# Topology Optimizations with applications in Microfluidics

## a comparison of level set methods

Andy R Terrel[1] Kevin Long[2]

[1]Department of Computer Science
University of Chicago

[2]Computational Sciences and Mathematics Deptartment
Sandia Livermore National Lab

September 18, 2006

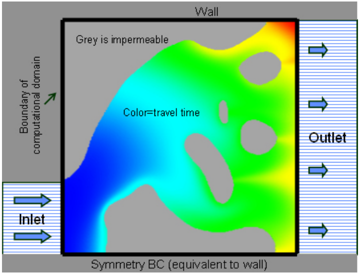
# Outline

# The Microfluidics Device Model.

- Flow is electrokinetically driven
  - Charge confined to a thin Debye layer near walls
  - High viscosity $\rightarrow$ velocity proportional to electric field $\rightarrow$ potential flow
  - Travel time to a point computable from advection equation
- Important constraints are imposed by acid-etch manufacturing process
  - Channel depth can vary stepwise only
  - Features have minimum radius of curvature equal to channel depth
  - Curvature of bottom near walls is important when width is comparable to depth

# Optimizing the Microfluidics Device.

- To optimize flow, change channel geometry
  - Varying channel walls has little effect by itself
  - Options: islands or depth
  - Islands require *topology optimization*

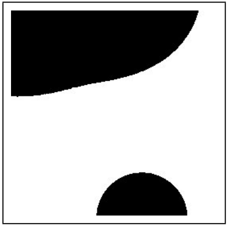



- For narrow channels: use shape of etch mask as design variable
  - Design automatically satisfies manufacturability constraint
  - Requires simulation of etching - more expensive than electrokinetic flow simulation!
  - Includes depth variation near walls

# Goal of Topology Optimization

Given some domain, $\Omega$, we wish to partition it into two subsets:

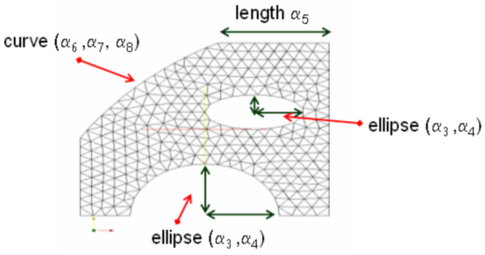
- The interior, $\mathcal{I}$, which will be the domain of some PDE,
- The exterior, $\mathcal{E}$, everything else.



# Using Parametric Curves

One idea is to use a parametric curve. But ...

- Small feasible region
- Mesh dependent
  - Causes mesh deformation
  - Topology changes require remeshing

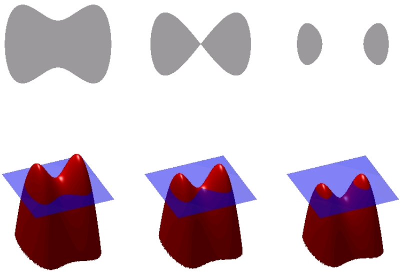


# Using a Level Set Method

Let the interface between $\mathcal{E}$ and $\mathcal{I}$ be defined by the zero contour of a level set function $\phi$.

$$\phi : \mathbf{x} \in \Omega \rightarrow \mathbb{R}$$

A level set function defines a geometric shape one dimension higher than $\Omega$



# Advantages of Level Set Method





By using a Level Set Method we gain several advantages.

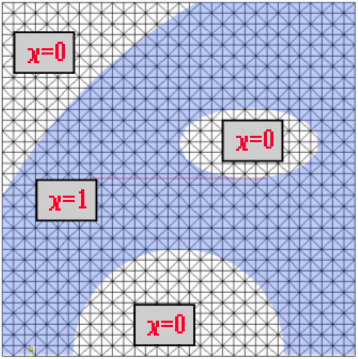
- Only need to manipulate $\phi$ not the mesh.
- Dimensionality of $\phi$ allows *topology* changes effortlessly.
- Much richer design space.

# Defining $\mathcal{E}$ and $\mathcal{I}$.



Now we can define $\mathcal{E}$ and $\mathcal{I}$ by a $\chi$ distribution:

$$\chi(\phi) = \begin{cases} 0 & \text{if} \phi < 0 \\ 1 & \text{otherwise} \end{cases}$$



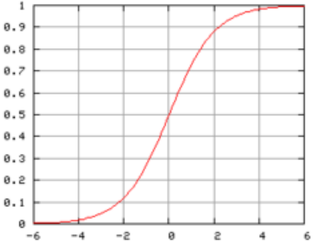

But this makes a large boolean optimization problem, which is very hard so we relax $\chi$ to the reals by the use of a sigmoid function.

