Exploiting graphical processing units to enable quantum chemistry calculation of large solvated molecules with conductor-like polarizable continuum models

Exploiting Graphical Processing Units to Enable Quantum Chemistry Calculation of Large Solvated Molecules with Polarizable Continuum Models

Fang Liu,[1,2,3] David M. Sanchez,[1,2] Heather J. Kulik,[3] Todd J. Martínez[*,1,2]

[1]*Department of Chemistry and The PULSE Institute, Stanford University, Stanford, CA 94305*

[2]*SLAC National Accelerator Laboratory, Menlo Park, CA 94025*

[3] *Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA*

Abstract: The conductor-like polarizable continuum model (C-PCM) with switching/Gaussian smooth discretization is a widely used implicit solvation model in quantum chemistry. We have previously implemented C-PCM solvation for Hartree-Fock (HF) and density functional theory (DFT) on graphical processing units (GPUs), enabling the quantum mechanical treatment of large solvated biomolecules. Here, we first propose a GPU-based algorithm for the PCM conjugate gradient linear solver that greatly improves the performance for very large molecules. The overhead for PCM related evaluations now consumes less than 15% of the total runtime for DFT calculations on large molecules. Second, we demonstrate that our algorithms tailored for ground state C-PCM are transferrable to excited state properties. Using a single GPU, our method evaluates the analytic gradient of the linear response PCM time-dependent density functional theory (TDDFT) energy up to 80X faster than a conventional central processing unit (CPU)-based implementation. In addition, our C-PCM algorithms are transferable to other methods that require electrostatic potential evaluations. For example, we achieve speedups of up to 130x for restricted electrostatic potential (RESP) based atomic charge evaluations, compared to CPU-based codes. We also summarize and compare the different PCM cavity discretization schemes used in some popular quantum chemistry packages as a reference for both users and developers.

1. Introduction

The evolution of computational hardware and algorithms has enabled quantum mechanical (QM) studies for large molecules, including not only static properties but also dynamic processes. This has led to fully QM calculations of biomolecular dynamics in solvated environments.[1] Implicit solvent models based on a dielectric continuum approximation are an attractive conceptual framework to describe solvent effects within a QM approach due to their high efficiency.[2, 3] Methods such as the polarizable continuum model[4] (PCM) and its variants such as conductor-like models (COSMO,[5-7] C-PCM[8], also known as GCOSMO[9], and IEF-PCM[10-12]) are the most popular and accurate of these implicit solvent models in the context of *ab initio* quantum chemistry.

In early implementations of PCM methods, geometry optimization and molecular dynamics were troublesome, as most models led to discontinuous gradients of the solvation energy. These discontinuities were caused by the discretization of the cavity surface: apparent surface charges (ASCs) at the boundary of two adjacent atoms were abruptly switched on and off (exposed or buried in the cavity surface), making the solvated potential energy surface non-smooth. This problem has recently been overcome by using improved discretization schemes and Gaussian smearing of the ASCs.[13-15]

Unfortunately, the application of PCM in QM calculations of large-scale biomolecular systems can still be limited by CPU bottlenecks.[16] On the one hand, calculation of the solvent-solute Coulomb interaction involves one-electron integrals (OEI), with effort scaling as $O(MN^2)$, where M and N denote the number of ASCs and basis functions, respectively. For small proteins with hundreds of atoms, the typical values of M and N are of the order of 10^4 and 10^3, respectively. On the other hand, the most straightforward algorithm (direct inversion) for solving the linear equations for PCM requires $O(M^3)$ time and $O(M^2)$ memory. Many efforts have aimed to make PCM methods more efficient. Fast summation techniques like the fast multipole method[17, 18] (FMM) were introduced to PCM[19] to reduce the cost of evaluating Coulomb interactions. Recently, an elegant domain decomposition strategy (ddCOSMO)[20-24] has been proposed as a linear scaling approach to COSMO and implemented with semiempirical methods.

Recently, we presented a completely different strategy to solve the C-PCM problem for large molecules.[25] We exploited the high parallelizability of GPUs to accelerate the evaluation of PCM-related OEIs. The linear equations were solved with a highly efficient Randomized Block-Jacobi (RBJ) preconditioner for conjugate gradient (CG),[26] and we introduced a dynamic CG convergence threshold to greatly reduce the number of CG iterations without affecting the final accuracy. Since the publication of that work,[25] we now also build the matrix-vectors products on-the-fly on the GPUs, greatly reducing the memory requirement for large molecules. With these techniques, we are able to perform C-PCM HF[27] or DFT[28] calculations of energies and gradients for any molecule that is tractable in isolation. The GPU-based PCM-related OEI builds and linear solver techniques are the foundation for accelerating PCM calculations in more advanced quantum chemistry methods, such as TDDFT.[29-32] The algorithms are also transferrable to other methods that require evaluation of the electrostatic potential (ESP), as is often used for determining atomic charges.[33-37]

The article is organized as follows: 1) We briefly review the basic formalism of C-PCM, summarizing the different cavity discretization schemes implemented in various quantum chemistry packages. 2) We then explain the necessity of accelerating C-PCM quantum chemistry calculations with GPUs, and briefly review the strategies we used. 3) We introduce recent improvements to the linear solver for very large molecules. 4) Then we demonstrate the transferability of our algorithms to other quantum chemistry methods and benchmark their performance. 5) Finally, we conclude with comments about future extensions of GPU-accelerated PCM.

2. Basic Formalism

2.1 Conductor-like Polarizable Continuum Model

Prior to explaining the basics of the C-PCM method, it is useful to review key notation used in this article: atomic orbitals (AOs) will be indexed with μ,ν,κ,γ and τ. Molecular orbitals (MOs) will be indexed with *i, j* for occupied orbitals, and *a, b* for virtual orbitals. Primitive Gaussian basis functions will be indexed with χ_p, χ_q. We will use brackets to denote integrals over primitive basis functions and parentheses to denote

integrals for contracted basis functions. Energies will be differentiated with respect to an external perturbation (e.g., the nuclear coordinate or an external field) denoted as ξ.

The basic idea of C-PCM is shown schematically in Figure 1. The reaction potential generated by the solute charge distribution is described in terms of an apparent charge distribution spread over the solute cavity surface. Outside the surface is a dielectric continuum with dielectric constant ε_2, whereas the cavity containing the solute has unit dielectric constant. For numerical convenience, the cavity boundary is usually discretized into *M* surface elements (tesserae). Each of these tesserae has an ASC, q_k, which describes the electric field of the polarized continuum. The values, **q**, of the ASCs are determined with a set of linear equations:[2, 14]

$$\mathbf{Kq} = \mathbf{RV} \tag{0}$$

where **K** and **R** depend on the particular PCM method, and represent geometric information concerning the cavity and the dielectric screening, respectively, and **V** is a vector that represents the solute electrostatic potential at each tesserae. Then the electrostatic component of the solvation free energy is given by the interaction between the polarization charges and solute, in addition to the self-energy of the surface charges. In the context of a self-consistent field (SCF) calculation, it is given as

$$\Delta G_{\text{solv}} = \frac{1}{2}\mathbf{V}^{\dagger}\mathbf{q} \tag{0}$$

In a PCM calculation of a QM solute molecule, two quantities must be determined: the polarization charges **q** and the solute electron density matrix **P**. In the simplest case, assuming closed shell HF or KS calculations, the solute density matrix is computed as:

$$\mathbf{F}^{\text{solvated}}\mathbf{C} = \mathbf{SC}\epsilon \tag{0}$$

$$P_{\mu\nu} = 2\sum_{i=1}^{\text{occ}} C_{\mu i}C_{\nu i}^{*} \tag{0}$$

where $\mathbf{F}^{\text{solvated}}$ is the Fock matrix of the solvated system, **C** are the MO coefficients, and **S** is the AO overlap matrix.

The solvent contributes to $\mathbf{F}^{\text{solvated}}$ through terms containing cavity charges $\mathbf{q}$

$$\mathbf{F}^{\text{solvated}} = \mathbf{F}^{0} + \Delta\mathbf{F}^{S}(\mathbf{q}) \tag{0}$$

where $\mathbf{F}^{0}$ is the gas phase Fock matrix, and $\Delta\mathbf{F}^{\text{S}}(\mathbf{q})$ is given as:

$$\Delta\mathbf{F}^{\text{S}}(\mathbf{q}) = \sum_{k=1}^{M} q_k L_{\mu\nu}^{k} \tag{0}$$

$$L_{\mu\nu}^{k} == -\int \hat{J}_{k}^{screened} \phi_{\mu}(\vec{r}) \phi_{\nu}(\vec{r}) dr \tag{0}$$

where $L_{\mu\nu}^{k}$ represents the electrostatic potential of the basis pair, $\phi_{\mu}(r)\phi_{\nu}(r)$ at tessera *k,* and the exact form of the operator $\hat{J}_{k}^{screened}$ depends on the cavity discretization scheme. For example, in the crudest treatment, when the ASCs are represented by point charges,

$$\hat{J}_{k}^{screened} = \frac{1}{|\mathbf{r} - \mathbf{r}_k|} \tag{0}$$

and $L_{\mu\nu}^{k}$ has the same form as the usual nuclear-electron attraction integrals.[38]

To determine $\mathbf{q}$, one can construct the ESP, V_k, for the *k*th tessera from contributions of the nuclei and solute electron density:

$$V_k = V_k^{\text{nuc}} + V_k^{\text{e}} \tag{0}$$

$$V_k^{\text{nuc}} = \sum_{J=1}^{n_A} Z_J \int \hat{J}_{k}^{\text{screened}} \delta(\mathbf{r} - \mathbf{R}_J) dr \tag{0}$$

$$V_k^{\text{e}} = \sum_{\mu\nu} P_{\mu\nu} L_{\mu\nu}^{k} \tag{0}$$

where $\mathbf{V}^{\text{nuc}}$ and $\mathbf{V}^{\text{e}}$ are the ESP contributions from the solute nuclei and electrons, respectively; $\mathbf{R}_J$ is the location of the *J*th nucleus with atomic radius R_J. The construction

of $\mathbf{V}^e$ is very similar to $\Delta\mathbf{F}^S(\mathbf{q})$, but the integrals $\left(L_{\mu\nu}^k\right)$ are contracted over basis function pairs (μ and ν) rather than tesserae (k).

Therefore, in a PCM-SCF iteration, the computationally intensive steps include three parts:

(1) constructing $\mathbf{V}^e$, i.e. Eq. (0);

(2) solving the linear system in Eq. (0);

(3) constructing $\Delta\mathbf{F}^s$, i.e. Eq. (0)

We will revisit how to reduce the computational cost in these steps in Section 2.3 and Section 4.

2.2 Comparison of Cavity Surface Discretization Schemes

There exist many versions of PCM in different quantum chemistry packages that give slightly different results, which can sometimes cause confusion. Here we briefly summarize the similarities and differences of these implementations.

