Level Set Method in a Finite Element Setting

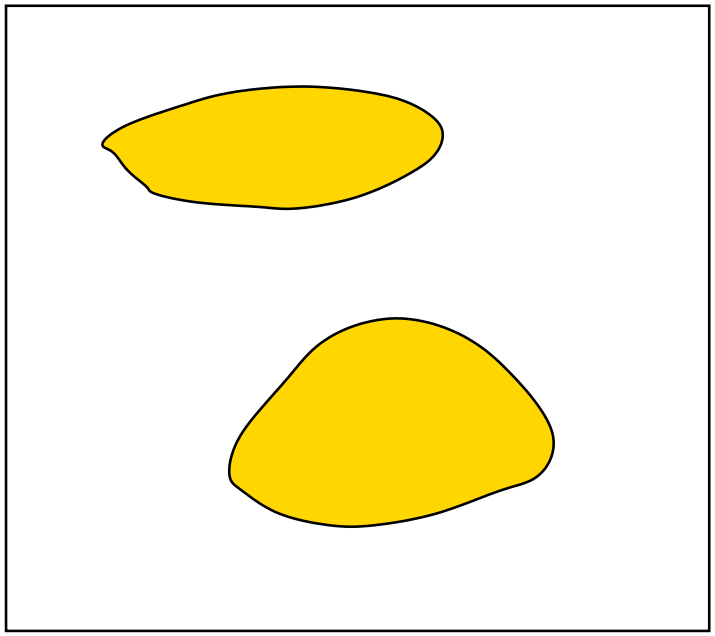
John Shopple

University of California, San Diego

November 6, 2007

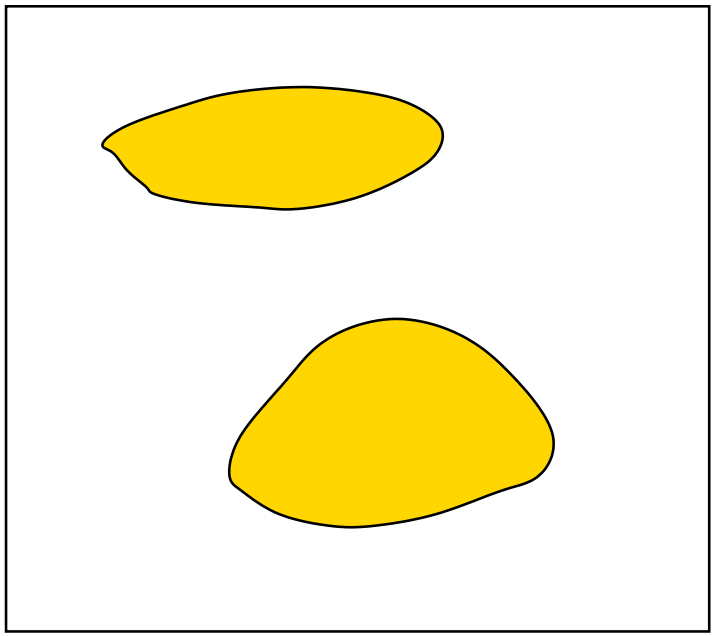
Outline

Types of problems



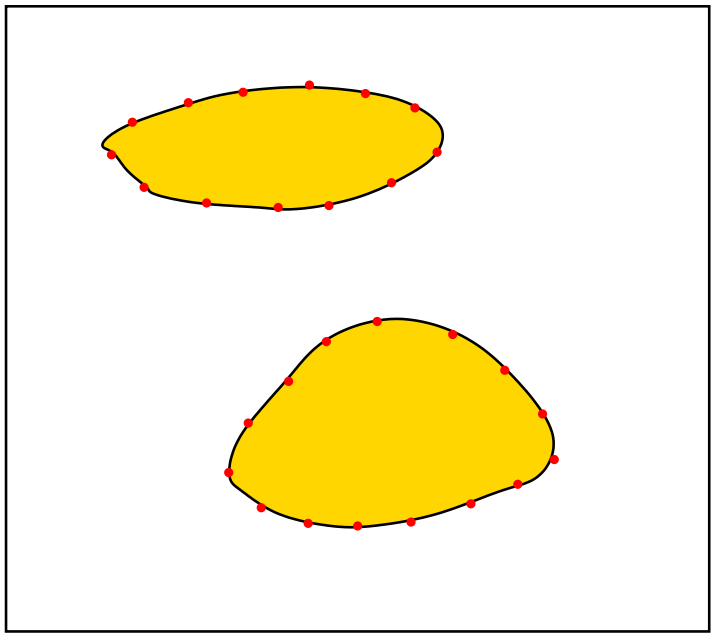
- Geometric based motion
  - Motion based on curvature
  - Normal direction motion
  - Simple “constant” motion
- Physical based motion
  - Fluid dynamics
    - Two-phase flow
    - Combustion
    - Liquid-gas interactions
  - Biology
    - Biomolecular surfaces
    - Membranes
  - Material science
    - Solidification
    - Epitaxial growth of thin films

Front Tracking Methods



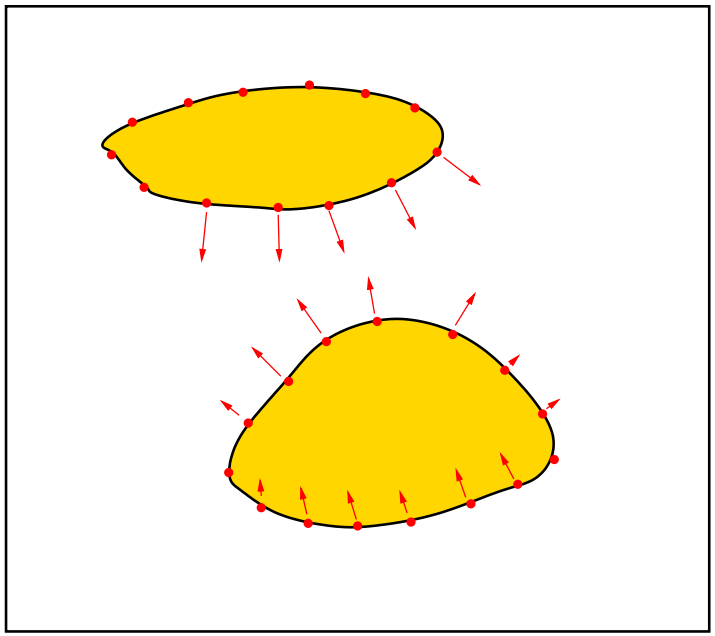
1. Place markers on interface
2. Calculate velocity for each marker
3. Propagate markers one time step
4. Check for topological changes of interfaces
   - add/remove markers
   - split/join interfaces if necessary
5. Go to step 2

Front Tracking Methods



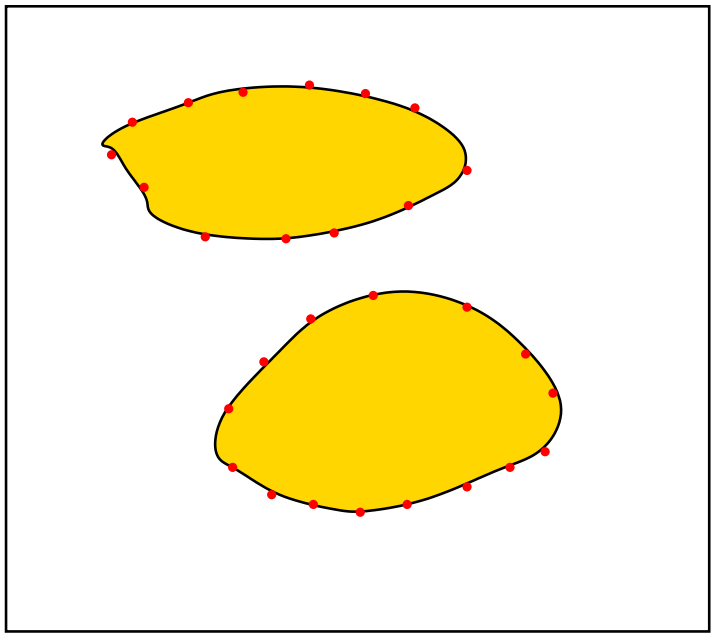
1. Place markers on interface
2. Calculate velocity for each marker
3. Propagate markers one time step
4. Check for topological changes of interfaces
   - add/remove markers
   - split/join interfaces if necessary
5. Go to step 2

