Revista Mexicana de Física 22 (1973) 199 - 214

THE BRASHINSKET AND ITS COMPUTATION*

T. A. Brody**

Instituto de Física, Universidad Nacional de México

(Recibido: mayo 15, 1973)

ABSTRACT: The principal computational schemes for obtaining the harmonic oscillator transformation bracket (or brashinsket) are briefly described and compared as regards speed. The technique of building a table of already calculated values for reuse is then discussed; when used with a recursive computational scheme, this technique is better by a factor of 10 or more in an extensive calculation.

I

The nuclear shell model is useful only to the extent that explicit calculations can be carried out. One of the most convenient forms of doing this is to expand the model wave functions in terms of harmonic-oscillator wave functions, because these have the remarkable property that a two-parti-

* Work supported in part by the Consejo Nacional de Ciencia y Tecnología, México.

** Technical Adviser, Instituto Nacional de Energía Nuclear, México.

cle function is a finite sum over products of centre-of-mass and relative-coordinate functions which are again harmonic-oscillator functions. (The group theory behind this property has been extensively discussed by Moshinsky[1]). Since the residual interactions between the particles may always (by means of fractional-parentage coefficients or other methods) be expressed in terms of two-particle interactions, this property now yields a great simplification in the calculation of a matrix element: after the transformation to centre-of-mass and relative coordinates, one can integrate immediately over the first; this avoids having to expand the potential in terms of Legendre polynomials and then compute the Slater integrals of the expansion coefficients – which may be all but impossible in nuclear physics.

The advantages of postulating a harmonic-oscillator well as common potential in the shell model were first pointed out by Talmi[2]; his ideas for simplifying the evaluation of the Slater coefficients as found in the standard methods of atomic spectroscopy[3] were later supplemented by Thieberger's tables[4]. But the real advantages of Talmi's idea began to show up once it appeared that the Slater coefficients could be bypassed altogether, - that in fact the harmonic oscillator permitted connecting the two-particle matrix element directly to the integrals in terms of which Talmi had expressed the Slater coefficients and which now bear his name. This step was taken by Moshinsky[5]. He found explicit recursion relations for what he called harmonic-oscillator transformation brackets, by means of which it was possible to tabulate them numerically[6, 7*]. The technique was completed by the B coefficient[5] for expanding the reduced matrix element of the two-body force in terms of Talmi integrals. In other words, we can write the matrix element (in LS coupling) of any residual interaction as a finite sum

$$< n_1 l_1 n_2 l_2 \lambda\mu \mid V \mid n_1' l_1' n_2' l_2' \lambda' \mu > = \sum_p C_V(p)\, I_p \quad . \tag{1}$$

Here I_p is the Talmi integral,

$$I_p = \frac{2}{\Gamma(p+3/2)} \int_0^\infty v(r)\, e^{-r^2} r^{2p+2}\, dr \tag{2}$$

* The neat contraction "brashinsket" for Moshinsky's transformation bracket was suggested by Dr. J.M. Lozano, and will be used here.

in which $v(r)$ is the radial dependence of the potential V. The coefficients $C_V(p)$ depend on the nature of the potential, but not on $v(r)$, and are in their turn finite sums over products of the brashinskets and the B coefficients, of a form that can be computed quite systematically. Thus for a central force we have

$$C_V \equiv C_c(n_1 l_1 n_2 l_2; n_1' l_1' n_2' l_2'; \lambda; p)$$

$$= \delta_{\lambda\lambda'} \sum_{nn'lNL} \langle n_1 l_1 n_2 l_2 \lambda | nlNL\lambda \rangle B(nl, n'l, p) \langle n'lNL\lambda | n_1' l_1' n_2' l_2' \lambda \rangle \tag{3}$$

and for other types of force there are similar but somewhat more complicated expressions. (A small table of the central-force coefficients has been computed explicitly.[9]) These topics are pursued and the details of the notation explained in other publications[1,7,8]; in this paper I propose to discuss the techniques of numerical computation of the brashinskets, without which the theory remains sterile.

