Calculation of general recoupling coefficients using graphical methods

V. Fack, S.N. Pitre, J. Van der Jeugt*
Department of Applied Mathematics and Computer Science,
Universiteit Gent, Krijgslaan 281–S9, B–9000 Gent, Belgium
E-mail : Veerle.Fack@rug.ac.be

November 22, 2000

Abstract

A new program to generate a summation formula in terms of 6-j coefficients for a general angular momentum recoupling coefficient is described. This algorithm makes use of the graphical techniques as developed by Yutsis, Vanagas and Levinson [1]. Attention is paid to providing an appropriate data structure for the graph representing a general recoupling coefficient, and to reducing the coefficient to summation formula with a minimal number of summation variables. Our results are compared to those of Bar-Shalom and Klapisch [5], who developed the program `NJGRAF` for the same purpose, also using graphical methods.

*Senior Research Associate N.F.W.O. (National Fund for Scientific Research, Belgium)

PROGRAM SUMMARY

Title of program : `NEWGRAPH`

Catalogue number :

Program obtainable from : CPC Program Library, Queen's University of Belfast, N. Ireland (see application form in this issue)

Computers used (Operating systems) : 486-based PCs (MS-DOS, Linux [13]), Sun Sparc (Unix).

Programming language used : C
(Compilers : Turbo C++ [14], GNU CC [15], SPARCompiler C).

No. of lines in source program : ??

Keywords : atomic structure, nuclear structure, scattering, general recoupling coefficient, angular momentum, Racah coefficient, $3n$-j coefficient, coupling tree, binary trees, recursive search process, Yutsis graphs, cubic graphs.

Nature of physical problem : A general recoupling coefficient for an arbitrary number of (integer or half-integer) angular momenta is expressed as a multiple sum over products of 6-j coefficients, including phase factors and square root factors. This summation formula can then be evaluated for given values of the angular momenta (for this purpose we use the program `NJSUMMATION` [10]).

Method of solution : A summation formula for a general recoupling coefficient is obtained by representing the coefficient by a Yutsis graph, and by performing a number of reduction rules valid for such graphs. Each reduction rule contributes to the final summation formula either just by a numerical factor, or else by an additional summation variable. The purpose is to find an optimal summation formula, i.e. with a minimal number of summation variables. With this in mind special attention is paid to implementing the graphical rules whereby N-loops are reduced in the most optimal way.

Typical running time :

LONG WRITE-UP

1 Introduction

In this paper, we describe an efficient C program to calculate a general recoupling coefficient for an arbitrary number of integer or half integer angular momenta by using the graphical methods developed by Yutsis, Vanagas and Levinson [1]. A famous program of Burke [2], `NJSYM`, is dealing with the same problem. In Burke's approach, the problem is divided in essentially two parts : first, the general recoupling coefficient is expressed as a (multiple) sum over a product of 6-j coefficients (including phase factors and square root factors); secondly, this expression is evaluated for given data of the angular momenta. The Burke method to find a summation formula for a general recoupling coefficient is equivalent to finding a certain path between the two binary trees. A binary coupling scheme expresses the order in which n angular momenta are being coupled, e.g. ($n = 4$) :

$$\begin{aligned} &|\, ((j_1, j_2)j_5, (j_3, j_4)j_6)j_7\; m\, \rangle = \\ &\qquad \sum_{m_1,\ldots,m_6} C^{j_1 j_2 j_5}_{m_1 m_2 m_5} C^{j_3 j_4 j_6}_{m_3 m_4 m_6} C^{j_5 j_6 j_7}_{m_5 m_6 m} |\, j_1 m_1\, \rangle \otimes |\, j_2 m_2\, \rangle \otimes |\, j_3 m_3\, \rangle \otimes |\, j_4 m_4\, \rangle. \end{aligned}$$

Herein, $C^{jj'j''}_{mm'm''}$ is a vector-coupling (Wigner or Clebsch-Gordan) coefficient [3, 4]. The relation between such a coupling scheme and a binary tree is clear [2, 4]. A general recoupling coefficient is then a transformation coefficient between two such coupling schemes, e.g. :

$$\langle\, ((j_1, j_2)j_5, (j_3, j_4)j_6)j_7 \mid (j_1, ((j_2, j_3)j_8, j_4)j_9)j_7\, \rangle.$$

Here, the m-dependence is dropped since such coefficients are independent of m [3]. Burke's method of finding a transformation relating the two binary coupling trees often yields expressions which are far from optimal. In order to improve `NJSYM`, Bar-Shalom and Klapisch [5] developed a new program `NJGRAF`. Their way of producing a "best formula" (where the number and values of summation variables are minimal) was to use graphical methods as developed by Yutsis, Levinson and Vanagas [1] and explained in many books [4, 6, 7, 8]. This graphical method is very powerful, and for many transformation coefficients it will yield the most optimal expression in terms of 6-j coefficients.

Our own contributions to the problem are as follows. In the first paper [9], a recursive search process is used to find a sequence of transformations relating the two binary coupling schemes that is as short as possible, thus yielding an optimal summation formula. For all the test examples in literature and for examples which we gave ourselves, our program `NJFORMULA` [9] produces an expression which is often better than (and sometimes as good as) the one obtained by means of `NJGRAF`. By a better expression, we mean a (multiple) sum over products of 6-j coefficients with fewer summation variables and fewer 6-j's.

Our second paper [10] deals with the actual evaluation of such a summation formula. Our program `NJSUMMATION` is the equivalent of `GENSUM` in the programs `NJSYM` and `NJGRAF`, and uses a completely different method viz recursion to calculate the sum over several variables. A general algorithm for the evaluation of a summation formula with an arbitrary number of summation variables is developed, in which nested loops of arbitrary depth are 'simulated' by means of recursive programming techniques. The fact that each step in this recursive algorithm basically handles only one summation variable, also allows to formulate a clear algorithm for the determination of the range of a summation variable.

In the present paper, we present an efficient C program to calculate the summation formula using the powerful graphical methods developed by Yutsis, Vanagas and Levinson [1] which are also used by Bar-Shalom and Klapisch in `NJGRAF` [5] while the program used for the numerical evaluation of the formula remains the same i.e. `NJSUMMATION`.

The general recoupling coefficient is represented by a graph (section 2), and by reducing the graph according to a number of rules (section 3) a formula for the coefficient in terms of 6-j's is found. The graph is reduced by removing loops or cycles in it (section 4). We show that it is important to make the right choice not only of which loop is to be removed first, but also of how it is removed ("collapsed"). Another new aspect of the problem is the choice of the datastructure representing the graph : here, we simply make use of the two coupling trees as originally given (section 5). Section 6 gives further implementation details, and section 7 discusses our results and compares them with other programs. The main conclusion is that the present program performs better than any of the existing programs for the same purpose.

2 Graphical representation of coefficients

The general theory of Yutsis graphs in angular momentum theory is developed and explained in [1], see also [6, 7, 8]. Here, we only need the graphical representation of a general recoupling coefficient (this section), and the rules to reduce such a graph (next section).

Consider a general recoupling coefficient, for which the binary coupling schemes in the left and right bracket have $n+1$ leaves. The graph representing this coefficient will be a cubic graph with $2n$ nodes and $3n$ lines. Each line is labelled by an angular momentum j_i. The lines have a direction, and the nodes have a sign label ($+$ or $-$). For a coupling $(j_1 j_2) j_3$, the node has a $+$ label if the three lines representing the j_1, j_2 and j_3 are anti-clockwise oriented in the node; otherwise it gets a $-$ label.

To build the graph representing a general recoupling coefficient, one can start from the binary coupling trees and use the following steps:

- The tree which represents the coupling scheme on the right hand side of the bracket is drawn so that the nodes representing the triads have a '$+$' sign. The lines representing the compounded angular momenta should be directed towards the node while the lines representing the resultant should be directed away from the node.

