使用非均匀网格泰勒展开实现快速准确的 Boys 函数求值

Longxiang Yan，[1] Wei Wu，[1] 和 Peifeng Su[1]福建省理论与计算化学重点实验室、固体表面物理化学国家重点实验室，以及厦门大学化学与化工学院，福建厦门 *361005*，中国

（*电子邮箱：supi@xmu.edu.cn）

在中央处理器（CPU）和图形处理器（GPU）上进行高斯积分计算时，高效的 Boys 函数求值十分重要。网格泰勒展开提供了一种简单的求值方法，但在较大的自变量范围内，等间距参考表需要大量存储空间。为解决这一限制，我们采用间距随自变量增大而增加的非均匀网格。对于大自变量，我们引入了一种归一化辅助函数，其缩放和导数性质使其能够采用与主域Boys函数相同的泰勒求值方法。主域和大自变量参考表通过联合构建得到，并在精度和存储约束下选择多项式次数与网格参数。参考值中加入低位修正，以提高泰勒求值的精度。所得方法已在 Fast and Accurate BOys（FABO）库中实现，并在 QUICK 和 NWChem–Simint 中进行了评估。在所测试的实现中，对于报告的三种误差度量，FABO 的最大误差和平均误差均为最小，其中采样得到的最大误差约为 1.067 个末位单位（ULP）。在应用基准测试中，与原有的 Boys 函数求值器相比，FABO 在两个应用中均缩短了 Fock 构建时间，并且与其他测试实现相比具有竞争力。

arXiv:2611.32943v1 [physics.chem-ph] 24 Nov 2026

许可证。 除非另有说明，本文根据知识共享署名—非商业性使用 4.0 国际许可协议（CC BY-NC 4.0）授权。

I. 引言

高斯型轨道（GTO）上的分子积分求值是电子结构计算的基础组成部分。特别是，在 Hartree–Fock 和密度泛函理论计算中，会反复求值电子排斥积分（ERI）。因此，无论是在中央处理器（CPU）还是图形处理器（GPU）上进行计算，高效的积分求值都十分重要。[1,2]

Boys 函数，

$$F_n(x) = \int_0^1 t^{2n} e^{-xt^2}\,\mathrm{d}t, \tag{1}$$

是许多高斯积分求值算法的核心。它们提供了通过递推关系生成更高角动量积分所需的基本量。其自变量和所需输出阶数取决于目标分子几何结构、基组以及积分求值算法。因此，Boys 函数求值器必须能够高效处理一系列自变量和输出阶数，同时提供足够的精度，以供后续积分计算使用。

人们已经针对 Boys 函数求值开发了多种方法，包括级数展开、多项式或有理逼近、根据制表值进行插值，以及不同阶数之间的递推关系。算法的选择需要在近似精度、浮点运算量和存储需求之间取得平衡。近期研究还考察了这些选择对向量化和图形处理器执行的影响。例如，Vikhamar-Sandberg 和 Repisky 将有理极小极大近似与阶数递推结合使用，而 Laikov 则针对向量化计算开发了全局近似。[3,4] 其他方法使用制表值和局部多项式求值，并且通常针对其他参数范围结合使用不同的公式。[5,6]

在这些方法中，网格化泰勒求值长期以来一直用于分子积分计算[7]，并且至今仍具吸引力。它具有简单的计算结构，主要可通过融合乘加运算（FMAs）进行求值，而融合乘加运算在现代处理器上效率很高。在插值区域内，对于不同的自变量，可以使用相同的多项式求值过程，无需依赖自变量的分支；此外，还可以利用余项轻松分析泰勒截断误差。

尽管具有这些优势，但要在较宽的自变量范围内扩展等间距参考表，需要大量存储空间。因此，现有实现通常将网格化泰勒求值限制在自变量范围的一部分，并对较大的自变量使用递推关系或渐近公式。[5,8] 这些不同区域需要采用不同的求值过程，而随着舍入误差的累积，递推生成值的精度可能会降低。当来自不同区域的自变量出现在同一个图形处理器 warp 或中央处理器单指令多数据（单指令多数据）向量，不同的处理过程还可能增加处理它们所需的工作量，从而导致性能下降。

为解决这些局限，我们开发了一种用于 Boys 函数求值的非均匀网格化泰勒方法，并将其实现于一个名为 Fast and Accurate BOys (FABO) 的全新开源库中。[9] 在主要区域，我们采用间距随自变量增大而增加的非均匀网格，从而减少覆盖更宽自变量范围所需的参考表存储量。对于大自变量，我们利用一个辅助函数的缩放恒等式构建归一化参考表

。主要区域和大自变量区域的表格采用相同的泰勒求值过程，因此两个区域的输入都可以遵循相同的计算结构。针对请求的输出阶数，对参考值进行低位修正，以提高求值的数值精度。在规定的精度和存储需求下，我们联合选择表格参数和多项式次数，优先减少多项式计算量，再考虑减少存储量。总体而言，这些设计提供了一种精确的 Boys 函数求值器，具有紧凑的参考表，并能在宽广的自变量范围内采用统一的求值过程。

本文结构如下。第二节介绍非均匀网格化泰勒方法及其数值目标值。第三节介绍归一化的大自变量表示，以及参考表的联合构造。第四节讨论该方法在积分核中的实现。第五节介绍求值协议，第六节给出准确性和性能结果。最后，第七节总结结论。

II. 非均匀网格化泰勒求值

A. 泰勒表示与数值目标值

高斯积分计算通常需要针对同一自变量$F_0(x)$，$F_1(x)$ ……，$F_N(x)$ 这一系列 Boys 函数值。我们使用围绕一组参考点展开的泰勒级数来求这些值。对于每个自变量，从网格中选取一个参考点r_i ，然后将所需的每个 Boys 函数近似为关于相对于r_i的位移的多项式。网格的选择会影响达到给定精度所需的多项式次数以及参考数据量。

这些多项式的系数可以直接由 Boys 函数值获得。对公式 (1) 求导可得

$$\frac{\mathrm{d}^j F_n(x)}{\mathrm{d}x^j} = (-1)^j F_{n+j}(x), \qquad j = 0, 1, \ldots, \tag{2}$$

因此，F_n 在r_i 处的导数由同一点处的高阶 Boys 函数给出。为了进行只需计算该序列部分项的计算，我们考虑所请求阶数$a \le n \le b$的一般区间。定义$h = r_i - x$，可写为

$$F_n(x) = T_{n,p_n}(x; r_i) + R_{n,p_n}(x; r_i), \tag{3}$$

其中，R_{n,p_n} 为截断余项。阶数为n 、多项式次数为p_n 的 Taylor 多项式为

$$T_{n,p_n}(x; r_i) = \sum_{j=0}^{p_n} \frac{F_{n+j}(r_i)}{j!} h^j. \tag{4}$$

选择$h = r_i - x$ 将公式 (2) 中的交替符号纳入其中。

对于请求的输出范围 $[a, b]$,，我们将用于求值相应 Boys 函数的完整数值数据和参数集合称为一个数值模型。一个模型由主域和归一化大参数参考表、它们的网格参数及切换边界、分配给每个请求阶数的多项式次数 p_n ，以及相关的低位修正组成。每个模型都针对其请求的输出范围独立构建，并由求值器用于计算 $F_a(x)$……$F_b(x)$，覆盖受支持的自变量范围。

我们将最后一位的一个单位（ULP）误差作为构造参考网格和选择多项式次数的数值目标。ULP 衡量给定数量级下相邻可表示浮点数之间的间距，因此表示相对于可用浮点分辨率的误差。[10] 令 $\widehat{F}_n(x)$ 表示计算得到的 binary64 结果，并令 RN_{64} 表示使用就近舍入，偶数舍入将数值舍入为 binary64。[11] 该目标表示为

$$\frac{\left|\widehat{F}_n(x) - F_n(x)\right|}{\mathrm{ulp}(\mathrm{RN}_{64}(F_n(x)))} \le 1, \qquad a \le n \le b. \tag{5}$$

该目标的限制性低于正确舍入；正确舍入要求 $\widehat{F}_n(x) = \mathrm{RN}_{64}(F_n(x))$。

式（5）中的误差不仅包括泰勒余项，还反映了参考数据的表示误差以及浮点求值过程中的舍入误差。在第 III 节引入的大自变量区域中，该误差还包括用缩放辅助函数近似 F_n 所产生的误差。这些误差贡献会在计算结果中叠加，因此，仅有较小的截断误差并不能确保最终误差保持在一个 ULP 以内。

针对每个候选网格与阶数组合测试每个 binary64 输入，会使参数搜索变得不切实际。因此，我们在选定的一组验证输入上评估完整的浮点结果。这些输入对参考单元进行采样，其中包括预期数学近似误差较大的区域。只有当候选模型在规定的验证输入上满足式（5）时，才会被接受。这些输入的构造以及数值验证过程详见补充信息（第 S2.3 节）。

由于资格认定使用的是有限组输入，通过这些测试并不能保证式（5）对每个 binary64 输入都成立。特别是，浮点舍入误差未必会在数学近似误差最大的地方达到最大值。第 VIA 节的精度结果中可以观察到这一现象。

B. 非均匀参考间距的数学基组

网格化泰勒求值具有简单的计算结构：一旦获得参考值，就可以通过乘法和融合乘加求值局部多项式，而无需针对输入自变量求值指数函数、误差函数或倒数平方根。对于固定的多项式次数，选择不同的参考点会改变系数，但多项式求值的运算序列保持不变。泰勒余项还直接揭示了参考间距与近似精度之间的联系

。这些特性使网格化泰勒求值颇具吸引力，前提是能够以较低成本存储和访问参考数据。

一种自然的实现采用等间距的参考点，$r_i = i\Delta$。在参考阶数集合固定的情况下，覆盖$0 \le x \le X$大约需要X/Δ个参考行。因此，为在原点附近满足精度目标而采用的足够精细的间距会在整张表中一直保留，而扩展所支持的范围会使存储线性增加。限制这种存储的一种方法是仅在自变量范围的一部分上使用网格化泰勒求值，而在其他区域采用其他方法。例如，Tsuji 等人对较小的自变量使用均匀间距的参考值，并对其余区域结合使用递推和大自变量方法。[5] 在此，我们转而考察参考间距是否可以随x增大，从而使网格化泰勒求值能够覆盖更宽的范围，而存储不必成比例增加。

Boys 函数为这样的网格提供了数学基组。对于取值为正规数的结果，固定的 ULP 目标对应于一种相对误差尺度，该尺度随有效数的变化至多相差两倍。[10] 因此，我们考察相对于$F_n(x)$的泰勒余项。所需的导数界直接由积分定义得出。对于$y > 0$以及非负整数k，对$t^{2k+1}e^{-yt^2}$的导数进行积分可得

$$\begin{aligned} e^{-y} &= \int_0^1 \frac{\mathrm{d}}{\mathrm{d}t}\left(t^{2k+1}e^{-yt^2}\right)\mathrm{d}t \\ &= (2k+1)F_k(y) - 2yF_{k+1}(y). \end{aligned} \tag{6}$$

由于 Boys 函数和指数函数均为正，重新排列可得

$$\frac{F_{k+1}(y)}{F_k(y)} \le \frac{k+\frac{1}{2}}{y}. \tag{7}$$

在阶数 n、$n+1$、……、$n+m-1$处依次应用该不等式，可将高阶导数相对于函数本身界定为：

$$\frac{F_{n+m}(y)}{F_n(y)} \le \frac{1}{y^m}\prod_{j=0}^{m-1}\left(n+j+\frac{1}{2}\right), \qquad m \ge 1. \tag{8}$$

因此，每增加一次求导，就会引入一个有界因子，该因子等于某个常数除以自变量。

要将这一观察应用于泰勒近似，令$m = p_n + 1$、$d = |x - r_i|$，以及$y = \min(x,\ r_i) > 0$。泰勒定理将余项导数定位在x与r_i之间的某一点。由于F_{n+m}随其自变量增大而减小，

$$\left|R_{n,p_n}(x;r_i)\right| \le \frac{d^m}{m!}F_{n+m}(y). \tag{9}$$

式（8）中的界将该导数与 $F_n(y)$ 进行比较，而所需的误差尺度是 $F_n(x)$。为了将该界表示为相对于 $F_n(x)$ 的形式，我们还需要对比值 $F_n(y)/F_n(x)$ 进行界定。利用 $F_n' = -F_{n+1}$ 并对式（7）积分可得

$$\ln\frac{F_n(y)}{F_n(x)} = \int_y^x \frac{F_{n+1}(v)}{F_n(v)}\,\mathrm{d}v \le \left(n+\frac{1}{2}\right)\ln\frac{x}{y}. \tag{10}$$

综合这些结果并记 $\rho = d/y$，其中 $x/y \le 1+\rho$，得到

$$\frac{\left|R_{n,p_n}(x;r_i)\right|}{F_n(x)} \le (1+\rho)^{n+\frac{1}{2}}\frac{\rho^m}{m!}\prod_{j=0}^{m-1}\left(n+j+\frac{1}{2}\right). \tag{11}$$

对于固定的阶数和次数，式（11）的右侧取决于相对位移 ρ。例如，同时将x和r_i加倍，会使二者的间距加倍，但该界保持不变。因此，在远离原点处，参考区间的宽度可以近似按x的比例增大，同时保持相同的截断误差界。在接近零处，相对位移的描述并不适用，但可以改用绝对位移d来界定截断误差。由于

$$F_{n+m}(y) \le F_{n+m}(0) = \frac{1}{2n+2m+1}, \tag{12}$$

式（9）给出

$$|R_{n,p_n}(x;r_i)| \le \frac{d^m}{m!(2n+2m+1)}. \tag{13}$$

由于 $F_n(x)$ 在原点邻域内始终与零保持一定距离，因此，足够小的绝对网格间距也能控制该处的相对截断误差。结合远离原点处的相对位移界，这启发我们采用一种网格：在零点附近使用精细且有限的间距，而在自变量较大时逐步增大区间。

这种网格改变了存储量随支持范围的增长方式。一旦间距与自变量成正比，上限每扩大一倍，所需增加的参考点数量就大致相同。因此，在该区域内，参考点数量随上限呈对数增长，而不是像采用固定间距的均匀网格那样呈线性增长。这为在广泛定义域上扩展网格化泰勒求值提供了一种途径，同时无需在整个表中始终保留原点附近的分辨率。

C. 基于浮点偏移的扩展网格

增大的间距减少了参考点的数量，但高效的求值器还必须针对每个输入，以较低的成本定位适当的参考行。搜索存储的参考坐标会给每次 Boys 求值增加额外工作。因此，我们希望构造一种网格，将上述方法启发的间距模式与可直接从输入表示中获得的参考索引结合起来。

二进制浮点数提供了这样一种结构：在每个二的幂区间内，其间距是均匀的，而在下一个区间中则加倍。[11] 一个正的正规数值具有 $2^{\ell}(1+f)$ 的形式，其中 ℓ 是整数指数，$0 \le f < 1$。保留 b_g 个小数位后，每个区间中有 $Q = 2^{b_g}$ 个晶格点 $[2^{\ell}, 2^{\ell+1})$：

$$z_{\ell,j} = 2^{\ell}\left(1+\frac{j}{Q}\right), \qquad j = 0,\ldots,Q-1. \tag{14}$$

其间距为

$$\Delta_\ell = 2^{\ell - b_g}. \tag{15}$$

因此，指数和保留小数位同时描述了扩展网格以及点在其中的位置。

然而，直接将该晶格应用于 x 会在 x 趋近于零时保留越来越细的间距。通过引入一个正偏移量 s，并将晶格应用于 $x+s$，我们避免了这种不必要的细化。选择 s 为一个晶格点，并将保留的点 $z_i \geq s$ 平移回原坐标：

$$r_i^\star = z_i - s. \tag{16}$$

点 $z_i = s$ 随后在原点处提供一个参考点。在零附近，$x+s$ 与 s 位于同一个二的幂区间内，从而产生有限的均匀间距。在更远处，当 $x+s$ 越过下一个二的幂时，间距便会加倍。例如，$s=1$ 和 $b_g=2$ 在 $1/4$ 上的间距为 $[0,1]$，在 $1/2$ 上为 $[1,3]$,，在 $[3,7]$ 上为 1。偏移量决定原点附近的 Scale，而保留小数位的数量则控制网格密度。

相同的表示方式也使网格的索引开销很低。在运行时，移位后的坐标由以下方式构成：

$$z = \mathrm{RN}_{64}(x+s). \tag{17}$$

对其二进制编码进行整数舍入和移位，即可选出邻近的晶格点，并确定相应的参考行。根据这些位恢复选定的晶格值z_i，参考坐标为

$$r_i = \mathrm{RN}_{64}(z_i - s). \tag{18}$$

因此，参考点的选择既不需要搜索存储的坐标，也不需要对数运算或浮点缩放乘法：索引坐标通过一次浮点加法即可获得。

对于平移后坐标每个完整二的幂范围内的Q 个区间，该构造实现了第 IIB 节所述的近似对数式参考点数量增长，同时无需搜索参考坐标即可选定参考行。

数值构造使用实际的浮点映射，而不是理想化的单元边界。在重构后的坐标r_i处生成参考系数，并根据公式 (5) 中的最终输出判据确定网格和泰勒次数。详细的索引构造与限定过程见补充信息（第 S1.2 节和第 S2.3 节）。

图 1 比较了使用为 FABO 求值而构造的非均匀网格计算F_0 时的泰勒截断误差（该求值仅请求F_0）与均匀网格的误差；后者具有相同数量的参考点和相同的多项式次数。在每个内部边界处，两个相邻泰勒多项式中较大的误差展示了.....项。在均匀间距下，大参数区域的大部分范围内，相对余项远小于一个 ULP，但在原点附近则超过一个 ULP。因此，在相同的存储预算下，参考点被用于截断误差已经远低于目标值的区域，而在需要更细网格的区域，间距仍然过大。非均匀网格将参考点重新分配至小参数区域，并允许间距随x增大。因此，在所有测试的单元边界处，余项均低于一个 ULP，最大相对余项也从 3.12×10^{-15}降至 6.48×10^{-17}。此比较针对数学上的泰勒余项，不包括参考系数的舍入和运行时算术运算。网格参数和计算细节见补充信息（S3.1 节）。

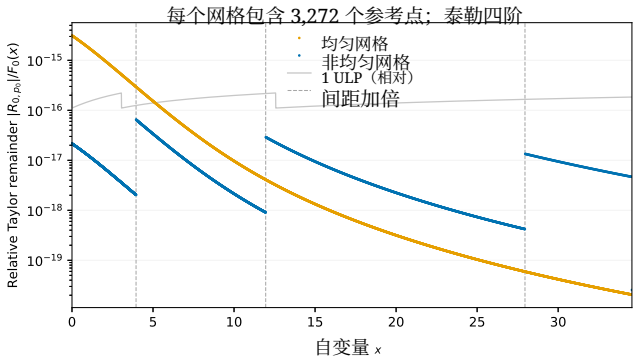

图 1. 使用具有相同参考点数量和多项式次数的均匀网格与非均匀网格时，F_0 在参考单元边界处的相对泰勒截断误差。

D. 参考表与多项式求值

不同输出阶数的泰勒展开需要重叠的参考值集合。例如，$F_{n+1}(r_i)$既是$F_{n+1}(x)$展开中的常数项，也是$F_n(x)$展开中h 的系数。因此，我们在每个参考点存储一个 Boys 值序列，而不是为每个输出分别存储一组阶乘加权系数。在多项式求值过程中引入阶乘因子，使同一个存储值能够用于不同展开中的不同位置。

为避免求值不必要的项，根据数值接受准则，针对每个输出阶数分别选择多项式次数p_n。对于给定阶数，该次数在整个参考网格上保持不变，因此选择不同的参考点只会改变系数，而不会改变多项式求值序列。对于请求区间 $[a,\ b]$,，所需的最高参考阶数为

$$H = \max_{a\leq n\leq b}(n+p_n). \tag{19}$$

因此，每个参考行都存储

$$G_{i,k} = \mathrm{RN}_{64}(F_k(r_i)), \qquad k = a,\ldots,H. \tag{20}$$

上述阶数b 提供的是泰勒系数，而不是额外的输出。对于N_r 个参考点，仅参考值就占用 $8N_r(H-a+1)$字节。它们可在不同输出阶数之间重复使用，

从而避免重复存储重叠系数序列。

式 (4) 中的阶乘因子可以直接应用，无需形成 h 的幂，也无需存储单独的阶乘缩放表。在精确算术中，霍纳法则的阶乘形式对于$p_n \geq 1$给出

$$T_{n,p_n}(x;r_i) = F_n(r_i) + h\,v_{n,1} \tag{21}$$

其中，中间值定义为

$$v_{n,p_n} = F_{n+p_n}(r_i), \tag{22a}$$

$$v_{n,j} = F_{n+j}(r_i) + \frac{h}{j+1} v_{n,j+1}, \qquad j = p_n - 1, \ldots, 1. \tag{22b}$$

例如，三次多项式可写为

$$T_{n,3}(x;r_i) = F_n(r_i) + h\left[F_{n+1}(r_i) + \frac{h}{2}\left(F_{n+2}(r_i) + \frac{h}{3}F_{n+3}(r_i)\right)\right]. \tag{23}$$

连续因子 h、$h/2$ 和 $h/3$ 生成所需的阶乘分母，同时保留对未缩放参考值的使用。

在浮点求值器中，非恒定霍纳链中的参考值由$G_{i,\,k}$替代，并且每个乘加阶段都使用融合乘加进行计算。对应于h/j 的因子通过将计算得到的位移与预先计算的正整数j的 binary64 倒数相乘来形成，而不是在运行时进行除法。由于这些因子取决于位移和泰勒位置，而与输出阶数无关，因此可以由所请求的多项式共享。不过，每个输出都有自己的霍纳累加器，并使用其参考系数进行求值，不会在x处通过阶数递推传播另一个计算得到的输出。

常数系数与非恒定项贡献的最终组合与霍纳链分开保留。尽管$G_{i,\,n}$ 经过正确舍入，但其与真实值$F_n(r_i)$之间的差值会在该组合中不带位移因子地参与计算。即使泰勒截断误差很小，该误差仍可能十分重要。下一小节将介绍一种紧凑的低位部分修正，在最终输出组装中保留有关常数系数的额外信息。

E. 泰勒求值的低位修正

将参考值舍入为 binary64 会丢弃可能影响最终泰勒结果的信息。对于常数系数，被丢弃的部分为

$$\rho_{i,n} = F_n(r_i) - G_{i,n}, \tag{24}$$

其中，$G_{i,\,n}$ 是公式（20）中定义的正确舍入的参考值。$F_{n+j}(r_i)$ 中的表示误差受到因子 $h_j\ /j!$ 的抑制，而常数系数残差不受位移 h 的任何幂次抑制。因此，将输入移得更接近参考点并不能抑制这一贡献。我们保留有关常数系数的附加信息，并将其纳入最终的输出组装。

该校正项会在 FMA 内部与非恒定项泰勒贡献结合，然后再加上常数系数的高位部分。这种顺序避免了先将一个较小的残差加到$G_{i,\,n}$上的情况，因为这样会直接将我们试图保留的信息舍入掉。将残差存储为另一个 binary64 值，需要为每个输出和参考点额外增加八个字节。相反，我们采用一个单字节近似，将额外精度集中在其误差不会被位移衰减的系数上。

为给泰勒多项式预留浮点执行资源，我们还希望解码该字节时无需进行整数到浮点的转换或浮点缩放。一种自然的方法是将存储的位直接放入浮点表示中，并根据$G_{i,\,n}$ 的指数推导其缩放比例。然而，正规二进制浮点数的有效数形式为 1。f，其中包含隐含前导一。[11]残差 $\rho_{i,\,n}$ 可能为正、为负、为零，或者小到任意程度。因此，无法通过在固定缩放比例下填充正规数的小数位来表示它。

我们通过改变高位部分与其校正项之间的划分来处理隐含前导一。对于正的正规参考值，令 $u_{i,\,n}$ =ulp($G_{i,\,n}$)。我们将调整后的高位部分定义为大于$G_{i,\,n}$的下一个可表示的 binary64 值：

$$G_{i,n}^{+} = \mathrm{nextUp}(G_{i,n}) = G_{i,n} + u_{i,n}. \tag{25}$$

相应的校正项定义为调整后的高位部分与精确参考值之差：

$$\Delta_{i,n} = G_{i,n}^{+} - F_n(r_i) = u_{i,n} - \rho_{i,n}. \tag{26}$$

正确舍入给出$|\rho_{i,\,n}| \leq u_{i,\,n}/2$，因此

$$\frac{1}{2}u_{i,n} \leq \Delta_{i,n} \leq \frac{3}{2}u_{i,n}. \tag{27}$$

因此，近零的有符号残差被一个限制在狭窄范围内的正校正项取代。其非零的有效数最高位如今发挥了作用，而不再构成障碍。减去该校正项，可补偿高位部分增加的一个最后一位单位。

