ELSEVIER

Computer Physics Communications 103 (1997) 51–73

Computer Physics Communications

Maple procedures for the coupling of angular momenta I. Data structures and numerical computations

S. Fritzsche [1]

Fachbereich Physik, Universität Kassel, Heinrich-Plett-Str. 40, D-34132 Kassel, Germany

Received 16 July 1996

Abstract

The theory of angular momentum and of spherical tensor operators leads to algebraic expressions which are usually written in terms of generalized Clebsch–Gordan coefficients and/or Wigner n–j symbols. In principle, the evaluation and simplification of such expressions is a straightforward task but it can also become extremely cumbersome in more complex applications, for instance, in atomic and nuclear structure theory or in the study of angular dependent properties. In these fields, simplification techniques are either based on graphical methods or on the explicit knowledge of special values and sum rules which can be found in some standard form in the literature. The direct application of these rules, however, is often laborious due to a large number of symmetric forms of the Wigner and related symbols and due to the complexity of the expressions in Racah algebra.

In order to facilitate the evaluation of Racah algebra expressions, a set of Maple procedures is presented for interactive work. In this paper, I first define proper data structures to deal with Racah algebra. These structures are the basis to provide procedures for various numerical computations. The use of recursion formulas and simplifications of typical expressions due to special values is also supported here. The impact of this interactive tool on atomic many-body perturbation theory is briefly discussed.

Keywords: Angular momentum; Vector coupling; Wigner n–j symbols; Racah algebra techniques; Spherical operators
PACS: 03.65.Fd; 02.90.+p

PROGRAM SUMMARY

Title of program: Racah

Catalogue identifier: ADFV

Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland

Licensing provisions: none

Computer for which the program is designed and others on which it is operable: All computers with a license of the computer algebra (CA) package Maple [1]

Operating systems under which the program has been tested: AIX 3.2.5

Program language used: Maple V, Release 3

[1] E-mail: fritzsch@physik.uni-kassel.de

PII S0010-4655(97)00032-5

Memory required to execute with typical data: 100000 words

No. of bytes in distributed program, including test data, etc.: 363852

Distribution format: ASCII

Keywords: Angular momentum, vector coupling, Wigner n–j symbols, Racah algebra techniques, spherical operators

Nature of the physical problem

Computer algebra (CA) is used to evaluate Wigner n–j symbols and vector coupling coefficients and to simplify typical expressions which appear by using Racah algebra techniques. A set of Maple commands for interactive work is presented.

Method of solution

The simplification of typical Racah algebra expressions is based on the numerical computation of Wigner n–j symbols and the explicit knowledge of symmetries, special values, orthogonality properties, and sum rules. To apply these rules by means of CA we first define the term *Racah expression* and also proper data structures for the internal representation of typical expressions. These structures form the basis to set up a general scheme for the simplification of Racah expressions in a series of steps. All symmetric forms of the Wigner n–j symbols are taken into account. This paper, in particular, deals with the implementation of numerical computations for Racah algebra expressions, recursion formulas, and with simplifications due to special values.

Restrictions onto the complexity of the problem

The Racah package is mainly based on the properties which are known for as Wigner 3–j and 6–j symbols. For 9–j symbols, we will only implement a few sum rules. The numerical computation of such symbols, however, is supported. There is currently a maximal number of 4 internal summation variables which are accepted in the explicit numerical evaluation of Racah algebra expressions. Higher n–j symbols ($n = 12, 15, \ldots$) could be defined in different ways; they are not included in this program.

In the present version, we only implement recursion relations for 3–j symbols. Furthermore, special values are restricted to a set of 3–j and 6–j symbols as they are given in the tabulation of Edmonds [2], i.e. with quantum numbers which differ by no more than 2. The full support for the simplification of general expressions due to orthogonality and sum rules which are known for the Wigner symbols, however, will be the subject of forthcoming work.

Unusual features of the program

All commands of the Racah package are available for interactive work. The program is based on data structures which are suitable for almost any complexity of Racah algebra expressions. More enhanced expressions which include the summation over various quantum numbers, weight factors, some phase as well as any number of n–j symbols are built up from simpler data structures. Wigner n–j symbols with all arguments having integer or half-integer values can be computed both as floating point number (with a precision due to the global `Digits` variable) or as algebraic expression. Three appendices summarize the most important mathematical relations for the manipulation and computation of typical expressions, the internal data structures, as well as all commands at the user level for quick reference. There is also some on-line help available. In addition to the classical symmetric forms of Wigner symbols, the package enables to deal with the full extended range of symmetries due to Regge [3].

Typical running time

All examples of the long write-up takes about 20 seconds on an IBM workstation.

References

[1] Maple is an established computer algebra program and a registered trademark of Waterloo Maple Inc.

[2] A.R. Edmonds, Angular Momentum in Quantum Mechanics (Princeton University Press, New York, 1957).

[3] T. Regge, Nuovo Cimento 10 (1958) 544.

LONG WRITE-UP

1. Introduction

The theory of angular momentum is practically indispensable for the study of quantum many-particle systems. This theory is not only concerned with closed physical systems where the total angular momentum is conserved and all properties, due to the isotropy of the space, are independent of rotations. Many other angular dependent properties, for instance in atomic and nuclear scattering, are also invariant under some coordinate rotation and can, thus, considerably be simplified by using the theory of angular momentum. To exploit rotational symmetries, a very powerful mathematical technique – the algebra of irreducible tensor operators – was developed by Racah [1] in the early forties. The first advanced applications of this technique dealt with atomic and nuclear structure theory [2,3]; nowadays, however, there exists a large number of other applications in particle physics as well as in atomic and molecular scattering.

The power of Racah's algebra is that it enables the analytic integration over angular coordinates in the evaluation of many-particle matrix elements. If, for the moment, we consider an N-electron system, the analytic treatment of angular coordinates allows one to reduce the problem which depends on $3N$ spatial (and further spin) coordinates to an equivalent one with only N (radial) coordinates. Moreover, the use of Racah algebra techniques separates geometrical factors from the actual physical interactions, thus providing a deeper insight into the physics of many-particle systems.

The theory of angular momentum and various applications of the Racah algebra have been presented in several texts and monographs during the past (for a few selected examples[2] see Ref. [2–7]). In earlier times, many different notations and symbols were first introduced into the literature. Nowadays, however, most presentations of this theory are now based on the Wigner n–j symbols. These n–j symbols are algebraic quantities with a high symmetry which obey various orthogonality relations and sum rules. The application of standard Racah algebra techniques often results in rather complex expressions including multiple sums of products of n–j symbols. Even though the simplification of such expressions is, in a certain sense, straightforward using the known symmetries and sum rules, work becomes extremely laborious in more difficult cases. For that reason, various graphical methods for the evaluation of Racah algebra expressions have been developed during the last decades [8–10]. In such graphical schemes, the diagrammatic representation of the Racah algebra can also be extended to include indications of phases and weight factors, but again the simplification process becomes elaborate in more complex applications.

A new alternative is now offered by use of computer algebra (CA). Several CA programs and languages like Maple[3], Mathematica[4], and others are available today. To exploit these facilities, I designed a program within the framework of Maple to evaluate and simplify typical Racah algebra expressions automatically. Attention has been given to developing an interactive tool which is flexible and powerful enough for most applications. Since the evaluation of more difficult expressions is known to be tedious and time-consuming, these facilities might be useful for both, the active work with Racah algebra techniques as well as for studying the literature if the reader wants to follow up some derivation in detail.

In many computer languages, the calculation of individual n–j symbols is now supported by library subprograms. Such programs, however, are only applicable in the final stage to perform numerical computation of some formulas; they usually do not allow for any simplification of an algebraic expression. This restriction also applies to special commands in Mathematica[5] which are available for the computation of n–j symbols. The main intention of this work, by contrast, is the evaluation of algebraic expressions by applying various techniques for simplification.

The Racah program will be presented in several parts. This paper here is first dedicated to define appropriate date structures and to perform numerical computations on Wigner n–j symbols and more complex expressions in Racah algebra. Various recursion formulas as well as the simplification of typical expressions by means of special values is also incorporated in the current version. The full support for the evaluation of Racah algebra expression by applying orthogonality and sum rules, on the other hand, is now under work and will be the subject of a different part in this series.

In the next two sections, I will give a brief introduction to the coupling of angular momenta and Wigner n–j symbols, followed by the definition of a *Racah expression*. At this stage, various strategies to simplify general expressions are also explained. A few simpler examples, a summary to the Racah package as well as various test cases are later described in Sections 4–6. The impact of such a computational tool on the study of many-particle systems is discussed with the example of atomic many-body perturbation theory. In this theory,

[2] A rather complete bibliography can be found in Ref. [7].

[3] Maple is a registered trademark of Waterloo Maple Inc.

[4] Mathematica is a registered trademark of Wolfram Research Inc.

[5] Mathematica recognizes a few special values known for n–j values but does not offer data structures to deal with more general Racah algebra expressions.

perturbation expansions for atomic properties are expressed in terms of Feynman–Goldstone diagrams [10]; for such expansions, then, Racah algebra techniques facilitate the analytic integration over angular coordinates and help to reduce the diagrams up to some radial integration.

