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

E. Haber*

July 22, 2003

Abstract

In this paper we consider optimal design problems that involve shape optimization. The goal is to determine the shape of a certain structure such that it is either as rigid or as soft as possible.

To achieve this goal we combine two new ideas for an efficient solution of the problem. First, we replace the eigenvalue problem with an approximation by using the inverse iteration.

Second, we use a level set method but rather than propagating the front we use constrained optimization methods combined with multilevel continuation techniques.

Combining these two ideas together we obtain a robust and rapid method for the solution of the problem.

1 Introduction and problem setup

In this paper we consider optimal design problems that involve shape optimization. The goal is to determine the shape of a certain structure such that it is either as rigid or as soft as possible. Although similar problem was originally posed by Lagrange and later by Rayleigh, only recent numerical treatment has been given to it [18, 16, 12, 11, 17]. In general, the mathematical problem can be represented as finding a distributed parameter $\rho(x)$

*Dept of Mathematics and Computer Science Emory University, Atlanta, GA

such that it solves the following constraint optimization problem

$$\begin{aligned} \min \text{ or } \max \quad & \lambda && \text{(1a)} \\ \text{s.t} \quad & \lambda \text{ is the minimal eiganvalue of } \ \mathcal{L}u = \lambda\rho(x)u && \text{(1b)} \\ & \rho \in \mathcal{S} && \text{(1c)} \\ & \int_\Omega \rho \, dV = M && \text{(1d)} \end{aligned}$$

where $\mathcal{L}$ is a self adjoint differential elliptic operator and $\mathcal{S}$ is a space such that ρ can have values of ρ_1 or ρ_2 only. If $\mathcal{L}$ is the minus Laplacian, then the problem is a model problem for many engineering design problems [3, 4].

In general, the eigenvalues are not continuously differentiable function with respect to the parameters we seek and this generates a considerable complication that had lead to semi-definite programming techniques [12]. In this paper we avoid this complication by assuming that the first eigenvalue is simple, that is, it does not have a multiplicity of more than one. This is theoretically justified for the model problem we solve here.

All the methods known to us for eigenvalue optimization use the eigenvalue equations themselves to define gradients of the eigenvalue with respect to the parameter (for a survey see [12, 17]). For large scale problems such approach is computationally expensive as it requires accurate evaluation of eigenvectors. Indeed, if we consider 3D structures then the discretized constraint optimization problem (1) can be very large as both ρ and u can have dimensionality of millions. We therefore take a very different approach. First, we approximate the (smallest) eigenvalue as a finite fixed point process and replace the eigenvalue equation with the inverse iteration. We then compute the gradients of this approximation (rather then of the true eigenvalue). Our approach tightly couples the way we compute the eigenvalue to the way we calculate its derivative and it is unlike the common approach where the process of obtaining the eigenvalue is divorced from the process of calculating its derivative. The advantages of our approach is that we are able to easily transform the problem to a constrained optimization problem and use standard techniques for its solution. Even more important, we are able to avoid a very exact eigenvalue computation early in our iteration and we can generate a natural continuation process to be used for highly nonlinear problems.

In order to deal with the constraint that $\rho \in \mathcal{S}$ we use a level set method. In a recent paper Osher and Santosa [15] have used a level-set method to solve this problem. Their approach requires the computation of the gener-

alized eigenvalues problem at each iteration and uses level set technology to track the evolving interface by solving the Hamilton Jacobi equations which describe the surface evolution. From an optimization stand point, their method is equivalent to a steepest descent method and therefore, converges slowly. We like the general idea of using level-set to represent the shape but our numerical treatment of the equations is very different. We avoid the Hamilton Jacobi equations all together and treat the problem as a numerical optimization problem. We solve the optimization problem using a reduced Hessian Sequential Quadratic Programming (SQP) method combined with multilevel continuation techniques. This allows us to quickly solve the optimization problem in a very few iterations at a fraction of the cost used when the problem is solved by traditional level set methods.

We choose to discretize the problem first and only then to optimize so we are able to use the overall framework and tools of numerical optimization techniques [14]. Care must be taken that at the end of the process the solution to the discrete problem is actually (an approximation to) the solution of the continuous problem. To use level set ideas on a grid we set $\rho = \chi_h(m)$ where χ is the usual characteristic function. To make this function differentiable we use similar ideas as in [10] and set

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

where α_h depends on the grid size h. The derivative of this function converges to a (scaled) delta function and approximates the interface. It has been proved in [10] that at the limit (as $\alpha \to \infty$) the solution m converges to the continuous solution. We choose α_h such that the width of the approximation to the delta function is roughly three pixels[1]. Discretizing the problem we define the following optimization problem

$$\begin{aligned}
\min \quad & \pm\lambda \; + \; \frac{1}{2}\hat{\gamma}\|\nabla_h m\|^2 && \text{(3a)}\\
\text{s.t} \quad & \lambda \text{ is the minimal eiganvalue of } \; Lu = \lambda D(m)u && \text{(3b)}\\
& h^d e^T \rho(m) = M && \text{(3c)}
\end{aligned}$$

where L is a finite volume or a finite element discretization of $\mathcal{L}$ and $D(m)$ is a diagonal mass matrix of the values of $\rho(m)$ on the grid, ∇_h is the discrete gradient and $\hat{\gamma}$ is a small regularization parameter. This regularization

