1 Level Set Method
2 Solute-Solvent Model
3 Reinitialization
4 Conclusion
Types of problems

- **Geometric based motion**
  - Motion based on curvature
  - Normal direction motion
  - Simple “constant” motion

- **Physical based motion**
  - Fluid dynamics
    - Two-phase flow
    - Combustion
    - Liquid-gas interactions
  - Biology
    - Biomolecular surfaces
    - Membranes
  - Material science
    - Solidification
    - Epitaxial growth of thin films
Front Tracking Methods

1. Place markers on interface
2. Calculate velocity for each marker
3. Propagate markers one time step
4. Check for topological changes of interfaces
   - add/remove markers
   - split/join interfaces if necessary
5. Go to step 2
Front Tracking Methods

1. Place markers on interface
2. Calculate velocity for each marker
3. Propagate markers one time step
4. Check for topological changes of interfaces
   - add/remove markers
   - split/join interfaces if necessary
5. Go to step 2
Front Tracking Methods

1. Place markers on interface
2. Calculate velocity for each marker
3. Propagate markers one time step
4. Check for topological changes of interfaces
   - add/remove markers
   - split/join interfaces if necessary
5. Go to step 2
Front Tracking Methods

1. Place markers on interface
2. Calculate velocity for each marker
3. Propagate markers one time step
4. Check for topological changes of interfaces
   - add/remove markers
   - split/join interfaces if necessary
5. Go to step 2
Front Tracking Methods

1. Place markers on interface
2. Calculate velocity for each marker
3. Propagate markers one time step
4. Check for topological changes of interfaces
   - add/remove markers
   - split/join interfaces if necessary

Go to step 2
Front Tracking Methods

1. Place markers on interface
2. Calculate velocity for each marker
3. Propagate markers one time step
4. Check for topological changes of interfaces
   - add/remove markers
   - split/join interfaces if necessary
5. Go to step 2
The interface is represented as the zero level set of a function

\[ \phi(x, y, t) = 0 \]

Topological changes will occur naturally as time progresses.
Derivation of the Level Set Evolution Equation

Let \((x(t), y(t))\) parameterize a particle that stays on the moving interface.

- \(\phi(x(0), y(0), t = 0) = 0\)
- \(\phi(x(1), y(1), t = 1) = 0\)
- \(\phi(x(2), y(2), t = 2) = 0\)
Let \((x(t), y(t))\) parameterize a particle that stays on the moving interface.

- \(\phi(x(0), y(0), t = 0) = 0\)
- \(\phi(x(1), y(1), t = 1) = 0\)
- \(\phi(x(2), y(2), t = 2) = 0\)
Let \((x(t), y(t))\) parameterize a particle that stays on the moving interface.

- \(\phi(x(0), y(0), t = 0) = 0\)
- \(\phi(x(1), y(1), t = 1) = 0\)
- \(\phi(x(2), y(2), t = 2) = 0\)
Let \((x(t), y(t))\) parameterize a particle that stays on the moving interface.

\[
\begin{align*}
\phi(x(0), y(0), t = 0) &= 0 \\
\phi(x(1), y(1), t = 1) &= 0 \\
\phi(x(2), y(2), t = 2) &= 0
\end{align*}
\]

\[
\frac{d\phi}{dt} = \phi_t + \phi_x x'(t) + \phi_y y'(t)
\]

\[
= \phi_t + \nabla \phi \cdot (x'(t), y'(t))
\]

\[
= \phi_t + \nabla \phi \cdot \mathbf{V} = 0
\]
Level Set Evolution Equation

For externally generated velocity fields:

\[ \phi_t + \nabla \phi \cdot V = 0 \]

For internally generated velocity fields:

\[ \phi_t + |\nabla \phi| V_n = 0 \]

where

\[ V_n = V \cdot n = V \cdot \frac{\nabla \phi}{|\nabla \phi|} \]

Interface = \( \Gamma(t) = \{(x, y) \mid \phi(x, y, t) = 0\} \)
Basic Types of Interface Motion

- Vector field $\mathbf{V}$
- Scalar field $V_n$ giving speed in level sets’ normal direction
- $V_n$ proportional to level set curvature $\kappa = \nabla \cdot \left( \frac{\nabla \phi}{|\nabla \phi|} \right)$
**Evolution equation**

\[ \phi_t + V_n |\nabla \phi| = 0 \]

1. Approximate \( |\nabla \phi| \) at each vertex \( v_i \) by the average of \( |\nabla \phi| \) over a patch around \( v_i \).

