



# AGBNP2 Implicit Solvent Library for Intel® MIC Architecture

Peicho Petkov<sup>a</sup>, Elena Lilkova<sup>a\*</sup>, Stoyan Markov<sup>a</sup>, Damyan Grancharov<sup>a</sup>, Nevena Ilieva<sup>a</sup>, Leandar Litov<sup>a</sup>

<sup>a</sup> National Centre for Supercomputing Applications, Akad. G. Bonchev Str., 25A, Sofia 1113, Bulgaria

---

## Abstract

A library, implementing the AGBNP2 [1,2] implicit solvation model, was developed, which calculates the nonbonded interactions of a solute with the solvent -- implicit water. The library is to be used in Molecular Dynamics (MD) packages for estimation of solvation free energies and studying of hydration effects.

The library was based on a previously developed Fortran code [3]. The presented in this paper code was rewritten in C and parallelized with OpenMP. The main objective was to parallelize the code in a way that allows it to run efficiently on Intel Xeon Phi in native mode. However, no efficient utilization of the MIC was observed, as the obtained performance on the coprocessor was an order of magnitude lower than on the CPU.

---

## 1. Motivation

Water is the most common solvent for most biological reactions and plays a vital role in determining the structures and dynamics, and hence the function, of globular proteins. Therefore it is of primary importance to account for the water-solute interactions in an MD simulation. In biomolecular modeling there are two alternatives for describing the solvent. Within the explicit solvent framework movements of individual water molecules are explicitly calculated. Due to the enormous number of solvent degrees of freedom this methodology is not particularly useful and suitable for molecular systems undergoing significant structural transitions, i.e. in protein folding, or for calculations of relative free energies of molecular conformations. The alternative approach is to replace the real water environment consisting of discrete molecules by a continuum with the dielectric and “hydrophobic” properties of water. This methodology allows enhanced sampling of conformational space due to lack of solvent viscosity and provides an effective way for free energy estimation by reducing the number of local “noise” minima, arising from the small variations in solvent configuration [4].

The implemented AGBNP2 (Analytical Generalized Born plus Nonpolar) model [2] uses a parameter-free and conformational-dependent algorithm to estimate the pairwise descreening scaling coefficients in the evaluation of Born radii. The same algorithm is also used to evaluate atomic surface areas. In addition a nonpolar estimator is introduced that does not depend exclusively on the solute surface area, but is based on the decomposition of the nonpolar hydration free energy into a cavity term, proportional to surface area, and an attractive dispersion energy term, which reproduces the continuum solvent solute-solvent Van der Waals interaction energy using a functional form based on the Born radius of each atom. The nonpolar model depends linearly on adjustable parameters that measure the effective surface tension and effective strength of solute-solvent Van der Waals interactions. The AGBNP2 model is applicable to a wide range of molecules and functional group topologies and types. The model is also suitable to study absolute hydration free energies as well as conformational equilibria.

## 2. AGBNP2

In the model the solute is described as a set of overlapping spheres of radius  $R_i$ , centered on the atomic positions  $\vec{r}_i$  and its volume is given by the Poincaré formula. The self-volume of atom  $i$  that measures the solute volume which belongs exclusively to that atom is:

$$V'_i = V_i - \frac{1}{2} \sum_j V_{ij} + \frac{1}{3} \sum_{i < j} V_{ijk} + \dots, \quad (1)$$

---

\* Corresponding author. *E-mail address*: elilkova@phys.uni-sofia.bg.

where  $V_i = \frac{3}{4}\pi R_i^3$  is the volume of atom  $i$ ,  $V_{ij}$  is the volume intersection of atoms  $i$  and  $j$  (second order intersection),  $V_{ijk}$  is the volume of intersection of atoms  $i$ ,  $j$ , and  $k$  (third order intersection), and so on. The total volume of the molecule is  $V = \sum_i V_i'$ . The overlap volume formed by  $n$  spheres is then approximated by the integral of the product of the  $n$  corresponding Gaussian functions:

$$V_{12\dots n}^g = p_{12\dots n} \exp(-K_{12\dots n}) \left(\frac{\pi}{\Delta_{12\dots n}}\right)^{\frac{3}{2}}, \quad (2)$$

where the coefficients  $p$ ,  $K$  and  $\Delta$  are defined as follows:

$$p_{12\dots n} = p^n, \quad p = \frac{4\pi}{3} \left(\frac{\kappa}{\pi}\right)^{\frac{3}{2}} \quad (3)$$

$$K_{12\dots n} = \frac{1}{\Delta_{12\dots n}} \sum_{i=1}^n \sum_{j=i+1}^n c_i c_j r_{ij}^2; \quad \Delta_{12\dots n} = \sum_{i=1}^n c_i; \quad c_i = \frac{\kappa}{R_i^2}; \quad \kappa = 2.227 \quad (4)$$

Only intersection volumes that are above a certain threshold are taken into account:

$$V_{12\dots n} = \begin{cases} 0, & V_{12\dots n}^g \leq v_1 \\ V_{12\dots n}^g f_w(u), & v_1 < V_{12\dots n}^g < v_2; \\ V_{12\dots n}^g, & V_{12\dots n}^g \geq v_2 \end{cases} \quad u = \frac{V_{12\dots n}^g - v_1}{v_2 - v_1}; \quad f_w(x) = x^3 (10 - 15x + 6x^2), \quad (5)$$

where  $v_1$  and  $v_2$  are predefined constants.

