Monte Carlo Methods for Partial Differential Equations: A Personal Journey

Prof. Michael Mascagni

Department of Computer Science
Department of Mathematics
Department of Scientific Computing
Graduate Program in Molecular Biophysics
Florida State University, Tallahassee, FL 32306 **USA**

E-mail: mascagni@fsu.edu or mascagni@math.ethz.ch
URL: http://www.cs.fsu.edu/∼mascagni

In collaboration with Drs. Marcia O. Fenley, Nikolai Simonov and Alexander Silalahi, Robert Harris, and Messrs. Travis Mackoy, and James McClain

Research supported by ARO, DOE, NATO, and NSF

Introduction

Introduction

Introduction

Introduction

Introduction

Introduction

Introduction

Early History of MCMs for PDEs

1. Courant, Friedrichs, and Lewy: Their pivotal 1928 paper has probabilistic interpretations and MC algorithms for linear elliptic and parabolic problems
2. Fermi/Ulam/von Neumann: Atomic bomb calculations were done using Monte Carlo methods for neutron transport, their success inspired much post-War work especially in nuclear reactor design
3. Kac and Donsker: Used large deviation calculations to estimate eigenvalues of a linear Schrödinger equation
4. Forsythe and Leibler: Derived a MCM for solving special linear systems related to discrete elliptic PDE problems

Early History of MCMs for PDEs

1. Courant, Friedrichs, and Lewy: Their pivotal 1928 paper has probabilistic interpretations and MC algorithms for linear elliptic and parabolic problems
2. Fermi/Ulam/von Neumann: Atomic bomb calculations were done using Monte Carlo methods for neutron transport, their success inspired much post-War work especially in nuclear reactor design
3. Kac and Donsker: Used large deviation calculations to estimate eigenvalues of a linear Schrödinger equation
4. Forsythe and Leibler: Derived a MCM for solving special linear systems related to discrete elliptic PDE problems

Early History of MCMs for PDEs

1. Courant, Friedrichs, and Lewy: Their pivotal 1928 paper has probabilistic interpretations and MC algorithms for linear elliptic and parabolic problems
2. Fermi/Ulam/von Neumann: Atomic bomb calculations were done using Monte Carlo methods for neutron transport, their success inspired much post-War work especially in nuclear reactor design
3. Kac and Donsker: Used large deviation calculations to estimate eigenvalues of a linear Schrödinger equation
4. Forsythe and Leibler: Derived a MCM for solving special linear systems related to discrete elliptic PDE problems

Early History of MCMs for PDEs

1. Courant, Friedrichs, and Lewy: Their pivotal 1928 paper has probabilistic interpretations and MC algorithms for linear elliptic and parabolic problems
2. Fermi/Ulam/von Neumann: Atomic bomb calculations were done using Monte Carlo methods for neutron transport, their success inspired much post-War work especially in nuclear reactor design
3. Kac and Donsker: Used large deviation calculations to estimate eigenvalues of a linear Schrödinger equation
4. Forsythe and Leibler: Derived a MCM for solving special linear systems related to discrete elliptic PDE problems

The First Passage (FP) Probability is the Green's Function

Back to our canonical elliptic boundary value problem:

$$\frac{1}{2}\Delta u(x) = 0, \quad x \in \Omega$$
$$u(x) = f(x), \quad x \in \partial\Omega$$

- Distribution of z is uniform on the sphere
- Mean of the values of $u(z)$ over the sphere is $u(x)$
- $u(x)$ has mean-value property and harmonic
- Also, $u(x)$ satisfies the boundary condition

$$u(x) = \mathbb{E}_x[f(X^{\mathbf{x}}(t_{\partial\Omega}))] \tag{1}$$

The First Passage (FP) Probability is the Green's Function

Back to our canonical elliptic boundary value problem:

$$\frac{1}{2}\Delta u(x) = 0, \quad x \in \Omega$$
$$u(x) = f(x), \quad x \in \partial\Omega$$

- Distribution of z is uniform on the sphere
- Mean of the values of $u(z)$ over the sphere is $u(x)$
- $u(x)$ has mean-value property and harmonic
- Also, $u(x)$ satisfies the boundary condition

$$u(x) = \mathbb{E}_x[f(X^{\mathbf{x}}(t_{\partial\Omega}))] \tag{1}$$

The First Passage (FP) Probability is the Green's Function

Back to our canonical elliptic boundary value problem:

$$\frac{1}{2}\Delta u(x) = 0, \quad x \in \Omega$$
$$u(x) = f(x), \quad x \in \partial\Omega$$

- Distribution of z is uniform on the sphere
- Mean of the values of $u(z)$ over the sphere is $u(x)$
- $u(x)$ has mean-value property and harmonic
- Also, $u(x)$ satisfies the boundary condition

$$u(x) = \mathbb{E}_x[f(X^{\mathbf{x}}(t_{\partial\Omega}))] \tag{1}$$

The First Passage (FP) Probability is the Green's Function

Back to our canonical elliptic boundary value problem:

$$\frac{1}{2}\Delta u(x) = 0, \quad x \in \Omega$$
$$u(x) = f(x), \quad x \in \partial\Omega$$

- Distribution of z is uniform on the sphere
- Mean of the values of $u(z)$ over the sphere is $u(x)$
- $u(x)$ has mean-value property and harmonic
- Also, $u(x)$ satisfies the boundary condition

$$u(x) = \mathbb{E}_x[f(X^{\mathbf{x}}(t_{\partial\Omega}))] \tag{1}$$

The First Passage (FP) Probability is the Green's Function

Reinterpreting as an average of the boundary values

$$u(x) = \int_{\partial\Omega} p(x,y)\, f(y)\, dy \tag{2}$$

Another representation in terms of an integral over the boundary

$$u(x) = \int_{\partial\Omega} \frac{\partial g(x,y)}{\partial \mathbf{n}}\, f(y)\, dy \tag{3}$$

$g(x,y)$ – Green's function of the Dirichlet problem in Ω

$$\Longrightarrow p(x,y) = \frac{\partial g(x,y)}{\partial \mathbf{n}} \tag{4}$$

'Walk on Spheres' (WOS) and 'Green's Function First Passage' (GFFP) Algorithms

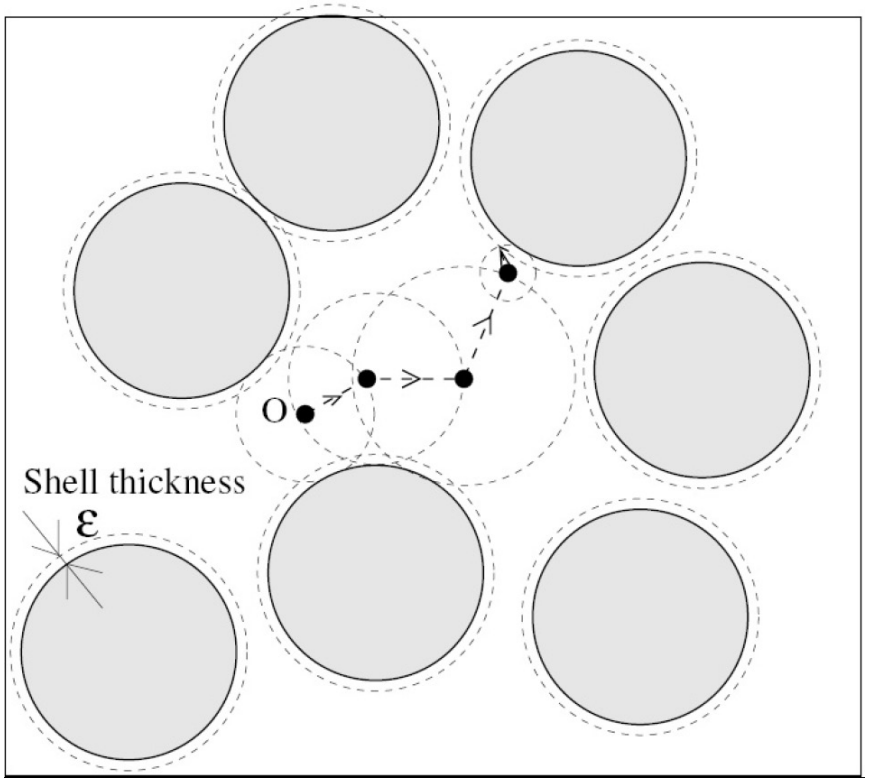
- Green's function is known
  $\Longrightarrow$ direct simulation of exit points and computation of the solution through averaging boundary values
- Green's function is unknown
  $\Longrightarrow$ simulation of exit points from standard subdomains of Ω, e.g. spheres
  $\Longrightarrow$ Markov chain of 'Walk on Spheres' (or GFFP algorithm)
  $x_0 = x, x_1, \ldots, x_N$
  $x_i \to \partial\Omega$ and hits ε-shell is $N = O(|\ln(\varepsilon)|)$ steps
  x_N simulates exit point from Ω with $O(\varepsilon)$ accuracy

'Walk on Spheres' (WOS) and 'Green's Function First Passage' (GFFP) Algorithms

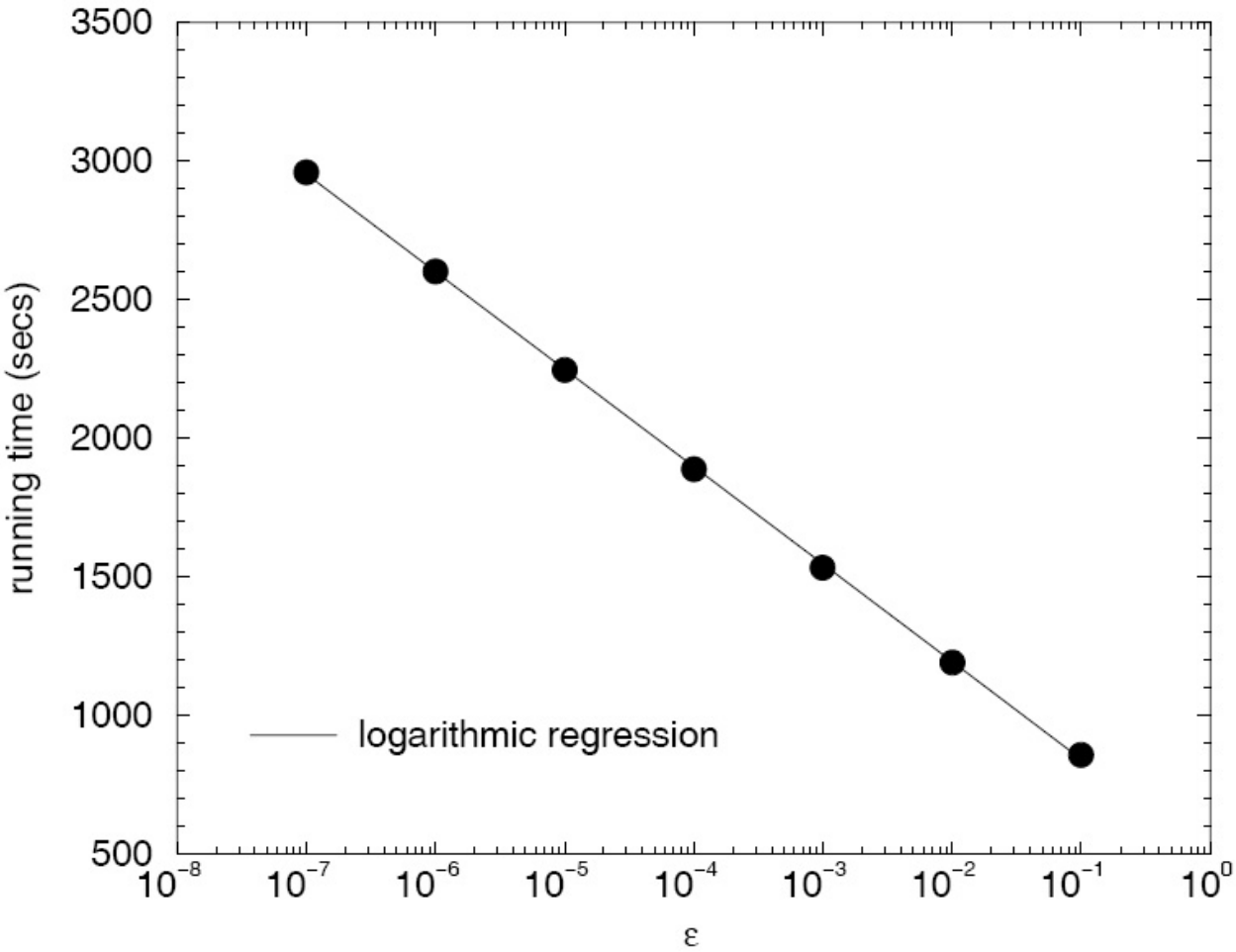
- Green's function is known
  $\Longrightarrow$ direct simulation of exit points and computation of the solution through averaging boundary values
- Green's function is unknown
  $\Longrightarrow$ simulation of exit points from standard subdomains of Ω, e.g. spheres
  $\Longrightarrow$ Markov chain of 'Walk on Spheres' (or GFFP algorithm)
  $x_0 = x, x_1, \ldots, x_N$
  $x_i \to \partial\Omega$ and hits ε-shell is $N = O(|\ln(\varepsilon)|)$ steps
  x_N simulates exit point from Ω with $O(\varepsilon)$ accuracy

'Walk on Spheres' (WOS) and 'Green's Function First Passage' (GFFP) Algorithms

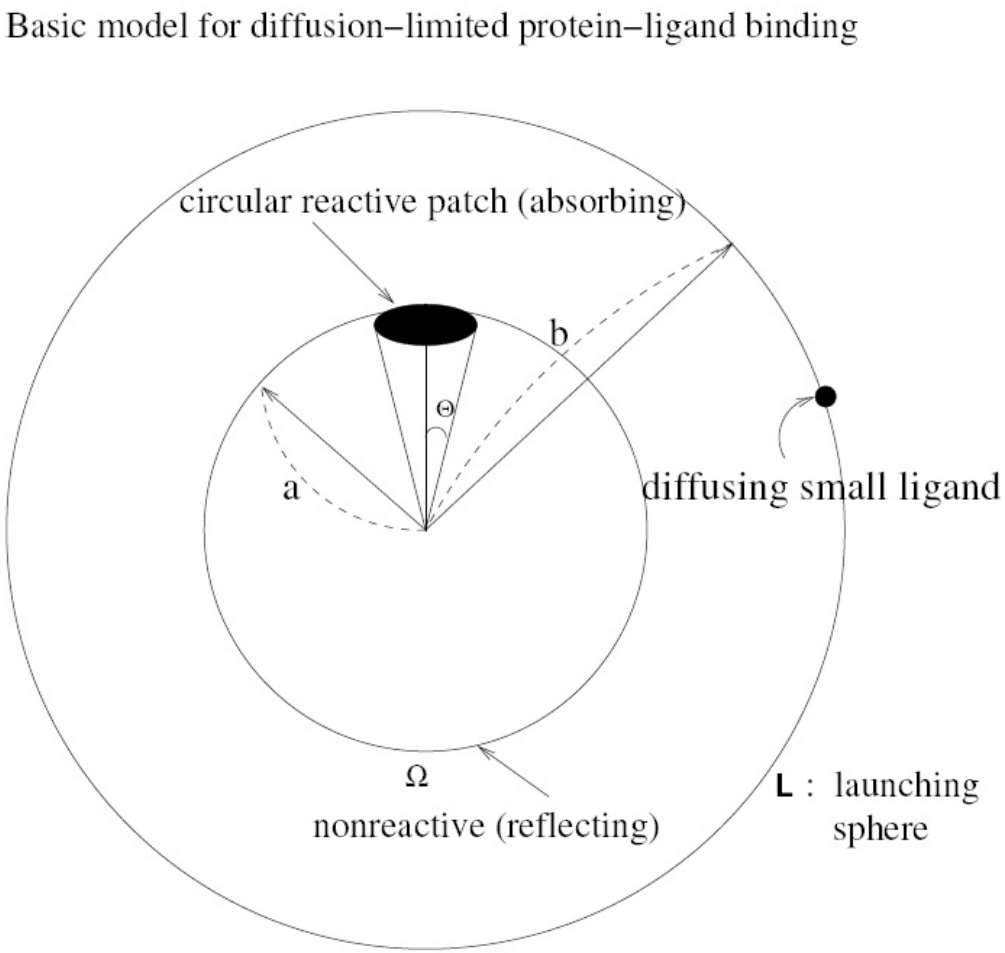
WOS:



Timing with WOS

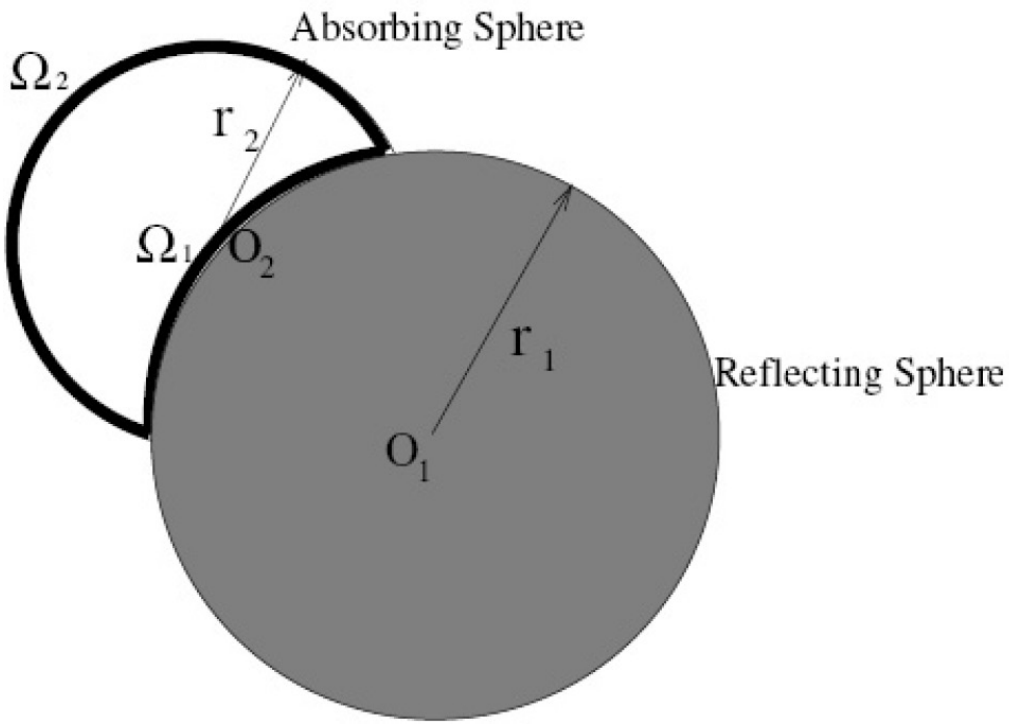


Solc-Stockmayer Model without Potential

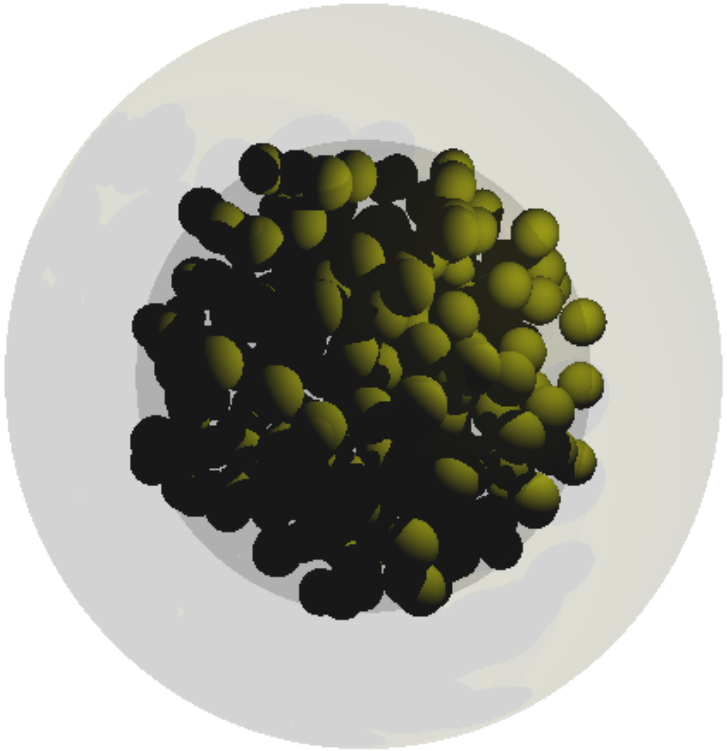


The Simulation-Tabulation (S-T) Method for Generalization

- Green's function for the non-intersected surface of a sphere located on the surface of a reflecting sphere



Porous Media: Complicated Interfaces



Computing Capacitance Probabilistically

- Hubbard-Douglas: can compute permeability of nonskew object via capacitance
- Recall that $C = \frac{Q}{u}$, if we hold conductor (Ω)at unit potential $u = 1$, then $C =$ total charge on conductor (surface)
- The PDE system for the potential is

$$\Delta u = 0, \quad x \notin \Omega; \quad u = 1, \quad x \in \partial\Omega; \quad u \to 0 \ \text{as} \ x \to \infty \tag{5}$$

- Recall $u(\mathbf{x}) = \mathbb{E}_x[f(X^{\mathbf{x}}(t_{\partial\Omega}))] =$ probability of walker starting at $\mathbf{x}$ hitting Ω before escaping to infinity
- Charge density is first passage probability
- Capacitance (relative to a sphere) is probability of walker starting at $\mathbf{x}$ (random chosen on sphere) hitting Ω before escaping to infinity

Computing Capacitance Probabilistically

- Hubbard-Douglas: can compute permeability of nonskew object via capacitance
- Recall that $C = \frac{Q}{u}$, if we hold conductor (Ω)at unit potential $u = 1$, then $C =$ total charge on conductor (surface)
- The PDE system for the potential is

$$\Delta u = 0, \quad x \notin \Omega; \quad u = 1, \quad x \in \partial\Omega; \quad u \to 0 \ \text{as} \ x \to \infty \tag{5}$$

- Recall $u(\mathbf{x}) = \mathbb{E}_x[f(X^{\mathbf{x}}(t_{\partial\Omega}))] =$ probability of walker starting at $\mathbf{x}$ hitting Ω before escaping to infinity
- Charge density is first passage probability
- Capacitance (relative to a sphere) is probability of walker starting at $\mathbf{x}$ (random chosen on sphere) hitting Ω before escaping to infinity

Computing Capacitance Probabilistically

- Hubbard-Douglas: can compute permeability of nonskew object via capacitance
- Recall that $C = \frac{Q}{u}$, if we hold conductor (Ω)at unit potential $u = 1$, then $C =$ total charge on conductor (surface)
- The PDE system for the potential is

$$\Delta u = 0, \quad x \notin \Omega; \quad u = 1, \quad x \in \partial\Omega; \quad u \to 0 \ \text{as} \ x \to \infty \tag{5}$$

- Recall $u(\mathbf{x}) = \mathbb{E}_x[f(X^{\mathbf{x}}(t_{\partial\Omega}))] =$ probability of walker starting at $\mathbf{x}$ hitting Ω before escaping to infinity
- Charge density is first passage probability
- Capacitance (relative to a sphere) is probability of walker starting at $\mathbf{x}$ (random chosen on sphere) hitting Ω before escaping to infinity