- The nodes representing the triads of the tree on the left hand side of the bracket should have a '$-$' sign. The direction of the lines and the mutual position of the first and the second line in a triad should be the reverse of the above.

- After drawing the 2 trees, the corresponding lines are contracted.

The graph thus obtained is invariant under any transformation which conserves the order of the lines around the nodes. The sign of a node in which j_1, j_2 and j_3 meet may be changed by multiplying the value it represents by $(-1)^{j_1+j_2+j_3}$. Change of direction of a line labelled by j involves a multiplication by $(-1)^{2j}$. Thus, directions of lines and labelling of nodes have an influence on phase factors only.

To illustrate the transition from a general recoupling coefficient to the graph, we give a very simple example here for $\langle\, ((a,b)d,c)f \mid (a,(b,c)e)f\, \rangle$ (figures 1, 2 and 3).

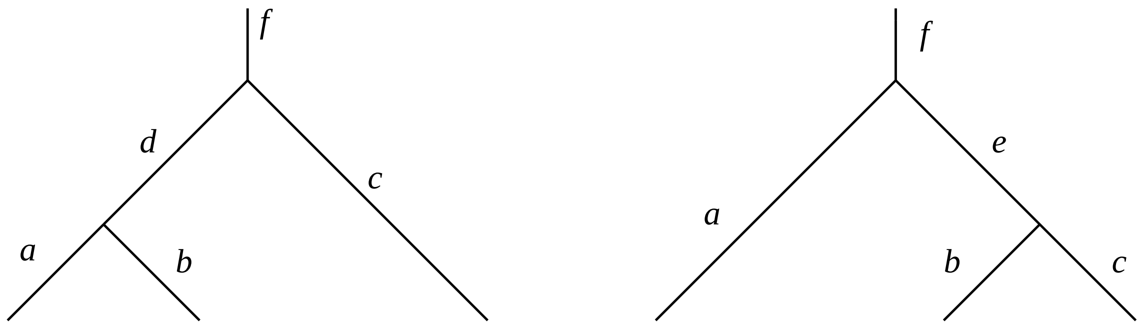


Figure 1: Binary trees drawn with each edge representing an angular momentum

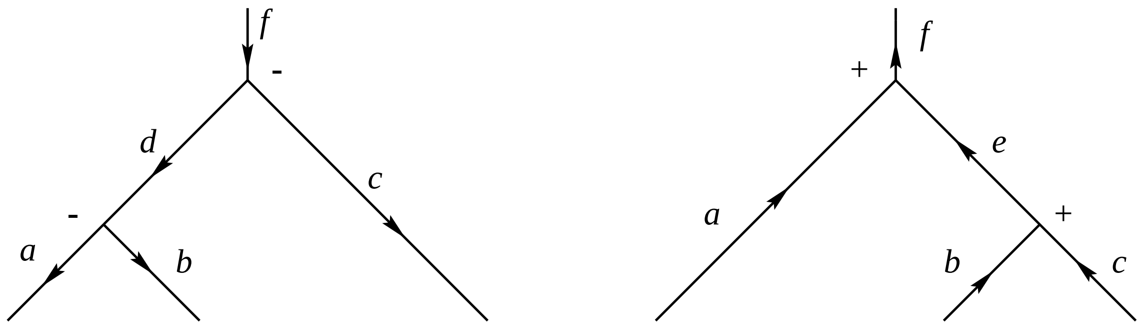


Figure 2: Direction of lines and signs of nodes

For a general recoupling coefficient, the transformation coefficient between the pair of binary coupling schemes is equal to the j-coefficient represented by this diagram multiplied by the following factor ([1] eqns. (22.1) and (22.2)) :

$$(-1)^{2(J+\sum_{i=1}^{n-2} b_i + S)} \times C,$$

where S is the sum of all 'first' coupled angular momenta, n is the number of leaves, and C is given by :

$$C = \left[\prod_{i=1}^{n-2} (2a_i+1)(2b_i+1)\right]^{1/2}.$$

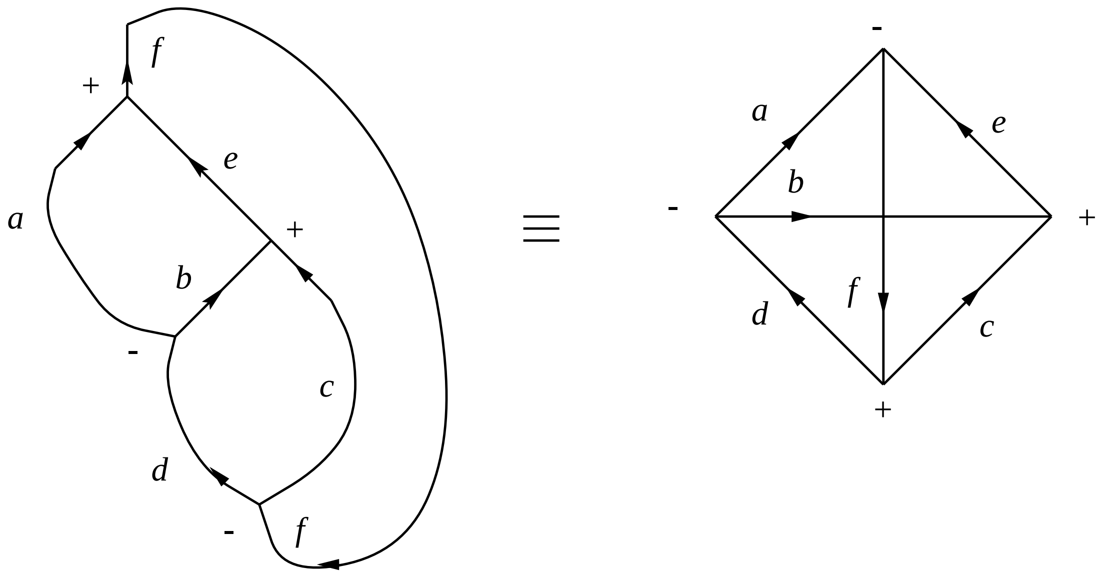


Figure 3: Joining the two parts to obtain a graph

Here a_i's are the intermediate angular momenta on the left side of the bracket and b_i's on the right side and J is the total angular momentum. For example, consider the following transformation coefficient :

$$\langle\, ((j_1,j_2)j_6,(j_3,(j_4,j_5)j_7)j_8)j_9|\;(((j_1,j_4)j_{10},(j_2,j_3)j_{11})j_{12},j_5)j_9\,\rangle$$

The a_i's are angular momenta in the binary tree corresponding to the left bracket other than the leaves and the root, i.e. j_6, j_7 and j_8. Similarly, the b_i's are angular momenta in the tree corresponding to the coupling scheme in the right bracket excluding the leaves and the root, thus : j_{10}, j_{11}, j_{12}. In the left bracket, the 'first' coupled momenta are all those momenta which appear first in a triad in the binary tree i.e. j_1, j_3, j_4 and j_6. For the right bracket these are : j_1, j_2, j_{10}, and j_{12}. Thus the sum of the first coupled momenta is :

$$S = 2j_1 + j_2 + j_3 + j_4 + j_6 + j_{10} + j_{12}.$$

Since $J = j_9$, the external phase factor is : $(-1)^{2(j_9+2j_1+j_2+j_3+j_4+j_6+2j_{10}+j_{11}+2j_{12})}$