Front Tracking Methods



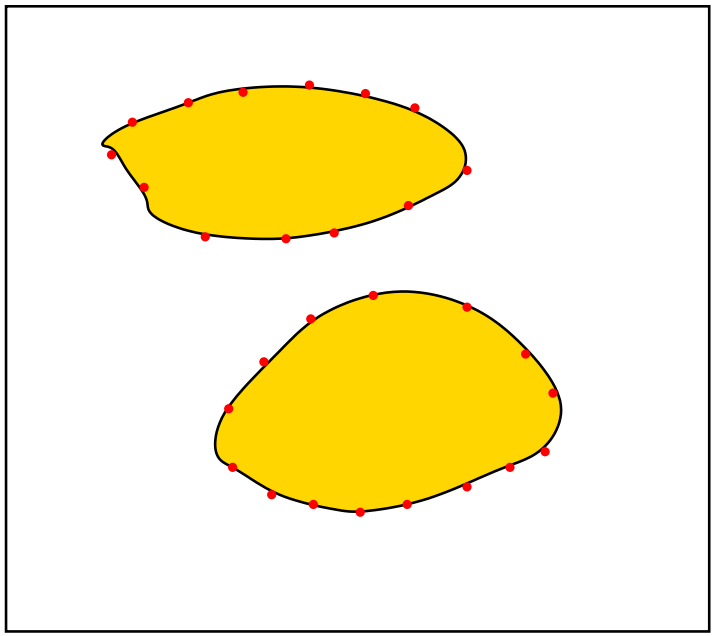
1. Place markers on interface
2. Calculate velocity for each marker
3. Propagate markers one time step
4. Check for topological changes of interfaces
   - add/remove markers
   - split/join interfaces if necessary
5. Go to step 2

Front Tracking Methods



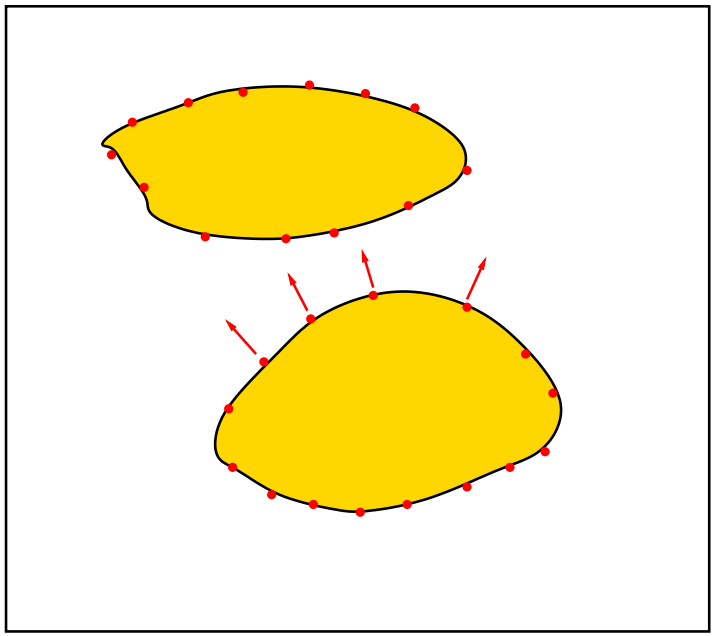
1. Place markers on interface
2. Calculate velocity for each marker
3. Propagate markers one time step
4. Check for topological changes of interfaces
   - add/remove markers
   - split/join interfaces if necessary
5. Go to step 2

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
5. Go to step 2

Front Tracking Methods



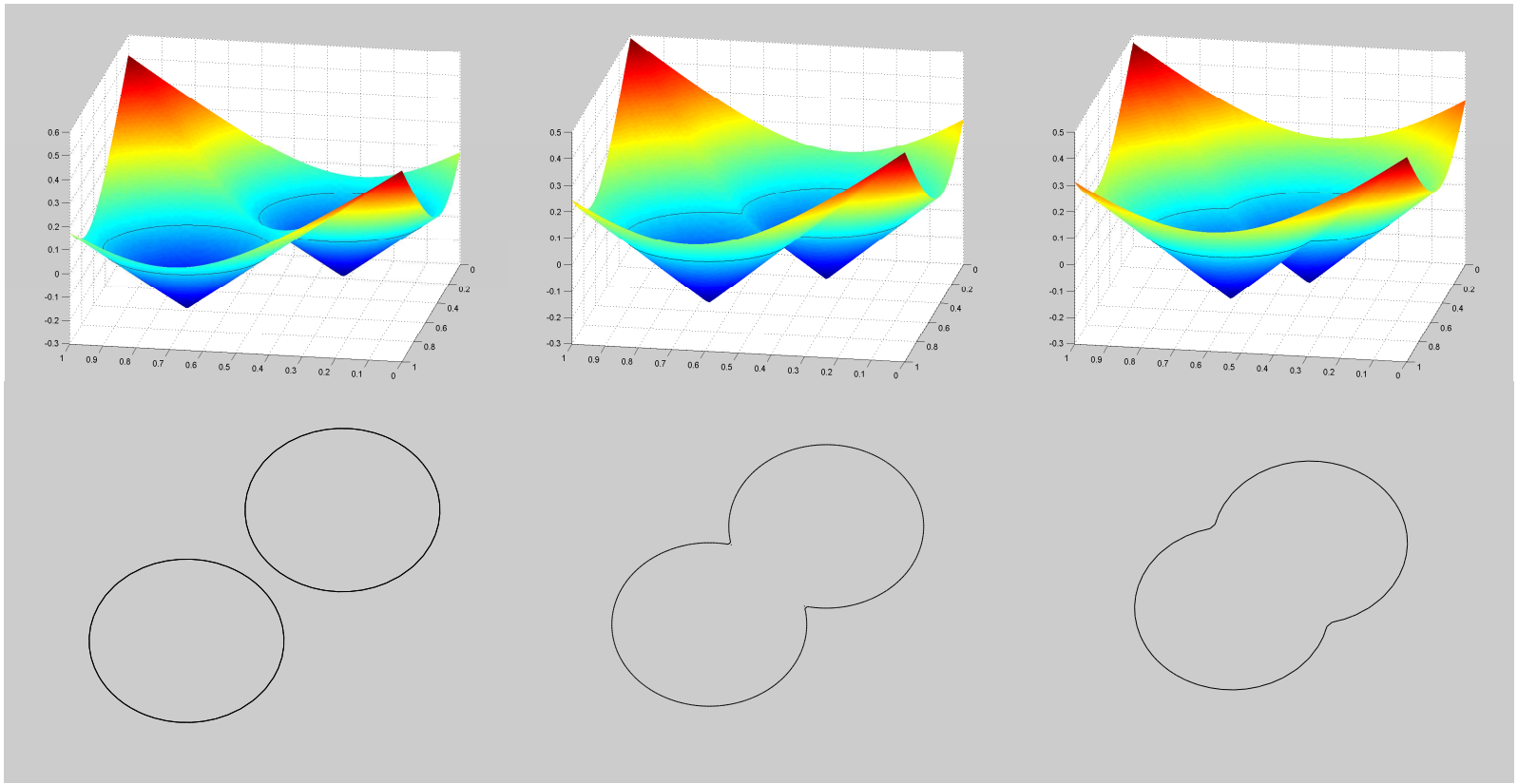
1. Place markers on interface
2. Calculate velocity for each marker
3. Propagate markers one time step
4. Check for topological changes of interfaces
   - add/remove markers
   - split/join interfaces if necessary
5. Go to step 2

Level Set Method

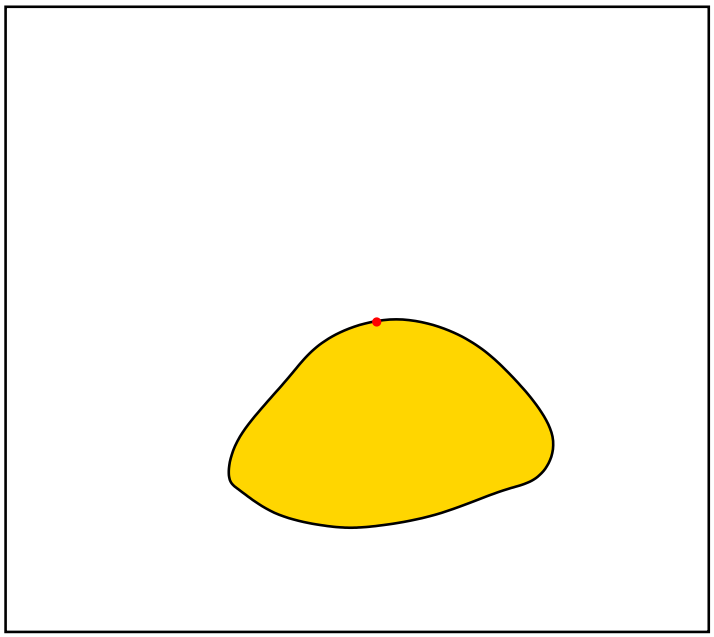
The interface is represented as the zero level set of a function

$$\phi(x, y, t) = 0$$

Topological changes will occur naturally as time progresses.



