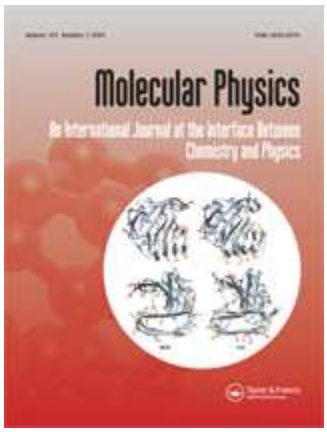
A complete list of symmetry adapted expressions to the fourth power for compact bending potentials in molecules with C3v and Td symmetry from a general symbolic algebra program

Roberto Marquardt, Kenneth Sagui



A complete list of symmetry adapted expressions to the fourth power for compact bending potentials in molecules with C3v and Td symmetry from a general symbolic algebra program

Journal:	*Molecular Physics*
Manuscript ID:	TMPH-2006-0112.R1
Manuscript Type:	Full Paper
Date Submitted by the Author:	24-Jan-2007
Complete List of Authors:	Marquardt, Roberto; Université Louis Pasteur, Institut de Chimie LC3-UMR7177-CNRS/ULP Sagui, Kenneth; Université de Marne-la-Vallée
Keywords:	symmetry adapted coordinates, reduction of tensor products, potential energy surfaces
Note: The following files were submitted by the author for peer review, but cannot be converted to PDF. You must view these files (e.g. movies) online.	
TMPH-2006-0112.R1.tex TMPH-2006-0112.R1-sup.tex	

A complete list of symmetry adapted expressions to the fourth power for compact bending potentials in molecules with $\mathbf{C}_{3v}$ and $\mathbf{T}_d$ symmetry from a general symbolic algebra program

Roberto Marquardt* and Kenneth Sagui

Laboratoire de Chimie Theorique, Université de Marne-la-Vallée

5 Bd Descartes (Champs-sur-Marne), F-77454 Marne-la-Vallée CEDEX 2, France

Prepared for publication in *Molecular Physics*

Pavel Rosmus special issue

with 10 manuscript pages, 2 Tables, 2 Appendices

version January 24, 2007

* present address and email: roberto.marquardt@chimie.u-strasbg.fr

correspondence and proofs to Prof. Roberto Marquardt

Laboratoire de Chimie Quantique - Institut de Chimie - LC3 UMR 7177 CNRS/ULP

Université Louis Pasteur - 4, rue Blaise Pascal - 67000 STRASBOURG - France

Fax 0(033)3 90 24 15 89

Abstract

An algorithm is proposed to perform the reduction of direct product representations of finite groups and, in particular, to perform the complete symmetry adaption of power representations, which may be used to obtain compact analytical representations of potential energy surfaces following an idea described in [R. Marquardt and M. Quack, *J. Chem. Phys.* 109, 10628 (1998)]. The algorithm is general in the sense that it can be applied to any finite group being characterized by its set of irreducible representations. It is automatic in the sense that, in case the reduction yields multiple degenerate irreducible subspaces of the same species, all degenerate irreducible subspaces are obtained with a coherent phase relation. The algorithm is based on the standard reduction rule of traditional representation theory. A symbolic algebra computer program based on `MAPLE` is presented and applied here to obtain the complete list of symmetry adapted expressions of bond angle coordinates to up to the fourth power in all irreducible representations of the $\mathbf{C}_{3v}$ and $\mathbf{T}_d$ point groups.

PACS: 31.15.Hz, 31.50.-x, 02.70.Wz, 02.20.Hj

Keywords: symmetry adapted coordinates, reduction of tensor products, potential energy surfaces

1 Introduction

The derivation of potential energy surfaces from *ab initio* calculations, and their accurate analytical representation in terms of functions of internal vibrational or reaction coordinates, has been of central interest to the work of Pavel Rosmus, the references [1–4] being some examples thereof. Analytical representations are quite often based on polynomial functions, which have the drawback of being interpolation formulae for *ab initio* derived energy points on the potential energy surface, rather than extrapolation formulae, and for adequate representations of large amplitude molecular rearrangements in highly anharmonic potentials, such as in B_4 [5] and SiC_3 [6], many high order polynomial terms are commonly needed.

While the description of anharmonicity of bond stretching potentials is well settled, e.g. with the use of Morse potentials [7] or alike, anharmonicity of bending potentials is considerably more difficult to describe analytically, in particular for highly symmetric molecules such as B_4. We have developed a method for deriving potential energy forms for bending vibrations that allow for the accurate description of large amplitude molecular motion and applied it to derive global analytical representations of methane [8, 9] and ammonia [10]. The derivation of these forms relies on extensive use of representation theory, for which a symbolic algebra program was developed. In particular, representation theory is used in this program to derive irreducible representations of products of coordinates that are used to build up compact potential energy forms [8, 10].

Both the derivation of the formulae and the program have not yet been published. The purpose of the present paper is to fill this gap and to show potential applications of the code to obtain symmetry adapted expressions of coordinates and products thereof for use in the derivation of potential energy surfaces for large amplitude molecular vibrations.

The program is based on a general reduction algorithm and is implemented as a symbolic algebra code within the `MAPLE` [11] program package. In the present version, the code is adapted to treat real representation spaces of some important finite point groups. Extension to other point groups has not yet been programmed, but is straightforward. The program is made available as supplementary material and the list of groups included

so far is given in a user guide.

The paper is organized as follows: In section 2 the basic steps used in the program are described, first by giving a brief review of the well known reduction procedure in section 2.2; then by explaining, in section 2.3, a new idea used to perform also a complete reduction in the representation space of the symmetric group, to reduce symmetric powers of representations, in addition to the reduction in the molecular point group. Section 3 describes, as an application, results for all symmetry adapted coordinates of power representations to up to the fourth power that arise in the context of the derivation of analytical potential energy surfaces for polyatomic molecules having $\mathbf{C}_{3v}$ and $\mathbf{T}_{d}$ point group symmetries. The results obtained finally are formulated in such a way that they can be interpreted independently of the underlying physical problem of the representation of potential energy surfaces and may indeed be used quite generally.

2 Theory

2.1 Representation theory

Symmetry is treated mathematically in group theory, and the representation of groups in vector spaces, the subject of representation theory, plays a major role in the study of the physical laws of nature [12]. A key question in this study is the reducibility of representations into simple sets of irreducible representations that will serve to label and to classify typical properties of the physical systems obeying these laws.

The reduction of a representation is performed by a linear transformation of the representation vector space such as to yield subspaces that are invariant under operations of the symmetry group. This procedure is called *symmetry adaption* and deriving symmetry adapted coordinates is one important application of representation theory. While the existence of the linear transformation is proven in representation theory, and the necessary tools to reduce representations are in principle well established [12–14], it is often quite difficult to calculate the transformation matrix in practice. Such a work can be performed more easily using a symbolic algebra computer program.

A commonly used procedure to obtain the transformation matrix for the reduction of a representation is to apply projection operators and ortho-normalize the projected vectors. Ortho-normalization yields vectors modulo a constant phase factor. In addition, projection into degenerate irreducible representation subspaces and subsequent ortho-normalization yields a set of orthonormal vectors that is unique only to up to an unitary transformation in the corresponding subspace. Transformation matrices resulting from the projection and ortho-normalization steps can therefore be different matrices, rather than unique, which are similar to each other. The non-uniqueness in the ortho-normalization has no consequences for non-degenerate irreducible representations. However, if a representation is to be reduced into multiple degenerate irreducible representations of the same species, the labeling of the irreducible vectors is ambiguous, and the relative phases of corresponding irreducible vectors may lack coherence.

The present work uses a standard reduction algorithm, which is based on the "generalized projection operator" [14, Eq. (4.52)] (see also Eq (3-184) in [13]). This algorithm is summarized in section 2.2.

We wish to note that the present code does not replace traditional methods for reducing powers of representations based on hand computation and using published tables of Clebsh-Gordan coefficients and for the reduction of direct products (see e.g. [15]). However, for those groups contained in the current version of the code, the computation is easier than by hand.

We also mention that the present algorithm does not solve the "missing label problem", since ambiguities remain because of the non-uniqueness in the ortho-normalization procedure. For a discussion of this problem and alternative methods for the reduction of representations we refer to [16]. However, this algorithm offers a straightforward means to obtain a coherent set of reduced representation spaces and symmetry adapted coordinates.

2.2 Reduction procedure

Representation theory asserts first that [12, 13], for a given finite symmetry group $\mathcal{G}$ (we shall constrain our considerations to finite groups in this paper) there is a finite number $N_{\rm irr}$ of irreducible representations $\mathcal{D}^{(\mu)} = \{\underline{\underline{D}}^{(\mu)}(g) \mid g \in \mathcal{G}\}$ of $\mathcal{G}$, each one defined in a $\mathbb{C}$-vector space V_{d_μ} of dimension d_μ ($\mu = 1, \ldots, N_{\rm irr}$); $\underline{\underline{D}}^{(\mu)} \in {V_{d_\mu}}^{V_{d_\mu}}$ are $d_\mu \times d_\mu$ unitary matrices. Secondly, if $\Gamma = \{\boldsymbol{\Gamma}(g) \mid g \in \mathcal{G}\}$ is any representation of $\mathcal{G}$ in a $\mathbb{C}$-vector space V_n of dimension n ($\boldsymbol{\Gamma} \in V_n^{V_n}$ are $n \times n$ unitary matrices), then to reduce such a representation into direct sums of irreducible representations means to find a unitary $n \times n$ matrix $\boldsymbol{Z}$, such that the transformed matrices

$$\boldsymbol{\Gamma}^{\rm red}(g) = \boldsymbol{Z}^\dagger \cdot \boldsymbol{\Gamma}(g) \cdot \boldsymbol{Z}, \tag{1}$$

where the dagger indicates Hermitean conjugation, are block-diagonal matrices composed of blocks for each irreducible representation and, within a given irreducible representation block labelled by the symbol μ, of γ_μ identical block-diagonal matrices $\underline{\underline{D}}^{(\mu)}(g)$, for all $g \in \mathcal{G}$. The number γ_μ is also called *reduction factor* of the irreducible representation $\mathcal{D}^{(\mu)}$ in Γ.

The transformation matrix $\boldsymbol{Z}$ may be composed of column vectors $\boldsymbol{z}^{(\mu;k;\alpha)}$, of column length n, where the superindices indicate the number μ of the irreducible representation, and running indices $k = 1, \ldots, \gamma_\mu$, and $\alpha = 1, \ldots, d_\mu$. The transformation matrix $\boldsymbol{Z}$ can then be obtained on the basis of "generalized projection operators" [13, equation (3-184) therein] $\boldsymbol{P}^{(\mu)}(\alpha, \alpha')$, defined here as

$$\boldsymbol{P}^{(\mu)}(\alpha, \alpha') = \frac{d_\mu}{\mid \mathcal{G} \mid} \sum_{g \in \mathcal{G}} {D^{(\mu)}_{\alpha\alpha'}}^*(g) \cdot \boldsymbol{\Gamma}(g), \tag{2}$$

where $D^{(\mu)}_{\alpha\alpha'}(g)$ is an element of the irreducible representation matrix $\underline{\underline{D}}^{(\mu)}(g)$. A reduction algorithm using these operators could then look as follows:

Step 1: For a given value of α (i.e. $\alpha = 1$), get column vectors $\boldsymbol{z}^{(\mu;k;\alpha)}$ by first applying $\boldsymbol{P}^{(\mu)}(\alpha, \alpha)$ from Eq. (2) on all basis vectors of V_n, and then applying an ortho-normalization procedure, such as that of Gram-Schmidt [17], on the resulting projected vector space. In practice, one has to ortho-normalize all column vectors

of $\boldsymbol{P}^{(\mu)}(\alpha,\alpha)$ to obtain γ_μ ortho-normalized vectors $\boldsymbol{z}^{(\mu;k;\alpha)}$, $k=1,\ldots,\gamma_\mu$. An alternative to the ortho-normalization procedure would be the diagonalization of an appropriate totally symmetric hermitian operator [18, 19].

Step 2: For k from 1 to γ_μ, and for $\beta \neq \alpha$ (i.e. for β from 2 to d_μ), get column vectors $\boldsymbol{z}^{(\mu;k;\beta)}$ by applying $\boldsymbol{P}^{(\mu)}(\beta,\alpha)$ from Eq. (2) on $\boldsymbol{z}^{(\mu;k;\alpha)}$,

$$\boldsymbol{z}^{(\mu;k;\beta)} = \boldsymbol{P}^{(\mu)}(\beta,\alpha)\cdot\boldsymbol{z}^{(\mu;k;\alpha)}, \tag{3}$$

within individual manifolds of equal k-labels.

Based on the orthogonality relations of irreducible representation matrices [13, Eq (3-143)], it can be shown that the n vectors obtained in this way define indeed an orthonormal basis:

$$\boldsymbol{z}^{(\mu;k;\alpha)^\dagger}\cdot\boldsymbol{z}^{(\mu';k';\alpha')} = \delta_{\mu\mu'}\,\delta_{kk'}\,\delta_{\alpha\alpha'}. \tag{4}$$

For a given coordinate vector $\boldsymbol{x}\in V_n$, the expressions

$$s_\alpha^{(\mu;k)} = \boldsymbol{z}^{(\mu;k;\alpha)^\dagger}\cdot\boldsymbol{x} \quad (\alpha=1,\ldots,d_\mu) \tag{5}$$

form a coherent set of d_μ symmetry adapted coordinates, for a given value of μ and k.

2.3 Direct product representations

In this section we use a simple superscript $^{(n)}$ on representation symbols in order to assign the dimension of the representation. Boldfaced symbols are used for matrices; a simple superscript indicates the rank of squared matrices, a double superscript indicates the number of rows and columns of rectangular matrices. Let $\Gamma^{(n)}$ be a product representation in a vector space V_n, which we interpret as a tensor space, for instance $V_n = V_{n_1}\otimes V_{n_2}$, with $n = n_1\times n_2$, and $\Gamma^{(n)} = \Gamma^{(n_1)}\otimes\Gamma^{(n_2)}$. Elements of representation matrices $\boldsymbol{\Gamma}^{(n)}(g)\in\Gamma^{(n)}$ are related to elements of representation matrices $\boldsymbol{\Gamma}^{(n_1)}(g)\in\Gamma^{(n_1)}$ and $\boldsymbol{\Gamma}^{(n_2)}(g)\in\Gamma^{(n_2)}$ as follows:

$$\Gamma^{(n)}_{k_r k_c}(g) = \Gamma^{(n_1)}_{i_{1r}(k_r)i_{1c}(k_c)}(g)\;\Gamma^{(n_2)}_{i_{2r}(k_r)i_{2c}(k_c)}(g), \tag{6}$$

where $k_r,k_c = 1,\ldots,n$, $i_{1r},i_{1c}=1,\ldots,n_1$ and $i_{2r},i_{2c}=1,\ldots,n_2$; the subindexes r and c differ row and column indexes. The algorithm proposed in section 2.2 may be used to reduce such representations.

In the special case when $\Gamma^{(n_1)} = \Gamma^{(n_2)}$, there is an additional symmetry related to the permutation of identical factors. It is then possible to consider reducing the representation in the symmetric group $\mathbf{S}_2$, in addition to reducing it in $\mathcal{G}$. Indeed, for commutative algebras only the A_1 symmetry adapted coordinates in $\mathbf{S}_2$ will have non-vanishing components in V_n (A_1 is the totally symmetric or trivial representation). Therefore, before reducing $\Gamma^{(n)}$ in $\mathcal{G}$, it is useful to reduce V_n to the space spanned by A_1-vectors.

In general, a power direct product representation of order m ($m = 2, 3, 4, \ldots$), $\Gamma^{(n^m)} = \Gamma^{(n)\otimes m}$, of any group $\mathcal{G}$ gives also rise to a representation $\Sigma^{(n^m)}$ of the symmetric group $\mathbf{S}_\mathrm{m}$, the construction of which is described below. A product representation matrix element of order m is calculated as

$$\Gamma^{(n^m)}_{k_r k_c}(g) = \Gamma^{(n)}_{i_{1r}(k_r) i_{1c}(k_c)}(g) \cdot \ldots \cdot \Gamma^{(n)}_{i_{mr}(k_r) i_{mc}(k_c)}(g). \tag{7}$$

Let $I_m = (i_1, \ldots, i_m)$ be a m-tuple of indexes $i_j = 1, \ldots, n$ ($j = 1, \ldots, m$) that describes a basis state vector of the tensor space V_{n^m}. There are n^m m-tuples of this form:

$$I_m(k) = (i_1(k), \ldots, i_m(k)) \quad k = 1, \ldots, n^m; \tag{8}$$

$i_j(k) = 1, \ldots, n; \;\; j = 1, \ldots, m$. We call the assignment $k \rightarrow I_m(k)$ in Eq. (8) a *product representation key*. The definition of the key implies the calculation of the elements of the product representation matrix from the factor matrix elements in Eq. (7).

Consider a specific key component $I_m(k)$. A given operation $p \in \mathbf{S}_\mathrm{m}$ performs a permutation of indexes i_j in $I_m(k)$ which will correspond in a unique way to a new key component $I_m(k')$. Representation matrix elements $\Sigma^{(n^m)}_{kk'}(p)$ will therefore have the values 1 or 0, depending on whether k and k' are connected by p or not. We calculate such matrices analytically using the regular representation of $\mathbf{S}_\mathrm{m}$ (the "Turmdarstellung").

