化 学 学 报
ACTA CHIMICA SINICA

研究论文
Article

DOI: 10.6023/A22070313

基于图形处理单元与密度拟合近似的单精度耦合簇 CCSD 和 CCSD(T)程序

王治钒*,[a]　何冰[a]　路艳朝[b]　王繁[b]

([a]成都师范学院　化学与生命科学学院/功能分子结构优化与应用四川省高校重点实验室　成都 611130)

([b]四川大学　原子与分子物理研究所　成都 610064)

摘要　作者此前工作表明, 在耦合簇 CCSD (Coupled-Cluster approaches within the singles and doubles approximation)与 CCSD(T) (CCSD approaches augmented by a perturbative treatment of triple excitations)计算中结合单精度数与消费型图形处理单元(GPU), 可以显著提高计算速度. 然而由于 CCSD(T)计算对内存的巨大需求以及消费型 GPU 的内存限制, 在利用消费型 GPU 进行加速时, 不考虑利用空间对称性的情况下, 此前开发的 CCSD(T)程序仅能用于计算 300～400 个基函数的体系. 利用密度拟合(Density-Fitting, DF)处理双电子积分可以显著降低 CCSD(T)计算过程中的内存需求, 本工作发展了基于密度拟合近似并结合单精度数进行运算的 DF-CCSD(T)程序, 该程序可用于包含 700 个基函数的无对称性体系的单点能计算, 以及包含 1700 个基函数的有对称性体系. 本工作所使用的计算节点配置了型号为 Intel I9-10900k 的 CPU 和型号为 RTX3090 的 GPU, 与用双精度数在 CPU 上的计算相比, 利用单精度数结合 GPU 进行运算可以将 CCSD 的计算速度提升 16 倍, (T)部分可提升 40 倍左右, 而使用单精度数引入的误差可忽略不计. 在程序开发过程中, 作者发展了一套可利用 GPU 或 CPU 结合单精度数或双精度数进行含空间对称性的矩阵操作代码库. 基于该套代码库, 可以显著降低开发含空间对称性的耦合簇代码的难度.

关键词　耦合簇; 密度拟合; 图形处理单元(GPU); 单精度

Single-precision CCSD and CCSD(T) Calculations with Density Fitting Approximations on Graphics Processing Units

Wang, Zhifan*,[a]　He, Bing[a]　Lu, Yanzhao[b]　Wang, Fan[b]

([a] *College of Chemistry and Life Science/Sichuan Provincial Key Laboratory for Structural Optimization and Application of Functional Molecules, Chengdu Normal University, Chengdu 611130, China*)

([b] *Institute of Atomic and Molecular Physics, Sichuan University, Chengdu 610064, China*)

Abstract　It has been reported by our group that using single-precision data and consumer graphics processing units (GPUs) can significantly improve computation speed of CCSD (Coupled-Cluster approaches within the singles and doubles approximation) and CCSD(T) (CCSD approaches augmented by a perturbative treatment of triple excitations). However, CCSD(T) can only be employed for small molecules with about 300～400 basis functions when using consumer GPUs for acceleration without spatial symmetry due to the memory limitation of GPU. Using density-fitting approximation can significantly reduce the memory requirements in CCSD(T) calculations. In this paper, DF-CCSD(T) codes based on the density fitting approximation together with single precision data was developed. All the matrix contractions were performed employing GEMM in CUBLAS on GPU or in Intel MKL on CPU. The other operations such as matrix expansion and transpose were performed using OpenACC on GPU or OpenMP on CPU. Those codes can be applied to single point energies for systems with around 700 basis functions without spatial symmetry and to molecules with about 1700 basis functions with symmetry on a GPU with 24 Gb memory. The server employed in this work has an Intel I9-10900k CPU and a RTX3090 GPU. CCSD calculations with single-precision data on GPU are about 16 times faster and it is about 40 times faster for the (T) part compared with the calculations on CPU using double precision data on this server. Error introduced by single precision data is negligible. A code library that can employ GPU or CPU using either single precision or double precision data to perform matrix operations with spatial symmetry was also reported in this work. Direct product decomposition (DPD) method was employed to deal with spatial symmetry. Complexity of developing coupled cluster codes with spatial symmetry can be significantly reduced with this library. The computational accuracy of single precision DF-CCSD(T) was compared with CCSD(T)-F12a and

* E-mail: wangzhifan1988@gmail.com

Received July 17, 2022; published August 17, 2022.

Supporting information for this article is available free of charge via the Internet at http://sioc-journal.cn.

Project supported by Natrual Science Foundation of Sichuan Province (No. 2022NSFSC1262), National Natural Science Foundation of China (Nos. 21973063, 21703020) and Foundation of Chengdu Normal University Talent Introduction Research Funding (No. 2021YJRC202020) and.

项目受四川省自然科学基金(No. 2022NSFSC1262)、国家自然科学基金(Nos. 21973063, 21703020)和成都师范学院人才引进项目(No. 2021YJRC202020)资助.

DLPNO-CCSD(T). The results shown that DF-CCSD(T) would be more stable than the other two approaches in describing chemical properties.

Keywords coupled-cluster; density fitting; graphics processing units (GPU); single precision

1 引言

在量子化学领域，耦合簇理论(Coupled-Cluster, CC)具有较高的电子相关能计算效率和精度[1]，其中CCSD(T) (CC approaches within the singles and doubles approximation augmented by a perturbative treatment of triple excitations)[2]方法被称为量子化学计算中的黄金标准[3]. 然而，耦合簇方法的计算量较大，CCSD的计算标度为迭代的N^6，N代表体系大小，而CCSD(T)的计算标度为非迭代的N^7. 以一个基函数数目为700，占据轨道数为50的分子为例，在CCSD计算中每一轮迭代的浮点运算次数约为10^{15}，而在(T)部分，浮点运算次数约为10^{17}. 巨大的计算量限制了CCSD(T)的应用范围，使其仅能用于较小的分子体系.

除了计算量较大以外，CCSD计算中面临的另一瓶颈是计算过程中频繁的输入/输出(input/output, I/O)操作. CCSD计算所需数据量为N^4，仍以基函数数目为700，占据轨道数为50的分子为例，在不考虑利用空间对称性的情况下，计算该分子单点能所需硬盘空间大小约1 Tb. 因此在CCSD的计算过程中，每次迭代均需要对总量约1 Tb的数据进行I/O操作. 目前的计算机硬盘中读取速度较快的是固态硬盘，其I/O速度也仅为1～3 GB/s，每轮迭代中仅I/O操作就需花费约500 s时间.

密度拟合(Density Fitting, DF)[4]可以降低量子化学计算过程中的存储空间，并已经被广泛应用于耦合簇计算中[5]. 在DF近似下，只需3指标的双电子积分而无需4指标双电子积分，DF-CC计算的数据量为N^3，因此I/O操作所消耗的时间显著减少. 文献结果显示，与常规CCSD(T)结果相比，DF引入的误差通常小于0.03 kJ/mol[5d].

目前，图形处理器(Graphic-processing units, GPU)在数值计算领域展现出非常优越的性能，并已经开始被广泛应用于各种量子化学计算中[4b,6]，比如密度泛函理论[7]，Hartree-Fock(HF)方法[8]，Møller-Plesset微扰理论[9]，组态相互作用理论[10]等. 在耦合簇计算中，也有越来越多的利用GPU进行加速的报道. DePrince等[5f]首次实现了可利用GPU进行运算的DF-CCSD代码，并充分利用3指标积分节省显存，使用单张NVIDIA Tesla C2070(Fermi) GPU进行运算的速度即可达到利用六颗Intel Core I7-3930k CPU的2.1倍. Fales等[11]基于高性能专业计算显卡NVIDIA V100 GPU发展并优化了DF-CCSD代码，最大能用于包含100原子、1300基函数的体系，并且在一天之内即得到单点能结果. 若利用CPU计算相同体系，则需要64台单节点16核的计算服务器进行并行计算，才可达到同等的计算效率.