# The Sigmoid Function.

We use the sigmoid function: $\sigma : \mathbb{R} \to \mathbb{R}|[0,1]$

$$\sigma(\phi) = \frac{1}{2}(1 + \tanh\left(\frac{\phi}{\Delta}\right))$$

where $\Delta$ is a given parameter.



# Ill posed.

The problem as it stands is ill posed.

- We only use the zero contour of $\phi$, but there are infinitely many choices.
- Without a smoothness condition, $\phi$ could be very perverse.

# Regularization.





To guide the optimization algorithm to a “nice” answer, we use a Tikhonov and total variation diminishing regularization terms.

- Tikhonov

$$\frac{\alpha_1}{2}\int_\Omega |\nabla\phi|^2 d\Omega$$

- TVD

$$\alpha_2\int_\Omega (\nabla\sigma^2)^{\frac{1}{2}}$$

# The interface between $\mathcal{E}$ and $\mathcal{I}$.

To promote boolean shapes, we require the transition of $\sigma$ from 0 to 1 to be small.

This transition is controlled by:

- The parameter, $\Delta$, something we control at runtime.
- The slope of $\phi$, a design variable.

By controlling the slope of $\phi$, we are able to pick the length of this transition.

# Slope Penalty Method.





Proposed by A. Cunha.

- Impose the approximate constraint that $|\nabla\phi| \approx 1$
- Don't use strict constraint because $\nabla\phi$ is also controlled by Tikhonov regularization
- Use penalties to determine the dominant feature in the objective function.

# Slope Barrier Method.

The slope penalty method defined the slope of $\phi$ over the entire domain.

A more desirable solution would be to only control the slope around the transition region of $\sigma$. So instead we use the inequality constraint:

$$(\left(\frac{\phi}{\Delta}\right)^2 + (\nabla\phi)^2) \geq 1 - \epsilon$$

This will allow the slope of $\phi$ vary freely outside our transition region.

# Quality Measure of the Shape





In our application we used a PDE to determine the quality of the shape

For testing purposes we are only going to look at matching a specified target shape, $\mathcal{E}^*$, and thus we will use a Heaviside Distance function to compare our shape with the target shape, that is:

$$d(\mathcal{E}, \mathcal{E}^*) = \frac{1}{2} \int_\Omega (\sigma - \sigma^*)^2 d\Omega$$

# Formal Definition of the Problem.





minimize the function $\mathcal{F}$ where:

$$\begin{aligned} \mathcal{F} &= \frac{1}{2}\int_\Omega (\sigma-\sigma^*)^2 d\Omega + \frac{\alpha_1}{2}\int_\Omega |\nabla\phi|^2 d\Omega + \\ &+ \alpha_2 \int_\Omega (\nabla\sigma^2)^{\frac{1}{2}} d\Omega + \\ &+ \frac{\beta}{4}\int_\Omega (1.0-\nabla\phi^2)^2 d\Omega \end{aligned}$$

augmented to the inequality constraint:

$$(\nabla\phi^2 + \left(\frac{\phi}{\Delta}\right)^2) \geq (1-\epsilon)$$

# Optimization Strategy.

For opimization we require a local an global strategy.

- Local Optimization
  - Use an adaptive limited-memory BFGS algorithm[1]
- Global Optimization
  - Use a Tunneling Method
    - Pick a new random direction to find a lower point.
    - Use method to catch perturbations at low spacial frequency.

$$\phi \leftarrow \phi + \sum_{m=-M,n=-N}^{M,N} A_{m,n} e^{-i\left(\frac{\pi m}{L}x + \frac{\pi n}{L}y\right)}.$$

  - For better results use simulated annealing to allow for a few uphill steps.

[1] Byrd and Boggs, publication in process

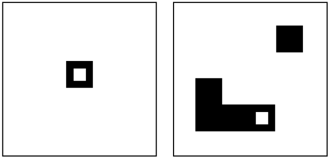
# Interpreting Results.

Reminder of what we what good results are:

- Match shape closely. (Heaviside distance small)
- Results should be boolean.
- Easy objective function to minimize.
  - Low number of iterations to reduce objective function.
  - Invariant to initial guess
  - Low number of artificial minimizers.

# General Boolean Results

Figure: The target images.



