Novel Stochastic Methods in Biochemical Electrostatics: (Stochastic Methods for PDEs Can Beat Deterministic Methods)

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, Alexander Silalahi, Robert Harris, and Messrs. Travis Mackoy, and James McClain

Research supported by ARO, DOE, NATO, NASA, and NSF
Introduction

Motivation

Mathematical Model
  Electrostatic Potential and Energy
  The Feynman-Kac Formula

Fast Exit Point Calculations
  ‘Walk-on-Spheres’ Algorithm
  Walk-in-Subdomains
  Monte Carlo Treatment of Boundary Conditions

Monte Carlo Estimates
  Monte Carlo Estimates
  Computational Geometry
  Correlated and Uncorrelated Sampling

Computational Results

Conclusions and Future Work
Introduction

Motivation

Mathematical Model
  Electrostatic Potential and Energy
  The Feynman-Kac Formula

Fast Exit Point Calculations
  ‘Walk-on-Spheres’ Algorithm
  Walk-in-Subdomains
  Monte Carlo Treatment of Boundary Conditions

Monte Carlo Estimates
  Monte Carlo Estimates
  Computational Geometry
  Correlated and Uncorrelated Sampling

Computational Results

Conclusions and Future Work
Introduction

Motivation

Mathematical Model
   Electrostatic Potential and Energy
   The Feynman-Kac Formula

Fast Exit Point Calculations
   ‘Walk-on-Spheres’ Algorithm
   Walk-in-Subdomains
   Monte Carlo Treatment of Boundary Conditions

Monte Carlo Estimates
   Monte Carlo Estimates
   Computational Geometry
   Correlated and Uncorrelated Sampling

Computational Results

Conclusions and Future Work
Introduction

Motivation

Mathematical Model
  Electrostatic Potential and Energy
  The Feynman-Kac Formula

Fast Exit Point Calculations
  ‘Walk-on-Spheres’ Algorithm
  Walk-in-Subdomains
  Monte Carlo Treatment of Boundary Conditions

Monte Carlo Estimates
  Monte Carlo Estimates
  Computational Geometry
  Correlated and Uncorrelated Sampling

Computational Results

Conclusions and Future Work
Introduction

Motivation

Mathematical Model
   Electrostatic Potential and Energy
   The Feynman-Kac Formula

Fast Exit Point Calculations
   ‘Walk-on-Spheres’ Algorithm
   Walk-in-Subdomains
   Monte Carlo Treatment of Boundary Conditions

Monte Carlo Estimates
   Monte Carlo Estimates
   Computational Geometry
   Correlated and Uncorrelated Sampling

Computational Results

Conclusions and Future Work
Introduction

Motivation

Mathematical Model
  Electrostatic Potential and Energy
  The Feynman-Kac Formula

Fast Exit Point Calculations
  ‘Walk-on-Spheres’ Algorithm
  Walk-in-Subdomains
  Monte Carlo Treatment of Boundary Conditions

Monte Carlo Estimates
  Monte Carlo Estimates
  Computational Geometry
  Correlated and Uncorrelated Sampling

Computational Results

Conclusions and Future Work
Introduction

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
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
Introduction

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
Introduction

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
Introduction

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
Introduction

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
Introduction

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
Introduction
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
Introduction

Monte Carlo Methods: Properties

- Monte Carlo method basics: \( I = \int_0^1 f(x)dx \)
  
  1. Random/stochastic process: \( x_i \sim U[0, 1) \)
  2. Random variable: \( f(x_i) \) where \( E[f(x_i)] = I \) and \( E[f^2(x_i)] < \infty \)

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

Monte Carlo Methods: Properties

- Monte Carlo method basics: \( I = \int_{0}^{1} f(x) \, dx \)
  - Random/stochastic process: \( x_i \sim U[0, 1) \)
  - Random variable: \( f(x_i) \) where \( E[f(x_i)] = I \) and \( E[f^2(x_i)] < \infty \)