The concept of the brashinsket has had a wide range of applications both inside and outside nuclear physics, and it is natural that it should have received various formulations, some of them based on group-theoretical considerations. A brief sketch of the more important formulations will provide the background needed for the discussion of the computational techniques.

That the evaluation of the Slater coefficients could be avoided was implicit in Talmi's work[2], and thus it was natural that Moshinsky and his group in Mexico were not the only ones to hit on the idea of obtaining explicitly the coefficients of the transformation from the centre of the potential well to the two-particle centre-of-mass and relative coordinates. Within a year of the appearance of Moshinsky's first paper[5], three other groups had found almost identical results[10,11,12]. None of them had obtained anything analogous to Moshinsky's quite general recursion relations; they derived spacial values and certain fairly limited recursion relations. However, Balashov and Eltekov[11] introduced a useful generalisation: they antisymmetrised the total wave function of the two particles, so that their transformation coefficients depend on the total spin and isospin as well. The same idea is mentioned by Arima and Terasawa[12], though it is not fully worked out. A clear discussion of the use of the Balashov–Eltekov coefficients and their relation to the brashinskets will be found in ref. 13, vol. II, ch. 6. None of these authors studied the remaining problem of expressing the relative-coordinate matrix element in terms of Talmi integrals, which gave rise to the B coefficient.

The brashinskets make possible the transformation to relative coordinates for particles of equal mass only; yet in many cases it is necessary to do this for particles of unequal masses. The generalisation to unequal masses was introduced by Smirnov[14]; several versions of a closed expression for the unequal-mass brashinsket have been given[15, 16, 17]. It may also be expressed in terms of the equal-mass ones[18], which yields a convenient computational procedure. The final step was taken by Kramer[19], who both found a simple closed formula for the unequal-mass brashinsket and extended the concept rather neatly to more than two particles.

Such developments became possible once the group-theoretical basis of the brashinsket was better understood. For two particles, the harmonic-oscillator Hamiltonian has the symmetry group $U(6)$; if we consider that the particle index (with values 1, 2) defines a two-dimensional space, then this group has a $SU(2)$ subgroup which acts on the particle-index space[20, 21]. The transformation from centre-of-well to centre-of-mass and relative coordinates then corresponds to a rotation in this group; the brashinskets, as the transformation coefficients of this rotation, are thus seen to constitute an in general reducible representation of $SU(2)$, characterised by ρ and λ, where

$$\rho = 2n_1 + l_1 + 2n_2 + l_2 = 2n + l + 2N + L \qquad (4)$$

and λ is, of course, the total angular momentum; $n_1 l_1 n_2 l_2$ and $nlNL$ are the (composite) row and column indices. For two particles only rotations in a plane are meaningful; if we write the Euler angles as $(0, \beta, 0)$, then for $\beta = \pi/4$ we have the equal-mass brashinskets, and in general the mass ratio of the two particles is $\tan^2\beta$. Apart from giving rise to the developments already mentioned, this approach allows one to deduce the symmetry relations of the brashinskets in a unified form[20, 22], as well as to find a series of sum rules often useful either for checking purposes or to simplify cumbersome algebraic expressions[23].

In what follows I will first of all briefly discuss several computing schemes to be found in the literature, of which the first two employ recursion relations, and then describe a technique I have adapted to the computation of the brashinskets which gives the two recursive methods a considerable advantage over the closed-formula type in extensive computations.

Since this paper is to appear in a special number of the Revista Mexicana de Física in honour of Prof. Marcos Moshinsky, a historical note may not be out of place: It was Marcos' merit to have recognised that the concept of the brashinsket was a most useful one which other research workers would develop[24] if we did not, and it was his infectious enthusiasm

which enlisted the help of many others in the theoretical group in Mexico for an enterprise which began with the brashinskets and in a few years ramified into the most diverse applications of group theory. Fortunately Mexico's first electronic computer, an IBM 650, had recently been installed at the University, and was still not saturated with work. After a hectic period of programming and formula checking, we could therefore take up over 200 hours of machine time to tabulate the brashinskets[7]. Even so, we had to work nights for several weeks, and during that time I still remember with great pleasure how Marcos would appear at half-past one in the morning bearing sandwiches and encouragement, while we were coaxing a recalcitrant computer with a few kicks in strategic places... The machine served us so well that by the time a second edition of the tables became necessary (published by Gordon and Breach in 1967), no errors had been detected – only a card dropped out before printing. Not many later machines could show such a record of reliability.

