

文章编号: 1000-5641(2019)01-0076-07

整数阶复宗量贝塞尔函数的计算程序研究

任宏红¹, 郭迎春¹, 王兵兵²

(1. 华东师范大学 物理与材料科学学院, 上海 200241;

2. 中国科学院物理研究所 凝聚态物理国家重点实验室 光物理实验室, 北京 100190)

摘要: 鉴于目前的算法程序集中没有现成的计算复宗量贝塞尔函数的程序, 本文基于贝塞尔函数的逆向递推关系编写了计算整数阶复宗量第一类贝塞尔函数的 Fortran 程序源代码. 与 Matlab 软件的计算结果比较, 两者至少有 12 位有效数字一致. 接着运用此程序, 分析了徐士良的《FORTRAN 常用算法程序集》中的纯虚宗量的贝塞尔函数, 即变形贝塞尔函数程序的准确度, 发现其准确度为 6 位有效数字. 最后, 对基于实宗量贝塞尔函数和纯虚宗量贝塞尔函数相乘然后用无限求和来计算复宗量贝塞尔函数值的方法的准确性进行了探讨. 证明其仅能对有限的贝塞尔函数进行准确计算. 这是由于当求和项中有远大于最终的求和项时, 会导致求和结果的有效数字减少甚至完全错误.

关键词: 复宗量贝塞尔函数; Fortran 源程序; 递推计算

中图分类号: O411.2 **文献标志码:** A **DOI:** 10.3969/j.issn.1000-5641.2019.01.009

Program for calculating the integer order of Bessel functions with complex arguments

REN Hong-hong¹, GUO Ying-chun¹, WANG Bing-bing²

(1. School of Physics and Materials Science, East China Normal University, Shanghai 200241, China;

2. Laboratory of Optical Physics, Beijing National Laboratory of Condensed Matter Physics, Institute of Physics, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Fortran source code for calculating the integer order of Bessel functions of the first kind with complex arguments is presented. The method is based on the backward recurrence relation of Bessel functions. Values of the Bessel function generated by our program are in-agreement with the values generated by Matlab to at least 12 significant digits. We use the program to calculate the integer order of Bessel functions of the first kind with pure imaginary arguments, provided by Xu Shiliang's Fortran algorithm assembly. The results show that the first 6 significant digits are accurate. We also analyze the algorithm for calculating Bessel functions with complex arguments, which use the infinite sum of the product of the real arguments of the Bessel function and the pure imaginary

收稿日期: 2017-12-20

基金项目: 国家自然科学基金 (61275128, 11474348)

第一作者: 任宏红, 女, 硕士研究生, 研究方向为理论物理.

通信作者: 郭迎春, 女, 副教授, 研究方向为强场与物质相互作用. E-mail: ycguo@phy.ecnu.edu.cn.

arguments of the Bessel function, provided by Xu Shiliang's algorithm assembly. The results show that this algorithm does not always get accurate values for Bessel functions with complex arguments. The reason lies with the fact that the term in the sum larger than the function value causes the loss of significant digits.

Keywords: Bessel function with complex arguments; Fortran source code; backward recurrence relation of Bessel function

0 引 言

整数阶贝塞尔函数是许多辐射、散射和波导问题的数学解, 贝塞尔函数中复宗量以及虚宗量和材料损耗、漏波等相关. 贝塞尔函数也应用于电磁场与物质相互作用的相关现象的数学描述, 如阈上电离及高次谐波等^[1-3]. 所以贝塞尔函数的精确计算尤为重要.

针对整数阶第一类贝塞尔函数, 在常用算法程序书中, 例如徐士良的《FORTRAN 常用算法程序集》^[4], 以及《Numerical Recipes: The Art of Scientific Computing》^[5]等已经有实宗量的贝塞尔函数的程序和纯虚宗量的贝塞尔函数即变形的贝塞尔函数的程序, 但是还没有计算复宗量贝塞尔函数的程序. Du Toit^[6]探讨了复宗量的贝塞尔函数的计算方法, 提出了采用逆向贝塞尔函数递推关系进行计算, 文中给出了一些典型的数值结果, 指出这种方法对于大的宗量不能计算, 而对于大的宗量, 采用幂级数展开截断的方法给出了结果. 魏彦玉等人^[7]采用级数展开方法对虚宗量的贝塞尔函数进行了计算, 对小的宗量计算结果很好. 张爽等人^[8]针对这种方法的计算进行了误差分析, 指出了此方法不适合大宗量贝塞尔函数计算的原因. 张善杰等人^[9]对任意实数阶复宗量贝塞尔函数的递推算法进行了研究, 并探讨了如何验证程序的正确性和分析了计算结果的精度.

