Parametric representation of molecular surfaces

M. Bugeanu and H. Harbrecht

Parametric representation of molecular surfaces

Monica Bugeanu*, Helmut Harbrecht*

February 17, 2018

Abstract

This article is dedicated to the computation of a parametric representation of solvent excluded surfaces and isosurfaces by smooth four-sided patches. Such surface representations allow for the isoparametric discretization of the boundary integral equations which arise from solvation continuum models. Thus, higher-order ansatz functions can be applied in the boundary element method that is commonly used to solve the underlying equations, yielding a more accurate approximation of the sought apparent surface charge. Numerical results are reported to illustrate the approach.

---

*Department of Mathematics and Computer Science, University of Basel, Spiegelgasse 1, 4051 Basel, Switzerland

INTRODUCTION

Many reactions in chemistry take place in complex realistic environments. The overall system is then far too large to allow for a full quantum chemical treatment, due to the large number of particles involved. Continuum models include the environment as a continuum, disregarding its atomistic structure. They make thus quantum chemical simulations feasible and require only a handful of parameters to define the molecular cavity and the properties of the continuum[21,25,26].

The interface between the solute and the continuum constitutes the molecule's surface. In general, the molecule is defined as a union of balls, centered in the nuclei positions and of radii related to the van der Waals radius. The resulting surface is called the van der Waals surface (VWS). Whereas, the solvent accessible surface (SAS) coincides with the VWS surface except for scaling of the radii. In this article, we will focus on solvent excluding surfaces (SES), also known as Connolly surfaces[4]. They are defined from the VWS model by "rolling" a spherical probe over the surface.

A considerable amount of work has to go into the description and discretization of the cavity, since –without a good discretization– no good approximation of the desired continuum model can be expected. Famous algorithms include the GEPOL algorithm[15,16,23] with the extensions for geometry optimization, continuous surface charge and switching Gaussians[11,12,22], the isosurface approach implemented in Gaussian[6], the algorithm found in DefPol[17,18], the FIXPVA approach from GAMESS[24], or the analytic description of the surface[9,19,20].

In this article, we aim at the construction of a parametric representation of the cavity's surface Γ by four-sided patches. This means that we intend to subdivide the surface into smooth four-sided *patches*

$$\Gamma = \bigcup_{i=1}^{M} \Gamma_i,$$

where the intersection $\Gamma_i \cap \Gamma_{i'}$ of two patches consists at most of a common vertex or a common edge if $i \neq i'$. For each patch, a smooth diffeomorphism is constructed, mapping from the reference domain $\square := [0,1]^2$ to the respective patch:

$$\gamma_i : \square \to \Gamma_i \quad \text{with} \quad \Gamma_i = \gamma_i(\square) \quad \text{for } i = 1, 2, \ldots, M.$$

The diffeomorphisms of neighbouring patches are required to coincide on the common edge except for orientation. This ensures that regular subdivision of the reference domain leads to a regular mesh on the surface, as illustrated in Figure 1 in case of ethyl alcohol.
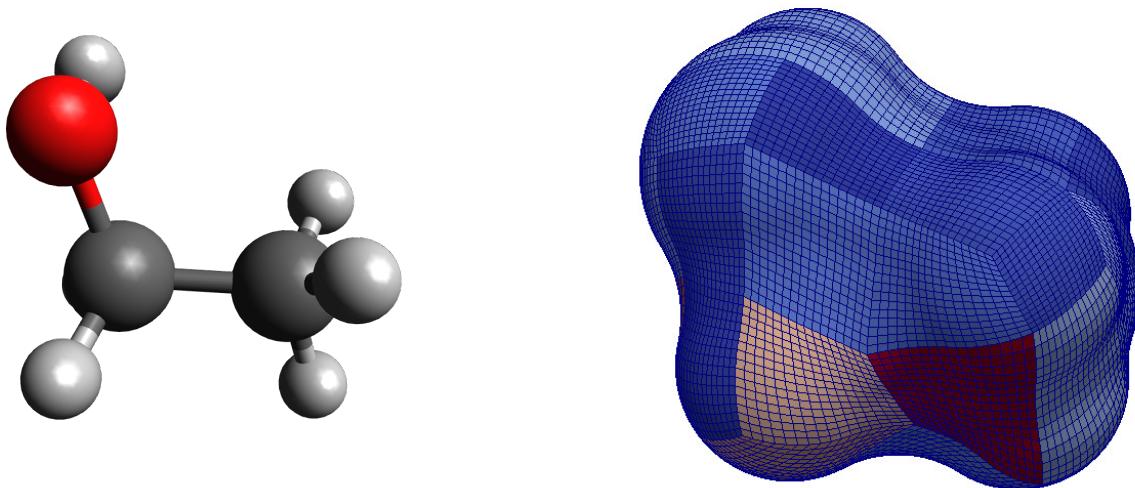

Figure 1: Ball and stick description of ethyl alcohol (left) and associated parametrization of the solvent excluded surface by 40 patches (right).

The main advantage of a parametric representation of the cavity's surface is that an exact representation is offered –no approximation by e.g. flat panels is performed– thus allowing for higher-order discretizations of the underlying equations[2,5,27]. In this article, we present a respective algorithm, which constructs the desired surface representation in case of SES and isosurfaces. As an intermediate step, a regular triangulation is derived, which can immediately be used in standard boundary element codes, such as the fast multipole method[7], the panel clustering[8], or the adaptive cross approximation[1]. Numerical results for both, SES and isosurfaces, are presented. Convergence studies for the computation of the apparent surface charge in the polarizable continuum model[14] by the wavelet PCM solver[2,27] are also given.

GENERATING THE CAVITY

The construction of the parametrization for the cavity starts with its mathematical description by means of a level-set function. It is constructed first a fine triangulation with an even number of triangles which resolve the geometry. This triangulation is coarsened step-by-step to get a coarse triangulation of the cavity, consisting of an even number of triangles. A graph based algorithm is then employed to find the best way to glue always two triangles together to form a quadrangular patch. Finally, the parametric representation is constructed by using a subdivision scheme.

Level-set function representation

The geometry can be defined by the position of the atoms of the molecule and the characteristic radius of the solvent. This results in a decomposition of the molecular surface into three types of partial surfaces: the atomic surface, stemming from the atoms of the molecule, the toroidal surface, appearing when two atoms interact with the radius of the solvent, and the spherical patch, a concave surface coming from the interaction of more than two atoms with the radius of the solvent. Separate level-set functions for each of these surfaces are constructed and used to determine whether or not a point is inside the molecular cavity.
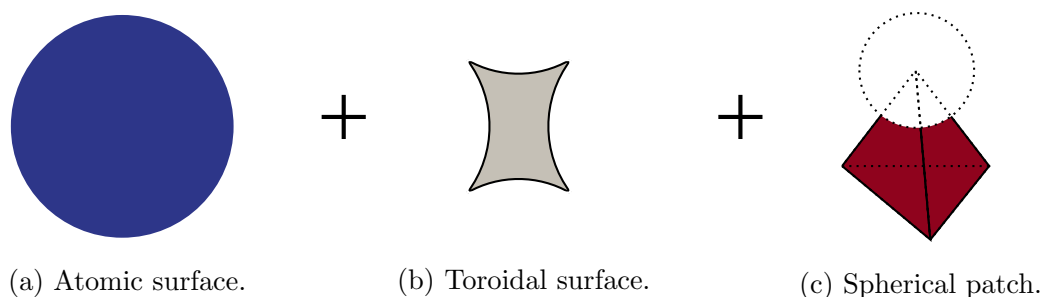
(a) Atomic surface.   (b) Toroidal surface.   (c) Spherical patch.