II

The method originally developed by Moshinsky[5] to compute the brashinskets is based on a recursion relation in the two quantum numbers n_1 and n_2, namely

$$\langle n l N L \lambda \,|\, n_1 + 1 \, l_1 n_2 l_2 \lambda \rangle = [(n_1+1)(n_1+l_1+\tfrac{3}{2})]^{-\frac{1}{2}} \times$$

$$\times \sum_{n' l' N' L'} \langle n l N L \lambda \mu \,|\, -r_1^2 \,|\, n' l' N' L' \lambda \mu \rangle \langle n' l' N' L' \lambda \,|\, n_1 l_1 n_2 l_2 \lambda \rangle \tag{5}$$

and the analogous relation in n_2. This recursion is much simpler to apply than appears at first sight, because the indices of the sum in (5) are restricted by the constancy of the energy, eq. (4), and the selection rules for angular momenta, so that at most six different terms survive; for these the matrix element of $-r_1^2$ in (5) can be evaluated quite easily by means of the well known recursion for the Laguerre polynomials[25].

By means of (5) and its n_2 analogue, one reduces the computation of any brashinsket to that for the case $n_1 = n_2 = 0$; for this Moshinsky was able quite straightforwardly to derive a closed expression,

$$<nlNL\lambda|\,0l_1 0l_2\lambda> = C\sum_x A(l_1 l l_2 L x) \begin{Bmatrix} l & L & \lambda \\ l_2 & l_1 & x \end{Bmatrix} , \qquad (6)$$

where C is a normalisation factor and A is a quantity the evaluation of which involves a single sum over a finite number of terms.

For the tabulation of the brashinskets, the use of (5) and (6) was quite convenient. First of all, an extensive table of the brashinskets with $n_1 = n_2 = 0$ was computed and checked by means of the symmetry and orthogonality relations; then the tables for n_1 and n_2 greater than 0 were successively computed. For $n_1 + n_2 > 1$ more than one path leads to the desired value; this fact served as a check and was also exploited in order to reduce the accumulated rounding errors by averaging over the different results.

The situation is, however, quite different when one does a machine computation of matrix elements, where the brashinskets are usually needed in an order that is both difficult to foresee and impossible to arrange so that the earlier values can be used later on in the recursion relations. As a result, machine time can become very large; for instance, 6.1×10^4 multiplication times would be needed on the average for n_1 and $n_2 = 5$. Even on a fast machine, a more or less complex shell-model computation needing, perhaps, 3000 brashinskets could thus take nearly an hour: not an economic proposition.

A first attempt to circumvent this difficulty is due to Baranger and Davies[26], who developed recursion relations in l. For $l = 0$ they obtain (I have adapted their notation somewhat)

$$<n0NL\lambda|\,n_1 l_1 n_2 l_2\lambda> = C\sum_{a,k} R(n_1 l_1 n_2 l_2 nNak)\,P(l_1 l_2 Lak)\,\delta_{\lambda L} , \qquad (7)$$

C is again a normalisation factor, P is a complicated but straightforward product of factorials, while R contains a double summation. For $l > 0$ they develop a recursion relation similar to an earlier one[12] in l. It is essentially of the form (removing some trivial errors in their expression (4.1))

$$< nl + 1NL\lambda \,|\, n_1 l_1 n_2 l_2 \lambda > = e \sum_{\lambda'} (2\lambda' + 1) \begin{Bmatrix} 1 & l & l+1 \\ L & \lambda & \lambda' \end{Bmatrix} \times$$

$$\times \left[a(\lambda') < nlNL\lambda' \,|\, n_1 l_1 n_2 - 1 l_2 + 1 \lambda' > + b(\lambda') < nlNL\lambda' \,|\, n_1 l_1 n_2 l_2 - 1 \lambda' > + \right.$$

$$\left. + c(\lambda') < nlNL\lambda' \,|\, n_1 - 1 l_1 + 1 n_2 n_2 \lambda' > + d(\lambda') < nlNL\lambda' \,|\, n_1 l_1 - 1 n_2 l_2 \lambda' > \right] , \tag{8}$$

where the coefficients a, b, c, d each involve a $6j$ coefficient. Eq. (8) is very convenient for $l = 0$, since then only one term appears in the sum, $\lambda' = \lambda$; but for higher values of l, they prefer to derive a closed expression rather like (7), though with a much more complicated function Q instead of P; the summation involved is ninefold.