[1]We define the width of the function w is an interval where $\int_{-w/2}^{w/2} \chi_h'(x)\,dx = 0.66$

term is added in order to obtain a smooth level-set function m; it does not intend to change the level-set of m by much but to rather generate a unique smooth function m which we are able to easily interpolate on different grids. Finally, the single integral equality constraint is discretized using the mid-point method, where h is the grid size and d is the dimension of the problem. In order to avoid the equality constraint (3c) directly we use regularization or penalty. We replace the constrained problem (3) with the unconstrained optimization problem

$$\begin{aligned} \min \quad & \pm\lambda \ + \beta \left(\frac{1}{2}\gamma \|\nabla_h m\|^2 \pm h^d e^T \rho(m) \right) && \text{(4a)} \\ \text{s.t} \quad & \lambda \text{ is the minimal eiganvalue of } \ Lu = \lambda D(m) u && \text{(4b)} \end{aligned}$$

where β is a fixed regularization or a penalty parameter and $\beta\gamma = \hat{\gamma}$. It is easy to show that β correspond to the Lagrange multiplier of the constrained problem which is positive for the minimization problem (the plus λ) and negative for the maximization problem. For the minimization problem, if β is very large, we require to minimize the total mass and therefore we obtain a very small mass, thus ρ is mainly made of material ρ_1 which is lighter. If, on the other hand, β is very small, the penalty on the total mass is small and this leads to a large mass thus ρ is mainly material ρ_2 which is heavier. The total mass in this formulation can be thought of as a map

$$M = M(\beta) = h^d e^T \rho(m(\beta)) \tag{5}$$

where $m(\beta)$ is obtained by solving to the optimization problem (4) for a specific β. The parameter β has to be tuned such that the constraint is satisfied. From a theoretical point of view, there is no guaranty that the map $M(\beta)$ is smooth. In some well known cases such a map can have jumps [7] and this implies that there is no minimizer subject to the constraint. This formulation is very similar to regularization of inverse problems [6] and to trust region methods [5] where an integral constraint is replaced with a penalty parameter. In our previous work [1] we have used multilevel continuation techniques to evaluate this parameter. In this paper we use a similar strategy with a few minor modifications.

The rest of this paper is divided as follows. In Section 2 we review the inverse iteration method to compute the smallest eigenvalue of $D^{-1}A$ and motivate its use. In Section 3 we formulate the discrete optimization problem and derive the Euler-Lagrange equations to be solved. In Section 4 we discuss

the solution of these equations utilizing constrained optimization techniques and the reduced Hessian SQP approach. In Section 5 we show how to use a multilevel continuation method to quickly solve the problem. In Section 6 we use yet another continuation process in the inverse iteration in order to obtain a stable algorithm. Combining the two continuation processes allow us to quickly converge even for highly nonlinear problems. Finally in Section 7 we carry out numerical experiments that demonstrate the advantages of our technique, we summarize the paper in Section 8.

2 The inverse iteration

As explained in the introduction, we would like to avoid the need to accurately compute the eigenvalue problem $Lu = \lambda D(m)u$ at each iteration, especially if we consider problems in 3D. We therefore reformulate the problem such that only the smallest eigenvalue is obtained in an increasing order of accuracy.

There are a few options for such a process. Maybe the fastest converging one is the Rayleigh Quotient iteration which converges in a cubic rate. The problem with this process is that it involves with solving linear systems of the form $(L - \lambda_k D)v = Du$ which can become very ill-conditioned. We therefore use the inverse iteration technique. This method converges only linearly but it has the three main advantages. First, only systems of the form $Lv = Du$ need to be solved. Using the fact that L is a discretization of an elliptic operator we use multigrid methods [19] to quickly calculate an approximate solution to these systems with the desired accuracy. Second, it can be shown that the convergence, although linear, is grid independent [13]. Finally, the approach generates simple expressions which are easily differentiated. This is an important advantage when we consider a constrained optimization process where derivatives have to be taken.

The inverse iteration process for the computation of the smallest eigenvalue of a positive definite system can be summarized as follows

The inverse iteration

- Choose a vector u_0 and an integer n such that $\|u_0\| = 1$

- For $j = 1...k$

$$\text{Solve} \quad Lu_j = \frac{1}{\sqrt{u_{j-1}^T u_{j-1}}} Du_{j-1}.$$

- Set $\lambda \approx \frac{1}{\sqrt{u_k^T u_k}}$

An important issue is the selection of u_0, the initial vector. If the vector contains mainly the eigenvector which corresponds to the smallest eigenvalue of $D^{-1}L$ then we may require a small k to obtain good results. We return to this point when we discuss the multilevel continuation process.

We reformulate the inverse iteration process as a nonlinear system of equations

$$C(m, u) = I(u)Au - B(m)u + b(m) = 0 \tag{6}$$

where

$$\begin{aligned}
&A = \text{diag}(L, L, ..., L); \\
&B(m) = \begin{pmatrix} 0 & & & \\ D(m) & 0 & & \\ & \cdot & & \\ & & D(m) & 0 \end{pmatrix} \\
&I(u) = \begin{pmatrix} I & & & \\ & \sqrt{u_1^T u_1} & & \\ & & & \sqrt{u_k^T u_k} \end{pmatrix} \\
&u = [u_1^T, ..., u_k^T]^T; \\
&b(m) = [(D(m)u_0)^T, 0, ..., 0]^T
\end{aligned}$$

We also define the matrix Q as a matrix of zeros and ones such that $Qu = u_k$. Obviously, the matrices A, B and $I(u)$ are never generated in practice and only matrix vector products of the form Lu and Du and are needed but it is easier to analyze our problem and to use numerical optimization techniques when the system is written in this form.

Using these definitions we replace the original eigenvalue optimization with the minimization or maximization of

$$\frac{1}{2\lambda^2} \approx \frac{1}{2} u^T Q^T Q u.$$

The advantage of this formulation is that while the original eigenvalue equation has, in general, n solutions however, the process we define here has a unique result. Also, by minimizing or maximizing the inverse of the eigenvalue square (rather than the eigenvalue itself) we get a simple expression to be optimized.

3 The discrete optimization problem

We now return to the discrete optimization problem. As explained in the introduction, we avoid the eigenvalue problem directly and replace it with the nonlinear system (6). This allows us to use simple optimization tools. Care must be taken such that we are solving the correct optimization problem, i.e. that the number of iterations in the inverse iteration process is sufficiently large to approximate the eigenvalue of the system. We return to this point later. For simplicity of notation we write the problem of maximizing the smallest eigenvalue. The minimization is done by simply changing the sign of the eigenvalue approximation and the level-set term.