在目前的量子化学计算中，主流的软件均使用双精度数进行运算. Knizia等[12]的工作表明单精度数也可以应用于量子化学计算，只要计算中不存在较多的线性累加，单精度数的计算结果往往是可靠的. Krylov等[13]报道了利用单精度数结合耦合簇理论进行单点能、激发能以及解析能量梯度的计算，结果显示单精度数引入的误差均可忽略不计. 在之前的工作中，我们报道了利用单精度数结合消费型GPU进行运算的CCSD(T)代码[14]，对于400个基函数的体系，在使用Intel(R) Core(TM) I9-9900 CPU和Nvidia GTX-2080Ti显卡的情况下，利用GPU进行(T)部分计算的速度可达CPU的12～20倍. 但对于CCSD计算，基函数数目较大时，GPU的加速效果并不显著. 这是因为CCSD部分涉及到的IO操作运算量太大. 为了能进一步用于更大的体系并提高CCSD计算的加速效果，在本工作中，我们报道了利用单精度数结合GPU进行运算的DF-CCSD和DF-CCSD(T)代码. 本工作先简单介绍了相关DF-CCSD(T)公式，然后介绍了代码实施的具体细节，最后通过一些算例，评价单精度DF-CCSD(T)计算的效率.

目前在耦合簇计算领域，对于较大体系，较为常用的两种近似方法为显相关的CCSD(T)-F12[15]方法与局域相关的DLPNO-CCSD(T)[16]方法，本工作还将对比DF-CCSD(T)和这两种方法的计算精度.

2 基本公式

CCSD能量公式为

$$E_{\mathrm{CCSD}} = \left\langle HF \middle| H \exp\left(T_1 + T_2\right) \middle| HF \right\rangle \tag{1}$$

振幅方程为

$$\left\langle \mu_i \middle| \exp(-T_1 - T_2) H \exp(T_1 + T_2) \middle| HF \right\rangle = 0 \tag{2}$$

其中$i=1, 2$，$\langle \mu_i |$表示HF参考态行列式波函数的单双激发行列式波函数. T_1和T_2是CCSD的单激发和双激发算符. CCSD方程的具体形式如方程(3)～(5)所示[17]

$$E_{\mathrm{CCSD}} = \sum_a \sum_i t_i^a f_{ia} + \frac{1}{4} \sum_{ab} \sum_{ij} \tau_{ij}^{ab} \left\langle ij \middle| \middle| ab \right\rangle \tag{3}$$

$$\begin{aligned} t_i^a D_i^a = {} & f_{ia} + \sum_e t_i^e F_{ae} - \sum_m t_m^a F_{mi} + \sum_{me} t_{im}^{ae} F_{me} - \\ & \frac{1}{2} \sum_{mef} t_{im}^{ef} \left\langle ma \middle| \middle| ef \right\rangle - \frac{1}{2} \sum_{men} t_{mn}^{ae} \left\langle nm \middle| \middle| ei \right\rangle - \sum_{nf} t_n^f \left\langle na \middle| \middle| if \right\rangle \end{aligned} \tag{4}$$

$$t_{ij}^{ab}D_{ij}^{ab}=\langle ij||ab\rangle+P_{a\text{-}b}\sum_e t_{ij}^{ae}(F_{be}-\frac{1}{2}\sum_m t_m^b F_{me})-$$
$$P_{i\text{-}j}\sum_m t_{im}^{ab}(F_{mj}+\frac{1}{2}\sum_e t_j^e F_{me})+\frac{1}{2}\sum_{mn}\tau_{mn}^{ab}W_{mnij}+$$
$$\frac{1}{2}\sum_{ef}\tau_{ij}^{ef}\langle ab||ef\rangle+P_{i\text{-}j}P_{a\text{-}b}\sum_{me}(t_{im}^{ae}W_{mbej}-t_i^e t_m^a\langle mb||ej\rangle)+ \quad (5)$$
$$P_{i\text{-}j}\sum_e t_i^e\langle ab||ej\rangle+P_{a\text{-}b}\sum_{efm}\tau_{ij}^{ef}\langle ef||am\rangle t_m^b-$$
$$P_{a\text{-}b}\sum_m t_m^a\langle mb||ij\rangle$$

其中 F 和 W 是公式的部分中间项，其具体形式见支撑材料. 方程中 i, j, m, n 为占据轨道指标, a, b, e, f 为空轨道指标, <pq||rs>为双电子积分, p, q, r, s 为分子轨道指标.

在 DF-CCSD 中，双电子积分可以由方程(6)得出，

$$\langle pq|rs\rangle_{\mathrm{DF}}=(pr\,|\,qs)_{\mathrm{DF}}=\sum_n^{N_{\mathrm{aux}}}(pr\,|\,n)(n\,|\,qs) \quad (6)$$

其中 $(pr\,|\,n)$ 为 DF 近似下的 3 指标积分，n 代表辅助基函数, N_{aux} 代表辅助基函数数目. 实现 DF-CCSD 的最简单的方法是直接利用 $(pr\,|\,n)$ 生成方程(3)～(5)中所需的所有双电子积分，但这种方式不能重复利用 DF 近似能够降低内存使用量的优势. 另一种方式是将方程(3)～(5)中的双电子积分利用(6)式进行代替，再设计合适的方案进行计算. 以 CCSD 中计算量最大的

$$\sum_{ef}\tau_{ij}^{ef}\langle ab||ef\rangle$$

为例，引入 3 指标积分之后，该项将变为方程(7)所示，

$$R_{ij}^{ab}=\sum_{ef}\tau_{ij}^{ef}(ea|n)(n|fb)-\sum_{ef}\tau_{ij}^{ef}(fa\,|\,n)(n\,|\,eb) \quad (7)$$

对于该项的程序实现有两种常用方式:

(1)将单双激发振幅矩阵 τ_{ij}^{ef} 进行分区处理，对其中每一对 i 和 j 指标，先计算 $t(ef)$*$(ae|n)$= $tmp(af|n)$，再计算 $tmp(af|n)$*$(n|bf)$=$r(ab)$. 再将 $r(ab)$放入 R_{ij}^{ab} 矩阵中的对应位置. 这种实现过程的计算量为 $2N_{\mathrm{o}}^2N_{\mathrm{v}}^3N_{\mathrm{aux}}$ (N_{v} 为空轨道数目, N_{o} 为占据轨道数目).

(2)对于 3 指标积分中每一个特定的 b，先计算 $(ea|n)$*$(n|f)$=$tmp(ea|f)$=<$ef|a$>，然后计算 $\tau_{ij}^{ef}\langle ef\,|\,a\rangle=R_{ij}^{a}$，再将 R_{ij}^{a} 放入 R_{ij}^{ab} 矩阵中的对应位置. 该方法计算量为 $N_{\mathrm{o}}^2N_{\mathrm{v}}^4+N_{\mathrm{v}}^4N_{\mathrm{aux}}$.