When one works with short-range forces, only small l values enter; but for other types of forces the computation time rises again quite rapidly: for $l = 5$, Baranger and Davies give data that correspond to approximately 4.25×10^4 multiplication times. I have found average times some 16% higher, since I allowed l_1 and l_2 to range over larger values than they did. Both the methods described so far can thus consume excessive computer time.

An entirely different computational scheme, due to Brink [27], therefore becomes of interest. He noted that most of the complexity of computing the brashinskets is due to the fact that the two kets $|n_1 l_1 m_1 >$ and $|n_2 l_2 m_2 >$ must be coupled to give a total angular momentum λ and similarly on the other side for $< nlm|$ and $< NLM|$. Making this coupling explicit by means of Clebsch-Gordan coefficients, we have

$$< nlNL\lambda \,|\, n_1 l_1 n_2 l_2 \lambda > < l_1 l_2 m_1 m_2 \,|\, \lambda\mu >$$

$$= \sum_{mM} < nlm, NLM \,|\, n_1 l_1 m_1 , n_2 l_2 m_2 > < lLmM \,|\, \lambda\mu > . \tag{9}$$

To compute the "uncoupled" brashinsket on the right-hand side of (9), Brink uses the complex form of cylindrical coordinates ($x + iy$, $x - iy$ and z): if η_i, $i = +, -, 0$, are the creation operators in these coordinates, then a one-particle ket may be written

$$|pqr> = (p!q!r!)^{-\frac{1}{2}} \eta_0^p \eta_+^q \eta_-^r |0> ,$$

and the transformation bracket to spherical coordinates is defined by

$$|nlm> = \sum_r <pqr|nlm>|pqr> , \; p = 2n + l - m - 2r , \; q = m + r . \tag{10}$$

The uncoupled brashinsket in (9) then becomes

$$<nlm, NLM | n_1 l_1 m_1, n_2 l_2 m_2> =$$

$$= \sum_{rRr_1r_2} <pqr|nlm><PQR|NLM><pqr, PQR|p_1 q_1 r_1, p_2 q_2 r_2> \times$$

$$\times <p_1 q_1 r_1 | n_1 l_1 m_1><p_2 q_2 r_2 | n_2 l_2 m_2> . \tag{11}$$

But since η_0, η_+ and η_- commute with each other, the cylindrical brashinsket in (11) factorises into three one-dimensional brashinskets which are all of the same type:

$$<pqr, PQR | p_1 q_1 r_1, p_2 q_2 r_2> = <pP|p_1 p_2><qQ|q_1 q_2><rR|r_1 r_2> . \tag{12}$$

These one-dimensional brashinskets are very easy to compute; they involve only one sum, as do the transformation brackets of (10). In (9) also only one summation is needed, since $m_1 + m_2 = M + m$. Putting together eqs. (9) to (12), one obtains a closed expression for the brashinsket. (I have simplified Brink's notation somewhat.)

This method is very elegant. Computationally it offers a definite advantage when, as sometimes happens, one wants the uncoupled brashinsket of (9) rather than the ordinary coupled one.*

* This was rediscovered by Richardson, Shapiro and Malik[28], whose coefficients c_l and t_l are simply the appropriate sums over uncoupled brashinskets and B coefficients for central and tensor forces; they seem to have been unaware of previous work and thus become involved in great algebraic complexity by deriving the coefficients directly in spherical coordinates.

A quite similar method was used by Smirnov[14] in deriving uncoupled unequal-mass brashinskets; however, he worked in Cartesian coordinates, which has the computational disadvantage that the transformation brackets from Cartesian to spherical coordinates are real only when n_y is even.