The problem we solve is the following constrained optimization problem

$$\begin{aligned} \min \quad & \frac{1}{2} u^T Q^T Q u + \beta \left(\frac{1}{2} \gamma \|\nabla_h m\|^2 - h^d e^T \rho(m) \right) && \text{(7a)} \\ \text{s.t} \quad & C(m,u) = I(u)Au - B(m)u - b(m) = 0 && \text{(7b)} \end{aligned}$$

Following our work [8, 2, 9] we form the Lagrangian

$$\begin{aligned} \mathcal{J}(u,m,\mu) \;=\; & \frac{1}{2} u^T Q^T Q u + \beta \left(\frac{1}{2} \gamma \|\nabla_h m\|^2 - h^d e^T \rho(m) \right) \; + \\ & \mu^T (I(u)Au - B(m)u - b(m)) \end{aligned} \qquad (8)$$

where μ is a vector of Lagrange multipliers.

Differentiating with respect to u, m and μ we obtain the following nonlinear system of equations to be solved

$$\begin{aligned} \mathcal{J}_u &= Q^T Q u + C_u^T \mu = 0 && \text{(9a)} \\ \mathcal{J}_m &= \beta \left(\gamma \nabla_h^T \nabla_h m + h^d \rho'(m) \right) + C_m^T \mu = 0 && \text{(9b)} \\ \mathcal{J}_\mu &= I(u)Au - B(m)u - b(m) = 0 && \text{(9c)} \end{aligned}$$

The matrices C_m and C_u are the differentiation of the constraint $C(m, u)$ with respect to m and u respectfully.

$$\begin{aligned} C_u &= I(u)A + [I(u)Au^{\text{fix}}]_u - B(m) \\ C_m &= \left[\text{diag}\left((\rho'(m))(u_0) \right) \text{diag}\left((\rho'(m))(u_1) \right), ..., \text{diag}\left((\rho'(m))(u_{k-1}) \right) \right]^T \end{aligned}$$

It is important to note that the matrix C_u can be thought of a discretization of a differential operator and the matrix C_m is simply a combination of a diagonal positive matrices which implies that it involves with only zero order derivatives. This fact is crucial to the effective solution of the problem. Also, its important to note that the matrix $[I(u)Au^{\text{fix}}]_u$ is a lower block triangular dense matrix but its product with a vector can be calculated in $\mathcal{O}(n)$ operations by simple rearrangement of the vectors. The matrix C_u is a lower triangular matrix with the Laplacian on its diagonal and therefore it is possible to solve a system of the form $C_u v = w$ and $C_u^T v = w$ by solving a few Poisson equations that can be done efficiently by using multigrid. Our multigrid is a standard geometrical multigrid method with bilinear prolongation and restriction. We use two steps of symmetric Gauss-Siedel as a smoother and combine everything into a W-cycle. The number of cycles needed to obtain a relative accuracy of 10^{-8} range between 5-6.

4 Solving the optimization problem

In order to solve the optimization problem we use the reduced Hessian Sequential-Quadratic-Programming method (SQP) [14], utilizing the properties of the matrices in the Euler-Lagrange equations in order to solve the resulting linear systems.

We start by approximating the full Hessian of the problem by a Gauss-Newton approximation (see for details [9])

$$\begin{pmatrix} C_u & 0 & C_m \\ Q^T Q & C_u^T & 0 \\ 0 & C_m^T & \beta R \end{pmatrix} \begin{pmatrix} s_u \\ s_\mu \\ s_m \end{pmatrix} = - \begin{pmatrix} \mathcal{L}_\mu \\ \mathcal{L}_m \\ \mathcal{L}_u \end{pmatrix} \qquad (10)$$

In the reduced Hessian method, we eliminate the unknowns s_u and s_μ obtaining an equation for s_m alone.

We therefore start by eliminating s_u

$$s_u = C_u^{-1}\mathcal{L}_\mu - C_u^{-1}C_m s_m.$$

To calculate the term $C_u^{-1}\mathcal{L}_\mu$ solve the lower triangular block system $C_u v = \mathcal{L}_\mu$ which is done by solving k Poisson equation.

After eliminating s_u we eliminate s_μ. To do that we substitute the computed s_u into the Hessian system.

$$s_\mu = C_u^{-T}(\mathcal{L}_m - Q^T Q s_u).$$

Note that this requires the solution of the system

$$C_u^T v = w$$

This system is an upper triangular block system with L^T on its diagonal. We therefore use the same multigrid solver for the solution of this problem as well.

Substituting s_u and s_μ into (10) we obtain an equation for s_m alone. The equation has the form

$$(J^T J + \beta R'')s_m = C_m^T C_u^{-T}(\mathcal{L}_m + Q^T Q C_u^{-1}\mathcal{L}_\mu) \;\equiv\; g_r \tag{11}$$

where the matrix J is

$$J = -QC_u^{-1}C_m$$

The right hand side of this equation, g_r, is referred to as the reduced gradient and the dense matrix on the left hand side is referred to as the reduced Hessian. The reduced Hessian is dense and therefore, we do not compute it in practice. There are two options to proceed. First, we can solve the reduced Hessian system using the conjugate gradient (CG) method. At each CG iteration we require to solve systems of the form $C_u v = w$ and $C_u^T v = w$ which can be done using a multigrid method. If the number of CG iterations is small then such an approach can quickly converge.

