The Level Set Method

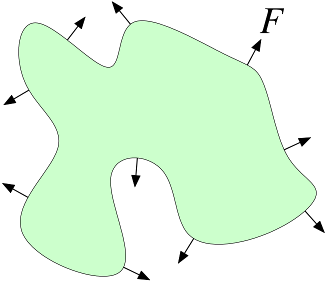
Per-Olof Persson
persson@berkeley.edu

Department of Mathematics
University of California, Berkeley

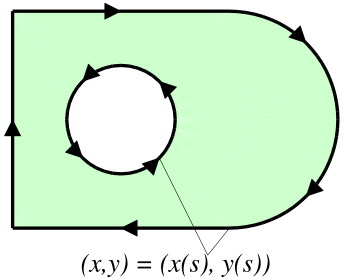
Math 228B Numerical Solutions of Differential Equations

Evolving Curves and Surfaces

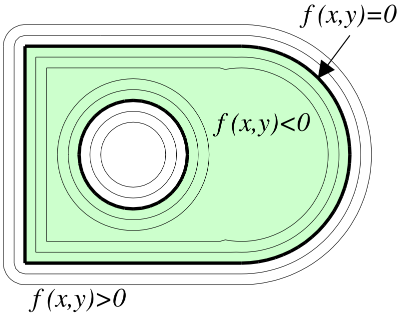
- Propagate curve according to speed function $\boldsymbol{v} = F\boldsymbol{n}$
- F depends on space, time, and the curve itself
- Surfaces in three dimensions



Explicit Geometry
Parameterized boundaries



Implicit Geometry
Boundaries given by zero levelset



Explicit Techniques

- Simple approach: Represent curve explicitly by a set of nodes $\{\boldsymbol{x}^{(i)}\}$, connected by lines or splines
- Propagate curve by solving ODEs

$$\frac{d\boldsymbol{x}^{(i)}}{dt} = \boldsymbol{v}(\boldsymbol{x}^{(i)}, t), \quad \boldsymbol{x}^{(i)}(0) = \boldsymbol{x}_0^{(i)},$$

- Calculate normal vectors, curvatures, etc by difference approximations, e.g.:

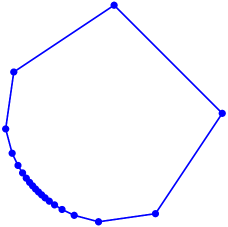
$$\frac{d\boldsymbol{x}^{(i)}}{ds} \approx \frac{\boldsymbol{x}^{(i+1)} - \boldsymbol{x}^{(i-1)}}{2\Delta s}$$

- MATLAB Demo

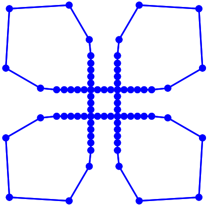
Explicit Techniques - Drawbacks

- Node redistribution required, introduces errors
- No entropy solution, sharp corners handled incorrectly
- Need special treatment for topology changes
- Stability constraints for curvature dependent speed functions

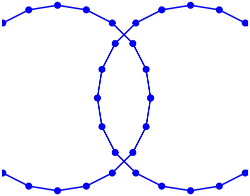
Node distribution



Sharp corners



Topology changes

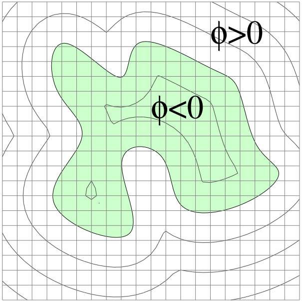


The Level Set Method

- Implicit geometries, evolve interface by solving PDEs
- Invented in 1988 by Osher and Sethian:
  - Stanley Osher and James A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.
- Introductory books:
  - James A. Sethian. *Level set methods and fast marching methods*. Cambridge University Press, Cambridge, second edition, 1999.
  - Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*. Springer-Verlag, New York, 2003.

Implicit Geometries

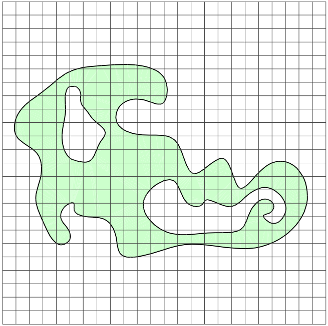
- Represent curve by zero level set of a function, $\phi(\boldsymbol{x}) = 0$
- Special case: *Signed distance function*:
  - $|\nabla\phi| = 1$
  - $|\phi(\boldsymbol{x})|$ gives (shortest) distance from $\boldsymbol{x}$ to curve



