JCTC Journal of Chemical Theory and Computation

Article

pubs.acs.org/JCTC

Fast Domain Decomposition Algorithm for Continuum Solvation Models: Energy and First Derivatives

Filippo Lipparini,*[,†] Benjamin Stamm,[‡] Eric Cancès,[§] Yvon Maday,[‡,∥,⊥] and Benedetta Mennucci[#]

[†]UPMC Université Paris 06, Institut du Calcul et de la Simulation, F-75005 Paris, France

[‡]UPMC Université Paris 06, UMR 7598, Laboratoire Jacques-Louis Lions, F-75005, Paris, France

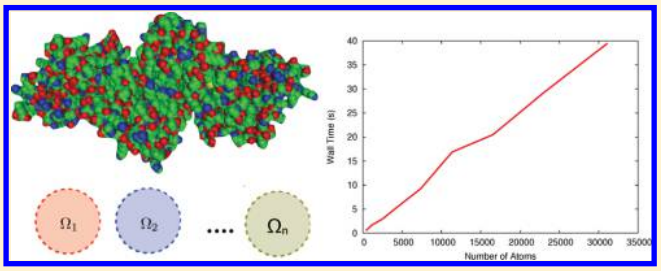
[§]Université Paris-Est, CERMICS, Project-team Micmac, INRIA-Ecole des Ponts, 6 & 8 avenue Blaise Pascal, 77455 Marne-la-Vallée Cedex 2, France

[∥]Institut Universitaire de France, 103 bd Saint-Michel 75005 Paris

[⊥]Brown University, Division of Applied Maths, Providence, Rhode Island, United States

[#]Dipartimento di Chimica e Chimica Industriale, Università di Pisa, Via Risorgimento 35, 56126 Pisa, Italy

ABSTRACT: In this contribution, an efficient, parallel, linear scaling implementation of the conductor-like screening model (COSMO) is presented, following the domain decomposition (dd) algorithm recently proposed by three of us. The implementation is detailed and its linear scaling properties, both in computational cost and memory requirements, are demonstrated. Such behavior is also confirmed by several numerical examples on linear and globular large-sized systems, for which the calculation of the energy and of the forces is achieved with timings compatible with the use of polarizable continuum solvation for molecular dynamics simulations.



1. INTRODUCTION

Continuum (or implicit) solvation models are nowadays largely used computational approaches to include solvent effects in the simulation of properties and processes of (supra)molecular systems in condensed phase.[1−7] Their easiness of use combined with a favorable cost/effectiveness ratio has allowed their application in very different research fields, from chemistry to biology and material sciences. Due to the very different typical ranges of accuracy and dimensions of these fields, continuum solvation models have been combined with both classical molecular mechanics (MM) approaches and quantum-mechanical (QM) descriptions.

Nowadays, many formulations are available in the most common computational softwares: they differentiate both in terms of specific modellistic aspects and computational implementations. In all cases, however, the starting point is common, namely the electrostatic problem of a charge distribution contained in a cavity of proper shape and dimension[6,8] built within a dielectric continuum characterized by its permittivity ε. Among the possible strategies developed so far to solve such a problem, one of the most successful approaches introduces the concept of apparent surface charge (ASC) to represent the polarization of the dielectric. The ASC allows, in fact, to reduce the entire problem to the cavity surface without the need to account for the volume of the dielectric. Within the ASC approach, different formulations have been given in the years. Among them, we quote here the Polarizable Continuum Model (PCM) by Tomasi and co-workers, the "conductor-like screening model" (COSMO) originally developed by Klamt and Schüürmann[9] and the "surface and simulation of volume polarization for electrostatics" (SS(V)PE) method by Chipman.[10,11] As a matter of fact, the PCM is now an acronym which collects a family of methods combining different solutions of the same electrostatic problem (namely DPCM,[12] IEFPCM,[13−15] CPCM[16,17]) with different formulations (VPCM[18,19]) and/or different computational implementations (IPCM,[20] CSCPCM[21]). Likewise, COSMO is now available in implementations which largely differentiate with respect to the original formulation, suffice here to quote the "smooth COSMO" (S-COSMO) model of York and Karplus[22] and the "Switching/Gaussian" (SWIG) approach by Lange and Herbert.[23,24] To further increase the complexity of the picture and the difficulties for a user to understand the differences among the many formulations, we have to recall that a given continuum model, for example COSMO or IEFPCM, can give different results when used in different softwares (Gaussian, QCHEM, ADF, GAMESS, just to quote the most used ones) as each software has its own definition of the cavity and of the numerical strategy used to mesh the surface cavity into the finite elements (or points) used to represent the ASC.

From this brief summary, it is clear that state-of-the-art continuum solvation models are the result of a complex combination of physical, mathematical, and numerical expertises

Received: April 5, 2013
Published: July 3, 2013

which, during the years, has made them more and more accurate in the solution of the original electrostatic problem and more and more efficient in their numerical performances. This increased accuracy and efficiency is indeed one of the main aspects that characterize the evolution of these models in the last years and their large applicability, now extending well beyond the field of small molecules in simple homogeneous and isotropic solvents.[25,26] Moreover, most of the recent formulations have been generalized to include derivatives with respect to internal (namely geometrical) and external parameters (such as electric and magnetic fields). This further extension has allowed to apply continuum solvation models to study geometries of solvated systems, to predict reaction mechanisms and to simulate spectroscopies.[27]

With the present paper, we add a further step in this evolution, reporting an efficient computational implementation of a new numerical approach to solve the electrostatic problem of continuum solvation models derived from the Domain Decomposition (dd) methods.[28] The common point of all the present implementations of the various continuum solvation methods is that the numerical methods used to solve the integral equations belongs to the family of Galerkin methods. In particular, two main strategies of discretization are currently employed, the first one being based on a boundary element approximation, the second one on a nonconformal Galerkin approximation, which makes use of Gaussian basis functions to expand the ASC. Nevertheless, all the implementations start by defining a suitable mesh on the surface of the cavity, which is used to position the basis functions, let them be spherical gaussians[21,22,24] or piecewise constant functions:[12,29] the approximation basis set is then used to represent the integral operators, so that the problem is transformed into the solution of a linear system, for which either matrix inversion or iterative approaches[30] can be used. As all the integral operators involved in continuum solvation produce, given the ASC, either a potential or a field, the fast multipole method (FMM)[29,31] can be used to compute the matrix-vector products implied in any iterative procedure.

The computational implementation that we propose here is based on a completely new path with respect to all the discretization schemes used so far to solve for the ASC. For now, only the COSMO (or C-PCM) equations will be considered: the extension of our numerical procedure to the IEF-PCM is object of active investigation. The main idea is to divide the domain (i.e., the cavity) in a set of subdomains simple enough that one can either solve the problem analytically or numerically but with a very limited computational effort. Here, the subdomains are the spheres composing the molecular cavity, which are centered on the atoms forming the solute molecule and for which a closed expression for the solution to the COSMO equation exists. We will show how the discretization method produces a very sparse linear system, which can be solved efficiently with an iterative procedure. Notice that the sparsity of the system implies the linear scaling properties of the method, which we will demonstrate in this paper, without relying on fast summation techniques. Such an efficient implementation paves the way to the use of PCM and similar methods both in molecular dynamics simulations and in large QM/MM/Continuum calculations, where the cost of solving the integral equations largely dominates the overall computational effort.

The details of the numerical approach will be given in section 2, whereas its extension to the analytical derivatives will be presented in section 3. In section 4, we will show how to achieve both computational costs and memory requirements that scale linearly with the number of atoms. Some numerical experiments will be used in section 5 to demonstrate the properties of our algorithm; finally, we will draw some conclusions and perspectives.

2. THEORY

In this section, we will briefly summarize the main aspects of the dd method that we will use to solve the COSMO[9] equations for a van der Waals molecular cavity. As the theory has been thoroughly described in a parallel paper by some of the authors,[32] we will present here only the discretized equations, on the implementation of which we will focus in section 4. We consider a solute molecule of M atoms carrying a classical charge distribution

$$\rho(\mathbf{r}) = \sum_{j=1}^{M} q_j \delta(\mathbf{r} - \mathbf{R}_j)$$

where q_j is the charge of the j^{th} atom and $\mathbf{R}_j$ its position. Notice that more complex classical distributions, including higher order or polarizable multipoles,[33−37] can be also treated with a straightforward generalization. The van der Waals molecular cavity is defined as a union of spheres centered at the atoms and radius the atomic van der Waals radius, scaled for a constant factor, that reads as

$$\Omega = \bigcup_{j=1}^{M} \Omega_j$$

Here, we call Ω_j the sphere with center $\mathbf{R}_j$ (i.e., the position of the j^{th} atom) and radius r_j. In COSMO, the solute and the solvent interact by mutually polarizing each other: the solvation energy of the molecule is defined as the interaction of the solute's density of charge with the reaction *potential* W, which is the contribution to the total electrostatic potential due to the polarization of the environment:

$$E_s = \frac{1}{2} f(\varepsilon_s) \int_\Omega \rho(\mathbf{r}) W(\mathbf{r}) = \frac{1}{2} f(\varepsilon_s) \sum_{j=1}^{M} q_j W(\mathbf{R}_j) \tag{1}$$

Here, $f(\varepsilon_s)$ is a constant scaling factor depending on the solvent dielectric constant ε_s used to take into account the nonconductor nature of the solvents and the reaction potential W is the unique solution to the boundary value problem[38]

$$\begin{cases} -\Delta W(\mathbf{r}) = 0 & \text{for} \quad \mathbf{r} \in \Omega \\ W(\mathbf{s}) = -\Phi(\mathbf{s}) & \text{for} \quad \mathbf{s} \in \Gamma \end{cases} \tag{2}$$

In the above equations, $\Gamma = \partial\Omega$ is the surface of the cavity Ω and Φ the potential generated by ρ *in vacuo*, given by

$$\forall\, \mathbf{r} \in \mathbb{R}^3, \qquad \Phi(\mathbf{r}) = \sum_{j=1}^{M} \frac{q_j}{|\mathbf{r} - \mathbf{R}_j|} \tag{3}$$

The usual way to compute the solvation energy E_s is to use the fact that it is possible to represent the reaction potential by means of an apparent surface charge (ASC) σ, which produces the potential W:[38]

$$\forall\, \mathrm{r} \in \Omega, \qquad W(\mathbf{r}) = \int_\Gamma \frac{\sigma_\Gamma(\mathbf{s})}{|\mathbf{r} - \mathbf{s}|} \mathrm{d}\mathbf{s} \tag{4}$$

The ASC is then computed by imposing that the potential it produces at Γ is equal to minus the solute's potential, as in eq 2. This defines an integral equation on Γ:

$$\forall\ \mathbf{s} \in \Gamma, \quad \int_\Gamma \frac{\sigma_\Gamma(\mathbf{s}')}{|\mathbf{s}-\mathbf{s}'|}\mathrm{d}\mathbf{s}' = (\mathcal{S}_\Gamma\sigma)(s) = -\Phi(\mathbf{s}) \tag{5}$$

Such an integral equation is usually discretized according to some boundary element scheme; here, we follow a completely different path.

