Journal of Mathematical Chemistry 20 (1996) 141–149

The permutational symmetry in matrix multiplications

Eduardo Hollauer

Departamento de Físico-Química, Instituto de Química, Universidade Federal Fluminense, Morro do Valonguinho s/n, Niterói, Centro, CEP 24210-150, Rio de Janeiro, Brazil
E-mail: gfqholl@vm.uff.br

Received 1 June 1995; revised 28 May 1996

This article presents slight modifications to algorithms for *in-core* symmetric matrix multiplications in order to optimize the computational number of multiplications required. The use of the *petite liste (Pl)* algorithm, a general procedure of treating spatial symmetry in molecular calculations, is extended to permutational symmetry in matrix multiplications. This implementation requires the same number of operations as a regular matrix multiplication when the dimensions of original and transformed matrices are the same. However, when the transformed space dimension is smaller, this algorithm provides savings of up to a factor of two in the overall number of multiplications involved. Such a method can be viewed as an alternative demonstration to Saunders and van Lenthe's two-index transformation technique, who developed similar expressions through the decomposition of the symmetric matrix into its upper and lower triangular parts. The final equations obtained by these authors are the same as the ones shown here. However, the present method is supported by a solid theoretical framework, permutational group theory, which makes it general and applicable over any permutational symmetry available.

1. Introduction

Basis transformations and sub-space projections are regular operations employed in many computational algorithms. When dealing with symmetric matrices it is usual to employ permutational symmetry, mainly with the purpose of memory saving. In such a case, just the triangular part of the original and the final matrices are stored and calculated. Although this procedure saves nearly half of the memory required, it does not reduce the number of multiplications performed, since each element, prior to the calculation, restores all the others. These elements are then processed as if there were no symmetry. Strictly speaking, the gains in using permutational symmetry are restricted to halving the number of elements in these matrices. Here the use of the *petite liste* algorithm applied to the permutational symmetry in matrix multiplications is discussed.

The *petite liste* algorithm is a well-known procedure of treating spatial symmetry in *ab initio* molecular calculations. It is based on point group theory and is extensively reported in the literature [1–5]. In this methodology all the redundancy due to

the spatial symmetry is eliminated and a unique list of elements is collected under the name of *petite liste (Pl)*. This reduced list has the property of regenerating the whole set of elements under the effects of the projection operator. These elements are scaled by the equivalency number and all properties are evaluated as if there were no symmetry. After this preliminary step the "*skeleton*" property clearly shows an improper behavior under the effects of symmetry operations since it was created on the basis of a reduced list. The property may be corrected by a procedure, called "*symmetrization step*", that is performed through the application of a projection operator belonging to the proper irreducible representation. In the case of symmetric operators a fully symmetric projection operator, $\hat{\mathbb{P}}^A$, is used, but a general case might be treated through the pertinent irreducible representation X projector, as shown in (1):

$$\hat{\mathbb{P}}^X = \frac{1}{n_G} \sum_{R \subset G} \chi^X(R) \hat{R}. \tag{1}$$

In this equation $\hat{\mathbb{P}}^X$ is a projection operator belonging to the X-irreducible representation for a particular point group and χ stands for its character under the action of symmetry operation R. By G we mean the whole set of symmetry operations in the point group, which number n_G. For detailed information about the meaning of these variables we recommend the reading of one of the point group textbooks listed in references.

Despite being a technique in use for almost twenty years, we are not aware of any work discussing the extension of this methodology to the permutational symmetry. This article shows an alternative way of performing this matrix transformation that could possibly reduce the number of operations when the final transformed dimension is smaller than the original one. Russel and van Lenthe [6,7] get the credited for the originality of this algorithm since they obtained previously the same final equations. Here, an alternative view of the same problem is presented which can easily be extended to the discussion of general permutational symmetries of tensor transformations of order two or greater. This is also the simplest practical example of the *Pl* algorithm yet known.