$\Sigma^{(n^m)}$ is reducible in $\mathbf{S}_\mathrm{m}$, and the reduction will always contain the A_1-representation. The algorithm proposed in section 2.2 may be used to to reduce this representation as well. A simpler way of obtaining the A_1-subspace of any representation space is, however, to use the projection operator introduced by Wigner [12, Eq. (12.12a) therein]. In practice, we proceed as follows:

Step 1: Let γ be the multiplicity of the $A_1(\mathbf{S}_\mathrm{m})$-representation of the Symmetric group

$\mathbf{S}_\mathrm{m}$ in the representation $\Sigma^{(n^m)}$, and let V_γ be the $\mathrm{A}_1(\mathbf{S}_\mathrm{m})$ invariant subspace of V_{n^m}. This subspace is defined in V_{n^m} by the matrix $\boldsymbol{T}^{(n^m;\gamma)}$, which has γ orthonormal columns and n^m rows. Calculate $\boldsymbol{T}^{(n^m;\gamma)}$ from the orthonormalization of

$$\boldsymbol{P}^{(n^m)}_{\mathrm{A}_1(\mathbf{S}_\mathrm{m})} = \sum_{g\in\mathbf{S}_\mathrm{m}} \boldsymbol{\Sigma}^{(n^m)}(g) \tag{9}$$

Then reduce $\Gamma^{(n)^{\otimes m}}$ to obtain the representation $\Gamma^{(\gamma)}$ of $\mathcal{G}$ in this subspace, e.g. for all $g \in \mathcal{G}$, calculate representation matrices

$$\boldsymbol{\Gamma}^{(\gamma)}(g) = \boldsymbol{T}^{\dagger(\gamma;n^m)} \cdot \boldsymbol{\Gamma}^{(n^m)}(g) \cdot \boldsymbol{T}^{(n^m;\gamma)}. \tag{10}$$

Step 2: Let then $\boldsymbol{Z}^{(\gamma)}$ be the unitary matrix of rank γ that reduces $\Gamma^{(\gamma)}$ in $\mathcal{G}$. This matrix is obtained as described in the procedure of section 2.2. $\mathcal{G}$-symmetry adapted coordinates $\boldsymbol{s}^{(\gamma)} \in V_\gamma$ are then calculated from Eq. (11), for a given coordinate vector $\boldsymbol{x}^{(n^m)} \in V_{n^m}$:

$$\boldsymbol{s}^{(\gamma)} = \boldsymbol{Z}^{\dagger(\gamma)} \cdot \boldsymbol{T}^{\dagger(\gamma;n^m)} \cdot \boldsymbol{x}^{(n^m)}. \tag{11}$$

As in section 2.2, components of $\boldsymbol{s}^{(\gamma)}$ may be partitioned into distinguished sets of coordinates that are internally coherent. These sets may be labelled according to the irreducible representation species μ of $\mathcal{G}$ and some additional multiplicity index k.

3 Results

The algorithm described in section 2.2 was used, as an application, to find all symmetry adapted expressions, according to Eq. (11), of products of symmetry adapted bending coordinates of methane [8,20] and ammonia [10]. The symmetry adapted coordinates used here are well known from spectroscopic studies [21]. They may as well be obtained using the general reduction algorithm of section 2.2 upon appropriate definition of representation matrices of the $\mathbf{T}_\mathrm{d}$ and $\mathbf{C}_\mathrm{3v}$ groups in the space of bond angle coordinates.

The validity of these expressions was tested by checking that the numerically obtained matrices $\boldsymbol{\Gamma}^{\mathrm{red}}(g)$ (Eq. (1)) are exactly blockdiagonal, for all $g \in \mathcal{G}$, and have identical block elements given by the matrices $\underline{\underline{D}}^{(\mu)}$. This test allows to find inconsistencies in the

computer algebra approach. Furthermore it is fundamental to ensure the coherence of all vectors. Coherent vectors are relevant for the reduction of tensor operators in many physical problems. For a further discussion on the relevance of coherent vectors for the derivation of potential energy surfaces see ref. [10].

All expressions given here are direct results from the numerical program, but were further simplified by hand computation, where constant multiplicative factors have been omitted.

3.1 NH_3

Ammonia (NH_3) has a $\mathbf{C}_{3v}$ point group structure at equilibrium (the stable stationary point on the potential energy surface). Symmetry adapted bending coordinates are [21]

$$SA1 = \frac{1}{\sqrt{3}}(\Delta\alpha_{12} + \Delta\alpha_{13} + \Delta\alpha_{23}) \tag{12}$$

$$SE = \begin{cases} SEa = \frac{1}{\sqrt{6}}(2\Delta\alpha_{12} - \Delta\alpha_{13} - \Delta\alpha_{23}) \\ SEb = \frac{1}{\sqrt{2}}(\Delta\alpha_{13} - \Delta\alpha_{23}) \end{cases} \tag{13}$$

where $\Delta\alpha_{ij} = \alpha_{ij} - \alpha_{ij}^{\mathrm{eq}}$ are displacements of bond angles α_{ij} between NH bonds i and j $(i, j = 1, \ldots, 3)$ from equilibrium values $\alpha_{ij}^{\mathrm{eq}}$.

The second order product representation space has dimension 9. Reduction in $\mathbf{S}_2$ yields a six-dimensional space which splits into a two-dimensional space of type A_1 and a two-dimensional space of type E. Symmetry adapted coordinates are

$$\left.\begin{aligned} SA1^{\{2\}} &= SA1^2 \\ SA1^{\{2\}} &= SEa^2 + SEb^2 \\ SEa^{\{2\}} &= (SEa - SEb)\,(SEb + SEa) \\ SEb^{\{2\}} &= -2\,SEa\,SEb \\ SEa^{\{2\}} &= SA1\,SEa \\ SEb^{\{2\}} &= SA1\,SEb \end{aligned}\right\} \tag{14}$$

Product representation spaces of order 3 and 4 have dimensions 27 and 81, respectively. Table 1 summarizes the resulting reductions, first in $\mathbf{S}_2$, $\mathbf{S}_3$ and $\mathbf{S}_4$, then in $\mathbf{C}_{3v}$. Symmetry

adapted coordinates are summarized in appendix A. Tab. 1 here

3.2 CH_4

Methane (CH_4) has a $\mathbf{T}_\mathrm{d}$ point group structure at equilibrium. Symmetry adapted bending coordinates are [21]

$$SA1 = \frac{1}{\sqrt{6}}(\Delta\alpha_{12} + \Delta\alpha_{13} + \Delta\alpha_{14} + \Delta\alpha_{23} + \Delta\alpha_{24} + \Delta\alpha_{34}) \quad (15)$$

$$SE = \begin{cases} SEa = \frac{1}{\sqrt{12}}(2(\Delta\alpha_{12} + \Delta\alpha_{34}) - (\Delta\alpha_{13} + \Delta\alpha_{24} + \Delta\alpha_{23} + \Delta\alpha_{14})) \\ SEb = \frac{1}{2}(\Delta\alpha_{13} + \Delta\alpha_{24} - \Delta\alpha_{23} - \Delta\alpha_{14}) \end{cases} \quad (16)$$

$$SF2 = \begin{cases} SFx = \frac{1}{\sqrt{2}}(\Delta\alpha_{12} - \Delta\alpha_{34}) \\ SFy = \frac{1}{\sqrt{2}}(\Delta\alpha_{13} - \Delta\alpha_{24}) \\ SFz = \frac{1}{\sqrt{2}}(\Delta\alpha_{14} - \Delta\alpha_{23}) \end{cases} \quad (17)$$

where $\Delta\alpha_{ij} = \alpha_{ij} - \alpha_{ij}^{\mathrm{eq}}$ are displacements of bond angles α_{ij} between CH bonds i and j $(i, j = 1, \ldots, 4)$ from equilibrium values $\alpha_{ij}^{\mathrm{eq}}$.

Product representation spaces of order 2, 3 and 4 have dimensions 36, 216 and 1296, respectively. Table 2 summarizes the resulting reductions, first in $\mathbf{S}_2$, $\mathbf{S}_3$ and $\mathbf{S}_4$, then in $\mathbf{T}_\mathrm{d}$. Symmetry adapted coordinates are summarized in appendix B. Tab. 2 here

4 Conclusions

The study of physical laws of nature often involves representations of symmetry groups in vector spaces. The reduction of such representations into simple sets of irreducible representations exploits a maximum of information embedded in the symmetry properties of these laws, which allows to label and classify typical properties of the corresponding physical systems. For instance, the structure and dynamics of polyatomic molecules may be studied on the basis of a classification of the complicated underlying spectroscopic data using properties of the symmetry inherent to these systems, such as the permutation of identical atoms [22–24]. The complete permutation-inversion group is also central to the derivation of selection rules for reactive collisions [25]. More recently, a new method for obtaining analytical representations of potential energy surfaces for polyatomic molecules was introduced, which requires the calculation of coherent sets of symmetry adapted coordinates corresponding to degenerate irreducible representations in high order tensor spaces [8, 10].

While the mathematical tools to reduce representations are well established [12, 13], it is often quite difficult and laborious to calculate the transformation matrices that yield the symmetry adapted coordinates in practice.

In the present work we have used a reduction algorithm based on projector operators such as the "generalized projection operator" [13]. The algorithm was implemented into a symbolic algebra program using the `MAPLE` [26] code, and applied to the problem of reducing direct product and power representations. The algorithm is general, as it can be used for any finite group defined by its set of irreducible representations. It is automatic in the sense that, in case the reduction yields multiple degenerate irreducible subspaces of the same species, all degenerate irreducible subspaces are obtained with a coherent phase relation. We note that the resulting standardization is similar to the one introduced for the construction of symmetry adapted vectors used in crystal- and ligand-field theory [27, 28]. For the reduction of a power representation $\Gamma^{\otimes m}$ in commutative algebras, the algorithm includes also reduction in the symmetric group $\mathbf{S}_{\mathrm{m}}$. The code allows currently to treat some of the finite groups, including the $\mathbf{C}_{2\mathrm{v}}$, $\mathbf{C}_{3\mathrm{v}}$ and $\mathbf{T}_{\mathrm{d}}$ groups, and powers of representations thereof to up to $m = 4$. An extension to include other groups is straightforward,

and we plan to apply it to derive a global potential energy surface for B_4, for which the $\mathbf{D}_{4h}$ group will play an important role [5]. The code may be of quite general use and is made available as supplementary material [29].

Acknowledgment

We thank Dr. D. Luckhaus and Dr. G. Dhont for early discussions, as well as Dr. V. Boudon, Dr. C. Leroy and Prof. F. Michelot for help and discussions in the final stage of this project. This work is supported financially by grants from the French Ministry of Research.

References

[1] U. Mänz, E. Reinsch, P. Rosmus, H. Werner, and S. O. Neil, J. Chem. Soc. Faraday Trans., **87**, 1809–1814 (1991).

[2] W. Gabriel, G. Chambaud, P. Rosmus, S. Carter, and N. C. Handy, Mol. Phys., **81**, 1445–1461 (1994).

[3] M. Rosenstock, P. Rosmus, E.-A. Reinsch, O. Treutler, S. Carter, and N. C. Handy, Mol. Phys., **93**, 853–865 (1998).

[4] P. Rosmus, P. Palmieri, and R. Schinke, J. Chem. Phys., **117**, 4871–4877 (2003).

[5] R. Linguerri, I. Navizet, P. Rosmus, S. Carter, and J. P. Maier, J. Chem. Phys., **122**, 34301 (2005).

[6] R. Linguerri, P. Rosmus, and S. Carter, J. Chem. Phys., **125**, 34305 (2006).

[7] P. M. Morse, Phys. Rev., **34**, 57–64 (1929).

[8] R. Marquardt and M. Quack, J. Chem. Phys., **109**, 10628–10643 (1998).

[9] R. Marquardt and M. Quack, J. Phys. Chem. A, **108**, 3166–3181 (2004).

[10] R. Marquardt, K. Sagui, W. Klopper, and M. Quack, J. Phys. Chem. B, **109**, 8439–8451 (2005).

[11] *Maple V Release 3 (ETH).* Copyright (c) 1981-1994 by Waterloo Maple Software and the University of Waterloo.

[12] E. Wigner. *Group Theory.* Academic Press, New York, (1959).

[13] M. Hamermesh. *Group Theory and Its Application to Physical Problems.* Addison-Wesley, Reading, Massachusetts, (1962).

[14] J. P. Elliot and P. G. Dawber. *Symmetry in Physics.* Oxford University Press, New York, (1979).

[15] S. L. Altmann and P. Herzig. *Point-group theory tables.* Clarendon, Oxford, (1994).

[16] J.-Q. Chen, M.-J. Gao, and G.-Q. Ma, Rev. Mod. Phys., **57**, 211–278 (1985).

[17] H.-J. Kowalsky. *Lineare Algebra.* de Gruyter, Berlin, New York, (1979).

[18] M. Rey, V. Boudon, C. Wenger, G. Pierre, and B. Sartakov, J. Mol. Spectrosc., **219**, 313–325 (2003).

[19] J.-P. Champion, G. Pierre, F. Michelot, and J. Moret-Bailly, Can. J. Phys., **55**, 512–520 (1977).

[20] R. Marquardt, , F. Mariotti, and M. Quack. Work in preparation.

[21] J. L. Duncan and I. M. Mills, Spectrochim. Acta, **20**, 523–546 (1964).

[22] G. Herzberg. *Molecular Spectra and Molecular Structure II. Infrared and Raman Spectra of Polyatomic Molecules.* Van Nostrand Reinhold Co., New York, reprint(1991) edition, (1945).

[23] G. Herzberg. *Molecular Spectra and Molecular Structure III. Electronic Spectra and Electronic Structure of Polyatomic Molecules.* Van Nostrand Reinhold Co., New York, reprint(1991) edition, (1966).

[24] P. R. Bunker. *Molecular Symmetry and Spectroscopy.* Academic Press, New York, (1979).

[25] M. Quack, Mol. Phys., **34**, 477–504 (1977).

[26] *Maple V Release 10.* Copyright (c) 2004-2005 by Maplesoft, a division of Waterloo Maple Software Inc.

[27] M. Kibler, Int. J. Quantum Chem., **3**, 795–822 (1969).

[28] M. R. Kibler, J. Mol. Spectrosc., **62**, 247–262 (1976).

[29] R. Marquardt and K. Sagui. REDTEN: A symbolic algebra program for the automatic reduction of powers of representations of finite groups. A hard copy version of the codes and user guide has been deposited with the British Library Document Supply Centre as Supplementary Publication No SUP ... (77 pages).

A C_{3v}-symmetry adapted expressions

A.1 Expressions of order 3

The product representation

$$\Gamma^{(3)} = \left(\mathcal{D}^{(A_1)} \oplus \mathcal{D}^{(E)}\right)^{\otimes 3} \tag{A.1}$$

reduces as

$$\Gamma^{\mathrm{red}(3)} = 3\,\mathcal{D}^{(A_1)} \oplus \mathcal{D}^{(A_2)} \oplus 3\,\mathcal{D}^{(E)}. \tag{A.2}$$

Symmetry adapted expressions are:

μ	k			
A_1	1	$SA1^{\{3\}}$	=	$SA1^3$
A_1	2	$SA1^{\{3\}}$	=	$SA1\ \left(SEa^2 + SEb^2\right)$
A_1	3	$SA1^{\{3\}}$	=	$SEa\ \left(SEa^2 - 3\,SEb^2\right)$
A_2	1	$SA2^{\{3\}}$	=	$SEb\ \left(3\,SEa^2 - SEb^2\right)$
E	1	$SEa^{\{3\}}$	=	$SA1^2 SEa$
		$SEb^{\{3\}}$	=	$SA1^2 SEb$
E	2	$SEa^{\{3\}}$	=	$SA1\ (SEa + SEb)\,(SEa - SEb)$
		$SEb^{\{3\}}$	=	$-2\,SA1\ SEa\ SEb$
E	3	$SEa^{\{3\}}$	=	$\left(SEa^2 + SEb^2\right) SEa$
		$SEb^{\{3\}}$	=	$\left(SEa^2 + SEb^2\right) SEb$

A.2 Expressions of order 4

The product representation

$$\Gamma^{(4)} = \left(\mathcal{D}^{(\mathrm{A}_1)} \oplus \mathcal{D}^{(\mathrm{E})}\right)^{\otimes 4} \tag{A.3}$$

reduces as

$$\Gamma^{\mathrm{red}(4)} = 4\,\mathcal{D}^{(\mathrm{A}_1)} \oplus \mathcal{D}^{(\mathrm{A}_2)} \oplus 5\,\mathcal{D}^{(\mathrm{E})}. \tag{A.4}$$

Symmetry adapted expressions are:

μ	k			
A_1	1	$SA1^{\{4\}}$	=	$SA1^4$
A_1	2	$SA1^{\{4\}}$	=	$\left(SEa^2 + SEb^2\right) SA1^2$
A_1	3	$SA1^{\{4\}}$	=	$SA1\ SEa\ \left(SEa^2 - 3\,SEb^2\right)$
A_1	4	$SA1^{\{4\}}$	=	$\left(SEa^2 + SEb^2\right)^2$
A_2	1	$SA2^{\{4\}}$	=	$SA1\ SEb\ \left(3\,SEa^2 - SEb^2\right)$
E	1	$SEa^{\{4\}}$	=	$1/2\,\sqrt{3}\,(SEa - SEb)\,(SEa + SEb)\,SA1^2$
		$SEb^{\{4\}}$	=	$-\sqrt{3} SA1^2 SEa\ SEb$
E	2	$SEa^{\{4\}}$	=	$SA1\ \left(SEa^2 + SEb^2\right) SEa$
		$SEb^{\{4\}}$	=	$SA1\ \left(SEa^2 + SEb^2\right) SEb$
E	3	$SEa^{\{4\}}$	=	$\frac{1}{300}\sqrt{10}\left(5\,SEa^2 - 3\,SEb^2 + 2\,SEb^2\sqrt{6}\right)\left(5\,SEa^2 - 3\,SEb^2 - 2\,SEb^2\sqrt{6}\right)\sqrt{3}$
		$SEb^{\{4\}}$	=	$-1/15\,\sqrt{10}\left(SEa^2 + 3\,SEb^2\right) SEa\ SEb\,\sqrt{3}$
E	4	$SEa^{\{4\}}$	=	$\left(3\,SEa^2 - SEb^2\right) SEb^2$
		$SEb^{\{4\}}$	=	$-\left(3\,SEa^2 - SEb^2\right) SEa\ SEb$
E	5	$SEa^{\{4\}}$	=	$SA1^3 SEa\,\sqrt{10}$
		$SEb^{\{4\}}$	=	$SA1^3 SEb\,\sqrt{10}$

B T_d-symmetry adapted expressions

B.1 Expressions of order 2

The product representation

$$\Gamma^{(2)} = \left(\mathcal{D}^{(A_1)} \oplus \mathcal{D}^{(E)} \oplus \mathcal{D}^{(F_2)}\right)^{\otimes 2} \tag{B.1}$$

reduces as

$$\Gamma^{\mathrm{red}(2)} = 3\,\mathcal{D}^{(A_1)} \oplus 3\,\mathcal{D}^{(E)} \oplus 1\,\mathcal{D}^{(F_1)} \oplus 3\,\mathcal{D}^{(F_2)} \tag{B.2}$$