2. Wigner n–j symbols

Clebsch–Gordan coefficients and Wigner n–j symbols ($n = 3, 6, 9$) play a dominant role in the coupling of angular momenta and the decomposition of reducible representations of composed states. Let us first consider some physical system which consists of two subsystems with definite angular momenta $|j_1 m_1\rangle$ and $|j_2 m_2\rangle$. A wavefunction of the total system with angular momentum (J, M) can then be constructed according to

$$|j_1 j_2 JM\rangle = \sum_{m_1 m_2} |j_1 m_1\rangle\, |j_2 m_2\rangle\, \langle j_1 m_1 j_2 m_2 \,|\, JM\rangle \tag{1}$$

from the wavefunctions of the subsystems. The coefficients $\langle j_1 m_1 j_2 m_2 \,|\, JM\rangle$ in Eq. (1) are the Clebsch–Gordan or vector-coupling coefficients; here, we use the notation of Brink and Satchler [6] which is also very similar to that one by Edmonds [4]. Nowadays, however, the Clebsch–Gordan coefficients are often expressed in terms of Wigner 3–j symbols,

$$\langle j_1 m_1 j_2 m_2 \,|\, j_3 m_3\rangle = (-1)^{j_1 - j_2 + m_3} [j_3]^{1/2} \begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & -m_3 \end{pmatrix}. \tag{2}$$

which possess much simpler symmetry properties. In Eq. (2), we applied the phase convention of Condon and Shortley [11] and the abbreviation $[a] = (2a + 1)$ frequently found in Racah algebra. The 3–j symbols also have some physical meaning in the coupling of three angular momenta where they represent the probability amplitude to yield a zero total angular momentum, $J = 0$. In much more detail, Clebsch–Gordan coefficients and Wigner 3–j symbols as well as important relations among them are explained, for example, in the textbooks of Edmonds [4], Brink and Satchler [6] and Varshalovich et al. [7].

Apart from Wigner 3–j symbols, there can often be found so-called 6–j and 9–j symbols in the recoupling of three and four angular momenta, respectively. They are written with curly brackets as $\begin{Bmatrix} j_1 & j_2 & j_3 \\ j_4 & j_5 & j_6 \end{Bmatrix}$ and $\begin{Bmatrix} j_1 & j_2 & j_3 \\ j_4 & j_5 & j_6 \\ j_7 & j_8 & j_9 \end{Bmatrix}$; a few examples for these symbols will occur in Sections 4 and 6 below. Other n–j symbols of even higher rank ($n = 12, 15, \ldots$) can also be defined (see Ref. [7]); their definition, however, is not anymore unique and they are much less important for practical applications. Moreover, in order to evaluate Wigner n–j symbols with $n \geq 9$, these symbols are usually expressed in terms of multiple sums over 3–j and/or 6–j symbols. Thus, n–j symbols with $n > 9$ will not be considered here.

Many different notations have been used in the earlier literature to express quantities closely related to the Wigner symbols. In the latest literature, however, the Wigner symbols are predominantly used. A rather complete compilation of earlier notations can, for instance, be found by Rotenberg et al. [12] and in the monograph of Varshalovich et al. [7]. I will also refer to these standard texts for most relations which are implemented in the Racah program.

The Wigner n–j symbols ($n = 3, 6, 9$) are algebraic functions of six or nine arguments which are only defined, i.e. nonzero, if all arguments fulfill proper conditions for the coupling of angular momenta. Additionally, the 3–j symbol, $\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix}$, is nonzero just for $m_1 + m_2 + m_3 = 0$. Tabulations for a large range of arguments as well as various explicit expressions for the 3–j and 6–j symbols as sums of factorials are given in different Refs. [4,12]. Appendix A shows the formulas which are applied for numerical computations in the present

program. There also exist various integral representations of these symbols which need not to be considered here. In a few special cases, the summation in the algebraic formulas can be carried out explicitly and the final expressions are then often called *special values*. The same term will also be used below to characterize one path in the simplification of Racah algebra expressions. Important special values are listed by Edmonds [4], Appendix 2, and to a much larger extent by Varshalovich and co-workers [7].

A crucial role in using Wigner n–j symbols is played by the symmetry of these quantities. For example, the 3–j symbol satisfies the symmetries

$$\begin{aligned}\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} &= \begin{pmatrix} j_2 & j_3 & j_1 \\ m_2 & m_3 & m_1 \end{pmatrix} = \begin{pmatrix} j_3 & j_1 & j_2 \\ m_3 & m_1 & m_2 \end{pmatrix} \\ &= (-1)^{j_1+j_2+j_3} \begin{pmatrix} j_1 & j_3 & j_2 \\ m_1 & m_3 & m_2 \end{pmatrix} = (-1)^{j_1+j_2+j_3} \begin{pmatrix} j_2 & j_1 & j_3 \\ m_2 & m_1 & m_3 \end{pmatrix} \\ &= (-1)^{j_1+j_2+j_3} \begin{pmatrix} j_3 & j_2 & j_1 \\ m_3 & m_2 & m_1 \end{pmatrix} \end{aligned} \tag{3}$$

due to the permutation of columns and

$$\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} = (-1)^{j_1+j_2+j_3} \begin{pmatrix} j_1 & j_2 & j_3 \\ -m_1 & -m_2 & -m_3 \end{pmatrix}$$

due to a change of signs in the momentum projections. Sometimes these symmetry relations are called the *classical symmetries*. They result in 12 formally different 3–j symbols with the same absolute value. There are additional symmetries known due to Regge [13] which are most easily explained in terms of Regge symbols [7]. For these details, however, the reader is referred to the literature. A similar distinction between classical symmetries and additional symmetric forms due to Regge can be made for the 6–j symbols. Table 1 lists the number of classical symmetries vs. the overall number of symmetric forms known for the Wigner symbols. In order to simplify Racah algebra expressions, it is highly essential to exploit these symmetries: they are therefore all incorporated in the program as it is explained below. For practical purposes, however, the classical symmetries are much more important and, thus, the distinction is kept. In our notation, the classical symmetries are a subset of the Regge symmetries.

The symmetry relations (3) represent some simple examples, where additional (phase) factors naturally appear in dealing with Racah algebra. Such factors as well as internal summations over various quantum numbers often occur during the evaluation and simplification of an expression. A typical structure of a Racah algebra expression will be shown in the next section. Of course, this structure must also find a reflection by the internal representation in any CA program on this subject.

A successful simplification of Racah algebra expressions must exploit orthogonality relations and a variety of important sum rules. Indeed, to a large extent, the literature on Racah algebra is devoted to find and to prove such relations among the Wigner symbols or to compile them in some useful form. In many instances, the standard presentation of the different orthogonality and sum rules already gives a first impression how more complex expressions should be simplified. In the practical use of such rules, however, the standard form is often not of great help. The main difficulty is to recognize the equivalence of some part of a Racah algebra expression, for instance with one side of a given sum rule, so that, in fact, this special rule can be applied. The most comprehensive compilation of these rules as far as I know can be found in the book by Varshalovich et al. [7].

Table 1
Number of classical symmetries of Wigner n–j symbols vs. the symmetries known due to Regge [13].

n–j symbol	Classical symmetries	Regge symmetries [13]
3–j	12	72
6–j	24	144
9–j	72	-

$$\text{Racahexpr} := \sum_{j_1,j_2,\ldots} (-1)^{2j_1-j_2+\cdots} j_1^{3/2} [j_2] \ldots \begin{pmatrix} \cdot & \cdot & j_1 \\ \cdot & \cdot & \cdot \end{pmatrix} \begin{pmatrix} j_1 & j_2 & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix} \begin{Bmatrix} \cdot & j_3 & \cdot \\ j_1 & \cdot & \cdot \\ J & \cdot & j_2 \end{Bmatrix} \ldots$$

Fig. 1. Typical structure of a Racah expression, i.e. an expression which appears in Racah algebra.