以上两种实现方式，对内存的最大需求均为 $N_{\mathrm{v}}^2N_{\mathrm{aux}}$，通常情况下，辅助基函数的数目为基函数数目的3～5倍，同时占据轨道数目远小于空轨道数目. 因此第二种实现方式的计算量会小于第一种实现方式. 同时，第二种实现方式也更容易通过修改已有的 CCSD 代码来实现.

以有 700 个基函数和 50 个占据轨道的分子体系为例，在利用常规 CCSD 方法进行计算时，空轨道双电子积分<VV|VV>所占空间为 1.8 Tb, <VV|VO>所占空间为 120 Gb (V 表示空轨道, O 表示占据轨道), <VV|OO>所占空间仅为 9 Gb. 如果采用单精度数，以上各双电子积分所占空间会减半. 为了充分利用我们此前已经实现的 CCSD 程序，我们只针对涉及<VV|VV>和<VV|VO>的项利用 3 指标积分进行重新编写，而对其他涉及<VV|OO>, <VO|VO>, <OO|OV>, <OO|OO>的相关项，均先通过(6)式由 3 指标积分得到相应的 4 指标双电子积分，再利用此前的 CCSD 代码进行运算.

在(T)的计算中，最大的双电子积分项为<VV|VO>，对于无对称性且基函数数目小于 700 的体系，该双电子积分的大小在 200 GB 以内. 我们此前开发的用单精度数结合 GPU 的(T)计算程序所用内存不大，因此在本工作的(T)计算中，我们首先利用(6)式得到(T)计算所需的双电子积分，再直接利用之前已经开发完成的代码计算(T)的能量校正.

3 应用细节

我们发展的 DF-CCSD(T)程序利用 Fortran90 编译器进行编写，并可进行独立编译与运行. 其中 DF 的 3 指标积分和分子轨道能量等关键信息均为通过 PySCF 程序[18]得到. 由于 PySCF 程序在计算分子轨道基的双电子积分时没有利用空间对称性，我们基于 PySCF 给出分子轨道的不可约表示信息，利用直积分解(direct product decomposition, DPD)方法[17]将 PySCF 得到的积分进行重新排布，以便在 CC 计算过程中利用空间对称性降低计算量.

图 1 为 DF-CCSD 程序框架图，为了便于对代码进行 GPU 加速运算，我们将 DF-CCSD 代码中的迭代计算分为 6 步. 其中，1～2 步完全在 CPU 上进行运算. 3～6 步可根据体系对存储空间的需求与服务器的内存进行判断并自动选择在 CPU 或者 GPU 上运行.

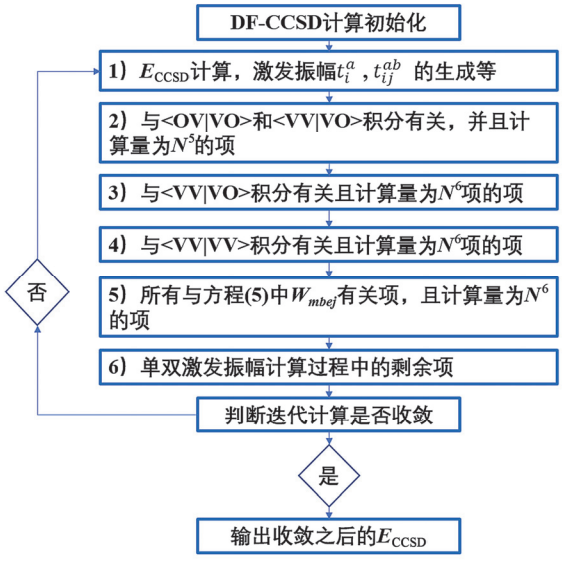

图 1 程序框架图
Figure 1 Program structure diagram

本工作将以迭代计算过程中的第4步为例介绍程序实现方式. 第4步仅有一项, 为

$$\sum_{ef}\tau_{ij}^{ef}\left\langle ab\middle\|ef\right\rangle_{\mathrm{DF}}$$

对于闭壳层体系, 需要进行计算的自旋情况为

$$\sum_{ef}\tau_{i\bar{j}}^{e\bar{f}}\left\langle a\bar{b}\middle\|e\bar{f}\right\rangle_{\mathrm{DF}}$$

由于$\left\langle a\bar{b}\middle\|e\bar{f}\right\rangle_{\mathrm{DF}}=\sum_n(ae|n)(n|bf)$, 将在整个计算过程中临时生成, 所以内存使用量约为 $N_{\mathrm{v}}^2N_{\mathrm{aux}}+N_{\mathrm{v}}^2N_{\mathrm{o}}^2$, 该内存占有量较小, 因此对于基函数总数小于700的体系, 都能够存储在内存中. 但消费型GPU显存容量通常不超过 24 GB. 因此, 在我们所发展的程序中, 将根据GPU 显存的大小, 判断该项内存的使用量是否超出GPU 显存, 进而选择是否使用 GPU 进行加速运算. 生成$\left\langle ab\middle\|ef\right\rangle_{\mathrm{DF}}$的计算量为 $N_{\mathrm{v}}^4N_{\mathrm{aux}}$, 因此该部分总计算量为$N_{\mathrm{v}}^4N_{\mathrm{o}}^2+N_{\mathrm{v}}^4N_{\mathrm{aux}}$. 为了降低计算量, 该方程可以变形为方程(8)和方程(9)所示.

$$\sum_{ef}\tau_{i\bar{j}}^{e\bar{f}}\left\langle e\bar{f}\middle\|a\bar{b}\right\rangle=\sum_{e\le f}Tp_{i\le j}^{e\le f}\,Ip_{a\le b}^{e\le f}+\sum_{e<f}Tm_{i<j}^{e<f}\,Im_{a<b}^{e<f}\quad(8)$$

$$Tp_{i\le j}^{e\le f}=t_{i\bar{j}}^{e\bar{f}}+t_{j\bar{i}}^{e\bar{f}}\qquad Ip_{a\le b}^{e\le f}=\left\langle e\bar{f}\middle\|a\bar{b}\right\rangle+\left\langle e\bar{f}\middle\|b\bar{a}\right\rangle$$

$$Tm_{i<j}^{e<f}=t_{i\bar{j}}^{e\bar{f}}-t_{j\bar{i}}^{e\bar{f}}\qquad Im_{a<b}^{e<f}=\left\langle e\bar{f}\middle\|a\bar{b}\right\rangle-\left\langle e\bar{f}\middle\|b\bar{a}\right\rangle\quad(9)$$

其计算量可降低为$1/4N_{\mathrm{v}}^4N_{\mathrm{o}}^2+2N_{\mathrm{v}}^4N_{\mathrm{aux}}$. 为了尽可能降低含对称性的 CC 程序的开发难度, 我们将 CC 近似中最常见的矩阵转置和矩阵收缩等相关代码进行了封装, 以减少代码开发过程中的重复性工作. 以方程(8)中的

$$\sum_{e<f}Tm_{i<j}^{e<f}\,Im_{a<b}^{e<f}$$

以及其对应的方程(9)中相关公式为例, 在图 2 中, 我们列出了封装之后的代码, 以及每行代码对数据矩阵所进行的操作.

