

A multilevel-level-set method for optimizing eigenvalues in shape design problems

Eldad Haber

Department of Mathematics and Computer Science,
Emory University

Outline

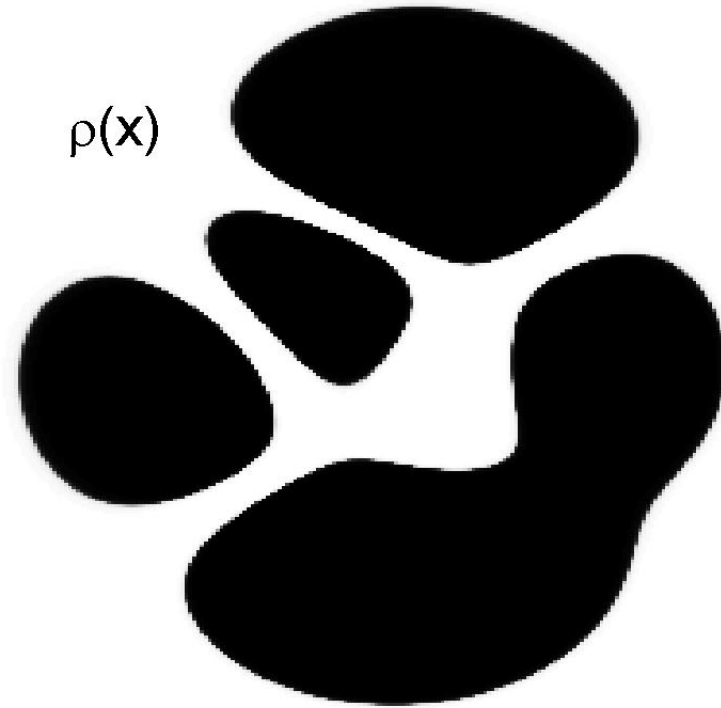
- Introduction and motivation
- Computing eigenvalues in practice - the inverse iteration
- Optimization and the inverse iteration
- Optimization and level-set methods
- Grid continuation techniques
- Examples

Introduction

Consider the following model design problem:

Find the optimal density distribution $\rho(x)$ such that the membrane has the minimal (maximal) frequency response

Similar problems appear in - structural design, photonic bandgap, antenna design ...



Mathematical formulation

min or max λ

s.t λ is the minimal eigenvalue of $\mathcal{L}u = \lambda\rho(x)u$

$$\rho \in \mathcal{S}$$

$$\int_{\Omega} \rho dV = M$$

\mathcal{L} is a self adjoint differential elliptic operator

\mathcal{S} is a space such that ρ can have values of ρ_1 or ρ_2 only.

Historical notes

- First formulated by Lagrange (max problem)
- Second formulation by Rayleigh (min problem)
- Analysis by Keller (60's)
- First numerical methods in the 90's (Overton)
- For large scale problems - still an open problem

Other notes

- The eigenvalues are not continuously differentiable with respect to ρ
- Solving an eigenvalue problem for a large scale application is computationally intensive
- For derivative information - eigenvectors are needed
- How to express the constraint $\rho \in \mathcal{S}$?

Constraint elimination - level set methods

The constraint: $\rho(x) \in \mathcal{S}$.

That is $\rho(x)$ can have only two values ρ_1 and ρ_2

Possible parameterizations

Track the interface between ρ_1 and ρ_2

Integer programming

Level set - the interface is the level-set of a smooth function

Constraint elimination - level set methods

Level set

$$\rho(x) = \begin{cases} 2 & \text{if } m(x) \geq 0 \\ 1 & \text{if } m(x) < 0 \end{cases}$$

But this function is not differentiable.

