Improved recursive computation of clebsch-Gordan coefficients

Guanglang Xu*

University of Helsinki, Helsinki, Finland

ABSTRACT

Fast, accurate, and stable computation of the Clebsch-Gordan (C-G) coefficients is always desirable, for example, in light scattering simulations, the translation of the multipole fields, quantum physics and chemistry. Current recursive methods for computing the C-G coefficients are often unstable for large quantum numbers due to numerical overflow or underflow. In this paper, we present an improved method, called the sign-exponent recurrence, for the recursive computation of C-G coefficients. The result shows that the proposed method can significantly improve the stability of the computation without losing its efficiency, producing accurate values for the C-G coefficients even with very large quantum numbers.

1. Introduction

The Clebsch-Gordan (C-G) coefficients arise whenever the coupling of two angular momenta is involved. The computation of C-G coefficients has a broad range of applications including particle light scattering simulations [1–4], fast multipole methods [5,6], spherical polar Fourier transform [7], quantum physics and chemistry [8]. In the context of light scattering by small particles, for instance, the C-G coefficients are needed for the realization of analytical random orientation average using T-matrix. Developed by M. Mishchenko [1], the analytical random orientation scheme is perhaps one of the greatest advantages of T-matrix method, which could save massive amount of computational time if the orientation averaged properties are needed. In addition, the C-G coefficient also arises when one need to compute the translation of multipole fields[4]. Consequently, the importance of obtaining fast, reliable and accurate computation of C-G coefficients should not be underestimated. Currently, the computation of C-G coefficients with large quantum numbers is often based on a modified recursion method, which was originally proposed by Schulten and Gordon [9,10] and later modified and implemented by M. Mishchenko [1] and Wielaard et al.[2] in T-matrix simulations. According to [11], the modified recursive method can compute the C-G coefficients with quantum number up to 150 in a stable and accurate manner. Although the size of T-matrix up to this order is already quite large, it is always desirable to develop a highly stable and

accurate method for calculating the C-G coefficients with quantum number as large as possible. In this paper, we present an improved method which could largely extend this limitation without losing the accuracy and efficiency.

The paper is organised as follows. In Section 2, we briefly introduce the widely applied recursion method for the computation of C-G coefficients and analyse its limitations. Section 3 presents our method. In Section 4, we demonstrate the stability, accuracy, and efficiency of the proposed method by comparing the results with those from the previous methods. Section 5 concludes this study.

2. The recursive computation of C-G coefficients

The C-G coefficients are related to the Wigner's 3j symbol in accordance with

$$C\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & -m_3 \end{pmatrix} = (-1)^{j_1-j_2+m_3}\sqrt{2j_3+1}\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix},$$
(1)

where the $j_{1,2,3}$ and $m_{1,2,3}$ are angular momentum and its projection on the quantization axis, and " C " here denotes the CG-coefficients. In this study, we use the term principle and magnetic angular-momentum quantum numbers for j and m respectively, as used by Schulten and Gordon [9]. The C-G coefficient will be zero unless the following conditions are satisfied simultaneously,

$$|j_1 - j_2| \le j_3 \le (j_1 + j_2),$$
(2)

$$m_3 = -(m_1 + m_2).$$
(3)

To be consistent with Schulten and Gordon [9] on notation, we focus on the discussion of Wigner's 3*j* symbol, while the C-G coefficients can be easily derived from Eq. 1. For small quantum numbers, the 3*j* symbols can be computed conveniently using Racah's formula [12]. For large quantum numbers, however, the computation using Racah's formula becomes very expansive and suffers numerical instabilities due to the high order factorials. It is believed that the 3*j* symbols with large quantum numbers can be evaluated most efficiently with the recurrence relations. In light scattering simulations, one needs to compute the 3*j* symbols with a range of principle quantum number *j* or magnetic quantum number *m*. Because the symmetry properties of the 3*j* symbols (see Appendix D of [11]), the recurrence relations for one of the three quantum numbers shall be enough to compute the recursion of all others. Without loss of generality, let us focus on the computation of j_3 and m_2. The recurrence relations for the two quantum numbers are [9]:

$$j_3 A(j_3+1)\begin{pmatrix} j_1 & j_2 & j_3+1 \\ m_1 & m_2 & m_3 \end{pmatrix} + B(j_3)\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix}$$
$$+ (j_3+1)A(j_3)\begin{pmatrix} j_1 & j_2 & j_3-1 \\ m_1 & m_2 & m_3 \end{pmatrix} = 0, \tag{4}$$

$$C(m_2+1)\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2+1 & m_3-1 \end{pmatrix} + D(m_2)\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix}$$
$$+ C(m_2)\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2-1 & m_3+1 \end{pmatrix} = 0, \tag{5}$$

where

$$A(j_3) = \big([(j_3)^2 - (j_1-j_2)^2][(j_1+j_2+1)^2 - (j_3)^2]$$
$$\times [(j_3)^2 - (m_3)^2]\big)^{1/2}, \tag{6}$$

$$B(j_3) = -(2j_3+1)[j_1(j_1+1)m_3 - j_2(j_2+1)m_3$$
$$- j_3(j_3+1)(m_2-m_1)], \tag{7}$$

$$C(m_2) = [(j_2-m_2+1)(j_2+m_2)(j_3+m_3+1)(j_3-m_3)]^{1/2}, \tag{8}$$

$$D(m_2) = j_2(j_2+1) + j_3(j_3+1) - j_1(j_1+1) + 2m_2m_3. \tag{9}$$

For recursive computation using Eq. 4, j_3 lies in the following range

$$\max(|m_1+m_2|, |j_1-j_2|) \leq j_3 \leq (j_1+j_2). \tag{10}$$

For recursive computation using Eq. 5, m_2 lies in the following range

$$-\min(j_2, j_3+m_1) \leq m_2 \leq \min(j_2, j_3-m_1). \tag{11}$$

In the original method proposed by Schulten and Gordon [9], the recursions can start from arbitrary real number, e.g., *unity*, and go both forward and backward from the minimum and maximum quantum number respectively. The method therefore requires the computation of a scaling factor such that the forward and backward recursions give the same number at the intermediate quantum number. The coefficients can then be determined by applying the unitary properties:

$$\sum_{j_3=j_{3_{\min}}}^{j_{3_{\max}}} (2j_3+1)\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix}^2 = 1 \tag{12}$$

$$\sum_{m_2=m_{2_{\min}}}^{m_{2_{\max}}} (2j_1+1)\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix}^2 = 1 \tag{13}$$

In addition to iteration direction[9], there are two more sources that could cause inaccuracy or numerical instability via recursion. Firstly, because the recursion starts with arbitrary number, the computed values need to be scaled twice, first by the scaling factor and then by the normalization factor. The errors of the factors, possibly arising from the inaccuracy of particular coefficients or their ratios, could propagate to the whole group of the computed values. To illustrate this, we compare the exact values with those computed by scaling and normalization as displayed in Fig. 1. It can be seen that most of the computed values can be shifted by certain magnitude due to the multiplication of the factors.

The second source of errors in the existing methods is the lack of a mechanism to avoid numerical overflows or underflows. Note that in the case of high order quantum numbers, the magnitude of quantities (or their ratios) involved in the computations could be extremely small or large.

In [13], Luscombe and Luban propose to iterate the ratio of two successive 3*j* symbols to avoid numerical overflows. Nevertheless, their method is not without drawbacks. Firstly, it has to perform normailzation, which could cause unnecessary shifting of values. Secondly, in their method, the values of C-G coefficients are to be obtained by multiplication of many ratios, and this could still induce numerical overflow/underflow, even though the iteration of the ratios may have no such risks. Thirdly, the necessity of identifying classical and nonclassical regions complicates the algorithm.

One way to remove the necessity of scaling and normalization is to start with an exact value of C-G coefficient at the minimum or maximum quantum number. As described in Appendix A, there are four different cases to be considered for $j_3 = j_{3_{\min}}$, while there is only one case for $j_3 = j_{3_{\max}}$. The usage of exact starting values exclude the necessity of scaling and normalization, which improves the computational stability and accuracy. However, it does not exclude the possibility of numerical underflow/overflow, because for large quantum numbers, the factorial computations for the starting values will likely exceed the precision of the arithmetic.