代码中的 t2abij 对应$t_{i\bar{j}}^{e\bar{f}}$, rivv 对应 DF 的 3 指标积分(n|vv), 在子程序 VmnpqVmnrs_to_Vpqrs_ri_in 中, 还自动完成了$\left\langle e\bar{f}\middle\|a\bar{b}\right\rangle=\sum_n(ae|n)(n|bf)$这一计算. 在计算过程中, 利用对部分控制参数的设置, 可以确定输入的 t2abij 和 DF 的 3 指标积分 rivv 的类型(单双精度)和计算使用的硬件(CPU 或 GPU), 由此图 2 中的代码即可同时兼容单精度与双精度数, 并利用 CPU 或 GPU 进行计算. 此外, 在使用这部分代码时, 还可以利用空间对称性降低计算量. 从图 2 中的代码示例可以看出, 通过对矩阵相关操作代码的封装, 能显著简化 DF-CC 程序的开发, 对图 2 代码示例的详细描述见支撑材料.

call iassymso(t2abij,ttmp,1,vrt,vrt,pop,pop,1)

$Ttmp_{i<j}^{e,f}=t_{ij}^{e\bar{f}}-t_{j\bar{i}}^{e\bar{f}}$

call icompso(ttmp,t-,vrt,vrt,1,0,pop,pop,0,1,1)

$Tm_{i<j}^{e<f}=Ttmp_{i<j}^{e,f}$

call isymtrso(t2abij,ttmp,vrt,vrt,1,1,pop,pop,1,0,1)

$ttmp_{ij}^{fe}=t_{ij}^{e\bar{f}}$

call axpyso(nsize,1.d0,tabij,1,ttmp,1)

$t_{ij}^{e\bar{f}}=ttmp_{ij}^{ef}+t_{ij}^{e\bar{f}}$

call VmnpqVmnrs_to_Vpqrs_ri_in(t-,icore,1,1,&
vrt,vrt,0,pop,pop,0,vrt,vrt,0,Rijab1,1,&
0.5d0,0.d0,rivv)

$Rtmp(i<j,e<f)=\frac{1}{2}\sum_{e<f}Tm_{i<j}^{e<f}\,Im_{a<b}^{e<f}$

call itranspso(Rijab1,Rabij1,pop,pop,0,vrt,vrt,0,1)

$R_{i<j}^{e<f}=Rtmp(i<j,e<f)$

图 2 代码示例
Figure 2 Example codes

4 计算结果与讨论

本工作计算所用服务器的GPU为单张RTX3090显卡(24 GB 显存), CPU 为单颗 Intel(R) Core(TM) I9-10900k CPU (10 cores@3.7GHz). 服务器的内存为128 GB. DF-CCSD(T)代码均为利用 PGI-20 结合 Intel MKL 与 CUBLAS 进行编译, 矩阵乘法运算主要使用CUBLAS中的GEMM子程序在GPU上计算或利用Intel MKL中的GEMM子程序在CPU上计算, 其他的矩阵转置等操作主要使用 OpenACC 在 GPU 上进行计算或OpenMP 在 CPU 上进行计算.

4.1 单精度 DF-CCSD(T)计算结果的准确性

在之前的工作中, 我们已经利用单精度CCSD(T)对部分含轻元素或重元素的分子体系进行了计算测试, 结果表明利用单精度数计算激发振幅, 最后利用混合精度数计算能量, 能得到可靠的结果, 并且单精度数引入的误差可以忽略不计. 和CCSD(T)相比, DF-CCSD(T)会增加一项矩阵收缩运算,

$$\left\langle pq|rs\right\rangle_{\mathrm{DF}}=(pr|qs)_{\mathrm{DF}}=\sum_{n}^{N_{\mathrm{aux}}}(pr|n)(n|qs)$$

为了考察单精度 DF-CCSD(T)计算的可靠性, 我们计算了$(H_2O)_7$团簇, C_6H_{14}, $C_{10}H_{18}$, AtF_5和I_3^-等分子的单点能. 其中, $(H_2O)_7$团簇, C_6H_{14}, $C_{10}H_{18}$使用 cc-pVTZ 基组[19], 辅助基组采用的是 cc-pVTZ-jkfit 基组[20]. AtF_5和I_3^-使用 aug-cc-pV5Z-PP 基组[21]以及对应的有效核势[21a,22], 其辅助基组则利用 PySCF 软件自动生成[23]. 从表 1 的数据中可以看到, 利用单精度数和双精度数计

表 1 利用单精度数(single precision, SP)在 GPU 和 CPU 上进行计算和利用双精度数(double precision, DP)在 CPU 上进行计算得到的 DF-CCSD 相关能与(T)计算得到的相关能(单位: a.u.)

Table 1 Correlation energy of DF-CCSD and (T) calculated on GPU and CPU with single precision (SP) data and calculated on CPU with double precision (DP) data (unit: a.u.)

Molecular	CCSD			(T)		
	SP GPU	SP CPU	DP CPU	SP GPU	SP CPU	DP CPU
$(H_2O)_7$	−1.9878328	−1.9878329	−1.9878329	−0.0585549	−0.0585566	−0.0585566
C_6H_{14}	−1.1398594	−1.1398593	−1.1398594	−0.0440448	−0.0440449	−0.0440449
$C_{10}H_{16}$	−1.7801039	−1.7801038	−1.7801038	−0.0798821	−0.0798823	−0.0798823
AtF_5	−2.2048644	−2.2048637	−2.2048637	−0.0941050	−0.0940953	−0.0941050
I_3^-	−1.3032696	−1.3032697	−1.3032697	−0.0498108	−0.0498108	−0.0498108

算得到的 CCSD(T)能量结果差别小于 10^{-6} a.u.. 该结果表明, 利用单精度 DF-CCSD(T)进行单点能计算时, 单精度数导致的误差很小, 对计算结果精度的影响可以忽略不计.

4.2 利用 GPU 结合单精度数对 DF-CCSD(T)计算的加速表现

在这一部分, 我们通过计算无对称性和有空间对称性的水团簇, 评价利用 GPU 结合单精度数对 DF-CCSD(T)计算的加速表现, 其中有对称性的水团簇的点群为 C_s, 本部分计算所用基组为 cc-pVTZ 基组[19], 辅助基组为 cc-pVTZ-jkfit[20]. 在表 2 第 3～4 列分别是利用 GPU 和 CPU 进行单精度 DF-CCSD 计算的单次迭代时间, 其中括号外数据是无对称性水团簇的计算时间, 括号内为计算有对称性水团簇所花费的时间. 第 5 列为 GPU/CPU 的计算加速比. 首先对比无对称性水团簇的计算时间, 从$(H_2O)_3$到$(H_2O)_9$, 相较于利用 CPU 的计算, 利用 GPU 可以得到 7 倍左右的加速比, 而当体系进一步增大, GPU 的加速效果降低. 我们在表 3 中列出了分别利用 GPU 和 CPU 结合单精度 DF-CCSD 计算$(H_2O)_9$和$(H_2O)_{10}$的过程中, 按照图 1 程序框架图所列各步骤的计算时间. 从表 3 的结果可以发现, 在$(H_2O)_{10}$的 CCSD 迭代中, 在选择利用 GPU 进行程序加速时, 第 5 步涉及 W_{mbej} 的项已经没有再使用 GPU 进行加速, 这是由于 W_{mbej} 所占存储空间为 27 GB, 已经超出服务器 GPU 的 24 GB 显存. 由此基于 GPU 的计算速度受到了较为显著的影响.