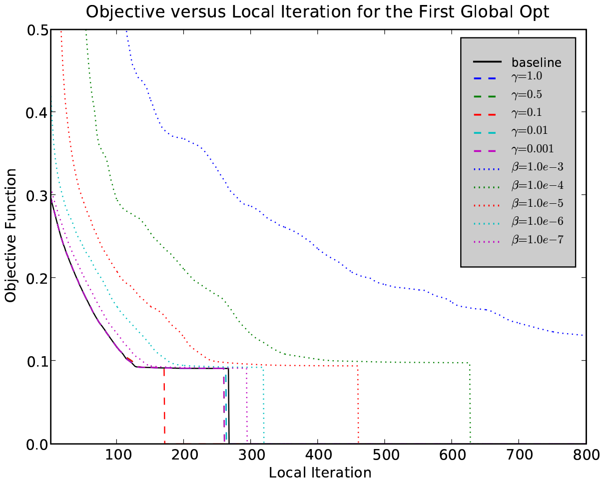
(a) Simple boolean image (b) Complex boolean image

- Tikhonov regularization and Slope barrier did well all around but slope barrier did beter in iteration count.
- Slope penalty method correctly reproduced the images but had a less boolean shape and had a larger number of local minimizers.

# Iteration Count.

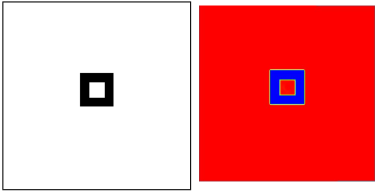


Figure: Sample convergence speed comparison



# Target Shape and Tikhonov Regularization.

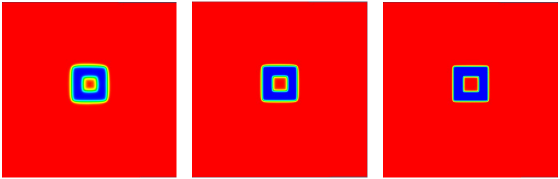
Figure: Sample Target and baseline configuration



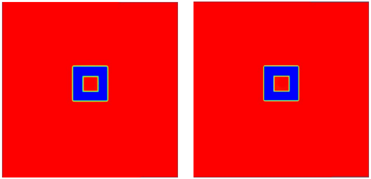
(a) target (b) baseline

# Slope Penalty Method.

Figure: Sample slope penalty configurations



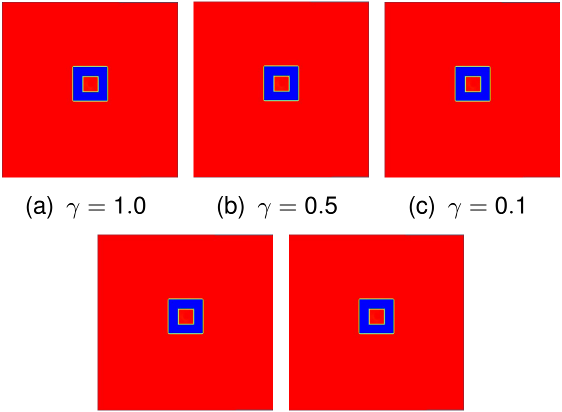
(a) $\beta = 1.0e-3$ (b) $\beta = 1.0e-4$ (c) $\beta = 1.0e-5$



(d) $\beta = 1.0e-6$ (e) $\beta = 1.0e-7$

# Slope Barrier Method.

Figure: Sample slope barrier configurations



(a) $\gamma = 1.0$ (b) $\gamma = 0.5$ (c) $\gamma = 0.1$ (d) $\gamma = 0.01$ (e) $\gamma = 0.001$

# Using Non-Boolean Images.

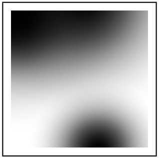
When using non-boolean targets to find a boolean approximation several features changed:

- The Slope Penalty method
  - matched the shape quickly
  - fewer minimizers
  - resulting image – non-boolean
- The Slope Barrier method
  - large Heaviside distance
  - lots of minimizers
  - resulting image – boolean

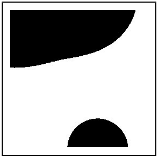
# Shape to Match.

For illustration we use the non-boolean solution to one of the microflow devices.
The next few slides will show resulting images from these tests.