Figure 2: Surface decomposition.

As used in the IPCM version implemented in Gaussian[6], the charge distribution in space can also be used to define the surface of the molecule. The isosurface is a set of points that have the same value, which in this case means the same charge value. The surface that will eventually be used as a description of the molecular cavity is then given by one isosurface.

The input is given in form of a cube file, where the charge values for voxels in a bounding box containing the molecule are defined. In order to check if a point is inside the molecule, a cubic Newton interpolation is used, placing the requested point in the middle voxel of the interpolant. If the charge value is larger than the value for the desired isosurface, the point is inside the molecule, otherwise it is located outside the molecular cavity.

Initial triangulation

Given a level-set function, an initial triangulation can be computed by using the marching cubes algorithm[13]. In short, this algorithm discretizes the bounding box of the molecule by a cubic mesh and checks where the surface of the molecule traverses the resulting voxels by checking whether or not there are both corner points of the cubes inside and outside of the molecule. In order to resolve the geometry accurately, the cube size for the marching cubes algorithm has to be small enough. The initial mesh is an unstructured grid with potentially very small elements or very small edges, as seen in Figure 3, whilst the sought after mesh is a triangular mesh that can be used to generate quadrangular patches.
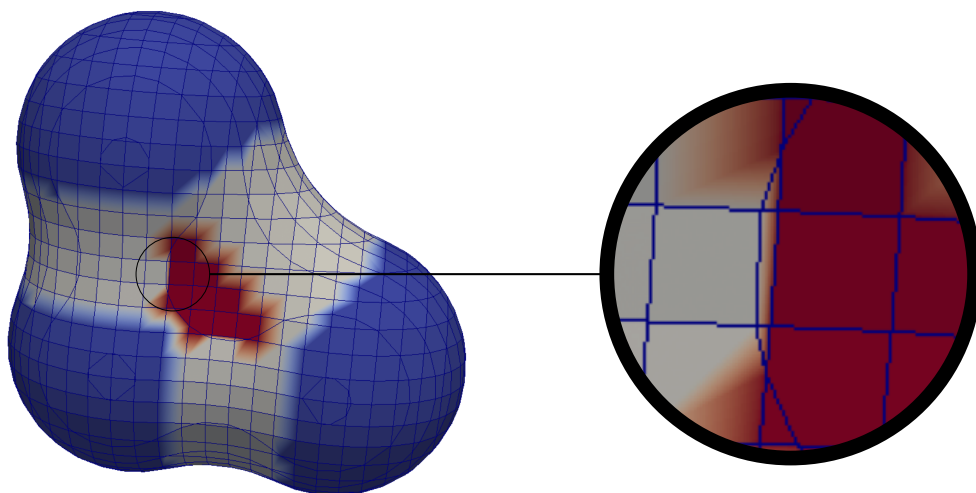

Figure 3: Unstructured grid resulting from the marching cubes algorithm.

Each cell of the unstructured grid is first subdivided into triangles by recursive insertion of edges. The subdivision can be done without hesitation, since all the cells are convex.

In our case, the shortest edge is always chosen. This results in an initial triangular mesh, with possibly ugly triangles and not necessarily an even amount of triangular cells. To each triangle Δ, a fitness measure can be associated

$$\text{fit}(\Delta) = \frac{4\sqrt{3}|\Delta|}{\sum_i |e_i|^2}, \tag{1}$$

where $|e_i|$ denotes the length of the edge e_i in Δ and $|\Delta|$ is the area of the triangle. This measure takes values between 0 and 1, with a maximum for the equilateral triangle. The further the triangle deviates from being equilateral, the smaller the fitness measure becomes.

The fitness measure can be used to give an overall assessment of the triangular mesh at hand. To this end, the smallest fitness value of the triangles a mesh consists of is associated as the fitness value of the mesh. The first improvement of the mesh replaces edges by points in order to achieve a better overall fitness. The replacement is done in ascending order of the replacement error and stops when the fitness of the mesh reaches a threshold or too few triangles are left. The algorithm that replaces the edge with a point has as input the mesh description and the index of the edge, $e = (\mathbf{q}_1, \mathbf{q}_2)$, to be replaced. The output of the replacement algorithm is the new point, $\mathbf{p}_{\text{new}}$, and the replacement error, $error(e)$. The point $\mathbf{p}_{\text{new}}$ is calculated as the least squares solution of the following problem:

$$\mathbf{p}_{\text{new}} = \arg\min_{\mathbf{q} \in \mathbb{R}^3} \|\mathbf{A}\mathbf{q} - \mathbf{b}\|_2. \tag{2}$$

Here, the matrix $\mathbf{A}$ is given by $\mathbf{n}_i$, the normals to the triangles containing either $\mathbf{q}_1$ or $\mathbf{q}_2$, and $\mathbf{b} = \langle \mathbf{n}_i, \mathbf{p}_i \rangle$, where $\mathbf{p}_i$ is a point in the triangle corresponding to the index i of the system matrix. In order to get a better understanding of the entries of the matrix $\mathbf{A}$ and the vector $\mathbf{b}$, let us consider the situation found in Figure 4. The matrix $\mathbf{A}$ and the vector $\mathbf{b}$ are here given by

$$\mathbf{A} = \begin{pmatrix} \mathbf{n}_1^T \\ \vdots \\ \mathbf{n}_4^T \\ \mathbf{n}_5^T \\ \vdots \\ \mathbf{n}_9^T \end{pmatrix} \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} \langle \mathbf{n}_1, \mathbf{q}_2 \rangle \\ \vdots \\ \langle \mathbf{n}_4, \mathbf{q}_2 \rangle \\ \langle \mathbf{n}_5, \mathbf{q}_1 \rangle \\ \vdots \\ \langle \mathbf{n}_9, \mathbf{q}_1 \rangle \end{pmatrix}.$$

6

The new point is calculated by a QR decomposition of the system matrix $\mathbf{A}$. The norm of the residuum of the solution is used to decide upon which edge to replace first. The error for the edge marked with magenta is thus given by

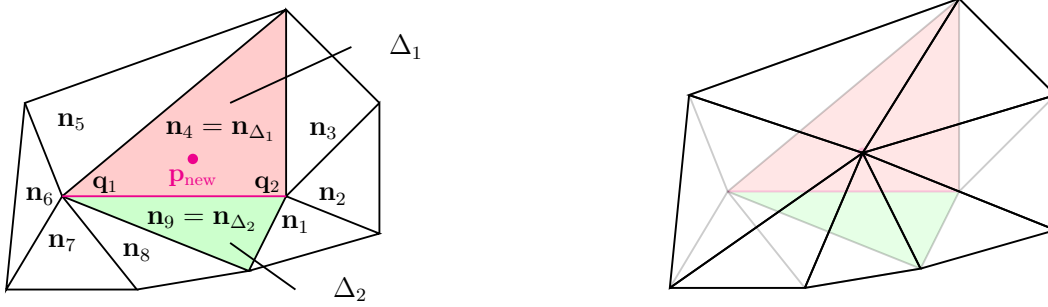$$\text{error}(e) = \|\mathbf{A}\mathbf{p}_{\text{new}} - \mathbf{b}\|_2.$$

Figure 4: Example of edge replacement.