Symmetry adapted expressions are:

μ	k			
A_1	1	$SA1^{\{2\}}$	=	$SA1^2$
A_1	2	$SA1^{\{2\}}$	=	$SEa^2 + SEb^2$
A_1	3	$SA1^{\{2\}}$	=	$SFx^2 + SFy^2 + SFz^2$
E	1	$SEa^{\{2\}}$	=	$SA1\ SEa$
		$SEb^{\{2\}}$	=	$SA1\ SEb$
E	2	$SEa^{\{2\}}$	=	$(SEa - SEb)\,(SEa + SEb)$
		$SEb^{\{2\}}$	=	$-2\,SEa\ SEb$
E	3	$SEa^{\{2\}}$	=	$2\,SFx^2 - SFy^2 - SFz^2$
		$SEb^{\{2\}}$	=	$\sqrt{3}\,(SFy - SFz)\,(SFy + SFz)$
F_1	1	$SFu^{\{2\}}$	=	$2\,SEb\ SFx$
		$SFv^{\{2\}}$	=	$-1/3\,\sqrt{3}\left(3\,SEa + \sqrt{3}SEb\right)\,SFy$
		$SFw^{\{2\}}$	=	$1/3\,\sqrt{3}\left(3\,SEa - \sqrt{3}SEb\right)\,SFz$
F_2	1	$SFx^{\{2\}}$	=	$2\,SEa\ SFx$
		$SFy^{\{2\}}$	=	$-SFy\ \left(SEa - \sqrt{3}SEb\right)$
		$SFz^{\{2\}}$	=	$-SFz\ \left(SEa + \sqrt{3}SEb\right)$
F_2	2	$SFx^{\{2\}}$	=	$SFy\ SFz$
		$SFy^{\{2\}}$	=	$SFx\ SFz$
		$SFz^{\{2\}}$	=	$SFx\ SFy$
F_2	3	$SFx^{\{2\}}$	=	$SA1\ SFx$
		$SFy^{\{2\}}$	=	$SA1\ SFy$
		$SFz^{\{2\}}$	=	$SA1\ SFz$

B.2 Expressions of order 3

The product representation

$$\Gamma^{(3)} = \left(\mathcal{D}^{(\mathrm{A}_1)} \oplus \mathcal{D}^{(\mathrm{E})} \oplus \mathcal{D}^{(\mathrm{F}_2)}\right)^{\otimes 3} \tag{B.3}$$

reduces as

$$\Gamma^{\mathrm{red}(3)} = 6\,\mathcal{D}^{(\mathrm{A}_1)} \oplus 2\,\mathcal{D}^{(\mathrm{A}_2)} \oplus 6\,\mathcal{D}^{(\mathrm{E})} \oplus 4\,\mathcal{D}^{(\mathrm{F}_1)} \oplus 8\,\mathcal{D}^{(\mathrm{F}_2)} \tag{B.4}$$

Symmetry adapted expressions are:

μ	k			
A_1	1	$SA1^{\{3\}}$	=	$SA1^3$
A_1	2	$SA1^{\{3\}}$	=	$SA1\ \left(SEa^2 + SEb^2\right)$
A_1	3	$SA1^{\{3\}}$	=	$SA1\ \left(SFx^2 + SFy^2 + SFz^2\right)$
A_1	4	$SA1^{\{3\}}$	=	$SEa\ \left(-3\,SEb^2 + SEa^2\right)$
A_1	5	$SA1^{\{3\}}$	=	$SEa\,SFx^2 - 1/2\,SEa\,SFy^2 - 1/2\,SEa\,SFz^2$
				$+1/2\,\sqrt{3}SEb\,SFy^2 - 1/2\,\sqrt{3}SEb\,SFz^2$
A_1	6	$SA1^{\{3\}}$	=	$SFx\,SFy\,SFz$
A_2	1	$SA2^{\{3\}}$	=	$SEb\ \left(3\,SEa^2 - SEb^2\right)$
A_2	2	$SA2^{\{3\}}$	=	$-2\,\sqrt{3}SEb\,SFx^2 + \sqrt{3}SEb\,SFy^2 + \sqrt{3}SEb\,SFz^2 + 3\,SEa\,SFy^2 - 3\,SEa\,SFz^2$
E	1	$SEa^{\{3\}}$	=	$SA1\ \left(SEa - SEb\right)\left(SEa + SEb\right)$
		$SEb^{\{3\}}$	=	$-2\,SA1\,SEa\,SEb$
E	2	$SEa^{\{3\}}$	=	$-\sqrt{3}SEb\,SFz^2 + \sqrt{3}SEb\,SFy^2 + 3\,SEa\,SFy^2 + 3\,SEa\,SFz^2$
		$SEb^{\{3\}}$	=	$SEb\,SFz^2 + SEb\,SFy^2 - \sqrt{3}\,SEa\,SFz^2 + 4\,SEb\,SFx^2 + \sqrt{3}\,SEa\,SFy^2$
E	3	$SEa^{\{3\}}$	=	$SA1^2 SEa$
		$SEb^{\{3\}}$	=	$SA1^2 SEb$
E	4	$SEa^{\{3\}}$	=	$SA1\ \left(2\,SFx^2 - SFy^2 - SFz^2\right)$
		$SEb^{\{3\}}$	=	$\sqrt{3}SA1\ \left(SFy + SFz\right)\left(SFy - SFz\right)$
E	5	$SEa^{\{3\}}$	=	$\left(SEa^2 + SEb^2\right) SEa$
		$SEb^{\{3\}}$	=	$\left(SEa^2 + SEb^2\right) SEb$
E	6	$SEa^{\{3\}}$	=	$4\,SEa\,SFx^2 + \sqrt{3}SEb\,SFz^2 - \sqrt{3}SEb\,SFy^2 + SEa\,SFy^2 + SEa\,SFz^2$
		$SEb^{\{3\}}$	=	$-\sqrt{3}\left(-\sqrt{3}SEb\,SFz^2 - \sqrt{3}SEb\,SFy^2 - SEa\,SFz^2 + SEa\,SFy^2\right)$

(cont.)

μ	k			
F_1	1	$SFu^{\{3\}}$	=	$SA1\ SEb\ SFx$
		$SFv^{\{3\}}$	=	$-1/6\sqrt{3}SA1\ \left(3\ SEa+\sqrt{3}SEb\right)\ SFy$
		$SFw^{\{3\}}$	=	$1/6\sqrt{3}SA1\ \left(3\ SEa-\sqrt{3}SEb\right)\ SFz$
F_1	2	$SFu^{\{3\}}$	=	$SFx\ (SFy+SFz)\,(SFy-SFz)$
		$SFv^{\{3\}}$	=	$-SFy\ (SFx-SFz)\,(SFx+SFz)$
		$SFw^{\{3\}}$	=	$SFz\ (SFx-SFy)\,(SFx+SFy)$
F_1	3	$SFu^{\{3\}}$	=	$SEb\ SFy\ SFz$
		$SFv^{\{3\}}$	=	$-1/6\sqrt{3}\left(3\ SEa+\sqrt{3}SEb\right)\ SFx\ SFz$
		$SFw^{\{3\}}$	=	$1/6\sqrt{3}\left(3\ SEa-\sqrt{3}SEb\right)\ SFx\ SFy$
F_1	4	$SFu^{\{3\}}$	=	$SEa\ SEb\ SFx$
		$SFv^{\{3\}}$	=	$1/12\sqrt{3}SFy\ \left(SEa-\sqrt{3}SEb\right)\left(3\ SEa+\sqrt{3}SEb\right)$
		$SFw^{\{3\}}$	=	$-1/12\sqrt{3}SFz\ \left(SEa+\sqrt{3}SEb\right)\left(3\ SEa-\sqrt{3}SEb\right)$

(cont.)

μ	k	
F_2	1	$SFx^{\{3\}} = SA1\ SFy\ SFz$
		$SFy^{\{3\}} = SA1\ SFx\ SFz$
		$SFz^{\{3\}} = SA1\ SFx\ SFy$
F_2	2	$SFx^{\{3\}} = SFx\ \left(SFz^2 + SFy^2\right)$
		$SFy^{\{3\}} = SFy\ \left(SFz^2 + SFx^2\right)$
		$SFz^{\{3\}} = SFz\ \left(SFy^2 + SFx^2\right)$
F_2	3	$SFx^{\{3\}} = SA1^2 SFx$
		$SFy^{\{3\}} = SA1^2 SFy$
		$SFz^{\{3\}} = SA1^2 SFz$
F_2	4	$SFx^{\{3\}} = SEa\ SFy\ SFz$
		$SFy^{\{3\}} = -1/2\ \left(SEa - \sqrt{3}SEb\right) SFx\ SFz$
		$SFz^{\{3\}} = -1/2\ \left(SEa + \sqrt{3}SEb\right) SFx\ SFy$
F_2	5	$SFx^{\{3\}} = SEb^2 SFx$
		$SFy^{\{3\}} = 1/12\ SFy\ \left(3\ SEa + \sqrt{3}SEb\right)^2$
		$SFz^{\{3\}} = 1/12\ SFz\ \left(3\ SEa - \sqrt{3}SEb\right)^2$
F_2	6	$SFx^{\{3\}} = SA1\ SEa\ SFx$
		$SFy^{\{3\}} = -1/2\ SA1\ \left(SEa - \sqrt{3}SEb\right) SFy$
		$SFz^{\{3\}} = -1/2\ SA1\ \left(SEa + \sqrt{3}SEb\right) SFz$
F_2	7	$SFx^{\{3\}} = SFx^3$
		$SFy^{\{3\}} = SFy^3$
		$SFz^{\{3\}} = SFz^3$
F_2	8	$SFx^{\{3\}} = SEa^2 SFx$
		$SFy^{\{3\}} = 1/4\ SFy\ \left(SEa - \sqrt{3}SEb\right)^2$
		$SFz^{\{3\}} = 1/4\ SFz\ \left(SEa + \sqrt{3}SEb\right)^2$

B.3 Expressions of order 4

The product representation

$$\Gamma^{(4)} = \left(\mathcal{D}^{(\mathrm{A}_1)} \oplus \mathcal{D}^{(\mathrm{E})} \oplus \mathcal{D}^{(\mathrm{F}_2)}\right)^{\otimes 4} \tag{B.5}$$

reduces as

$$\Gamma^{\mathrm{red}(4)} = 11\,\mathcal{D}^{(\mathrm{A}_1)} \oplus 3\,\mathcal{D}^{(\mathrm{A}_2)} \oplus 14\,\mathcal{D}^{(\mathrm{E})} \oplus 11\,\mathcal{D}^{(\mathrm{F}_1)} \oplus 17\,\mathcal{D}^{(\mathrm{F}_2)} \tag{B.6}$$

The symmetry adapted expressions are:

μ	k			
A_1	1	$SA1^{\{4\}}$	=	$SA1^4$
A_1	2	$SA1^{\{4\}}$	=	$\left(SEb^2 + SEa^2\right) SA1^2$
A_1	3	$SA1^{\{4\}}$	=	$\left(SFz^2 + SFy^2 + SFx^2\right) SA1^2$
A_1	4	$SA1^{\{4\}}$	=	$SA1\ SEa\ \left(-3\,SEb^2 + SEa^2\right)$
A_1	5	$SA1^{\{4\}}$	=	$SA1\ \left(-\sqrt{3}SEb\ SFz^2 + \sqrt{3}SEb\ SFy^2 + 2\,SEa\ SFx^2 - SEa\ SFy^2 - SEa\ SFz^2\right)$
A_1	6	$SA1^{\{4\}}$	=	$SA1\ SFx\ SFy\ SFz$
A_1	7	$SA1^{\{4\}}$	=	$\left(SEb^2 + SEa^2\right)^2$
A_1	8	$SA1^{\{4\}}$	=	$3\,SEb^2SFz^2 + 3\,SEb^2SFy^2 + SEa^2SFz^2 + SEa^2SFy^2$
				$+4\,SEa^2SFx^2 - 2\sqrt{3}SEa\ SEb\ SFy^2 + 2\sqrt{3}SEa\ SEb\ SFz^2$
A_1	9	$SA1^{\{4\}}$	=	$4\,SEb^2SFx^2 + SEb^2SFz^2 + SEb^2SFy^2 + 3\,SEa^2SFz^2$
				$+3\,SEa^2SFy^2 + 2\sqrt{3}SEa\ SEb\ SFy^2 - 2\sqrt{3}SEa\ SEb\ SFz^2$
A_1	10	$SA1^{\{4\}}$	=	$SFz^4 + SFy^4 + SFx^4$
A_1	11	$SA1^{\{4\}}$	=	$SFy^2SFz^2 + SFx^2SFz^2 + SFx^2SFy^2$
A_2	1	$SA2^{\{4\}}$	=	$SA1\ SEb\ \left(3\,SEa^2 - SEb^2\right)$
A_2	2	$SA2^{\{4\}}$	=	$SA1\ \left(-2\sqrt{3}SEb\ SFx^2 + \sqrt{3}SEb\ SFy^2 + \sqrt{3}SEb\ SFz^2\right)$
				$+SA1\ \left(3\,SEa\ SFy^2 - 3\,SEa\ SFz^2\right)$
A_2	3	$SA2^{\{4\}}$	=	$3\,SEb^2SFz^2 - 3\,SEb^2SFy^2 - 3\,SEa^2SFz^2 + 3\,SEa^2SFy^2$
				$+4\sqrt{3}SEa\ SEb\ SFx^2 - 2\sqrt{3}SEa\ SEb\ SFy^2 - 2\sqrt{3}SEa\ SEb\ SFz^2$

μ	k			
E	1	$SEa^{\{4\}}$	=	$25\,SEa^4 - 30\,SEa^2 SEb^2 - 15\,SEb^4$
		$SEb^{\{4\}}$	=	$-20\,\left(SEa^2 + 3\,SEb^2\right) SEa\,SEb$
E	2	$SEa^{\{4\}}$	=	$SA1\,\left(3\,SEa\,SFy^2 + 3\,SEa\,SFz^2 + \sqrt{3}SEb\,SFy^2 - \sqrt{3}SEb\,SFz^2\right)$
		$SEb^{\{4\}}$	=	$SA1\,\left(\sqrt{3}\,SEa\,SFy^2 - \sqrt{3}\,SEa\,SFz^2 + 4\,SEb\,SFx^2 + SEb\,SFz^2 + SEb\,SFy^2\right)$
E	3	$SEa^{\{4\}}$	=	$SA1\,\left(SEa\,SFy^2 + SEa\,SFz^2 - \sqrt{3}SEb\,SFy^2 + \sqrt{3}SEb\,SFz^2 + 4\,SEa\,SFx^2\right)$
		$SEb^{\{4\}}$	=	$-\sqrt{3}SA1\,\left(SEa\,SFy^2 - SEa\,SFz^2 - \sqrt{3}SEb\,SFz^2 - \sqrt{3}SEb\,SFy^2\right)$
E	4	$SEa^{\{4\}}$	=	$SA1\,\left(SEb^2 + SEa^2\right) SEa$
		$SEb^{\{4\}}$	=	$SA1\,\left(SEb^2 + SEa^2\right) SEb$
E	5	$SEa^{\{4\}}$	=	$SA1^2\left(2\,SFx^2 - SFy^2 - SFz^2\right)$
		$SEb^{\{4\}}$	=	$\sqrt{3}\,(SFy - SFz)\,(SFy + SFz)\,SA1^2$
E	6	$SEa^{\{4\}}$	=	$(SEa - SEb)\,(SEa + SEb)\,SA1^2$
		$SEb^{\{4\}}$	=	$-2\,SA1^2 SEa\,SEb$
E	7	$SEa^{\{4\}}$	=	$SA1^3 SEa$
		$SEb^{\{4\}}$	=	$SA1^3 SEb$
E	8	$SEa^{\{4\}}$	=	$-SFx^2 SFy^2 - SFx^2 SFz^2 + 2\,SFy^2 SFz^2$
		$SEb^{\{4\}}$	=	$-\sqrt{3}\,(SFy - SFz)\,(SFy + SFz)\,SFx^2$
E	9	$SEa^{\{4\}}$	=	$-SFy^4 + 2\,SFx^4 - SFz^4$
		$SEb^{\{4\}}$	=	$\sqrt{3}\,(SFy - SFz)\,(SFy + SFz)\left(SFy^2 + SFz^2\right)$
E	10	$SEa^{\{4\}}$	=	$SEa\,SFx\,SFy\,SFz$
		$SEb^{\{4\}}$	=	$SEb\,SFx\,SFy\,SFz$
E	11	$SEa^{\{4\}}$	=	$\left(-2\,\sqrt{3}SEb\,SFx^2 + \sqrt{3}SEb\,SFy^2 + \sqrt{3}SEb\,SFz^2 + 3\,SEa\,SFy^2 - 3\,SEa\,SFz^2\right) SEb$
		$SEb^{\{4\}}$	=	$\left(2\,\sqrt{3}SEb\,SFx^2 - \sqrt{3}SEb\,SFy^2 - \sqrt{3}SEb\,SFz^2 - 3\,SEa\,SFy^2 + 3\,SEa\,SFz^2\right) SEa$
E	12	$SEa^{\{4\}}$	=	$-5\,SEb^2 SFz^2 - 5\,SEb^2 SFy^2 - 8\,SEb^2 SFx^2$
				$+9\,SEa^2 SFz^2 + 9\,SEa^2 SFy^2 - 2\,\sqrt{3}SEa\,SEb\,SFy^2 + 2\,\sqrt{3}SEa\,SEb\,SFz^2$
		$SEb^{\{4\}}$	=	$-10\,SEa\,SEb\,SFz^2 - 10\,SEa\,SEb\,SFy^2 - 16\,SEa\,SEb\,SFx^2$
				$-\sqrt{3}\,SEa^2 SFy^2 + \sqrt{3}\,SEa^2 SFz^2 + 3\sqrt{3}\,SEb^2 SFz^2 - 3\sqrt{3}\,SEb^2 SFy^2$
E	13	$SEa^{\{4\}}$	=	$-2\,SEa^2 SFx^2 + 1/4\,SEa^2 SFy^2 + 1/2\,\sqrt{3}SEa\,SEb\,SFz^2$
				$-1/2\,\sqrt{3}SEa\,SEb\,SFy^2 + 1/4\,SEa^2 SFz^2 + 3/4\,SEb^2 SFy^2 + 3/4\,SEb^2 SFz^2$
		$SEb^{\{4\}}$	=	$-1/4\,\sqrt{3}\left(SEa\,SFy + SEa\,SFz - \sqrt{3}SEb\,SFy + \sqrt{3}SEb\,SFz\right)$
				$\times\left(SEa\,SFy - SEa\,SFz - \sqrt{3}SEb\,SFz - \sqrt{3}SEb\,SFy\right)$
E	14	$SEa^{\{4\}}$	=	$\left(3\,SEa^2 - SEb^2\right) SEb^2$
		$SEb^{\{4\}}$	=	$-\left(3\,SEa^2 - SEb^2\right) SEa\,SEb$

(cont.)