The main idea of the Schwarz Domain Decomposition method is to divide the domain, that is, the cavity (and not the surface), in a set of simple overlapping subdomains, which, for the COSMO problem in a van der Waals cavity, are obviously the single spheres composing the cavity. Let us consider a van der Waals cavity as the one sketched in Figure 1. Each of the spheres

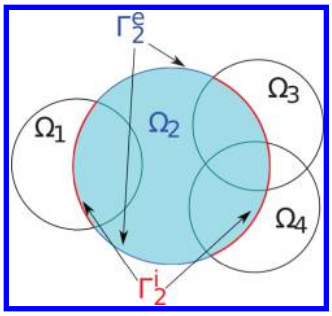


Figure 1. Domain decomposition of a van der Waals cavity.

composing the cavity has a portion of its surface Γ_j exposed to the solvent (Γ_j^e) and a portion which is buried into the cavity (Γ_j^i) and, in particular, is contained in one or more other neighboring spheres. Unfortunately, it is not possible to solve the standard COSMO problem as a collection of independent problems on a single sphere, which would be very easy, since, in order to do so, one should have a boundary condition not only on the external portion of the surface (which is provided by the original COSMO problem) but also on the internal one. To better clarify this point, let us consider the sphere Ω_2 in Figure 1: we know the boundary condition on Γ_2^e but not on Γ_2^i. The domain decomposition method gives a strategy to setup an iterative procedure where, at each step, for each sphere Γ_j, the boundary condition on Γ_j^i is computed as the reaction potential obtained at the previous step in the sphere that intersect Ω_j or, if more spheres intersect Ω_j at the same point, as the weighted average of the previous step potentials of each intersecting sphere. A pictorial representation of the traditional and dd approaches is given by Figure 2. Let us consider sphere Ω_1: at iteration "it", we will solve an integral equation

$$\forall\ \mathbf{s} \in \Gamma_1, \quad (\mathcal{S}_{\Gamma_1}\sigma^{1,(\mathrm{it})})(\mathbf{s}) = g_1^{(\mathrm{it})}(\mathbf{s})$$

where

$$g_1^{(\mathrm{it})}(\mathbf{s}) = \begin{cases} -\Phi(\mathbf{s}), & \mathbf{s} \in \Gamma_1^e \\ W_2^{(\mathrm{it}-1)}(\mathbf{s}), & \mathbf{s} \in \Gamma_1^i \end{cases}$$

Notice that we have introduced a local ASC $\sigma^1(\mathbf{s})$, which is not related to the global ASC. Always referring to Figure 2, for sphere Ω_2 we need to take into account multiple intersections. Let $\mathcal{N}_2(\mathbf{s})$ be the list of spheres that intersect Ω_2 at $\mathbf{s} \in \Gamma_2$ (see Figure 2 for a graphical description of the neighbor list $\mathcal{N}_2(\mathbf{s})$) and let $|\mathcal{N}_2(\mathbf{s})|$ be the number of intersecting spheres: the integral equation that we will solve for sphere Ω_2 will be

$$\forall\ \mathbf{s} \in \Gamma_2, \quad (\mathcal{S}_{\Gamma_2}\sigma^{1,(\mathrm{it})})(\mathbf{s}) = g_2^{(\mathrm{it})}(\mathbf{s})$$

where

$$g_2^{(\mathrm{it})}(\mathbf{s}) = \begin{cases} -\Phi(\mathbf{s}), & \mathbf{s} \in \Gamma_2^e \\ \displaystyle\sum_{k\in\mathcal{N}_2(\mathbf{s})} \frac{1}{|\mathcal{N}_2(s)|} W_k^{(\mathrm{it}-1)}(\mathbf{s}), & \mathbf{s} \in \Gamma_2^i \end{cases}$$

More in general, for each sphere Ω_j, we will solve

$$\forall\ \mathbf{s} \in \Gamma_j, \quad (\mathcal{S}_{\Gamma_j}\sigma^{j,(\mathrm{it})})(\mathbf{s}) = g_j^{(\mathrm{it})}(\mathbf{s}) \tag{6}$$

where

$$g_j^{(\mathrm{it})}(\mathbf{s}) = \begin{cases} -\Phi(\mathbf{s}), & \mathbf{s} \in \Gamma_j^e \\ \displaystyle\sum_{k\in\mathcal{N}_j(\mathbf{s})} \frac{1}{|\mathcal{N}_j(\mathbf{s})|} W_k^{(\mathrm{it}-1)}(\mathbf{s}), & \mathbf{s} \in \Gamma_j^i \end{cases} \tag{7}$$

For later convenience, we will introduce here a more compact notation for the definition of the boundary condition. Let χ_k be the characteristic function of Ω_k, that is,

$$\chi_k(\mathbf{r}) = \begin{cases} 1, & \mathbf{r} \in \Omega_k \\ 0, & \mathbf{r} \notin \Omega_k \end{cases}$$

A point $\mathbf{s} \in \Gamma_j$ will be external if and only if the characteristic function of all the neighbors of Ω_j is zero, that is, if

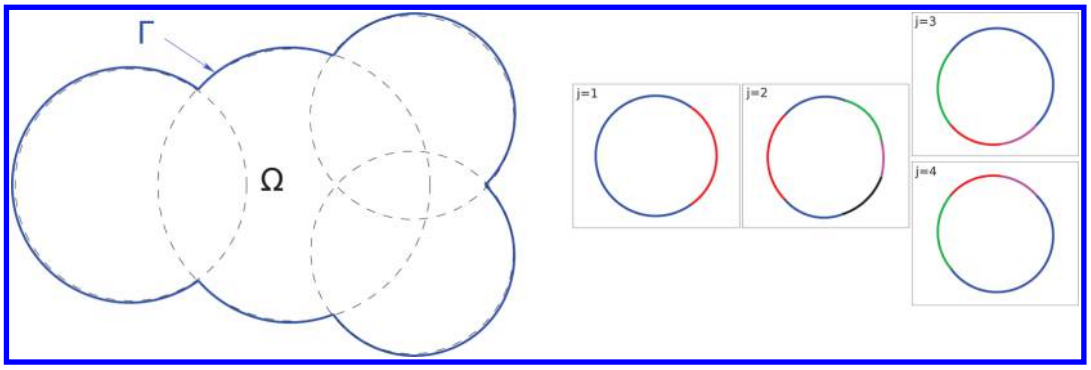


Figure 2. In the standard approach, the integral equation eq 5 is solved on the molecular cavity $\Gamma = \partial\Omega$ (left). In the ddCOSMO method, we solve a coupled system of integral equations on M copies of the unit sphere (right). For $j = 2$, the blue, red, green, black and magenta sets are those where $\mathcal{N}_2(\mathbf{s}) = \varnothing$, $\mathcal{N}_2(\mathbf{s}) = \{1\}$, $\mathcal{N}_2(\mathbf{s}) = \{3\}$, $\mathcal{N}_2(\mathbf{s}) = \{4\}$, and $\mathcal{N}_2(\mathbf{s}) = \{3,4\}$ respectively.

dx.doi.org/10.1021/ct400280b | J. Chem. Theory Comput. 2013, 9, 3637−3648

$$\sum_{k\in\mathcal{N}_j(\mathbf{s})}\frac{\chi_k(\mathbf{s})}{|\mathcal{N}_j(\mathbf{s})|}=\sum_{k\in\mathcal{N}_j(\mathbf{s})}\omega_{jk}(\mathbf{s})=0$$

where we have introduced the normalized weights $\omega_{jk}(\mathbf{s})=\chi_k(\mathbf{s})/|\mathcal{N}_j(\mathbf{s})|$. This allows us to rewrite eq 7 as

$$g_j^{(\mathrm{it})}(\mathbf{s})=-(1-\sum_{k\in\mathcal{N}_j(\mathbf{s})}\omega_{jk})\Phi(\mathbf{s})+\sum_{k\in\mathcal{N}_j(\mathbf{s})}\omega_{jk}W_k^{(\mathrm{it}-1)}(\mathbf{s}) \tag{8}$$

The final step in order to obtain the ddCOSMO linear system of coupled integral equations is to recognize that the reaction potential on the neighboring spheres can be expressed as the potential produced by the local apparent surface charge on such spheres by means of eq 4, which is also valid for the local ASC σ^k in Ω_k:

$$\forall\,\mathbf{s}\in\Gamma_j,\qquad W_k^{(it-1)}(\mathbf{s})=(\tilde{\mathcal{S}}_{jk}\sigma^{k,(it-1)})(\mathbf{s})=\int_{\Gamma_k}\frac{\sigma^{k,(it-1)}(\mathbf{y})}{|\mathbf{s}-\mathbf{y}|}\mathrm{d}\mathbf{y} \tag{9}$$

Notice that we have introduced the $\mathcal{S}_{jk}$ operator, which, given the local ASC on sphere Ω_k, computes the potential produced by σ^k at some point $\mathbf{s}$ on Γ_j. By substituting in eq 8 and putting everything together, we finally get the coupled equations for each sphere Ω_j:

$$\forall\,\mathbf{s}\in\Gamma_j,\qquad (\mathcal{S}_{\Gamma_j}\sigma^{j,(it)})(\mathbf{s})=-(1-\sum_{k\in\mathcal{N}_j(\mathbf{s})}\omega_{jk})\Phi(\mathbf{s})+\sum_{k\in\mathcal{N}_j(\mathbf{s})}\omega_{jk}(\tilde{*}_{jk}\sigma^{k,(it-1)})(\mathbf{y})(\mathbf{s}) \tag{10}$$