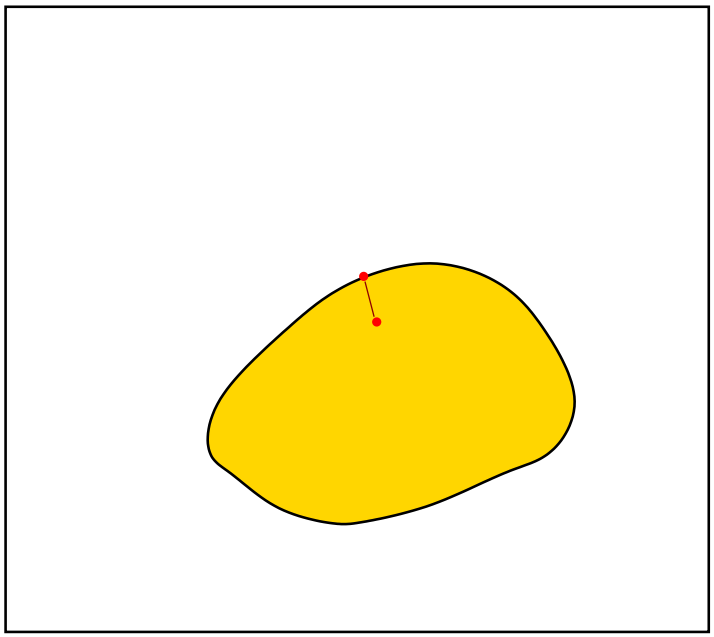
Derivation of the Level Set Evolution Equation



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

- $\phi(x(0), y(0), t = 0) = 0$
- $\phi(x(1), y(1), t = 1) = 0$
- $\phi(x(2), y(2), t = 2) = 0$

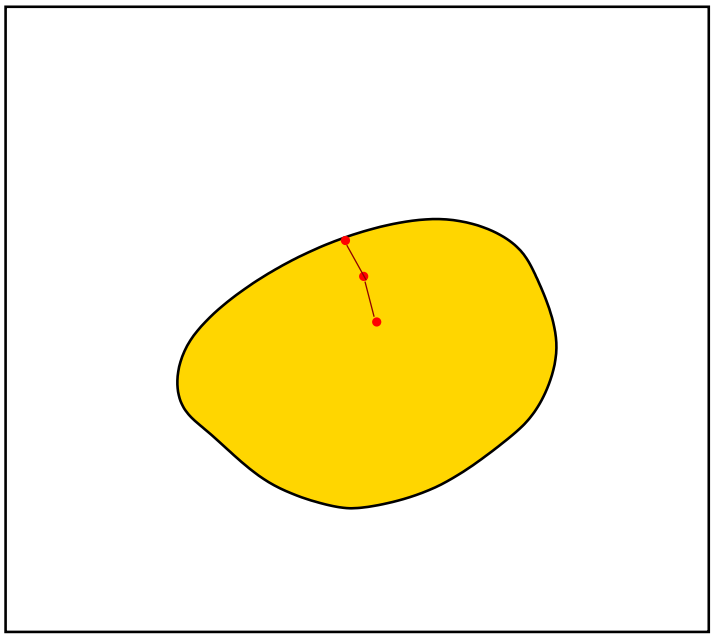
Derivation of the Level Set Evolution Equation



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

- $\phi(x(0), y(0), t = 0) = 0$
- $\phi(x(1), y(1), t = 1) = 0$
- $\phi(x(2), y(2), t = 2) = 0$

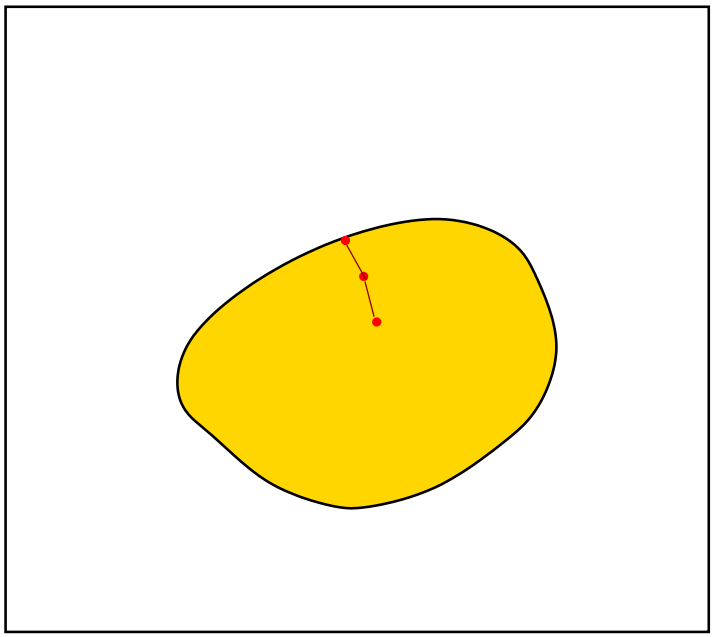
Derivation of the Level Set Evolution Equation



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

- $\phi(x(0), y(0), t = 0) = 0$
- $\phi(x(1), y(1), t = 1) = 0$
- $\phi(x(2), y(2), t = 2) = 0$

Derivation of the Level Set Evolution Equation



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

- $\phi(x(0), y(0), t = 0) = 0$
- $\phi(x(1), y(1), t = 1) = 0$
- $\phi(x(2), y(2), t = 2) = 0$

$$\begin{aligned}\frac{d\phi}{dt} &= \phi_t + \phi_x x'(t) + \phi_y y'(t) \\ &= \phi_t + \nabla\phi \cdot \big(x'(t), y'(t)\big) \\ &= \phi_t + \nabla\phi \cdot \mathbf{V} = 0\end{aligned}$$

Level Set Evolution Equation

For externally generated velocity fields:

$$\phi_t + \nabla\phi \cdot \mathbf{V} = 0$$

For internally generated velocity fields:

$$\phi_t + |\nabla\phi| V_n = 0$$

where

$$V_n = \mathbf{V} \cdot \mathbf{n} = \mathbf{V} \cdot \frac{\nabla\phi}{|\nabla\phi|}$$

Interface $= \Gamma(t) = \{(x,y) \mid \phi(x,y,t) = 0\}$

Basic Types of Interface Motion

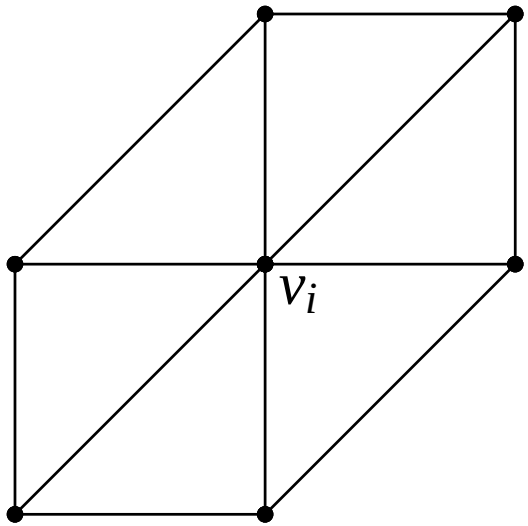
- Vector field $\mathbf{V}$
- Scalar field V_n giving speed in level sets' normal direction
- V_n proportional to level set curvature $\kappa = \nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|} \right)$

