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

\[
\begin{align*}
\min \text{ or } \max \quad & \lambda \\
\text{s.t.} \quad & \lambda \text{ is the minimal eigenvalue of } \mathcal{L}u = \lambda \rho(x)u \\
& \rho \in S \\
& \int_{\Omega} \rho \, dV = M
\end{align*}
\]

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

\(S\) is a space such that \(\rho\) can have values of \(\rho_1\) or \(\rho_2\) 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 $\rho$

• Solving an eigenvalue problem for a large scale application is computationally intensive

• For derivative information - eigenvectors are needed

• How to express the constraint $\rho \in S$?
Constraint elimination - level set methods

The constraint: $\rho(x) \in S$.
That is $\rho(x)$ can have only two values $\rho_1$ and $\rho_2$

Possible parameterizations
Track the interface between $\rho_1$ and $\rho_2$
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

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

\(\rho\) 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

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

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

Need regularization ...
Reformulation by penalty

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

\( \beta \) - Lagrange multiplier
\( \gamma \) - regularization parameter
The traditional approach

Take the steepest decent approach [Santosa & Osher 01]

Calculate Frech’et derivatives by perturbation

\[ \lambda^T \delta \rho = -\lambda \frac{u^T D(\delta \rho)u}{u^T D(\rho)u} \]

by computing the eigenpair \((u, \lambda)\)
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 $\lambda_1$; see Figure 6 for corresponding densities.
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 $j = 1 \ldots k$
  - Solve $Lu_j = \frac{1}{\sqrt{u_{j-1}^Tu_{j-1}}}Du_{j-1}$.

- Set $\lambda \approx \frac{1}{\sqrt{u_k^Tu_k}}$
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
\[ A = \text{diag}(L, L, ..., L); \]

\[ B(m) = \begin{pmatrix} 0 \\ D(m) \\ 0 \\ . \\ D(m) \\ 0 \end{pmatrix} \]

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

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

$$\min \quad \frac{1}{2} u^T Q^T Q u + \beta \left( \frac{1}{2} \gamma \|\nabla_h m\|^2 - h^d e^T \rho(m) \right)$$

s.t. \quad C(m, u) = I(u) A u - B(m) u - b(m) = 0

This is a class of PDE constraint optimization problem
[ Heinkenslusch, Ghattas, Burger, Ascher & Haber]

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

The Lagrangian

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

$\mu$ - a vector of Lagrange multipliers.
Differentiating we obtain a large nonlinear system of equations for \( m, u, \mu \)

\[
\mathcal{I}_u = Q^T Qu + C_u^T \mu = 0
\]

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

\[
\mathcal{I}_\mu = C(m, u) = 0
\]

\[
C(m, u) = I(u)Au - B(m)u - b(m)
\]
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

\[
\begin{pmatrix}
C_u & 0 & C_m \\
Q^T Q & C_u^T & 0 \\
0 & C_m^T & \beta R
\end{pmatrix}
\begin{pmatrix}
s_u \\
s_\mu \\
s_m
\end{pmatrix}
= -
\begin{pmatrix}
J_\mu \\
J_m \\
J_u
\end{pmatrix}
\]

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

The KKT system

\[
\begin{pmatrix}
C_u & 0 & C_m \\
Q^T Q & C_u^T & 0 \\
0 & C_m^T & \beta R
\end{pmatrix}
\begin{pmatrix}
s_u \\
s_\mu \\
s_m
\end{pmatrix}
=
\begin{pmatrix}
J_\mu \\
J_m \\
J_u
\end{pmatrix}
\]

For large $\beta$'s - equivalent to a diagonally dominant system of PDE's

But $\beta$ is small - Highly coupled system of PDE's

[Haber & Ascher 01]
Solving the linear subproblem - the reduced Hessian method

\[
\begin{pmatrix}
C_u & 0 & C_m \\
Q^TQ & C_u^T & 0 \\
0 & C_m^T & \beta R
\end{pmatrix}
\begin{pmatrix}
s_u \\
s_\mu \\
s_m
\end{pmatrix}
=
-\begin{pmatrix}
J_\mu \\
J_m \\
J_u
\end{pmatrix}
\]

Eliminate

\[
s_u = -C_u^{-1}(J_\mu + C_m s_m)
\]
\[
s_\mu = -C_u^{-T}(J_m + Q^T Q s_u)
\]
The reduced Hessian method

Obtain an equation for $s_m$

$$(J^T J + \beta R'')s_m = C_m^T C_u^{-T} (L_m + Q^T Q C_u^{-1} L_\mu) \equiv g_r$$

where

$$J = -Q C_u^{-1} C_m$$
Solving the The reduced Hessian system

**method I**
Use Conjugate Gradient
At each CG iteration solve $Cuv = w$ and $C^Tv = w$

**method II**
Approximate the reduced Hessian using L-BFGS method.
No matrix inversion needed
Example I - min $\lambda$

Problem in 2D
\[ \mathcal{L} = -\Delta \]
Discretize on a $65^2$

Number of iterations - 44

Cost per iteration - 14 solves of the Poison 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: $m_H, \lambda_H, \beta, k$ 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 $H \leftarrow h.$
Table 1: Iterations per grid for maximizing $\lambda$

<table>
<thead>
<tr>
<th>Grid</th>
<th>Iterations</th>
<th>Final mass</th>
</tr>
</thead>
<tbody>
<tr>
<td>$5^2$</td>
<td>29</td>
<td>1.3601</td>
</tr>
<tr>
<td>$9^2$</td>
<td>14</td>
<td>1.2592</td>
</tr>
<tr>
<td>$17^2$</td>
<td>16</td>
<td>1.1584</td>
</tr>
<tr>
<td>$33^2$</td>
<td>4</td>
<td>1.1538</td>
</tr>
<tr>
<td>$65^2$</td>
<td>3</td>
<td>1.1539</td>
</tr>
</tbody>
</table>
## Example II - max $\lambda$

<table>
<thead>
<tr>
<th>Grid</th>
<th>Iterations</th>
<th>Final mass</th>
</tr>
</thead>
<tbody>
<tr>
<td>$5^2$</td>
<td>38</td>
<td>1.603</td>
</tr>
<tr>
<td>$9^2$</td>
<td>19</td>
<td>1.588</td>
</tr>
<tr>
<td>$17^2$</td>
<td>12</td>
<td>1.567</td>
</tr>
<tr>
<td>$33^2$</td>
<td>6</td>
<td>1.562</td>
</tr>
<tr>
<td>$65^2$</td>
<td>4</td>
<td>1.562</td>
</tr>
</tbody>
</table>

Table 2: Iterations per grid for minimizing $\lambda$
Example III - max \( \lambda \) - complicated domain

Use \( \mathcal{L} = -\Delta \)

Domain has "holes"

Neumann BC inside the holes, Dirichlet outside

Test for changing topology
Example IV - max $\lambda$ in 3D

<table>
<thead>
<tr>
<th>Grid</th>
<th>Iterations</th>
<th>Final mass</th>
</tr>
</thead>
<tbody>
<tr>
<td>$5^2$</td>
<td>52</td>
<td>1.261</td>
</tr>
<tr>
<td>$9^2$</td>
<td>13</td>
<td>1.132</td>
</tr>
<tr>
<td>$17^2$</td>
<td>9</td>
<td>1.116</td>
</tr>
<tr>
<td>$33^2$</td>
<td>5</td>
<td>1.110</td>
</tr>
<tr>
<td>$65^2$</td>
<td>2</td>
<td>1.112</td>
</tr>
</tbody>
</table>

Table 3: Iterations per grid for maximizing $\lambda$ 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?