Notice how eq 10 represents a Jacobi iteration for a linear system of (integral) equations, which defines the ddCOSMO model:

$$\forall\,\mathbf{s}\in\Gamma_j,\qquad (\mathcal{S}_{\Gamma_j}\sigma^{j})(\mathbf{s})=-(1-\sum_{k\in\mathcal{N}_j(\mathbf{s})}\omega_{jk})\Phi(\mathbf{s})+\sum_{k\in\mathcal{N}_j(\mathbf{s})}\omega_{jk}(\tilde{\mathcal{S}}_{jk}\sigma^{k})(\mathbf{y})(\mathbf{s}) \tag{11}$$

Explicit Computation of the ddCOSMO Energy. In this subsection, we will discretize and compute explicitly all the contributions to eq 11. In order to do so, we will expand each local ASC in a truncated basis set of real spherical harmonics $\{Y_l^m\}_{l=0,N}^{m=-l,l}$ (see Appendix A for more details). This is particularly convenient, as all the quantities involved in eq 11 are either the potential produced by the solute or the potential produced by a local ASC: in order to compute the latter, we can easily use a truncated multipolar expansion. For the sake of simplicity, before proceeding we will rescale all the spheres to the unit sphere $\mathbb{S}^2$ and introduce accordingly the scaled local ASCs as

$$\forall\,\mathbf{s}\in\mathbb{S}^2,\qquad \sigma_j(\mathbf{s})=r_j\sigma^j(\mathbf{R}_j+r_j\mathbf{s})$$

Let us then represent σ_j in terms of a truncated expansion in spherical harmonics:

$$\sigma_j(\mathbf{s})=\sum_{lm}[X_j]_l^m Y_l^m(\mathbf{s})$$

where we put for brevity

$$\sum_{lm}:=\sum_{l=0}^{N}\sum_{m=-l}^{l}$$

The left-hand side of eq 11 is easily computed in the spherical harmonics representation:

$$(\mathcal{S}_{\mathbb{S}^2}\sigma_j)(\mathbf{s})=\int_{\mathbb{S}^2}\frac{\sum_{lm}[X_j]_l^m Y_l^m(\mathbf{s}')}{|\mathbf{s}-\mathbf{s}'|}\mathrm{d}\mathbf{s}'=\sum_{lm}\frac{4\pi}{2l+1}[X_j]_l^m Y_l^m(\mathbf{s})$$

This means that the $\mathcal{S}_{\mathbb{S}^2}$ operator is diagonal in the spherical harmonics basis, and in particular, if we call L_{jj} its representation (the jj subscript means that both $\mathbf{s}$ and the integration variable belong to the same sphere Γ_j):

$$L_{ll'}^{mm'}=\frac{4\pi}{2l+1}\delta_{ll}\delta_{mm'} \tag{12}$$

The computation of the reaction potential produced by a local ASC on a neighboring sphere is more complex. Let $\mathbf{v}^{jk}(\mathbf{s})$ be the vector pointing from the center of sphere Ω_k to $\mathbf{s}\in\Gamma_j$, that is,

$$\mathbf{v}^{jk}(\mathbf{s})=\mathbf{R}_j+r_j\mathbf{s}-\mathbf{R}_k$$

and let $t^{jk}(\mathbf{s})$ be the length of $\mathbf{v}^{jk}(\mathbf{s})$ divided by the radius of sphere Ω_k and $\mathbf{s}^{jk}(\mathbf{s})$ be the unit vector that represents the direction of $\mathbf{v}^{jk}(\mathbf{s})$, that is,

$$t^{jk}(\mathbf{s}):=\frac{|\mathbf{R}_j+r_j\mathbf{s}-\mathbf{R}_k|}{r_k}\quad\text{and}\qquad \mathbf{s}^{jk}(\mathbf{s}):=\frac{\mathbf{R}_j+r_j\mathbf{s}-\mathbf{R}_k}{|\mathbf{R}_j+r_j\mathbf{s}-\mathbf{R}_k|}$$

If Ω_k is a sphere intersecting Ω_j at $\mathbf{s}\in\Gamma_j^i$ and σ_k is the (scaled) solution on Ω_k, the potential produced by σ_k at $\mathbf{s}$ (see eq 9) is

$$(\tilde{\mathcal{S}}_{jk}\sigma_k)(\mathbf{s})=\int_{\mathbb{S}^2}\frac{\sigma_k(\mathbf{s}')}{|t^{jk}(\mathbf{s})\mathbf{s}^{jk}(\mathbf{s})-\mathbf{s}'|}\mathrm{d}\mathbf{s}' \tag{13}$$

Introducing the spherical harmonics expansion for σ_k and using the spherical harmonics addition theorem to expand the denominator, one gets

$$(\tilde{\mathcal{S}}_{jk}\sigma_k)(\mathbf{s})=\sum_{lm}\frac{4\pi}{2l+1}[X_k]_k^m[t^{jk}(\mathbf{s})]^l Y_l^m[\mathbf{s}^{jk}(\mathbf{s})]$$

In order to get the representation matrix $[L_{jk}]$ of $\tilde{\mathcal{S}}_{jk}$, we need to perform a further integration. Unfortunately, this integral can not be computed analytically: a numerical quadrature is mandatory. For this particular problem, the Lebedev quadrature rule is particularly appropriate. Let $\{\mathbf{y}_n,\boldsymbol{w}_n\}_{n=1}^{N_g}$ be the N_g Lebedev points and weights for the unit sphere:

$$[L_{jk}]_{ll'}^{mm'}=-\int_{\mathbb{S}^2}\omega^{jk}(\mathbf{s})[\tilde{\mathcal{S}}_{jk}Y_{l'}^{m'}](\mathbf{s})Y_l^m(\mathbf{s})d\mathbf{s}\simeq-\sum_{n=1}^{N_g}w_n Y_l^m(\mathbf{y}_n)\omega_n^{jk}\frac{4\pi}{2l'+1}(t_n^{jk})^{l'}Y_{l'}^{m'}(\mathbf{s}_n^{jk})$$

where we have added the weight function as we need to integrate only on the portion of surface which is inside Ω_k and, for brevity, we have put

dx.doi.org/10.1021/ct400280b | J. Chem. Theory Comput. 2013, 9, 3637−3648

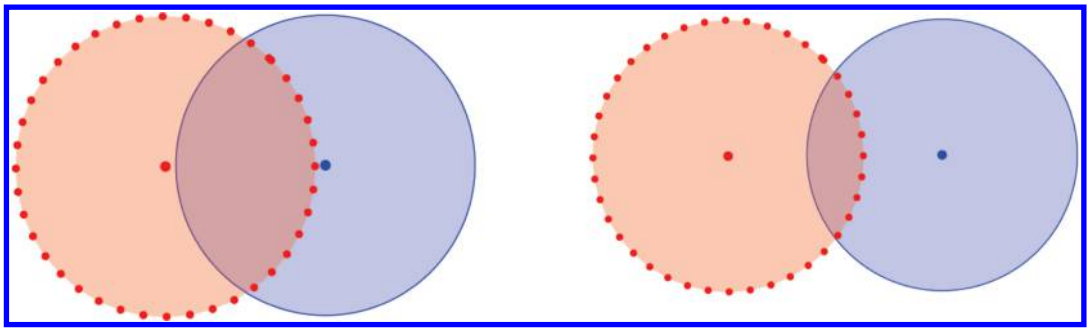


Figure 3. By changing the distance of two spheres, some integration point can suddenly become buried or exposed.

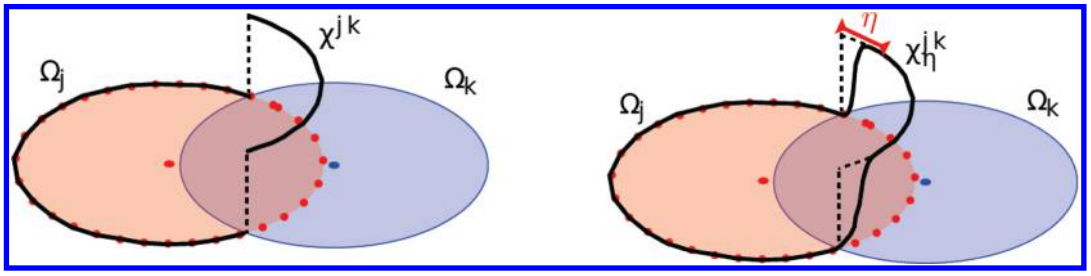


Figure 4. Sharp and smooth switching between external and internal points.

$$t_n^{jk} = t^{jk}(\mathbf{y}_n), \quad \mathbf{s}_n^{jk} = \mathbf{s}^{jk}(\mathbf{y}_n), \quad \omega_n^{jk} = \omega^{jk}(\mathbf{y}_n) \tag{14}$$

The numerical quadrature introduces a source of discontinuity of the computed quantities with respect to the atomic positions, as changing the distance between two atoms (and therefore, between two spheres), an integration point can become suddenly exposed or buried (see Figure 3 for a pictorial representation of the problem). In order to avoid this problem, it is possible to regularize the switching between internal and external surface by introducing a smoothed characteristic function $\chi_\eta(t)$ defined as follows:

$$\chi_\eta(t) = \begin{vmatrix} 1 & \text{if} \quad t \le 1 - \eta \\ p_\eta(t) & \text{if} \quad 1 - \eta < t < 1 \\ 0 & \text{if} \quad t \ge 1 \end{vmatrix} \tag{15}$$

where the parameter η identifies a "switching region" (see Figure 4 for a pictorial view) between internal and external and where

$$p_\eta(t) = \eta^{-5}(1 - t)^3(6t^2 + (15 - 12)t + 10\eta^2 - 15\eta + 6)$$

Using such a definition for the characteristic function, we can introduce a new set of weights for the right-hand side of eq 11, which allows for a smooth definition of all the computed quantities. In particular, let

$$\chi_n^{jk} = \chi_\eta(t_n^{jk}), \quad f_n^j = \sum_{k \in \mathcal{N}_j} \chi_n^{jk}, \quad d_n^j = \frac{\min(f_n^j, 1)}{f_n^j} \tag{16}$$

where $\mathcal{N}_j$ is the list of all neighbors of sphere j (we use the convention $0/0 = 0$ in the definition of d_n^j). We can now define

$$W_n^{jk} = d_n^j \chi_n^{jk}, \quad U_n^j = 1 - \sum_{k \in \mathcal{N}_j} W_n^{jk} \tag{17}$$