3. Racah expressions and techniques for simplification

The main effort in dealing with Racah algebra techniques concerns the attempt to simplify complex expressions. To explain more details about the process of simplification, I will first expound the structure of a typical expression which is later called a *Racah expression* (or in many applications even shorter Racahexpr). A useful relation among the Wigner symbols is the algebraic equivalence between a 6–j symbol and a summation involving four 3–j symbols (see Rotenberg et al. [12], Eq. (2.18)),

$$\begin{Bmatrix} j_1 & j_2 & j_3 \\ l_1 & l_2 & l_3 \end{Bmatrix} = \sum_{m_1 m_2 m_3 n_1 n_2 n_3} (-1)^S \begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} \begin{pmatrix} j_1 & l_2 & l_3 \\ m_1 & n_2 & -n_3 \end{pmatrix} \times \begin{pmatrix} l_1 & j_2 & l_3 \\ -n_1 & m_2 & n_3 \end{pmatrix} \begin{pmatrix} l_1 & l_2 & j_3 \\ n_1 & -n_2 & m_3 \end{pmatrix} \tag{4}$$

$$= \sum_{m_1 m_2 n_1 n_2 n_3} (-1)^S [j_3] \begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} \begin{pmatrix} j_1 & l_2 & l_3 \\ m_1 & n_2 & -n_3 \end{pmatrix} \times \begin{pmatrix} l_1 & j_2 & l_3 \\ -n_1 & m_2 & n_3 \end{pmatrix} \begin{pmatrix} l_1 & l_2 & j_3 \\ n_1 & -n_2 & m_3 \end{pmatrix}, \tag{5}$$

with $S = l_1 + l_2 + l_3 + n_1 + n_2 + n_3$. The right-hand side of Eq. (5) already displays typical terms of a general expression in Racah algebra, i.e. some phase and weight factors, different 3–j symbols and a summation over various quantum numbers. The summation variables may also appear in the phase and/or weight factors. More generally, a *Racah expression* may include any number of Wigner n–j symbols of different kinds as well as Kronecker and triangular $\delta(j_1, j_2, j_3)$ symbols. We will apply the symbol $\delta(j_1, j_2, j_3)$ to represent $+1$ if the triangle relation is satisfied by the angular momenta j_1, j_2, and j_3 and to represent zero otherwise. The general structure of a Racah expression is symbolically shown in Fig. 1. To provide a quick reference to various parts of a Racah expression we will also use a few short-hand notations as explained in Appendix B; the terms which are most frequently used are an *wnj* to denote an arbitrary Wigner n–j symbol and furthermore an *w3j*, *w6j*, and *w9j* with an obvious meaning.

A structure more complex than a single Racah expression is a sum of different Racah expressions, which is sometimes called a *Racahsum* below. Such a sum often occurs due to recursion relations if they are applied to some Wigner n–j symbol which, of course, could also be part of a more complex Racah expression.

A numerical evaluation of a Racah expression can be carried out only if all quantum numbers in the n–j symbols apart from summation variables have *numerical* (integral or half-integral) values. Thus, a numerical evaluation of a Racah expression is not possible if the input angular momentum quantum numbers correspond to unspecified symbolic Maple variables. However, already a simple estimation shows that it is no longer feasible

to carry it out if more than a very few summation variables appear. Of course, the main strength of using Racah algebra techniques is that such expressions can often be simplified considerably by algebraic transformations. From the properties of Wigner symbols which have briefly been discussed in the previous section three different strategies for the simplification of Racah expressions can be derived:

(i) The application of known special values. Here it is assumed that a n–j symbol is replaced by a (much) simpler expression which, in particular, does not contain any explicit summation. Note that all n–j symbols of a given Racah expression can be analyzed and replaced independently by special-value rules.
(ii) Use of orthogonality properties.
(iii) Use of various sum rules.

In the present version, I focused the attention onto path (i) and on the numerical evaluation of Racah expressions. The main obstacle to apply the last two alternatives, (ii) and (iii), is that they require a careful analysis of the Racah expression as a whole. Since all orthogonality relations and sum rules include summations over formal quantum numbers, these summation variables not only have to be in the correct position in the Wigner n–j symbols, but they must also contribute to a correct phase and weight of the overall expression. Moreover, the same variables may not occur in other Wigner symbols of the Racah expression which are not part of the selected rule. These difficulties in the simplification process are further enhanced by the large number of equivalent forms of the total Racah expression due to the various symmetric forms of the Wigner n–j symbols. Therefore, in order to simplify a Racah expression by a given sum rule, in practice one has to start with a certain part of the expression and then try to identify equivalence with some relation by means of the various symmetries of the n–j symbols. Once the equivalence has been proven, this part of the Racah expression can be replaced by a corresponding simpler term. In this context, simplification of a Racah expression always means reducing the number of summation indices and/or the number of Wigner n–j symbols.

This rather cumbersome procedure of running through all symmetric forms of a Racah expression and of identifying algebraic equivalent parts makes the simplification of such expressions very suitable for CA applications. An efficient scheme to perform simplifications of Racah expressions by means of the steps (i)–(iii) is realized in the Racah program. The main part of this program serves to perform certain transformations and to replace algebraic equivalent structures. Attention has been paid to developing an interactive tool for rather a wide range of applications. However, before I want to present further details about the Racah program, the interactive use is shown for a few individual procedures. An overview to the Racah package will later be summarized in Section 5 and in Appendices B and C.

4. Interactive work using the Racah procedures

We now display the interactive dialog to perform some simple tasks. For the time being, the reader is not supposed to be familiar with a specific computer algebra system even though the syntax of Maple is used throughout. More advanced examples as well as the way to get access to all procedures at the beginning of a session will be described in the next sections.

A straightforward example is the computation of a Wigner n–j symbol with only numerical arguments, say $\begin{Bmatrix} 7/2 & 3 & 9/2 \\ 3/2 & 4 & 3/2 \end{Bmatrix}$. To find the value of this 6–j symbol, we have to enter two lines at Maple's prompt,

```
> w6ja := Racah_set(w6j,7/2,3,9/2,3/2,4,3/2):
> Racah_compute(w6ja);

    -.09258200999
```

or to obtain the algebraic value

```
> Racah_compute(w6ja,algebraic);
```

```
                                1/2   1/2   1/2    1/2
      - 1/269500 2310     70     35     110
```

The first line assigns the 6–j symbol to the variable `w6ja` but does not evaluate it immediately; this is done by the second command Racah_compute(). To compare the result we might look up some appropriate table which covers the six arguments above. With two similar lines we can obtain the numerical value of any n–j symbol ($n = 3, 6, 9$) with almost no restrictions on the size of the arguments.

The angular momenta in Wigner n–j symbols and general Racah expressions are certainly not always fixed numerical values. They are often just variables or some other expressions. This has been taken into account in the Racah program. The only restriction within the program is that all variables and expressions which appear like angular momentum quantum numbers represent either integer or half-integer values and fulfill proper coupling conditions. This restriction is necessary since the validity of some expression is often difficult to prove during the process of evaluation. A failure in the coupling conditions can lead to syntax errors in some later step.

Next, let us consider the Racah expression

$$\sum_{jm} (-1)^{j-1/2} [j] \begin{pmatrix} 3/2 & j & 2 \\ -1/2 & m & 1 \end{pmatrix} \begin{pmatrix} j & 2 & 3/2 \\ -m & -1 & 1/2 \end{pmatrix}, \tag{6}$$

where again only numerical arguments appear apart from summation variables. This is necessary in order to *compute* the full expression, i.e. to determine its numerical value. Here, all summations within the expression can be carried out explicitly. The whole expression is entered in 6 lines:

```
> w3ja := Racah_set(w3j,3/2,j,2,-1/2,m,1):
> w3jb := Racah_set(w3j,j,2,3/2,-m,-1,1/2):
> wexpra := Racah_set(Racahexpression,w3ja,w3jb):
> wexpra := Racah_add(factor,2*j+1,wexpra):
> wexpra := Racah_add(phase,j-1/2,wexpra):
> wexpra := Racah_add(sum,{j,m},wexpra):
```

We suppress the reply of Maple by a colon at the end of the input lines. A semicolon, instead, would demonstrate the list structure of all data types as they are shown in Appendix B. To obtain now the result of the numerical evaluation, we just type

```
> Racah_compute(wexpra);

                    .9999999998
```

The answer, 1.0, is not surprising since the Racah expression (6) is the r.h.s. of a well-known orthogonality relation where we have rewritten the 3–j symbols by means of various classical symmetries.

So far, we have seen features for the numerical computation of Racah expressions. In analytic work with angular momenta we might also need recursion relations and the simplification of Racah expressions. There are different recursions known for the 3–j symbol

$$\begin{pmatrix} j_a & j_b & j_c \\ m_a & m_b & m_c \end{pmatrix} \tag{7}$$

in terms of two or more 3–j symbols with different arguments. For example, two of the j-quantum numbers may be decreased simultaneously by 1/2 if we type

```
> w3jc := Racah_set(w3j,ja,jb,jc,ma,mb,mc):
> wexprb := Racah_recursionforw3j(halfstep,w3jc):
> Racah_print(wexprb):
```

```
--->
                                        1/2
                 ((jb + mb) (jc - mc))
             ---------------------------------------
                                                   1/2
             ((ja + jb + jc + 1) (- ja + jb + jc))

             w3j(ja,jb-1/2,jc-1/2,ma,mb-1/2,mc+1/2)

-------------------------------------------------------------------------  1
                           -1
                                        1/2
                 ((jb - mb) (jc + mc))
             ---------------------------------------
                                                   1/2
             ((ja + jb + jc + 1) (- ja + jb + jc))

             w3j(ja,jb-1/2,jc-1/2,ma,mb+1/2,mc-1/2)

-------------------------------------------------------------------------  2
```

that is

$$\begin{pmatrix} j_a & j_b & j_c \\ m_a & m_b & m_c \end{pmatrix} = \left[\frac{(j_b+m_b)(j_c-m_c)}{(j_a+j_b+j_c+1)(j_b+j_c-j_a)}\right]^{1/2} \begin{pmatrix} j_a & j_b-1/2 & j_c-1/2 \\ m_a & m_b-1/2 & m_c+1/2 \end{pmatrix}$$
$$- \left[\frac{(j_b-m_b)(j_c+m_c)}{(j_a+j_b+j_c+1)(j_b+j_c-j_a)}\right]^{1/2} \begin{pmatrix} j_a & j_b-1/2 & j_c-1/2 \\ m_a & m_b+1/2 & m_c-1/2 \end{pmatrix}.$$

In order to apply another recursion relation instead, it would be enough to modify the keyword *halfstep* into of the keywords *intstep, Louck* or *magnetic*. The different recursion relations for the 3-j symbols which are implemented in the present program are listed in Appendix A. This appendix also includes the explicit formulas used for the numerical evaluation of the Wigner n-j symbols. There are currently four recursions known by the package which can also be accessed by using integer flags instead of the given keywords.