在耦合簇计算过程中, 如果体系具有空间对称性, 则利用空间对称性可以显著降低计算量. 表 2 中括号内的数据均为针对具有 C_s 对称性的水分子团簇的计算时间. 在利用 C_s 对称性进行计算时, 由于 C_s 对称性有两个不可约表示, 双电子积分所占空间将降至无对称性的体系的 1/2 左右, 而计算量则根据计算过程中公式不同, 可能降至原计算量的 1/2 至 1/4, 由此, 综合计算速度可以提升 2～4 倍. 第 3 列的结果显示, 用 GPU 进行 CCSD 计算时, 对于$(H_2O)_3$～$(H_2O)_8$, 利用空间对称性之后的计算速度为不利用空间对称性计算速度的 2～2.5 倍. 这一系列结果基本与利用空间对称性之后的理论提速相符合. 而随着体系的进一步增大, 利用空间对称性对$(H_2O)_9$的提速为 3 倍, 对$(H_2O)_{10}$的提速达到 6.5 倍, 已经超过了理论上的最大提速倍数. 这是由于随着体系的增大, 在不含对称性时, $(H_2O)_{10}$的 W_{mbej} 项的计算已经不能利用 GPU 进行加速计算, 而利用了空间对称性之后, 由于所需存储空间较小, 计算 W_{mbej} 时仍然能够利用 GPU 进行加速, 由此带来了显著的计算速度的差异. 在利用 CPU 的单精度 RI-CCSD 计算中, 从$(H_2O)_3$至$(H_2O)_9$, 引入对称性有 2～2.5 倍的提速, $(H_2O)_{10}$达到 3 倍提速, 均在理论提速范围之内, 这表明本工作发展的 DF-CCSD 代码能够充分利用对称性以提升计算效率.

(T)部分的计算量为单次的 $N_v^4 * N_o^3$, 我们继续评价了代码在(T)部分, 利用 GPU 结合单精度数进行计算的加速效果. 第 7 列和第 8 列分别为利用 GPU 和 CPU 进行(T)计算所耗费时间, 其中括号外为无对称性的 H_2O 团簇的计算时间, 括号内数据是对称性为 C_s 的 H_2O 团簇的计算时间, 第 9 列为 GPU 与 CPU 计算的时间比. 从中可以看出 GPU 在(T)部分的总体加速效果非常显著, 在计算不含对称性的 H_2O 团簇时, 对于含有 4～8 个水分子的团簇, 可提速 20 倍以上, 而对于 9～10 个水分子的团簇, 提速分别为约 18 和 16 倍, 这是由于随着体系的增大, 在进行(T)计算时, 会引入更多的 CPU 和 GPU 之间的数据传输操作, 由此降低 GPU 的提速效果. 另外, 对于三个水分子这种较小的体系, 计算提速仅为 8 倍, 这是由于体系非常小的时候, GPU 的部分内存指令操作也会消耗一定的时间, 导致计算提速效果降低.

从第 7 列的结果可以看到, 利用 GPU 进行(T)计算加速时, 对于$(H_2O)_4$～$(H_2O)_7$, 利用空间对称性可以将计算速度提升 2～3 倍, 而对于$(H_2O)_8$～$(H_2O)_{10}$, 利用对称性可以将计算速度提升 3～4 倍. 另一方面, 由第 8 列数据可得, 利用 CPU 进行(T)的计算时, 使用对称性的提速效果基本在 3 倍左右. 在计算$(H_2O)_3$时, 无论用 CPU 还是用 GPU, 利用对称性对计算速度的提升均小于 2 倍, 这也是由于体系过小, 计算总体时间过短所导致. 总体而言, 不管是用 CPU 还是 GPU 计算这些分子时, 利用空间对称性均可以将(T)部分的计算速度提高 3 倍左右.

我们还利用双精度数结合 CPU 进行了 CCSD(T)单点能计算, CCSD 和(T)的计算时间分别列在第 5 列与第

表 2 DF-CCSD(T)分别利用单精度或双精度数在 CPU 或 GPU 上进行$(H_2O)_n$团簇单点能计算的计算耗时, 其中括号内为含 C_s 空间对称性的计算耗时, 括号外为不含空间对称性的计算耗时(单位: s)

Table 2 Wall-time for DF-CCSD and DF-CCSD(T) calculations on CPU and GPU with single-precision (SP) and double-precision (DP). Data inside brackets is the computation time with C_s spatial symmetry and data outside brackets is the computation time without spatial symmetry (unit: s)

		Wall-time of a single RI-CCSD iteration				Wall-time of (T) correction calculation			
		SP	SP	SP	DP	SP	SP	SP	DP
n	Number of basis	GPU Time	CPU Time	$\frac{\text{CPUTime}}{\text{GPUTime}}$	CPU Time	GPU Time	CPU Time	$\frac{\text{CPUTime}}{\text{GPUTime}}$	CPU Time
3	174	0.9 (0.4)	5.46 (2.24)	6.07 (5.6)	12.4 (5.2)	2.8 (1.8)	22.4 (13.9)	8 (7.7)	40 (33)
4	232	3.1 (1.4)	17.7 (9.3)	5.71 (6.64)	37.2 (19.0)	14 (6.7)	336 (105)	24 (15.7)	672 (244)
5	290	7.8 (3.7)	50.9 (24.6)	6.53 (6.65)	106 (52.1)	62 (23)	1514 (492)	24.4 (21.4)	2698 (1035)
6	348	18 (8)	129 (58)	7.17 (7.25)	243 (122)	189 (69)	4866 (1555)	25.7 (22.5)	8476 (3174)
7	406	35 (14)	283 (111)	8.09 (7.93)	523 (226)	542 (179)	12588 (4359)	23.2 (24.4)	22312 (8399)
8	464	66 (28)	508 (220)	7.7 (7.86)	1012 (436)	1514 (445)	31404 (11527)	20.7 (25.9)	53604 (19188)
9	522	131 (44)	913 (389)	6.97 (8.84)	1815 (774)	3601 (898)	65888 (22203)	18.3 (24.7)	113226 (40485)
10	580	529 (81)	1768 (656)	3.34 (8.1)	(1626)	7608 (2313)	119307 (46606)	15.7 (20.2)	(84115)

表 3 不含空间对称性的$(H_2O)_9$与$(H_2O)_{10}$水团簇单次 CCSD 迭代中, 图 1 程序框架图所定义各步所用时间(单位: s)

Table 3 Wall-time of each step defined in Figure 1 in a single CCSD iteration for $(H_2O)_9$ and $(H_2O)_{10}$ without spatial symmetry (unit: s)

	$(H_2O)_9$		$(H_2O)_{10}$	
	GPU	CPU	GPU	CPU
1) Initio	15.6	15.3	233.7	235
2) N^5	60	53.6	84.5	87.1
3) Iabci	3.5	22	7	37.6
4) Iabcd	37.7	623	66	1050
5) Wmbej	10.79	169.9	320.2	319.2
6) left	2.5	18.9	3.5	30

9 列中. 对比分别利用双精度数和单精度数的 CPU 计算耗时可以看到, 对于同一分子, 使用双精度数的计算耗时约为使用单精度数的 2 倍.

从水团簇的计算结果可以看到, 利用单精度数结合 GPU 进行 CCSD(T)单点能计算时, 相比利用双精度数结合 CPU 计算而言, 在 CCSD 部分, 可以得到 13～16 倍的提速效果, 在(T)部分, 则可以达到 40～50 倍的提速效果, 而计算精度的损失基本可以忽略不计. 与无对称性体系相比, 对于 C_s 对称性的体系, 利用对称性可以将计算速度提升 2～3 倍. 若体系有更高的对称性, 提速效果会更加显著.