For the coupled brashinsket, additional computing time is required in Brink's method to find the Clebsch-Gordan coefficients. Furthermore, one must choose m_1 and m_2 such that the Clebsch-Gordan coefficient on the left of (9) is not zero. This should always be possible. However, the computing time depends on the choice of m_1 and m_2, and it may also happen that the accuracy of the result suffers from an unsuitable choice of these numbers. It would be agreeable if a fast algorithm could be found for optimising this choice.

Experiments carried out with Brink's method have shown that computing times for it fall roughly between those of the two recursive methods described above. I selected for these experiments a test set of brashinskets by choosing combinations of n_1, l_1, n_2 and l_2 such that $\rho = 0, 2, 6, 12$ and computed all possible brashinskets for these combinations. The average times for computing a brashinsket for the three computational schemes were as follows (as approximate multiples of the machine's floating-point multiplication time, which will be called τ in what follows; this enables comparisons to be made between different computers):

ρ	Moshinsky	Baranger-Davies	Brink
0	47	112	419
2	9760	9010	11700
6	22900	20300	29600
12	98200	115400	83300

This test is, in a sense, unfair to all three methods, because it does not take into account any of their peculiar advantages; but it shows that none of them is satisfactory as a single, general-purpose algorithm.

A slight improvement on Brink's method is to tabulate at the start the transformation brackets of (10) and the one-dimensional brashinskets of (12), so that the computation is reduced to carrying out the sum in (11) and then recoupling with two Clebsch-Gordan coefficients, as before[29]. The tables needed for this purpose remain reasonably small and exceed $4k$ words in size only when $\rho > 19$ (on the supposition that one word is being used for the argu-

ment values, another for the function value). A not very complete experiment with this method suggests that the time saving is of the order of 25% once the tables have been computed. Better table-lookup procedures might improve this a good deal.

An entirely different computational scheme has been proposed by Talman and Lande[30]. They observe that, using as basis the relative-coordinate wave functions $|nlNL\lambda>$, the matrix that diagonalises the operators h_1, h_2, l_1^2 and l_2^2 – the single-particle Hamiltonian and angular-momentum operators in the original coordinates – is unitary and its elements are, apart from phase factors, the brashinskets. By suitable choice of the four coefficients, one can build an operator

$$H = a_1 h_1 + a_2 h_2 + b_1 l_1^2 + b_2 l_2^2$$

which has no degeneracies; Talman and Lande obtain explicitly its matrix elements and then simply diagonalise numerically the resulting matrix so as to get its eigenvectors, taking advantage of the fact that the eigenvalues are already known. This method does, indeed, yield a considerable improvement in speed so long as ρ remains relatively small. But for $\rho > 20$ the matrix to be diagonalised may easily exceed 400×400, and quite apart from the questions of machine time and memory size, the accumulation of rounding errors tends to affect the orthogonality of the eigenvectors. To get satisfactory results one must use the rather slower Jacobi diagonalisation method[31] or calculate certain critical eigenvectors by inverse iteration[32]. Because of these problems, I have not made any extensive experiments concerning the computing times for this method.

For completeness' sake I may mention that there exist at least two further approaches which have yielded very similar closed formulae for the unequal-mass brashinsket. One is due to Bakri[16], who points out that the wave functions $|n_1 l_1 n_2 l_2 \lambda>$ transform into centre-of-mass and relative coordinates exactly as the much simpler functions

$$\Phi(r_1, r_2) = A(n_1 l_1) A(n_2 l_2)\, r_1^{2n_1 + l_1}\, r_2^{2n_2 + l_2} [Y_{l_1}(\hat{r}_1)\, Y_{l_2}(\hat{r}_2)]_{\lambda\mu}$$

do; he then obtains by algebraic transformations the transformation brackets for these functions. The other approach (Kumar[15] and, independently, Buck[17]) is the use of generating functions for the polynomials $r^l L_n^{l+\frac{1}{2}}(r^2)$; simple re-

lations between these generating functions may be derived, which lead to closed expressions for the brashinskets. These do not, however, seem to offer any special advantages over the formulae already discussed.