本文针对第一类整数阶复宗量贝塞尔函数, 重新考察了基于逆递推公式对贝塞尔函数的计算, 编写了适用于任意大小的复宗量的第一类整数阶贝塞尔函数 Fortran 源代码. 本程序计算的贝塞尔函数的结果与 Matlab 软件结果进行比较, 二者一致到第 12 位有效数字. 由于常用算法程序书(如文献[4])中, 很容易找到实宗量的贝塞尔函数程序和纯虚宗量贝塞尔函数的程序, 基于二者相乘求和, 原则上可以计算复宗量的贝塞尔函数. 本文探讨了这种方法计算贝塞尔函数的可行性. 在这之前还探讨了算法程序集中变形贝塞尔函数的计算精度.

本文安排如下, 第 1 节简要阐述复宗量贝塞尔函数的理论基础, 并举例比较此程序的结果与 Matlab 的结果; 第 2 节运用给出的程序分析变形贝塞尔函数程序的准确度; 第 3 节分析基于实宗量贝塞尔函数和虚宗量贝塞尔函数相乘无穷求和的方法, 计算复宗量贝塞尔函数的可行性; 附录中给出源代码.

1 计算复宗量的贝塞尔函数的原理

贝塞尔函数满足递推关系:

$$J_{n-1}(z) = \frac{2n}{z} J_n(z) - J_{n+1}(z), \quad (1)$$

令

$$J_n(z) = \frac{B_n(z)}{S}, \quad (2)$$

从而有

$$B_{n-1}(z) = \frac{2n}{z} B_n(z) - B_{n+1}(z). \quad (3)$$

因为 $J_n(z)$ 随 $|z|$ 的增加而减小, 所以采用逆向的递推关系, 即令 $B_q(z) = 1$, $B_{q+1}(z) = 0$, 忽略大于 q 阶的所有项, 反复运用上式计算可得到 $B_n(z)$.

下面计算式 (2) 中的 S . 因为贝塞尔函数有下列性质:

$$1 = J_0(z) + 2 \sum_{k=1}^{\infty} J_{2k}(z), \quad (4)$$

$$\cos z = J_0(z) + 2 \sum_{k=1}^{\infty} J_{2k}(z)(-1)^k, \quad (5)$$

所以 S 满足:

$$S = B_0(z) + 2 \sum_{k=1}^{\infty} B_{2k}(z) \approx B_0(z) + 2 \sum_{k=1}^{q/2} B_{2k}(z), \quad (6)$$

$$S \cos z = B_0(z) + 2 \sum_{k=1}^{\infty} B_{2k}(z)(-1)^k \approx B_0(z) + 2 \sum_{k=1}^{q/2} B_{2k}(z)(-1)^k. \quad (7)$$

当 $\text{Im}(z) \leq 1$ 时, 采用式 (6) 计算 S , 当 $\text{Im}(z) > 1$ 时, 采用式 (7) 计算 S . 这样做可以保障 S 中无穷求和的每一项都小于 S 值, 从而不会带来有效数字的减少.