For clarity's sake, in this article some notations of general use will be introduced. The values $m * (m + 1)/2$ will be denoted by the function $H(m)$. In order to set clearly all half-transformations performed Latin letters, i, j, k and l will be used to denote the original space indices while the Greek letters α, β, γ and δ will refer to variables belonging to the transformed space. When referring to matrices, S will be used for the general n-rank tensor represented in the original space, P for a generic half-transformed matrix and T for the final m-rank transformed matrix. The corresponding elements will be written in lower case letters. The order of the original matrices will be represented by N, while the variable M refers to the order of the final transformed ones.

The aim of this article is to present a simple discussion of the mathematical basis

of the permutational symmetry in matrix multiplications followed by some possible applications.

2. The permutational symmetry

Consider a regular unitary transformation given by eq. (2), where the original matrix S and the transformed matrix T show a regular permutational symmetry $s_{ij} = s_{ji}$ and $t_{\alpha\beta} = t_{\beta\alpha}$, and the matrix C represents the transformation between these spaces.

$$T = C^T S C . \tag{2}$$

To study this kind of permutational symmetry the two-index transformation must be represented as a single supermatrix multiplication where all S matrix elements are collected as a column vector v multiplied by a supermatrix, whose elements are given by $\mathbb{B}_{(\alpha\beta,ij)} = c_{i\alpha} c_{j\beta}$. The result for the operation $\mathbb{B}v$ generates u, the column-vector representation of T.

$$u = \mathbb{B}v , \tag{3}$$

$$\begin{pmatrix} t_{11} \\ t_{12} \\ \cdot \\ t_{1m} \\ \cdot \\ t_{\alpha\beta} \\ \cdot \\ t_{m1} \\ t_{m2} \\ \cdot \\ t_{mm} \end{pmatrix}_G = \begin{pmatrix} c_{11}c_{11} & c_{11}c_{21} & \cdot & \cdot & c_{n1}c_{n1} \\ c_{11}c_{12} & c_{11}c_{22} & \cdot & \cdot & c_{n1}c_{n2} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ c_{11}c_{1m} & c_{11}c_{2m} & \cdot & \cdot & c_{n1}c_{nm} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & c_{i\alpha}c_{j\beta} & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ c_{1m}c_{11} & c_{1m}c_{21} & \cdot & \cdot & c_{nm}c_{n1} \\ c_{1m}c_{12} & c_{1m}c_{22} & \cdot & \cdot & c_{nm}c_{n2} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ c_{1m}c_{1m} & c_{1m}c_{2m} & \cdot & \cdot & c_{nm}c_{nm} \end{pmatrix} \cdot \begin{pmatrix} s_{11} \\ s_{12} \\ \cdot \\ s_{1n} \\ \cdot \\ s_{ij} \\ \cdot \\ s_{n1} \\ s_{n2} \\ \cdot \\ s_{nn} \end{pmatrix}_G .$$

It may be observed that such a form, despite being computationally inadequate, will be useful as a guide to assess the *Pl* algorithm validity. Although this example is concerned with the two-index transformation, similar expressions might be obtained for the general case of an n-index transformation. It may be observed that the dimension of the v-space is n^2 while the dimension of the transformed u-space is m^2. Therefore, $\mathbb{B}$ will be, in general, a non-square matrix.

The matrix $\mathbb{B}$ does not show any apparent symmetry since its rows and columns are related to different indexes. However, all the permutational features of eq. (1) lead to permutational symmetries among rows and columns in this matrix. For

instance, the permutation of two rows is equivalent to the transposed permutations in the columns of this matrix, i.e. $\mathbb{B}_{(\beta\alpha,ij)} = \mathbb{B}_{(\alpha\beta,ji)}$, as shown in eq. (4):

$$\mathbb{B}_{(\beta\alpha,ij)} = c_{i\beta}c_{j\alpha} = c_{j\alpha}c_{i\beta} = \mathbb{B}_{(\alpha\beta,ji)} \,. \tag{4}$$