III

In recent years the size of computer memories available to research in physics has greatly increased, and beyond the growth in physical capacity the development of effective swapping monitors has put almost unlimited virtual memory space at the disposal of the programmer. This alters the tradeoff balance between execution time and program size, and I have accordingly reprogrammed the computation of the brashinsket as a memo-function[33]: in other words, a procedure which keeps a table of function values computed earlier, updates it dynamically, and does a table look-up rather than recompute the function whenever a repetition of earlier argument values is called for.

In order to identify correctly which function value is needed (or to decide that it is not yet in the table), both arguments and corresponding values must be stored. Of the 9 arguments the brashinsket has, one is redundant, because of relation (4). The remaining 8 can, in most machines, be packed into one computer word. If one of the angular momenta is eliminated, a reasonable choice would be L, which leaves 4 radial quantum numbers and 4 angular momenta. I have found it convenient to require the angular momenta to be less than a limit K, and the radial numbers to be less than another limit M; I then define a packing function

$$f = M(M(K(K(K(K(Mn + N) + l_2) + l_1) + l) + \lambda) + n_1) + n_2 \quad . \qquad (13)$$

For a 36-bit machine, $M = 18$ and $K = 28$ are suitable limits if the sign bit can be used; this may mean that the packing routine must be coded in machine language, but since just one added bit increases the limits by about 20% this is well worthwhile. On the Burroughs 6500 (which is a 48-bit machine but uses only 39 bits to represent integers) I have used $M = 23$ and $K = 37$, and so far the Mexican group has not needed a brashinsket beyond these limits.

The size of the table may be adjusted as needed in a full Algol system which has dynamic own arrays[34]; unfortunately this feature is often not implemented. Where it exists, the monitor will usually need to swap the

program out onto discs and remap on swapping back; hence one should increase the table size in fairly blocks, $-\frac{1}{2}k$ words, say – so as to reduce the overhead.

Where fixed-size tables must be used*, it may be useful as an indication that the Mexican group has usually found a table of $2k$ arguments (packed according to (13)) and $2k$ function values to be sufficient for shell-model calculations up to about $\Lambda = 100$.

A considerable reduction in the necessary table size is achieved by making use of the known symmetries of the brashinsket (see e.g. ref. 5). They permit exchanging the arguments until a standard order is achieved, for instance that defined by the inequalities

$$n_1 + n_2 < n + N$$

$$l_1 < l_2$$

$$l < L \ .$$

Since a linear table search is rather time-consuming, I decided to use a hash-coded table look-up[36, 37]. The hash code (i.e. the index of the first place in the table that is examined) is obtained as the modulus of the packing function with respect to the largest prime number smaller than the table size; if the value at this point is not the required one, successive table entries are examined until either the value is found or an empty place is detected, which indicates that the arguments looked for are not in the table. The table is used in a circular fashion.

This scheme makes it necessary to start with a fairly large table size, since the efficient use of a very much expanded table would mean remapping it – a time-wasting operation. After some experimentation, I have used a table size of 2048, hash-coded modulo 2039, for some years now.

Looking at table entries beyond the first one is necessary to resolve the collisions that may occur between different argument sets in this method (known as open addressing). If the table is less than two-thirds full, a hash-coding procedure that effectively randomises the search argument, will require

* It is worth noting that on the PDP-10 dynamic own arrays have been implemented not only in Algol, but also in Fortran[35].

on the average only one more entry to be looked at[38].

Better search methods are now known (see references given in ref. 37); the table-search time is however sufficiently small compared to the computation time that I have not felt it necessary to improve in the search technique.

Of the three computational schemes described in section II, the two recursive methods are clearly better than Brink's for use with the memo-function technique, since they permit saving time in the calculation of a new brashinsket by looking up in the table those brashinsket that are needed in the recursion. The modification by Chasman and Wahlborn[29] of Brink's method can also be made to work along the lines described; two memo-functions would be involved, however, and since the sums in eqs. (9) and (11) would still have to be carried out, the time saving is not as considerable as one might hope. I have done some preliminary experiments which confirm these ideas.