Computing Capacitance Probabilistically

- Hubbard-Douglas: can compute permeability of nonskew object via capacitance
- Recall that $C = \frac{Q}{u}$, if we hold conductor (Ω)at unit potential $u = 1$, then $C =$ total charge on conductor (surface)
- The PDE system for the potential is

$$\Delta u = 0, \quad x \notin \Omega; \quad u = 1, \quad x \in \partial\Omega; \quad u \to 0 \ \text{as} \ x \to \infty \tag{5}$$

- Recall $u(\mathbf{x}) = \mathbb{E}_x[f(X^{\mathbf{x}}(t_{\partial\Omega}))] =$ probability of walker starting at $\mathbf{x}$ hitting Ω before escaping to infinity
- Charge density is first passage probability
- Capacitance (relative to a sphere) is probability of walker starting at $\mathbf{x}$ (random chosen on sphere) hitting Ω before escaping to infinity

Computing Capacitance Probabilistically

- Hubbard-Douglas: can compute permeability of nonskew object via capacitance
- Recall that $C = \frac{Q}{u}$, if we hold conductor (Ω)at unit potential $u = 1$, then $C =$ total charge on conductor (surface)
- The PDE system for the potential is

$$\Delta u = 0, \quad x \notin \Omega; \quad u = 1, \quad x \in \partial\Omega; \quad u \to 0 \ \text{as} \ x \to \infty \tag{5}$$

- Recall $u(\mathbf{x}) = \mathbb{E}_x[f(X^{\mathbf{x}}(t_{\partial\Omega}))] =$ probability of walker starting at $\mathbf{x}$ hitting Ω before escaping to infinity
- Charge density is first passage probability
- Capacitance (relative to a sphere) is probability of walker starting at $\mathbf{x}$ (random chosen on sphere) hitting Ω before escaping to infinity

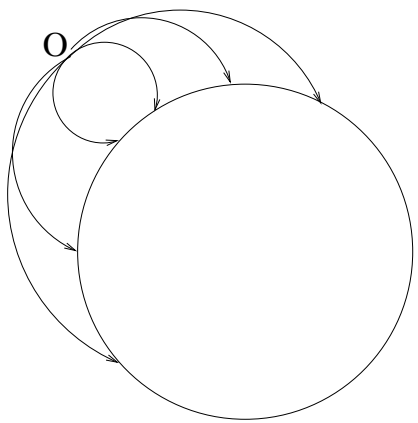
Computing Capacitance Probabilistically

- Hubbard-Douglas: can compute permeability of nonskew object via capacitance
- Recall that $C = \frac{Q}{u}$, if we hold conductor (Ω)at unit potential $u = 1$, then $C =$ total charge on conductor (surface)
- The PDE system for the potential is

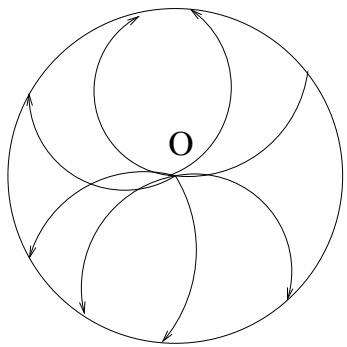
$$\Delta u = 0, \quad x \notin \Omega; \quad u = 1, \quad x \in \partial\Omega; \quad u \to 0 \ \text{as} \ x \to \infty \tag{5}$$

- Recall $u(\mathbf{x}) = \mathbb{E}_x[f(X^{\mathbf{x}}(t_{\partial\Omega}))] =$ probability of walker starting at $\mathbf{x}$ hitting Ω before escaping to infinity
- Charge density is first passage probability
- Capacitance (relative to a sphere) is probability of walker starting at $\mathbf{x}$ (random chosen on sphere) hitting Ω before escaping to infinity

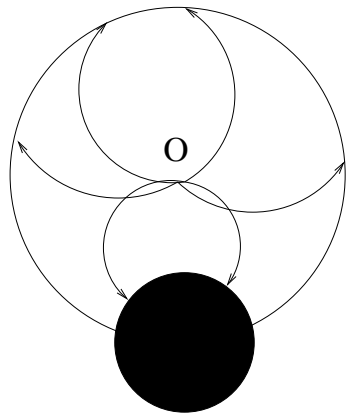
Various Laplacian Green's Functions for Green's Function First Passage (GFFP)



(a) Putting back



(b) Void space



(c) Intersecting

Escape to ∞ in A Single Step

- Probability that a diffusing particle at $r_0 > b$ will escape to infinity

$$P_{esc} = 1 - \frac{b}{r_0} = 1 - \alpha \tag{6}$$

- Putting-back distribution density function

$$\omega(\theta, \phi) = \frac{1 - \alpha^2}{4\pi[1 - 2\alpha\cos\theta + \alpha^2]^{3/2}} \tag{7}$$

  - (b, θ, ϕ) ; spherical coordinates of the new position when the old position is put on the polar axis

Escape to ∞ in A Single Step

- Probability that a diffusing particle at $r_0 > b$ will escape to infinity

$$P_{esc} = 1 - \frac{b}{r_0} = 1 - \alpha \tag{6}$$

- Putting-back distribution density function

$$\omega(\theta, \phi) = \frac{1 - \alpha^2}{4\pi[1 - 2\alpha\cos\theta + \alpha^2]^{3/2}} \tag{7}$$

  - (b, θ, ϕ) ; spherical coordinates of the new position when the old position is put on the polar axis

Escape to ∞ in A Single Step

- Probability that a diffusing particle at $r_0 > b$ will escape to infinity

$$P_{esc} = 1 - \frac{b}{r_0} = 1 - \alpha \tag{6}$$

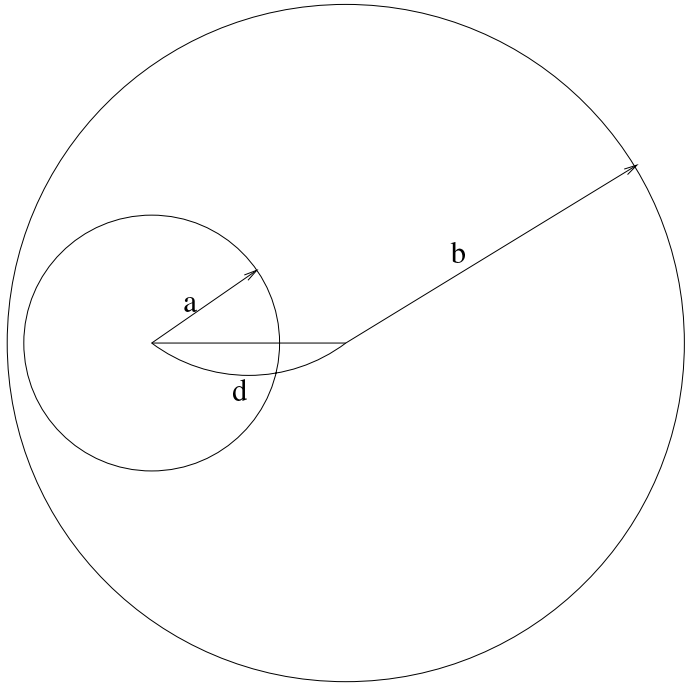
- Putting-back distribution density function

$$\omega(\theta, \phi) = \frac{1 - \alpha^2}{4\pi[1 - 2\alpha\cos\theta + \alpha^2]^{3/2}} \tag{7}$$

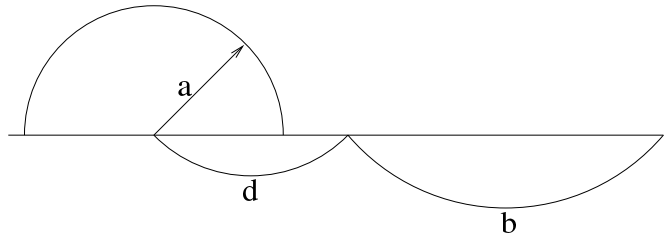
  - (b, θ, ϕ) ; spherical coordinates of the new position when the old position is put on the polar axis

Charge Density on a Circular Disk via Last-Passage

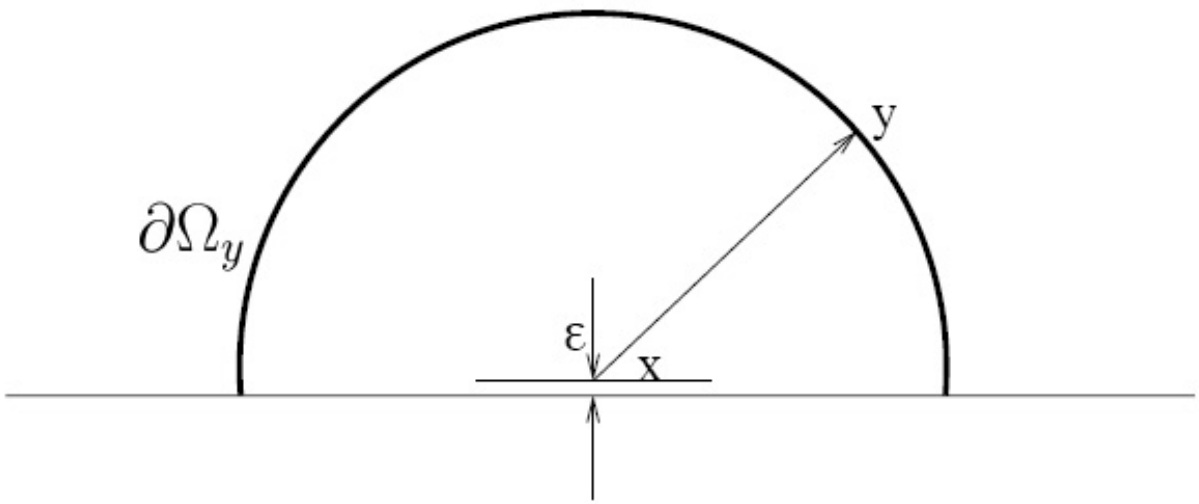
From the top



From the side



Time Reversal Brownian Motion: Approach from the Outside



Approach from the Outside

- $P(x)$: prob. of diffusing from ϵ above lower FP surface to ∞

$$P(x) = \int_{\partial\Omega_y} g(x,y,\epsilon)p(y,\infty)dS \tag{8}$$

$$\sigma(x) = -\frac{1}{4\pi}\frac{d}{d\epsilon}\bigg|_{\epsilon=0}\phi(x) = \frac{1}{4\pi}\frac{d}{d\epsilon}\bigg|_{\epsilon=0}P(x) \tag{9}$$

$$\sigma(x) = \frac{1}{4\pi}\int_{\partial\Omega_y} G(x,y)p(y,\infty)dS \tag{10}$$

where

$$G(x,y) = \frac{d}{d\epsilon}\bigg|_{\epsilon=0} g(x,y,\epsilon) \tag{11}$$

- $G(x,y)$ satisfies a point dipole problem

Approach from the Outside

- $P(x)$: prob. of diffusing from ϵ above lower FP surface to ∞

$$P(x) = \int_{\partial\Omega_y} g(x,y,\epsilon)p(y,\infty)dS \tag{8}$$

$$\sigma(x) = -\frac{1}{4\pi}\frac{d}{d\epsilon}\bigg|_{\epsilon=0}\phi(x) = \frac{1}{4\pi}\frac{d}{d\epsilon}\bigg|_{\epsilon=0}P(x) \tag{9}$$

$$\sigma(x) = \frac{1}{4\pi}\int_{\partial\Omega_y} G(x,y)p(y,\infty)dS \tag{10}$$

where

$$G(x,y) = \frac{d}{d\epsilon}\bigg|_{\epsilon=0} g(x,y,\epsilon) \tag{11}$$

- $G(x,y)$ satisfies a point dipole problem

Charge Density on the Circular Disk

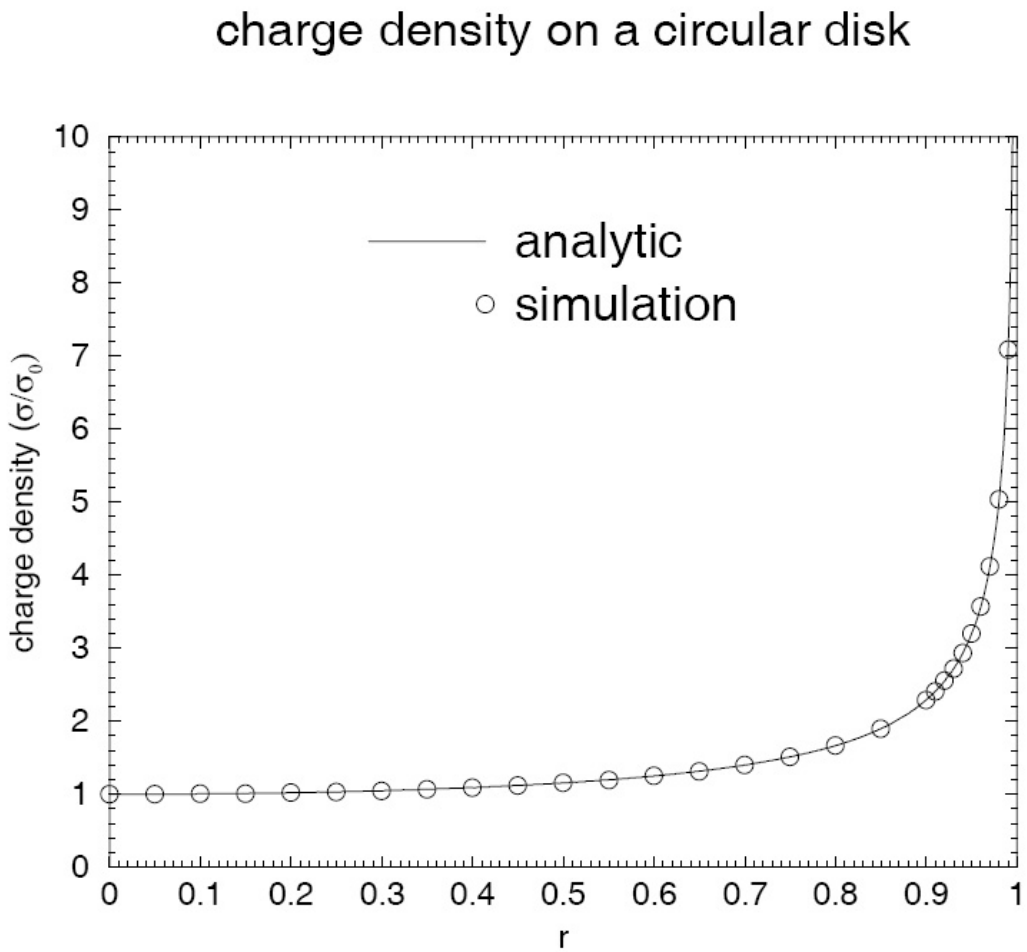
$$G = \frac{3}{4}\frac{\cos\theta}{a^3} \tag{12}$$

$$\sigma(x) = \frac{3}{16\pi}\int_{\partial\Omega_{\mathbf{r}}} \frac{\cos\theta}{a^3} p(\mathbf{r},\infty)\, dS \tag{13}$$

where

$$p(\mathbf{r},\infty) = 1 - \frac{2}{\pi}\arctan\left(\frac{\sqrt{2}b}{\sqrt{r^2 - b^2 + \sqrt{(r^2-b^2)^2 + 4b^2x^2}}}\right) \tag{14}$$

Charge Density on the Circular Disk



Edge Distribution on the Circular Disk

$$\sigma(r) = \frac{1}{4\pi}\frac{1}{\sqrt{1-r^2}} \tag{15}$$

Let $r = 1 - x$:

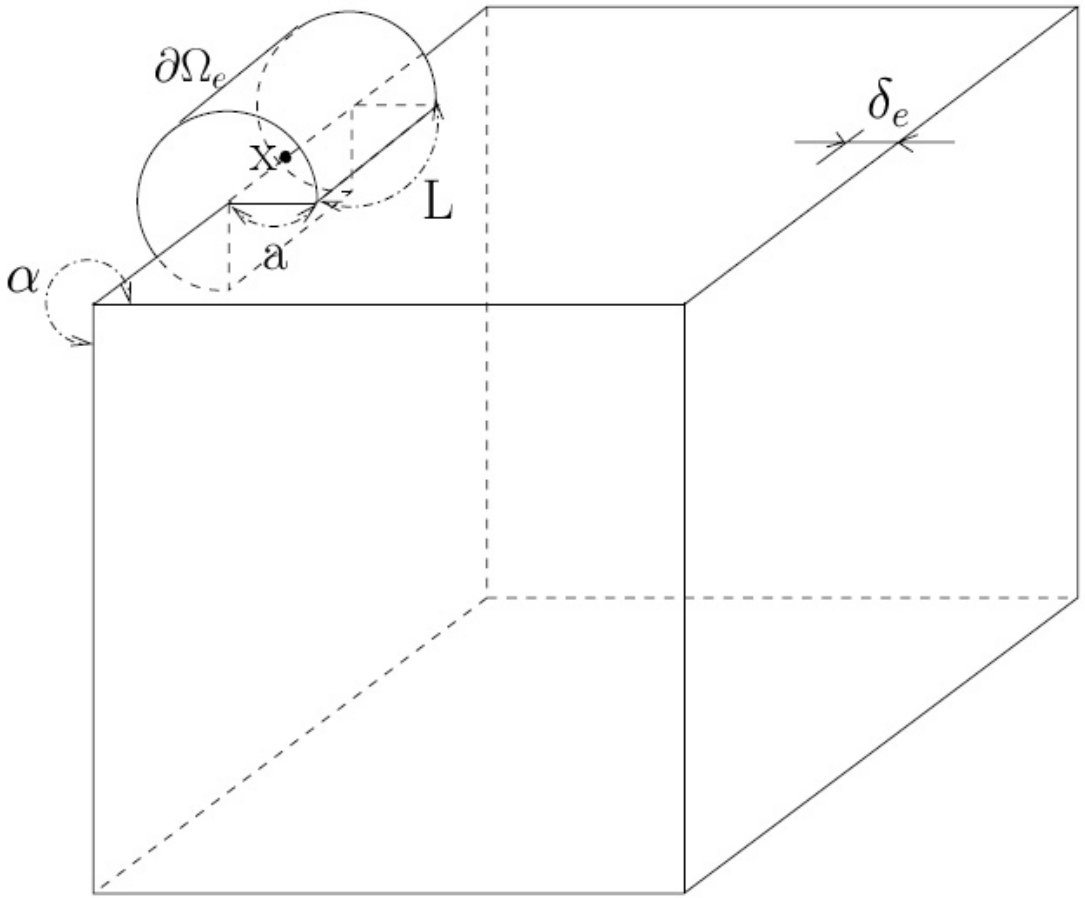
$$\sigma(x) = \frac{1}{4\pi}\frac{1}{\sqrt{2x}}(1 - x/2)^{-1/2} \tag{16}$$

when x is small enough,

$$\sigma(x) \simeq \frac{1}{4\sqrt{2}\pi}\frac{1}{\sqrt{x}} \tag{17}$$

$$\sigma(x) \simeq \sigma_e \frac{1}{\sqrt{x}} \tag{18}$$

Unit Cube Edge Distribution



Unit Cube Edge Distribution

$$\sigma(x, \delta_e) = \delta_e^{\pi/\alpha - 1} \sigma_e(x) \tag{19}$$

- $\sigma(x, \delta_e)$: charge on a curve parallel to the edge separated by δ_e
- $\sigma_e(x)$: edge distribution
- α: angle between the two intersecting surfaces, here $\alpha = 3\pi/2$

$$\sigma_e(x) = \frac{1}{4\pi} \lim_{\delta_e \to 0} \delta_e^{1-\pi/\alpha} \int_{\partial\Omega_e} G(x, y) p(y, \infty) dS \tag{20}$$

- $\partial\Omega_e$: cylindrical surface that intersects the pair of absorbing surfaces meeting at angle α

Unit Cube Edge Distribution

$$\sigma(x, \delta_e) = \delta_e^{\pi/\alpha - 1} \sigma_e(x) \tag{19}$$

- $\sigma(x, \delta_e)$: charge on a curve parallel to the edge separated by δ_e
- $\sigma_e(x)$: edge distribution
- α: angle between the two intersecting surfaces, here $\alpha = 3\pi/2$

$$\sigma_e(x) = \frac{1}{4\pi} \lim_{\delta_e \to 0} \delta_e^{1-\pi/\alpha} \int_{\partial\Omega_e} G(x, y) p(y, \infty) dS \tag{20}$$

- $\partial\Omega_e$: cylindrical surface that intersects the pair of absorbing surfaces meeting at angle α

Unit Cube Edge Distribution

$$\sigma(x, \delta_e) = \delta_e^{\pi/\alpha - 1} \sigma_e(x) \tag{19}$$

- $\sigma(x, \delta_e)$: charge on a curve parallel to the edge separated by δ_e
- $\sigma_e(x)$: edge distribution
- α: angle between the two intersecting surfaces, here $\alpha = 3\pi/2$

$$\sigma_e(x) = \frac{1}{4\pi} \lim_{\delta_e \to 0} \delta_e^{1-\pi/\alpha} \int_{\partial\Omega_e} G(x, y) p(y, \infty) dS \tag{20}$$

- $\partial\Omega_e$: cylindrical surface that intersects the pair of absorbing surfaces meeting at angle α

Unit Cube Edge Distribution

$$\sigma(x, \delta_e) = \delta_e^{\pi/\alpha - 1} \sigma_e(x) \tag{19}$$

- $\sigma(x, \delta_e)$: charge on a curve parallel to the edge separated by δ_e
- $\sigma_e(x)$: edge distribution
- α: angle between the two intersecting surfaces, here $\alpha = 3\pi/2$

$$\sigma_e(x) = \frac{1}{4\pi} \lim_{\delta_e \to 0} \delta_e^{1-\pi/\alpha} \int_{\partial\Omega_e} G(x, y) p(y, \infty) dS \tag{20}$$

- $\partial\Omega_e$: cylindrical surface that intersects the pair of absorbing surfaces meeting at angle α

Unit Cube Edge Distribution

- $G(x, y)$:

$$G(x, y) = \left. \frac{d}{d\delta_\epsilon} \right|_{\delta_\epsilon = 0} g(x, y, \delta_\epsilon) \tag{21}$$

  - $g(x, y, \delta_\epsilon)$: Laplace Green's function on the surface, $\partial\Omega_e$, with source point x at a distance δ_ϵ from the absorbing surface
- $p(y, \infty)$: probability that a diffusing particle, initiated at point $y \in \partial\Omega_e$, diffuses to infinity without returning to the absorbing surface

Unit Cube Edge Distribution

- $G(x, y)$:

$$G(x, y) = \left. \frac{d}{d\delta_\epsilon} \right|_{\delta_\epsilon = 0} g(x, y, \delta_\epsilon) \tag{21}$$

  - $g(x, y, \delta_\epsilon)$: Laplace Green's function on the surface, $\partial\Omega_e$, with source point x at a distance δ_ϵ from the absorbing surface
- $p(y, \infty)$: probability that a diffusing particle, initiated at point $y \in \partial\Omega_e$, diffuses to infinity without returning to the absorbing surface