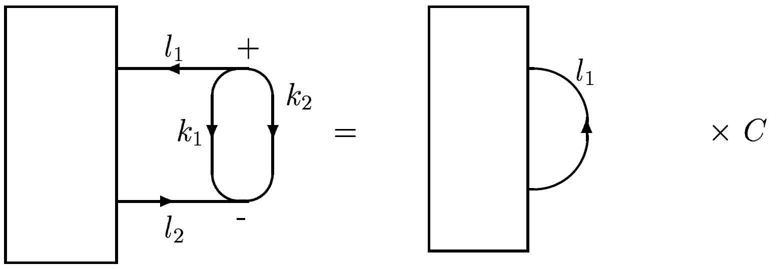
and

$$C = [(2j_6+1)(2j_7+1)(2j_8+1)(2j_{10}+1)(2j_{11}+1)(2j_{12}+1)]^{1/2}.$$

3 Rules for reducing Yutsis graphs

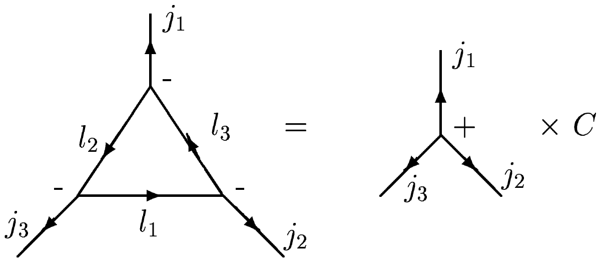
Once the graph is generated, it can be simplified with the help of the rules developed by Yutsis, Vanagas and Levinson [1]. These rules describe some possible transformations on the graph and their effect [5]. The specific rules employed in our program are the location of N-loops in the graph like a 2-loop or a bubble, 3-loop or a triangle, a 4-loop or a square and their elimination using the rules given below along with the operation of interchange which comes into play when these N-loops are not present for $N = 2, 3, 4$.

Rule 1. *2-loop or a bubble.*



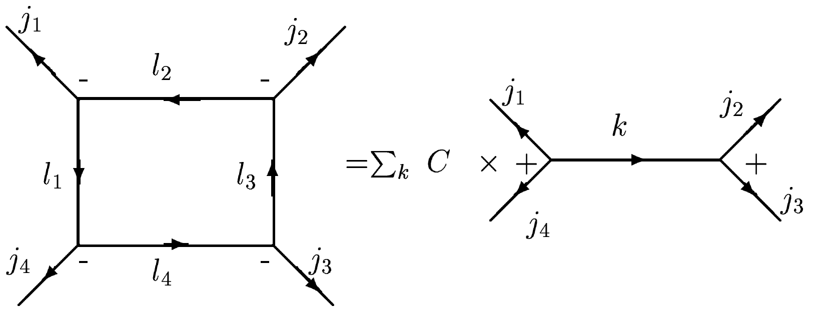
where $C = (2l_1 + 1)^{-1}\ \delta(l_1, l_2)$.

Rule 2. *3-loop or a triangle.*



where $C = \left\{ \begin{array}{ccc} j_1 & j_2 & j_3 \\ l_1 & l_2 & l_3 \end{array} \right\}$.

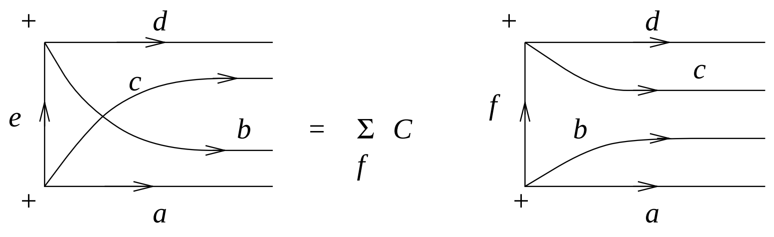
Rule 3. *4-loop or a square.*



where $C = (-1)^{k+l_2-l_4} \times\ (2k+1) \left\{ \begin{array}{ccc} j_1 & j_4 & k \\ l_4 & l_2 & l_1 \end{array} \right\} \left\{ \begin{array}{ccc} j_2 & j_3 & k \\ l_4 & l_2 & l_3 \end{array} \right\}$.

Rule 4. *Interchange.*

where $C = (-1)^{b+c+e+f}\ (2f+1) \left\{ \begin{array}{ccc} a & b & f \\ d & c & e \end{array} \right\}$.



4 Method of solution

4.1 General ideas

Each such loop when removed according to the above rules, will give a contribution to the final formula. The complete formula is said to be generated when after all simplifications of the graph, it reduces to a 'triangular delta' shown in figure 4 which is numerically equal to unity if the three j's involved form a triad and zero otherwise.

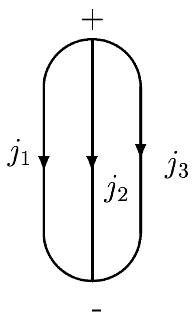


Figure 4: The graphical representation of the triangular delta.

In **NJGRAF**, Bar-Shalom and Klapisch have used three kinds of transformations on the graph : along with the location of different N-loops and their elimination, they also use a cut on m lines ($m = 1,\ 2,\ 3$) and the disposal of a zero branch. The N-loops of the lowest order which are detected are eliminated. They do not consider which loop will be the best choice to be eliminated given a list of N-loops of the same order present in the graph. The program **RECOUP** [11] also uses the 'elimination of N-element chains or N-loops' without spending time on this decision making.

In **NEWGRAPH**, it is first checked if there are any 2-loops or bubbles present in the graph and they are then disposed of. They correspond to a delta function in terms of the formula along with some phases and constants. There is no addition of either a 6-j coefficient or a sum variable in the formula. The process of the removal of this 2-loop detected actually involves redrawing the graph without this loop, after having joined the remaining 2 lines of the 2 nodes which do not form the loop as shown in Rule 1 of section 3. It does not matter if there are one or more 2-loops present, this process is continued until all of them have been removed and there are no 2-loops left.

The 3-loops are the next to be disposed of, if they are located, because this results in the addition of only a 6-j coefficient to the formula [1] and not a variable of summation and the aim is

[1]By addition of a quantity to a formula we mean that the new formula is obtained from the previous one by

to have a formula with the minimum number of sum variables for optimal calculation speeds. The removal of 3-loops in a graph replaces 3 nodes originally present in the graph by one node and thus simplifies it. All the 3-loops present are removed in the sequence in which they are detected, there is no priority decision to be made here since they are equivalent as regards to the simplifications their removal introduces in the graphs.

4.2 Removal of 4-loops

If no 3-loops are located, the next step is to search for and reduce a 4-loop. The 4-loop can be reduced as per rules given and this leads to an addition of two 6-j coefficients and one summation variable k to the formula. It is a powerful tool to simplify the graph if used in the right way. If there are several 4-loops present, we are now faced with a choice having far-reaching implications. Not only is it now a question of which 4-loop is to be disposed of first but also in which manner. This is crucial since the additions to a formula given by the disposal of every 4-loop are the same and they are substantial, but the changes in the overall graph produced on removal of each 4-loop are different. The aim is to collapse that 4-loop which will lead to a simpler graph in terms of the formula i.e. so that it produces a maximum number of triangles in the graph whose subsequent disposal will not add new sum variables to the formula. The information of other existing 4- and 5-loops can be used to decide which 4-loop should be reduced first so as to get a more simplified graph.

To illustrate the importance of this decision, consider the following situation where a part of the graph has been drawn.