Define

$$\rho_h(m) = \frac{1}{2} \tanh(\alpha_h m) + 1.5$$

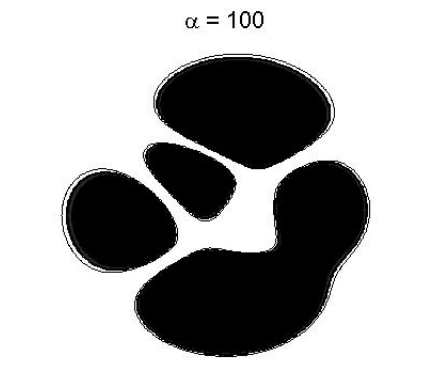
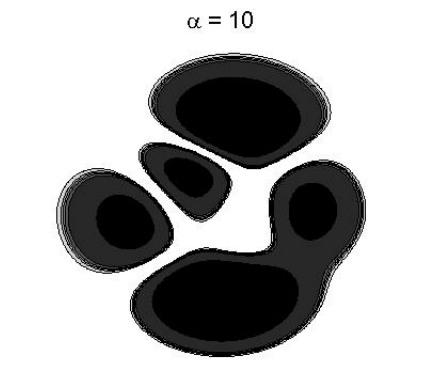
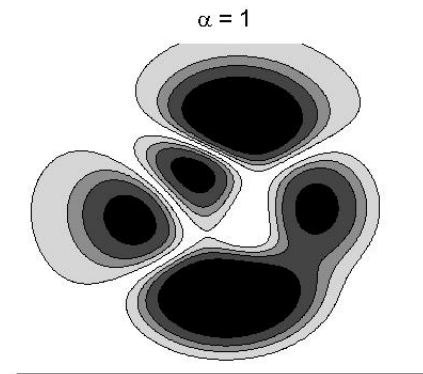
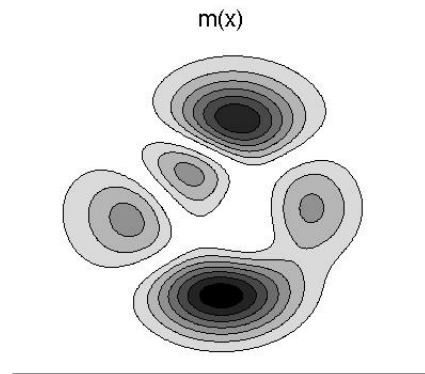
ρ is the (smoothed) level-set of m

Constraint elimination - level set methods

As $\alpha_h \rightarrow \infty$ the solution of the continuous problem converge to the solution of the discontinuous problem

[Scherzer 02]

The function m is any smooth function



Discretization and reformulation

Use a regular grid for discretization and obtain

$$\begin{aligned} \text{min or max} \quad & \lambda \\ \text{s.t} \quad & \lambda \text{ is the minimal eigenvalue of } Lu = \lambda D(\rho(m))u \\ & h^d e^T \rho = M \end{aligned}$$

But there may be infinite vectors m that solve this problem.

Need regularization ...

Reformulation by penalty

$$\begin{aligned} \min \quad & \pm\lambda + \beta \left(\frac{1}{2}\gamma \|\nabla_h m\|^2 \pm h^d e^T \rho(m) \right) \\ \text{s.t} \quad & \lambda \text{ is the minimal eigenvalue of } Lu = \lambda D(m)u \end{aligned}$$

β - Lagrange multiplier

γ - regularization parameter

The traditional approach

Take the steepest decent approach [Santosa & Osher 01]

Calculate Frech'et derivatives by perturbation

$$\lambda_{\rho}^T \delta \rho = -\lambda \frac{u^T D(\delta \rho) u}{u^T D(\rho) u}$$

by computing the eigenpair (u, λ)

Problems with the traditional approach

- Need to obtain eigen-pairs for each iteration.
- Steepest descent is slowly converging
- Example in 2D - 400 - 4000 iterations are needed