4.3 单精度 DF-CCSD(T)计算碳氢化合物的反应能

在这一部分, 我们利用 CCSD(T), 单精度 DF-CCSD(T), CCSD(T)-F12a 和 DLPNO-CCSD(T)计算了 Minnesota 数据库中的 7 个碳氢化合物的反应能(HC7), 所涉及化合物的几何结构均来自该数据库[24]. CCSD(T)-F12a 方法利用 Molpro[25] 进行计算, DLPNO-CCSD(T)利用 ORCA[26]进行计算, 在利用 DLPNO-CCSD(T)进行计算时, 使用 ORCA 默认的截断阈值, 计算结果列在表 4 中. 计算过程中使用的基组为 cc-pVnZ[19]系列基组以及对应的 cc-pVnZ-jkfit[20]辅助基组. 对于表 4 中的反应(1)～(5), 在使用 cc-pVTZ 基组时, CCSD(T)和 DF-CCSD(T)计算得到的反应能差别小于 0.4 kJ/mol. 对于反应(6)和反应(7), 反应能差别小于 1.2 kJ/mol. 而使用 cc-pVTZ 基组的 DLPNO-CCSD(T)所得反应能与 CCSD(T)结果差别达到 3.3 kJ/mol.

采用更大的基组时, 常规的 CCSD(T)难以计算这些体系的能量. 为了估计采用更大基组后 DF 近似的误差, 我们分别利用 MP2 和 DF-MP2 结合 cc-pV5Z 基组以及 cc-pV5Z-jkfit 辅助基组计算了反应(4)～(7)的反应能, 结果显示二者之间差别均小于 0.5 kJ/mol. 这说明在 cc-pV5Z 基组下, DF 近似对反应能的影响依然较小. 我们采用 cc-pVQZ 基组时, DF-CCSD(T)所得反应能与 DLPNO-CCSD(T) 结果的最大差别达到了 8 kJ/mol. 之前有报道称 CCSD(T)-F12a 结合 cc-pVDZ 基组, 可以得到常规 CCSD(T)结合 cc-pVQZ 的计算精度[15], 然而表中数据显示, 采用 cc-pVDZ 基组的 CCSD(T)-F12a 所得反应能与采用 cc-pVQZ 的 DF-CCSD(T)以及 DLPNO-CCSD(T)的结果都有非常显著的差距, 最大可达 70 kJ/mol. 此前文献结果也显示, 采用 cc-pVDZ 基组的 CCSD(T)-F12a 对一些体系也存在较大误差[15].

采用 cc-pV5Z 基组时, 由于所计算的体系过大, 我们所开发的 DF-CCSD(T)程序只能用于计算反应(4)～(7). 在此基组下, DF-CCSD(T)与 DLPNO-CCSD(T)的计算结果相差最大为 5 kJ/mol.

已有报道称 CCSD(T)-F12a 结合 cc-pVTZ 基组可以得到采用 cc-pV5Z 基组的 CCSD(T)计算精度. 然而对比表 4 中反应(4)～(7)的反应能数据, 可以看到采用

表 4 基于 Minnesota 数据库的分子结构(HC7), 利用不同的基组与 CC 方法计算得到的碳氢化合物的反应能(单位: kJ/mol)
Table 4 Hydrocarbon reaction energies calculated with different basis sets and CC method based on the molecular structure of Minnesota database (HC7) (unit: kJ/mol)

Reaction	cc-pVTZ CCSD(T)	cc-pVTZ DF-CCSD(T)	cc-pVTZ DLPNO-CCSD(T)	cc-pVQZ DF-CCSD(T)	cc-pVQZ DLPNO-CCSD(T)	cc-pVDZ CCSD(T)-F12a	cc-pV5Z DF-CCSD(T)	cc-pV5Z DLPNO-CCSD(T)	cc-pVTZ CCSD(T)-F12a	Ref. [a]
(1) $E_{22}\rightarrow E_1$	56.028	55.823	56.551	53.145	57.618	85.713		59.865	66.35	59.999
(2) $E_{31}\rightarrow E_1$	94.847	94.508	94.583	86.768	95.42	133.955		97.345	107.106	104.684
(3) $(CH_3)_3CC(CH_3)_3\rightarrow C_8H_{18}$	8.694	8.715	6.904	7.736	6.464	9.301		6.209	8.556	7.95
(4) $C_6H_{14}+4CH_4\rightarrow 5C_2H_6$	35.204	35.179	33.359	36.53	34.623	35.539	38.777	35.585	35.514	41.045
(5) $C_8H_{18}+6CH_4\rightarrow 7C_2H_6$	53.099	53.057	49.773	55.078	51.953	53.321	58.501	53.597	53.48	62.091
(6) adamantane$\rightarrow 3C_2H_4+2C_2H_2$	819.855	818.754	817.298	821.675	820.608	890.255	822.089	825.968	840.009	811.654
(7) bicyclo_octane$\rightarrow 3C_2H_4+C_2H_2$	530.996	530.251	527.661	532.824	530.464	581.706	531.841	535.088	547.024	532.288

[a] Reference 23.

cc-pVTZ 基组的 CCSD(T)-F12a 得到的反应能与 DF-CCSD(T)的计算结果之间最大相差 18 kJ/mol.

值得注意的是, 如 C_8H_{18} 等分子体系, 在使用 cc-pV5Z 基组时, 虽然基函数数目已达 1700, 但是由于体系具有 C_s 对称性, 使用单精度结合 GPU 进行 DF-CCSD(T)计算仅需花费 40 h.

在 DF-CCSD(T)计算中, 对于反应(4)～(7), cc-pV5Z 与 cc-pVTZ 得到的反应能之间差距最大为反应(5), 其结果为 5.5 kJ/mol. 其他反应能之间差距均小于 5 kJ/mol. DLPNO-CCSD(T)计算中, cc-pVTZ 与 cc-pV5Z 基组计算的反应能最大差距均小于 10 kJ/mol. 这说明对于该系列反应体系, 利用 cc-pVTZ 基组即可合理描述相关化学性质. 最后一列数据为 Minnesota 数据库提供的参考值, 该参考值利用 cc-pVDZ 基组进行计算, 本工作中计算数据应比该数据更加可靠.

4.4 单精度 DF-CCSD(T)在构建势能面中的表现

我们分别利用 CCSD(T)、DF-CCSD(T)、DLPNO-CCSD(T)与 CCSD(T)-F12a 计算了线性结构 CO_2 二聚体的势能面, 计算过程中使用基组为 aug-cc-pVnZ[19]系列基组以及 aug-cc-pVnZ-jkfit[20]辅助基组. 所得势能面展示在图 3 中. 在绘制势能曲线时, 以各种近似方法下计算出来的 CO_2 单点能作为参考零点. 由于 CO_2 二聚体之间相互作用较弱, 因此对势能面的构建需要较高的计算精度. 在利用 aug-cc-pVTZ 和 aug-cc-pVQZ 基组进行计算时, 如图 3(a)所示的 CCSD(T)得到的势能面与 DF-CCSD(T)结合单精度数得到的势能面几乎完全重合, 因此单精度 DF-CCSD(T)对势能面的计算结果是可靠的. 在图 3(b)中, 我们对比了 CCSD(T)-F12a 与 DF-CCSD(T)所得势能面, 从图 3(b)可以得到, CCSD(T)-F12a 结合 aug-cc-pVTZ 基组得到的势能面与 DF-CCSD(T)结合 aug-cc-pV5Z 基组得到的结果相近, 在平衡构型附近, 采用 aug-cc-pVDZ 的 CCSD(T)-F12a 所得势能面与采用 aug-cc-pVQZ 的 DF-CCSD(T)结果比较接近, 但是两 CO_2 之间的距离较远时, 这两个方法的结果有较大差距. 在图 3(c)中, 我们对比了采用 aug-cc-pVTZ 的 CCSD(T)-F12a 势能面,