Another approach is to use a quasi-Newton method in order to approximate the reduced Hessian and solve instead.

$$\widetilde{H_r} s_m = g_r$$

where $\widetilde{H_r}$ is an approximation to the reduced Hessian. In this case no or very simple matrix inversion is needed and the approximation to the reduced Hessian is obtained through the sequence of reduced gradients g_r. In this work we have chosen to use the L-BFGS method [14]. The advantage of the method is that the only matrix inversion that is needed for the solution of the approximate reduced Hessian is its initial guess and therefore we can quickly calculate the product of the inverse reduced Hessian times a vector. In general, the number of steps of this method can be larger compared with the Newton type approach however, for this problem, as we demonstrate in our numerical examples, the number of steps needed was very small. Our implementation of L-BFGS is standard (see [14]). The reduced Hessian is initiated to

$$H_0 = \beta(I + \gamma \nabla_h^T \nabla_h).$$

and we invert it at each iteration by using the same multigrid solver.

In order to globalize the optimization algorithm and to guaranty the convergence to a (local) minimum we use the l_1 merit function with a simple Armijo backtracking line-search (see [14] for details). We have found that using our continuation strategy (see Sections 5 and 6) this line search was more than sufficient and never fail thus we avoided the more involved and more expensive line search which involves evaluating reduced gradients in addition to objective functions.

5 Multilevel continuation

In order to speed-up computations, deal with possible high nonlinearities, gain extra accuracy and evaluate the penalty parameter we have embedded our nonlinear optimization solver within a multilevel continuation iteration. Multilevel methods are especially effective for this problem due to two main reasons.

- The smallest eigenvalue of an elliptic differential operator corresponds to a smooth eigenvector and therefore can be computed on a coarse grid with (relative) high accuracy.
- The function m is smooth and therefore we are able to represent it on a coarse grid.

As we now demonstrate, utilizing these properties we obtain a multilevel iteration that is highly efficient and reduces the computational time significantly. We base our method on our previous work [1] with some modifications to deal with the problem at hand. The algorithm is as follows:

Algorithm 1 - Grid continuation

- Initialize the parameter m_H and choose a vector q_H as initial guess to the first eigenvector on the coarse grid.
- while not converge
  1. Use the methods discussed in Section 4 to solve the optimization problem on the current grid.
  2. Use a search technique in the regularization parameter to approximately solve the mass constraint (5) $M = M(\beta)$.
  3. Output: m_H, λ_H, β, k and q_H where q_H is the approximation to the first eigenvector, β is the regularization parameter evaluated on grid H and k is the number of inverse iteration needed to compute the eigenvalue to a prescribed accuracy.
  4. Check for convergence
  5. Refine the m_H and q_H grid to h using bilinear interpolation.
  $$m_h = I_H^h \, m_H; \quad q_h = I_H^h \, q_H$$
  6. Set the initial guess for the eigenvector q_h.
  7. Set $H \leftarrow h$

In order to check for convergence we compare the difference between the computed models $\tau_m = ||I_H^h m_H - m_h||$ and the computed eigenvectors $\tau_u = ||I_H^h q_H - q_h||$ on fine and coarse grids. We stop the iteration when $\max(\tau_m, \tau_u) \leq \tau$. Typically in our implementation we choose $\tau = 10^{-3}$. In other cases we will use a pre-prescribe fine grid. In these cases we stop the process only when we have solved the equations on the pre-prescribe finest grid.

To evaluate the regularization parameter we use a simple secant method. This allow us to approximately solve the equation

$$h^d e^T \rho(m_h(\beta)) = M_h \tag{12}$$

by only a few evaluations of m for different β's. Our basic assumption is that $M(\beta)$ is a smooth function. If this is not the case then β on the coarse grid may not indicate the correct β on the fine grid. In virtually all of our numerical experiments we have found this process to be efficient. It is important to note that we should not try to over-solve equation (12). Recall that due to our parametrization, the mass is almost a discrete variable. We therefore solve (12) only approximately and stop when

$$\|h^d e^T \rho(m_h(\beta)) - M\| \leq l h^d$$

where l is a small integer (typically 2-4). Thus we allow some flexibility which is proportional to the mass of each pixel. We have found in our experiments that we are able to evaluate β accurately on a coarse grid and avoid expensive fine grid iterations.

There are a few other points in our continuation procedure which are different then standard continuation methods. The major being the output of the initial guess q_H and setting it as the initial guess for the next level. As we have discussed in the introduction, the choice of a good initial vector can reduce the number of fixed point iterations. If the initial vector is close enough to the first eigenvector then one requires a very small number of inverse iterations. We have found that using this strategy, the initial vector on the finest grid is very good and that we are typically able to reduce the number of inverse iterations on the finest grid to $2 - 3$.

Furthermore, using this approach we needed very few fine grid iterations (usually less than 5) and thus the total work on the fine grid was very minimal indeed.

6 Continuation in the fixed point iteration

Although our multilevel iteration is highly efficient we found that for some cases, the solution on the coarsest level required many iterations and in some cases fail to converge if we start from a very bad guess. In order to be able to always converge, even on coarse grids, and to obtain a more robust algorithm we have used a continuation in the fixed point parameter, that is, we start by doing very few inverse iterations and increase this number until we converge. The reason that such a method works well is that it is easy to verify by inspecting the equations that the problem becomes more nonlinear as the number of inverse iterations increase. Therefore, problems with small

number of inverse iterations tend to converge faster and give good initial guesses to the next problem with more inverse iterations. We summarize the algorithm as follows.

Algorithm 2 - Fixed point continuation

- Initialize the parameter m and choose a vector q as initial guess to the first eigenvector on the coarse grid. Set the number of fixed point iterations $k = 2$
- while not converge
  1. Use the methods discussed in Section 4 to solve the optimization problem.
  2. Use a search technique in the regularization parameter to approximately solve the mass constraint
  3. Output: m, λ, β and q
  4. Check for convergence
  5. Set $k \leftarrow k + 1$.

Similar to the grid continuation we successively improve our initial guess and obtain high accuracy in the computed eigenvectors and eigenvalues.