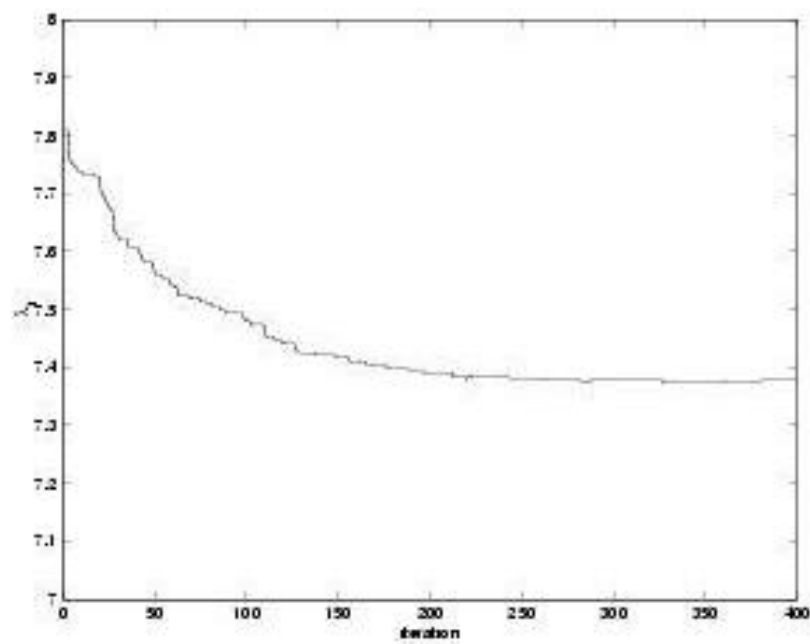


Figure 5: Minimization of λ_1 ; see Figure 6 for corresponding densities. 16

Question - can we do better?

Computing eigenvalues in practice the inverse iteration

Goal - avoid an exact computation of eigenpair during the optimization process

Combine the way we compute eigenvalue with the optimization algorithm.

Computing eigenvalues in practice the inverse iteration

Use the Inverse Iteration.

Converge linearly but involves with “easy to solve” subproblems.

Convergence is grid independent

The Inverse Iteration

- Choose a vector u_0 and an integer k s.t $\|u_0\| = 1$
- For $j = 1 \dots k$
Solve $Lu_j = \frac{1}{\sqrt{u_{j-1}^T u_{j-1}}} Du_{j-1}$.
- Set $\lambda \approx \frac{1}{\sqrt{u_k^T u_k}}$

Reformulating the inverse iteration

Reformulate the inverse iteration as a nonlinear system of equations

$$C(m, u) = I(u)Au - B(m)u + b(m) = 0$$

where

$$A = \text{diag}(L, L, \dots, L);$$

$$B(m) = \begin{pmatrix} 0 & & & \\ D(m) & 0 & & \\ & \cdot & & \\ & & D(m) & 0 \end{pmatrix}$$

$$I(u) = \begin{pmatrix} I & & & \\ & \sqrt{u_1^T u_1} & & \\ & & & \\ & & & \sqrt{u_k^T u_k} \end{pmatrix}$$

$$u = [u_1^T, \dots, u_k^T]^T; \quad b(m) = [(D(m)u_0)^T, 0, \dots, 0]^T$$

Define the matrix Q such that $Qu = u_k$.

The smallest eigenvalue is

$$\lambda^{-2} = u^T Q^T Q u$$

The matrices $A, B, I(u), Q$ never generated in practice and only matrix vector products are calculated

The discrete optimization problem

We can now rewrite the optimization problem as

$$\begin{aligned} \min \quad & \frac{1}{2}u^T Q^T Q u + \beta \left(\frac{1}{2}\gamma \|\nabla_h m\|^2 - h^d e^T \rho(m) \right) \\ \text{s.t} \quad & C(m, u) = I(u)Au - B(m)u - b(m) = 0 \end{aligned}$$

This is a class of PDE constraint optimization problem

[Heinkenslush, Ghattas, Burger, Ascher & Haber]

Use Inexact Sequential-Quadratic-Programming (SQP) methods for the solution of the problem

SQP framework

The Lagrangian

$$\mathcal{J}(u, m, \mu) = \frac{1}{2}u^T Q^T Q u + \beta \left(\frac{1}{2}\gamma \|\nabla_h m\|^2 - h^d e^T \rho(m) \right) + \mu^T (I(u) A u - B(m) u - b(m))$$

μ - a vector of Lagrange multipliers.

