1. Методы вычисления дальних взаимодействий

2. Метод граничных Элементов

3. Метод быстрых Мультиполей

Сергей Грудинин, 07.05.2010

Методы вычисления дальних взаимодействий

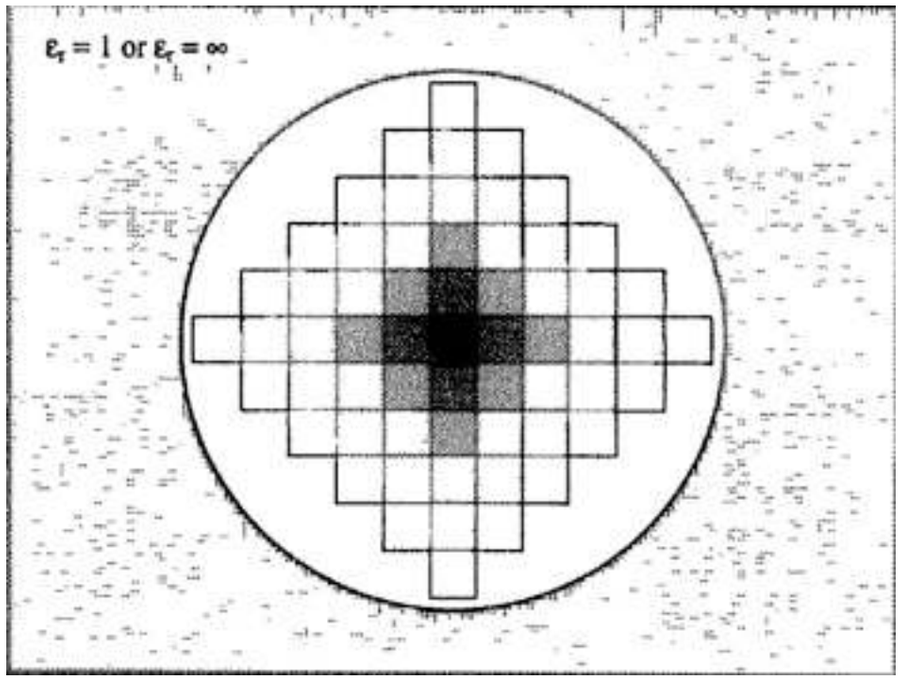
Метод Эвальда

потенциал в центральной коробке

$$\mathcal{V} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{q_i q_j}{4\pi\varepsilon_0 r_{ij}}$$

взаимодействия с соседями:

$$\mathcal{V} = \frac{1}{2}\sum_{n\,\mathrm{box}=1}^{6}\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{q_i q_j}{4\pi\varepsilon_0|\mathbf{r}_{ij}+\mathbf{r}_{\mathrm{box}}|}$$



для $\mathbf{n}\ (= (n_x L, n_y L, n_z L)$

$$\mathcal{V} = \frac{1}{2}\sum_{|\mathbf{n}|=0}^{\prime}\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{q_i q_j}{4\pi\varepsilon_0|\mathbf{r}_{ij}+\mathbf{n}|}$$

где в центральной коробке не считаются взаимодействия i=j

Метод Эвальда

теперь разложим потенциал на сумму двух быстро сходящихся членов:

$$\frac{1}{r} = \frac{f(r)}{r} + \frac{1-f(r)}{r}$$

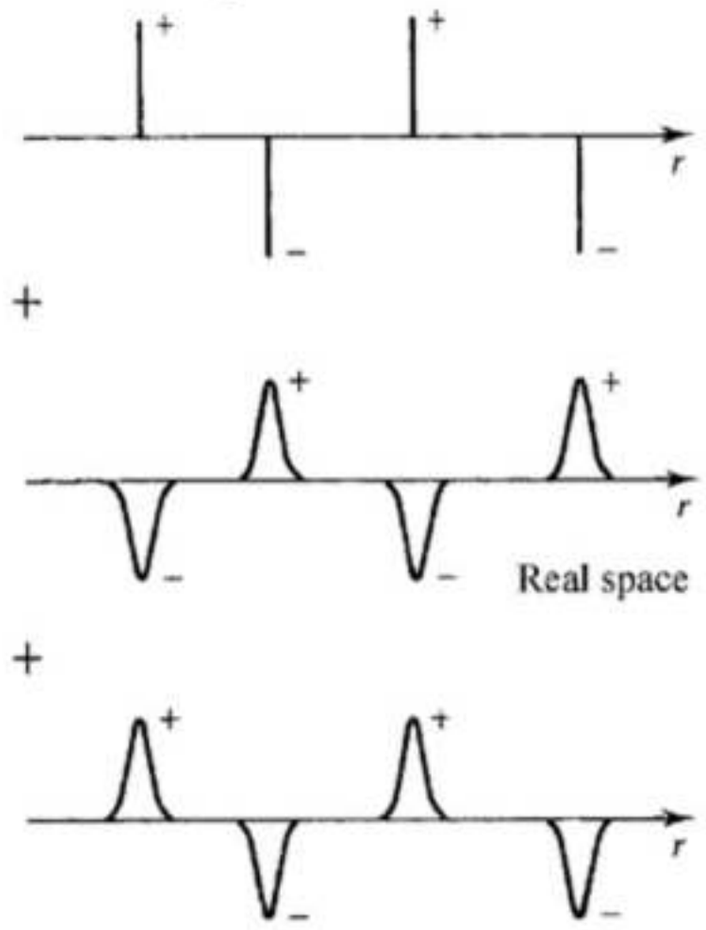
каждый заряд окружен двумя гауссовыми облаками противоположного заряда

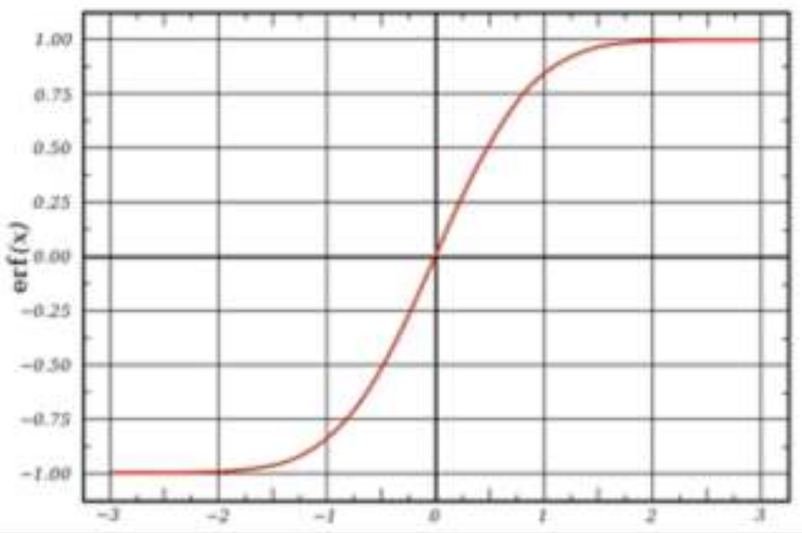
$$\rho_{\text{Gauss}}(r) = -q_i(\alpha/\pi)^{\frac{3}{2}} \exp(-\alpha r^2)$$

быстро затухающий член:

$$\begin{aligned} \phi_{\text{short-range}}(r) &= \frac{q_i}{r} - \frac{q_i}{r}\text{erf}\left(\sqrt{\alpha}r\right) \\ &= \frac{q_i}{r}\text{erfc}\left(\sqrt{\alpha}r\right), \end{aligned}$$

$$U_{\text{short-range}} = \frac{1}{2}\sum_{i\neq j}^{N} q_i q_j \text{erfc}\left(\sqrt{\alpha}r_{ij}\right)/r_{ij}$$





Метод Эвальда

уравнение Пуассона

$$-\nabla^2\phi(\mathbf{r}) = 4\pi\rho(\mathbf{r})$$

решение в Фурье-пространстве:

$$\tilde{\phi}(k) = \frac{4\pi}{k^2}\tilde{\rho}_P(k) \qquad\qquad f(\mathbf{r}) = \frac{1}{V}\sum_{l=-\infty}^{\infty}\tilde{f}(k)e^{ik\cdot r}$$

Фурье часть суммы Эвальда:

$$\begin{aligned}
\phi_1(r) &= \frac{1}{V}\sum_{\mathbf{k}\neq 0}\phi_1(\mathbf{k})\exp(i\mathbf{k}\cdot\mathbf{r}) \\
&= \sum_{\mathbf{k}\neq 0}\sum_{i=1}^{N}\frac{4\pi q_j}{k^2}\exp[i\mathbf{k}\cdot(\mathbf{r}-\mathbf{r}_j)]\exp(-k^2/4\alpha), \\
\mathcal{U}_1 &\equiv \frac{1}{2}\sum_i q_i\phi_1(r_i) \\
&= \frac{1}{2}\sum_{\mathbf{k}\neq 0}\sum_{i,j=1}^{N}\frac{4\pi q_i q_j}{Vk^2}\exp[i\mathbf{k}\cdot(\mathbf{r}_i-\mathbf{r}_j)]\exp(-k^2/4\alpha) \\
&= \frac{1}{2V}\sum_{\mathbf{k}\neq 0}\frac{4\pi}{k^2}|\rho(\mathbf{k})|^2\exp(-k^2/4\alpha),
\end{aligned}$$