7 Numerical experiments

In this section we report on numerical experiments in 2 and 3D. The goal of these experiments is to demonstrate the effectiveness of the techniques

7.1 Maximizing and minimizing λ in 2D

Our first experiment we run the model problem in 2D where $\mathcal{L} = -\nabla^2$ which Drichlet boundary conditions, on a 65^2 grid points and use the LBFGS(10) method. We shoot for a total mass of 1.154. To obtain an eigenvalue with at least three digit accuracy we set (after some experiments) the number of fixed point iterations to 7. We then solve the problem first on the single fine grid without any continuation and follow the iteration. To solve the problem

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

on a single fine grid, we needed 44 fine grid iterations to reduce the absolute value of the gradient to 10^{-6}. This number is better than the iteration counts reported in [15] in a factor of $3-4$ but as we see next, we can be much more ambitions then that. Recall that each iteration requires 14 solutions of the Poison equation thus even in 2D using multigrid methods, the problem is computationally expensive.

When we use a multilevel continuation technique we can do much better. First, initializing the first eigenvector with the coarse grid interpolation, we can reduce the number of inverse iterations to 2 keeping the same accuracy and reducing nonlinearity. The number of iterations and the total mass on each grid is summarized in Table 1.

As we can see, the number of the iterations on the coarsest grid is somewhat large but this iteration is very cheap. Using the coarse grid solution as an initial guess, the number of finer grid iteration reduces dramatically. Finally the number of the finest grid iteration is reduced to only 3. Since we solve only 4 Poison equations at each iteration (due to the good initial approximation to the first eigenvector), compared with 14 when we do not use multilevel methods, the cost of each fine grid iteration in the multilevel setting is roughly 3.5 times cheaper than the iteration on a single grid. Thus the overall saving is more then a factor of a hundred.

In this case, even though the number of iterations on the coarse grid was large, the line search did not fail and we did not need to use the continuation in the fixed point iteration. However, when starting from different starting models we have used this continuation on the coarse grid if the line search fails.

The final result of the density on each grid in these experiments are plotted in Figure 1. The level-set function is plotted in Figures 2 Finally, we plot the convergence curve of this process in Figure 3. We can see that using

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

the multilevel process, the initial iteration on the finest grid had roughly a gradient norm of 10^{-5}.

We repeat the experiment but this time we minimize λ by changing the sign in Equation (4). The results in this case are recorded in Table 2. We see that we keep the overall efficiency and that our formulation is insensitive to this change.

7.2 Maximizing λ on a complicated domain

In the next experiment we demonstrate the ability of the level-set to change topology. We solve the problem on a square domain with two holes with a butterfly shape. Drichlet boundary conditions are imposed inside and outside. In this case we do not use a multilevel scheme as we are interested in following the changes in the density through the iteration.

We set the grid to 65×65 We set the total mass to 1.33 and trace the iteration. The stopping criteria is set to when the absolute value of the gradient is smaller than 10^{-6}. We start our iteration with a connected shape. We see that the shape quickly disconnect into two separate shapes that "travel" to the right place.

In this case there is no analytic solution to compare our results with. However, starting from different starting models yield identical results. The results are plotted in Figure 4.

7.3 Maximizing λ in 3D

As a third experiment we solve the same problem in 3D on a simple domain. The finest grid is set to 65^3 thus the size of the matrix is 274625^2. We set the total mass in this case to 1.11. Evaluating eigenvalues and eigenvectors

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

for this problem is a computationally demanding task. Here we did not use the single grid option but rather used our multilevel algorithm alone. The results of this experiment are summarized in Table 3

The final density from this experiment is plotted in Figure 5. We see again that our method is highly effective and allow us to work with large scale problems.

8 Summary and discussion

In this paper we have used multilevel methods combined with level-sets to solve an eigenvalue optimization problem. It is important to note that in this case as well as many other cases of optimal design and inverse problems, we do not care about the surface evolution. This is very different from the case of front propagation where tracking the front is important. Furthermore, evaluating the speed of propagation for the interface is computationally intensive as it involves with computing the solution to an eigenvalue problem. Thus the bottleneck of this computation is the solution of the forward problem. Using fast marching scheme and other front propagation techniques to solve such problems is very inefficient. Instead, one can use standard numerical optimization techniques combined with multilevel methods to obtain solutions in only a fraction of the cost that is needed for following the front.

The second technique we have demonstrated in this paper is the use of the inverse iteration process for eigenvalue optimization. As we stated in the introduction, using this approach, we can compute the eigenvalues and eigenvectors with increasing accuracy, avoiding the need to compute a very exact and time consuming eigenvector for gradient computation. Our work here involved only the first eigenvalue but we intend to expand our work to

the case of minimizing the gap between eigenvalues as well as other eigenvalue problems.

Acknowledgment

The author would like to thank Stanley Osher for introducing the problem.

References

[1] U. Ascher and E. Haber. Grid refinement and scaling for distributed parameter estimation problems. *Inverse Problems*, 17:571–590, 2001.

[2] U. Ascher and E. Haber. A multigrid method for distributed parameter estimation problems. *Preprint*, 2001.

[3] M. Bendsoe and C. Mota Soares. *Topology design of structuts*. Kluwer Academic, Dordrecht, MA, 1993.

[4] S. Cox and D. Dobson. Band structure optimization of two dimensional photonic crystals in h-polarization. *J. Comput. Phys.*, 158:214–223, 2000.

[5] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM, Philadelphia, 1996.

[6] H.W. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems*. Kluwer, 1996.

[7] C. Farquharson and D. Oldenburg. Non-linear inversion using general measures of data misfit and model structure. *Geophysics J.*, 134:213–227, 1998.

[8] E. Haber and U. Ascher. Preconditioned all-at-one methods for large, sparse parameter estimation problems. *Inverse Problems*, 2001. To appear.

[9] E. Haber, U. Ascher, and D. Oldenburg. On optimization techniques for solving nonlinear inverse problems. *Inverse problems*, 16:1263–1280, 2000.

[10] A Leito and O. Scherzer. On the relation between constraint regularization, level sets, and shape optimization. *Inverse Problems*, 19 (1):1–11, 2003.