The first difference lies in the integral equations, as reflected in the detailed formulas for both the **K** and **R** matrices in Eq. 1. This distinguishes the conductor-like models from other PCM models such as SS(V)PE[39] or IEF-PCM.[10-12] For further detail, see the detailed discussion by Lange and Herbert.[14]

The second difference lies in the cavity surface discretization scheme. Even with the same model, e.g. C-PCM, there still exist many variations in different quantum chemistry packages that are distinguished only by the discretization scheme, as embodied by the **K** matrix in Eq. (0). To define a cavity surface discretization scheme, two elements are needed: cavity type (defining the shape of the molecular surface) and tessellation scheme (discretization of the molecular surface).

Cavity type: Most cavity types fall into one of the following three categories: (1) the van de Waals cavity; (2) the solvent accessible surface (SAS);[5] or (3) the solvent excluded surface (SES or Connolly Surface[40]) which are shown schematically in Figure 2. It is worth noting that in the literature, the same terminology may refer to different types of cavities. For instance, SAS was originally defined as shown in Figure 2d.[5] The fused vdW surface (Figure 2b, scaling factor α=1) is first expanded by the solvent radius (Figure 2c) and then contracted by a distance δ^{SC}, leading to a cavity

with some creases without tesserae at the boundary between atoms. However, in more recent publications,[13,14] SAS refers to Figure 2b, a fused vdW surface with expanded atomic radii (α>1). The SES removes cusps at boundaries by introducing "probe spheres" (see Figures 2e-f). In our implementation, we always employ the modern definition of SAS (Figure 2b).

Tessellation scheme: There are more ways to tessellate than types of cavity construction. In total, there are two important aspects regarding tessellation. 1) How is the surface charge distribution on each tessera represented? 2) How is the molecular surface discretized into small segments? For the first aspect, one could represent the charge distribution as a point charge or a Gaussian distribution centered on the tessera. This choice, combined with the techniques defining exposure and burial of the tesserae, determines the continuity of the analytic gradient of the solvation energy. For the second aspect, the discretization methods can be grouped into two categories: polyhedron based and Lebedev grid[41] based. The original COSMO model by Klamt generates the grid points on a sphere from iterative refinement of triangles starting from an icosahedron.[5] Although developed independently, GEPOL[42-45] has a similar procedure for polyhedron refinement, and is still widely used in modern implementations of C-PCM methods in popular electronic structure packages, such as GAMESS-US,[46, 47] GAUSSIAN,[48] and ORCA.[49] The Lebedev grid, which is often used for numerical quadrature in DFT,[50, 51] was introduced to solvent surface discretization by Karplus and coworkers.[13] The Lebedev grid benefits from its advantages in numerical quadrature stability and ease of construction compared to the polyhedron method. It is used in more recent implementations of C-PCM, for example, in Q-CHEM[14, 52, 53] and TERACHEM.[26-28,34,50] The choices of charge representation and surface discretization are independent and can be combined in different ways. For example, a surface discretized with GEPOL can either be represented by point charges, as implemented in GAMESS-US,[42-45, 54] or by Gaussian-smeared distributions, as implemented in GAUSSIAN.[15] Similarly, the Lebedev grid can be either used in the SWIG/ISWIG scheme to obtain continuous energy gradients, or it can be used with point charges associated with fixed tesserae area. Both of these methods are implemented in Q-CHEM and TERACHEM. A summary of different cavity discretization schemes used in some popular quantum chemistry packages is provided in Table 1.

In the following sections, we focus on the SWIG/ISWIG implementation of C-PCM, which benefits from a simple discretization scheme and a smooth analytical gradient of solvation energy. In this context, **R** in Eq. (0) is a scalar

$$\mathbf{R} = -\frac{\varepsilon - 1}{\varepsilon} \tag{0}$$

and the **K** matrix is given as:

$$K_{kl} = \frac{\mathrm{erf}(\zeta'_{kl} \mid \vec{r}_k - \vec{r}_l \mid)}{\mid \vec{r}_k - \vec{r}_l \mid} \quad \forall k \neq l \tag{0}$$

$$K_{kk} = \frac{\zeta_k}{\sqrt{2\pi}} \mathbf{S}_k^{-1} \tag{0}$$

where $\vec{r}_k$ is the location of the *k*th Lebedev point, $\zeta_k = \frac{\zeta}{R_{J(k)}\sqrt{w_k}}$ is the Gaussian exponent for the *k*th point where ζ is an optimized exponent for the specific Lebedev quadrature level being used (as tabulated[13] by York and Karplus), w_k is the Lebedev quadrature weight for the *k*th point, and $R_{J(k)}$ is the radius of the atom (J) associated with the *k*th point. Finally, $\zeta'_{kl} = \frac{\zeta_k \zeta_l}{\sqrt{\zeta_k^2 + \zeta_l^2}}$ is the combined exponent that emerges when describing the Coulomb interaction of two Gaussian charge distributions. The matrix $\mathbf{S}_k$ in Eq. (0) is the switching function which smoothes the boundary of adjacent vdW spheres corresponding to different atoms (and thus makes the solvation energy continuous), and is a function of only the positions of the PCM grid points and solute nuclei. Using Gaussian distributions to represent the ASC, the operator $\hat{J}_k^{screened}$ for the *k*th tessera has the form

$$\hat{J}_k^{screened} = \frac{\mathrm{erf}(\zeta_k \mid \vec{r} - \vec{r}_k \mid)}{\mid \vec{r} - \vec{r}_k \mid} \tag{0}$$

and thus

$$L_{\mu\nu}^{k} = -\int \phi_{\mu}(\vec{r}) \frac{\mathrm{erf}(\zeta_k |\vec{r} - \vec{r}_k|)}{|\vec{r} - \vec{r}_k|} \phi_{\nu}(\vec{r}) dr \tag{0}$$

$$V_{k}^{\mathrm{nuc}} = \sum_{J=1}^{n_A} Z_J \frac{\mathrm{erf}(\zeta_k |\vec{r}_k - \vec{R}_J|)}{|\vec{r}_k - \vec{R}_J|} \tag{0}$$

where Eq. (0) is a generalized version of the nuclear-electron attraction integral, and is sometimes referred to as the Coulomb attenuated nuclear-electron attraction integral. With SWIG/ISWIG, smooth analytical gradients are available for the solvated SCF energy gradient with respect to the nuclear coordinates ξ. The PCM contribution to the gradient can be obtained by differentiating Eq. (0):

$$\begin{aligned} \Delta G_{\mathrm{solv}}^{(\xi)} &= \frac{1}{2}(\mathbf{V}^{\dagger}\mathbf{q})^{(\xi)} = \frac{1}{2}(\mathbf{R}\mathbf{V}^{\dagger}\mathbf{K}^{-1}\mathbf{V})^{(\xi)} \\ &= \mathbf{q}^{\dagger}\mathbf{V}^{(\xi)} - \frac{1}{2}\mathbf{R}^{-1}\mathbf{q}^{\dagger}\mathbf{K}^{(\xi)}\mathbf{q} \end{aligned} \tag{0}$$

In the derivation we utilize Eqs. (0), (0) and the equation for the derivative of the inverse of a matrix: $(\mathbf{K}^{-1})^{(\xi)} = -\mathbf{K}^{-1}\mathbf{K}^{(\xi)}\mathbf{K}^{-1}$

2.3 Need for GPU Acceleration of PCM

GPUs are characterized as stream processors, and are especially suitable for massively data-parallel computing. To take the most advantage of CPU/GPU heterogeneous programming in quantum chemistry calculations, there are two important considerations: (1) which parts are most time-consuming; (2) which parts are most "stream-friendly", or benefit most from highly parallelizable architectures. For typical molecules with up to ~15,000 basis functions at the HF/DFT level of theory, building the electron repulsion integrals (ERI) is the most computationally expensive part. However, this cost can be dramatically decreased if we take advantage of the high ratio of compute to data transfer in constructing the ERIs. Therefore, it is a common approach for many developers to accelerate just the ERIs on GPUs and leave the less costly portions of the calculations, e.g. OEIs and linear algebra, on CPUs.[55-59]

As mentioned above, the three computationally intensive steps are: 1) constructing V^e, 2) solving the linear system in Eq. (1), and 3) computing $\Delta \mathbf{F}^s$. Steps 1) and 3) involve OEIs only, and step 2) is linear algebra. One might think that there was no need to accelerate these PCM components on GPUs, since the cost of the SCF is dominated by the evaluation of two-electron repulsion integrals (ERI). However, in large systems the PCM-related OEIs become comparable in cost to ERIs, and GPU acceleration becomes greatly needed. Figure 3 shows the percentage of time taken by OEI evaluations on CPUs and GPUs for the PCM calculation of a peptide. If only the ERIs are accelerated with GPUs, while the PCM related evaluations remain on CPUs, the PCM portion will take 88% of the total runtime. Therefore, one should expect a considerable speed-up if PCM evaluations are also accelerated with GPUs.

3. Methods

Our GPU accelerated PCM and related methods have been implemented within TERACHEM.[57-59] The excited state C-PCM implementation is currently available in a development version of TERACHEM. All other C-PCM functionalities discussed in this article are available in the commercial release v1.9. Timings for this program have been obtained using an Intel Xeon E5-2680 v2 CPU clocked at 2.80GHz with a NVIDIA Tesla P100 GPU. For comparison, ground and excited state PCM calculations have been conducted with the CPU-based electronic structure package Q-CHEM,[14, 52, 53] which uses the same cavity definition and PCM scheme. The restrained ESP (or RESP) charge[60, 61] calculations have been conducted with another CPU based electronic structure package GAMESS-US,[46, 47] since it has the same definition of molecular electrostatic potential (MEP)[33-37] points as TERACHEM. Timings of Q-CHEM and GAMESS-US have been obtained using the faster Intel Xeon E5-2643 CPU clocked at 3.30GHz.

For a performance test of the ground state PCM implementation, we use microsolvated trans-penta-2,4-dieniminium cation ($C_5H_8N^+$, trans-PSB3),[62] including up to 186 explicit water molecules (and a dielectric continuum surrounding the microsolvated cluster). Molecular geometries for the microsolvated clusters were obtained using the Amber 12 simulation package[63] by taking snapshots from a classical molecular dynamics simulation. These dynamics simulations used the general AMBER

force field (GAFF)[64, 65] to describe the solute molecule and the three-site transferable intermolecular potential (TIP3P)[66] to describe water molecules. In the PCM calculations, the ωPBEh[67] exchange-correlation functional was used with the 6-31G* basis set.[68] For Fock matrix diagonalization, the Matrix Algebra on GPU and Multicore Architectures (MAGMA)[69-71] library is used for matrices larger than 1500×1500. Smaller Fock matrices are diagonalized with Intel MKL[50] on the CPU.

For performance tests of linear response (LR)[72-74] PCM for TDDFT/TDA, we examine a series of microsolvated umbelliferone molecules with explicit water molecules in a sphere with radius ranging from 2-10Å. The TDDFT/TDA energy and analytical gradient calculations were conducted with the 6-31G* basis set and the ωPBEh[67] functional, which is widely used in TDDFT calculations in order to avoid spurious low-lying charge transfer states.