μ	k			
F_1	1	$SFu^{\{4\}}$	=	$2\sqrt{3}SEa^2\,SEb\,SFx$
		$SFv^{\{4\}}$	=	$-1/4\,SFy\,\left(3\,SEa+\sqrt{3}SEb\right)\left(SEa-\sqrt{3}SEb\right)^2$
		$SFw^{\{4\}}$	=	$1/4\,SFz\,\left(3\,SEa-\sqrt{3}SEb\right)\left(SEa+\sqrt{3}SEb\right)^2$
F_1	2	$SFu^{\{4\}}$	=	$SA1\,SFx\,\left(SFy-SFz\right)\left(SFy+SFz\right)$
		$SFv^{\{4\}}$	=	$-SA1\,\left(SFx-SFz\right)\left(SFx+SFz\right)SFy$
		$SFw^{\{4\}}$	=	$SA1\,\left(SFx-SFy\right)\left(SFx+SFy\right)SFz$
F_1	3	$SFu^{\{4\}}$	=	$2\sqrt{6}SA1\,SEa\,SEb\,SFx$
		$SFv^{\{4\}}$	=	$1/2\sqrt{2}SA1\,\left(SEa-\sqrt{3}SEb\right)\left(3\,SEa+\sqrt{3}SEb\right)SFy$
		$SFw^{\{4\}}$	=	$-1/2\sqrt{2}SA1\,\left(SEa+\sqrt{3}SEb\right)\left(3\,SEa-\sqrt{3}SEb\right)SFz$
F_1	4	$SFu^{\{4\}}$	=	$2\,SEb\,SFx^3$
		$SFv^{\{4\}}$	=	$-1/3\sqrt{3}\left(3\,SEa+\sqrt{3}SEb\right)SFy^3$
		$SFw^{\{4\}}$	=	$1/3\sqrt{3}\left(3\,SEa-\sqrt{3}SEb\right)SFz^3$
F_1	5	$SFu^{\{4\}}$	=	$2\,SEb^3SFx$
		$SFv^{\{4\}}$	=	$-1/36\sqrt{3}SFy\,\left(3\,SEa+\sqrt{3}SEb\right)^3$
		$SFw^{\{4\}}$	=	$1/36\sqrt{3}SFz\,\left(3\,SEa-\sqrt{3}SEb\right)^3$
F_1	6	$SFu^{\{4\}}$	=	$SEb\,SFx\,\left(SFy^2+SFz^2\right)$
		$SFv^{\{4\}}$	=	$-1/6\sqrt{3}\left(3\,SEa+\sqrt{3}SEb\right)\left(SFx^2+SFz^2\right)SFy$
		$SFw^{\{4\}}$	=	$1/6\sqrt{3}\left(3\,SEa-\sqrt{3}SEb\right)\left(SFx^2+SFy^2\right)SFz$
F_1	7	$SFu^{\{4\}}$	=	$2\sqrt{3}SA1^2SEb\,SFx$
		$SFv^{\{4\}}$	=	$-\left(3\,SEa+\sqrt{3}SEb\right)SFy\,SA1^2$
		$SFw^{\{4\}}$	=	$SFz\,\left(3\,SEa-\sqrt{3}SEb\right)SA1^2$
F_1	8	$SFu^{\{4\}}$	=	$SEa\,SFx\,\left(SFy-SFz\right)\left(SFy+SFz\right)$
		$SFv^{\{4\}}$	=	$1/2\,\left(SEa-\sqrt{3}SEb\right)\left(SFx-SFz\right)\left(SFx+SFz\right)SFy$
		$SFw^{\{4\}}$	=	$-1/2\,\left(SEa+\sqrt{3}SEb\right)\left(SFx-SFy\right)\left(SFx+SFy\right)SFz$
F_1	9	$SFu^{\{4\}}$	=	$2\sqrt{6}SEa\,SEb\,SFy\,SFz$
		$SFv^{\{4\}}$	=	$1/2\sqrt{2}\left(SEa-\sqrt{3}SEb\right)\left(3\,SEa+\sqrt{3}SEb\right)SFx\,SFz$
		$SFw^{\{4\}}$	=	$-1/2\sqrt{2}\left(SEa+\sqrt{3}SEb\right)\left(3\,SEa-\sqrt{3}SEb\right)SFx\,SFy$

(cont.)

μ	k	
F_1	10	$SFu^{\{4\}} = 2\sqrt{6}SA1\ SEb\ SFy\ SFz$
		$SFv^{\{4\}} = -\sqrt{2}SA1\ \left(3\ SEa + \sqrt{3}SEb\right)\ SFx\ SFz$
		$SFw^{\{4\}} = \sqrt{2}SA1\ \left(3\ SEa - \sqrt{3}SEb\right)\ SFx\ SFy$
F_1	11	$SFu^{\{4\}} = (SFy - SFz)\,(SFy + SFz)\ SFy\ SFz$
		$SFv^{\{4\}} = -(SFx - SFz)\,(SFx + SFz)\ SFx\ SFz$
		$SFw^{\{4\}} = (SFx - SFy)\,(SFx + SFy)\ SFx\ SFy$

(cont.)

μ	k			
F_2	1	$SFx^{\{4\}}$	=	$2\,SA1\,SEa\,SFy\,SFz$
		$SFy^{\{4\}}$	=	$-SA1\,\left(SEa-\sqrt{3}SEb\right)SFx\,SFz$
		$SFz^{\{4\}}$	=	$-SA1\,\left(SEa+\sqrt{3}SEb\right)SFx\,SFy$
F_2	2	$SFx^{\{4\}}$	=	$SEa\,SFx\,\left(SFy^2+SFz^2\right)$
		$SFy^{\{4\}}$	=	$-1/2\,\left(SEa-\sqrt{3}SEb\right)\left(SFx^2+SFz^2\right)SFy$
		$SFz^{\{4\}}$	=	$-1/2\,\left(SEa+\sqrt{3}SEb\right)\left(SFx^2+SFy^2\right)SFz$
F_2	3	$SFx^{\{4\}}$	=	$SA1\,SFx^3$
		$SFy^{\{4\}}$	=	$SA1\,SFy^3$
		$SFz^{\{4\}}$	=	$SA1\,SFz^3$
F_2	4	$SFx^{\{4\}}$	=	$SEb\,SFx\,(SFy-SFz)\,(SFy+SFz)$
		$SFy^{\{4\}}$	=	$1/6\,\sqrt{3}\left(3\,SEa+\sqrt{3}SEb\right)(SFx-SFz)\,(SFx+SFz)\,SFy$
		$SFz^{\{4\}}$	=	$1/6\,\sqrt{3}\left(3\,SEa-\sqrt{3}SEb\right)(SFx+SFy)\,(SFx-SFy)\,SFz$
F_2	5	$SFx^{\{4\}}$	=	$2\,SEa\,SFx^3$
		$SFy^{\{4\}}$	=	$-\left(SEa-\sqrt{3}SEb\right)SFy^3$
		$SFz^{\{4\}}$	=	$-\left(SEa+\sqrt{3}SEb\right)SFz^3$
F_2	6	$SFx^{\{4\}}$	=	$2\,SEb^2SFy\,SFz$
		$SFy^{\{4\}}$	=	$1/6\,SFx\,SFz\,\left(3\,SEa+\sqrt{3}SEb\right)^2$
		$SFz^{\{4\}}$	=	$1/6\,SFx\,SFy\,\left(3\,SEa-\sqrt{3}SEb\right)^2$
F_2	7	$SFx^{\{4\}}$	=	$SA1\,SFx\,\left(SFy^2+SFz^2\right)$
		$SFy^{\{4\}}$	=	$SA1\,\left(SFx^2+SFz^2\right)SFy$
		$SFz^{\{4\}}$	=	$SA1\,\left(SFx^2+SFy^2\right)SFz$
F_2	8	$SFx^{\{4\}}$	=	$2\,SEa^2SFy\,SFz$
		$SFy^{\{4\}}$	=	$1/2\,SFx\,SFz\,\left(SEa-\sqrt{3}SEb\right)^2$
		$SFz^{\{4\}}$	=	$1/2\,SFx\,SFy\,\left(SEa+\sqrt{3}SEb\right)^2$
F_2	9	$SFx^{\{4\}}$	=	$2\,SEa^3SFx$
		$SFy^{\{4\}}$	=	$-1/4\,SFy\,\left(SEa-\sqrt{3}SEb\right)^3$
		$SFz^{\{4\}}$	=	$-1/4\,SFz\,\left(SEa+\sqrt{3}SEb\right)^3$
F_2	10	$SFx^{\{4\}}$	=	$2\,SA1^2SEa\,SFx$
		$SFy^{\{4\}}$	=	$-\left(SEa-\sqrt{3}SEb\right)SFy\,SA1^2$
		$SFz^{\{4\}}$	=	$-\left(SEa+\sqrt{3}SEb\right)SFz\,SA1^2$

(cont.)

μ	k			
F_2	11	$SFx^{\{4\}}$	$=$	$2\,SA1\,SEa^2SFx$
		$SFy^{\{4\}}$	$=$	$1/2\,SA1\,SFy\,\left(SEa-\sqrt{3}SEb\right)^2$
		$SFz^{\{4\}}$	$=$	$1/2\,SA1\,SFz\,\left(SEa+\sqrt{3}SEb\right)^2$
F_2	12	$SFx^{\{4\}}$	$=$	$2\,SEa\,SEb^2SFx$
		$SFy^{\{4\}}$	$=$	$-1/12\,\left(SEa-\sqrt{3}SEb\right)SFy\,\left(3\,SEa+\sqrt{3}SEb\right)^2$
		$SFz^{\{4\}}$	$=$	$-1/12\,\left(SEa+\sqrt{3}SEb\right)SFz\,\left(3\,SEa-\sqrt{3}SEb\right)^2$
F_2	13	$SFx^{\{4\}}$	$=$	$\left(SFy^2+SFz^2\right)SFy\,SFz$
		$SFy^{\{4\}}$	$=$	$\left(SFx^2+SFz^2\right)SFx\,SFz$
		$SFz^{\{4\}}$	$=$	$\left(SFx^2+SFy^2\right)SFx\,SFy$
F_2	14	$SFx^{\{4\}}$	$=$	$SA1^2SFy\,SFz$
		$SFy^{\{4\}}$	$=$	$SA1^2SFx\,SFz$
		$SFz^{\{4\}}$	$=$	$SA1^2SFx\,SFy$
F_2	15	$SFx^{\{4\}}$	$=$	$SFx^2SFy\,SFz$
		$SFy^{\{4\}}$	$=$	$SFx\,SFy^2SFz$
		$SFz^{\{4\}}$	$=$	$SFx\,SFy\,SFz^2$
F_2	16	$SFx^{\{4\}}$	$=$	$SA1^3SFx$
		$SFy^{\{4\}}$	$=$	$SA1^3SFy$
		$SFz^{\{4\}}$	$=$	$SA1^3SFz$
F_2	17	$SFx^{\{4\}}$	$=$	$2\,SA1\,SEb^2SFx$
		$SFy^{\{4\}}$	$=$	$1/6\,SA1\,SFy\,\left(3\,SEa+\sqrt{3}SEb\right)^2$
		$SFz^{\{4\}}$	$=$	$1/6\,SA1\,SFz\,\left(3\,SEa-\sqrt{3}SEb\right)^2$

Table captions

Table 1. Reduction of product representations $\Gamma^{(m)} = \left(\mathcal{D}^{(\mathrm{A}_1)} \oplus \mathcal{D}^{(\mathrm{E})}\right)^{\otimes m}$ of the three-dimensional representation $\mathcal{D}^{(\mathrm{A}_1)} \oplus \mathcal{D}^{(\mathrm{E})}$ of $\mathbf{C}_{3\mathrm{v}}$ (Eq. (12) to Eq. (13)) (space of bending coordinates in ammonia). The reduced dimension at order m is the dimension of the A_1-subspace in $\mathbf{S}_{\mathrm{m}}$.

Table 2. Reduction of product representations $\Gamma^{(m)} = \left(\mathcal{D}^{(\mathrm{A}_1)} \oplus \mathcal{D}^{(\mathrm{E})} \oplus \mathcal{D}^{(\mathrm{F}_2)}\right)^{\otimes m}$ of the six-dimensional representation $\left(\mathcal{D}^{(\mathrm{A}_1)} \oplus \mathcal{D}^{(\mathrm{E})} \oplus \mathcal{D}^{(\mathrm{F}_2)}\right)$ of $\mathbf{T}_{\mathrm{d}}$ (Eq. (15) to Eq. (17)) (space of bending coordinates in methane). The reduced dimension at order m is the dimension of the A_1-subspace of $\mathbf{S}_{\mathrm{m}}$.

Table 1 **Marquardt and Sagui**

order (m)	dimension (n^m)	reduced dimension	reduction in $\mathbf{C}_{3v}$
2	9	6	$2\,\mathcal{D}^{(A_1)} \oplus 2\,\mathcal{D}^{(E)}$
3	27	10	$3\,\mathcal{D}^{(A_1)} \oplus \mathcal{D}^{(A_2)} \oplus 3\,\mathcal{D}^{(E)}$
4	81	15	$4\,\mathcal{D}^{(A_1)} \oplus 1\,\mathcal{D}^{(A_2)} \oplus 5\,\mathcal{D}^{(E)}$

Table 2 **Marquardt and Sagui**

order (m)	dimension (n^m)	reduced dimension	reduction in $\mathbf{T}_\mathrm{d}$
2	36	21	$3\,\mathcal{D}^{(\mathrm{A}_1)} \oplus 3\,\mathcal{D}^{(\mathrm{E})} \oplus 1\,\mathcal{D}^{(\mathrm{F}_1)} \oplus 3\,\mathcal{D}^{(\mathrm{F}_2)}$
3	216	56	$6\,\mathcal{D}^{(\mathrm{A}_1)} \oplus 2\,\mathcal{D}^{(\mathrm{A}_2)} \oplus 6\,\mathcal{D}^{(\mathrm{E})} \oplus 4\,\mathcal{D}^{(\mathrm{F}_1)} \oplus 8\,\mathcal{D}^{(\mathrm{F}_2)}$
4	1296	126	$11\,\mathcal{D}^{(\mathrm{A}_1)} \oplus 3\,\mathcal{D}^{(\mathrm{A}_2)} \oplus 14\,\mathcal{D}^{(\mathrm{E})} \oplus 11\,\mathcal{D}^{(\mathrm{F}_1)} \oplus 17\,\mathcal{D}^{(\mathrm{F}_2)}$

REDTEN: A symbolic algebra program for the automatic reduction of powers of representations of finite groups

Roberto Marquardt* and Kenneth Sagui

Laboratoire de Chimie Theorique, Université de Marne-la-Vallée
5 Bd Descartes (Champs-sur-Marne), F-77454 Marne-la-Vallée CEDEX 2, France

USER GUIDE and MAPLE code **REDTEN**

prepared as Supplementary Publication
to be deposited with the British Library Document Supply Centre

* present address (21/01/2007) and email: roberto.marquardt@chimie.u-strasbg.fr
correspondence and proofs to Prof. Roberto Marquardt
Laboratoire de Chimie Quantique - Institut de Chimie - LC3 UMR 7177 CNRS/ULP
Université Louis Pasteur - 4, rue Blaise Pascal - 67000 STRASBOURG - France
Fax 0(033)3 90 24 15 89

Contents

1 Introduction

This code implements an algorithm to perform the reduction of representations in general, and the symmetry adaption for power representations of some finite groups in particular, according to the procedure described in the article "A complete list of symmetry adapted expressions to the fourth power for compact bending potentials in molecules with $\mathbf{C}_{3v}$ and $\mathbf{T}_{d}$ symmetry from a general symbolic algebra program" [1] that is related with it.

The implementation of the code is based on the `MAPLE` program package for symbolic algebra. The present version (060527) runs under `MAPLE 10` release [2].

This document contains information on the distribution of the code and instructions for installation and use. A hard copy of the code related to the distribution list described below is given in the appendices.

2 Name

The code has the name **REDTEN**.

3 Distribution list (version 060527)

A distribution package is available from the corresponding author at the address given in the cover page. This package contains the following files:

`main.cmd`	source code
`turm.cmd`	source code
`redtenlib.cmd`	source code
`run-3-3-C3v.cmd`	sample input
`run-3-3-C3v.out`	sample output
`reduce.com`	unix script

All files are open source. The code may be changed or appended by interested users when they make a reference to the original article [1].

4 Installation

The package is delivered as a zip file. Unpacking yields a directory, named `package`, and a pdf file containing the present document. To run the code, enter the directory `package`, or any other working directory, into which the contents of `package` have been copied.

Give the unix script `reduce.com` "execute permission", then run it from the working directory (see section 6 below).

5 Input

Input files must be named according to the following nomenclature:

`run-`M`-`$NDIM1$`-`$GROUP$`.cmd`

where M is a positive integer that gives the order of the tensor space, $NDIM1$ is a positive integer that gives the dimension of the first order linear space V and $GROUP$ is a code that defines the finite symmetry group to be considered.

Currently, accepted codes are: $GROUP \in$ {Cs, C2v, C3v, Td} (with size sensitive letters).