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

Monte Carlo Methods: Properties

- Monte Carlo method basics: \( I = \int_0^1 f(x) \, dx \)
  1. Random/stochastic process: \( x_i \sim U[0, 1] \)
  2. Random variable: \( f(x_i) \) where \( E[f(x_i)] = I \) and \( E[f^2(x_i)] < \infty \)

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

Monte Carlo Methods: Properties

- **Monte Carlo method basics:** \( I = \int_0^1 f(x)dx \)
  1. Random/stochastic process: \( x_i \sim U[0,1] \)
  2. Random variable: \( f(x_i) \) where \( E[f(x_i)] = I \) and \( E[f^2(x_i)] < \infty \)

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

Monte Carlo Methods: Properties

- Monte Carlo method basics: \( I = \int_{0}^{1} f(x) dx \)
  1. Random/stochastic process: \( x_i \sim U[0, 1] \)
  2. Random variable: \( f(x_i) \) where \( E[f(x_i)] = I \) and \( E[f^2(x_i)] < \infty \)

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

Monte Carlo Methods: Properties

- Monte Carlo method basics: \( I = \int_{0}^{1} f(x)dx \)
  1. Random/stochastic process: \( x_i \sim U[0, 1) \)
  2. Random variable: \( f(x_i) \) where \( E[f(x_i)] = I \) and \( E[f^2(x_i)] < \infty \)

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

Monte Carlo Methods: Properties

- Monte Carlo method basics: \( I = \int_0^1 f(x)dx \)
  1. Random/stochastic process: \( x_i \sim U[0, 1) \)
  2. Random variable: \( f(x_i) \) where \( E[f(x_i)] = I \) and \( E[f^2(x_i)] < \infty \)

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

Monte Carlo Methods: Properties

- **Monte Carlo method basics:** \( I = \int_0^1 f(x)dx \)
  1. Random/stochastic process: \( x_i \sim U[0, 1) \)
  2. Random variable: \( f(x_i) \) where \( E[f(x_i)] = I \) and \( E[f^2(x_i)] < \infty \)

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

Figure: Biomolecule with dielectric $\epsilon_i$ and region region $G_i$ is in solution with dielectric $\epsilon_e$ 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, $\Phi_i$, 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, \( \Phi_i \), and point charges, \( Q_m \), inside a molecule (in CGS units):

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

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

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

but we only consider the linearized PBE:

\[
\Delta \Phi_e(x) - \kappa^2 \Phi_e(x) = 0, \quad x \in G_e
\]

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

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

\( G_i \) and \( G_e \) are defined as appropriate.
Mathematical Model: Partial Differential Equations

- Poisson equation for the electrostatic potential, $\Phi_i$, 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):

\[
\kappa^2 = \frac{8\pi N_A e^2 C_s}{\epsilon_e 1000k_B T},
\]

- \(C_s\) – concentration of ions (in moles)
- \(N_A\) – Avogadro’s number
- \(e\) – elementary protonic charge
- \(k_B\) – Boltzmann’s constant
- \(\epsilon_e\) – dielectric permittivity outside the molecule
Mathematical Model: Debye-Hückle Parameter

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

\[ \kappa^2 = \frac{8\pi N_A e^2 C_s}{\epsilon_e 1000 k_B T}, \]

where

- \( C_s \) – concentration of ions (in moles)
- \( N_A \) – Avogadro’s number
- \( e \) – elementary protonic charge
- \( k_B \) – Boltzmann’s constant
- \( \epsilon_e \) – dielectric permittivity outside the molecule
Mathematical Model: Debye-Hückle Parameter

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

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

- \( C_s \) – concentration of ions (in moles)
- \( N_A \) – Avogadro’s number
- \( e \) – elementary protonic charge
- \( k_B \) – Boltzmann’s constant
- \( \epsilon_e \) – dielectric permittivity outside the molecule
Mathematical Model: Debye-Hückle Parameter

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

\[ \kappa^2 = \frac{8\pi N_A e^2 C_s}{\epsilon_e 1000 k_B T} \]

- \( C_s \) – concentration of ions (in moles)
- \( N_A \) – Avogadro’s number
- \( e \) – elementary protonic charge
- \( k_B \) – Boltzmann’s constant
- \( \epsilon_e \) – dielectric permittivity outside the molecule
Mathematical Model: Debye-Hückle Parameter