Notice that U_n^j can be used to identify whether a point $\boldsymbol{y}_n$ belongs to the external portion of Γ_j ($U_n^j = 1$), to a zone of switching ($0 < U_n^j < 1$) or whether the point is completely buried inside the cavity ($U_n^j = 0$). Notice that $f_n^j = 0$ if the point $\mathbf{y}_n$ belongs to the Γ_j^e, $f_n^j \ge 1$ if $\mathbf{y}_n$ is buried inside the cavity and $0 < f_n^j < 1$ if the point belongs to a switching region. We can use the newly defined W_n^{jk} weights in the definition of the L_{jk} matrix blocks

$$[L_{jk}]_{ll'}^{mm'} = -\sum_{n=1}^{N_g} w_n Y_l^m(\mathbf{y}_n) W_n^{jk} \frac{4\pi}{2l' + 1} (t_n^{jk})^{l'} Y_{l'}^{m'}(\mathbf{s}_n^{jk}) \tag{18}$$

and the U_n^j weights to compute the spherical harmonics expansion of the molecular potential

$$[g_j]_l^m = -\sum_{n=1}^{N_g} w_n Y_l^m(\mathbf{y}_n) U_n^j \Phi_n^j \tag{19}$$

where we have introduced

$$\Phi_n^j = \sum_{k=1}^{M} \frac{q_k}{|\mathbf{R}_j + r_j \mathbf{y}_n - \mathbf{R}_k|}$$

We have now all the pieces to define the linear system that represents the discretized ddCOSMO coupled integral equations:

$$\begin{pmatrix} L_{11} & \cdots & L_{1M} \\ \vdots & \ddots & \vdots \\ L_{M1} & \cdots & L_{MM} \end{pmatrix} \begin{pmatrix} X_1 \\ \vdots \\ X_M \end{pmatrix} = \begin{pmatrix} g_1 \\ \vdots \\ g_M \end{pmatrix} \tag{20}$$

For the sake of brevity, we will also write eq 20 as $LX = g$. Notice that, as W_n^{jk} vanishes if the spheres j and k do not overlap, only the off-diagonal blocks relative to intersecting spheres are nonzero: the matrix L is therefore block sparse. To conclude, the COSMO polarization energy can be calculated as

$$E_s = \sqrt{\pi} f(\varepsilon_s) \sum_{j=1}^{M} q_j [X_j]_0^0 = f(\varepsilon_s) \sum_{j=1}^{M} \langle X_j, \Psi_j \rangle \tag{21}$$

dx.doi.org/10.1021/ct400280b | J. Chem. Theory Comput. 2013, 9, 3637−3648

where we have introduced the auxiliary function

$$[\Psi_j]_l^m = \sqrt{\pi}\, q_j \delta_{l0}\delta_{m0}$$

and the notation

$$\langle a, b\rangle = \sum_{lm} [a]_l^m [b]_l^m$$

We will show in section 4 how it is possible to implement an efficient, iterative solution to the ddCOSMO equations achieving linear scaling in both computational cost and memory requirements.

3. ANALYTICAL DERIVATIVES

In this section, we will describe the procedure to compute the analytical derivatives of the COSMO solvation energy. By differentiating eq 21 with respect to the position of the center of the i-th sphere:

$$\begin{aligned}
-\mathbf{F}_i &= \nabla_i E_s \\
&= f(\varepsilon_s) \sum_{j=1}^{M} \langle \Psi_j, \nabla_i X_j\rangle \\
&= f(\varepsilon_s) \sum_{j=1}^{M} \langle \Psi_j, \nabla_i (\sum_{k=1}^{M} L_{jk}^{-1} g_k)\rangle \\
&= f(\varepsilon_s) \sum_{j=1}^{M} \langle \Psi_j, \sum_{k} L_{jk}^{-1} \nabla_i g_k\rangle - f(\varepsilon_s) \\
&\quad \sum_{k=1}^{M} \langle \Psi_j, \sum_{k=1}^{M} (L^{-1}(\nabla_i L) L^{-1})_{jk} g_k\rangle
\end{aligned} \tag{22}$$

By rearranging

$$-\mathbf{F}_i = f(\varepsilon_s) \sum_{j=1}^{M} \left\langle \underbrace{\sum_{k=1}^{M} (L_{jk}^{-1})^T \Psi_k}_{s_j}, \underbrace{\left(\nabla_i g_j - \sum_{k=1}^{M} (\nabla_i L_{jk}) X_k\right)}_{h_{ij}} \right\rangle = f(\varepsilon_s) \sum_{j=1}^{M} \langle s_j, h_{ij}\rangle, \tag{23}$$

where $s = (s_1, s_2, \ldots s_M)^T$ is the solution to the adjoint linear system

$$L^T s = \Psi \tag{24}$$

$\Psi = (\Psi_1, \Psi_2, \ldots \Psi_M)^T$ and

$$[\mathbf{h}_{ij}]_l^m = \nabla_i [g_j]_l^m - \sum_k (\nabla_i [L_{jk}]_{ll'}^{mm'}) [X_k]_{l'}^{m'} \tag{25}$$

The computation of the forces therefore involves two separate steps: solving the adjoint linear system and assembling the explicit derivative terms $[\mathbf{h}_{ij}]_l^m$. For the adjoint linear system we notice that

$$L^T = \begin{pmatrix} L_{11}^T & \ldots & L_{M1}^T \\ \vdots & \ddots & \vdots \\ L_{1M}^T & \ldots & L_{MM}^T \end{pmatrix} \tag{26}$$

where the diagonal blocks of the L matrix are self-adjoined:

$$[L_{jj}^T]_{ll'}^{mm'} = \delta_{ll'}\delta_{mm'} \frac{4\pi}{2l+1}$$

and the off-diagonal blocks are defined as follows:

$$\begin{aligned}
[L_{kj}^T]_{ll'}^{mm'} &= -\frac{4\pi}{2l+1} [c_{kj}]_{l'l}^{m'm} \\
&= -\frac{4\pi}{2l+1} \sum_{n}^{I(j,k)} w_n Y_{l'}^{m'}(\mathbf{y}_n) W_n^{kj} (t_n^{kj})^l Y_l^m(\mathbf{s}_n^{kj})
\end{aligned}$$

Notice that, of course, also L^T is a block sparse matrix.

The explicit contributions can be divided into two sets, as in eq 25: the ones involving the derivatives of the RHS, and therefore of the solute's potential, and the ones involving the derivatives of the L matrix. The first term is obtained by differentiating eq 19:

$$[\mathbf{G}_{ij}]_l^m = \sum_{n=1}^{N_g} w_n Y_l^m(\mathbf{y}_n)(\Phi_n^j \nabla_i U_n^j + U_n^j \nabla_i \Phi_n^j) \tag{27}$$

Here, the derivatives of the potential are standard quantities. The derivatives of the weights are

$$\nabla_i U_n^i = -\mathbf{Z}_n^i, \qquad \nabla_i U_n^j = \frac{1}{R_i} p_\eta'(t_n^{ji}) \mathbf{s}_n^{ji}, \qquad i \in \mathcal{N}_j \tag{28}$$

where

$$\mathbf{Z}_n^i = \sum_{k \in \mathcal{N}_i} \frac{1}{R_k} p_\eta'(t^{ik}) \mathbf{s}^{ik}$$

and

$$p_\eta'(t) = -\frac{30}{\eta^5}(1-t)^2(t-1+\eta)^2$$

is the derivative of the switching function.

The second family of contributions arising from eq 25, that is, the ones of the form

$$[\mathbf{K}_{ij}]_l^m = \sum_{l'm'} \sum_{k \in \mathcal{N}_j} \nabla_i [L_{jk}]_{ll'}^{mm'} [X_k]_{l'}^{m'} \tag{29}$$

involves the evaluation of the gradient of the product of several terms, which produces very cumbersome expressions. Notice that three different cases arise, depending on whether $i = j \neq k$, $i = k \neq j$ or $j \neq i \neq k$. For the sake of brevity, we will report here only the final expression of the $i = j$ case:

$$\begin{aligned}
[\mathbf{K}^{ii}]_l^m = \sum_{k \in \mathcal{N}_i} \sum_{l'm'} \frac{4\pi}{2l'+1} \delta_{ij} \Bigg\{ & \sum_{n}^{I(i,k)} \frac{w_n}{R_k} Y_l^m(\mathbf{s}_n) W_n^{ik} \delta_{l'\geq 1} \\
& \times (t_n^{ik})^{l'-1} (l' Y_{l'}^{m'}(\mathbf{s}_n^{ik}) \mathbf{s}_n^{ik} + \nabla Y_{l'}^{m'}(\mathbf{s}_n^{ik})) \\
& + \sum_{n}^{S(i,k)} \frac{w_n}{R_k} Y_l^m(\mathbf{y}_n) (t_n^{ik})^{l'} Y_{l'}^{m'}(\mathbf{s}_n^{ik}) d_n^i p'_\eta(t_n^{ik}) \mathbf{s}_n^{ik} \\
& - \delta_{f_n^i > 1} \sum_{n}^{I(i,k)} w_n Y_l^m(\mathbf{y}_n) (t_n^{ik})^{l'} Y_{l'}^{m'}(\mathbf{s}_n^{ik}) (d_n^i)^2 \chi_n^{ik} Z_n^i \Bigg\} [\sigma_k]_{l'}^{m'}
\end{aligned} \tag{30}$$

where $\delta_{f_n^i} > 1 = 1$ if $f_n^i > 1$ and zero otherwise. For the sake of brevity, we have identified with $I(j,k)$ the set of points on Γ_j that are inside Ω_k, that is,

$$I(j,k) = \{\mathbf{y}_n | t_n^{jk} < 1\}$$

and with $S(j,k) \subset I(j,k)$ the set of points on Γ_j that are in the switching region between spheres j and k, that is,

$$S(j,k) = \{\mathbf{y}_n | 1 - \eta < t_n^{jk} < 1\}$$

dx.doi.org/10.1021/ct400280b | J. Chem. Theory Comput. 2013, 9, 3637−3648

The other cases produce similar expressions that are reported in Appendix B. Such contributions can be assembled with the geometrical quantities already defined and the gradients of the basis functions. We will show in section 4 how it is possible to compute the forces efficiently and with linear scaling in computational cost and memory requirements.