The atomic surface is the derivative of the volume with respect to the atomic radius  $R_i$ ,

$$A_i = f_a \left(\frac{\partial V}{\partial R_i}\right); \quad f_a(x) = \begin{cases} \frac{x^3}{a^2 + x^2}, & x > 0 \\ 0, & x \leq 0 \end{cases} \quad (6)$$

where  $f_a$  ensures positivity of  $A$ .

Once the geometric properties of the molecule are calculated, we can calculate the different contributions to the solvation free energy, given in the AGBNP2 model by

$$\Delta G_h = \Delta G_{gb} + \Delta G_{cav} + \Delta G_{vdw} + \Delta G_{hb}. \quad (7)$$

The non-polar term consist of a term, describing cavity formation energy  $\Delta G_{cav}$ , and a Van der Waals interaction energy term  $\Delta G_{vdw}$  with solvent molecules:

$$\Delta G_{cav} = \sum_i \gamma_i A_i, \quad (8)$$

where  $\gamma_i$  is the surface tension parameter assigned to atom  $i$  and  $A_i$  is its Van der Waals surface area;

$$\Delta G_{vdw} = \sum_i \frac{\alpha_i a_i}{(B_i + R_w)^3}; \quad \alpha_i \approx 1; \quad a_i = -\frac{16}{3} \pi \rho_w \epsilon_{iw} \sigma_{iw}^6, \quad (9)$$

where  $\alpha_i$  is an adjustable dimensionless parameter of the order of 1 and  $\rho_w$  is the number density of water at standard conditions,  $\sigma_{iw}$  and  $\epsilon_{iw}$  are the Lennard-Jones parameters for the interaction between atom  $i$  and an oxygen atom of the water model, and  $R_w = 1.4 \text{ \AA}$ .

The solute-solvent electrostatic interaction is given by:

$$\Delta G_{gb} = u_\epsilon \sum_i \frac{q_i^2}{B_i} + 2u_\epsilon \sum_{i < j} \frac{q_i q_j}{f_{ij}}. \quad (10)$$

The Born radius  $B_i$  of a solute atom is calculated using

$$B_i^{-1} = f_b(\beta_i) = \begin{cases} \sqrt{b^2 + \beta_i^2}, & \beta_i > 0 \\ b, & \beta_i \leq 0 \end{cases}; \quad b^{-1} = 50 \text{ \AA}. \quad (11)$$

Again, the function  $f_b(\beta)$  keeps the result finite. Its parameter  $\beta_i$  is calculated using the volume scaling  $s_{ij}$  and the pair descreening functions  $Q_{ij}$  (Appendix B in ref [1])

$$\beta_i = \frac{1}{R_i} - \frac{1}{4\pi} \sum_{i \neq j} s_{ij} Q_{ij}; \quad s_{ij} = s_j + \frac{v_j'}{v_j}; \quad s_j = \frac{v_j' - d_j A_j}{v_j}; \quad d_j = \frac{1}{3} R_j' \left[1 - \left(\frac{R_j}{R_j'}\right)^3\right] \quad (12)$$

$$v_{ji}' = v_{ij}' = \frac{1}{2} V_{ij} - \frac{1}{3} \sum_k V_{ijk} + \frac{1}{4} \sum_{k < l} V_{ijkl} - \dots, \quad (13)$$

where  $R'_j$  is the augmented radius  $R' = R + dR$ ,  $dR = 0.5 \text{ \AA}$ .

### 3. Library Development

The library was based on a previously developed Fortran code[3]. The library was rewritten in C and parallelized with OpenMP. The main objective was to parallelize the code efficiently to run on Intel Xeon Phi in native mode.

Two different structures are used to store data. The first is the atomic properties structure, where indices run over the atom number. The second is the so called multiplet structure, where indices depend on the thread ID and the doublet/triplet/quadruplet number.