Local Method to Solve Level Set Equation

Evolution equation

$$\phi_t + V_n|\nabla\phi| = 0$$

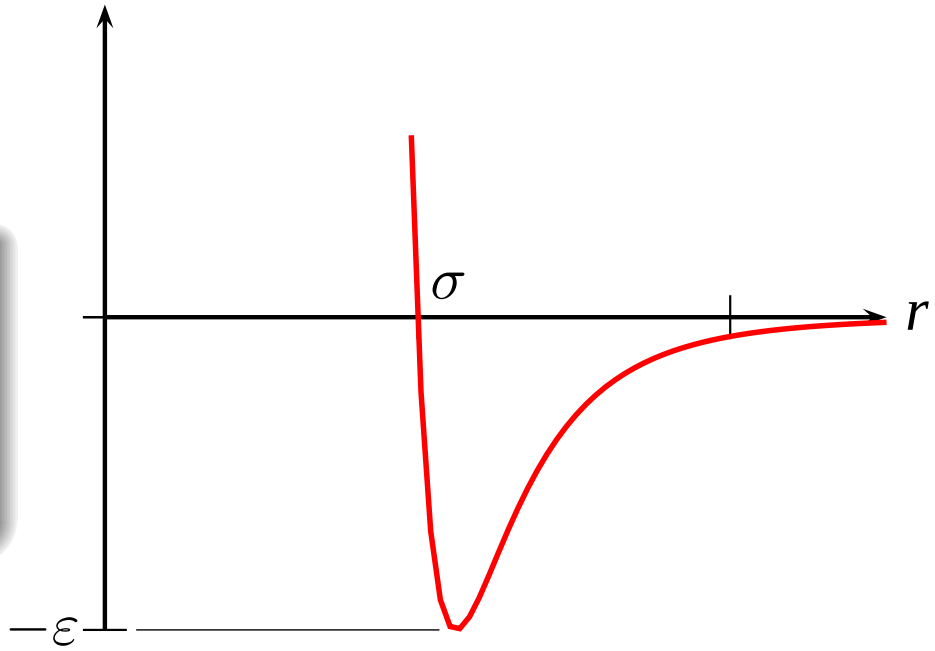
1. Approximate $|\nabla\phi|$ at each vertex v_i by the average of $|\nabla\phi|$ over a patch around v_i
2. Use a Runga-Kutta type method using the values of V_n at each vertex.



Lennard-Jones Potential

Lennard-Jones potential

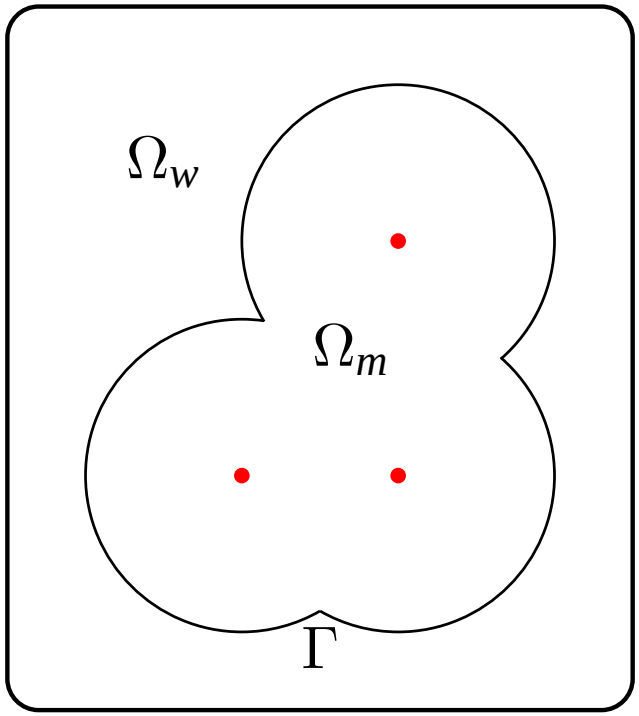
$$U(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$



- $U(r = \sigma) = 0$
- Minimum:
  $U(r = \sigma\sqrt[6]{2} \approx 1.12\sigma) = -\varepsilon$
- $r > \sigma$ attracts
- $r < \sigma$ repels

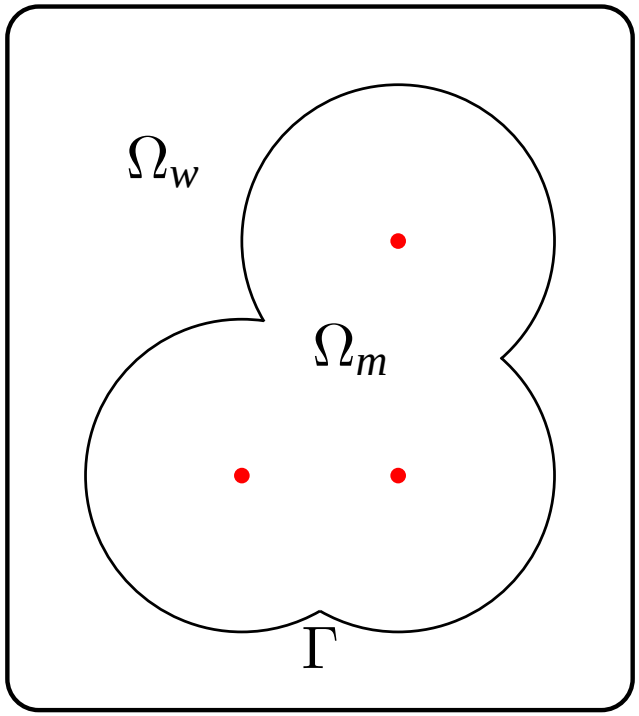
Solute-Solvent Model

- Molecules consist of atoms at locations $\mathbf{x}_1, \ldots, \mathbf{x}_N$
- Solvent region Ω_w
- Solute region Ω_m
- A solute-solvent interface Γ



Solute-Solvent Model

- Molecules consist of atoms at locations $\mathbf{x}_1, \ldots, \mathbf{x}_N$
- Solvent region Ω_w
- Solute region Ω_m
- A solute-solvent interface Γ



A free energy model

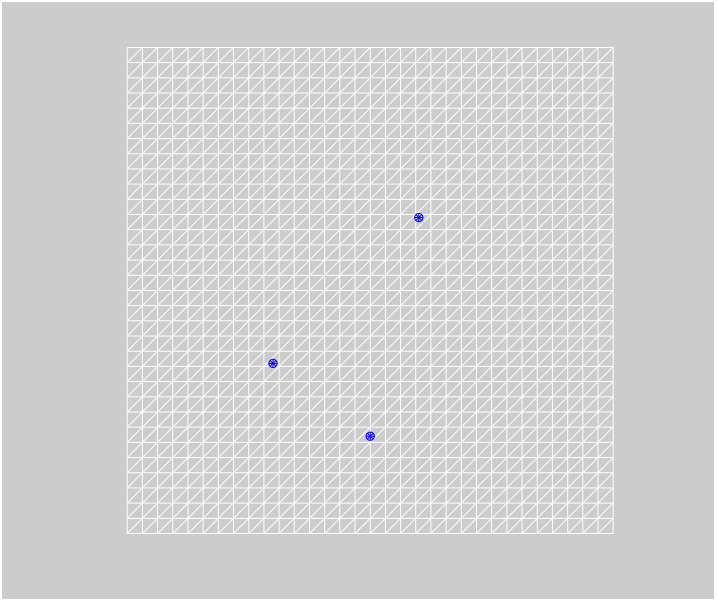
$$G[\Gamma; \mathbf{x}_1, \ldots, \mathbf{x}_N] = \int_\Gamma \gamma \, dS + \rho_w \sum_{i=1}^{N} \int_{\Omega_w} U_{sw}(|\mathbf{x} - \mathbf{x}_i|) \, dV$$

Algorithm

1. Define a regular grid
2. Define atoms (with radii σ_i)
3. Construct initial surface
4. At each time step:
   1. body-fit mesh to interface Γ
   2. calculate L-J potential on Γ
   3. calculate curvature κ
   4. on Γ 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

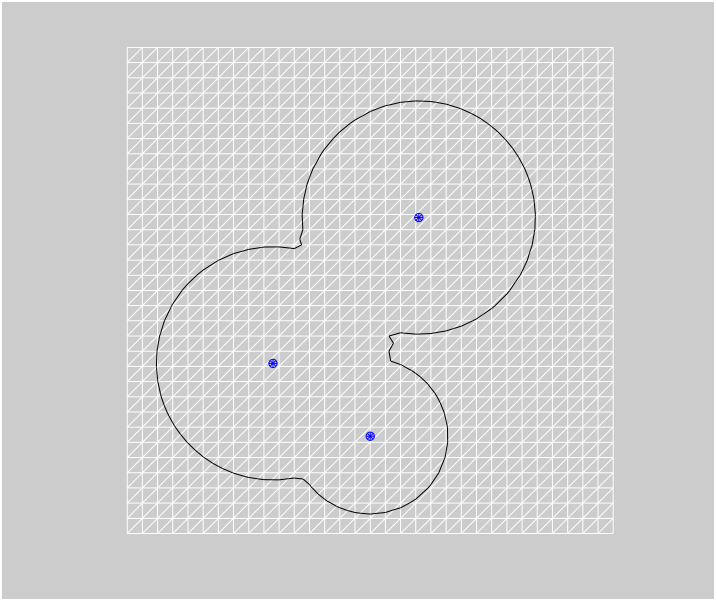
Algorithm

1. Define a regular grid
2. Define atoms (with radii σ_i)
3. Construct initial surface
4. At each time step:
   1. body-fit mesh to interface Γ
   2. calculate L-J potential on Γ
   3. calculate curvature κ
   4. on Γ 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



