Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Physics Communications 173 (2005) 140–161
Computer Physics Communications
www.elsevier.com/locate/cpc

Symbolic algorithms for the computation of Moshinsky brackets and nuclear matrix elements ☆

D. Ursescu [a,*], M. Tomaselli [a,b], T. Kuehl [a], S. Fritzsche [c]

[a] *Gesellschaft für Schwerionenforschung, Planckstr. 1, D-64291 Darmstadt, Germany*
[b] *Institut für Kernphysik, Universität Darmstadt, Schlossgartenstr. 9, D-64289 Darmstadt, Germany*
[c] *Fachbereich Physik, Universität Kassel, Heinrich-Plett-Str. 40, D-34132 Kassel, Germany*

Received 17 August 2004; accepted 6 September 2004

Available online 25 August 2005

Abstract

To facilitate the use of the extended nuclear shell model (NSM), a FERMI module for calculating some of its basic quantities in the framework of MAPLE is provided. The Moshinsky brackets, the matrix elements for several central and non-central interactions between nuclear two-particle states as well as their expansion in terms of Talmi integrals are easily given within a symbolic formulation. All of these quantities are available for interactive work.

Program summary

Title of program: Fermi
Catalogue identifier: ADVO
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/ADVO
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 is has been tested: All computers with a licence for the computer algebra package MAPLE [Maple is a registered trademark of Waterloo Maple Inc., produced by MapleSoft division of Waterloo Maple Inc.]
Instalations: GSI-Darmstadt; University of Kassel (Germany)
Operating systems or monitors under which the program has beentested: WindowsXP, Linux 2.4
Programming language used: MAPLE 8 and 9.5 from MapleSoft division of Waterloo Maple Inc.
Memory required to execute with typical data: 30 MB
No. of lines in distributed program including test data etc.: 5742

☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail address: d.ursescu@gsi.de (D. Ursescu).

0010-4655/$ – see front matter
doi:10.1016/j.cpc.2004.09.012

No. of bytes in distributed program including test data etc.: 288 939
Distribution program: tar.gz
Nature of the physical problem: In order to perform calculations within the nuclear shell model (NSM), a quick and reliable access to the nuclear matrix elements is required. These matrix elements, which arise from various types of forces among the nucleons, can be calculated using Moshinsky's transformation brackets between relative and center-of-mass coordinates [T.A. Brody, M. Moshinsky, Tables of Transformation Brackets, Monografias del Instituto de Fisica, Universidad Nacional Autonoma de Mexico, 1960] and by the proper use of the nuclear states in different coupling notations.
Method of solution: Moshinsky's transformation brackets as well as two-nucleon matrix elements are provided within the framework of MAPLE. The transformation brackets are evaluated recursively for a given number of shells and utilized for the computation of the two-particle matrix elements for different coupling schemes and interactions. Moreover, a simple notation has been introduced to handle the two-particle nuclear states in ll-, LSJ-, and jj-coupling, both in the center-of-well and the relative and center-of-mass coordinates.
Restrictions onto the complexity of the problem: The program supports in principle an arbitrary number of shell states with the only limitation given by the computer resources themselves. Typically, the time requirements for the recursive computation of the Moshinsky brackets and matrix elements increase rapidly with the number of the allowed shell states but can be reduced significantly by the *pre-calculation* of the transformation brackets.
Unusual features of the program: Moshinsky brackets are computed and provided in either numeric, algebraic or some symbolic form. In addition, the two-particle matrix elements are calculated for a scalar potential, spin-orbit coupling and tensorial forces, both in floating-point and algebraic notation. All two-particle matrix elements are expressed in terms of the Talmi integrals but can be evaluated also explicitly for several predefined types of the interaction. To simplify the handling of the program, a short but very powerful notation has been introduced which help the user to *deal with* the two-particle states in various coupling notations. The main commands of the current version of the program are described in detail in Appendix B.
Typical running time: The computation of all Moshinsky brackets in floating-point notation, up to $\rho = 6$, takes about 5 s at a 2.26 GHz Intel Pentium IIII processor with 512 MB; in algebraic form, the same computations take about 13 s. Similarly, the computation of these brackets up to $\rho = 10$ requires in numeric and algebraic form about 5 and 15 min, respectively. Once these coefficients have been stored, however, the program replies rather promptly on most further requests.

PACS: 21.60.-n

Keywords: Angular momentum; Center-of-well coordinates; Harmonic potential well; jj-coupling; LSJ-coupling; Moshinsky bracket; Nuclear matrix element; Nuclear shell model; Nuclear two-particle states; Relative and center-of-mass coordinates; Transformation bracket

1. Introduction

For several decades, the nuclear shell model (NSM) has been applied successfully to the study of the structure and properties of a large range of nuclei. Besides the excitation energies, it has been used to predict the charge and magnetization distributions. The NSM is based on the assumption that the nucleon–nucleon interaction can be treated quite effectively by means of a mean-field, i.e. in terms of a single-particle potential well while the residual interaction is treated perturbatively.

However, a theory based on a single potential well does in general not describe the deformation of the experimentally observed nuclei correctly. These deformations are understood only if the residual two-body interactions are taken into account.

By using a harmonic oscillator well to approximate the mean field potential, shell model calculations can reproduce the order of ground- and the low-lying excited levels, because the long-range behavior of the potential is only of little importance for these states. However, highly-excited states or nuclear reaction cross sections are not reliably estimated by a harmonic well without introducing a residual interaction. By including the residual interaction in the model, a large number of nuclear phenomena could be explained. The NSM has considerably increased our present understanding of nuclear structure far off the stability lines. Moreover, non-linear shell-model corrections were used to investigate successfully the nuclear-halo structures of exotic nuclei.

However, there are two major difficulties associated with the use of NSM. First, the large class of nuclear structure calculations considered up to now, depend strongly on the evaluation of the matrix elements of the residual interaction. This calculation is complicated by the increasing number of terms to be calculated especially if one consider more particles outside a closed shell or if different types of interactions between the pairs of nucleons are assumed. Typically, it is quite time-consuming to evaluate the matrix elements and to carry out detailed computations for nuclear effects. Second, since the nucleon–nucleon interactions are not known from first principles, the computational methods must be sufficiently general to allow various assumptions and to repeat some computation for a different choice of the form of the interaction.

From their introduction [1,2], the oscillator brackets have been computed in different ways [3–7]. Using the oscillator brackets and Racah algebra, the two body matrix elements can be computed, see for example [2,8].

All these programs were written in the FORTRAN language. They find most of their applications in NSM. Nevertheless, they have also been used in other fields of physics, including diatomic molecules [9] or the computation of Boltzmann collision matrix elements [10].

Still, at this time, there is no complete tool to provide easy access to the oscillator brackets and to the Wigner $n-j$ symbols. Our program provides simple access to the oscillator brackets and to the two particle matrix elements for nuclear physics using MAPLE [11]. Together with the RACAH package [12–15] it forms a powerful tool for theoreticians in nuclear structure and other fields of physics.

2. Theoretical background

2.1. NSM with harmonic potential wells

Following the standard procedure in the NSM, the nucleon–nucleon interaction of the (many-nucleon) Hamiltonian can be factorized in terms of Slater integrals formed by Slater determinants which include both geometrical factors and radial integrals. While the techniques from Racah's algebra are then used to evaluate all the angular parts of the matrix elements, the radial integrals are calculated either analytically (for simple models of the nucleon–nucleon interaction) or need to be determined numerically. Compared to the evaluation of atomic Slater integrals, however, the computation of the nuclear ones is more sophisticated. A rather large number of different forces are known and since, even after 50 years of nuclear physics, these forces are by far less well established.

Therefore, the calculations should be performed using an easy and quick algorithm. As shown by Talmi [1], a first simplification is possible if the central (one-particle) potential is taken to be a harmonic oscillator potential as in the phenomenological theory. In this case, it is possible to reduce the nuclear radial integrals to a set of analytical integrals which are used also as fitting parameters in the case that an *effective* nuclear model is constructed.

2.2. Effective one-particle matrix elements in a harmonic potential

For the harmonic oscillator well, the Schrödinger equation can be solved analytically as for the hydrogen atom. In polar coordinates, the solution is

$$|nlm\rangle = R_{nl}(r)Y_{lm}(\vartheta,\varphi), \tag{1}$$

where the $Y_{lm}(\vartheta,\varphi)$ denote the spherical harmonics and the radial functions are given in terms of the Laguerre polynomials $L_{\mu,\nu}(x)$ by

$$R_{nl}(r) = \left[\frac{2n!}{\Gamma(n+l+3/2)}\right]^{1/2} r^l \exp\left(\frac{r^2}{2}\right) L_{n,l+1/2}(r^2). \tag{2}$$

The radial coordinate r is taken in the nuclear length unit $b_o = (\hbar/M\omega)^{1/2}$ as the nuclear 'size parameter'. This length unit relates the mass M of the nucleon to the frequency ω as associated with the harmonic potential. In

contrast to the atomic notation, however, a principal quantum number $n \geqslant 0$ is used in nuclear physics and gives rise to the one-particle states $0s, 0p, 1s, 1p, 0f, 2s, \ldots$ with energies $\epsilon_{nl} = 2n + l + 3/2$ in units of $\hbar\omega$.