The library needs as input the coordinates of the atoms, their atom type, partial charge and Lennard Jones parameters. As a first step the derived constants for the model ( $R_i, c_i, a_i, \alpha_i$ ) are calculated and stored in arrays of length equal to the number of atoms. Then neighbor lists are constructed for each atom, given a certain cutoff value. In our tests we used a cutoff of  $15 \text{ \AA}$ . In addition, two auxiliary neighbor lists are constructed – a short neighbor list within a cutoff, equal to twice the largest Lennard-Jones sigma plus  $dR$ ; and a complementary lists with the neighbors between the two cutoffs. All these computations are done in parallel over the atoms and the data is stored in an atomic properties structure.

After that, based on the short neighbor lists doublet lists are constructed. This task is performed in parallel over the atoms, whereas each thread creates its own list. For each doublet its properties ( $\Delta_{ij}, K_{ij}, V_{ij}^g, \vec{r}_{ij}^c$ ) are calculated. If the second-order cross section volume  $V_{ij}^g > 0$  holds, the data is stored in the doublet fields of the multiplets structure, otherwise it is stored in the zerodoublet fields.

Once the doublet and zerodoublet lists are created and their size is determined, the program passes on the triplet lists construction. This is performed again in parallel over the doublet number. The triplets are based on the doublets, whereas an auxiliary list is used with the cross section of the short neighbor lists of the two atoms, forming the doublet. The properties of the triplet ( $\Delta_{ijk}, K_{ijk}, V_{ijk}^g, \vec{r}_{ijk}^c$ ) are computed and stored in the triplet fields of the multiples structure, only if the third-order cross section volume  $V_{ijk}^g > 0$ .

Finally, based on the triplets, the quadruplet lists are constructed in a similar way and stored in the quadruplet fields of the multiplets structure.

Once all multiplets and their properties are computed, the self-volumes and the surfaces as well as the scaling parameters  $s_j$  and  $d_j$  of each atom are calculated and held in the atomic properties structure.

The  $V'_{ij}$  values are calculated using auxiliary lists of the triplets in which each doublet participates and of the quadruplets in which each triplet is part of. These lists are created during the doublet/triplet and quadruplet lists construction. These values are used for calculation of the atomic scaling factors  $s_{ij}$  wherever needed. In addition to the inverse Born radii  $\beta_i$ , several other atomic properties ( $Y_i, W_i, \vec{\gamma}_i, U_i$ , please refer to [1,2] for further details on these quantities and the various derivatives that are necessary for computation of the forces) are also calculated in parallel and stored in the atomic properties structure. The different force contributions are calculated using neighbor lists and the multiplets structure.

At the end, the free energy terms are computed. In our implementation we did not include the hydrogen bonding energy correction of the AGBNP2 model and its contribution to the forces. A cutoff scheme is used for calculation of the GB and the VdW energies, as well as  $Q$ -term of the VdW forces and the  $Q$  and  $Q,s$ -terms of the GB forces.

### 4. Optimizing Performance for Intel Xeon Phi

For optimized OpenMP parallelization private copies of the doublets, zerodoublets, triplets quadruplets and  $V'_{ij}$  lists are used for every thread without overlapping data. The atomic parameters and properties, stored in the atomic properties structure, are shared between the threads. The main goal of the so implemented parallel algorithm is to loop over private mutliplets or to use parallel FOR loops over shared data. At this stage we have not yet padded the data in the parallel FOR loops.

In accordance with Intel MIC programming guidelines [5] all data was aligned to 64B. Where possible, elemental functions were vectorized and loops were modified to enable vectorization. The results are obtained with the Intel compiler with O3 optimization level and using of Intel's *fast memset/memcpy libirc* library.

The library was tested on different sized cases, varying from 360 to 17 700 atoms. It was found that the most time consuming functions are the one that create the neighbor lists, the calculation of the forces and the creation

of the doublet and triplet lists and the calculation of their properties. The execution times for the tests on different number of threads on the host and on the coprocessor are given in Table 1 and 2 respectively.

Table 1. Execution time in seconds on the Intel Xeon host

# Atoms	1 thread	2 threads	6 threads	8 threads	12 threads	16 threads
362	0.055	0.028	0.031	0.026	0.021	0.220
1155	0.260	0.138	0.065	0.064	0.057	0.057
10903	5.322	2.796	1.080	0.832	0.627	0.589
17661	10.720	5.234	1.963	1.510	1.166	1.078

Table 2. Execution time in seconds on the Intel Xeon Phi coprocessor

# Atoms	1 thread	2 threads	8 threads	16 threads	32 threads	64 threads	120 threads	180 threads	240 threads
362	0.637	0.340	0.176	0.108	0.154	0.192	0.310	0.555	0.671
1155	3.331	1.752	0.754	0.515	0.410	0.252	0.290	0.428	0.447
10903	70.999	35.742	11.132	6.804	5.238	3.757	3.385	3.317	3.229
17661	137.292	64.132	20.523	11.860	8.288	6.803	5.979	5.617	5.783

The implemented AGBNP2 implicit solvation model contains about 3000 lines of code, organized in 78 functions, and proved to be very difficult to vectorize properly. A major issue in this respect was that vectorization could not be enforced in loops with calls to elemental functions containing if statements. The library scales similarly on both the Intel Xeon and on the Intel Xeon Phi (Fig. 1). However, for all tested cases the host performs more than 10 times better than the coprocessor. The code scales linearly with the size of the simulated system.