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

\[ \kappa^2 = \frac{8\pi N_A e^2 C_s}{\epsilon e 1000 k_B T}, \]  

where

- \( C_s \) – concentration of ions (in moles)
- \( N_A \) – Avogadro’s number
- \( e \) – elementary protonic charge
- \( k_B \) – Boltzmann’s constant
- \( \epsilon e \) – 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 \( \Phi \):

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

- 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:

\[
\Delta G_{solv}^{elec} = W_{rf}(\epsilon_i, \epsilon_e, \kappa) - W_{rf}(\epsilon_i, 1, 0)
\]
Electrostatic Potential and Energy

- Point values of the potential: $\Phi(x) = \Phi_{rf}(x) + \Phi_c(x)$
  Here, singular part of $\Phi$:
  
  $$\Phi_c(x) = \sum_{m=1}^{M} \frac{Q_m}{|x - x^{(m)}|}$$

- 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:
  
  $$\Delta G_{solv}^{elec} = W_{rf}(\epsilon_i, \epsilon_e, \kappa) - W_{rf}(\epsilon_i, 1, 0)$$
Electrostatic Potential and Energy

- Point values of the potential: \( \Phi(x) = \Phi_{rf}(x) + \Phi^c(x) \)
  Here, singular part of \( \Phi \):
  \[
  \Phi^c(x) = \sum_{m=1}^{M} \frac{Q_m}{|x - x^{(m)}|}
  \]
- 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:
  \[
  \Delta G_{solv}^{elec} = W_{rf}(\epsilon_i, \epsilon_e, \kappa) - W_{rf}(\epsilon_i, 1, 0)
  \]
Mathematical Model

The Feynman-Kac Formula

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

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

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 $\Omega \subseteq \mathbb{R}^d$

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

- 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

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

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

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

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

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

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 \( m, x_i = x_{i-1} + d_i \omega_i \), \( i = 1, 2, \ldots \), where
  1. \( d_i = d(x_{i-1}) \) is distance from \( x_{i-1} \) to \( \Gamma \)
  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 \( q(\kappa, d_i) = \frac{\kappa d_i}{\sinh(\kappa d_i)} \) 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 \( \Gamma \)
  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 \( q(\kappa, d_i) = \frac{\kappa d_i}{\sinh(\kappa d_i)} \) 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 \( \Gamma \)
  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 \( q(\kappa, d_i) = \frac{\kappa d_i}{\sinh(\kappa d_i)} \) 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 \( \Gamma \)
  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 \( q(\kappa, d_i) = \frac{\kappa d_i}{\sinh(\kappa d_i)} \) 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 \( \Gamma \)
  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 \( q(\kappa, d_i) = \frac{\kappa d_i}{\sinh(\kappa d_i)} \) 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 \( \Gamma \)
  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 \( q(\kappa, d_i) = \frac{\kappa d_i}{\sinh(\kappa d_i)} \) 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 \( \Gamma \)
  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 \( q(\kappa, d_i) = \frac{\kappa d_i}{\sinh(\kappa d_i)} \) 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, $B(x^m_i, R^m_i)$, exit points are distributed in accordance with the Poisson kernel:
     $$\frac{1}{4\pi R^m_i} \left| x_i - x^m_i \right|^2 - \left( R^m_i \right)^2$$
     $$\left| x_i - x_{i+1} \right|^3$$
‘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, $B(x^m, R_i^m)$, exit points are distributed in accordance with the Poisson kernel:

$$\frac{1}{4\pi R_i^m} \frac{|x_i - x^m|^2 - (R_i^m)^2}{|x_i - x_{i+1}|^3}$$
‘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, $B(x^m, R^m)$, exit points are distributed in accordance with the Poisson kernel:

$$ \frac{1}{4\pi R^m_i} \frac{|x_i - x^m|^2 - (R^m)^2}{|x_i - x_{i+1}|^3} $$
‘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, $B(x^m, R^m)$, exit points are distributed in accordance with the Poisson kernel:

\[
\frac{1}{4\pi R^m_i} \left( \frac{|x_i - x^m|^2 - (R^m_i)^2}{|x_i - x_{i+1}|^3} \right)
\]
‘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, $B(x^m_i, R^m_i)$, exit points are distributed in accordance with the Poisson kernel:

$$\frac{1}{4\pi R^m_i} \frac{|x_i - x^m_i|^2 - (R^m_i)^2}{|x_i - x_{i+1}|^3}$$
‘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, \( B(x^m_i, R^m_i) \), exit points are distributed in accordance with the Poisson kernel:

\[
\frac{1}{4\pi R^m_i} \frac{|x_i - x^m_i|^2 - (R^m_i)^2}{|x_i - x_{i+1}|^3}
\]
‘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 \( S(y, a) \):

\[
\begin{align*}
  u(y) &= \frac{\epsilon_e}{\epsilon_e + \epsilon_i} \int_{S_e(y, a)} \frac{1}{2\pi a^2} \frac{\kappa a}{\sinh(\kappa a)} u_e \\
  &\quad + \frac{\epsilon_i}{\epsilon_e + \epsilon_i} \int_{S_i(y, a)} \frac{1}{2\pi a^2} \frac{\kappa a}{\sinh(\kappa a)} u_i \\
  &\quad - \frac{\epsilon_e - \epsilon_i}{\epsilon_e + \epsilon_i} \int_{\Gamma \cap B(y, a) \setminus \{y\}} \frac{\cos \varphi_{yx}}{2\pi |y - x|^2} Q_{\kappa, a} u \\
  &\quad + \frac{\epsilon_i}{\epsilon_e + \epsilon_i} \int_{B_i(y, a)} [-2\kappa^2 \Phi_{\kappa}] u_i
\end{align*}
\]
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 $S(y, a)$:
  $$u(y) = \frac{\epsilon_e}{\epsilon_e + \epsilon_i} \int_{S_e(y,a)} \frac{1}{2\pi a^2 \sinh(\kappa a)} u_e$$
  $$+ \frac{\epsilon_i}{\epsilon_e + \epsilon_i} \int_{S_i(y,a)} \frac{1}{2\pi a^2 \sinh(\kappa a)} u_i$$
  $$- \frac{\epsilon_e - \epsilon_i}{\epsilon_e + \epsilon_i} \int_{\Gamma \cap B(y,a) \setminus \{y\}} \frac{\cos \varphi_{yx}}{2\pi |y - x|^2} Q_{\kappa, a} u$$
  $$+ \frac{\epsilon_i}{\epsilon_e + \epsilon_i} \int_{B_i(y,a)} [-2\kappa^2 \Phi_{\kappa}] u_i$$  \hfill (1)
Monte Carlo Treatment of Boundary Conditions

Randomized approximation to (1): \( u(y) = \mathbb{E}u(x) + O((a/2R)^3) \):

- **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 \( \frac{\kappa a}{\sinh(\kappa a)} \)

- **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 (1): \( u(y) = \mathbb{E} u(x) + O((a/2R)^3) \):

- 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 \( \frac{\kappa a}{\sinh(\kappa a)} \)

- 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 \( B_\kappa(y, a) \) (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 (1): \( u(y) = \mathbb{E}u(x) + O((a/2R)^3) \):

- 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 \( \frac{\kappa a}{\sinh(\kappa a)} \)

- 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 \( B_r(y, a) \) (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 (1): $u(y) = \mathbb{E}u(x) + O((a/2R)^3)$:

- 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 $\frac{\kappa a}{\sinh(\kappa a)}$

- 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 (1): \( u(y) = \mathbb{E}u(x) + O((a/2R)^3) \):

- 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 \( \frac{\kappa a}{\sinh(\kappa a)} \)

- 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 (1): \( u(y) = \mathbb{E}u(x) + O((a/2R)^3) \):

- 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 \( \frac{\kappa a}{\sinh(\kappa a)} \)

- 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 (1): \( u(y) = \mathbb{E} u(x) + O((a/2R)^3) \):

- **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 \( \frac{\kappa a}{\sinh(\kappa a)} \)

- **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 (1): \( u(y) = \mathbb{E}u(x) + O((a/2R)^3) \):

1. 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 \( \frac{\kappa a}{\sinh(\kappa a)} \)

2. 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 $\kappa$). Estimates for point values of the potential and free energy are $O(h)$-biased

- New treatment of boundary conditions provides $O(\bar{a})^2$-biased and more efficient Monte Carlo algorithm. Mean number of steps is $O((\bar{a})^{-1} \log(\bar{a}) f(\kappa))$, $\bar{a} = a/2R$.