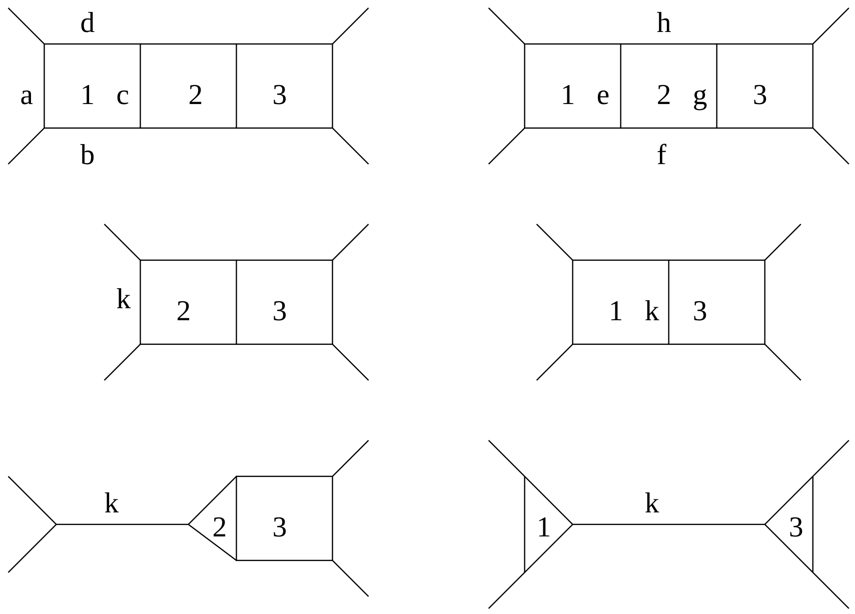


Figure 5: Part of a graph with 4-loops

This part of a graph has 3 adjoining 4-loops. The orientations of the nodes and the directions

(multiplicative) concatenation of that quantity.

of the edges have been neglected here since they are to be taken into account only to arrive at the correct phase factor which is not the issue at this point. Different possible choices are investigated here to show what kind of a simplification they give in the graph. On the left hand side, the 4-loop number 1 is decided to be collapsed first and the collapse is carried out on the sides a and c first. The resulting graph now has two 4-loops remaining as shown. However consider the collapse of the same loop but now carried out on the sides b and d. This gives a 3-loop along with a 4-loop. Thus, a collapse on the sides b and d simplifies the graph more because the removal of a 3-loop does not add a sum variable to the formula. Thus a collapse on the sides b and d is the best choice if one decides to collapse the loop 1 first. Consider now the diagrams on the right which show the removal of the loop 2 first in the same graph. This can also be done in the following 2 ways. A collapse on the sides e and g also leads to a graph with two 4-loops in it whereas a collapse on the sides f and h gives two triangles or 3-loops. Clearly, collapsing the loop on the sides f and h is most advantageous since it creates 2 triangles which will not lead to further sum variables. This goes to show that the loop 2 is the best choice to be removed first among the three 4-loops shown here only if it is collapsed on the sides f and h. Thus these two considerations are equally important.

Given a list of all 4-loops present, which one of them should be removed first and the choice of which two sides in that 4-loop will become the new k is made by making use of the following criteria : find those two sides in a 4-loop which are most common to other 4-loops present i.e. with sides which have a maximum number of overlaps with other 4-loops. It will then be advantageous to collapse the 4-loop in a way that this pair of sides disappears altogether. If a decision cannot be made just by taking into account all other 4-loops present (when the 'maximum' occurs several times), the list of 5-loops is considered. The 4-loop which has sides overlapping with many other 5-loops will be the optimal choice since a 5-loop can be converted into a loop of a lower order after the collapse of the chosen 4-loop and the more 5-loops which can be reduced to 4-loops as a side result the better. The decision to remove a particular 4-loop is thus made by taking into account the list of other 4-loops present and also a list of existing 5-loops. One could at this point use more information in making this choice, like higher order loops present in the graph and more sophisticated methods, in making the decision. Our approach is a compromise between spending time making the best choice on one hand and generating a possibly more optimal formula on the other hand.

4.3 Removal of larger loops

If there is no 4-loop present in the graph, the next step is to locate loops of higher order i.e. 5-loops. A 5-loop (or an N-loop in general) can be converted to a loop of a lower order by an operation called the *interchange* also shown in the set of rules.

There is also a choice while performing the operation of *interchange* as to which 5-loop should be reduced to a 4-loop or less among a list with a view to get the most simplified graph and finally a better formula. Consider an interchange in figure 6 here where we have two adjoining 5-loops. The wrong choice of a and b for the interchange does not lead to any simplification in this diagram. The 5-loops are different now but they do not reduce to a 4-loop.

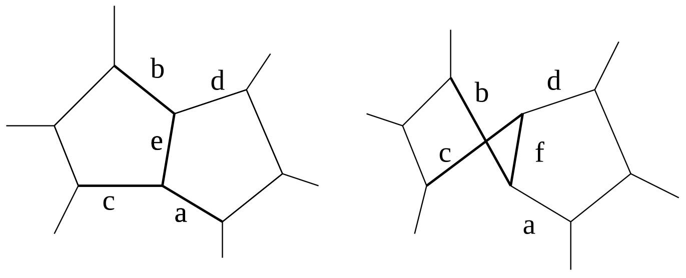


Figure 6: Incorrect choice of a and b for an interchange

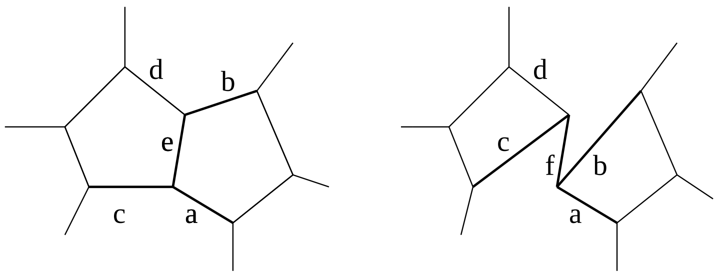


Figure 7: The correct choice of a and b for an interchange

In this case shown in figure 7, the choice of the two edges in a 5-loop on both sides of e as a and b gives two 4-loops on doing the interchange. This is the choice which is made once e has been fixed. Apart from 5-loops, loops of higher order can also be simplified with the help of *interchange.*

In this part of the algorithm, one has to find a compromise between (a) the time spent in making this choice of which loop is to be first reduced among a list of loops yielding an optimal formula that can be computed in a short time, and (b) spending less time on this choice, yielding a formula that is perhaps less optimal and more time consuming for computation.

4.4 Estimates from graph theory

Since the smallest loop detected is always removed first, it is interesting to study the particular types of graphs we deal with and get an estimate on the size of graphs that contain no triangles or squares etc. We would particularly like to know when it is necessary to employ the interchange rule to reduce a graph i.e. when are we likely to come across a graph which has no 4-loops. The size of the smallest loop (or cycle) in a graph is called the *girth* of the graph. A graph is said to be regular of valency v if each of its vertices has a valency v (i.e. has v edges). A regular graph with valency v and girth g is called a (v, g)-graph. A (v, g)-graph with the least possible number of vertices is called a (v, g)-cage. The number of vertices of a (v, g)-cage is denoted by $f(v, g)$. A theorem in graph theory [12] says that a minimal (v, g)-graph exists only if

- $g = 5$ and $v = 3, 7$ or 57,
- $g = 6, 8, 12$.

From the defined graphical representation of a j-coefficient, it follows that all graphs in this particular problem will be *3-regular* or *cubic graphs* (valency of each vertex can be 3 and 3 only). Thus from the above theorem, the smallest graph which does not have any 4-loop or 'square' is the $(3,5)$-cage. This is the Petersen graph with 10 nodes and 15 edges ($n = 5$). This graph contains only 5-loops. The minimal $(3,6)$-graph, i.e. a cubic graph with a girth of 6 is the Heawood graph where $f(3,6) = 14$. This graph has 14 nodes and 21 edges and the smallest loop this graph has is a 6-loop. The only known cage with a girth of 7 is the $(3,7)$-cage. This cage has 24 vertices and 36 edges. This information is helpful in deciding for which recoupling coefficients, an *interchange* is necessary. Thus the smallest graph without any 5-loops can be constructed for an input file consisting of 21 j's and one without any 6-loops can be constructed for 36 j's. Alternatively, every graph with less than 10, 21, and 36 j's has at least one 4-, 5- or 6-loop respectively.

