Parallel Solution of the Poisson-Boltzmann Equation for Proteins

Shura Hayryan[1], Chin-Kun Hu[1], Edik Hayryan[2], Imrikh Pokorny[3]

[1] Institute of Physics, Academia Sinica, Taipei 11529, Taiwan
[2] Laboratory of Computational Techniques and Automation, J.I.N.R., Dubna 141980, Russia
[3] Technical University in Košice, Slovakia

Abstract. A fortran code PBSOLVE is created for numerical solution of a linear Poisson - Boltzman equation. Finite difference discretization and successive overrelaxation (SOR) iterations on the sequence of grids are used to obtain the approximate electrostatic potential on the grid. Error control is provided by comparison of solutions on nested grids. A parallel version of the PBSOLVE which exploits the message passing interface (MPI) has also been constructed. The performance of the program and the efficiency of parallelization have been tested on the small peptide Met-Enkephalin.

1 Introduction

The biologically active 3D structure of the protein results from several types of complicated interactions which occur between the atoms comprising the protein molecule as well as between the protein and the surrounding solvent. These include a solvent hydrophobic effect, Van der Waals forces, electrostatic interactions, hydrogen bonding, entropy contributions due to the chain flexibility, etc. The importance of the interactions between the solute protein and the solvent is emphasized by the fact that the main force, driving the protein molecule to the folded state is a hydrophobic force [1]. The contribution to the free energy of folding from the intramolecular interactions is relatively easy to calculate by means of molecular dynamic simulations, Monte Carlo procedures or using a molecular force-field calculations. Much more severe is a problem of calculation of the solvent contribution to the free energy because of the tremendous number of degrees of freedom.

There are two basic approaches to calculate the free energy of the solvation of proteins. In the *molecular* solvent models, hundreds of thousands of water molecules and salt ions are introduced explicitly into simulation. This approach is, of course, the most realistic and precise from the standpoint of physics. The difficult point here is that the simulation processes often fail to converge to reasonable results in feasible time period. In the so called *continuum* solvent models, the solvent is treated as a continuous medium, the average characteristics of which are close to those of the real solvent. Though cumbersome by themselves,

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2657, pp. 54–62, 2003.

the simulations with continuum solvent models require several orders of magnitude less computational time than the simulations with molecular models. In present paper we will discuss only the continuum models of the solvent.

Since water is a highly polar liquid, the electrostatic interactions play a central role in the behavior of protein-solvent system [2]. Yet an accurate calculation of these interactions is the most complicated problem. While in the calculations *in vacuo* the simple Coulomb potential may be used successfully, the calculation of the electrostatic interactions between water and solute protein remains a very difficult task. One of the methods currently used is based on the classical elecrodynamics, where the energy of the electrostatic field of charged systems in the water environment is calculated by solving the Poisson-Boltzmann equation.

The purpose of the present work is to develop an effective computer code for solving numerically the Poisson-Boltzmann equation for protein molecule placed in the water.

2 Theoretical Background

2.1 Protein-Solvent Interaction Energy

Perhaps the simplest way to calculate the electrostatic interactions of protein with solvent (Fig.1) is to assume a distance dependent electrostatic constant and calculate the electrostatic energy by the formula [3].

$$\varepsilon(r) = D - \frac{D-2}{2}[(sr)^2 + 2sr + 2]e^{-sr}. \tag{1}$$

Here empirical values for the parameters D and s are being used such that on the large distances ε takes the value of that of bulk water (≈ 80) and the value $= 2$ for the short distances (protein interior space). However, this approach is oversimplified. One of the obvious drawbacks is that the atoms which are close to each other in the space may not occur in the protein interior simultaneously, and reversely, the atoms which are far from each other in space may occur both in the protein interior.

Generally speaking the protein-water interactions, may be treated on the macroscopical level by exploiting two kinds of interactions: electrostatic interactions for polar components and surface tension for non-polar components [2]. So, the total free energy of protein-water system may be written as a sum of two terms

$$G_{tot} = G_{np} + G_p. \tag{2}$$

Here G_{np} is the non-polar part of the free energy of solvation; it actually includes two parts: the free energy of creating a cavity of the shape and size of the molecule in the bulk solvent, and the energy of Van der Waals interactions between solute molecule and the solvent. G_p is the part of free energy which corresponds to the mono- and multipole electrostatic interactions between charged atoms of the protein and the surrounding polar solvent molecules or solved salt ions.