- More subtle approximation to (1) 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, \[ \text{Mean number of returns to the boundary is } O(d_0)^{-1} \]

- 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 $\kappa$). Estimates for point values of the potential and free energy are $O(h)$-biased

- New treatment of boundary conditions provides $O(\bar{a}^2)$-biased and more efficient Monte Carlo algorithm. Mean number of steps is $O((\bar{a})^{-1} \log(\bar{a}) f(\kappa))$, $\bar{a} = a/2R$.

- More subtle approximation to (1) 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 $\kappa$). Estimates for point values of the potential and free energy are $O(h)$-biased

- New treatment of boundary conditions provides $O(\bar{a})^2$-biased and more efficient Monte Carlo algorithm. Mean number of steps is $O((\bar{a})^{-1} \log(\bar{a}) f(\kappa))$, $\bar{a} = a/2R$.

- More subtle approximation to (1) will provide even more efficient Monte Carlo estimates
Monte Carlo Estimates

The estimate for the reaction-field potential point value:
\[
\xi[\Phi_{rf}](x^{(m)}) = -\Phi^c(x_1^*) + \sum_{j=2}^{N_{ins}} F_j(\kappa) (\Phi^c(x_{j,ins}^*) - \Phi^c(x_{j,ins}^*)) \tag{2}
\]

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:
\[
\xi[W_{rf}] = \frac{1}{2} \sum_{m=1}^{M} Q_m \xi[\Phi_{rf}](x^{(m)}) \tag{3}
\]
Novel Stochastic Methods in Biochemical Electrostatics: (Stochastic Methods for PDEs Can Beat Deterministic Methods)

Monte Carlo Estimates

The estimate for the reaction-field potential point value:

\[
\xi[\Phi_{rf}](x^{(m)}) = -\Phi^c(x_1^*)
+ \sum_{j=2}^{N_{ins}} F_j(\kappa) \left( \Phi^c(x_{j,ins}^*) - \Phi^c(x_{j,ins}^*) \right)
\]  

(2)

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:

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

(3)
Monte Carlo Estimates

- The estimate for the reaction-field potential point value:
  \[
  \xi[\Phi_{rf}](x^{(m)}) = -\Phi^c(x_1^*) + \sum_{j=2}^{N_{ins}} F_j(\kappa) (\Phi^c(x_{j,ins}^*) - \Phi^c(x_{j,ins}^*))
  \]  
  (2)

- 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:
  \[
  \xi[W_{rf}] = \frac{1}{2} \sum_{m=1}^{M} Q_m \xi[\Phi_{rf}](x^{(m)})
  \]  
  (3)
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 $\infty$ in One Step

Figure: $\kappa = 0$, $p_\infty = 1 - R_{Enclosed}/\text{dist}$
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
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.
Monte Carlo Estimates

Correlated and Uncorrelated Sampling

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 $\kappa$ used in simulations, $F_j(\kappa) = 1$
- For an arbitrary $\kappa' > \kappa$:
  $F_j(\kappa')$ is multiplied by the ratio $\frac{q(\kappa', d)}{q(\kappa, d)}$ on every step of the WOS in the exterior
- The results obtained with the estimates (2) and (3) for different values of $\kappa$ 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 $\kappa$ used in simulations, $F_j(\kappa) = 1$

- For an arbitrary $\kappa' > \kappa$:
  
  \[ F_j(\kappa') \text{ is multiplied by the ratio } \frac{q(\kappa', d)}{q(\kappa, d)} \text{ on every step of the WOS in the exterior} \]