5 Algorithm description and structure of the program

5.1 Datastructure

The input to the program is given in the form of coupling trees where each of the two trees represents a coupling scheme of angular momenta, which is the same input format as other programs to calculate a general recoupling coefficient like `NJGRAF` [5] and `NJFORMULA` [9]. Once we have the input, the job on hand is to construct a graph for the purpose of identifying the various types of loops present in the graph. Once the loops are detected, the graph can be simplifed as per the rules explained in the previous section.

The task is to look for a datastructure to store the graph which will enable the job of tracing the loops of different orders. Many algorithms exist in literature to identify loops or cycles in general graphs. Some of these methods generate a spanning tree for a given general graph. However our specific problem already involves two coupling trees, from which the graph is constructed. The trees themselves will never contain a loop or a cycle by definition. Looking carefully at the loops in such graphs constructed from two given coupling trees, we observe that each loop in this graph contains parts (or couplings) from both the given trees and that every loop contains two leaves as two sides. Thus every cycle in this graph is a combination of a path in one tree and a path in the other tree. A loop can be manually detected by finding the shortest path in both trees between two leaves, which is done by following the edges up until the nearest common parent node of both leaves in each tree (taking care that we count both the leaves only once). Proceeding to perform this calculation for all possible combinations of leaves gives an exhaustive list of *all* loops present in the graph. Thus for our purposes, it is possible to represent the graph in terms of a datastructure which tabulates this information of the distances between any two leaves (or alternatively the different number of edges joining them) in a coupling tree for both the trees. This has been accomplished here by making a special table which maintains the distance or depth of the different leaves from the different coupled nodes for both trees.

Consider the following two coupling trees from the input file `f1.dat` corresponding to the bracket (F_1) (see also section 7) :

$$(F_1) \qquad \langle\, ((j_1,j_2)j_6,(j_3,(j_4,j_5)j_7)j_8)j_9 |\, (((j_1,j_4)j_{10},(j_2,j_3)j_{11})j_{12},j_5)j_9\, \rangle$$

```
f1-left                                   f1-right
-------                                   --------

            | 1  2  4  5  3  9                         | 1  4  2  3  5  9
            |-----------------                         |-----------------
1  2  6     | 1  1  -  -  -  -             1  4 10     | 1  1  -  -  -  -
4  5  7     | -  -  1  1  -  -             2  3 11     | -  -  1  1  -  -
3  7  8     | -  -  2  2  1  -            10 11 12     | 2  2  2  2  -  -
6  8  9     | 2  2  3  3  2  1            12  5  9     | 3  3  3  3  1  1
```

Here, `f1-left` is one coupling scheme whereas `f1-right` is the other. The two tables of distances for both the trees are given next to them. In the table, the first row represents the number of different *leaves* or angular momenta which are themselves not coupled like 1, 2, 4, 5 and 3 (corresponding to j_1, j_2, j_4, j_5, j_3) and the first column has a list of all *coupled* nodes 6, 7, and 8 (corresponding to j_6, j_7 and j_8) for this tree which is in fact exactly the last column of the tree concerned and so is not listed separately again. Notice that the *root* node 9 has been represented as both a *coupled* node as well as a *leaf.* The other entries in the table then indicate the distance at which a particular *coupled* node is from a *leaf.* e.g. the entry at position (1,1) is 1 showing that 6 is only at a distance '1' from the leaf 1. The *leaves* 4 and 5 are also at a distance 1 each from the *coupled* node 7. The coupled node 8 is at a distance 1 from the leaf 3 but at distance 2 each from the leaves 4 and 5. The distances between 2 different leaves are not a direct entry in this table. They can be obtained by making a few simple calculations. Thus if we want to find the (shortest) distance between the leaves say 3 and 5, we look in the columns of 3 and 5 till we come across the *first* non-zero entry in *both* the columns. This corresponds to the coupled node 8 (the row entry). This means that these 2 leaves are connected through the coupled node 8. The distance from 8 to the leaf 3 is 1 and that from 8 to the leaf 5 is 2; adding this gives the distance between the two leaves 3 and 5 as 3. Following the same procedure for `f1-right`, we find that the distance between the same two leaves 3 and 5 in `f1-right` is 4. Adding the two distances between the leaves 3 and 5 from both the trees and subtracting 2 from it (since both the leaves have been counted twice) gives us 5 which indicates that there exists a 5-loop in the graph that will be formed from these two trees which includes the leaves 3 and 5 as its two edges. Thus it is possible to detect all the loops present in the graph with the aid of this *table of distances* which is built directly from the two trees of couplings given as input.

5.2 The removal of loops

One scan of the table gives a list of *all* loops of all orders present in the graph (by looking at each different combination of two leaves in both tables). They are not all stored however. As described earlier, the process of tabulating loops of different orders is immediately disrupted if we come across a loop of order 2 or 3. When a 2-loop or a bubble is detected, all further search for loops is immediately suspended since there is no choice to be made at this stage. Every bubble detected is immediately removed since its contribution to the formula is only that of a constant whereas it simplifies the graph by reducing two nodes in it.

How is a loop disposed of from a graph which has not really been built at all but which exists in the form of the two trees of couplings and tables of distances? Look at both the graph and the tree structure of a recoupling coefficient at the same time. Simplify the graph by first removing this 2-loop from it and redraw it. Now write down the two trees of couplings for this new graph generated. Comparison of the two original trees and the new trees of couplings shows how one can effectively work on these coupling trees while indirectly using the graphical methods to simplify the graph. The advantage here is that we do not need a complicated data structure to store the graph, which will have to be updated at every step. Instead we have two trees of couplings which are simple 2-dimensional arrays that can be updated easily. The tables of distances are then rebuilt from the new updated trees. However, for more complex loops like the 4-loop and the interchange, updating the trees is not very simple because of the many possibilities that have to be taken into account. For example for a 4-loop there can be two couplings from each tree involved in the loop, or one coupling from the first tree and two from the second and vice-versa.

The two couplings which result in the bubble are identified and removed from the trees. We then proceed to relabel one j in either one of the trees (the delta function) as shown in Rule 1 section 3 and update the formula by adding the appropriate constant square root factor and phase factors if necessary (this depends on the orientations of the two nodes which form the bubble and also the directions of the j's in these nodes). The information of the direction of each edge of the graph can be extracted from the couplings themselves. Depending on which tree a particular coupling belongs to, the directions of leaves and coupled nodes are known by definition. Two new tables of distances are constructed for the new updated trees now and a new scan for all possible loops present in this new and simplified graph can begin.

When a 3-loop or a triangle is detected in the scan for loops, all search for further loops is also stopped and one proceeds directly with the removal of this loop in a fashion similar to a 2-loop. The couplings from the two trees which are involved in this 3-loop are collected and removed from the respective trees and instead the single coupling that remains on removing the 3-loop (by using the rules of collapsing a triangle) is put back into the right tree and in the right place. Thus it is not necessary to reorder the two trees of couplings after such an operation. The formula is updated, this time it results in the addition of a 6-j coefficient and of course a few phase factors depending on the directions of the j's involved in this particular loop and the orientations of the nodes. The new trees now correspond to a simpler graph and again the new tables of distances can be constructed and the scan for loops in this different graph can be started.