文献 [5] 指出, 编程计算的误差有舍入误差(round off error)、截止误差(truncation error)和计算的稳定度误差(stability error). 舍入误差是由计算机精度所决定的, 如双精度型的变量含 16 位有效数字, 误差是最后一位有效数字的量级. 此数据精度所引入的误差为舍入误差. 舍入误差一般会随计算次数的增加而增加, 一般会影响结果的最后两位有效数字. 在某些特殊情况下会引入很大的误差, 从而减少有效数字. 如两个几乎相等的同型数据相减. 截止误差是由计算过程中将无穷求和截取为有限求和而引入的误差. 稳定度误差是由于计算方法不当(称为不稳定误差)使最初阶段的舍入误差被连续放大而导致的误差, 这种误差甚至会淹没真实结果. 减小这三种误差是我们程序设计的出发点, 文献 [6] 已证实我们所采用的这种逆递推方法是稳定的. 不同情况下选取不同的 S , 目的是减小舍入误差. q 的选取决定了截止误差, 只要 q 值取得足够大, 截止误差就会变小, 直到小于舍入误差. 具体表现为, q 增大到某值后, 如果再继续增大, 计算的结果将保持不变. 据此, 为计算准确又节省时间, 我们将 q 取值为

$$q = n + \sqrt{40n} + 100 + 2|z|. \quad (8)$$

可见, 贝塞尔函数的阶数越大, 宗量的模越大, 需要的 q 值就越大.

计算结果表明, 我们的结果和 Matlab 软件的结果一致到至少 12 位有效数字, 另外, 我们计算了文献 [4] 表 1 中的几个典型数据, 再现了文献的全部结果, 并给出了文献中没有计算出的结果, 列在表 1 中. 对于小的宗量, 小的阶数, 由公式 (8) 可见, q 值会相对很小, 此时不仅 q 所决定的截止误差小于舍入误差, 由于 q 相对小, 计算的次数少, 和大阶数大宗量的情况

相比, 舍入误差也不会大, 所以计算的结果应该更为精确. 如表 1 中的前两行所示, 我们计算的结果与 Matlab 的结果一致到 14 位有效数字. 文献 [4] 强调这种基于逆向递推公式的方法不适合计算大的宗量, 认为计算如 $10\,000\,000+i333$ 大宗量的贝塞尔函数是不可行的, 该文献提出运用幂级数展开截断的方法来计算大宗量贝塞尔函数, 而我们就是采用逆递推公式的方法, 将如 $10\,000\,000+i333$ 这样大宗量的贝塞尔函数计算了出来. 得到的结果与该文献中基于展开截断的方法的结果基本一致 (见表 1). 我们认为文献中采用这种逆向递推的方法没有计算出来的原因, 一方面是我们的逆递推起点 q 值比文献中的大, 还有可能是计算机的性能的局限造成的.