For performance tests of RESP, the SCF and RESP fitting is conducted on the same set of benchmark molecules for LR-PCM. The HF/6-31G* level of theory is used, since this choice has been made in the development of many popular force fields[64, 65, 75-77] based on RESP charges. The MEP grid used for fitting the charges is built from 4 layers of Connolly surfaces with radii scaling factor starting from 1.4 and incrementing by 0.2 per layer, with a grid density of 5.0 points/Å^2.

4. Results and Discussion

4.1 Acceleration of C-PCM on GPUs

In previous work,[25] we introduced a GPU-accelerated SCF-PCM code where the SCF iterations were accelerated by: (1) computing the PCM related OEIs with a parallel algorithm on GPUs and (2) using a dynamic CG threshold with a RBJ preconditioner. In addition, we have now developed an implementation where the matrix-vector products are built on GPUs without explicit storage of the matrices. This further enhanced the performance for large systems.

Benchmarking results for C-PCM single point energy evaluation are shown in Figure 4. The total run time includes both gas phase DFT evaluations and the PCM terms. The observed scaling of $O(N^{1.63})$ is very similar to the scaling of the gas phase counterpart. The scaling of the PCM integrals and the linear solver are sub-quadratic and quadratic,

respectively. Based on Eq. (0), the total number of elementary integrals scales as $O(MN^2)$, as discussed in section 1. Since M and N both grow linearly with the number of atoms for these benchmarking systems (as shown in Figure 4b), the PCM OEI might be expected to scale as $\sim O(N^3)$. However, our implementation is observed to scale as $O(N^{1.58})$ for medium/large sized molecules (with more than 250 atoms and 1500 basis sets). The scaling gains come from exploiting sparsity. The OEI in Eq. (0) are calculated as summations of OEIs over primitive basis functions

$$L_{\mu\nu}^{k} = \sum_{p=1}^{l_\mu}\sum_{q=1}^{l_\nu} c_{\mu p} c_{\nu q} [\chi_p \mid \hat{J}_k^{screened} \mid \chi_q] \tag{0}$$

where $c_{\mu p}$ is the contraction coefficient. When we contract $L_{\mu\nu}^{k}$ to build the Fock matrix contribution $\Delta\mathbf{F}^{S}(\mathbf{q})$ in Eq. (0), only significant primitive pairs ($\chi_p\chi_q$) are included, as determined by using a Schwartz-like bound[78] with a cutoff, ε^{screen}, of 10^{-12} atomic units:

$$[\chi_p \mid \hat{J}_k^{screened} \mid \chi_q] \approx 0 \quad \forall\, [\chi_p\chi_q \mid \chi_p\chi_q]^{1/2} < \varepsilon^{screen} \tag{1}$$

When building the ESP, V_k^{e} in Eq. (0), the screening criteria is augmented with the corresponding density matrix element:

$$[\chi_p \mid \hat{J}_k^{screened} \mid \chi_q] \approx 0 \quad \forall \left|P_{\mu\nu}\right| [\chi_p\chi_q \mid \chi_p\chi_q]^{1/2} < \varepsilon^{screen} \tag{2}$$

In both cases, only the surviving pair quantities are preloaded to the GPU global memory and participate in the integral calculation.

For small systems, the observed scaling looks even better, but it is mainly because the GPU cores are not saturated with tasks for very small systems. Our linear solver is observed to scale as $O(N^{1.92})$, close to the expected $O(N^2)$ scaling of matrix-vector multiplication in CG. With the acceleration of a single GPU, the C-PCM ωPBEh/6-31G* calculation for the largest system studied in this work (707 atoms, 4495 basis functions) can be completed within 8 minutes.

We previously showed that the GPU-accelerated RBJ preconditioned conjugate gradient method achieved significant speed-ups over both direct inversion and Jacobi preconditioned CG on CPUs. However, as our previous algorithm explicitly stored the **K** matrix in memory, it required $O(M^2)$ memory which becomes prohibitive for molecules

with M > 10^4 (ca. 200 atoms assuming a Lebedev grid density of 110 pts/atom). To improve upon this, we implemented the matrix-vector product, **Kq,** on GPUs without storing the **K** matrix. This is accomplished in two steps: 1) Pre-computation (on the CPU) of the diagonal elements $\mathbf{K}_{kk}$ based on Eq. (0) 2) Construction of the off-diagonal elements $\mathbf{K}_{kl}$ on the GPU simultaneously with evaluation of the matrix-vector product **Kq**, as shown schematically in Figure 5. In detail, each thread is assigned a tessera, i ($\mathbf{r}_i$, ζ_i), and accumulates contributions to the ESP at this tessera from all other tesserae: $q_j \mathrm{erf}(\zeta'_{ij} |\mathbf{r}_i - \mathbf{r}_j|)/|\mathbf{r}_i - \mathbf{r}_j|$ for $i \neq j$. We pre-load $(\zeta_j, \mathbf{r}_j)$ to shared memory in a coalesced pattern to reduce memory traffic. Finally, the **Kq** vector is copied back to the CPU and the contributions from diagonal elements of **K** are added in (on the CPU). The whole process requires only O(M) memory on both GPU and CPU. The standard Jacobi preconditioner is used in this algorithm for simplicity. However, the RBJ and other preconditioners could also be applied with minor modification.

In Figure 6, we compare the GPU-based CG solver with the CPU based RBJ CG solver. For small molecules (less than ≈32 atoms with grid density of 110 points/atom), the overhead for CPU-GPU data transfer is relatively significant and the CPU based solver is slightly more efficient (Figure 6a). For larger molecules, the GPU solver is significantly faster than the CPU version. Furthermore, we can take advantage of efficient single-precision arithmetic on GPUs, which can significantly accelerate the calculation. For our tests on P100 GPUs, the CG runtime is reduced by about half for large systems. Larger gains will be achieved when using commodity cards that have poor double precision support. Our benchmark data shows that the use of single-precision for the GPU calculation of off-diagonal contributions does not introduce significant errors to the total energy of the solvated system (Figure 6c). In this case, we use double precision for the diagonal elements that are calculated on the CPU. Thus, what we call "single precision" here is actually a mixed-precision scheme. Because of this, the absolute errors in both the total energy and solvation energy introduced by single precision arithmetic are always less than 10^{-5} Hartree even for large molecules with thousands of basis functions and tens of thousands of tesserae. Thus, CG implemented with mixed GPU single precision/CPU double precision will usually be preferred. Due to the significant acceleration achieved in

CG, the ratio of time spent on PCM related evaluations (PCM integrals and CG) relative to total time is greatly reduced (Figure 6d). With the CPU-based CG solver, PCM takes 25%-38% of the total runtime and this percentage increases with system size. The GPU-based mixed-precision solver decreases this ratio to less than 15% for large molecules, significantly outperforming our previously reported[25] typical ratio of 30% for both CPU- and GPU-based implementations.

4.2 Extensions of C-PCM implementation on GPUs: other electronic structure methods

Our C-PCM implementation with GPU acceleration is not only important for ground state HF/DFT, but also lays the foundation for GPU accelerated C-PCM implementation of higher-level quantum chemistry methods. For gas phase quantum chemistry methods, the GPU algorithms for the *J* and *K* builds (Coulomb and exchange operators, respectively)[56-59] are the basis for the implementation of higher-level methods. GPU acceleration of CIS/TDDFT[79] and CASSCF[80, 81] can be easily achieved through reformulation of the methods in terms of *J* and *K* builds. Similarly, for C-PCM, the OEI build ($\Delta\mathbf{F}^{S}$ and V^{e}) and the linear solver play a pivotal role. Implementation of C-PCM in more complicated quantum chemistry methods can be realized by formulation in terms of these fundamental builds. Such high-level methods might include CIS/TDDFT, CASSCF,[82-86] CASPT2,[87, 88] and newly emergent multi-reference DFT methods like REKS.[89-92] We are especially interested in multi-state versions of the last three methods,[93-97] which are able to treat electronic excited and ground states in a balanced way,[98-101] to correctly describe conical intersections.[102] As an example, here we present our GPU accelerated C-PCM implementation of CIS/TDDFT.

To study the excited state properties of solvated molecules, we implemented both the state-specific (SS)[72, 103] and linear response (LR)[72-74] PCM for TDDFT/TDA, with both equilibrium and non-equilibrium[104, 105] solvation. For SS-PCM, the PCM term only enters the ground state SCF iterations, leaving the working equations of TDDFT virtually unchanged. Only minor modifications need to be performed to separate the "fast" and "slow" polarizations,[104, 105] and an external iteration[103, 106] is performed for TDDFT. The LR-PCM implementation is more complicated, as PCM terms directly enter the TDDFT working equations. These terms also change the coupled-perturbed Kohn-Sham

(CPKS)[107-110] equations needed for analytical gradient evaluation. However, they can be reformulated in terms of the PCM OEI builds for both energy and gradient evaluation. The LR-PCM TDDFT/TDA energies are computed by solving Eq. (16), which represents a solvated version of the gas-phase Casida equation, shown here for TDDFT with the Tamm-Dancoff approximation (TDA):[29-32]

$$\mathbf{A}^{\mathrm{sol}}\mathrm{X} = \omega\mathrm{X} \tag{2}$$

The matrix $\mathbf{A}^{\mathrm{sol}}$ is the response matrix of the solvated system and defined as the sum of the gas phase response matrix **A** and a PCM correction term **M**

$$\mathbf{A}^{\mathrm{sol}}_{ia\sigma,jb\sigma'} = \mathbf{A}_{ia\sigma,jb\sigma'} + \mathbf{M}_{ia\sigma,jb\sigma'} \tag{2}$$

$$\mathbf{M}_{ia\sigma,jb\sigma'} = \sum_{kl} L^{k}_{ia\sigma} Q(\epsilon_{\mathrm{lr}})_{kl} L^{l}_{jb\sigma'} \tag{2}$$

Here, $L^{k}_{ia\sigma}$ represents the electrostatic potential at tessera k induced by the charge distribution of the KS orbital pair $\psi_{i\sigma}(\mathbf{r})\psi_{a\sigma}(\mathbf{r})$, and is the MO transform of $L^{k}_{\mu\nu}$ in Eq. (0). The matrix, $Q(\varepsilon) = \mathbf{K}^{-1}\mathbf{R}$, is often called the response matrix,[74] because the action of $Q(\varepsilon)$ on **V** produces the surface charges, **q,** that are induced in response to the electrostatic potential **V.** The linear-response dielectric constant is represented by ϵ_{lr}, which can represent the solvent static dielectric constant for equilibrium solvation, or the optical dielectric constant for non-equilibrium solvation. In the Davidson[111] iteration of TDDFT/TDA, we only evaluate the matrix-vector product of the response matrix with a trial eigenvector, $\mathbf{A}^{\mathrm{solv}}\mathbf{X}$, which leads to a PCM contribution in the AO basis of the form

$$I_{\mu\nu} = \sum_{\gamma\tau} M_{\mu\nu,\gamma\tau} X_{\gamma\tau} = \sum_{\gamma\tau}\sum_{kl} L^{k}_{\mu\nu} Q(\epsilon_{\mathrm{lr}})_{kl} L^{l}_{\gamma\tau} X_{\gamma\tau} \tag{2}$$

This term can be reformulated in terms of ground state PCM builds in the following steps:

(1) Build the ESP induced on the cavity surface by the transition density X

$$V_k^{\mathrm{e,lr}} = \sum_{\gamma\tau} L_{\gamma\tau}^k X_{\gamma\tau} \tag{2}$$

Notice, this step reuses the GPU module for building V^{e} of the ground state.