3. The sign-exponent recurrence method

If the quantum numbers satisfy the conditions of Eq. 2 and Eq. 3, the 3*j* symbols are generally non-zero. However, it is well-known that some of the coefficients can be "accidentally" zero even if the conditions are fulfilled [14]. Such zeros are called non-trivial zeros. But the non-trivial zeros are quite rarely encountered, for the moment, let us assume that all the coefficients involved are non-zero. In this case, one can write arbitrary 3*j* symbols as

$$f(j_3) = \begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} = s(j_3)\exp(k(j_3)), \tag{14}$$

$$g(m_2) = \begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & -m_2-m_1 \end{pmatrix} = s(m_2)\exp(h(m_2)), \tag{15}$$

where $k(j_3)$ and $h(m_2)$ are real functions of j_3 and m_3 respectively, and

$$\begin{cases} s(j_3) = sign(f(j_3)) \\ s(m_2) = sign(g(m_2)) \end{cases} \tag{16}$$

By introducing Eq. 14 into Eq. 4, we obtain

$$s(j_3+1)\exp(k(j_3+1)) = \exp(k(j_3))[\alpha(j_3)s(j_3)$$
$$+ \beta(j_3)s(j_3-1)\exp(-\Delta(j_3))], \tag{17}$$

where

$$\Delta(j_3) = k(j_3) - k(j_3-1), \tag{18}$$
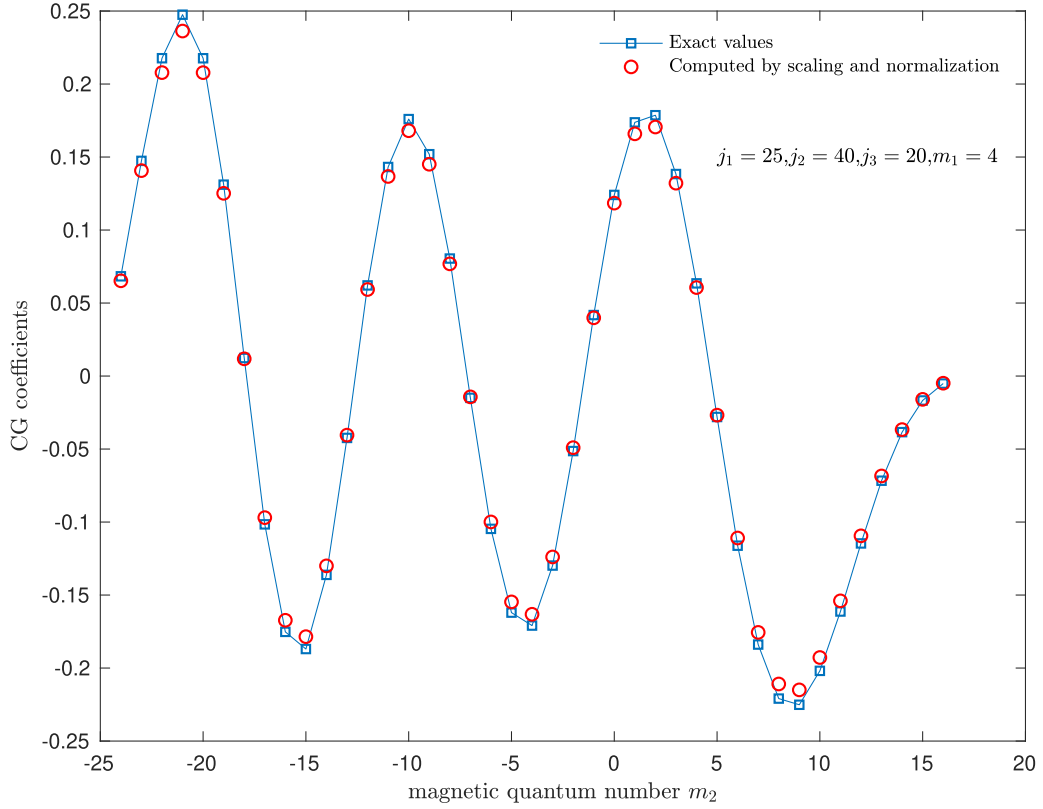
$$\alpha(j_3) = -\frac{B(j_3)}{j_3 A(j_3+1)}, \tag{19}$$

Fig. 1. Global shifting caused by scaling and normalization.