Unit Cube Edge Distribution

- $G(x, y)$:

$$G(x, y) = \frac{d}{d\delta_\epsilon}\bigg|_{\delta_\epsilon=0} g(x, y, \delta_\epsilon) \tag{21}$$

  - $g(x, y, \delta_\epsilon)$: Laplace Green's function on the surface, $\partial\Omega_e$, with source point x at a distance δ_ϵ from the absorbing surface
- $p(y, \infty)$: probability that a diffusing particle, initiated at point $y \in \partial\Omega_e$, diffuses to infinity without returning to the absorbing surface

Unit Cube Edge Distribution

$$G(\rho = a, \phi, z) = \frac{1}{\Gamma(5/3)2^{2/3}} \frac{4}{9\pi La} \sum_{n=1}^{\infty} \sin\left(\frac{2}{3}\phi\right) \sin\left(\frac{n\pi z}{L}\right) \sin\left(\frac{n\pi z'}{L}\right) \times \left(\frac{n\pi}{L}\right)^{2/3} \frac{1}{I_{2/3}(\frac{n\pi a}{L})}$$

$$G(\rho, \phi, z = 0) = \frac{1}{\Gamma(5/3)2^{2/3}} \frac{4}{9\pi L} \sum_{n=1}^{\infty} \sin\left(\frac{2}{3}\phi\right) \left(\frac{n\pi}{L}\right)^{5/3} \sin\left(\frac{n\pi z'}{L}\right) \times \frac{1}{I_{2/3}(\frac{n\pi a}{L})} \left[I_{2/3}\left(\frac{n\pi a}{L}\right) K_{2/3}\left(\frac{n\pi \rho}{L}\right) - K_{2/3}\left(\frac{n\pi a}{L}\right) I_{2/3}\left(\frac{n\pi \rho}{L}\right) \right]$$

Unit Cube Edge Distribution

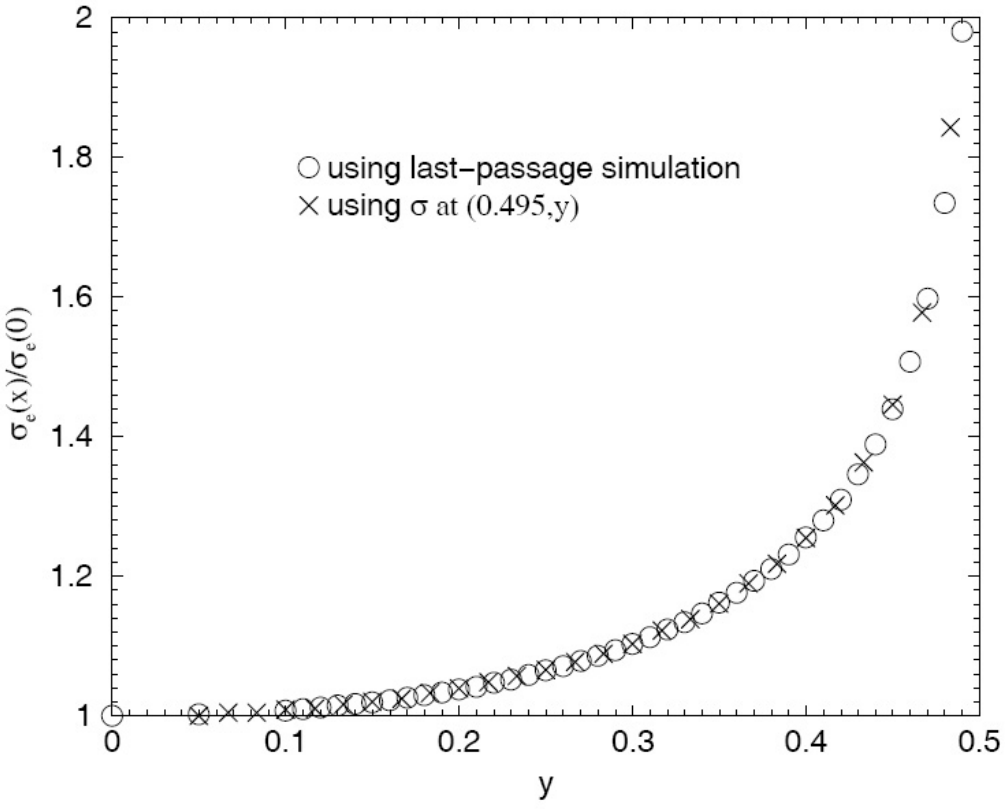


Figure: First- and last-passage edge computations

Unit Cube Edge Distribution

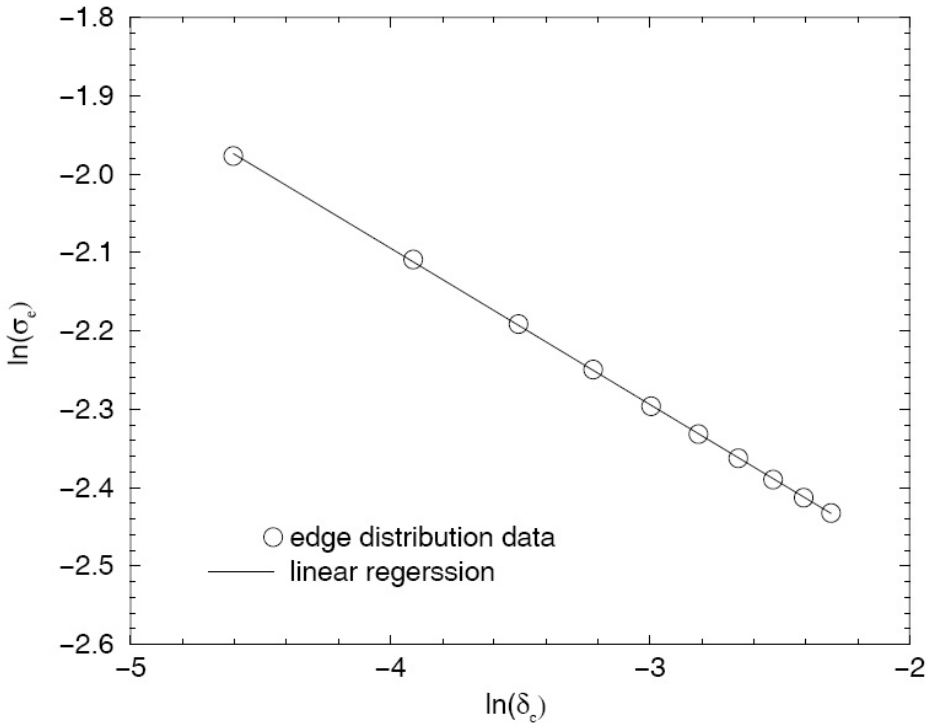


Figure: The slope, that is, the exponent of the edge distribution near the corner is approximately -0.20, that is, $\sigma_e \sim \delta_c^{-1/5}$

Walk on the Boundary Algorithm

- $\mu(y) = -\frac{1}{4\pi}\frac{\partial \phi}{\partial n}(y)$; surface charge density
- $\phi(x) = \int_{\partial\Omega} \frac{1}{|x-y|}\mu(y)d\sigma(y)$; electrostatic potential

Limit properties of the normal derivative ($x \to y$ outside of Ω):

$$\mu(y) = \int_{\partial\Omega} \frac{n(y)\cdot(y-y')}{2\pi|y-y'|^3}\mu(y')d\sigma(y')$$

By the ergodic theorem (convex Ω)

$$\int_{\partial\Omega} v(y)\pi_\infty(y)d\sigma(y) = \lim_{N\to\infty}\frac{1}{N}\sum_{n=1}^{N} v(y_n)$$

Walk on the Boundary Algorithm

- $\mu(y) = -\frac{1}{4\pi}\frac{\partial \phi}{\partial n}(y)$; surface charge density
- $\phi(x) = \int_{\partial\Omega} \frac{1}{|x-y|}\mu(y)d\sigma(y)$; electrostatic potential

Limit properties of the normal derivative ($x \to y$ outside of Ω):

$$\mu(y) = \int_{\partial\Omega} \frac{n(y)\cdot(y-y')}{2\pi|y-y'|^3}\mu(y')d\sigma(y')$$

By the ergodic theorem (convex Ω)

$$\int_{\partial\Omega} v(y)\pi_\infty(y)d\sigma(y) = \lim_{N\to\infty}\frac{1}{N}\sum_{n=1}^{N} v(y_n)$$

Walk on the Boundary Algorithm

- π_∞ - stationary distribution of Markov chain $\{y_n\}$ with transition density $p(y_n \to y_{n+1}) = \dfrac{n(y_{n+1}) \cdot (y_{n+1} - y_n)}{2\pi |y_{n+1} - y_n|^3}$
- $\mu = C\pi_\infty$
- C - capacitance if $\phi|_{\partial\Omega} = 1$
- $\phi(x) = 1$ for $x \in \Omega$

$$C = (\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} v(y_n))^{-1} \quad \text{for} \quad v(y) = \frac{1}{x - y}$$

Walk on the Boundary Algorithm

- π_∞ - stationary distribution of Markov chain $\{y_n\}$ with transition density $p(y_n \to y_{n+1}) = \dfrac{n(y_{n+1}) \cdot (y_{n+1} - y_n)}{2\pi |y_{n+1} - y_n|^3}$
- $\mu = C\pi_\infty$
- C - capacitance if $\phi|_{\partial\Omega} = 1$
- $\phi(x) = 1$ for $x \in \Omega$

$$C = (\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} v(y_n))^{-1} \quad \text{for} \quad v(y) = \frac{1}{x - y}$$

Walk on the Boundary Algorithm

- π_∞ - stationary distribution of Markov chain $\{y_n\}$ with transition density $p(y_n \to y_{n+1}) = \dfrac{n(y_{n+1}) \cdot (y_{n+1} - y_n)}{2\pi |y_{n+1} - y_n|^3}$
- $\mu = C\pi_\infty$
- C - capacitance if $\phi|_{\partial\Omega} = 1$
- $\phi(x) = 1$ for $x \in \Omega$

$$C = (\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} v(y_n))^{-1} \quad \text{for} \quad v(y) = \frac{1}{x - y}$$

Walk on the Boundary Algorithm

- π_∞ - stationary distribution of Markov chain $\{y_n\}$ with transition density $p(y_n \to y_{n+1}) = \dfrac{n(y_{n+1}) \cdot (y_{n+1} - y_n)}{2\pi |y_{n+1} - y_n|^3}$
- $\mu = C\pi_\infty$
- C - capacitance if $\phi|_{\partial\Omega} = 1$
- $\phi(x) = 1$ for $x \in \Omega$

$$C = (\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} v(y_n))^{-1} \quad \text{for} \quad v(y) = \frac{1}{x - y}$$

Capacitance of the Unit Cube

Reitan-Higgins (1951)	0.6555
Greenspan-Silverman (1965)	0.661
Cochran (1967)	0.6596
Goto-Shi-Yoshida (1992)	$0.6615897 \pm 5 \times 10^{-7}$
Conjectured Hubbard-Douglas (1993)	0.65946...
Douglas-Zhou-Hubbard (1994)	0.6632 ± 0.0003
Given-Hubbard-Douglas (1997)	0.660675 ± 0.00001
Read (1997)	0.6606785 ± 0.000003
First passage method (2001)	0.660683 ± 0.000005
Walk on boundary algorithm (2002)	0.6606780 ± 0.0000004

Continuum Biochemical Electrostatics

Motivation

- Experimental Data: Folding, stability & binding behavior of biomolecules can be modulated by changes in salt concentration
- Physical Model: Implicit solvent-based Poisson-Boltzmann model can provide accurate predictions of salt dependent behavior of biomolecules
- Mathematical Model: Elliptic boundary-value problems

Specific Problems

- Electrostatic free energy for linear case: only finite number of electrostatic potential point values
- Dependence of energy on geometry: needs accurate treatment
- Singularities in solution: have to be taken into account analytically
- Behavior at infinity: must be exactly enforced
- Functional dependence on salt concentration: needs accurate estimate

Continuum Biochemical Electrostatics

Motivation

- Experimental Data: Folding, stability & binding behavior of biomolecules can be modulated by changes in salt concentration
- Physical Model: Implicit solvent-based Poisson-Boltzmann model can provide accurate predictions of salt dependent behavior of biomolecules
- Mathematical Model: Elliptic boundary-value problems

Specific Problems

- Electrostatic free energy for linear case: only finite number of electrostatic potential point values
- Dependence of energy on geometry: needs accurate treatment
- Singularities in solution: have to be taken into account analytically
- Behavior at infinity: must be exactly enforced
- Functional dependence on salt concentration: needs accurate estimate

Continuum Biochemical Electrostatics

Motivation

- Experimental Data: Folding, stability & binding behavior of biomolecules can be modulated by changes in salt concentration
- Physical Model: Implicit solvent-based Poisson-Boltzmann model can provide accurate predictions of salt dependent behavior of biomolecules
- Mathematical Model: Elliptic boundary-value problems

Specific Problems

- Electrostatic free energy for linear case: only finite number of electrostatic potential point values
- Dependence of energy on geometry: needs accurate treatment
- Singularities in solution: have to be taken into account analytically
- Behavior at infinity: must be exactly enforced
- Functional dependence on salt concentration: needs accurate estimate

Continuum Biochemical Electrostatics

Motivation

- Experimental Data: Folding, stability & binding behavior of biomolecules can be modulated by changes in salt concentration
- Physical Model: Implicit solvent-based Poisson-Boltzmann model can provide accurate predictions of salt dependent behavior of biomolecules
- Mathematical Model: Elliptic boundary-value problems

Specific Problems

- Electrostatic free energy for linear case: only finite number of electrostatic potential point values
- Dependence of energy on geometry: needs accurate treatment
- Singularities in solution: have to be taken into account analytically
- Behavior at infinity: must be exactly enforced
- Functional dependence on salt concentration: needs accurate estimate

Continuum Biochemical Electrostatics

Motivation

- Experimental Data: Folding, stability & binding behavior of biomolecules can be modulated by changes in salt concentration
- Physical Model: Implicit solvent-based Poisson-Boltzmann model can provide accurate predictions of salt dependent behavior of biomolecules
- Mathematical Model: Elliptic boundary-value problems

Specific Problems

- Electrostatic free energy for linear case: only finite number of electrostatic potential point values
- Dependence of energy on geometry: needs accurate treatment
- Singularities in solution: have to be taken into account analytically
- Behavior at infinity: must be exactly enforced
- Functional dependence on salt concentration: needs accurate estimate

Continuum Biochemical Electrostatics

Motivation

- Experimental Data: Folding, stability & binding behavior of biomolecules can be modulated by changes in salt concentration
- Physical Model: Implicit solvent-based Poisson-Boltzmann model can provide accurate predictions of salt dependent behavior of biomolecules
- Mathematical Model: Elliptic boundary-value problems

Specific Problems

- Electrostatic free energy for linear case: only finite number of electrostatic potential point values
- Dependence of energy on geometry: needs accurate treatment
- Singularities in solution: have to be taken into account analytically
- Behavior at infinity: must be exactly enforced
- Functional dependence on salt concentration: needs accurate estimate

Continuum Biochemical Electrostatics

Motivation

- Experimental Data: Folding, stability & binding behavior of biomolecules can be modulated by changes in salt concentration
- Physical Model: Implicit solvent-based Poisson-Boltzmann model can provide accurate predictions of salt dependent behavior of biomolecules
- Mathematical Model: Elliptic boundary-value problems

Specific Problems

- Electrostatic free energy for linear case: only finite number of electrostatic potential point values
- Dependence of energy on geometry: needs accurate treatment
- Singularities in solution: have to be taken into account analytically
- Behavior at infinity: must be exactly enforced
- Functional dependence on salt concentration: needs accurate estimate

Continuum Biochemical Electrostatics

Motivation

- Experimental Data: Folding, stability & binding behavior of biomolecules can be modulated by changes in salt concentration
- Physical Model: Implicit solvent-based Poisson-Boltzmann model can provide accurate predictions of salt dependent behavior of biomolecules
- Mathematical Model: Elliptic boundary-value problems

Specific Problems

- Electrostatic free energy for linear case: only finite number of electrostatic potential point values
- Dependence of energy on geometry: needs accurate treatment
- Singularities in solution: have to be taken into account analytically
- Behavior at infinity: must be exactly enforced
- Functional dependence on salt concentration: needs accurate estimate

Monte Carlo Methods: Properties

- Monte Carlo methods for solving Poisson and linearized Poisson-Boltzmann equations (PBEs)
- Analytical treatment of geometry, singularities, behavior at infinity
- Capability to compute point values of solution (energies) and its spatial derivatives (forces)
- New methods for the flux boundary conditions (exact integral formulation)
- Simultaneous correlated computation of values at different salt concentrations

Monte Carlo Methods: Properties

- Monte Carlo methods for solving Poisson and linearized Poisson-Boltzmann equations (PBEs)
- Analytical treatment of geometry, singularities, behavior at infinity
- Capability to compute point values of solution (energies) and its spatial derivatives (forces)
- New methods for the flux boundary conditions (exact integral formulation)
- Simultaneous correlated computation of values at different salt concentrations

Monte Carlo Methods: Properties

- Monte Carlo methods for solving Poisson and linearized Poisson-Boltzmann equations (PBEs)
- Analytical treatment of geometry, singularities, behavior at infinity
- Capability to compute point values of solution (energies) and its spatial derivatives (forces)
- New methods for the flux boundary conditions (exact integral formulation)
- Simultaneous correlated computation of values at different salt concentrations

Monte Carlo Methods: Properties

- Monte Carlo methods for solving Poisson and linearized Poisson-Boltzmann equations (PBEs)
- Analytical treatment of geometry, singularities, behavior at infinity
- Capability to compute point values of solution (energies) and its spatial derivatives (forces)
- New methods for the flux boundary conditions (exact integral formulation)
- Simultaneous correlated computation of values at different salt concentrations

Monte Carlo Methods: Properties

- Monte Carlo methods for solving Poisson and linearized Poisson-Boltzmann equations (PBEs)
- Analytical treatment of geometry, singularities, behavior at infinity
- Capability to compute point values of solution (energies) and its spatial derivatives (forces)
- New methods for the flux boundary conditions (exact integral formulation)
- Simultaneous correlated computation of values at different salt concentrations

Mathematical Model: Molecular Geometry

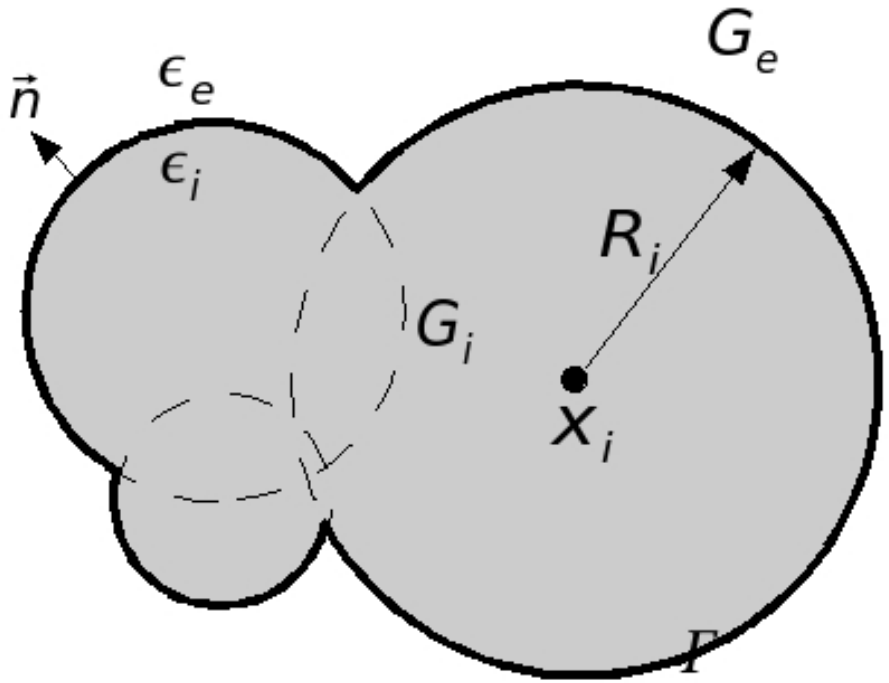


Figure: Biomolecule with dielectric ϵ_i and region region G_i is in solution with dielectric ϵ_e and region G_e. On the boundary of the biomolecule, electrostatic potential and normal component of dielectric displacement continue

Mathematical Model: Partial Differential Equations

- Poisson equation for the electrostatic potential, Φ_i, and point charges, Q_m, inside a molecule (in CGS units):

$$\epsilon_i \Delta \Phi_i(x) + 4\pi \sum_{m=1}^{M} Q_m \delta(x - x^{(m)}) = 0 \ , \ x \in G_i$$

- For 1-1 salt (such as *NaCl*) Poisson-Boltzmann equation (PBE):

$$\Delta \Phi_e(x) - \kappa^2 \sinh(\Phi_e(x)) = 0 \ , \ x \in G_e \ ,$$

but we only consider the linearized PBE:

$$\Delta \Phi_e(x) - \kappa^2 \Phi_e(x) = 0 \ , \ x \in G_e$$

- For one-surface model: continuity condition on the dielectric boundary

$$\Phi_i = \Phi_e \ , \ \epsilon_i \frac{\partial \Phi_i}{\partial n(y)} = \epsilon_e \frac{\partial \Phi_e}{\partial n(y)} \ , \ y \in \Gamma$$

Mathematical Model: Partial Differential Equations

- Poisson equation for the electrostatic potential, Φ_i, and point charges, Q_m, inside a molecule (in CGS units):

$$\epsilon_i \Delta \Phi_i(x) + 4\pi \sum_{m=1}^{M} Q_m \delta(x - x^{(m)}) = 0 \ , \ x \in G_i$$

- For 1-1 salt (such as *NaCl*) Poisson-Boltzmann equation (PBE):

$$\Delta \Phi_e(x) - \kappa^2 \sinh(\Phi_e(x)) = 0 \ , \ x \in G_e \ ,$$

but we only consider the linearized PBE:

$$\Delta \Phi_e(x) - \kappa^2 \Phi_e(x) = 0 \ , \ x \in G_e$$

- For one-surface model: continuity condition on the dielectric boundary

$$\Phi_i = \Phi_e \ , \ \epsilon_i \frac{\partial \Phi_i}{\partial n(y)} = \epsilon_e \frac{\partial \Phi_e}{\partial n(y)} \ , \ y \in \Gamma$$

Mathematical Model: Partial Differential Equations

- Poisson equation for the electrostatic potential, Φ_i, and point charges, Q_m, inside a molecule (in CGS units):

$$\epsilon_i \Delta \Phi_i(x) + 4\pi \sum_{m=1}^{M} Q_m \delta(x - x^{(m)}) = 0 \ , \ x \in G_i$$

- For 1-1 salt (such as *NaCl*) Poisson-Boltzmann equation (PBE):

$$\Delta \Phi_e(x) - \kappa^2 \sinh(\Phi_e(x)) = 0 \ , \ x \in G_e \ ,$$

but we only consider the linearized PBE:

$$\Delta \Phi_e(x) - \kappa^2 \Phi_e(x) = 0 \ , \ x \in G_e$$

- For one-surface model: continuity condition on the dielectric boundary

$$\Phi_i = \Phi_e \ , \ \epsilon_i \frac{\partial \Phi_i}{\partial n(y)} = \epsilon_e \frac{\partial \Phi_e}{\partial n(y)} \ , \ y \in \Gamma$$

