corrections of any order,

$$\langle i;0|n;i\rangle = \delta(n=0) , \tag{9}$$

where $\delta(n=0)$ stands for a LKD.

4. Brillouin–Wigner perturbation theory

In the Brillouin–Wigner perturbation formalism, the following identity is used [4]:

$$H^0|n;i\rangle + V|n-1;i\rangle = E_i|n;i\rangle + E_i^{(n)}|0;i\rangle . \tag{10}$$

Combining eqs. (8) and (10) it can be easily found that the nth order wavefunction correction is given by [4]

$$|n;i\rangle = \sum_{j_1}{}' \sum_{j_2}{}' \dots \sum_{j_n}{}' \frac{|0;j_1\rangle U_{j_1 j_2} U_{j_2 j_3} \dots U_{j_n 0}}{(E_i - E_{j_1}^{(0)})(E_i - E_{j_2}^{(0)}) \dots (E_i - E_{j_n}^{(0)})}; \quad n>0, \quad (11)$$

being the vector $|0;i\rangle$ defined in eq. (5) and where the terms $U_{ij} = \langle i;0|V|0;j\rangle$ constitute the representation of the perturbation operator V within the characteristic bases set of the unperturbed Hamiltonian H^0. In eq. (11) the primed summation symbols are attached to sums performed over all index values except the ith.

The nth order correction for the energy takes the form [4]

$$E_i^{(n)} = \sum_{j_1}{}' \sum_{j_2}{}' \dots \sum_{j_{n-1}}{}' \frac{U_{ij_1} U_{j_1 j_2} \dots U_{j_{n-1} i}}{(E_i - E_{j_1}^{(0)})(E_i - E_{j_2}^{(0)}) \dots (E_i - E_{j_{n-1}}^{(0)})}; \quad n>1, \quad (12)$$

being $E_i^{(0)}$ and $E_i^{(1)} = U_{ii}$ defined in eqs. (5) and (8), respectively.

Equations (11) and (12) can be rewritten using the NSS formalism. The corrections for the wavefunction take now the simple form:

$$|n;i\rangle = \sum_n (j = \mathbf{1}, \infty\mathbf{1}, \boldsymbol{L}) R_i(j) |0;i\rangle; \quad n>0 \quad (13)$$

and the corrections over the energies are expressed by eq. (8).

In eq. (13) the vectors $\mathbf{1}$ and $\boldsymbol{L}$ are n-dimensional and $\boldsymbol{L}$ components are LKDs of the type $\{\delta(j_k \neq i); \forall k = 1, n\}$. The operator $R_i(j)$ is written as

$$R_i(j) = \prod_{p=1}^{n} Z_{i,j_p} \quad (14)$$

where $Z_{p,q}$ is a projector-like operator defined in turn as

$$Z_{p,q} = |0;q\rangle\langle q;0|V(E_p - E_q^{(0)})^{-1}. \quad (15)$$

Thus, one can see NSS as a useful device which permits to write eqs. (11) and (12) in a compact manner. Also it allows one to easily obtain these formulae by means of the NSS straightforward implementation, the GNDL algorithm, as it is shown in appendix B.

In fact, the Brillouin–Wigner notation is not very adequate for many problems. This is so because in order to obtain the perturbed wavefunction, it is necessary to use the corresponding perturbed energy. The perturbed energy is not known in advance in many cases although an iterative algorithm can be employed, using eq. (8) combined with (13). The method starts the computation with a trial perturbed energy and then iterates over eqs. (8) and (13) until a selfconsistent situation is reached. Appendix B shows a practical implementation of this algorithm.

5. General Rayleigh–Schrödinger perturbation theory

As it can be seen in eq. (13), the NSS notation permits to write some equations

in an elegant and compact manner. In fact, this is due to the fact that NSS opens a new door in order to obtain algebraic expressions. In this sense we propose the use of NSS as an ideal framework to construct a *really general perturbation theory* scheme. The following discussion will try to prove this.