4. EFFICIENT, PARALLEL, AND LINEAR SCALING IMPLEMENTATION

This section is devoted to the description of the actual implementation of the ddCOSMO equations. We will show the feasibility of computing all the needed quantities with both computational costs and memory requirements linear with respect to the number of atoms. We will start by discussing the iterative procedure to solve the linear system eq 20; we will then show how the contributions to the forces can also be computed achieving linear scaling cost by discussing the terms already presented in section 3. In Appendix A we will dedicate some attention to an efficient procedure for the evaluation of a SH set given a vector on the unit sphere, which is the most CPU intensive task involved in the algorithm.

To compute the interaction energy (eq 21), we need to compute the discretized local ASCs, that is, their expansion coefficients $[X_j]_l^m$, by solving the linear system in eq 20. This can be efficiently done by means of Jacobi iterations, that is, solving for each j, at iteration "it":

$$[L_{jj}X_j^{(it)}]_l^m = [g_j]_m^l - \sum_{k\in\mathcal{N}_j}\sum_{l'm'}[L_{jk}]_{ll'}^{mm'}[X_k^{(it-1)}]_{l'}^{m'}:$$
$$= [V_j^{(it)}]_l^m \tag{31}$$

Notice that eq 20 is the discretized equivalent to eq 10. The diagonal blocks of the L matrix are diagonal (see eq 12), which means that, once the right-hand side (RHS) has been assembled, the linear system is trivially solved. Not only: the sparsity of the L matrix ensures that only the spheres that intersect j will give a contribution to the RHS: each iteration will therefore require a computational effort which is, for each sphere, proportional to the number of neighbors, which is much smaller than the total number of atoms for a large enough molecule. Notice that the number of neighbors is determined by the molecular connectivity: the scaling properties will hence be similar for a linear or a globular molecule. This is a key feature of the ddCOSMO algorithm that is not shared by other fast algorithms, similar to the Fast Multipole Method (FMM) used in conjunction with a standard finite element discretization of the COSMO equations.[29] We will now detail our iterative procedure.

Preliminary Steps. Read the positions of the centers of the spheres and the charges; choose and build the Lebedev grid $\{\mathbf{y}_n\}_{n=1}^{N_g}$ and the associated weights $\{w_n\}_{n=1}^{N_g}$; at each grid point, compute $\{Y_l^m(\mathbf{y}_n)\}_{0\le l\le N}^{-l\le m\le l}$. Assemble a neighbor list and other quantities needed to handle the sparse storage: all the quantities that formally depend on two sphere indexes such as the t_n^{jk} can be precomputed and stored in memory using $O(N_g \times M)$ words of memory. Allocate the needed amount of memory and compute all the geometrical quantities defined in eqs 14, 16, and 17. As a last preliminary step, compute the solute's potential Φ_n^j. This can be done efficiently using standard fast summation techniques, such as the FMM. In this implementation, we employed the fully adaptive FMM3DLIB[39,40] package for such a purpose. In simulation softwares, this step is already handled by the standard machinery and can be considered as the input to the ddCOSMO module. The RHS can finally be assembled by weighting the potential with the U_n^j factors. Notice that the RHS will be equal to the electrostatic potential on the external points, but will be nonvanishing also in the switching region. The total cost of this step is linear thanks to the FMM and requires one to allocate a vector g_n^j of size $M \times N_g$, plus some (linear) scratch for the FMM procedure.

The iterative procedure consists of three main steps: calculation of the RHS, solution of the diagonal system, and test of the convergence.

Step 1: Calculation of the RHS. The first step is the most computationally demanding and can be described as follows. We will need a scratch vector P_n^j of size $M \times N_g$. For each sphere j, loop over the grid points $\mathbf{y}_n$ and check whether they are external (i.e., if $U_n^j = 1$) or internal. If the points are external, put $P_n^j = g_n^j$, else loop over the neighboring spheres $k \in \mathcal{N}_j$. If $\mathbf{y}_n \in \Omega_k$, that is, if $t_n^{jk} < 1$, compute $\{Y_m^l(\mathbf{s}_n^{jk})\}$, where $\mathbf{s}_n^{jk}$ is the projection of $\mathbf{y}_n$ on Γ_k, then compute

$$\alpha = \sum_{lm}\frac{4\pi}{2l+1}(t_n^{jk})^l Y_m^l(\mathbf{s}_n^{jk})[X_k^{(it-1)}]_l^m \tag{32}$$

We can now accumulate the contribution to P_n^j from the k-th sphere

$$P_n^j = P_n^j + \alpha \times W_n^{jk} \tag{33}$$

and, when the loop over the neighboring spheres is complete, add the contribution due to the solute's potential in the switching region (i.e., where $0 < U_n^j < 1$). Finally, by numerical quadrature, we compute

$$[V_j^{(it)}]_m^l = \sum_{n=1}^{N_g} w_n Y_l^m(\mathbf{y}_n)P_n^j \tag{34}$$

which is the right-hand side of eq 31. See algorithm 1 for a sketched summary of the computation of $\mathbf{P}_n^j$.

Algorithm 1 Computation of the P_n^j vector for sphere j

```
for n = 1, N_g do
    if U_n^j = 1 then
        P_n^j = g_n^j
    else
        for k ∈ N_i do
            if t_n^jk < 1 then
                Assemble {Y_l^m(s_n^jk)}
                Compute α as in eq. 32 using [X_k^(it-1)]
                P_n^j = P_n^j + α × W_n^jk
            end if
        end for
        P_n^j = P_n^j + g_n^j
    end if
end for
```

Here, we notice some important features of the algorithm. First, the total computational cost of a single iteration can be estimated by $O(M \times N_g)$, where the prefactor depends on the average number of neighbors, on the number of points which are buried inside the cavity and on the maximum angular momentum N used for the SH expansion. Second, the quantity of memory needed scales linearly with respect to the size of the system as well, as the only quantities that we need to store are two vectors of size $M \times N_g$ (i.e., P_n^j and g_n^j) and three vectors of size $M \times (N+1)^2$ (i.e., $[V_m^{(it)}]_m^l$, $[X_m^{(it)}]_m^l$, and $[X_m^{(it-1)}]_m^l$). Finally, for each sphere j, the only information needed to perform the iteration are either geometrical parameters, the solute's potential and quantities that can be computed on the fly, such as the SH, or the solution vector

dx.doi.org/10.1021/ct400280b | J. Chem. Theory Comput. 2013, 9, 3637−3648

on the neighboring spheres at the previous iteration: it is hence possible to parallelize the algorithm by distributing each sphere or group of spheres on a separate processor.

Step 2: Solution of the Diagonal System. Once the RHS of eq 31 has been computed, it is possible to compute the solution at the iteration step. Due to the fact that the $\mathcal{S}_{\mathbb{S}^2}$ operator is diagonal in the SH representation, this step can be done trivially:

$$[X_j^{(\mathrm{it})}]_l^m = \frac{2l+1}{4\pi}[V_j^{(\mathrm{it})}]_m^l \tag{35}$$

Step 3: Convergence Check and Acceleration. For each sphere, it is now possible to evaluate

$$\Delta_j = \left\| X_j^{(\mathrm{it})} - X_j^{(\mathrm{it}-1)} \right\| = \sqrt{\sum_{lm} \frac{([X_j^{(it)}]_l^m - [X_j^{(\mathrm{it}-1)}]_l^m)^2}{l+1}} \tag{36}$$

where we have chosen to use the energy norm of the increment. When the vector Δ has been assembled, we exit from the parallel region, evaluate the RMS norm of Δ, and compare it with a threshold to check the convergence. This simple iterative procedure is known as the Jacobi iterative method. The incremental solution $\boldsymbol{X}_m^{(\mathrm{it})} - \boldsymbol{X}_j^{(\mathrm{it}-1)}$ can be also used as an error vector for the DIIS[41] extrapolation, which has proven to be a good way to accelerate the convergence of the Jacobi iterations. A summary of the iterative procedure is sketched in Algorithm 2. A similar procedure can be used to solve the adjoint linear system, which is a mandatory step for the computation of the forces.

Algorithm 2 Algorithm for the iterative solution of the linear equations.

```
Read the spheres centers and radii, compute the Lebedev grid and SH at each grid point
Compute the neighbor lists and the quantities t_n^jk, s_n^jk, χ_n^jk, d_n^j, W_n^jk, U_n^j, Z_n^j and Q_n^j
Compute g_n^j = U_n^j Φ_n^j
for it = 1, ItMax do
    Parallel Region (j)
    for j = 1, M do
        Assemble the P_n^j vector as in algorithm 1 using [X^(it-1)]
        Assemble [V_j^(it)]_l^m as in eq. 34
        Solve for [X_k^(it)]_lm^l as in eq. 35
        Evaluate Δ_j
    end for
    End Parallel Region
    Compute ||Δ||
    if Converged then
        Return
    else
        Perform DIIS extrapolation and save [X^(it)].
    end if
end for
```

We will now show that also the explicit contributions to the forces can be computed with linear effort. As said in section 3, the contributions involving the derivatives of the matrix are very cumbersome: we will focus on the one reported in eq 30. By fully expanding the contribution arising from such term (which we will denote $\mathbf{K}_{\mathrm{A}}^i$) we get:

$$\begin{aligned}\mathbf{K}_{\mathrm{A}}^i = \sum_{lm}[\mathbf{s}_i]_l^m \sum_{k\in\mathcal{N}_i}\sum_{l'm'}\frac{4\pi}{2l'+1}\Bigg\{ &\sum_n^{I(i,k)}\frac{w_n}{R_k}Y_l^m(\mathbf{y}_n)\\ &W_n^{ik}\delta_{l',\geq 1}(t_n^{ik})^{l'-1}(l'Y_{l'}^{m'}(\mathbf{s}_n^{ik})\mathbf{s}_n^{ik} + \nabla Y_{l'}^{m'}(\mathbf{s}_n^{ik}))\\ &+ \sum_n^{S(i,k)}\frac{w_n}{R_k}Y_l^m(\mathbf{y}_n)(t_n^{ik})^{l'}Y_{l'}^{m'}(\mathbf{s}_n^{ik})d_n^i\mathbf{p}'_\eta(t_n^{ik})\mathbf{s}_n^{ik}\\ &- \delta_{f_n^i>1}\sum_n^{I(i,k)} w_n Y_l^m(y_n)(t_n^{ik})^{l'}Y_{l'}^{m'}(\mathbf{s}_n^{ik})(d_n^i)^2\chi_n^{ik}\mathbf{Z}_n^i\Bigg\}[\sigma_k]_{l'}^{m'}\end{aligned} \tag{37}$$