5. The Racah package

The Racah package facilitates the evaluation and simplification of expressions which appear in the application of Racah algebra techniques. It is designed as an interactive tool in the framework of Maple V and might simply be considered as an extension of this computer algebra system to the theory of angular momentum. Due to its interactive character, the Racah package will be helpful both for occasional applications as well as for more advanced work dealing with angular momenta in quantum mechanics.

Let us, however, begin with a few general remarks. In this section I assume that the reader has some practice with computer algebra and with procedural languages. In a programming language as it is provided in Maple, a set of procedures form a hierarchy. Each procedure can be used as a *selfcontained* command in interactive work as well as a basic element to build up procedures at some higher level of the hierarchy. The input and output data are handled as logical objects which might have an in principle arbitrary complex data structure. Thus, this concept allows one to define data structures appropriate to some special field of application. In the coupling of angular momenta such structures are, for instance, Wigner's n-j symbols, Clebsch-Gordan coefficients, or,

more generally, Racah expressions as introduced in Section 3. Within a CA environment, the formulation of *typical* structures leads to the *internal representation* of this data type. For example, a general Racah expression is here of type Racahexpr and may contain various other data types like the overall phase, weight factors, etc. I explain all data structures which are used in this package in Appendix B[6]. The choice and definition of appropriate structures is crucial for the successful work with CA systems.

Another important feature for interactive work is the *concept of generic names* in CA programs. Many commands are able to recognize the type of the input data and to perform an appropriate action in correspondence to this type. I.e., they often invoke subprocedures which may remain *hidden* at the user level. This concept simplifies the work considerably, because the user has to know much less about individual commands and their syntax. This feature is here also supported, i.e. only the higher-level procedures need to be described below.

In the current version the Racah package contains a total of about 60 procedures at different levels. Due to the usage of generic names, however, only 16 commands have to be explained in detail. The most important ones are summarized with a brief description in Table 2. More detailed information about these procedures which can all be used interactively is shown in Appendix C. The presentation therein follows the style of The Maple Handbook by Redfern [14] which, in particular, seems to me suitable for quick reference. Appendix C does not teach the theoretical background of the individual procedures but provides a reference to all commands in alphabetic order. For each command I describe the type of input and output data and, if available, the form of optional arguments. Additionally, further necessary information and cross reference to related commands is also given.

Most inherent commands in Maple V have rather short names. This certainly facilitates the frequent use of these functions. By contrast, the choice of long names keeps the commands selfexplained, and one also needs not refer so often to the manual in order to find correct *abbreviations*. The Racah package therefore uses rather long names. Moreover, all commands begin with the prefix Racah_ to make a clear distinction from inherent functions of Maple V.

Even though the Racah package is a straightforward extension of Maple's V functionality to Racah algebra, it is not intended to support all features of this CA system. This concerns (i) the access of the commands by means of `with()` or the standard long form and (ii) a full on-line help. A list of all current commands is, however, available from Racah_help().

To access the commands of the Racah package the user has to read in the file `racah1` by

```
> read racah1;
```

at the beginning of each session with which all procedures are loaded at the same time. This takes about 80 000 words of memory; the procedures, however, do not define any global variables which would have to be taken into account in interactive work.

We will now conclude the description of the Racah package with a few more advanced examples and, finally, with an outlook to some forthcoming application in atomic many-body theory.

6. Test cases

This section describes four independent tests of the package which will also provide a deeper insight into how different commands can be use interactively.

Let us again start with the numerical evaluation of Wigner n–j symbols. A known relation for 9–j symbols with a single zero and a few repeated parameters is [12]

[6] The knowledge of these structures is the minimum requirement for the design of own programs using the Racah package.

Table 2
Important commands of the Racah package. A more detailed description of these commands and of other procedures which are less frequently used is given in Appendix C. The internal representations of the various data types for input and output are explained in Section 3 and Appendix B.

Racah_add()	Adds some n–j symbols, phase contributions, weight factors, or summation variables to a Racah expression.
Racah_compute()	Computes the numerical value of some Racah expression by carrying out all summations explicitly. Here, it is supposed that all arguments apart from summation variables have to be integer or half-integer constants and have to fulfill the conditions of angular coupling. The expression can be either of type wnj or Racahexpr.
Racah_delete()	Removes explicitly various terms from a Racah expression.
Racah_evaluate()[a]	Attempts to simplify a general Racah expression by using a list of special values, orthogonality properties and a variety of sum rules which are known for the Wigner n–j symbols. Simplification *means* the reduction of the number of summation variables *and/or* the number of n–j symbols in the Racah expression.
Racah_help()	Displays a list of all commands implemented in the current version. A full on-line help for valid keywords and the syntax of individual commands is not given.
Racah_print()	Prints a Racah expression or some part of it in a neat format. No pretty print.
Racah_recursionforw3j()	Applies a specified recursion relation to a Wigner 3–j symbol.
Racah_set()	Enters some expression in Racah algebra into the internal representation. Valid expression are Wigner n–j symbols, Clebsch–Gordan coefficients, or, more general, Racah expressions.

[a] The full support of this command is postponed to forthcoming work; the present version only exploits special values and some orthogonality properties for the n–j symbols in a Racah expression.

$$\begin{Bmatrix} a & b & e \\ c & d & e \\ f & f & 0 \end{Bmatrix} = \frac{(-1)^{b+c+e+f}}{\left[(2e+1)(2f+1)\right]^{1/2}} \begin{Bmatrix} a & b & e \\ d & c & f \end{Bmatrix}, \tag{8}$$

where, in our notation, the left-hand side is of type w9j, i.e. a single 9–j symbol, and the r.h.s. is a simple Racah expression. We calculate both sides of Eq. (8) independently for some numerical parameters, say $a = 3/2, b = 7/2, c = 2, d = 3, e = 4, f = 3/2$.

```
> w9ja := Racah_set(w9j,3/2,7/2,4,2,3,4,3/2,3/2,0):
> Racah_compute(w9ja);
```

-.008132500612

The r.h.s. of Eq. (8) can be evaluated in different ways. We consider this term as a Racah expression and type

```
> w6jb := Racah_set(w6j,3/2,7/2,4,3,2,3/2):
> wexprc := Racah_set(Racahexpression,w6jb):
> wexprc := Racah_add(phase,3/2+2+4+3/2,wexprc):
> wexprc := Racah_add(factor,1/sqrt((2*4+1)*(2*3/2+1)),wexprc):
```

in order to enter the whole expression. The numerical evaluation of wexprc confirms

```
> Racah_compute(wexprc);
```

-.008132500610

the equivalence of relation (8).

Next, Clebsch–Gordan coefficients are computed explicitly in the limit that one of the angular momenta involved is small compared with the other two j-values. In this case, these coefficients can be approximately expressed by matrix elements of a finite rotation (cf. Appendix 2 of Ref. [4]). A simple example is

Table 3
Clebsch–Gordan coefficients (9) for various pairs of quantum numbers (J, M). See text for explanation.

J	M	$\langle 11JM \mid J+1\, M+1\rangle$	$\left[\frac{J(J+1)+2MJ+M(M+1)}{(2J+1)(2J+2)}\right]^{1/2}$
10	1	0.581087	0.534522
10	10	1.0	0.953463
100	1	0.508680	0.503718
100	10	0.553346	0.548383
100	100	1.0	0.995037
1000	1	0.500874	0.500375
1000	10	0.505371	0.504871
1000	100	0.550337	0.549837
1000	1000	1.0	0.999500

$$\langle 11JM \mid J+1\, M+1\rangle \approx \left[\frac{J(J+1)+2MJ+M(M+1)}{(2J+1)(2J+2)}\right]^{1/2}. \tag{9}$$

Table 3 shows the Clebsch–Gordan coefficients for a few selected pairs (J, M) of quantum numbers[7] and compares the results with the values from the r.h.s. of Eq. (9). The explicit calculation with Racah_compute() remains rather fast even for values with $J = 1000$.