If a 3-loop is not detected, a list of all loops of higher order encountered (upto a certain predecided limit) is now made. The question now is not only which of the listed 4-loops (if present) will be most advantageous to remove but also in which way we should proceed in removing the selected 4-loop. This can make a lot of difference to the final formula generated as described in subsection 4.2. The decision to remove a particular 4-loop is made by taking into account the list of other 4-loops present and also a list of existing 5-loops. Once this is decided, the actual process of collapsing the 4-loop is carried out according to the Rule 3 of section 3. This involves collecting the couplings in the two trees which are involved in the 4-loop and removing them from the trees and instead adding the new couplings along with the new k. The number of nodes in the graph reduces by 2 on completion of this. The formula is updated and there are two 6-j coefficients which are added to it along with one new summation variable and some phase factors. Again new tables of distances are made and a new scan for loops in this reduced graph gets underway.

If no 4-loop is present in the graph, the task on hand is to look at the list of existing 5-loops with a view of converting one of them into a 4-loop or less using the *interchange*. That 5-loop which on its removal converts one or more existing 5-loops in the graph into a 4-loop is the best candidate for removal given a choice of such loops. This choice is made using similar criteria as for the 4-loop : find that side of a 5-loop which has the maximum number of overlaps with other 5-loops and this side will be the side e on which the interchange can be made as shown in the diagram. Edge e has to be a coupled node of one of the trees and this has to be checked before making the interchange. Then the adjoining two sides in this 5-loop will act as a and b. The interchange results in the addition of one more 6-j coefficient to the formula along with a new summation variable (plus phase and square root factors).

Loops of order higher than 5 can also be converted to ones of lower order by means of the *interchange*. However, as has been discussed in subsection 4.4, this situation will only occur when the number of j's is greater than or equal to 21.

As an example, consider the datafile `f2.dat` and the graph for it ((F_2) of section 7) :

$$\langle\, ((j_1,(j_2,j_3)j_7)j_8,(j_4,(j_5,j_6)j_9)j_{10})j_{11}|\, (((j_1,(j_4,j_5)j_{12})j_{13},j_2)j_{14},(j_3,j_6)j_{15})j_{11}\,\rangle.$$

```
f2-left                                     f2-right
-------                                     --------
           0 | 2  3  1  5  6  4 11                     0 | 4  5  3  6  1  2 11
          --|--------------------                    ---|--------------------
 2  3  7   7 | 1  1  -  -  -  -  -           4  5 12  12 | 1  1  -  -  -  -  -
 1  7  8   8 | 2  2  1  -  -  -  -           3  6 15  15 | -  -  1  1  -  -  -
 5  6  9   9 | -  -  -  1  1  -  -           1 12 13  13 | 2  2  -  -  1  -  -
 4  9 10  10 | -  -  -  2  2  1  -          13  2 14  14 | 3  3  -  -  2  1  -
 8 10 11  11 | 3  3  2  3  3  2  1          14 15 11  11 | 4  4  2  2  3  2  1
```

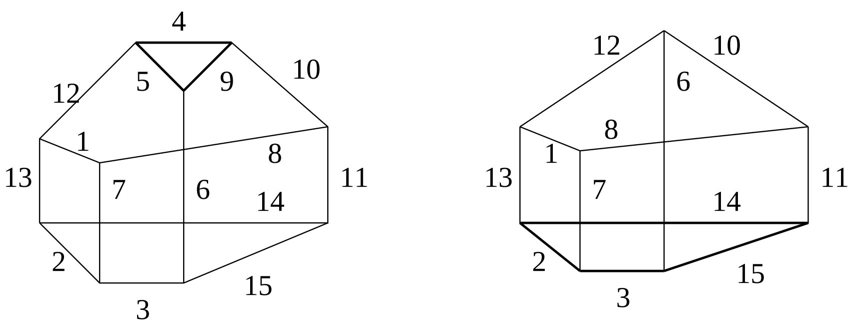


Figure 8: Removal of the 3-loop `(4-5-9)` and then the 4-loop `2-3-15-14` from the graph for `f2.dat`

The left picture in figure 8 is the graph for these two coupling trees. It clearly shows the presence of a triangle which is the first loop to be removed from the graph. The right picture in figure 8 shows the graph after the removal of the 3-loop (4-5-9). Apart from phase factors, the only contribution to the formula is $\left\{ \begin{array}{ccc} j_{12} & j_6 & j_{10} \\ j_9 & j_4 & j_5 \end{array} \right\}$. Now the new trees and their tables of distances are :

```
           0 | 2  3  1  6 12 11               0 | 3  6  1 12  2 11
           --|-----------------             ---|-----------------
2  3  7    7 | 1  1  -  -  -  -   3  6 15   15 | 1  1  -  -  -  -
1  7  8    8 | 2  2  1  -  -  -   1 12 13   13 | -  -  1  1  -  -
6 12 10   10 | -  -  -  1  1  -  13  2 14   14 | -  -  2  2  1  -
8 10 11   11 | 3  3  2  2  2  1  14 15 11   11 | 2  2  3  3  2  1
```

The program now detects several 4-loops and 5-loops as follows :

```
4-loops detected :                5 -loops also detected:
0:   3 15 14  2                    0:  3  6 10  8  7
1:   6 15 11 10                    1:  3 15 11  8  7
2:   1 12 10  8                    2:  6 15 14 13 12
3:   1 13  2  7                    3:  1 13 14 11  8
                                   4: 12 13 14 11 10
                                   5:  2 14 11  8  7
```

The decision is made to collapse the 4-loop (3-15-14-2) just by taking into account the given list of 4-loops for this particular case because the sides 15 and 2 are present in two more 4-loops and if the 4-loop is collapsed on 15 and 2 so that these sides disappear altogether, two more triangles will result in the graph.

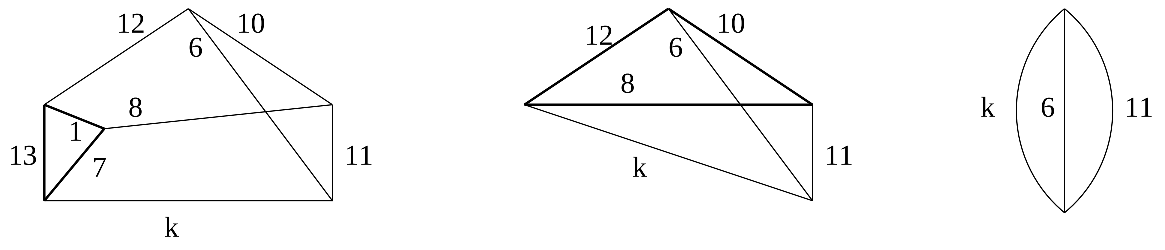


Figure 9: Removing the 3-loops from the simplified graph

The contribution to the formula on removing the 4-loop is :

$$(-1)^{k_1+j_3-j_{14}}(2k_1+1)\left\{ \begin{array}{ccc} j_7 & j_{13} & k_1 \\ j_{14} & j_3 & j_2 \end{array} \right\}\left\{ \begin{array}{ccc} j_6 & j_{11} & k_1 \\ j_{14} & j_3 & j_{15} \end{array} \right\}.$$

The new diagram on collapsing the 4-loop is shown on the left in figure 9. The diagram now has many triangles out of which (1-13-7) is removed first as it is detected first. The next graph is what remains on removal of the 3-loop and the contribution to the formula now is : $\left\{ \begin{array}{ccc} j_{12} & k_1 & j_8 \\ j_7 & j_1 & j_{13} \end{array} \right\}$. One more triangle (12-8-10) is the next candidate to be removed giving an addition to the formula of $\left\{ \begin{array}{ccc} k_1 & j_{11} & j_6 \\ j_{10} & j_{12} & j_8 \end{array} \right\}$. What remains now is just a triangular delta as shown. It is equal to unity if the three angular momenta form a triad. Thus the total formula generated is :

$$\sum_{k_1} (2k_1+1) \left\{ \begin{array}{ccc} j_{12} & j_6 & j_{10} \\ j_9 & j_4 & j_5 \end{array} \right\} \left\{ \begin{array}{ccc} j_7 & j_{13} & k_1 \\ j_{14} & j_3 & j_2 \end{array} \right\} \left\{ \begin{array}{ccc} j_6 & j_{11} & k_1 \\ j_{14} & j_3 & j_{15} \end{array} \right\}$$

$$\times \left\{ \begin{array}{ccc} j_{12} & k_1 & j_8 \\ j_7 & j_1 & j_{13} \end{array} \right\} \left\{ \begin{array}{ccc} k_1 & j_{11} & j_6 \\ j_{10} & j_{12} & j_8 \end{array} \right\}$$

The addition of phase factors is a rather technical detail presently ignored, but of course included in the final program.