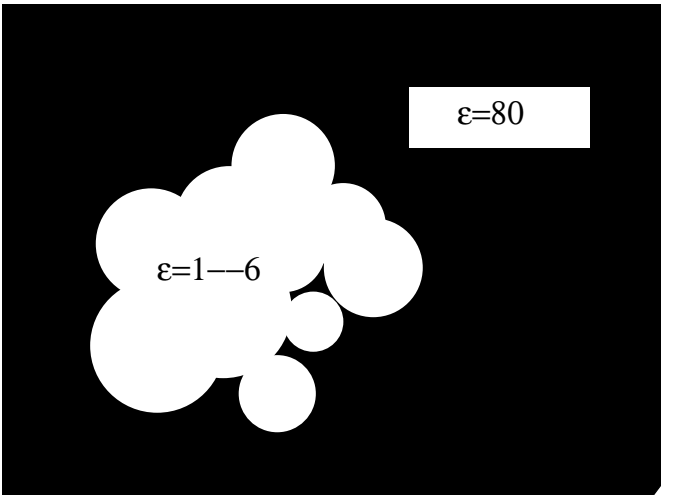

Fig. 1. The protein molecule in the solvent.

The non-polar interactions are usually calculated under the assumption that the free energy of solvation of the uncharged hydrocarbons are proportional to the surface accessible area of the atoms [4].

$$G_{np} = \sum\nolimits_i \sigma_i S_i. \tag{3}$$

Here the sum is over all atoms of the protein, S_i is the solvent accessible area [5] of the $i-th$ atom, σ_i is some coefficient of proportionality determined by experiments on solvation of small hydrocarbons (octanol is the most typical solute). The values of σ are different for each type of atoms and depend strongly on the forcefield within which they have been calculated. The quantities S_i are conformational dependent and their calculation requires a complicated geometrical algorithm for evaluation of the solvent-accessible surface of the protein molecule [5]. Two types of algorithms are now exploited for this purpose: an analytical algorithm by Connolly [6] or some numerical algorithm, such as double cubic method [7].

The second term in Eq.(2) requires exact calculation of the electrostatic polarization energy of solved molecules and their dielectric environment. This is usually done by solving the Poisson-Boltzmann equation for the charged particles placed in the dielectric medium [8].

2.2 The Poisson-Boltzmann Equation

Exact classical approach for the electrostatic interactions in solution is based on the solution of Poisson-Boltzmann equation [8].

$$\nabla \cdot [\varepsilon(r)\nabla \cdot \phi(r)] - \varepsilon(r)h(r)^2 sinh[\phi(r)] + 4\pi\rho(r)/kT = 0. \tag{4}$$

Here $\phi(r)$ is the dimensionless electrostatic potential in units of kT/q, with q being an electrical charge, T is the temperature, $\varepsilon(r)$ is the electrostatic constant, $\rho(r)$ is the fixed charge density, and $h(r)^2 = 1/\lambda^2 = 8\pi q^2 I/ekT$ with λ, I, k, and e being, respectively, the Debye damping length, the ionic strength of the solvent, the Boltzmann constant, and the elementary charge. Quantities ϕ, ϵ, h and ρ are all functions of the vector $\mathbf{r}$.

For small electrostatic fields Eq. (4) may be linearized

$$\nabla \cdot [\varepsilon(r)\nabla \cdot \phi(r)] - \varepsilon(r)h(r)^2\phi(r) = -4\pi\rho(r)/kT. \tag{5}$$

In the case of a single point charge placed in the medium with uniform dielectric constant $\epsilon(r) = \epsilon$, and equilibrium distribution of the ions, Eq.(5) has a simple spherically symmetrical solution

$$\phi(r) = q\frac{e^{-hr}}{r}, \tag{6}$$

which becomes the Coulomb's law $\phi(r) = q/r$, when there are no solved ions in the solvent ($h(r) = 0$).

However, due to its compact packing the protein molecule creates another medium (protein interior) with the electrostatic constant different from that of the bulk solvent (Fig.1), and thus the electrostatic constant of the whole system is not uniform anymore. In the protein interior space ε is assumed to be constant and having the value 1 to 6, according to different authors. In the bulk solvent medium the value of ε is taken to be close to the electrostatic constant of water (≈ 80). So the electrostatic constant is discontinuous along the *protein-water* boundary. In this case we have a boundary problem [9] and should use the numerical methods to solve the Poisson-Boltzmann equation. Usually finite-difference methods are used for this purpose.