Метод Эвальда

взаимодействие заряда с самим собой:

$$(1/2)q_i\phi_{self}(r_i)$$

$$\begin{aligned} \mathcal{U}_{self} &= \frac{1}{2}\sum_{i=1}^{N} q_i\phi_{self}(r_i) \\ &= (\alpha/\pi)^{\frac{1}{2}}\sum_{i=1}^{N} q_i^2. \end{aligned}$$

окончательное выражение:

$$\begin{aligned} \mathcal{U}_{Coul} = {} & \frac{1}{2V}\sum_{\mathbf{k}\neq 0}\frac{4\pi}{k^2}|\rho(\mathbf{k})|^2\exp(-k^2/4\alpha) \\ & - (\alpha/\pi)^{\frac{1}{2}}\sum_{i=1}^{N} q_i^2 \\ & + \frac{1}{2}\sum_{i\neq j}^{N}\frac{q_i q_j \operatorname{erfc}\left(\sqrt{\alpha}r_{ij}\right)}{r_{ij}}. \end{aligned}$$

Неявное водное окружение

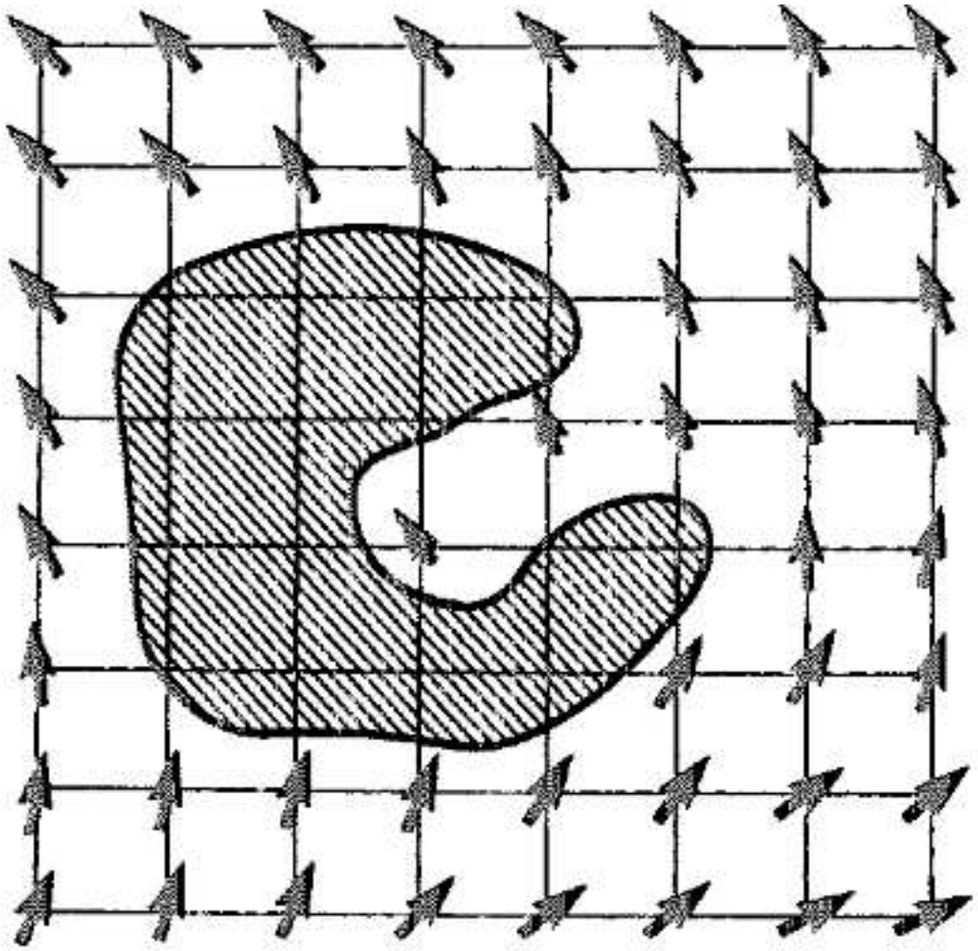
обобщенная модель Борна

$$G_{\text{elec}} = \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{q_i q_j}{\varepsilon r_{ij}} - \frac{1}{2}\left(1 - \frac{1}{\varepsilon}\right) \sum_{i=1}^{N} \frac{q_i^2}{a_i}$$

метод диполей Ланжевена

$$\mu_i = \mu_0 \frac{\mathbf{E}_i}{|\mathbf{E}_i|} \left[\frac{\exp[C\mu_0|\mathbf{E}_i|/k_B T] + \exp[-C\mu_0|\mathbf{E}_i|/k_B T]}{\exp[C\mu_0|\mathbf{E}_i|/k_B T] - \exp[-C\mu_0|\mathbf{E}_i|/k_B T]} - \frac{1}{C\mu_0|\mathbf{E}_i|/k_B T} \right]$$

$$\Delta G_{\text{sol}} = -\frac{1}{2} \sum_i \boldsymbol{\mu}_i \cdot \mathbf{E}_i^0$$



уравнение Пуассона-Больцмана

$$\nabla^2\phi(\mathbf{r}) = -\frac{4\pi\rho(\mathbf{r})}{\varepsilon}$$

или

$$\nabla\cdot\varepsilon(\mathbf{r})\nabla\phi(\mathbf{r}) = -4\pi\rho(\mathbf{r})$$

распределение ионов:

$$n(\mathbf{r}) = \mathcal{N}\exp(-\mathcal{V}(\mathbf{r})/k_{\mathrm{B}}T)$$

уравнение Пуассона-Больцмана:

$$\nabla\cdot\varepsilon(\mathbf{r})\nabla\phi(\mathbf{r}) - \kappa'\sinh[\phi(\mathbf{r})] = -4\pi\rho(\mathbf{r})$$

можно разложить в ряд:

$$\nabla\cdot\varepsilon(\mathbf{r})\nabla\phi(\mathbf{r}) - \kappa'\phi(\mathbf{r})\left[1 + \frac{\phi(\mathbf{r})^2}{6} + \frac{\phi(\mathbf{r})^4}{120} + \cdots\right] = -4\pi\rho(\mathbf{r})$$

первый член разложения даст:

$$\nabla\cdot\varepsilon(\mathbf{r})\nabla\phi(\mathbf{r}) - \kappa'\phi(\mathbf{r}) = -4\pi\rho(\mathbf{r})$$

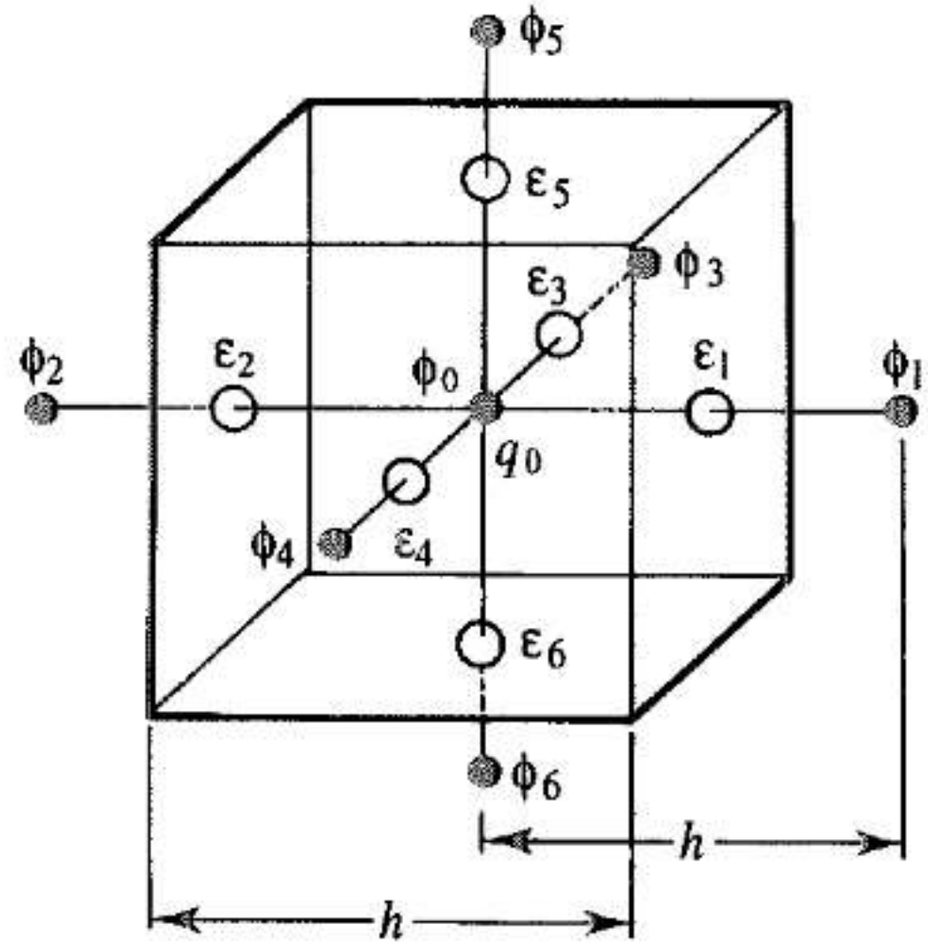
решение на сетке

проинтегрируем обе стороны:

$$\nabla \cdot \varepsilon(\mathbf{r}) \nabla \phi(\mathbf{r}) - \kappa' \phi(\mathbf{r}) = -4\pi\rho(\mathbf{r})$$