表 1 $J_n(z)$ 的计算举例Tab. 1 Numerical examples of $J_n(z)$

n	z	本文结果	[1] 中基于逆向递推的结果
5	$18+0i$	$-0.155\,370\,098\,779\,049$	$-0.155\,370\,098\,779\,049$
5	$0+18i$	$i3.057\,827\,717\,566\,10 \times 10^6$	$i3.057\,827\,717\,566\,10 \times 10^6$
50	$-180+100i$	$3.235\,775\,172\,706\,374 \times 10^{40} - i2.054\,938\,087\,958\,093 \times 10^{40}$	$3.235\,775\,172\,706\,37 \times 10^{40} - i2.054\,938\,087\,958\,09 \times 10^{40}$
600	$-4\,000-300i$	$-2.767\,691\,224\,061\,821 \times 10^{126} + i3.172\,834\,828\,322\,395 \times 10^{126}$	$-2.767\,691\,224\,061\,89 \times 10^{126} + i3.172\,834\,828\,322\,395 \times 10^{126}$
1 000	$600-200i$	$5.373\,339\,700\,916\,123 \times 10^{-104} + i1.658\,289\,459\,267\,489 \times 10^{-103}$	$5.373\,339\,700\,915\,97 \times 10^{-104} + i1.658\,289\,459\,267\,45 \times 10^{-103}$
0	$10\,000\,000+333i$	$-1.810\,334\,213\,201\,858 \times 10^{140} - i4.938\,482\,320\,281\,676 \times 10^{140}$	Not computable with this algorithm
4 090	$10\,000\,000+333i$	$4.878\,566\,679\,363\,567 \times 10^{140} - i1.965\,697\,327\,877\,443 \times 10^{140}$	Not computable with this algorithm
3 300	$3\,000+10i$	$-3.031\,921\,315\,712\,407 \times 10^{-42} - i2.477\,259\,163\,766\,169 \times 10^{-41}$	$-3.031\,921\,315\,712\,61 \times 10^{-42} - i2.477\,259\,163\,766\,18 \times 10^{-41}$
n	z	[1] 中基于幂级数展开截断的结果	Matlab 的结果
5	$18+0i$	$-0.155\,370\,098\,779\,049$	$-0.155\,370\,098\,779\,049$
5	$0+18i$	$-3.383\,628\,015\,819\,71 \times 10^{-8} + i3.057\,827\,717\,566\,10 \times 10^6$	$1.872\,379\,463\,330\,692 \times 10^{-9} + i3.057\,827\,717\,566\,103 \times 10^6$
50	$-180+100i$	$3.235\,775\,172\,710\,25 \times 10^{40} - i2.054\,938\,087\,958\,88 \times 10^{40}$	$3.235\,775\,172\,706\,450 \times 10^{40} - i2.054\,938\,087\,958\,026 \times 10^{40}$
600	$-4\,000-300i$	$-2.767\,691\,223\,671\,43 \times 10^{126} + i3.172\,834\,828\,322\,395 \times 10^{126}$	$-2.767\,691\,224\,059\,726 \times 10^{126} + i3.172\,834\,828\,323\,653 \times 10^{126}$
1 000	$600-200i$	Not computable with this algorithm	$5.373\,339\,700\,915\,873 \times 10^{-104} + i1.658\,289\,459\,267\,151 \times 10^{-103}$
0	$10\,000\,000+333i$	$-1.810\,334\,215\,650\,01 \times 10^{140} - i4.938\,482\,319\,383\,89 \times 10^{140}$	$-1.810\,334\,213\,201\,893 \times 10^{140} - i4.938\,482\,320\,281\,608 \times 10^{140}$
4 090	$10\,000\,000+333i$	$4.878\,566\,815\,077\,401 \times 10^{140} - i1.965\,697\,328\,649\,25 \times 10^{140}$	$4.878\,566\,679\,363\,453 \times 10^{140} - i1.965\,697\,327\,877\,465 \times 10^{140}$
3 300	$3\,000+10i$	Not computable with this algorithm	$-3.031\,921\,315\,712\,416 \times 10^{-42} - i2.477\,259\,163\,766\,451 \times 10^{-41}$

2 变形贝塞尔函数的准确度

文献 [4] 中给出了计算变形贝塞尔函数 (即纯虚数的贝塞尔函数) 的程序, 下面将采用第 1 节的程序来探讨此变形贝塞尔函数的准确度.

变形的贝塞尔函数 $I_n(y)$ 与纯虚数的贝塞尔函数 $J_n(iy)$ 是等价的, 它们的关系是

$$J_n(iy) = i^n I_n(y), \quad (9)$$

程序集中 $I_n(y)$ 的计算和第 1 节一样, 也是采用逆向的贝塞尔函数递推公式, 即

$$I_{n-1}(y) = \frac{2n}{y} I_n(y) - I_{n+1}(y). \quad (10)$$

令

$$I_n(y) = \frac{B_n}{S}, \quad (11)$$

并令 $B_q(y) = 1$, $B_q(y) = 0$, 忽略大于 q 阶的所有项, 反复运用上式计算得到 $B_q(y)$, 与第 2 部分中我们的做法不同的是, S 不是采用公式 (6) 和 (7) 求得, 而是运用拟合的方法首先计算出 $I_0(y)$,

从而有

$$S = \frac{B_0(y)}{I_0(y)}. \quad (12)$$

所以 $I_n(y)$ 的准确度不仅取决于 q 是否足够大, 还取决于 $I_0(y)$ 的准确度. 所以, 我们采用第 1 节提供的程序对 $I_0(y)$ 进行了计算, 结果列于表 2.