Another problem about solving the PB equation is very complicated shape of the surface along which the electrostatic constant has discontinuity. This means that one can not set the boundary conditions analytically and it is necessary to assign a pointer to each mesh point of the difference scheme, which shows whether a given site belongs to the protein interior or to the bulk solvent. For this, again, a solvent accessible surface must be evaluated.

After the solution of the Poisson-Boltzmann equation is found one can calculate the energy of the electrostatic field of distributed charges by the formula

$$G_p = \int d^3r\rho(r)\phi(r). \tag{7}$$

Here the integration is over the whole space, or, in the case of the numerical solution, over the calculating domain, where the PB equation has been solved.

Obviously, this approach for calculating the solvation free energy is reasonable only in the cases when one does not have to calculate the protein energy too many times within one simulation. In simulations such as molecular dynamics or Monte Carlo, one has to sample many hundreds of thousands conformations,

and to solve numerically the non-linear equation (4) is not feasible. One of computationally cheaper approaches is the Born approximation method which is based on the Born treatment of the monoatomic spherical ions in the dielectric medium [10].

3 Results

3.1 The Calculating Domain

The solution of Poisson-Boltzmann equation for the proteins in the water environment is a typical boundary problem [9]. It means that we deal with a system in which some quantities (the electrostatic constant in the present case) is discontinuous along some surface. In the case of the proteins the boundary is very complicated and it is impossible to define it analytically or even with arbitrary precision. This difficulty imposes some specific limitations on the definition of calculation domain. When one tries to define the boundaries of the calculation domain very close to the molecular surface this introduces very large errors in the solution because the molecular surface itself is defined not exactly. On the other hand it is not recommended to take very large calculation domain because involving a large water volume increases the calculation time. In our calculations we define the domain as a cube which includes the protein molecule and has a volume of $30 - 40\%$ larger than the protein.

3.2 The Structure of the Computer Code PBSOLVE

Input: *ksf* - the number of atoms in the molecule,
xsf(ksf), ysf(ksf), zsf(ksf) - Cartesian coordinates of atoms in the molecule,
rsf(ksf) - the Van der Waals radii of the atoms,
perbox - percent of molecule size increment to set a computational domain,
numq - number of the atom where the charge is placed,
q - the value of the charge .

Definition of the calculating domain: the size of calculating domain is determined by the size of minimal box containing the molecule. Each dimension of the box is increased by the value of perbox and round-off to $1\mathring{A}$ to have an initial grid size h of $1\mathring{A}$.

Construction of the grid: As mentioned above, the size of calculating domain is determined by the size of minimal box, containing the molecule. Each dimension of the box is increased by the value of *perbox* and round-off to $1\mathring{A}$ to have an initial grid size h of $1\mathring{A}$. After the computational domain is defined, a mesh is constructed:

x(m), y(n), z(k) - the coordinates of grid points,
u(m,n,k) - the values of electrostatic potential at grid points,
c(m,n,k) - each cell of the grid has an indicator showing whether that cell belongs to the protein interior or not,
f(m,n,k) - the fixed charge density.

Iterations: The method of successive overrelaxation (SOR) [15] is used for solution. Iterations are stopped when the relative error in solution is less than *eps*. Computations are organized in the multigrid approach by using the sequence of grids. This method provides quick convergence as well as a better control and analysis of obtained approximations. First the problem is solved on the mesh with step size $h = 1\mathring{A}$ by SOR iteration process. The obtained solution is then interpolated on the mesh with a grid size half of the original and new iteration is performed. Similarly, the solution on the fine grid with mesh size of 0.25Åis obtained.

Output: $u(m, n, k)$ - the electrostatic potential on the fine grid,
x(m), y(n), z(k) - coordinates of grid nodes,
h - the step size.

3.3 Parallel Realization of Finite Difference Scheme

We use a finite difference scheme to solve our 3d problem by the overrelaxation method. We find the solution in the three-dimensional perpendicular area. The natural way for parallelization of the solution process is to divide the computation domain into p subdomains by horizontal planes, where p is the number of the processors to be involved in the solution (See Fig.2.).