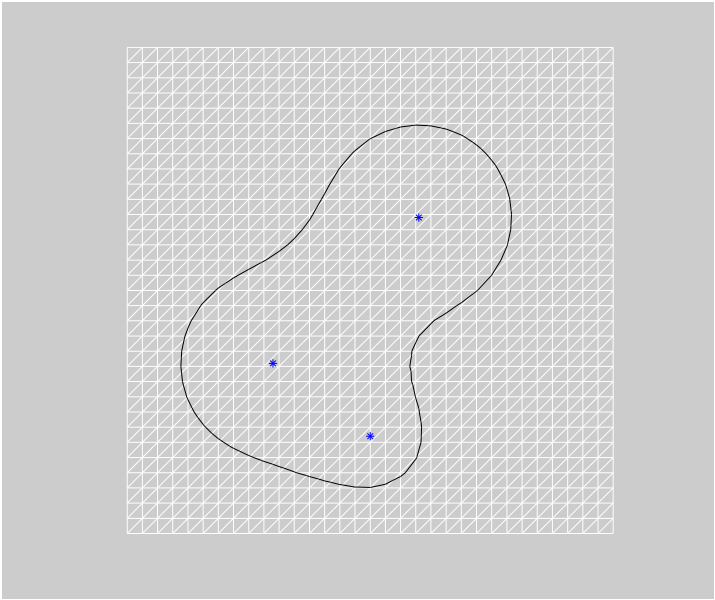
Algorithm

1. Define a regular grid
2. Define atoms (with radii σ_i)
3. Construct initial surface
4. At each time step:
   1. body-fit mesh to interface Γ
   2. calculate L-J potential on Γ
   3. calculate curvature κ
   4. on Γ 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



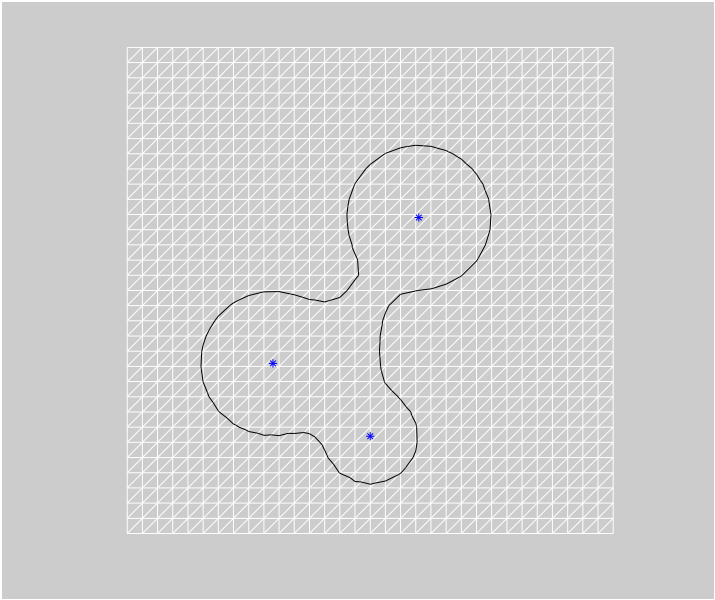
Algorithm

1. Define a regular grid
2. Define atoms (with radii σ_i)
3. Construct initial surface
4. At each time step:
   1. body-fit mesh to interface Γ
   2. calculate L-J potential on Γ
   3. calculate curvature κ
   4. on Γ 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



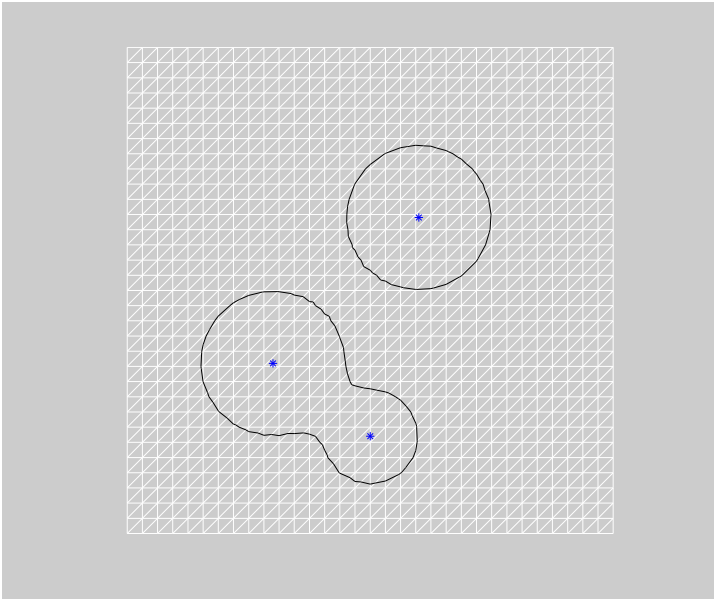
Algorithm

1. Define a regular grid
2. Define atoms (with radii σ_i)
3. Construct initial surface
4. At each time step:
   1. body-fit mesh to interface Γ
   2. calculate L-J potential on Γ
   3. calculate curvature κ
   4. on Γ 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

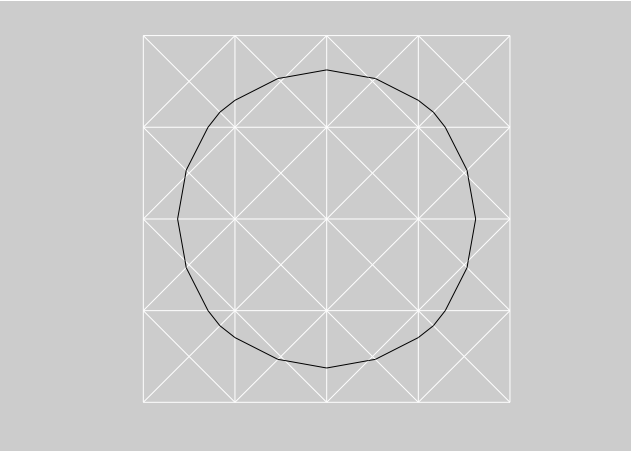


Algorithm

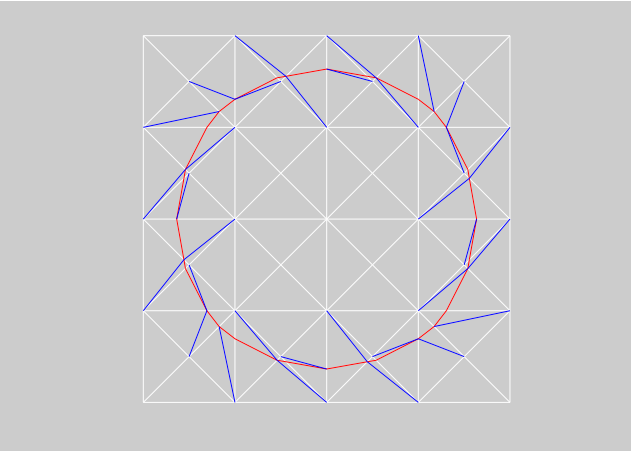
1. Define a regular grid
2. Define atoms (with radii σ_i)
3. Construct initial surface
4. At each time step:
   1. body-fit mesh to interface Γ
   2. calculate L-J potential on Γ
   3. calculate curvature κ
   4. on Γ let $V_n = -\gamma_0\kappa + \rho_0 \sum_{i=1}^{N} U(|\mathbf{x}-\mathbf{x}_i|)$
 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