6 Implementation details

The main program which joins the two modules `NEWGRAPH` and `NJSUMMATION` is `newgmain.c`. All different subroutines responsible for generating the formula are present in `NEWGRAPH` while `NJSUMMATION` calculates the formula numerically. The input to the program is given in the form of coupling trees where each of the two trees represents a coupling scheme of angular momenta. This is the same input format as other programs to calculate a general recoupling coefficient like `NJGRAF` [5] and `NJFORMULA` [9]. There is one additional value `nlps` which is added at the end of this input file of trees. This value indicates the maximum number upto which order a list of detected loops is collected and stored in memory e.g. `nlps`=5 indicates that in the search for the lowest loop to be reduced/collapsed, all loops upto the order 5 are stored in an array and this information is used in deciding which loop is to be removed first so as to have a simpler graph after this operation.

6.1 Subroutines

The different subroutines in `NEWGRAPH` are described here.

- `coup()` : After reading the two input trees and storing them in the structure `TREE t1` and `t2`, the subroutine `coup()` checks if the couplings are correctly ordered.
- `arrange()` : The *tables of distances* for the two trees are built in this subroutine.
- `ini_phase()` : The external phase and square root factors are calculated here and added to the formula.
- `cycle()` : The subroutine responsible for detecting loops upto the order `nlps` specified by the user, making use of the information in the two tables of distances together.
- `bubble()` : Removal of a 2-loop or a *bubble* takes place here.
- `triangle()` : Responsible for the disposal of a 3-loop or a *triangle* and updating the formula correspondingly.
- `four()` : Removes a 4-loop from the graph and updates the formula.

- `which_four_4()` : Decides which 4-loop should be disposed of first given a list of 4-loops present in the graph.
- `which_four_5()` : Decides which 4-loop should be disposed of by taking into account the list of 5-loops present in the graph.
- `which_five_four()` : Decides which 5-loop should be chosen for the operation of *interchange* to be performed, given a list of 5-loops, to convert the chosen 5-loop into a 4-loop.
- `init()`: Initializes the arrays and values of variables in the structure `FORMULA`.
- `directn_tr()`: Ensures that all the lines involved in the collapse of a 3-loop or a triangle have the correct directions and updates the phase factors in the formula accordingly.
- `directn_sq()`: Ensures that all the lines involved in the collapse of a 4-loop or a square have the correct directions and updates the phase factors in the formula accordingly.
- `swap_nodes()`: Adds the correct phase to the formula when the orientation of a node is changed which is equivalent to swapping the positions of the two leaves in a coupling.
- `rotate()`: Performs a cyclic permutation of the elements of a 'triad' or a coupling in such a way that a particular element is in the required place without changing their order i.e. without a change of orientation of the node. Hence no update to the formula is necessary here.

These are the main subroutines of the program `NEWGRAPH`. Different tools described by Yutsis *et al* [1] for reducing the graph like a cut on 2 and 3 lines (which may be useful rules when doing the reduction of a graph by hand) do not need to be implemented explicitly in the program. These cut-rules do not yield better formulae, they only split the graph in two subgraphs which can be treated separately. However, since the size of the graphs is so small (the total number of nodes in a graph is always much less than say 100), there is no need to implement these cut-rules.

6.2 Datastructures

The input to the program is given in the form of coupling trees where each of the two trees represents a coupling scheme of angular momenta, which is the same for other programs to calculate a general recoupling coefficient like `NJGRAF` [5] and `NJFORMULA` [9]. The two trees of couplings are 2-dimensional arrays which are each represented in a record structure `TREE` which also contains other relevant information pertaining to these: `ncoups` is the number of couplings in each tree, `k1` is the number of 'leaf' nodes which also includes the root, `cp[][]` is the actual array of couplings, `dist[][]` is the table of distances for each tree. The datastructure used for representing the generated formula is the record structure `FORMULA` (see section 4.1 of [9]), to assure compatibility with the module `NJSUMMATION` [10] which does the numerical evaluation of the generated summation formula. The structure `GRAPH` contains the fields `low` which has the size of the smallest loop present in the graph, `nol[MNLPS]` which is an array which stores the exact number of 4, 5, etc. loops present in the graph and `loop[MNLPS][MAXNRLOOPS][MAXLOOPSIZE]` which is a 3-dimensional array storing lists of 4, 5 etc. loops found in the graph. Here the variables `MNLPS`, `MAXNRLOOPS` and `MAXLOOPSIZE` are fixed in the beginning of the program and stand for the upper limit for the order of the loops detected, the upper limit for the number of loops of each order which are stored and the upper limit for the size of each loop of each order detected.

7 Results and discussion

The program `NEWGRAPH` has been written in C and can be run on any system providing a C (or C++) compiler. The test cases used have also been discussed in [9] and [10] while discussing the performance of `NJFORMULA` :

(G_1) $\langle\, ((j_1,j_2)j_5,(j_3,j_4)j_6)j_7 \mid (j_1,((j_2,j_3)j_8,j_4)j_9)j_7 \,\rangle$

(G_2) $\langle\, (((j_1,j_2)j_8,((j_3,j_5)j_9,(j_6,j_7)j_{10})j_{11})j_{12},j_4)j_{13}$
$\mid (((j_1,j_2)j_{14},((j_3,(j_7,j_5)j_{15})j_{16},j_6)j_{17})j_{18},j_4)j_{13} \,\rangle$

(G_4) $\langle\, ((j_1,j_2)j_{12},(j_3,(j_4,(j_5,((j_6,j_7)j_{13},(j_8,(j_9,(j_{10},j_{11})j_{14})j_{15})j_{16})j_{17})j_{18})j_{19})j_{20})j_{21}$
$\mid (j_5,(j_7,((j_6,(j_{11},(j_9,(j_{10},j_8)j_{22})j_{23})j_{24})j_{25},(j_1,(j_3,(j_2,j_4)j_{26})j_{27})j_{28})j_{29})j_{30})j_{21} \,\rangle$

(F_0) $\langle\, (j_1,j_2)j_5,(j_3,j_4)j_6)j_7 \mid ((j_1,j_3)j_8,(j_2,j_4)j_9)j_7 \,\rangle$

(F_1) $\langle\, ((j_1,j_2)j_6,(j_3,(j_4,j_5)j_7)j_8)j_9 \mid (((j_1,j_4)j_{10},(j_2,j_3)j_{11})j_{12},j_5)j_9 \,\rangle$

(F_2) $\langle\, ((j_1,(j_2,j_3)j_7)j_8,(j_4,(j_5,j_6)j_9)j_{10})j_{11} \mid (((j_1,(j_4,j_5)j_{12})j_{13},j_2)j_{14},(j_3,j_6)j_{15})j_{11} \,\rangle$