One shall collect the *unique* elements into the *petite liste*, v_P, of matrix elements. This *Pl* is defined such that, under the action of the projection operator, it regenerates the whole list of elements present in the original matrix. The projection operator representation, $\mathbb{P}$, depends on the dimensions and permutational symmetry of the elements but eq. (5) presents an acceptable representation for the v_P-space permutational projector of the two-index transformation of order two:

$$v_G = \mathbb{P}v_P \,, \tag{5}$$

$$\mathbb{P} = \{\mathbb{I} + \mathbb{O}\}/2 \,,$$

where $\mathbb{I}$ denotes the identity and $\mathbb{O}$ the permutation operator, $\hat{O}s_{ij} = s_{ji}$, in the representation defined by the ordered *grande liste* of the elements. The permutational group order is 2. This order is defined in such a way that S_{ij} precedes $S_{i'j'}$ if $i < i'$ and $j < j'$ when $i = i'$.

$$\mathbb{I} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}; \quad \mathbb{O} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} . \tag{6}$$

The projection operator, $\mathbb{P}$, is represented below:

$$\begin{pmatrix} t_{11} \\ t_{12} \\ t_{21} \\ t_{22} \end{pmatrix}_G = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 \\ 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} t_{11} \\ t_{12} \\ t_{21} \\ t_{22} \end{pmatrix}_P . \tag{7}$$

The expression shows that any arbitrary choice of the *petite liste* elements lead, under the effects of the projection operator, to the same *grande liste* of elements. The sets $\{S_{11}, 2 * S_{12}, 0, S_{22}\}$ and $\{S_{11}, 0, 2 * S_{12}, S_{22}\}$ are equally acceptable propositions for the *petite liste*. The constituency factor multiplying the non-diagonal elements are remarkable in the sets above. Under the effects of the projection operator, all of them generate the proper *grande liste* of elements. Of course, the desired choice is to make the elements in v_P assume null values as far as possible in order to save computation time. It is also interesting to note the net effect of the symmetrization step employed. Each element is the result of an average among the permuted equivalent elements.

The *petite liste* algorithm is based on the symmetrization of the transformed space, u_P, calculated over the *petite liste* of two-index elements, v_P, i.e.,

$$u_G = \mathbb{P}^{(m)} u_P \,. \tag{8}$$

In this equation a superscript m to indicate the projection operator dimension was aggregated. The identity above can be transformed into a more convenient form by using the definition of the transformed matrix, u_P, presented in eq. (3):

$$u_G = \mathbb{P}^{(m)} u_P = \mathbb{P}^{(m)} \mathbb{B} v_P \,. \tag{9}$$

Comparing expression (9) with its equivalent definition for u_G from the *Pl* of elements $\mathbb{P}^{(n)} v_P$ one has

$$u_G = \mathbb{B} v_G = \mathbb{B}\mathbb{P}^{(n)} v_P \,, \tag{10}$$

leading to the identity

$$\mathbb{P}^{(m)}\mathbb{B} = \mathbb{B}\mathbb{P}^{(n)} \,. \tag{11}$$

This equation is the matrix form of eq. (4), relating the action of the permutation operator in both spaces and is a consequence of the permutational symmetries observed in the original and the transformed matrices. Eq. (11) presents a simple demonstration for the use of permutational *Pl*. Therefore present algorithms might be modified by using just the *unique* scaled elements followed by a final symmetrization step represented by eq. (8). As the final space might be of smaller dimensions, the overall cost could be improved by a factor close to two. This factor does not consider improvements related to vectorization or *cache*-oriented algorithms. In the next section some potential applications will be analyzed.