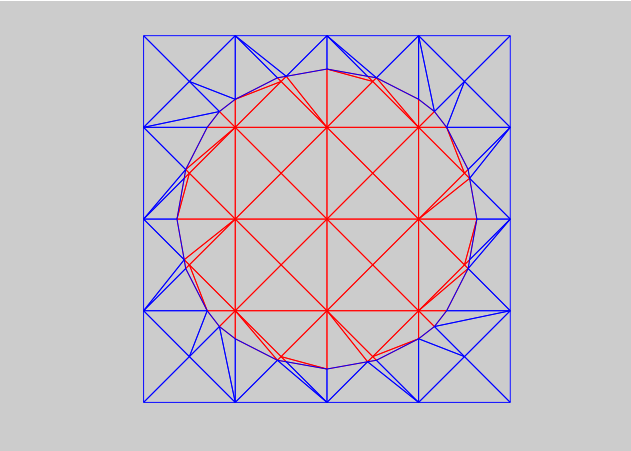


Body Fitted Meshes

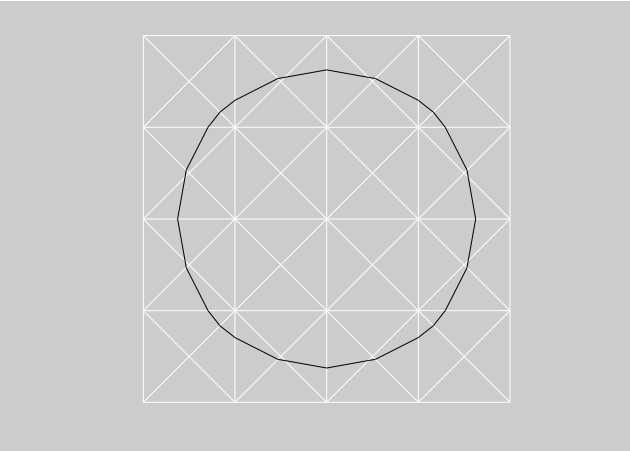


- Refine mesh into interior and exterior regions.
- Add edges at zero level set
- Add edges to make conforming mesh

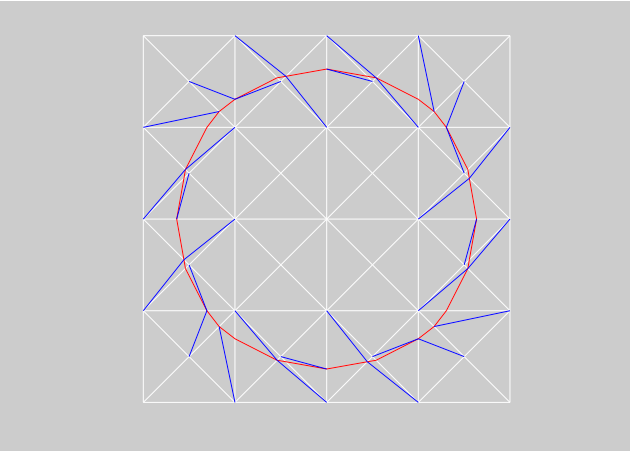


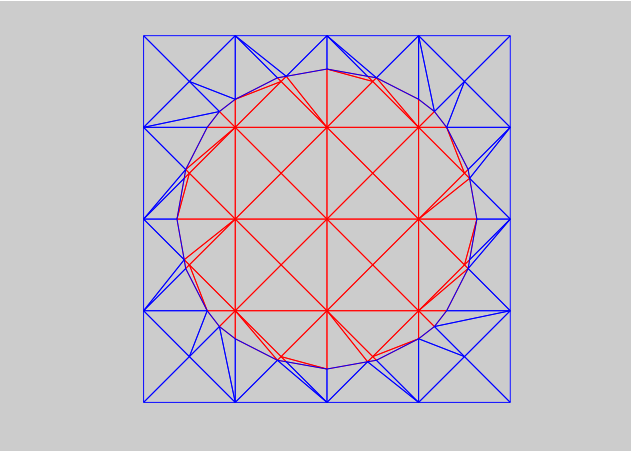


Body Fitted Meshes



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





Reinitialization

What, when, and why?

Reinitialization

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.

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.

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

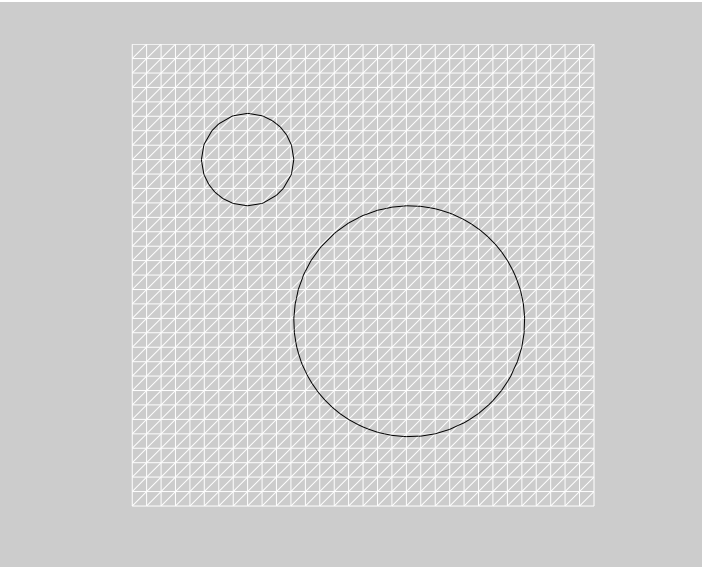
- Doesn’t move the interface
- $|\nabla\phi| \approx 1$ (especially near the interface)
- Efficient
- Preserves key quantities near the interface (curvature, normal)
- Signed distance function?

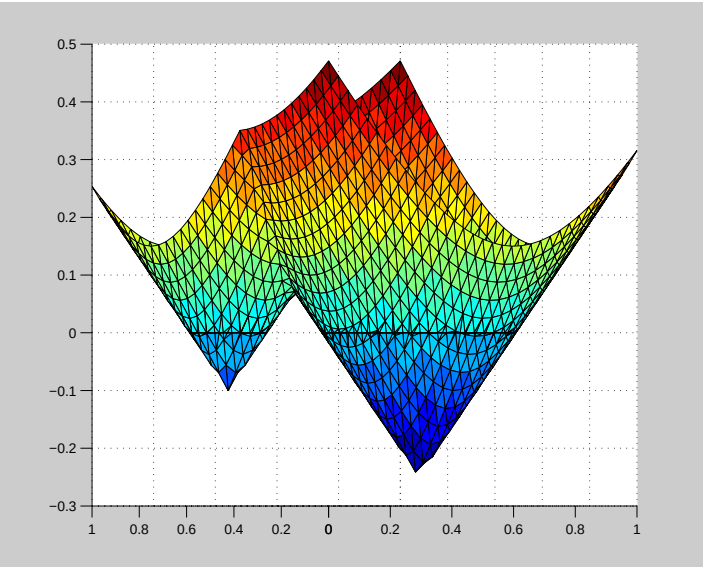
“Ideal” Reinitialization

- 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





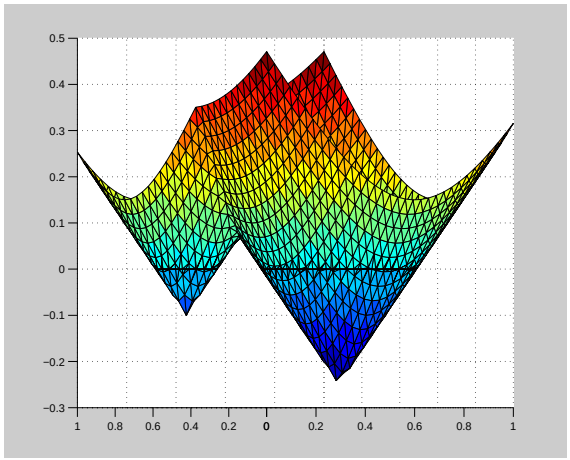
Reinitialization, Poisson's Equation Method 1

Method 1

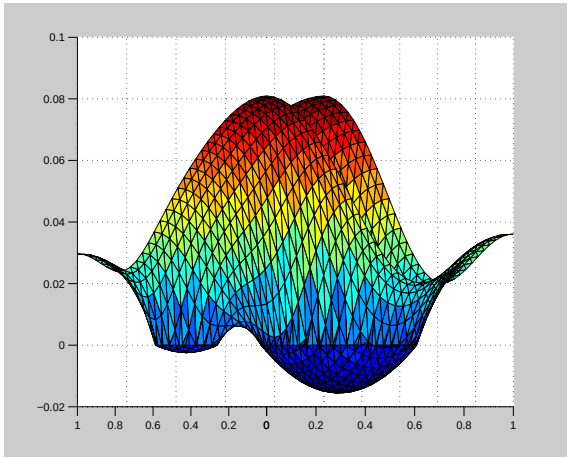
1. Body-fit mesh to interface
2. Solve

$$
\begin{aligned}
\Delta\phi &= G_- && \text{in } \Omega_- \\
\Delta\phi &= G_+ && \text{in } \Omega_+ \\
\phi &= 0 && \text{on } \Gamma \\
\frac{\partial\phi}{\partial n} &= 0 && \text{on } \partial\Omega
\end{aligned}
$$