2. Use a Runga-Kutta type method using the values of \( V_n \) at each vertex.
The Lennard-Jones potential is given by:

\[ U(r) = 4\varepsilon \left( \left( \frac{\sigma}{r} \right)^{12} - \left( \frac{\sigma}{r} \right)^6 \right) \]

- \( U(r = \sigma) = 0 \)
- Minimum: \( U(r = \sigma \sqrt{2} \approx 1.12\sigma) = -\varepsilon \)
- \( r > \sigma \) attracts
- \( r < \sigma \) repels
Solute-Solvent Model

- Molecules consist of atoms at locations $x_1, \ldots, x_N$
- Solvent region $\Omega_w$
- Solute region $\Omega_m$
- A solute-solvent interface $\Gamma$
Molecules consist of atoms at locations $x_1, \ldots, x_N$

Solvent region $\Omega_w$

Solute region $\Omega_m$

A solute-solvent interface $\Gamma$

A free energy model

$$G[\Gamma; x_1, \ldots, x_N] = \int_{\Gamma} \gamma \, dS + \rho_w \sum_{i=1}^{N} \int_{\Omega_w} U_{sw}(|x - x_i|) \, dV$$
Algorithm

1. Define a regular grid
2. Define atoms (with radii $\sigma_i$)
3. Construct initial surface
4. At each time step:
   1. body-fit mesh to interface $\Gamma$
   2. calculate L-J potential on $\Gamma$
   3. calculate curvature $\kappa$
   4. on $\Gamma$ let
      $V_n = -\gamma_0 \kappa + \rho_0 \sum_{i=1}^{N} U(|x - x_i|)$
   5. extend $V_n$ to regular grid
   6. evolve LS function on regular grid
5. Stop when free energy stops decreasing
Algorithm

1. Define a regular grid
2. Define atoms (with radii $\sigma_i$)
3. Construct initial surface
4. At each time step:
   1. body-fit mesh to interface $\Gamma$
   2. calculate L-J potential on $\Gamma$
   3. calculate curvature $\kappa$
   4. on $\Gamma$ let
      \[ V_n = -\gamma_0\kappa + \rho_0 \sum_{i=1}^{N} U(|x-x_i|) \]
   5. extend $V_n$ to regular grid
   6. evolve LS function on regular grid
5. Stop when free energy stops decreasing
Algorithm

1. Define a regular grid
2. Define atoms (with radii $\sigma_i$)
3. Construct initial surface
4. At each time step:
   1. body-fit mesh to interface $\Gamma$
   2. calculate L-J potential on $\Gamma$
   3. calculate curvature $\kappa$
   4. on $\Gamma$ let
      \[ V_n = -\gamma_0 \kappa + \rho_0 \sum_{i=1}^{N} U(|\mathbf{x} - \mathbf{x}_i|) \]
   5. extend $V_n$ to regular grid
   6. evolve LS function on regular grid
5. Stop when free energy stops decreasing
Algorithm

1. Define a regular grid
2. Define atoms (with radii $\sigma_i$)
3. Construct initial surface
4. At each time step:
   1. body-fit mesh to interface $\Gamma$
   2. calculate L-J potential on $\Gamma$
   3. calculate curvature $\kappa$
   4. on $\Gamma$ let
      \[ V_n = -\gamma_0 \kappa + \rho_0 \sum_{i=1}^{N} U(|x-x_i|) \]
   5. extend $V_n$ to regular grid
   6. evolve LS function on regular grid
5. Stop when free energy stops decreasing
Algorithm

1. Define a regular grid
2. Define atoms (with radii $\sigma_i$)
3. Construct initial surface
4. At each time step:
   1. body-fit mesh to interface $\Gamma$
   2. calculate L-J potential on $\Gamma$
   3. calculate curvature $\kappa$
   4. on $\Gamma$ let
   \[ V_n = -\gamma_0 \kappa + \rho_0 \sum_{i=1}^{N} U(|x-x_i|) \]
   5. extend $V_n$ to regular grid
   6. evolve LS function on regular grid
5. Stop when free energy stops decreasing
Algorithm

1. Define a regular grid
2. Define atoms (with radii $\sigma_i$)
3. Construct initial surface
4. At each time step:
   1. body-fit mesh to interface $\Gamma$
   2. calculate L-J potential on $\Gamma$
   3. calculate curvature $\kappa$ on $\Gamma$
   4. let $V_n = -\gamma_0 \kappa + \rho_0 \sum_{i=1}^{N} U(|x-x_i|)$
   5. extend $V_n$ to regular grid
   6. evolve LS function on regular grid
5. Stop when free energy stops decreasing
Body Fitted Meshes

- Refine mesh into interior and exterior regions.
- Add edges at zero level set
- Add edges to make conforming mesh
Body Fitted Meshes

- Can calculate length of interface easily
- Explicitly separates interior and exterior
- May need extra refinement in some applications
Reinitialization

What, when, and why?
What is reinitialization?

Redefining the level set function to make it nicer without moving the location of the interface (too much).
Reinitialization

What is reinitialization?
Redefining the level set function to make it nicer without moving the location of the interface (too much).

Why reinitialize?
Over time the level set function can become flat, causing errors in calculations of certain quantities.
What is reinitialization?
Redefining the level set function to make it nicer without moving the location of the interface (too much).

Why reinitialize?
Over time the level set function can become flat, causing errors in calculations of certain quantities.