该字节存储一个指数位和七个小数位。将其无符号编码写作 $c = 128e_c + f_c$，其中 $e_c \in \{0,1\}$和 $0 \leq f_c < 128$，则表示的校正项为

$$\widehat{\Delta}_{i,n} = u_{i,n}\,2^{e_c-1}\left(1 + \frac{f_c}{128}\right). \tag{28}$$

从 0 到 192 的编码值覆盖公式（27）中的区间。特别地，编码值 128 表示 $u_{i,\,n}$ 因此可以处理原始残差恰好为零的情况，而无需使用值为零的解码后的校正值。在表构造过程中，$\Delta_{i,\,n}$ 会舍入到最近编码值；若恰好处于中间，则向偶数舍入，得到

$$\left|\widehat{\Delta}_{i,n} - \Delta_{i,n}\right| \leq \frac{u_{i,n}}{256}. \tag{29}$$

公式（28）中的运算描述的是所表示的值，而不是运行时解码器：binary64 编码由 $G_{i,\ n}$ 的原始指数，以及通过整数运算和按位运算处理的校正字节组装而成。下一个更大的高位部分同样通过递增正参考值的编码获得。

令 $\widehat{\ }v_{n,}1$ 表示式（21）中的非恒定霍纳累加器，该累加器使用原始参考系数进行求值。我们将 $\mathrm{FMA}_{64}(a,b,c) = \mathrm{RN}_{64}(ab+c)$ 记作由电气与电子工程师协会浮点运算标准（IEEE 754）定义的融合乘加，其中乘积和总和的计算不经过中间舍入。[11] 校正后的输出为

$$\widehat{F}_n(x) = \mathrm{RN}_{64}\left[G^{+}_{i,n} + \mathrm{FMA}_{64}\left(h, \widehat{v}_{n,1}, -\widehat{\Delta}_{i,n}\right)\right]. \tag{30}$$

这种方法通过最终的融合累加保留了低位部分的信息，同时每个输出和参考点仅需额外一个字节，且无需通过浮点运算来构造解码后的校正值。位级解码器以及确保移位后的高位部分有限、解码后的校正值为正规数的构造检查，详见补充信息（第 S1.3 节）。

III. 使用缩放渐近函数进行大参数求值

仅使用参考表将网格化泰勒求值扩展至整个 binary64 输入范围并不切实际。因此，传统实现对于大自变量会采用渐近近似和递推关系，但在精度要求严格时，这些方法会带来数值和执行结构方面的挑战。我们通过构造一个缩放渐近函数来同时解决这两个问题；该函数可采用与原始 Boys 函数相同的泰勒求值流程进行求值。

A. 传统大参数求值的局限性

尽管前一节引入的扩展网格减少了参考存储的增长，但在存储多个 Boys 阶数及其泰勒系数时，要覆盖整个 binary64 输入范围仍需要一个相当大的表。这样的表可以容纳在设备内存中，但其占用空间很容易超过我们希望为积分内核中的高效求值预留的缓存容量。因此，传统的 Boys 实现通常会对足够大的自变量使用渐近近似和递推关系。[3,5,6]

渐近近似可以直接由积分定义推导得出。对于 $n = 0$，将 $y = \sqrt{x}t$ 代入式（1）可得

$$\begin{aligned} F_0(x) &= \int_0^1 e^{-xt^2}\,\mathrm{d}t \\ &= \frac{1}{\sqrt{x}}\int_0^{\sqrt{x}} e^{-y^2}\,\mathrm{d}y \\ &= \frac{\sqrt{\pi}}{2\sqrt{x}}\,\mathrm{erf}(\sqrt{x}). \end{aligned} \tag{31}$$

由于随着 x 增大，$\mathrm{erf}(\sqrt{x})$ 趋近于 1，因此首项渐近近似为

$$F_0(x) \approx \frac{\sqrt{\pi}}{2\sqrt{x}}. \tag{32}$$

更高阶数可以通过向上递推获得。重新整理式 (6) 可得

$$F_{n+1}(x) = \frac{(2n+1)F_n(x) - e^{-x}}{2x}, \qquad x > 0. \tag{33}$$

尽管这种递推提供了一种低成本生成连续阶数的方法，但随着递推进行，其数值精度可能会下降。当所需精度显著高于传统应用通常容许的误差时，这一点尤为重要。

为评估向上递推引入的误差，我们从 $F_0(x)$ 和 e^{-x} 的正确舍入的 binary64 值开始。对于每个自变量，我们计算 $t = \mathrm{RN}_{64}(x+x)$、$\rho = \mathrm{RN}_{64}(1/t)$ 和 $q = \mathrm{RN}_{64}(e^{-x}\rho)$。随后将递推求值为

$$\widehat{F}_n = \mathrm{FMA}_{64}(a_n, \widehat{F}_{n-1}, -q), \tag{34}$$

其中，$a_n = \mathrm{RN}_{64}((2n-1)\rho)$ 以及 $\widehat{\ }F_0 = \mathrm{RN}_{64}(F_0(x))$。舍入指数和倒数会在每一步重复使用。

对于每个请求的最高阶数 $N = 1, \ldots\ldots, 12$，以及 1、2、4、8 或 16 个 ULP 的容差，我们扫描自变量 $x_i = \mathrm{RN}_{64}(i/10^6)$，其中 $i = 1, \ldots\ldots, 109$，并确定第一个包含一百万个连续样本的窗口，在该窗口中每个计算得到的$\widehat{\ }$ $\widehat{\ }$值 $F1, \ldots\ldots, F_N$ 均满足该容差。报告的边界是该窗口中的第一个自变量。误差相对于高精度数学参考值进行测量，并使用正确舍入的参考值处的 binary64 间距来定义 ULP。正如第 II A 节所述，在采样输入上满足误差容差，并不能保证在其他输入上也满足该容差。

图 2 显示，随着请求的最高阶数增加，第一个通过窗口通常会向更大的自变量移动。收紧误差容差也会产生类似影响；对于若干阶数与容差的组合，即使在已经很大的扫描范围内，也未找到完整的通过窗口。特别是，在一 ULP 容差下，所测试的阶数范围均未产生通过窗口，即便起始值经过正确舍入也是如此。这些结果说明，使用向上递推保持高精度十分困难，尤其是在需要计算多个阶数时。

除了数值限制外，传统的混合求值器通常会采用不同的计算过程

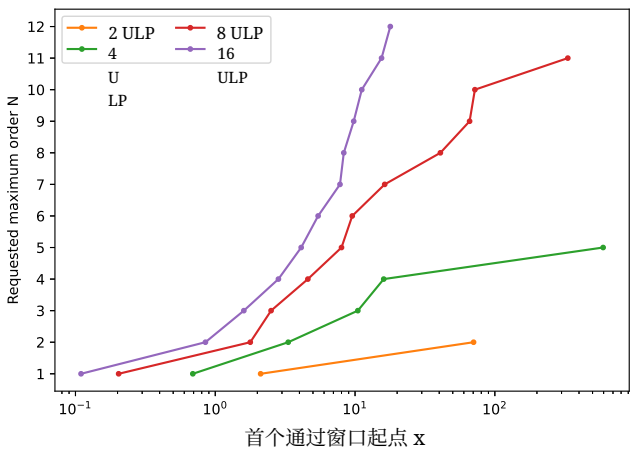

图 2. 不同指定最高阶数和误差容差下，binary64 向上递推的首个通过窗口起点。对于 1-ULP 容差，未找到这样的窗口。

分别处理小参数区域和大参数区域。当来自两个区域的自变量出现在同一个 GPU warp 中时，相应的分支可能会在不同的线程掩码下执行，从而增加串行工作量。CPU SIMD向量的不同通道也可能遇到同样的问题。这两个区域还需要不同的系数访问和求值过程，因此难以共享主要的计算工作。

这些观察促使我们采用一种不同于传统渐近递推的方法：一种紧凑的大参数表示，同时保留原始 Boys 函数的局部泰勒求值结构。

B. 大参数缩放与归一化

式 (32) 中的渐近近似可理解为将 Boys 积分的积分上限从 1 延伸至无穷大。我们将相应的函数族定义为

$$A_n(x) = \int_0^\infty t^{2n} e^{-xt^2}\,\mathrm{d}t, \qquad x > 0. \tag{35}$$

对于 $n = 0$，高斯积分给出

$$A_0(x) = \frac{\sqrt{\pi}}{2\sqrt{x}}, \tag{36}$$

这恰好是$F_0(x)$ 的主导渐近近似。

更高阶项可通过分部积分得到。由于

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(t^{2n+1}e^{-xt^2}\right) = (2n+1)t^{2n}e^{-xt^2} - 2xt^{2n+2}e^{-xt^2}, \tag{37}$$

对 $[0,\infty$ 积分并利用消失的边界项可得

$$A_{n+1}(x) = \frac{2n+1}{2x}A_n(x). \tag{38}$$

因此，无需计算误差函数或指数函数，即可根据式（36）生成整个渐近族。通过比较式（1）和式（35），模型与原始 Boys 函数之间的差异恰好为

$$A_n(x) - F_n(x) = \int_1^\infty t^{2n} e^{-xt^2}\,\mathrm{d}t. \tag{39}$$

因此，该模型会高估 Boys 函数，偏差大小取决于将积分上限延伸至无穷大所引入的额外尾部。这种近似是否足够准确，取决于自变量和所需阶数，并将在完整求值器的数值资格认定中予以考量。

该模型的一个重要性质是，自变量变为原来的四倍时，其缩放规律为。由式 (36) 可得，

$$A_0(4x) = \frac{1}{2}A_0(x). \tag{40}$$

依次应用式 (38) 中的递推关系可得

$$A_n(4x) = 2^{-(2n+1)}A_n(x). \tag{41}$$

因此，对于每个阶数，缩放因子都是 2 的整数次幂。例如，自变量增大为原来的四倍时，零阶值减半，一阶值则变为原来的八分之一。

我们利用这一恒等式将较大的自变量映射到一个固定区间。任意 $x \geq 1$ 均可唯一表示为

$$x = 4^{k_x}u \tag{42}$$

其中 $k_x \in \mathbb{Z}_{\geq 0}$ 且 $1 \leq u < 4$。反复应用式 (41) 即可得到

$$A_n(x) = 2^{-(2n+1)k_x}A_n(u). \tag{43}$$

这种归一化将自变量的大小与函数求值分离开来。无论原始输入变得多大，渐近模型的参考网格只需覆盖 $[1,4)$。此外，对于正常的 binary64 输入，k_x 和 u 都可以直接根据其指数和有效数获得：二进制指数每增加 2，就对应于k_x增加 1。无需进行昂贵的对数运算或浮点除法。

因此，大参数模型能够在任意宽的输入范围内提供紧凑的表示。更重要的是，正如下面将展示的，其导数结构与原始 Boys 函数完全相同，因此两个域可以使用相同的泰勒求值过程。

C. 共享的泰勒求值路径

归一化大参数表示可以使用与原始 Boys 函数相同的泰勒表示进行求值。为此，在积分号下对式（35）求导可得：

$$\frac{\mathrm{d}A_n(x)}{\mathrm{d}x} = -\int_0^\infty t^{2n+2}e^{-xt^2}\,\mathrm{d}t = -A_{n+1}(x). \tag{44}$$

因此，反复求导可得

$$\frac{\mathrm{d}^j A_n(x)}{\mathrm{d}x^j} = (-1)^j A_{n+j}(x), \qquad j = 0, 1, \ldots. \tag{45}$$

这与式（2）中原始 Boys 函数的导数关系相同。因此，这两个函数族具有相同的泰勒展开结构。

令 r_i 表示从主域网格中选取的一个参考点，令 u_i 表示归一化大参数网格中的一个参考点。对于主域中的输入x，定义位移 $h = r_i - x$。对于大参数输入，首先利用公式（42）获得其归一化坐标 u，并定义$h = u_i - u$。相应的泰勒多项式为

$$T^{(F)}_{n,p_n}(x; r_i) = \sum_{j=0}^{p_n} \frac{F_{n+j}(r_i)}{j!} h^j, \tag{46a}$$

$$T^{(A)}_{n,p_n}(u; u_i) = \sum_{j=0}^{p_n} \frac{A_{n+j}(u_i)}{j!} h^j. \tag{46b}$$

这两个多项式的参考系数和位移量不同，但具有相同的代数形式。特别是，二者都可以使用第 II D 节和第 II E 节中介绍的阶乘霍纳格式以及低位部分修正进行求值。

从数学上讲，这两个域无需使用相同的多项式次数。我们有意施加这一约束：对于每个输出阶数n，两个域均使用相同的次数p_n。不同的输出阶数仍可使用不同的次数。这样可确保对于每个输出阶数，两次求值都需要相同数量的多项式阶段以及相同顺序的系数访问，从而使它们能够共享多项式求值过程，而无需根据域的不同改变其中的算术运算。

两个域所需的参考阶数也由公共次数向量决定。对于请求的输出区间 $[a,b]$, ，所需的最高参考阶数为

$$H = \max_{a \le n \le b} (n + p_n). \tag{47}$$

因此，每个主域参考行都存储舍入值序列 $F_k(r_i)$，其中 $a \le k \le H$；而每个归一化大参数行则存储相应的序列 $A_k(u_i)$。两者采用相同的逻辑行布局，参考系数和输出低位修正项占据对应位置。数值因域而异，但它们在求值过程中的含义和用法完全相同。

最终的求值过程将依赖于定义域的预处理与通用的多项式计算分开。主域输入直接根据x选取其参考行和位移。大参数输入首先进行归一化，随后根据u获取其参考行和位移；同时保留相应的整数缩放指数k_x。参考行和位移准备就绪后，两个域都遵循相同的系数加载、校正项解码和霍纳求值流程。主域

算法 1 求值 $F_0(x), \ldots, F_N(x)$。

输入： 有限 binary64 自变量 $x \ge 0$（包括 -0），最大输出阶数N **输出：** $\widehat{F}_0(x), \ldots, \widehat{F}_N(x)$

如果 $x \le x_{\text{switching}}$ **则** ▷ 主域求值
  $z \leftarrow \mathrm{RN}_{64}(x + s)$
  $i \leftarrow$ 主网格索引(z)
  $r_i \leftarrow$主参考点(z,i)
  $h \leftarrow \mathrm{RN}_{64}(r_i - x)$
否则 ▷归一化大参数求值
  $(k_x, u) \leftarrow$ 归一化(x)
  $i \leftarrow$ 尾部网格索引(u)
  $u_i \leftarrow$尾部参考点(i)
  $h \leftarrow \mathrm{RN}_{64}(u_i - u)$
结束 if
对于$n = 0, \ldots, N$ **do**
  $v \leftarrow C_{i,n+p_n}$
  对于$j = p_n - 1, \ldots, 1$ **执行**
    $\alpha_j \leftarrow \mathrm{RN}_{64}(h \times \mathrm{RN}_{64}(1/(j+1)))$
    $v \leftarrow$融合乘加$_{64}(\alpha_j, v, C_{i,\,n+j})$
  结束循环
  $G_{i,n}^{+} \leftarrow \mathrm{nextUp}(C_{i,n})$
  根据式 (28) 解码 $\widehat{\Delta}_{i,n}$
  $q_n \leftarrow$融合乘加$_{64}(h, v, -\widehat{\Delta}_{i,n})$
  $y_n \leftarrow \mathrm{RN}_{64}(G_{i,n}^{+} + q_n)$
  如果 $x > x_{\text{switching}}$ **则**
    $\widehat{F}_n(x) \leftarrow \mathrm{SCALEBYPOWEROFTWO}(y_n, -(2n+1)k_x)$
  else
    $\widehat{F}_n(x) \leftarrow y_n$
  结束 if
结束 for
返回 $\widehat{F}_0(x), \ldots, \widehat{F}_N(x)$

多项式直接逼近$F_n(x)$，而归一化后的多项式则逼近$A_n(u)$。对于后者，在多项式求值之后，根据公式（43）得到相应的 Boys 值。

这种构造使 Boys 函数的两种不同表示能够共用同一泰勒计算。核心运行时求值过程概述于算法 1 中。$x_{\text{switching}}$ 表示主域与归一化大参数域之间依赖于模型的切换边界，而$C_{i,m}$ 表示主域中的普通舍入参考值 $\mathrm{RN}_{64}(F_m(r_i))$，或归一化大参数域中的普通舍入参考值 $\mathrm{RN}_{64}(A_m(u_i))$。

剩余问题是构造兼容的参考网格和公共次数向量，在遵守共享存储预算的同时满足数值目标。接下来，我们将讨论这一联合构造及其数值验证。

D. 联合参考表构造与数值验证

主域表和归一化大参数表采用联合构造，因为它们共用相同的泰勒次数和参考行布局。两者的网格参数可以不同，但多项式次数的选择会影响两张表的精度和存储需求。因此，我们

在总存储预算下，寻找一对兼容的表格。

搜索优先降低泰勒阶数。对于候选大参数网格，会针对请求的输出阶数考虑一个统一的阶数上限。对大参数表示进行数值验证后确定切换边界，该边界也同时决定主域的范围。

为大参数表分配存储后，剩余预算可用于主表。其网格密度由公式（17）中的偏移量 s 控制：在保留的有效数位数固定时，增大 s 通常会使输入区间上的网格变得更稀疏，并减少参考点的数量。因此，我们寻求在剩余预算允许的范围内尽可能小的偏移量，尽可能充分利用可用存储，以获得精细的主网格。该网格可减小泰勒位移，并可能允许采用更低的多项式阶数。

建立两个网格后，在数值验证允许的情况下，降低每个输出阶数所对应的阶数。在两个自变量区域中，对每个阶数保留相同的阶数，从而保持共享泰勒求值过程。选定阶数后，我们增大 s，以减小主表占用空间，同时保留这些阶数并满足数值验证要求。这样，搜索过程首先利用可用存储来减少多项式计算量，然后在不引入额外多项式阶段的情况下减少存储。

图3展示了联合构造的主要阶段。两张表均以相同的逻辑行布局存储参考系数和输出低位修正项。两者的总存储量（包括行填充）不得超过 1 MiB 的预算。数值验证基于选定输入下完整计算出的 Boys 值，并考虑近似、参考数据表示、浮点求值和大参数缩放的综合影响。参数搜索、参考数据构建、验证流程以及选定的模型参数详见补充信息（第 S2 节和表 S1）。

IV. 在积分内核中的实现

我们在 FABO 中实现了所提出的方法，并提供统一计算设备架构（CUDA）和 CPU SIMD 后端。为了评估该方法在成熟量子化学应用中的应用层面福克构建性能，我们将该库集成到 QUICK 中以进行图形处理器计算，并通过其 Simint 积分引擎集成到 NWChem 中以进行 CPU 计算。[1,2,12] 这些集成在选定的积分路径中替换了 Boys 求值，同时保留周围的筛选、积分递推、收缩和福克累加。因此，该实现遵循应用现有的请求模式，无需将自变量收集到单独的 Boys 求值批次中。

A. GPU 线程映射与阶数特化

在 QUICK 的双电子积分内核中，每个参与的线程都会提供一个自变量 x 并请求前缀$F_0(x)\ldots,F_N(x)$，以供后续积分计算使用。因此，我们在同一线程中对一个自变量及其所请求的输出进行求值。求值器以内联方式嵌入积分内核，并将结果直接写入 QUICK 现有的按阶步长排列的工作区。

尽管各线程所需的端点 N 可能不同，但积分调用点和构建配置确定了一个共同的上限。例如，普通 SP 和 SPD 请求的上限分别为 4 和 8，而在 ENABLEF 构建中，SPD 和 SPDF 路径可能需要达到 12 阶。我们在编译时使用相应的上限来选择数值模型，并确定参考表参数、系数偏移量和泰勒阶数。实际端点仍是每个线程的运行时输入，因此，选择支持更高阶数的模型并不意味着每次调用都必须计算这些阶数。

为将这种特化与依赖请求的工作相结合，生成的求值器包含按升序展开的输出阶段序列。每个阶段都有固定的多项式次数和固定的参考数据偏移量。写入结果后，如果已到达请求端点，该阶段便从 Boys 求值器返回；否则，执行将继续进入下一阶段。对请求区间施加的编译期限制会移除不包含在内的阶段，而运行时端点则决定每个参与的线程何时停止。

提前终止还应避免执行仅用于未请求输出的数据访问。预加载整行参考数据会保留这些访问，即使后续多项式阶段已被跳过。反过来，若分别加载每个多项式的系数，就会反复读取彼此重叠的参考值。因此，我们将每次系数加载安排在其首次使用时进行，并保留该值供后续多项式使用。系数以具名标量变量表示，因此在生成的阶段中，其访问和复用都是显式的，而不是通过动态索引的每线程数组来实现。

例如，对于输出 F_0 至F_4 ，所选模型对F_0 和F_1均使用四次多项式。第一阶段需要参考阶数 0 至 4，而第二阶段仅引入阶数 5。若请求在F_1 处结束，则会在后续阶段及其额外的系数加载之前返回。更一般地，加载计划遵循截至目前已处理输出所需系数范围的并集；它不会假定每推进一个输出就恰好需要一个新系数。

终止条件对每个线程单独生效。需要更高阶数的线程会继续执行剩余阶段，无需进行 warp 范围最大阶数归约，也无需全 warp 参与。这适用于 QUICK 筛选积分计算中出现的混合请求和部分参与情况。相同的生成输出阶段既可处理主域参考行，也可处理归一化大参数参考行；这些行的准备和访问方式将在下文介绍

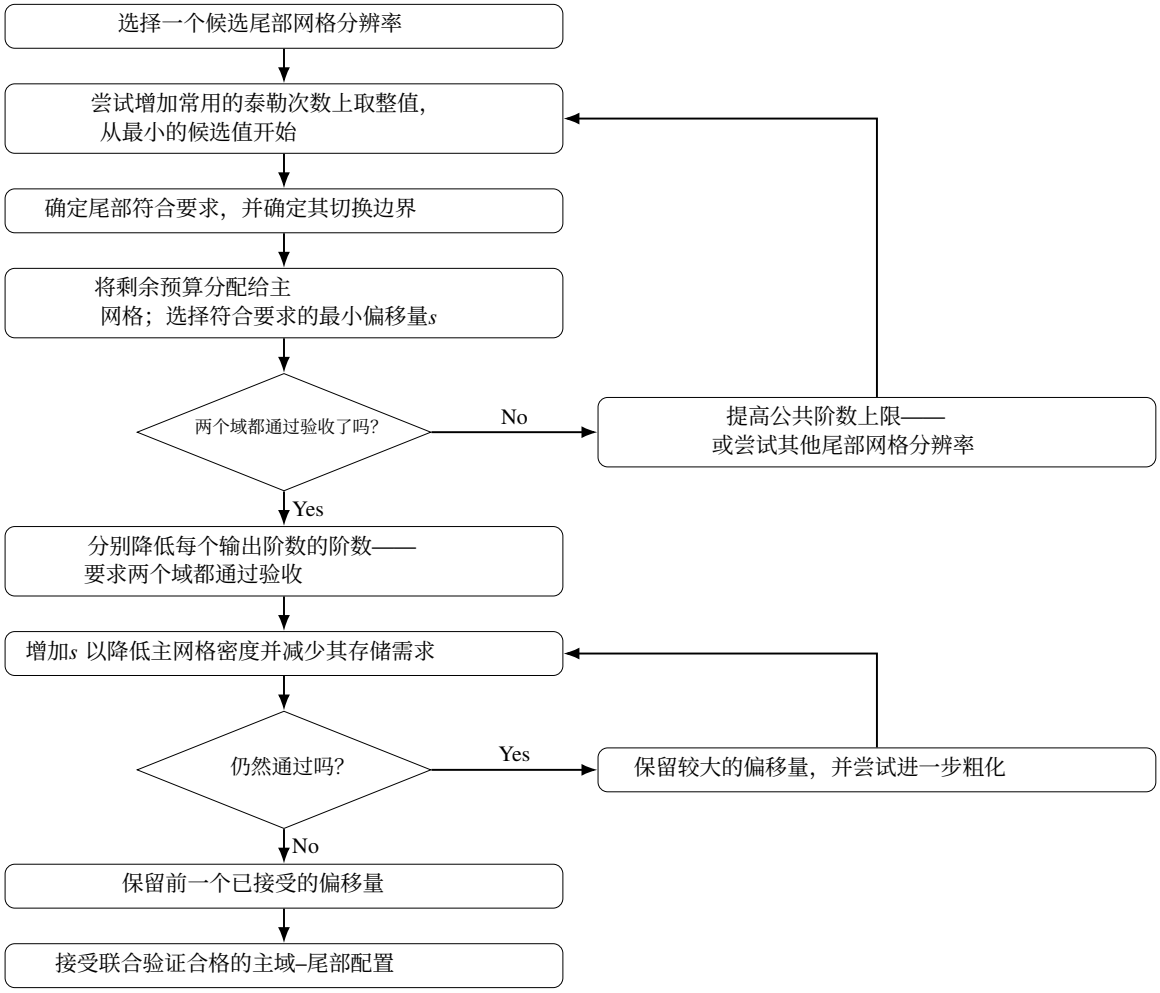

图3. 示意性　　用于联合构造主域和归一化大参数参考表的工作流程　　s.