Another example concerns the numerical computation of more complex expressions. The command Racah_compute() also attempts to evaluate all Racah expressions as long as they are equivalent to some numerical value. For instance, we consider a sum rule which is satisfied by combinations of the 3–j and 6–j symbols [12]

$$\sum_{m_3} \begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} \begin{pmatrix} l_1 & l_2 & j_3 \\ n_1 & n_2 & -m_3 \end{pmatrix} = \sum_{l_3 m_3} (-1)^{j_3+l_3+m_3+n_3} [l_3] \begin{Bmatrix} j_1 & j_2 & j_3 \\ l_1 & l_2 & l_3 \end{Bmatrix} \times \begin{pmatrix} l_1 & j_2 & l_3 \\ n_1 & m_2 & n_3 \end{pmatrix} \begin{pmatrix} j_1 & l_2 & l_3 \\ m_1 & n_2 & -n_3 \end{pmatrix} \tag{10}$$

for the special case $j_1 = 2, j_2 = 3, j_3 = 2, l_1 = 3, l_2 = 4, m_1 = -1, m_2 = 1, n_1 = 1, n_2 = -1$. Again, we compute both sides of this relation in turn

```
> w3jd := Racah_set(w3j,2,3,2,-1,1,m3):
> w3je := Racah_set(w3j,3,4,2,1,-1,-m3):
> wleft := Racah_set(Racahexpression,w3jd,w3je):
> wleft := Racah_add(sum,{m3},wleft):
> Racah_compute(wleft);
```

```
.01844277784
```

With analogous steps we find for the r.h.s.

```
> w6jc := Racah_set(w3j,2,3,2,3,4,l3):
> w3jf := Racah_set(w3j,3,3,l3,1,1,n3):
> w3jg := Racah_set(w3j,2,4,l3,-1,-1,-n3):
> wright := Racah_set(Racahexpression,w6jc,w3jf,w3jg):
> wright := Racah_add(factor,2*l3+1,wright):
> wright := Racah_add(phase,2+l3-1+1,wright):
```

[7] To enter a Clebsch–Gordan coefficient use the command Racah_set().

```
> wright := Racah_add(sum,{13,n3},wright):
> Racah_compute(wright);
```

.01844277785

The sum rule (10) is thus directly proven for the given set of quantum numbers.

A final test searches for special values of one or more Wigner n–j symbols within a Racah expression. At the first glance, the expression

$$\sum_J (-1)^{-2J+M-b-c}\,[J+1, J+1/2, J, J-1/2]^{1/2} \begin{Bmatrix} c-2 & 2 & c \\ b & J & b-2 \end{Bmatrix} \times \begin{pmatrix} J+1/2 & 3/2 & J \\ -M & -1/2 & M+1/2 \end{pmatrix} \begin{Bmatrix} d & e & c \\ 1/2 & 0 & 1/2 \\ J & e & b \end{Bmatrix} \tag{11}$$

looks rather lengthy and tedious to evaluate. - Could this expression be simplified by the knowledge of special values for the Wigner n–j symbols ? To study this question we assign the full expression (11) to the variable wexprd as we have done it before. An attempt to simplify expression (11) is done by typing just

```
>wexpre := Racah_evaluate(wexprd);
>Racah_print(wexpre);

--->
                                    SUM{J}

                                      (3 J + 2 c)
                                  (-1)

                                                                    1/2
(J + 3 M + 3/2)(4 J - 4 M + 2)  ((J + b + c - 2)(J + b + c - 1)(J + b + c)

     (J + b + c + 1) (- J + b + c - 3) (- J + b + c - 2) (- J + b + c - 1)
                                                    /
    (- J + b + c))^1/2  /  (((2 b - 3) (2 b - 2) (2 b - 1) b (2 b + 1)
                       /
                                                                        1/2
    (2 c - 3) (2 c - 2) (2 c - 1) c (2 c + 1))^1/2 (4 e + 2)    )

                                  w6j(d,c,e,b,J,1/2)
```

Thus, the result of the evaluation is

$$\text{expression}(11) = \sum_J (-1)^{3J+2c}(J+3M+3/2)\left[\frac{(4J-4M+2)(J+b+c-2)(J+b+c-1)}{(2b-3)(2b-2)(2b-1)b(2b+1)} \times \frac{(J+b+c)(J+b+c+1)(-J+b+c-3)(-J+b+c-2)(-J+b+c-1)(-J+b+c)}{(2c-3)(2c-2)(2c-1)c(2c+1)(4e+2)}\right]^{1/2} \times \begin{Bmatrix} d & c & e \\ b & J & 1/2 \end{Bmatrix}.$$

By contrast, if the command fails to simplify the expression a NULL list [ ] is returned.

7. Applications in atomic many-body theory

Atomic structure is one important field which extensively exploits the techniques of Racah algebra. Most theoretical support in the interpretation of open-shell spectra is nowadays based on such techniques. So far, the development of powerful computers has allowed for fast number crunching of formulas which were derived before and coded in some computer program. With a deeper insight in the electronic structure of atoms and ions, however, models will become more sophisticated. Therefore, computers might also be used to work out details of some model in the future. Computer algebra is the modern and powerful attempt to perform these derivations. It results in faster solutions and is often much more reliable.

The relativistic many-body perturbation theory (MBPT) is able to account for both relativistic and correlation effects simultaneously. The theory was developed by several groups during the last decade using either *local* or *global* basis functions [15,16]. MBPT also provides a systematic approach to the calculation of a large variety of properties which are expressed in terms of perturbation expansions. The derivation of these expansions can become very laborious and is rather complicated for all atoms with a nontrivial open-shell structure. The complexity of the expressions which occur during the derivation are the main reason that only *effective* one- and two-particle systems (with no more than two electrons *and/or* holes outside closed shells) have so far been studied accurately within this framework. In order to extent the domain of many-body perturbation techniques towards more complex atoms and ions one needs support by computer algebra.

Most practical schemes in atomic MBPT start from the generalized Bloch equation in the Rayleigh-Schrödinger formulation [10],

$$[\Omega, H_0]\, P = V\Omega P - \Omega P V \Omega P \,, \tag{12}$$

where H_0 is the zero-order part of the Hamiltonian, $H = H_0 + V = H_0 + V_1 + V_2 + \cdots$, and Ω is the wave operator for the states of interest. The operator V represents one or more perturbations beyond H_0, for instance the instantaneous Coulomb repulsion, $\sum_{i<j} 1/r_{ij}$.

The Bloch equation (12) is naturally expressed in second quantization. In this algebraic form, the derivation of proper perturbation expansions leads to the evaluation of products of primitive operators like $H_0, V_1, V_2, \ldots$. In current treatments, these operators and all products are represented by Feynman-Goldstone diagrams as it is, for example, developed and explained in the textbook by Lindgren and Morrison [10]. Again, however, the derivation of the corresponding diagrammatic expansions becomes lengthy for open-shell atoms. Even for some fixed order of perturbation theory, say second order, the number of diagrams increases rapidly if the atom includes two or more valence-particles *and/or* valence-holes with respect to a closed-shell reference state or if more than one perturbation has to be included. Furthermore, several open-shells in some atom usually also increases the dimension of the model space which has to be taken into account.

Symbolic computations provide a useful alternative to derive the diagrammatic expansions in a given approximation scheme; it results, of course, in algebraic expressions which must later be computed numerically. A key to the successful derivation of perturbation expansions is the ability to simplify operator products in second quantization like

$$a_{i_1}^{+} a_{i_2}^{+} a_{i_3} a_{i_4} \cdots \frac{(-1)^k}{D} \langle \text{amplitude}_1 \rangle \langle \text{amplitude}_2 \rangle \cdots \,, \tag{13}$$

where the integer k characterizes the phase of the operator expression, and D is a typical *energy denominator* which appears in MBPT. The amplitudes $\langle \text{amplitude}_i \rangle$ define any one- or two-particle matrix elements, for example of the Coulomb repulsion, $\langle ij | 1/r_{12} | kl \rangle$.

From my point of view, the application of computer algebra tools will become essential for the study of complex atomic spectra. I will conclude this description here with a brief outlook upon a project which has recently been started.

8. Further developments and outlook

To treat operator products like expression (13), a Maple program has been designed which is based on Wick's theorem and the anticommutation relations of the (fermion) creation and annihilation operators. The setup includes the rules of second quantization and is also written for interactive use. There is neither a formal restriction on the number of creation and annihilation operators nor on the number of amplitudes. This program also allows for to include one ore more single perturbations. A current test version enables to solve the Bloch equation for closed-shell and effective one-particle atoms up to some given order n. This program for deriving Feynman–Goldstone perturbation expansions will be presented elsewhere. One difficulty in the evaluation of operator products and the projection onto the model space, however, is to recognize equivalent parts in different expressions, a very well-known problem in CA programs.

The direct combination of the Racah package with this work on atomic perturbation theory needs the implementation of basic formulas for spherical tensors. A rather general design for this implementation would enable steps to derive automatically all Feynman–Goldstone diagrams in their final *radial–angular* representation and to perform numerical computations. Moreover, the angular parts of these diagrams can then further be reduced by means of the given Racah package.

Acknowledgements

I wish to thank the referee for helpful comments on the manuscript. Parts of this work has been supported by a fellowship from the Swedish Natural Science Research Council (NFR).

Appendix A. Applied mathematical relations

Here, we summarize the most important mathematical relations which in the present version of the program are applied for the computation and manipulation of Racah expressions.