$$\beta(j_3) = -\frac{(j_3+1)A(j_3)}{j_3 A(j_3+1)}. \tag{20}$$

From Eq. 17, one can obtain the recurrence relations for both $s(j_3)$ and $k(j_3)$, i.e.,

$$\begin{cases} s(j_3+1) = sign[\alpha(j_3)s(j_3) + \beta(j_3)s(j_3-1)\exp(-\Delta(j_3))] \\ k(j_3+1) = k(j_3) + \ln|\alpha(j_3)s(j_3) + \beta(j_3)s(j_3-1) \\ \quad \exp(-\Delta(j_3))|. \end{cases} \tag{21}$$

Similarly, recurrence relation for m_2 can be obtained by introducing Eq. 15 into Eq. 5, i.e.,

$$\begin{cases} s(m_2+1) = sign[\lambda(m_2)s(m_2) + \eta(m_2)s(m_2-1) \\ \quad \exp(-\Delta(m_2))] \\ h(m_2+1) = h(m_2) + \ln|\lambda(m_2)s(m_2) + \eta(m_2)s(m_2-1) \\ \quad \exp(-\Delta(m_2))| \end{cases} \tag{22}$$

where

$$\Delta(m_2) = h(m_2) - h(m_2-1), \tag{23}$$

$$\lambda(m_2) = -\frac{D(m_2)}{C(m_2+1)}, \tag{24}$$

$$\eta(m_2) = -\frac{C(m_2)}{C(m_2+1)}. \tag{25}$$

At the starting minimum quantum number, we have

$$\begin{cases} s(j_{3_{min}}-1) = 0 \\ s(m_{2_{min}}-1) = 0. \end{cases} \tag{26}$$

Therefore the iteration becomes

$$\begin{cases} s(j_{3_{min}}+1) = sign[\alpha(j_{3_{min}})s(j_{3_{min}})] \\ k(j_{3_{min}}+1) = k(j_{3_{min}}) + \ln|\alpha(j_{3_{min}})s(j_{3_{min}})|, \end{cases} \tag{27}$$

and

$$\begin{cases} s(m_{2_{min}}+1) = sign[\lambda(m_{2_{min}})s(m_{2_{min}})] \\ h(m_{2_{min}}+1) = h(m_{2_{min}}) + \ln|\lambda(m_{2_{min}})s(m_{2_{min}})|. \end{cases} \tag{28}$$

It can be seen that we have separated the original recurrence relation into sign-recursion and exponent-recursion. Note that the above relations are valid for the forward direction which increases the quantum number. Similarly, it would be straightforward to derive the backward recurrence relation that reduces the quantum number. The basic idea behind such recurrence relations is that we could focus on computing the sign and exponent of the coefficient, rather than the coefficient itself. In principle, as long as the sign and exponent are computed accurately, the coefficient can always be calculated accurately via Eq. 14 or Eq. 15. Because the 3j symbols can vary many orders of magnitude, the sign-exponent recurrence can significantly reduce the risk of numerical underflows/overflows. To apply the derived relations, we just need to compute the starting exponent and sign of the coefficient at the minimum or maximum quantum numbers. According to our numerical tests, the method would not induce numerical underflow/overflow even with quantum number larger than 10 million.