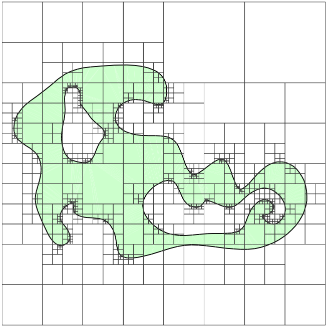
Discretized Implicit Geometries

- Discretize implicit function ϕ on *background grid*
- Obtain $\phi(\boldsymbol{x})$ for general $\boldsymbol{x}$ by interpolation

Cartesian



Quadtree/Octree



Geometric Variables

- Normal vector $\boldsymbol{n}$ (without assuming distance function):

$$\boldsymbol{n} = \frac{\nabla \phi}{|\nabla \phi|}$$

- Curvature (in two dimensions):

$$\kappa = \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} = \frac{\phi_{xx}\phi_y^2 - 2\phi_y\phi_x\phi_{xy} + \phi_{yy}\phi_x^2}{(\phi_x^2 + \phi_y^2)^{3/2}}.$$

- Write material parameters, etc, in terms of ϕ:

$$\rho(\boldsymbol{x}) = \rho_1 + (\rho_2 - \rho_1)\theta(\phi(\boldsymbol{x}))$$

Smooth Heaviside function θ over a few grid cells.

The Level Set Equation

- Solve convection equation to propagate $\phi = 0$ by velocities $\boldsymbol{v}$

$$\phi_t + \boldsymbol{v} \cdot \nabla\phi = 0.$$

- For $\boldsymbol{v} = F\boldsymbol{n}$, use $\boldsymbol{n} = \nabla\phi / |\nabla\phi|$ and $\nabla\phi \cdot \nabla\phi = |\nabla\phi|^2$ to obtain the *Level Set Equation*

$$\phi_t + F|\nabla\phi| = 0.$$

- Nonlinear, hyperbolic equation (Hamilton-Jacobi).

Discretization

- Use upwinded finite difference approximations for convection
- For the level set equation $\phi_t + F|\nabla\phi| = 0$:

$$\phi_{ijk}^{n+1} = \phi_{ijk}^{n} - \Delta t \left(\max(F,0)\nabla_{ijk}^{+} + \min(F,0)\nabla_{ijk}^{-} \right),$$

where

$$\begin{aligned}\nabla_{ijk}^{+} = \big[&\max(D^{-x}\phi_{ijk}^{n},0)^2 + \min(D^{+x}\phi_{ijk}^{n},0)^2 + \\ &\max(D^{-y}\phi_{ijk}^{n},0)^2 + \min(D^{+y}\phi_{ijk}^{n},0)^2 + \\ &\max(D^{-z}\phi_{ijk}^{n},0)^2 + \min(D^{+z}\phi_{ijk}^{n},0)^2 \big]^{1/2},\end{aligned}$$

Discretization

and

$$\begin{aligned}\nabla^-_{ijk} = \Big[&\min(D^{-x}\phi^n_{ijk},0)^2 + \max(D^{+x}\phi^n_{ijk},0)^2 + \\ &\min(D^{-y}\phi^n_{ijk},0)^2 + \max(D^{+y}\phi^n_{ijk},0)^2 + \\ &\min(D^{-z}\phi^n_{ijk},0)^2 + \max(D^{+z}\phi^n_{ijk},0)^2\Big]^{1/2}.\end{aligned}$$

- D^{-x} backward difference operator in the x-direction, etc
- For curvature dependent part of F, use central differences
- Higher order schemes available
- MATLAB Demo

Reinitialization

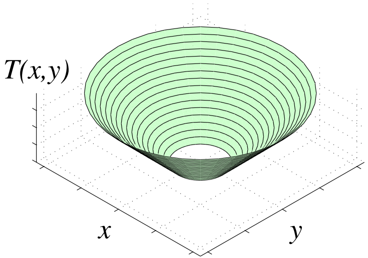
- Even if the initial level set function is a distance function, general speed functions F will give large variations in $|\nabla\phi|$
- This gives poor accuracy and performance, and requires smaller timesteps for stability
- *Reinitialize* the level set function by finding a new ϕ with same zero level set but $|\nabla\phi| = 1$
- Different approaches:
  1. Integrate the *reinitialization equation* for a few time steps
     $$\phi_t + \operatorname{sign}(\phi)(|\nabla\phi| - 1) = 0$$
  2. Compute distances from $\phi = 0$ explicitly for nodes close to boundary, use Fast Marching Method for remaining nodes