2.3. Coupling of two-particle states in a given reference frame

Two particles moving in a harmonic potential can be described either by using the center-of-well (cw) coordinates $\mathbf{r}_1$ and $\mathbf{r}_2$ or the relative and center-of-mass (rcm) coordinates $\mathbf{r}$ and $\mathbf{R}$, respectively, which are related to each other by

$$\mathbf{r} = \frac{1}{\sqrt{2}}(\mathbf{r}_1 - \mathbf{r}_2), \qquad \mathbf{R} = \frac{1}{\sqrt{2}}(\mathbf{r}_1 + \mathbf{r}_2). \tag{3}$$

For the rcm coordinates, the form of the harmonic oscillator wave functions is still the same as in (1) and (2) with the only difference that the principal and orbital quantum numbers n, l characterize the relative motion while N, L refer to the center-of-mass motion. As usual, the orbital angular momenta of the two subsystems can be coupled to a total angular momentum λ which must obey the usual coupling conditions

$$|l_1 - l_2| \leqslant \lambda \leqslant l_1 + l_2, \qquad |l - L| \leqslant \lambda \leqslant l + L. \tag{4}$$

For the unperturbed (no interaction) two-particle system, the total energy of the system is simply the sum of the one-particle energies. The energy index of the two-particle system must be the same in the different coordinates systems due to the conservation of energy:

$$\rho = 2n_1 + l_1 + 2n_2 + l_2 = 2n + l + 2N + L. \tag{5}$$

This quantity is often used to introduce the selection rules for transformation brackets and matrix elements.

The two particle states are rotationally invariant, i.e. one-particle states $|n_1 l_1\rangle$ and $|n_2 l_2\rangle$ in the cw coordinates or, respectively, $|nl\rangle$ and $|NL\rangle$ in the rcm coordinates, are coupled to a well-defined total orbital momentum λ with projection μ

$$\big|(n_1 l_1, n_2 l_2)\lambda\mu\big\rangle = \sum_{m_1, m_2} |n_1 l_1 m_1; n_2 l_2 m_2\rangle\langle l_1 m_1, l_2 m_2|\lambda\mu\rangle, \tag{6}$$

$$\big|(nl, NL)\lambda\mu\big\rangle = \sum_{m, M} |nlm; NLM\rangle\langle lm, LM|\lambda\mu\rangle. \tag{7}$$

In (6) and (7) we use the standard Clebsch–Gordan definition, where m_1, m_2 and m, M denote the magnetic quantum numbers.

2.4. Transformation brackets and coefficients

For any two body wave function which has the total angular momentum λ and projection μ, there is a simple (and orthogonal) transformation between the center-of-well and reduced center-of-mass coordinates

$$\big|(n_1 l_1, n_2 l_2)\lambda\mu\big\rangle = \sum_{nlNL} \big|(nl, NL)\lambda\mu\big\rangle\big\langle(nl, NL)\lambda\big|(n_1 l_1, n_2 l_2)\lambda\big\rangle, \tag{8}$$

where the transformation coefficients $\langle(nl, NL)\lambda|(n_1 l_1, n_2 l_2)\lambda\rangle$ are independent of μ and are known as the Moshinsky brackets in the literature. Owing to the conservation of energy, moreover, these brackets are zero for all pairs of rcm and cw two-particle states which do not have the same energy index ρ. In practice, the Moshinsky brackets are calculated recursively, by starting from expressions for $n_1 = n_2 = 0$. For the sake of completeness, these formulas are given below in Appendix A.

2.5. Effective two-particle matrix elements

Taking a look at the standard tabulations of the two-particle matrix elements such as Ref. [2], a first observation is that the use of computer-algebraic techniques makes the traditional distinction between the diagonal and non-diagonal terms introduced for interaction potential matrix elements unnecessary. In the FERMI program, of course, both types of these interaction matrix elements can be obtained from the same procedures, making their use more transparent than before. In this subsection we consider the computation of single matrix elements, leaving aside their storage and application. The further manipulation of matrix elements and transformation brackets is provided by MAPLE having a wide range of built-in features which need not to be described here.

To compute a matrix element between two states in the cw coordinates, we start from the definition of these states and first transform them into the rcm coordinates by using the expansion (8). The use of rcm coordinates is justified by the fact that most realistic interaction potentials do not depend on the center-of-mass coordinate **R** of (3). In many applications, therefore, the quantum numbers N and L are simply omitted from the notation and are shown explicitly only when required to avoid ambiguities.

2.5.1. Reduced matrix elements

Making use of rcm coordinates we introduce the reduced matrix elements of a given central potential $V(r) \equiv V(|\mathbf{r}_1 - \mathbf{r}_2|)$ as given by Moshinsky in [2]: $\langle nl \| V(r) \| n'l' \rangle$. These reduced elements describe the *physical part* of the interaction among the nucleons while its geometrical part can easily be factorized using the Wigner–Eckart theorem. Using the expansion (2) for the radial function $R_{nl}(r)$, the reduced matrix element can be written

$$\langle nl \| V(r) \| n'l' \rangle = \sum_p B(nl, n'l', p), I_p, \tag{9}$$

where I_p are the Talmi integrals [1]

$$I_p = \frac{2}{\Gamma(p+3/2)} \int_0^\infty \mathrm{d}r\, r^{2p+2} \mathrm{e}^{-r^2} V(r) \tag{10}$$

and the $B(nl, n'l', p)$ are angular coefficients whose explicit form is given in Appendix A. In practice, this decomposition is useful if the Talmi integrals in (9) are used as adjustable parameters which fit the theoretical predictions or experiments. The reduced matrix elements and the Talmi integrals are provided by our FERMI package.

2.5.2. Matrix elements for various interactions

Different matrix elements for the two-particle states in cw coordinates are of interest in applications. Apart from a pure central potential $V(r) \equiv V(|\mathbf{r}_1 - \mathbf{r}_2|)$ and states in a ll-coupled basis, there are two further types of matrix elements: the spin-orbit coupling potential $V_{LS}(r)\mathbf{l} \cdot \mathbf{s}$ as well as tensor forces V_T for LSJ-coupled states with well defined total angular momentum J and its z-projection M. The above interactions are not scalar but contain first- and second-rank spherical tensors. In the FERMI program, we support the evaluation and computation of all of these matrix elements using a simple notation. The origin of these interactions is well discussed in the literature. Here we display only the basic decomposition of the matrix elements and leave further details for Appendix A.

Central-field interaction matrix in the ll-coupled basis:

$$\langle n_1 l_1, n_2 l_2, \lambda | V(r) | n'_1 l'_1, n'_2 l'_2, \lambda' \rangle = \sum_p \delta_{\lambda\lambda'} C_c(n_1 l_1, n_2 l_2, n'_1 l'_1, n'_2 l'_2; \lambda; p) I_p. \tag{11}$$

Spin-orbit interaction matrix in the LSJ-coupled basis:

$$\left\langle n_1 l_1, n_2 l_2, \lambda; \frac{1}{2}\frac{1}{2} s; JM \middle| V_{LS}(r)\mathbf{l}\cdot\mathbf{s} \middle| n_1' l_1', n_2' l_2', \lambda'; \frac{1}{2}\frac{1}{2} s'; J'M' \right\rangle = \hbar \sum_p \delta_{JJ'}\delta_{MM'}\delta_{ss'}\delta_{s1} C_{LS}\big(n_1 l_1, n_2 l_2, \lambda, n_1' l_1', n_2' l_2'; \lambda'; J; p\big) I_p. \tag{12}$$

Tensor-force interaction matrix in the LSJ-coupled basis: To describe the spin–spin interaction among two nucleons with spins $\mathbf{s}_1$ and $\mathbf{s}_2$, a product interaction of two second-rank tensors

$$V_T = \left(\frac{32}{5}\pi\right)^{1/2} V(r) Y_2(\theta,\phi)\cdot X_2 \tag{13}$$

is used, where $Y_2(\theta,\phi)$ is the tensor of the second-rank spherical harmonics $Y_{2m}(\theta,\phi), m = -2,-1,\ldots,2$, and X_2 a tensor whose $m=0$ component is given by

$$X_{20} = \frac{1}{\sqrt{2}}\big(3S_z^2 - S^2\big) \tag{14}$$

with $\mathbf{S} = \mathbf{s}_1 + \mathbf{s}_2$. With these conventions, the interaction matrix for the tensor force is written as [2]

$$\left\langle n_1 l_1, n_2 l_2, \lambda; \frac{1}{2}\frac{1}{2} S; JM \middle| V_T \middle| n_1' l_1', n_2' l_2', \lambda'; \frac{1}{2}\frac{1}{2} S'; J'M' \right\rangle = \hbar^2 \sum_p \delta_{JJ'}\delta_{MM'}\delta_{SS'}\delta_{S1} C_T\big(n_1 l_1, n_2 l_2, \lambda, n_1' l_1', n_2' l_2'; \lambda'; J; p\big) I_p. \tag{15}$$

In all three decompositions (11), (12), and (15) introduced for the two-particle matrix elements, the coefficients C_c, C_{LS} and C_T are expressed in terms of the Moshinsky brackets. The B-coefficients from (9) as well as the Wigner $3-j$ and $6-j$ symbols are calculated within the FERMI program (see Appendix A).