Of the two recursive methods, Moshinsky's is now marginally better than that of Baranger and Davies. This is because with the memo-function technique what dominates the total time for calculating a representative set of brashinskets is no longer the recursion depth but rather the time for a single level of recursion; and in Moshinsky's procedure fewer $6j$ coefficients need be computed. (Baranger and Davies's one-step recursion from any l value to $l = 0$ turns out to be slower than the use of eq. (7); I have therefore compared this with the speed yielded by eq. (4).)

It is not possible to compare directly the timing for a memo-function with any of the previous data, because it depends on what calculations have been called for earlier. Instead I used the entire test set mentioned above and eliminated all the argument sets that were redundant under a symmetry of the brashinskets; after the first occurrence these would merely be looked up. The remainder were then placed in a random order. The set of 2112 brashinsket calculations which resulted represents a rather worse case than would usually occur during, say, a shell-model calculation. The values were calculated, once with the memo-function and once with the table look-up inhibited. The results were as follows:

Average Time for	No Table Look-up	With Table Look-up
first 200	----	67600
last 200	----	816
all 2112	69900	6440

The average times are in units of τ, the floating-point multiplication time, and are averages for the computation of one brashinsket. The timing with table look-up is higher than is suggested by the data in section II, because in the test set used here high values of ρ predominate.

The improvement in speed by a factor of better than 10 is striking and amply justifies the use of the memo-function technique. Provided the value of λ has already occurred in earlier calculations, the use of recursion relations allows great savings in time even for new argument sets. It is thus the combination of recursive computation with the meo-function method which brings about the improvement. It is worth not that here Algol shows up to great advantage compared to Fortran; it incorporates a machine-language coded mechanism for carrying out the recursion and in some machines uses special hardware operations not available to the Fortran programmer.

The gain in speed due to this method should not blind one to the need for careful programming. Much time may be lost, for instance, in the evaluation of the factorials; I have found it very convenient for the computation of the brashinskets (and other coefficients derived from group theory) to set up at the start a table of square roots of factorials (and sometimes of double factorials as well), divided by a suitable constant to the power of the argument – thus saving much time in the evaluation of square roots; where a ratio of products of factorials is needed, the powers of the constant tend to cancel, and the squaring is carried out only once. The table of factorials should in most machines be calculated in double precision, else the higher factorials show excessive rounding errors.

The program that uses the brashinskets also requires care, and some thought should be given to reducing the number of brashinskets needed, since their computation – even with the technique just described – can still take up to 30% of the total time. Among other useful ways of cutting time, the use of the known symmetries[5] and sum rules[23] has the added advantage of improving numerical accuracy.

ACKNOWLEDGEMENTS

My thanks go above all to Marcos Moshinsky, who first set my feet on this path, but also to the many members of the Mexican group of theoretical physics who have helped me in debugging programs or pointing out errors and possible improvements. I am grateful to Brian Buck for several discussions and a copy of his unpublished results. I would like to acknowledge the hospi-

tality of the Nuclear Physics Laboratory of Oxford University, where a large part of this work was carried out. Finally I would like to place on record my appreciation of the help received from the staff of the four computing centres whose facilities I have made use of: the Atlas Computer Laboratory of the Science Research Council of Great Britain; the PDP-10 computer at the Nuclear Physics Laboratory of Oxford; the Burroughs 6500 at the University of Mexico; and the PDP-10 of the Centro Nuclear of the Instituto Nacional de Energía Nuclear, Mexico.