For now, we shall consider the problem of encountering non-trivial zeros. Once zero-value is encountered, the *sign* value will become 0 and the exponent will become $-\infty$, making the computation of $\Delta(m_2)$ or $\Delta(j_3)$ meaningless. Therefore, the above recurrence relations must be avoided and the original three-term linear recurrence relations should be applied. The condition for using the sign-exponent recursions shall be that both $s(j_3-1)$ and $s(j_3)$ are non-zero (same for m_2 case), otherwise the three-term linear relations should be invoked. For large quantum numbers, the non-trivial zeros are rarely encountered. For most of the cases, only the sign-exponent iteration is invoked in the computation.
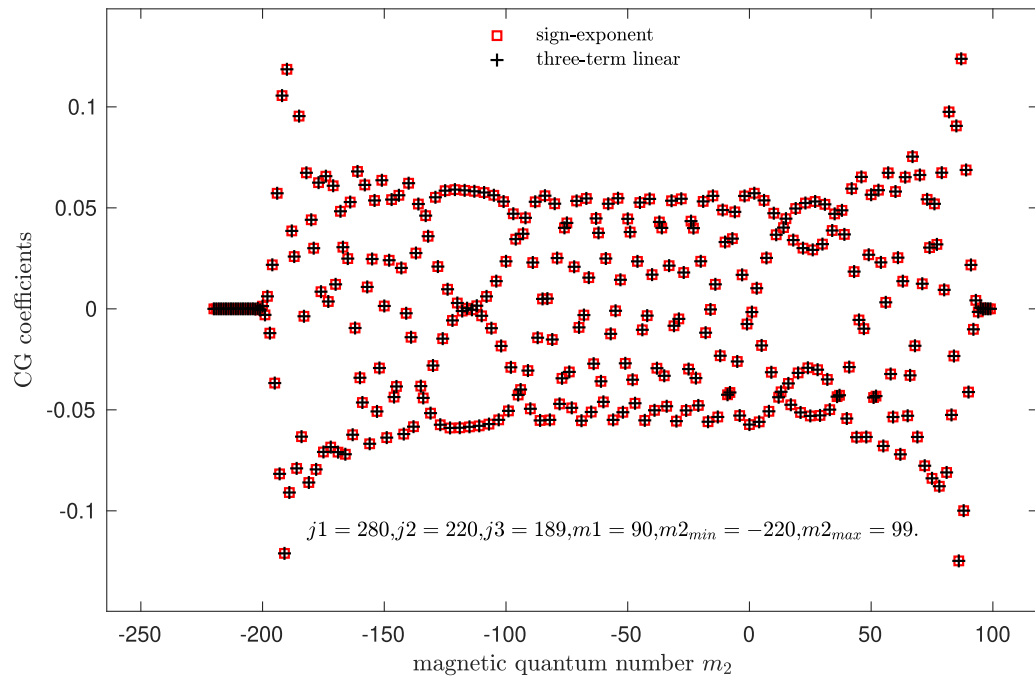
Fig. 2. A test comparison between the widely-applied three-term linear recursion and the sign-exponent recursion method introduced in this study. The values are scaled back to C-G coefficients. The black *plus − sign* marks the values computed by the three-term linear recurrence method, while the red *circle* marks the values computed by the sign-exponent recurrence method. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 3. Test comparison between the three-term linear recurrence and sign-exponent recurrence for large quantum numbers. The values have been scaled back to C-G coefficients. The *red* dot marks the values from the sign-exponent recurrence method, while the *black* dot marks the values from the three-term linear recurrence method. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)
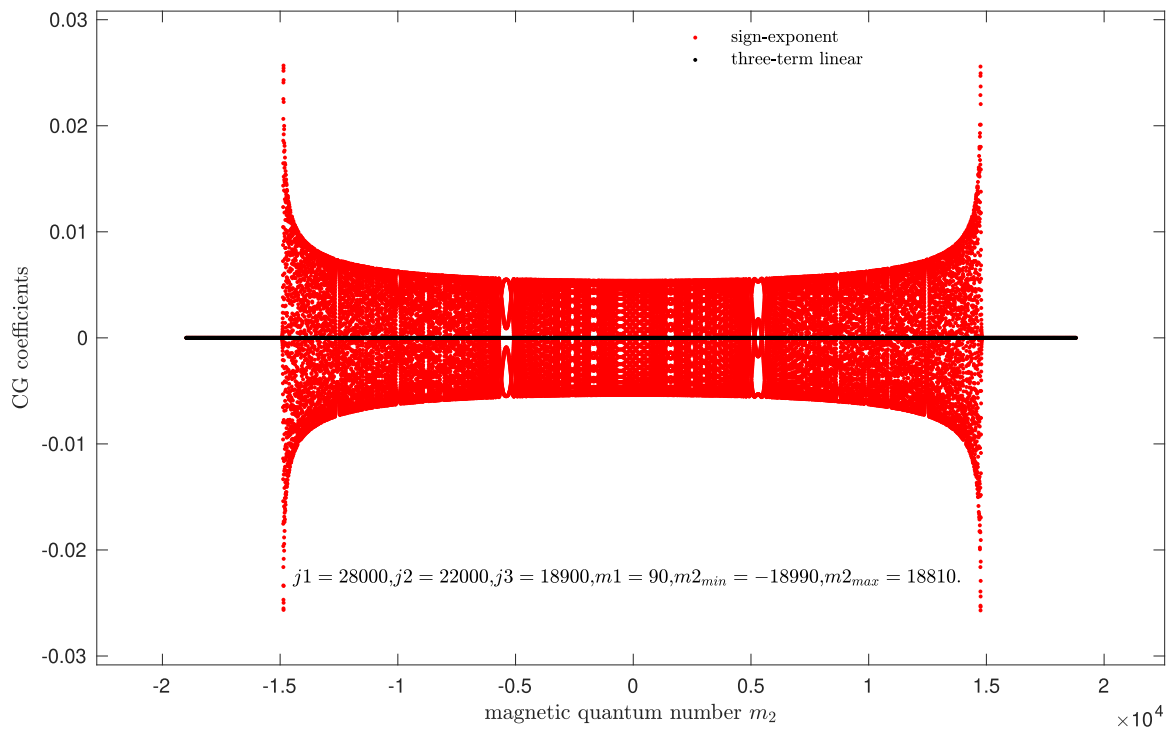
4. Results and discussion