3. Program organization

3.1. Overview and main commands

The combination of symbolic and numerical manipulations as provided by most computer-algebra systems offers a great advantage compared to traditional programming languages. In addition to providing a common environment for performing both the algebraic and numerical analysis of a given problem, it allows to define mathematical objects and a number of basic quantities.

In the FERMI package we introduce symbolical objects to define two-particle nuclear states and to evaluate the Talmi–Moshinsky integrals (or shortly the so-called Moshinsky brackets). In addition, we support the computation of the two-particle matrix elements for the computation of matrix elements for scalar as well as spin-orbit and tensorial interactions between two particle states.

At the user's level, only very few procedures have to be known in order to obtain the transformation brackets or matrix elements as discussed above. For a quick overview, Table 1 displays a short list of the main commands. Owing to the complexity of many internal expressions, however, a large number of hidden procedures had to be implemented at a lower level of the program hierarchy.

Table 1
Commands of the FERMI module for calculating various basic quantities used in the NSM, i.e. for nucleons in an harmonic potential. A more detailed description of these procedures, which are provided for interactive work, is given in Appendix B. Our commands also include a number of auxiliary procedures in order to facilitate the *communication* with and within the program

cw_ll(), cw_LSJ(), cw_LSJM(), cw_jj()	Provide a nuclear two-particle state in the center-of-well (cw) coordinates in either ll-, LSJ-, or jj-coupling, respectively.
rcm_ll(), rcm_LSJ(), rcm_jj()	Provide a nuclear two-particle state in the relative and center-of-mass (rcm) coordinates in either ll-, LSJ- or jj-coupling, respectively.
Fermi_rho()	Returns the energy index of a given nuclear two-particle state.
Fermi_shell()	Generates a list of quantum numbers for all (non-interacting) two-particle states which belong to the same energy shell, i.e. all states with the same energy index ρ and λ.
Fermi_tabulate()	Returns a table with the arguments from an auxilliary procedure to facilitate work and programming.
Fermi_w3j(), Fermi_w6j(), Fermi_w9j()	Computes the Wigner $3-j$, $6-j$, and $9-j$ symbols, respectively.
RacahW()	Computes the Racah W function.
Moshinsky_bracket()	Returns the transformation bracket $\langle (nl, NL)\lambda \vert (n_1l_1, n_2l_2)\lambda\rangle$.
Moshinsky_predefine()	Calculates and stores all the Moshinsky brackets $\langle (nl, NL)\lambda \vert (n_1l_1, n_2l_2)\lambda\rangle$ up to a specified energy index for the nuclear two-particle states; these values are used for the computation of matrix elements and elsewhere.
Moshinsky_orthogonality()	Checks the orthogonality relation for the Moshinsky brackets.
Talmi_integral()	Returns the Talmi integral for a given order p and potential $V(r)$.
Fermi_V_central()	Calculates the matrix element $\langle n_1l_1, n_2l_2\lambda \vert V(r) \vert n'_1l'_1, n'_2l'_2\lambda'\rangle$ between two ll-coupled two-particle states and for a central potential $V(r)$.
Fermi_V_reduced()	Computes the reduced matrix element $\langle nl \Vert V \Vert n'l'\rangle$ for a given central potential.
Fermi_V_spin_orbit()	Calculates the matrix element $\langle n_1l_1n_2l_2\lambda SJM \vert V_{ls}(r) \vert n'_1l'_1n'_2l'_2\lambda' S'J'M'\rangle$ between two LSJ-coupled states and for the spin-orbit interaction from (12).
Fermi_V_tensor_X2()	Calculates the matrix element $\langle n_1l_1n_2l_2\lambda SJM \vert V_{ss}(r) \vert n'_1l'_1n'_2l'_2\lambda' S'J'M'\rangle$ between two LSJ-coupled states and for the spin–spin tensorial interaction from (14).

3.2. Use of auxiliary procedures

Computations involving nuclear states must provide the user with a quick and simple access to the basic functions. To support the *definition* and communication of such functions, seven *auxiliary* procedures have been designed and implemented in the FERMI program. They keep all necessary information about a two particle state together.

In this part of the program we define only two particle states in different coupling schemes and in two different reference coordinates. We designate by cw and rcm nuclear states in the center-of-well and in the reduced center-of-mass reference systems, respectively. All the computations are performed in the LS-coupling scheme; the jj-coupling was defined here for later use. We also define reduced states designated by "_", while in many of the procedures it is enough to know the quantum numbers for the energy and kinetic momentum for the two particles without taking into account the spin S and the total J.

These procedures such as cw_ll(), cw_LSJ(), cw_LSJM() and cw_jj() or rcm_ll(), rcm_LSJ() and rcm_jj(), respectively, are defined separately for each coupling scheme, i.e. for LS- or jj-coupling, and, hence, could be easily extended to include other coupling schemes in the future.

3.3. Distribution and installation

To facilitate the installation of the program, the FERMI package is distributed as a whole by the zip file `Fermi2004.zip` from which the `Fermi` root directory is (re-)generated. This directory contains the source

code (tested with MAPLE 8 and 9.5), a `Read.me` for the installation of the program as well as the document `Fermi-commands.ps`. In fact, this document may serve as a short manual which provides an alphabetic list of all user relevant (and exported) commands. Although emphasis was placed to preserve the compatibility of the program with respect to earlier releases of MAPLE, this cannot always be ensured due to changes in the MAPLE syntax.

4. Examples. Computation of transformation brackets and interaction matrix elements

A few examples from nuclear shell calculations are displayed below to illustrate the use of the Moshinsky brackets as introduced in Section 2. Beside of the computation of a particular transformation bracket, we briefly explain how one can accelerate their application in the computation of reduced matrix elements by first calculating all the necessary brackets at the beginning of the execution.

We start by first defining two (two-particle) states in the cw and rcm coordinates with well defined total angular momentum $\lambda = 3$

```
> a := cw_ll(1,2,0,2,3);
```

$$\mathrm{a} := \mathrm{cw\_ll}(1, 2, 0, 2, 3)$$

```
> b := rcm_ll(0,1,1,3,3);
```

$$\mathrm{b} := \mathrm{rcm\_ll}(0, 1, 1, 3, 3)$$

and with the same energy index ρ in order to obtain a non-zero transformation bracket:

```
> Fermi_rho(a); Fermi_rho(b);
```

6

6

With this definition, we may now compute the Moshinsky brackets either numerically

```
> Moshinsky_bracket(b,a);
```

0.4008918627

or in *algebraic* form by using this keyword as a third parameter in the list below

```
> m := Moshinsky_bracket(b,a,algebraic);
```

$$\mathrm{m} := \frac{3\sqrt{2}\sqrt{7}}{28}$$

```
> evalf(m,11);
```

0.40089186287

which, of course, gives the same value up to some rounding error. Moreover, since the Moshinsky brackets are finally reduced to some linear combination of Wigner $6-j$ symbols, this explicit representation is obtained by the keyword *symbolic*

```
> Moshinsky_bracket(b,a,symbolic);
```

$$-\frac{3}{20}\sqrt{2}\sqrt{3}\sqrt{5}\text{w6j\_}(1,3,3,2,2,1)+\frac{3}{4}\sqrt{5}\text{w6j\_}(1,3,3,2,2,2)+\frac{3}{20}\sqrt{7}\sqrt{3}\sqrt{5}\text{w6j\_}(1,3,3,2,2,3)$$

and has been introduced mainly for teaching purposes. To make use of the two-particle states in the different coupling and coordinate systems, their quantum numbers can be extracted by the procedure Fermi_tabulate() which returns a table

```
> q:= Fermi_tabulate(a);
```

$$q := T$$

```
> q[n1_]; q[l1_]; q[n2_]; q[lambda_]; a;
```

$$1$$

$$2$$

$$0$$

$$3$$

$$\text{cw\_ll}(1,2,0,2,3)$$

The procedure Fermi_shell() can be used to generate a list of states from the same subshells. Since the transformation brackets are evaluated recursively, it's often helpful to compute and store all brackets up to some specified energy index ρ. In the FERMI program, this is achieved by calling the command Moshinsky_predefine(rho) by which the coefficients are kept in the global variable Moshinsky_stored_ while another global variable Moshinsky_level_ is set to *rho*. To modify this storage, it is enough to call the procedure Moshinsky_predefine() again, using a negative integer to give the storage *free* for other use. The default value for Moshinsky_level_ is -2.

```
> Moshinsky_level_; Moshinsky_stored_;
```

$$-2$$

$$[]$$

```
> Moshinsky_predefine(6);
```

$$\text{ok}$$

```
> Moshinsky_level_;
```

$$6$$

Note that the recursive algorithm as discussed above enables one to compute the transformation brackets for any higher value of ρ by using the brackets from the previous run. For example, we may *add* the brackets for $\rho = 7$ and 8 by calling

```
> Moshinsky_predefine(8);
```

$$\text{ok}$$

To check the coefficients, an 'orthogonality test' according to formulae (A.5), (A.6) can be performed by using two states with the same coupling scheme and energy index as well as for the same choice of coordinates

```
> Moshinsky_orthogonality(a,cw_ll(0,4,0,2,3));
```

$$-0.3 \cdot 10^{-9}$$

```
> Moshinsky_orthogonality(b,b);
```

0.9999999988

Having provided the access to the Moshinsky brackets, we are now prepared to compute the Talmi integrals and the reduced matrix elements for states with a different coupling and defined in the center-of-well coordinates. The Talmi integral is the basic *radial* part in the computations and can be specified either by means of a (user-defined) function Talmi_I()

```
> Talmi_integral(2);
```