[11] A.S. Lewis. The mathematics of eigenvalue optimization. *Mathematical Programming*, to appear, 2003.

[12] A.S. Lewis and M.L. Overton. Eigenvalue optimization. *Acta Numerica*, 5:149–190, 1996.

[13] K. Neymeyr. *Solving mesh eigenproblems with multigrid efficiency, in Numerical Methods for Scientific Computing. Variational problems and applications*. CIMNE, Barcelona, 2003.

[14] J. Nocedal and S. Wright. *Numerical Optimization*. New York: Springer, 1999.

[15] S. Osher and F. Santosa. Level set methods for optimization problems involving geometry and constraints i. frequencies of a two density inhomogeneous drum. *JCP*, 171:272–288, 2001.

[16] M. Overton. On minimizing the maximum eigenvalue of a symmetric matrix. *SIAM J. on Matrix Analysis and Applications*, 9(2):256–268, 1988.

[17] M.L. Overton. Large-scale optimization of eigenvalues. *SIAM J. Optimization*, 2:88–120, 1992.

[18] A. Shapiro and M.K.H. Fan. On eigenvalue optimization. *SIAM J. on Optimization*, 5:552–569, 1995.

[19] U. Trottenberg, C. Oosterlee, and A. Schuller. *Multigrid*. Academic Press, 2001.

