Topology Optimization of a Multiphysics Problem in Semiconductor Laser Design: Code Documentation

L. Adam

May 7, 2017

1 Problem description

The code solves the following problem:

$$\begin{aligned} \text{minimize} \quad & -\int_\Omega \varphi_{Ge}^p \Theta^q \operatorname{tr} e(\boldsymbol{u}) dx + \alpha f_{GL}(\boldsymbol{\varphi}) \\ \text{subject to} \quad & (\boldsymbol{\varphi}, \boldsymbol{u}) \text{ satisfies the elasticity equation (E)}, \\ & (\boldsymbol{\varphi}, \Theta) \text{ satisfies the Helmholtz equation (H)}, \\ & \boldsymbol{\varphi} \in \mathcal{G}_{ad}. \end{aligned} \tag{1}$$

Here, $\alpha > 0$ is a weighting parameter and the Ginzburg-Landau energy is of form

$$f_{\mathrm{GL}}(\boldsymbol{\varphi}) := \int_\Omega \frac{\varepsilon}{2} |\nabla \boldsymbol{\varphi}|^2 + \frac{1}{2\varepsilon} \boldsymbol{\varphi} \cdot (1 - \boldsymbol{\varphi}) dx, \tag{2}$$

The linear elasticity reads

$$\begin{aligned} -\operatorname{div}[\mathbb{C}(\boldsymbol{\varphi}) e(\boldsymbol{u}) - F(\boldsymbol{\varphi})] &= 0 && \text{in} \quad \Omega, \\ \boldsymbol{u} &= \epsilon_0 (x, y)^\top && \text{on} \quad \partial\Omega, \end{aligned} \tag{E}$$

where $e(\mathbf{u}) := \frac{1}{2}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^\top)$ is the symmetric strain of the displacement vector $\mathbf{u}$, $\mathbb{C}(\boldsymbol{\varphi})$ is a fourth-order tensor and

$$F(\boldsymbol{\varphi}) := \epsilon_0 \mathbb{C}(\boldsymbol{\varphi}) \begin{pmatrix} \varphi_{SiO_2} & 0 \\ 0 & \varphi_{SiO_2} \end{pmatrix} - \sigma_0 \begin{pmatrix} \varphi_{SiN} & 0 \\ 0 & \varphi_{SiN} \end{pmatrix}, \tag{3}$$

The Helmholtz equation is defined via

$$\begin{aligned} -\Delta \Theta - g(\boldsymbol{\varphi}) \Theta &= \lambda \Theta && \text{in} \quad \Omega, \\ \Theta &= 0 && \text{on} \quad \partial\Omega, \end{aligned} \tag{4}$$

where

$$g(\boldsymbol{\varphi}) := \left(\frac{2\pi}{\mu}\right)^2 \sum_{i=1}^N \varepsilon_{r,i} \varphi_i, \tag{5}$$

where $\varepsilon_{r,i}$ are relative permittivities of individual materials and then (H) is defined via

$$\begin{aligned} &\lambda \text{ is the smallest eigenvalue of (4)} \\ &\Theta \text{ is the corresponding eigenfunction of (4) with } \|\Theta\|_{L^r(\Omega)} = 1. \end{aligned} \tag{H}$$

Finally, using the standard definition of the Gibbs simplex

$$\mathcal{G} := \{\boldsymbol{\varphi} |\ \boldsymbol{\varphi} \geq 0,\ \mathbb{1} \cdot \boldsymbol{\varphi} = 1 \text{ a.e. in } \Omega\}, \tag{6}$$

we define the set of feasible solution as

$$\mathcal{G}_{ad} := \{\boldsymbol{\varphi} \in \mathcal{G} \cap H^1(\Omega, \mathbb{R}^N) |\ \varphi_i = 1 \text{ a.e. on } \Pi_i,\ i = 1, \ldots, N\},$$

thus material i has to be present at Π_i.

2 Solution methods

There are two implemented method: one based on a game theory and the other one on a direct solution of (1).