Mathematical Model: Debye-Hückle Parameter

Dependence on salt in the Debye-Hückle parameter (units as per Kirkwood):

$$\kappa^2 = \frac{8\pi N_A e^2 C_s}{\epsilon_e 1000 k_B T}, \text{ where}$$

- C_s – concentration of ions (in moles)
- N_A – Avogadro's number
- e – elementary protonic charge
- k_B – Boltzmann's constant
- ϵ_e – dielectric permittivity outside the molecule

Mathematical Model: Debye-Hückle Parameter

Dependence on salt in the Debye-Hückle parameter (units as per Kirkwood):

$$\kappa^2 = \frac{8\pi N_A e^2 C_s}{\epsilon_e 1000 k_B T}, \text{ where}$$

- C_s – concentration of ions (in moles)
- N_A – Avogadro's number
- e – elementary protonic charge
- k_B – Boltzmann's constant
- ϵ_e – dielectric permittivity outside the molecule

Mathematical Model: Debye-Hückle Parameter

Dependence on salt in the Debye-Hückle parameter (units as per Kirkwood):

$$\kappa^2 = \frac{8\pi N_A e^2 C_s}{\epsilon_e 1000 k_B T}, \text{ where}$$

- C_s – concentration of ions (in moles)
- N_A – Avogadro's number
- e – elementary protonic charge
- k_B – Boltzmann's constant
- ϵ_e – dielectric permittivity outside the molecule

Mathematical Model: Debye-Hückle Parameter

Dependence on salt in the Debye-Hückle parameter (units as per Kirkwood):

$$\kappa^2 = \frac{8\pi N_A e^2 C_s}{\epsilon_e 1000 k_B T}, \text{ where}$$

- C_s – concentration of ions (in moles)
- N_A – Avogadro's number
- e – elementary protonic charge
- k_B – Boltzmann's constant
- ϵ_e – dielectric permittivity outside the molecule

Mathematical Model: Debye-Hückle Parameter

Dependence on salt in the Debye-Hückle parameter (units as per Kirkwood):

$$\kappa^2 = \frac{8\pi N_A e^2 C_s}{\epsilon_e 1000 k_B T}, \text{ where}$$

- C_s – concentration of ions (in moles)
- N_A – Avogadro's number
- e – elementary protonic charge
- k_B – Boltzmann's constant
- ϵ_e – dielectric permittivity outside the molecule

Electrostatic Potential and Energy

- Point values of the potential: $\Phi(x) = \Phi_{rf}(x) + \Phi^c(x)$
  Here, singular part of Φ:

$$\Phi^c(x) = \sum_{m=1}^{M} \frac{Q_m}{|x - x^{(m)}|}$$

- Reaction field electrostatic free energy of a molecule is linear combination of point values of the regular part of the electrostatic potential:

$$W_{rf} = \frac{1}{2} \sum_{m=1}^{M} \Phi_{rf}(x^{(m)}) Q_m \ ,$$

- Electrostatic solvation free energy = difference between the energy for a molecule in solvent with a given salt concentration and the energy for the same molecule in vacuum:

$$\Delta G^{elec}_{solv} = W_{rf}(\epsilon_i, \epsilon_e, \kappa) - W_{rf}(\epsilon_i, 1, 0)$$

Electrostatic Potential and Energy

- Point values of the potential: $\Phi(x) = \Phi_{rf}(x) + \Phi^c(x)$
  Here, singular part of Φ:

$$\Phi^c(x) = \sum_{m=1}^{M} \frac{Q_m}{|x - x^{(m)}|}$$

- Reaction field electrostatic free energy of a molecule is linear combination of point values of the regular part of the electrostatic potential:

$$W_{rf} = \frac{1}{2} \sum_{m=1}^{M} \Phi_{rf}(x^{(m)}) Q_m \ ,$$

- Electrostatic solvation free energy = difference between the energy for a molecule in solvent with a given salt concentration and the energy for the same molecule in vacuum:

$$\Delta G^{elec}_{solv} = W_{rf}(\epsilon_i, \epsilon_e, \kappa) - W_{rf}(\epsilon_i, 1, 0)$$

Electrostatic Potential and Energy

- Point values of the potential: $\Phi(x) = \Phi_{rf}(x) + \Phi^c(x)$
  Here, singular part of Φ:

$$\Phi^c(x) = \sum_{m=1}^{M} \frac{Q_m}{|x - x^{(m)}|}$$

- Reaction field electrostatic free energy of a molecule is linear combination of point values of the regular part of the electrostatic potential:

$$W_{rf} = \frac{1}{2} \sum_{m=1}^{M} \Phi_{rf}(x^{(m)}) Q_m \ ,$$

- Electrostatic solvation free energy = difference between the energy for a molecule in solvent with a given salt concentration and the energy for the same molecule in vacuum:

$$\Delta G^{elec}_{solv} = W_{rf}(\epsilon_i, \epsilon_e, \kappa) - W_{rf}(\epsilon_i, 1, 0)$$

The Feynman-Kac Formula

- Consider the Dirichlet problem for the Poisson equation in the domain $\Omega \in \mathbb{R}^d$

$$-\frac{1}{2}\Delta u(x) = g(x), \quad x \in \Omega, \qquad u(x) = f(x), \quad x \in \partial\Omega$$

- If we assume $g(x) = 0$, then we have the Laplace equation, and the solution at the point $y \in \Omega$ is given as the following Brownian motion expectation:

$$u(y) = \mathbb{E}[f(\beta_y(\tau_{\partial\Omega}))],$$

where $\beta_y(\cdot)$ is Brownian motion starting at the point y, and $\tau_{\partial\Omega}$ is the first-passage time of this Brownian motion,
i.e. $\tau_{\partial\Omega} = \inf_t\{\beta_y(t) \in \partial\Omega\}$

The Feynman-Kac Formula

- Consider the Dirichlet problem for the Poisson equation in the domain $\Omega \in \mathbb{R}^d$

$$-\frac{1}{2}\Delta u(x) = g(x), \quad x \in \Omega, \qquad u(x) = f(x), \quad x \in \partial\Omega$$

- If we assume $g(x) = 0$, then we have the Laplace equation, and the solution at the point $y \in \Omega$ is given as the following Brownian motion expectation:

$$u(y) = \mathbb{E}[f(\beta_y(\tau_{\partial\Omega}))],$$

where $\beta_y(\cdot)$ is Brownian motion starting at the point y, and $\tau_{\partial\Omega}$ is the first-passage time of this Brownian motion,
i.e. $\tau_{\partial\Omega} = \inf_t\{\beta_y(t) \in \partial\Omega\}$

The Feynman-Kac Formula

- If we set $f(x) = 0$ and have $g(x) \neq 0$, the solution is

$$u(y) = \mathbb{E}\left[\int_0^{\tau_{\partial\Omega}} g(\beta_y(s))\, ds\right]$$

- By linear superposition, the solution to Poisson equation is given probabilistically as

$$u(y) = \mathbb{E}\left[\int_0^{\tau_{\partial\Omega}} g(\beta_y(s))\, ds + f(\beta_y(\tau_{\partial\Omega}))\right]$$

- The linearized Poisson-Boltzmann equation is given by

$$\Delta u(x) - \kappa^2 u(x) = 0,\ \ x \in \Omega,\ \ u(x) = f(x),\ \ x \in \partial\Omega,\ \ u \to 0 \text{ as } |x| \to \infty$$

and has Wiener integral representation:

$$u(y) = \mathbb{E}\left[f(\beta_y(\tau_{\partial\Omega}))e^{-\int_0^{\tau_{\partial\Omega}} \kappa^2\, ds}\right]$$

The Feynman-Kac Formula

- If we set $f(x) = 0$ and have $g(x) \neq 0$, the solution is

$$u(y) = \mathbb{E}\left[\int_0^{\tau_{\partial\Omega}} g(\beta_y(s))\, ds\right]$$

- By linear superposition, the solution to Poisson equation is given probabilistically as

$$u(y) = \mathbb{E}\left[\int_0^{\tau_{\partial\Omega}} g(\beta_y(s))\, ds + f(\beta_y(\tau_{\partial\Omega}))\right]$$

- The linearized Poisson-Boltzmann equation is given by

$$\Delta u(x) - \kappa^2 u(x) = 0, \;\; x \in \Omega, \;\; u(x) = f(x), \;\; x \in \partial\Omega, \;\; u \to 0 \text{ as } |x| \to \infty$$

and has Wiener integral representation:

$$u(y) = \mathbb{E}\left[f(\beta_y(\tau_{\partial\Omega}))e^{-\int_0^{\tau_{\partial\Omega}} \kappa^2\, ds}\right]$$

The Feynman-Kac Formula

- If we set $f(x) = 0$ and have $g(x) \neq 0$, the solution is

$$u(y) = \mathbb{E}\left[\int_0^{\tau_{\partial\Omega}} g(\beta_y(s))\, ds\right]$$

- By linear superposition, the solution to Poisson equation is given probabilistically as

$$u(y) = \mathbb{E}\left[\int_0^{\tau_{\partial\Omega}} g(\beta_y(s))\, ds + f(\beta_y(\tau_{\partial\Omega}))\right]$$

- The linearized Poisson-Boltzmann equation is given by

$$\Delta u(x) - \kappa^2 u(x) = 0, \;\; x \in \Omega, \;\; u(x) = f(x), \;\; x \in \partial\Omega, \;\; u \to 0 \text{ as } |x| \to \infty$$

and has Wiener integral representation:

$$u(y) = \mathbb{E}\left[f(\beta_y(\tau_{\partial\Omega}))e^{-\int_0^{\tau_{\partial\Omega}} \kappa^2\, ds}\right]$$

'Walk-on-Spheres' Algorithm

- Walk-on-spheres (WOS) algorithm for general domains with a regular boundary
- Define a Markov chain $\{x_i,\ i = 1, 2, \ldots\}$
- Set $x_0 = x^{(m)}$ for some m, $x_i = x_{i-1} + d_i \omega_i\ ,\quad i = 1, 2, \ldots$, where
  1. $d_i = d(x_{i-1})$ is distance from x_{i-1} to Γ
  2. $\{\omega_i\}$ is sequence of independent unit isotropic vectors
  3. x_i is the exit point from the ball, $B(x_{i-1}, d(x_{i-1}))$, for a Brownian motion starting at x_{i-1}
- Outside the molecule, on every step, walk-on-spheres terminates with probability $1 - q(\kappa, d_i)$, where $q(\kappa, d_i) = \dfrac{\kappa d_i}{\sinh(\kappa d_i)}$ to deal with LPBE

'Walk-on-Spheres' Algorithm

- Walk-on-spheres (WOS) algorithm for general domains with a regular boundary
- Define a Markov chain $\{x_i,\ i = 1, 2, \ldots\}$
- Set $x_0 = x^{(m)}$ for some m, $x_i = x_{i-1} + d_i \omega_i\ ,\quad i = 1, 2, \ldots$, where
  1. $d_i = d(x_{i-1})$ is distance from x_{i-1} to Γ
  2. $\{\omega_i\}$ is sequence of independent unit isotropic vectors
  3. x_i is the exit point from the ball, $B(x_{i-1}, d(x_{i-1}))$, for a Brownian motion starting at x_{i-1}
- Outside the molecule, on every step, walk-on-spheres terminates with probability $1 - q(\kappa, d_i)$, where $q(\kappa, d_i) = \dfrac{\kappa d_i}{\sinh(\kappa d_i)}$ to deal with LPBE

'Walk-on-Spheres' Algorithm

- Walk-on-spheres (WOS) algorithm for general domains with a regular boundary
- Define a Markov chain $\{x_i,\ i = 1, 2, \ldots\}$
- Set $x_0 = x^{(m)}$ for some m, $x_i = x_{i-1} + d_i\omega_i\ ,\ \ i = 1, 2, \ldots$, where
  1. $d_i = d(x_{i-1})$ is distance from x_{i-1} to Γ
  2. $\{\omega_i\}$ is sequence of independent unit isotropic vectors
  3. x_i is the exit point from the ball, $B(x_{i-1}, d(x_{i-1}))$, for a Brownian motion starting at x_{i-1}
- Outside the molecule, on every step, walk-on-spheres terminates with probability $1 - q(\kappa, d_i)$, where $q(\kappa, d_i) = \dfrac{\kappa d_i}{\sinh(\kappa d_i)}$ to deal with LPBE

'Walk-on-Spheres' Algorithm

- Walk-on-spheres (WOS) algorithm for general domains with a regular boundary
- Define a Markov chain $\{x_i,\ i = 1, 2, \ldots\}$
- Set $x_0 = x^{(m)}$ for some m, $x_i = x_{i-1} + d_i\omega_i\ ,\ \ i = 1, 2, \ldots$, where
  1. $d_i = d(x_{i-1})$ is distance from x_{i-1} to Γ
  2. $\{\omega_i\}$ is sequence of independent unit isotropic vectors
  3. x_i is the exit point from the ball, $B(x_{i-1}, d(x_{i-1}))$, for a Brownian motion starting at x_{i-1}
- Outside the molecule, on every step, walk-on-spheres terminates with probability $1 - q(\kappa, d_i)$, where $q(\kappa, d_i) = \dfrac{\kappa d_i}{\sinh(\kappa d_i)}$ to deal with LPBE

'Walk-on-Spheres' Algorithm

- Walk-on-spheres (WOS) algorithm for general domains with a regular boundary
- Define a Markov chain $\{x_i,\ i = 1, 2, \ldots\}$
- Set $x_0 = x^{(m)}$ for some m, $x_i = x_{i-1} + d_i\omega_i\ ,\ \ i = 1, 2, \ldots$, where
  1. $d_i = d(x_{i-1})$ is distance from x_{i-1} to Γ
  2. $\{\omega_i\}$ is sequence of independent unit isotropic vectors
  3. x_i is the exit point from the ball, $B(x_{i-1}, d(x_{i-1}))$, for a Brownian motion starting at x_{i-1}
- Outside the molecule, on every step, walk-on-spheres terminates with probability $1 - q(\kappa, d_i)$, where $q(\kappa, d_i) = \dfrac{\kappa d_i}{\sinh(\kappa d_i)}$ to deal with LPBE

'Walk-on-Spheres' Algorithm

- Walk-on-spheres (WOS) algorithm for general domains with a regular boundary
- Define a Markov chain $\{x_i,\ i = 1, 2, \ldots\}$
- Set $x_0 = x^{(m)}$ for some m, $x_i = x_{i-1} + d_i\omega_i\ ,\quad i = 1, 2, \ldots$, where
  1. $d_i = d(x_{i-1})$ is distance from x_{i-1} to Γ
  2. $\{\omega_i\}$ is sequence of independent unit isotropic vectors
  3. x_i is the exit point from the ball, $B(x_{i-1}, d(x_{i-1}))$, for a Brownian motion starting at x_{i-1}
- Outside the molecule, on every step, walk-on-spheres terminates with probability $1 - q(\kappa, d_i)$, where $q(\kappa, d_i) = \dfrac{\kappa d_i}{\sinh(\kappa d_i)}$ to deal with LPBE

'Walk-on-Spheres' Algorithm

- Walk-on-spheres (WOS) algorithm for general domains with a regular boundary
- Define a Markov chain $\{x_i,\ i = 1, 2, \ldots\}$
- Set $x_0 = x^{(m)}$ for some m, $x_i = x_{i-1} + d_i\omega_i\ ,\quad i = 1, 2, \ldots$, where
  1. $d_i = d(x_{i-1})$ is distance from x_{i-1} to Γ
  2. $\{\omega_i\}$ is sequence of independent unit isotropic vectors
  3. x_i is the exit point from the ball, $B(x_{i-1}, d(x_{i-1}))$, for a Brownian motion starting at x_{i-1}
- Outside the molecule, on every step, walk-on-spheres terminates with probability $1 - q(\kappa, d_i)$, where $q(\kappa, d_i) = \dfrac{\kappa d_i}{\sinh(\kappa d_i)}$ to deal with LPBE

'Walk-on-Spheres' and 'Walk-in-Subdomains'

- For general domains, an efficient way to simulate exit points is a combination of
  1. Inside the molecule: 'walk-in-subdomains'
  2. Outside the molecule 'walk-on-spheres'
- The whole domain, G_i, is represented as a union of intersecting subdomains:

$$G_i = \bigcup_{m=1}^{M} G^m$$

- 'Walk-in-Subdomains': Simulate exit point separately in every G^m

  1. $x_0 = x, x_1, \ldots, x_N$ – Markov chain, every x_{i+1} is an exit point from the corresponding subdomain for Brownian motion starting at x_i
  2. For spherical subdomains, $B(x_i^m, R_i^m)$, exit points are distributed in accordance with the Poisson kernel:

$$\frac{1}{4\pi R_i^m} \frac{|x_i - x_i^m|^2 - (R_i^m)^2}{|x_i - x_{i+1}|^3}$$

'Walk-on-Spheres' and 'Walk-in-Subdomains'

- For general domains, an efficient way to simulate exit points is a combination of
  1. Inside the molecule: 'walk-in-subdomains'
  2. Outside the molecule 'walk-on-spheres'
- The whole domain, G_i, is represented as a union of intersecting subdomains:

$$G_i = \bigcup_{m=1}^{M} G^m$$

- 'Walk-in-Subdomains': Simulate exit point separately in every G^m

  1. $x_0 = x, x_1, \ldots, x_N$ – Markov chain, every x_{i+1} is an exit point from the corresponding subdomain for Brownian motion starting at x_i
  2. For spherical subdomains, $B(x_i^m, R_i^m)$, exit points are distributed in accordance with the Poisson kernel:

$$\frac{1}{4\pi R_i^m} \frac{|x_i - x_i^m|^2 - (R_i^m)^2}{|x_i - x_{i+1}|^3}$$

'Walk-on-Spheres' and 'Walk-in-Subdomains'

- For general domains, an efficient way to simulate exit points is a combination of
  1. Inside the molecule: 'walk-in-subdomains'
  2. Outside the molecule 'walk-on-spheres'
- The whole domain, G_i, is represented as a union of intersecting subdomains:

$$G_i = \bigcup_{m=1}^{M} G^m$$

- 'Walk-in-Subdomains': Simulate exit point separately in every G^m

  1. $x_0 = x, x_1, \ldots, x_N$ – Markov chain, every x_{i+1} is an exit point from the corresponding subdomain for Brownian motion starting at x_i
  2. For spherical subdomains, $B(x_i^m, R_i^m)$, exit points are distributed in accordance with the Poisson kernel:

$$\frac{1}{4\pi R_i^m} \frac{|x_i - x_i^m|^2 - (R_i^m)^2}{|x_i - x_{i+1}|^3}$$

'Walk-on-Spheres' and 'Walk-in-Subdomains'

- For general domains, an efficient way to simulate exit points is a combination of
  1. Inside the molecule: 'walk-in-subdomains'
  2. Outside the molecule 'walk-on-spheres'
- The whole domain, G_i, is represented as a union of intersecting subdomains:

$$G_i = \bigcup_{m=1}^{M} G^m$$

- 'Walk-in-Subdomains': Simulate exit point separately in every G^m

  1. $x_0 = x, x_1, \ldots, x_N$ – Markov chain, every x_{i+1} is an exit point from the corresponding subdomain for Brownian motion starting at x_i
  2. For spherical subdomains, $B(x_i^m, R_i^m)$, exit points are distributed in accordance with the Poisson kernel:

$$\frac{1}{4\pi R_i^m} \frac{|x_i - x_i^m|^2 - (R_i^m)^2}{|x_i - x_{i+1}|^3}$$

'Walk-on-Spheres' and 'Walk-in-Subdomains'

- For general domains, an efficient way to simulate exit points is a combination of
  1. Inside the molecule: 'walk-in-subdomains'
  2. Outside the molecule 'walk-on-spheres'
- The whole domain, G_i, is represented as a union of intersecting subdomains:

$$G_i = \bigcup_{m=1}^{M} G^m$$

- 'Walk-in-Subdomains': Simulate exit point separately in every G^m

  1. $x_0 = x, x_1, \ldots, x_N$ – Markov chain, every x_{i+1} is an exit point from the corresponding subdomain for Brownian motion starting at x_i
  2. For spherical subdomains, $B(x_i^m, R_i^m)$, exit points are distributed in accordance with the Poisson kernel:

$$\frac{1}{4\pi R_i^m} \frac{|x_i - x_i^m|^2 - (R_i^m)^2}{|x_i - x_{i+1}|^3}$$

'Walk-on-Spheres' and 'Walk-in-Subdomains'

- For general domains, an efficient way to simulate exit points is a combination of
  1. Inside the molecule: 'walk-in-subdomains'
  2. Outside the molecule 'walk-on-spheres'
- The whole domain, G_i, is represented as a union of intersecting subdomains:

$$G_i = \bigcup_{m=1}^{M} G^m$$

- 'Walk-in-Subdomains': Simulate exit point separately in every G^m

  1. $x_0 = x, x_1, \ldots, x_N$ – Markov chain, every x_{i+1} is an exit point from the corresponding subdomain for Brownian motion starting at x_i
  2. For spherical subdomains, $B(x_i^m, R_i^m)$, exit points are distributed in accordance with the Poisson kernel:

$$\frac{1}{4\pi R_i^m} \frac{|x_i - x_i^m|^2 - (R_i^m)^2}{|x_i - x_{i+1}|^3}$$

'Walk-on-Spheres' and 'Walk-in-Subdomains'

- For general domains, an efficient way to simulate exit points is a combination of
  1. Inside the molecule: 'walk-in-subdomains'
  2. Outside the molecule 'walk-on-spheres'
- The whole domain, G_i, is represented as a union of intersecting subdomains:

$$G_i = \bigcup_{m=1}^{M} G^m$$

- 'Walk-in-Subdomains': Simulate exit point separately in every G^m

  1. $x_0 = x, x_1, \ldots, x_N$ – Markov chain, every x_{i+1} is an exit point from the corresponding subdomain for Brownian motion starting at x_i
  2. For spherical subdomains, $B(x_i^m, R_i^m)$, exit points are distributed in accordance with the Poisson kernel:

$$\frac{1}{4\pi R_i^m} \frac{|x_i - x_i^m|^2 - (R_i^m)^2}{|x_i - x_{i+1}|^3}$$

'Walk-on-Spheres' and 'Walk-in-Subdomains'