Let us write a perturbed Hamiltonian by a set of k independent perturbation operators using the following expression involving a NSS:

$$H = \sum_k(\boldsymbol{p} = \boldsymbol{0}, \boldsymbol{K})\lambda(\boldsymbol{p})H(\boldsymbol{p}), \tag{16}$$

where the vector $\boldsymbol{s}$ and $\boldsymbol{L}$ of the NSS are omitted, assuming that $\boldsymbol{s} = \boldsymbol{1}$ and all the possible forms of vector $\boldsymbol{p}$ have to be generated. In eq. (16) the first parameter vector values gives the unperturbed Hamiltonian $H(\boldsymbol{0})$, thus the convention $\lambda(\boldsymbol{0}) = 1$ must hold, and any other vector index $\boldsymbol{p}$ structure generates a set of perturbation operators $\{H(\boldsymbol{p}); \boldsymbol{p} \neq \boldsymbol{0}\}$. The final parameter vector $\boldsymbol{K}$ contains the maximal order of the perturbation, which can be different for every operator. The symbol $\lambda(\boldsymbol{p})$ is an element of the scalar set of perturbation parameters. Both $H(\boldsymbol{p})$ and $\lambda(\boldsymbol{p})$ can be considered products of perturbation operators and the attached parameters.

That is:

$$H(\boldsymbol{p}) = \prod_{i=1}^{k} H_i^{(p_i)} \tag{17}$$

and

$$\lambda(\boldsymbol{p}) = \prod_{i=1}^{k} \lambda_i^{p_i}. \tag{18}$$

The adequate technique here is to substitute the usual Rayleigh–Schrödinger scalar perturbation order by a *vector perturbation order* $\boldsymbol{n}$.

The perturbed energies and wavefunctions for the ith system state can be expressed in a similar way as in scalar perturbation theory,

$$E_i = \sum_k(\boldsymbol{n} = \boldsymbol{0}, \infty\boldsymbol{1})\lambda(\boldsymbol{n})E_i(\boldsymbol{n}), \tag{19}$$

and

$$|i\rangle = \sum_k(\boldsymbol{n} = \boldsymbol{0}, \infty\boldsymbol{1})\lambda(\boldsymbol{n})|\boldsymbol{n}; i\rangle, \tag{20}$$

the expressions (19) and (20) being the generalization of eqs. (6) and (7), respectively.

Substituting eqs. (16), (19) and (20) into the perturbed Schrödinger secular equation produces the $\boldsymbol{n}$th order equation:

$$\sum_k(\boldsymbol{p} \oplus \boldsymbol{q}, \delta(\boldsymbol{p} + \boldsymbol{q} = \boldsymbol{n}))H(\boldsymbol{p})|\boldsymbol{q}; i\rangle = \sum_k(\boldsymbol{p} \oplus \boldsymbol{q}, \delta(\boldsymbol{p} + \boldsymbol{q} = \boldsymbol{n}))E_i(\boldsymbol{p})|\boldsymbol{q}; i\rangle, \tag{21}$$

which when $\boldsymbol{n} = \mathbf{0}$ yields the unperturbed Schrödinger equation.

Thus, the nth order energy correction for the ith system's state can be written as

$$E_i(\boldsymbol{n}) = \sum_k(\boldsymbol{p} = \mathbf{0}, \boldsymbol{n}, \delta(\boldsymbol{p} \neq \mathbf{0}))\langle i; \mathbf{0}|H(\boldsymbol{p})|\boldsymbol{n} - \boldsymbol{p}; i\rangle \tag{22}$$

provided that the orthogonality condition

$$\langle i; \mathbf{0}|\boldsymbol{p}; i\rangle = \delta(\boldsymbol{p} = \mathbf{0}) \tag{23}$$

holds between the unperturbed state wavefunction and their perturbation corrections up to any order.