Talmi_I(2)

or simply in terms of some predefined interactions potentials such as an *harmonic* one

```
> Talmi_integral(4,k,harmonic);
```

$$-\frac{11 \cdot k}{2}$$

Instead of using a *harmonic* potential between the particles, other forms of the potential can be selected using the keywords *Gaussian*, *unity*, or *rInverse* where one may have to specify also further parameters as described in Appendix B. In general, the Talmi integrals (10) arise in the representation of the reduced interaction matrix elements given by (A.11) and of the interaction matrix elements (A.14), (A.16), (A.19).

```
> Fermi_V_reduced(1,2,0,4);
```

$$\frac{7}{6}\sqrt{2}\text{Talmi\_I}(3) - \frac{3}{2}\sqrt{2}\text{Talmi\_I}(4)$$

and, hence, can be inserted numerically in the computations or are left *unevaluated* as in the example above. Both the Moshinsky brackets and the reduced matrix elements are needed for calculating the matrix elements for different types of interactions and the various definitions of the two-particle states. If we first define a second ll-coupled state in the cw coordinates

```
> c:=cw_ll(0,4,0,2,3);
```

c := cw_ll(0, 4, 0, 2, 3)

we can evaluate, for instance, the matrix element $\langle n_1 l_1, n_2 l_2 \lambda | V(r) | n_1' l_1', n_2' l_2' \lambda' \rangle$ (in our case: $\langle 04, 02, 3 | V(r) | 12, 02, 3 \rangle$) between two ll-coupled two-particle states and for a central potential $V(r)$ in terms of the Talmi integrals by typing

```
> Fermi_V_central(a,c);
```

$$-0.1796840341 \cdot \text{Talmi\_I}(1) + 0.8385254919 \cdot \text{Talmi\_I}(2) - 0.4791574244 \cdot \text{Talmi\_I}(3)$$
$$- 0.8385254911 \cdot \text{Talmi\_I}(4) + 0.6588414574 \cdot \text{Talmi\_I}(5)$$

or explicitly, if we assume either a *unit* potential $V(r) = V_1$ (in this case we should obtain the value 0):

```
> Fermi_V_central(a,c,V1,unity);
```

$$-0.3000000000 \cdot 10^{-9} \cdot V1$$

or a harmonic form $V(r) = -kr^2$ so the k is positive for an attractive (negative) potential

```
> Fermi_V_central(a,c,k,harmonic);
```

$$0.1000000000 \cdot 10^{-8} \cdot \mathrm{k}$$

The result should be again zero in the previous harmonic potential case. The non-zero result is numerical noise; to avoid it we have to compute this matrix element in algebraic way. Of course, both the constants V_1 and k could be given also numerically.
In a similar way, we may proceed for other types of interaction operators such as the spin-orbit or the tensorial interaction as defined in (A.16) and (A.19), respectively. For these types of interactions, information about the spins and the total angular momenta is needed and can be provided by defining the two LSJ-coupled states

```
> d := cw_LSJM(2,0,0,2,2,1,3,1);
  e := cw_LSJM(0,1,1,3,2,1,3,1);
```

$$\mathrm{d} := \mathrm{cw\_LSJM}(2, 0, 0, 2, 2, 1, 3, 1)$$

$$\mathrm{e} := \mathrm{cw\_LSJM}(0, 1, 1, 3, 2, 1, 3, 1)$$

Again, different potentials can be used for the interaction among the nucleons

```
> Fermi_V_spin_orbit(d,e,k,harmonic);
```

$$-0.6200000000 \cdot 10^{-8} \cdot hbar\_ \cdot \mathrm{k}$$

```
> Fermi_V_tensor_X2(d,e,V0,rInverse);
```

$$-0.01459703364 \cdot \mathrm{VO} \cdot hbar\_^{2}$$

In order to obtain the matrix elements in algebraic or symbolic form, one has first to *store* the Moshinsky brackets in their *algebraic* representation. We achieve this by calling the procedure

```
> Moshinsky_predefine(-2);
```

"no pre-defined values for Moshinsky brackets will be further used"

ok

to 'delete' all pre-defined numerical values and to recalculate them with the proper optional argument

```
> Moshinsky_predefine(6,algebraic);
```

ok

In this way, we can confirm the *zero*-values for the two matrix elements

```
> Fermi_V_central(a,c,V1,unity);
```

0

```
> Fermi_V_central(a,c,k,harmonic);
```

0

and find the algebraic representation for the matrix element

```
> Fermi_V_tensor_X2(d,e,algebraic,V0,rInverse);
```

$$-\frac{23 hbar\_^{2}\sqrt{7} V0}{2352\sqrt{\pi}}$$

5. Outlook

Providing a set of useful commands, the FERMI program facilitates the transformation of two-particle wave functions from the center-of-well to the reduced and center-of-mass coordinates and *vice versa*. In addition to the (geometrical) transformation brackets, the interaction matrix elements for various interactions can be computed by using the Talmi integrals. The results obtained from the FERMI package can be used as input for more complex computations such as many particle models.

To facilitate the computations, advantage has been taken from the symbolic features as offered by MAPLE. Mixed numeric–algebraic–symbolic computation of the harmonic oscillator brackets (Moshinsky brackets) and interaction matrix elements can be performed for a specific problem and results are returned in parameterized form. With the implementation of data structures for the quantum description of the nuclear states [cf. the auxiliary procedures `cw_ll(), rcm_ll(), ...`], we also provide a standard notation to be used for various tasks. Their use will be extended in the further developments of the FERMI package where we will implement procedures for computation of the interaction matrix elements for certain nuclear models as known from the literature.

Acknowledgements

We thank to Thomas Stöhlker for providing us the access to a MAPLE 8 installation under the Windows XP operating system.

Appendix A. Definitions and basic relations

A.1. Moshinsky's transformation brackets

The energy index for a (non-interacting) two-particle state in rcm or in cw coordinates is defined as:

$$\rho = 2n + l + 2N + L, \quad \rho = 2n_1 + l_1 + 2n_2 + l_2. \tag{A.1}$$

The transformation brackets $\langle (nl, NL)\lambda | (n_1 l_1, n_2 l_2)\lambda \rangle$ between the nuclear two-particle states in the cw and rcm coordinates are generated recursively, starting from an expression for $n_1 = n_2 = 0$ [2]

$$\begin{aligned}&\big\langle (nl, NL)\lambda \big| (0l_1, 0l_2)\lambda \big\rangle \\ &\quad = \left[\frac{l_1! l_2!}{(2l_1)!(2l_2)!} \frac{(2l+1)(2L+1)}{2^{l+L}} \frac{(n+l)!}{n!(2n+2l+1)!} \frac{(N+L)!}{N!(2N+2L+1)!} \right] \\ &\qquad \times (-1)^{n+l+L-\lambda} \sum_x (2x+1) A(l_1 l, l_2 L, x) W(l L l_1 l_2; \lambda x),\end{aligned} \tag{A.2}$$

where $W(l L l_1 l_2; \lambda x)$ is the Racah W coefficient and the coefficient A is given by

$$\begin{aligned}&A(l_1 l, l_2 L, x) \\ &\quad = \left[\frac{(l_1 + l + x + 1)!(l_1 + l - x)!(l_1 + x - l)!}{(l + x - l_1)!} \frac{(l_2 + L + x + 1)!(l_2 + L - x)!(l_2 + x - L)!}{(L + x - l_2)!} \right]^{1/2} \\ &\qquad \times \sum_q (-1)^{\frac{l+q-l_1}{2}} \frac{(l+q-l_1)!}{(\frac{l+q-l_1}{2})!(\frac{l+l_1-q}{2})!} \frac{1}{(q-x)!(q+x+1)!} \frac{(L+q-l_2)!}{(\frac{L+q-l_2}{2})!(\frac{L+l_2-q}{2})!}.\end{aligned} \tag{A.3}$$

The ranges of summation over x and q in the expressions (A.2) and (A.3) are given by the angular momentum coupling rules and the fact that q must not be negative.