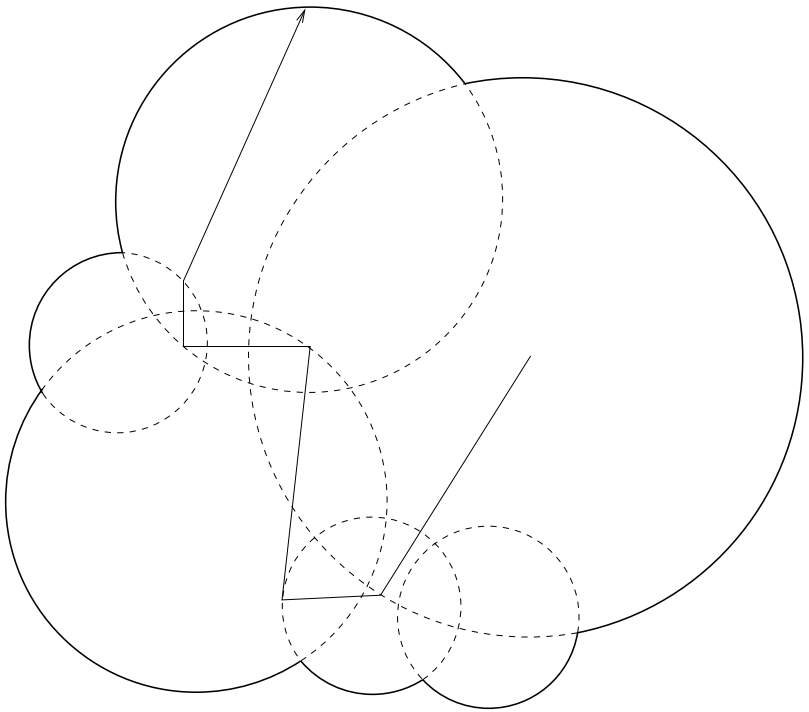


Figure: Walk in subdomains example

Monte Carlo Treatment of Boundary Conditions

- Randomization of finite-difference approximation with step, h. $u(y) = \mathbb{E}u(x) + O(h^2)$
- Exact treatment of boundary conditions (mean-value theorem) for boundary point, y, in the ball $B(y,a)$ with surface $S(y,a)$:

$$\begin{aligned} u(y) &= \frac{\epsilon_e}{\epsilon_e+\epsilon_i}\int_{S_e(y,a)}\frac{1}{2\pi a^2}\frac{\kappa a}{\sinh(\kappa a)}u_e \\ &+ \frac{\epsilon_i}{\epsilon_e+\epsilon_i}\int_{S_i(y,a)}\frac{1}{2\pi a^2}\frac{\kappa a}{\sinh(\kappa a)}u_i \\ &- \frac{\epsilon_e-\epsilon_i}{\epsilon_e+\epsilon_i}\int_{\Gamma\cap B(y,a)\setminus\{y\}}\frac{\cos\varphi_{yx}}{2\pi|y-x|^2}Q_{\kappa,a}u \\ &+ \frac{\epsilon_i}{\epsilon_e+\epsilon_i}\int_{B_i(y,a)}[-2\kappa^2\Phi_\kappa]u_i \end{aligned} \tag{22}$$

Monte Carlo Treatment of Boundary Conditions

- Randomization of finite-difference approximation with step, h. $u(y) = \mathbb{E}u(x) + O(h^2)$
- Exact treatment of boundary conditions (mean-value theorem) for boundary point, y, in the ball $B(y, a)$ with surface $S(y, a)$:

$$\begin{aligned} u(y) &= \frac{\epsilon_e}{\epsilon_e + \epsilon_i} \int_{S_e(y,a)} \frac{1}{2\pi a^2} \frac{\kappa a}{\sinh(\kappa a)} u_e \\ &+ \frac{\epsilon_i}{\epsilon_e + \epsilon_i} \int_{S_i(y,a)} \frac{1}{2\pi a^2} \frac{\kappa a}{\sinh(\kappa a)} u_i \\ &- \frac{\epsilon_e - \epsilon_i}{\epsilon_e + \epsilon_i} \int_{\Gamma \bigcap B(y,a)\setminus\{y\}} \frac{\cos\varphi_{yx}}{2\pi|y-x|^2} Q_{\kappa,a} u \\ &+ \frac{\epsilon_i}{\epsilon_e + \epsilon_i} \int_{B_i(y,a)} [-2\kappa^2 \Phi_\kappa] u_i \end{aligned} \tag{22}$$

Monte Carlo Treatment of Boundary Conditions

Randomized approximation to (22): $u(y) = \mathbb{E}u(x) + O((a/2R)^3)$:

- With probability p_e exit to solvent:
  1. x is chosen isotropically on the surface of auxiliary sphere, $S_+(y, a)$, that lies above tangent plane
  2. Walker survives with probability $\dfrac{\kappa a}{\sinh(\kappa a)}$
- With probability $p_i = 1 - p_e$:
  1. x is chosen isotropically in the solid angle below tangent plane; with probability $-2\kappa^2\Phi_\kappa$ & sampled in $B_i(y, a)$ (reenter molecule)
  2. With the complementary probability x is sampled on the surface of auxiliary sphere, $S_-(y, a)$, that lies below tangent plane
  3. x reenters molecule with conditional probability $1 - a/2R$ and
  4. x exits to solvent with conditional probability $a/2R$

Monte Carlo Treatment of Boundary Conditions

Randomized approximation to (22): $u(y) = \mathbb{E}u(x) + O((a/2R)^3)$:

- With probability p_e exit to solvent:
  1. x is chosen isotropically on the surface of auxiliary sphere, $S_+(y, a)$, that lies above tangent plane
  2. Walker survives with probability $\dfrac{\kappa a}{\sinh(\kappa a)}$
- With probability $p_i = 1 - p_e$:
  1. x is chosen isotropically in the solid angle below tangent plane; with probability $-2\kappa^2\Phi_\kappa$ & sampled in $B_i(y, a)$ (reenter molecule)
  2. With the complementary probability x is sampled on the surface of auxiliary sphere, $S_-(y, a)$, that lies below tangent plane
  3. x reenters molecule with conditional probability $1 - a/2R$ and
  4. x exits to solvent with conditional probability $a/2R$

Monte Carlo Treatment of Boundary Conditions

Randomized approximation to (22): $u(y) = \mathbb{E}u(x) + O((a/2R)^3)$:

- With probability p_e exit to solvent:
  1. x is chosen isotropically on the surface of auxiliary sphere, $S_+(y, a)$, that lies above tangent plane
  2. Walker survives with probability $\dfrac{\kappa a}{\sinh(\kappa a)}$
- With probability $p_i = 1 - p_e$:
  1. x is chosen isotropically in the solid angle below tangent plane; with probability $-2\kappa^2\Phi_\kappa$ & sampled in $B_i(y, a)$ (reenter molecule)
  2. With the complementary probability x is sampled on the surface of auxiliary sphere, $S_-(y, a)$, that lies below tangent plane
  3. x reenters molecule with conditional probability $1 - a/2R$ and
  4. x exits to solvent with conditional probability $a/2R$

Monte Carlo Treatment of Boundary Conditions

Randomized approximation to (22): $u(y) = \mathbb{E}u(x) + O((a/2R)^3)$:

- With probability p_e exit to solvent:
  1. x is chosen isotropically on the surface of auxiliary sphere, $S_+(y, a)$, that lies above tangent plane
  2. Walker survives with probability $\dfrac{\kappa a}{\sinh(\kappa a)}$
- With probability $p_i = 1 - p_e$:
  1. x is chosen isotropically in the solid angle below tangent plane; with probability $-2\kappa^2\Phi_\kappa$ & sampled in $B_i(y, a)$ (reenter molecule)
  2. With the complementary probability x is sampled on the surface of auxiliary sphere, $S_-(y, a)$, that lies below tangent plane
  3. x reenters molecule with conditional probability $1 - a/2R$ and
  4. x exits to solvent with conditional probability $a/2R$

Monte Carlo Treatment of Boundary Conditions

Randomized approximation to (22): $u(y) = \mathbb{E}u(x) + O((a/2R)^3)$:

- With probability p_e exit to solvent:
  1. x is chosen isotropically on the surface of auxiliary sphere, $S_+(y, a)$, that lies above tangent plane
  2. Walker survives with probability $\dfrac{\kappa a}{\sinh(\kappa a)}$
- With probability $p_i = 1 - p_e$:
  1. x is chosen isotropically in the solid angle below tangent plane; with probability $-2\kappa^2\Phi_\kappa$ & sampled in $B_i(y, a)$ (reenter molecule)
  2. With the complementary probability x is sampled on the surface of auxiliary sphere, $S_-(y, a)$, that lies below tangent plane
  3. x reenters molecule with conditional probability $1 - a/2R$ and
  4. x exits to solvent with conditional probability $a/2R$

Monte Carlo Treatment of Boundary Conditions

Randomized approximation to (22): $u(y) = \mathbb{E}u(x) + O((a/2R)^3)$:

- With probability p_e exit to solvent:
  1. x is chosen isotropically on the surface of auxiliary sphere, $S_+(y, a)$, that lies above tangent plane
  2. Walker survives with probability $\dfrac{\kappa a}{\sinh(\kappa a)}$
- With probability $p_i = 1 - p_e$:
  1. x is chosen isotropically in the solid angle below tangent plane; with probability $-2\kappa^2\Phi_\kappa$ & sampled in $B_i(y, a)$ (reenter molecule)
  2. With the complementary probability x is sampled on the surface of auxiliary sphere, $S_-(y, a)$, that lies below tangent plane
  3. x reenters molecule with conditional probability $1 - a/2R$ and
  4. x exits to solvent with conditional probability $a/2R$

Monte Carlo Treatment of Boundary Conditions

Randomized approximation to (22): $u(y) = \mathbb{E}u(x) + O((a/2R)^3)$:

- With probability p_e exit to solvent:
  1. x is chosen isotropically on the surface of auxiliary sphere, $S_+(y, a)$, that lies above tangent plane
  2. Walker survives with probability $\dfrac{\kappa a}{\sinh(\kappa a)}$
- With probability $p_i = 1 - p_e$:
  1. x is chosen isotropically in the solid angle below tangent plane; with probability $-2\kappa^2\Phi_\kappa$ & sampled in $B_i(y, a)$ (reenter molecule)
  2. With the complementary probability x is sampled on the surface of auxiliary sphere, $S_-(y, a)$, that lies below tangent plane
  3. x reenters molecule with conditional probability $1 - a/2R$ and
  4. x exits to solvent with conditional probability $a/2R$

Monte Carlo Treatment of Boundary Conditions

Randomized approximation to (22): $u(y) = \mathbb{E}u(x) + O((a/2R)^3)$:

- With probability p_e exit to solvent:
  1. x is chosen isotropically on the surface of auxiliary sphere, $S_+(y, a)$, that lies above tangent plane
  2. Walker survives with probability $\dfrac{\kappa a}{\sinh(\kappa a)}$
- With probability $p_i = 1 - p_e$:
  1. x is chosen isotropically in the solid angle below tangent plane; with probability $-2\kappa^2\Phi_\kappa$ & sampled in $B_i(y, a)$ (reenter molecule)
  2. With the complementary probability x is sampled on the surface of auxiliary sphere, $S_-(y, a)$, that lies below tangent plane
  3. x reenters molecule with conditional probability $1 - a/2R$ and
  4. x exits to solvent with conditional probability $a/2R$

Monte Carlo Treatment of Boundary Conditions

In the exterior, probability of terminating Markov chain depends linearly on the initial distance to the boundary, d_0. Therefore, $\Rightarrow$ Mean number of returns to the boundary is $O(d_0)^{-1}$

- Finite-difference approximation of boundary conditions, $\varepsilon = h^2$ Mean number of steps in the algorithm is $O(h^{-1}\log(h)\, f(\kappa))$, f is a decreasing function ($f(\kappa) = O(\log(\kappa))$ for small κ). Estimates for point values of the potential and free energy are $O(h)$-biased
- New treatment of boundary conditions provides $O(\overline{a})^2$-biased and more **efficient** Monte Carlo algorithm. Mean number of steps is $O((\overline{a})^{-1}\log(\overline{a})\, f(\kappa))$, $\overline{a} = a/2R$.
- More subtle approximation to (22) will provide even more efficient Monte Carlo estimates

Monte Carlo Treatment of Boundary Conditions

In the exterior, probability of terminating Markov chain depends linearly on the initial distance to the boundary, d_0. Therefore, $\Rightarrow$ Mean number of returns to the boundary is $O(d_0)^{-1}$

- Finite-difference approximation of boundary conditions, $\varepsilon = h^2$ Mean number of steps in the algorithm is $O(h^{-1}\log(h)\, f(\kappa))$, f is a decreasing function ($f(\kappa) = O(\log(\kappa))$ for small κ). Estimates for point values of the potential and free energy are $O(h)$-biased
- New treatment of boundary conditions provides $O(\overline{a})^2$-biased and more **efficient** Monte Carlo algorithm. Mean number of steps is $O((\overline{a})^{-1}\log(\overline{a})\, f(\kappa))$, $\overline{a} = a/2R$.
- More subtle approximation to (22) will provide even more efficient Monte Carlo estimates

Monte Carlo Treatment of Boundary Conditions

In the exterior, probability of terminating Markov chain depends linearly on the initial distance to the boundary, d_0. Therefore, $\Rightarrow$ Mean number of returns to the boundary is $O(d_0)^{-1}$

- Finite-difference approximation of boundary conditions, $\varepsilon = h^2$ Mean number of steps in the algorithm is $O(h^{-1}\log(h)\, f(\kappa))$, f is a decreasing function ($f(\kappa) = O(\log(\kappa))$ for small κ). Estimates for point values of the potential and free energy are $O(h)$-biased
- New treatment of boundary conditions provides $O(\overline{a})^2$-biased and more **efficient** Monte Carlo algorithm. Mean number of steps is $O((\overline{a})^{-1}\log(\overline{a})\, f(\kappa))$, $\overline{a} = a/2R$.
- More subtle approximation to (22) will provide even more efficient Monte Carlo estimates

Monte Carlo Estimates

- The estimate for the reaction-field potential point value:

$$\begin{aligned} \xi[\Phi_{rf}](x^{(m)}) &= -\Phi^c(x_1^*) \\ &+ \sum_{j=2}^{N_{ins}} F_j(\kappa)\left(\Phi^c(x_j^{ins}) - \Phi^c(x_{j,ins}^*)\right) \end{aligned} \tag{23}$$

- Here $\{x_{j,ins}^*\}$ is a sequence of boundary points, after which the random walker moves inside the domain, G_i, to x_j^{ins}
- The estimate for the reaction-field energy:

$$\xi[W_{rf}] = \frac{1}{2}\sum_{m=1}^{M} Q_m\, \xi[\Phi_{rf}](x^{(m)}) \tag{24}$$

Monte Carlo Estimates

- The estimate for the reaction-field potential point value:

$$\begin{aligned} \xi[\Phi_{rf}](x^{(m)}) &= -\Phi^c(x_1^*) \\ &+ \sum_{j=2}^{N_{ins}} F_j(\kappa)\left(\Phi^c(x_j^{ins}) - \Phi^c(x_{j,ins}^*)\right) \end{aligned} \tag{23}$$

- Here $\{x_{j,ins}^*\}$ is a sequence of boundary points, after which the random walker moves inside the domain, G_i, to x_j^{ins}
- The estimate for the reaction-field energy:

$$\xi[W_{rf}] = \frac{1}{2}\sum_{m=1}^{M} Q_m\, \xi[\Phi_{rf}](x^{(m)}) \tag{24}$$

Monte Carlo Estimates

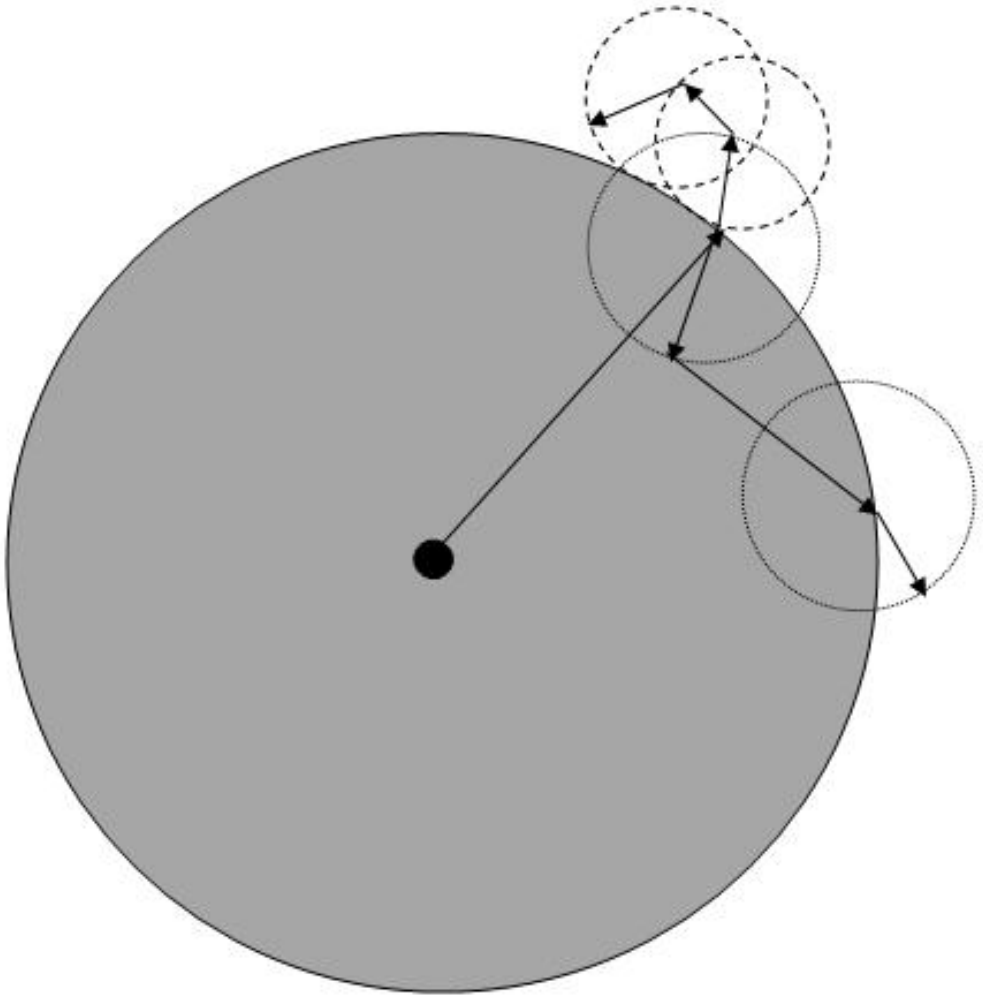
- The estimate for the reaction-field potential point value:

$$\begin{aligned}\xi[\Phi_{rf}](x^{(m)}) &= -\Phi^c(x_1^*) \\ &+ \sum_{j=2}^{N_{ins}} F_j(\kappa)\left(\Phi^c(x_j^{ins}) - \Phi^c(x_{j,ins}^*)\right)\end{aligned} \tag{23}$$

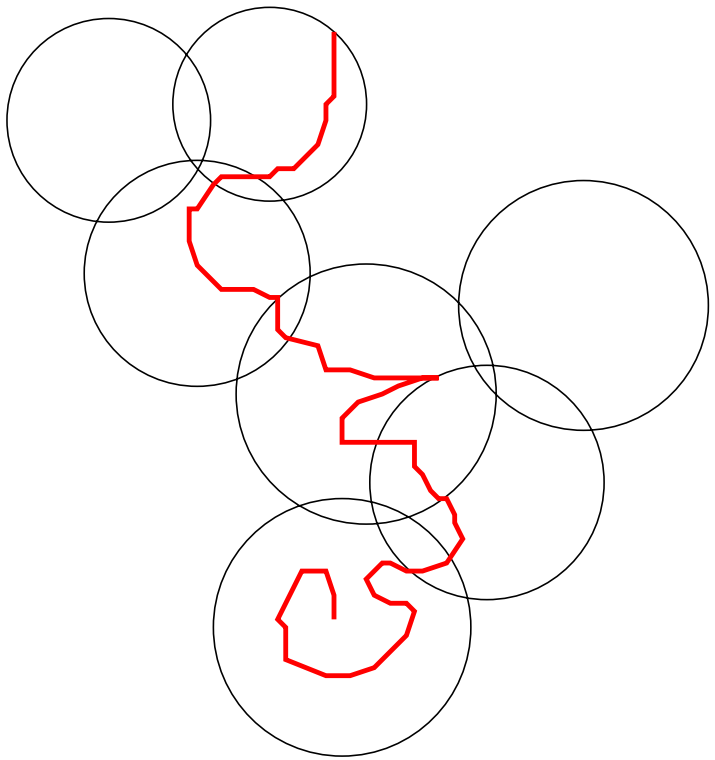
- Here $\{x_{j,ins}^*\}$ is a sequence of boundary points, after which the random walker moves inside the domain, G_i, to x_j^{ins}
- The estimate for the reaction-field energy:

$$\xi[W_{rf}] = \frac{1}{2}\sum_{m=1}^{M} Q_m\, \xi[\Phi_{rf}](x^{(m)}) \tag{24}$$

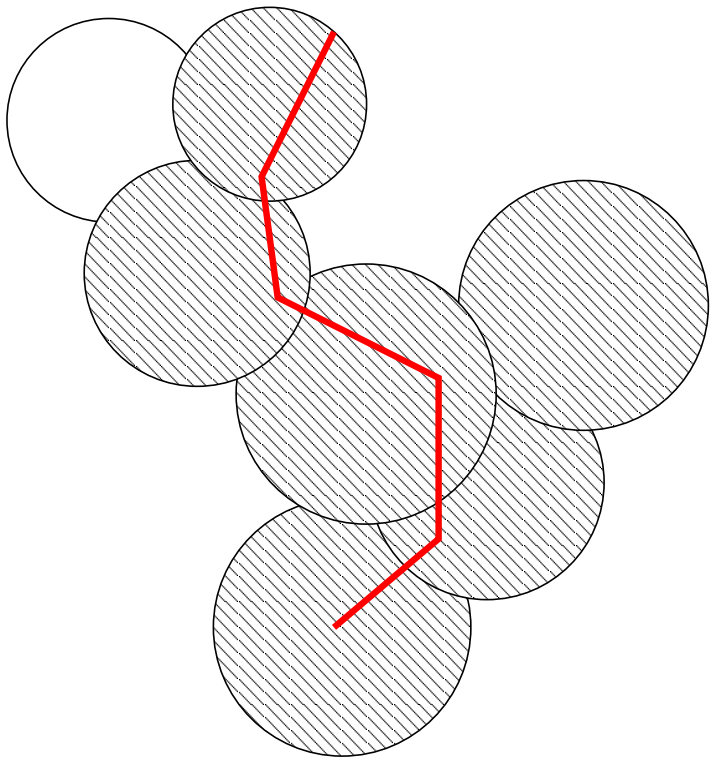
A Picture: The Algorithm for a Single Spherical Atom