Before an edge is replaced, the new point is projected to the surface in the direction of the normal and an admissibility test is performed in order to guarantee that the new resulting surface mesh does not have self-intersections.

After achieving a nice mesh, according to the fitness measure from Equation (1), the next step is to ensure an even number of triangles. This can easily be done by refining each triangle into four subtriangles and projecting the new points onto the surface. Since the mesh that is refined has a good fitness, the new even numbered mesh will also have a good fitness property.
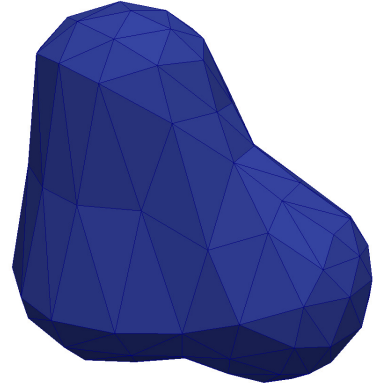
Quadrangular patches

The improved triangular mesh consists now of an even number of triangles which are coarsened until the number of triangles is double the desired number of patches, $n_\Delta = 2n_\square$. The desired number of patches can either be a predefined number given by the user or a number determined by the algorithm. In the second case, the even numbered triangular mesh with the best fitness with respect to Equation (1) is chosen out of a range of possible meshes.
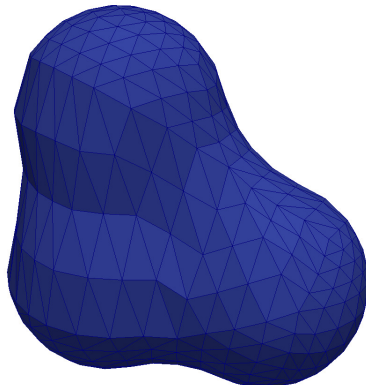
The next big step is to find out which two triangles should be glued together to form a quadrangular patch. To this end, the coarse triangles are refined again in order to estimate the curved surface better, as seen in Figure 5.
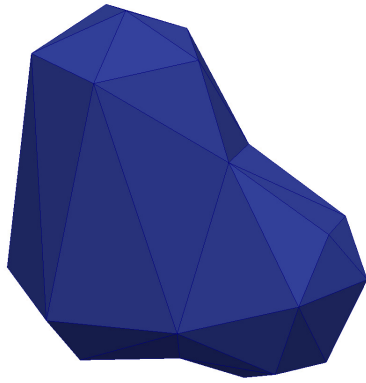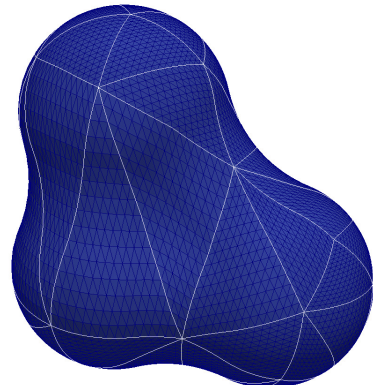

(a) Coarse mesh.

(b) Refinement on level one.

(c) Refinement on level two.

(d) Edges and fine mesh on level four.

Figure 5: Coarse mesh and refinement up to the required level for quadrangular patch calculation.

Each triangle can be combined with one of its three neighbours. The blossom algorithm[10] is used to find a minimum cost perfect matching in a weighted graph and can thus be applied to find the best possible combination of pairs of triangles. Therefore, a fitness measure is introduced for the combination of two triangles. Ideally, all angles of the resulting quadrangle
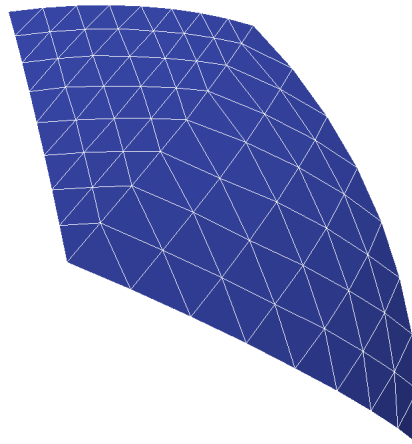
would be close to $\frac{\pi}{2}$. This is accomplished if the measure

$$\mathrm{fit}(\square) = \frac{1}{\min\{\sin^2\widehat{\alpha}, \sin^2\widehat{\beta}, \sin^2\widehat{\gamma}, \sin^2\widehat{\delta}\}}$$

is closest to 1, where the angles can be seen in Figure 6 and are measured on the fine mesh. If one of the angles is over π or too close to 0, a very high value is assigned to the fitness, such that the combination is rarely chosen.
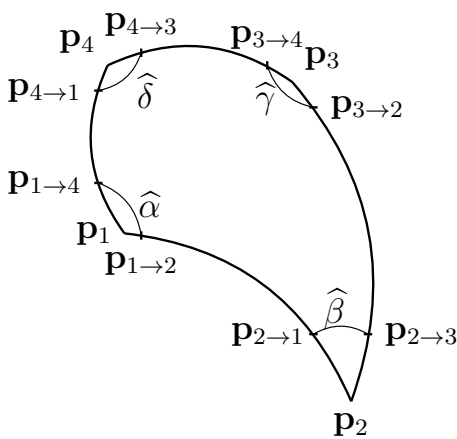

(a) Quadrangle angles used.

(b) Refined triangles.

Figure 6: Quadrangle approximation from two refined triangles.