3. Applications

3.1. TWO-INDEX TRANSFORMATIONS

A regular unitary transformation given by eq. (2), where the original matrix S and the final transformed matrix T show the same permutational symmetry, is considered. In order to have this transformation computationally optimized, the overall sum must be broken into two smaller half-transformations as shown by eqs. (12) and (13):

$$P = SC \,, \tag{12}$$

$$T = C^T P \,. \tag{13}$$

Algorithm 1 presents an ordinary computational program based on formulas (12) and (13). There was no attempt to exploit vectorization, *cache* memory or specific improved algorithms [9]. The computational cost of these two algorithms can be compared by counting the number of multiplications, $N^2 * M + M^2 * N/2 + M * N/2$ (roughly $3 * N^3/2$ if $N = M$) as shown in Algorithm 1. It is interesting

```
      Integer  I,J,K,IA,IB,N,M
C
      Do I=1,N
      Do J=1,N
      Do IA=1,M
      P(I,IA)=P(I,IA)+S(J,I)*C(J,IA)
      end do
      end do
C                                                          N**2 * M
C     ---- Second transformation
C
      Do IA=1,M                                  Upper part evaluation
      Do IB=1,IA
      VAL=0.0
      Do K=1,N
      VAL=VAL+P(K,IA)*C(K,IB)
      end do
      T(IA,IB)=VAL
      end do
C                                                          H(M)*N
C
C     ---- Total
C                                        N**2 * M + M**2 * N/2 + M * N/2
C
```

Algorithm 1. Fluxogram showing a sequence of FORTRAN commands for a two-index transformation. On the right side we indicate the number of operations in each step. The initialization of variables was omitted. Comments were added.

to notice that the use of permutational symmetry is restricted to the second transformation, allowing the half-calculation for the transformed matrix. The permutational symmetry could be used in the original matrix as well by storing the upper part of the S matrix only. However, it is pointed out that this expedience does not lead to any improvement in the number of multiplications performed since each element, prior to the calculation, must generate the equivalent elements.

A second algorithm based on the use of the *petite liste* is presented. The matrix elements are scaled by its constituency number, formally the number of equivalencies under permutational symmetry (all non-diagonal elements are multiplied by two!) and the *unique* elements are then processed normally, with no concern for regenerating the equivalent elements prior to the first transformation. The symmetrization step is performed at the end by averaging the permuted elements, eq. (1), as $T^G_{\alpha\beta} = \{T^P_{\alpha\beta} + T^P_{\beta\alpha}\}/2$.

The gains with this procedure can be compared to Algorithm 1 through the two half-transformations. In the first, the multiplication is performed over a restricted number of elements with a saving of $H(N) * M$ operations. In the second half-transformation this method shows a small overhead due to the need to create a full transformed matrix in spite of the triangular part. The final number of multiplications by the end of the process is kept the same if the dimension of the final transformed space is the same as the original one, i.e. $3N^2/2 + N^2/2$. However, it is important to remark that, when the transformed space dimensions are smaller than the original number, this algorithm can provide a speed-up close to 2 when com-

```
      Integer  I,J,K,IA,IB,Ndelta(I,J),N,M
C
      Do I=1,N
      Do J=1,I
      SIJ=S(J,I)/(NDELTA(I,J)+1)                          Diff. Scaling
      Do IA=1,M
      P(I,IA)=P(I,IA)+SIJ*C(J,IA)                 Single Multiplication
      end do
      end do
C                                                              H(N) * M
C     ---- Second transformation
C
      Do IA=1,M                                        Full T evaluation
      Do IB=1,M
      VAL=0.0
      Do K=1,N
      VAL=VAL+P(K,IA)*C(K,IB)
      end do
      T(IA,IB)=VAL
      end do
C                                                                M^2 * N
C     ---- Symmetrization
C
      Do IA=1,M
      Do IB=1,IA
      VAL=T(IA,IB)+T(IB,IA)                          Permutational Sym.
      T(IA,IB)=VAL
      T(IB,IA)=VAL
      end do
C
C     ---- Total
C                                    N^2* M/2 + N * M^2 + M * N/2
C
```

Algorithm 2. Fluxogram showing a sequence of FORTRAN statements of a two-index transformation exploiting the permutational symmetry through the *Pl* algorithm. On the right side we indicate the number of operations in each step. *IA* and *IB* denote indexes of the transformed spaces. Initialization was omitted. Comments were added.

pared to the regular transformation described in the first algorithm. Restricted sub-space use is rather frequent in many fields and coupling it with the present methodology would provide faster results and a solid theoretical background for matrix multiplications with arbitrary permutational symmetries.