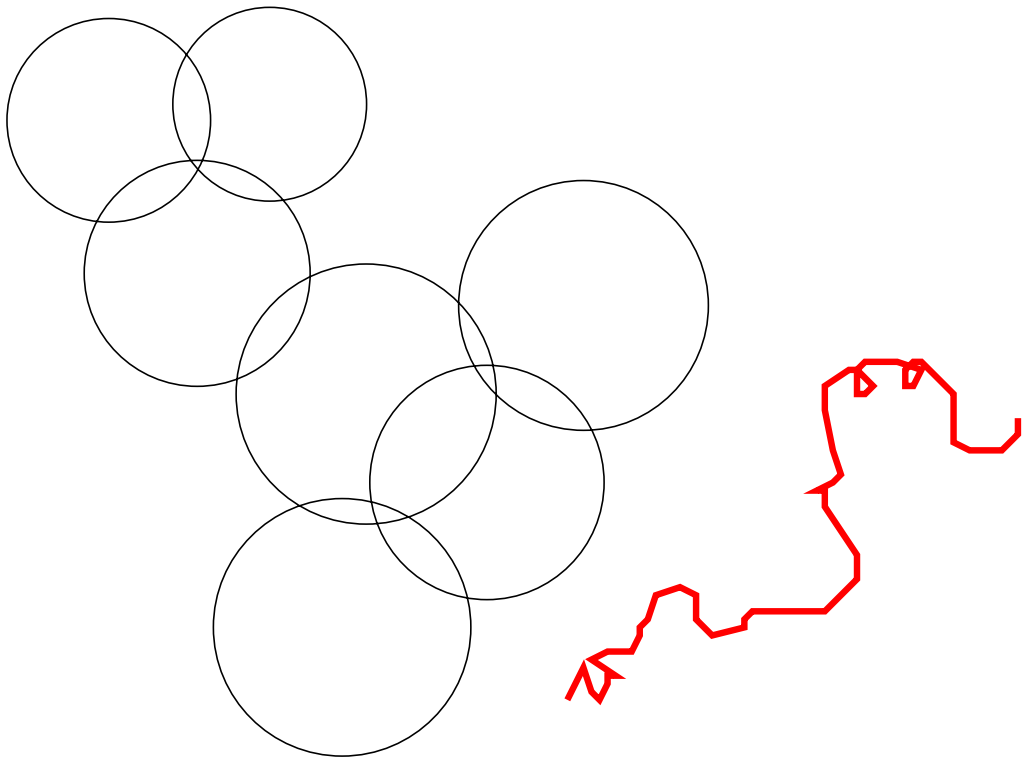
The Algorithm in Pictures: Walk Inside



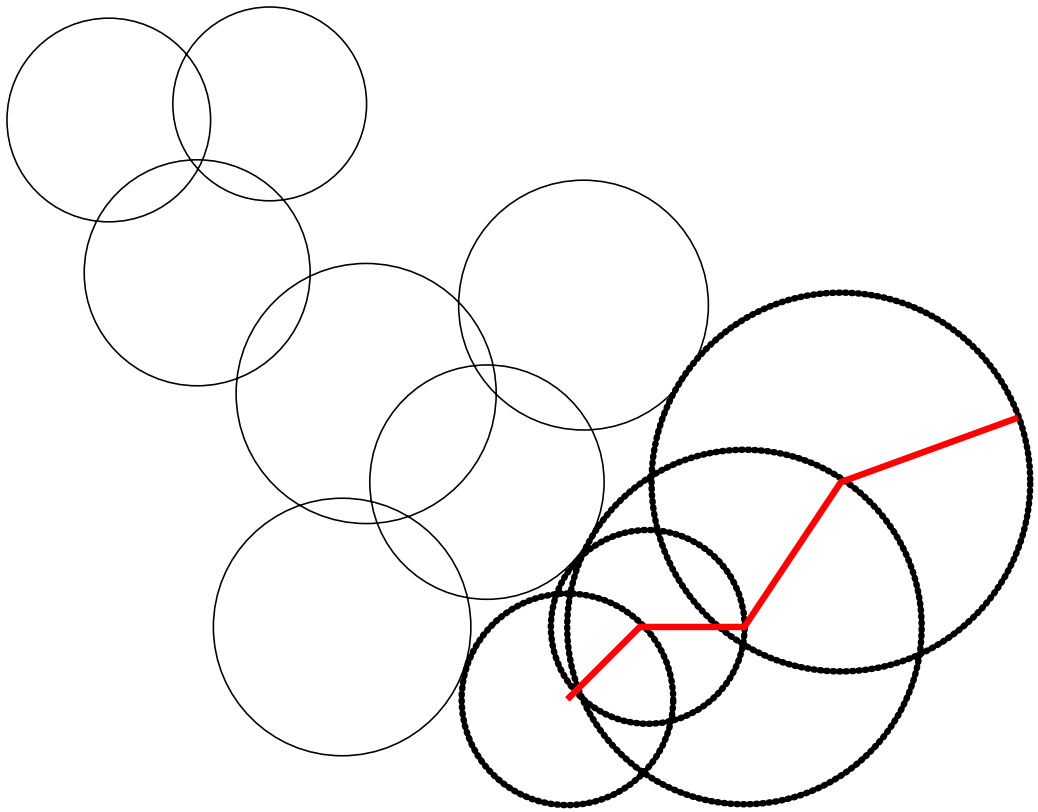
The Algorithm in Pictures: Walk Inside



The Algorithm in Pictures: Walk Outside



The Algorithm in Pictures: Walk Outside



The Algorithm in Pictures: Walk to ∞ in One Step

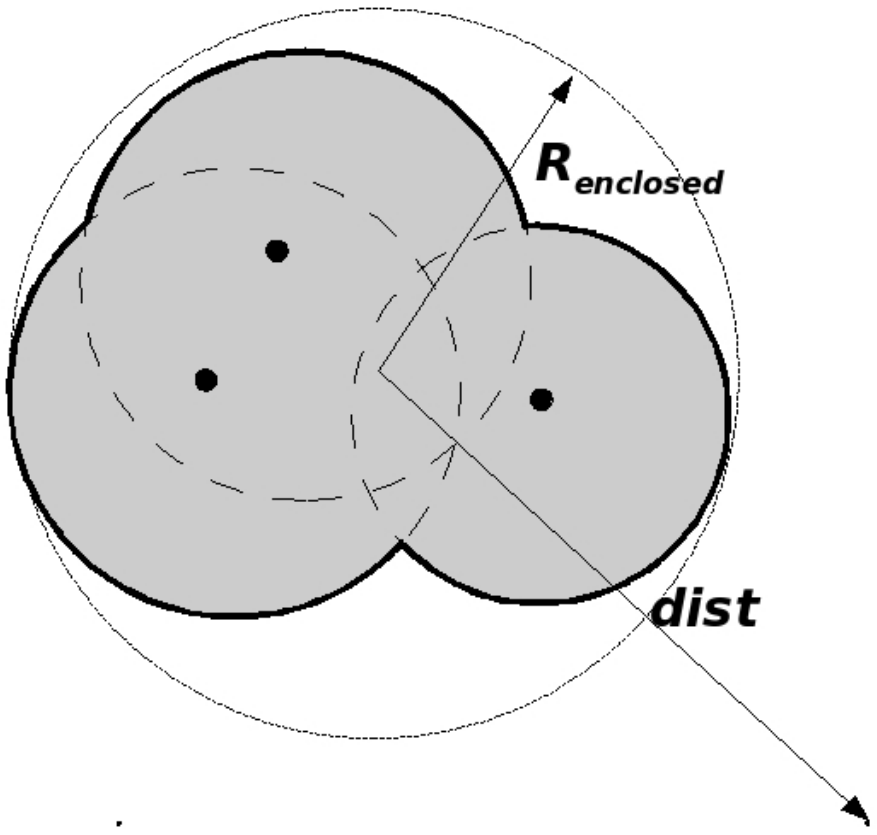


Figure: $\kappa = 0$, $p_\infty = 1 - R_{Enclosed}/dist$

Monte Carlo Algorithm's Computational Complexity

Cost of a single trajectory

- Number of steps is random walk is not dependent on M, the number of atoms
- The cost of finding the nearest sphere is $M \log_2(M)$ due to optimizations

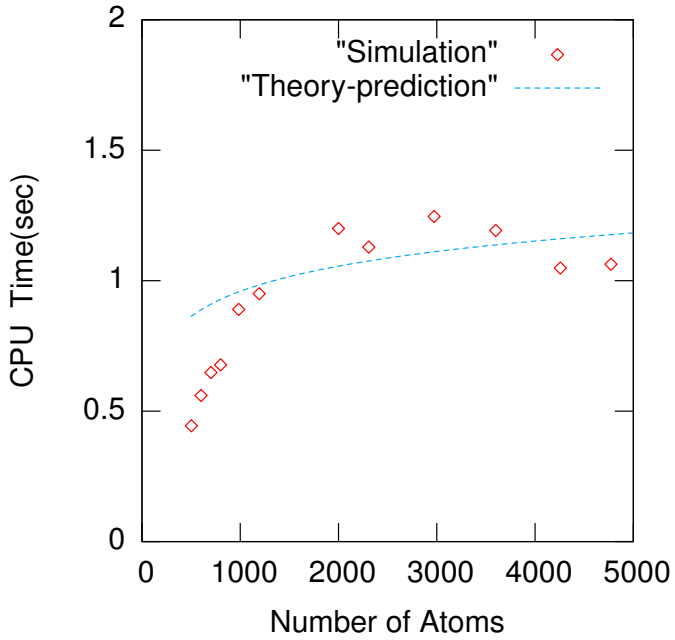


Figure: The CPU time per atom per trajectory is plotted as function of number of atoms. For small number of atoms the CPU time scales linearly and for large number of atoms it asymptotically scales logarithmically

Monte Carlo Algorithm's Computational Complexity

Cost of a single trajectory

- Number of steps is random walk is not dependent on M, the number of atoms
- The cost of finding the nearest sphere is $M \log_2(M)$ due to optimizations

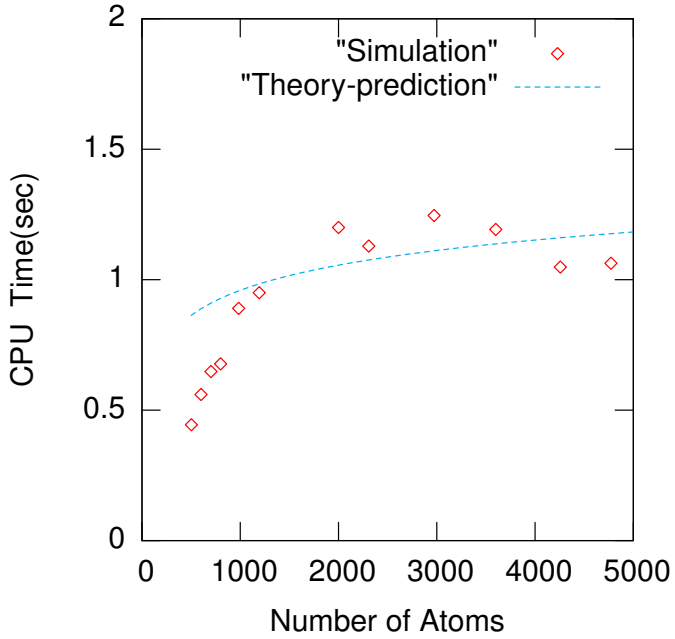


Figure: The CPU time per atom per trajectory is plotted as function of number of atoms. For small number of atoms the CPU time scales linearly and for large number of atoms it asymptotically scales logarithmically

Geometry: Problem Descriptions

There are many geometric problems that arise in this algorithm:

- Efficiently determining if a point is on the surface of the molecule or inside of it (for interior walks)
- Efficiently determining the closest sphere to a given exterior point (for walks outside molecule)
- Efficiently determining if a query point is inside of the convex hull of the molecule
- Efficiently finding the largest possible sphere enclosing a query point for external walks

Geometry: Problem Descriptions

There are many geometric problems that arise in this algorithm:

- Efficiently determining if a point is on the surface of the molecule or inside of it (for interior walks)
- Efficiently determining the closest sphere to a given exterior point (for walks outside molecule)
- Efficiently determining if a query point is inside of the convex hull of the molecule
- Efficiently finding the largest possible sphere enclosing a query point for external walks

Geometry: Problem Descriptions

There are many geometric problems that arise in this algorithm:

- Efficiently determining if a point is on the surface of the molecule or inside of it (for interior walks)
- Efficiently determining the closest sphere to a given exterior point (for walks outside molecule)
- Efficiently determining if a query point is inside of the convex hull of the molecule
- Efficiently finding the largest possible sphere enclosing a query point for external walks

Geometry: Problem Descriptions

There are many geometric problems that arise in this algorithm:

- Efficiently determining if a point is on the surface of the molecule or inside of it (for interior walks)
- Efficiently determining the closest sphere to a given exterior point (for walks outside molecule)
- Efficiently determining if a query point is inside of the convex hull of the molecule
- Efficiently finding the largest possible sphere enclosing a query point for external walks

Correlated and Uncorrelated Sampling

- Correlated sampling in Monte Carlo is essential for two important reasons
  1. To obtain smooth curves with a minimum of sampling (function-wise vs. point-wise sampling)
  2. To obtain accurate results from quantities defined as the differences of Monte Carlo estimates
- With this correlated sampling sampling you can get a “smooth curve" with three orders of magnitude less sampling, note: you still have $O(N^{-1/2})$ errors, just in “curve space," not point by point

Correlated and Uncorrelated Sampling

- Correlated sampling in Monte Carlo is essential for two important reasons
  1. To obtain smooth curves with a minimum of sampling (function-wise vs. point-wise sampling)
  2. To obtain accurate results from quantities defined as the differences of Monte Carlo estimates
- With this correlated sampling sampling you can get a “smooth curve" with three orders of magnitude less sampling, note: you still have $O(N^{-1/2})$ errors, just in “curve space," not point by point

Correlated and Uncorrelated Sampling

- Correlated sampling in Monte Carlo is essential for two important reasons
  1. To obtain smooth curves with a minimum of sampling (function-wise vs. point-wise sampling)
  2. To obtain accurate results from quantities defined as the differences of Monte Carlo estimates
- With this correlated sampling sampling you can get a “smooth curve" with three orders of magnitude less sampling, note: you still have $O(N^{-1/2})$ errors, just in “curve space," not point by point

Correlated and Uncorrelated Sampling

- Correlated sampling in Monte Carlo is essential for two important reasons
  1. To obtain smooth curves with a minimum of sampling (function-wise vs. point-wise sampling)
  2. To obtain accurate results from quantities defined as the differences of Monte Carlo estimates
- With this correlated sampling sampling you can get a “smooth curve" with three orders of magnitude less sampling, note: you still have $O(N^{-1/2})$ errors, just in “curve space," not point by point

Correlated Sampling: Salt Concentration

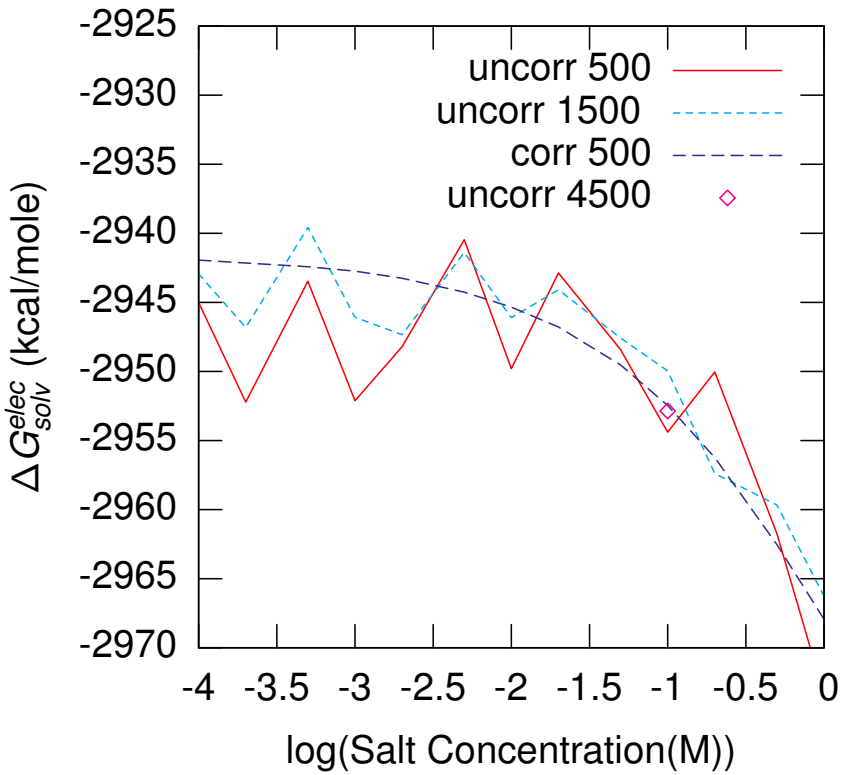


Figure: Electrostatic Solvation free Energy of `3icb` calculated with three four conditions: uncorrelated sampling with 500 number of trajectories per concentration, uncorrelated sampling with 1500 number of trajectories per concentration, uncorrelated sampling with 4500 number of iterations, and correlated sampling with 500 number of trajectories

Dependence on Salt Concentration

- Values of scalar energies as a function of external salt concentration are important
  1. Smooth curves of internal energy vs. salt concentration (see above)
  2. Numerical estimate of the derivative as salt concentration vanishes
- For κ used in simulations, $F_j(\kappa) = 1$
- For an arbitrary $\kappa' > \kappa$:

  $F_j(\kappa')$ is multiplied by the ratio $\dfrac{q(\kappa', d)}{q(\kappa, d)}$ on every step of the WOS in the exterior
- The results obtained with the estimates (23) and (24) for different values of κ are **highly correlated**

Dependence on Salt Concentration

- Values of scalar energies as a function of external salt concentration are important
  1. Smooth curves of internal energy vs. salt concentration (see above)
  2. Numerical estimate of the derivative as salt concentration vanishes
- For κ used in simulations, $F_j(\kappa) = 1$
- For an arbitrary $\kappa' > \kappa$:

  $F_j(\kappa')$ is multiplied by the ratio $\dfrac{q(\kappa', d)}{q(\kappa, d)}$ on every step of the WOS in the exterior
- The results obtained with the estimates (23) and (24) for different values of κ are **highly correlated**

Dependence on Salt Concentration

- Values of scalar energies as a function of external salt concentration are important
  1. Smooth curves of internal energy vs. salt concentration (see above)
  2. Numerical estimate of the derivative as salt concentration vanishes
- For κ used in simulations, $F_j(\kappa) = 1$
- For an arbitrary $\kappa' > \kappa$:

  $F_j(\kappa')$ is multiplied by the ratio $\dfrac{q(\kappa', d)}{q(\kappa, d)}$ on every step of the WOS in the exterior
- The results obtained with the estimates (23) and (24) for different values of κ are **highly correlated**

Dependence on Salt Concentration

- Values of scalar energies as a function of external salt concentration are important
  1. Smooth curves of internal energy vs. salt concentration (see above)
  2. Numerical estimate of the derivative as salt concentration vanishes
- For κ used in simulations, $F_j(\kappa) = 1$
- For an arbitrary $\kappa' > \kappa$:

  $F_j(\kappa')$ is multiplied by the ratio $\dfrac{q(\kappa', d)}{q(\kappa, d)}$ on every step of the WOS in the exterior
- The results obtained with the estimates (23) and (24) for different values of κ are **highly correlated**

Dependence on Salt Concentration

- Values of scalar energies as a function of external salt concentration are important
  1. Smooth curves of internal energy vs. salt concentration (see above)
  2. Numerical estimate of the derivative as salt concentration vanishes
- For κ used in simulations, $F_j(\kappa) = 1$
- For an arbitrary $\kappa' > \kappa$:

  $F_j(\kappa')$ is multiplied by the ratio $\dfrac{q(\kappa', d)}{q(\kappa, d)}$ on every step of the WOS in the exterior
- The results obtained with the estimates (23) and (24) for different values of κ are **highly correlated**

Dependence on Salt Concentration

- Values of scalar energies as a function of external salt concentration are important
  1. Smooth curves of internal energy vs. salt concentration (see above)
  2. Numerical estimate of the derivative as salt concentration vanishes
- For κ used in simulations, $F_j(\kappa) = 1$
- For an arbitrary $\kappa' > \kappa$:

  $F_j(\kappa')$ is multiplied by the ratio $\dfrac{q(\kappa', d)}{q(\kappa, d)}$ on every step of the WOS in the exterior
- The results obtained with the estimates (23) and (24) for different values of κ are **highly correlated**

Correlated Sampling: Binding Calculations

- Binding computation requires three energy computations $E(A+B) - E(A) - E(B)$
- Monte Carlo requires “help" when differencing
- We use the reproducibility in SPRNG to do this effectively
  1. Unbound: when exiting the molecule the seed is stored using SPRNG tools
  2. Bound: walks resume at the exit points with the same random number streams and reusing
  3. At this exit point, only the exit point information is required
- The leads to correlation between unbound and bound energy computations that decreases as the walk length increases (κ^2 decreases)

Correlated Sampling: Binding Calculations

- Binding computation requires three energy computations $E(A+B) - E(A) - E(B)$
- Monte Carlo requires “help" when differencing
- We use the reproducibility in SPRNG to do this effectively
  1. Unbound: when exiting the molecule the seed is stored using SPRNG tools
  2. Bound: walks resume at the exit points with the same random number streams and reusing
  3. At this exit point, only the exit point information is required
- The leads to correlation between unbound and bound energy computations that decreases as the walk length increases (κ^2 decreases)

Correlated Sampling: Binding Calculations

- Binding computation requires three energy computations $E(A+B) - E(A) - E(B)$
- Monte Carlo requires “help" when differencing
- We use the reproducibility in SPRNG to do this effectively
  1. Unbound: when exiting the molecule the seed is stored using SPRNG tools
  2. Bound: walks resume at the exit points with the same random number streams and reusing
  3. At this exit point, only the exit point information is required
- The leads to correlation between unbound and bound energy computations that decreases as the walk length increases (κ^2 decreases)

Correlated Sampling: Binding Calculations

- Binding computation requires three energy computations $E(A+B) - E(A) - E(B)$
- Monte Carlo requires “help" when differencing
- We use the reproducibility in `SPRNG` to do this effectively
  1. Unbound: when exiting the molecule the seed is stored using `SPRNG` tools
  2. Bound: walks resume at the exit points with the same random number streams and reusing
  3. At this exit point, only the exit point information is required
- The leads to correlation between unbound and bound energy computations that decreases as the walk length increases (κ^2 decreases)

Correlated Sampling: Binding Calculations

- Binding computation requires three energy computations $E(A+B) - E(A) - E(B)$
- Monte Carlo requires “help" when differencing
- We use the reproducibility in `SPRNG` to do this effectively
  1. Unbound: when exiting the molecule the seed is stored using `SPRNG` tools
  2. Bound: walks resume at the exit points with the same random number streams and reusing
  3. At this exit point, only the exit point information is required
- The leads to correlation between unbound and bound energy computations that decreases as the walk length increases (κ^2 decreases)