下一节。

B. 图形处理器参考数据访问与共享路径执行

通用泰勒求值器要求两个数值域以相同格式提供参考数据。因此，我们将主域参考行和归一化大参数参考行存储在同一连续分配中。主域行位于大参数行之前，后者的起始位置由编译时偏移量 tail_row_offset 指定。两个区域采用相同的行大小，并以相同的逻辑布局存放参考系数和输出修正量。因此，可以通过单个表指针和一组通用的系数偏移量访问任一域。

在每一行中，输出修正量与若干组参考系数交错排列。每组最多包含八个对应的输出阶数和参考系数，其前面都有一个修正字；仅供泰勒展开使用的更高阶系数则直接排列在后，不再添加修正字。这种布局提升了缓存局部性，降低了访问这些数据时需要额外读取缓存行的可能性，从而——

——。我们还在两个域中保留相同的布局，使生成的求值器能够使用相同的偏移量访问任一参考序列。

这两个域在参考坐标的准备方式上有所不同。对于主域中的输入，求值器会形成平移后的坐标$x + s$，选取相应的网格点，重构其参考坐标，并计算相对于x的泰勒位移。对于大自变量输入，求值器首先根据公式（42）对自变量进行归一化，得到u及相应的缩放指数k_x。随后，它从归一化网格中选取一个参考点，并计算相对于u的位移。通过将编译时偏移量加到局部行索引上，即可访问所选的尾部行。

一旦完成参考行和泰勒位移的准备，两个域便进入相同的求值过程。它们遵循完全相同的参考系数加载、校正解码和阶乘霍纳运算序列。所得泰勒多项式可直接近似主域输入的$F_n(x)$，而对于大参数输入，则近似$A_n(u)$。因此，与域相关的运算

仅限于参考坐标准备，而系数访问和多项式求值则是共用的。

大参数结果在执行常规泰勒求值后还需要一个额外步骤。利用公式（43）中的缩放恒等式，可将归一化结果转换回原始自变量：

$$\widehat{F}_n(x) = 2^{-(2n+1)k_x} T_n, \tag{48}$$

其中，T_n 是针对归一化大参数模型计算得到的泰勒结果。主域结果无需进行此类重新缩放，可以直接存储。在实现中，这两种情况通过一个通用的输出缩放接口处理，其中主域情况退化为恒等操作。

这种组织方式将主域/尾部的区别限制在坐标准备和最终输出组装阶段，从而可以共享计算量较大的系数加载和多项式求值阶段。它不要求相邻线程选择相同的参考行或自变量域。具有不同所请求输出终点的线程仍可如第 IVA 节所述，在不同阶段终止，但域选择和参考数据访问均不需要全线程束操作。

C. CPU SIMD 扩展

中央处理器实现遵循现有的 NWChem–Simint 积分求值组织方式。Simint 的生成内核对原始积分任务进行向量化，每个 SIMD 通道提供自己的 Boys 参数。在单次向量调用中，所有通道请求相同的输出阶数范围，该范围在编译时已知。然而，连续调用不必请求相同的范围，因为 NWChem 可能会组合不同的壳层四元组组。因此，我们根据每个生成的积分内核的输出阶数范围对求值器进行专门化，而不是假设整个批次具有统一的阶数。

基于 Highway 的单指令多数据求值器为每个通道分配一个自变量，并将所需的 Boys 值直接写入 Simint 现有的向量工作区。[13] 它沿用与图形处理器实现相同的数值模型、组合参考表和行布局。由于不同通道可以选择不同的参考点，甚至不同的自变量域，因此在收集所需的参考系数和校正项之前，会分别确定它们的逻辑行索引。收集完这些数据后，各通道执行相同的向量泰勒求值，包括低位部分校正和最终缩放。

因此，CPU 后端沿用与 CUDA 实现相同的数值构造和组合参考表，先根据通道选择参考值，再进行共享的向量泰勒求值。这在传统 CPU 积分引擎中实现了所提出方法的单指令多数据实现。

V. 实验方法

我们通过函数级精度测试和应用级基准测试评估了所提出的方法。主要性能比较采用 GPU 上的 QUICK，而 NWChem–Simint 则提供了补充的 CPU SIMD 求值。[1,2,12] 我们还比较了 Boys 函数实现对分子计算结果的影响。本节介绍参与比较的实现和计算平台，随后说明工作负载、计时流程和精度评估。

A. 应用、平台与对比实现

在图形处理器基准测试中，我们将 FABO 与 QUICK 原始的 Boys 求值器，以及改编自 LibintX、Vikhamar-Sandberg–Repisky (VSR) 和 GANSU 的求值器进行了比较。[2,3,6,8] LibintX 和 GANSU 仅在接口和输出布局层面进行了改编，同时保留其数值求值过程；而作者提供的 VSR OpenACC 实现则移植到了 CUDA。每个后端都编译为单独的 QUICK 可执行文件，外围的筛选、积分递推、收缩和福克矩阵累加均保持不变。

对于中央处理器基准测试，我们在 NWChem 中使用 Simint 积分引擎，将 FABO 与 Simint 原始的 Boys 求值器、LibintX、VSR 以及 Laikov 的近似方法进行了比较。[1,3,4,6] LibintX 保留了其原生的单指令多数据实现，而 VSR 和 Laikov 则通过基于 Highway 的单指令多数据移植版本进行评估。[13] 我们将 Boys 后端替换进 Simint 生成的积分核中，并将其分别链接到独立的 NWChem 可执行文件。

QUICK 计算使用 NVIDIA GeForce RTX 5090 D v2 图形处理器。CUDA 代码使用 NVCC 13.4.92 编译，目标架构为 sm_120，同时保留 QUICK 的 -O2 和 –use_fast_math 选项。我们没有启用其 f 计算路径，该路径默认处于禁用状态。NWChem–Simint 计算在 Intel Xeon Gold 6526Y 处理器上以单线程运行，使用 Intel 高级矢量扩展 512（AVX-512）单指令多数据指令集。所有中央处理器积分变体均使用 GCC 14.4.0 编译，并采用 -O3、-ffast-math 和 -march=sapphirerapids 选项。中央处理器线性代数运算使用单线程 Intel 数学核心函数库（MKL），并启用自动指令集分派。这些设置适用于计时版本；数值验证版本见第 VD 节。完整的硬件规格、软件版本、源代码修订版本和构建设置见补充信息（第 S5.1 节和表 S3）。

B. 工作负载与模型选择

应用基准测试采用限制性 Hartree–Fock (RHF) 方法，对水、尿嘧啶、苯和戊烷二聚体进行计算，分别对应 S66 中的系统 1、17、24 和 34。[14] 我们使用平衡分离几何构型以及 6-31G*、[15,16] def2-SVP、[17] 和 cc-pVDZ[18] 基组。

每项基准测试都从固定的初始密度出发，评估一次密度更新。

为获得足够长的执行时间，我们将每种二聚体的多个平移副本线性堆叠，同时不改变其内部几何结构。我们分别为 QUICK 和 NWChem 选择堆叠规模。所得图形处理器堆叠中，水、尿嘧啶、苯和戊烷分别包含 128、8、16 和 16 个二聚体；中央处理器工作负载则包含 8 个水二聚体，以及其他每种体系各 1 个二聚体。选定后，所有三种基组和所有 Boys 后端均使用固定的几何结构和堆叠数量。完整的几何结构及规模记录见补充信息（第 S5.2 节和表 S4）。

FABO 的数值模型采用第 III D 节中的联合构造方法生成。我们针对每个输出范围 $[0,N]$, $0 \le N \le 24$ 分别构建了一个模型，并要求其满足基于采样的一 ULP 验收准则，且每个模型的主表–尾表总预算为 1MiB。该预算包括存储系数、低位修正和行填充。

在 QUICK 中，s/p 和 s/p/d 积分内核系列分别使用 $[0,4]$ 和 $[0,8]$ 模型，并且每个线程仅求值其积分计算所请求的阶数。在 NWChem–Simint 中，每个生成的积分内核都已知所请求的最大 Boys 阶数，并使用相应的 $[0,N]$ 模型。两个应用使用相同的数值选定模型数据；区别仅在于它们的求值接口和执行后端。

C. 性能测量

每次测量均从保存的初始状态执行一次福克构建和一次轨道/密度更新。QUICK 使用其默认的初始 SAD 猜测，而 NWChem 使用其原子初始猜测。在每个应用和分子案例中，对于每个 Boys 后端及每次重复，都重放相同的保存状态。初始化、初始猜测准备、测试固件加载以及 Boys函数表分配和上传均在计时前完成。

对于 QUICK，福克构建区间包含 scf_operator(.false.)，该操作执行完整的、而非密度差分式的福克构造。其中包括依赖密度的筛选准备、相关的主机–设备传输、通过 gpu_get_cshell_eri 进行的双电子积分求值，以及福克矩阵组装。单电子矩阵已提前准备好并重复使用。墙钟时间使用 clock_gettime(CLOCK_MONOTONIC) 测量，并在起始时间戳之前和结束时间戳之前调用 cudaDeviceSynchronize()，因此测量包含图形处理器工作负载的完成时间。

对于 NWChem–Simint，计时区间包含 fock_2e(...)，用于测量对电子库仑作用和交换贡献的 Fock 构建。这些经过的墙钟时间使用 NWChem 的 util_wallsec() 进行测量。因此，中央处理器计时仅测量对电子构建，而 QUICK 计时则包括周围的 Fock 构建操作。矩阵序列化和诊断输出均在计时完成后执行。

每个后端针对每个分子案例测量五次，每次重复使用一个独立进程，并在各次重复测量之间循环调整后端顺序。我们报告五次耗时的算术平均值。各次测量结果及其离散程度保留在补充信息中（第 S6 节以及表 S6 和表 S7）。

D. 函数级精度和分子精度

对于每个被比较的实现以及每个请求的输出范围 $[0,N]$, $0 \le N \le 12$，均评估了函数级精度。上限阶数的选择覆盖了启用 ENABLEF 选项时 QUICK 中 RHF 计算所需的最大 Boys函数阶数。评估采用覆盖 $[0,1000]$ 的均匀网格，间距为 10^{-6}。在采样的 binary64参数处计算高精度参考值；在计算误差之前，不将参考值舍入为 binary64。

针对每个请求的范围，记录绝对误差、相对误差和 ULP 误差的最大值及算术平均值，并将所有采样自变量和所有返回阶数等权汇总。严格算术和快速数学验证结果分别报告。

我们将分子结果与使用可提供正确舍入的binary64 Boys函数值的后端所得结果进行了比较。对于每个案例，我们计算了能量的绝对差异，以及福克矩阵和更新后的密度矩阵中逐元素绝对差异的最大值和平均值。跨案例汇总的结果，是这些量在各案例中的最大值。

为在相同条件下比较不同Boys函数求值器的影响，对于每个分子体系，所有计算均从同一个初始密度矩阵开始。这一点很重要，因为Boys函数求值的差异可能会在连续的自洽场（SCF）迭代中传播，导致不同的密度矩阵，并可能产生不同的依赖密度的筛选决策。因此，各次计算所求值的积分集合及相关计算工作量也可能有所不同。通过固定初始密度矩阵，并仅执行一次Fock构建，随后进行一次对角化和密度更新，我们避免了此前SCF迭代历史所造成的这些差异。该流程能够在相同的应用工作流程内，受控地评估Boys函数求值误差对福克矩阵及后续密度更新的影响。

对于 QUICK，由于动态 GPU 调度，浮点累加的顺序可能因运行而异。由于浮点加法不满足结合律，即使使用相同的 Boys 求值器，这种变化也可能导致计算结果出现差异。因此，在计算差异之前，我们先对每个后端（包括参考实现）运行五次得到的能量和每个矩阵元素取平均值。对于单线程NWChem计算，我们检查了重复输出的可重复性，并将其与相同保存初始状态下的参考值进行比较。

VI. 结果与讨论

我们首先考察 Boys 函数实现的精度，以及 FABO 所需的参考存储。随后，我们比较这些实现应用于 QUICK 和 NWChem–Simint 时的性能。[1,2,12] 最后，我们评估各实现所得分子计算结果与正确舍入的 Boys 参考值之间的差异。

A. Boys 函数精度与参考值存储

图 4 比较了严格算术模式下每个请求输出范围的最大 ULP 误差和平均值。表 I 总结了所有请求范围内的绝对误差、相对误差和 ULP 误差，其中包括严格算术和快速数学结果。误差均以未经舍入的高精度参考值为基准进行测量。对于每个平均值，所有返回值均采用等权重。完整的统计量以及出现最大值的输入见配套数据（参见补充信息中的 S4 节和表 S2）。

对三种误差度量的比较为评估所测试的实现提供了一致的依据：在两个平台和两种算术模式下，无论采用绝对误差、相对误差还是 ULP 误差，FABO 的最大误差和平均值均最小。因此，其精度优势并不取决于误差度量的选择。这些函数级差异在多大程度上会影响后续的分子计算，将在 SVI D 节中单独考察。

尽管 FABO 在测试的实现中实现了最小误差，但其最大的采样 ULP 误差约为 1.067，略高于构造过程中采用的 1 ULP目标。这并不与构造结果矛盾，因为该目标仅针对规定的验证输入进行约束，而当前的密集网格扫描还评估了其他自变量。正如第 IIA 节所述，通过验证测试并不能保证每个 binary64 输入都满足该目标。因此，观察到的超出情况体现的是有限数值验证的局限性，而不是未能满足规定的构造准则。相应的输入x 和输出阶数见补充信息（表 S2）。

其他实现的误差会随请求的阶数以及各自变量区域采用的近似方法而变化。原始 QUICK 求值器在切换为向上递推后不久，误差便会显著增大，此时 $x = 0.1$。对于如此小的自变量，随着递推推进到更高阶，舍入误差会迅速放大。例如，在严格算术下，计算得到的F_{12}(0.100019)约为 -2.36×10^5。LibintX 在中央处理器和图形处理器上的误差表现截然不同，中央处理器上的最大误差大得多。这种差异源于渐近切换点不同：对于 $[0, 8]$ 输出范围，中央处理器实现会在$x = 36$处切换，此时其误差会增大；而图形处理器序列实现则在$x = 117$处切换，因此近似误差较小。

快速数学模式下的结果通常与严格算术模式下的结果接近，不过有少数误差统计量略有改善。我们认为，这些改善是因为编译器能够将独立的乘法和加法操作替换为融合乘加，从而减少舍入步骤。[11]

图5展示了所构造模型的参考存储。所有合并后的主表和尾表均可容纳在分配给每个模型的 1 MiB 预算内。堆叠区域显示了主表和尾表的大小，其中包括存储系数、低阶校正项和行填充。存储量不会随N单调增加，因为网格和多项式次数会针对每个请求范围分别选取，并且网格参数与多项式次数会在相同存储预算下进行联合优化。增加N 会加入更多请求的 Boys 函数，并可能增加每个参考行中存储的系数数量。然而，较高的泰勒次数也可能允许采用更粗的网格，从而减少参考点数量，抵消额外系数带来的影响。图5右轴所示的最大泰勒次数在测试范围内从四次到七次逐步增加。由此产生的存储量变化反映了每个参考行中的系数数量与参考行数量之间的权衡。尽管对于缓存利用率而言，这一 1 MiB 预算可能看似不利，但其对整体性能的影响必须通过端到端应用基准测试进行评估，具体讨论见第 VI B 和 VI C 节。

精度和存储结果展示了 FABO构造 背后的权衡关系。非均匀网格将参考点重新分配到需要更小间距的区域，在不增加参考点数量的情况下减小最大泰勒余项，如图 1 所示。低位修正会引入额外存储开销，但该开销已计入合并表预算，并使最终求值能够保留参考系数舍入为 binary64 后丢失的信息。图5中的存储结果进一步表明，主表和尾表大小共同取决于请求的输出阶数、多项式次数和网格参数。其随N 呈现的非单调变化，反映了在相同存储约束下对每个模型进行的独立优化。重要的是，紧凑表示不需要针对两个自变量区域分别采用多项式求值过程。准备好参考数据和位移量后，两个区域均使用相同的系数加载、修正解码和泰勒求值流程。这种共享的计算结构使得存储量减少和数值精度提升得以实现，而无需针对大自变量引入额外的多项式阶段。然而，这些设计选择对执行时间的影响还取决于周围的积分计算，包括筛选、收缩和 Fock 矩阵组装。因此，以下各节将通过应用级 Fock构建基准测试评估其实际性能。

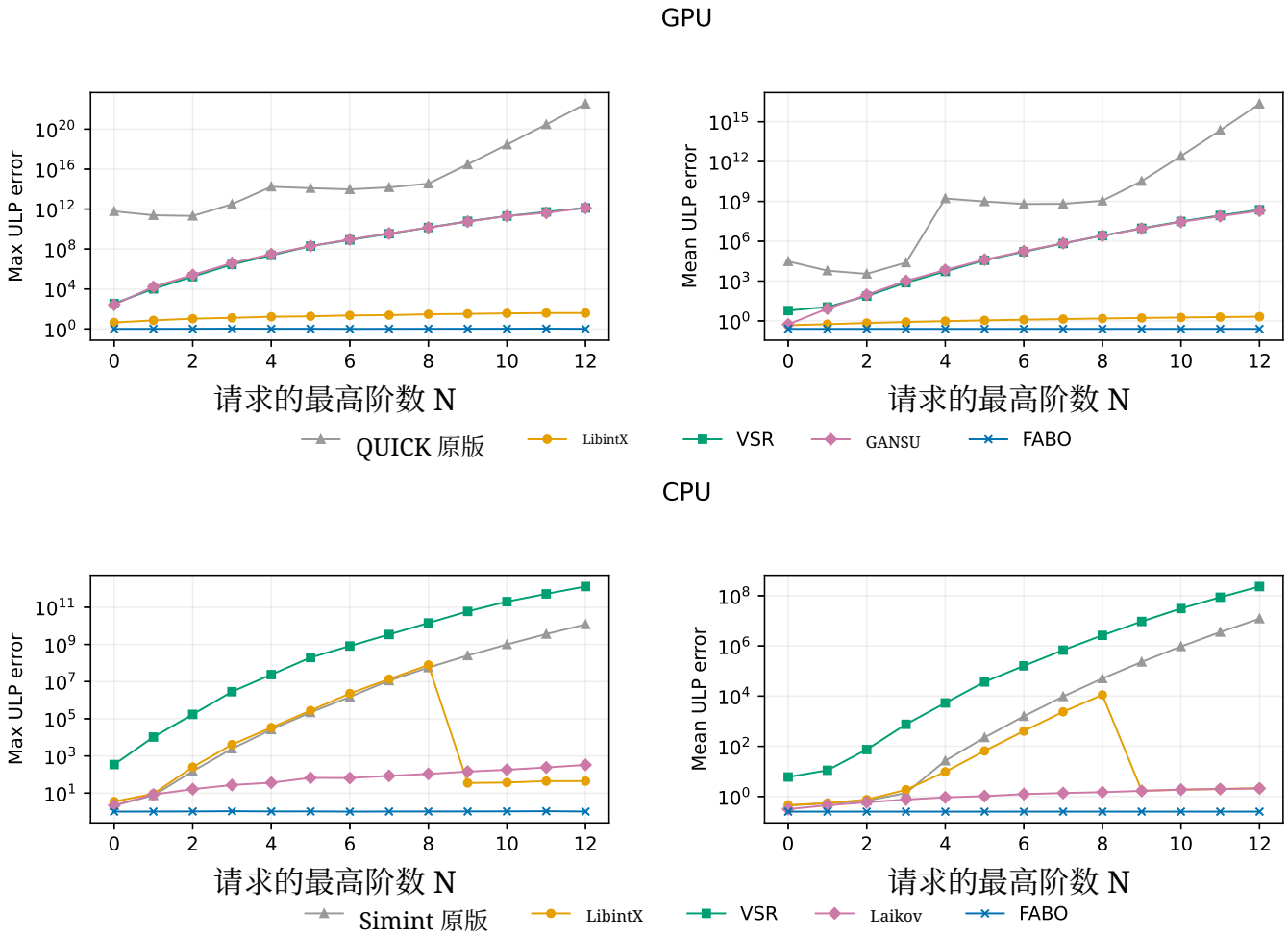

图4. 在严格算术下，针对每个请求的输出范围，图形处理器和中央处理器实现的最大采样 ULP 误差和平均采样 ULP 误差 $[0,N]$ 。每个范围均在 10^{-6} 网格上进行了评估，范围为 $[0,1000]$。

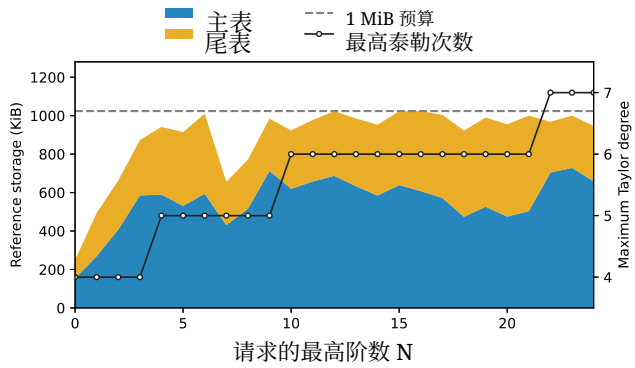

图5. 覆盖输出范围$[0,N]$的每个独立构建数值模型的参考存储和最高泰勒次数。

B. QUICK 中的 GPU 性能

图 6 比较了 QUICK 中五种 Boys 函数求值器得到的 Fock构建时间。相应的数值列于表 II。FABO 在全部十二个测试案例中均取得最低的平均值 Fock 构建时间，并且始终优于 QUICK 原有的 Boys 求值器。这种改进在全部四个分子体系和三种基组中均可观察到，但其幅度会随工作负载而变化。

值得注意的是，函数级精度与应用级性能之间存在差异。如第 VIA 节所示，与其他经过测试的实现相比，FABO 显著降低了 Boys 函数求值误差。然而，测得的 Fock 构建时间不仅包括 Boys 求值，还包括筛选、积分递推、收缩和 Fock 矩阵组装。根据阿姆达尔定律[19]，，如果 Boys 求值占原始执行时间的比例为f，并且加速了 S_{B}倍，则总体

表 I. 各 Boys 求值器在所有请求范围内的最大和平均绝对误差、相对误差及 ULP 误差；请求范围为 $[0,N], N=0,\ldots,12$，参数则在 10^{-6} 网格上于 $[0,1000]$ 范围内采样。