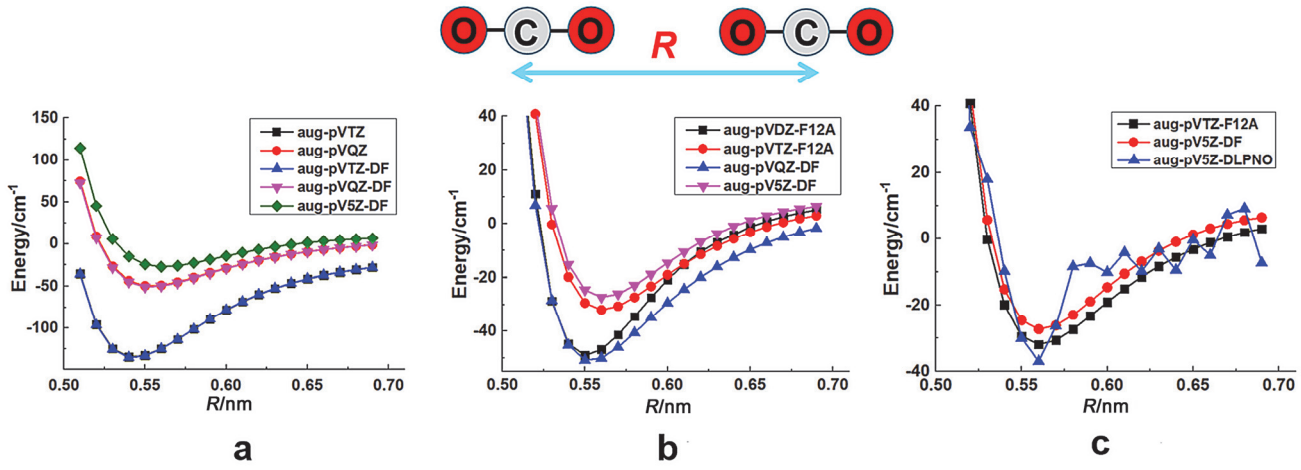

图 3 线性 CO_2 二聚体的势能面
Figure 3 Potential energy surface of linear CO_2 dimer

以及采用 aug-cc-pV5Z 的 DF-CCSD(T)和 DLPNO-CCSD(T)势能面. 可以看到, DLPNO-CCSD(T)在计算 CO_2-CO_2 这种能量变化较小的势能面时, 不能得到光滑的势能面曲线, 而 DF-CCSD(T)与 CCSD(T)-F12a 计算得到的势能面吻合度高, 这说明 DF-CCSD(T)与 CCSD(T)对势能面的计算结果均是可靠的.

通常来说, CCSD(T)-F12 方法的计算量略大于 CCSD(T), 而 DLPNO-CCSD(T)方法由于采用了局域相关近似, 它的计算量一般小于 CCSD(T). 但是各方法的计算效率依赖于所使用的基组、体系大小以及具体的实现方式. 在计算 CO_2 二聚体势能面时, 我们分别利用了 Molpro, ORCA 和 PySCF 结合我们自行发展的代码开展了 CCSD(T)-F12a, DLPNO-CCSD(T), DF-CCSD(T)计算. 由于采用了不同程序, 而且 CCSD(T)-F12a 计算所用基函数与 DF-CCSD(T)以及 DLPNO-CCSD(T)计算中使用的基函数并不相同, 因此计算时间难以完全反映各个方法的计算效率. 这里我们给出各个方法的计算时间, 仅供参考: CCSD(T)-F12a/aug-cc-pVTZ 计算中, 利用了 D_{2h} 对称性后, CC 部分计算耗时为 285 s; DLPNO-CCSD(T)/aug-cc-pV5Z 计算中, 不能利用空间对称性, 其 CC 部分计算耗时为 1018 s, 而基于单精度数在 GPU 上进行 DF-CCSD(T)/aug-cc-pV5Z 运算时, 利用了 D_{2h} 空间对称性, CC 部分计算耗时为 370 s.

5 结论

本工作中, 我们发展了可利用 GPU 结合单精度数进行运算的 DF-CCSD(T)计算程序, 并可以利用空间对称性降低计算量, 提高计算效率. 计算结果显示, 在 DF-CCSD(T)计算过程中, 单精度数所引入的计算误差可以忽略不计. 在 DF-CCSD(T)计算过程中, 对于 CCSD, 利用 RTX3090 显卡结合单精度数比使用 I9-10900k CPU 结合双精度数可获得 14 倍左右加速效果, 而在(T)部分, 则可以获得 40 倍左右的加速效果. 该程序的 GPU 计算过程对空间对称性的支持表现良好, 在目标体系具有一定对称性的前提下, 已经可以应用于基函数数目达 1700 的体系.

在计算一些碳氢化合物的反应能时, DF-CCSD(T)与 DLPNO-CCSD(T)的计算结果总体相差较小, 而采用 cc-pVDZ 基组的 CCSD(T)-F12a 对一些体系则有较大差距. 在计算 CO_2 二聚体势能面时, DF-CCSD(T)与 CCSD(T)-F12a 均可以得到光滑的势能面, 而 DLPNO-CCSD(T)不能得到. 由此可得, 相较于 CCSD(T)-F12a 和 DLPNO-CCSD(T)而言, DF-CCSD(T)在描述各类化学性质时, 表现更加稳定.

本工作还发展了可利用 GPU 或 CPU 结合单精度数或双精度数进行含空间对称性的矩阵操作的代码库, 利用此代码库可以显著降低编写耦合簇程序的难度. 在后期工作中, 我们将基于该代码库, 进一步开发可利用单精度数结合消费型 GPU 进行运算的开壳层 DF-CC 方法, 解析能量梯度计算以及基于 EOM-CC 的激发能计算程序.