For each sphere i, loop over the $\boldsymbol{y}_n$ grid points and, for each $\mathbf{y}_n$, loop over the neighboring spheres $k \in \mathcal{N}_i$. If $\mathbf{y}_n \in \Omega_k$, that is, if $t_n^{ik} < 1$, compute a SH set $\{Y_l^m(\mathbf{s}_n^{ik})\}$ and its derivatives $\{\nabla Y_l^m(\mathbf{s}_n^{ik})\}$. Now, compute all the sums involving $[X_k]_l^m$:

$$\mathbf{a} = \sum_{l'=1}^{N}\sum_{m'=-l}^{l'}\frac{4\pi}{2l'+1}(t_n^{ik})^{l'-1}(l'Y_{l'}^{m'}(\mathbf{s}_n^{ik})\mathbf{s}_n^{ik} + \nabla Y_{l'}^{m'}(\mathbf{s}_n^{ik}))[X_k]_{l'}^{m'}$$

$$b = \sum_{l'm'}\frac{4\pi}{2l'+1}(t_n^{ik})^{l'}Y_{l'}^{m'}(\mathbf{s}_n^{ik})[X_k]_{l'}^{m'}$$

Then, accumulate the sum over the neighbors: put

$$\mathbf{v} = \mathbf{v} + \frac{W_n^{ik}}{R_k}\times\mathbf{a} + (d_n^i)^2(\chi_n^{ik})^2\mathbf{Z}_n^i\times b$$

If $\mathbf{y}_n$ belongs to the switching region between i and k (i.e., if $1-\eta < t_n^{ik} < 1$), accumulate also the following contribution:

$$\mathbf{v} = \mathbf{v} + \frac{2}{R_k}d_n^i\chi_n^{ik}p'_\eta(t_n^{ik})\mathbf{s}_n^{ik}b$$

Finally, evaluate the sum involving $[s_i]_l^m$

$$\alpha = \sum_{lm}[s_i]_l^m Y_l^m(\mathbf{y}_n)$$

and accumulate all the contributions for each $\boldsymbol{y}_n$:

$$\mathbf{K}_{\mathrm{A}}^i = \mathbf{K}_{\mathrm{A}}^i - w_n\alpha\mathbf{v} \tag{38}$$

Notice that we used a quantity of scratch memory that does not depend on the size of the system and that we did an amount of work, which is $O(M\times N_g)$, where the prefactor depends again on the average number of neighbors, on the number of points buried inside the cavity and on the maximum angular momentum N used for the SH expansion. Notice also that each i contribution can be computed in parallel by distributing each sphere or group of spheres on a separate processor. The other two terms involving the derivatives of the matrix can be computed in a similar fashion starting from the same quantities already used for eq 38.

The terms arising from the derivatives of the RHS of eq 11 involve both the computation of the derivatives of the solute's potential, for which the FMM can be used to achieve linear scaling in computational cost, and the evaluation of eq 28. For these latter terms, as all the two-index quantities are nonzero only for neighboring spheres, both the computational cost and the memory requirements remain linear with respect to the number of spheres M.

5. NUMERICAL RESULTS

In this section, we present some test cases to show the performance of our methodology. In particular, we will confirm numerically the linear scaling properties of our algorithm for both linear and globular systems, together with the overall excellent timings. We choose here three categories of systems: a chain of alanine peptides, a large, tetrameric protein and its subunits, and a spherical cluster of water molecules. The first system is linear, while the latter two are globular; furthermore, both the alanine chain and the water cluster can be made arbitrarily large by simply adding more and more residues.

The algorithm, as described in section 4, has been implemented in a stand-alone FORTRAN 77 code. OpenMP

shared memory parallelism has been used to parallelize the calculation, by distributing the spheres among the processors. All the calculations for which we report absolute timings have been performed on a single node equipped with two Xeon X5690@ 3.47 GHz hexacore processors and 24 GB of DDR3@1333 MHz physical memory. The ifort FORTRAN compiler from the Intel Parallel Studio XE 2013 suite of compilers has been used together with machine specific optimization flags (-xSSE4.2) and aggressive optimization (-O3). As the elapsed time of a computation can be subject to small variation, due for instance to little differences in the background activity of the operating system, all the computation have been repeated ten times: in the figures, the standard deviation on the timings is reported as an error bar. We report in the following the elapsed time for the solution of the ddCOSMO linear system eq 20 and adjoint problem eq 24 and the time needed to compute the explicit contributions to the forces that involve ddCOSMO-related quantities. These three operations are, in fact, the ones which depend explicitly on the algorithm used to solve the ddCOSMO equations; to obtain the total elapsed time of a computation, one needs to add the time to compute the cavity-related quantities, the potential and the potential derivatives. As we have already mentioned, the potential and potential derivatives are computed in our code by means of the FMM3DLIB; as far as the cavity-related quantities (i.e., the Lebedev quadrature points and weights and the neighbor lists) efficient implementations already exist:[21,29,42] we will not deal with these aspects in this communication. We also report the quantity of memory allocated for the computation, which we divide in basic (i.e., the memory needed to store the solution, the scratch arrays, the external potential the grid and other basic quantities) and geometrical (i.e., the memory needed to store the geometrical parameters t_n^{ik},$\mathbf{s}_n^{ik}$,...): this latter memory storage can be avoided by computing the geometrical quantities on-the-fly. The discretization parameters used for all the computations are $\eta = 0.1$, $N = 10$, and $N_g = 302$; the threshold for the convergence of the iterative solution (see eq 36) has been fixed to 10^{-6}. In Figure 5, the results for various alanine chains are reported. The number of residues per chain varies in a range from 20 units (i.e., 2000 atoms) to 400 units (40 000 atoms). To give a taste of the

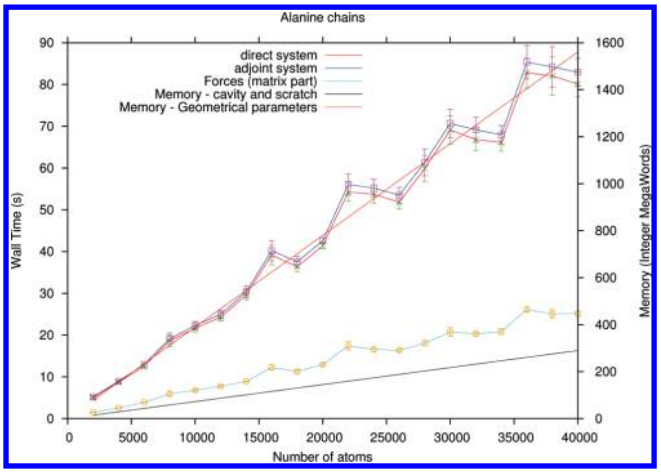


Figure 5. Linear molecules: alanine chains as a test case. The elapsed time to solve the ddCOSMO equations, the adjoint equations, and to compute the explicit contributions to the forces are reported together with the memory needed to perform the computation and to precompute the geometrical parameters.

computational size of the problem, the number of basis functions (i.e., the size of the linear system) varies from more than two hundred thousands up to almost five millions, the average number of neighbors per sphere is approximately 20 and, on average, approximately 255 points out of 302 lye on the internal surface. As stated in section 4, these latter two quantities give a measure of the computational cost per iteration per sphere. It is possible to see how the predicted linear scaling behavior shows in Figure 5, where the noisy curves are to be ascribed to the nonuniform memory access of the various cores and to the use of the processors cache, which is determined by the compiler and non trivial to analyze. It is remarkable that also the memory requirements scale perfectly linearly with the size of the system. Notice also how the quantity of memory necessary to store the geometrical parameters is larger than the one needed for the computation: even smaller memory requirements can be obtained by computing the geometrical parameters on-the-fly. To see how this affects the computation, we report in Figure 6

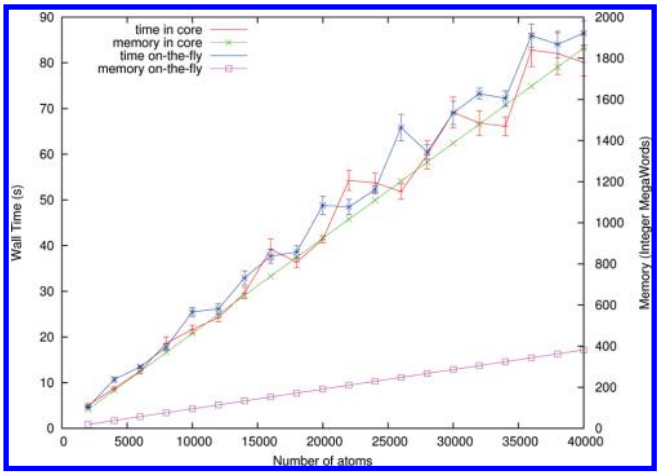


Figure 6. In core vs on-the-fly solution of the linear system for alanine chains.

the wall time required to solve the equations by precomputing the geometrical parameters ("in core"), which is taken from Figure 5, and the time to solve the same equations but by calculating all the parameters on-the-fly. The differences are minimal, with the on-the-fly algorithm requiring only a very small amount of time more than its in core counterpart; on the other hand, the memory required is much smaller: the on-the-fly algorithm is the way to go when moving to very large systems. Finally, the absolute timings are *per se* impressive: considering the size of the linear systems implied (up to almost 5×10^6), it is noteworthy that only a few minutes are required to compute the COSMO polarization energy and contributions to the forces even for very large systems on a single node. Similar results are obtained for nonlinear molecules, such as a protein.