平台 / 模式	Boys 后端	绝对误差		相对误差		ULP 误差	
		Max	Mean	Max	Mean	Max	Mean
图形处理器 / 严格模式	QUICK 原版	2.36×10^{5}	2.21×10^{-2}	6.49×10^{6}	6.11×10^{-1}	3.41×10^{22}	3.19×10^{15}
	LibintX	4.30×10^{-16}	6.70×10^{-19}	5.06×10^{-15}	2.34×10^{-16}	3.93×10^{1}	1.52×10^{0}
	VSR	4.37×10^{-14}	4.52×10^{-17}	1.97×10^{-4}	7.75×10^{-9}	1.30×10^{12}	4.97×10^{7}
	GANSU	7.34×10^{-15}	6.04×10^{-18}	1.67×10^{-4}	6.63×10^{-9}	1.30×10^{12}	4.32×10^{7}
	FABO	7.67×10^{-17}	3.27×10^{-19}	1.87×10^{-16}	4.01×10^{-17}	1.07×10^{0}	2.50×10^{-1}
图形处理器 / 快速数学	QUICK 原版	2.34×10^{5}	2.21×10^{-2}	6.43×10^{6}	6.11×10^{-1}	3.38×10^{22}	3.19×10^{15}
	LibintX	4.30×10^{-16}	6.70×10^{-19}	5.06×10^{-15}	2.34×10^{-16}	3.94×10^{1}	1.52×10^{0}
	VSR	4.37×10^{-14}	4.52×10^{-17}	1.97×10^{-4}	7.75×10^{-9}	1.30×10^{12}	4.97×10^{7}
	GANSU	7.34×10^{-15}	6.04×10^{-18}	1.67×10^{-4}	6.63×10^{-9}	1.30×10^{12}	4.32×10^{7}
	FABO	7.67×10^{-17}	3.27×10^{-19}	1.87×10^{-16}	4.01×10^{-17}	1.07×10^{0}	2.50×10^{-1}
中央处理器 / 严格模式	Simint 原版	2.00×10^{-7}	4.33×10^{-12}	1.37×10^{-6}	3.74×10^{-10}	1.19×10^{10}	2.35×10^{6}
	LibintX	2.91×10^{-16}	6.12×10^{-19}	9.73×10^{-9}	2.01×10^{-13}	7.80×10^{7}	1.35×10^{3}
	VSR	4.37×10^{-14}	4.52×10^{-17}	1.97×10^{-4}	7.75×10^{-9}	1.30×10^{12}	4.97×10^{7}
	Laikov	1.57×10^{-15}	1.07×10^{-18}	3.95×10^{-14}	2.35×10^{-16}	3.29×10^{2}	1.52×10^{0}
	FABO	7.67×10^{-17}	3.27×10^{-19}	1.87×10^{-16}	4.01×10^{-17}	1.07×10^{0}	2.50×10^{-1}
中央处理器 / 快速数学	Simint 原版	2.00×10^{-7}	4.33×10^{-12}	1.37×10^{-6}	3.74×10^{-10}	1.19×10^{10}	2.35×10^{6}
	LibintX	2.90×10^{-16}	6.05×10^{-19}	9.73×10^{-9}	2.01×10^{-13}	7.80×10^{7}	1.35×10^{3}
	VSR	4.37×10^{-14}	4.52×10^{-17}	1.97×10^{-4}	7.75×10^{-9}	1.30×10^{12}	4.97×10^{7}
	Laikov	1.57×10^{-15}	1.09×10^{-18}	3.95×10^{-14}	2.35×10^{-16}	3.29×10^{2}	1.52×10^{0}
	FABO	7.67×10^{-17}	3.27×10^{-19}	1.87×10^{-16}	4.01×10^{-17}	1.07×10^{0}	2.50×10^{-1}

表 II. 图 6 所示 QUICK 基准测试的平均 Fock 构建时间（秒）。

分子	基组	QUICK 原版	LibintX	VSR	GANSU	FABO
128×水二聚体	6-31G*	2.347	2.122	2.221	2.237	2.103
	def2-SVP	2.553	2.410	2.463	2.490	2.408
	cc-pVDZ	3.703	3.350	3.462	3.530	3.297
8×尿嘧啶二聚体	6-31G*	3.674	3.439	3.624	3.669	3.348
	def2-SVP	2.237	2.128	2.212	2.251	2.091
	cc-pVDZ	3.818	3.473	3.667	3.760	3.370
16×苯二聚体	6-31G*	5.918	5.380	5.701	5.765	5.290
	def2-SVP	4.540	4.264	4.424	4.473	4.234
	cc-pVDZ	7.955	7.113	7.503	7.665	6.984
16×戊烷二聚体	6-31G*	5.192	4.668	4.916	5.037	4.567
	def2-SVP	6.192	5.765	5.977	6.082	5.699
	cc-pVDZ	8.823	7.947	8.298	8.535	7.800

加速比受限于

$$S_{\text{total}} = \frac{1}{(1-f)+f/S_{\text{B}}}. \tag{49}$$

这个表达式说明了为什么 Boys 函数求值性能的提升并不能直接转化为完整福克构建性能的相应提升。

因此，人们可能会问，为什么不单独测量 Boys 求值所耗费的时间。主要原因在于，其性能无法脱离周围的积分内核进行评估。该求值器被内联到图形处理器积分核中，其指令与积分递推及其他操作交错执行。特别是，完整内核的寄存器需求决定了每个流式多处理器上能够并发驻留的线程束数量，从而影响隐藏指令和内存延迟的能力。单独测量求值器会改变这种资源使用情况，以及其操作与周围计算相重叠的机会。参考表访问还会与其他内核数据竞争缓存容量，因此其性能取决于完整积分内核的内存访问模式。因此，孤立测量无法可靠地代表 Boys 求值对应用层面 Fock 构建时间的贡献。

在这些替代评估器中，LibintX 的用时与 FABO 最为接近。FABO 相对于 LibintX 的几何平均加速比约为 1.016，这意味着在这十二个案例中，平均执行时间减少了 1.62%。对于采用 def2- 的水分子堆叠体而言，两者差异尤其小，

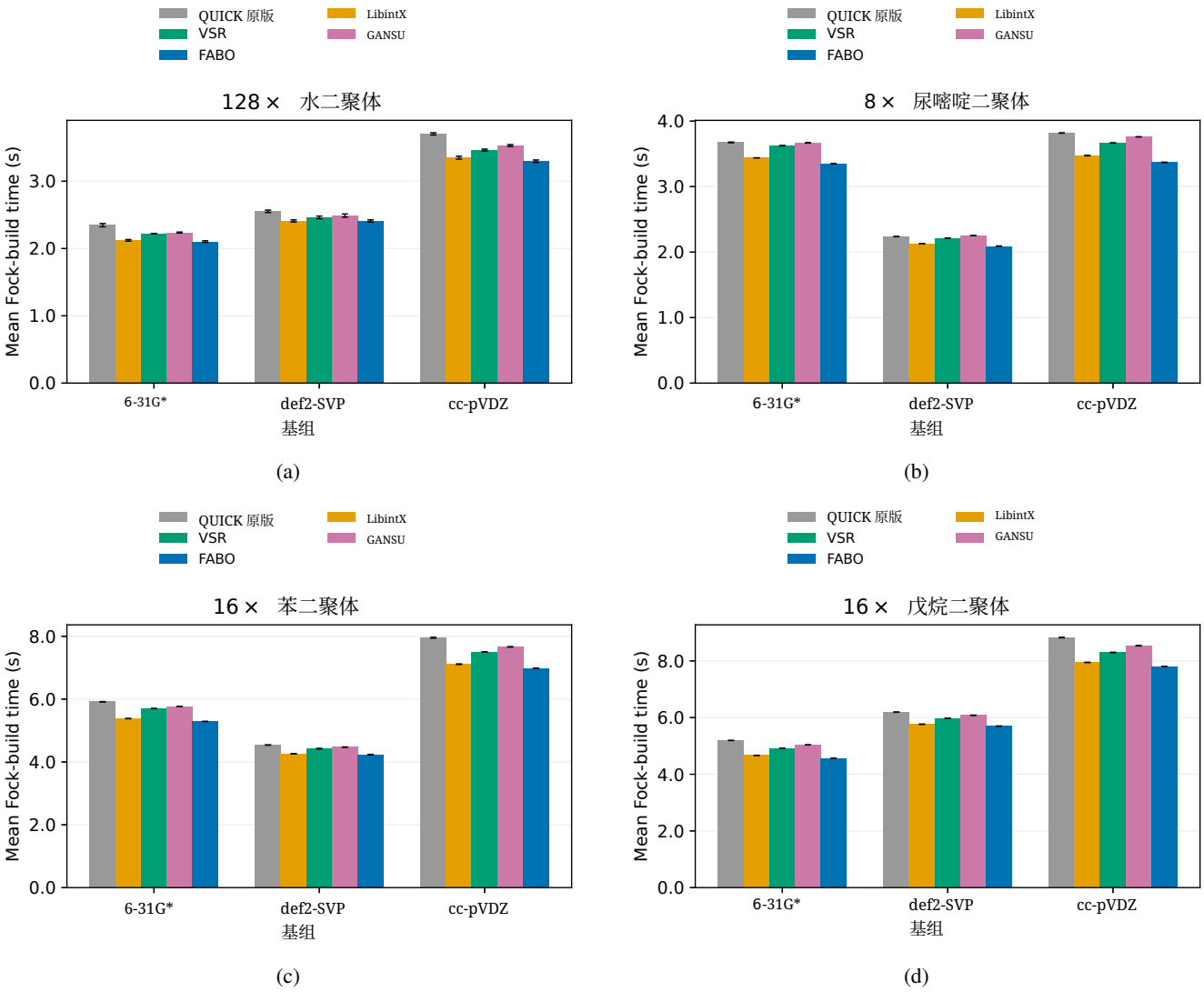

图 6。QUICK 中水、尿嘧啶、苯和戊烷二聚体堆叠的 Fock 构建时间，分别对应 (a)、(b)、(c) 和 (d)。柱状图表示五次测量的平均值，误差线表示一个样本标准差。各面板采用独立的纵轴刻度。

SVP 情况下，FABO 和 LibintX 的平均用时分别为 2.408 秒和 2.410 秒。这一差异远小于任一实现的多次运行间变化，因此在该案例中，两者的性能应视为相当。相对于 VSR 和 GANSU，其对应的几何平均加速比分别为 1.062 和 1.080。相较于 QUICK 的原始求值器，FABO 实现了 1.107 的几何平均加速比，使 Fock 构建时间减少了 9.64%。

这些结果表明，FABO 引入的数值改进可以纳入 QUICK，而不会牺牲应用级性能。具体而言，非均匀参考网格和归一化大参数表示提供了紧凑的参考表，而低位修正提高了最终泰勒求值的精度。尽管存在这些额外的数值特性以及相关的参考数据访问，所得求值器仍能与已有的 GPU 实现保持竞争力。

C. NWChem–Simint 中的 CPU 性能

图 7 和表 III 给出了相应的 NWChem–Simint 结果。中央处理器和图形处理器的测量使用了不同的工作负载，因此它们的绝对时间不能直接比较。

对于每个测试系统和基组，与 Simint 的原始求值器相比，FABO 都能降低平均双电子构建时间。其相对于其他替代方案的性能取决于工作负载。在水计算中，LibintX 的平均时间最低；而在尿嘧啶、苯和戊烷计算中，FABO 的平均时间最低。对于后面这些系统，VSR 的性能也与 FABO 十分接近。

VSR 在水计算中表现相对较佳，可能与其依赖区域的求值策略有关。VSR 在两个有限参数区间内结合使用有理极小极大近似和递推关系，并针对大自变量采用渐近处理。其单指令多数据

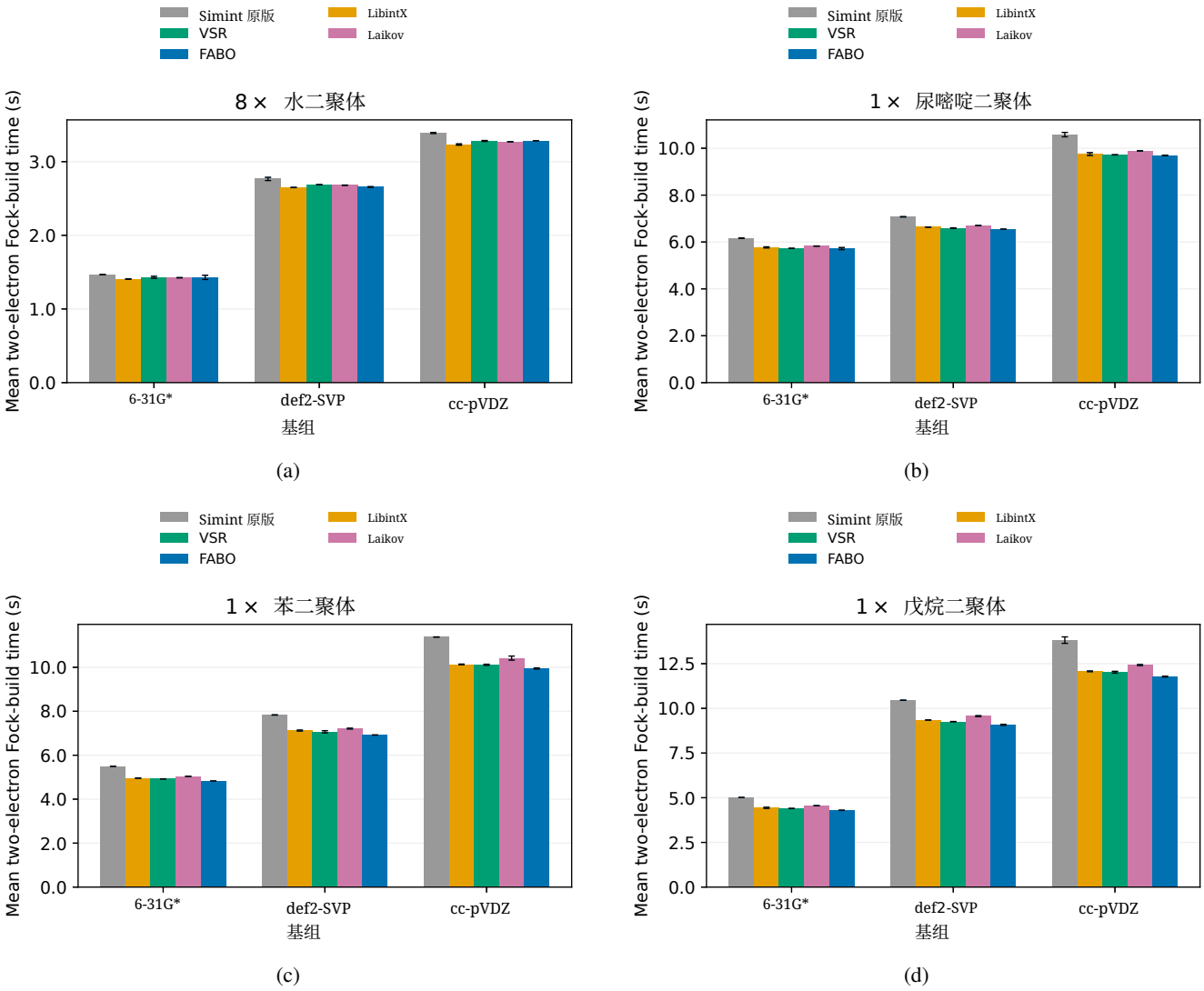

图 7。在 NWChem–Simint 中，对 (a) 水、(b) 尿嘧啶、(c) 苯和 (d) 戊烷二聚体进行的双电子福克构建时间。柱状图表示五次测量的平均值，误差线表示一个样本标准差。各面板采用独立的纵轴刻度。

表 III。图 7 所示 NWChem–Simint 基准测试中的双电子福克构建平均时间（单位：秒）。

分子	基组	Simint 原版	LibintX	VSR	Laikov	FABO
8×水二聚体	6-31G*	1.466	1.405	1.430	1.424	1.429
	def2-SVP	2.765	2.649	2.687	2.680	2.656
	cc-pVDZ	3.389	3.232	3.281	3.269	3.283
1×尿嘧啶二聚体	6-31G*	6.162	5.770	5.728	5.827	5.717
	def2-SVP	7.075	6.635	6.594	6.700	6.543
	cc-pVDZ	10.581	9.752	9.718	9.885	9.693
1×苯二聚体	6-31G*	5.496	4.957	4.923	5.038	4.825
	def2-SVP	7.830	7.122	7.066	7.210	6.918
	cc-pVDZ	11.378	10.131	10.114	10.420	9.945
1×戊烷二聚体	6-31G*	5.011	4.436	4.396	4.552	4.296
	def2-SVP	10.452	9.344	9.257	9.564	9.079
	cc-pVDZ	13.819	12.072	12.021	12.423	11.776

当向量中的所有通道都属于同一求值区域时，实现可以跳过其他求值区域。此行为在中央处理器上可能尤其有利：AVX-512 向量只能容纳 8 个双精度值，而 GPU线程束包含 32 个线程。对于分布相似的输入自变量，规模较小的一组输入更有可能全部落在大参数区域内，从而使 VSR 能够跳过小参数区域的求值。这种情况与水计算有关，因为不同二聚体之间的空间距离可能产生相对较大的 Boys参数。因此，相对较大的自变量与较窄的SIMD向量相结合，可能会提高仅使用渐近求值路径的频率。

除了 VSR 在水计算中表现出良好的性能外，在中央处理器基准测试中，FABO 相较于其他优化求值器的优势也相对有限。这可能部分反映了两种架构上大参数计算的执行方式存在差异。如第 III A 节所述，传统渐近算法会像式（32）那样计算倒平方根。在图形处理器上，双精度倒平方根通常通过近似初始值再进行细化来获得，而细化操作会与周边计算争用浮点执行资源。在中央处理器上，同一操作通常通过平方根和除法来实现。尽管这些操作的延迟相对较长，但正如英特尔优化手册所述，积分内核周边的独立计算可以隐藏其延迟。[20] 因此，传统大参数求值的相对成本在中央处理器上可能较低，从而缩小 FABO 的性能优势。

综合来看，QUICK 和 NWChem–Simint 基准测试表明，FABO 在图形处理器和中央处理器平台上，相较于原始 Boys 求值器，均能持续提升应用级性能。在 QUICK 中，FABO 在所有测试案例中都实现了最低的平均 Fock 构建时间；在 NWChem–Simint 中，FABO 在 75% 的案例中实现了最低的平均时间。特别是，结果表明，FABO 提高数值精度的同时，不会显著损失应用级性能。因此，中央处理器和图形处理器基准测试证明，FABO 在不同计算架构上均具有实际应用价值。

D. 对分子计算结果的影响

表 IV 和图 8 比较了各 Boys 求值器得到的分子结果与使用正确舍入的 Boys 值所得结果。比较内容包括从固定初始状态更新一次后的能量、Fock矩阵和密度矩阵。对于每个矩阵，先对其元素分别取最大值和平均值，再从测试案例中选取最大值。不同列中取得最大值的案例可能不同。完整的逐案例结果见补充信息（表 S8 和 S9）。

在 QUICK 中，四种替代评估器产生的偏差都显著小于原始 Boys 求值器相对于参考值的偏差。其中，FABO 和 LibintX 的能量与福克矩阵结果最接近参考值，最大矩阵误差和平均矩阵误差的绝对值也相近。不过，它们的密度矩阵最大差异略大于 VSR 和 GANSU。这个排序与函数级精度测试中观察到的结果不同。相比之下，FABO 和 LibintX 的密度矩阵平均差异仍小于 VSR 和 GANSU，这与它们更高的函数级精度一致。

这种差异可能源于图形处理器动态调度导致的浮点累加运行间变化。由于浮点加法不满足结合律，即使使用同一个 Boys 求值器，不同的累加顺序也可能产生不同的矩阵元。正确舍入的参考计算中也观察到了这种运行间变化。因此，尽管 FABO 和 LibintX 的平均差异较小，这种变化仍可能导致观测到它们的最大密度矩阵差异较大。比较中采用的五次运行平均降低了单次运行的影响，但无法消除这种变化。因此，QUICK 中的最大密度矩阵差异未必仅反映 Boys 函数求值器的相对精度。

NWChem–Simint 的结果提供了有用的比较，因为单线程 CPU 计算在重复运行时具有可重复性。在这里，FABO 的最大密度矩阵差异最小，其次是 Laikov 的近似方法、LibintX 和 VSR。密度矩阵差异的平均值也呈现相同的排序。尤其是，FABO、LibintX 和 VSR 的相对排序与它们的函数级精度结果一致。这种一致性支持以下解释：QUICK 中最大密度矩阵误差的排序不同，是由图形处理器累加的变化性导致，而非 Boys 函数求值器的内在精度所致。

能量、Fock矩阵和密度矩阵中的误差不一定遵循相同的排序。能量是由输入密度和相应的Fock矩阵得到的标量，而密度矩阵则是在对所得轨道进行对角化和占据处理后获得的。因此，Fock矩阵中的差异可能以不同方式传播到能量和密度矩阵中；此外，后者还可能取决于轨道特征向量对微小扰动的敏感性。矩阵元素差异的最大值和平均值进一步刻画了这些偏差的不同方面。因此，在每种情况下，都不能期望某一个分子量能够重现函数级精度的排序。

总体而言，分子计算表明，FABO 将较高的 Boys 函数精度与具有竞争力的应用级性能结合起来。在 QUICK 和 NWChem–Simint 中，使用 FABO 获得的结果均与使用正确舍入的 Boys 值所得结果保持接近，其中能量和 Fock矩阵的偏差尤其小。其密度矩阵差异在可复现的 CPU 计算中也属于最小之列。所观

表 IV. 相对于正确舍入的 Boys 参考值，分子结果中各案例的最大绝对差异。

应用	Boys 后端	最大值 $\lvert\Delta E\rvert$	最大值 $\lvert\Delta F\rvert$	平均值 $\lvert\Delta F\rvert$	最大值 $\lvert\Delta P\rvert$	平均值 $\lvert\Delta P\rvert$
QUICK（GPU）	QUICK 原版	5.98×10^{-5}	1.78×10^{-6}	1.30×10^{-10}	1.55×10^{-4}	8.00×10^{-8}
	LibintX	2.47×10^{-11}	5.80×10^{-13}	2.58×10^{-16}	7.36×10^{-10}	1.83×10^{-14}
	VSR	2.44×10^{-10}	2.27×10^{-11}	3.97×10^{-15}	3.32×10^{-10}	5.51×10^{-14}
	GANSU	1.69×10^{-10}	2.27×10^{-11}	4.75×10^{-15}	4.63×10^{-10}	6.87×10^{-14}
	FABO	2.18×10^{-11}	5.74×10^{-13}	2.58×10^{-16}	7.11×10^{-10}	1.79×10^{-14}
NWChem–Simint（CPU）	Simint 原版	9.78×10^{-12}	5.10×10^{-11}	5.56×10^{-14}	1.07×10^{-11}	1.11×10^{-13}
	LibintX	4.55×10^{-13}	1.42×10^{-14}	5.95×10^{-17}	2.27×10^{-12}	1.49×10^{-14}
	VSR	1.36×10^{-11}	1.41×10^{-11}	1.66×10^{-14}	1.64×10^{-11}	1.76×10^{-13}
	Laikov	4.55×10^{-13}	2.13×10^{-14}	7.55×10^{-17}	1.60×10^{-12}	1.42×10^{-14}
	FABO	2.27×10^{-13}	1.42×10^{-14}	3.78×10^{-17}	1.22×10^{-12}	9.27×10^{-15}

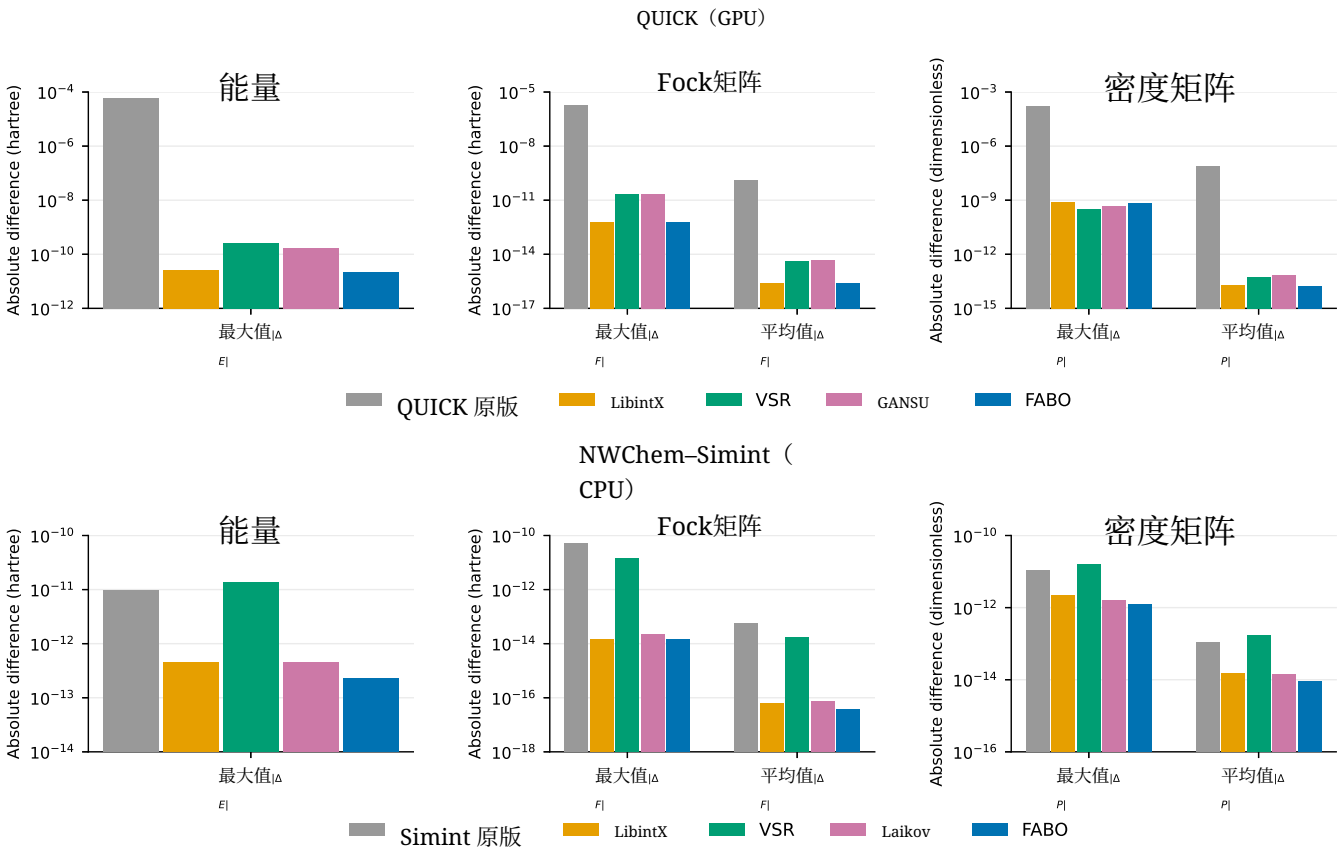

图 8. 相对于正确舍入的 Boys 参考值，能量、Fock矩阵和更新后的密度矩阵在各个算例中的最大差异。柱状图对应表 IV 中的数值。每个面板均使用独立的对数刻度。