- The results obtained with the estimates (2) and (3) for different values of $\kappa$ 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 $\kappa$ used in simulations, $F_j(\kappa) = 1$

- For an arbitrary $\kappa' > \kappa$:
  $F_j(\kappa')$ is multiplied by the ratio $\frac{q(\kappa', d)}{q(\kappa, d)}$ on every step of the WOS in the exterior

- The results obtained with the estimates (2) and (3) for different values of $\kappa$ 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 $\kappa$ used in simulations, $F_j(\kappa) = 1$

- For an arbitrary $\kappa' > \kappa$:
  $F_j(\kappa')$ is multiplied by the ratio $\frac{q(\kappa', d)}{q(\kappa, d)}$ on every step of the WOS in the exterior

- The results obtained with the estimates (2) and (3) for different values of $\kappa$ 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 $\kappa$ used in simulations, $F_j(\kappa) = 1$

- For an arbitrary $\kappa' > \kappa$:
  
  $F_j(\kappa')$ is multiplied by the ratio $\frac{q(\kappa', d)}{q(\kappa, d')}$ on every step of the WOS in the exterior

- The results obtained with the estimates (2) and (3) for different values of $\kappa$ 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 $\kappa$ used in simulations, $F_j(\kappa) = 1$
- For an arbitrary $\kappa' > \kappa$:
  $F_j(\kappa')$ is multiplied by the ratio $\frac{q(\kappa', d)}{q(\kappa, d')}$ on every step of the WOS in the exterior
- The results obtained with the estimates (2) and (3) for different values of $\kappa$ 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 (\(\kappa^2\) 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 \textit{SPRNG} to do this effectively
  1. Unbound: when exiting the molecule the seed is stored using \textit{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 (\( \kappa^2 \) 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 (\( \kappa^2 \) 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 \texttt{SPRNG} to do this effectively
  1. Unbound: when exiting the molecule the seed is stored using \texttt{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
- This leads to correlation between unbound and bound energy computations that decreases as the walk length increases (\( \kappa^2 \) 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 \texttt{SPRNG} to do this effectively
  1. Unbound: when exiting the molecule the seed is stored using \texttt{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 (\( \kappa^2 \) 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 \textit{SPRNG} to do this effectively.
  1. Unbound: when exiting the molecule the seed is stored using \textit{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 ($\kappa^2$ 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 \textit{SPRNG} to do this effectively:
  1. Unbound: when exiting the molecule the seed is stored using \textit{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 (\( \kappa^2 \) decreases)
Novel Stochastic Methods in Biochemical Electrostatics: (Stochastic Methods for PDEs Can Beat Deterministic Methods)

Computational Results

**Accuracy: Monte Carlo vs. Deterministic**

![Graph comparing Monte Carlo and Deterministic Electrostatic Solvation Free Energies](image)
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 ($\epsilon$): bias is $O(\epsilon)$
     - Auxiliary sphere radius ($a$): bias is $O(a^3)$
     - Effective Van der Waals sphere radius, $R$
     - Overall bias: $(\frac{a}{2R})^3 + (\frac{\epsilon}{2R})$
  3. $\text{Var}[\sum_i q_i \Phi(x_i)] = \sum_i q_i^2 \text{Var}[\Phi(x_i)]$
  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 $\epsilon$, $a$ 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 ($\epsilon$): bias is $O(\epsilon)$
     - Auxiliary sphere radius ($a$): bias is $O(a^3)$
     - Effective Van der Waals sphere radius, $R$
     - Overall bias: $(\frac{a}{2R})^3 + (\frac{\epsilon}{2R})$
  3. $\text{Var}[\sum_i q_i \Phi(x_i)] = \sum_i q_i^2 \text{Var}[\Phi(x_i)]$
  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 $\epsilon$, $a$ 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 ($\epsilon$): bias is $O(\epsilon)$
     - Auxiliary sphere radius ($a$): bias is $O(a^3)$
     - Effective Van der Waals sphere radius, $R$
     - Overall bias: $(\frac{a}{2R})^3 + (\frac{\epsilon}{2R})$
  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 $\epsilon$, $a$ and $ANN$
  7. Large $ANN$ means errors in drawing the largest sphere outside the molecule for WOS
Computational Results