In Figure 7, the various timings are reported for a biological structure (PDB structure 3S48,[43] to which we will refer improperly as Hemoglobin), by starting from one subunit up to the full protein. It is interesting to notice how both the average number of internal points (257−259) and of neighbors per sphere (18) are actually smaller than for the linear chains: we can expect the cost per iteration for the globular protein to be similar or even smaller than the one for the alanine chains. To confirm this in a more systematic way, we run the usual set of calculations on spherical clusters of water molecules, which we cut starting from a pre-equilibrated cubic box. The timings are reported in

dx.doi.org/10.1021/ct400280b | J. Chem. Theory Comput. 2013, 9, 3637−3648

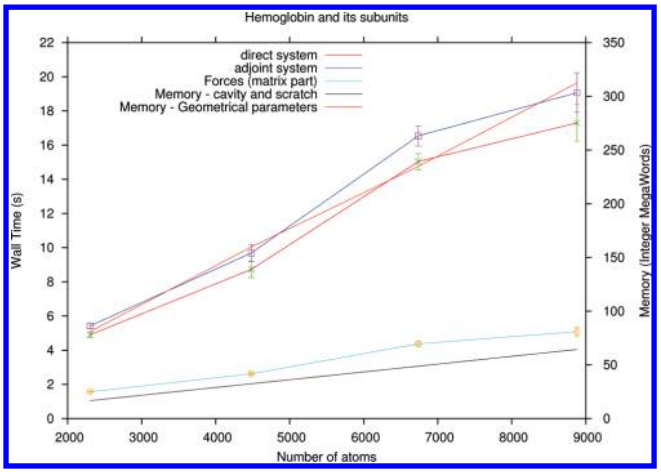


Figure 7. Globular molecules: Hemoglobin chains as a test case. The elapsed time to solve the ddCOSMO equations, the adjoint equations, and to compute the explicit contributions to the forces are reported together with the memory needed to perform the computation and to precompute the geometrical parameters.

Figure 8; the sizes of the clusters vary between 411 and 31104 atoms, the average number of neighbors between 12 and 15 and

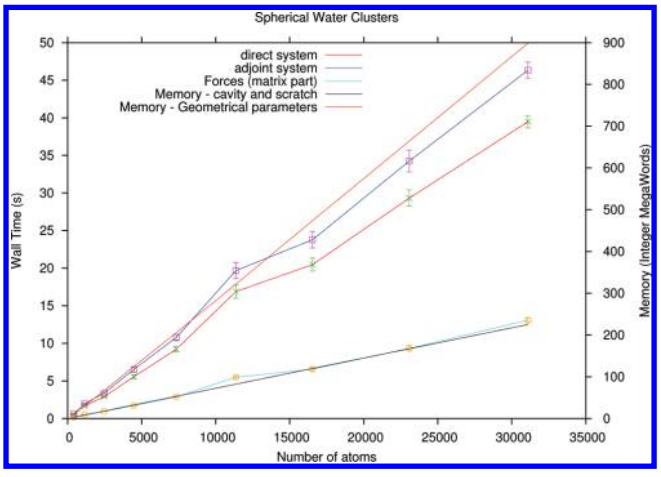


Figure 8. Globular systems: spherical water clusters as a test case. The elapsed time to solve the ddCOSMO equations, the adjoint equations, and to compute the explicit contributions to the forces is reported together with the memory needed to perform the computation and to precompute the geometrical parameters.

the average number of internal points between 236 and 255. Again, a very nice linear scaling behavior is shown also for this very globular system, both for the CPU time and for the memory requirements. Here, we see that the average number of neighbors is again much lower while, for large enough clusters, the number of internal points is similar to the one for the alanine chains: we expect the cost of the computation for the spherical water cluster to be smaller than for the linear molecules. To make the comparison easier, the time to solve the ddCOSMO equations are reported for each class of systems in Figure 9, showing that the computations for the water clusters are in fact faster than the ones for the linear molecules. This is a very remarkable feature of the ddCOSMO algorithm, as the nice, linear scaling behavior depends on the topology of the molecular system rather than on its size: the linear scaling regime can thus be accessed also for globular, not so large systems.

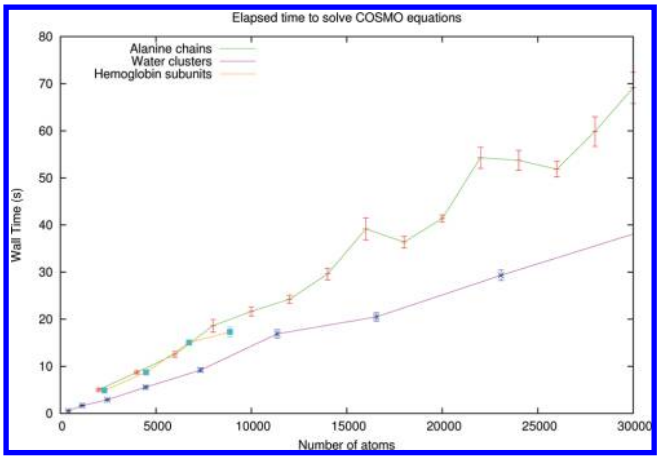


Figure 9. Comparison of the elapsed times to solve the ddCOSMO equations for different classes of molecules.

As already mentioned, the ddCOSMO algorithm is suitable for a parallel implementation. In this work, only some basic shared memory parallelism has been introduced; nevertheless, it is possible to see from Figure 10 that the parallel behavior of the code is good up to 40−48 cores, after which a plateau is observed. An improved parallel implementation is currently under development.

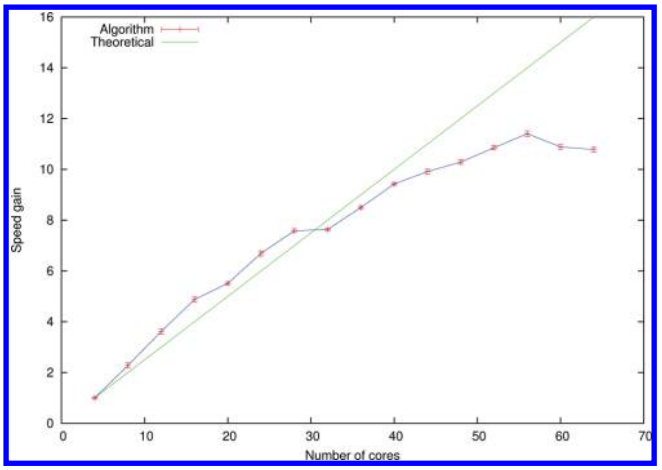


Figure 10. Speed gain by using more shared memory processors.

We conclude this section with a remark on the accuracy of the computed forces with respect to the tightness of the convergence of the linear equations. In Figure 11, the RMS error on the forces with respect to both numerical derivatives and a very tight computation for increasingly tight convergence thresholds is reported for a chain of 5 alanine peptides. We notice that even with the sleaziest threshold, an accuracy better than 10^{-5} Hartree/Bohr is already obtained: an accuracy of 10^{-6} Hartree/Bohr, which can be considered safe both for molecular dynamics simulations and for geometry optimizations is achieved with the 10^{-6} threshold used for all the simulations reported here.

6. CONCLUSIONS

In this paper, an implementation of the domain decomposition paradigm recently presented by some of us is described. The linear scaling properties in both CPU time and memory are detailed and illustrated by means of several numerical examples. The methodology is promising and paves the way to the

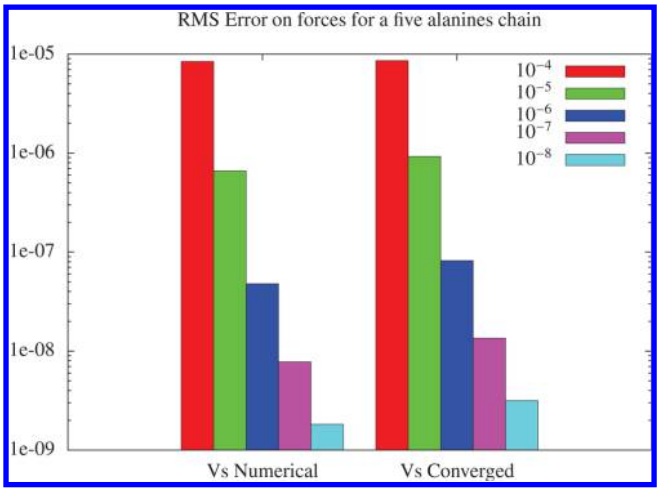


Figure 11. Error on the computed forces for a 5 alanines chain depending on the convergence threshold for the iterative solver. In the left group of columns, we report the RMS difference between the forces computed by means of analytical and numerical derivatives (with a finite step of 10^{-5} Bohrs) for different stopping criteria; in the right group of columns, we report the RMS difference between the analytical derivatives for a given stopping criterion and the ones obtained with a 10^{-10} threshold.

application of polarizable continuum methods to molecular dynamics simulation. Starting from the present, several directions are worthy to be explored, both from a theoretical and a computational point of view. In particular, the extension of the algorithm to the PCM model and to solutes described by means of a quantum mechanical methodology are currently actively investigated. Further refinements in the definition of the cavity, or in the way to decompose the cavity in spheres may also improve the computational efficiency of the ddCOSMO scheme.

APPENDIX A: EFFICIENT EVALUATION OF THE SPHERICAL HARMONICS AND THEIR GRADIENT

In this appendix, we illustrate how to efficiently compute a real SH set (and possibly its gradient) at a given point on the unit sphere $\mathbf{s} = (x,y,z)$, like any $\mathbf{s}_n^{jk}$ vector. We recall that

$$Y_l^m(\theta, \phi) = \begin{cases} C_l^m P_l^{|m|}(\cos\theta)\sqrt{2}\cos(m\phi), & 0 < m \leq l \\ C_l^0 P_l^0(\cos\theta) & m = 0 \\ C_l^m P_l^{|m|}(\cos\theta)\sqrt{2}\sin(m\phi), & -l \leq m < 0 \end{cases}$$

where

$$C_l^m = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}$$

is a normalization constant and $P_l^{|m|}$ is an associated Legendre polynomial. To compute the SH set, one needs therefore to assemble the associated Legendre polynomials and the trigonometric functions. First, remembering that $\mathbf{s} = (x,y,z)$ is a unit vector, let

$$\cos\theta = z, \qquad \sin\theta = \sqrt{1 - z^2}$$

and if $\sin\theta \neq 0$,

$$\cos\phi = \frac{x}{\sin\theta}, \qquad \sin\phi = \frac{y}{\sin\theta}$$

notice that if $\sin\theta = 0$ it is safe to put $\cos\phi = 1$ and $\sin\phi = 0$.

The Legendre functions can be computed by recursion, exploiting the relation

$$(l-m)P_l^m(\cos\theta) = \cos\theta(2l-1)P_{l-1}^m(\cos\theta) - (l+m-1)P_{l-2}^m(\cos\theta)$$

together with $P_l^l(\cos\theta) = (-1)^l(2l-1)!!(1-\cos^2\theta))^{1/2}$.