察到的 QUICK 中相对较大的最大密度矩阵差异，可能源于非确定性的图形处理器调度以及浮点累加的非结合性。结合函数级精度结果，这些比较表明，FABO 所实现的精度提升在实际分子积分计算中得以保持。此外，在大多数测试案例中，FABO 都提供了适度的应用级加速比，表明其数值精度的提升并未以牺牲计算效率为代价。

VII. 结论

我们开发了一种网格化Taylor方法，用于在分子积分计算中求值 Boys 函数。由移位浮点表示构造的非均匀参考网格，在原点附近提供较小的间距，并在自变量增大时逐步扩大间距。这既减少了参考存储的增长，又使参考点能够直接从输入中选取。紧凑的低位部分修正保留了每个输出的额外信息

常数泰勒系数。

为将该方法扩展到大自变量情形，我们引入了一种基于辅助函数的归一化渐近表示，该辅助函数的导数遵循与 Boys 函数导数相同的算术关系。因此，主域展开和归一化大自变量展开可以共用一系列系数访问和泰勒运算。我们联合构建了它们的参考表，并在综合存储预算下选取了共享的、依赖于阶数的多项式次数。选定模型在规定的验证输入上达到了 1 ULP 目标；独立测试也发现少量超出该目标的情况，这与抽样验证的局限性相符。

我们在 FABO 中实现了该方法，并将其集成到 QUICK 和 NWChem–Simint 中。在可复现的 NWChem–Simint 计算中，对于所考察的每项能量和矩阵度量，FABO 与正确舍入的 Boys 参考值之间的偏差都是最小的。在 QUICK 中，由于累加顺序可能因运行而异，FABO 的总体偏差也处于最小之列，其中能量差异和福克矩阵差异均为最小。对于单次福克构建，与其他受测 Boys 实现相比，FABO 可实现 1.22%–10.67% 的几何平均加速比，具体取决于硬件和用于比较的实现。

致谢

本研究得到国家自然科学基金（项目编号：22473092、22373077）和中央高校基本科研业务费专项资金（项目编号：20720260032）的资助。

数据可用性声明

FABO 的源代码可从 https://github.com/LXYan2333/fabo 获取。图表所依据的紧凑记录及其复现说明可从 FABO release paper-si-v1 获取。

[1]B. P. Pritchard 和 E. Chow，《电子排斥积分的水平向量化》，Journal of Computational Chemistry **37**，2537–2546（2016）。[2]V. W. D. Cruzeiro、M. Manathunga、K. M. Merz, Jr. 和 A. W. Götz，《使用AMBER和QUICK进行的开源多GPU加速QM/MM模拟》，Journal of Chemical Information and Modeling **61**，2109–2115（2021）。[3]R. Vikhamar-Sandberg 和 M. Repisky，《使用极小极大近似高效求值 Boys 函数》，The Journal of Chemical Physics **164**，154111（2026）。[4]D. N. Laikov，《向量化计算中 Boys 函数的全局近似》，The Journal of Chemical Physics **163**，024104（2025）。[5]S. Tsuji、Y. Ito、K. Nakano 和 A. Kasagi，《计算量子化学中 Boys 函数求值的图形处理器加速》，并发与计算：实践与经验 **37**，e8328（2025）。[6]A. Asadchev 和 E. F. Valeev，《适用于SIMD处理器的高斯AO积分McMurchie–Davidson算法实现》，The Journal of Physical Chemistry A **129**，9788–9797（2025）。[7]L. E. McMurchie 和 E. R. Davidson，《笛卡尔高斯函数上的单电子和双电子积分》，Journal of Computational Physics **26**，218–231（1978）。[8]Y. Ito、S. Tsuji、K. Nakano 和 A. Kasagi，《GANSU：面向GPU的量子化学框架，用于高效的Hartree–Fock和后HF计算》，Eng **7**，205（2026）。[9]https://github.com/LXYan2333/fabo。[10]D. Goldberg，《每位计算机科学家都应了解的浮点运算知识》，ACM Computing Surveys **23**，5–48（1991）。[11]IEEE，《IEEE浮点运算标准》，IEEE Std 754-2019（修订版 IEEE 754-2008），1–84（2019）。[12]E. Aprà、E. J. Bylaska、W. A. de Jong、N. Govind、K. Kowalski、T. P. Straatsma、M. Valiev、H. J. J. van Dam、Y. Alexeev、J. Anchell、V. Anisimov、F. W. Aquino、R. Atta-Fynn、J. Autschbach、N. P. Bauman、J. C. Becca、D. E. Bernholdt、K. Bhaskaran-Nair、S. Bogatko、P. Borowski、J. Boschen、J. Brabec、A. Bruner、E. Cauët、Y. Chen、G. N. Chuev、C. J. Cramer、J. Daily、M. J. O. Deegan、T. H. Dunning, Jr.、M. Dupuis、K. G. Dyall、G. I. Fann、S. A. Fischer、A. Fonari、H. Früchtl、L. Gagliardi、J. Garza、N. Gawande、S. Ghosh、K. Glaesemann、A. W. Götz、J. Hammond、V. Helms、E. D. Hermes、K. Hirao、S. Hirata、M. Jacquelin、L. Jensen、B. G. Johnson、H. Jónsson、R. A. Kendall、M. Klemm、R. Kobayashi、V. Konkov、S. Krishnamoorthy、M. Krishnan、Z. Lin、R. D. Lins、R. J. Littlefield、A. J. Logsdail、K. Lopata、W. Ma、A. V. Marenich、J. Martin del Campo、D. Mejia-Rodriguez、J. E. Moore、J. M. Mullin、T. Nakajima、D. R. Nascimento、J. A. Nichols、P. J. Nichols、J. Nieplocha、A. Otero-de-la-Roza、B. Palmer、A. Panyala、T. Pirojsirikul、B. Peng、R. Peverati、J. Pittner、L. Pollack、R. M. Richard、P. Sadayappan、G. C. Schatz、W. A. Shelton、D. W. Silverstein、D. M. A. Smith、T. A. Soares、D. Song、M. Swart、H. L. Taylor、G. S. Thomas、V. Tipparaju、D. G. Truhlar、K. Tsemekhman、T. Van Voorhis、Á. Vázquez-Mayagoitia、P. Verma、O. Villa、A. Vishnu、K. D. Vogiatzis、D. Wang、J. H. Weare、M. J. Williamson、T. L. Windus、K. Woliński、A. T. Wong、Q. Wu、C. Yang、Q. Yu、M. Zacharias、Z. Zhang、Y. Zhao 和 R. J. Harrison，《NWChem：过去、现在与未来》，The Journal of Chemical Physics**152**，184102（2020）。[13]Google LLC，《Highway：支持性能可移植、长度无关SIMD及运行时分派》，https://github.com/google/highway（2026）。[14]J. Řezáč、K. E. Riley 和 P. Hobza，《S66：一个均衡良好的基准相互作用能数据库，与生物分子结构相关》，Journal of Chemical Theory and Computation**7**，2427–2438（2011）。[15]W. J. Hehre、R. Ditchfield 和 J. A. Pople，《自洽分子轨道方法。XII. 用于有机分子分子轨道研究的高斯型基组的进一步扩展》，The Journal of Chemical Physics **56**，2257–2261（1972）。[16]P. C. Hariharan 和 J. A. Pople，《极化函数对分子轨道氢化能的影响》，Theoretica Chimica Acta **28**，213–222（1973）。[17]F. Weigend 和 R. Ahlrichs，《适用于H至Rn的分裂价层、三ζ价层和四ζ价层质量的均衡基组：设计与精度评估》，Physical Chemistry Chemical Physics **7**，3297–3305（2005）。[18]T. H. Dunning, Jr.，《用于相关分子计算的高斯基组。I. 硼至氖的原子及氢》，The Journal of Chemical Physics **90**，1007–1023（1989）。[19]G. M. Amdahl，《实现大规模计算能力的单处理器方法的有效性》，载于*1967*年*4*月*18*日至*20*日春季联合计算机会议论文集（*AFIPS '67*）（ACM Press，1967年），第483–485页。[20]Intel Corporation，*Intel® 64*和*IA-32*架构优化参考手册：第*1*卷，Intel Corporation（2026），文档编号248966-051US，第17.25节，第17-77页。

《使用非均匀网格泰勒展开快速准确地求值 Boys 函数》的补充信息

Longxiang Yan、Wei Wu 和 Peifeng Su厦
门大学，厦门 361005，中国

本补充材料介绍 FABO（Fast and Accurate BOys）的数值构造、浮点求值、函数测试和分子测试。数值验证采用选定的输入，而非穷举所有 binary64 参数。定向参考包围区间控制测试输入处的参考不确定性；但这并不能确立实现误差在连续定义域上的界。配套数据保留了完整的配置和测量结果。

目录

S1 数值表示与实现细节 2
S1.1 算术约定.2 S1.2 主网格索引与位移.2 S1.3 校正编码与最终组装.2 S1.4 尾部归一化与缩放.3 S1.5 共享表与输出阶段.3

S2 参考数据构建与数值验证 3
S2.1 参考算术.3 S2.2 联合搜索与保留的选项.4 S2.3 采样、细化与验收.4 S2.4 选定参数.6

S3 数值示例详情 6
S3.1 匹配参考网格.6 S3.2 向上递推通过窗口.7

S4 独立的函数级精度评估 8

S5 应用程序集成和基准测试设置 11
S5.1 平台和编译.11 S5.1.1 源代码与构建映射.12 S5.2 冻结初始状态和工作负载.12 S5.3 计时边界.14

S6单项计时和加速比统计量 14

S7 分子结果比较和配套数据 17
S7.1 正确舍入的 Boys 后端.17 S7.2 诊断信息和重复输出削减.17 S7.3 配套数据指南.21 S7.3.1 重新运行计算.22

S1 数值表示与实现细节

S1.1 算术约定

输入为有限的 binary64 值 $x \geq 0$，包括负零。请求的阶数位于所选模型的范围内。算术约定为就近舍入，偶数舍入，并采用渐进下溢；用户不应启用浮点陷阱。NaN、无穷大以及负的非零输入均不在该约定范围内。在严格构建中，显式的融合乘加操作仍保持融合状态：禁用收缩会阻止编译器合并原本分开的操作，而不会阻止使用显式 FMA。

令 U 将正的 binary64 值映射为其无符号 64 位表示，并令 U^{-1} 为逆位重解释。全文始终使用主文稿中的 r_i、 $h = r_i - x$、 p_n 和 $G_{i,\,n} = \mathrm{RN}_{64}(F_n(r_i))$。binary64 参考系数与用于构造它们的数学值相区分。

S1.2 主网格索引与位移

对于保留尾数位数 b_g，令 $d = 52 - b_g$。主网格使用

$$z = \mathrm{RN}_{64}(x + s), \qquad K = (U(z) + 2^{d-1}) \gg d, \tag{S1}$$

$$i = K - K_{\min}, \qquad r_i = \mathrm{RN}_{64}(U^{-1}(K \ll d) - s), \tag{S2}$$

$$h = \mathrm{RN}_{64}(r_i - x). \tag{S3}$$

整数量化器将中点情形舍入到较大的键。因此，它不同于采用舍入到偶数规则的浮点加法和减法。SIMD 实现先加上舍入位，然后进行移位，如上面对 K 的表达式所示。在 CUDA 中，当 $d > 32$ 时，仅高 32 位字就包含进行此量化所需的全部位。这会改变指令宽度，但不会改变所选的键。

主域包括 $x = C$；尾部为 $x > C$ 选取。定义 $K_{\min} = K(0)$ 和 $K_{\max} = K(C)$，得到 $K_{\max} - K_{\min} + 1$ 行。偏移码为 $U(s) \gg d$； s 位于保留格点上，因此在零处得到一个节点。最后一个主节点不一定等于 C。在构造过程中，按照非负 binary64 位序进行单调搜索，通过完整舍入映射定位分配给每个键的第一个和最后一个可表示自变量。资格认定使用这些实际单元，并将其裁剪到允许定义域，而不是使用理想实数中点。

S1.3 校正编码与最终组装

写作 $G = \mathrm{RN}_{64}(F_n(r_i))$、 $G^+ = U^{-1}(U(G) + 1)$ 和 $u_G = \mathrm{ulp}(G)$。编码器构造 $\delta = (G^+ - F_n(r_i))/u_G$ 并选择

$$c = \begin{cases} \mathrm{RN}_{\mathrm{int}}(256(\delta - 1/2)), & \delta \leq 1, \\ 128 + \mathrm{RN}_{\mathrm{int}}(128(\delta - 1)), & \delta > 1, \end{cases} \tag{S4}$$

其中整数舍入采用舍入到最近值，平局时舍入到偶数。编码值限制为 $0 \leq c \leq 192$。定向界必须在舍入系数和编码值上均一致；未解决的情况不予接受。该构造还会检查正性、介于 54 和 2046 之间的偏置系数指数，以及有限的 G^+。

解码后的校正值无需经过浮点转换即可组装：

$$\widehat{\Delta} = U^{-1}[(U(G) \,\&\, \texttt{0x7ff0000000000000}) - (53 \ll 52) + (c \ll 45)]\,. \tag{S5}$$

这些加法是整数加法，而非按位或：编码高位会影响指数。指数取自常规的 G，即使 G^+ 跨越二进制阶边界也是如此。最低存储阶已在离线预偏置为 G^+，因为它不可能作为任何支持的输出中的非常数系数出现。解码时减去 1，即可恢复其原始位。其他系数保留常规舍入值；在求值时，其最终高位部分会递增。

对于每个所需的 j，计算 $h_j = \mathrm{RN}_{64}(h\mathrm{RN}_{64}(1/j))$。非常数霍纳链使用常规舍入系数和显式融合乘加。如果其最终累加器为 q_n，则输出组装为

$$t_n = \mathrm{FMA}_{64}(h, q_n, -\widehat{\Delta}_n), \qquad v_n = \mathrm{RN}_{64}(G_n^+ + t_n). \tag{S6}$$

外层加法是一次独立的舍入。算术屏障和显式 CUDA 舍入内建函数可保持这一结构；不得将其替换为以 G_n^+ 为加数的一次融合乘加。

S1.4 尾部归一化与缩放

对于尾部参数，令 e 为其无偏二进制指数，$k = \lfloor e/2 \rfloor$，并保留指数为 $e \bmod 2$ 的有效数。由此得到 $x = 4^k u$，其中 $1 \le u < 4$。尾部网格使用相同的向上取整量化器，保留 u 的 b_t 个小数位。其局部键起点为 $1023\,2^{b_t}$，参考节点包含 4：紧邻 4 以下的自变量可能会舍入到该终端节点。位移为 $\mathrm{RN}_{64}(u_i - u)$。

共享多项式在尾部近似 $A_n(u)$，并按 $2^{-(2n+1)k}$ 缩放。尾部修正编码完全采用主网格规则，只需将 $F_n(r_i)$ 替换为 $A_n(u_i)$。对于偏置指数为 E_v 的正正规多项式结果，若 $(2n+1)k < E_v$，则从其表示中执行 $((2n+1)k) \ll 52$ 整数减法，即可实现精确正规缩放。对于次正规数和零结果，CUDA 则使用 ldexp。include/fast_boys/detail/simd/ops-inl.h 中的单指令多数据辅助函数 Scale 在所有通道仍为正规数时立即返回；否则，它会使用 Highway 的 MulByPow2，并传入带符号的负缩放指数。该路径要求采用渐进下溢和最近舍入。主域预处理会设置 $k = 0$，使缩放成为恒等操作。

S1.5 共享表与输出阶段

主行和尾部行占用同一块连续分配的内存，且行宽相同。尾部索引以主行数为偏移量。每个修正字最多包含八个字节，较低的输出阶数占用较低位，并位于对应的系数组之前。仅供非恒定泰勒项使用的系数随后排列，不再附加修正字。对于 $[0, 4]$ 模型，一个字存放五个修正字节和三个填充字节，之后是系数序列。1 MiB 限制适用于每个模型的主域与尾域的合并分配，包括填充，而不是分别限制每个域，也不是限制所有模型的总和。

CUDA 按输出阶数升序生成直线执行阶段，在每个系数首次使用时加载，并在线程到达请求的端点时返回。这里没有跨整个线程束的端点归约，也没有动态索引系数数组。单指令多数据输出边界是编译时常量，结果通过受限向量引用写入。原生聚集和软件聚集使用相同的参考数据和数值算法。

S2 参考数据构建与数值验证 n

S2.1 参考算术

固定版本生成器记录了 Python 3.12.15、gmpy2 2.3.2、GMP 6.3.0 和 MPFR 4.2.2。MPFR 的定向舍入用于提供包围区间，而不是依赖两个浮点近似值的一致性。只有当 Binary64 系数的包围区间端点舍入到同一数值时，才会接受这些系数；修正码也要求端点一致。对于未决判定，精度会从 256 位逐步提高至 4096 位。无法判定某项决策与已证实的数值失败会分别处理。

对于非负的 x，生成器使用的正项表示为

$$F_n(x) = e^{-x} \sum_{j=0}^{\infty} t_j, \qquad t_0 = \frac{1}{2n+1}, \qquad t_{j+1} = t_j \frac{2x}{2n+2j+3}. \tag{S7}$$

求和至 t_J 后，设 $r_J = 2x/(2n+2J+3)$。后续比值逐渐减小，因此，当 $r_J < 1$ 时，未求和级数由 $t_J r_J/(1-r_J)$ 进行界定。下界项和上界项以及部分和分别采用 MPFR 向负无穷和正无穷舍入的方式传播。上界比值、向下舍入的 $1 - r_J$ 以及向上舍入的除法给出保守余项 R_J。在精度 q 下，当 $R_J \le S_J^{\dagger} 2^{-q+16}$ 时停止求和，其中右侧采用向下舍入。将下界部分和与上界部分和加上 R_J 分别乘以 e^{-x} 的定向界，即可包围 $F_n(x)$。如果在满足该测试前达到 100000 个项递推步数的上限，则参考计算失败；它不会接受最后一个部分和。实现位于 data/records/numerical/paired-1ulp/producer/workflow/additive/reference.py，其中包含函数 enclosure 和 reference_sequence。在 $x \ge \max(n+1,\ q+2048)$ 时，直接辅助函数则改为界定完整积分，并从其下端点中减去 $2^{-(q+2048)}$。

此数学包围区间独立于候选评估器。生成器的 rn64、add64、mul64 和 fma64 函数先使用精确有理数操作数，再对每个指定操作执行一次舍入到偶数的 binary64 舍入；FMA 仅对精确乘积与加数的和舍入一次。因此，资格认定会复现候选项的舍入序列，而不是在 MPFR 中求值其多项式后仅对最终结果舍入。

局部泰勒区间参考点可加速资格认定。辅助函数 taylor_reference 使用 32 项（31 次）来计算所需的最高阶数 m。在中心点 r 处，其截断误差至多为

$$\frac{|x-r|^{32}}{32!}F_{m+32}(\min(x,r)) \le \frac{|x-r|^{32}}{32!}\min\left\{\frac{1}{2(m+32)+1}, A_{m+32}(\min(x,r))\right\}. \tag{S8}$$

在零左端点处，第一个界就足够了。定向系数区间和定向霍纳运算包含参考数据和求值不确定性，并将截断界向外扩展。较低阶数通过带有指数包络的定向向下递推获得。对于含糊的主域和近切换点误差判定，将返回直接的 MPFR精化。辅助函数由 $A_0(u) = \sqrt{\pi}/(2\sqrt{u})$ 和$A_{n+1}(u) = (2n+1)A_n(u)/(2u)$ 构造，并使用定向界；这些是离线参考运算，而不是运行时的系数重构。

尾部验证将缩放后的多项式与 Boys 函数进行比较，而不仅仅是与 A_n 比较。令 $a_n = \max(n-1/2, 0)$。对于 $y \ge 1$, $y^{n-1/2} \le \exp(a_n(y-1))$，根据对数 $y \le y-1$。因此，

$$B_n(x) = A_n(x) - F_n(x) = \frac{1}{2}\int_1^\infty y^{n-1/2}e^{-xy}\,dy \le \frac{e^{-x}}{2(x-a_n)}, \qquad x > a_n. \tag{S9}$$

该实现的条件 $x \ge n+1$ 已足够。对于其支持的远尾参考范围 $0 \le n \le 128$ 和原始输入 $x > 10^7$，分母大于一且 $e^{-x} < 2^{-x} < 2^{-10000000}$，从而证明固定保守界是合理的。此阈值涉及 x，而不是归一化坐标 $u \in [1,4)$。

如果 $x = 4^k u$、 $P = (2n+1)k$ 以及 $[A_L(u)\ A_U(u)]$ 包含 $A_n(u)$，则参考包围区间为

$$\left[\max\{0, 2^{-P}A_L(u) - B_{\mathrm{up}}(x)\},\ 2^{-P}A_U(u)\right]. \tag{S10}$$

首先对辅助区间进行缩放，然后通过向外舍入减去原变量省略积分界。在切换点附近，生成器也可以改用直接 Boys包络。省略积分界只控制其中一项贡献，而不是完整的浮点误差。

S2.2 联合搜索与保留选择

完成后的配置考虑主网格位数 $b_g = 4$、.. . 、12 以及公共多项式阶数上限$p = 1$、. . . 、12。尾部位数候选值从 0 枚举到 20；当即使是最小阶数的尾部加上一行主网格也会超出预算时停止。每个尾部网格包含 $2^{b_t+1}+1$ 个节点。存储元数据用于标识所选候选值；名义组合不会作为已评估候选项计数报告。

对于每一对网格，按递增顺序尝试阶数上限。稀疏尾部筛选先于完整的尾部验证和切换点确定。逐输出切换点中的最大值定义了 C。计入尾部容量分配后，主网格采用行数符合剩余容量的最小格点偏移量。该首次适配操作涉及存储：生成器在二分搜索之前检查单调行数几何关系，并在检查失败时使用精确扫描回退。

先筛选主候选项，再在共同上限处进行资格认定。随后从 1 开始逐阶测试各阶数，要求两个域均通过验收。在此缩减过程中，初始主网格、尾部网格和共同切换点保持不变；只有当较小的尾部次数所对应的资格认定切换点不晚于 C 时，才接受该次数。

固定这些次数后，数值粗化会以指数增长的步长增大主偏移代码，直到某个测试失败点与一个通过点构成区间。通过二分搜索缩小该区间，并测试最终区间周围四个代码值范围内的邻域。保留测试通过的最大偏移量。数值验收不假定具有单调性，因此这是一种有界启发式方法，并不能证明该偏移量是全局最大可容许偏移。正文中用于说明的主工作流程概述了这一策略，而非这些逐项探测过程。

候选项按以下指标依次进行字典序排序：最大次数、次数总和、合计字节数、记录的最大主资格验证误差、次数向量、尾部位数、主位数和偏移码。不会根据分子性能重新调整先前的数值决策。 10^7 值是尾部资格锚点，而不是最终的主域端点。

S2.3 采样、细化和验收

廉价的主筛选会选取最多 17 个均匀分布的行索引，并对每个选定单元使用四个细分。完整的主资格验证会遍历每个舍入索引单元 $[a, z]$ ，并进行 16 个细分。内部样本的构造方式为