Surface parametrization

The input parameter, n_L, defines the level of refinement of the quadrangular mesh. The output of the blossom algorithm already defines the corners of the quadrangular patches, and thus the refinement on level 0 of the mesh. The refinement on level n_L is then obtained by a subdivision scheme, recursively refining each element of the quadrangular patches. This is illustrated in Figure 7, where the normals and the elements are represented for each level of refinement.
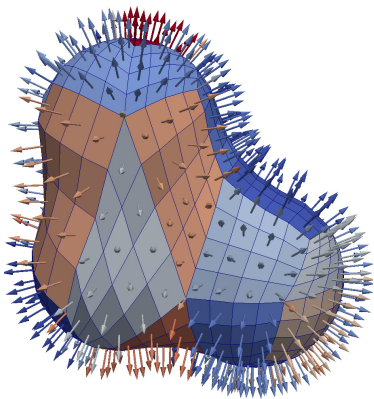
The refinement uses the mapping from the unit square, $\square$, to the bilinear element described by the corner points, $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$, and $\mathbf{p}_4$. The element is refined and the new points
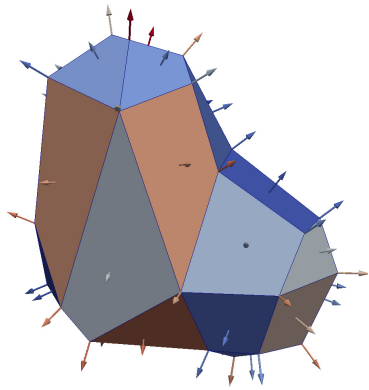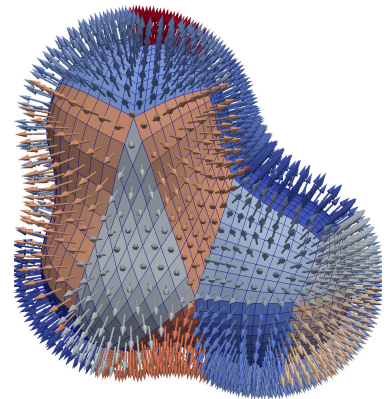
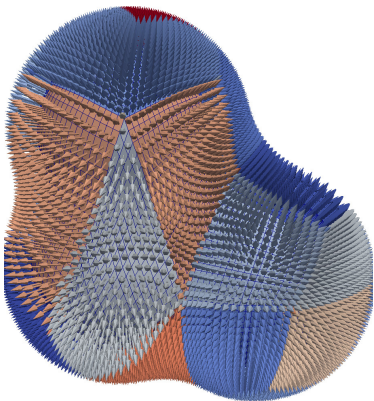(a) Coarse mesh, including element normals.
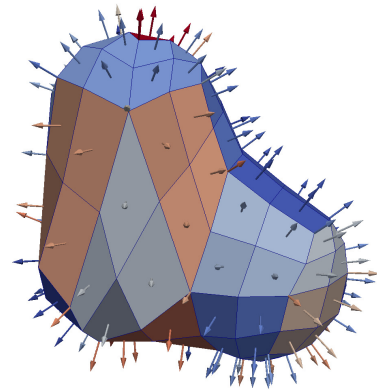
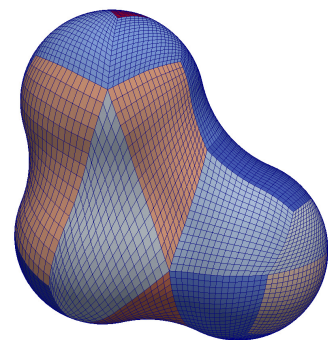(b) Refinement on level one, including element normals.

(c) Refinement on level two, including element normals.

(d) Refinement on level three, including element normals.

(e) Refinement on level four, including element normals.

(f) Refinement on level four.

Figure 7: Coarse patches and refinement up to required level.

are lifted onto the surface in normal direction. The bilinear interpolation is given by:

$$\mathbf{Q}_{\square_i}(s,t) = (1-s)\big((1-t)\mathbf{p}_1 + t\mathbf{p}_4\big) + s\big((1-t)\mathbf{p}_2 + t\mathbf{p}_3\big).$$

The surface point corresponding to $(s,t) \in \square$ is calculated by projecting the result of the bilinear interpolation, $\mathbf{Q}_{\square_i}(s,t)$, in the direction of the normal at this point, $\mathbf{n}_{\square_i}(s,t)$. The normal is calculated by taking the partial derivatives in s and t and applying the cross product

$$\mathbf{n}_{\square_i}(s,t) = \frac{\partial \mathbf{Q}_{\square_i}}{\partial s}(s,t) \times \frac{\partial \mathbf{Q}_{\square_i}}{\partial t}(s,t).$$

On the boundary between two elements, the arithmetic mean between the normals from both sides is taken. For one refinement step, the inner point found at $(0.5, 0.5)$ and the boundary points at $(0, 0.5)$, $(1, 0.5)$, $(0.5, 0)$, and $(0.5, 1)$ have to be computed for each element. All points on the element boundaries are calculated only once.

If the initial position of the nodes on the coarse quadrangular representation of the surface that comes out of the blossom algorithm does not lead to satisfying results, improvements can be applied. This is done by taking the optimal angle into account. For a given node, $\mathbf{p}_i$, of the coarse quadrangular mesh the optimal angles around that node depend on the degree of the node, $\alpha_{\mathrm{opt}} = \frac{2\pi}{\deg \mathbf{p}_i}$. The optimization is done by means of steepest descent, such that the angles around a node become optimal. The fitness associated to a point is given by

$$\mathrm{fit}(\mathbf{p}_i) = \frac{\sum_j (\langle \mathbf{p}^j_{i \to i-1} - \mathbf{p}_i, \mathbf{p}^j_{i \to i+1} - \mathbf{p}_i \rangle - \cos(\alpha_{\mathrm{opt}}))^2}{\deg \mathbf{p}_i}. \tag{3}$$

Here, each angle of the node, $\mathbf{p}_i$, taken inside the patch j is schematically seen in Figure 8 and thus approximated by $\angle(\mathbf{p}^j_{i \to i-1}, \mathbf{p}_i, \mathbf{p}^j_{i \to i+1})$ on the finest level.

Two special optimization cases are taken into account. The first one eliminates impossible combinations of patches, meaning quadrangular patches with two common edges. This is always combined with a convexity problem for the resulting patches and can only happen during the blossom algorithm if no better option is found. The solution in this case is to merge the two quadrangular patches into one. The second special case is looking at the lengths of the edges and puts a halt to the optimization in case an edge becomes too short. This can happen, since the steepest descent optimization only takes the angles into account. The steepest descent optimization is first done for all corner points of the patches
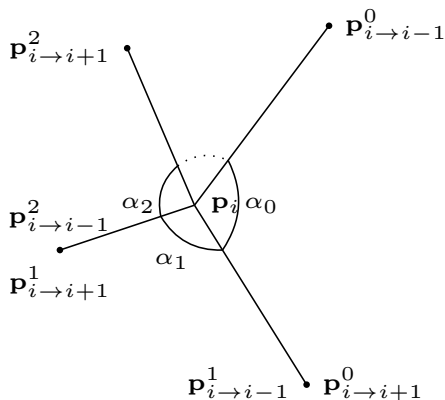
Figure 8: Description of angles around a patch vertex.

simultaneously until an optimality condition is reached and thereafter only for the corner point with the worst fitness. In both cases, the optimization also stops if the mesh becomes worse with respect to the fitness defined in Equation (3).

A control flow of the mesh generation is found in Figure 9 and shows the three main stages of the algorithm: (i) the initialization phase where the input is read into the program, (ii) the calculation of a nice coarse triangular mesh, and (iii) the combination of the triangular mesh into a quadrangular one ready to be used with the wavelet method.

VALIDATION

As already mentioned, the only information needed by the mesh generator is a characteristic function or a level-set function. The results of the description given by atoms and automatic detection of patches are found in Figure 1 for ethyl alcohol (on level 4, $n_\square = 40$, fit $= 0.43$).