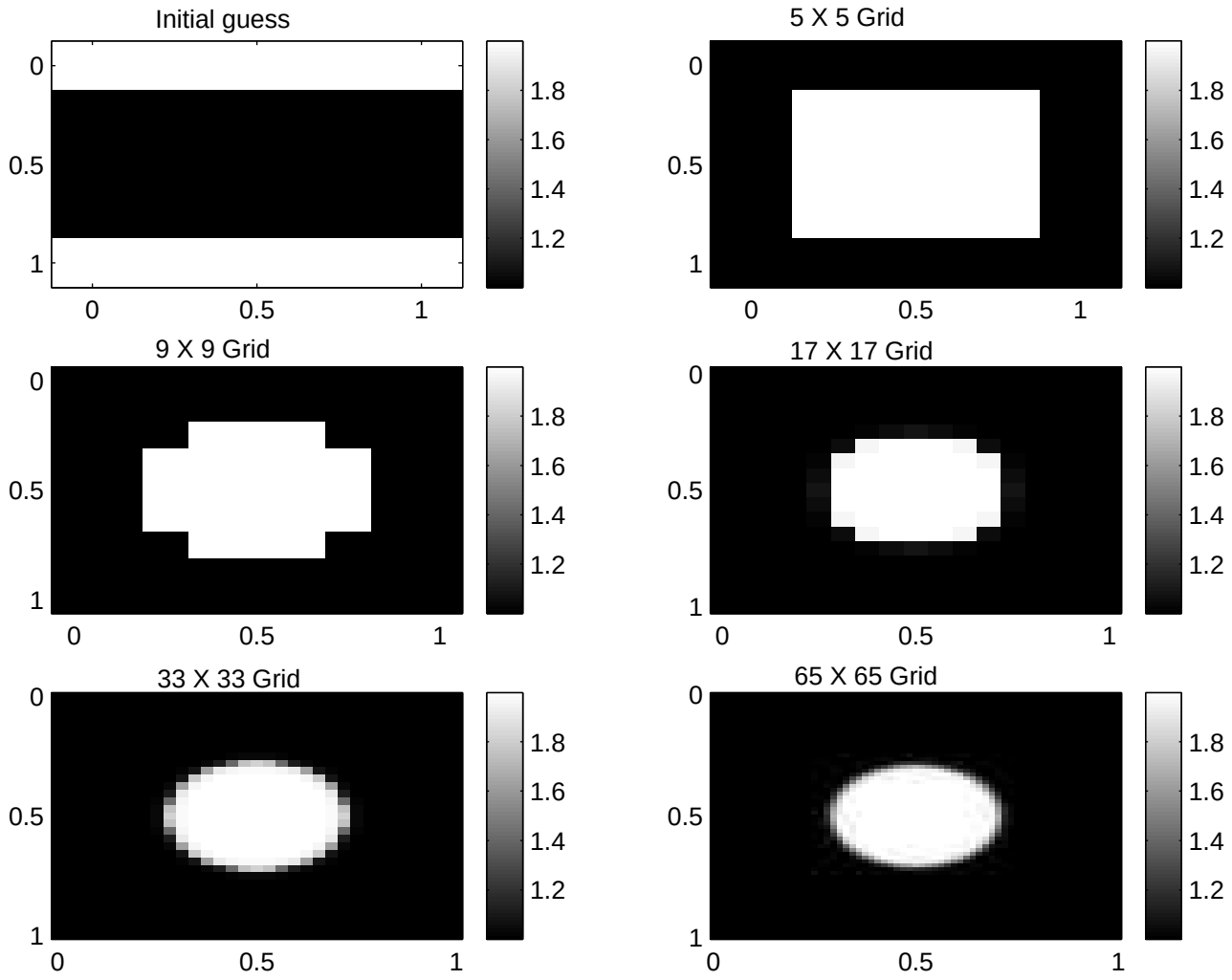

Figure 1: Evolution of the density when using the multilevel approach

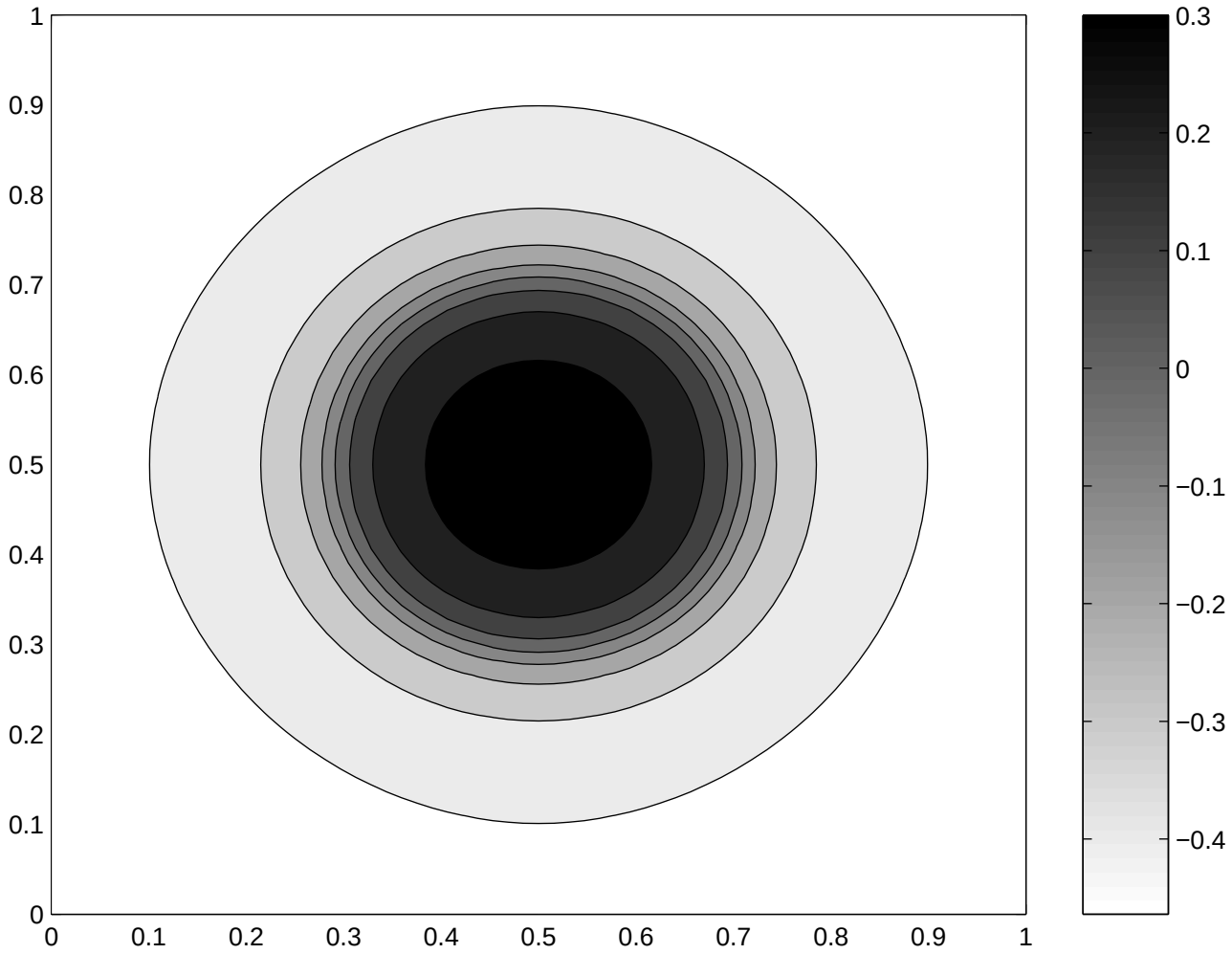

Figure 2: The final level set function on the finest grid

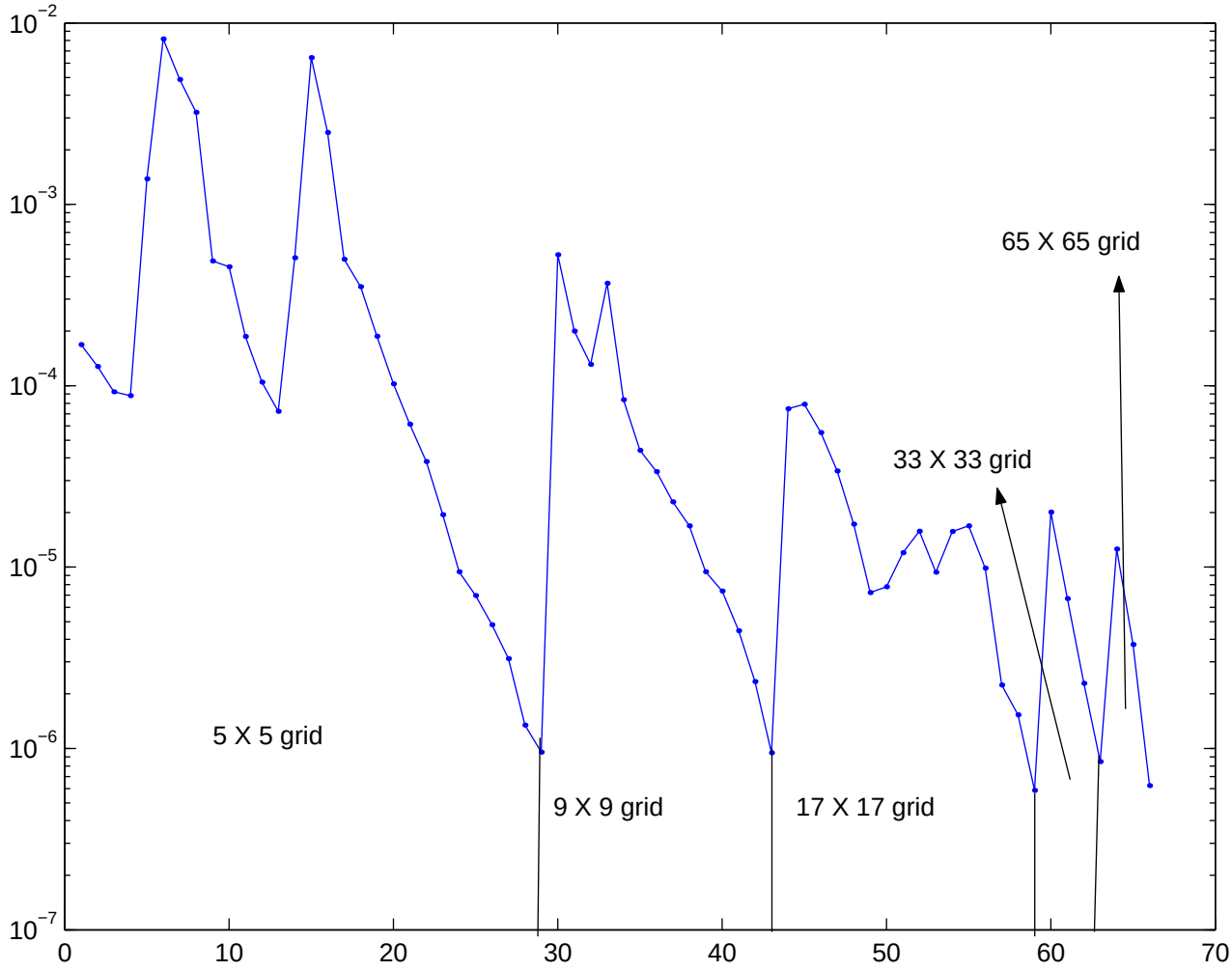