二进制表达式 $a+(z-a)(j/16)$，$j=1$、……、15，并分别对每次减法、除法、乘法和加法进行舍入。根据定义域限制，纳入端点、单元内的参考值，以及端点和参考值沿两个可表示方向的相邻值；重复项会被移除。在细化级别 $\ell=0$、……、3，会在 $w\pm(z-a)/(16\,2^{\ell+1})$ 以及 nextafter(w, 0) 和 nextafter(w, $+\infty$) 处探测当前输出的全局最差见证点 w，并保留可接受输入。这并非对每个主单元都进行细化。保留的见证点会以至少 512 位精度直接重新求值。

尾部筛选使用行索引 0、$M-1$、$\lfloor (M-1)/2 \rfloor$ 以及 $\lfloor (M-1)\,i/16 \rfloor$、$i=0$、……、16，其中 M 为尾部行数，并去除重复项。测试五个 binary64 点 $a+(z-a)\,j/4$、$j=0$、……、4，测试时使用$x=2^{24}u$（归一化指数 $k=12$）。完整尾部采样包括端点、4 以下的参考节点，以及 $\mathrm{RN}_{64}(a+(z-a)\,j/17)$、$j=1$、……、16；每个内部表达式都先表示为精确有理数，再进行一次舍入。还包括锚点、其相邻点和最大有限输入的归一化形式。四轮细化将每个单元最差点周围相邻的采样区间二等分，并再次将精确有理数中点舍入一次。

对于每个归一化的 u 和输出 n，令 $k_{\min}$ 为使 $x=4^k u$ 高于资格判定下限的最小指数。远尾检查包括 $k_{\min}$、511，以及所有满足 $-1077 \le \lfloor \log_2 A_U(u) \rfloor-(2n+1)k \le -1020$ 的 $k\in[k_{\min},511]$。这些类别涵盖正规数/次正规数和舍入为零转换；其他正规数类别使用缩放等价性。精确的采样生成函数位于冻结的生成器目录下的 qualify.py、tail.py 和 paired.py 中。

切换点发现测试将归一化样本缩放到 $k=0$、……、11，并在 $[1,10^7]$ 上进行测试。找到最右侧采样失败点后，经过四轮处理，在该失败点与其下一个通过点之间加入 16 个有理数细分点。保留的切换点及其紧邻的后继点都必须通过测试。主分支包含该切换点；尾部分支从严格高于该点的位置开始。该采样集中的通过后缀并不能证明其间的每个 binary64参数都通过。

主域和近切换点判定接受误差上界不超过一个 ULP 的候选项，拒绝下界高于一个 ULP 的候选项，并对模糊参考点进行自适应细化。远尾资格判定器则筛选保守的 256 位上界，并在采样上界超过目标值时拒绝候选项；保留的最差见证点会另行以 512 位重新评估。仅凭这种保守拒绝，并不能证明真实误差超过一个 ULP。未决判定不会取代已接受候选项。逐模型记录保留了样本数、见证点以及采样 ULP 误差的最大报告上界。

S2.4 选定参数

表 S1：选定模型： $H = \max_{0 \le n \le N}(n + p_n)$ 是存储的最高参考阶数。存储内容包括系数、修正项和填充；完整向量及十六进制参数位于 data/models.csv 中。

N	H	b_g	b_t	主行	尾部行	行字节数	总字节数
0	4	10	10	3272	2049	48	255408
1	5	11	11	4931	4097	56	505568
2	6	11	11	6512	4097	64	678976
3	7	11	11	8324	4097	72	894312
4	9	11	11	6861	4097	88	964304
5	10	11	11	5659	4097	96	936576
6	11	11	11	5849	4097	104	1034384
7	12	10	10	3941	2049	112	670880
8	13	10	10	4123	2049	128	790016
9	14	11	10	5365	2049	136	1008304
10	16	10	10	4174	2049	152	945896
11	17	10	10	4207	2049	160	1000960
12	18	10	10	4188	2049	168	1047816
13	19	10	10	3685	2049	176	1009184
14	20	10	10	3254	2049	184	975752
15	21	10	10	3408	2049	192	1047744
16	22	10	10	2991	2049	208	1048320
17	23	10	10	2712	2049	216	1028376
18	24	10	10	2166	2049	224	944160
19	25	10	10	2323	2049	232	1014304
20	26	11	10	2027	2049	240	978240
21	27	12	10	2080	2049	248	1023992
22	29	9	9	2731	1025	264	991584
23	30	9	9	2741	1025	272	1024352
24	31	10	9	2341	1025	288	969408

配套数据 models.csv 提供次数向量、精确的十六进制 C 和 s、两种分辨率、行数、填充以及特定域的字节总数。data/records/numerical/paired-1ulp/ 下的摘要保留了资格验证见证和样本数。规范数值载荷和输出数组的哈希值彼此独立，因为二者的序列化布局不同；两者均已包含最低阶预偏置。README 列出了相应的序列化器和清单。

S3 数值示例详情

S3.1 匹配参考网格

比较采用 $[0, 0]$ 主模型，次数为四，包含 3272 个参考点。保留分数计数为 10，偏移量为 0x1.0300000000000p+2，端点为 0x1.146b4b4e3f86dp+5。均匀比较在 $r_i = iC/3271$ 处放置精确有理数节点，包括两个端点。两个网格使用相同数量的系数行和相同次数；不包括尾部。

在每个精确的相邻中心中点处，计算两侧相邻的数学泰勒多项式，并保留绝对相对余项较大者。定义域端点也会进行求值。本数值示例采用理想中点单元，不同于模型资格验证所用的舍入索引单元。系数是数学上的 Boys 值，而非舍入后的表项；不包含校正解码或运行时求值舍入。

计算采用 MPFR 解析方法 F_0，并通过高精度向上递推计算泰勒系数。256 位和 512 位下的结果相对容差一致，为 10^{-10}；这项精度一致性测试不同于定向

包围证明。记录的绘图环境为 gmpy2 2.3.1 和 MPFR 4.2.2。数学导数余项界提供了额外检验。非均匀网格和均匀网格的最大相对余项分别为 $6.48463556794868 \times 10^{-17}$ 和 $3.11676804346875 \times 10^{-15}$ 。

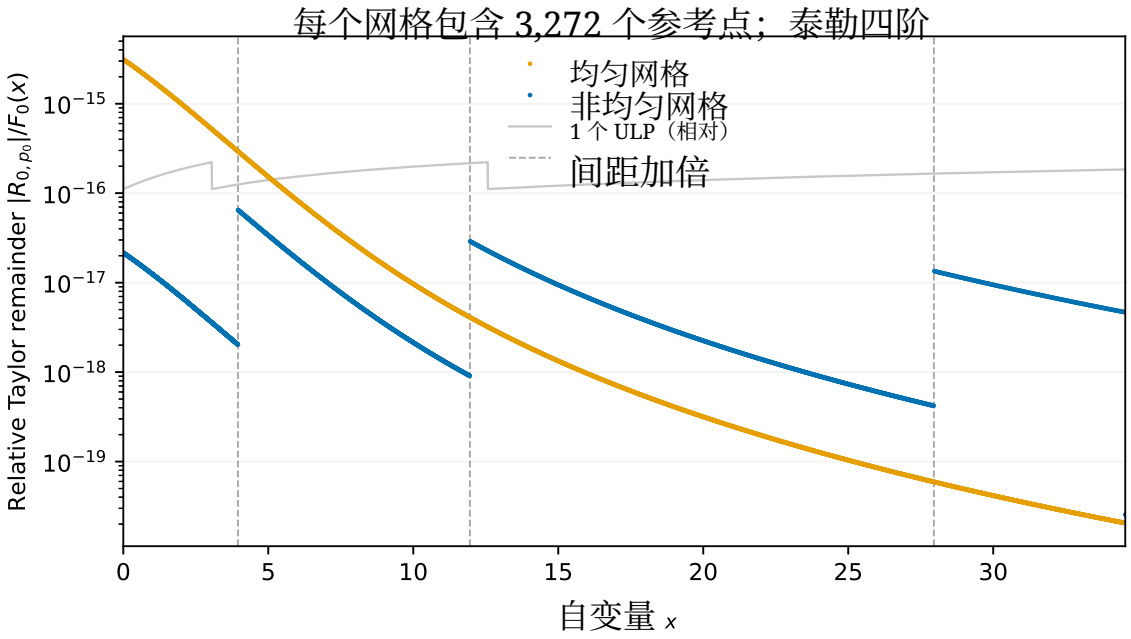

图 S1：主文匹配网格示意图的复现。灰色表示 $2^{\lfloor \log_2 F_0(x) \rfloor - 52}/F_0(x)$；虚线标记 $x + s = 8$、16、32 或 $x = 3.953125$、11。953125、27。953125。对数坐标轴上省略了精确为零的余项。这不是对求值器误差或性能的测量。

S3.2 向上递推通过窗口

正确舍入的 binary64 F_0 和 $E = e^{-x}$ 初始化该独立实验。它一次性生成 $t = \mathrm{RN}_{64}(x + x)$、$\rho = \mathrm{RN}_{64}(1/t)$ 和 $q = \mathrm{RN}_{64}(E\rho)$，然后求值

$$a_n = \mathrm{RN}_{64}((2n-1)\rho), \qquad \widehat{F}_n = \mathrm{FMA}_{64}(a_n, \widehat{F}_{n-1}, -q). \tag{S11}$$

指数函数可能正确地下溢为零。不使用近似倒数或渐近替代。参考区间采用 MPFR 生成的、宽度为 $1/16$ 的 12 阶泰勒展开，使用 binary128 求值，并针对舍入歧义或门控条件直接通过 4096 位 MPFR 进行精化。

输入为 $x_i = \mathrm{RN}_{64}(i/10^6)$、$i = 1$、……、$10^9$。对于 $N = 1$、……、12 以及 1、2、4、8 和 16 ULP 的容差，每个输出 1、……、N 都必须在一百万个连续输入上通过。窗口在 x 中跨度为 0.999999。包含 250000 个输入的分片保留了失败边界；连接通过前缀和后缀即可定位最早的完整窗口。未找到完整的单 ULP 窗口。缺失窗口仍在机器可读记录中明确列出。

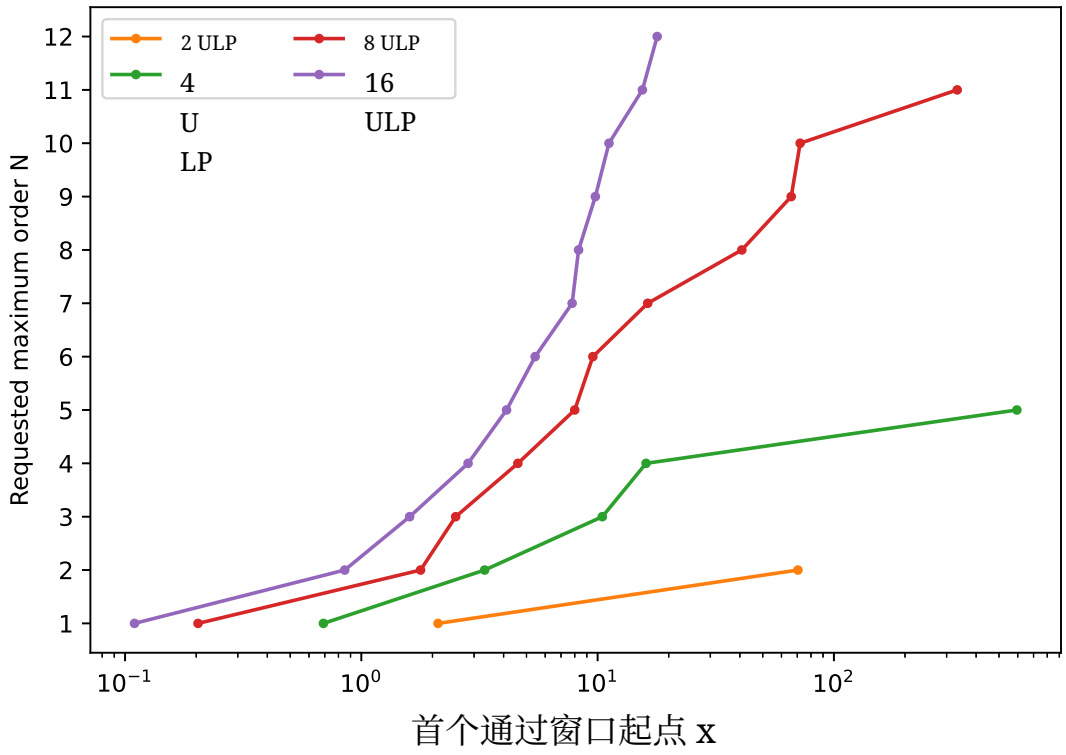

图 S2：重现正文中的向上递推示意图：首个完整通过窗口的起点，而非保证在所有更大的自变量处均能通过验收的边界。随附的诊断记录保留了完整的容差/阶数覆盖情况，包括缺失的窗口。

S4 独立函数级精度评估

对于每个请求范围 $[0, N]$、$N = 0$、……、12，已完成的扫描会在 $x_i = \mathrm{RN}_{64}(i/10^6)$、$i = 0$、……、$10^9$ 上评估所有输出。输入由整数索引和分母构成，而不是通过重复加法生成舍入步长。参考点会在这些实际的 binary64 输入上进行评估。此验证范围独立于构造范围 $N \le 24$。

在本补充材料中，间距定义为

$$\mathrm{ulp}(y) = \begin{cases} 2^{-1074}, & 0 \le y < 2^{-1022}, \\ 2^{\lfloor \log_2 y \rfloor - 52}, & y \ge 2^{-1022}, \end{cases} \tag{S12}$$

对于有限非负 y，精度测量使用 $u = \mathrm{ulp}(\mathrm{RN}_{64}(F_n(x)))$，但在减去计算值之前不会对参考值进行舍入。对于 $L > 0$ 和 $[L, U] \ni F_n(x)$，有限计算得到的 v 具有以下误差包围

$$a_- = \max(0, L - v, v - U), \qquad a_+ = \max(|v - L|, |v - U|), \tag{S13}$$

$$e_{\mathrm{abs}} \in [a_-, a_+], \qquad e_{\mathrm{rel}} \in [a_-/U, a_+/L], \qquad e_{\mathrm{ULP}} \in [a_-/u, a_+/u]. \tag{S14}$$

所有界均向外计算。扫描器将区间端点舍入为 binary64，并检查其二进制指数（ilogb）是否一致；若不一致，则直接使用 512 位参考值并重复检查。它要求下界为正且间距已确定，而不要求每个舍入有效数位都一致。因此，与分子参考后端不同，稠密参考区间不必证明唯一的 binary64 值。

扫描参考值使用 4000 个宽度为 $1/4$、位于 $[0, 1000]$，上且中心为中点 $r = a + 1/8$ 的单元。每个阶数存储 25 个系数区间，用于构造 24 阶泰勒多项式；这些区间通过定向的 256 位 MPFR 求值构造，并向外转换为 binary128。对于 $\delta = x - r$，余项的界为

$$|R_{n,24}(x)| \le F_{n+25}(a)\frac{|\delta|^{25}}{25!}, \tag{S15}$$

其中 a 是单元左端点。上界因子通过向外舍入单独制表。Binary128 区间霍纳求值会传播系数和算术不确定性；余项向两个方向分别加上。最终端点采用最后一个单元，而 $x = 0$ 使用针对 $1/(2n+1)$ 的定向界。此配置独立于用于说明递推的 12 阶参考。源代码和构建标识编入 S5.1.1 节索引。

保留的见证点会在 512 位精度下直接重新计算。每个分片的求和结果以 binary128 为界；合并时使用定向舍入的 80 位十进制算术。扫描包含 10 亿个输入的分片，以及一个仅含 1 个输入的末尾分片，并以最多 4096 个输入为一批进行处理。覆盖检查会拒绝缺失、重复或不兼容的分片。保留的最大值和总和的下界与上界分别提供；当参考区间重叠时，报告的最大上界不一定对应唯一的真实最大值点。

每个返回值的权重相同。合并所有请求范围时，每种后端和模式均计入 $91 \times 1\,000\,000\,001$ 个值。不同请求范围测试中重复出现的物理积分阶数不会去重。合并均值是总和除以计数得出的，而非 13 个范围均值的未加权平均值。完整的逐阶和合并记录均保留上下界、计数，以及精确的见证输入和计算结果。

CPU 严格适配器使用 -fno-fast-math 和 -ffp-contract=off；快速适配器使用 -ffast-math 和 -ffp-contract=fast。CUDA 严格适配器使用 --fmad=false，快速适配器使用 --use_fast_math。显式的融合乘加仍保持融合。收集器采用严格模式，不链接快速数学启动对象，并在适配器调用后检查舍入至偶数、已禁用的 FTZ/DAZ 和已屏蔽的陷阱。这些检查针对主机环境；CUDA 设备算术遵循其适配器的编译模式。混合布局试验会分别测试部分向量/块和混合域输入，与密集的相邻布局扫描相区分。这些适配器并非分子计时可执行文件。

中央处理器覆盖范围包括原始 Simint、FABO、原生 LibintX SIMD，以及 VSR 和 Laikov 的 Highway 移植版本。CUDA 覆盖范围包括原始 QUICK、FABO、LibintX、VSR 和 GANSU。原始 QUICK 和 Simint 例程直接纳入测试。LibintX 的中央处理器输出存储按其累加接口的要求进行初始化。随附的构建记录列明了确切的依赖版本修订、适配器命令和源代码哈希；每个库均在这些设置下进行评估，而不是以 FABO 的构造目标作为评估标准。

FABO 报告的采样 ULP 误差最大上界为 1.0669248898976357，出现在 $N = n = 3, x =$0x1.6b0ff5c6c11a1p+5，适用于两个平台和两种算术模式。该值略高于构造目标。报告的最大值相同，并不意味着其他所有输入都逐位相等。QUICK 在 $x = 0.100019$ 附近的 $N = 12$ 见证值，来自一项范围超出 $N \le 8$ 分子工作负载的函数测试。

表 S2：按实现和算术模式列出的采样 ULP 误差最大报告上界。精确见证样本及所有误差度量见 data/maximum-witnesses.csv。

平台	Mode	后端	N	n	x	ULP 上界
cpu	严格模式	FABO	3	3	45.382793	1.066925
cpu	严格模式	FABO（软件）	3	3	45.382793	1.066925
cpu	严格模式	Laikov	12	12	0.269971	329.2376
cpu	严格模式	LibintX	8	8	36.000000	7.804764e+07
cpu	严格模式	Simint	12	7	36.500000	1.185666e+10
cpu	严格模式	VSR	12	12	29.491339	1.298784e+12
cpu	fast	FABO	3	3	45.382793	1.066925
cpu	fast	FABO（软件）	3	3	45.382793	1.066925
cpu	fast	Laikov	12	12	0.269971	329.2376
cpu	fast	LibintX	8	8	36.000000	7.804764e+07
cpu	fast	Simint	12	7	36.500000	1.185666e+10
cpu	fast	VSR	12	12	29.491339	1.298784e+12
cuda	严格模式	FABO	3	3	45.382793	1.066925
cuda	严格模式	GANSU	12	12	29.491339	1.298784e+12
cuda	严格模式	LibintX	11	11	18.581440	39.33279
cuda	严格模式	QUICK	12	12	0.100019	3.407894e+22
cuda	严格模式	VSR	12	12	29.491339	1.298784e+12
cuda	fast	FABO	3	3	45.382793	1.066925
cuda	fast	GANSU	12	12	29.491339	1.298784e+12
cuda	fast	LibintX	11	11	18.583692	39.44849
cuda	fast	QUICK	12	12	0.100055	3.376171e+22
cuda	fast	VSR	12	12	29.491339	1.298784e+12

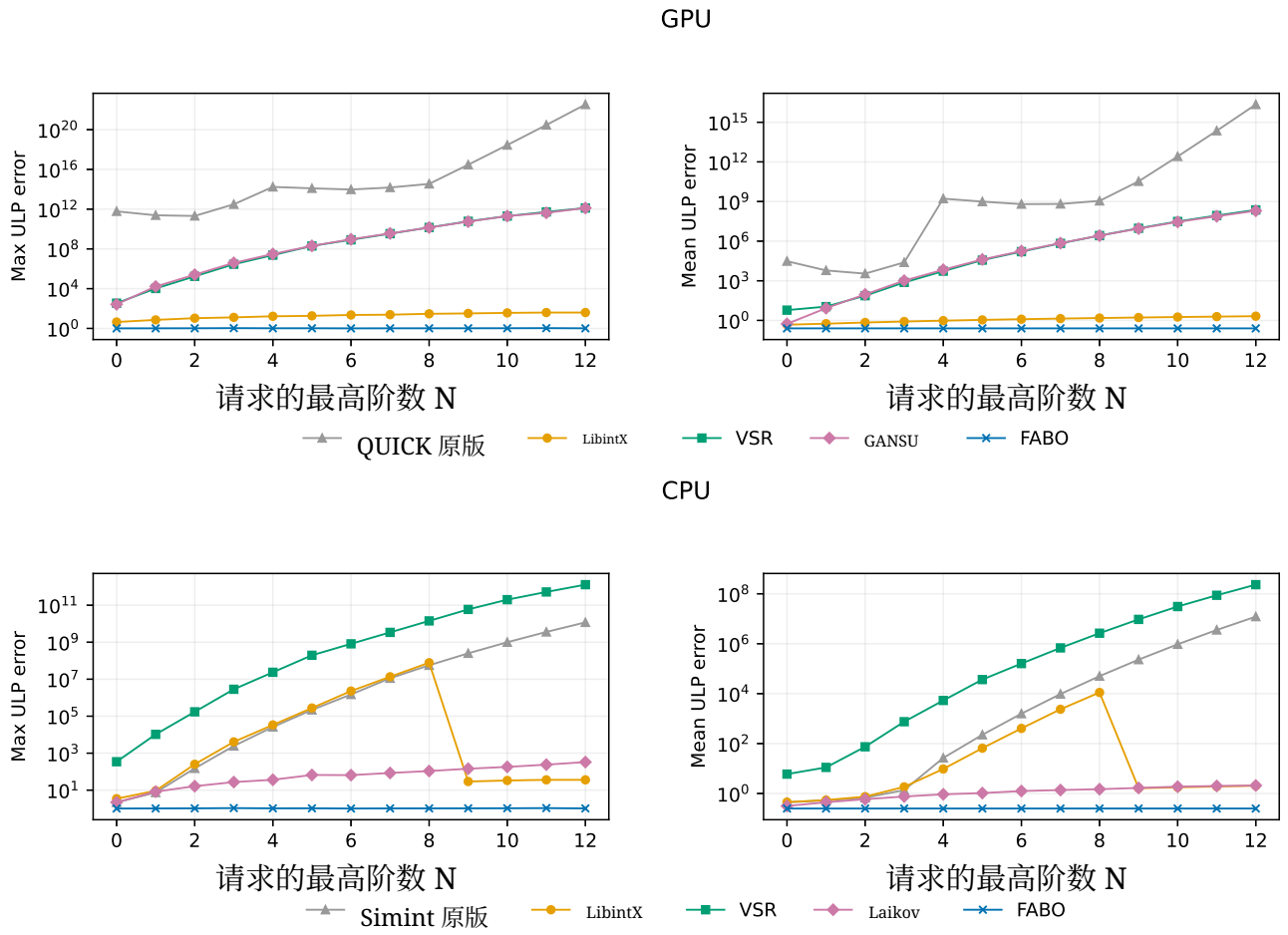

图 S3：已完成扫描所得的快速数学精度补充结果。数值为采样误差上界，并非连续域界。原生聚集和软件聚集是同一 FABO 算法的策略；完整的 CPU 记录包含这两种策略。

S5 应用程序集成与基准测试设置

S5.1 平台和编译

图形处理器测量使用 NVIDIA GeForce RTX 5090 D v2、NVCC 13.4.92，并以 sm_120 为目标，同时保留 QUICK 的 -O2 和 --use_fast_math 构建选项。中央处理器测量在 Intel Xeon Gold 6526Y 上使用一个线程，编译器为 GCC 14.4.0，编译选项为 -O3 -ffast-math -march=sapphirerapids，并启用 AVX-512。Intel MKL 使用一个线程和自动分派。这些应用标志与 S4 节中的独立验证模式不同；指令检查探测不定义应用构建设置。

QUICK 的双电子能量积分 Boys 调用点在分别编译的可执行文件中被替换。筛选、收缩、积分递推和累加均保持不变。普通的 s/p 和 s/p/d 路径分别选择 FABO $[0, 4]$ 和 $[0, 8]$ 模型，并且每个线程在其实际请求的端点处停止。ENABLEF 处于关闭状态。NWChem 的替换发生在 Simint 生成的积分核中；这些积分核的请求端点在编译时已知，并据此选择相应的模型。比较算法和应用已在其原始出版物中进行了描述。[1, 2, 3, 4, 5, 6]

中央处理器应用的编译标志中也包含 -fopenmp-simd；它会启用单指令多数据指令，但不会建立多线程执行。归档的链接方式使用顺序版 MKL。记录的运行时环境为 Intel oneAPI MKL 2026.1（构建版本 20260612），使用一个线程，并通过自动分派加载 libmkl_avx512.so.3；未使用 MKL_CBWR 强制采用较低级别的指令集。CUDA 验证在 Ryzen 9 7950X 主机上使用驱动程序 615.71.09 运行；

表 S3：记录的编译与执行差异。数据附有完整命令和已加载库标识。

工件	目标平台/编译器	算术策略
QUICK 计时	SM120 / NVCC 13.4.92	O2、CUDA 快速数学
NWChem 积分计时	AVX-512 / GCC 14.4.0	O3、快速数学
CPU 精度适配器	Sapphire Rapids / GCC	严格模式和快速模式分别
CUDA 精度适配器	SM120 / NVCC	关闭收缩 / 快速数学
精度收集器	主机 CPU	严格模式，舍入至偶数，关闭 FTZ/DAZ

主机 r 参考算术与设备适配器算术是该实验的不同组成部分 t.