表 2 $I_0(y)$ 的计算举例

Tab. 2 Numerical examples of $I_0(y)$

y	本文第 1 节的结果	程序集 [4] 中的结果
1	1.266 065 877 752 008	1.266 065 848 034 260
71	3.243 091 297 746 851E+029	3.243 089 889 079 756E+029
141	5.783 741 126 182 957E+059	5.783 738 566 300 272E+059
211	1.188 946 782 454 909E+090	1.188 946 347 893 052E+090
281	2.591 193 569 443 138E+120	2.591 192 781 417 313E+120
351	5.831 422 925 433 985E+150	5.831 421 416 694 420E+150
421	1.339 291 152 932 047E+181	1.339 290 852 152 675E+181
491	3.119 396 717 782 502E+211	3.119 396 099 566 693E+211
561	7.340 570 892 169 514E+241	7.340 569 591 071 896E+241
631	1.741 003 196 211 549E+272	1.741 002 917 182 085E+272
701	4.154 898 016 095 451E+302	4.154 897 408 481 159E+302

可见程序集中 $I_0(y)$ 的结果准确到第 6 位有效数字. 进而确定程序集中的纯虚数的贝塞尔函数即变形的贝塞尔函数 $I_n(y)$ 的准确度为 6 个有效数字, 远小于第 1 节提供的 12 位有效数字.

3 无穷求和计算贝塞尔函数

由于程序集 [4] 中已经提供了实宗量的贝塞尔函数的程序以及纯虚宗量的贝塞尔函数的程序, 很自然想到由下面的无穷求和 (式 (13)) 来计算复宗量的贝塞尔函数:

$$J_n(z = x + iy) = \sum_{k=-\infty}^{+\infty} J_k(x) I_{n-k}(y) (i)^{n-k}. \quad (13)$$

所以这一节, 我们将探讨基于这种无穷求和来计算复宗量贝塞尔函数的准确度.

通过与第 1 节中提供的程序计算结果的比较发现, 采用无穷求和来计算的结果仅仅在有限范围内是正确的. 一般来讲, 对于确定的自变量, 阶数变大时, 计算结果将会不准确. 我们定义准确的结果指的是计算结果的有效数字与第 1 节提供的程序的结果达到前 6 位一致. 表 3 给出了几个特定自变量, 无穷求和方法能给出正确结果的贝塞尔函数的阶数范围, 如: 对于 $J_n(50 + i100)$, 当 $-21 < n < 21$ 时, 无穷求和能够给出正确的结果准确到 6 位有效数字.

无穷求和的方法会给出不好的结果的原因在于式子 (13) 左边的贝塞尔函数值小于求和中的个别的项值, 从而减少了最终求和中准确的有效数字的个数. 例如

$$J_{35}(50 + i40) = \sum_{k=-\infty}^{+\infty} J_k(50) I_{35-k}(40) (i)^{35-k}. \quad (14)$$

表 3 运用程序集 [4] 中的现有程序结合公式 (13) 能准确计算 $J_n(x + iy)$ 的 n 的范围

Tab. 3 The range of the orders of $J_n(x + iy)$ which can be accurately calculated based on Eq. 13 and programs from [4]

(x, y)	n 的范围	(x, y)	n 的范围	(x, y)	n 的范围	(x, y)	n 的范围
(1,1)	$(-\infty, +\infty)$	(40,1)	$(-\infty, +\infty)$	(100,1)	$(-\infty, +\infty)$	(500,1)	$(-\infty, +\infty)$
(1,10)	$(-\infty, +\infty)$	(40,10)	$(-131, 131)$	(100,10)	$(-51, 51)$	(500,10)	$(-\infty, +\infty)$
(1,50)	$(-\infty, +\infty)$	(40,50)	$(-41, 41)$	(100,50)	$(-21, 21)$	(500,50)	$(-246, 246)$
(1,80)	$(-\infty, +\infty)$	(40,80)	$(-46, 46)$	(100,80)	$(-26, 26)$	(500,80)	$(-56, 56)$
(1,120)	$(-\infty, +\infty)$	(40,120)	$(-56, 56)$	(100,120)	$(-41, 41)$	(500,120)	$(-56, 56)$
(1,300)	$(-\infty, +\infty)$	(40,300)	$(-131, 131)$	(100,300)	$(-61, 61)$	(500,300)	$(-61, 61)$