The wavefunction corrections can be obtained similarly through a resolvent operator technique which will be discussed below. The nth wavefunction correction for the ith state of the perturbed system can be written in the same manner as it is customary when developing some scalar perturbation theory scheme: by means of a linear combination of the unperturbed state wavefunctions, excluding the ith unperturbed state. That is

$$|\boldsymbol{n}; i\rangle = \sum_j{}' a_{ji}|\mathbf{0}; i\rangle\,. \tag{24}$$

Using expression (24) into eq. (21), after some straightforward manipulation, one can obtain the equivalent rule in order to construct the nth order wavefunction correction

$$|\boldsymbol{n}; i\rangle = \sum_k(\boldsymbol{p}, \delta(\boldsymbol{p} \neq \mathbf{0}))R_i(\boldsymbol{p})|\boldsymbol{n} - \boldsymbol{p}; i\rangle\,, \tag{25}$$

where a set of resolvent operators $\{R_i(\boldsymbol{p})\}$ for the ith state are easily defined as follows:

$$R_i(\boldsymbol{p}) = Z_i(\mathbf{0})(H(\boldsymbol{p}) - E_i(\boldsymbol{p})) \tag{26}$$

with the weighted projector sum $Z_i(\mathbf{0})$ defined in turn as

$$Z_i(\mathbf{0}) = \sum_j{}'(E_i(\mathbf{0}) - E_j(\mathbf{0}))^{-1}P_j(\mathbf{0})\,, \tag{27}$$

$\{P_j(\mathbf{0})\}$ being the set of projectors over the unperturbed states,

$$P_j(\mathbf{0}) = |\mathbf{0}; j\rangle\langle j; \mathbf{0}|\,. \tag{28}$$

In this context eqs. (22) and (25) can be considered forming a completely general perturbation theory for *nondegenerate* systems.

6. Conclusions

A mathematical device, the NSS, which can be related to artificial intelligence techniques, has been defined and applied in order to solve a concrete quantum che-

mical problem. This symbol is related to computer formulae generation. Not only can the present perturbation theory problem be solved by means of the use of NSSs: many other applications of such symbols can be found in mathematical applications as well as in mathematical chemistry in particular [1,2].

Apart of being able to simplify typographical structures, the NSS symbols constitute the basic elements of a completely general framework, allowing to write mathematical formulae, in such a manner that *immediate translation* to any high level programming language is feasible, producing a *complete general* code, which can be kept sequential of parallelized in a simple manner, as it is explained in appendix A below.

Appendix A. Computational implementation of a GNDL

GENERAL CONSIDERATIONS

In standard high level language programmation the dimension of the NSS, n, signals the number of nested do loops which are necessary to reproduce the structure in a computational environment. But the mathematical usefulness of this entity can be easily recognized when the particular characteristic of this symbolic unit is analyzed: the involved vector parameters could be chosen with *arbitrary and variable dimensions*. There are many scientific and mathematical formulae which will benefit of this property, when written in a paper or computationally implemented.

NSS symbolism constitutes a link between mathematical formalism and program implementation techniques, because successive generation of j index vector elements can be programmed in a general but simple way under any high level language. This can be achieved using a unique "do" or "for" loop statement construct, which is *general and independent of the dimension of the involved nested sums*. This kind of programming structure constitutes the GNDL algorithm.

NSS have *not a direct* translation to the usual high level languages. Present day compilers or standard language rules ignore such an interesting feature, see for example the practical final form of the standard Fortran 90 language [11]. Even high level language compilers have no capacity of processing more than a limited number of classical do loops in a nest, for example VAX Fortran [12] has a limit of 20 nested do loops. Thus, the GNDL structure is a good candidate to circumvent these limitations in any compiler.

It looks simple to introduce GNDL in the family of repetitive sentences found in high level languages. So we feel that a claim in this direction to language and compiler builders can be made here. Some immediate computational benefits in order to construct *really* general programs may be obtained.

In order to show in a practical manner the computational implementation of a NSS, program 1 represents a Fortran source code corresponding to the NSS structure. The NSS implementation using a GNDL generates all the forms of vector j

obeying the logical expressions constituting the components of the vector L. According to this, program 1 generates the indices of the n-dimensional NSS $\sum_n(j = i, f, s, L)$, where the components of the vector L are $\{\delta(j_k \neq l_k); \forall k = 1, n\}$. The dimension n and the initial, final and step index values collected in the vectors i, f, and s have not been specified and the question mark symbol stands for their possible values. These values depend on the concrete application given to the algorithm. Here it is assumed that the step vector s has all its components positive definite.