Table 2
The matrix elements of $-r_1^2$

n'	l'	N'	L'	$\langle (nl, NL)\lambda \vert -r_1^2 \vert (n'l', N'L')\lambda\rangle$
$n-1$	l	N	L	$\frac{1}{2}[n(n+l+\frac{1}{2})]^{1/2}$
n	l	$N-1$	L	$\frac{1}{2}[N(N+L+\frac{1}{2})]^{1/2}$
$n-1$	$l+1$	$N-1$	$L+1$	$[nN(l+1)(L+1)]^{1/2}(-1)^{\lambda+L+l}W(ll+1LL+1;1\lambda)$
$n-1$	$l+1$	N	$L-1$	$[n(N+L+1/2)(l+1)L]^{1/2}(-1)^{\lambda+L+l}W(ll+1LL-1;1\lambda)$
n	$l-1$	$N-1$	$L+1$	$[(n+l+1/2)Nl(L+1)]^{1/2}(-1)^{\lambda+L+l}W(ll-1LL+1;1\lambda)$
n	$l-1$	N	$L-1$	$[(n+l+1/2)(N+L+1/2)lL]^{1/2}(-1)^{\lambda+L+l}W(ll-1LL-1;1\lambda)$

From (A.2) for the case $n_1 = n_2 = 0$, the transformation brackets with different n_1 and n_2 are obtained with the recursion relation

$$\begin{aligned}&\big\langle (nl, NL)\lambda \big\vert (n_1+1l_1, n_2l_2)\lambda\big\rangle \\ &\quad = \big[(n_1+1)(n_1+l_1+3/2)\big]^{-1/2} \\ &\qquad \times \sum_{n'l'N'L'} \big\langle (nl, NL)\lambda\big\vert -r_1^2\big\vert (n'l', N'L')\lambda\big\rangle\big\langle (n'l', N'L')\lambda\big\vert (n_1l_1, n_2l_2)\lambda\big\rangle, \end{aligned} \tag{A.4}$$

where all two-particle states involved must belong to the same energy shell: $2n + l + 2N + L = 2(n_1 + 1) + l_1 + 2n_2 + l_2$.

The matrix elements of $-r_1^2$ are non-zero for just six two-particle states $\vert (n'l', N'L')\lambda\rangle$ as displayed in Table 2 due to the conservation of energy and the properties of the radial functions. Moreover, the same expression of Table 2 can be applied if, instead of the index n_1 the value of n_2 is to be increased by 1; in this case, the last four expressions of Table 2 must be taken with a minus sign.

Orthogonality and symmetry relations for the transformation brackets: There are two orthogonality relations known for the Moshinsky brackets:

$$\sum_{n_1l_1,n_2l_2} \big\langle (nl, NL)\lambda\big\vert (n_1l_1, n_2l_2)\lambda\big\rangle\big\langle (n'l', N'L')\lambda\big\vert (n_1l_1, n_2l_2)\lambda\big\rangle = \delta_{nn'}\delta_{NN'}\delta_{ll'}\delta_{LL'}, \tag{A.5}$$

$$\sum_{nl,NL} \big\langle (nl, NL)\lambda\big\vert (n_1l_1, n_2l_2)\lambda\big\rangle\big\langle (nl, NL)\lambda\big\vert (n_1'l_1', n_2'l_2')\lambda\big\rangle = \delta_{n_1n_1'}\delta_{n_2n_2'}\delta_{l_1l_1'}\delta_{l_2l_2'}. \tag{A.6}$$

And we have the following symmetry relations:

$$\big\langle (nl, NL)\lambda\big\vert (n_1l_1, n_2l_2)\lambda\big\rangle = (-1)^{L-\lambda}\big\langle (nl, NL)\lambda\big\vert (n_2l_2, n_1l_1)\lambda\big\rangle \tag{A.7}$$

$$= (-1)^{l_1-\lambda}\big\langle (NL, nl)\lambda\big\vert (n_1l_1, n_2l_2)\lambda\big\rangle \tag{A.8}$$

$$= (-1)^{l_1+l}\big\langle (NL, nl)\lambda\big\vert (n_2l_2, n_1l_1)\lambda\big\rangle \tag{A.9}$$

$$= (-1)^{l_2+L}\big\langle (n_1l_1, n_2l_2)\lambda\big\vert (NL, nl)\lambda\big\rangle. \tag{A.10}$$

A.2. $B(nl, NL, p)$ coefficients and Talmi integrals

For any central potential $V(r)$, the matrix elements

$$\langle nl\|V(r)\|n'l'\rangle = \sum_p B(nl, n'l', p)I_p \tag{A.11}$$

can be expressed in terms of the Talmi integrals (10) [1].

The B coefficients are required only if the matrix elements are to be expressed in terms of the Talmi integrals. This decomposition has been found useful, for instance, when no radial dependence of the two-particle interaction has been given explicitly and, thus, the set of Talmi integrals in (A.11) can be used as parameters to adjust the experimental data.

An explicit representation of the $B(nl, n'l', p)$ coefficients in the expansion of the effective one-particle matrix elements for a central interaction $V(r)$ are given in [2]

$$B(nl, n'l', p) = (-1)^{p-\frac{l+l'}{2}} \frac{(2p+1)!}{2^{n+n'}p!} \left[\frac{n!n'!(2n+2l+1)!(2n'+2l'+1)!}{(n+l)!(n'+l')!}\right]^{1/2} \times \sum_{k=\alpha}^{\beta} \frac{(l+k)!(p-\frac{l-l'}{2}-k)!}{k!(2l+2k+1)!(n-k)!(2p-l+l'-2k+1)!(n'-p+\frac{l+l'}{2}+k)!(p-\frac{l+l'}{2}-k)!}, \tag{A.12}$$

where the summation over k is limited to $\max[0, p-1/2(l+l')-n'] \leqslant k \leqslant \min[n, p-1/2(l+l')]$. The possible values of p are therefore

$$1/2(l+l') \leqslant p \leqslant 1/2(l+l')+n+n'. \tag{A.13}$$

A.3. Matrix elements for different interactions

For a central potential $V(r)$, we have the formula (11) for the matrix element where the C_c coefficient is given by

$$C_c(n_1l_1, n_2l_2, n_1'l_1', n_2'l_2'; \lambda; p) = \sum_{nlNL} \left[\left\langle (nl, NL)\lambda \middle| (n_1l_1, n_2l_2)\lambda \right\rangle \left\langle (n'l, NL)\lambda \middle| (n_1'l_1', n_2'l_2')\lambda \right\rangle B(n'l, nl, p)\right], \tag{A.14}$$

where the possible values for the index p are given by (A.13) and

$$n' = n + n_1' + n_2' - n_1 - n_2 + \frac{1}{2}\left(l_1' + l_2' - l_1 - l_2\right). \tag{A.15}$$

For a spin-orbit coupling potential $V_{LS}(r)\vec{l}\cdot\vec{s}$ we have the formula (12) for the matrix element, where the C_{LS} coefficient is given by

$$C_{LS}(n_1l_1, n_2l_2, \lambda, n_1'l_1', n_2'l_2'; \lambda'; J; p) = (-1)^{J+\rho}\sqrt{6}W(\lambda\lambda' 11; 1J)\sqrt{(2\lambda+1)(2\lambda'+1)} \sum_{nlNL\lambda'} \left[W(\lambda\lambda' ll; 1L)\sqrt{l(l+1)(2l+1)} \times \left\langle (nl, NL)\lambda \middle| (n_1l_1, n_2l_2)\lambda \right\rangle \left\langle (n'l, NL)\lambda' \middle| (n_1'l_1', n_2'l_2')\lambda' \right\rangle B(n'l, nl, p)\right], \tag{A.16}$$

where the possible values for the index p are given by (A.13), n' is given by (A.15), ρ is given by (A.1) and

$$\lambda' = \lambda, \lambda \pm 1. \tag{A.17}$$

For a tensor force V_T acting between two particles of spin $\vec{\sigma}_1$ and $\vec{\sigma}_2$ we have the relations (13), (14) and (15) where $Y_2(\theta, \phi)$ is a second order Racah tensor whose components are the spherical harmonics $Y_{2m}(\theta, \phi)$ and X_2 is the second order Racah tensor with

$$\vec{s} = (\vec{\sigma} + \vec{\sigma})/2 \tag{A.18}$$

and the C_T coefficient is given by:

$$C_T(n_1l_1, n_2l_2, \lambda, n_1'l_1', n_2'l_2'; \lambda'; J; p) = (-1)^{\rho-J+1}\sqrt{120(2\lambda+1)(2\lambda'+1)}W(\lambda\lambda'11; 2J)\sum_{l'}\sum_{nlNL}\Big[\sqrt{2l+1}\langle l200|l'0\rangle W(\lambda\lambda' ll'; 2L) \times \big\langle (nl, NL)\lambda\big|(n_1l_1, n_2l_2)\lambda\big\rangle\big\langle (n'l', NL)\lambda'\big|(n_1'l_1', n_2'l_2')\lambda'\big\rangle B(n'l', nl, p)\Big], \tag{A.19}$$

where the possible values for the index p are given by (A.13), n' is given by (A.15) and l' is restricted to

$$l' = l, l \pm 2. \tag{A.20}$$

ρ is given in (A.1), $\langle l200|l'0\rangle$ is the Clebsch–Gordan coefficient and

$$\lambda' = \lambda, \lambda \pm 1, \lambda \pm 2. \tag{A.21}$$

Appendix B. Commands of the FERMI module

For quick reference, we compile and describe here the *main* commands of the FERMI module which are accessible for interactive work. Of course, there are further procedures which are *hidden* from the user and, hence, need not to be explained in detail. Overall, the FERMI module currently comprises about 50 sub-procedures in a hierarchical order. For the presentation of the procedures below, we follow a style similar to MAPLE's internal help pages.