S5.1.1 源代码与构建映射

QUICK 26.03 固定于修订版本 730af1533da8；NWChem 7.3.1 固定于 e2869a2c8144；Simint 生成器固定于 38b24bb040c6。CPU 对比记录使用 LibintX dd5bb8b9536c 和 Highway 9ca61efcab7f，以及来自快照 728ec72 的 VSR/Laikov 单指令多数据移植版本。这些是依赖项标识，并非为所有观测结果指定的单一修订版本。QUICK 链接其捆绑的 BLAS 和 LAPACK；NWChem 链接顺序 ILP64 MKL。完整的编译器命令、依赖项哈希值和已加载库记录保留在以下相对于软件包的路径中。

GPU GANSU 适配器使用 v2026.8.10 版本；VSR 系数及所提供的实现归属于作者的补充资料集，DOI 为 10.60893/figshare.jcp.c.8390929。其源代码溯源信息保存在 data/application-inputs/comparators/ 下。该适配器和系数头文件与已完成的 CUDA 精度构建中的哈希值一致；这些标识将它们与另行存档的 CPU SIMD 移植版本区分开来。

QUICK 计时二进制程序。 data/application-inputs/quick/variants/ 包含变体配方以及 FABO 的 build.json 和 compile_commands.json；data/records/benchmarks/quick-fixed-density-2 0261106/reuse.json 将最终运行映射到可执行文件、测试装置和参考标识。

NWChem 计时二进制程序。 data/application-inputs/nwchem/built.json 记录六个数值二进制程序（包括未计时的参考程序）、标志、编译器版本和链接方式；该目录中的 inputs.json 对其源输入进行哈希处理。data/records/benchmarks/nwchem-fixed-density-20261106/config.json 记录固定依赖项和执行设置。

函数精度适配器和收集器。 在 data/records/benchmarks/function-accuracy-202 61107/ 下，cpu/ 和 cuda/ 各自包含 built.json、run.json 和 scan.build.json。适配器命令及其哈希值与严格主机收集器的命令和参考表哈希分开提供。匹配的归档源文件位于 data/methods/function-accuracy/ 下。

测试装置、参考点和归约。 每个应用的 reuse.json 都标识了重复使用的初始状态和所接受的参考输出。QUICK molecular-means.manifest.json 和 NWChem matrix-absol ute.manifest.json 验证了后续归约。data/method-sources.csv 对本补充材料随附的参考代码和报告代码进行了索引。

所有最终的计时观测结果均为新近测量所得。此前经过验证的尺寸决策、初始状态和正确舍入的参考值，均按照这些清单中的标识予以重复使用。报告脚本随后对这些观测结果进行了归约；这些脚本的身份信息与应用程序构建和数值模型生成彼此分开保存。

S5.2 冻结的初始状态和工作负载

QUICK 26.03 每个测试案例仅准备一次默认的原子密度叠加猜测。在每次更新之前，都会重放已保存的 P_0、H_{core}、重叠矩阵和正交化矩阵。NWChem 准备原子猜测，重放轨道、占据数、重叠矩阵和正交化矩阵，并验证重建的输入密度和核心哈密顿量哈希值。初始化和测试固件加载不计时。每次运行都会构建 $F[P_0]$、进行对角化并构造 P_1；这不是收敛轨迹。保留正常的积分筛选，且各后端接收的筛选输入保持不变。

工作负载使用平衡 S66 体系 1、17、24 和 34：水、尿嘧啶、苯和戊烷二聚体。[7] 基组为 6-31G*、def2-SVP 和 cc-pVDZ。[8, 9, 10, 11] 两个应用都执行中性、闭壳层单重态 RHF 能量计算，不使用密度拟合。QUICK 使用笛卡尔基组函数和直接图形处理器积分，不进行分子对称性约化。NWChem 使用球形分段基组、直接积分和显式C_1 对称性（noautosym noautoz）；其已保存的输入会禁用备选的 Texas 和原生积分路径，并要求使用 Simint 进行复制式 Fock 构造。经过修补的单次更新入口会拒绝降秩 NWChem 状态。下方列出了可接受的维度；这些维度不是根据基组名称推断得出的。

QUICK 输入文件指定的密度 RMS 判据为 10^{-6}，但固定状态钩子会绕过收敛循环。在固定的源码版本中，这些输入文件将双电子截断设为 10^{-7}，原始积分截断设为 10^{-8}，密度截断设为 10^{-10}；既未选择粗略积分选项，也未选择严格积分选项。这些值由 scf_ 算子上传。单电子预筛选截断为 10^{-12}，重叠截断为 10^{-5}。提供的源码摘录保留了相关的初始化和关键字解析。NWChem 保留其依赖密度的筛选，对于这些满秩输入，tol2e$= 10^{-7}$，dentolmax $= 10^{-6}$。其复现构建会针对非零的最大壳层块 RMS 密度 D，将仅积分阈值计算为 min(10^{-6}, $10^{-7}/D$, $10^{-7}/D^2$)，并在 $D = 0$ 时取10^{-7}。因此，筛选取决于固定密度，而非仅取决于某个统一阈值。

副本沿包围盒轴中最短的轴平移，平移距离为该轴上的二聚体尺度加 3.4 埃。内部几何结构保持不变。工作负载大小调整使用原始后端和 def2-SVP，并选择第一个达到两秒福克构建时间的已配置副本数量。冻结的图形处理器副本数量为 128、8、16 和 16；中央处理器数量为 8、1、1 和 1。所得几何结构在三个基组和后端之间复用，每个基组分别使用独立的初始状态装置。此大小调整并非 Boys 参数调优。

表 S4：可接受的工作负载维度。QUICK 使用笛卡尔函数；NWChem 使用球函数。在这些情况下，轨道秩等于基组维度。

平台	分子	基组	AO 维度	电子数
GPU	128× 水二聚体	6-31G*	4864	2560
GPU	128× 水二聚体	def2-SVP	6400	2560
GPU	128× 水二聚体	cc-pVDZ	6400	2560
GPU	8×尿嘧啶二聚体	6-31G*	2048	928
GPU	8×尿嘧啶二聚体	def2-SVP	2240	928
GPU	8×尿嘧啶二聚体	cc-pVDZ	2240	928
GPU	16×苯二聚体	6-31G*	3264	1344
GPU	16×苯二聚体	def2-SVP	3840	1344
GPU	16×苯二聚体	cc-pVDZ	3840	1344
GPU	16×戊烷二聚体	6-31G*	3168	1344
GPU	16×戊烷二聚体	def2-SVP	4320	1344
GPU	16×戊烷二聚体	cc-pVDZ	4320	1344
CPU	8× 水二聚体	6-31G*	288	160
CPU	8× 水二聚体	def2-SVP	384	160
CPU	8× 水二聚体	cc-pVDZ	384	160
CPU	1×尿嘧啶二聚体	6-31G*	240	116
CPU	1×尿嘧啶二聚体	def2-SVP	264	116
CPU	1×尿嘧啶二聚体	cc-pVDZ	264	116
CPU	1×苯二聚体	6-31G*	192	84
CPU	1×苯二聚体	def2-SVP	228	84
CPU	1×苯二聚体	cc-pVDZ	228	84
CPU	1×戊烷二聚体	6-31G*	188	84
CPU	1×戊烷二聚体	def2-SVP	260	84
CPU	1×戊烷二聚体	cc-pVDZ	260	84

data/workloads.csv 将每个案例映射到其几何结构、基组和维度；data/application-inputs/ 提供以埃为单位的 XYZ文件、保存的输入文件、基组记录和尺寸决策。每条通过验收的观测结果均保留其测试装置和重放哈希。输入文件中设定的 SCF迭代上限不会覆盖已修补的单次更新协议。

S5.3 计时边界

QUICK 福克计时在 scf_operator(.false.) 外围使用 clock_gettime(CLOCK_MONOTONIC)，并在每次取时间戳之前执行 CUDA同步。计时涵盖依赖密度的筛选准备、相关数据传输、图形处理器上的双电子计算和福克矩阵组装；单电子矩阵则预先准备。NWChem 的 jk_seconds 使用 util_wallsec() 对 fock_2e 计时，仅测量双电子库仑/交换构建。这两种计时范围不能视为相同。Boys函数表的分配、初始化和主机到设备上传均在计时开始前完成。矩阵序列化和能量诊断在计时后进行。完整更新计时会归档，但不会代替上述任何一种福克度量。

S6 单次计时和加速比统计量

每个应用包含 12 个案例、5 个计时后端和 5 次新进程重复，因此共有 300 个观测值和 60 个案例/后端组。不进行分子预热。后端顺序在各次重复中循环轮换。随附的原始计时CSV保留重复索引；原始记录和执行溯源则分别保留。参考后端运行时间从不作为性能观测值纳入。

对于计时值 t_1、……、t_5，报告的平均值为 $\bar{t} = \sum_j t_j/5$，样本标准差为 $[\sum_j (t_j - \bar{t})^2/4]^{1/2}$。二者均根据各次单独观测值计算。表中数值以秒为单位，保留三位小数；机器可读文件保留完整精度。

对于固定的比较基准 b 和案例 c，定义 $S_{b,c} = \bar{t}_{b,c}/\bar{t}_{\mathrm{FABO},c}$。总体比率为

$$G_b = \exp\left[\frac{1}{12}\sum_c \log S_{b,c}\right]. \tag{S16}$$

百分比加速为 100($G_b - 1$)，而非百分比时间缩减 100($1 - 1/G_b$)。所有情况均使用同一个比较基准；并非针对每种情况分别选取最快的替代方案。重新计算后的比率得出正文中的 1.22–10.67 百分比范围。

表 S5：12 种情况下比较基准/FABO 平均时间比的几何平均

平台	比较基准	比率	加速比（百分比）
GPU	GANSU	1.080227	8.02
GPU	LibintX	1.016422	1.64
GPU	QUICK	1.106705	10.67
GPU	VSR	1.061879	6.19
CPU	Laikov	1.030355	3.04
CPU	LibintX	1.013004	1.30
CPU	Simint	1.103530	10.35
CPU	VSR	1.012189	1.22

表 S6：图形处理器计时（单位：秒）；五次观测的算术平均值和样本标准差。

分子	基组	后端	平均值（秒）	标准差（秒）
128× 水二聚体	6-31G*	QUICK	2.347	0.025
128× 水二聚体	6-31G*	LibintX	2.122	0.012
128× 水二聚体	6-31G*	VSR	2.221	0.004
128× 水二聚体	6-31G*	GANSU	2.237	0.008
128× 水二聚体	6-31G*	FABO	2.103	0.012

接下页

分子	基组	后端	平均值（秒）	标准差（秒）
128× 水二聚体	def2-SVP	QUICK	2.553	0.018
128× 水二聚体	def2-SVP	LibintX	2.410	0.016
128× 水二聚体	def2-SVP	VSR	2.463	0.020
128× 水二聚体	def2-SVP	GANSU	2.490	0.024
128× 水二聚体	def2-SVP	FABO	2.408	0.017
128× 水二聚体	cc-pVDZ	QUICK	3.703	0.016
128× 水二聚体	cc-pVDZ	LibintX	3.350	0.022
128× 水二聚体	cc-pVDZ	VSR	3.462	0.014
128× 水二聚体	cc-pVDZ	GANSU	3.530	0.014
128× 水二聚体	cc-pVDZ	FABO	3.297	0.019
8×尿嘧啶二聚体	6-31G*	QUICK	3.674	0.005
8×尿嘧啶二聚体	6-31G*	LibintX	3.439	0.001
8×尿嘧啶二聚体	6-31G*	VSR	3.624	0.003
8×尿嘧啶二聚体	6-31G*	GANSU	3.669	0.005
8×尿嘧啶二聚体	6-31G*	FABO	3.348	0.004
8×尿嘧啶二聚体	def2-SVP	QUICK	2.237	0.002
8×尿嘧啶二聚体	def2-SVP	LibintX	2.128	0.003
8×尿嘧啶二聚体	def2-SVP	VSR	2.212	0.002
8×尿嘧啶二聚体	def2-SVP	GANSU	2.251	0.001
8×尿嘧啶二聚体	def2-SVP	FABO	2.091	0.004
8×尿嘧啶二聚体	cc-pVDZ	QUICK	3.818	0.003
8×尿嘧啶二聚体	cc-pVDZ	LibintX	3.473	0.004
8×尿嘧啶二聚体	cc-pVDZ	VSR	3.667	0.002
8×尿嘧啶二聚体	cc-pVDZ	GANSU	3.760	0.002
8×尿嘧啶二聚体	cc-pVDZ	FABO	3.370	0.004
16×苯二聚体	6-31G*	QUICK	5.918	0.007
16×苯二聚体	6-31G*	LibintX	5.380	0.004
16×苯二聚体	6-31G*	VSR	5.701	0.003
16×苯二聚体	6-31G*	GANSU	5.765	0.003
16×苯二聚体	6-31G*	FABO	5.290	0.004
16×苯二聚体	def2-SVP	QUICK	4.540	0.005
16×苯二聚体	def2-SVP	LibintX	4.264	0.006
16×苯二聚体	def2-SVP	VSR	4.424	0.009
16×苯二聚体	def2-SVP	GANSU	4.473	0.007
16×苯二聚体	def2-SVP	FABO	4.234	0.010
16×苯二聚体	cc-pVDZ	QUICK	7.955	0.012
16×苯二聚体	cc-pVDZ	LibintX	7.113	0.009
16×苯二聚体	cc-pVDZ	VSR	7.503	0.004
16×苯二聚体	cc-pVDZ	GANSU	7.665	0.006
16×苯二聚体	cc-pVDZ	FABO	6.984	0.009
16×戊烷二聚体	6-31G*	QUICK	5.192	0.004
16×戊烷二聚体	6-31G*	LibintX	4.668	0.003
16×戊烷二聚体	6-31G*	VSR	4.916	0.004
16×戊烷二聚体	6-31G*	GANSU	5.037	0.002
16×戊烷二聚体	6-31G*	FABO	4.567	0.005
16×戊烷二聚体	def2-SVP	QUICK	6.192	0.005
16×戊烷二聚体	def2-SVP	LibintX	5.765	0.007
16×戊烷二聚体	def2-SVP	VSR	5.977	0.007
16×戊烷二聚体	def2-SVP	GANSU	6.082	0.006
16×戊烷二聚体	def2-SVP	FABO	5.699	0.007

续见下页

分子	基组	后端	平均值（秒）	标准差（秒）
16×戊烷二聚体	cc-pVDZ	QUICK	8.823	0.008
16×戊烷二聚体	cc-pVDZ	LibintX	7.947	0.006
16×戊烷二聚体	cc-pVDZ	VSR	8.298	0.010
16×戊烷二聚体	cc-pVDZ	GANSU	8.535	0.009
16×戊烷二聚体	cc-pVDZ	FABO	7.800	0.008

表 S7：CPU 计时（秒）；五次观测的算术平均值和样本标准差。

分子	基组	后端	平均值（秒）	标准差（秒）
8× 水二聚体	6-31G*	Simint	1.466	0.003
8× 水二聚体	6-31G*	LibintX	1.405	0.003
8× 水二聚体	6-31G*	VSR	1.430	0.013
8× 水二聚体	6-31G*	Laikov	1.424	0.002
8× 水二聚体	6-31G*	FABO	1.429	0.029
8× 水二聚体	def2-SVP	Simint	2.765	0.022
8× 水二聚体	def2-SVP	LibintX	2.649	0.001
8× 水二聚体	def2-SVP	VSR	2.687	0.002
8× 水二聚体	def2-SVP	Laikov	2.680	0.002
8× 水二聚体	def2-SVP	FABO	2.656	0.005
8× 水二聚体	cc-pVDZ	Simint	3.389	0.008
8× 水二聚体	cc-pVDZ	LibintX	3.232	0.009
8× 水二聚体	cc-pVDZ	VSR	3.281	0.006
8× 水二聚体	cc-pVDZ	Laikov	3.269	0.002
8× 水二聚体	cc-pVDZ	FABO	3.283	0.001
1×尿嘧啶二聚体	6-31G*	Simint	6.162	0.011
1×尿嘧啶二聚体	6-31G*	LibintX	5.770	0.024
1×尿嘧啶二聚体	6-31G*	VSR	5.728	0.004
1×尿嘧啶二聚体	6-31G*	Laikov	5.827	0.004
1×尿嘧啶二聚体	6-31G*	FABO	5.717	0.046
1×尿嘧啶二聚体	def2-SVP	Simint	7.075	0.008
1×尿嘧啶二聚体	def2-SVP	LibintX	6.635	0.002
1×尿嘧啶二聚体	def2-SVP	VSR	6.594	0.010
1×尿嘧啶二聚体	def2-SVP	Laikov	6.700	0.003
1×尿嘧啶二聚体	def2-SVP	FABO	6.543	0.006
1×尿嘧啶二聚体	cc-pVDZ	Simint	10.581	0.094
1×尿嘧啶二聚体	cc-pVDZ	LibintX	9.752	0.059
1×尿嘧啶二聚体	cc-pVDZ	VSR	9.718	0.005
1×尿嘧啶二聚体	cc-pVDZ	Laikov	9.885	0.007
1×尿嘧啶二聚体	cc-pVDZ	FABO	9.693	0.010
1×苯二聚体	6-31G*	Simint	5.496	0.003
1×苯二聚体	6-31G*	LibintX	4.957	0.006
1×苯二聚体	6-31G*	VSR	4.923	0.003
1×苯二聚体	6-31G*	Laikov	5.038	0.004
1×苯二聚体	6-31G*	FABO	4.825	0.003
1×苯二聚体	def2-SVP	Simint	7.830	0.007
1×苯二聚体	def2-SVP	LibintX	7.122	0.022
1×苯二聚体	def2-SVP	VSR	7.066	0.050
1×苯二聚体	def2-SVP	Laikov	7.210	0.018
1×苯二聚体	def2-SVP	FABO	6.918	0.003

续于下一页

分子	基组	后端	平均值（秒）	标准差（秒）
1×苯二聚体	cc-pVDZ	Simint	11.378	0.005
1×苯二聚体	cc-pVDZ	LibintX	10.131	0.014
1×苯二聚体	cc-pVDZ	VSR	10.114	0.025
1×苯二聚体	cc-pVDZ	Laikov	10.420	0.092
1×苯二聚体	cc-pVDZ	FABO	9.945	0.025
1×戊烷二聚体	6-31G*	Simint	5.011	0.003
1×戊烷二聚体	6-31G*	LibintX	4.436	0.032
1×戊烷二聚体	6-31G*	VSR	4.396	0.005
1×戊烷二聚体	6-31G*	Laikov	4.552	0.002
1×戊烷二聚体	6-31G*	FABO	4.296	0.006
1×戊烷二聚体	def2-SVP	Simint	10.452	0.004
1×戊烷二聚体	def2-SVP	LibintX	9.344	0.009
1×戊烷二聚体	def2-SVP	VSR	9.257	0.010
1×戊烷二聚体	def2-SVP	Laikov	9.564	0.027
1×戊烷二聚体	def2-SVP	FABO	9.079	0.029
1×戊烷二聚体	cc-pVDZ	Simint	13.819	0.181
1×戊烷二聚体	cc-pVDZ	LibintX	12.072	0.021
1×戊烷二聚体	cc-pVDZ	VSR	12.021	0.051
1×戊烷二聚体	cc-pVDZ	Laikov	12.423	0.028
1×戊烷二聚体	cc-pVDZ	FABO	11.776	0.023

S7 分子结果比较及配套数据

S7.1 正确舍入的 Boys 后端

参考实现改变的是 Boys 求值，而非后续的应用算术。QUICK 使用 binary128 对 MPFR 生成的泰勒区间进行求值，只有当两个端点舍入为相同的 binary64 位模式时才接受该值。对于有歧义的调用，会记录下来以便通过自适应定向 MPFR 精化处理；同一测试装置会在全新进程中重放。未解决的尝试不计入五次已接受的参考更新。队列溢出、无效输入和重放次数耗尽都会导致失败，而不会提供猜测值。参考计时不计入。

NWChem 预言器使用共享的 MPFR 生成的表和 113 位 MPFR 算术进行区间求值，并在需要时通过定向精化将精度提升至 8192 位。通过两次未经计时的新进程参考更新检查可复现性。预言器会临时设置所需的 MXCSR 控制，并恢复调用方的完整寄存器状态。Boys 值的正确舍入并不能使积分递推、矩阵累加或特征值求解达到精确。

S7.2 诊断信息和重复输出削减

使用自旋求和输入密度，能量诊断为

$$E[P_0] = E_{\mathrm{nuc}} + \frac{1}{2}\sum_{\mu\nu}(P_0)_{\mu\nu}\big(H_{\mathrm{core}} + F[P_0]\big)_{\mu\nu}. \tag{S17}$$

能量和福克差值的单位为哈特里；密度条目采用应用的无量纲原子轨道表示。该能量与 P_0 及其 Fock矩阵相关，而不是与更新后的 P_1相关。

对于 QUICK，首先分别对每个后端接受的五个能量，以及其五个福克矩阵和更新密度矩阵的每个元素取平均。参考输出则单独取平均。随后计算这些均值之间的差值。以观测值为中心进行求和，并保留均值的高位/低位部分，可以保留亚 ULP 偏移；这既不是绝对误差的平均值，也不是根据平均后的福克矩阵重建的密度。图形处理器的累加顺序可能在不同运行之间发生变化，因此，即使 Boys 调用采用正确舍入，参考离散度仍然具有意义。

对于 NWChem，会检查所有重复运行的福克矩阵、输入/输出密度、核心哈密顿量及能量的字节是否完全一致。矩阵归约使用经验证的首次输出。此举仅能确保在保存的二进制文件/节点/环境下具有可重复性，并不能保证在任意系统上都可重复。

对于任一矩阵 M，统计量为 $\max_{\mu\nu}|\Delta M_{\mu\nu}|$ 和 $d^{-2}\sum_{\mu\nu}|\Delta M_{\mu\nu}|$，其中 d 是其维度。所有 d^2条目均纳入计算，包括零值和对称两半。本文不报告相对矩阵误差。正文摘要中的每一列均独立取各案例统计量的最大值，因此达到最大值的案例未必相同。参考运行样本标准差衡量的是离散程度，而非参考均值的严格误差。