Application is a called procedure where the n nested loops converge and where their leading indices can be arbitrarily used in the desired internal application. The j index values generation is sequential but the execution of *Application* can be performed into separate CPUs. In this manner, the full computation can be parallelized if desired. In fact, this is a general algorithm, enabling to perform a parallel

Program 1: NSS implementation. GNDL Fortran code

* The innermost loop corresponds to the index $k = n$.

```
      Parameter (n=?)
      Integer j(n), i(n), f(n), s(n), l(n)
*     〈NSS starting parameter values〉
      do k=1, n
        i(k)=?
        f(k)=?
        s(k)=?
        l(k)=?
        if (i(k). eq. l(k)) i(k)=i(k)+s(k)
        j(k)=i(k)
      end do

*     〈Start nested sum procedure〉

      k=n
      do while (k.gt.0)
        if (j(k).gt.f(k)) then
          j(k)=i(k)
          k=k-1
        else
          call Application (n, j)
          k=n
        end if
        if (k. gt. 0) then
          j(k)=j(k)+s(k)
          if (j(k). eq. l(k)) j(k)=j(k)+s(k)
        end if
      end do
      END

*     〈End of NSS Program.〉
```

Application implementation if the nature of the problem asks for such a process and the available hardware allows to run it in this manner. A previous tentative description on GNDL, in a sequential programming framework, was initially made by Carbó and Bunge [10].

Also, another GNDL structure can be put inside *Application*, constituting the first step towards a *generalized nest of GNDL* program structure. We have developed some programming experience on this subject. Construction of a generalized nest of GNDL, aside of being a good no trivial programming exercise, as far as practice has shown to us, it also constitutes a very good test to seek flaws, potentially present, into Fortran and other high level language compilers.

Various application examples can be obtained upon request from the authors. Some Fortran source codes on combinatorial analysis, product of an arbitrary number of matrices and determinant evaluation in a parallel transputer [13] environment are available.

Appendix B. Numerical example

Here, we describe the Fortran source code listed in program 2. This function constitutes a simple codification of the iterative algorithm described above in order to compute the energy corrections of a perturbed system. The program works within the Brillouin–Wigner formalism and uses eq. (12). The code computes the ith eigenvalue of an m-dimensional perturbed matrix. The arguments of the function are the following: the matrix dimension (m), the eigenvalue number to be computed (i), the maximal order correction which one wants to achieve (k), the eigenvalues of

Program 2: Brillouin–Wigner formula implementation

```
      DOUBLE PRECISION FUNCTION BRIWIG (m, i, k, E, U)
      implicit double precision (A-H, O-Z)
      parameter (maxd=9, maxorder=15)
      dimension E(maxd), U(maxd, maxd)   ! Energies and matrix elements
      dimension j (0: maxorder), EE (maxd)
      j (0)=0
      mn = k - 1
      ini=1
      if (i. eq. 1) ini=2
*
*     Define initial Energy differences
*
        Ener=E(i)+U(i, i)
        Ener_old=0.0D0
        do k=1, m
          EE(k)=Ener-E(k)
        end do
*
*     Starting iterations
*
```

```
      Iter=0            ! Iteration counter
      Criterion=0.0D0 ! Finish iteration criterion
      DO WHILE (dabs(1.0D0-Criterion). gt. 1. 0d-12. and. Iter. lt. 99)

*
*   Corrections: Zeroth & First order correction
*
        Ener=E(i)                    ! Zeroth order correction
        Ener=Ener+U(i, i)            ! Adds first order correction
*
*   Successive Correction orders
*
        do n=1, mn                   ! The order of the correction is n+1
          En=0.0D0                   ! Energy correction
*   ---------------------------------------------------------------
*   NSS of dimension n: Initital values of NSS indices
*
          do k=1, n
            j(k)=ini
          end do
*
          jn=n
          do while (jn. gt. 0)
            if (j(jn). gt. m) then
              j(jn)=ini
              jn=jn-1
            else
*
*   Correction Computation.
*
              Denominator=1.0D0
              Terms_Uij=U(i, j(1))
              do k=1, n-1
                Denominator=Denominator*EE(j(k))
                Terms_Uij=Terms_Uij*U(j(k), j(k+1))
              end do
              Terms_Uij=Terms_Uij*U(j(n), i)
              Denominator=Denominator*EE(j(n))
              En=En+Terms_Uij/Denominator
              jn=n
            end if
            j(jn)=j(jn)+1
            if (j(jn). eq. i) j(jn)=j(jn)+1
          end do
*   ---------------------------------------------------------------------
          Ener=Ener+En               ! Adds Correction
*
*   Redefine Energy differences
*
          do k=1, m
            EE(k)=Ener-E(k)          ! Ener must tend to the perturbed energy
          end do
        end do
        Iter=Iter+1
        Criterion=Ener_old/Ener
        Ener_old=Ener
      END DO                         ! New iteration
      if (Iter. ge. 99) stop 'Non convergence'.
      BRIWIG=Ener
      END
```