In this section, we shall discuss the stability, accuracy, and efficiency of the proposed method by comparing with the widely applied three-term linear recurrence with exact starting values. For

large quantum numbers, in general, all recursion-based methods are much faster than those using the direct definition or formula. The efficiency of the proposed method is almost the same as the previous recursion methods. To quantify the accuracy of the computation, we define the following error term, which is consistent

with [4].

$$R = |1 - \sum_{m_2=m_{2_{min}}}^{m_{2_{max}}} (2j_1+1) \begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix}^2| \tag{29}$$

Please note that all the results are obtained by applying the double-precision arithmetic. We compute the starting values for m_2 recurrence by firstly using the recurrence relation for j_3, therefore the comparison on m_2 recurrence is preferred. The following figure displays a comparison between the three-term linear recurrence and sign-exponent recurrence method.

It can be seen that the results of the two are indistinguishable from Figure 2. In fact, for the three-term linear recursion, $R = 1.0334 \times 10^{-12}$, while for the sign-exponent recursion, $R = 1.0214 \times 10^{-12}$. The values of R suggest that the sign-exponent recurrence method has at least the same performance as the three-term linear recurrence on numerical accuracy. The biggest advantage of the proposed method perhaps is that it can avoid numerical underflow when computing the starting values and this is crucial for dealing with large quantum numbers because the starting values could be extremely small. Fig. 3 demonstrates the comparison for the case of very large quantum numbers. The three-term linear recurrence suffers numerical underflow and becomes zero throughout the iteration. On the contrary, the sign-exponent recurrence method excludes the risk of obtaining zero starting values. Consequently, for the three-term linear recurrence, $R = 1$, while for the sign-exponent recursion, $R = 5.9769 \times 10^{-10}$, which still maintains high accuracy. Based on our extensive tests, the proposed method is generally much more stable than the original linear recursion method, while having the same level of numerical accuracy and efficiency. To further demonstrate the accuracy and stability of our method, in Appendix B, we provide a link to our Matlab code and more test examples in comparisons with the most accurate package Python SymPy.

5. Conclusion

In this paper, an improved recursion method for computing the C-G coefficients is introduced. Specifically, the method separates the recursion process into sign-recursion and exponent-recursion, while the C-G value itself is not involved in the recursion except a non-trivial zero occurs. The C-G values can be obtained after the computation of their signs and exponents. By using the sign-exponent recursion, the method removes the risk of generating numerical overflows or underflows. The results presented in this paper, together with our extensive tests, have shown that the sign-exponent recurrence method is in general more stable than the original three-term linear recurrence method, while having the same level of accuracy and efficiency. As the sign-exponent method provides a mechanism to avoid numerical overflows/underflows, it could also potentially improve the computation of multipole fields and their translations and this would be a future direction to pursue.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

CRediT authorship contribution statement

Guanglang Xu: Conceptualization, Methodology, Software, Writing - original draft.