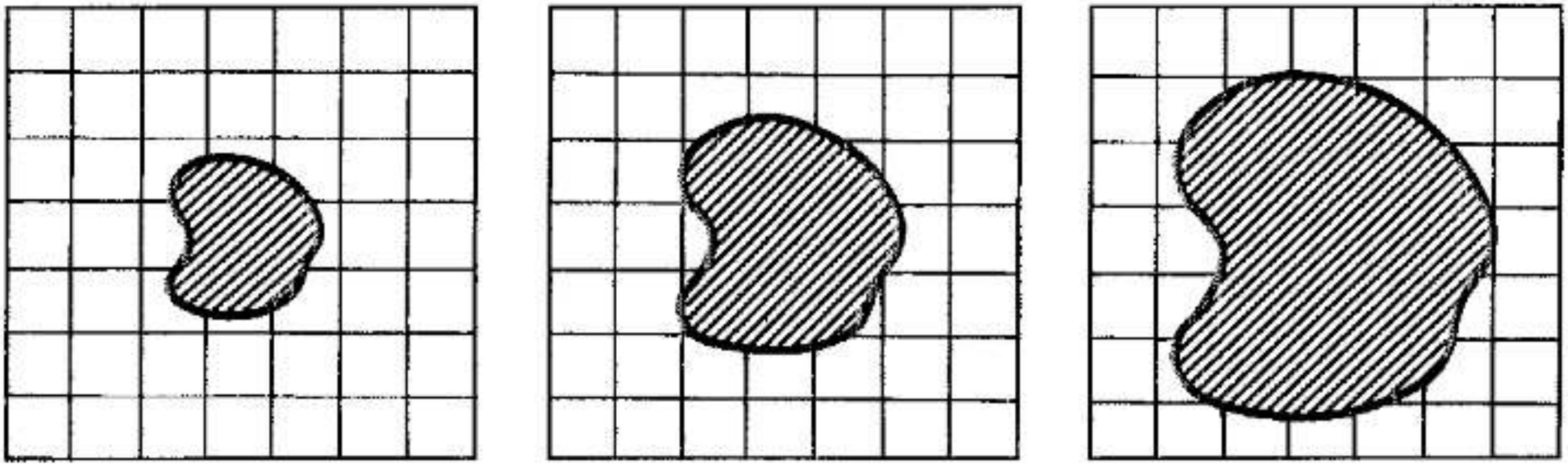
после применения формулы Остроградского получим:

$$\phi_0 = \frac{\sum \varepsilon_i \phi_i + 4\pi \dfrac{q_0}{h}}{\sum \varepsilon_i + k_0'^2 f(\phi_0)}$$



- решается при помощи линейной системы уравнений
- полученная матрица - разряженная, решается за линейное кол-во операций
- проблема с граничными условиями!
- применяем метод нескольких разрешений

метод нескольких разрешений



- последовательно находим решение уравнения Пуассона-Больцмана на сетке
- уменьшаем шаг сетки
- граничные условия берем из предыдущего решения
- на первом шаге потенциал на границе считаем аналитически или берем равным нулю

Метод Быстрых Мультиполей

→ отдельная презентация

Метод Граничных Элементов

(на ангийском)

Introduction

Explicit solvent representation requires a lot of computational time.

In some systems water molecules are $\approx$ 90% of the system size.

- empirical models:
  - solvation free energy is a sum of atom or group contributions
  - contributions are linear functions of SAS areas or volumes
  - these models incorporate the hydrophobic and electrostatic components of solvation, but omit the solvent charges screening
- continuum electrostatics:
  - different dielectric constants for the solvent and the solute interior
  - solve the Poisson-Boltzmann equation

Implicit models

- empirical models:
  - a simple model: $G^{solvation} = \sum_i \sigma_i ASA_i$
  - effective energy function:

    $\Delta G^{solvation} = \Delta G^{reference} - \sum_j \int_{V_j} f_i(r_{ij}) d^3r$
  - GB approximations: $\Delta E_{ij}^{screening} = (1 - \frac{1}{\epsilon}) \frac{q_i q_j}{f(r_i, r_j)}$

    $\Delta G^{solvation} = \sum_i (\Delta E_i^{self} - \frac{1}{2} \sum_{j \neq i} E_{ij}^{screening} + \Delta E_i^{nonpolar})$
- finite difference PB:
  - $\Delta G^{PB} = \Delta G^{PB}_{water} - \Delta G^{PB}_{vacuum}$
  - $\Delta G^{solvation} = \Delta G^{PB} + \gamma ASA$

Poisson-Boltzmann Equation

Gauss law in vacuum:

$$\nabla \cdot \vec{E}(\vec{r}) = 4\pi\rho(\vec{r})$$

In the media with dielectric constant ϵ:

$$\vec{D}(\vec{r}) = \epsilon\vec{E}(\vec{r}),$$

$$\nabla \cdot \epsilon\nabla\phi(\vec{r}) = -4\pi\rho,$$

Charge distribution:

$$\rho = \rho_{fixed} + \rho_{mobile},$$

$$\rho_{fixed} = \sum q_i\delta(r - r_i),$$

$$\rho_{mobile} = \sum z_i n_0 exp^{\frac{-z_i\phi}{k_B T}}$$

For 1-1 salt:

$$\rho_{mobile} = \sum z_i n_0 sinh(\frac{-z_i\phi}{k_B T})$$

Or in the linear case:

$$\rho_{mobile} = \sum z_i n_0 \frac{-z_i\phi}{k_B T}$$

Boundary Element Method

Electrostatic boundary conditions on the molecular surface:

$$\mathbf{D_{in}} \cdot \mathbf{n} = \mathbf{D_{out}} \cdot \mathbf{n}$$

$$(\mathbf{E_{out}} - \mathbf{E_{in}}) \cdot \mathbf{n} = 4\pi\sigma$$

Taking into account that $\mathbf{D} = \epsilon\mathbf{E}$:

$$\sigma = \left(\frac{\epsilon_{in} - \epsilon_{out}}{4\pi\epsilon_{in}}\right)\mathbf{E_{out}} \cdot \mathbf{n}$$

On the other hand from Poisson equation:

$$\mathbf{E_{out}}(\mathbf{r}) = \sum_{\mathbf{i}} \frac{\mathbf{q_i}(\mathbf{r}-\mathbf{r_i})}{\epsilon_{\mathbf{in}}|\mathbf{r}-\mathbf{r_i}|^3} + \mathbf{2}\pi\sigma(\mathbf{r})\mathbf{n}(\mathbf{r}) + \oint \frac{(\mathbf{r}-\mathbf{r_s})}{|\mathbf{r}-\mathbf{r_s}|^{\mathbf{3}}}\sigma_{\mathbf{s}}\mathbf{ds}$$

And finally we get:

$$\sigma(\mathbf{r}) = \left(\frac{\epsilon_{\mathbf{in}} - \epsilon_{\mathbf{out}}}{4\pi\epsilon_{\mathbf{in}}}\right)\sum_{\mathbf{i}} \frac{\mathbf{q_i}(\mathbf{r}-\mathbf{r_i})\cdot\mathbf{n}(\mathbf{r})}{\epsilon_{\mathbf{in}}|\mathbf{r}-\mathbf{r_i}|^3} + \left(\frac{\epsilon_{\mathbf{in}} - \epsilon_{\mathbf{out}}}{\mathbf{2}\epsilon_{\mathbf{in}}}\right)\sigma(\mathbf{r}) + \left(\frac{\epsilon_{\mathbf{in}} - \epsilon_{\mathbf{out}}}{4\pi\epsilon_{\mathbf{in}}}\right)\oint \frac{(\mathbf{r}-\mathbf{r_s})\cdot\mathbf{n}(\mathbf{r})}{|\mathbf{r}-\mathbf{r_s}|^3}\sigma_{\mathbf{s}}\mathbf{ds}$$

Or after the regrouping terms and discretization:

$$\sigma_j = \frac{1}{2\pi}\frac{\epsilon_{in} - \epsilon_{out}}{\epsilon_{in} + \epsilon_{out}} s_j \left[\sum_i \frac{q_i(\mathbf{r_j}-\mathbf{r_i})}{\epsilon_{in}|\mathbf{r_j}-\mathbf{r_i}|^3} + \sum_{k\neq j} \frac{(\mathbf{r_j}-\mathbf{r_k})}{|\mathbf{r_j}-\mathbf{r_k}|^3}\sigma_k\right] \cdot \mathbf{n_j}$$