(2) Solve for surface charges in response to $V_k^{\mathrm{e,lr}}$:

$$\tilde{\mathbf{q}} = \mathbf{Q}(\epsilon_{\mathrm{lr}})\mathbf{V}^{\mathrm{e,lr}} \tag{2}$$

i.e. $\mathbf{K}\tilde{\mathbf{q}}=\mathbf{R}\mathbf{V}^{\mathrm{e,lr}}$ with accelerated CG.

(3) Build a 1-electron Fock-like contribution $I_{\mu\nu} = \sum_k L_{\mu\nu}^k \tilde{q}_k$. Again, this benefits from GPU code reuse, exploiting the module for building $\Delta F_{\mu\nu}^S$.

To evaluate the analytical gradient of the LR-PCM excited state energy,[112, 113] the PCM correction matrix **M** of Eq. (2), and its partial derivatives with respect to nuclear gradient, enter both the CPKS equation and the analytical gradient of the excited state energy. The detailed formulas are available elsewhere.[112, 113] Here we just describe how these terms can be evaluated with the ground state GPU accelerated PCM module. In the CPKS of LR-PCM, all PCM terms are expressed in terms of the matrix-vector product shown in Eq. (2). For the final gradient evaluation, the derivatives of **M** with respect to nuclear coordinates are also needed:

$$\begin{aligned}
&\sum_{\mu\nu,\kappa\lambda} M(\boldsymbol{\varepsilon}_{lr})_{\mu\nu,\kappa\lambda}^{(\xi)} X_{\mu\nu} X_{\kappa\lambda} \\
&= \sum_{\mu\nu,\kappa\lambda}\sum_{kl}\left(L_{\mu\nu}^k\right)^{(\xi)} Q(\epsilon_{\mathrm{lr}})_{kl} L_{\mu\nu}^l X_{\mu\nu} X_{\kappa\lambda} + \sum_{\mu\nu,\kappa\lambda}\sum_{kl} L_{\mu\nu}^k Q(\epsilon_{\mathrm{lr}})_{kl}\left(L_{\mu\nu}^l\right)^{(\xi)} X_{\mu\nu} X_{\kappa\lambda} \\
&\quad + \sum_{\mu\nu,\kappa\lambda}\sum_{kl} L_{\mu\nu}^k Q(\epsilon_{\mathrm{lr}})_{kl}^{(\xi)} L_{\mu\nu}^l X_{\mu\nu} X_{\kappa\lambda} \\
&= 2\left(\sum_k \tilde{\mathbf{q}}^\dagger\left(\mathbf{V}^{\mathrm{e,lr}}\right)^{(\xi)} - \frac{1}{2} R^{-1}\tilde{\mathbf{q}}^\dagger \mathbf{K}^{(\xi)}\tilde{\mathbf{q}}\right)
\end{aligned} \tag{2}$$

The right-hand-side has similar form as the ground state solvation energy gradients in Eq. (0). The only difference is that the ESP, $\mathbf{V}^{\mathrm{e,lr}}$, and the PCM charge vector, $\tilde{\mathbf{q}}$, now correspond to the transition density, as defined in Eq. (2) and (2). Therefore, it can be calculated with the same subroutine used for the ground state solvation energy gradients implementing Eq. (0).

We tested the performance of the GPU based LR-PCM analytical gradient evaluation with a series of microsolvated 7-hydroxycoumarin (Umbelliferone) molecules with explicit water molecules in spheres ranging from 2-10 Å as shown in Figure 7. Umbelliferone ($C_9H_6O_3$, Scheme 1) is a well-known fluorophore with high quantum yield[114-116] and has industrial uses as a sunscreen agent.[117] Its lowest electronic excitation ($S_0 \rightarrow S_1$) is a HOMO-LUMO transition consisting of a $\pi \rightarrow \pi^*$ excitation. The number of basis functions and the number of PCM grid points increase linearly with system size, as shown in Figure 7. Figure 8 shows the total wall time for each calculation of the analytic gradient in microsolvated Umbelliferone. The GPU-based LR-PCM calculation is only negligibly more time-consuming than the GPU-based gas phase TDDFT implementation, and both of these are significantly faster than CPU-based implementations, as demonstrated by comparison with the CPU-based Q-CHEM code. The speed-up compared to the CPU-based Q-CHEM code grows rapidly from about 20x for small molecules to 80x for medium sized molecules. Speed-ups for larger molecules are not shown because the CPU-based calculations are too time-consuming.

4.3 Extensions of C-PCM implementation on GPUs: ESP charge fitting

Apart from the obvious application to solvation models, our GPU-based PCM algorithm also benefits other types of calculations that require evaluation of the ESP. A typical example is ESP-based calculation of atomic charges,[33-37] which are crucial for the development of empirical force fields.[118] The ESP charges are fitted to reproduce the molecular electrostatic potential (MEP)[33-37] at a large number of points defined on three-dimensional surfaces around the molecule of interest. The procedure for building the MEP grid is very similar to building the PCM cavity surface, and evaluating the MEP is equivalent to evaluating V^e in Eq. (0). Due to the similarities between PCM and ESP evaluation, the GPU based PCM OEI modules can be directly adapted for computing the ESP. The MEP grid discretization schemes[35, 40,119-121] are often different from the ones used in PCM, and vary for different types of force field development. For the Amber force fields,[65, 122, 123] the MEP is usually calculated on a series of Connolly surfaces.[35, 40] Multiple layers of Connolly surfaces are built outside the vdW surface of the molecule with a constant increment of the radius (Figure 9). Figure 6 shows how the number of

MEP and PCM grid points increase with molecular size for a representative example. The restrained ESP (or RESP) charge model[60, 61] is a variation of ESP charge fitting developed by Kollman and coworkers. It applies weak hyperbolic restraints to address the ill-behaved ESP charges associated with statistically poorly determined centers ("buried" atoms[124, 125]). In the following discussion we focus on the RESP charge fitting implemented in TERACHEM. However, our MEP evaluation algorithm could also be easily adapted to calculate any atomic charges based on the ESP.

We compared the runtime of RESP charge calculation of TERACHEM and GAMESS on the same set of molecules used for benchmarking LR-PCM calculations in the previous section (Figure 10). With the TERACHEM GPU implementation, the total runtime of RESP (including HF/DFT SCF, MEP evaluation and charge fitting) is only 1.2x-1.4x of the time for a pure SCF run. In other words, the extra cost to obtain RESP charges is almost negligible compared to the preceding SCF calculation. This fast RESP implementation is not only useful for force field parameterization, but also applicable to other methods that require calculation of electrostatic interaction energies. Examples include various embedding methods, as well as fragment-based methods[126] such as the recently developed ab initio exciton model.[127]

5 Conclusions

We have briefly reviewed the formalism of C-PCM and its implementations in many quantum chemistry packages. We demonstrated the necessity of accelerating PCM on GPUs as a result of the GPU acceleration of the solvent-free calculations. Motivated by this, we developed the techniques to accelerate the SWIG/ISWIG version of C-PCM in our previous work. We further demonstrated our recent implementation of matrix-free CG build on GPUs significantly improves both the memory and the time scaling for PCM calculations of very large molecules. The $O(N^{1.63})$ time scaling for C-PCM HF/DFT calculations shows that our C-PCM code is one of the most efficient ones available.

Our GPU algorithms are highly transferrable and enable C-PCM implementation for other quantum chemistry methods besides HF and DFT, by exploiting modular PCM OEI and linear equation solver routines. As an example, we discussed the reformulation of PCM-related terms in LR-PCM TDDFT/TDA energy and gradient evaluations. We achieve speedups compared to conventional CPU-based codes of 20-80X for small to

medium-size molecules. We further demonstrated the transferability of our algorithm to methods other than PCM that require ESP evaluations, with the RESP charge derivation as an example. We achieved speedups compared to CPU-based codes of 20-130X for small to medium-size molecules, and the computational overhead introduced by RESP calculation is less than 50%. This enables efficient atomic charge evaluation for force field development and embedding methods.

In the future, we will extend our acceleration strategies to PCM implementation in multi-reference quantum chemistry methods, such as CASSCF, CASPT2, and REKS. Accelerating PCM in these methods will enable excited state *ab initio* molecular dynamics of molecules in solvent environment, thus shedding light on photochemical mechanisms in biological systems.

6. Acknowledgement

The authors acknowledge support by the Department of Energy under grant number DE-SC0018096 (to H.J.K. and F.L). H.J.K. holds a Career Award at the Scientific Interface from the Burroughs Welcome Fund. TJM is grateful to the Department of Defense (Office of the Assistant Secretary of Defense for Research and Engineering) for a National Security Science and Engineering Faculty Fellowship (NSSEFF). D.M.S. is grateful to the National Science Foundation (NSF) for a graduate fellowship. This work was carried out in part using computational resources from the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562. This work used the XStream computational resource, supported by the National Science Foundation Major Research Instrumentation program (ACI-1429830).

Author Information

Corresponding Author

*E-mail: todd.martinez@stanford.edu

Table 1. Combinations of tessellation schemes (rows) and cavity types (columns) in C-PCM implementations for some representative quantum chemistry software packages. Abbreviations used are: GAMESS (GM), Gaussian (GS), NWChem (NW), ORCA (OR), Q-CHEM (QC) and TERACHEM (TC).

			Scaled VDW/SAS	**SES**
non-smooth gradient	polyhedron	GEPOL-GB[42-45]	GM,OR	GM,OR
		GEPOL-AS[54]	GM	GM
		GEPOL-RT[54]	GM	GM
		Klamt's[5]	TC	
	Lebedev grid	Fixed	QC, TC	QC
smooth gradient	polyhedron	SWIG[13] (Klamt's[5])	NW	
		CSC (GEPOL)[15]	GS	GS
	Lebedev grid	SWIG[13]	TC, QC	QC
		ISWIG[14]	TC,QC	QC
	other	FIXPVA[128]	GM	GM

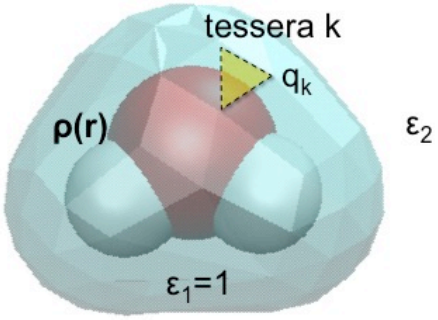


Figure 1: Schematic demonstration of polarizable continuum model.