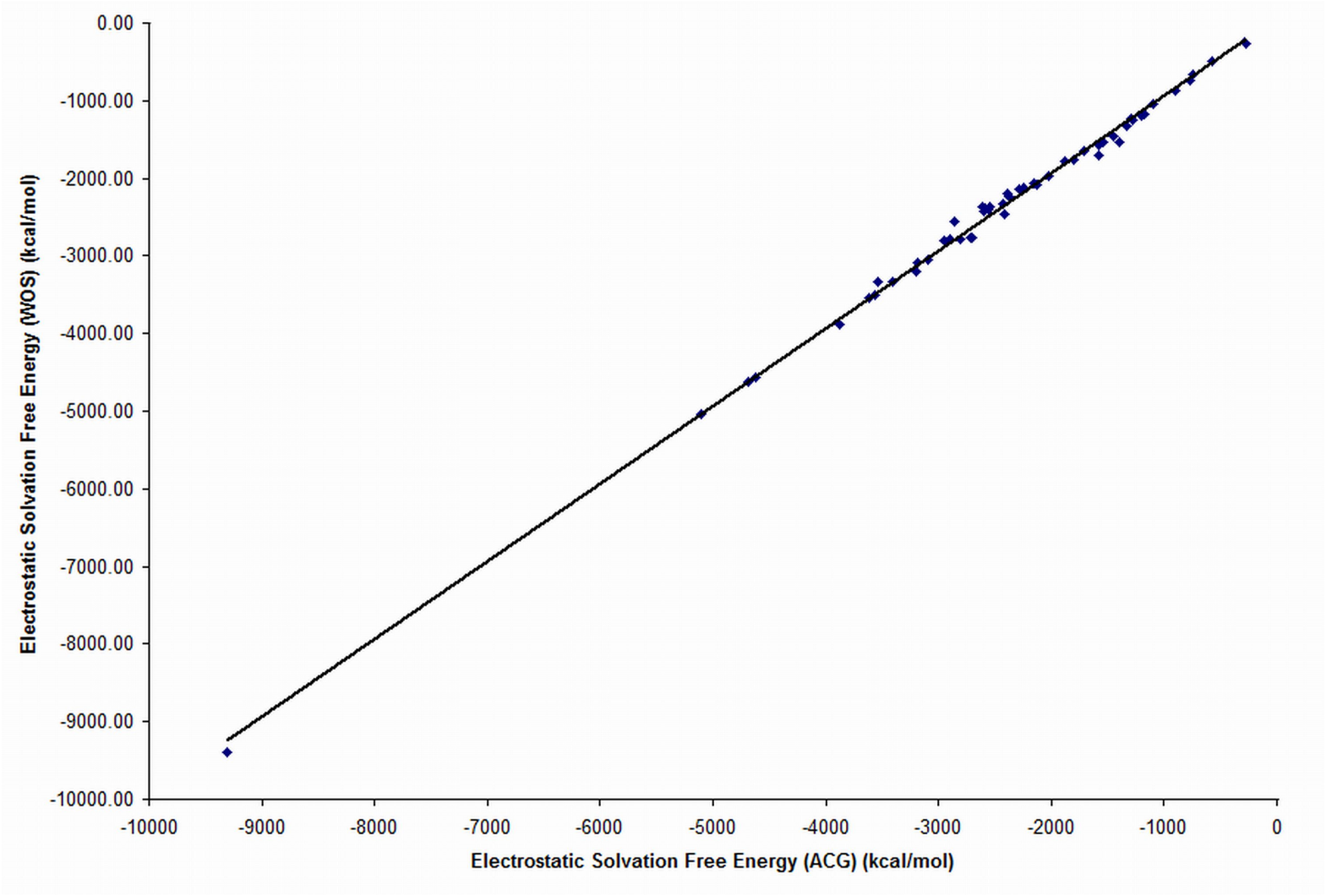
Correlated Sampling: Binding Calculations

- Binding computation requires three energy computations $E(A+B) - E(A) - E(B)$
- Monte Carlo requires “help" when differencing
- We use the reproducibility in `SPRNG` to do this effectively
  1. Unbound: when exiting the molecule the seed is stored using `SPRNG` tools
  2. Bound: walks resume at the exit points with the same random number streams and reusing
  3. At this exit point, only the exit point information is required
- The leads to correlation between unbound and bound energy computations that decreases as the walk length increases (κ^2 decreases)

Correlated Sampling: Binding Calculations

- Binding computation requires three energy computations $E(A+B) - E(A) - E(B)$
- Monte Carlo requires “help" when differencing
- We use the reproducibility in `SPRNG` to do this effectively
  1. Unbound: when exiting the molecule the seed is stored using `SPRNG` tools
  2. Bound: walks resume at the exit points with the same random number streams and reusing
  3. At this exit point, only the exit point information is required
- The leads to correlation between unbound and bound energy computations that decreases as the walk length increases (κ^2 decreases)

Accuracy: Monte Carlo vs. Deterministic



Sampling Error and Bias

- In Monte Carlo there are biases (errors) and sampling error
  1. Sampling error is based on standard error $O(N^{-1/2})$
  2. Difference between expected value and PDE solution is bias
     - Capture thickness (ϵ): bias is $O(\epsilon)$
     - Auxiliary sphere radius (a): bias is $O(a^3)$
     - Effective Van der Waals sphere radius, R
     - Overall bias: $\left(\frac{a}{2R}\right)^3 + \left(\frac{\epsilon}{2R}\right)$
  3. $Var[\sum_i q_i \Phi(x_i)] = \sum_i q_i^2 Var[\Phi(x_i)]$
  4. Given a desired variance, divide it evenly over this sum
  5. Running time $\propto \frac{|\ln(\epsilon)|}{a}$
  6. Can reduce running time by 2 orders of magnitude by bias/variance balancing and using larger ϵ, a and ANN
  7. Large ANN means errors in drawing the largest sphere outside the molecule for WOS

Sampling Error and Bias

- In Monte Carlo there are biases (errors) and sampling error
  1. Sampling error is based on standard error $O(N^{-1/2})$
  2. Difference between expected value and PDE solution is bias
     - Capture thickness (ϵ): bias is $O(\epsilon)$
     - Auxiliary sphere radius (a): bias is $O(a^3)$
     - Effective Van der Waals sphere radius, R
     - Overall bias: $\left(\frac{a}{2R}\right)^3 + \left(\frac{\epsilon}{2R}\right)$
  3. $Var[\sum_i q_i \Phi(x_i)] = \sum_i q_i^2 Var[\Phi(x_i)]$
  4. Given a desired variance, divide it evenly over this sum
  5. Running time $\propto \frac{|\ln(\epsilon)|}{a}$
  6. Can reduce running time by 2 orders of magnitude by bias/variance balancing and using larger ϵ, a and ANN
  7. Large ANN means errors in drawing the largest sphere outside the molecule for WOS

Sampling Error and Bias

- In Monte Carlo there are biases (errors) and sampling error
  1. Sampling error is based on standard error $O(N^{-1/2})$
  2. Difference between expected value and PDE solution is bias
     - Capture thickness (ϵ): bias is $O(\epsilon)$
     - Auxiliary sphere radius (a): bias is $O(a^3)$
     - Effective Van der Waals sphere radius, R
     - Overall bias: $\left(\frac{a}{2R}\right)^3 + \left(\frac{\epsilon}{2R}\right)$
  3. $Var[\sum_i q_i \Phi(x_i)] = \sum_i q_i^2 Var[\Phi(x_i)]$
  4. Given a desired variance, divide it evenly over this sum
  5. Running time $\propto \frac{|\ln(\epsilon)|}{a}$
  6. Can reduce running time by 2 orders of magnitude by bias/variance balancing and using larger ϵ, a and ANN
  7. Large ANN means errors in drawing the largest sphere outside the molecule for WOS

Sampling Error and Bias

- In Monte Carlo there are biases (errors) and sampling error
  1. Sampling error is based on standard error $O(N^{-1/2})$
  2. Difference between expected value and PDE solution is bias
     - Capture thickness (ϵ): bias is $O(\epsilon)$
     - Auxiliary sphere radius (a): bias is $O(a^3)$
     - Effective Van der Waals sphere radius, R
     - Overall bias: $\left(\frac{a}{2R}\right)^3 + \left(\frac{\epsilon}{2R}\right)$
  3. $Var[\sum_i q_i \Phi(x_i)] = \sum_i q_i^2 Var[\Phi(x_i)]$
  4. Given a desired variance, divide it evenly over this sum
  5. Running time $\propto \frac{|\ln(\epsilon)|}{a}$
  6. Can reduce running time by 2 orders of magnitude by bias/variance balancing and using larger ϵ, a and ANN
  7. Large ANN means errors in drawing the largest sphere outside the molecule for WOS

Sampling Error and Bias

- In Monte Carlo there are biases (errors) and sampling error
  1. Sampling error is based on standard error $O(N^{-1/2})$
  2. Difference between expected value and PDE solution is bias
     - Capture thickness (ϵ): bias is $O(\epsilon)$
     - Auxiliary sphere radius (a): bias is $O(a^3)$
     - Effective Van der Waals sphere radius, R
     - Overall bias: $\left(\frac{a}{2R}\right)^3 + \left(\frac{\epsilon}{2R}\right)$
  3. $Var[\sum_i q_i \Phi(x_i)] = \sum_i q_i^2 Var[\Phi(x_i)]$
  4. Given a desired variance, divide it evenly over this sum
  5. Running time $\propto \frac{|\ln(\epsilon)|}{a}$
  6. Can reduce running time by 2 orders of magnitude by bias/variance balancing and using larger ϵ, a and ANN
  7. Large ANN means errors in drawing the largest sphere outside the molecule for WOS

Sampling Error and Bias

- In Monte Carlo there are biases (errors) and sampling error
  1. Sampling error is based on standard error $O(N^{-1/2})$
  2. Difference between expected value and PDE solution is bias
     - Capture thickness (ϵ): bias is $O(\epsilon)$
     - Auxiliary sphere radius (a): bias is $O(a^3)$
     - Effective Van der Waals sphere radius, R
     - Overall bias: $\left(\frac{a}{2R}\right)^3 + \left(\frac{\epsilon}{2R}\right)$
  3. $Var[\sum_i q_i \Phi(x_i)] = \sum_i q_i^2 Var[\Phi(x_i)]$
  4. Given a desired variance, divide it evenly over this sum
  5. Running time $\propto \frac{|\ln(\epsilon)|}{a}$
  6. Can reduce running time by 2 orders of magnitude by bias/variance balancing and using larger ϵ, a and ANN
  7. Large ANN means errors in drawing the largest sphere outside the molecule for WOS

Sampling Error and Bias

- In Monte Carlo there are biases (errors) and sampling error
  1. Sampling error is based on standard error $O(N^{-1/2})$
  2. Difference between expected value and PDE solution is bias
     - Capture thickness (ϵ): bias is $O(\epsilon)$
     - Auxiliary sphere radius (a): bias is $O(a^3)$
     - Effective Van der Waals sphere radius, R
     - Overall bias: $\left(\frac{a}{2R}\right)^3 + \left(\frac{\epsilon}{2R}\right)$
  3. $Var[\sum_i q_i \Phi(x_i)] = \sum_i q_i^2 Var[\Phi(x_i)]$
  4. Given a desired variance, divide it evenly over this sum
  5. Running time $\propto \frac{|\ln(\epsilon)|}{a}$
  6. Can reduce running time by 2 orders of magnitude by bias/variance balancing and using larger ϵ, a and ANN
  7. Large ANN means errors in drawing the largest sphere outside the molecule for WOS

Sampling Error and Bias

- In Monte Carlo there are biases (errors) and sampling error
  1. Sampling error is based on standard error $O(N^{-1/2})$
  2. Difference between expected value and PDE solution is bias
     - Capture thickness (ϵ): bias is $O(\epsilon)$
     - Auxiliary sphere radius (a): bias is $O(a^3)$
     - Effective Van der Waals sphere radius, R
     - Overall bias: $\left(\frac{a}{2R}\right)^3 + \left(\frac{\epsilon}{2R}\right)$
  3. $Var[\sum_i q_i \Phi(x_i)] = \sum_i q_i^2 Var[\Phi(x_i)]$
  4. Given a desired variance, divide it evenly over this sum
  5. Running time $\propto \frac{|\ln(\epsilon)|}{a}$
  6. Can reduce running time by 2 orders of magnitude by bias/variance balancing and using larger ϵ, a and ANN
  7. Large ANN means errors in drawing the largest sphere outside the molecule for WOS

Sampling Error and Bias

- In Monte Carlo there are biases (errors) and sampling error
  1. Sampling error is based on standard error $O(N^{-1/2})$
  2. Difference between expected value and PDE solution is bias
     - Capture thickness (ϵ): bias is $O(\epsilon)$
     - Auxiliary sphere radius (a): bias is $O(a^3)$
     - Effective Van der Waals sphere radius, R
     - Overall bias: $\left(\frac{a}{2R}\right)^3 + \left(\frac{\epsilon}{2R}\right)$
  3. $Var[\sum_i q_i \Phi(x_i)] = \sum_i q_i^2 Var[\Phi(x_i)]$
  4. Given a desired variance, divide it evenly over this sum
  5. Running time $\propto \frac{|\ln(\epsilon)|}{a}$
  6. Can reduce running time by 2 orders of magnitude by bias/variance balancing and using larger ϵ, a and ANN
  7. Large ANN means errors in drawing the largest sphere outside the molecule for WOS

Sampling Error and Bias

- In Monte Carlo there are biases (errors) and sampling error
  1. Sampling error is based on standard error $O(N^{-1/2})$
  2. Difference between expected value and PDE solution is bias
     - Capture thickness (ϵ): bias is $O(\epsilon)$
     - Auxiliary sphere radius (a): bias is $O(a^3)$
     - Effective Van der Waals sphere radius, R
     - Overall bias: $\left(\frac{a}{2R}\right)^3 + \left(\frac{\epsilon}{2R}\right)$
  3. $Var[\sum_i q_i \Phi(x_i)] = \sum_i q_i^2 Var[\Phi(x_i)]$
  4. Given a desired variance, divide it evenly over this sum
  5. Running time $\propto \frac{|\ln(\epsilon)|}{a}$
  6. Can reduce running time by 2 orders of magnitude by bias/variance balancing and using larger ϵ, a and ANN
  7. Large ANN means errors in drawing the largest sphere outside the molecule for WOS

Sampling Error and Bias

- In Monte Carlo there are biases (errors) and sampling error
  1. Sampling error is based on standard error $O(N^{-1/2})$
  2. Difference between expected value and PDE solution is bias
     - Capture thickness (ϵ): bias is $O(\epsilon)$
     - Auxiliary sphere radius (a): bias is $O(a^3)$
     - Effective Van der Waals sphere radius, R
     - Overall bias: $\left(\frac{a}{2R}\right)^3 + \left(\frac{\epsilon}{2R}\right)$
  3. $Var[\sum_i q_i \Phi(x_i)] = \sum_i q_i^2 Var[\Phi(x_i)]$
  4. Given a desired variance, divide it evenly over this sum
  5. Running time $\propto \frac{|\ln(\epsilon)|}{a}$
  6. Can reduce running time by 2 orders of magnitude by bias/variance balancing and using larger ϵ, a and *ANN*
  7. Large *ANN* means errors in drawing the largest sphere outside the molecule for WOS

Sampling Error and Bias

- In Monte Carlo there are biases (errors) and sampling error
  1. Sampling error is based on standard error $O(N^{-1/2})$
  2. Difference between expected value and PDE solution is bias
     - Capture thickness (ϵ): bias is $O(\epsilon)$
     - Auxiliary sphere radius (a): bias is $O(a^3)$
     - Effective Van der Waals sphere radius, R
     - Overall bias: $\left(\frac{a}{2R}\right)^3 + \left(\frac{\epsilon}{2R}\right)$
  3. $Var[\sum_i q_i \Phi(x_i)] = \sum_i q_i^2 Var[\Phi(x_i)]$
  4. Given a desired variance, divide it evenly over this sum
  5. Running time $\propto \frac{|\ln(\epsilon)|}{a}$
  6. Can reduce running time by 2 orders of magnitude by bias/variance balancing and using larger ϵ, a and ANN
  7. Large ANN means errors in drawing the largest sphere outside the molecule for WOS

Timing: Better Than Expected

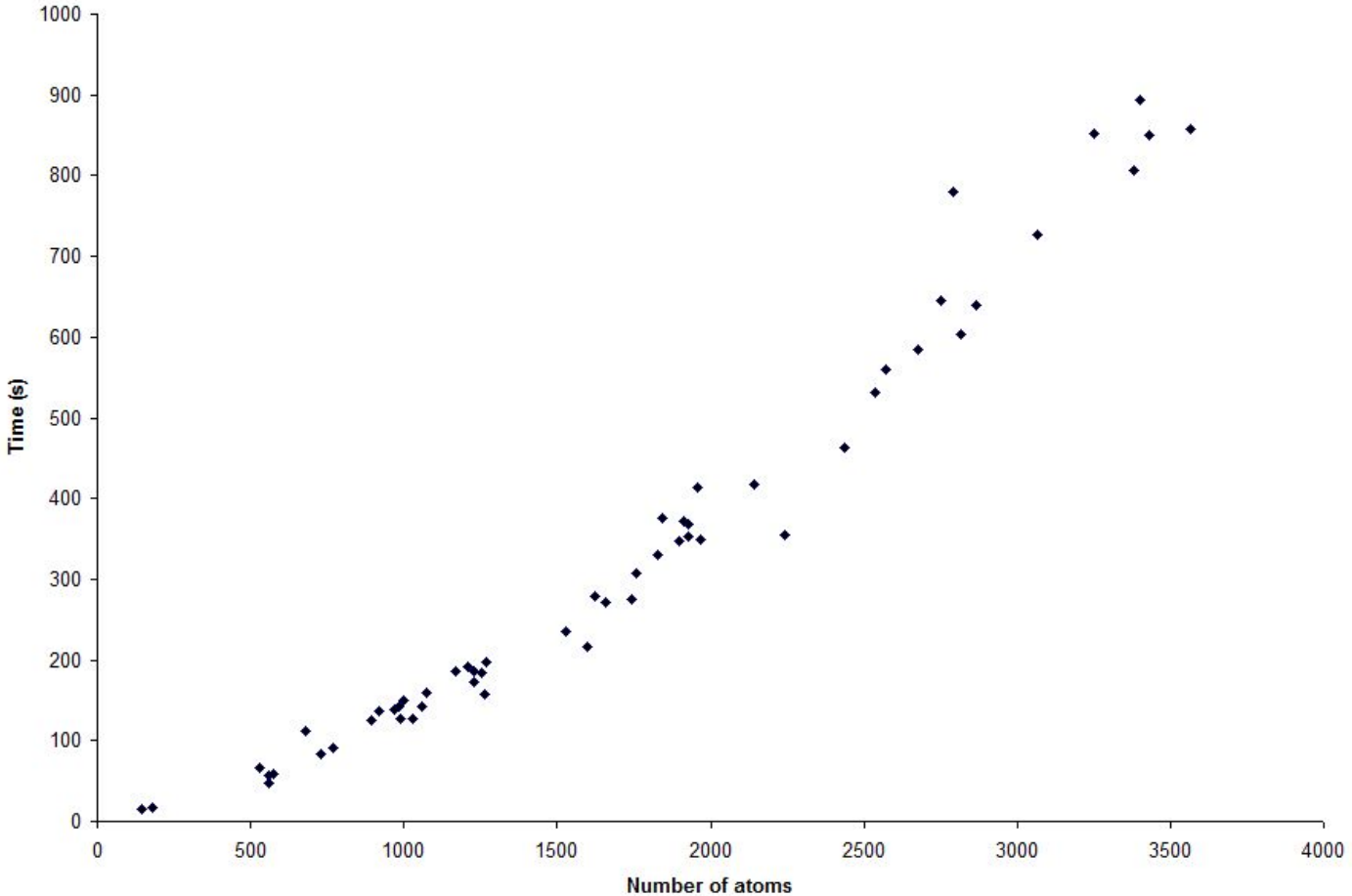


Figure: $O(M \log M)$?

Conclusions

- Over the years we have developed many MC tools for PDEs and more recently:
- We have developed a novel stochastic linear PBE solver that can provide highly accurate salt-dependent electrostatic properties of biomolecules in a single PBE calculation
- Advantages of the stochastic linear PBE solver over the more mature deterministic methods include: the subtle geometric features of the biomolecule can be treated with higher precision, the continuity and outer boundary conditions are accounted for exactly, a singularity free scheme is employed and straightforward implementation on parallel computer platform is possible
- Codes provide higher accuracy (on demand) and do not suffer losses in accuracy near the boundary
- Only way to handle large ($M >> 10000$) molecules

Conclusions

- Over the years we have developed many MC tools for PDEs and more recently:
- We have developed a novel stochastic linear PBE solver that can provide highly accurate salt-dependent electrostatic properties of biomolecules in a single PBE calculation
- Advantages of the stochastic linear PBE solver over the more mature deterministic methods include: the subtle geometric features of the biomolecule can be treated with higher precision, the continuity and outer boundary conditions are accounted for exactly, a singularity free scheme is employed and straightforward implementation on parallel computer platform is possible
- Codes provide higher accuracy (on demand) and do not suffer losses in accuracy near the boundary
- Only way to handle large ($M >> 10000$) molecules

Conclusions

- Over the years we have developed many MC tools for PDEs and more recently:
- We have developed a novel stochastic linear PBE solver that can provide highly accurate salt-dependent electrostatic properties of biomolecules in a single PBE calculation
- Advantages of the stochastic linear PBE solver over the more mature deterministic methods include: the subtle geometric features of the biomolecule can be treated with higher precision, the continuity and outer boundary conditions are accounted for exactly, a singularity free scheme is employed and straightforward implementation on parallel computer platform is possible
- Codes provide higher accuracy (on demand) and do not suffer losses in accuracy near the boundary
- Only way to handle large ($M >> 10000$) molecules

Conclusions

- Over the years we have developed many MC tools for PDEs and more recently:
- We have developed a novel stochastic linear PBE solver that can provide highly accurate salt-dependent electrostatic properties of biomolecules in a single PBE calculation
- Advantages of the stochastic linear PBE solver over the more mature deterministic methods include: the subtle geometric features of the biomolecule can be treated with higher precision, the continuity and outer boundary conditions are accounted for exactly, a singularity free scheme is employed and straightforward implementation on parallel computer platform is possible
- Codes provide higher accuracy (on demand) and do not suffer losses in accuracy near the boundary
- Only way to handle large ($M >> 10000$) molecules

Conclusions

- Over the years we have developed many MC tools for PDEs and more recently:
- We have developed a novel stochastic linear PBE solver that can provide highly accurate salt-dependent electrostatic properties of biomolecules in a single PBE calculation
- Advantages of the stochastic linear PBE solver over the more mature deterministic methods include: the subtle geometric features of the biomolecule can be treated with higher precision, the continuity and outer boundary conditions are accounted for exactly, a singularity free scheme is employed and straightforward implementation on parallel computer platform is possible
- Codes provide higher accuracy (on demand) and do not suffer losses in accuracy near the boundary
- Only way to handle large ($M >> 10000$) molecules

Future Work

- Binding computations: using correlated sampling by directly reprocessing walks
- Simple code interface for distribution with
  1. Desired accuracy as input that allows a precalculation of the number of needed trajectories
  2. Importance sampling for optimal estimation of scalar energy values
  3. Built-in CONDOR support for distribution of concurrent tasks
  4. Multicore distributed computing support for the code: OpenMP/OpenMPI
  5. Precompiled code module distribution to protect IP
  6. Webpage to describe the method and the mathematical background and application
- Exploit the implicit inverse computation this methods provides
  1. Can do computation without knowing charges until the end (an inverse)
  2. Simple to examine many charge distributions in a perfectly correlated setting

Future Work

- Binding computations: using correlated sampling by directly reprocessing walks
- Simple code interface for distribution with
  1. Desired accuracy as input that allows a precalculation of the number of needed trajectories
  2. Importance sampling for optimal estimation of scalar energy values
  3. Built-in CONDOR support for distribution of concurrent tasks
  4. Multicore distributed computing support for the code: OpenMP/OpenMPI
  5. Precompiled code module distribution to protect IP
  6. Webpage to describe the method and the mathematical background and application
- Exploit the implicit inverse computation this methods provides
  1. Can do computation without knowing charges until the end (an inverse)
  2. Simple to examine many charge distributions in a perfectly correlated setting