Differentiating we obtain a large nonlinear system of equations for m, u, μ

$$\mathcal{J}_u = Q^T Q u + C_u^T \mu = 0$$

$$\mathcal{J}_m = \beta \left(\gamma \nabla_h^T \nabla_h m + h^d \rho'(m) \right) + C_m^T \mu = 0$$

$$\mathcal{J}_\mu = C(m, u) = 0$$

$$C(m, u) = I(u) A u - B(m) u - b(m)$$

Properties of the matrices

C_u - a discretization of a elliptic differential operator.
Lower triangular - solve using multigrid methods

C_m - a diagonal positive matrix (mass)

The linear subproblem

Approximating the Hessian by a Gauss-Newton approximation

$$\begin{pmatrix} C_u & 0 & C_m \\ Q^T Q & C_u^T & 0 \\ 0 & C_m^T & \beta R \end{pmatrix} \begin{pmatrix} s_u \\ s_\mu \\ s_m \end{pmatrix} = - \begin{pmatrix} \mathcal{J}_\mu \\ \mathcal{J}_m \\ \mathcal{J}_u \end{pmatrix}$$

Note - its an non-symmetric form of a KKT system, putting the "meat" on the diagonal

The linear subproblem

The KKT system

$$\begin{pmatrix} C_u & 0 & C_m \\ Q^T Q & C_u^T & 0 \\ 0 & C_m^T & \beta R \end{pmatrix} \begin{pmatrix} s_u \\ s_\mu \\ s_m \end{pmatrix} = - \begin{pmatrix} \mathcal{J}_\mu \\ \mathcal{J}_m \\ \mathcal{J}_u \end{pmatrix}$$

For large β 's - equivalent to a diagonally dominant system of PDE's

But β is small - Highly coupled system of PDE's

[Haber & Ascher 01]

Solving the linear subproblem - the reduced Hessian method

$$\begin{pmatrix} C_u & 0 & C_m \\ Q^T Q & C_u^T & 0 \\ 0 & C_m^T & \beta R \end{pmatrix} \begin{pmatrix} s_u \\ s_\mu \\ s_m \end{pmatrix} = - \begin{pmatrix} \mathcal{J}_\mu \\ \mathcal{J}_m \\ \mathcal{J}_u \end{pmatrix}$$

Eliminate

$$\begin{aligned} s_u &= -C_u^{-1}(\mathcal{J}_\mu + C_m s_m) \\ s_\mu &= -C_u^{-T}(\mathcal{J}_m + Q^T Q s_u) \end{aligned}$$

The reduced Hessian method

Obtain an equation for s_m

$$(J^T J + \beta R'')s_m = C_m^T C_u^{-T} (\mathcal{L}_m + Q^T Q C_u^{-1} \mathcal{L}_\mu) \equiv g_r$$

where

$$J = -Q C_u^{-1} C_m$$

Solving the The reduced Hessian system

method I

Use Conjugate Gradient

At each CG iteration solve $C_u v = w$ and $C_u^T v = w$

method II

Approximate the reduced Hessian using L-BFGS method.

No matrix inversion needed

Example I - min λ

Problem in 2D

$$\mathcal{L} = -\Delta$$

Discretize on a 65^2

Number of iterations - 44

Cost per iteration - 14 solves of the Poisson equation

Continuation

Fast convergence - start at a “near-by” point.

How do we get close, cheap?

Continuation techniques

Natural continuation - grid, iterations in the power method

Grid Continuation - Multilevel methods

Effective because

- The smallest eigenvalue of an elliptic differential operator corresponds to a smooth eigenvector.
- The level-set function m is smooth.