One would expect that the solving process might be accelerated p times by parallel computation in a cluster with p processores. Of course this is only an idealization; in reality the computers need some preparatory work and a certain amount of time for the data transfer between the processors. Each processor does calculation only at the points in its "own" area and the necessary data are exchanged between the processors which need them. We have been solving the boundary problem by using three different types of grids. At first we tried to implement the iteration process using the approach that all processors work simultaneously and *independently*, each one being calculating the function at the mesh points inside its own subdomain. In this case the overrelaxation method does not lead to correct results at the mesh points near the boundaries of subdomains. The problem is that since the function values have been calculated independently by different processors, the whole dataset is not consistent and needs to ensure the continuity of the function values on the boundaries of adjacent subdomains. We have overcome this problem (see below) by increasing number of iterations. For example, the number of iterations on the first grid was increased from 43 for 1 processor to 81 for 8 processors.

The modified working schedule of the processors looks as following:

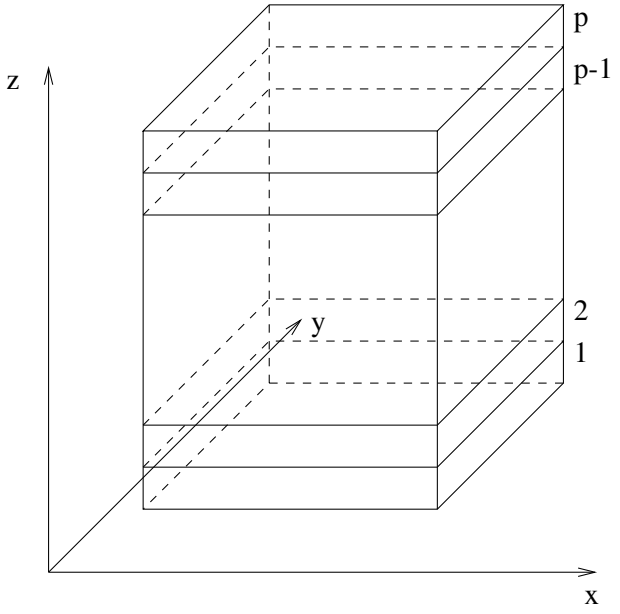

Fig. 2. Partitioning of the calculating domain for parallelization.

– For each grid we divide the calculating domain into p subdomains. Then we organize a multistage calculation process in this way:
  - In the first cycle only the first processor works on the whole mesh and after finishing his work it transmits the function values at the points on the last (the upper) level of the mesh in the first subdomain to the second processor.
  - In the second cycle first two processors work. After the first step of this process, the second processor transmits the function values on at the points in the first (lower) level of the mesh to the first processor which uses these data to calculate of its last-level data. After finishing the calculation the first processor transmits its last level data to the second processor.
  - In the third cycle the first three processors work. After the first step of this process is over, the second and the third processors transmit their first level data to the previous processor (2. to 1. and 3. to 2.) which uses these data to calculate its last level data. After finishing the calculation each processor (except the last one) transmits its last level data to the next processor.
  - *etc.*
  - In the pth cycle and up to the end of the solution process all processors are working. After the first step of this process each processor (except the first one) transmits its first level data to the previous processor (2. to 1., 3. to 2. etc.) which uses these data to calculate its last level data. After

finishing the calculation each processor (except the last one) transmits its last level data to the next processor.

- After finishing the iteration process each nonzero processor sends its calculated data to the first processor (its number is 0) and after data completization the first processor sends whole solution to other ones and the process is repeated for the next grid after interpolating of the previous result to use it in the start iteration.

In this work we use:

- a cubical solution domain,
- three grids with the equidistant nodes (the number of nodes was 21, 41, and 81 in turn),
- the steps of each direction were identical ($h = 1, 0.5, 0.25$).