Future Work

- Binding computations: using correlated sampling by directly reprocessing walks
- Simple code interface for distribution with
  1. Desired accuracy as input that allows a precalculation of the number of needed trajectories
  2. Importance sampling for optimal estimation of scalar energy values
  3. Built-in CONDOR support for distribution of concurrent tasks
  4. Multicore distributed computing support for the code: OpenMP/OpenMPI
  5. Precompiled code module distribution to protect IP
  6. Webpage to describe the method and the mathematical background and application
- Exploit the implicit inverse computation this methods provides
  1. Can do computation without knowing charges until the end (an inverse)
  2. Simple to examine many charge distributions in a perfectly correlated setting

Future Work

- Binding computations: using correlated sampling by directly reprocessing walks
- Simple code interface for distribution with
  1. Desired accuracy as input that allows a precalculation of the number of needed trajectories
  2. Importance sampling for optimal estimation of scalar energy values
  3. Built-in CONDOR support for distribution of concurrent tasks
  4. Multicore distributed computing support for the code: OpenMP/OpenMPI
  5. Precompiled code module distribution to protect IP
  6. Webpage to describe the method and the mathematical background and application
- Exploit the implicit inverse computation this methods provides
  1. Can do computation without knowing charges until the end (an inverse)
  2. Simple to examine many charge distributions in a perfectly correlated setting

Future Work

- Binding computations: using correlated sampling by directly reprocessing walks
- Simple code interface for distribution with
  1. Desired accuracy as input that allows a precalculation of the number of needed trajectories
  2. Importance sampling for optimal estimation of scalar energy values
  3. Built-in `CONDOR` support for distribution of concurrent tasks
  4. Multicore distributed computing support for the code: OpenMP/OpenMPI
  5. Precompiled code module distribution to protect IP
  6. Webpage to describe the method and the mathematical background and application
- Exploit the implicit inverse computation this methods provides
  1. Can do computation without knowing charges until the end (an inverse)
  2. Simple to examine many charge distributions in a perfectly correlated setting

Future Work

- Binding computations: using correlated sampling by directly reprocessing walks
- Simple code interface for distribution with
  1. Desired accuracy as input that allows a precalculation of the number of needed trajectories
  2. Importance sampling for optimal estimation of scalar energy values
  3. Built-in `CONDOR` support for distribution of concurrent tasks
  4. Multicore distributed computing support for the code: OpenMP/OpenMPI
  5. Precompiled code module distribution to protect IP
  6. Webpage to describe the method and the mathematical background and application
- Exploit the implicit inverse computation this methods provides
  1. Can do computation without knowing charges until the end (an inverse)
  2. Simple to examine many charge distributions in a perfectly correlated setting

Future Work

- Binding computations: using correlated sampling by directly reprocessing walks
- Simple code interface for distribution with
  1. Desired accuracy as input that allows a precalculation of the number of needed trajectories
  2. Importance sampling for optimal estimation of scalar energy values
  3. Built-in `CONDOR` support for distribution of concurrent tasks
  4. Multicore distributed computing support for the code: OpenMP/OpenMPI
  5. Precompiled code module distribution to protect IP
  6. Webpage to describe the method and the mathematical background and application
- Exploit the implicit inverse computation this methods provides
  1. Can do computation without knowing charges until the end (an inverse)
  2. Simple to examine many charge distributions in a perfectly correlated setting

Future Work

- Binding computations: using correlated sampling by directly reprocessing walks
- Simple code interface for distribution with
  1. Desired accuracy as input that allows a precalculation of the number of needed trajectories
  2. Importance sampling for optimal estimation of scalar energy values
  3. Built-in `CONDOR` support for distribution of concurrent tasks
  4. Multicore distributed computing support for the code: OpenMP/OpenMPI
  5. Precompiled code module distribution to protect IP
  6. Webpage to describe the method and the mathematical background and application
- Exploit the implicit inverse computation this methods provides
  1. Can do computation without knowing charges until the end (an inverse)
  2. Simple to examine many charge distributions in a perfectly correlated setting

Future Work

- Binding computations: using correlated sampling by directly reprocessing walks
- Simple code interface for distribution with
  1. Desired accuracy as input that allows a precalculation of the number of needed trajectories
  2. Importance sampling for optimal estimation of scalar energy values
  3. Built-in `CONDOR` support for distribution of concurrent tasks
  4. Multicore distributed computing support for the code: OpenMP/OpenMPI
  5. Precompiled code module distribution to protect IP
  6. Webpage to describe the method and the mathematical background and application
- Exploit the implicit inverse computation this methods provides
  1. Can do computation without knowing charges until the end (an inverse)
  2. Simple to examine many charge distributions in a perfectly correlated setting

Future Work

- Binding computations: using correlated sampling by directly reprocessing walks
- Simple code interface for distribution with
  1. Desired accuracy as input that allows a precalculation of the number of needed trajectories
  2. Importance sampling for optimal estimation of scalar energy values
  3. Built-in `CONDOR` support for distribution of concurrent tasks
  4. Multicore distributed computing support for the code: OpenMP/OpenMPI
  5. Precompiled code module distribution to protect IP
  6. Webpage to describe the method and the mathematical background and application
- Exploit the implicit inverse computation this methods provides
  1. Can do computation without knowing charges until the end (an inverse)
  2. Simple to examine many charge distributions in a perfectly correlated setting

Future Work

- Binding computations: using correlated sampling by directly reprocessing walks
- Simple code interface for distribution with
  1. Desired accuracy as input that allows a precalculation of the number of needed trajectories
  2. Importance sampling for optimal estimation of scalar energy values
  3. Built-in `CONDOR` support for distribution of concurrent tasks
  4. Multicore distributed computing support for the code: OpenMP/OpenMPI
  5. Precompiled code module distribution to protect IP
  6. Webpage to describe the method and the mathematical background and application
- Exploit the implicit inverse computation this methods provides
  1. Can do computation without knowing charges until the end (an inverse)
  2. Simple to examine many charge distributions in a perfectly correlated setting

Future Work

- Further algorithmic development
  1. Computation of gradients using existing Markov chains
  2. Global computation of field variables and their visualization
  3. Nonlinear BVPs perhaps via branching processes
  4. Using “Walk-on-the-Boundary" (WOB) techniques
- Geometric Issues
  1. Computation of the three region model problem
  2. More complicated surfaces (solvent-excluded and ion-excluded)
  3. Accuracy issues related to the Van der Waals surface
- Optimize the performance
  1. Error/bias/variance balancing
  2. Importance sampling and the outer walks
  3. WOB to eliminate walks outside
  4. QMC methods

Future Work

- Further algorithmic development
  1. Computation of gradients using existing Markov chains
  2. Global computation of field variables and their visualization
  3. Nonlinear BVPs perhaps via branching processes
  4. Using “Walk-on-the-Boundary" (WOB) techniques
- Geometric Issues
  1. Computation of the three region model problem
  2. More complicated surfaces (solvent-excluded and ion-excluded)
  3. Accuracy issues related to the Van der Waals surface
- Optimize the performance
  1. Error/bias/variance balancing
  2. Importance sampling and the outer walks
  3. WOB to eliminate walks outside
  4. QMC methods

Future Work

- Further algorithmic development
  1. Computation of gradients using existing Markov chains
  2. Global computation of field variables and their visualization
  3. Nonlinear BVPs perhaps via branching processes
  4. Using “Walk-on-the-Boundary" (WOB) techniques
- Geometric Issues
  1. Computation of the three region model problem
  2. More complicated surfaces (solvent-excluded and ion-excluded)
  3. Accuracy issues related to the Van der Waals surface
- Optimize the performance
  1. Error/bias/variance balancing
  2. Importance sampling and the outer walks
  3. WOB to eliminate walks outside
  4. QMC methods

Future Work

- Further algorithmic development
  1. Computation of gradients using existing Markov chains
  2. Global computation of field variables and their visualization
  3. Nonlinear BVPs perhaps via branching processes
  4. Using “Walk-on-the-Boundary" (WOB) techniques
- Geometric Issues
  1. Computation of the three region model problem
  2. More complicated surfaces (solvent-excluded and ion-excluded)
  3. Accuracy issues related to the Van der Waals surface
- Optimize the performance
  1. Error/bias/variance balancing
  2. Importance sampling and the outer walks
  3. WOB to eliminate walks outside
  4. QMC methods

Future Work

- Further algorithmic development
  1. Computation of gradients using existing Markov chains
  2. Global computation of field variables and their visualization
  3. Nonlinear BVPs perhaps via branching processes
  4. Using “Walk-on-the-Boundary" (WOB) techniques
- Geometric Issues
  1. Computation of the three region model problem
  2. More complicated surfaces (solvent-excluded and ion-excluded)
  3. Accuracy issues related to the Van der Waals surface
- Optimize the performance
  1. Error/bias/variance balancing
  2. Importance sampling and the outer walks
  3. WOB to eliminate walks outside
  4. QMC methods

Future Work

- Further algorithmic development
  1. Computation of gradients using existing Markov chains
  2. Global computation of field variables and their visualization
  3. Nonlinear BVPs perhaps via branching processes
  4. Using “Walk-on-the-Boundary" (WOB) techniques
- Geometric Issues
  1. Computation of the three region model problem
  2. More complicated surfaces (solvent-excluded and ion-excluded)
  3. Accuracy issues related to the Van der Waals surface
- Optimize the performance
  1. Error/bias/variance balancing
  2. Importance sampling and the outer walks
  3. WOB to eliminate walks outside
  4. QMC methods

Future Work

- Further algorithmic development
  1. Computation of gradients using existing Markov chains
  2. Global computation of field variables and their visualization
  3. Nonlinear BVPs perhaps via branching processes
  4. Using “Walk-on-the-Boundary" (WOB) techniques
- Geometric Issues
  1. Computation of the three region model problem
  2. More complicated surfaces (solvent-excluded and ion-excluded)
  3. Accuracy issues related to the Van der Waals surface
- Optimize the performance
  1. Error/bias/variance balancing
  2. Importance sampling and the outer walks
  3. WOB to eliminate walks outside
  4. QMC methods

Future Work

- Further algorithmic development
  1. Computation of gradients using existing Markov chains
  2. Global computation of field variables and their visualization
  3. Nonlinear BVPs perhaps via branching processes
  4. Using “Walk-on-the-Boundary" (WOB) techniques
- Geometric Issues
  1. Computation of the three region model problem
  2. More complicated surfaces (solvent-excluded and ion-excluded)
  3. Accuracy issues related to the Van der Waals surface
- Optimize the performance
  1. Error/bias/variance balancing
  2. Importance sampling and the outer walks
  3. WOB to eliminate walks outside
  4. QMC methods

Future Work

- Further algorithmic development
  1. Computation of gradients using existing Markov chains
  2. Global computation of field variables and their visualization
  3. Nonlinear BVPs perhaps via branching processes
  4. Using “Walk-on-the-Boundary" (WOB) techniques
- Geometric Issues
  1. Computation of the three region model problem
  2. More complicated surfaces (solvent-excluded and ion-excluded)
  3. Accuracy issues related to the Van der Waals surface
- Optimize the performance
  1. Error/bias/variance balancing
  2. Importance sampling and the outer walks
  3. WOB to eliminate walks outside
  4. QMC methods

Future Work

- Further algorithmic development
  1. Computation of gradients using existing Markov chains
  2. Global computation of field variables and their visualization
  3. Nonlinear BVPs perhaps via branching processes
  4. Using “Walk-on-the-Boundary" (WOB) techniques
- Geometric Issues
  1. Computation of the three region model problem
  2. More complicated surfaces (solvent-excluded and ion-excluded)
  3. Accuracy issues related to the Van der Waals surface
- Optimize the performance
  1. Error/bias/variance balancing
  2. Importance sampling and the outer walks
  3. WOB to eliminate walks outside
  4. QMC methods

Future Work

- Further algorithmic development
  1. Computation of gradients using existing Markov chains
  2. Global computation of field variables and their visualization
  3. Nonlinear BVPs perhaps via branching processes
  4. Using “Walk-on-the-Boundary" (WOB) techniques
- Geometric Issues
  1. Computation of the three region model problem
  2. More complicated surfaces (solvent-excluded and ion-excluded)
  3. Accuracy issues related to the Van der Waals surface
- Optimize the performance
  1. Error/bias/variance balancing
  2. Importance sampling and the outer walks
  3. WOB to eliminate walks outside
  4. QMC methods

Future Work

- Further algorithmic development
  1. Computation of gradients using existing Markov chains
  2. Global computation of field variables and their visualization
  3. Nonlinear BVPs perhaps via branching processes
  4. Using “Walk-on-the-Boundary" (WOB) techniques
- Geometric Issues
  1. Computation of the three region model problem
  2. More complicated surfaces (solvent-excluded and ion-excluded)
  3. Accuracy issues related to the Van der Waals surface
- Optimize the performance
  1. Error/bias/variance balancing
  2. Importance sampling and the outer walks
  3. WOB to eliminate walks outside
  4. QMC methods

Future Work

- Further algorithmic development
  1. Computation of gradients using existing Markov chains
  2. Global computation of field variables and their visualization
  3. Nonlinear BVPs perhaps via branching processes
  4. Using “Walk-on-the-Boundary" (WOB) techniques
- Geometric Issues
  1. Computation of the three region model problem
  2. More complicated surfaces (solvent-excluded and ion-excluded)
  3. Accuracy issues related to the Van der Waals surface
- Optimize the performance
  1. Error/bias/variance balancing
  2. Importance sampling and the outer walks
  3. WOB to eliminate walks outside
  4. QMC methods

Future Work

- Further algorithmic development
  1. Computation of gradients using existing Markov chains
  2. Global computation of field variables and their visualization
  3. Nonlinear BVPs perhaps via branching processes
  4. Using “Walk-on-the-Boundary" (WOB) techniques
- Geometric Issues
  1. Computation of the three region model problem
  2. More complicated surfaces (solvent-excluded and ion-excluded)
  3. Accuracy issues related to the Van der Waals surface
- Optimize the performance
  1. Error/bias/variance balancing
  2. Importance sampling and the outer walks
  3. WOB to eliminate walks outside
  4. QMC methods

Bibliography


[T. Mackoy, R. C. Harris, J. Johnson, M. Mascagni and M. O. Fenley (2011)] Numerical Optimization of a Walk-on-Spheres Solver for the Linear Poisson-Boltzmann Equation *Communications in Computational Physics*, in the press.


[M. Fenley, M. Mascagni, J. McClain, A. Silalahi and N. Simonov (2010)] Using Correlated Monte Carlo Sampling for Efficiently Solving the Linearized Poisson-Boltzmann Equation Over a Broad Range of Salt Concentrations *Journal of Chemical Theory and Computation*, **6(1)**: 300–314.


[N. Simonov and M. Mascagni and M. O. Fenley (2007)] Monte Carlo Based Linear Poisson-Boltzmann Approach Makes Accurate Salt-Dependent Solvation Energy Predictions Possible *Journal of Chemical Physics*, **187(18)**, article #185105, 6 pages.

Bibliography


[T. Mackoy, R. C. Harris, J. Johnson, M. Mascagni and M. O. Fenley (2011)] Numerical Optimization of a Walk-on-Spheres Solver for the Linear Poisson-Boltzmann Equation *Communications in Computational Physics*, in the press.


[M. Fenley, M. Mascagni, J. McClain, A. Silalahi and N. Simonov (2010)] Using Correlated Monte Carlo Sampling for Efficiently Solving the Linearized Poisson-Boltzmann Equation Over a Broad Range of Salt Concentrations *Journal of Chemical Theory and Computation*, **6(1)**: 300–314.


[N. Simonov and M. Mascagni and M. O. Fenley (2007)] Monte Carlo Based Linear Poisson-Boltzmann Approach Makes Accurate Salt-Dependent Solvation Energy Predictions Possible *Journal of Chemical Physics*, **187(18)**, article #185105, 6 pages.

Bibliography


[T. Mackoy, R. C. Harris, J. Johnson, M. Mascagni and M. O. Fenley (2011)] Numerical Optimization of a Walk-on-Spheres Solver for the Linear Poisson-Boltzmann Equation *Communications in Computational Physics*, in the press.


[M. Fenley, M. Mascagni, J. McClain, A. Silalahi and N. Simonov (2010)] Using Correlated Monte Carlo Sampling for Efficiently Solving the Linearized Poisson-Boltzmann Equation Over a Broad Range of Salt Concentrations *Journal of Chemical Theory and Computation*, **6(1)**: 300–314.


[N. Simonov and M. Mascagni and M. O. Fenley (2007)] Monte Carlo Based Linear Poisson-Boltzmann Approach Makes Accurate Salt-Dependent Solvation Energy Predictions Possible *Journal of Chemical Physics*, **187(18)**, article #185105, 6 pages.

Bibliography


[M. Mascagni and N. A. Simonov (2004)] Monte Carlo Methods for Calculating Some Physical Properties of Large Molecules *SIAM Journal on Scientific Computing*, **26(1)**: 339–357.


[N. A. Simonov and M. Mascagni (2004)] Random Walk Algorithms for Estimating Effective Properties of Digitized Porous Media *Monte Carlo Methods and Applications*, **10**: 599–608.


[M. Mascagni and N. A. Simonov (2004)] The Random Walk on the Boundary Method for Calculating Capacitance *Journal of Computational Physics*, **195**: 465–473.


[A. Karaivanova, N. A. Simonov and M. Mascagni(2004)] Parallel Quasirandom Walks on the Boundary *Monte Carlo Methods and Applications*, **11**: 311–320.

Bibliography

 [M. Mascagni and N. A. Simonov (2004)] Monte Carlo Methods for Calculating Some Physical Properties of Large Molecules *SIAM Journal on Scientific Computing*, **26(1)**: 339–357.

 [N. A. Simonov and M. Mascagni (2004)] Random Walk Algorithms for Estimating Effective Properties of Digitized Porous Media *Monte Carlo Methods and Applications*, **10**: 599–608.

 [M. Mascagni and N. A. Simonov (2004)] The Random Walk on the Boundary Method for Calculating Capacitance *Journal of Computational Physics*, **195**: 465–473.

 [A. Karaivanova, N. A. Simonov and M. Mascagni(2004)] Parallel Quasirandom Walks on the Boundary *Monte Carlo Methods and Applications*, **11**: 311–320.

Bibliography

 [M. Mascagni and N. A. Simonov (2004)] Monte Carlo Methods for Calculating Some Physical Properties of Large Molecules *SIAM Journal on Scientific Computing*, **26(1)**: 339–357.

 [N. A. Simonov and M. Mascagni (2004)] Random Walk Algorithms for Estimating Effective Properties of Digitized Porous Media *Monte Carlo Methods and Applications*, **10**: 599–608.

 [M. Mascagni and N. A. Simonov (2004)] The Random Walk on the Boundary Method for Calculating Capacitance *Journal of Computational Physics*, **195**: 465–473.

 [A. Karaivanova, N. A. Simonov and M. Mascagni(2004)] Parallel Quasirandom Walks on the Boundary *Monte Carlo Methods and Applications*, **11**: 311–320.

Bibliography


[M. Mascagni and N. A. Simonov (2004)] Monte Carlo Methods for Calculating Some Physical Properties of Large Molecules *SIAM Journal on Scientific Computing*, **26(1)**: 339–357.


[N. A. Simonov and M. Mascagni (2004)] Random Walk Algorithms for Estimating Effective Properties of Digitized Porous Media *Monte Carlo Methods and Applications*, **10**: 599–608.


[M. Mascagni and N. A. Simonov (2004)] The Random Walk on the Boundary Method for Calculating Capacitance *Journal of Computational Physics*, **195**: 465–473.


[A. Karaivanova, N. A. Simonov and M. Mascagni(2004)] Parallel Quasirandom Walks on the Boundary *Monte Carlo Methods and Applications*, **11**: 311–320.

Bibliography


[C.-O. Hwang and M. Mascagni (2003)] Analysis and Comparison of Green's Function First-Passage Algorithms with "Walk on Spheres" Algorithms *Mathematics and Computers in Simulation*, **63**: 605–613.


[C.-O. Hwang and M. Mascagni (2001)] Efficient modified "walk on spheres" algortihm for the linearized Poisson-Boltzmann equation *Applied Physics Letters*, **78**: 787–789.


[C.-O. Hwang, J. A. Given and M. Mascagni (2001)] The Simulation-Tabulation Method for Classical Diffusion Monte Carlo *Journal of Computational Physics*, **174**: 925–946.


[C.-O. Hwang, J. A. Given and M. Mascagni (2000)] On the Rapid Calculation of Permeability for Porous Media Using Brownian Motion Paths *Physics of Fluids*, **12**: 1699–1709.

Bibliography


[C.-O. Hwang and M. Mascagni (2003)] Analysis and Comparison of Green's Function First-Passage Algorithms with "Walk on Spheres" Algorithms *Mathematics and Computers in Simulation*, **63**: 605–613.


[C.-O. Hwang and M. Mascagni (2001)] Efficient modified "walk on spheres" algortihm for the linearized Poisson-Boltzmann equation *Applied Physics Letters*, **78**: 787–789.


[C.-O. Hwang, J. A. Given and M. Mascagni (2001)] The Simulation-Tabulation Method for Classical Diffusion Monte Carlo *Journal of Computational Physics*, **174**: 925–946.


[C.-O. Hwang, J. A. Given and M. Mascagni (2000)] On the Rapid Calculation of Permeability for Porous Media Using Brownian Motion Paths *Physics of Fluids*, **12**: 1699–1709.

Bibliography

- [C.-O. Hwang and M. Mascagni (2003)] Analysis and Comparison of Green's Function First-Passage Algorithms with "Walk on Spheres" Algorithms *Mathematics and Computers in Simulation*, **63**: 605–613.
- [C.-O. Hwang and M. Mascagni (2001)] Efficient modified "walk on spheres" algortihm for the linearized Poisson-Boltzmann equation *Applied Physics Letters*, **78**: 787–789.
- [C.-O. Hwang, J. A. Given and M. Mascagni (2001)] The Simulation-Tabulation Method for Classical Diffusion Monte Carlo *Journal of Computational Physics*, **174**: 925–946.
- [C.-O. Hwang, J. A. Given and M. Mascagni (2000)] On the Rapid Calculation of Permeability for Porous Media Using Brownian Motion Paths *Physics of Fluids*, **12**: 1699–1709.

Bibliography

- [C.-O. Hwang and M. Mascagni (2003)] Analysis and Comparison of Green's Function First-Passage Algorithms with "Walk on Spheres" Algorithms *Mathematics and Computers in Simulation*, **63**: 605–613.
- [C.-O. Hwang and M. Mascagni (2001)] Efficient modified "walk on spheres" algortihm for the linearized Poisson-Boltzmann equation *Applied Physics Letters*, **78**: 787–789.
- [C.-O. Hwang, J. A. Given and M. Mascagni (2001)] The Simulation-Tabulation Method for Classical Diffusion Monte Carlo *Journal of Computational Physics*, **174**: 925–946.
- [C.-O. Hwang, J. A. Given and M. Mascagni (2000)] On the Rapid Calculation of Permeability for Porous Media Using Brownian Motion Paths *Physics of Fluids*, **12**: 1699–1709.