Figure 3: Convergence curve for the multilevel iteration

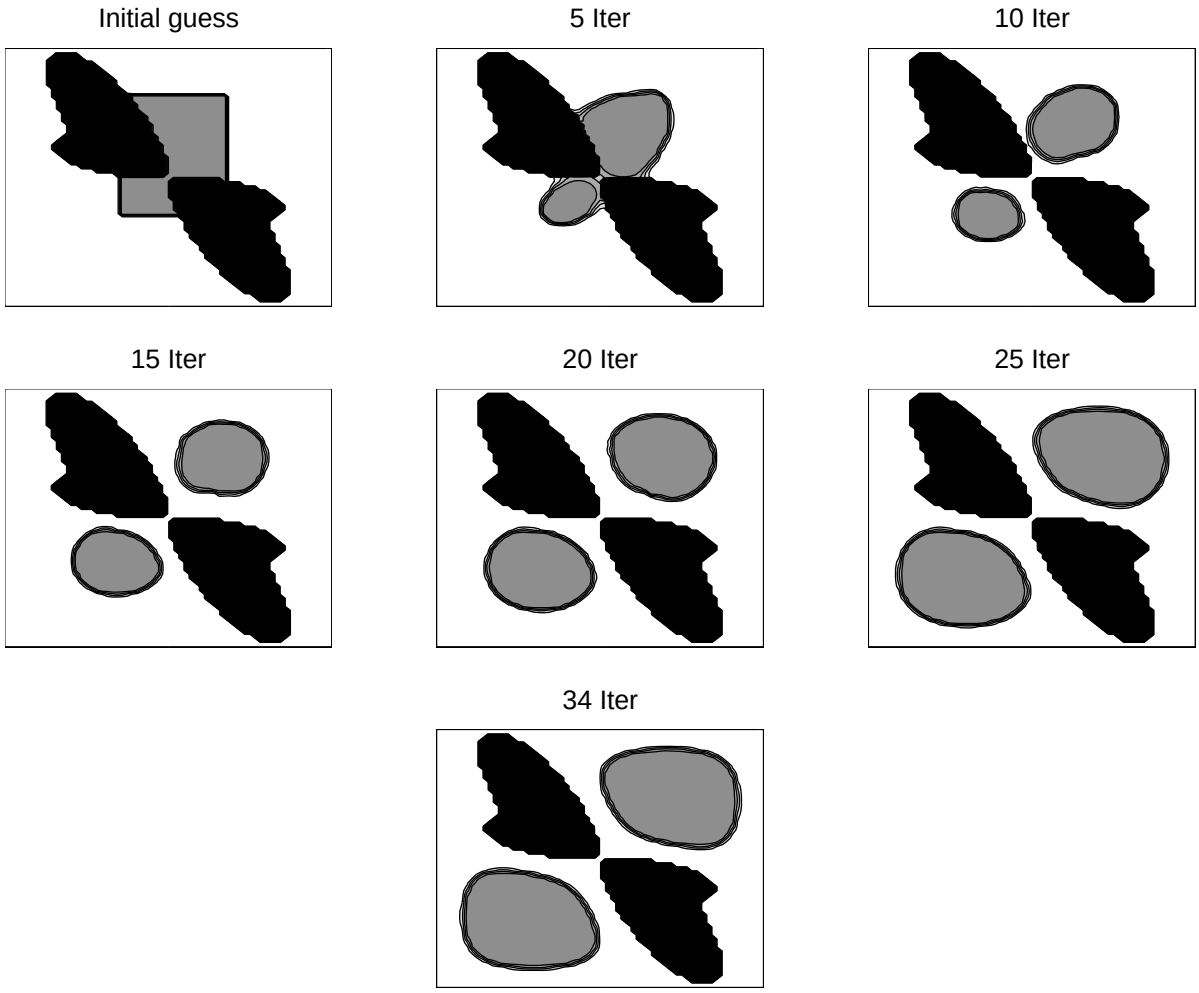

Figure 4: Density distributions with the iteration

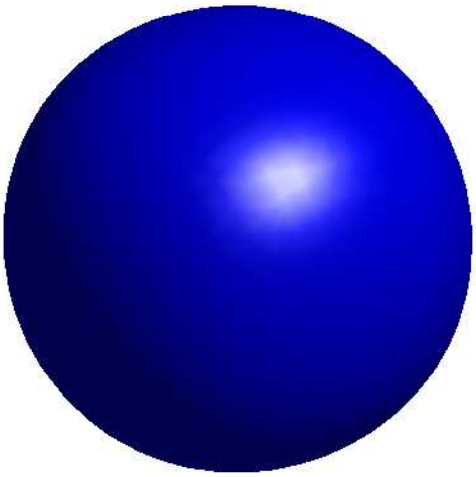

Figure 5: The final density on the finest 3D grid