2.1 Separating the Physics: An Alternating Minimization Approach

In this approach we consider the elasticity equation (E) and the Helmholtz equation (H) as separate "players" who look for an equilibrium. The first player maximizes the strain in the optical cavity D by solving

$$\begin{aligned} \text{minimize} \quad & -\int_D \operatorname{tr} e(\boldsymbol{u}) + \alpha f_{GL}(\boldsymbol{\varphi}) \\ \text{subject to} \quad & (\boldsymbol{\varphi}, \boldsymbol{u}) \text{ satisfies the elasticity equation (E)}, \\ & \boldsymbol{\varphi} \in \mathcal{G}_{ad}, \\ & \varphi_{Ge} = 1 \text{ on } D \end{aligned} \tag{7}$$

for $\boldsymbol{\varphi}$. The second player then takes the material distribution $\boldsymbol{\varphi}$ and determines the new optical cavity via

$$D = \{x \in \Omega |\ S_\Theta(\boldsymbol{\varphi}) \geq c_\Theta \|S_\Theta(\boldsymbol{\varphi})\|_\infty\}. \tag{8}$$

Since $\Theta > 0$ a.e. on Ω, we have to apply a cutoff by some small $c_\Theta \in (0, 1)$. This process is iterated until an equilibrium is reached.

2.2 A Simultaneous Approach

For solving directly (1) we use the projected gradients method. Denoting the reduced objective of (1) by $\hat{\mathcal{J}}$, we solve at each step

$$\boldsymbol{\varphi}^{k+1} = Proj_{\mathcal{G}}\big(\boldsymbol{\varphi}^k - t^k \hat{\mathcal{J}}'(\boldsymbol{\varphi}^k)_{Riesz}\big), \tag{9}$$

where we employ the Riesz representation of the derivative in the primal space. This is computed by solving

$$\begin{aligned} -\Delta \boldsymbol{\xi} + \boldsymbol{\xi} &= \hat{\mathcal{J}}'(\boldsymbol{\varphi}) \qquad \text{in} \quad \Omega, \\ \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{n}} &= 0 \qquad\quad \text{on} \quad \partial\Omega. \end{aligned} \tag{10}$$

In an implementation, we make use of the generalized Armijo step size rule, where for a given $\sigma > 0$ we choose some $t^k > 0$ such that the following inequality

$$\hat{\mathcal{J}}(\boldsymbol{\varphi}^k) - \hat{\mathcal{J}}(\boldsymbol{\varphi}^{k+1}) \geq \sigma \frac{\|\boldsymbol{\varphi}^k - \boldsymbol{\varphi}^{k+1}\|^2_{H^1(\Omega,\mathbb{R}^N)}}{t^k} \tag{11}$$

is satisfied. Note that if $\mathcal{G}$ were the whole space, then (11) would reduce to the classical Armijo rule. For all $c > 0$ the optimality condition reads

$$\boldsymbol{\varphi} - Proj_{\mathcal{G}}\left(\boldsymbol{\varphi} - c\hat{\mathcal{J}}'(\boldsymbol{\varphi})_{Riesz}\right) = 0. \tag{12}$$

Thus, we stop the algorithm whenever the residual of (12) is smaller than some $\text{tol}_{PG} > 0$.

3 Code description

In this section, parameters and then the most important part of individual functions are described. The whole optimization is considered only in $\boldsymbol{\varphi}$, the remaining variables $\boldsymbol{u}$ and Θ are uniquely determined by it. Since $\boldsymbol{\varphi}$ is prespribed on $\cup_i \Pi_i$, the optimization is considered only on $\Omega \setminus \cup_i \operatorname{cl} \Pi_i$. But since all matrices are considered for all nodes, the reduced $\boldsymbol{\varphi}$ have to be prolonged to the whole domain. This is done in function `ProlongPhi`. Mathematically it is realized via a linear mapping $\boldsymbol{\varphi}_{\text{full}} = P\boldsymbol{\varphi}_{\text{red}} + \hat{\boldsymbol{\varphi}}$, where $\hat{\varphi}_i = \chi_{\Pi_i}$. Then in P^* has to be inserted to all derivatives computed in the paper. This is done in `ShortenPhi`. Note that every time when the optical cavity changes, the domain of P changes as well (since φ_{Ge} is prescribed there) and needs to be recomputed. This is done in `ModifyMatrices`.

3.1 Parameters

The parameters with their Matlab name, notation from this file and short description are summarized in Tables 1, 2 and 3.