The Boundary Value Formulation

- For $F > 0$, formulate evolution by an arrival function T
- $T(\boldsymbol{x})$ gives time to reach $\boldsymbol{x}$ from initial Γ
- time * rate = distance gives the *Eikonal equation*:

$$|\nabla T| F = 1, \quad T = 0 \text{ on } \Gamma.$$

- Special case: $F = 1$ gives distance functions



The Fast Marching Method

- Discretize the Eikonal equation $|\nabla T|F = 1$ by

$$\left[\begin{array}{r} \max(D^{-x}_{ijk}T, 0)^2 + \min(D^{+x}_{ijk}T, 0)^2 \\ + \max(D^{-y}_{ijk}T, 0)^2 + \min(D^{+y}_{ijk}T, 0)^2 \\ + \max(D^{-z}_{ijk}T, 0)^2 + \min(D^{+z}_{ijk}T, 0)^2 \end{array}\right]^{1/2} = \frac{1}{F_{ijk}}$$

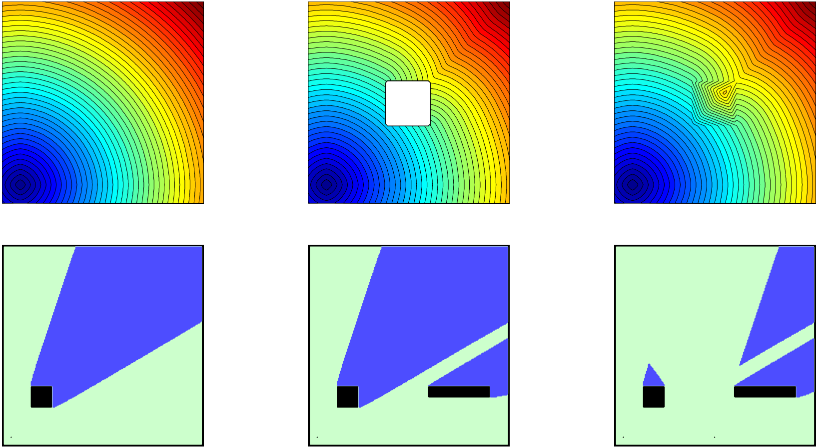
  or

$$\left[\begin{array}{r} \max(D^{-x}_{ijk}T, -D^{+x}_{ijk}T, 0)^2 \\ + \max(D^{-y}_{ijk}T, -D^{+y}_{ijk}T, 0)^2 \\ + \max(D^{-z}_{ijk}T, -D^{+z}_{ijk}T, 0)^2 \end{array}\right]^{1/2} = \frac{1}{F_{ijk}}$$

The Fast Marching Method

- Use the fact that the front propagates outward
- Tag known values and update neighboring T values (using the difference approximation)
- Pick unknown with smallest T (will not be affected by other unknowns)
- Update new neighbors and repeat until all nodes are known
- Store unknowns in priority queue, $\mathcal{O}(n \log n)$ performance for n nodes with heap implementation

Application: First arrivals and shortest geodesic paths



Visibility around obstacles

Application: Structural Vibration Control

- Consider eigenvalue problem

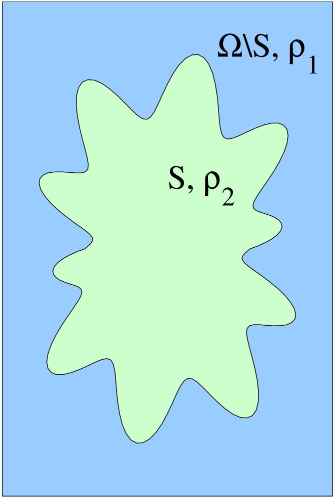
$$
\begin{aligned}
-\Delta u &= \lambda \rho(\boldsymbol{x}) u, && x \in \Omega \\
u &= 0, && x \in \partial\Omega.
\end{aligned}
$$

with

$$
\rho(\boldsymbol{x}) = \begin{cases} \rho_1 & \text{for } x \notin S \\ \rho_2 & \text{for } x \in S. \end{cases}
$$

- Solve the optimization

$$
\min_S \lambda_1 \text{ or } \lambda_2 \text{ subject to } \|S\| = K.
$$



Application: Structural Vibration Control

- Level set formulation by Osher and Santosa:
  - Finite difference approximations for Laplacian
  - Sparse eigenvalue solver for solutions λ_i, u_i
  - Calculate descent direction $\delta\phi = -v(\boldsymbol{x})|\nabla\phi|$ with $v(\boldsymbol{x})$ from shape sensitivity analysis
  - Find a Lagrange multiplier that enforces the area constraint using Newton's method
  - Represent interface implicitly, propagate using level set method