Reinitialization, Poisson's Equation Method 1

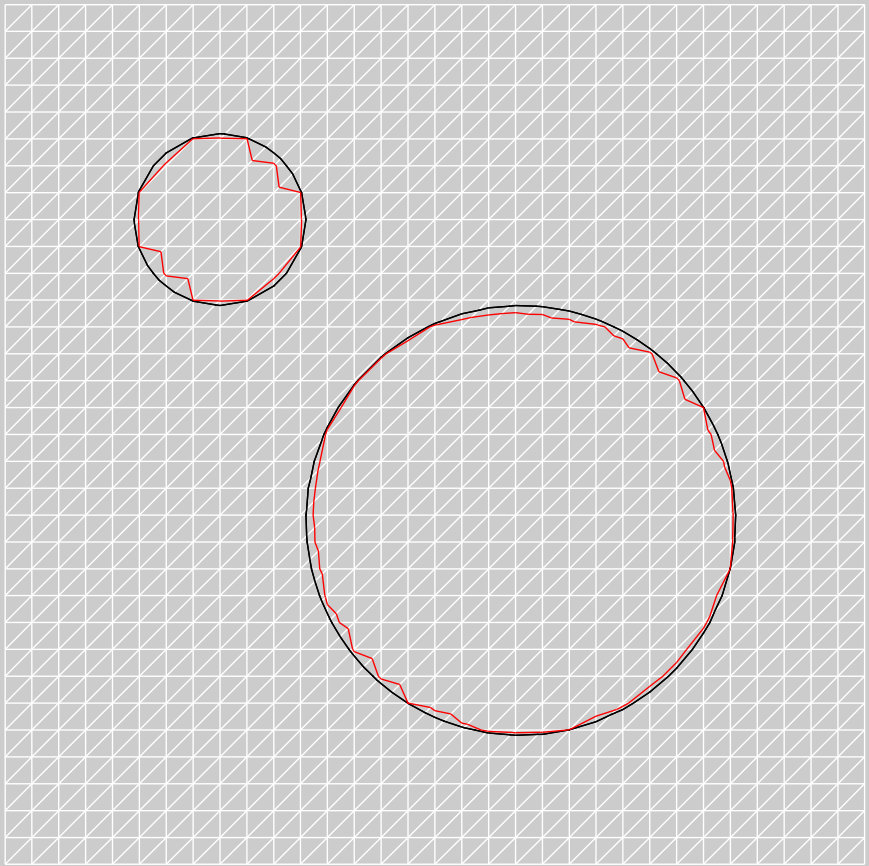


$$\begin{aligned} \Delta\phi &= 1 && \text{in } \Omega_- \\ \Delta\phi &= -1 && \text{in } \Omega_+ \\ \phi &= 0 && \text{on } \Gamma \\ \frac{\partial\phi}{\partial n} &= 0 && \text{on } \partial\Omega \end{aligned}$$



Reinitialization, Poisson's Equation Method 1

Black: original interface
Red: new interface



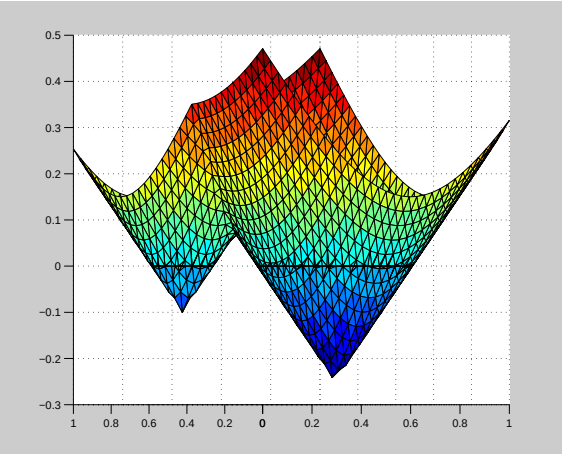
Reinitialization, Poisson's Equation Method 2

Method 2

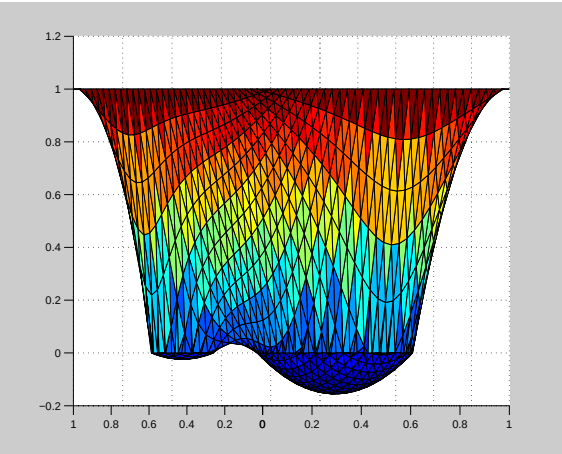
1. Body-fit mesh to interface
2. Solve

$$\begin{aligned} \Delta\phi &= G_- && \text{in } \Omega_- \\ \Delta\phi &= 0 && \text{in } \Omega_+ \\ \phi &= 0 && \text{on } \Gamma \\ \phi &= 1 && \text{on } \partial\Omega \end{aligned}$$

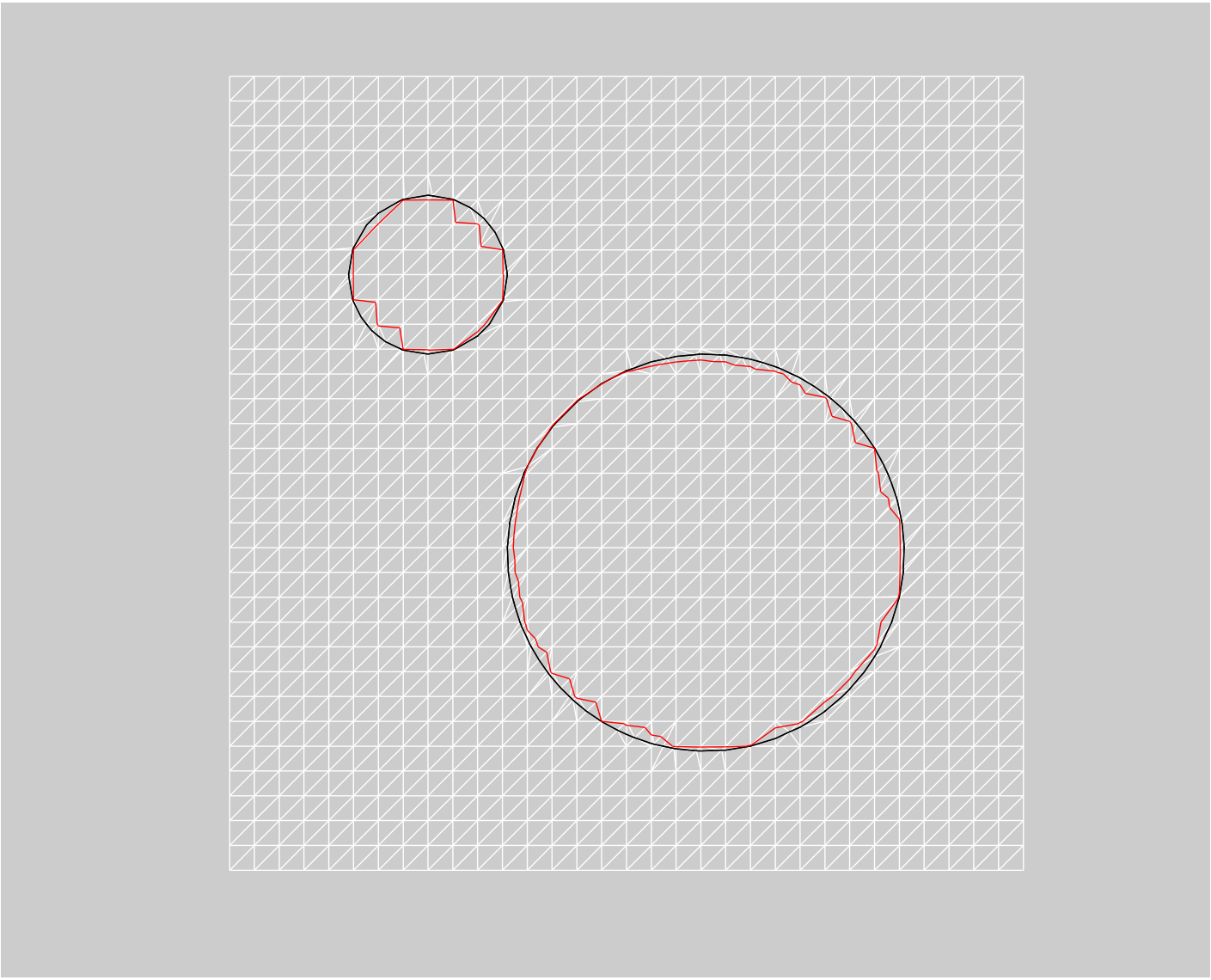
Reinitialization, Poisson's Equation Method 2



$$\begin{aligned} \Delta\phi &= 10 \quad \text{in } \Omega_- \\ \Delta\phi &= 0 \quad \text{in } \Omega_+ \\ \phi &= 0 \quad \text{on } \Gamma \\ \phi &= 1 \quad \text{on } \partial\Omega \end{aligned}$$