Matrix Form of the BEM equation

The BEM equation can be written in a matrix form:

$$\sum_{ij} A_{ij}\sigma_j = b_i$$

with the matrix

$$A_{ij} = 2\pi\frac{\epsilon_{in}+\epsilon_{out}}{\epsilon_{in}-\epsilon_{out}}\delta_{ij} - s_i\frac{(\mathbf{x_i}-\mathbf{x_j})\cdot\mathbf{n_i}}{|\mathbf{x_i}-\mathbf{x_j}|^3}$$

and the vector

$$b_i = s_i\sum_k \frac{q_k(\mathbf{x_i}-\mathbf{x_k})\cdot\mathbf{n_i}}{\epsilon_{in}|\mathbf{x_i}-\mathbf{x_k}|^3}$$

To solve the system directly:

- either calculation and storage of all the matrix $\mathbf{A}$ elements, storage is $O(N^2)$
- or the A_{ij} calculated on the fly, CPU usage is $O(N^2)$ operations per iteration

Matrix Form of the BEM equation

Diagonal elements of matrix A: $A_{ii} =$

$$\begin{cases} 2\pi\frac{\epsilon_{in}+\epsilon_{out}}{\epsilon_{in}-\epsilon_{out}},\ flat\ area\ elements \\ 2\pi\frac{\epsilon_{in}+\epsilon_{out}}{\epsilon_{in}-\epsilon_{out}} - \sqrt{\frac{\pi s_i}{K_i^2}},\ area\ elements\ with\ Gaussian\ curvature\ K \\ \frac{4\pi\epsilon_{out}}{\epsilon_{in}-\epsilon_{out}} + \sum_{i\neq j} A_{ij}^T,\ normalization\ conditions \end{cases}$$

- first approximation works rather well
- curvature is not easy to estimate
- normalization did not advance much in our tests

Iterative Solvers

The BEM system is linear, so a number of iterative solvers exist:

- Stationary Methods: can be expressed as $x^{(k)} = Bx^{(k+1)} + c$
  - **Jacobi** - The resulting method is easy to understand and implement, but convergence is slow.
  - **Gauss-Seidel** - Similar to Jacobi, but a bit faster
  - **Successive Overrelaxation (SOR)** - May converge faster than Gauss-Seidel by an order of magnitude.

  To use these methods the BEM equation can be written as:

  $$\sigma_i = U_i(\sigma_j)$$

- Convergence depends on the matrix form
- May never converge

Iterative Solvers

- Nonstationary Methods for general matrices:
  - **Generalized Minimal Residual (GMRES)** - should be restarted, otherwise requires too much memory
  - **BiConjugate Gradient (BiCG)** - convergence may be irregular
  - **Quasi-Minimal Residual (QMR)** - smooth out the irregular convergence behavior of BiCG
  - **Conjugate Gradient Squared (CGS)** - convergence may be much more irregular than for BiCG
  - **Biconjugate Gradient Stabilized (Bi-CGSTAB)** - obtains smoother convergence than CGS
  - **Chebyshev Iteration** - knowledge of the extremal eigenvalues is required
- GMRES and QMR perform the best

Iterative Solvers

Comparison for different iterative algorithms for the BPTI protein with 2500 surface elements:

	CGS	BiCGSTAB	GMRES	BiCG	QMR	Cheby
steps	24	23	30	36	37	-

Convergence of iterative methods. BPTI protein with point density 5 points/nm.

Tolerance = 1e-6, 2500 surface elements, no preconditioner was used

BEM against FDPB

FDPB:

- - solve PB equation on a 3D grid. Storage is $O(N^3)$
- - boundary conditions must be always implied
- - solved several times iteratively
- - solvent/solute boundary has a finite size
- - atoms are mapped on the grid
- + different dielectric constants can be used

BEM:

- + solve PB equation on a 2D grid. Storage is $O(N^2)$
- + exact atom positions
- + solvent/solute boundary is defined by the mesh
- - every region has a unique dielectric constant

Preconditioners

Preconditioned problem

$$\mathcal{P}\mathbf{A}\mathbf{x} = \mathcal{P}\mathbf{b}$$

- How do we choose/construct $\mathcal{P}$?
- Simple idea: Use a (cheap!) approximation to the inverse.

$$\mathcal{P} \approx \mathbf{A}^{-1}.$$

- Pointless in its exact form!
- Some popular choices:
  - Diagonal preconditioner (also in block form):

$$\mathcal{P} = (diag\mathbf{A})^{-1}$$

  - Incomplete LU factorisation:

$$\mathcal{P} = (\mathbf{L}_{inc}\mathbf{U}_{inc})^{-1}$$

- In general, the choice of preconditioner is problem dependent.

BEM Forces

Electrostatic energy evaluated as:

$$U = \frac{1}{\epsilon_{in}} \sum_{i \neq j} \frac{q_i q_j}{|\mathbf{r_i} - \mathbf{r_j}|} + \sum_{k,j} \frac{\sigma_k q_j}{|\mathbf{r_j} - \mathbf{r_k}|}$$

Its derivatives are:

$$\frac{\partial}{\partial x_j} U =$$
$$\frac{\partial}{\partial x_j} \left(\frac{1}{\epsilon_{in}} \sum_{i \neq j} \frac{q_i q_j}{|\mathbf{r_i} - \mathbf{r_j}|} \right) + \sum_{k,j} \sigma_k \frac{\partial}{\partial x_j} \frac{q_j}{|\mathbf{r_j} - \mathbf{r_k}|} + \sum_{k,j} \frac{q_j}{|\mathbf{r_j} - \mathbf{r_k}|} \frac{\partial}{\partial x_j} \sigma_k$$

The only problem arises with $\frac{\partial}{\partial x_j} \sigma_k$ in the last term. It can be found from the matrix equation derivatives:

$$\frac{\partial A}{\partial x_j} \sigma + A \frac{\partial}{\partial x_j} \sigma = \frac{\partial b}{\partial x_j}$$

- requires derivatives of molecular surface

Another beautiful approximation for the solvation forces (Cancès and Menucci):

$$\frac{\partial}{\partial x_j} G^{solv} \approx \frac{4\pi\epsilon}{\epsilon - 1} \oint \sigma^2(k) (U_\Gamma^{(j)}(k) \cdot n(k)) dk$$

- requires calculation of the moving front $U_\Gamma^{(j)}(k)$

Molecular Surfaces

Parametric surfaces:

- spherical blobs with a local smoothing
- Gaussian kernels

The density D in every point of the system is:

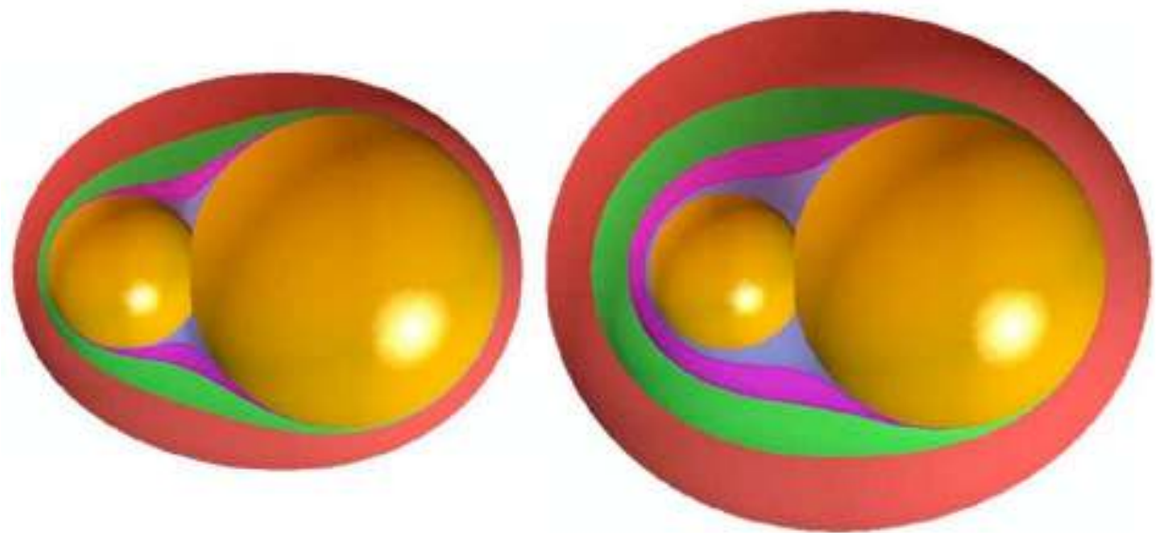
$$D = \sum_i \exp^{-a_i(\frac{|\vec{r}-\vec{r_i}|^2}{R_i^2}-1)}$$

applying $C = a_i/R_i$:

$$D = \sum_i \exp^{-C(|\vec{r}-\vec{r_i}|^2-R_i^2)}$$

Surface is then defined as $D = 1$

The difference between C(a) and a_i(b) (from C. Bajaj):



Molecular Surface Derivatives

Surface D derivatives are:

$$\frac{\partial D}{\partial x_j} = 2C(\vec{r} - \vec{r_j})\exp^{-C(|\vec{r}-\vec{r_j}|^2 - R_j^2)}$$

Surface gradient is:

$$\vec{n} = -2C\sum_i(\vec{r} - \vec{r_i})\exp^{-C(|\vec{r}-\vec{r_i}|^2 - R_i^2)} = -\sum_j \frac{\partial D}{\partial x_j} = -\nabla D$$