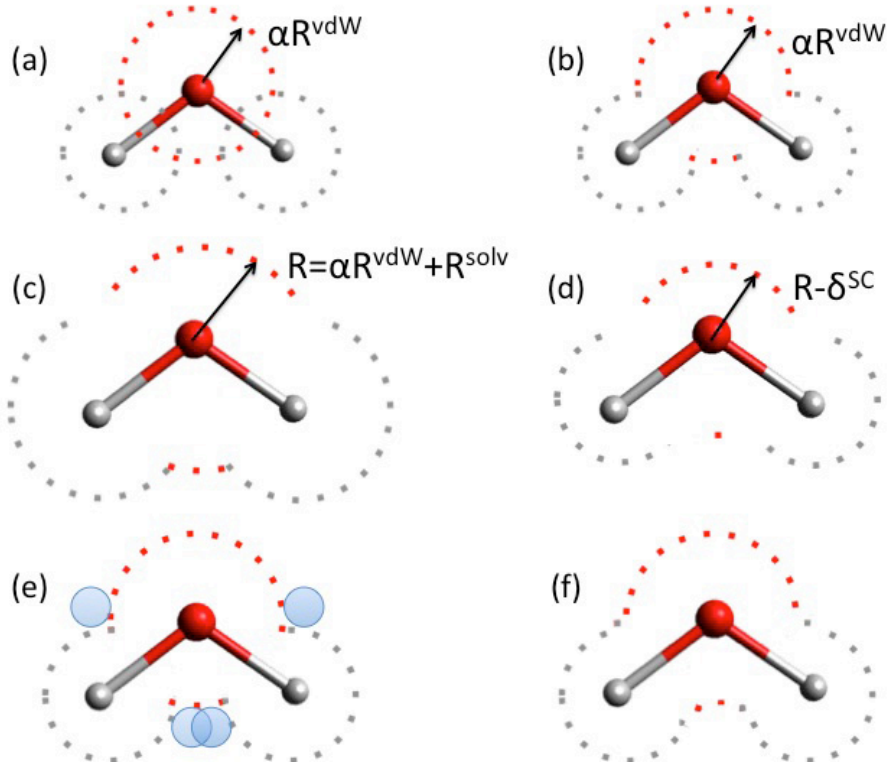


Figure 2: The construction of different types of solute cavities (schematic), using a water molecule as an example solute. (a) Interlocking spheres with scaled van der Waals radii centered on the solute atoms (scaling factor $\alpha \geq 1$). (b) Remove the surface buried inside the cavity and form the scaled van der Waals surface. This is the currently used definition of solvent accessible surface (SAS). (c) Form the surface built in (b), expand by the solvent radii R^{solv}. (d) Contract the radii of surface built in (c) by δ^{SC} and form the Klamt solvent accessible surface (SAS(K)) in the original COSMO method. (e) Roll a probe (representing the solvent) around the SAS in (b). (f) Take the area explored by center of the probe as the solvent excluded surface (SES, or Connolly surface).

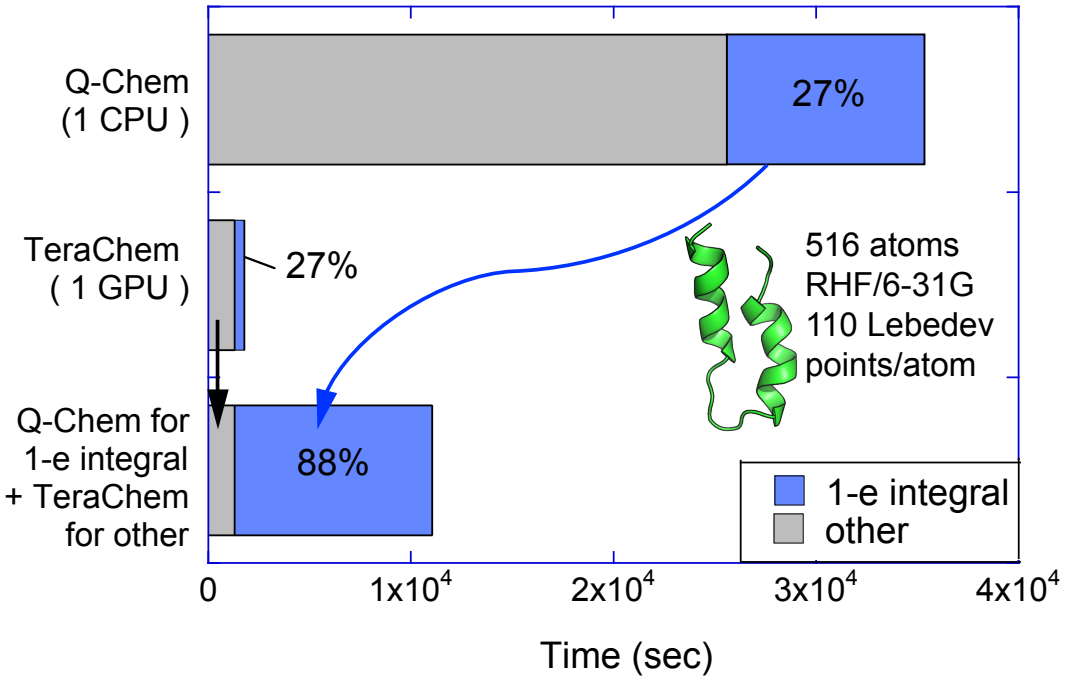


Figure 3: Percentage of time taken by OEIs in a polarizable continuum model calculation of a peptide (PDB ID: 2KJM) with TERACHEM and Q-CHEM.

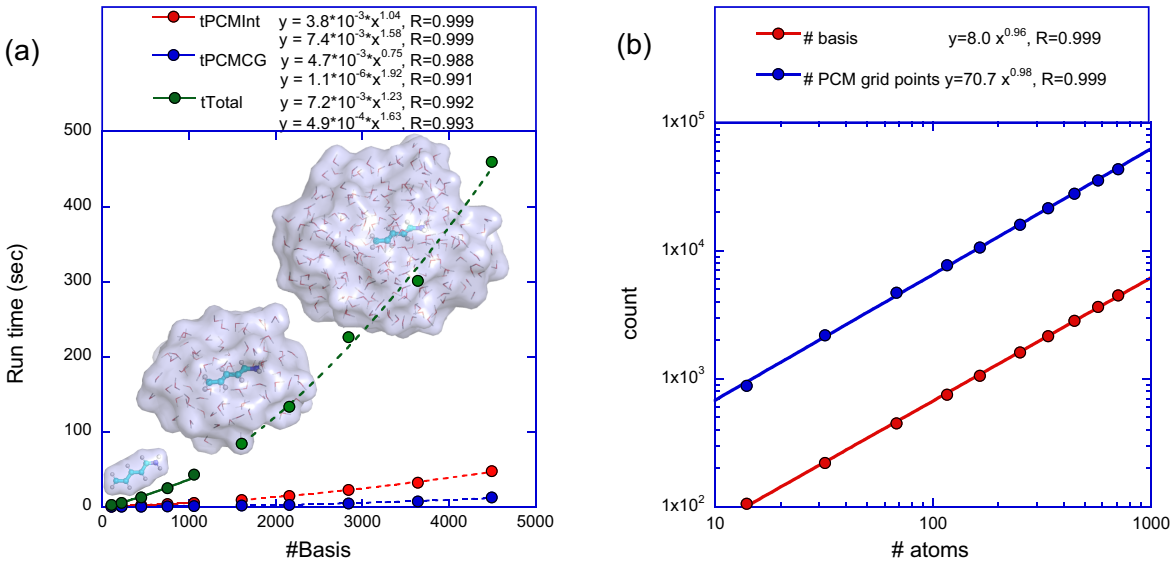


Figure 4: Timing for C-PCM ωPBEh/6-31G* single point energy evaluation for trans-PSB3 and a series of microsolvated trans-PSB3 molecules. (a) The total timing involves the whole SCF calculation (both gas phase portion and PCM related). The time for PCM integrals includes calculating the electrostatic potential and building the Fock matrix contribution. The structures for trans-PSB3 molecule and its microsolvated forms in water spheres of radii 6 Å and 10 Å are shown with respective cavity surfaces in the inset. C, N, O, H atoms are colored with cyan, blue, red, and white, respectively. Timings were obtained using 1 NVIDIA Tesla P100 GPU accelerator. For the run time, a power law fit is conducted separately for small-medium systems (up to 1067 basis functions) and larger systems. Intel MKL and GPU-based MAGMA libraries are used for the Fock matrix diagonalization, respectively. (b) Number of basis functions and number of PCM grid points as a function of the number of atoms in the benchmark molecules.