REFERENCES

1. M. Moshinsky, *The Harmonic Oscillator in Modern Physics: From Atoms to Quarks*, (Gordon & Breach, New York 1969).
2. I. Talmi, Helv. Phys. Acta 39 (1952) 346.
3. J.C. Slater, Phys. Rev. 34 (1929) 1293; see also E.U. Condon and G.W. Shortley, *Theory of Atomic Spectra*, (Cambridge University Press 1935).
4. R. Thieberger, Nuclear Phys. 2 (1956/7) 533; K.W. Ford and E.J. Konopinski, Nuclear Phys. 9 (1958/9) 218.
5. M. Moshinsky, Nuclear Phys. 13 (1959) 104.
6. T.A. Brody, Rev. Mex. Fís., 8 (1959) 139.
7. T.A. Brody and M. Moshinsky, *Tables of Transformation Brackets*, Universidad Nacional Autónoma de México, Mexico 1960; 2nd edition, (Gordon & Breach, New York 1967).
8. T.A.Brody, G. Jacob and M. Moshinsky, Nuclear Phys. 17 (1960) 16.
9. J.B. Langworthy and S. Podgor, NRL Report 6733, Washington 1968.
10. R.D. Lawson and M. Goeppert-Mayer, Phys. Rev. 117 (1960) 174.
11. V.V. Balashov and V.A. Eltekov, Nuclear Phys. 16 (1960) 423.
12. A. Arima and T. Terasawa, Prog. Theor. Phys. 23 (1960) 115.
13. F. Dönau and G. Flach, *Gruppentheoretische Methoden im Schalenmodell der Kerne* II, (Akademie-Verlag, Berlin 1969).
14. Yu. F. Smirnov, Nuclear Phys. 27 (1961); Nuclear Phys. 39 (1962) 346.
15. K. Kumar, J. Math. Phys. 7 (1966) 671.
16. M.M. Bakri, Nuclear Phys. A96 (1967) 115, 377.
17. B. Buck, unpublished, 1968.
18. A. Gal, Ann. Phys. (N.Y.) 49 (1968) 341.
19. P. Kramer, Rev. Mex. Fís., 19 (1970) 241.
20. T.A. Brody and M. Moshinsky, Rev. Mex. Fís., 9 (1960) 181.
21. B. Kaufmann and C. Noack, J. Math. Phys. 6 (1965) 142.

22. M. Moshinsky, *Group Theory and Collective Motions*, Notes of the Latin American School of Physics, Mexico 1962.
23. V.C. Aguilera-Navarro, T.A. Brody and J. Flores, Rev. Mex. Fís., 19 (1970) 303.
24. The concept was foreshadowed in a number of papers, e.g. H.J. Mang, Z. f. Physik 148 (1957) 582.
25. A. Erdélyi (dir.), *Higher Transcendental Functions*, vol. 2, (McGraw-Hill, New York 1953).
26. M. Baranger and K.T.R. Davies, Nuclear Phys. 79 (1966) 403.
27. D.M. Brink, unpublished results reported in J.H. Gunn, Algorithm A.P.L.9, Niels Bohr Institutet, Regnemaskine Afdeling, NORDITA, Copenhague 1965.
28. R.H. Richardson, J. Shapiro and F.B. Malik, Phys. Rev. C3 (1971) 84.
29. R.R. Chasman and S. Wahlborn, Nuclear Phys. A90 (1967) 401.
30. j.D. Talman and A. Lande, Nuclear Phys. A163 (1971) 249.
31. See e.g. H. Rutishauser, Numer. Math. 9 (1966) 1.
32. G. Peters and J.H. Wilkinson, Paper II/18 in Wilkinson and Reinsch (eds.), *Handbook for Automatic Computation*, II, Linear Algebra, (Springer-Verlag, Berlin 1971).
33. D. Michie, Proceedings, IFIPS Congress, Edinburgh 1968.
34. P. Naur et al., Numer. Math. 2 (1963) 106; also appeared in Comm. ACM 6 (1963) 1.
35. P.J. Hagan, Algorithm DECUS 10-70, Digital Equipment Corp., Maynard, Mass., 1970.
36. A.P. Ershov, Dokladi Akad. Nauk SSSR 118 (1958) 427.
37. D.E. Knuth, *The Art of Computer Programming*, vol. 3, Sorting and Searching, (Addison-Wesley, Reading, Mass., 1973).
38. W. Buchholz, IBM Systems J. 2 (1963) 86.

RESUMEN

Se describen brevemente los principales procedimientos computacionales para obtener los paréntesis de transformación del oscilador armónico (o brashinsket) y se comparan en cuanto a su velocidad de ejecución. Luego se discute la técnica de construir una tabla de los valores ya obtenidos para volver a usarlos; cuando se emplea esta técnica combinada con un método recursivo de computación, se obtiene, en un cálculo extenso, una reducción en tiempo por un factor de 10 o más.