A.1. Computation of Wigner n–j symbols

Let us first introduce the Δ symbol by

$$\Delta(a,b,c) = \left[\frac{(a+b-c)!(a-b+c)!(-a+b+c)!}{(a+b+c+1)!}\right]^{1/2} . \tag{A.1}$$

Then, the numerical value of a Wigner 3–j symbol is calculated by the expression [1]

$$\begin{aligned}\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} &= \delta_{m_1+m_2+m_3,0}(-1)^{j_1-j_2-m_3}\Delta(j_1,j_2,j_3) \\ &\times \left[(j_1-m_1)!(j_1+m_1)!(j_2-m_2)!(j_2+m_2)!(j_3-m_3)!(j_3+m_3)!\right]^{1/2} \\ &\times \sum_l \left[\frac{(-1)^l}{l!(j_1+j_2-j_3-l)!(j_1-m_1-l)!(j_2+m_2-l)!}\right. \\ &\times \left.\frac{1}{(j_3-j_2+m_1+l)!(j_3-j_1-m_2+l)!}\right] . \end{aligned} \tag{A.2}$$

This expression has a nonzero value only if the arguments of all factorials are nonnegative integers.

The computation of a Wigner 6–j symbol follows internally the expression [4]

$$\begin{Bmatrix} j_1 & j_2 & j_3 \\ l_1 & l_2 & l_3 \end{Bmatrix} = \Delta(j_1,j_2,j_3)\Delta(j_1,l_2,l_3)\Delta(l_1,j_2,l_3)\Delta(l_1,l_2,j_3)$$
$$\times \sum_l \left[\frac{(-1)^l (l+1)!}{(l-j_1-j_2-j_3)!(l-j_1-l_2-l_3)!(l-l_1-j_2-l_3)!(l-l_1-l_2-j_3)!} \right.$$
$$\left. \times \frac{1}{(j_1+j_2+l_1+l_2-l)!(j_2+j_3+l_2+l_3-l)!(j_3+j_1+l_3+l_1-l)!} \right] . \qquad (A.3)$$

Finally, the 9–j symbol with numeric arguments is calculated by the known sum rule over three Wigner 6–j symbols

$$\begin{Bmatrix} j_{11} & j_{12} & j_{13} \\ j_{21} & j_{22} & j_{23} \\ j_{31} & j_{32} & j_{33} \end{Bmatrix} = \sum_j (-1)^{2j} \begin{Bmatrix} j_{11} & j_{21} & j_{31} \\ j_{32} & j_{33} & j \end{Bmatrix} \begin{Bmatrix} j_{12} & j_{22} & j_{32} \\ j_{21} & j & j_{23} \end{Bmatrix} \begin{Bmatrix} j_{13} & j_{23} & j_{33} \\ j & j_{11} & j_{12} \end{Bmatrix} . \qquad (A.4)$$

From the properties of the 6–j symbols one can easily see that the 9–j symbol is zero unless the arguments in each raw and column satisfy the triangular relation.

A.2. Recursion relations for Wigner 3–j symbols

There are four recursion formulas which are currently known to the Racah program (cf. Ref. [12], Eqs. (1.45)–(1.48)). They can be accessed, for instance, by a keyword as displayed in *italic* mode below (see also the description of the procedure Racah_recursionforw3j() in Appendix C). To simplify the notation the abbreviation $J = j_1 + j_2 + j_3$ is used in this subsection.

One *halfstep* recursion [4] allows to decrease two j-values by $1/2$

$$\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} = \left[\frac{(j_2+m_2)(j_3-m_3)}{(J+1)(J-2j_1)} \right]^{1/2} \begin{pmatrix} j_1 & j_2-1/2 & j_3-1/2 \\ m_1 & m_2-1/2 & m_3+1/2 \end{pmatrix}$$
$$- \left[\frac{(j_2-m_2)(j_3+m_3)}{(J+1)(J-2j_1)} \right]^{1/2} \begin{pmatrix} j_1 & j_2-1/2 & j_3-1/2 \\ m_1 & m_2+1/2 & m_3-1/2 \end{pmatrix} . \qquad (A.5)$$

If the magnetic quantum numbers explicitly fulfill the condition $m_1 = -m_2 - m_3$, the $1/2$-step recursion due to *Louck* [17] can be applied,

$$[j_3] \begin{pmatrix} j_1 & j_2 & j_3 \\ -m_2-m_3 & m_2 & m_3 \end{pmatrix}$$
$$= - \left[\frac{(J-2j_1)(J+1)(j_3+m_3)}{(j_2-m_2)} \right]^{1/2} \begin{pmatrix} j_1 & j_2-1/2 & j_3-1/2 \\ -m_2-m_3 & m_2+1/2 & m_3-1/2 \end{pmatrix}$$
$$- \left[\frac{(J-2j_3)(J-2j_2+1)(j_3-m_3+1)}{(j_2-m_2)} \right]^{1/2} \begin{pmatrix} j_1 & j_2-1/2 & j_3+1/2 \\ -m_2-m_3 & m_2+1/2 & m_3-1/2 \end{pmatrix} . \qquad (A.6)$$

An integral decrease of one j-value follows from the recursion with the keyword *intstep* [4]

$$\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} = \left[\frac{(j_2-m_2)(j_2+m_2+1)(j_3+m_3)(j_3+m_3-1)}{(J+1)(J-2j_1)(J-2j_2)(J-2j_3+1)} \right]^{1/2} \begin{pmatrix} j_1 & j_2 & j_3-1 \\ m_1 & m_2+1 & m_3-1 \end{pmatrix}$$
$$-2m_2 \left[\frac{(j_3+m_3)(j_3-m_3)}{(J+1)(J-2j_1)(J-2j_2)(J-2j_3+1)} \right]^{1/2} \begin{pmatrix} j_1 & j_2 & j_3-1 \\ m_1 & m_2 & m_3 \end{pmatrix}$$

$$-\left[\frac{(j_2+m_2)(j_2-m_2+1)(j_3-m_3)(j_3-m_3-1)}{(J+1)(J-2j_1)(J-2j_2)(J-2j_3+1)}\right]^{1/2}\begin{pmatrix} j_1 & j_2 & j_3-1 \\ m_1 & m_2-1 & m_3+1 \end{pmatrix}. \tag{A.7}$$

A further recursion relation among *magnetic* quantum numbers is [12]

$$\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} = -\left[\frac{j_1(j_1+1)-m_1(m_1+1)}{j_3(j_3+1)-m_3(m_3-1)}\right]^{1/2}\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1+1 & m_2 & m_3-1 \end{pmatrix}$$
$$-\left[\frac{j_2(j_2+1)-m_2(m_2+1)}{j_3(j_3+1)-m_3(m_3-1)}\right]^{1/2}\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2+1 & m_3-1 \end{pmatrix}. \tag{A.8}$$

A.3. Special values for Wigner n–j symbols

The special values of the 3–j and 6–j symbols, which are implemented in the Racah package, are basically taken from Edmonds tabulation (Tables 2 and 5 in Appendix 2 of Ref. [4]). There are 20 different special-value rules for 3–j symbols and 19 rules for 6–j symbols. A more comprehensive tabulation for special Clebsch–Gordan coefficients contains the monograph of Varshalovich and co-workers [7] but is not incorporated in the present version.

Appendix B. Internal data structures of Racah expressions

A concise description of the Racah package is simplified by introducing a few short-hand notations to refer to typical structures of Racah algebra expressions. These short-hand notations have been used in this paper and in many in-line comments in the source code of the Maple procedures; they are listed below in alphabetic order. Each notation also corresponds to a specific internal list data structure of Maple with a proper keyword as first operand.

Racahexpr A general sum of products of Wigner n–j symbols (→ Rproduct), weight factors (→ Rfactor), and a phase (→ Rphase) including any summation over internal quantum numbers (→ Rsummationset). In general, a Racahexpr can contain any number of Wigner 3–j, 6–j, and/or 9–j symbols, Kronecker and triangular $\delta(j_1, j_2, j_3)$ deltas, typical weights like $(2j+1)^{n/2}$ (n being an integer), and an arbitrary phase representing a *real* quantity. A Racahexpr is internally a Maple list

Racahexpr := [*Racahexpr*♯, Rsummationset, Rphase, Rfactor, Rdelta, Rproduct] ,

i.e. a list of list structures.

Racahsum A sum of Racah expressions (→ Racahexpr) which are put together internally in a common list structure. For instance, such a sum appears due to recursion relations which might be applied to a Wigner 3–j symbol. The sign of each Racahexpr in a Racahsum is determined due to the phase factor (→ Rphase). A Racahsum has the internal representation

Racahsum := [*Racahsum*♯, Racahexpr$_1$, Racahexpr$_2$, ...] .

Rdelta A product of any number of Kronecker $\delta_{j_1 j_2}$ and/or triangular $\delta(j_1, j_2, j_3)$ delta which appear due to the evaluation and simplification of a general Racah expression. Internally, a Rdelta is a list of (→) tdelta lists,

Rdelta := [*Rdelta*♯, tdelta$_1$, ..., tdelta$_n$] .