Acknowledgement

Appendix A. The computation of starting values

The recursive computation without using the scaling and normalization relies on the exact computation of the starting values. For backward recursion, the starting value is unique, i.e.,

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

$$\times \left[\frac{(2j_1)!(2j_2)!(j_1+j_2-m_1-m_2)!(j_1+j_2+m_1+m_2)!}{(2j_1+2j_2+1)!(j_1-m_1)!(j_1+m_1)!(j_2-m_2)!(j_2+m_2)!} \right]^{1/2} \tag{A.1}$$

For the forward iteration, we have four possibilities, depending on the values of $j_{3_{min}}$, i.e.,

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

$$\times \left[\frac{(2j_1-2j_2)!(2j_2)!(j_1-m_1)!(j_1+m_1)!}{(j_1-j_2-m_1-m_2)!(j_2-m_2)!(j_2+m_2)!(j_1-j_2+m_1+m_2)!(2j_1+1)!} \right]^{1/2} \tag{A.2}$$

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

$$\times \left[\frac{(2j_2-2j_1)!(2j_1)!(j_2-m_1)!(j_2+m_1)!}{(j_2-j_1-m_1-m_2)!(j_1-m_1)!(j_1+m_1)!(j_2-j_1+m_1+m_2)!(2j_2+1)!} \right]^{1/2} \tag{A.3}$$

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

$$\times \left[\frac{(j_1+m_1)!(j_2+m_2)!(j_2+j_1-m_1-m_2)!(2m_1+2m_2)!}{(j_1-m_1)!(j_2-m_2)!(j_1-j_2+m_1+m_2)!(j_2-j_1+m_1+m_2)!(j_1+j_2+m_1+m_2+1)!} \right]^{1/2} \tag{A.4}$$

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

$$\times \left[\frac{(j_1-m_1)!(j_2-m_2)!(j_2+j_1+m_1+m_2)!(-2m_1-2m_2)!}{(j_1+m_1)!(j_2+m_2)!(j_1-j_2-m_1-m_2)!(j_2-j_1-m_1-m_2)!(j_1+j_2-m_1-m_2+1)!} \right]^{1/2} \tag{A.5}$$

In the computation of the above values, we can compute the logarithm of the factorials to avoid overflow. To be more specific, one can write arbitrary starting value as

$$f(j_{start}) = s(j_{start}) \exp(k(j_{start})) = (-1)^l \left[\frac{a_1! a_2! a_3! a_4!}{b_1! b_2! b_3! b_4! b_5!} \right]^{1/2}$$
(A.6)

Obviously, the starting values for s and k functions are

$$s(j_{start}) = (-1)^l$$
(A.7)

$$k(j_{start}) = \frac{1}{2} [\ln(a_1!) + \ln(a_2!) + \ln(a_3!) + \ln(a_4!)$$
$$- \ln(b_1!) - \ln(b_2!) - \ln(b_3!) - \ln(b_4!) - \ln(b_5!))]$$
(A.8)

To compute the logarithm of the factorials, we may use the following formula to avoid numerical overflow,

$$\ln(N!) = \sum_{i=1}^{N} \ln(i).$$
(A.9)

Alternatively, a formula (Eq. 23) in [4] can be applied.

Appendix B. Comparison with Python library SymPy

To demonstrate the accuracy of the proposed method, we compile some of the results from Python library SymPy and compare with the proposed method in this study. The code for our method can be accessed from [15]. The code for SymPy can be obtained from [16]. The SymPy is based on symbolic manipulation, which could be considered as the most accurate method. The computational time by the recursion methods ranges between 0.01 to 0.03 seconds for most of the cases with quantum number smaller than 1000, while the SymPy package could take up to roughly a few seconds. It can be seen that the two recursion methods are both very accurate comparing to SymPy and their levels of accuracy are pretty much the same. However, as we increase the quantum numbers, the three-term linear recursion becomes unstable and generates zeros due to numerical underflows. The Python SymPy simply produces errors at very large quantum numbers, while the sign-exponent method is still stable and produces reasonable results.

$j_1 = 280$, $j_2 = 220$, $j_3 = 189$, $m_1 = 90$

m_2	Sign-Exponent	Three-Term Linear	Python SymPy
−120	0.002887948213256	0.002887948213256	0.00288794821325701
−125	0.058508415288557	0.058508415288558	0.0585084152885739
−128	0.020928845109162	0.020928845109162	0.020928845109168
−130	-0.028070293415027	-0.028070293415028	-0.0280702934150357
−135	-0.038257492867934	-0.038257492867934	-0.038257492867945

(continued on next page)

$j_1 = 480$, $j_2 = 320$, $j_3 = 300$, $m_1 = 90$

m_2	Sign-Exponent	Three-Term Linear	Python SymPy
−125	-0.046262518791471	-0.046262518791468	-0.0462625187915161
−128	-0.041062246328595	-0.041062246328592	-0.0410622463286351
−120	0.041718263408549	0.041718263408546	0.0417182634085898
−130	0.047752989423874	0.047752989423870	0.0477529894239198
−135	-0.047799931849649	-0.047799931849645	-0.0477999318496948