Shape operator:

$$\mathbf{S} = \frac{1}{\|\nabla \mathrm{D}\|^3}\begin{pmatrix} d_{xx}(d_y^2 + d_z^2) & -d_x d_y d_{yy} & -d_x d_z d_{zz} \\ -d_y d_x d_{xx} & d_{yy}(d_x^2 + d_z^2) & -d_y d_z d_{zz} \\ -d_z d_x d_{xx} & -d_z d_y d_{yy} & d_{zz}(d_x^2 + d_y^2) \end{pmatrix}$$

Gaussian K and mean H curvatures:

$$\begin{aligned} K &= det(\mathbf{S}) \\ H &= 1/2\, Tr(\mathbf{S}) \end{aligned}$$

Molecular Surface Area

Gauss-Bonnet theorem:

$$\int_M K dA + \int_{\partial M} k_g ds = 2\pi\chi(M)$$

If the boundary ∂M is a union of geodesic curves:

$$\int_M K dA = 2\pi\chi(M) - \sum \alpha_i$$

For triangular meshes it can be rewritten as:

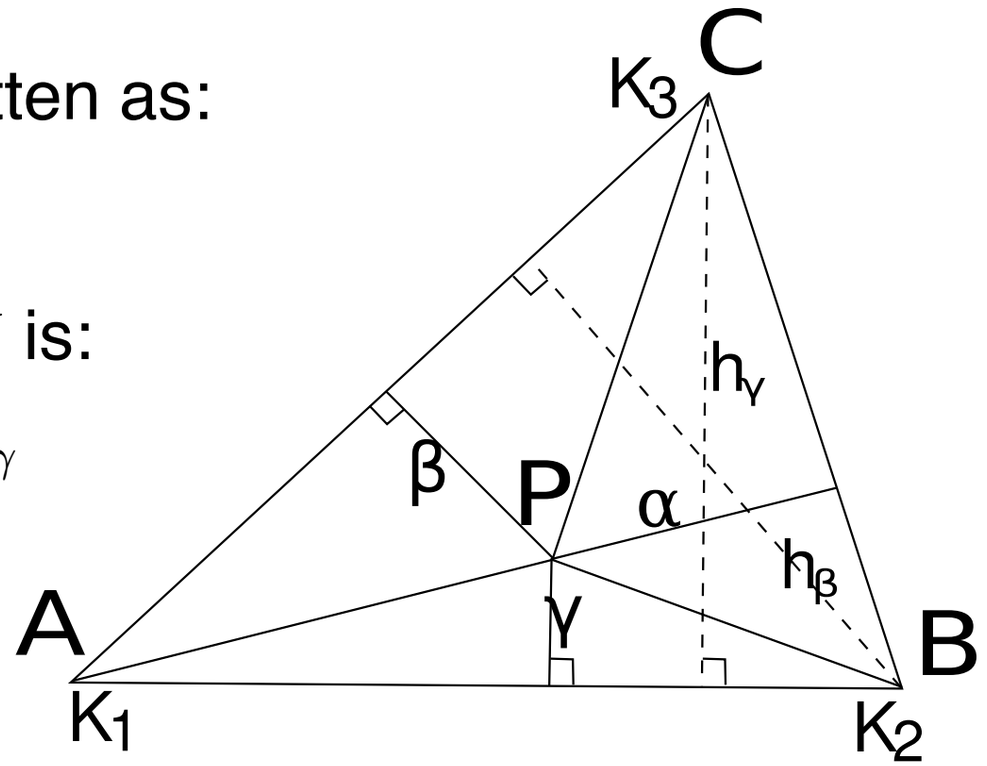
$$\int_M K dA = \sum A_i - \pi$$

For a triangle ΔABC the curvature K is:

$$K = K_1\alpha/h_\alpha + K_2\beta/h_\beta + K_3\gamma/h_\gamma$$

where:

$$\alpha/h_\alpha + \beta/h_\beta + \gamma/h_\gamma = 1$$

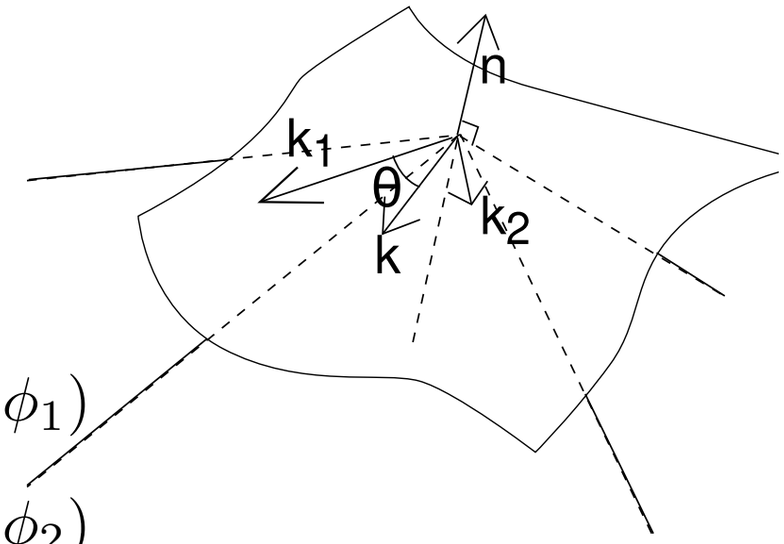


Molecular Surface Area

Gaussian curvature without Shape operator: Euler equation

$$k = k_1 cos^2(\theta) + k_2 sin^2(\theta)$$

Having normal curvatures c_i in 3 directions:



$$\begin{cases} c_1 = k_1 cos^2(\theta) + k_2 sin^2(\theta) \\ c_2 = k_1 cos^2(\theta - \phi_1) + k_2 sin^2(\theta - \phi_1) \\ c_3 = k_1 cos^2(\theta - \phi_2) + k_2 sin^2(\theta - \phi_2) \end{cases}$$

$$\begin{cases} cos^2(\theta) = \frac{c_1 - k_2}{k_1 - k_2};\ sin^2(\theta) = \frac{k_1 - c_1}{k_1 - k_2} \\ c_2 - k_2 = cos^2(\phi_1)(c_1 - k_2) + sin^2(\phi_1)(k_1 - c_1) + sin(2\phi_1)\sqrt{(c_1 - k_2)(k_1 - c_1)} \\ c_3 - k_2 = cos^2(\phi_2)(c_1 - k_2) + sin^2(\phi_2)(k_1 - c_1) + sin(2\phi_2)\sqrt{(c_1 - k_2)(k_1 - c_1)} \end{cases}$$

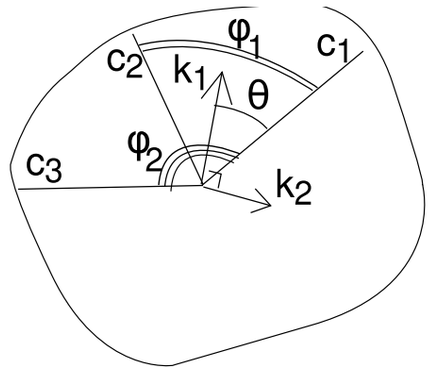
The latter system of equation can be rewritten in terms of the Gaussian curvature $K = k_1 k_2$ and the mean curvature $H = (k_1 + k_2)/2$:

$$\begin{cases} k_1 + k_2 = \dfrac{\frac{c_2 - c_1 cos2\phi_1}{sin2\phi_1} - \frac{c_3 - c_1 cos2\phi_2}{sin2\phi_2}}{\frac{sin^2\phi_2}{sin2\phi_2} - \frac{sin^2\phi_1}{sin2\phi_1}} \\ k_1 k_2 = c_1(k_1 + k_2) - c_1^2 - (\frac{c_2 - c_1 cos2\phi_1}{sin2\phi_1} - \frac{sin^2\phi_1 (k_1 + k_2)}{sin2\phi_1})^2 \end{cases}$$

Molecular Surface Area

Or in a simpler form:

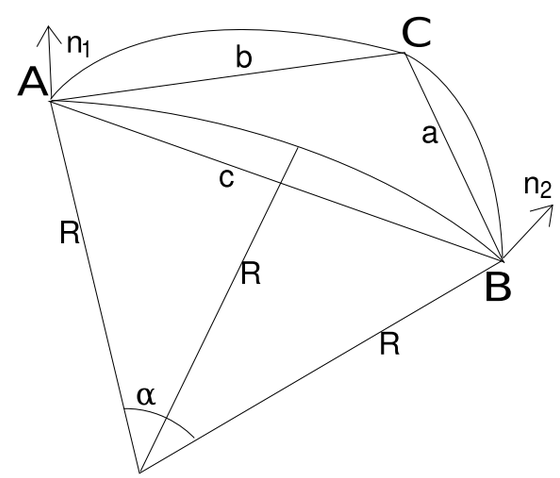
$$\begin{cases} k_1 + k_2 = \frac{c_2 sin2\phi_2 - c_3 sin2\phi_1 - c_1 sin(2\phi_2 - 2\phi 1)}{2 sin\phi_1 sin\phi_2 sin(\phi_2 - \phi_1)} \\ k_1 k_2 = c_1(k_1 + k_2) - c_1^2 - (\frac{c_2 - c_1 cos2\phi_1}{sin2\phi_1} - \frac{sin^2\phi_1(k_1 + k_2)}{sin2\phi_1})^2 \end{cases}$$