The input file must contain the following cards:

M:= the order of the tensor space $V^{\otimes m}$

GROUP:= the code for the finite symmetry group

NDIM1:= the dimension of the first order vector space V

MUE1:= $[\mu_1, \ldots, \mu_N]$, `MAPLE`-table of N irreducible representations of group $GROUP$ which are used to build up the V space; μ_j is the number of the j-th irreducible representation in the corresponding character table (see section 9 below for a list of character tables used in the code); of course, $\sum_{n=1}^{N} d_{\mu_n} \cdot \gamma_{\mu_n} = NDIM1$ must

hold, where d_{μ_n} is the dimension and γ_{μ_n} is a possible multiplicity of the irreducible representation species μ_n in V (see next card)

MMU1:= $[\gamma_1, \ldots, \gamma_N]$, MAPLE-table of multiplicities of each irreducible representation species used to build up the first order vector space V

V1I[1][1]:= MAPLE-table of symbols to represent the components of the first irreducible vector of species 1 (the table has d_1 entries)

⋮

V1I[1][γ_1]:= MAPLE-table of symbols to represent the components of the irreducible vector number γ_1 of species 1 (the table has d_1 entries)

⋮

⋮

V1I[μ][1]:= MAPLE-table of symbols to represent the components of the first irreducible vector of species μ (the table has d_μ entries)

⋮

V1I[μ][γ_μ]:= MAPLE-table of symbols to represent the components of the irreducible vector number γ_μ of species μ (the table has d_μ entries)

⋮

⋮

read "turm.cmd": Call of routine `turm.cmd` which is designed to get the representation of the symmetric Group $\mathbf{S}_\mathrm{m}$ in tensor space $V^{\otimes m}$ and the reduction thereof in $\mathbf{S}_\mathrm{m}$; this routine essentially calculates the rectangular transformation matrix $\boldsymbol{T}^{(NDIM1^M;\gamma)}$ from equation (13) of the article, where γ is the multiplicity of the $\mathrm{A}_1(\mathbf{S}_\mathrm{M})$-species; when this routine is run for the first time, an intermediate output file is generated that is read in the subsequent calls of the program, for the same value of M; this card can therefore be commented in subsequent calls by inserting the symbol ♯ in the first column of the card

read "main.cmd": Call of routine `main.cmd` which concentrates all the stages of the algorithm, as indicated by its name.

To illustrate this, we give the example of the input file designed to run a calculation of the reduction of direct product representations of $\mathbf{C}_{3v}$ of order 3 for a three-dimensional first-order vector, as showed in the article that goes with this program [1].

First, the input file in question must be named `run-3-3-C3v.cmd`.

Secondly, it must be presented in the form given in Figure 1.

```
♯ Input values:
M:=3: ♯ order of tensor space
GROUP:=C3v: ♯ symmetry group
NDIM1:=3: ♯ dimension of first order linear space
MUE1:=[1,3]: ♯ irreps of GROUP used to build up V space
MMU1:=[1,1]: ♯ multiplicity of irreps
V1I[1][1]:=[SA1]: ♯ choice of component symbols
V1I[3][1]:=[SEa,SEb]: ♯ choice of component symbols
read "turm.cmd":
read "main.cmd":
```

Figure 1: Example of an input file : `run-3-3-C3v.cmd`; the symbol ♯ is followed of a comment in the maple language.

6 Running the calculation

The program can be launched by using the unix script `reduce.com` as follows:

`reduce.com` M $NDIM1$ $GROUP$

The three arguments have the same meaning as for the naming of the files. The script opens `MAPLE`, runs the code, writes output files and closes `MAPLE` automatically. The

program may be run as a background job. For the example given above, type for instance `reduce.com 3 3 C3v&`.

7 Output files

Several output files are generated:

`turmdata-`M`-`$NDIM1$`.res`

Data file in `MAPLE` language to be used in routine `main.cmd`, containing the rectangular transformation matrix $\boldsymbol{T}^{(NDIM1^M;\gamma)}$.

`turm-`M`-`$NDIM1$`.out`

Logfile for routine `turm.cmd`.

`REPMDATA-`M`-`$NDIM1$`-`$GROUP$`.res`

Data file in `MAPLE` language containing the representation matrices $\boldsymbol{\Gamma}^{(NDIM1^M)}$ obtained from equation (11) of the article.

`GAMMAmr-`M`-`$NDIM1$`-`$GROUP$`.res`

Data file in `MAPLE` language containing the reduced representations $\boldsymbol{\Gamma}^{(\gamma)}$ obtained from equation (14).

`run-`M`-`$NDIM1$`-`$GROUP$`.res`

File in `MAPLE` language containing the following data, in the given order:

NIRRP, the number of irreducible representions;

REDF, `MAPLE`-table with NIRRP entries containing the reduction factors (multiplicities) γ_μ into the irreductible representations μ;

DIRRP, `MAPLE`-table with NIRRP entries containing the dimensions d_μ of the irreducible representations;

GLABE, `MAPLE`-table with NIRRP entries containing labels for the irreducible representations;

Vms, `MAPLE`-table containing entries Vms[μ]; only irreducible representations μ for which $\gamma_\mu > 0$ are given; each entry is a vector of length $\gamma_\mu \times d_\mu$ and contains symmetry adapted expressions;

`run-`M`-`$NDIM1$`-`$GROUP$`.out`

Logfile of routine `main.cmd` containing self-explanatory output.

8 Brief description of different subroutines used in the algorithm

In `main.cmd` the program library `redtenlib.cmd` is loaded which gathers all the subroutines used in the algorithm with the exception of `turm.cmd`. In what follows, the function of each routine will be briefly described.

`prtime` displays the duration of each stage of the calculation

`irrep` definition of irreducible representations

`defk` definition of tensor space key

`repm` calculation of group representation matrices in power M tensor space

`simtra` calculation of similarity transformation

`Sm` calculation of permutation matrices (regular representation of the symmetry group $\mathbf{S}_{\mathrm{M}}$)

`SmInVm` calculation of representation matrices $\boldsymbol{\Sigma}^{(NDIM1^M)}$

`redA1` calculation of symmetry reduction by projection on A_1 irreducible representation vector subspace using Wigner's projection operator [3]

`redfac` calculation of symmetry reduction factors

`reduce` calculation of symmetry reduction using generalized projection operators as given by Hamermesh [4, 5]

`orthonorm` calculates a set of ortho-normalized vectors following the Gram-Schmidt procedure [6]

`vecnorm` calculation of a vector norm.

9 Character tables

We used the following conventions for irreducible representations:

$\mathbf{C}_\mathrm{s}$	μ	label	E	σ
	1	A	1	1
	2	B	1	-1

$\mathbf{C}_\mathrm{2v}$	μ	label	E	C_2	σ_v	σ_h
	1	A_1	1	1	1	1
	2	A_2	1	1	-1	-1
	3	B_1	1	-1	-1	1
	4	B_2	1	-1	1	-1

$\mathbf{C}_\mathrm{3v}$	μ	label	E	$2C_3$	$3\sigma_\mathrm{v}$
	1	A_1	1	1	1
	2	A_2	1	1	-1
	3	E	2	-1	0

$\mathbf{T}_\mathrm{d}$	μ	label	E	$6C_2'$	$8C_3$	$6C_4$	$3C_2$
	1	A_1	1	1	1	1	1
	2	A_2	1	-1	1	-1	1
	3	E	2	0	-1	0	2
	4	F_1	3	-1	0	1	-1
	5	F_2	3	1	0	-1	-1

References

[1] Roberto Marquardt and Kenneth Sagui. A complete list of symmetry adapted expressions to the fourth power for compact bending potentials in molecules with $\mathbf{C}_{3v}$ and $\mathbf{T}_{d}$ symmetry from a general symbolic algebra program, 2007. Submitted to Molecular Physics.

[2] *Maple V Release 10.* Copyright (c) 2004-2005 by Maplesoft, a division of Waterloo Maple Software Inc.

[3] E. Wigner. *Group Theory.* Academic Press, New York, 1959.

[4] M. Hamermesh. *Group Theory and Its Application to Physical Problems.* Addison-Wesley, Reading, Massachusetts, 1962.

[5] J. P. Elliot and P. G. Dawber. *Symmetry in Physics.* Oxford University Press, New York, 1979.

[6] H.-J. Kowalsky. *Lineare Algebra.* de Gruyter, Berlin, New York, 1979.

A main.cmd

```
# This file is the program REDTEN. It runs an automatic reduction of a tensor product
# representation of order m.
#
# Authors: R. Marquardt and K. Sagui
#          Université de Marne-la-Vallée
#          5 Bd Descartes, 77454 Marne-la-Vallée CEDEX 2
#          email for contact: roberto.marquardt@univ-mlv.fr
#
# Version REDTEN-060527
#
# Needed programs: MAPLE 9.5 or MAPLE 10
#
# Input parameters      M (order of tensor space),
#                   GROUP (symmetry group label),
#                   NDIM1 (dimension of first order linear space for check),
#                    MUE1 (list of irreps of GROUP used to build up V1),
#                    MMU1 (list of multiplicities of irreps),
# V1I[MUE1[mu]][MMU1[mu]] (list of component symbols for irrep MUE1[mu])
# must be set in calling program.
```

```
#
  restart:
  time0:=time():
# 1) Load subroutines and libraries:
  read ‘redtenlib.cmd‘:
  with(linalg):
  with(LinearAlgebra):
  outputfilename:=cat("run-",M,"-",NDIM1,"-",GROUP,".out"):
  outputfile:=fopen(outputfilename,WRITE):
#  writeto(outputfile):
#  interface(quiet=true):
#
  prtime(outputfile,1,"loading    ",time0):
# 2) Load information on irreductible representations of the symmetry groups.
  GDATA:=irrep(GROUP):
# Procedure irrep returns a list of 6 items:
  NORDR:=GDATA[1]: # the group order.
  NIRRP:=GDATA[2]: # the number of irreductible representations of the group.
  GCHAR:=GDATA[3]: # a NIRRPxNORDR matrix containing the character table.
  GLABE:=GDATA[4]: # a 1..NIRRP array  containing usual labels of the irr. rep.
  GIRRP:=GDATA[5]: # a nested list of matrices of the form GIRRP[i][n][k,j],
```

```
                    # where i is the number of the irreductible representation,
                    #       n is the number of the operation,
                    #       k and j are indices from 1..NDIM(i),
                    #       where NDIM(i) = GCHAR[i,1].
  NBG:=GDATA[6]:    # number of any operation with nondiagonal
                    # degenerate irreductible representation; set to 1 if none
  prtime(outputfile,2,"read irreps",time0):
#
# 3) Build up 1st order vector space and group representations:
  for mu from 1 to NIRRP do
      DIRRP[mu]:=GCHAR[mu,1]:            # dimension of irrep mu
  od:
  NIRRPE1:=nops(MUE1):  # number of irreps of G used to build up V1 space
  if (nops(MMU1) <> NIRRPE1) then
     print("Mismatch between MMU1 and MUE1 on input. Program stopped."):
     stop
  end if:
  mdim:=0:
  for mu from 1 to NIRRPE1 do
      mmu:=MMU1[mu]:
      for im from 1 to mmu do
```

```
            for id from 1 to DIRRP[MUE1[mu]] do
                mdim:=mdim+1:
            od:
            di:=nops(V1I[MUE1[mu]][im]):
            if (di <> DIRRP[MUE1[mu]]) then
               error "Mismatch with dimensions of V1I on input. Program interrupted."
            end if:
            V1I[MUE1[mu]][im]:=convert(V1I[MUE1[mu]][im],vector):
        od:
    od:
    if (mdim <> NDIM1) then
       error "Mismatch with NDIM1 on input. Program interrupted."
    end if:
    V1:=array(1..NDIM1,[seq(seq(seq(V1I[MUE1[mu]][im][id],id=1..DIRRP[MUE1[mu]]),im=1..MMU1[mu]),mu=1..NIRRPE1)]):
    for n from 1 to NORDR do
        GAMMA1[n]:=matrix(NDIM1,NDIM1):
        GAMMA1[n]:=diag(seq(seq(eval(GIRRP[MUE1[mu]][n]),im=1..MMU1[mu]),mu=1..NIRRPE1)):
    od:
    prtime(outputfile,3,"setup GAMMA1",time0):
#
# 4) Define tensor space and key:
```

```
  KDATA:=defk(M,NDIM1):
# Procedure defk returns 2 items:
  NDIMm:=KDATA[1]: # tensor space dimension
  KEY:=KDATA[2]:   # matrix MxNDIMm defining the tensor space.
  Vm:=array(1..NDIMm):
  for nd from 1 to NDIMm do    # definition of tensor space
      Vm[nd]:=1:
      for mm from 1 to M do
          Vm[nd]:=Vm[nd]*V1[KEY[mm,nd]]:
      od:
  od:
  prtime(outputfile,4,"define tensor space key",time0):
# 5) Build up order m vector space and group representations:
  GAMMAm:=array(1..NORDR):
  for n from 1 to NORDR do
      GAMMAm[n]:=repm(M,NDIMm,KEY,GAMMA1[n]):
      nout:=500+n:
      prtime(outputfile,nout,"setup tensor space, operation ",time0):
# Procedure repm returns 1 item: the representation matrix NDIMmxNDIMm
  od:
  datfile:=cat("REPMDATA-",M,"-",NDIM1,"-",GROUP,".res"):
```

```
  save GAMMAm,datfile:
# read datfile:
  prtime(outputfile,5,"setup tensor space",time0):
#
# 6) Reduce tensor space Vm into A1 space of Sm by similarity transformation.
  turmdatafile:=cat("turmdata-",M,"-",NDIM1,".res"):
  read turmdatafile:
# turmdatafile returns REDSM, a list of 5 items:
  MM:=REDSM[1]:    # the order, must match M.
  MDIM1:=REDSM[2]: # dimension of 1st order space, must match NDIM1.
  MIRRP:=REDSM[3]: # number of irreductible representations (irreps) of Sm.
  GAMMS:=REDSM[4]: # list 1..MIRRP of reduction factors
                   # of Sm representation in Vm.
  TS:=REDSM[5]:    # NDIMmXGAMMS[1] transformation matrix into A1 space.
  if (MM <> M) then
     error "Order mismatch with data on file",turmdatafile
  end if:
  if (MDIM1 <> NDIM1) then
     error "NDIM1 mismatch with data on file",turmdatafile
  end if:
  NDIMmr:=GAMMS[1]: # Reduced dimension of Vm; only A1 species of Sm is kept
```

```
  TSr:=matrix(NDIMm,NDIMmr):
  TSr:=submatrix(TS,1..NDIMm,1..NDIMmr):   # get reduction trafo
  TSrT:=transpose(TSr):
  GAMMAmr:=array(1..NORDR):
  for n from 1 to NORDR do
      GAMMAmr[n]:=simtra(TSr,TSrT,GAMMAm[n]): # Procedure simtra returns TT*G*T
      nout:=600+n:
      prtime(outputfile,nout,"reduction in Sm, operation ",time0):
  od:
#  GAMMAmr:=eval(GAMMAmr,4):
  datfile:=cat("GAMMAmr-",M,"-",NDIM1,"-",GROUP,".res"):
  save GAMMAmr,datfile:
#  read datfile:
  Vmr:=array(1..NDIMmr):
  Vmr:=multiply(TSrT,Vm): # Vector components in A1 space of Sm.
  TT:=matrix(NDIMmr,NDIMm):
  TT:=evalm(TSrT):        # backup final transformation of Vm
  prtime(outputfile,6,"reduction in Sm",time0):
# 7) Get reduction factors of GAMMAmr in GROUP
  REDF:=redfac(NORDR,NIRRP,GCHAR,GLABE,NDIMmr,GAMMAmr):
# Procedure redfac returns 1 item: a list 1..NIRRP of reduction factors
```

```
  prtime(outputfile,7,"reduction factors",time0):
# 8) Reduce GAMMAmr irrep by irrep
  for mu from 1 to NIRRP do
      gammu:=REDF[mu]:
      dmu:=DIRRP[mu]:
      nmu:=gammu*dmu:
      if (gammu > 0) then
         Ti[mu]:=matrix(NDIMmr,nmu):
 Ti[mu]:=reduce(mu,gammu,NORDR,GIRRP,GCHAR,NDIMmr,GAMMAmr,time0):
# Procedure reduce returns a NDIMmrx(gammu*nmu) matrix of orthogonal vectors
#        print(Ti[mu]):
 TiT[mu]:=transpose(Ti[mu]):
      end if:
  od:
  prtime(outputfile,8,"reduce GAMMAmr",time0):
# 9) Tensor transformation
  for mu from 1 to NIRRP do
      gammu:=REDF[mu]:
      dmu:=DIRRP[mu]:
      if (gammu > 0) then
         Vms[mu]:=evalm(TiT[mu]&*Vmr):
```

```
Vms[mu]:=map(factor,Vms[mu]):
Vms[mu]:=map(combine,Vms[mu],radical,symbolic):
        for nop from 1 to NORDR do
            Gred[mu][nop]:=simtra(Ti[mu],TiT[mu],GAMMAmr[nop]):
        od:
    end if:
 od:
 prtime(outputfile,9,"tensor transformations",time0):
# 10) Testing reduced representations
#  appendto(outputfilename):
 printf("    \n\n"):
 printf(" Testing reduction; reduced representations:\n\n"):
 for mu from 1 to NIRRP do
     sdiff:=0:
     if (REDF[mu] > 0) then
         gammu:=REDF[mu]:
         for nop from 1 to NORDR do
     rept:=diag(seq(eval(GIRRP[mu][nop]),im=1..gammu)):
             repr:=evalm(eval(Gred[mu][nop])):
     diffr:=evalm(rept-repr):
     diffn:=linalg[norm](diffr):
```

```
        diffn:=evalf(diffn):
        sdiff:=sdiff+diffn**2:
                printf(" type %-4s , operation %-3d , difference %-5.1f \n",GLABE[mu],nop,diffn);
                fprintf(outputfile," type %-4s , operation %-3d , difference %-5.1f \n",GLABE[mu],nop,diffn);
#               print(repr);
            od:
        end if:
    od:
    if (sdiff = 0.0) then
       printf(" Testing reduction successful \n");
       fprintf(outputfile, " Testing reduction successful \n");
    else
       printf(" ATTENTION: Testing reduction FAILED \n");
       fprintf(outputfile, " ATTENTION: Testing reduction FAILED \n");
    end if:
#   writeto(terminal):
# 11) Printing final results
    fprintf(outputfile,"%-60s \n\n","------------------------------------------------------------------");
    fprintf(outputfile,"Final results: \n\n");
    fprintf(outputfile,"1) %-38s %-.0f \n"  ,"Original tensor space dimension:",NDIMm);
    fprintf(outputfile,"   %-38s %-.0f \n\n","Reduced  tensor space dimension:",NDIMmr);
```

```
fprintf(outputfile,"2) %-38s %-10s %-10s \n","Reduction factors: ","LABEL[mu]","GAMMA[mu]");
fprintf(outputfile,"   %-38s %-10s %-10s \n","                    ","---------","---------");
for mu from 1 to NIRRP do
  fprintf(outputfile,"   %-38s %-10a %-10d \n","                     ",GLABE[mu],REDF[mu]);
od:
fprintf(outputfile,"3) Symmetrized expressions: \n\n");
fprintf(outputfile,"   (k=1..GAMMA[mu], n=1..DIM[mu]) \n");
fprintf(outputfile,"   %-10s %-3s  %-3s %s \n","LABEL[mu]","k","n","S[mu][k][n]");
fprintf(outputfile,"   %-10s %-3s  %-3s %s \n","---------","---","---","-------------------------");
savfile:=cat("run-",M,"-",NDIM1,"-",GROUP,".res"):
REDF:=eval(REDF):
DIRRP:=eval(DIRRP):
GLABE:=eval(GLABE):
save NIRRP,REDF,DIRRP,GLABE,Vms, savfile:
for mu from 1 to NIRRP do
    gammu:=REDF[mu]:
    dmu:=DIRRP[mu]:
    if (gammu > 0) then
for k from 1 to gammu do
   for n from 1 to dmu do
               fprintf(outputfile,"   %-10s %-3d  %-3d %-a \n\n",GLABE[mu],k,n,Vms[mu][(k-1)*dmu+n]);
```

```
            od:
        od:
    end if:
od:
prtime(outputfile,11,"**************",time0):
gc():
fclose(outputfilename):
quit
```