Figure: Smooth target and boolean solution



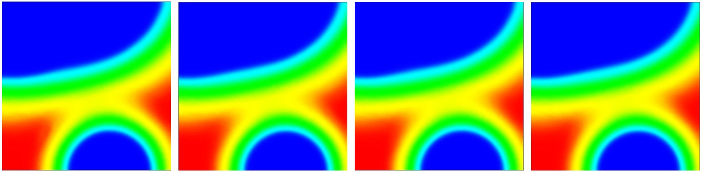
(a) target



(b) solution

# Large Slope Penalty.

Figure: Larger slope penalty ($\beta = 1.0e - 4$) Final Global iteration



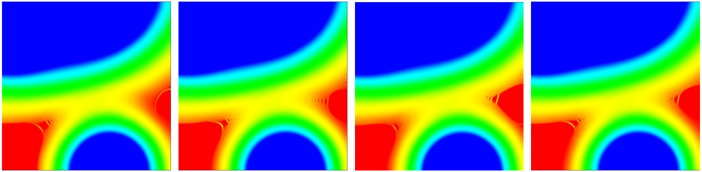
(a) 4 circles (b) 9 circles (c) 16 circles (d) 25 circles

# Small Slope penalty.

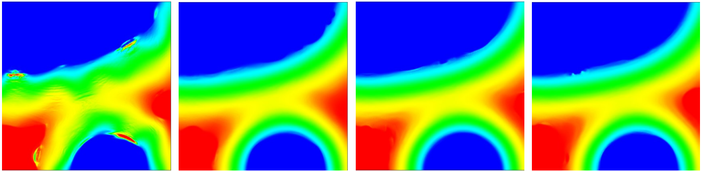




Figure: Smaller slope penalty $\beta = 1.0e - 6$, First and Final Global iteration



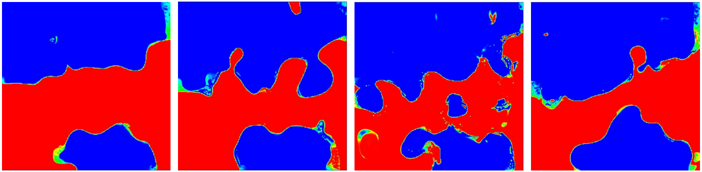
(a) 4 circles, first (b) 9 circles, first (c) 16 circles, first (d) 25 circles, first



(e) 4 circles, final (f) 9 circles, final (g) 16 circles, final (h) 25 circles, final

# Slope Barrier Method.

Figure: Slope Barrier $\gamma = 0.001$, Final Global iteration



(a) 4 circles (b) 9 circles (c) 16 circles (d) 25 circles

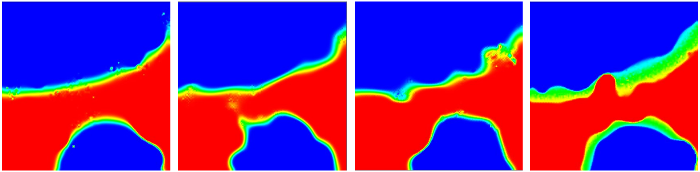
# Compromising Boolean Image for a Better Shape

- In this scenario, neither method has produce satisfactory results.
- There seems to be a tradeoff between the boolean-ness and shape.

To help alleviate this problem, we try a large $\Delta$ and reduce it after a number of iterations. This should have no affect on the Slope Penalty Method but gives better results for the Slope Barrier Method.

# Using a Variable Δ.

Figure: Slope Barrier Reduction Method ($\gamma = 0.001$, reduces = 20, opts = 5, reduction = 0.9)



(a) 4 circles (b) 9 circles (c) 16 circles (d) 25 circles

# Summary

- Controlling the slope of a Level Set Function provides a natural way to embed a topology optimization in a PDE constrained problem.
- For boolean targets, the methods are closely comparable in effectiveness.
- Non-boolean targets provide challenges that depend highly on the necessary conditions. In our application the boolean condition was as important as the shape and thus the Slope Barrier wins.

- Outlook
  - Test out different slope controlling mechanisms for the level set function.
  - Work out some solid theory behind these types of methods.

# Questions

Any Questions?

aterrel@uchicago.edu

Acknowledgements:

- Kevin Long - original simulations and frameworks.
- Paul Boggs - development of opimization schemes we used.
- Sandia National Labs - funding and work environment.

# The Sundance Simulation Environment





- High-level, symbolic components simplify simulator development
  - Write an entire multiphysics simulator in a few dozen lines (Python or C++)
- Symbolic representation allowing efficient derivative evaluation
- Performance is superb
  - Abstract representation allows automated performance optimizations
  - User interface dedicated to human readibility, computational core dedicated to performance
- Fully parallel
  - Uses Trilinos parallel solver components