The benzene molecule is used to prove the applicability of the isosurface approach. The input file was generated by Gaussian09, using Hartree-Fock and the 6-311++G** basis set. A point is inside the molecule if the value of the interpolated voxels evaluated at the given point is larger than the constant `isoSurfaceValue` $= 0.02$. A resulting mesh for the benzene example using $n_\square = 50$ patches can be found in Figure 10.
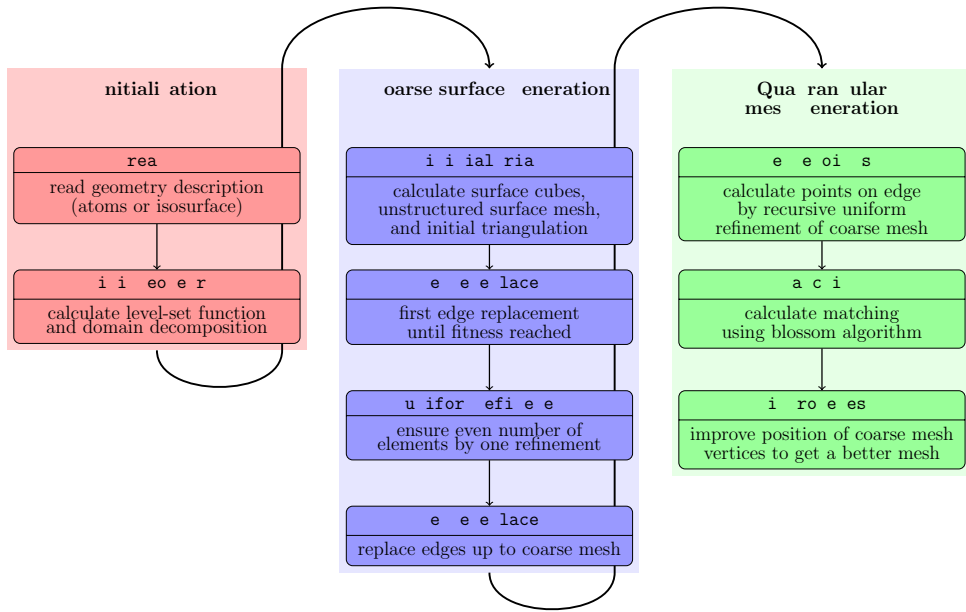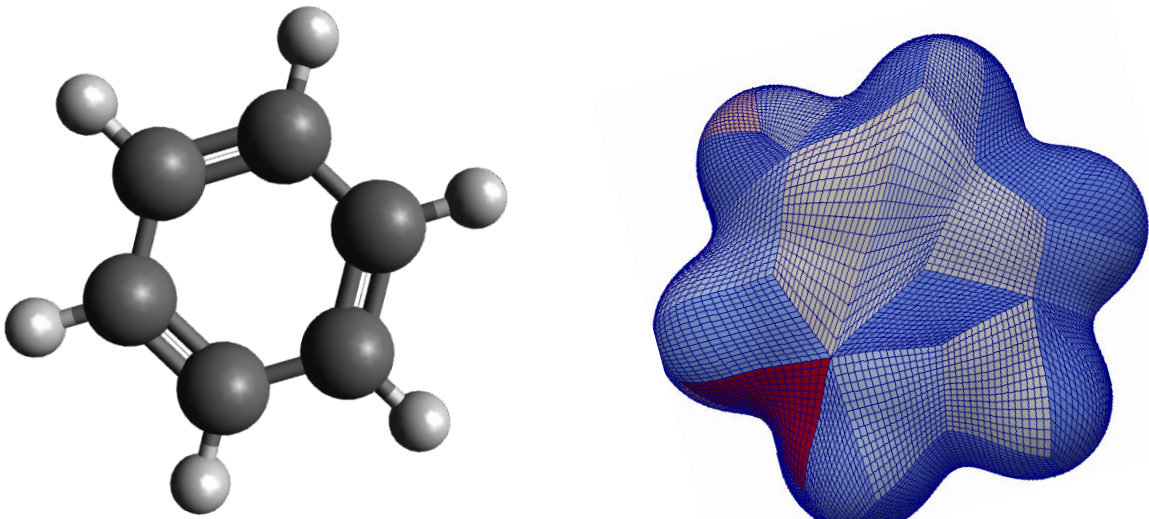
Figure 9: Control flow for the cavity generator.

Figure 10: Ball and stick description of benzene and the mesh generated with the isosurface description, $n_\square = 50$ patches on level 4, fit = 3.3.

Mesh generation time versus number of patches

The time needed to compute the surface is split into nine main components. The first (■) point in time recorded is after determining the cubes situated on the surface. The second (■) point in time marks the computation of the surface points by means of the bisection algorithm. This time period depends on the number of surface cubes, which in turn also factors in the constant dimension of the cubes. The time needed to triangulate the resulting unstructured mesh is negligible and is thus not taken into account. The third (■) point in time recorded is the time needed to compute the fine mesh by means of edge replacement using the QR decomposition. The fourth (■) point in time is closely related to the third and marks the computation of the coarse mesh. The fifth (■) point in time pinpoints the time needed for the refinement needed for the matching. The quadrangular patch initialization and refinement towards the boundary that will be used for the improvement strategy are stored in the sixth (■) point in time. The seventh (■) and the eighth (■) point in time mark the time needed for the global and the local improvement strategies, respectively. The last and ninth (■) point in time contains the projection onto the surface with a higher accuracy, the refinement up to the required level, n_L, and the output of the grid on all relevant levels in a format that can be read in by the boundary element code.

The time needed for the refinement and the output depends on the number of patches and the time needed to project the points onto the surface. This is a balancing act: if the number of patches is small, then the approximation of the surface is poorer and the time needed to project the refined patches onto the surface might take longer. On the other hand, the approximation of the surface is better if the number of patches is larger, but more points need to be projected onto the surface.

Figure 11 contains a representation of the time needed for the ethyl alcohol molecule given by the atom description. The pie chart on the left shows the time distribution for the automatic patch detection. The total time there is 116 sec. On the right the time for a fixed number of patches varies between 150 sec and 220 sec.

The conclusion that can be reached is that most of the time is spent in the edge replacement algorithm and the final refinement and output steps. While the output cannot

14

be optimized, a better, coarser starting mesh would save a lot of time currently spent in the edge replacement algorithm. The starting mesh, however, has to resolve all the fine structures of the geometry. This is not automatically guaranteed with a larger cube size for the marching cubes algorithm.
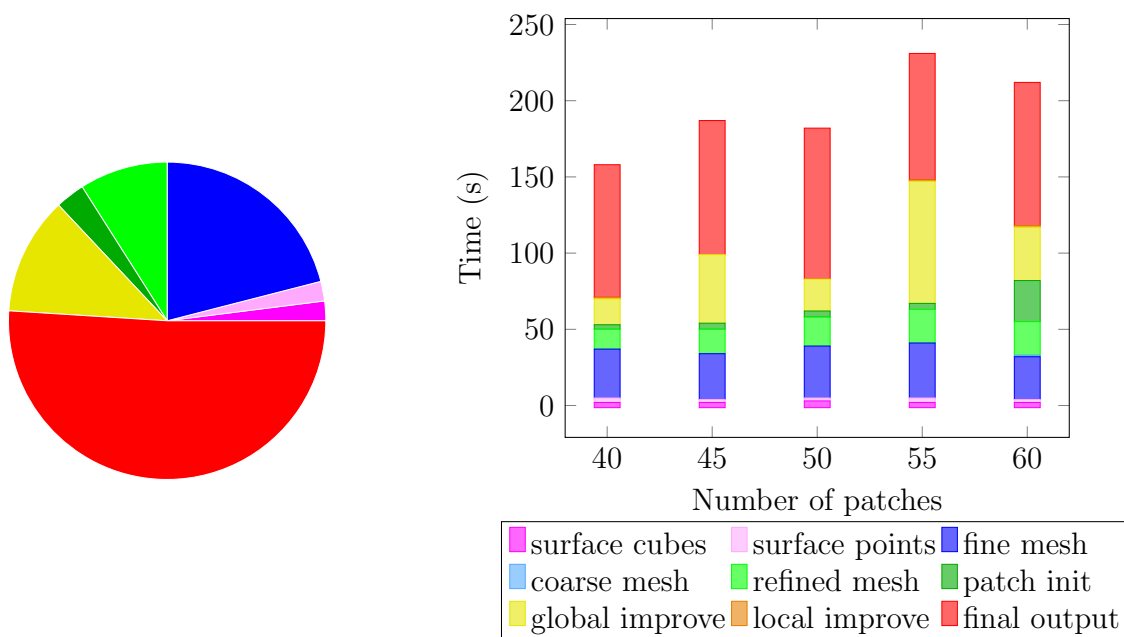

Figure 11: Time distribution for ethyl alcohol: On the left, the time for the automatic patch number detection can be seen, total time = 116 sec. On the right, the version for fixed number of patches is depicted.