Rfactor An algebraic or numerical weight factor in a Racah expression (→ Racahexpr). Typically, such a weight factorizes into a rational factor and typical powers of some angular momenta like $j_i^{n_i}, j_k^{n_k/2}, \ldots, (2j_i+1)^{n_i/2}, \ldots$

with all n being integers. The specification of Rfactor is, however, not restricted to this form, i.e. other expressions may appear. In Maple terminology, Rfactor must be only of type *algebraic*.

Rphase A sum of integer and half-integer constants, variables, and/or expressions which characterize the sign of a $(\rightarrow)$ Racahexpr by $(-1)^{\text{Rphase}}$. Rphase is supposed to represent an integer expression so that the total phase remains always *real*.

Rproduct A product of any number of Wigner n–j symbols $(\rightarrow)$ wnj; the sequence of these n–j symbols has no meaning. An wnj can represent either a Wigner 3–j, 6–j, or 9–j symbol. The simplification of Racah expressions including 9–j symbols will not fully be supported by using orthogonality and sum rules known in the present version. A Rproduct has the internal Maple representation

Rproduct := [*wnj*♯, $\text{wnj}_1, \ldots, \text{wnj}_n$] .

Rsummationset The internal representation of the $\sum$, the summation indices, and all ranges for summation. In the Racah package, it is represented by a set of (integer or half-integer valued) variables and/or index-range equations, i.e. the sequence of the entries is arbitrary. Note that no constant value may be assigned to a summation variable.

Rsummationset := { $\text{var}_1, \ldots, \text{var}_n$ } .

tdelta A short-hand notation to describe either a Kronecker delta δ_{j_1,j_2} or triangular delta $\delta(j_1, j_2, j_3)$ which depend on two or three angular momentum quantum numbers, respectively. Both δ factors are internally represented by a list where a keyword as first operand characterizes the individual meaning, i.e.

tdelta_1 := [*delta*♯, j_1, j_2] or
tdelta_2 := [*triangle*♯, j_1, j_2, j_3] .

wnj Short-hand notation for either a Wigner 3–j, 6–j, or 9–j symbol.

w3j Short-hand notation for a Wigner 3–j symbol $\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix}$. The internal list representation is

w3j := [*w3j*♯, j_1, j_2, j_3, m_1, m_2, m_3] .

w6j Short-hand notation for a Wigner 6–j symbol $\begin{Bmatrix} j_1 & j_2 & j_3 \\ j_4 & j_5 & j_6 \end{Bmatrix}$. The internal list representation is

w6j := [*w6j*♯, j_1, j_2, j_3, j_4, j_5, j_6] .

w9j Short-hand notation for a Wigner 9–j symbol $\begin{Bmatrix} j_1 & j_2 & j_3 \\ j_4 & j_5 & j_6 \\ j_7 & j_8 & j_9 \end{Bmatrix}$. The internal list representation is

w9j := [*w9j*♯, j_1, j_2, j_3, j_4, j_5, j_6, j_7, j_8, j_9] .

Appendix C. Command listing to the Racah package

This appendix serves users of the Racah package for quick reference. It provides a list of all commands which are necessary for the interactive work at the user level. Auxiliary procedures which are invoked at some hidden level are not explained here. The whole package contains a total of about 60 subprograms.

In this appendix, the presentation follows the style of The Maple Handbook by Redfern [14] which provides a complete reference to Maple V in the form of logical subsets. Within each subset the individual commands are ordered alphabetically. The Racah package forms an additional subset to manipulate expressions from Racah algebra. For each command it is briefly described how it works, the structure of the input and output data, and, if available, optional arguments. Reference to all internal data structures can be found in Appendix B.

Two further sections for each command provide some additional information and cross references to related commands within the Racah package.

According to the restrictions we made in this paper with respect to the full simplification of Racah expressions, we list below only procedures which are currently available.

Racah_add(*njsymbol*,wnj$_1$,wnj$_2$, . . .,wnj$_n$,Racahexpr)
Appends one or more Wigner n–j symbols to Racahexpr.

Output:A Racahexpr is returned.

Argument options:(*delta*,tdelta$_1$,. . .,tdelta$_n$,Racahexpr) adds one or more tdelta factors to Racahexpr. ♣(*factor*,Rfactor,Racahexpr) multiplies Rfactor to the current internal weight of Racahexpr. ♣(*phase*,Rphase,Racahexpr) adds Rphase to the internal phase of Racahexpr. ♣(*sum*,Rsummationset,Racahexpr) adds the variables and ranges of Rsummationset (which has to be a set structure) to the internal set of summation variables. ♣(*Racahexpression*,Racahexpr$_1$,Racahexpr$_2$, . . .,Racahexpr$_n$) multiplies Racahexpr$_1$ × Racahexpr$_2$ [× . . . × Racahexpr$_n$] together.

Additional information: The first argument always denote a *keyword* which specifies the type of the other arguments and how the procedure works. ♣A notation like wnj$_1$,wnj$_2$,. . .,wnj$_n$ indicates that the command can be used with any number of these quantities. The last argument, however, must be of type Racahexpr. ♣If the *keyword* is *sum*, the summation variables are supposed to be nonconstant as well as uniquely defined in the full Racahexpr due to the set structure of Rsummation. ♣The *multiplication* of two or more Racahexpr often appears during the simplification of a more complex Racahexpr. This applies, for instance, if some part of a Racahexpr (including any internal summation) can be replaced by a simpler expression.

See also: Racah_delete().

Racah_compute(wexpr)
Computes the numerical value of wexpr by performing an explicit summation and numerical evaluation of all included Wigner n–j symbols. wexpr can be either of type wnj or Racahexpr.

Output:A (floating-point) number is returned if the calculation is feasible.

Argument options:(wexpr,*algebraic*) attempts to perform the same computation algebraically correct without numerical floating point evaluation.

Additional information: All arguments of wexpr (apart from summation variables in a Racah expression) have to evaluate to integer or half-integer constants and have to fulfill the conditions of angular coupling. ♣An algebraically exact calculation can only be carried out for single Wigner n–j symbols. ♣Explicit formulas for the internal computation of Wigner n–j symbols are shown in Appendix A.1.

See also: Racah_set(), Racah_evaluate().

Racah_delete(*njsymbol*,wnj$_1$,wnj$_2$, . . .,wnj$_n$,Racahexpr)
Removes one or more Wigner n–j symbols from Racahexpr without any replacement.

Output:A Racahexpr is returned.

Argument options:(*delta*,tdelta$_1$,. . .,tdelta$_n$,Racahexpr) removes one or more tdelta factors from Racahexpr. ♣(*factor*,Rfactor,Racahexpr) divides the internal weight of Racahexpr by Rfactor. ♣(*phase*,Rphase,Racahexpr) subtracts Rphase from the internal phase of Racahexpr. ♣(*sum*,Rsummationset,Racahexpr) removes the variables and index range equations of Rsummationset (which must be a set structure) from the internal set of summation variables of Racahexpr.

Additional information: The first argument always denotes a *keyword* which specifies the type of the other arguments and how the procedure works. ♣A notation like wnj$_1$,wnj$_2$,. . ., wnj$_n$ indicates that the command can

be used with any number of these quantities. The last argument, however, must be of type Racahexpr. ♣The procedure terminates with a proper ERROR message if one of the list structures to be deleted is not found in Racahexpr. ♣If the *keyword* is *sum*, the summation variables are supposed to be uniquely defined in the full Racahexpr in agreement with a formal mathematical view due to the set structure of Rsummation.

See also: Racah_add().

Racah_evaluate(wexpr)
Attempts to simplify wexpr which can be either of type wnj or Racahexpr.

Output:A Racahexpr is returned if the simplification was successful and a [NULL] list otherwise.

Additional information: Simplification of a Racahexpr always means the reduction of the number of summation indices and/or the number of Wigner n–j symbols. ♣The argument wexpr is converted into a Racahexpr no matter what the type of the argument is. ♣The procedure will attempt three steps of simplification by

(i) Applying a set of *special values* for the Wigner n–j symbols.
(ii) Using the orthogonality properties for sums of products of equivalent n–j symbols.
(iii) Using a variety of important sum rules and incomplete orthogonality relations which can be found in the literature.

Whereas special-value evaluation applies to individual n–j symbols, the methods in steps (ii) and (iii) require the analysis of the whole Racah expression including all summation indices, the dependence on variables in the remaining parts of Racahexpr as well as the overall phase. The current version only supports step (i) in full detail. ♣A list of important special values can be found in the Refs. [4,7]. ♣The simplification of complex Racahexpr can be both very time- and memory-consuming.

See also: Racah_add(), Racah_compute(), Racah_delete().

Racah_extractfactor(var,Rfactor)
Extracts the *product part* of Rfactor which purely depends on the variable var. This is usually some power of this variable or of the root $(2\ \mathrm{var} + 1)^{1/2}$.

Output: An expression is returned if some valid factor can be extracted and FAILs otherwise.

Additional information: The present version assumes that Rfactor factorizes into any (half-integer) powers of the variable $\mathrm{var}^{n/2}$ and/or the typical root $(2\ \mathrm{var} + 1)^{n/2}$ with n being an integer. ♣The procedure recognizes only a limited number of structures of Rfactor; it terminates with an appropriate ERROR message if more sophisticated structures appear during the evaluation.