$J_{35}(50 + i40)$ 无穷求和方法给出的结果是: $-5.142\ 06 - i2.859\ 89$, 相较于第 1 节给出的结果 $-5.141\ 64 - i3.226\ 74$, 在第一位有效数字就有误差, 原因在于求和项中, 很多项远远超过求和后的结果, 如: k 为 45 时, $J_k(50)I_{35-k}(40)(i)^{35-k} = -i8.554\ 382\ 770\ 939\ 035 \times 10^8$, 由第 2 节的讨论可知, 此数据前 6 位有效数字是准确的, 十位上的数据是不准确的, 所以求和后的结果, 十位以后都是不准确的. 而 $J_{35}(50 + i40)$ 的结果的虚部最大在个位上, 所以造成了结果的完全错误.

4 结 论

我们基于逆向的贝塞尔函数的递推关系, 给出了计算第一类复宗量整数阶贝塞尔函数的 Fortran 源代码, 探讨了现有的算法程序集^[4]中的纯虚数的贝塞尔函数的准确度为 6 个有效数字, 针对运用算法程序集中的实宗量的贝塞尔函数和纯虚宗量贝塞尔函数的程序, 采用它们的乘积对阶数的无穷求和来计算复宗量的贝塞尔函数的方法, 讨论了其计算的准确度, 通过与本文所给程序的结果比较, 得出其仅能对有限的贝塞尔函数进行计算. 这是由于无穷求和项中有大于最终求和的项而导致结果的准确的有效数字减少甚至结果完全错误, 这也是物理量的求和计算中需要注意的问题.

[参 考 文 献]

- [1] LEWENSTEIN M, BALCOU P, IVANOV M Y, et al. Theory of high harmonic generation by low frequency laser fields [J]. Physical Review A, 1994, 49(3): 2117-2132.
- [2] GUO Y, FU P, YAN Z C, et al. Imaging the geometrical structure of the H_2^+ molecular ion by high order above-threshold ionization in an intense laser field [J]. Physical Review A, 2009, 80(6): 3694-3697.
- [3] JIA X Y, LI W D, FAN J, et al. Suppression effect in the nonsequential double ionization of molecules by an intense laser field [J]. Physical review A, 2008, 77(6): 3195-3199.
- [4] 徐士良. FORTRAN 常用算法程序集 [M]. 2 版. 北京: 清华大学出版社, 2012.
- [5] PRESS W H, TEUKOLSKY S A, VETTERLING W T, et al. Numerical Recipes: The Art of Scientific Computing [M]. 3rd ed. Cambridge: Cambridge University Press, 2007.
- [6] DU TOIT C F. The numerical computation of Bessel functions of the first and second kind for integer orders and complex arguments [J]. IEEE Transactions on Antennas and Propagation, 1990, 38(9): 1341-1349.
- [7] 魏彦玉, 宫玉彬, 王文祥. 任意阶复宗量贝塞尔函数的数值计算 [J]. 电子科技大学学报, 1998, 27(2): 171-176.
- [8] 张爽, 郭欣, 宋立军. 利用贝塞尔函数的级数形式进行数值计算的误差分析 [J]. 长春大学学报, 2004, 14(2): 57-59.
- [9] 张善杰, 唐汉. 任意实数阶复宗量第一类和第二类 Bessel 函数的精确计算 [J]. 电子学报, 1996, 24(3): 77-81.

附 录

计算整数阶复宗量第一类贝塞尔函数的Fortran源代码

```
function xbessj(nw,xw) !nw: 贝塞尔函数的阶数, xw: 复宗量
implicit double precision (a-h,o-w)
implicit double complex(x-z)
parameter (iacc=40,bigno=1.d10,bigni=1.d-10)
n=abs(nw)
aw=cdabs(xw)
if(aw.lt.1.e-11)then
xbessj=0.
else
xtox=2./xw
m=2*((n+int(sqrt(float(iacc*n))))/2)+100+2*int(aw)
xbessj=0.
jsum=0
xsum=0.
xbjp=0.
xbj=1.
do 12 j=m,1,-1
xbjm=j*xtox*xbj-xbjp
xbjp=xbj
xbj=xbjm
if(cdabs(xbj).gt.bigno) then
xbj=xbj*bigni
xbjp=xbjp*bigni
xbessj=xbessj*bigni
xsum=xsum*bigni
endif
if((jsum.ne.0).and. (cdabs(dimag(xw)).le.1.0d0)) xsum=xsum+xbj
if((jsum.ne.0) .and. (cdabs(dimag(xw)).gt.1.0d0))then
k=mod((j-1)/2,2)
xsum=xsum+xbj*(-1.0d0)**k
endif
endif
```