As mentioned above, the communication with the user (and also among the various procedures of the FERMI module) can be simplified considerably if a number of auxiliary procedures are introduced for the two-particle states. These procedures refer to the definition of nuclear states in the different coupling schemes and coordinate systems as frequently applied within the NSM. Indeed, the fast and unique specification of these quantum states are often one of the basic requests by the user in dealing with Moshinsky's transformation brackets and all the necessary matrix elements. Although a few tests are made on the parameters of the auxiliary procedures, they return *unevaluated* and mainly serve for keeping all the necessary information together. Since these procedures occur very frequently during input and output (and, in fact, because no *real* data manipulation is made), *no* prefix `Fermi_` has been added to these help procedures. Moreover, the same omission of the prefix also applies to a number of other procedures such as Moshinsky_bracket(), Talmi_integral() and RacahW(), which are known as *standard* quantities from the literature.

B.1. Auxiliary procedures

Presently, seven auxiliary procedures are provided to specify the two-particle nuclear states in the center-of-well (cw) as well as the relative and center-of-mass (rcm) coordinates.

cw_ll(n_1,l_1,n_2,l_2,lambda)

Auxiliary procedure to represent (i.e. define) a ll-coupled two-particle state $|(n_1l_1, n_2l_2)\lambda\rangle$ with total orbital angular momentum λ in the center-of-well coordinates.
Output: An unevaluated call to cw_ll(n_1, l_1, n_2, l_2, lambda) is returned.
Argument options: (..., *no_check*) to do *no check* on the consistency of the given quantum numbers; this may save some time in lengthy computations.
Additional information: All quantum numbers in the parameter list must evaluate to valid names or integers ($\geqslant 0$) and must fulfill the proper coupling conditions.
See also: cw_LSJ(), cw_LSJM(), cw_jj(), rcm_ll().

cw_LSJ(n1,l1,n2,l2,lambda,S,J)

Auxiliary procedure to represent a LSJ-coupled state $|((n_1l_1, n_2l_2)\lambda, S)J\rangle$ with orbital angular momentum λ, spin S, and total angular momentum J in the center-of-well coordinates.
Output: An unevaluated call to cw_LSJ(n1, l1, n2, l2, lambda, S, J) is returned.
Argument options: (..., *no_check*) to do *no check* on the consistency of the given quantum numbers; this may save some time in lengthy computations.
Additional information: All quantum numbers in the parameter list must evaluate to valid names or integers ($\geqslant 0$) and must fulfill the proper coupling conditions.
See also: cw_ll(), cw_jj(), cw_LSJM(), rcm_LSJ().

cw_LSJM(n1,l1,n2,l2,lambda,S,J,M)

Auxiliary procedure to represent a LSJ-coupled state $|((n_1l_1, n_2l_2)\lambda, S)JM\rangle$ with orbital angular momentum λ, spin S, total angular momentum J and the projection M in the center-of-well coordinates.
Output: An unevaluated call to cw_LSJM(n1, l1, n2, l2, lambda, S, J, M) is returned.
Argument options: (..., *no_check*) to do *no check* on the consistency of the given quantum numbers; this may save some time in lengthy computations.
Additional information: All quantum numbers in the parameter list must evaluate to valid names or integers ($\geqslant 0$) and must fulfill the proper coupling conditions.
See also: cw_ll(), cw_LSJ(), cw_jj(), rcm_LSJ().

cw_jj(n1,l1,j1,n2,l2,j1,J)

Auxiliary procedure to represent a jj-coupled two-particle state $|(n_1l_1j_1, n_2l_2j_2)J\rangle$ with total angular momentum J in the center-of-well coordinates.
Output: An unevaluated call to cw_jj(n1, l1, j1, n2, l2, j1, J) is returned.
Argument options: (..., *no_check*) to do *no check* on the consistency of the given quantum numbers; this may save some time in lengthy computations.
Additional information: All quantum numbers in the parameter list must evaluate to valid names or integers and half-integers ($\geqslant 0$) and must fulfill the proper coupling conditions.
See also: cw_ll(), cw_LSJ(), rcm_jj().

rcm_ll(n,l,N,L,lambda)

Auxiliary procedure to represent a ll-coupled two-particle state $|(nl, NL)\lambda\rangle$ with total orbital angular momentum λ in the relative and center-of-mass coordinates.
Output: An unevaluated call to rcm_ll(n,l,N,L,lambda) is returned.
Argument options: (..., *no_check*) to do *no check* on the consistency of the given quantum numbers; this may save some time in lengthy computations.
Additional information: All quantum numbers in the parameter list must evaluate to valid names or integers ($\geqslant 0$) and must fulfill the proper coupling conditions.
See also: cw_ll(), rcm_LSJ(), rcm_jj().

rcm_LSJ(n,l,N,L,lambda,S,J)

Auxiliary procedure to represent a LSJ-coupled two-particle state $|((nl, NL)\lambda, S)J\rangle$ with orbital angular momentum λ, spin S and total angular momentum J in the relative and center-of-mass coordinates.
Output: An unevaluated call to rcm_LSJ(n, l, N, L, lambda, S, J) is returned.
Argument options: (..., *no_check*) to do *no check* on the consistency of the given quantum numbers; this may save some time in lengthy computations.

Additional information: All quantum numbers in the parameter list must evaluate to valid names or integers ($\geqslant 0$) and must fulfill the proper coupling conditions.
See also: cw_LSJ(), rcm_ll(), rcm_jj().

rcm_jj(n,l,j,N,L,J$_L$,J)

Auxiliary procedure to represent a jj-coupled two-particle state $|(nlj, NLJ_L)J\rangle$ with total angular momentum J in the relative and center-of-mass coordinates.
Output: An unevaluated call to rcm_jj(n, l, j, N, L, J^L, J) is returned.
Argument options: (..., *no_check*) to do *no check* on the consistency of on the given quantum numbers; this may save some time in lengthy computations.
Additional information: All quantum numbers in the parameter list must evaluate to valid names or integers and half-integers ($\geqslant 0$) and must fulfill the proper coupling conditions.
See also: cw_jj(), rcm_ll(), rcm_LSJ().

B.2. Main commands of the FERMI module

The (auxiliary) notation from the last subsection helps the user with a compact but still rather flexible notation for his or her interactive work. For example, a notation like `...,cw_ll`$_a$`,cw_ll`$_b$`,...` means that the user may call the procedures explicitly by `...,cw_ll(n`$_{1a}$`,l`$_{1a}$`,n`$_{2a}$`,l`$_{2a}$`,lambda`$_a$`), cw_ll(n`$_{1b}$`, l`$_{1b}$`,n`$_{2b}$`,l`$_{2b}$`,lambda`$_{2b}$`)...` in the parameter list or that these (unevaluated) calls to cw_ll() are first assign to some variables, say `wa, wb`, and later only these variables appear at input time: `...,wa,wb,...`. To 'extract' the quantum numbers from these unevaluated calls, the command Moshinsky_tabulate() is used internally.

For two-particle states in the cw coordinates, the matrix elements are given algebraically if all the required Moshinsky brackets are stored in *algebraic* form before by using the command Moshinsky_predefine(ρ, *algebraic*). Additionally, one has to use the option *algebraic* in the Fermi_V_...() procedures after specifying the quantum states arguments and before specifying the interaction potential parameters (if any). An exception from this is the procedure Fermi_V_central() which does not need the option *algebraic* but only the algebraic form of the Moshinsky brackets.

Fermi_rho(n$_a$,l$_a$,n$_a$,l$_a$)

Calculates the energy index $\rho = 2n_a + l_a + 2n_b + l_b$ for a (non-interacting) two-particle state in a harmonic potential well.
Output: An expression or a number is returned.
Argument options: (cw_ll$_a$) to calculate the corresponding energy index ρ for a ll-coupled two-particle state in the cw coordinates. ♣ (rcm_ll$_a$) to calculate the corresponding energy index ρ for a ll-coupled two-particle state in the rcm coordinates. ♣ (n$_a$,l$_a$,...,n$_q$,l$_q$) to calculate the corresponding energy index ρ for a non-interacting q-particle state in a harmonic potential well.
Additional information: The energy index ρ can be used to define *shells* of equal energy.
See also: cw_ll(), rcm_ll().

Fermi_V_reduced(n$_a$, l$_a$,n$_b$,l$_b$)

Evaluates the reduced matrix element

$$\langle nl \| V(r) \| n'l' \rangle = \sum_p B(nl, n'l', p) I_p \tag{B.1}$$

in terms of the Talmi integrals I_p [1].

Output: An expression in terms of unevaluated calls to Talmi_I() is returned.
Argument options: (n_a,l_a,n_b,l_b,k,*harmonic*) to calculate the corresponding reduced matrix element for a harmonic interaction $V(r) = -kr^2$; an (algebraic) number or a symbolic expression in terms of the parameter k is returned in this case. ♣ (n_a, l_a, n_b, l_b, V_o, *unity*) to calculate the corresponding reduced matrix element for the a unity interaction $V(r) \equiv V_o$; an (algebraic) number or a symbolic expression in terms of V_o is returned. ♣ ($n_a, l_a, n_b, l_b, V_o, \beta$, *Gaussian*) to calculate the corresponding reduced matrix element for a Gaussian-type interaction $V(r) = -V_o e^{-\beta^2 r^2}$; an (algebraic) number or a symbolic expression in terms of V_o and β is returned. ♣ (n_a, l_a, n_b, l_b, V_o, *rInverse*) to calculate the corresponding reduced matrix element for the interaction $V(r) = -V_o/r$; an (algebraic) number or a symbolic expression in terms of V_o is returned.
Additional information: The values of Talmi_I() can be user defined; for instance, it could be used to return the result of an explicit computation or simply values from a given list. ♣ The B-coefficients are always computed in *algebraic* form if possible.
See also: Talmi_integral().