B turm.cmd

```
# This file is the routine turm.cmd of program REDTEN. This routine gets
# the representation of the symmetric Group Sm in tensor space Vm=V**m,
# and the reduction thereof in Sm.
#
# Authors: R. Marquardt and K. Sagui
#          Université de Marne-la-Vallée
#          5 Bd Descartes, 77454 Marne-la-Vallée CEDEX 2
#          email for contact: roberto.marquardt@univ-mlv.fr
#
# Version REDTEN-060527
#
# Needed programs: MAPLE 9.5 or MAPLE 10
#
# Input parameters M and NDIM1 must be set in calling program.
#
# 1) Load subroutines and libraries:
  time0:=time():
  read ‘redtenlib.cmd‘:
  with(linalg):
```

```
  with(LinearAlgebra):
  outputfilename:=cat("turm-",M,"-",NDIM1,".out"):
  outputfile:=fopen(outputfilename,WRITE):
#
# 2) Define tensor space and key:
  KDATA:=defk(M,NDIM1):
# Procedure defk returns 2 items:
  NDIMm:=KDATA[1]: # tensor space dimension
  KEY:=eval(KDATA[2]):   # matrix MxNDIMm defining the tensor space.
  prtime(outputfile,2,"TURM: key definition",time0):
# 3) Define permutation matrices of order M, the "turmdarstellung":
  SDATA:=Sm(M):
# Procedure Sm returns a list of 5 items:
  MORDR:=SDATA[1]: # the group order.
  MIRRP:=SDATA[2]: # the number of irreductible representations of the group.
  SCHAR:=SDATA[3]: # a MIRRPxMORDR matrix containing the character table.
  SLABE:=array(1..MIRRP):
  SLABE:=evalm(SDATA[4]): # a 1..NIRRP array  containing usual labels of the irr. rep.
  PMAT:=SDATA[5]:  # a list of matrices of the form PMAT[n][k,j],
                   # where n is the number of the operation,
                   #       k and j are indices from 1..M, the "turmdarstellung".
```

```
  prtime(outputfile,3,"TURM: setup turmdarstellung",time0):
# 4) Get representation of Sm in Vm
  TREP:=evalm(SmInVm(M,NDIMm,MORDR,KEY,PMAT)):
# Procedure SmInVm returns 1 item: a list of M! matrices NDIMmxNDIMm
#                  defining the permutation matrices of M elements in
#                  tensor space V**M.
#
  prtime(outputfile,4,"TURM: setup permutation matrices",time0):
# 5) Reduce permutation representation in Sm, get only A1
  REDTS:=redA1(MORDR,MIRRP,SCHAR,NDIMm,TREP,time0):
# Procedure redA1 returns a list of 2 items:
  REDF:=REDTS[1]: # the list 1..MIRRP of reduction factors
  TS:=matrix(NDIMm,NDIMm):
  TS:=REDTS[2]:    # the NDIMmxREDF[1] transformation matrix into the
                   # irreductible space A1
  prtime(outputfile,5,"TURM: reduction of perm. matrices",time0):
# 6) Saving data:
  REDSM:=[M,NDIM1,MIRRP,eval(REDF),evalm(TS)]:
  datafile:=cat("turmdata-",M,"-",NDIM1,".res"):
  save REDSM,datafile:
  prtime(outputfile,6,"TURM: saving data",time0):
```

```
fclose(outputfilename):
```