See also: Racah_extractphase().

Racah_extractphase(var,Racahexpr)
Extracts the phase contribution of Racahexpr which purely depends on the variable var.

Output: An expression in var is returned if the phase depends on this variable and 0 otherwise.

Additional information: The procedure recognizes only a limited number of structures for the representation of the phase; it terminates with an appropriate ERROR message if more sophisticated structures appear during the evaluation.

See also: Racah_extractfactor().

Racah_isRacahexpr(wexpr)
Tests whether wexpr is of type Racahexpr.

Output:A Boolean value of either true or false is returned.

Additional information: To test other internal structures there are analog commands available: Racah_isdelta(), Racah_isRacahsum(), Racah_iswnj(), Racah_isw3j(), Racah_isw6j(), Racah_isw9j().

Racah_istriangle(j_1,j_2,j_3)
Tests whether j_1, j_2, and j_3 form a angular momentum coupling triangle $\delta(j_1, j_2, j_3)$.

Output:A Boolean value of either true or false is returned.

Additional information: The procedure checks the inequalities $j_1 + j_2 \geq j_3$ (and all cyclic permutations) as well as that $j_1 + j_2 + j_3$ is of type integer. ♣All arguments must be integer or half-integer constants. ♣The procedure terminates with an appropriate ERROR message if the arguments $2 * j_k$ are not of type *nonnegint*.

Racah_print(wexpr)
Prints wexpr in a fast neat format. wexpr can be either of type Racahsum, Racahexpr, or wnj. No pretty print.

Output:A NULL expression is returned.

Racah_recursionforw3j(nrule,w3j)
Applies a recursion relation to the Wigner 3–j symbol w3j. The type of recursion is specified by the *integer* or *keyword* nrule.

Output:A Racahsum which contains two or more Racahexpr is returned.

Additional information: The parameter nrule which specifies the type of recursion can be an integer $n = 1, \ldots, 4$ or one of the allowed keywords {*halfstep, Louck, intstep, magnetic*}. The given sequence of keywords corresponds to $n = 1, 2, 3$, and 4, respectively. ♣The applied recursion relations are based on Eqs. (A.5)–(A.8) in Appendix A.

Racah_replace(var,expr,Racahexpr)
Replaces variable var by expr at all occurrences in Racahexpr.

Output:A Racahexpr is returned.

Argument options:([$\mathrm{var}_1, \ldots, \mathrm{var}_n$],[$\mathrm{expr}_1, \ldots, \mathrm{expr}_n$],Racahexpr) to perform the same replacement for each pair (var_i,expr_i).

Additional information: The output has always Rsummationset = { NULL } .

Racah_set(*Racahexpression*,$\mathrm{wnj}_1, \ldots, \mathrm{wnj}_n$)
Enters one or more Wigner n–j symbols into the (internal Maple) representation of a Racahexpr.

Output:A Racahexpr is returned.

Argument options:() to set up the internal list representation of a Racahexpr without any arguments. ♣(*ClebschGordan*,j_1,m_1,j_2,m_2,j_3,m_3) enters a Clebsch–Gordan coefficient $\langle j_1 m_1, j_2 m_2 \mid j_3 m_3 \rangle$ into the internal Maple representation. ♣(*w3j*,j_1,j_2,j_3,m_1,m_2,m_3) enters a Wigner 3–j symbol. An expression of type w3j is returned. ♣(*w6j*,j_1,j_2,j_3,j_4,j_5,j_6) enters a Wigner 6–j symbol. An expression of type w6j is returned. ♣(*w9j*,j_1,j_2,j_3,j_4,j_5,j_6,j_7,j_8,j_9) enters a Wigner 9–j symbol. An expression of type w9j is returned.

Additional information:A notation like $\mathrm{wnj}_1, \ldots, \mathrm{wnj}_n$ indicates that there can be any number of these data structures in the parameter list ♣If *keyword* is *Racahexpression*, then, the output has always Rfactor = 1 and Rphase = 0. ♣For Clebsch–Gordan coefficients we use the phase convention (2) of Condon and Shortley [11] for the conversion into 3–j symbols. ♣All j_i and m_i must be integer or half-integer constants or expressions and have to fulfill the conditions of angular coupling.

See also: Racah_add(), Racah_compute(), Racah_delete().

Racah_simplifydeltas(Racahexpr)
Simplifies Racahexpr due to the internal list of δ factors.

Output:A Racahexpr or NULL is returned.

Additional information: The procedure also checks the validity of Racahexpr. It prints a short message if Racahexpr evaluates identically to zero and returns NULL in this case.

Racah_symmetricw3j(n,w3j)
Evaluates the *n*th *symmetric form* of a Wigner 3–*j* symbol.

Output:A Racahexpr is returned.

Argument options:(n,w3j,*Regge*) to access one of the 72 symmetric forms of the 3–*j* symbol due to Regge [13], i.e. *n* must be in the range $1 \leq n \leq 72$.

Additional information: There are 12 basic *symmetric forms* (3) for a Wigner 3–*j* symbol whose sequence is internally defined, i.e. the allowed range of *n* is $1 \leq n \leq 12$. ♣The symmetric form for $n = 1$ is identical to the argument w3j. ♣The output has always Rsummationset = { NULL } and Rfactor = 1.

See also: Racah_symmetricw6j(), Racah_symmetricw9j().

Racah_symmetricw6j(n,w6j)
Evaluates the *n*th *symmetric form* of a Wigner 6–*j* symbol.

Output:A Racahexpr is returned.

Argument options:(n,w6j,*Regge*) to access one of the 144 symmetric forms of the 6–*j* symbol due to Regge [13], i.e. *n* must be in the range $1 \leq n \leq 144$.

Additional information: There are 24 basic *symmetric forms* for a 6–*j* symbol whose sequence is internally defined, i.e. the allowed range of *n* is $1 \leq n \leq 24$. ♣The symmetric form for $n = 1$ is identical to the argument w6j. ♣The output has always Rsummationset = { NULL }, Rphase = 0, and Rfactor = 1.

See also: Racah_symmetricw3j(), Racah_symmetricw9j().

Racah_symmetricw9j(n,w9j)
Evaluates the *n*th *symmetric form* of a Wigner 9–*j* symbol.

Output:A Racahexpr is returned.

Additional information: There are 72 basic *symmetric forms* for a 9–*j* symbol whose sequence is internally defined, i.e. the allowed range of *n* is $1 \leq n \leq 72$. ♣The symmetric form for $n = 1$ is identical to the argument w9j. ♣The output has always Rsummationset = { NULL } and Rfactor = 1.

See also: Racah_symmetricw3j(), Racah_symmetricw6j().

References

[1] G. Racah, Phys. Rev. 61 (1941) 186; 62 (1942) 438; 63 (1943) 367.
[2] E.P. Wigner, Group Theory and its Application to the Quantum Mechanics of Atomic Spectra (Academic Press, New York, 1959).
[3] A. De-Shalit and I. Talmi, Nuclear Shell Theory (Academic Press, New York, 1963)
[4] A.R. Edmonds, Angular Momentum in Quantum Mechanics (Princeton University Press, New York, 1957).
[5] B.R. Judd, Operator Techniques in Atomic Spectroscopy (McGraw-Hill, New York, 1963).
[6] D.M. Brink and G.R. Satchler, Angular Momentum (Clarondon Press, Oxford, 1968).
[7] D.A. Varshalovich, A.N. Moskalev, V.K. Khersonskii, Quantum Theory of Angular Momentum (World Scientific, Singapore, 1988).
[8] A.P. Yutsis and A.A. Bandzaitis, The Theory of Angular Momentum in Quantum Mechanics (Mintus, Vilnius, 1965).
[9] E. El-Baz and B. Castel, Graphical Methods of Spin Algebras in Atomic, Nuclear, and Particle Physics (Marcel Decker, New York, 1972).
[10] I. Lindgren and J. Morrison, Atomic Many-Body Theory, 2nd ed. (Springer, Berlin, 1986).
[11] E.U. Condon and Q.W. Shortley, The Theory of Atomic Spectra (Cambridge Univ. Press, Cambridge, 1935).
[12] M. Rotenberg, R. Bivins, N. Metrapolis, and J.K. Wooten Jr., The 3–*j* and 6–*j* Symbols (The Technology Press, Cambridge, Massachusetts, 1959).

[13] T. Regge, Nuovo Cimento 10 (1958) 544.
[14] D. Redfern, The Maple Handbook (Springer, New York, Berlin, 1994).
[15] W.R. Johnson, S.A. Blundell and J. Sapirstein, Phys. Rev. A 37 (1988) 2764; A 38 (1988) 2699; A 41 (1990) 1698; A 42 (1990) 1087.
[16] I. Lindgren, Phys. Scr. 36 (1987) 591; in: Proc. of the Program on Relativistic, Quantum Electrodynamics, and Weak Interaction Effects in Atoms, ITP Santa Barbara, AIP Conference Series 189 (1989) 371.
[17] J.D. Louck, Phys. Rev. 110 (1958) 815.