(F_3) $\langle\, (((j_1,j_2)j_7,(j_3,j_4)j_8)j_9,(j_5,j_6)j_{10})j_{11} \mid ((j_3,j_6)j_{12},((j_2,j_4)j_{13},(j_1,j_5)j_{14})j_{15})j_{11} \,\rangle$

(F_4) $\langle\, (((j_1,j_2)j_7,(j_3,j_4)j_8)j_9,(j_5,j_6)j_{10})j_{11} \mid ((j_1,j_6)j_{12},((j_3,j_5)j_{13},(j_2,j_4)j_{14})j_{15})j_{11} \,\rangle$

(F_5) $\langle\, (((j_1,j_2)j_8,(j_3,j_4)j_9)j_{10},((j_5,j_6)j_{11},j_7)j_{12})j_{13}$
$\mid ((j_1,j_7)j_{14},(((j_3,j_5)j_{15},(j_2,j_4)j_{16})j_{17},j_6)j_{18})j_{13} \,\rangle$

(F_6) $\langle\, (((j_1,j_2)j_8,(j_3,j_4)j_9)j_{10},(j_5,(j_6,j_7)j_{11})j_{12})j_{13}$
$\mid (j_5,((j_6,(j_2,j_4)j_{14})j_{15},(j_3,(j_1,j_7)j_{16})j_{17})j_{18})j_{13} \,\rangle$

(F_7) $\langle\, ((j_1,(j_2,j_3)j_8)j_9,((j_4,j_5)j_{10},(j_6,j_7)j_{11})j_{12})j_{13}$
$\mid (((j_1,j_4)j_{14},j_6),j_{15},((j_5,j_2)j_{16},(j_7,j_3)j_{17})j_{18})j_{13} \,\rangle$

(F_8) $\langle\, (((j_1,j_2)j_9,j_3)j_{10},(((j_4,j_5)j_{11},j_6)j_{12},(j_7,j_8)j_{13})j_{14})j_{15}$
$\mid (((j_1,j_4)j_{16},j_7)j_{17},(((j_2,j_5)j_{18},(j_8,j_3)j_{19})j_{20},j_6)j_{21})j_{15} \,\rangle$

(F_9) $\langle\, (((j_1,(j_2,j_3)j_{11})j_{12},((j_4,j_5)j_{13},j_6)j_{14})j_{15},(((j_7,j_8)j_{16},j_9)j_{17},j_{10})j_{18})j_{19}$
$\mid (((j_2,j_4)j_{20},j_7)j_{21},((((j_1,j_8)j_{22},(j_9,j_5)j_{23})j_{24},j_{10})j_{25},(j_6,j_3)j_{26})j_{27})j_{19} \,\rangle$

To compare the results of the program `NEWGRAPH` with `NJGRAF` and `NJFORMULA`, a table is provided here. For the above data, it gives the formula generated by these three programs in terms of the number of summation variables (k's) and the number of 6-j's. The smaller the number of k's, the better the formula. For `NEWGRAPH`, there is an additional column which shows the number of steps in which we arrived at the formula and each step here is either the disposal of a loop or an interchange. The numbers in the bracket indicate the number of 2-loops, triangles, squares, interchanges and triangular delta encountered respectively.

The first three test cases listed here are those used by Bar-Shalom and Klapisch [5] for testing the performance of `NJGRAF`. G_4 is a complicated case devised by them and for this, `NEWGRAPH` gives

Test case	#j's	NJGRAF		NJFORMULA		NEWGRAPH		
		#k's	#6-j's	#k's	#6-j's	#k's	#6-j's	#steps
(G_1)	9	0	2	0	2	0	2	3 (0-2-0-0-1)
(G_2)	18	0	2	0	2	0	2	6 (3-2-0-0-1)
(G_4)	18	3	11	3	11	2	10	10 (1-6-2-0-1)
(F_0)	9	1	3	1	3	1	3	3 (0-1-1-0-1)
(F_1)	12	2	5	1	4	1	4	4 (0-2-1-0-1)
(F_2)	15	1	5	1	5	1	5	5 (0-3-1-0-1)
(F_3)	15	2	6	2	6	2	6	5 (0-2-2-0-1)
(F_4)	15	3	7	2	6	2	6	5 (0-2-2-0-1)
(F_5)	18	4	9	2	7	2	7	6 (0-3-2-0-1)
(F_6)	18	2	7	2	7	2	7	6 (0-3-2-0-1)
(F_7)	18	–	–	4	9	4	9	6 (0-1-4-0-1)
(F_8)	21	–	–	7	13	6	12	8 (0-1-5-1-1)
(F_9)	27	–	–	9	17	8	16	11 (0-2-6-2-1)

Table 1: Results obtained by `NJGRAF`, `NJFORMULA` and `NEWGRAPH` for a range of recoupling coefficients.

the best formula that has yet been obtained among all existing programs as the table shows. For the datafiles F_0 to F_6, `NEWGRAPH` consistently gives a formula which is the same as that given by our own `NJFORMULA` and which is sometimes better than if not as good as that given by `NJGRAF`. For more complicated datafiles F_7 to F_9, `NJGRAF` crashes while `NEWGRAPH` performs better than our own `NJFORMULA`, which is not surprising. The complexity of this algorithm is also of a lower order than that used in `NJFORMULA` since the decision of reduction of a loop (which is one step for this graphical algorithm) is straightforward while in the case of the method of binary tree transformations, this involves a recursive search process on a tree which is a brute force method still yielding surprisingly good results. Every step of removing a loop greatly simplifies the graph, thus the total number of steps to obtain the complete formula is small. The conclusion is that the graphical method is superior to the method of binary tree transformations for this particular problem of generating the summation formula for a general recoupling coefficient.

Acknowledgements

This work was partly supported by the EEC (contract no. CI1*–CT92–0101).

References

[1] A.P. Yutsis, I.B. Levinson and V.V. Vanagas, The Theory of Angular Momentum (Israel Program for Scientific Translation, Jerusalem, 1962).

[2] P.G. Burke, Comput. Phys. Commun. **1** (1970) 241.

[3] A. Edmonds, Angular momentum in quantum mechanics (Princeton University Press, Princeton, 1957).

[4] L.C. Biedenharn and J.D. Louck, The Racah-Wigner algebra in Quantum Theory, Encyclopedia of Mathematics and its Applications, Vol. 9, ed. G.-C. Rota (Addison Wesley, Reading, MA, 1981).

[5] A. Bar-Shalom and M. Klapisch, Comput. Phys. Commun. **50** (1988) 375.

[6] D.M. Brink and G.R. Satchler, Theory of Angular Momentum (Clarendon Press, Oxford, 1968).

[7] B.R. Judd, Operator Techniques in Atomic Spectroscopy (McGraw-Hill, New York, 1963).

[8] E. El Baz and B. Castel, Graphical Methods of Spin Algebras (Marcel Dekker, New York, 1972).

[9] V. Fack, S.N. Pitre and J. Van der Jeugt, Comput. Phys. Commun. **83** (1994) 275.

[10] V. Fack, S.N. Pitre and J. Van der Jeugt, Comput. Phys. Commun. **86** (1995) 105.

[11] P. M. Lima, Comput. Phys. Commun. **66** (1991) 89.

[12] P.-K. Wong, J. Graph Theory **6** (1982) 1.

[13] Linux (Unix clone for 386/486-based PCs) version 1.0; publicly available via anonymous ftp to `nic.funet.fi` in directory `/pub/OS/Linux`.

[14] Turbo C++ (version 1.01) User Guide, Borland International Inc. (1990).

[15] GNU CC version 2.5, Free Software Foundation, Cambridge, MA, USA; publicly available via anonymous ftp to `prep.ai.mit.edu` in directory `/pub/gnu`.