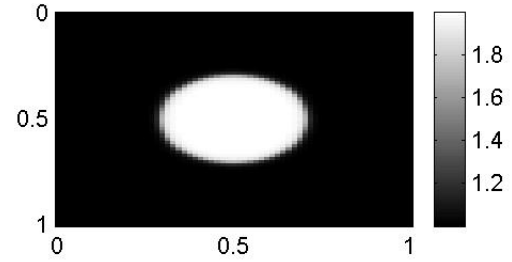
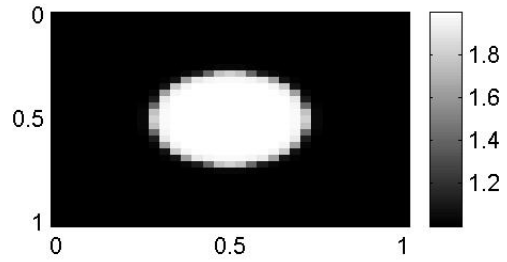
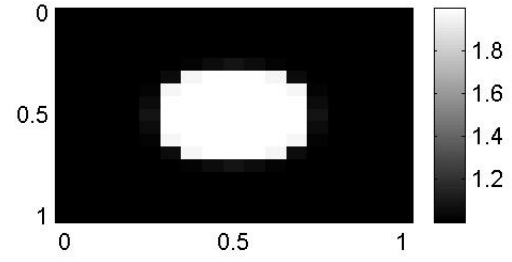
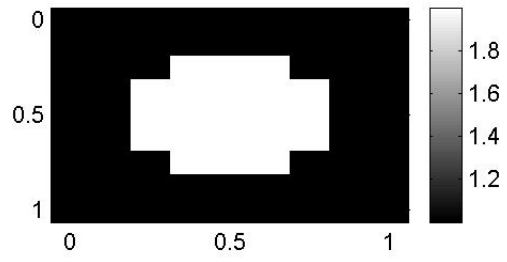
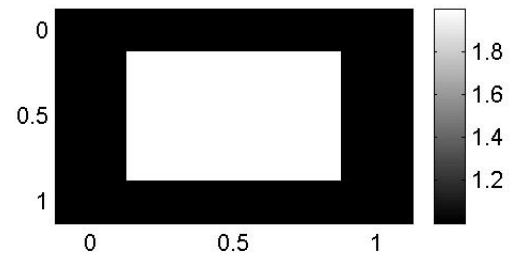
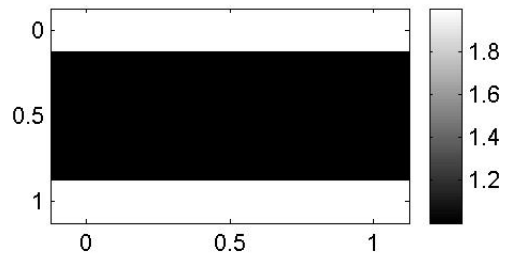
Grid continuation

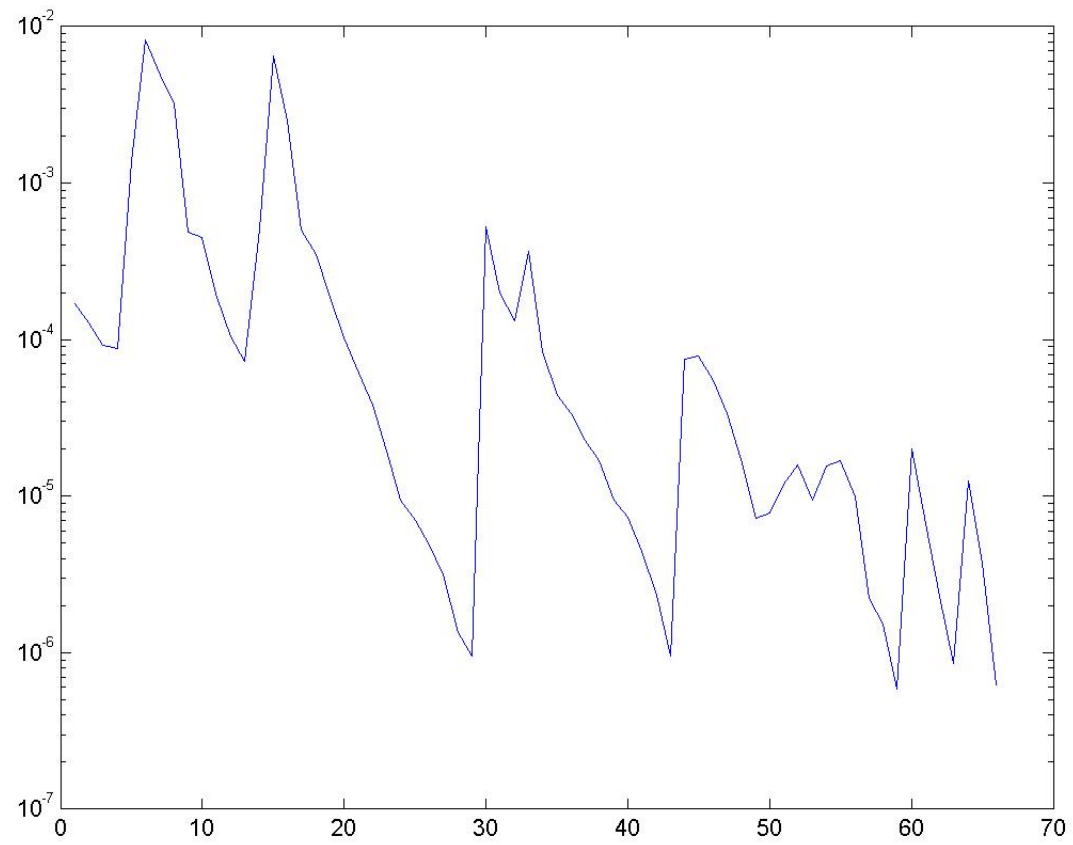
- Initialize m_H and choose a vector q_H as initial guess to the first eigenvector.
- while not converge
 1. Solve the optimization problem on the current grid.
 2. Output: m_H, λ_H, β, k and q_H
 q_H - the approximation to the first eigenvector.
 3. Refine the m_H and q_H grid to h using bilinear interpolation.
 $m_h = I_H^h m_H; \quad q_h = I_H^h q_H$
 4. Set the initial guess for the eigenvector q_h .
 5. Set $H \leftarrow h$.