The time needed for the implementation using isosurfaces is much shorter than for the description given by atoms. This can be seen in Figure 12 where the y-scale is in the order of seconds. The test if one point is inside or outside the molecule is done much faster, which also leads to a different distribution of the time. The final point in time containing the refinement and the output is no longer dominant, now most of the time is spent in the edge replacement algorithm, as seen in the fixed number of patches. For the automatic patch detection, more time is spent in the refinement step, due to the fact that we have a large number of patches, namely $n_\square = 243$.
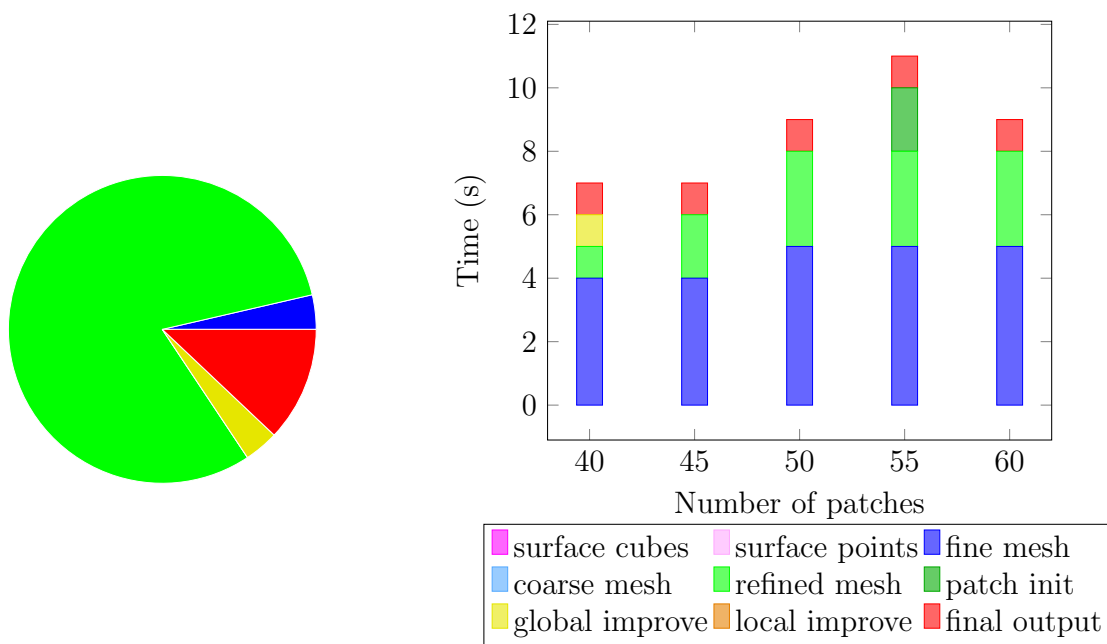
Figure 12: Time distribution or benzene described as isosurface: On the left, the time for the automatic patch number detection can be seen, total time = 39 sec. On the right, the version for fixed number of patches is shown.

Mesh quality versus number of patches

The next validation test shows the influence of the number of patches on the fitness of the resulting mesh for different molecules. The number of patches has been varied from 40 to 60. The quality of the mesh is taken as the worst value of the node fitness from Equation (3). The fitness values vary between 0.28 and 3.6 and do not directly correlate with the number of patches. This is due to the fact that the coarsening is governed by the error of the edge replacement and the resulting quadrangular patches are not taken into account. In Figure 13 however only fitness values between 0.2 and 0.6 are shown for the chosen test molecules found in the legend.
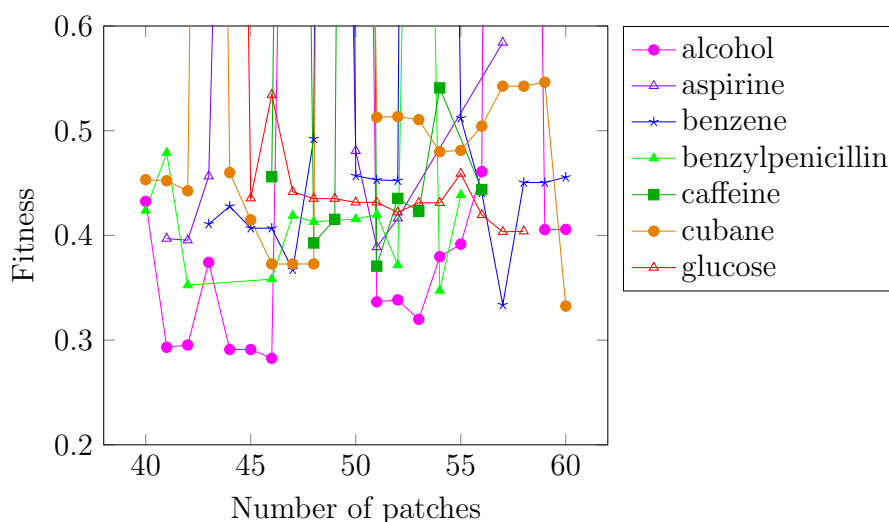
Figure 13: The worst node fitness versus the number of patches for a set of molecules calculated on level 4.

Scalability

A simple scalability test that can be done for the cavity generator is to take a look at the polyalkane chains C_nH_{2n+2}, $n = 8, 16, 32$. We look at the time needed to generate the mesh starting from the atom description and using an automatic patch number detection with the resulting number of patches $n_\square(C_8H_{18}) = 399$, $n_\square(C_{16}H_{34}) = 603$, and $n_\square(C_{32}H_{66}) = 1999$. The patches that have been generated, coloured corresponding to their fitness value, are

17

shown on the left side in Figure 14. On the right hand side, the pie chart of the time distribution for the automatic patch detection on refinement level 4 is shown. One can see that the time required for each step is similar for every molecule.

(a) C_8H_{18} patch representation with automatic patch detection, $n_\square = 399$, fit = 3.6.

(b) $C_{16}H_{34}$ patch representation with automatic patch detection, $n_\square = 603$, fit = 3.4.

(c) $C_{32}H_{66}$ patch representation with automatic patch detection, $n_\square = 1999$, fit = 1.56.

Figure 14: Surface description with automatic number of patch detection.