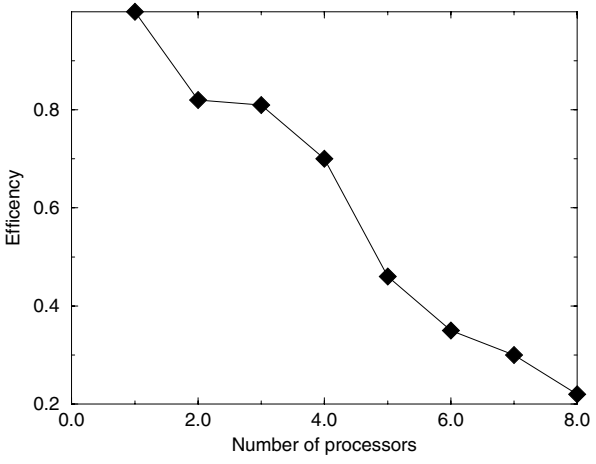

Fig. 3. Parallelization efficiency vs number of computers.

The MPI package [16] was used for the parallel implementation of the algorithm.

The parallelization efficiency is shown in Fig.3. One can see that the parallelization is effective up to 4 processors. For larger number of processors, data transfer time is increased drastically and the efficiency drops.

4 Conclusions and Future Prospects

Future prospects deal with including the nonlinear term in iteration process to obtain the solution of nonlinear PB equation. The relaxation processes used to solve linear equation may be applied to the nonlinear equation. Next improvement of the program concerns the treatment of the solute-solvent interface boundary. There are two ways to improve the mapping of the interface

boundary onto the grid. The first one is multilevel mesh refinement technique [17]. The method allows for accurate calculation of electrostatic potential over domains with local mesh refinement patches. The second is the exploitation of well-developed finite [18] and boundary [19] element techniques to complex molecular surface. To describe in more exact way the behavior of the potential at the infinity, we may use coupled boundary integral finite elements formalism.

As shown in Fig.3 the efficiency of parallelization decreases drastically after 4 processors. Of course we donot satisfy with such a result. Recently, we developed a parallelization method for protein energy calculation [20] which is working highly effective up to 30 processors. Now the work is underway to find new parallelization strategies for solution of Poisson-Boltzmann equation which can provide a higher efficiency.

ACKNOWLEDGMENTS: One of the authors is grateful to A.P. Sapozhnikov and T.F Sapozhnikova for fruitful consultations on parallelization and calculations on the SPP2000.

References

1. C. Chotia, J.Janin, Nature 256 (1974) 705-708.
2. B. Honig, A. Nicholls, Science 268 (1995) 1144-1149.
3. B. Hingerty, R. H. Richie, T. L. Ferrel, J. Turner, Biopolymers 24 (1985) 427-439.
4. T. Ooi, M. Oobatake, G. Nemethy, H. A. Scheraga, Proc. Natl. Acad. Sci. USA, 84 (1987) 3086-3090.
5. T. J. Richmond, J. Mol. Biol. 178 (1984) 63-89.
6. M. Connolly, J. Appl. Cryst. 16 (1983) 548.
7. F. Eisenhaber, *et al*, J. Comp. Chem. 16 (1995) 273-284.
8. M. K. Gilson, B. Honig, Biopolymers 25 (1986) 2097.
9. D. Jackson, Classical Electrodynamics (Wiley, New York, 1975).
10. M. Born,Z. Phys. 1 (1920) 45.
11. W. C. Still,W, A. Tempczyk, R. C. Hawley, T. J. Hendricson , J. Am. Chem. Soc. 112 (1990) 6127.
12. Di Qui, P. Shenkin, F.P. Hollinger, W.C. Still, J. Phys. Chem. A 101 (1997) 3005-3014.
13. B. N. Domany and Ch. L. Brooks III, J. Phys. Chem. B 103 (1999) 3765-3773.
14. F. Eisenmenger, U. Hansmann, Sh. Hayryan, C.-K. Hu, [SMMP] The Modern Package for Protein Simulations, *to be submitted in 2000*.
15. L.A. Hageman, D.M. Young, in: Applied Iterative Methods (Academic Press, New York, 1981).
16. M. Snir, *et al*, MPI (The MIT Press, Cambrage (USA), 1997).
17. D. Bai, A. Brandt, SIAM, J. Sci. Stat. Comput., 8 (1987) 109.
18. O.C. Zienkiewics, in: The Finite Element Method (McGraw-Hill, London,1977).
19. C.A. Brebbia, J.C.F. Telles, L.C. Wrobel, in: Boundary Element Techniques: Theory and Applications in Engineering (Springer-Verlag, Berlin, 1984).
20. Sh. Hayryan, C.-K. Hu, S.-Y. Hu, R.-J. Shang, J. Comp. Chem., submitted.