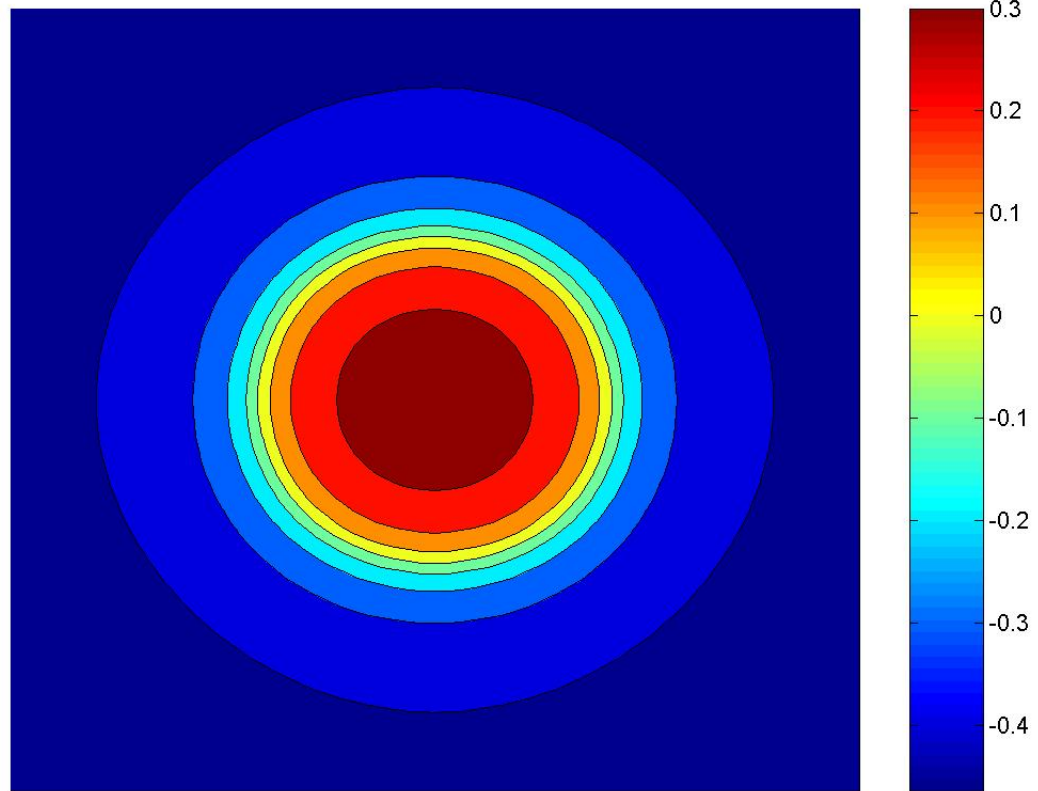
Example I - revisited

Grid	Iterations	Final mass
5^2	29	1.3601
9^2	14	1.2592
17^2	16	1.1584
33^2	4	1.1538
65^2	3	1.1539