Convergence of PCM solver

We consider the polarizable continuum model (PCM)[14], solved by means of the integral equation formalism (IEFPCM)[3]. We will thus demonstrate the interplay between the parametric surface representation and our wavelet PCM solver[2,27]. The wavelet PCM solver was used to compute the apparent surface charge (ASC) by the boundary integral equation

$$\mathcal{S}\sigma = \frac{1}{\varepsilon - 1}\left(\frac{\varepsilon + 1}{2(\varepsilon - 1)} - \mathcal{D}\right)^{-1}\mathcal{N}_\rho - \mathcal{N}_\rho \quad \text{on } \Gamma, \tag{4}$$

where $\mathcal{S}$ and $\mathcal{D}$ are the single and double layer potential operators associated with the Laplace equation

$$\mathcal{S}f(\boldsymbol{x}) = \int_\Gamma \frac{f(\boldsymbol{y})}{\|\boldsymbol{x} - \boldsymbol{y}\|}\,\mathrm{d}s_{\boldsymbol{y}}, \quad \mathcal{D}f(\boldsymbol{x}) = \int_\Gamma \frac{\partial}{\partial \boldsymbol{n}_{\boldsymbol{y}}}\frac{f(\boldsymbol{y})}{\|\boldsymbol{x} - \boldsymbol{y}\|}\,\mathrm{d}s_{\boldsymbol{y}},$$

and $\mathcal{N}_\rho$ is the potential generated by the charge distribution ρ located inside the cavity.

For our test, the solvent's dielectric constant is $\varepsilon = 78.39$ and corresponds to water. The charge distribution ρ are point charges, located in the nuclei positions. In this situation, the interaction energy is known exactly, thus serving as an error measure. The ASC was computed on levels $\ell = 2, \ldots, 6$, yielding roughly $4^{\ell}M$ piecewise constant or bilinear ansatz functions in the wavelet PCM solver.

The results are shown in Figure 15 for piecewise constant ansatz functions and in Figure 16 for piecewise bilinear ansatz functions. The figures show the difference in the ASC compared to the expected theoretical value for different molecules. Both choices of ansatz functions converge for all molecules tested, starting with errors in the region of 10^{-1} and reaching up to 10^{-3} for the piecewise constant ansatz functions and reaching errors of around 10^{-5} for the piecewise bilinear ansatz functions. Hence, the piecewise bilinear ansatz functions perform better than the piecewise constant ansatz functions.
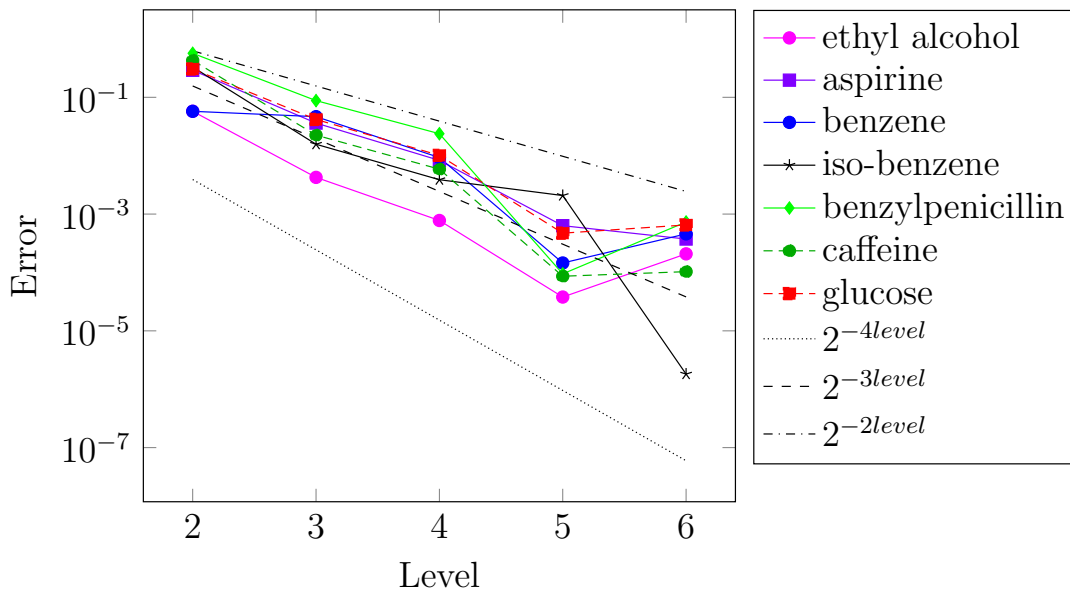

Figure 15: Convergence of the wavelet PCM solver using piecewise constant ansatz functions, calculated as a difference to the theoretical value.
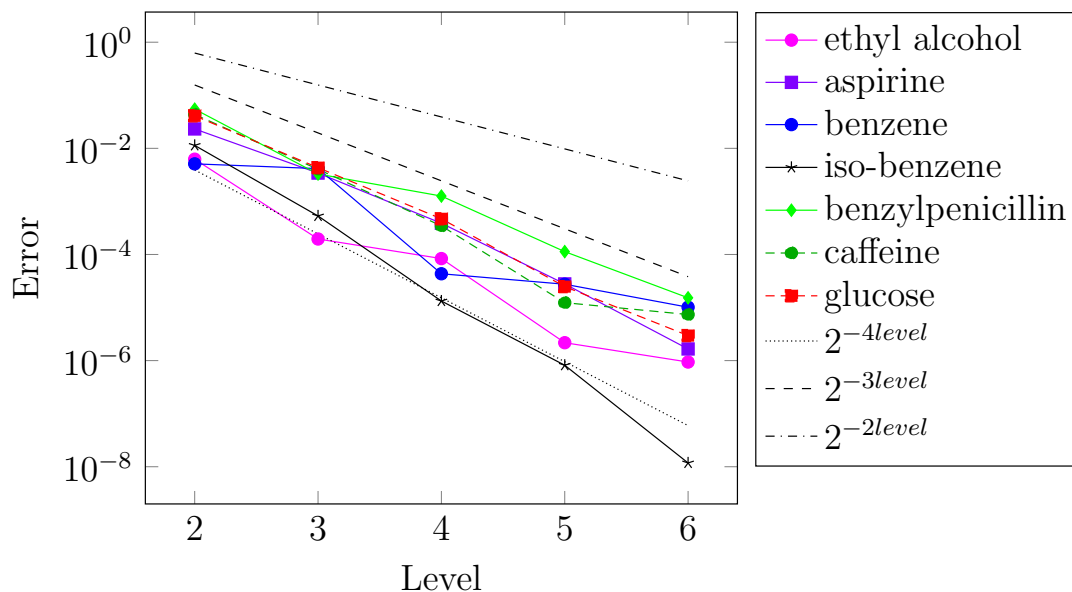
Figure 16: Convergence of the wavelet PCM solver using piecewise bilinear ansatz functions, calculated as a difference to the theoretical value.

CONCLUSION

Simulation of molecules in their environment can be performed by continuum models. The discretization of the underlying equations require in general also a suitable discretization of the molecule's surface. By starting with a level-set description of this surface, we have presented an algorithm that computes a parametric representation of the surface by smooth four-sided patches. Such surface representations allow for higher-order discretization techniques, especially in the integral equation formulation of the polarizable continuum model. Numerical results for solvent accessible surfaces and isosurfaces of several molecules have been given to present the approach.