Normal curvatures c_i are approximated by:

$$\begin{cases} cos(\alpha) = \vec{n}_1 \cdot \vec{n}_2 \\ sin(\alpha/2) = \frac{c}{2R} \end{cases}$$

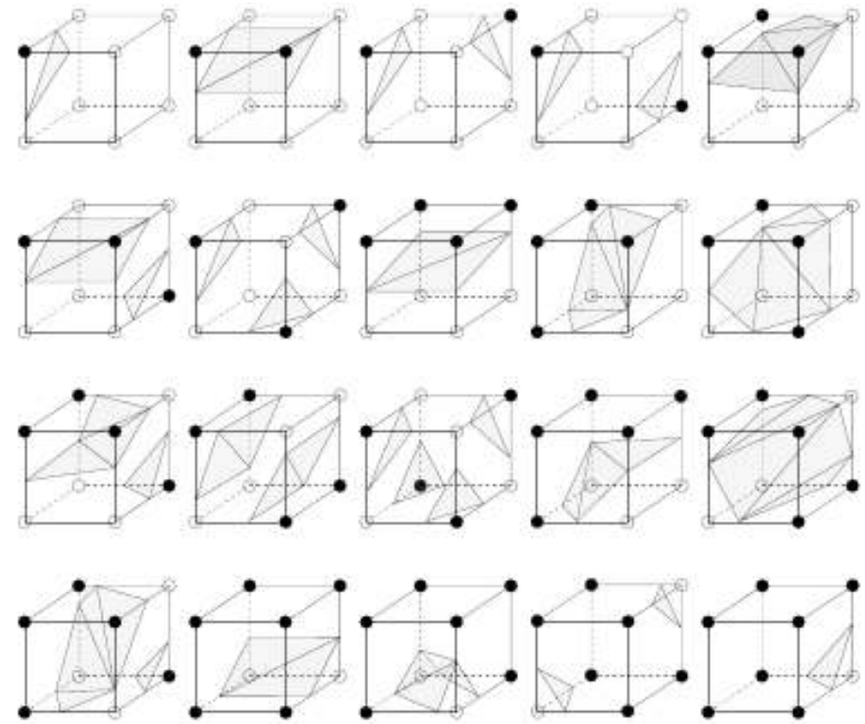
$$c \equiv \frac{1}{R} = \frac{\sqrt{2 - 2\vec{n}_1 \cdot \vec{n}_2}}{c}$$

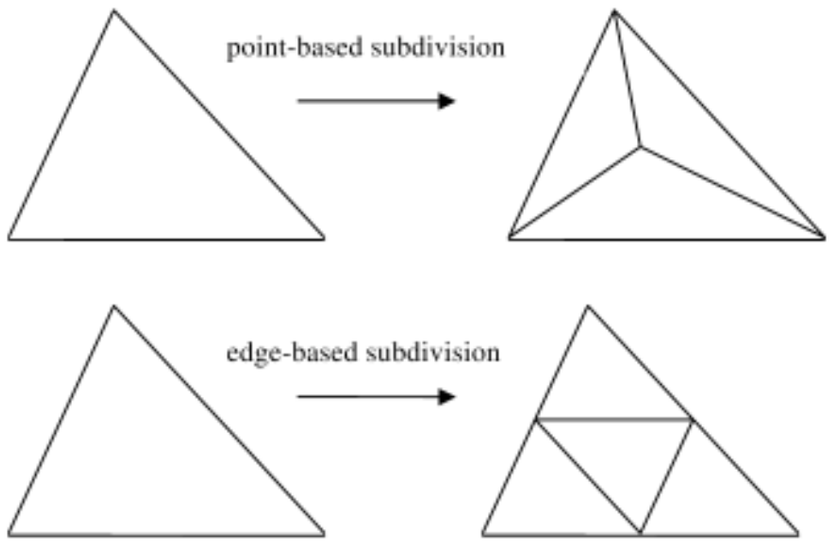


- this approximation is much simpler than the Shape operator
- works very well only for meshes with good quality triangles
- large errors when $\phi \approx 180°$

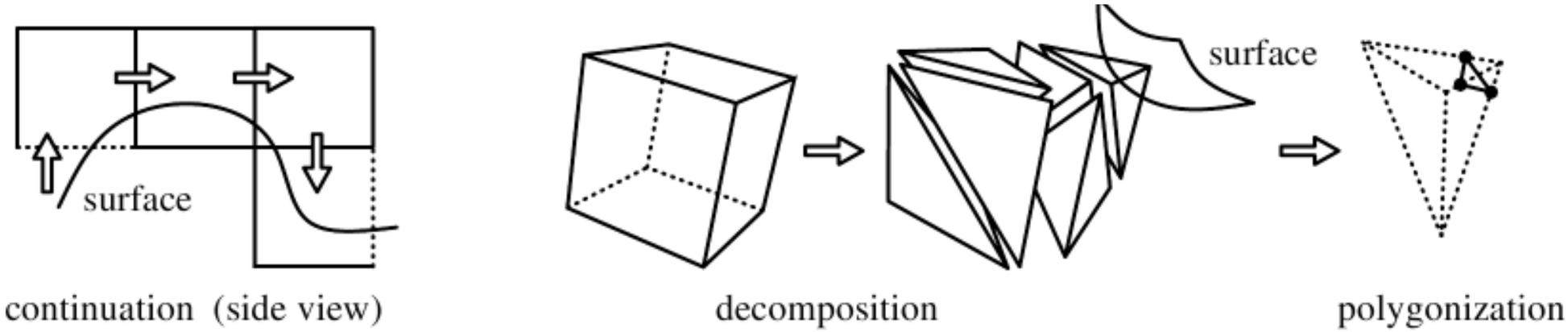
Meshing Techniques

- marching algorithms
  - marching cubes
  - marching tetrahedra
  - marching triangles
- decimation algorithms
  - edge decimation
  - vertex decimation
- refinement algorithms
  - Loop algorithm
  - SQRT3 algorithm
- postprocessing techniques





Marching Cubes



continuation (side view) decomposition polygonization

- + very robust
- - poor mesh quality

Improvements:

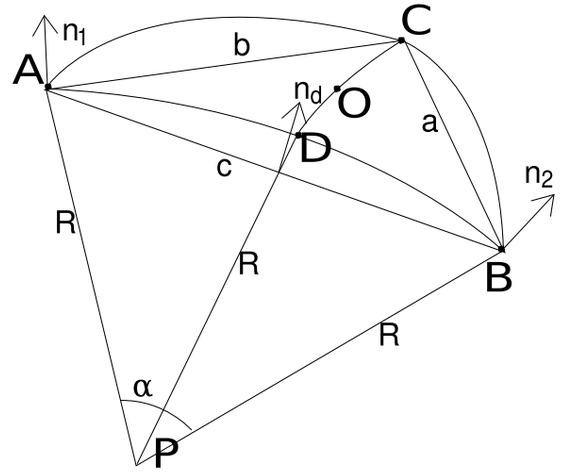
- edge decimation
- subsequent prediction-correction SQRT3-like refinement

Result:

- good mesh quality
- adaptivity, e.g. to curvature

Mesh Refinement

Prediction step:



- calculate position D on the middle of the arc AB
- calculate normal n_d as an average of $\vec{n}_1$ and $\vec{n}_2$
- calculate normal curvature of the arc DOC
- calculate the refined O position from the triangle centroid

Correction step:

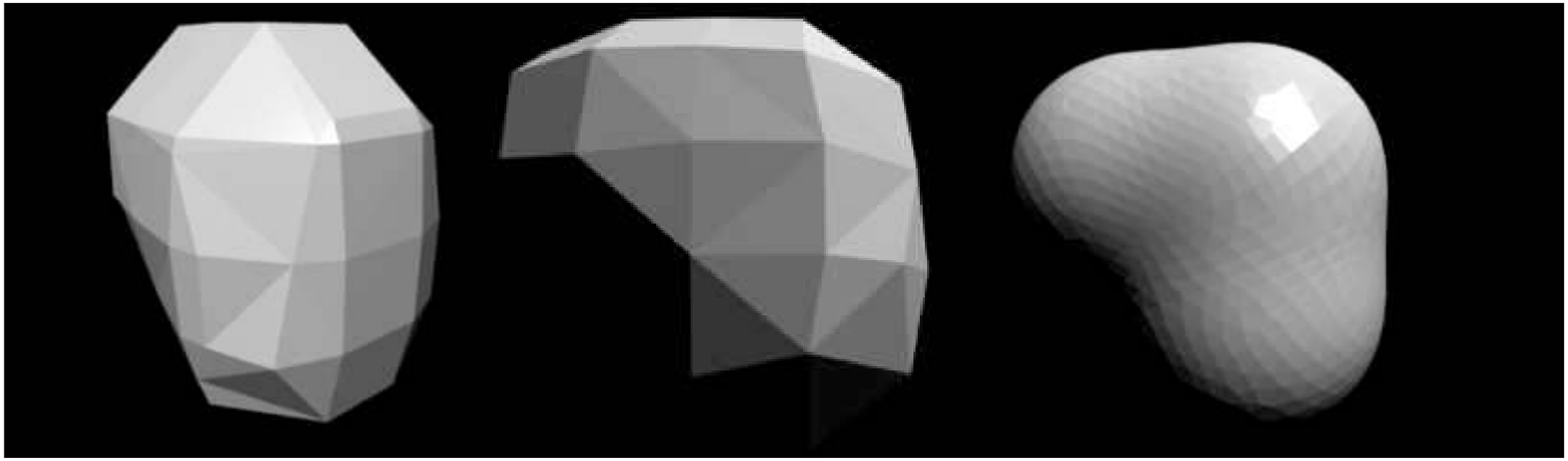
- calculate function value v and normal $\vec{n}$ at point O
- correct position of O by $\vec{n}\frac{v+|n|x}{v|n|}$