The trigonometric functions $\cos(m\phi)$ and $\sin(m\phi)$ can be computed without the need of evaluating an inverse trigonometric function and $2 \times N$ transcendental functions, which would be computationally very demanding. In fact, it is possible to exploit the definition of the Chebyshev polynomials of the first kind

$$T_m(\cos\phi) = \cos(m\phi)$$

and to evaluate the Chebyshev polynomials by recursion:

$$T_{m+2}(\cos\phi) = 2\cos(\phi)T_{m+1}(\cos\phi) - T_m(\cos\phi)$$

together with

$$T_0(\cos\phi) = 1, \qquad T_1(\cos\phi) = \cos\phi$$

Notice that the same expression can be used also for the $\sin(m\phi)$ functions by modifying the starting values according to

$$T_0(\sin\phi) = 0, \qquad T_1(\sin\phi) = \sin\phi$$

Finally, the gradient of the SH can be computed with the same machinery, as

$$\nabla Y_l^m(\mathbf{s}) = \frac{\partial Y_l^m(\mathbf{s})}{\partial\theta}\mathbf{e}_\theta + \frac{\partial Y_l^m(\mathbf{s})}{\partial\phi}\mathbf{e}_\phi$$

where

$$\mathbf{e}_\theta = (\cos\theta\cos\phi, \cos\theta\sin\phi, -\sin\theta)$$

and if $\sin\theta \neq 0$,

$$\mathbf{e}_\phi = \left(-\frac{\sin\phi}{\sin\theta}, \frac{\cos\phi}{\sin\theta}, 0\right)$$

The derivatives with respect to ϕ are trivially calculated; for the ones with respect to θ, it is possible to exploit the relation

$$\frac{dP_l^m(\cos\theta)}{d\theta} = -\frac{1}{2}[(l+m)(l-m+1)P_l^{m-1}(\cos\theta) - P_l^{m+1}(\cos\theta)]$$

Notice that, to assemble either the SH or its gradient, only three scratch vectors are needed: one for the associated Legendre polynomials and two for the trigonometric functions, for a total of $(N+1)^2 + 2(N+1)$ integer words of memory. Of course, if the code is executed in parallel, such an amount of memory will be needed for each processor even in the case of shared memory parallelization.

APPENDIX B OTHER CONTRIBUTIONS TO THE FORCES

We report here the contributions arising from eq 29. For $i = k$:

dx.doi.org/10.1021/ct400280b | J. Chem. Theory Comput. 2013, 9, 3637−3648

$$\mathbf{K}_{\mathrm{B}}^{ji} = \sum_{l'm'} \frac{4\pi}{2l'+1}\delta_{ik}\Bigg\{\sum_{n}^{I(j,i)} \frac{w_n}{R_i} Y_l^m(\mathbf{y}_n) W_n^{ji} \delta_{l'\geq 1} (t_n^{ji})^{l'-1} (l' Y_{l'}^{m'}(\mathbf{s}_n^{ji})\mathbf{s}_n^{ji} + \nabla Y_{l'}^{m'}(\mathbf{s}_n^{ji})) + \sum_{n}^{S(j,i)} \frac{w_n}{R_i} Y_l^m(\mathbf{y}_n)(1 - \delta_{f_n^j>1} d_n^j \chi_n^{ji}) d_n^j p'_\eta(t_n^{ji}) (t_n^{ji})^{l'} Y_{l'}^{m'}(\mathbf{s}_n^{ji})\mathbf{s}_n^{ji}\Bigg\}[\sigma_i]_{l'}^{m'}$$

Finally, for $j \neq i \neq k$:

$$\mathbf{K}_{\mathrm{C}}^{i} = \delta_{f_n^j>1} \sum_{k\in N_j\setminus\{i\}} \sum_{l'm'} \frac{4\pi}{2l'+1}\frac{1}{R_i} \sum_{n}^{I(j,k)\cap S(j,i)} w_n Y_l^m(\mathbf{y}_n) (d_n^j)^2 \chi_n^{jk} (t_n^{jk})^{l'} Y_{l'}^{m'}(\mathbf{s}_n^{jk}) p'_\eta(t_n^{ji}) \mathbf{s}_n^{ji} [\sigma_k]_{l'}^{m'}$$

AUTHOR INFORMATION

Corresponding Author

*E-mail: filippo.lipparini@courriel.upmc.fr.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

This work was supported in part by the France-Berkeley Fund, the ANR Manif, and the French state funds managed by CALSIMLAB and the ANR within the Investissements d'Avenir programme under reference ANR-11-IDEX-0004-02.

REFERENCES

(1) Tomasi, J.; Persico, M. *Chem. Rev.* **1994**, *94*, 2027−2094.
(2) Honig, B.; Nicholls, A. *Science* **1995**, *268*, 1144−9.
(3) Roux, B.; Simonson, T. *Biophys. Chem.* **1999**, *78*, 1−20.
(4) Cramer, C. J.; Truhlar, D. G. *Chem. Rev.* **1999**, *99*, 2161−2200.
(5) Orozco, M.; Luque, F. *Chem. Rev.* **2000**, *100*, 4187−4225.
(6) Tomasi, J.; Mennucci, B.; Cammi, R. *Chem. Rev.* **2005**, *105*, 2999−3093.
(7) Klamt, A. *WIREs Comput. Mol. Sci.* **2011**, *1*, 699−709.
(8) Marenich, A. V.; Cramer, C. J.; Truhlar, D. G. *J. Phys. Chem. B* **2009**, *113*, 6378−6396.
(9) Klamt, A.; Schuurmann, G. *J. Chem. Soc., Perkin Trans. 2* **1993**, 799−805.
(10) Chipman, D. *J. Chem. Phys.* **1999**, *110*, 8012−8018.
(11) Chipman, D. M. *J. Chem. Phys.* **2006**, *124*, 224111.
(12) Miertus, S.; Scrocco, E.; Tomasi, J. *Chem. Phys.* **1981**, *55*, 117−129.
(13) Cancès, E.; Mennucci, B.; Tomasi, J. *J. Chem. Phys.* **1997**, *107*, 3032−3041.
(14) Cancès, E.; Mennucci, B. *J. Math. Chem.* **1998**, *23*, 309−326.
(15) Mennucci, B.; Cancès, E.; Tomasi, J. *J. Phys. Chem. B* **1997**, *101*, 10506−10517.
(16) Barone, V.; Cossi, M. *J. Phys. Chem. A* **1998**, *102*, 1995−2001.
(17) Cossi, M.; Rega, N.; Scalmani, G.; Barone, V. *J. Comput. Chem.* **2003**, *24*, 669−681.
(18) Lipparini, F.; Scalmani, G.; Mennucci, B.; Cancès, E.; Caricato, M.; Frisch, M. J. *J. Chem. Phys.* **2010**, *133*, 014106.
(19) Lipparini, F.; Scalmani, G.; Mennucci, B.; Frisch, M. J. *J. Chem. Theory Comput.* **2011**, *7*, 610−617.
(20) Foresman, J.; Keith, T.; Wiberg, K.; Snoonian, J.; Frisch, M. *J. Phys. Chem.* **1996**, *100*, 16098−16104.
(21) Scalmani, G.; Frisch, M. J. *J. Chem. Phys.* **2010**, *132*, 114110.
(22) York, D.; Karplus, M. *J. Phys. Chem. A* **1999**, *103*, 11060−11079.
(23) Lange, A. W.; Herbert, J. M. *J. Phys. Chem. Lett.* **2010**, *1*, 556−561.
(24) Lange, A. W.; Herbert, J. M. *J. Chem. Phys.* **2010**, *133*, 244111.
(25) Mennucci, B. *J. Phys. Chem. Lett.* **2010**, *1*, 1666−1674 and references therein..
(26) Mennucci, B. *WIREs Comput. Mol. Sci.* **2012**, *2*, 386−404.
(27) *Continuum Solvation Models in Chemical Physics*; Mennucci, B., Cammi, R., Eds.; Wiley: New York, 2007.
(28) Quarteroni, A.; Valli, A. *Domain Decomposition Methods for Partial Differential Equations*; Oxford Science Publications: Oxford, 1999.
(29) Scalmani, G.; Barone, V.; Kudin, K.; Pomelli, C.; Scuseria, G.; Frisch, M. *Theor. Chem. Acc.* **2004**, *111*, 90−100.
(30) Cammi, R.; Tomasi, J. *J. Comput. Chem.* **1995**, *16*, 1449−1458.
(31) Greengard, L.; Rokhlin, V. *J. Comput. Phys.* **1987**, *73*, 325−348.
(32) Stamm, B.; Cancès, E.; Maday, Y. Domain decomposition for implicit solvation models. Submitted for publication.
(33) Steindal, A. H.; Ruud, K.; Frediani, L.; Aidas, K.; Kongsted, J. *J. Phys. Chem. B* **2011**, *115*, 3027−3037.
(34) Lipparini, F.; Barone, V. *J. Chem. Theory Comput.* **2011**, *7*, 3711−3724.
(35) Caprasecca, S.; Curutchet, C.; Mennucci, B. *J. Chem. Theory Comput.* **2012**, *8*, 4462−4473.
(36) Boulanger, E.; Thiel, W. *J. Chem. Theory Comput.* **2012**, *8*, 4527−4538.
(37) Lipparini, F.; Cappelli, C.; Barone, V. *J. Chem. Theory Comput.* **2012**, *8*, 4153−4165.
(38) Cancès, E. In *Continuum Solvation Models in Chemical Physics*; Mennucci, B., Cammi, R., Eds.; Wiley: New York, 2007; Chapter 1.2, pp 29−48.
(39) Cheng, H.; Greengard, L.; Rokhlin, V. *J. Comput. Phys.* **1999**, *155*, 468−498.
(40) The software package FMM3DLIB is publicly available at http://www.cims.nyu.edu/cmcl/fmm3dlib/fmm3dlib.html (accessed April 13, 2012).
(41) Pulay, P. *Chem. Phys. Lett.* **1980**, *73*, 393−398.
(42) Cossi, M.; Scalmani, G.; Rega, N.; Barone, V. *J. Chem. Phys.* **2002**, *117*, 43−54.
(43) The structure can be downloaded at http://www.rcsb.org/pdb/explore/explore.do?structureId=3S48.