Matlab	Latex	Description
`coarsenCount`	-	Number of times the elements are coarsened when passing to a finer mesh
`cutThreshold`	c_Θ	Cutoff parameter from (8)
`fixGe`	D	The optical cavity from the alternating approach, see (8)
`iterMax`	-	Number of maximal iterations for the projected gradients
`iterMaxIn`	-	Number of outer iterations for the alternating approach; the number of solves of (7) and (8) on each mesh
`refineCount`	-	Number of times the elements are refined when passing to a finer mesh
`refineMesh`	-	Number the meshes, the first mesh is refined `refineMesh-1` times
`wavelength`	μ	The wavelength from (5)

Table 1: Parameters scattered all over the code

Matlab	Latex	Description
`alpha`	α	Weighting parameter in front of the Ginzburg-Landau energy
`epsilon`	ε	Interfacial thickness and part of the Ginzburg-Landau energy (2)
`cOptimality`	-	Stepsize for the verification of optimality conditions
`epsilonR`	$(\frac{2\pi}{\mu})^2\varepsilon_{r,i}$	Combination of wavelength and relative permittivities from (5)
`eps0`	ε_0	Eigenstrain generated by SiO_2 on Ge from (3)
`lambda`	-	First Lamé parameter
`mu`	-	Second Lamé parameter
`sigma0`	σ_0	Thermal (pre-)stress generated by SiN from (3)

Table 2: Parameters in the structs `constants` and `material`

Matlab	Latex	Description
`computeG`	-	Determines whether the gradient should be computed. Used only for a speed-up.
`cutoffs`	-	Determines whether cutoffs are applied within $\mathbb{C}$ and g
`jointObjectivePhiLp`	p	Exponent from (1)
`jointObjectiveThetaLp`	q	Exponent from (1)
`method`	-	Determines whether the alternating minimization (`method = 1`) or the simultaneous approach (`method = 0`) is used
`normalizationLp`	r	Normalization space from (H)
`originalEps0`	-	Determines whether the original elasticity (where ε_0 and σ_0 had different values and the eigenstrain was placed in germanium) should be used
`symmetrize`	-	Determines whether $\boldsymbol{\varphi}$ and Θ should be symmetrized along the x-axis

Table 3: Parameters in the struct `options`

3.2 Function `Main.m`

Since it is not clear how to choose some parameters, the code can be run in parallel thanks to

```
parfor parsIndex = 1:size(parsAll, 2)
```

The important part starts at

```
for meshIndex = 1:refineMesh
```

where, after the parameters being assigned, the solution is found on the first mesh, the mesh is refined and the data are prolonged onto this new mesh.

Since ε corresponds to the interfacial thickness, it is chosen as twice the size of smallest element `2*max(meshMaxElement)`. For the alternating minimization method we store in `fixGe` the initial optical cavity which will (usually) start shrinking based on the value of c_Θ. To refine the mesh, all data which are to be prolonged are stored in `dataToProlong`. In

```
if any(values(:) < 1 - refinementTol & values(:) > refinementTol) || ...
```

we identify all elements, where the phases are not pure (extract above) or where there is transition between phases (in the code after `...`). The element is not refined if it is already too small due to

```
if side1 >= 0.9*meshMaxElement(1) && side2 >= 0.9*meshMaxElement(2)
```

This process is repeated `refineCount` times. After doing so, the mesh is coarsened `coarsenCount` times based on similar rules. Since the mesh is changed, all matrices have to be recomputed (this is done in the simplest possible way). After doing so, `TriInfo`, `Transformation` and `matrices` are recomputed and ε is decreased by half (since the width of the intefacial region was halfed as well). Since the Lamé parameters of air depend on ε, they have to be recomputed in

```
ObtainData(epsilon, alpha, wavelength, options);
```

After doing so, the actual optimization process may start. For the alternating minimization `method == 1`, the optical cavity `fixGe` is iteratively updated until a fixed point is found. Since the optical cavity D in (E) equals to `fixGe == 1`, the operator in the objective has to be recomputed in

```
ModifyMatrices(fixGe, TriInfo, matrices, Transformation, phi);
```

Moreover, due to numerical errors, the solution may be non-symmetric along the x-axis. Even though this does not matter much when purely the elasticity is considered, when the Helmholtz equation is added, it may cause the support of Θ to be located only on the left/right part of the domain and a possible collapse of the method. To prevent this, the solution is symmetrized in

```
SymmetryCompute(phi, TriInfo, 1, 1, 1e-8);
```

Whenever the symmetrization error is greater than 10^{-8}, a warning will occur. For the joint approach, described in the paper, `method == 0` it suffices to run the projected gradients.