Fermi_shell()

Generates a list of quantum numbers for all (non-interacting) two-particle states which belong to the same energy shell, i.e. which have the energy index $\rho = 2n_a + l_a + 2n_b + l_b$ (and the same λ).
Output: A list of lists $[[n_{a1}, l_{a1}, n_{b1}, l_{b1}, \lambda], [n_{a2}, l_{a2}, n_{b2}, l_{b2}\lambda], \ldots]$ is returned.
Argument options: (cw_ll) to generate the list with ρ and λ taken from ll-coupled two-particle state in the cw coordinates. ♣ (rcm_ll) to generate the list with ρ and λ taken from ll-coupled two-particle state in the rcm coordinates. ♣ (ρ) to generate the list with the given energy index ρ and with all possible λ values.
Additional information: This procedure is used to generate all possible states with the same energy index. A complete list of such two-particle states is often required to evaluate the transformation bracket and matrix elements. ♣ The quantum numbers from the input are also returned in the output list.
See also: cw_ll(), rcm_ll(), Fermi_rho().

Fermi_tabulate()

Returns a table with all defined quantum numbers of an ll-coupled nuclear two-particle state $|(n_1 l_1, n_2, l_2)\lambda\rangle$ in the cw coordinates.
Output: A table `T` with entries `T[n1_], T[l1_], T[n2_], T[l2_]`, and `T[lambda_]` is returned.
Argument options: (cw_LSJ) to return a table with all defined quantum numbers of a LSJ-coupled state $|((n_1 l_1, n_2 l_2)\lambda, S)J\rangle$ in the cw coordinates. A table with entries `T[n1_], T[l1_], T[n2_], T[l2_] T[lambda_], T[S_]`, and `T[J_]` is returned in this case. ♣ (cw_LSJM) to return a table with all defined quantum numbers of a $LSJM$-coupled state $|((n_1 l_1, n_2 l_2)\lambda, S)JM\rangle$ in the cw coordinates. A table with entries `T[n1_], T[l1_], T[n2_], T[l2_] T[lambda_], T[S_], T[J_]` and `T[M_]` is returned in this case. ♣ (cw_jj) to return a table with all defined quantum numbers of a jj-coupled state $|(n_1 l_1 j_1, n_2 l_2 j_2)J\rangle$ in the cw coordinates. A table with entries `T[n1_], T[l1_], T[j1_], T[n2_], T[l2_], T[j2_]`, and `T[J_]` is returned in this case. ♣ (rcm_ll) to return a table with all defined quantum numbers of a ll-coupled state $|(nl, NL)\lambda\rangle$ in the rcm coordinates. A table with entries `T[n_], T[l_], T[N_], T[L_]`, and `T[lambda_]` is returned in this case. ♣ (rcm_LSJ) to return a table with all defined quantum numbers of a LSJ-coupled state $|((nl, NL)\lambda, S)J\rangle$ in the rcm coordinates. A table with entries `T[n_], T[l_], T[N_], T[L_] T[lambda_], T[S_]`, and `T[J_]` is returned in this case. ♣ (rcm_jj) to return a table with all defined quantum numbers of a jj-coupled state $|(nlj, NLJ_L)J\rangle$ in the rcm coordinates. A table with entries `T[n_], T[l_], T[j_], T[N_], T[L_], T[JL_]`, and `T[J_]` is returned in this case.
Additional information:
See also: Moshinsky_bracket().

Fermi_w3j()

Calculates the Wigner $3-j$ symbol.
Output: A (floating-point) number is returned.
Argument options: (..., *algebraic*) to return the Fermi_w3j function in an algebraic form.
Additional information: It was defined merely for internal use. For extended use of the Racah algebra we recommend the use of RACAH package.
See also: Fermi_w6j(), Fermi_w9j().

Fermi_w6j()

Calculates the six-j-symbol coefficients.
Output: A (floating-point) number is returned.
Argument options: (..., *algebraic*) to return the Fermi_w6j function in an algebraic form.
Additional information: It was defined merely for internal use. For extended use of the Racah algebra we recommend the use of RACAH package.
See also: Fermi_w3j(), Fermi_w9j(), RacahW().

Fermi_w9j()

Calculates the nine-j-symbol coefficients.
Output: A (floating-point) number is returned.
Argument options: (..., *algebraic*) to return the Fermi_w9j function in an algebraic form.
Additional information: It was defined merely for internal use. For extended use of the Racah algebra we recommend the use of RACAH package.
See also: Fermi_w3j(), Fermi_w6j().

Fermi_V_central(cw_ll$_1$,cw_ll$_2$)

Evaluates the (two-particle) reduced matrix element $\langle (n_1 l_1, n_2 l_2)\lambda \| V \| (n_1' l_1', n_2' l_2')\lambda \rangle$ for an arbitrary central potential $V(r)$.
Output: An expression in terms of the Talmi integrals I_p, a (floating-point) number or a symbolic expression is returned.
Argument options: (cw_ll$_1$, cw_ll$_2$, V_o, *unity*) to return the same matrix element for a constant interaction $V(r) \equiv V_o$; a (floating-point) number or a symbolic expression in terms of V_o is returned in this case. ♣ (cw_ll$_1$, cw_ll$_2$,k,*harmonic*) to return the same matrix element for a harmonic interaction $V(r) = -kr^2$; a (floating-point) number or a symbolic expression in terms of k is returned. ♣ (cw_ll$_1$,cw_ll$_2$,V_o,beta,*Gaussian*) to return the same matrix element for a Gaussian-type interaction $V(r) = -V_o \mathrm{e}^{-\beta^2 r^2}$; a (floating-point) number or a symbolic expression in terms of V_o and β is returned. ♣ (cw_ll$_1$,cw_ll$_2$,V_o,*rInverse*) to return the same matrix element for the interaction $V(r) = -V_o/r$; a (floating-point) number or a symbolic expression in terms of V_o is returned.
Additional information: To evaluate the matrix elements in the cw coordinates, the two-particle states are first transformed into the rcm system by using the Moshinsky brackets and then the matrix elements for the rcm functions are utilized. ♣ For two-particle states in the cw coordinates, the matrix elements are given algebraically if all the required Moshinsky brackets are stored in *algebraic* form before by using the command Moshinsky_predefine(ρ, *algebraic*). ♣ Internally, the evaluation of the matrix elements is based on Ref. [2], Eqs. (4.7)–(4.9).
See also: Fermi_V_spin_orbit(), Fermi_V_tensor_X2(), Moshinsky_predefine().

Fermi_V_spin_orbit(cw_LSJM$_1$,cw_LSJM$_2$)

Evaluates the (two-particle) reduced matrix element $\langle ((n_1 l_1, n_2 l_2)\lambda, S) J, M \| V(r) \mathbf{l}\cdot\mathbf{s} \| ((n_1' l_1', n_2' l_2')\lambda', S') J' M' \rangle$ for a central potential $V(r)$.

Output: An expression in terms of Talmi integrals I_p, a (floating-point) number or a symbolic expression is returned.
Argument options: (cw_LSJM$_1$,cw_LSJM$_2$,V_o,*unity*) to return the same matrix element for a constant interaction $V(r) \equiv V_o$; a (floating-point) number or a symbolic expression in terms of V_o is returned in this case. ♣ (cw_LSJM$_1$,cw_LSJM$_2$,k,*harmonic*) to return the same matrix element for a harmonic interaction $V(r) = -kr^2$; a (floating-point) number or a symbolic expression in terms of k is returned. ♣ (cw_LSJM$_1$,cw_LSJM$_2$,V_o,beta, *Gaussian*) to return the same matrix element for a Gaussian-type interaction $V(r) = -V_o \mathrm{e}^{-\beta^2 r^2}$; a (floating-point) number or a symbolic expression in terms of V_o and β is returned. ♣ (cw_LSJM$_1$,cw_LSJM$_2$,V_o,*rInverse*) to return the same matrix element for the interaction $V(r) = -V_o/r$; a (floating-point) number or a symbolic expression in terms of V_o is returned. ♣ (cw_LSJM$_1$,cw_LSJM$_2$,*algebraic*) to evaluate the matrix elements in algebraic form as far as possible.
Additional information: The matrix elements are given algebraically if all the required Moshinsky brackets are stored in *algebraic* form before by using the command Moshinsky_predefine(ρ, *algebraic*). ♣ Internally, the evaluation of the matrix elements is based on Ref. [2], Eqs. (4.36). ♣ To obtain an algebraic result also for some predefined potential, the parameters for the potential msut be entered after the option *algebraic* as, for instance, in: Fermi_V_spin_orbit(cw_LSJM$_1$,cw_LSJM $_2$, *algebraic*,V_o,beta,*Gaussian*).
See also: Fermi_V_central(), Fermi_V_tensor_X2(), Moshinsky_predefine().