C redtenlib.cmd

```
# This file is a library of routines to be used within program REDTEN.
#
# Authors: R. Marquardt and K. Sagui
#          Université de Marne-la-Vallée
#          5 Bd Descartes, 77454 Marne-la-Vallée CEDEX 2
#          email for contact: roberto.marquardt@univ-mlv.fr
#
# Version REDTEN-060527
#
# Needed programs: MAPLE 9.5 or MAPLE 10
#
prtime:=proc(file,i,txt::string,time0)
        local n,rtime,H,M,S:
rtime:=time()-time0:
H:=trunc(rtime/3600.):
M:=trunc((rtime-H*3600.)/60.):
S:=rtime-(H*3600.+M*60.):
        fprintf(file,"Step %-5.0f ( %-34s ) done; %3.0f h %2.0f m %2.3f s \n",i,txt,H,M,S);
        printf("Step %-5.0f ( %-34s ) done; %3.0f h %2.0f m %2.3f s \n",i,txt,H,M,S);
```

```
end proc:
irrep:=proc(G)
       local i,n,ng,nirrep,chartab,label,Girr,nnb,RESULT,a,b,c,d,e,f,g,h:
       "Definition of irreductible representations.":
       "Currently available: CS, C2v, C3v, Td":
       a:=1/2:
       b:=sqrt(3)/2:
       c:=1/3:
       d:=sqrt(2)/3:
       e:=sqrt(6)/3:
       f:=5/6:
       g:=sqrt(3)/6:
       h:=sqrt(8)/3:
       if (G='CS') then
          nnb:=1:   # number of operation with non-diagonal representation
          ng:=2:    # group order
  nirrep:=2:# number of irreps (number of classes)
  chartab:=matrix(nirrep,ng,[[ 1, 1],
                             [ 1,-1]]):
          label:=array(1..nirrep):
#         #1st irrep
```

```
            label[1]:=A:
            Girr[1][1]:=matrix(1,1,[[ 1]]):
            Girr[1][2]:=matrix(1,1,[[ 1]]):
#           #2nd irrep
            label[2]:=B:
            Girr[2][1]:=matrix(1,1,[[ 1]]):
            Girr[2][2]:=matrix(1,1,[[-1]]):
        elif ((G='C2v') or (G='S2')) then
           nnb:=1:   # number of operation with non-diagonal representation
           ng:=4:    # group order
  nirrep:=4:# number of irreps (number of classes)
  chartab:=matrix(nirrep,ng,[[ 1, 1, 1, 1],
                             [ 1, 1,-1,-1],
                             [ 1,-1,-1, 1],
                             [ 1,-1, 1,-1]]):
           label:=array(1..nirrep):
#          #1st irrep
            label[1]:=A1:
            Girr[1][1]:=matrix(1,1,[[ 1]]):
            Girr[1][2]:=matrix(1,1,[[ 1]]):
            Girr[1][3]:=matrix(1,1,[[ 1]]):
```

```
            Girr[1][4]:=matrix(1,1,[[ 1]]):
#           #2nd irrep
            label[2]:=A2:
            Girr[2][1]:=matrix(1,1,[[ 1]]):
            Girr[2][2]:=matrix(1,1,[[ 1]]):
            Girr[2][3]:=matrix(1,1,[[-1]]):
            Girr[2][4]:=matrix(1,1,[[-1]]):
#           #3rd irrep
            label[3]:=B1:
            Girr[3][1]:=matrix(1,1,[[ 1]]):
            Girr[3][2]:=matrix(1,1,[[-1]]):
            Girr[3][3]:=matrix(1,1,[[-1]]):
            Girr[3][4]:=matrix(1,1,[[ 1]]):
#           #4th irrep
            label[4]:=B2:
            Girr[4][1]:=matrix(1,1,[[ 1]]):
            Girr[4][2]:=matrix(1,1,[[-1]]):
            Girr[4][3]:=matrix(1,1,[[ 1]]):
            Girr[4][4]:=matrix(1,1,[[-1]]):
        elif ((G='C3v') or (G='S3')) then
           nnb:=2:   # number of operation with non-diagonal representation
```

```
          ng:=6:    # group order
  nirrep:=3:# number of irreps (number of classes)
  chartab:=matrix(nirrep,ng,[[ 1, 1, 1, 1, 1, 1],
                             [ 1, 1, 1,-1,-1,-1],
                       [ 2,-1,-1, 0, 0, 0]]):
          label:=array(1..nirrep):
#         #1st irrep
           label[1]:=A1:
           Girr[1][1]:=matrix(1,1,[[ 1]]):
           Girr[1][2]:=matrix(1,1,[[ 1]]):
           Girr[1][3]:=matrix(1,1,[[ 1]]):
           Girr[1][4]:=matrix(1,1,[[ 1]]):
           Girr[1][5]:=matrix(1,1,[[ 1]]):
           Girr[1][6]:=matrix(1,1,[[ 1]]):
#         #2nd irrep
           label[2]:=A2:
           Girr[2][1]:=matrix(1,1,[[ 1]]):
           Girr[2][2]:=matrix(1,1,[[ 1]]):
           Girr[2][3]:=matrix(1,1,[[ 1]]):
           Girr[2][4]:=matrix(1,1,[[-1]]):
           Girr[2][5]:=matrix(1,1,[[-1]]):
```

```
            Girr[2][6]:=matrix(1,1,[[-1]]):
#          #3rd irrep
            label[3]:=E:
            Girr[3][1]:=matrix(2,2,[[         1,          0],
                                    [         0,          1]]):
            Girr[3][2]:=matrix(2,2,[[      -1/2, -sqrt(3)/2],
                                    [ sqrt(3)/2,       -1/2]]):
            Girr[3][3]:=matrix(2,2,[[      -1/2,  sqrt(3)/2],
                                    [-sqrt(3)/2,       -1/2]]):
            Girr[3][4]:=matrix(2,2,[[         1,          0],
                                    [         0,         -1]]):
            Girr[3][5]:=matrix(2,2,[[      -1/2, -sqrt(3)/2],
                                    [-sqrt(3)/2,        1/2]]):
            Girr[3][6]:=matrix(2,2,[[      -1/2,  sqrt(3)/2],
                                    [ sqrt(3)/2,        1/2]]):
         elif ((G='Td') or (G='S4')) then
            nnb:=3:   # number of operation with non-diagonal representation
            ng:=24:   # group order
   nirrep:=5:# number of irreps (number of classes)
          chartab:=matrix(nirrep,ng,[ [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
```

```
                                [ 1,-1,-1,-1,-1,-1,-1, 1, 1, 1, 1, 1,
     1, 1, 1,-1,-1,-1,-1,-1,-1, 1, 1, 1],
                                [ 2, 0, 0, 0, 0, 0, 0,-1,-1,-1,-1,-1,
    -1,-1,-1, 0, 0, 0, 0, 0, 0, 2, 2, 2],
   [ 3,-1,-1,-1,-1,-1,-1, 0, 0, 0, 0, 0,
     0, 0, 0, 1, 1, 1, 1, 1, 1,-1,-1,-1],
   [ 3, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0,
     0, 0, 0,-1,-1,-1,-1,-1,-1,-1,-1,-1] ]):
label:=array(1..nirrep):
#          #1st irrep
label[1]:=A1:
for n from 1 to ng do
    Girr[1][n]:=matrix(1,1,[[ 1]]):
od:
#          #2nd irrep
 label[2]:=A2:
 Girr[2][1] :=matrix(1,1,[[ 1]]):
 Girr[2][2] :=matrix(1,1,[[-1]]):
 Girr[2][3] :=matrix(1,1,[[-1]]):
 Girr[2][4] :=matrix(1,1,[[-1]]):
 Girr[2][5] :=matrix(1,1,[[-1]]):
```

```
Girr[2][6] :=matrix(1,1,[[-1]]):
Girr[2][7] :=matrix(1,1,[[-1]]):
Girr[2][8] :=matrix(1,1,[[ 1]]):
Girr[2][9] :=matrix(1,1,[[ 1]]):
Girr[2][10]:=matrix(1,1,[[ 1]]):
Girr[2][11]:=matrix(1,1,[[ 1]]):
Girr[2][12]:=matrix(1,1,[[ 1]]):
Girr[2][13]:=matrix(1,1,[[ 1]]):
Girr[2][14]:=matrix(1,1,[[ 1]]):
Girr[2][15]:=matrix(1,1,[[ 1]]):
Girr[2][16]:=matrix(1,1,[[-1]]):
Girr[2][17]:=matrix(1,1,[[-1]]):
Girr[2][18]:=matrix(1,1,[[-1]]):
Girr[2][19]:=matrix(1,1,[[-1]]):
Girr[2][20]:=matrix(1,1,[[-1]]):
Girr[2][21]:=matrix(1,1,[[-1]]):
Girr[2][22]:=matrix(1,1,[[ 1]]):
Girr[2][23]:=matrix(1,1,[[ 1]]):
Girr[2][24]:=matrix(1,1,[[ 1]]):
#          #3rd irrep
label[3]:=E:
```

```
Girr[3][1] :=matrix(2,2,[[ 1, 0],    #id
                         [ 0, 1]]):
Girr[3][2] :=matrix(2,2,[[ 1, 0],    #{12}
                         [ 0,-1]]):
Girr[3][3] :=matrix(2,2,[[-a,-b],    #{13}
                         [-b, a]]):
Girr[3][4] :=matrix(2,2,[[-a, b],    #{14}
                         [ b, a]]):
Girr[3][5] :=matrix(2,2,[[-a, b],    #{23}
                         [ b, a]]):
Girr[3][6] :=matrix(2,2,[[-a,-b],    #{24}
                         [ -b, a]]):
Girr[3][7] :=matrix(2,2,[[ 1, 0],    #{34}
                         [ 0,-1]]):
Girr[3][8] :=evalm(eval(Girr[3][2 ],4)&*eval(Girr[3][5],4)):   #{123} = {12}{23}
Girr[3][9] :=evalm(eval(Girr[3][3 ],4)&*eval(Girr[3][5],4)):   #{132} = {13}{23}
Girr[3][10]:=evalm(eval(Girr[3][2 ],4)&*eval(Girr[3][6],4)):   #{124} = {12}{24}
Girr[3][11]:=evalm(eval(Girr[3][4 ],4)&*eval(Girr[3][6],4)):   #{142} = {14}{24}
Girr[3][12]:=evalm(eval(Girr[3][3 ],4)&*eval(Girr[3][7],4)):   #{134} = {13}{34}
Girr[3][13]:=evalm(eval(Girr[3][4 ],4)&*eval(Girr[3][7],4)):   #{143} = {14}{34}
Girr[3][14]:=evalm(eval(Girr[3][5 ],4)&*eval(Girr[3][7],4)):   #{234} = {23}{34}
```

```
Girr[3][15]:=evalm(eval(Girr[3][6 ],4)&*eval(Girr[3][7],4)):    #{243} = {24}{34}
Girr[3][16]:=evalm(eval(Girr[3][8 ],4)&*eval(Girr[3][7],4)):    #{1234} = {123}{34}
Girr[3][17]:=evalm(eval(Girr[3][10],4)&*eval(Girr[3][7],4)):    #{1243} = {124}{34}
Girr[3][18]:=evalm(eval(Girr[3][9 ],4)&*eval(Girr[3][6],4)):    #{1324} = {132}{24}
Girr[3][19]:=evalm(eval(Girr[3][12],4)&*eval(Girr[3][6],4)):    #{1342} = {134}{24}
Girr[3][20]:=evalm(eval(Girr[3][11],4)&*eval(Girr[3][5],4)):    #{1423} = {142}{23}
Girr[3][21]:=evalm(eval(Girr[3][13],4)&*eval(Girr[3][5],4)):    #{1432} = {143}{23}
Girr[3][22]:=evalm(eval(Girr[3][2 ],4)&*eval(Girr[3][7],4)):    #{12}{34}
Girr[3][23]:=evalm(eval(Girr[3][3 ],4)&*eval(Girr[3][6],4)):    #{13}{24}
Girr[3][24]:=evalm(eval(Girr[3][4 ],4)&*eval(Girr[3][5],4)):    #{14}{23}
#          #4th irrep
 label[4]:=F1:
 Girr[4][1] :=matrix(3,3,[[ 1, 0, 0],    #id
                          [ 0, 1, 0],
                          [ 0, 0, 1]]):
 Girr[4][2] :=matrix(3,3,[[-1, 0, 0],    #{12}
                          [ 0, 0, 1],
                          [ 0, 1, 0]]):
 Girr[4][3] :=matrix(3,3,[[ 0, 0, 1],    #{13}
                          [ 0,-1, 0],
                          [ 1, 0, 0]]):
```

```
Girr[4][4] :=matrix(3,3,[[ 0, 1, 0],    #{14}
                         [ 1, 0, 0],
                         [ 0, 0,-1]]):
Girr[4][5] :=matrix(3,3,[[ 0,-1, 0],    #{23}
                         [-1, 0, 0],
                         [ 0, 0,-1]]):
Girr[4][6] :=matrix(3,3,[[ 0, 0,-1],    #{24}
                         [ 0,-1, 0],
                         [-1, 0, 0]]):
Girr[4][7] :=matrix(3,3,[[-1, 0, 0],    #{34}
                         [ 0, 0,-1],
                         [ 0,-1, 0]]):
Girr[4][8] :=evalm(eval(Girr[4][2 ],4)&*eval(Girr[4][5],4)):    #{123} = {12}{23}
Girr[4][9] :=evalm(eval(Girr[4][3 ],4)&*eval(Girr[4][5],4)):    #{132} = {13}{23}
Girr[4][10]:=evalm(eval(Girr[4][2 ],4)&*eval(Girr[4][6],4)):    #{124} = {12}{24}
Girr[4][11]:=evalm(eval(Girr[4][4 ],4)&*eval(Girr[4][6],4)):    #{142} = {14}{24}
Girr[4][12]:=evalm(eval(Girr[4][3 ],4)&*eval(Girr[4][7],4)):    #{134} = {13}{34}
Girr[4][13]:=evalm(eval(Girr[4][4 ],4)&*eval(Girr[4][7],4)):    #{143} = {14}{34}
Girr[4][14]:=evalm(eval(Girr[4][5 ],4)&*eval(Girr[4][7],4)):    #{234} = {23}{34}
Girr[4][15]:=evalm(eval(Girr[4][6 ],4)&*eval(Girr[4][7],4)):    #{243} = {24}{34}
Girr[4][16]:=evalm(eval(Girr[4][8 ],4)&*eval(Girr[4][7],4)):    #{1234} = {123}{34}
```

```
Girr[4][17]:=evalm(eval(Girr[4][10],4)&*eval(Girr[4][7],4)):    #{1243} = {124}{34}
Girr[4][18]:=evalm(eval(Girr[4][9 ],4)&*eval(Girr[4][6],4)):    #{1324} = {132}{24}
Girr[4][19]:=evalm(eval(Girr[4][12],4)&*eval(Girr[4][6],4)):    #{1342} = {134}{24}
Girr[4][20]:=evalm(eval(Girr[4][11],4)&*eval(Girr[4][5],4)):    #{1423} = {142}{23}
Girr[4][21]:=evalm(eval(Girr[4][13],4)&*eval(Girr[4][5],4)):    #{1432} = {143}{23}
Girr[4][22]:=evalm(eval(Girr[4][2 ],4)&*eval(Girr[4][7],4)):    #{12}{34}
Girr[4][23]:=evalm(eval(Girr[4][3 ],4)&*eval(Girr[4][6],4)):    #{13}{24}
Girr[4][24]:=evalm(eval(Girr[4][4 ],4)&*eval(Girr[4][5],4)):    #{14}{23}
#          #5th irrep
 label[5]:=F2:
 Girr[5][1] :=matrix(3,3,[[ 1, 0, 0],    #id
                          [ 0, 1, 0],
                          [ 0, 0, 1]]):
 Girr[5][2] :=matrix(3,3,[[ 1, 0, 0],    #{12}
                          [ 0, 0,-1],
                          [ 0,-1, 0]]):
 Girr[5][3] :=matrix(3,3,[[ 0, 0,-1],    #{13}
                          [ 0, 1, 0],
                          [-1, 0, 0]]):
 Girr[5][4] :=matrix(3,3,[[ 0,-1, 0],    #{14}
                          [-1, 0, 0],
```

```
                              [ 0, 0, 1]]):
Girr[5][5] :=matrix(3,3,[[ 0, 1, 0],    #{23}
                              [ 1, 0, 0],
                              [ 0, 0, 1]]):
Girr[5][6] :=matrix(3,3,[[ 0, 0, 1],    #{24}
                              [ 0, 1, 0],
                              [ 1, 0, 0]]):
Girr[5][7] :=matrix(3,3,[[ 1, 0, 0],    #{34}
                              [ 0, 0, 1],
                              [ 0, 1, 0]]):
Girr[5][8] :=evalm(eval(Girr[5][2 ],4)&*eval(Girr[5][5],4)):    #{123} = {12}{23}
Girr[5][9] :=evalm(eval(Girr[5][3 ],4)&*eval(Girr[5][5],4)):    #{132} = {13}{23}
Girr[5][10]:=evalm(eval(Girr[5][2 ],4)&*eval(Girr[5][6],4)):    #{124} = {12}{24}
Girr[5][11]:=evalm(eval(Girr[5][4 ],4)&*eval(Girr[5][6],4)):    #{142} = {14}{24}
Girr[5][12]:=evalm(eval(Girr[5][3 ],4)&*eval(Girr[5][7],4)):    #{134} = {13}{34}
Girr[5][13]:=evalm(eval(Girr[5][4 ],4)&*eval(Girr[5][7],4)):    #{143} = {14}{34}
Girr[5][14]:=evalm(eval(Girr[5][5 ],4)&*eval(Girr[5][7],4)):    #{234} = {23}{34}
Girr[5][15]:=evalm(eval(Girr[5][6 ],4)&*eval(Girr[5][7],4)):    #{243} = {24}{34}
Girr[5][16]:=evalm(eval(Girr[5][8 ],4)&*eval(Girr[5][7],4)):    #{1234} = {123}{34}
Girr[5][17]:=evalm(eval(Girr[5][10],4)&*eval(Girr[5][7],4)):    #{1243} = {124}{34}
Girr[5][18]:=evalm(eval(Girr[5][9 ],4)&*eval(Girr[5][6],4)):    #{1324} = {132}{24}
```

```
            Girr[5][19]:=evalm(eval(Girr[5][12],4)&*eval(Girr[5][6],4)):    #{1342} = {134}{24}
            Girr[5][20]:=evalm(eval(Girr[5][11],4)&*eval(Girr[5][5],4)):    #{1423} = {142}{23}
            Girr[5][21]:=evalm(eval(Girr[5][13],4)&*eval(Girr[5][5],4)):    #{1432} = {143}{23}
            Girr[5][22]:=evalm(eval(Girr[5][2 ],4)&*eval(Girr[5][7],4)):    #{12}{34}
            Girr[5][23]:=evalm(eval(Girr[5][3 ],4)&*eval(Girr[5][6],4)):    #{13}{24}
            Girr[5][24]:=evalm(eval(Girr[5][4 ],4)&*eval(Girr[5][5],4)):    #{14}{23}
    for i from 3 to 5 do
        for n from 1 to ng do
            Girr[i][n]:=map(combine,Girr[i][n],radical,symbolic):
                od:
            od:
        else
            print("Group ",G," not identified; program stopped.");
    stop
        end if:
        [ng,nirrep,chartab,label,Girr,nnb]:
end proc:
defk:=proc(m,nd1)
      local ndm,nd,mm,k,j,j1,j2,j3,j4,key:
      "Definition of order m tensor space key.":
      ndm:=nd1**m:
```

```
key:=matrix(m,ndm):
nd:=0:
if (m<1) then
   error "Tensor product defined only for m > 0; program stopped"
elif (m=1) then
    for j1 from 1 to nd1 do
        nd:=nd+1:
            key[1,nd]:=j1: # tensor key definition
    od:
elif (m=2) then
    for j1 from 1 to nd1 do
        for j2 from 1 to nd1 do
            nd:=nd+1:
            key[1,nd]:=j1: # tensor key definition
            key[2,nd]:=j2: # tensor key definition
        od:
    od:
elif (m=3) then
    for j1 from 1 to nd1 do
        for j2 from 1 to nd1 do
            for j3 from 1 to nd1 do
```

```
                    nd:=nd+1:
                    key[1,nd]:=j1: # tensor key definition
                    key[2,nd]:=j2: # tensor key definition
                    key[3,nd]:=j3: # tensor key definition
                od:
            od:
        od:
    elif (m=4) then
        for j1 from 1 to nd1 do
            for j2 from 1 to nd1 do
                for j3 from 1 to nd1 do
                    for j4 from 1 to nd1 do
                        nd:=nd+1:
                        key[1,nd]:=j1: # tensor key definition
                        key[2,nd]:=j2: # tensor key definition
                        key[3,nd]:=j3: # tensor key definition
                        key[4,nd]:=j4: # tensor key definition
                    od:
                od:
            od:
        od:
```

```
        else
            error "Tensor product definition limited to m = 4; program stopped"
        end if:
        [ndm,key]:
    end proc:
    repm:=proc(m,ndm,key,G1)
        local mm,k,j,Gm:
        "Definition of group representations in order m tensor space.":
        Gm:=matrix(ndm,ndm):
        for k from 1 to ndm do
            for j from 1 to ndm do
                Gm[k,j]:=1:
                for mm from 1 to m do
                    Gm[k,j]:=Gm[k,j]*G1[key[mm,k],key[mm,j]]:
                    Gm[k,j]:=simplify(Gm[k,j]):
                od:
            od:
        od:
    #       Gm:=map(combine,Gm,radical,symbolic):
    #      combine needs 3 times more memory!
        Gm:
```

```
end proc:
simtra:=proc(T,TT,G)
        local Gs:
        description "similarity transformation":
        Gs:=evalm(&*(TT,G,T)):
#        Gs:=linalg[multiply](TT,G,T):
Gs:=map(simplify,Gs):
#        Gs:=map(combine,Gs,radical,symbolic):
end proc:
Sm:=proc(m)
      local MP,chartab,ng,nirrep,label:
      description " Turmdarstellung of symmetric group Sm":
      if (m<1) then
         print("Permutations defined only for m > 0; program stopped");
 stop
      elif (m>4) then  #work in progress below for m=5
          print("Permutation matrices limited to m = 4; program stopped");
  stop
      end if:
      ng:=factorial(m):
      if (m=1) then
```

```
      nirrep:=1:
      MP[1]:=matrix(m,m,[[1]]):  # E
      chartab:=matrix(nirrep,ng,[[ 1]]):
      label:=array(1..nirrep,[A1]):
   elif (m=2) then
      nirrep:=2:
      MP[1]:=matrix(m,m,[[1,0],
         [0,1]]): # E
      MP[2]:=matrix(m,m,[[0,1],
         [1,0]]): # {12}
      chartab:=matrix(nirrep,ng,[[ 1, 1],
                                 [ 1,-1]]):
      label:=array(1..nirrep,[A1,A2]):
   elif (m=3) then
      nirrep:=3:
      MP[1]:=matrix(m,m,[[1,0,0],
[0,1,0],
[0,0,1] ]): # E
      MP[2]:=matrix(m,m,[[0,0,1],
[1,0,0],
[0,1,0] ]): # {123}
```

```
        MP[3]:=matrix(m,m,[[0,1,0],
    [0,0,1],
    [1,0,0] ]): # {132}
        MP[4]:=matrix(m,m,[[0,1,0],
    [1,0,0],
    [0,0,1] ]): # {12}
        MP[5]:=matrix(m,m,[[0,0,1],
    [0,1,0],
    [1,0,0] ]): # {13}
        MP[6]:=matrix(m,m,[[1,0,0],
    [0,0,1],
    [0,1,0] ]): # {23}
chartab:=matrix(nirrep,ng,[[ 1, 1, 1, 1, 1, 1],
                           [ 1, 1, 1,-1,-1,-1],
                      [ 2,-1,-1, 0, 0, 0]]):
        label:=array(1..nirrep,[A1,A2,E]):
  elif (m=4) then
        nirrep:=5:
        MP[1]:=matrix(m,m,[ [1,0,0,0],
                            [0,1,0,0],
                            [0,0,1,0],
```

```
[0,0,0,1]  ]): # E
    MP[2]:=matrix(m,m,[ [0,1,0,0],
                        [1,0,0,0],
                        [0,0,1,0],
[0,0,0,1]  ]): # {12}
    MP[3]:=matrix(m,m,[ [0,0,1,0],
                        [0,1,0,0],
                        [1,0,0,0],
[0,0,0,1]  ]): # {13}
    MP[4]:=matrix(m,m,[ [0,0,0,1],
                        [0,1,0,0],
                        [0,0,1,0],
[1,0,0,0]  ]): # {14}
    MP[5]:=matrix(m,m,[ [1,0,0,0],
                        [0,0,1,0],
                        [0,1,0,0],
[0,0,0,1]  ]): # {23}
    MP[6]:=matrix(m,m,[ [1,0,0,0],
                        [0,0,0,1],
                        [0,0,1,0],
[0,1,0,0]   ]): # {24}
```

```
    MP[7]:=matrix(m,m,[ [1,0,0,0],
                        [0,1,0,0],
                        [0,0,0,1],
[0,0,1,0]  ]): # {34}
    MP[8]:=matrix(m,m,[ [0,0,1,0],
                        [1,0,0,0],
                        [0,1,0,0],
[0,0,0,1]  ]): # {123}
    MP[9]:=matrix(m,m,[ [0,1,0,0],
                        [0,0,1,0],
                        [1,0,0,0],
[0,0,0,1] ]): # {132}
   MP[10]:=matrix(m,m,[ [0,0,0,1],
                        [1,0,0,0],
                        [0,0,1,0],
[0,1,0,0]  ]): # {124}
   MP[11]:=matrix(m,m,[ [0,1,0,0],
                        [0,0,0,1],
                        [0,0,1,0],
[1,0,0,0]  ]): # {142}
   MP[12]:=matrix(m,m,[ [0,0,0,1],
```

```
                              [0,1,0,0],
                              [1,0,0,0],
[0,0,1,0]  ]): # {134}
   MP[13]:=matrix(m,m,[ [0,0,1,0],
                              [0,1,0,0],
                              [0,0,0,1],
[1,0,0,0]  ]): # {143}
   MP[14]:=matrix(m,m,[ [1,0,0,0],
                              [0,0,0,1],
                              [0,1,0,0],
[0,0,1,0]  ]): # {234}
   MP[15]:=matrix(m,m,[ [1,0,0,0],
                              [0,0,1,0],
                              [0,0,0,1],
[0,1,0,0]  ]): # {243}
   MP[16]:=matrix(m,m,[ [0,0,0,1],
                              [1,0,0,0],
                              [0,1,0,0],
[0,0,1,0]  ]): # {1234}
   MP[17]:=matrix(m,m,[ [0,0,1,0],
                              [1,0,0,0],
```

```
                          [0,0,0,1],
[0,1,0,0]  ]): # {1243}
   MP[18]:=matrix(m,m,[ [0,0,0,1],
                        [0,0,1,0],
                        [1,0,0,0],
[0,1,0,0]  ]): # {1324}
   MP[19]:=matrix(m,m,[ [0,1,0,0],
                        [0,0,0,1],
                        [1,0,0,0],
[0,0,1,0]  ]): # {1342}
   MP[20]:=matrix(m,m,[ [0,0,1,0],
                        [0,0,0,1],
                        [0,1,0,0],
[1,0,0,0]  ]): # {1423}
   MP[21]:=matrix(m,m,[ [0,1,0,0],
                        [0,0,1,0],
                        [0,0,0,1],
[1,0,0,0]  ]): # {1432}
   MP[22]:=matrix(m,m,[ [0,1,0,0],
                        [1,0,0,0],
                        [0,0,0,1],
```

```
[0,0,1,0]  ]): # {12}{34}
   MP[23]:=matrix(m,m,[ [0,0,1,0],
                        [0,0,0,1],
                        [1,0,0,0],
[0,1,0,0]  ]): # {13}{24}
   MP[24]:=matrix(m,m,[ [0,0,0,1],
                        [0,0,1,0],
                        [0,1,0,0],
[1,0,0,0]  ]): # {14}{23}
   chartab:=matrix(nirrep,ng,[ [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                              [ 1,-1,-1,-1,-1,-1,-1, 1, 1, 1, 1, 1,
        1, 1, 1,-1,-1,-1,-1,-1,-1, 1, 1, 1],
                              [ 2, 0, 0, 0, 0, 0, 0,-1,-1,-1,-1,-1,
       -1,-1,-1, 0, 0, 0, 0, 0, 0, 2, 2, 2],
       [ 3, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0,
         0, 0, 0,-1,-1,-1,-1,-1,-1,-1,-1,-1],
       [ 3,-1,-1,-1,-1,-1,-1, 0, 0, 0, 0, 0,
         0, 0, 0, 1, 1, 1, 1, 1, 1,-1,-1,-1] ]):
   label:=array(1..nirrep,[A1,A2,E,F1,F2]):
 elif (m=5) then  #in work, needs completion
```

```
    nirrep:=7:
    MP[1]:=matrix(m,m,[ [1,0,0,0,0],
                        [0,1,0,0,0],
                        [0,0,1,0,0],
[0,0,0,1,0],
                        [0,0,0,0,1] ]): # E
    MP[2]:=matrix(m,m,[ [0,1,0,0,0],
                        [1,0,0,0,0],
                        [0,0,1,0,0],
[0,0,0,1,0],
                        [0,0,0,0,1] ]): # {12}
    MP[3]:=matrix(m,m,[ [0,0,1,0,0],
                        [0,1,0,0,0],
                        [1,0,0,0,0],
[0,0,0,1,0],
                        [0,0,0,0,1] ]): # {13}
    MP[4]:=matrix(m,m,[ [0,0,0,1,0],
                        [0,1,0,0,0],
                        [0,0,1,0,0],
[1,0,0,0,0],
                        [0,0,0,0,1] ]): # {14}
```

```
    MP[5]:=matrix(m,m,[ [0,0,0,0,1],
                        [0,1,0,0,0],
                        [0,0,1,0,0],
[0,0,0,1,0],
                        [1,0,0,0,0] ]): # {15}
    MP[6]:=matrix(m,m,[ [1,0,0,0,0],
                        [0,0,1,0,0],
                        [0,1,0,0,0],
[0,0,0,1,0],
                        [0,0,0,0,1] ]): # {23}
    MP[7]:=matrix(m,m,[ [1,0,0,0,0],
                        [0,0,0,1,0],
                        [0,0,1,0,0],
[0,1,0,0,0],
                        [0,0,0,0,1] ]): # {24}
    MP[8]:=matrix(m,m,[ [1,0,0,0,0],
                        [0,0,0,0,1],
                        [0,0,1,0,0],
[0,0,0,1,0],
                        [0,1,0,0,0] ]): # {25}
    MP[9]:=matrix(m,m,[ [1,0,0,0,0],
```

```
                              [0,1,0,0,0],
                              [0,0,0,1,0],
        [0,0,1,0,0],
                              [0,0,0,0,1] ]): # {34}
           MP[10]:=matrix(m,m,[ [1,0,0,0,0],
                              [0,1,0,0,0],
                              [0,0,0,0,1],
        [0,0,0,1,0],
                              [0,0,1,0,0] ]): # {35}
           MP[11]:=matrix(m,m,[ [1,0,0,0,0],
                              [0,1,0,0,0],
                              [0,0,1,0,0],
        [0,0,0,0,1],
                              [0,0,0,1,0] ]): # {45}
    MP[12]:=evalm(eval(MP[2],4)&*eval(MP[9],4)):    # {12}{34}
    ### needs to be completed, 120 operations to be defined as products
      # of the 10 foregoing primitive permutations
           chartab:=matrix(nirrep,ng,
            [ [ 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