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 ($\epsilon$): bias is $O(\epsilon)$
     - Auxiliary sphere radius ($a$): bias is $O(a^3)$
     - Effective Van der Waals sphere radius, $R$
     - Overall bias: $(\frac{a}{2R})^3 + (\frac{\epsilon}{2R})$
  3. $\text{Var}[\sum_i q_i \Phi(x_i)] = \sum_i q_i^2 \text{Var}[\Phi(x_i)]$
  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 $\epsilon$, $a$ 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 ($\epsilon$): bias is $O(\epsilon)$
    - Auxiliary sphere radius ($a$): bias is $O(a^3)$
    - Effective Van der Waals sphere radius, $R$
    - Overall bias: $(\frac{a}{2R})^3 + (\frac{\epsilon}{2R})$
  3. $\text{Var} \left[ \sum_i q_i \Phi(x_i) \right] = \sum_i q_i^2 \text{Var}[\Phi(x_i)]$
  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 $\epsilon$, $a$ 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 ($\epsilon$): bias is $O(\epsilon)$
     - Auxiliary sphere radius ($a$): bias is $O(a^3)$
     - Effective Van der Waals sphere radius, $R$
     - Overall bias: $(\frac{a}{2R})^3 + \left(\frac{\epsilon}{2R}\right)$
  3. $\text{Var}\left[\sum_i q_i \Phi(x_i)\right] = \sum_i q_i^2 \text{Var}[\Phi(x_i)]$
  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 $\epsilon$, $a$ 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 ($\epsilon$): 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. $\text{Var}[\sum_i q_i \Phi(x_i)] = \sum_i q_i^2 \text{Var}[\Phi(x_i)]$
  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 $\epsilon$, $a$ 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 ($\epsilon$): bias is $O(\epsilon)$
   - Auxiliary sphere radius ($a$): bias is $O(a^3)$
   - Effective Van der Waals sphere radius, $R$
   - Overall bias: $(\frac{a}{2R})^3 + (\frac{\epsilon}{2R})$
3. $\text{Var}[\sum_i q_i \Phi(x_i)] = \sum_i q_i^2 \text{Var}[\Phi(x_i)]$
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 $\epsilon$, $a$ 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 ($\epsilon$): 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. $\text{Var}\left[\sum_i q_i \Phi(x_i)\right] = \sum_i q_i^2 \text{Var}[\Phi(x_i)]$
  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 $\epsilon$, $a$ 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 ($\epsilon$): 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. $\text{Var}[\sum_i q_i\Phi(x_i)] = \sum_i q_i^2 \text{Var}[\Phi(x_i)]$
  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 $\epsilon$, $a$ 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 ($\epsilon$): bias is $O(\epsilon)$
    - Auxiliary sphere radius ($a$): bias is $O(a^3)$
    - Effective Van der Waals sphere radius, $R$
    - Overall bias: $(\frac{a}{2R})^3 + (\frac{\epsilon}{2R})$
  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 $\epsilon$, $a$ 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 ($\epsilon$): bias is $O(\epsilon)$
    - Auxiliary sphere radius ($a$): bias is $O(a^3)$
    - Effective Van der Waals sphere radius, $R$
    - Overall bias: $(\frac{a}{2R})^3 + (\frac{\epsilon}{2R})$
  3. $\text{Var}[\sum_i q_i \Phi(x_i)] = \sum_i q_i^2 \text{Var}[\Phi(x_i)]$
  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 $\epsilon$, $a$ and ANN
  7. Large ANN means errors in drawing the largest sphere outside the molecule for WOS
Novel Stochastic Methods in Biochemical Electrostatics: (Stochastic Methods for PDEs Can Beat Deterministic Methods)

Computational Results

Timing: Better Than Expected

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

- 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

- 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 ($M \gg 10000$) molecules.
Conclusions

- 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 ($M \gg 10000$) molecules.
Conclusions

- 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 ($M \gg 10000$) 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


Bibliography


Bibliography


[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


Bibliography


Bibliography


Bibliography