the unperturbed system (E) and the matrix elements (U) constituting the representation of the perturbation operator in the characteristic basis of the eigenvectors of the unperturbed matrix. The returned value is the perturbed eigenvalue attached to the ith eigenvector.

The routine stops if there is a zero division or no convergence is encountered. The code has been tested using as unperturbed matrix a Hilbert matrix (H) and the perturbation was constructed adding the scalar 0.01 over the H_{11} matrix element.

Acknowledgement

This work is a contribution of the "Grup de Química Quàntica del Institut d'Estudis Catalans" and it has been financed by the CICYT–CIRIT Fine Chemicals Programme of the "Generalitat de Catalunya" through a grant #QFN91-4206. One of us (EB) benefits of a grant of the "Departement d'Ensenyament de la Generalitat de Catalunya".

References

[1] R. Carbó and E. Besalú, Adv. Quant. Chem., accepted.
[2] E. Besalú and R. Carbó, J. Comp. Chem., submitted.
[3] R. Carbó and J.M. Riera, *A General SCF Theory* (Springer, Berlin, 1978);
R. Carbó and O. Gropen, Adv. Quant. Chem. 12 (1980) 159;
R. Carbó, Ll. Domingo and J.J. Peris, Adv. Quant. Chem. 15 (1982) 215;
R. Carbó, J. Miró, J.J. Novoa and J.J. Domingo, Adv. Quant. Chem. 20 (1989) 375.
[4] C.H. Wilcox, ed., *Perturbation Theory and its Applications in Quantum Mechanics* (Wiley, New York, 1966).
[5] F.L. Pilar, *Elementary Quantum Chemistry* (McGraw-Hill, New York, 1968).
[6] R. Carbó and J.A. Hernández, *Introducción a la Teoría de Matrices* (Editorial Alhambra, Madrid, 1983).
[7] A. Szabo and N.S. Ostlund, *Modern Quantum Chemistry* (McGraw-Hill, New York, 1989).
[8] J.O. Hirschfelder, in: *Perturbation Theory and its Applications in Quantum Mechanics* (Wiley, New York, 1966) p. 3.
[9] R. Carbó, Theor. Chim. Acta 17 (1970) 74;
R. Carbó, Rev. Roum. Chim. 16 (1971) 1155;
R. Carbó and R. Gallifa, Nuovo Cimento 10 (1972) 576;
R. Carbó, Int. J. Quant. Chem. 6 (1972) 609;
R. Carbó and R. Gallifa, An. Física 68 (1972) 197;
R. Carbó and R. Gallifa, Nuovo Cimento 17 (1973) 46;
R. Carbó and R. Gallifa, An. Física 69 (1973) 331.
[10] R. Carbó and C. Bunge, PC Actual Magazine (September 1989) 124.
[11] Fortran 90, X3J2 Internal document s8.118; ANSI x3.198-1991, May 1991.
[12] VAX Fortran Document AA-D034D-TE, p. E28 (Digital Equipment Corporation, Maynard, MA, 1984).
[13] Microway, Quadputer2, Owner's Manual (Microway, Kingston, 1989).