```
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
[ 1,
 -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
 -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
[ 4,
  2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1],
[ 4,
 -2,-2,-2,-2,-2,-2,-2,-2,-2,-2,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1],
[ 5,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[ 5,
 -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
 -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[ 6,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 -2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,
```

```
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]):
          label:=array(1..nirrep,[A1,A2,G1,G2,H1,H2,I]):
        end if:
        [ng,nirrep,chartab,label,eval(MP)]:
    end proc:
    SmInVm:=proc(m,ndm,ns,key,pmat)
        local v,TD,nop,k,j,jj,u,sss,kki,kk,kks,kkl,kkk:
        description "Calculation of SmInVm using turmdarstellung of Sm";
        v:=array(1..m):
        for nop from 1 to ns do
    #       TD[nop]:=LinearAlgebra[Matrix](ndm,ndm,shape=zero):
            TD[nop]:=Matrix(ndm,ndm):
            kks:=ndm:
      for kk from 1 to ndm do
            kki[kk]:=kk:
          od:
          for k from 1 to ndm do
            for j from 1 to m do
```

```
                    v[j]:=key[j,k]:
                od:
                u:=linalg[multiply](pmat[nop],v):
                for kk from 1 to kks do
                    sss:=0:
                    for j from 1 to m do
                        sss:=sss+(key[j,kki[kk]]-u[j])**2:
                    od:
                    if (sss=0) then
                       TD[nop][kki[kk],k]:=1:
                       kks:=kks-1:
      for kkk from kk to kks do
                           kki[kkk]:=kki[kkk+1]:
                       od:
                       break:
                    end if
                od:
            od:
        od:
        TD:
end proc:
```

```
redA1:= proc(ng,nirr,chartab,ndim,rep,runtime)
        local nop,mu,k,l,km,chi,sss,redfac,T,NC,VC,GS,DT,dirrp,label,TT,ID,nD,repn,gammu:
        description "calculation of symmetry reduction by projection":
        chi:=array(1..ng):
redfac:=array(1..nirr):
label:=array(1..ndim):
dirrp:=array(1..nirr,[seq(chartab[k,1],k=1..nirr)]): #dimension of irrep
        for nop from 1 to ng do # get character of representation rep
            chi[nop]:=0:
            for k from 1 to ndim do
                chi[nop]:=chi[nop]+rep[nop][k,k]:
            od:
            chi[nop]:=simplify(expand(chi[nop])):
        od:
        for mu from 1 to nirr do # calculation of reduction factors
            sss:=0:
            for nop from 1 to ng do
                sss:=sss+chi[nop]*chartab[mu,nop]:
            od:
            sss:=simplify(expand(sss)):
            redfac[mu]:=sss/ng:
```

```
            if not type(redfac[mu],integer) then
               print("Reduction factor for irrep ",mu," is not integer."):
               print("Check representation. Program stopped."):
#              stop
            end if:
        od:
        for mu from 1 to 1 do # calculate projection on A1 only
            T:=Matrix(ndim,ndim,shape=zero):
            for nop from 1 to ng do
        repn:=matrix(ndim,ndim):
                repn:=eval(rep[nop]):
                T:=evalm(T+chartab[mu,nop]*repn):
            od:
            gammu:=redfac[mu]*dirrp[mu]:
            GS:=orthonorm(ndim,ndim,T,gammu,runtime): # Procedure orthonorm returns 2 items:
            NC[mu]:=GS[1]:                # number of non-zero vectors
    VC[mu]:=evalm(GS[2]):        # list of NC column vectors
                                  # of length ndim
            #print(NC[mu]);
            #print(VC[mu]);
            if (NC[mu] <> redfac[mu]*dirrp[mu]) then
```

```
                print("Mismatch in projection onto irrep space mu =",mu):
                print(NC[mu],redfac[mu]):
                error "Program stopped."
            end if:
        od:
        T:=matrix(ndim,redfac[1]):
        k:=0:
        for mu from 1 to 1 do
            for km from 1 to NC[mu] do
                k:=k+1:
                for l from 1 to ndim do
                    T[l,k]:=VC[mu][km][l]:
                od:
            od:
        od:
        redfac:=eval(redfac,4):
        T:=eval(T,4):
[redfac,T]:
end proc:
redfac:=proc(ng,nirr,chartab,labtab,ndim,rep)
        local nop,mu,k,l,km,chi,sss,redf,T,NC,VC,GS,DT,dirrp,label,TT,ID,nD,repn:
```

```
        description "calculation of symmetry reduction by projection":
        chi:=array(1..ng):
redf:=array(1..nirr):
label:=array(1..ndim):
dirrp:=array(1..nirr,[seq(chartab[k,1],k=1..nirr)]): #dimension of irrep
        for nop from 1 to ng do # get character of representation rep
            chi[nop]:=0:
            for k from 1 to ndim do
                chi[nop]:=chi[nop]+rep[nop][k,k]:
            od:
            chi[nop]:=simplify(expand(chi[nop])):
        od:
        for mu from 1 to nirr do # calculation of reduction factors
            sss:=0:
            for nop from 1 to ng do
                sss:=sss+chi[nop]*chartab[mu,nop]:
            od:
            sss:=simplify(expand(sss)):
            redf[mu]:=sss/ng:
            if not type(redf[mu],integer) then
               print("Reduction factor for irrep ",mu," is not integer.");
```

```
                error "Check representation. Program stopped."
            end if:
        od:
redf:
end proc:
reduce:=proc(mu,gammu,ng,Gi,chartab,ndim,G,runtime)
        local dmu,rdim,nmu,nop,k,l,P,NC,VC,GS,repn,S,ST,sss,nopst,R,A,DA,U,AD,mmu,T,no,nno,kmu,ph:
        description "calculation of symmetry reduction in subspace mu by projection; hammermesh idea":
dmu:=eval(chartab[mu,1]): #dimension of irrep mu
        no:=6000:
rdim:=dmu*gammu:
nmu:=1:
nopst:=1:
P:=Matrix(ndim,ndim,shape=zero):
for nop from 1 to ng do
    P:=evalm(P+dmu/ng*eval(Gi[mu][nop][nmu,nmu])*G[nop]):
od:
# print(P);
GS:=orthonorm(ndim,ndim,P,gammu,runtime): # Procedure orthonorm returns 2 items:
        NC:=GS[1]:              # number of non-zero vectors
VC:=evalm(GS[2]):       # list of NC column vectors
```

```
                                        # of length ndim
        unassign('GS'):
        #print(NC);
#print(VC);
if (NC <> gammu) then
   print("Mismatch in projection onto irrep space mu =",mu):
   print(NC,gammu):
   error "Program stopped."
end if:
        S[nmu]:=matrix(ndim,gammu):
for k from 1 to NC do
    for l from 1 to ndim do
S[nmu][l,k]:=VC[k][l]
    od:
od:
        nno:=no+1:
prtime(outputfile,nno,"REDUCE: get S1",runtime):
#        print(S[nmu]);
for mmu from 2 to dmu do
    P:=Matrix(ndim,ndim,shape=zero):
    for nop from 1 to ng do
```

```
            P:=evalm(P+dmu/ng*eval(Gi[mu][nop][mmu,nmu])*G[nop]):
        od:
        S[mmu]:=evalm(P&*S[nmu]):
#     for k from 1 to gammu do              #normalize S
# VC:=array(1..ndim,[seq(S[mmu][l,k],l=1..ndim)]):
# NC[k]:=vecnorm(VC):
# if (NC[k] = 0) then
#    error "trivial projection stop"
# end if:
#     od:
    for k from 1 to gammu do
     for l from 1 to ndim do
#     S[mmu][l,k]:=combine(S[mmu][l,k]/NC[k],radical,symbolic):
    S[mmu][l,k]:=combine(S[mmu][l,k],radical,symbolic):
od:
            od:
#     print(S[nmu]);
            nno:=no+nmu:
    prtime(outputfile,nno,"REDUCE: get Si, i>1",runtime):
od:
T:=matrix(ndim,rdim):
```

```
        k:=0:
for mmu from 1 to gammu do
            A:=matrix(ndim,dmu):
            for nmu from 1 to dmu do
        for l from 1 to ndim do
    A[l,nmu]:=S[nmu][l,mmu]:
                od:
            od:
            ST:=linalg[transpose](A):
#      print(G[nopst]):
            GS:=simtra(A,ST,G[nopst]):
            GS:=eval(GS):
#      print(GS,Gi[mu][nopst]):
    for nmu from 1 to dmu do
        ph[nmu]:=1:
            od:
    for nmu from 1 to dmu do
        k:=k+1:
for kmu from 1 to dmu do   #phase determination
                    if (kmu <> nmu) then   #just for testing
     sss:=eval(Gi[mu][nopst][nmu,kmu],4):
```

```
            if (sss <> 0) then
                                ph[nmu]:=ph[nmu]*ph[kmu]*eval(GS[nmu,kmu],4)/sss:
                                break:
            end if:
                            end if:
                        od:
    #                   print(mmu,nmu,kmu,sss,ph[nmu]);
    if (evalf(ph[nmu]**2) <> 1) then
       error "Phase error in reduce; program stopped"
                        end if:
    for l from 1 to ndim do
        T[l,k]:=A[l,nmu]*ph[nmu]:
    #     T[l,k]:=S[nmu][l,mmu]:
                        od:
                    od:
            od:
    if (k <> rdim) then
       error "Mismatch in reduce: nb of columns <> rdim; program stopped."
            end if:
    T:=map(combine,T,radical,symbolic):
    end proc:
```

```
orthonorm:=proc(nl,nk,mat,gamma,runtime)
            local v,vk,vc,nv,nc,k0,cn,k,l,n,no,nno:
            "Orthonormalization of nlXnc matrix mat following Gram-Schmidt":
            no:=1000:
            nc:=0:
            v:=array(1..nl):
            vk:=array(1..nl):
            k0:=nk+1:
            for k from 1 to nk do
                for l from 1 to nl do
                    v[l]:=mat[l,k]:
                od:
                nv:=vecnorm(v):
                if (evalf(nv) > 0) then
                   nc:=1:                # first non-zero vector
                   vc[nc]:=array(1..nl):
  vc[nc]:=evalm(1/nv*v):
                   vc[nc]:=map(simplify,vc[nc]):
                   k0:=k:
  k:=nk+1:
                   nno:=no+nc:
```

```
            prtime(outputfile,nno,"ORTHONORM     ",runtime):
        end if:
    od:
    for k from (k0+1) to nk do
        if (nc=gamma) then
            break:
        end if:
        for l from 1 to nl do
            vk[l]:=mat[l,k]:
            v[l]:=vk[l]:
        od:
        for n from 1 to nc do
            cn:=evalm(vk&*vc[n]):
    v:=evalm(v-cn*vc[n]):
            v:=map(simplify,v):
        od:
        nv:=vecnorm(v):
        if (evalf(nv) <> 0) then
           nc:=nc+1:
           vc[nc]:=array(1..nl):
           vc[nc]:=evalm(1/nv*v):
```

```
                    vc[nc]:=map(simplify,vc[nc]):
                    nno:=no+nc:
                    prtime(outputfile,nno,"ORTHONORM     ",runtime):
                end if:
            od:
            [nc,vc]:
end proc:
vecnorm:=proc(v)
      local nv,lv,i:
      nv:=0:
      lv:=nops(convert(v,list)):
      for i from 1 to lv do
          nv:=nv+v[i]**2:
      od:
      nv:=sqrt(nv):
end proc:
```

D sample input

```
# Input values:
  M:=3:          # order of tensor space
  GROUP:=C3v:    # symmetry group
  NDIM1:=3:      # dimension of first order linear space
  MUE1:=[1,3]:   # irreps of GROUP used to build up V1 space
  MMU1:=[1,1]:   # multiciply of irreps
  V1I[1][1]:=[SA1]:          # choice of component symbols
  V1I[3][1]:=[SEa,SEb]:      # choice of component symbols
  read "turm.cmd":
  read "main.cmd":
```

E sample output

```
Step 1     ( loading                          ) done;   0 h  0 m 0.200 s
Step 2     ( read irreps                      ) done;   0 h  0 m 0.205 s
Step 3     ( setup GAMMA1                     ) done;   0 h  0 m 0.220 s
Step 4     ( define tensor space key          ) done;   0 h  0 m 0.222 s
Step 501   ( setup tensor space, operation    ) done;   0 h  0 m 0.332 s
Step 502   ( setup tensor space, operation    ) done;   0 h  0 m 0.538 s
Step 503   ( setup tensor space, operation    ) done;   0 h  0 m 0.654 s
Step 504   ( setup tensor space, operation    ) done;   0 h  0 m 0.760 s
Step 505   ( setup tensor space, operation    ) done;   0 h  0 m 0.874 s
Step 506   ( setup tensor space, operation    ) done;   0 h  0 m 1.034 s
Step 5     ( setup tensor space               ) done;   0 h  0 m 1.035 s
Step 601   ( reduction in Sm, operation       ) done;   0 h  0 m 1.131 s
Step 602   ( reduction in Sm, operation       ) done;   0 h  0 m 1.229 s
Step 603   ( reduction in Sm, operation       ) done;   0 h  0 m 1.361 s
Step 604   ( reduction in Sm, operation       ) done;   0 h  0 m 1.428 s
Step 605   ( reduction in Sm, operation       ) done;   0 h  0 m 1.493 s
Step 606   ( reduction in Sm, operation       ) done;   0 h  0 m 1.556 s
Step 6     ( reduction in Sm                  ) done;   0 h  0 m 1.566 s
Step 7     ( reduction factors                ) done;   0 h  0 m 1.567 s
```

```
      Step 1001  ( ORTHONORM                        ) done;   0 h  0 m 1.688 s
      Step 1002  ( ORTHONORM                        ) done;   0 h  0 m 1.713 s
      Step 1003  ( ORTHONORM                        ) done;   0 h  0 m 1.756 s
      Step 6001  ( REDUCE: get S1                   ) done;   0 h  0 m 1.757 s
      Step 1001  ( ORTHONORM                        ) done;   0 h  0 m 1.875 s
      Step 6001  ( REDUCE: get S1                   ) done;   0 h  0 m 1.875 s
      Step 1001  ( ORTHONORM                        ) done;   0 h  0 m 2.008 s
      Step 1002  ( ORTHONORM                        ) done;   0 h  0 m 2.027 s
      Step 1003  ( ORTHONORM                        ) done;   0 h  0 m 2.068 s
      Step 6001  ( REDUCE: get S1                   ) done;   0 h  0 m 2.069 s
71    Step 6001  ( REDUCE: get Si, i>1              ) done;   0 h  0 m 2.152 s
      Step 8     ( reduce GAMMAmr                   ) done;   0 h  0 m 2.193 s
      Step 9     ( tensor transformations           ) done;   0 h  0 m 2.703 s
       type A1   , operation 1   , difference 0.0
       type A1   , operation 2   , difference 0.0
       type A1   , operation 3   , difference 0.0
       type A1   , operation 4   , difference 0.0
       type A1   , operation 5   , difference 0.0
       type A1   , operation 6   , difference 0.0
       type A2   , operation 1   , difference 0.0
       type A2   , operation 2   , difference 0.0
```

```
 type A2   , operation 3   , difference 0.0
 type A2   , operation 4   , difference 0.0
 type A2   , operation 5   , difference 0.0
 type A2   , operation 6   , difference 0.0
 type E    , operation 1   , difference 0.0
 type E    , operation 2   , difference 0.0
 type E    , operation 3   , difference 0.0
 type E    , operation 4   , difference 0.0
 type E    , operation 5   , difference 0.0
 type E    , operation 6   , difference 0.0
-----------------------------------------------------------------

Final results:

1) Original tensor space dimension:       27
   Reduced  tensor space dimension:       10

2) Reduction factors:                     LABEL[mu]  GAMMA[mu]
                                          ---------  ---------
                                          A1         3
                                          A2         1
```

E 3

3) Symmetrized expressions:

(k=1..GAMMA[mu], n=1..DIM[mu])

LABEL[mu]	k	n	S[mu][k][n]
A1	1	1	SA1^3
A1	2	1	1/2*SA1*(SEa^2+SEb^2)*6^(1/2)
A1	3	1	1/2*SEa*(SEa^2-3*SEb^2)
A2	1	1	1/2*SEb*(3*SEa^2-SEb^2)
E	1	1	3^(1/2)*SA1^2*SEa
E	1	2	3^(1/2)*SA1^2*SEb
E	2	1	1/2*SA1*(SEa+SEb)*(SEa-SEb)*6^(1/2)
E	2	2	-6^(1/2)*SA1*SEa*SEb

```
      E          3    1   1/2*3^(1/2)*(SEa^2+SEb^2)*SEa

      E          3    2   1/2*3^(1/2)*(SEa^2+SEb^2)*SEb

Step 11    ( **************                    ) done;   0 h  0 m 2.928 s
```

74

F unix script

```
#!/bin/tcsh
# command file for running reduce program: reducing tensor representations
  if ( $#argv == 3 ) goto goodargs
  echo "Usage: reduce.com <NORD> <NDIM (1st order)> <Group>"
  exit 1
#
  goodargs:
  set NORD=$argv[1]
  set NDIM=$argv[2]
  set GROUP=$argv[3]
  set INPUT=run-$NORD-$NDIM-$GROUP.cmd
  maple -i $INPUT
  exit
```