In Algorithm 2 slightly different scaling factors are adopted and all original elements were halved in order to avoid a final multiplication by 1/2 at the symmetrization step. These scaling factors were suggested by Saunders and van Lenthe [7], who achieved similar equations working with configuration interaction calculations.

3.2. FOUR-INDEX TRANSFORMATION [6,7]

Most of the semi-empirical and the *ab initio* methods handle and process two-electron repulsion integrals, $(i, j/k, l)$. Their number grows with the fourth power of the basis function number and, due to this explosive scaling, much effort has been devoted to processing of existing symmetries in such integrals.

Concerning the permutational symmetries, these repulsion integrals show the following permutational symmetries, $(i, j/k, l) = (i, j/l, k) = (j, i/k, l) = (j, i/l, k) = (k, l/i, j) = (k, l/j, i) = (l, k/i, j) = (l, k/j, i)$, which, if considered, may provide a speed-up factor of close to eight in basis transformations of these properties. Besides the consideration of the permutational symmetries involved, the optimization of this transformation step is absolutely necessary since the straightforward multiplication would scale with the eighth power of the basis function. In fact, optimized algorithms exist in which the full transformation is performed through four half-transformations [6]. The overall cost of this algorithm comes down to nearly the fifth power of the same basis function number. Saunders and van Lenthe explored the permutational symmetry by the factorization of the S matrix into its upper and lower triangular parts. The method provides an interesting exploitation of these symmetries.

3.3. SUB-SPACE REDUCTION [10–12]

In many fields, the iterative diagonalization of large symmetric matrices is a common numerical step. Among the methods dealing with iterative space reductions, one may cite the coordinate relaxation based methods and those dealing with general Krylov based sequences. Such methods involve the iterative sub-space reduction from the original space to the one formed by some trial eigenvectors. Since this reduction involves the evaluation of matrix elements among iterative vectors, (v_i/Av_j), where the sub-space dimensions are often much smaller than the original ones, it would be helpful to use permutational symmetries as presented in this article. Savings of up to a factor of two in the matricial multiplications could be obtained.

4. Conclusions

As a general conclusion, computer codes involving intensive matrix multiplications of this kind might make use of the permutational symmetry as presented here. Work is in progress in order to provide routines for vectorization and *in-cache* oriented multiplications to be used in a new molecular computer program. This article has been written in the belief that the use of symmetry through the *petite liste* algorithm might be reasonably explored in other fields, for which historically such developments were not so decisive.

Acknowledgements

I would like to acknowledge Dr. José Karam Filho (LNCC/CNPq) for helpful comments and suggestions and Fundação Vitae and CNPq for financial support.

References

[1] R.M. Pitzer, J. Chem. Phys. 58 (1973) 3111.
[2] P.D. Dacre, Chem. Phys. Lett. 7 (1970) 47.
[3] M. Elder, Int. J. Quant. Chem. 7 (1973) 75.
[4] M. Dupuis and H.F. King, I, Int. J. Quant. Chem. 11 (1977) 613.
[5] E. Hollauer and M. Dupuis, J. Chem. Phys. 96 (1992) 5220.
[6] S. Wilson, Four-index transformations, *Methods in Computational Chemistry*, ed. S. Wilson (Plenum Press, New York, NY, 1987) p. 251.
[7] V.R. Saunders and J.H. van Lenthe, Molec. Phys. 48 (1983) 923.
[8] H. Sanukawa, IBM Syst. J. 27 (1988) 453.
[9] I. Shavitt, C.F. Bender, A. Pipano and R.P. Hosteny, J. Comp. Phys. 11 (1973) 90.
[10] R.C. Raffenetti, J. Comp. Phys. 32 (1979) 403.
[11] J. Cullum and R.A. Willoughby, J. Comp. Phys. 44 (1981) 329.