Results:

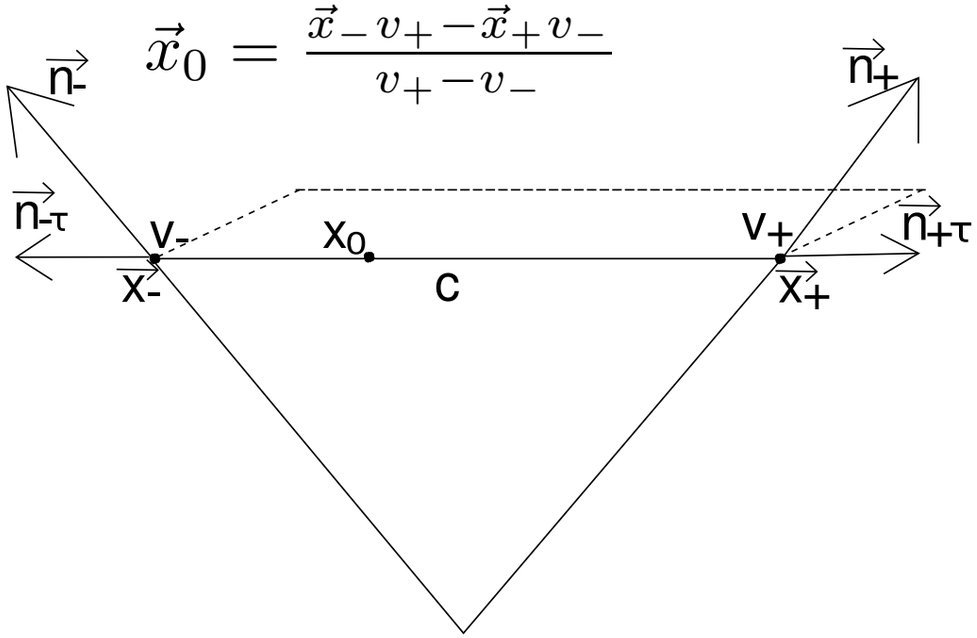
- One prediction-correction procedure requires only a single function evaluation and reduces the error by $\approx 10^2$ times

Mesh Refinement

Original mesh, mesh after decimation and the refined mesh:

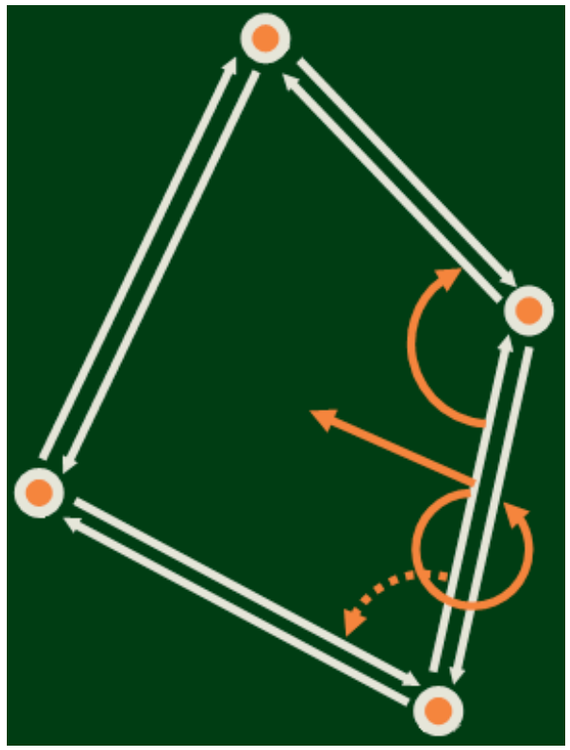


Original Marching cube algorithm uses linear point interpolation:



Data Structure

Open Mesh library:



- one vertex
- one face
- the next halfedge
- the opposite halfedge
- optional: the previous halfedge

- fast iterators over faces, vertexes, etc.
- fast circulators over faces, vertexes, etc.

Fast Electrostatic Summation

N-body solvers:

- direct $O(N^2)$
- multipole $O(N) - O(N \log N)$
  - analytical derivatives do not exist at atom positions
  - forces fluctuate at low number of expansion terms
  - it is possible to achieve a given accuracy with certain number of expansion terms
- multigrid $O(N) - O(N \log N)$
  - at low accuracy energy conserves better
  - there is a limit of accuracy
- Ewald sums -$O(N \log N)$
  - easy and robust
  - classical schemes exist only for periodic systems

Multipole Summation

- DPMTA algorithm has been extended to calculate the transpose of matrix $\mathbf{A}$
- The multipole expansion $\mathbf{M}^{\mathrm{T}}_{\mathrm{n,m}}$ has a vector form and contains three new components:

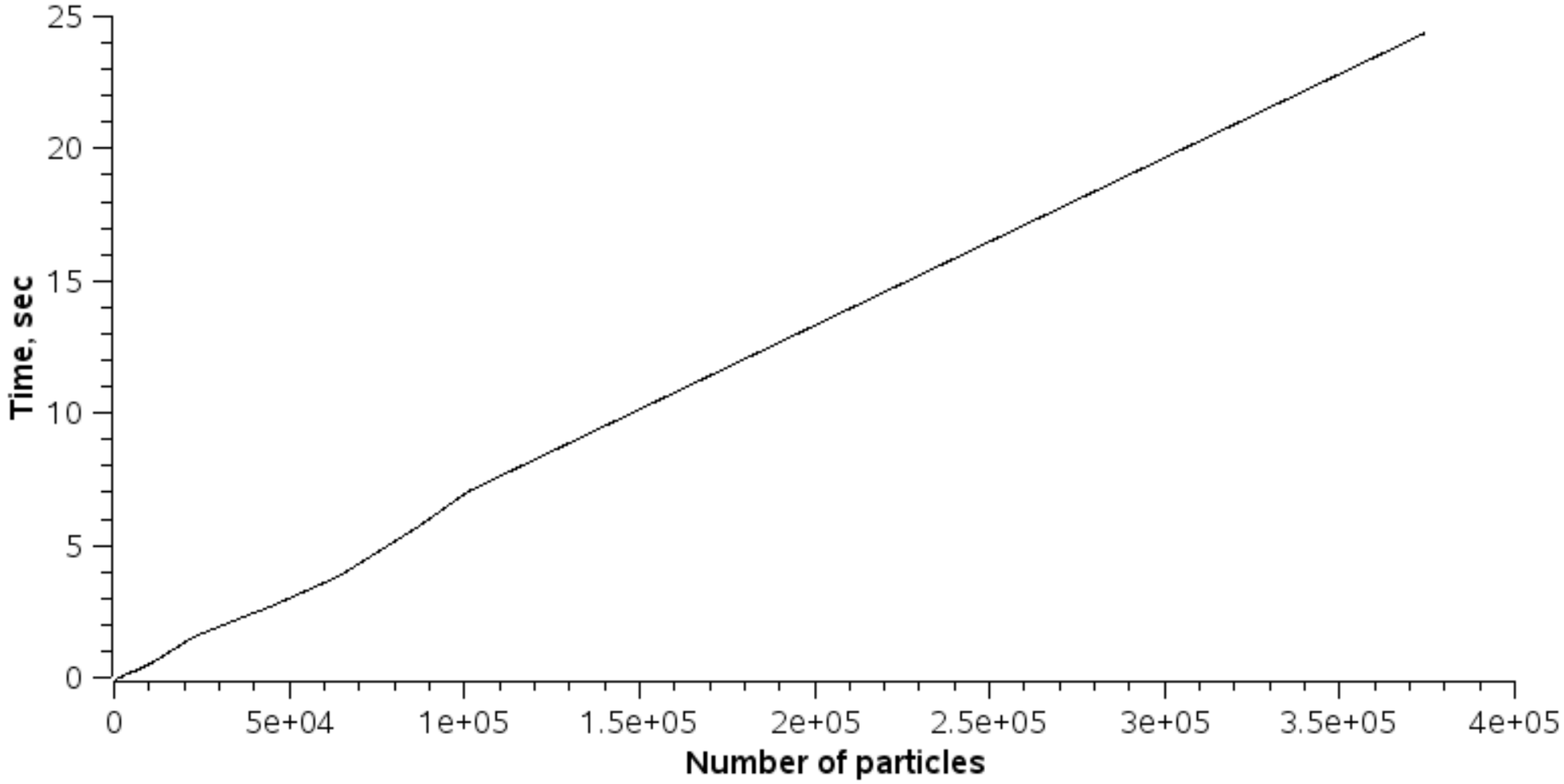
$$\vec{\mathbf{M}}^{T}_{n,m}(\vec{r}) = \sum_{i=1}^{k} q_i s_i \vec{n}_i \mathbf{F}^{*}_{n,m}(\vec{r}_i) \tag{1}$$