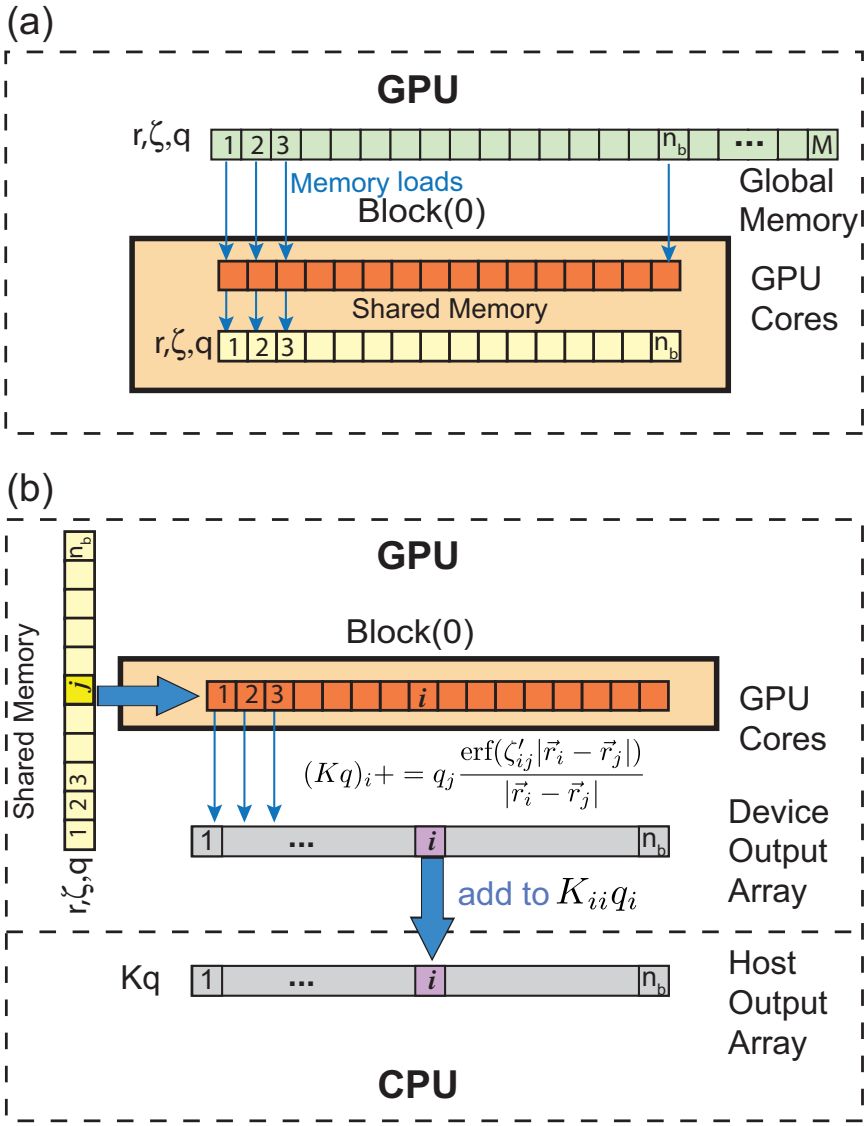


Figure 5: Algorithm for bulding **Kq** product on GPUs. At the top of the graph, the pale green array represents data for each tessera (Gaussian charge distribution), including the center coordinates, Gaussian exponents, and charge. The GPU cores are represented by orange squares (threads) embedded in pale yellow rectangles (1 dimensional blocks with n_b=16 threads/block). The algorithm includes two steps. (a) Each thread fetches the data of the tessera corresponding to its thread index and copies it to shared memory as a coalesced transfer. (b) Each thread is assigned to the data of a tessera corresponding to its ID, ***i***, and loops over all tesserae ***j*** loaded in the shared memory to calculate the contribution to the ***i***-th entry of the **Kq** product: $K_{ij}q_j$. Since the number of tesserae is usually larger than the block size, the tessera data will be loaded in batches into shared memory, and the two steps will be repeated until ***j*** loops over all tesserae. We show only the 0th block. For block *b,* the tessara indices ***i*** corresponding to each thread in the block are $b*n_b$, $b*n_b+1$, …, $(b+1)*n_b-1$

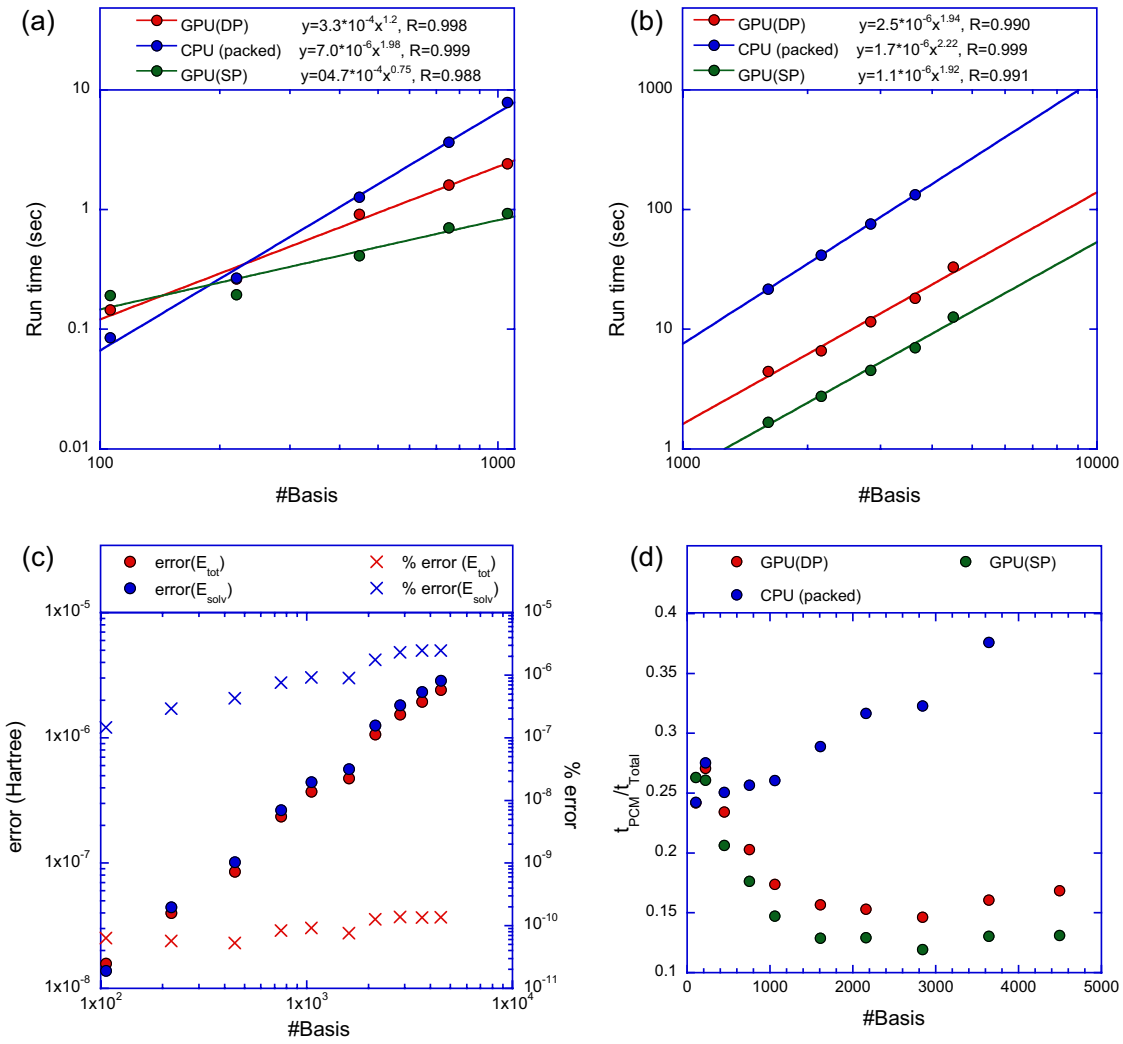
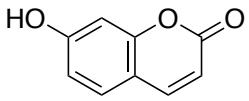


Figure 6: Timing for different TERACHEM PCM CG solver implementations for trans-PSB3 and a series of microsolvated trans-PSB3 molecules. The CPU (packed) RBJ solver explicitly stores the upper-triangle of the PCM matrix and uses MKL on the CPU to perform the matrix-vector product. GPU(DP) and GPU(SP) refer to forming the matrix-vector product on the GPU (without explicit storage of the matrix) using double precision (DP) and single precision (SP). Empirical scaling of run time is obtained from a power law fit. The fit is done separately for (a) small/medium (up to 1067 basis functions) and (b) large systems, respectively. (c) Absolute error in total energy of the solvated system (E_{tot}) and solvation energy (E_{solv}) calculated with GPU(SP) compared with GPU(DP). (d) Ratio of time spent on PCM related evaluations (PCM integrals and CG solver) with respect to the total run time (PCM + gas phase).



Umbelliferone

Scheme 1: Chemical structure of Umbelliferone (IUPAC name: 7-Hydroxychromen-2-one)

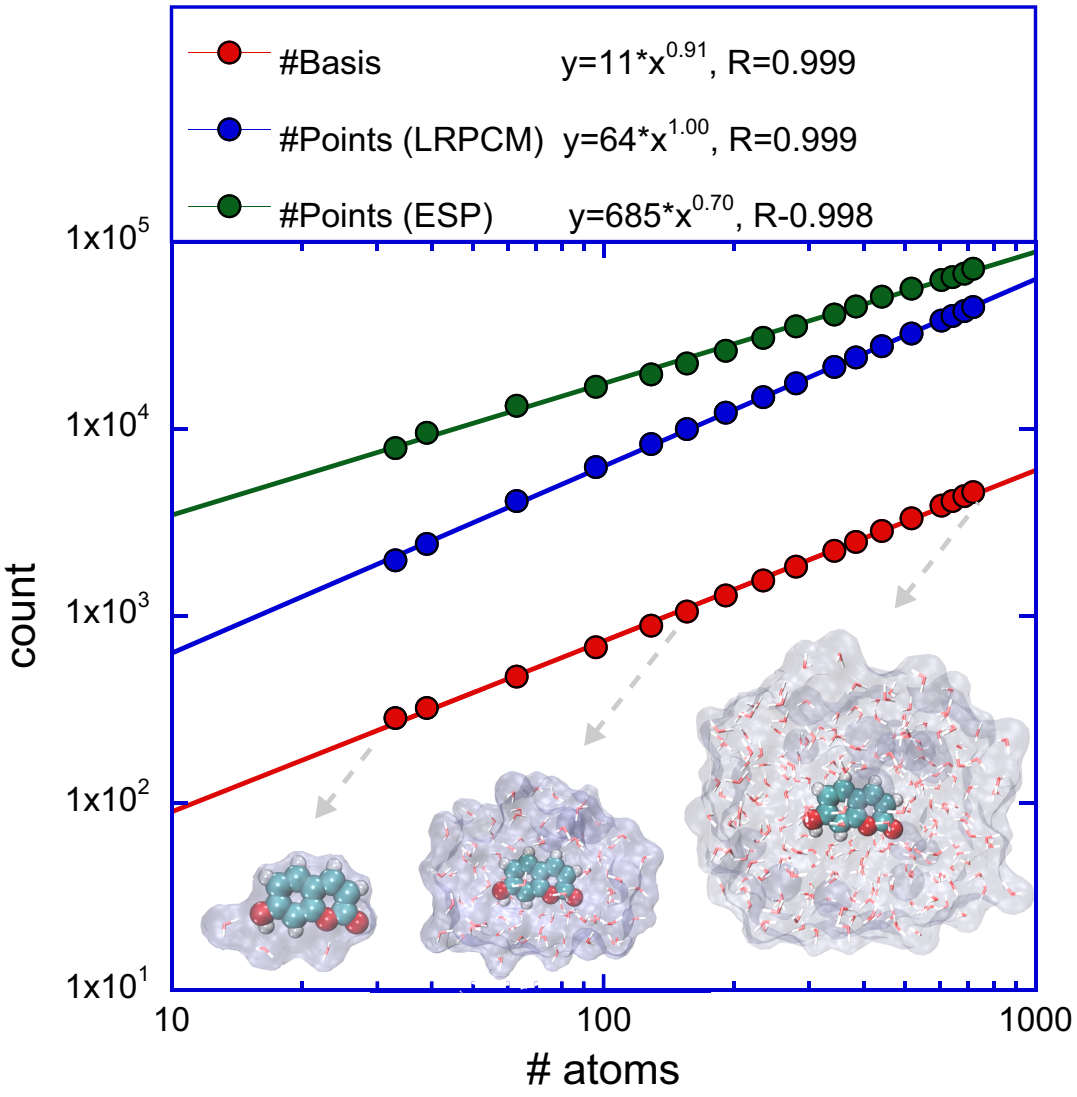


Figure 7: Systems used for benchmarking LR-PCM and RESP calculations: a series of microsolvated umbelliferone molecules with increasing numbers of explicit water molecules. Red dots show the number of basis functions in each system (6-31G* basis set). Blue dots show the number of cavity surface points used for LRPCM benchmarking, with Lebedev grid density of 110 points/atom and cavity radii scaling factor of 1.2. Green dots show the number of ESP grid points used for RESP charge benchmarking, with 4 layers of Connolly surfaces with radii scaling factor starting from 1.4 and an increment of 0.2 per layer. Grid density is 5.0 points/Å^2.