$j_1 = 700$, $j_2 = 620$, $j_3 = 230$, $m_1 = 300$

m_2	Sign-Exponent	Three-Term Linear	Python SymPy
−200	-0.029578200668109	-0.029578200668109	-0.0295782006677839
−250	-0.033722189882640	-0.033722189882640	-0.0337221898822695
−300	-0.000885723206928	-0.000885723206928	-0.0008857232069200
−350	0.032668945677003	0.032668945677002	0.0326689456767071
−400	0.032449523658905	0.032449523658904	0.0324495236586107

$j_1 = 7000$, $j_2 = 6200$, $j_3 = 2300$, $m_1 = 3000$

m_2	Sign-Exponent	Three-Term Linear	Python SymPy
−2000	0.001244977301861	0.001244977301860	Error.
−2500	-0.007275107384171	-0.007275107384171	Error.
−3000	0.002712153629703	-0.000000000000000	Error.
−3500	0.006665616564930	-0.000000000000000	Error.
−4000	-0.010583441967577	-0.000000000000000	Error.

$j_1 = 9000000$, $j_2 = 620000$, $j_3 = 7800000$, $m_1 = 3000000$

m_2	Sign-Exponent	Three-Term Linear	Python SymPy
−2000000	-0.000258969176674	0.000000000000000	Error.
−2500000	-0.000236627240189	-0.000000000000000	Error.
−3000000	-0.000146400518567	-0.000000000000000	Error.
−3500000	-0.000144722745846	0.000000000000000	Error.
+3500000	0.000396746931467	-0.000000000000000	Error.

References

[1] Mishchenko M. Light scattering by randomly oriented axially symmetric particles. JOSA A 1991;8(6):871–82.

[2] Wielaard DJ, Mishchenko MI, Macke A, Carlson BE. Improved t-matrix computations for large, nonabsorbing and weakly absorbing nonspherical particles and comparison with geometrical-optics approximation. Appl Opt 1997;36(18):4305–13.

[3] Mackowski DW, Mishchenko MI. Calculation of the t matrix and the scattering matrix for ensembles of spheres. JOSA A 1996;13(11):2266–78.

[4] Xu Y-L. Efficient evaluation of vector translation coefficients in multiparticle light-scattering theories. J Comput Phys 1998;139(1):137–65.

[5] Danos M, Maximon L. Multipole matrix elements of the translation operator. J Math Phys 1965;6(5):766–78.

[6] Epton MA, Dembart B. Multipole translation theory for the three-dimensional laplace and helmholtz equations. SIAM Journal on Scientific Computing 1995;16(4):865–97.

[7] Ritchie DW. High-order analytic translation matrix elements for real-space six-dimensional polar fourier correlations. J Appl Crystallogr 2005;38(5):808–18.

[8] Schulten K, Gordon RG. Semiclassical approximations to 3 j-and 6 j-coefficients for quantum-mechanical coupling of angular momenta. J Math Phys 1975;16(10):1971–88.

[9] Schulten K, Gordon R. Recursive evaluation of 3j and 6j coefficients. Comput Phys Commun 1976;11(2):269–78.

[10] Schulten K, Gordon RG. Exact recursive evaluation of 3 j-and 6 j-coefficients for quantum-mechanical coupling of angular momenta. J Math Phys 1975;16(10):1961–70.

[11] Mishchenko MI, Travis LD, Lacis AA. Scattering, absorption, and emission of light by small particles. Cambridge university press; 2002.

[12] Racah G. Theory of complex spectra. ii. Physical Review 1942;62(9–10):438.

[13] Luscombe JH, Luban M. Simplified recursive algorithm for wigner 3 j and 6 j symbols. Physical Review E 1998;57(6):7274.

[14] Heim T, Hinze J, Rau A. Some classes of nontrivial zeroes of angular momentum addition coefficients. J Phys A: Math Theor 2009;42(17):175203.

[15] https://www.github.com/GXhelsinki/Clebsch-Gordan-Coefficients-.

[16] SymPy. https://docs.sympy.org/latest/modules/physics/quantum/cg.html.