- Forces then calculated as:

$$q_i \nabla \Phi^{T}(\vec{r}) = Tr\left[q_i \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \vec{\mathbf{M}}^{T}_{n,m} \nabla \mathbf{G}_{n,m}(\vec{r}) \right] \tag{2}$$

Multipole Summation

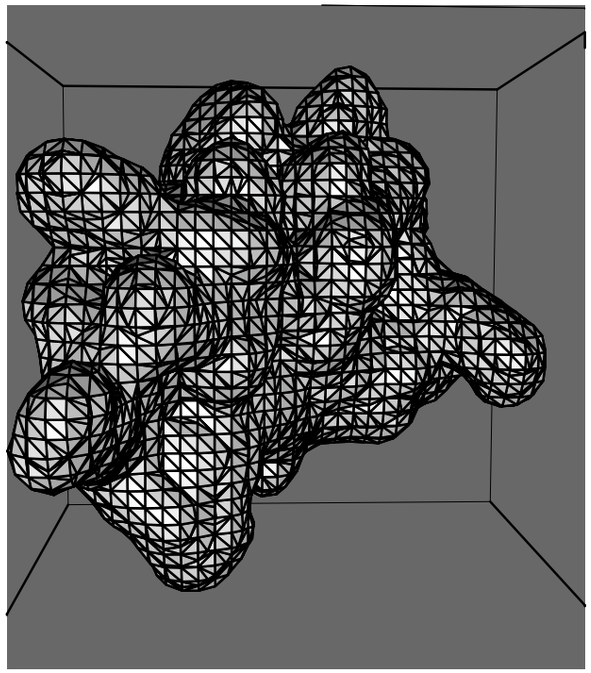
- DPMTA scales linearly:

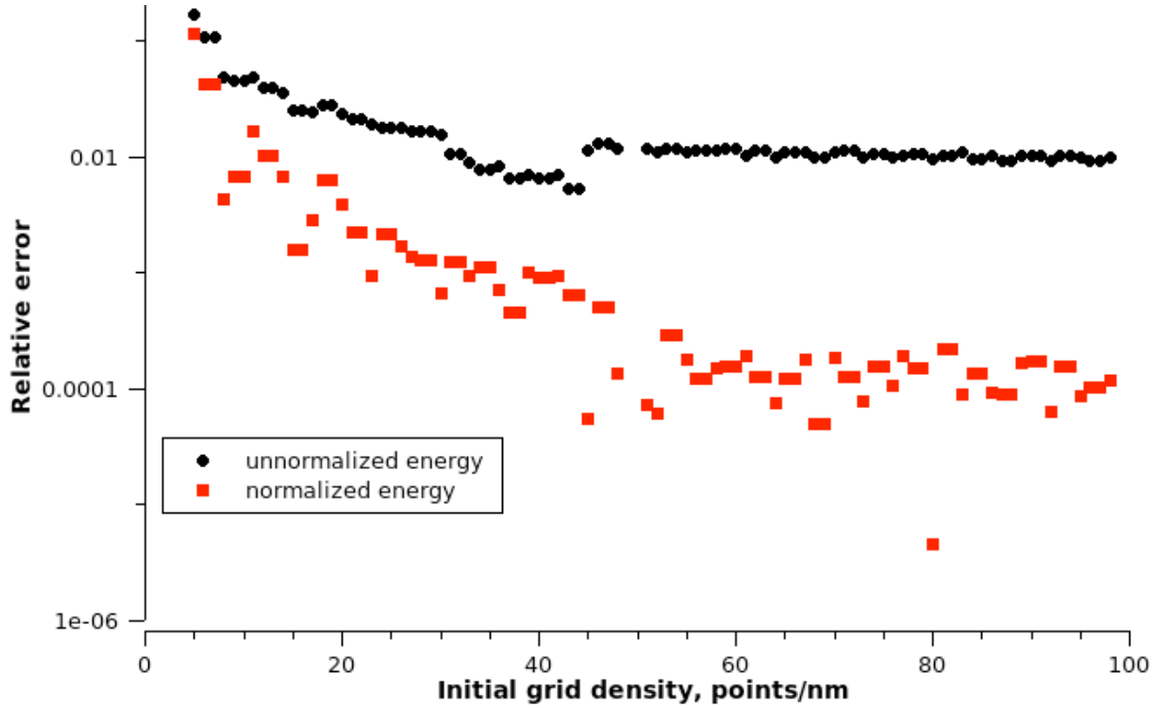


Time for calculation of one cycle of the BEM algorithm as a function of number of the boundary particles

- it is faster than the direct solver only at high $\approx 10^4$ number of particles

Numerical Tests

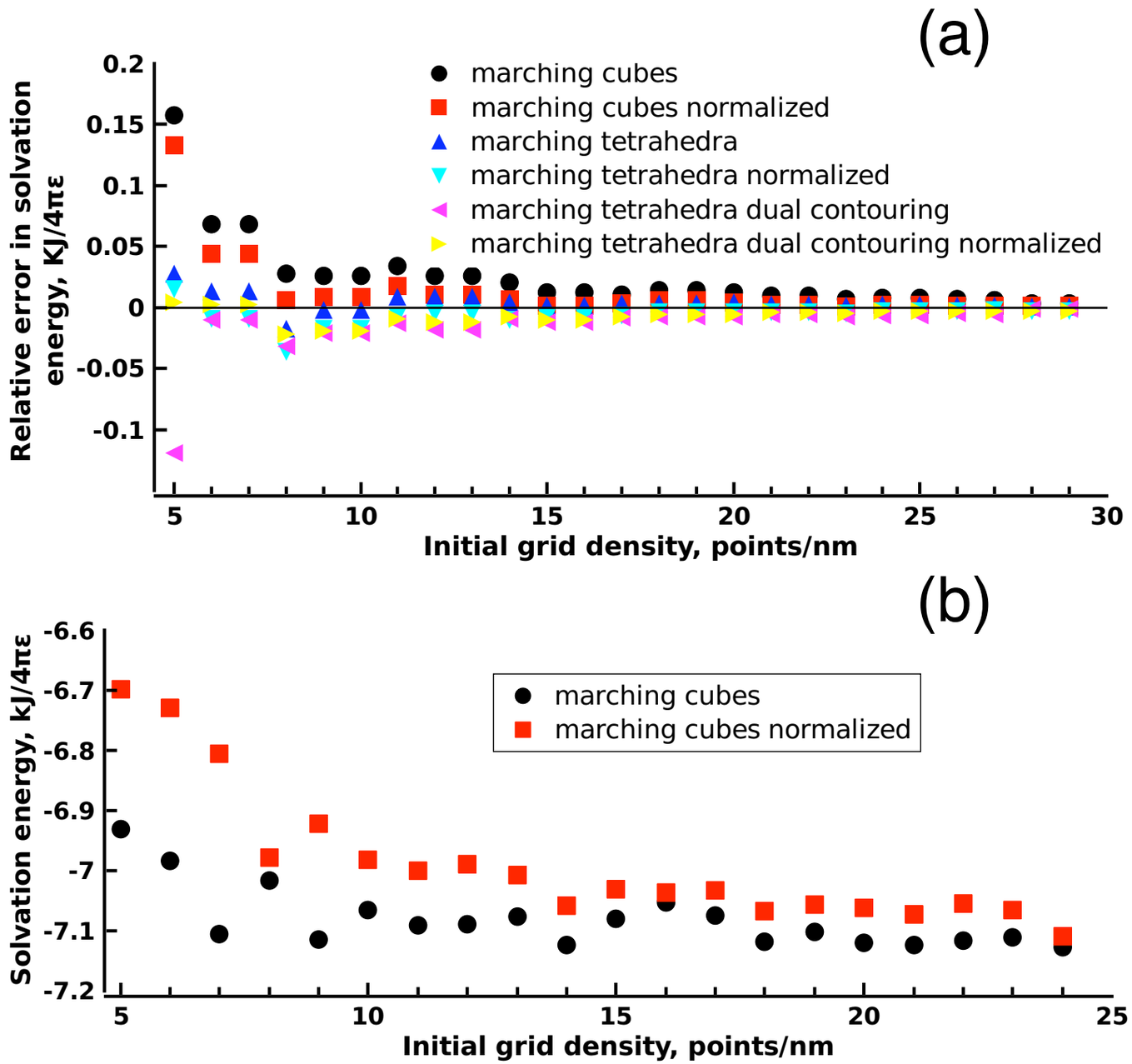




(a) (b)

(a) Molecular surface of the BPTI protein. Marching cubes algorithm for the surface extraction has been used. (b) Error in solvation energy versus density of surface point for the marching cubes algorithm.

Numerical Tests



(a) Three different marching schemes are compared. (b) Solvation energy for the BPTI protein at different grid point densities.