When do we reinitialize?
Often enough that the level set function never becomes “bad”. Not so often to introduce more errors.
“Ideal” Reinitialization

- Doesn’t move the interface
- $|\nabla \phi| \approx 1$ (especially near the interface)
- Efficient
- Preserves key quantities near the interface (curvature, normal)
- Signed distance function?
“Ideal” Reinitialization

- Doesn’t move the interface
- $|\nabla \phi| \approx 1$ (especially near the interface)
- Efficient
- Preserves key quantities near the interface (curvature, normal)
- Signed distance function?
“Ideal” Reinitialization

- Doesn’t move the interface
- $|\nabla \phi| \approx 1$ (especially near the interface)
- Efficient
- Preserves key quantities near the interface (curvature, normal)
- Signed distance function?
“Ideal” Reinitialization

- Doesn’t move the interface
- $|\nabla \phi| \approx 1$ (especially near the interface)
- Efficient
- Preserves key quantities near the interface (curvature, normal)
- Signed distance function?
“Ideal” Reinitialization

- Doesn’t move the interface
- $|\nabla \phi| \approx 1$ (especially near the interface)
- Efficient
- Preserves key quantities near the interface (curvature, normal)
- Signed distance function?
Reinitialization Examples

- A model starting problem
Method 1

1. Body-fit mesh to interface
2. Solve

\[
\begin{align*}
\Delta \phi &= G_- & \text{in } \Omega_- \\
\Delta \phi &= G_+ & \text{in } \Omega_+ \\
\phi &= 0 & \text{on } \Gamma \\
\frac{\partial \phi}{\partial n} &= 0 & \text{on } \partial \Omega
\end{align*}
\]
Reinitialization, Poisson’s Equation Method 1

\[
\Delta \phi = 1 \quad \text{in } \Omega_-
\]
\[
\Delta \phi = -1 \quad \text{in } \Omega_+
\]
\[
\phi = 0 \quad \text{on } \Gamma
\]
\[
\frac{\partial \phi}{\partial n} = 0 \quad \text{on } \partial \Omega
\]
Reinitialization, Poisson’s Equation Method 1

Black: original interface
Red: new interface
Reinitialization, Poisson’s Equation Method 2

Method 2

1. Body-fit mesh to interface
2. Solve

\[
\begin{align*}
\Delta \phi &= G_- \quad \text{in } \Omega_- \\
\Delta \phi &= 0 \quad \text{in } \Omega_+ \\
\phi &= 0 \quad \text{on } \Gamma \\
\phi &= 1 \quad \text{on } \partial \Omega
\end{align*}
\]
Reinitialization, Poisson’s Equation Method 2

\[ \Delta \phi = 10 \quad \text{in } \Omega_- \]
\[ \Delta \phi = 0 \quad \text{in } \Omega_+ \]
\[ \phi = 0 \quad \text{on } \Gamma \]
\[ \phi = 1 \quad \text{on } \partial \Omega \]
Reinitialization, Poisson’s Equation Method 2
Constraint

Add additional constraint

\[
\left[ \frac{\partial \phi}{\partial n} \right]_e = 0
\]

for all new body-fitted edges \( e \), so unrefinement won’t move interface.

- Constraint is enforced by basis test functions along interface
- Overdetermined system can only be solved in a least squares way
- Constraint can be weighted more
Constrained Gradient Jump Across Interface Method

Constraint

Add additional constraint

\[
\left[ \frac{\partial \phi}{\partial n} \right]_e = 0
\]

for all new body-fitted edges \( e \), so unrefinement won’t move interface.

- Constraint is enforced by basis test functions along interface
- Overdetermined system can only be solved in a least squares way
- Constraint can be weighted more
Constraint

Add additional constraint

\[
\left[ \frac{\partial \phi}{\partial n} \right]_e = 0
\]

for all new body-fitted edges \( e \), so unrefinement won’t move interface.

- Constraint is enforced by basis test functions along interface
- Overdetermined system can only be solved in a least squares way
- Constraint can be weighted more
Constraint

Add additional constraint

\[
\left[ \frac{\partial \phi}{\partial n} \right]_e = 0
\]

for all new body-fitted edges \( e \), so unrefinement won’t move interface.

- Constraint is enforced by basis test functions along interface
- Overdetermined system can only be solved in a least squares way
- Constraint can be weighted more
Constrained Gradient Jump Comparison

Black: original interface
Red: new interface
2 atom systems: $1.6\sigma$, $2.0\sigma$, $2.4\sigma$, $2.8\sigma$.

- $\sigma = 1.6$
  - Energy: $-3.111 \times 10^{-5}$

- $\sigma = 2.0$
  - Energy: $-2.543 \times 10^{-5}$

- $\sigma = 2.4$
  - Energy: $-2.172 \times 10^{-5}$

- $\sigma = 2.8$
  - Energy: $-3.283 \times 10^{-5}$
Conclusion

Future work:

- Implement 3 dimensional algorithm
- Speed up algorithm (various ways)
- Add more terms to free energy functional