表 S8：图形处理器与正确舍入的 Boys 后端之间的逐案例差异。能量和 Fock 量的单位为哈特里；密度条目为无量纲。矩阵平均值包括每个元素。

分子	基组	后端	$\lvert\Delta E\rvert$	最大值 $\lvert\Delta F\rvert$	平均值$\lvert\Delta F\rvert$	最大值 $\lvert\Delta P\rvert$	平均值$\lvert\Delta P\rvert$
128× 水二聚体	6-31G*	QUICK	3.07e-07	4.92e-07	4.89e-11	3.31e-05	4.76e-10
128× 水二聚体	6-31G*	LibintX	1.16e-11	1.94e-13	2.75e-17	7.36e-10	1.93e-15
128× 水二聚体	6-31G*	VSR	2.44e-10	1.65e-12	1.62e-16	3.32e-10	2.88e-15
128× 水二聚体	6-31G*	GANSU	1.69e-10	1.66e-12	1.52e-16	4.63e-10	2.29e-15
128× 水二聚体	6-31G*	FABO	1.46e-11	2.83e-13	2.75e-17	7.11e-10	1.89e-15
128× 水二聚体	def2-SVP	QUICK	2.78e-05	3.60e-07	3.01e-11	1.55e-04	1.16e-09
128× 水二聚体	def2-SVP	LibintX	0.00e+00	1.78e-13	1.92e-17	1.44e-10	8.23e-16
128× 水二聚体	def2-SVP	VSR	5.82e-11	8.92e-13	6.14e-17	1.47e-10	1.59e-15
128× 水二聚体	def2-SVP	GANSU	6.69e-11	1.06e-12	6.04e-17	3.69e-10	1.84e-15
128× 水二聚体	def2-SVP	FABO	0.00e+00	1.95e-13	1.92e-17	1.49e-10	8.61e-16
128× 水二聚体	cc-pVDZ	QUICK	4.31e-07	7.98e-07	3.50e-11	7.93e-06	2.25e-10
128× 水二聚体	cc-pVDZ	LibintX	1.46e-12	1.54e-13	2.00e-17	8.16e-11	6.86e-16
128× 水二聚体	cc-pVDZ	VSR	9.90e-11	1.25e-12	6.42e-17	1.24e-10	1.75e-15
128× 水二聚体	cc-pVDZ	GANSU	6.84e-11	1.17e-12	5.68e-17	1.07e-10	1.53e-15
128× 水二聚体	cc-pVDZ	FABO	5.82e-12	1.50e-13	2.01e-17	5.60e-11	6.60e-16
8×尿嘧啶二聚体	6-31G*	QUICK	2.35e-07	7.00e-07	9.42e-11	6.20e-06	8.98e-09
8×尿嘧啶二聚体	6-31G*	LibintX	1.75e-11	5.80e-13	2.58e-16	2.74e-11	1.52e-14
8×尿嘧啶二聚体	6-31G*	VSR	1.46e-10	2.27e-11	3.97e-15	5.20e-11	4.66e-14
8×尿嘧啶二聚体	6-31G*	GANSU	1.60e-10	2.27e-11	4.75e-15	4.25e-11	5.10e-14
8×尿嘧啶二聚体	6-31G*	FABO	2.18e-11	5.66e-13	2.58e-16	2.46e-11	1.50e-14
8×尿嘧啶二聚体	def2-SVP	QUICK	5.98e-05	1.78e-06	1.30e-10	9.72e-05	8.00e-08
8×尿嘧啶二聚体	def2-SVP	LibintX	1.46e-12	5.63e-13	1.79e-16	4.71e-11	1.83e-14
8×尿嘧啶二聚体	def2-SVP	VSR	1.41e-10	1.41e-11	2.55e-15	8.32e-11	5.51e-14
8×尿嘧啶二聚体	def2-SVP	GANSU	1.34e-10	1.41e-11	3.02e-15	1.54e-10	6.87e-14
8×尿嘧啶二聚体	def2-SVP	FABO	7.28e-12	5.74e-13	1.78e-16	4.76e-11	1.73e-14
8×尿嘧啶二聚体	cc-pVDZ	QUICK	4.81e-05	1.32e-06	1.07e-10	5.62e-05	3.14e-08
8×尿嘧啶二聚体	cc-pVDZ	LibintX	4.37e-12	4.52e-13	2.13e-16	2.53e-11	1.15e-14
8×尿嘧啶二聚体	cc-pVDZ	VSR	1.21e-10	1.18e-11	3.02e-15	3.64e-11	4.25e-14
8×尿嘧啶二聚体	cc-pVDZ	GANSU	1.34e-10	1.18e-11	3.47e-15	4.22e-11	4.39e-14
8×尿嘧啶二聚体	cc-pVDZ	FABO	2.91e-12	4.46e-13	2.12e-16	2.39e-11	1.12e-14
16×苯二聚体	6-31G*	QUICK	4.75e-07	4.84e-07	1.02e-10	5.64e-06	4.04e-09
16×苯二聚体	6-31G*	LibintX	1.16e-11	2.54e-13	1.17e-16	2.35e-11	8.13e-15
16×苯二聚体	6-31G*	VSR	6.98e-11	1.22e-11	1.50e-15	3.01e-11	1.54e-14
16×苯二聚体	6-31G*	GANSU	6.98e-11	1.21e-11	1.61e-15	2.73e-11	1.39e-14
16×苯二聚体	6-31G*	FABO	1.16e-11	2.98e-13	1.16e-16	1.96e-11	7.96e-15
16×苯二聚体	def2-SVP	QUICK	9.75e-06	2.10e-07	6.20e-11	2.30e-05	9.76e-09
16×苯二聚体	def2-SVP	LibintX	1.89e-11	3.95e-13	9.56e-17	8.14e-11	1.79e-14
16×苯二聚体	def2-SVP	VSR	8.88e-11	1.37e-11	1.18e-15	1.06e-10	2.55e-14
16×苯二聚体	def2-SVP	GANSU	1.46e-10	1.36e-11	1.40e-15	8.65e-11	2.46e-14
16×苯二聚体	def2-SVP	FABO	1.60e-11	4.02e-13	9.51e-17	8.08e-11	1.79e-14

续见下页

分子	基组	后端	$\lvert\Delta E\rvert$	最大值 $\lvert\Delta F\rvert$	平均值$\lvert\Delta F\rvert$	最大值 $\lvert\Delta P\rvert$	平均值$\lvert\Delta P\rvert$
16×苯二聚体	cc-pVDZ	QUICK	3.11e-07	7.01e-07	5.42e-11	6.18e-06	4.16e-09
16×苯二聚体	cc-pVDZ	LibintX	1.46e-12	3.31e-13	1.05e-16	4.07e-11	1.05e-14
16×苯二聚体	cc-pVDZ	VSR	9.31e-11	5.67e-12	1.25e-15	6.86e-11	2.73e-14
16×苯二聚体	cc-pVDZ	GANSU	6.69e-11	1.20e-11	1.49e-15	4.39e-11	1.78e-14
16×苯二聚体	cc-pVDZ	FABO	1.46e-12	3.21e-13	1.06e-16	4.11e-11	1.02e-14
16×戊烷二聚体	6-31G*	QUICK	6.22e-07	4.86e-07	9.88e-11	4.84e-06	1.80e-09
16×戊烷二聚体	6-31G*	LibintX	4.37e-12	2.84e-13	1.19e-16	1.58e-11	2.15e-15
16×戊烷二聚体	6-31G*	VSR	6.40e-11	8.02e-12	8.72e-16	1.94e-11	6.28e-15
16×戊烷二聚体	6-31G*	GANSU	5.97e-11	1.31e-11	9.89e-16	1.76e-11	5.91e-15
16×戊烷二聚体	6-31G*	FABO	5.82e-12	2.77e-13	1.18e-16	1.53e-11	2.10e-15
16×戊烷二聚体	def2-SVP	QUICK	3.92e-06	1.53e-07	4.49e-11	1.01e-05	1.88e-09
16×戊烷二聚体	def2-SVP	LibintX	1.46e-12	3.25e-13	7.76e-17	2.66e-11	3.16e-15
16×戊烷二聚体	def2-SVP	VSR	9.17e-11	1.87e-12	4.53e-16	4.67e-11	9.19e-15
16×戊烷二聚体	def2-SVP	GANSU	1.03e-10	2.10e-12	5.11e-16	6.44e-11	1.25e-14
16×戊烷二聚体	def2-SVP	FABO	1.16e-11	3.14e-13	7.73e-17	2.54e-11	3.10e-15
16×戊烷二聚体	cc-pVDZ	QUICK	4.24e-07	7.08e-07	6.31e-11	6.17e-06	1.89e-09
16×戊烷二聚体	cc-pVDZ	LibintX	2.47e-11	2.96e-13	8.80e-17	9.69e-12	2.00e-15
16×戊烷二聚体	cc-pVDZ	VSR	6.26e-11	8.33e-12	7.09e-16	1.92e-11	5.56e-15
16×戊烷二聚体	cc-pVDZ	GANSU	9.46e-11	1.12e-11	8.26e-16	1.32e-11	6.15e-15
16×戊烷二聚体	cc-pVDZ	FABO	1.31e-11	2.77e-13	8.79e-17	8.46e-12	1.97e-15

表 S9：中央处理器各案例与正确舍入的 Boys 后端之间的差异。能量和福克量以哈特里为单位；密度条目为无量纲。矩阵平均值包含每个元素。

分子	基组	后端	$\lvert\Delta E\rvert$	最大值 $\lvert\Delta F\rvert$	平均值$\lvert\Delta F\rvert$	最大值 $\lvert\Delta P\rvert$	平均值$\lvert\Delta P\rvert$
8× 水二聚体	6-31G*	Simint	5.00e-12	1.22e-11	1.21e-14	4.92e-13	4.22e-15
8× 水二聚体	6-31G*	FABO	0.00e+00	3.55e-15	6.33e-18	3.28e-14	4.92e-16
8× 水二聚体	6-31G*	LibintX	0.00e+00	7.11e-15	1.04e-17	3.42e-14	4.83e-16
8× 水二聚体	6-31G*	VSR	1.36e-11	1.86e-12	3.04e-15	2.40e-12	1.06e-14
8× 水二聚体	6-31G*	Laikov	0.00e+00	7.11e-15	1.07e-17	3.86e-14	5.04e-16
8× 水二聚体	def2-SVP	Simint	1.36e-12	1.97e-12	2.13e-15	2.29e-13	9.47e-16
8× 水二聚体	def2-SVP	FABO	0.00e+00	3.55e-15	6.95e-18	2.72e-14	2.39e-16
8× 水二聚体	def2-SVP	LibintX	4.55e-13	1.07e-14	1.02e-17	4.35e-14	2.41e-16
8× 水二聚体	def2-SVP	VSR	5.91e-12	8.60e-13	1.05e-15	9.41e-13	3.51e-15
8× 水二聚体	def2-SVP	Laikov	4.55e-13	7.11e-15	1.18e-17	4.94e-14	2.41e-16
8× 水二聚体	cc-pVDZ	Simint	9.09e-13	1.18e-12	1.07e-15	2.62e-13	1.00e-15
8× 水二聚体	cc-pVDZ	FABO	0.00e+00	7.11e-15	7.21e-18	2.64e-14	2.21e-16
8× 水二聚体	cc-pVDZ	LibintX	0.00e+00	7.11e-15	1.12e-17	3.25e-14	2.31e-16
8× 水二聚体	cc-pVDZ	VSR	5.46e-12	9.08e-13	9.90e-16	1.30e-12	3.92e-15
8× 水二聚体	cc-pVDZ	Laikov	0.00e+00	7.11e-15	1.16e-17	3.04e-14	2.27e-16
1×尿嘧啶二聚体	6-31G*	Simint	9.78e-12	4.77e-11	4.91e-14	8.82e-12	7.58e-14
1×尿嘧啶二聚体	6-31G*	FABO	2.27e-13	1.07e-14	3.42e-17	3.94e-13	2.80e-15
1×尿嘧啶二聚体	6-31G*	LibintX	2.27e-13	1.07e-14	5.03e-17	4.13e-13	2.55e-15
1×尿嘧啶二聚体	6-31G*	VSR	6.37e-12	9.05e-12	1.39e-14	1.35e-11	1.07e-13
1×尿嘧啶二聚体	6-31G*	Laikov	0.00e+00	2.13e-14	6.28e-17	3.36e-13	2.82e-15
1×尿嘧啶二聚体	def2-SVP	Simint	8.19e-12	3.50e-11	3.37e-14	1.07e-11	5.49e-14
1×尿嘧啶二聚体	def2-SVP	FABO	2.27e-13	1.42e-14	2.85e-17	1.22e-12	2.88e-15
1×尿嘧啶二聚体	def2-SVP	LibintX	4.55e-13	1.07e-14	4.51e-17	8.24e-13	3.18e-15
1×尿嘧啶二聚体	def2-SVP	VSR	6.82e-12	1.41e-11	9.01e-15	1.09e-11	1.07e-13
1×尿嘧啶二聚体	def2-SVP	Laikov	2.27e-13	1.42e-14	5.57e-17	8.13e-13	3.56e-15

下页续

分子	基组	后端	$\lvert\Delta E\rvert$	最大值	$\lvert\Delta F\rvert$ 平均值	$\lvert\Delta F\rvert$ 最大值	$\lvert\Delta P\rvert$ 平均值	$\lvert\Delta P\rvert$
1×尿嘧啶二聚体	cc-pVDZ	Simint	7.05e-12	5.10e-11	3.45e-14	7.27e-12	6.47e-14	
1×尿嘧啶二聚体	cc-pVDZ	FABO	2.27e-13	1.07e-14	3.23e-17	4.32e-13	2.07e-15	
1×尿嘧啶二聚体	cc-pVDZ	LibintX	0.00e+00	1.42e-14	4.61e-17	1.17e-12	2.65e-15	
1×尿嘧啶二聚体	cc-pVDZ	VSR	6.14e-12	1.03e-11	9.59e-15	1.01e-11	9.15e-14	
1×尿嘧啶二聚体	cc-pVDZ	Laikov	0.00e+00	1.42e-14	5.99e-17	4.86e-13	2.26e-15	
1×苯二聚体 6-31G*		Simint	4.09e-12	1.31e-11	3.84e-14	2.82e-12	4.79e-14	
1×苯二聚体 6-31G*		FABO	0.00e+00	1.07e-14	3.78e-17	2.51e-13	5.63e-15	
1×苯二聚体 6-31G*		LibintX	0.00e+00	7.11e-15	5.78e-17	3.40e-13	5.47e-15	
1×苯二聚体 6-31G*		VSR	2.96e-12	7.89e-12	1.66e-14	4.92e-12	1.27e-13	
1×苯二聚体 6-31G*		Laikov	0.00e+00	1.78e-14	7.55e-17	4.93e-13	6.29e-15	
1×苯二聚体 def2-SVP Simint			5.00e-12	1.21e-11	3.14e-14	8.70e-12	7.34e-14	
1×苯二聚体 def2-SVP FABO			0.00e+00	7.11e-15	3.50e-17	9.74e-13	9.27e-15	
1×苯二聚体 def2-SVP LibintX			0.00e+00	7.11e-15	5.28e-17	2.27e-12	1.49e-14	
1×苯二聚体 def2-SVP VSR			3.18e-12	7.19e-12	1.18e-14	1.64e-11	1.58e-13	
1×苯二聚体 def2-SVP Laikov			0.00e+00	1.07e-14	6.70e-17	1.60e-12	1.42e-14	
1×苯二聚体 cc-pVDZ		Simint	6.59e-12	3.11e-11	5.56e-14	1.00e-11	1.11e-13	
1×苯二聚体 cc-pVDZ		FABO	0.00e+00	1.07e-14	3.71e-17	5.34e-13	7.05e-15	
1×苯二聚体 cc-pVDZ		LibintX	0.00e+00	1.07e-14	5.95e-17	7.50e-13	7.22e-15	
1×苯二聚体 cc-pVDZ		VSR	2.96e-12	3.55e-12	1.41e-14	1.24e-11	1.76e-13	
1×苯二聚体 cc-pVDZ		Laikov	0.00e+00	1.42e-14	7.36e-17	6.68e-13	6.84e-15	
1×戊烷二聚体 6-31G*		Simint	2.16e-12	2.50e-11	4.19e-14	1.80e-12	2.35e-14	
1×戊烷二聚体 6-31G*		FABO	0.00e+00	7.11e-15	3.00e-17	1.35e-13	1.30e-15	
1×戊烷二聚体 6-31G*		LibintX	0.00e+00	7.11e-15	4.47e-17	1.81e-13	1.30e-15	
1×戊烷二聚体 6-31G*		VSR	3.30e-12	6.59e-12	1.09e-14	2.00e-12	3.37e-14	
1×戊烷二聚体 6-31G*		Laikov	1.14e-13	1.07e-14	5.56e-17	1.86e-13	1.56e-15	
1×戊烷二聚体 def2-SVP Simint			2.27e-12	2.71e-11	2.51e-14	2.42e-12	2.33e-14	
1×戊烷二聚体 def2-SVP FABO			0.00e+00	7.11e-15	2.54e-17	2.60e-13	1.73e-15	
1×戊烷二聚体 def2-SVP LibintX			0.00e+00	1.07e-14	3.51e-17	3.31e-13	2.01e-15	
1×戊烷二聚体 def2-SVP VSR			2.84e-12	1.38e-12	5.27e-15	6.18e-12	3.84e-14	
1×戊烷二聚体 def2-SVP Laikov			0.00e+00	1.42e-14	4.55e-17	3.02e-13	1.79e-15	
1×戊烷二聚体 cc-pVDZ		Simint	1.59e-12	6.99e-12	1.46e-14	2.88e-12	2.37e-14	
1×戊烷二聚体 cc-pVDZ		FABO	0.00e+00	7.11e-15	2.68e-17	1.07e-13	9.78e-16	
1×戊烷二聚体 cc-pVDZ		LibintX	0.00e+00	7.11e-15	3.72e-17	1.08e-13	1.10e-15	
1×戊烷二聚体 cc-pVDZ		VSR	2.61e-12	7.59e-12	9.43e-15	5.06e-12	3.58e-14	
1×戊烷二聚体 cc-pVDZ		Laikov	0.00e+00	1.42e-14	5.10e-17	1.07e-13	1.08e-15	

表 S10：QUICK 参考重复分散度：每个矩阵的能量样本标准差和最大逐元素样本标准差。

分子	基组	能量标准差（哈特里）	最大福克矩阵标准差（哈特里）	最大密度标准差
128×水二聚体	6-31G*	0.000e+00	1.863e-14	1.674e-11
128×水二聚体	def2-SVP	0.000e+00	1.577e-14	5.682e-12
128×水二聚体	cc-pVDZ	3.985e-12	1.577e-14	6.045e-12
8×尿嘧啶二聚体	6-31G*	0.000e+00	4.426e-14	3.171e-12
8×尿嘧啶二聚体	def2-SVP	0.000e+00	2.947e-14	4.081e-12
8×尿嘧啶二聚体	cc-pVDZ	6.508e-12	4.094e-14	4.028e-12
16× 苯二聚体	6-31G*	7.970e-12	3.495e-14	1.989e-12
16× 苯二聚体	def2-SVP	3.254e-12	3.178e-14	5.409e-12
16× 苯二聚体	cc-pVDZ	0.000e+00	3.495e-14	4.333e-12
16×戊烷二聚体	6-31G*	0.000e+00	3.846e-14	1.126e-12
16×戊烷二聚体	def2-SVP	0.000e+00	4.713e-14	2.427e-12
16×戊烷二聚体	cc-pVDZ	7.970e-12	3.233e-14	9.160e-13

S7.3 配套数据指南

补充数据和复现说明可从 FABO 发布论文 si-v1 中获取。请解压 si-companion.zip 以获取清单、图表和表格。归档数据包 data.zip 可还原 data/目录；data-manifest.json 标识其中的普通记录，以及单独提供的图表和生成的表格。以下索引使用相对于 data/ 的路径。重新生成需要使用主 FABO代码库中的报告工具和 Python环境，不能仅依靠数据归档完成。包根目录README.md 在“代码库与环境前置条件”、“输入准备与完整性”和“重新生成命令”部分，规定了报告提交版本、设置、输入位置和命令；“完整归档”则标识了单独的矩阵和可执行文件归档。

表S11：配套数据索引和代码库工具入口。简短工具名称指 workflow.scripts 模块，但 workflow.upward_recurrence_limits 除外；完整命令请参见软件包根目录中的 README。

用途	data/ 下的文件	章节/入口
选定模型和资格认定	models.csv; records/numerical/paired-1ulp/summary.json, config.json、identity.json 以及其中的 producer/ 目录	S1–S2 节；导出_支持ing_信息
函数误差和见证点	function-accuracy.csv、maximum-witnesses.csv；	S4 节；
数值示例	records/benchmarks/function-accuracy-20261107/nonuniform-grid-remainder.json, nonuniform-grid-remainder.csv; records/diagnostics/upward-recurrence-limits/	报告_函数_精度 S3 节; 绘制_网格_余项, 向上_递推_限制 plot
工作负载与测试夹具	workloads.csv；application-inputs/quick/ 和 application-inputs/nwchem/；每次运行的 reuse.json	S5.1.1 节；README, “应用程序记录”
源代码与构建	method-sources.csv、methods/、manifest.json, simd-manifest.json；S5.1.1 节列出的构建记录	README、“源代码与载荷标识”
重复计时和加速比	timing-observations.csv、timing-summary.csv, geometric-speedups.csv	第 S6 节； plot_QUICK_性能, 绘图_nwchem_性能
分子比较	molecular-differences.csv；QUICK molecular-means.json 和 NWChem	S7节；报告_分子_准确性
参考变异性	records/benchmarks/ 下的 matrix-absolute.json reference-variability.csv；QUICK摘要和均值矩阵记录	S7节； quick_分子_均值（完整矩阵档案）

数据中保留原始后端标识符；backend-names.csv 将其映射为显示名称。具体而言，official/quick 表示 QUICK，original/simint 表示 Simint，fabo_additive/fabo 表示 FABO，而 libintx_gpu/libintx 表示 LibintX。fabo_software 表示带有软件聚集的 FABO，并非另一种近似。每个矩阵统计量均使用其所在行中指定的情形和后端；单位及合并行约定定义于 README 中。

使用仓库工具和环境，紧凑记录足以重新生成报告中的归约结果和图表。完整执行归档总计约 94 GiB，体积过大，无法随补充材料分发，因此保存在 Slater 和 Heitler 上。README 在“完整归档”一节中列出了其位置和访问要求。要根据原始矩阵重新计算差值，需要这些归档；若要根据输入生成新的矩阵或扫描记录，请按照 S7.3.1 节的说明操作。

S7.3.1 重新运行计算

包根目录的 README.md 在“重建和重新运行”一节中，链接到独立 Boys 扫描（workflow/library_accuracy/README.md）和固定密度分子计算（workflow/quick/FIXED_DENSITY.md 和 workflow/NWCHEM_FIXED_DENSITY.md）的仓库说明。要重复某个分子案例，请使用记录的几何结构、基组和初始态方案。QUICK 的矩阵元在重复运行时可能有所不同；请按照上文所述比较五次运行的平均值。根据冻结元数据重建系数表或根据紧凑记录重新生成报告，并不意味着重新运行这些实验。

参考点

[1] Vinícius Wilian D. Cruzeiro、Madushanka Manathunga、Kenneth M. Merz, Jr. 和 Andreas W. Götz。使用AMBER和QUICK进行的开源多GPU加速QM/MM模拟。*Journal of Chemical Information and Modeling*，61(5):2109–2115，2021。[2] Benjamin P. Pritchard 和 Edmond Chow。电子排斥积分的水平向量化。*Journal of Computational Chemistry*，37(28):2537–2546，2016。[3] Andrey Asadchev 和 Edward F. Valeev。三中心高斯积分的高效递归求值。*Journal of Chemical Theory and Computation*，19(6):1698–1710，2023。[4] Rasmus Vikhamar-Sandberg 和 Michal Repisky。使用极小极大近似高效求值 Boys 函数。*The Journal of Chemical Physics*，164(15):154111，2026。[5] Dimitri N. Laikov。用于向量化计算的 Boys 函数全局近似。*The Journal of Chemical Physics*，163(2):024104，2025年7月。[6] Yasuaki Ito、Satoki Tsuji、Koji Nakano 和 Akihiko Kasagi。GANSU：用于高效 Hartree–Fock 和后 HF 计算的 GPU 原生量子化学框架。*Eng*，7(5):205，2026。[7] Jan Řezáč、Kevin E. Riley 和 Pavel Hobza。S66：一个均衡的数据库，包含与生物分子结构相关的基准相互作用能。*Journal of Chemical Theory and Computation*，7(8):2427–2438，2011。[8] W. J. Hehre、R. Ditchfield 和 J. A. Pople。自洽分子轨道方法。XII. 用于有机分子分子轨道研究的高斯型基组的进一步扩展。*The Journal of Chemical Physics*，56(5):2257–2261，1972。[9] P. C. Hariharan 和 J. A. Pople。极化函数对分子轨道氢化能的影响。*Theoretica Chimica Acta*，28(3):213–222，1973。[10] Florian Weigend 和 Reinhart Ahlrichs。适用于 H 至 Rn 的分裂价层、三ζ价层和四ζ价层质量均衡基组：设计与精度评估。*Physical Chemistry Chemical Physics*，7 3305，2005 3297: – 。[11] Thom H. Dunning, Jr.。用于相关分子计算的高斯基组。I. 硼至氖原子及氢。*The Journal of Chemical Physics*，90(2):1007–1023，1989。