References

1. Bebendorf, M., *Numer. Math.*, **2000**, 86, 565–589.

2. Bugeanu, M., Di Remigio, R., Mozgawa, K., Reine, S., Harbrecht, H., Frediani, L., *Phys. Chem. Chem. Phys.*, **2015**, 17, 31566–31581.

3. Cancès, E., Mennucci, B., *J. Math. Chem.*, **1998**, 23, 309–326.

4. Connolly, M. *J. Appl. Cryst.*, **1983**, 16, 548–558.

5. Dölz, J., Harbrecht, H., Peters, M.D., *Int. J. Numer. Meth. Eng.*, **2016**, 108(13), 1705–1728.

6. Foresman, J.B., Keith, T.A., Wiberg, K.B., Snoonian, J., Frisch, M.J., *J. Phys. Chem.*,

7. Greengard, L., Rokhlin, V., *J. Comput. Phys.*, **1987**, 73, 325–348. **1996**, 100, 16098–16104.

8. Hackbusch, W., Nowak, Z.P., *Numer. Math.*, **1989**, 54, 463–491.

9. Harbrecht, H., Randrianarivony, M., *Computing*, **2011**, 92, 335–364.

10. Kolmogorov, V., *Math. Prog. Comp.*, **2009**, 1, 43–67.

11. Lange, A.W., Herbert, J.M., *J. Phys. Chem. Lett.*, **2010**, 1, 556–561.

12. Lange, A.W., Herbert, J.M., *J. Chem. Phys.*, **2010**, 133, 244111.

13. Lorensen, W.E., Cline, H.E., *SIGGRAPH Comput. Graph.*, **1987**, 21, 163–169.

14. Miertŭs, S., Scrocco, E., Tomasi, J., *J. Chem. Phys.*, **1981**, 55(1), 117–129.

15. Pascual-Ahuir, J.L., Silla, E., *J. Comput. Chem.*, **1990**, 11, 1047–1060.

16. Pascual-Ahuir, J.L., Silla, E., and Tuñon, I., *J. Comput. Chem.*, **1994**, 15, 1127–1138.

17. Pomelli, C.S., Tomasi, J., *J. Comput. Chem.*, **1998**, 19, 1758–1776.

18. Pomelli, C.S., Tomasi, J., Cossi, M., Barone, V., *J. Comput. Chem.*, **1999**, 20, 1693–1701.

19. Quan, C., Stamm, B., *J. Comput. Phys.*, **2016**, 322, 760–782.

20. Quan, C., Stamm, B., *J. Mol. Graph. Model.*, **2017**, 71, 200–210.

21. Rivail, J.-L., Rinaldi, D., *Chem. Phys.*, **1976**, 18(1–2), 233–242.

22. Scalmani, G., Frisch, M.J., *J. Chem. Phys.*, **2010**, 132, 114110.

23. Silla, E., Tuñón, I., Pascual-Ahuir, J. L., *J. Comput. Chem.*, **1991**, 12, 1077–1088.

24. Su, P., Li, H., *J. Chem. Phys.*, **2009**, 130, 074109.

25. Tomasi, J., Mennucci, B., Cammi, R., *Chem. Rev.*, **2005**, 105, 2999–3094.

26. Warshel, A., Levitt, M., *J. Mol. Biol.*, **1976**, 103(2), 227–249.

27. Weijo, V., Randrianarivony, M., Harbrecht, H., Frediani, L., *J. Comput. Chem.*, **2010**, 31(7), 1469–1477.

LATEST PREPRINTS

No.	Author:	Title

No.   **Author:**   *Title*

2016-33   **S. Iula**
A note on the Moser-Trudinger inequality in Sobolev-Slobodeckij spaces in dimension one

2016-34   **C. Bürli, H. Harbrecht, P. Odermatt, S. Sayasone, N. Chitnis**
Mathematical analysis of the transmission dynamics of the liver fluke, Opisthorchis viverrini

2017-01   **J. Dölz and T. Gerig, M. Lüthi, H. Harbrecht and T. Vetter**
Efficient computation of low-rank Gaussian process models for surface and image registration

2017-02   **M. J. Grote, M. Mehlin, S. A. Sauter**
Convergence analysis of energy conserving explicit local time-stepping methods for the wave equation

2017-03   **Y. Bilu, F. Luca, D. Masser**
Collinear CM-points

2017-04   **P. Zaspel**
Ensemble Kalman filters for reliability estimation in perfusion inference

2017-05   **J. Dölz and H. Harbrecht**
Hierarchical Matrix Approximation for the Uncertainty Quantification of Potentials on Random Domains

2017-06   **P. Zaspel**
Analysis and parallelization strategies for Ruge-Stüben AMG on many-core processor

2017-07   **H. Harbrecht and M. Schmidlin**
Multilevel Methods for Uncertainty Quantification of Elliptic PDEs with Random Anisotropic Diffusion

2017-08   **M. Griebel and H. Harbrecht**
Singular value decomposition versus sparse grids: Refined complexity Estimates

2017-09   **J. Garcke and I. Kalmykov**
Efficient Higher Order Time Discretization Schemes for Hamilton-Jacobi-Bellman Equations Based on Diagonally Implicit Symplectic Runge-Kutta Methods

2017-10   **M. J. Grote and U. Nahum**
Adaptive Eigenspace Regularization For Inverse Scattering Problems

---

Preprints are available under *https://math.unibas.ch/research/publications*

LATEST PREPRINTS

No.	**Author:** *Title*
2017-11	**J. Dölz, H. Harbrecht, S. Kurz, S. Schöps and F. Wolf** *A Fast Isogeometric BEM for the Three Dimensional Laplace- and Helmholtz Problems*
2017-12	**P. Zaspel** *Algorithmic patterns for H-matrices on many-core processors*
2017-13	**R. Brügger, R. Croce and H. Harbrecht** *Solving a free boundary problem with non-constant coefficients*
2017-14	**M. Dambrine, H. Harbrecht and B. Puig** *Incorporating knowledge on the measurement noise in electrical impedance tomography*
2017-15	**C. Bürli, H. Harbrecht, P. Odermatt, S. Sayasone and N. Chitnis** *Analysis of Interventions against the Liver Fluke, Opisthorchis viverrini*
2017-16	**D. W. Masser** *Abcological anecdotes*
2017-17	**P. Corvaja, D. W. Masser and U. Zannier** *Torsion hypersurfaces on abelian schemes and Betti coordinates*
2017-18	**F. Caubet, M. Dambrine and H. Harbrecht** *A Newton method for the data completion problem and application to obstacle detection in Electrical Impedance Tomography*
2018-01	**H. Harbrecht and P. Zaspel** *On the algebraic construction of sparse multilevel approximations of elliptic tensor product problems*
2018-02	**F. Ghiraldin and X. Lamy** *Optimal Besov differentiability for entropy solutions of the eikonal equation*
2018-03	**H. Harbrecht and M. Schmidlin** *Multilevel quadrature for elliptic problems on random domains by the coupling of FEM and BEM*
2018-04	**M. Bugeanu and H. Harbrecht** *Parametric representation of molecular surfaces*

Preprints are available under *https://math.unibas.ch/research/publications*