References

[1] Bartlett, R. J.; Musiał, M. *Rev. Mod. Phys.* **2007**, *79*, 291.
[2] Raghavachari, K.; Trucks, G. W.; Pople, J. A.; Head-Gordon, M. *Chem. Phys. Lett.* **1989**, *157*, 479.
[3] Řezáč, J.; Hobza, P. *J. Chem. Theor. Comput.* **2013**, *9*, 2151.
[4] (a) Vahtras, O.; Almlöf, J.; Feyereisen, M. W. *Chem. Phys. Lett.* **1993**, *213*, 514. (b) Van Alsenoy, C. *J. Comput. Chem.* **1988**, *9*, 620.
[5] (a) Kats, D.; Korona, T.; Schütz, M. *J. Chem. Phys.* **2007**, *127*, 064107. (b) Korona, T.; Jeziorski, B. *J. Chem. Phys.* **2008**, *128*, 144107. (c) Boström, J.; Pitoňák, M.; Aquilante, F.; Neogrády, P.; Pedersen, T. B.; Lindh, R. *J. Chem. Theor. Comput.* **2012**, *8*, 1921. (d) DePrince, A. E.; Sherrill, C. D. *J. Chem. Theor. Comput.* **2013**, *9*, 2687. (e) Epifanovsky, E.; Zuev, D.; Feng, X.; Khistyaev, K.; Shao, Y.; Krylov, A. I. *J. Chem. Phys.* **2013**, *139*, 134105. (f) DePrince, A. E.; Kennedy, M. R.; Sumpter, B. G.; Sherrill, C. D. *Mol. Phys.* **2014**, *112*, 844. (g) Lesiuk, M. *J. Chem. Phys.* **2020**, *152*, 044104. (h) Bozkaya, U.; Sherrill, C. D. *J. Chem. Phys.* **2017**, *147*, 044104. (i) Shen, T.; Zhu, Z.; Zhang, I. Y.; Scheffler, M. *J. Chem. Theor. Comput.* **2019**, *15*, 4721.
[6] (a) Harvey, M. J.; De Fabritiis, G. *WIREs: Comput. Mol. Sci.* **2012**, *2*, 734. (b) Walker, R. C.; Götz, A. W. *Electronic Structure Calculations on Graphics Processing Units*. (Wiley Online Library, **2016**). (c) Bao, J. Z.; Feng, X. T.; Yu, J. G. *Acta Phys. Chim. Sin.* **2011**, *27*, 2019 (in Chinese). (鲍建樟, 丰鑫田, 于建国, 物理化学学报, **2011**, *27*, 2019.)
[7] (a) Stopper, D.; Roth, R. *J. Chem. Phys.* **2017**, *147*, 064508. (b) Nitsche, M. A.; Ferreria, M.; Mocskos, E. E.; Gonzalez Lebrero, M. C. *J. Chem. Theor. Comput.* **2014**, *10*, 959. (c) Jia, W.; Fu, J.; Cao, Z.; Wang, L.; Chi, X.; Gao, W.; Wange, L. W. *J. Comput. Phys.* **2013**, *251*, 102. (d) Genovese, L.; Ospici, M.; Deutsch, T.; Mehaut, J. F.; Neelov, A.; Goedecker, S. *J. Chem. Phys.* **2009**, *131*, 034103.
[8] (a) Tornai, G. J.; Ladjanszki, I.; Rak, A.; Kis, G.; Cserey, G. *J. Chem. Theor. Comput.* **2019**, *15*, 5319. (b) Rak, A.; Cserey, G. *Chem. Phys. Lett.* **2015**, *622*, 92. (c) Asadchev, A.; Gordon, M. S. *J. Chem. Theor. Comput.* **2012**, *8*, 4166. (d) Wang, Y.; Tian, Y. Q.; Jin, Z.; Suo, B. B. *Acta Chim. Sinica* **2021**, *79*, 653 (in Chinese). (王岩, 田英齐, 金钟, 索兵兵, 化学学报, **2021**, *79*, 653.)
[9] (a) Bykov, D.; Kjaergaard, T. *J. Comput. Chem.* **2017**, *38*, 228. (b) Angel Martinez-Martinez, L.; Amador-Bedolla, C. *J. Mex. Chem. Soc.* **2017**, *61*, 60. (c) Song, C.; Martinez, T. J. *J. Chem. Phys.* **2016**, *144*, 174111. (d) Maurer, S. A.; Kussmann, J.; Ochsenfeld, C. *J. Chem. Phys.* **2014**, *141*, 051106.
[10] Fales, B. S.; Levine, B. G. *J. Chem. Theor. Comput.* **2015**, *11*, 4708.
[11] Fales, B. S.; Curtis, E. R.; Johnson, K. G.; Lahana, D.; Seritan, S.; Wang, Y.; Weir, H.; Martínez, T. J.; Hohenstein, E. G. *J. Chem. Theor. Comput.* **2020**, *16*, 2021.
[12] Knizia, G.; Li, W.; Simon, S.; Werner, H. J. *J. Chem. Theor. Comput.* **2011**, *7*, 2387.
[13] Pokhilko, P.; Epifanovsky, E.; Krylov, A. I. *J. Chem. Theor. Comput.* **2018**, *14*, 4088.
[14] Wang, Z.; Guo, M.; Wang, F. *Phys. Chem. Chem. Phys.* **2020**, *22*, 25103.
[15] Knizia, G.; Adler, T. B.; Werner, H.-J. *J. Chem. Phys.* **2009**, *130*, 054104.
[16] (a) Guo, Y.; Riplinger, C.; Becker, U.; Liakos, D. G.; Minenkov, Y.; Cavallo, L.; Neese, F. *J. Chem. Phys.* **2018**, *148*, 011101. (b) Liakos, D. G.; Guo, Y.; Neese, F. *J. Phys. Chem. A* **2020**, *124*, 90.
[17] Gauss, J.; Stanton, J. F.; Bartlett, R. J. *J. Chem. Phys.* **1991**, *95*, 2623.
[18] (a) Sun, Q.; Berkelbach, T. C.; Blunt, N. S.; Booth, G. H.; Guo, S.; Li, Z.; Liu, J.; McClain, J. D.; Sayfutyarova, E. R.; Sharma, S.; Wouters, S.; Chan, G. K.-L. *WIREs Comput. Mol. Sci.* **2018**, *8*, 1340. (b) Sun, Q.; Zhang, X.; Banerjee, S.; Bao, P.; Barbry, M.; Blunt, N. S.; Bogdanov, N. A.; Booth, G. H.; Chen, J.; Cui, Z.-H.; Eriksen, J. J.; Gao, Y.; Guo, S.; Hermann, J.; Hermes, M. R.; Koh, K.; Koval, P.; Lehtola, S.; Li, Z.; Liu, J.; Mardirossian, N.; McClain, J. D.; Motta, M.; Mussard, B.; Pham, H. Q.; Pulkin, A.; Purwanto, W.; Robinson, P. J.; Ronca, E.; Sayfutyarova, E. R.; Scheurer, M.;

Schurkus, H. F.; Smith, J. E. T.; Sun, C.; Sun, S.-N.; Upadhyay, S.; Wagner, L. K.; Wang, X.; White, A.; Whitfield, J. D.; Williamson, M. J.; Wouters, S.; Yang, J.; Yu, J. M.; Zhu, T.; Berkelbach, T. C.; Sharma, S.; Sokolov, A. Y.; Chan, G. K.-L. *J. Chem. Phys.* **2020**, *153*, 024109.
[19] Dunning, T. H. *J. Chem. Phys.* **1989**, *90*, 1007.
[20] Weigend, F. *Phys. Chem. Chem. Phys.* **2002**, *4*, 4285.
[21] (a) Peterson, K. A.; Figgen, D.; Goll, E.; Stoll, H.; Dolg, M. *J. Chem. Phys.* **2003**, *119*, 11113. (b) Peterson, K. A.; Shepler, B. C.; Figgen, D.; Stoll, H. *J. Phys. Chem. A* **2006**, *110*, 13877.
[22] Figgen, D.; Rauhut, G.; Dolg, M.; Stoll, H. *Chem. Phys.* **2005**, *311*, 227.
[23] Stoychev, G. L.; Auer, A. A.; Neese, F. *J. Chem. Theor. Comput.* **2017**, *13*, 554.
[24] Peverati, R.; Truhlar, D. G. *Philos. T. R. Soc. A* **2014**, *372*, 20120476.
[25] Werner, H.-J.; Knowles, P. J.; Manby, F. R.; Black, J. A.; Doll, K.; Heßelmann, A.; Kats, D.; Köhn, A.; Korona, T.; Kreplin, D. A.; Ma, Q.; Miller, T. F.; Mitrushchenkov, A.; Peterson, K. A.; Polyak, I.; Rauhut, G.; Sibaev, M. *J. Chem. Phys.* **2020**, *152*, 144107.
[26] Neese, F. *WIREs Comput. Mol. Sci.* **2012**, *2*, 73.

(Cheng, B.)