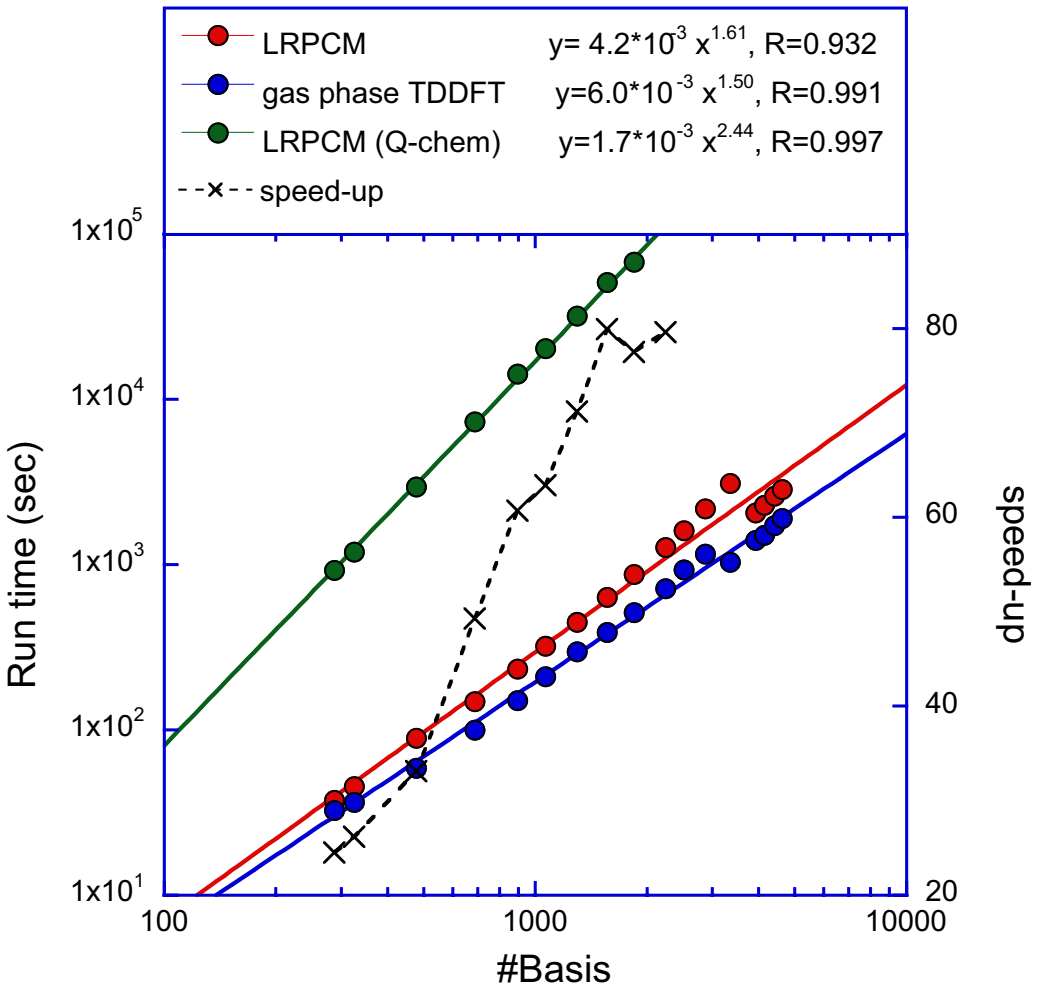


Figure 8: Timings for TDDFT/TDA ωPBEh/6-31G* gradient evaluation in gas phase and in LR-PCM equilibrium solvation for a series of microsolvated umbelliferone molecules. Red and blue dots show timings for TERACHEM using 1 NVIDIA Tesla P100 GPU accelerator. Green dots are the timings for Q-CHEM LR-PCM calculations using 1 Intel Xeon E5-2643 CPU clocked at 3.30GHz. The speed-up for TERACHEM vs Q-CHEM for LRPCM is shown by black Xs.

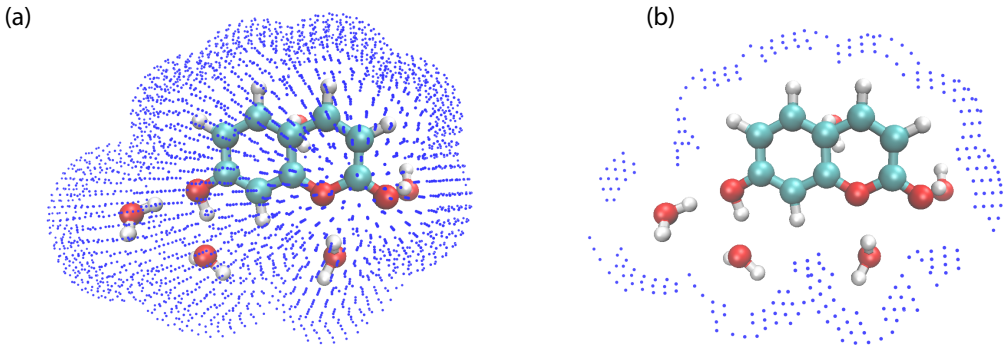


Figure 9 (a) RESP grid points for microhydrated umbelliferone. MEP grid points were built from 4 layers of Connolly surface with radii scaling factor starting from 1.4 and an increment of 0.2 per layer. Grid density is 5.0 points/Å^2. The molecules are represented with ball and sticks, with C, O, H colored as cyan, red, and white, respectively. Blue dots represent grid points. (b) Slice of the grid points in the plane of the aromatic ring, highlighting the four-layer structure of the RESP grid.

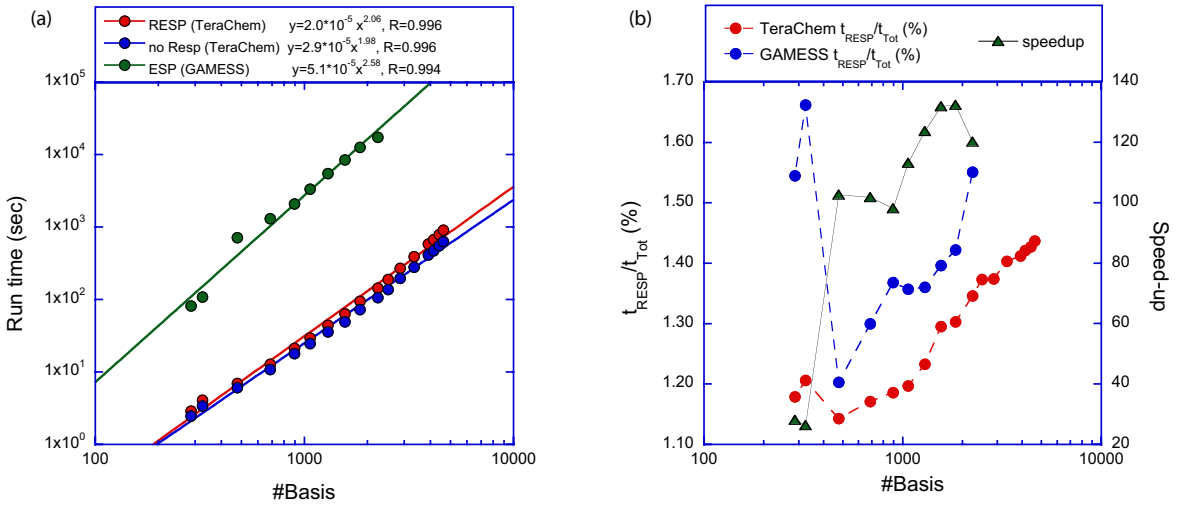


Figure 10: Timings for HF/6-31G* RESP charge calculation for a series of microhydrated umbelliferone molecules. (a) Red and blue dots represent the timings for TERACHEM using 1 NVIDIA Tesla P100 GPU accelerator. Green dots are the timings for GAMESS LR-PCM calculations using 1 Intel Xeon E5-2643 CPU clocked at 3.30GHz. RESP timing includes both SCF and the following RESP charge evaluation. (b) The percentage of time taken by RESP charge evaluation in an SCF calculation followed by RESP charge evaluation is shown with red dots and blue dots, for TERACHEM and GAMESS, respectively. The speed-up of TERACHEM compared to GAMESS for the RESP charge evaluation is shown as blue dots.

References

[1] I. S. Ufimtsev, N. Luehr, and T. J. Martinez, J. Phys. Chem. Lett. **2** (2011) 1789.

[2] B. Mennucci, WIREs: Comput. Mol. Sci. **2** (2012) 386.

[3] J. Tomasi, and M. Persico, Chem. Rev. **94** (1994) 2027.

[4] S. Miertuš, E. Scrocco, and J. Tomasi, Chem. Phys. **55** (1981) 117.

[5] A. Klamt, and G. Schuurmann, J. Chem. Soc. Perkin Trans. 2 (1993) 799.

[6] J. Andzelm, C. Kölmel, and A. Klamt, J. Chem. Phys. **103** (1995) 9312.

[7] A. Klamt, WIREs: Comput. Mol. Sci. **8** (2017) e1338.

[8] V. Barone, and M. Cossi, J. Phys. Chem. A **102** (1998) 1995.

[9] T. N. Truong, and E. V. Stefanovich, Chem. Phys. Lett. **240** (1995) 253.

[10] E. Cancès, B. Mennucci, and J. Tomasi, J. Chem. Phys. **107** (1997) 3032.

[11] B. Mennucci, E. Cancès, and J. Tomasi, J. Phys. Chem. B **101** (1997) 10506.

[12] J. Tomasi, B. Mennucci, and E. Cancès, J. Mol. Struct: THEOCHEM **464** (1999) 211.

[13] D. M. York, and M. Karplus, J. Phys. Chem. A **103** (1999) 11060.

[14] A. W. Lange, and J. M. Herbert, J. Chem. Phys. **133** (2010) 244111.

[15] G. Scalmani, and M. J. Frisch, J. Chem. Phys. **132** (2010) 114110.

[16] M. Orozco, and F. J. Luque, Chem. Rev. **100** (2000) 4187.

[17] L. Greengard, and V. Rokhlin, J. Comp. Phys. **73** (1987) 325.

[18] P. Li, H. Johnston, and R. Krasny, J. Comp. Phys. **228** (2009) 3858.

[19] G. Scalmani *et al.*, Theo. Chem. Acc. **111** (2004) 90.

[20] B. Stamm *et al.*, J. Chem. Phys. **144** (2016) 054101.

[21] F. Lipparini *et al.*, J. Phys. Chem. Lett. **5** (2014) 953.

[22] E. Cancès, Y. Maday, and B. Stamm, J. Chem. Phys. **139** (2013) 054111.