Reinitialization, Poisson's Equation Method 2



Constrained Gradient Jump Across Interface Method

Constraint

Add additional constraint

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

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

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

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

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

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

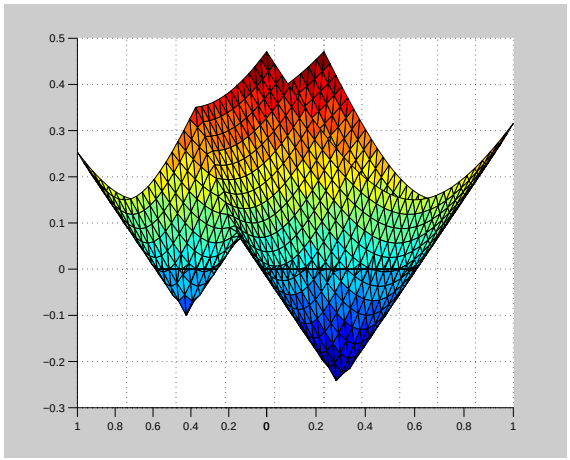
Add additional constraint

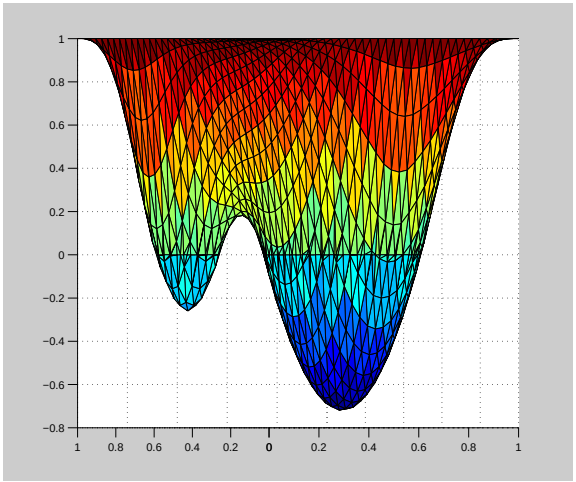
$$\left[\!\left[\frac{\partial \phi}{\partial n}\right]\!\right]_e = 0$$

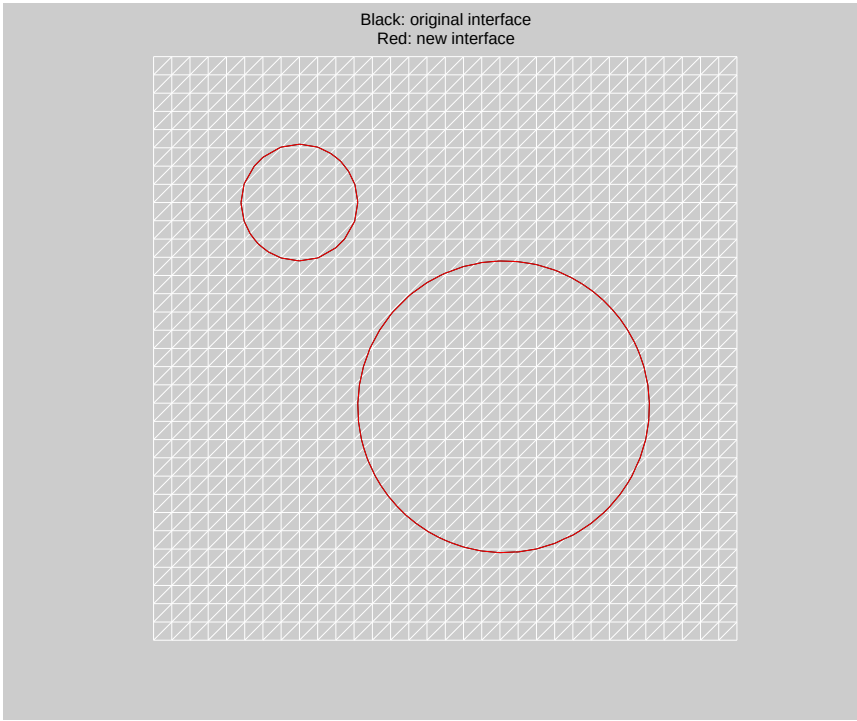
for all new body-fitted edges e, so unrefinement won't move interface.

- 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

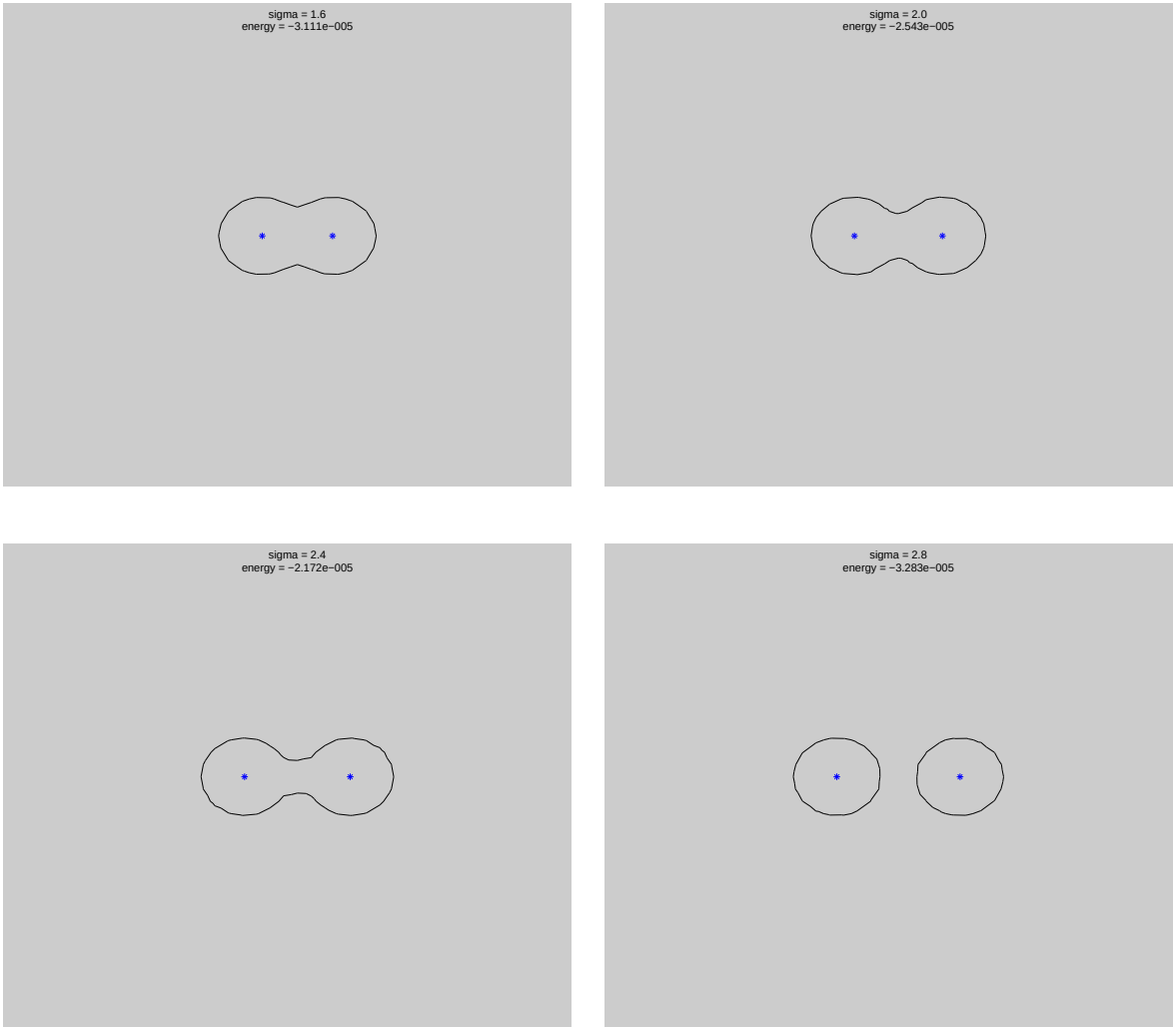






Solute-Solvent Model Examples

2 atom systems: 1.6σ, 2.0σ, 2.4σ, 2.8σ.



Conclusion

Future work:

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