Table 1: Iterations per grid for maximizing λ







Example II - max λ

Grid	Iterations	Final mass
5^2	38	1.603
9^2	19	1.588
17^2	12	1.567
33^2	6	1.562
65^2	4	1.562

Table 2: Iterations per grid for minimizing λ

Example III - max λ - complicated domain

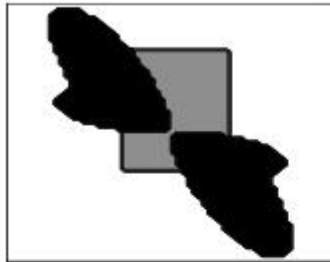
Use $\mathcal{L} = -\Delta$

Domain has "holes"

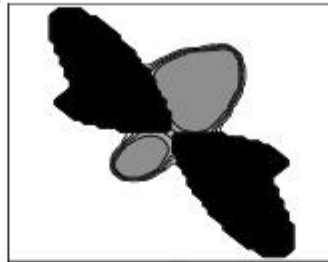
Nuemann BC inside the holes, Dirichlet outside

Test for changing topology

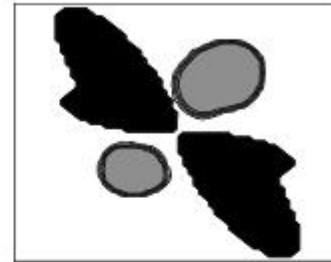
Initial guess



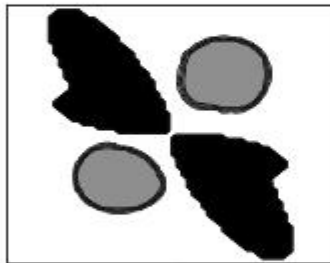
5 lter



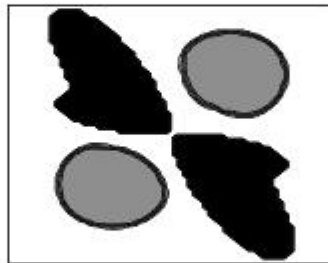
10 lter



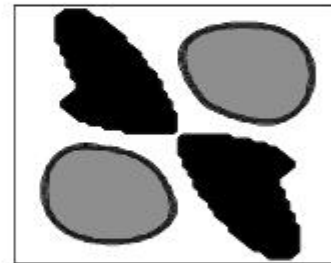
15 lter



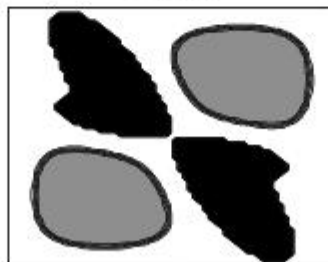
20 lter



25 lter



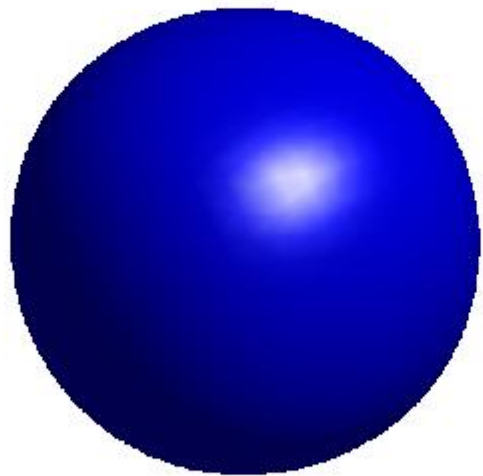
34 lter



Example IV - max λ in 3D

Grid	Iterations	Final mass
5^2	52	1.261
9^2	13	1.132
17^2	9	1.116
33^2	5	1.110
65^2	2	1.112

Table 3: Iterations per grid for maximizing λ in 3D



Summary Conclusions

- Computing eigenvalues in practice - the inverse iteration
- Optimization and the inverse iteration
- Optimization and level-set methods
- Grid continuation techniques

Future work

- How should we deal with problem of minimizing other eigenvalues or the gap in eigenvalues?
- What other constraints we need to add to make the solution useful for different applications?