(下转第 92 页)

- [24] COSSINS B P, FOUCHER S, EDGE C M, et al. Assessment of nonequilibrium free energy methods [J]. *J Phys Chem B*, 2009, 113: 5508-5519.
- [25] GOETTE M, GRUBMULLER H. Accuracy and convergence of free energy differences calculated from nonequilibrium switching processes [J]. *J Comput Chem*, 2009, 30: 447-456.
- [26] JARZYNSKI C. Equilibrium free-energy differences from nonequilibrium measurements: A master-equation approach [J]. *Phys Rev E*, 1997, 56: 5018-5035.
- [27] HENDRIX D A, JARZYNSKI C. A fast growth method of computing free energy differences [J]. *J Chem Phys*, 2001, 114: 5974-5981.
- [28] HUMMER G. Fast-growth thermodynamic integration: Error and efficiency analysis [J]. *J Chem Phys*, 2001, 114: 7330-7337.
- [29] YTREBERG F M, ZUCKERMAN D M. Single-ensemble nonequilibrium path-sampling estimates of free energy differences [J]. *J Chem Phys*, 2004, 120: 10876-10879.
- [30] LECHNER W, OBERHOFER H, DELLAGO C, et al. Equilibrium free energies from fast-switching trajectories with large time steps [J]. *J Chem Phys*, 2006, 124: 044113.
- [31] BAYLY C I, CIEPLAK P, CORNELL W, et al. A well-behaved electrostatic potential based method using charge restraints for deriving atomic charges: The RESP model [J]. *J Phys Chem*, 1993, 97(40): 10269-10280.
- [32] FRISCH M J, TRUCKS G W, SCHLEGEL H B, et al. Gaussian 09, Revision B.01. [Z], Wallingford: Gaussian Inc, 2010.
- [33] STEINBRECHER T, MOBLEY D L, CASE D A. Nonlinear scaling schemes for lennard-jones interactions in free energy calculations [J]. *J Chem Phys*, 2007, 127: 214108.
- [34] DARDEN T, YORK D, PEDERSEN L. Particle mesh ewald: An nlog(N) method for ewald sums in large systems [J]. *J Chem Phys*, 1993, 98: 10089-10092.
- [35] WENNMOHS F, SCHINDLER M. Development of a multipoint model for sulfur in proteins: A new parametrization scheme to reproduce high-level ab initio interaction energies [J]. *J Comput Chem*, 2005, 26(3): 283-293.
- [36] OLIVET A, VEGA L F. Optimized molecular force field for sulfur hexafluoride simulations [J]. *J Chem Phys*, 2007, 126(14): 144502.
- [37] ZHANG X J, GONG Z, LI J, et al. Intermolecular sulfur...oxygen interactions: Theoretical and statistical investigations [J]. *J Chem Inf Model*, 2015, 55: 2138-2153.
- [38] WANG M T, LI P F, JIA X Y, et al. An efficient strategy for the calculations of solvation free energies in water and chloroform at quantum mechanical/molecular mechanical level [J]. *J Chem Inf Model*, 2017, 57: 2476-2489.

(责任编辑: 张 晶)

(上接第 82 页)

```

jsum=1-jsum
if(j.eq.n) xbessj=xbjp
12 continue
if(n.eq.0) xbessj=xbj
xsum=2.0d0*xsum-xbj
if (cdabs(dimag(xw)).gt.1.0d0) then
xcos=(cdexp(dcmplx(0.0d0,1.0d0)*xw)+
*cexp(dcmplx(0.0d0,-1.0d0)*xw))/2.0d0
xsum=xsum/xcos
endif

```

(责任编辑: 林 磊)