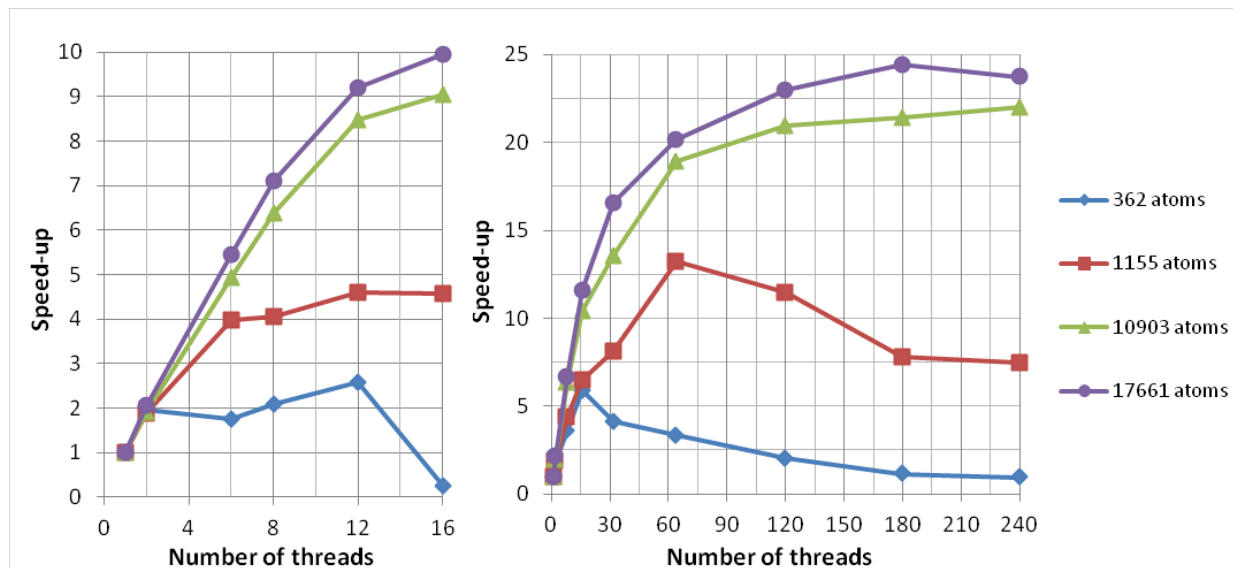


Figure 1: Speed-up of the library on Intel Xeon (left) and Intel Xeon Phi coprocessor (right) for systems with different size.

## 5. Conclusions

Based on a previously developed Fortran code, a library, implementing the AGBNP2 implicit solvation model, was written in C with OpenMP parallelization and ported on Intel Xeon Phi in native mode. The library proved to be quite complex, considering the specific of the coprocessor. The results clearly indicate both a performance and scalability potential, however the speed-up of the present code is rather poor. The available time of the project was not sufficient to properly address vectorization issues and other performance/scalability problems.

Future efforts will focus on these issues and also on plugging the library into popular MD packages like GROMACS. The current code of the library can be found at <http://phys.uni-sofia.bg/~peicho/agbnp2-mic-1.0.tgz>.

## References

- [1] E. Gallicchio and R.M. Levy, AGBNP: An Analytic Implicit Solvent Model, *J. Comput. Chem.* 25(4) 479-499 (2004).
- [2] E. Gallicchio, K. Paris and R.M. Levy, The AGBNP2 Implicit Solvation Model, *J. Chem. Theory Comput.* 5(9), 2544–2564 (2009).
- [3] P. Petkov, S. Markov, and I. Todorov, Development of AGBNP2 Implicit Solvent Model Library for MD Simulations, PRACE Whitepaper 111, <http://www.prace-ri.eu/IMG/pdf/wp111.pdf>
- [4] Alexey Onufriev, The generalized Born model: its foundation, applications, and Limitations, September 8, 2010, <http://people.cs.vt.edu/~onufriev/PUBLICATIONS/gbreview.pdf>
- [5] Jim Jeffers and James Reinders, Intel Xeon Phi Coprocessor High Performance Programming, 1st Edition, 2013, Morgan Kaufmann.

## Acknowledgements

This work was financially supported by the PRACE project funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-261557. The project was realized using the PRACE Research Infrastructure resources at Cineca, Italy.