Fermi_V_tensor_X2(cw_LSJM$_1$,cw_LSJM$_2$)

Evaluates the (two-particle) reduced matrix element $\langle((n_1 l_1, n_2 l_2)\lambda, S)J, M|V_T|((n'_1 l'_1, n'_2 l'_2)\lambda', S')J', M'\rangle$ for the tensorial interaction $V_T = (\frac{32}{5}\pi)^{1/2} V(r)\mathbf{Y}_2 \cdot \mathbf{X}_2$ where $\mathbf{X}_2$ is supposed to be a second-rank tensor with zero-component $X_{20} = \frac{1}{\sqrt{2}}(3s_z^2 - s^2)$ (see (A.19), (14)).
Output: An expression in terms of Talmi integrals I_p, a (floating-point) number or a symbolic expression is returned.
Argument options: (cw_LSJM$_1$,cw_LSJM$_2$,V_o,*unity*) to return the same matrix element for a constant interaction $V(r) \equiv V_o$; a (floating-point) number or a symbolic expression in terms of V_o is returned in this case. ♣ (cw_LSJM$_1$,cw_LSJM$_2$,k,*harmonic*) to return the same matrix element for a harmonic interaction $V(r) = -kr^2$; a (floating-point) number or a symbolic expression in terms of k is returned. ♣ (cw_LSJM$_1$,cw_LSJM$_2$,V_o,beta, *Gaussian*) to return the same matrix element for a Gaussian-type interaction $V(r) = -V_o \mathrm{e}^{-\beta^2 r^2}$; a (floating-point) number or a symbolic expression in terms of V_o and β is returned. ♣ (cw_LSJM$_1$,cw_LSJM$_2$,V_o,*rInverse*) to return the same matrix element for the interaction $V(r) = -V_o/r$; a (floating-point) number or a symbolic expression in terms of V_o is returned.
Additional information: The matrix elements are given algebraically if all the required Moshinsky brackets are stored in *algebraic* form before by using the command Moshinsky_predefine(ρ, *algebraic*). ♣ Internally, the evaluation of the matrix elements is based on Ref. [2], Eqs. (4.50)–(4.51). ♣ To obtain an algebraic result also for some predefined potential, the parameters for the potential msut be entered after the option *algebraic* as, for instance, in: Fermi_V_tensor_X2(cw_LSJM$_1$,cw_LSJM$_2$,*algebraic*,V_o,beta,*Gaussian*).
See also: Fermi_V_central(), Fermi_V_spin_orbit(), Moshinsky_predefine().

Moshinsky_bracket(rcm_ll,cw_ll)

Calculates the Moshinsky bracket $\langle(nl, NL)\lambda|(n_1 l_1, n_2 l_2)\lambda\rangle$, i.e. the transformation coefficient in going from the center-of-well (cw) to the relative and center-of-mass (rcm) coordinates.
Output: A (floating-point) number is returned.
Argument options: (..., *algebraic*) to return the same transformation bracket in algebraic form. ♣ (..., *symbolic*) to return the same transformation bracket as a sum of unevaluated Wigner $6-j$ symbols denoted by w6j_().
Additional information: Moshinsky's transformation brackets are nonzero only if the two-particle states on the left- and right-hand side belong to the same energy shell and have the same lambda.

See also: Moshinsky_orthogonality(), Moshinsky_predefine().

Moshinsky_orthogonality()

Tests the orthogonality of the Moshinsky brackets according to (A.5) and (A.6).
Output: A (floating-point) number is returned.
Argument options: (rcm_ll$_a$,rcm_ll$_b$) to return the result A according to (A.5) ♣ (cw_ll$_a$,cw_ll$_b$) to return the result A according to (A.6).
Additional information: The procedure is meant for checking the orthogonality of the (stored) Moshinsky brackets.
See also: Moshinsky_bracket(), Moshinsky_predefine().

Moshinsky_predefine(ρ_0)

Calculates and stores all Moshinsky brackets $\langle (nl, NL)\lambda | (n_1 l_1, n_2 l_2)\lambda \rangle$ up to given maximal energy index ρ_0.
Output: A NULL expression is returned.
Argument options: (ρ_0,*algebraic*) to return the same transformation brackets in algebraic form.
Additional information: This is an auxiliary procedure to calculate all the required brackets once; it can be used, for instance, together with the evaluation of matrix elements by the procedures Fermi_V_...() in order to accelerate their computation. ♣ The procedure builds a list of lists [[rcm_ll(), cw_ll(), Moshinsky_bracket(rcm_ll(), cw_ll())], [...], ...] and stores it in the global variable Moshinsky_store_. It also define another global variable Moshinsky_level_, which is used by most other procedure whenever they make a call to Moshinsky_bracket(rcm_ll(),cw_ll()). ♣ The value for a Moshinsky bracket is read from the global variable Moshinsky_store_ if Moshinsky_level_ $\leqslant$ Fermi_rho(rcm_ll()). ♣ A parameter $0 \leqslant \rho_0$ must be given in order to compute all the Moshinsky brackets up to the energy index ρ_0. ♣ The range of pre-calculated Moshinsky brackets can be enlarged by a later call to this procedure; in this case, all the global variables are reset to their current values. ♣ When the Moshinsky brackets are stored in *algebraic* form, one can compute the matrix elements by means of the procedures Fermi_V_...() in either algebraic or symbolic form, respectively.
See also: Moshinsky_bracket().

RacahW()

Calculates the Racah W function. This is equivalent to the $6-j$ symbol up to a phase and the order of arguments:

$$\mathit{Fermi\_w6j}(j_1, j_2, j_3, j_4, j_5, j_6) \equiv (-1)^{j_1+j_2+j_4+j_5} \cdot \mathrm{Racah}\, W(j_1, j_2, j_5, j_4; j_3, j_6). \tag{B.2}$$

Output: A (floating-point) number is returned.
Argument options: (..., *algebraic*) to return the Racah W function in an algebraic form.
Additional information: It was defined merely for internal use. For extended use of the Racah algebra we recommend you to use Fermi_w6j() or the RACAH package.
See also: Moshinsky_...(), Fermi_V_...(), Fermi_w6j().

Talmi_integral(p)

Calculates the Talmi integral according to (10) for a given potential.
Output: A number or an unevaluated call to Talmi_I(p) is returned.
Argument options: (p, V_o, *unity*) to return the Talmi integral for a constant interaction $V(r) \equiv V_o$; a (floating-point) number or a symbolic expression in terms of V_o is returned in this case. ♣ (p,k,*harmonic*) to return the Talmi integral for a harmonic interaction $V(r) = -kr^2$; a (floating-point) number or a symbolic expression in terms of k is returned. ♣ (p,V_o,beta,*Gaussian*) to return the the Talmi integral for a Gaussian-type interaction $V(r) = -V_o \mathrm{e}^{-\beta^2 r^2}$; a (floating-point) number or a symbolic expression in terms of V_o and β is returned. ♣ (p,V_o,*rInverse*)

to return the Talmi integral for the interaction $V(r) = -V_o/r$; a (floating-point) number or a symbolic expression in terms of V_o is returned.
Additional information: The user can provide an additional procedure Talmi_I() in order to make an unevaluated call explicit. ♣ A floating-point number is returned whenever the parameters of the potentials are given by floating-point numbers; otherwise an algebraic number is returned.
See also: Fermi_V_central(), Fermi_V_spin_orbit(), Fermi_V_reduced().

References

[1] I. Talmi, Helv. Phys. Acta 25 (1952) 185.
[2] T.A. Brody, M. Moshinsky, Tables of Transformation Brackets, Monografias del Instituto de Fisica, Universidad Nacional Autonoma de Mexico, 1960.
[3] A. Lejeune, J.P. Jeukenne, Comput. Phys. Comm. 2 (1971) 231.
[4] M. Sotona, M. Gmitro, Comput. Phys. Comm. 3 (1972) 53.
[5] O. Zohni, Comput. Phys. Comm. 3 (1972) 61.
[6] D.H. Feng, T. Tamura, Comput. Phys. Comm. 10 (1975) 87.
[7] J. Dobes, Comput. Phys. Comm. 16 (1979) 373.
[8] A. Etchegoyen, M.C. Etchegoyen, E.G. Vergini, Comput. Phys. Comm. 55 (2) (1989) 227.
[9] G. Fieck, J. Phys. B: At. Mol. Phys. 12 (1979) 1063.
[10] P. Das, Comput. Phys. Comm. 55 (2) (1989) 177.
[11] D. Redfern, The Maple Handbook, Springer, New York, Berlin, a.o., 1996.
[12] S. Fritzsche, Comput. Phys. Comm. 103 (1997) 51;
S. Fritzsche, S. Varga, D. Geschke, B. Fricke, Comput. Phys. Comm. 111 (1998) 167.
[13] G. Gaigalas, S. Fritzsche, B. Fricke, Comput. Phys. Comm. 135 (2001) 219.
[14] T. Inghoff, S. Fritzsche, B. Fricke, Comput. Phys. Comm. 139 (2001) 297.
[15] S. Fritzsche, T. Inghoff, T. Bastug, B. Fricke, Comput. Phys. Comm. 139 (2001) 314.