[23] F. Lipparini *et al.*, J. Chem. Theo. Comp. **9** (2013) 3637.

[24] F. Lipparini *et al.*, J. Chem. Phys. **141** (2014) 184108.

[25] F. Liu *et al.*, J. Chem. Theo. Comp. **11** (2015) 3131.

[26] G. H. Golub, and C. F. Van Loan, *Matrix computations* (Johns Hopkins University Press, Baltimore, 2013), 4th edn., Johns Hopkins studies in the mathematical sciences.,

[27] A. Szabo, and N. S. Ostlund, *Modern Quantum Chemistry* (Dover, New York, 1996),

[28] W. Kohn, and L. J. Sham, Phys. Rev. **140** (1965) A1133.

[29] S. Hirata, and M. Head-Gordon, Chem. Phys. Lett. **302** (1999) 375.

[30] S. Hirata, M. Head-Gordon, and R. J. Bartlett, J. Chem. Phys. **111** (1999) 10774.

[31] A. Dreuw, and M. Head-Gordon, Chem. Rev. **105** (2005) 4009.

[32] F. Furche, and R. Ahlrichs, J. Chem. Phys. **117** (2002) 7433.

[33] F. A. Momany, J. Phys. Chem. **82** (1978) 592.

[34] S. Cox, and D. Williams, J. Comp. Chem. **2** (1981) 304.

[35] U. C. Singh, and P. A. Kollman, J. Comp. Chem. **5** (1984) 129.

[36] L. E. Chirlian, and M. M. Francl, J. Comp. Chem. **8** (1987) 894.

[37] D. E. Williams, Rev. Comp. Chem. **2** (2007) 219.

[38] L. E. McMurchie, and E. R. Davidson, J. Comp. Phys. **26** (1978) 218.

[39] D. M. Chipman, J. Chem. Phys. **112** (2000) 5558.

[40] M. L. Connolly, J. Appl. Cryst. **16** (1983) 548.

[41] V. I. Lebedev, USSR Comp. Math. Math. Phys. **16** (1976) 10.

[42] J. L. Pascual‐Ahuir, E. Silla, and I. Tunon, J. Comp. Chem. **15** (1994) 1127.

[43] J. L. Pascual‐Ahuir, and E. Silla, J. Comp. Chem. **11** (1990) 1047.

[44] E. Silla, I. Tunon, and J. L. Pascual‐Ahuir, J. Comp. Chem. **12** (1991) 1077.

[45] E. Silla *et al.*, J. Mol. Graph. **8** (1990) 168.

[46] M. W. Schmidt *et al.*, J. Comp. Chem. **14** (1993) 1347.

[47] M. S. Gordon, and M. W. Schmidt, in *Theory and applications of computational chemistry* (Elsevier, 2005), pp. 1167.

[48] M. J. Frisch *et al.*, Wallingford, CT, 2016).

[49] F. Neese, WIREs: Comput. Mol. Sci. **2** (2012) 73.

[50] O. Treutler, and R. Ahlrichs, J. Chem. Phys. **102** (1995) 346.

[51] C. W. Murray, N. C. Handy, and G. J. Laming, Mol. Phys. **78** (1993) 997.

[52] Y. Shao *et al.*, Mol. Phys. **113** (2015) 184.

[53] A. W. Lange, and J. M. Herbert, J. Chem. Phys. **134** (2011) 117102.

[54] H. Li, and J. H. Jensen, J. Comp. Chem. **25** (2004) 1449.

[55] K. Yasuda, J. Comp. Chem. **29** (2008) 334.

[56] I. S. Ufimtsev, and T. J. Martinez, Comp Sci. Eng. **10** (2008) 26.

[57] I. S. Ufimtsev, and T. J. Martinez, J. Chem. Theo. Comp. **4** (2008) 222.

[58] I. S. Ufimtsev, and T. J. Martinez, J. Chem. Theo. Comp. **5** (2009) 1004.

[59] I. S. Ufimtsev, and T. J. Martinez, J. Chem. Theo. Comp. **5** (2009) 2619.

[60] C. I. Bayly *et al.*, J. Phys. Chem. **97** (1993) 10269.

[61] W. D. Cornell *et al.*, J. Am. Chem. Soc. **115** (1993) 9620.

[62] M. Garavelli *et al.*, J. Am. Chem. Soc. **119** (1997) 6891.

[63] D. A. Case *et al.*, (University of California, 2012).

[64] J. Wang *et al.*, J. Mol. Graph. Model. **25** (2006) 247.

[65] J. Wang *et al.*, J. Comp. Chem. **25** (2004) 1157.

[66] W. L. Jorgensen *et al.*, J. Chem. Phys. **79** (1983) 926.

[67] M. A. Rohrdanz, K. M. Martins, and J. M. Herbert, J. Chem. Phys. **130** (2009)

[68] Ditchfie.R, W. J. Hehre, and J. A. Pople, J. Chem. Phys. **54** (1971) 724.

[69] S. Tomov, J. Dongarra, and M. Baboulin, Par. Comp. **36** (2010) 232.

[70] S. Tomov *et al.*, in *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on* (IEEE, 2010), pp. 1.

[71] J. Dongarra *et al.*, in *Numerical Computations with GPUs* (Springer, 2014), pp. 3.

[72] R. Cammi *et al.*, J. Chem. Phys. **122** (2005) 104513.

[73] G. Scalmani *et al.*, J. Chem. Phys. **124** (2006) 094107.

[74] M. Cossi, and V. Barone, J. Chem. Phys. **115** (2001) 4708.

[75] W. D. Cornell *et al.*, J. Am. Chem. Soc. **117** (1995) 5179.

[76] R. J. Woods *et al.*, J. Phys. Chem. **99** (1995) 3832.

[77] K. N. Kirschner *et al.*, J. Comp. Chem. **29** (2008) 622.

[78] J. L. Whitten, J. Chem. Phys. **58** (1973) 4496.

[79] C. M. Isborn *et al.*, J. Chem. Theo. Comp. **7** (2011) 1814.

[80] E. G. Hohenstein *et al.*, J. Chem. Phys. **142** (2015)

[81] J. W. Snyder *et al.*, J. Chem. Phys. **143** (2015)

[82] H. B. Schlegel, and M. A. Robb, Chem. Phys. Lett. **93** (1982) 43.

[83] D. Hegarty, and M. A. Robb, Mol. Phys. **38** (1979) 1795.

[84] N. Yamamoto *et al.*, Chem. Phys. Lett. **250** (1996) 373.

[85] R. H. Eade, and M. A. Robb, Chem. Phys. Lett. **83** (1981) 362.

[86] M. Frisch *et al.*, Chem. Phys. Lett. **189** (1992) 524.

[87] K. Andersson *et al.*, J. Phys. Chem. **94** (1990) 5483.

[88] K. Andersson, P. Å. Malmqvist, and B. O. Roos, J. Chem. Phys. **96** (1992) 1218.

[89] M. Filatov, and S. Shaik, Chem. Phys. Lett. **304** (1999) 429.

[90] I. d. P. R. Moreira *et al.*, J. Chem. Theo. Comp. **3** (2007) 764.

[91] M. Filatov, WIREs: Comput. Mol. Sci. **5** (2015) 146.

[92] M. Filatov, in *Density-functional methods for excited states*, edited by N. Ferre, M. Filatov, and M. Huix-Rotllant (Springer, Heidelberg, 2016), pp. 97.

[93] J. W. Snyder Jr, R. M. Parrish, and T. J. Martínez, J. Phys. Chem. Lett. **8** (2017) 2432.

[94] R. N. Diffenderfer, and D. R. Yarkony, J. Phys. Chem. **86** (1982) 5098.

[95] J. Finley *et al.*, Chem. Phys. Lett. **288** (1998) 299.

[96] T. Shiozaki *et al.*, J. Chem. Phys. **135** (2011) 081106.

[97] M. Filatov, F. Liu, and T. J. Martínez, J. Chem. Phys. **147** (2017) 034113.

[98] B. G. Levine *et al.*, Mol. Phys. **104** (2006) 1039.

[99] M. Huix-Rotllant *et al.*, J. Chem. Theo. Comp. **9** (2013) 3917.

[100] S. L. Li *et al.*, J. Phys. Chem. Lett. **5** (2014) 322.

[101] S. Gozem *et al.*, J. Chem. Theo. Comp. **8** (2012) 4069.

[102] D. R. Yarkony, Chem. Rev. **112** (2011) 481.

[103] R. Improta *et al.*, J. Chem. Phys. **125** (2006) 054103.

[104] M. Cossi, and V. Barone, J. Chem. Phys. **112** (2000) 2427.

[105] M. Cossi, and V. Barone, J. Phys. Chem. A **104** (2000) 10614.

[106] R. Improta *et al.*, J. Chem. Phys. **127** (2007) 074504.

[107] R. E. Stratmann, G. E. Scuseria, and M. J. Frisch, J. Chem. Phys. **109** (1998) 8218.

[108] H. Weiss, R. Ahlrichs, and M. Häser, J. Chem. Phys. **99** (1993) 1262.

[109] A. Görling *et al.*, J. Chem. Phys. **110** (1999) 2785.

[110] S. Van Gisbergen, J. Snijders, and E. Baerends, Comp. Phys. Comm. **118** (1999) 119.

[111] E. R. Davidson, J. Comp. Phys. **17** (1975) 87.

[112] J. Liu, and W. Liang, J. Chem. Phys. **138** (2013) 024101.

[113] Y. Wang, and H. Li, J. Chem. Phys. **133** (2010) 034108.

[114] J.-J. Aaron *et al.*, J. Fluorescence **5** (1995) 337.

[115] T. Moriya, Bull. Chem. Soc. Japan **61** (1988) 1873.

[116] K. Azuma *et al.*, Photochem. Photobio. Sci. **2** (2003) 443.

[117] B. Ramesh, and K. Pugalendi, Life sciences **79** (2006) 306.

[118] F.-Y. Dupradeau *et al.*, Phys. Chem. Chem. Phys. **12** (2010) 7821.

[119] C. M. Breneman, and K. B. Wiberg, J. Comp. Chem. **11** (1990) 361.

[120] M. A. Spackman, J. Comp. Chem. **17** (1996) 1.

[121] R. H. Henchman, and J. W. Essex, J. Comp. Chem. **20** (1999) 483.

[122] J. A. Maier *et al.*, J. Chem. Theo. Comp. **11** (2015) 3696.

[123] V. Hornak *et al.*, Proteins: Structure, Function, and Bioinformatics **65** (2006) 712.

[124] D. E. Williams, Biopolymers **29** (1990) 1367.

[125] K. M. Merz, J. Comp. Chem. **13** (1992) 749.

[126] A. Goez, and J. Neugebauer, in *Frontiers of Quantum Chemistry* (Springer, 2018), pp. 139.

[127] X. Li *et al.*, J. Chem. Theo. Comp. **13** (2017) 3493.

[128] P. Su, and H. Li, J. Chem. Phys. **130** (2009) 074109.