User's Manual for the code for Helmholtz transmission eigenvalues

Xia Ji, Jiguang Sun, Tiara Turner

November 21, 2011

1 Introduction

This is a program written in MATLAB for solving the transmission eigenvalues for Helmholtz equation. For the case of scattering of time-harmonic acoustic waves by a bounded simply connected inhomogeneous medium $D \in \mathbb{R}^2$, the transmission eigenvalue problem is to find $k \in \mathbb{C}, u, v \in L^2(D), u-v \in H^2(D)$ such that

$$\begin{aligned} \Delta u + k^2(1+q(x))u &= 0, \\ \Delta v + k^2 v &= 0, \\ u - v &= 0, \\ \tfrac{\partial u}{\partial \nu} - \tfrac{\partial v}{\partial \nu} &= 0, \end{aligned}$$

where $n(x) = 1+q(x)$ is the index of refraction. Using the mixed method, we can obtain the following weak problem, find $(k^2, u, v) \in \mathbb{C} \times H_0^1(D) \times H^1(D)$ such that

$$\begin{aligned} (\Delta v, \Delta \phi) &= k^2(v, \phi), \quad \forall \phi \in H_0^1(D), \\ (\Delta u, \Delta \phi) + (qv, \phi) &= k^2((1+q)u, \phi), \quad \forall \phi \in H^1(D). \end{aligned}$$

Given finite dimensional spaces $S_h \subset H^1(D)$ and $S_h^0 \subset H_0^1(D)$ such that $S_h^0 \subset S_h$, the discrete problem is to find $(k_h^2, u_h, v_h) \in \mathbb{C} \times S_h^0 \times S_h$ such that

$$\begin{aligned} (\Delta v_h, \Delta \phi_h) &= k_h^2(v_h, \phi_h), \quad \forall \phi_h \in S_h^0, \\ (\Delta u_h, \Delta \phi_h) + (qv_h, \phi_h) &= k_h^2((1+q)u_h, \phi_h), \quad \forall \phi \in S_h. \end{aligned}$$

Standard piecewise linear finite elements are used to discretize the problem

$$\begin{aligned} S_h &= \text{the space of continuous piecewise linear finite elements on } D, \\ S_h^0 &= S_h \cap H_0^1(D) \\ &= \text{the subspace of functions in } S_h \text{ that have vanishing DoF on } \partial D, \end{aligned}$$

where DoF stands for degree of freedom. Let $\psi_1, \ldots, \psi_K$ be a basis for S_h^0 and $\psi_1, \ldots, \psi_K, \psi_{K+1}, \ldots, \psi_T$ be a basis for S_h. Let $u_h = \sum_{i=1}^K u_i \psi_i$ and $v_h = \sum_{i=1}^T v_i \psi_i$. Furthermore, let $\mathbf{u} = (u_1, \ldots, u_K)^T$ and $\mathbf{v} = (v_1, \ldots, v_T)^T$. Then the corresponding matrix problem is

$$\begin{aligned} S_{K\times T}\mathbf{v} &= k_h^2 M_{K\times T}\mathbf{v}, \\ S_{T\times K}\mathbf{u} + M^q_{T\times T}\mathbf{v} &= k_h^2 M^{1+q}_{T\times K}\mathbf{u}, \end{aligned}$$

where the matrices are defined as follows

Matrix	Dimension	Definition
$S_{K\times T}$	$K \times T$	$S^{i,j}_{K\times T} = (\nabla\psi_i, \nabla\psi_j), 1 \le i \le K, 1 \le j \le T$
$S_{T\times K}$	$T \times K$	$S^{i,j}_{T\times K} = (\nabla\psi_i, \nabla\psi_j), 1 \le i \le T, 1 \le j \le K$
$M_{K\times T}$	$K \times T$	$M^{i,j}_{K\times T} = (\psi_i, \psi_j), 1 \le i \le K, 1 \le j \le T$
$M^{1+q}_{T\times K}$	$T \times K$	$(M^{1+q}_{T\times K})^{i,j} = ((1+q)\psi_i, \psi_j), 1 \le i \le T, 1 \le j \le K$
$M^q_{T\times T}$	$T \times T$	$(M^q_{T\times T})^{i,j} = (q\psi_i, \psi_j), 1 \le i \le T, 1 \le j \le T$

The generalized eigenvalue problem we need to solve is

$$\begin{pmatrix} S_{K\times T} & 0_{K\times K} \\ M^q_{T\times T} & S_{T\times K} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{u} \end{pmatrix} = k^2 \begin{pmatrix} M_{K\times T} & 0_{K\times K} \\ 0_{T\times T} & M^{1+q}_{T\times K} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{u} \end{pmatrix}.$$

For simplicity, we shall write this problem as

$$A\mathbf{x} = \lambda B\mathbf{x} \tag{1.1}$$

where

$$A = \begin{pmatrix} S_{K\times T} & 0_{K\times K} \\ M^q_{T\times T} & S_{T\times K} \end{pmatrix}, \quad B = \begin{pmatrix} M_{K\times T} & 0_{K\times K} \\ 0_{T\times T} & M^{1+q}_{T\times K} \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} \mathbf{v} \\ \mathbf{u} \end{pmatrix}.$$

The idea of the code is to construct the matrices first, then solve the generalized eigenvalue problems.

2 How to run this code

The main m-file is **'mixFemTE'**. To compute transmission eigenvalues, first type **'mixFemTE'** in Matlab Command Window. The program first asks the user to input the mesh file by displaying

```
Input name of the mesh file -->
```

The user types the mesh file name, for example,

```
'halfcircle'
```

Then the program asks the user to input the supremum of index of refraction by displaying

```
Input the supremum of index of refraction -->
```

The user could type, for example,

```
16
```

Note that the actual definition for the index of refraction $n(x)$ is defined in **'Rindex'**. 16 is the supremum of index in the current **'Rindex'**. You must change this file to change the index of refraction.

Finally the program asks how many transmission eigenvalues to be computed by displaying

```
Input the number of transmission eigenvalues -->
```

The user could type, for example,

```
4
```

Taking all above input, the program does the following

1. Construct the stiffness matrix S, mass matrix M and weighted mass matrix Mn (by calling subroutine **'assemble'**).
2. Identify the interior and boundary nodes and store them in 'Inode' and 'Bnode' (by calling subroutine **'intnode'**).
3. Compute the first Dirichlet eigenvalue (by calling subroutine **'DirichletEig'**).
4. Construct the matrices A and B for the generalized eigenvalue problem (by calling subroutine **'MixMethod'**).
5. Compute transmission eigenvalues (by calling subroutine **'sptarnite'**).

3 Modula of this code

This program consists of the following parts:

- Mesh generation.
- Construct the mass matrix, stiffness matrix and weighted mass matrix.
- Identify the interior and boundary nodes.
- Compute the first Dirichlet eigenvalue for $-\Delta$ in D.
- Construct the matrices for the transmission eigenvalues problem.
- Solve the eigenvalue problem by an iterative algorithm using restarted Arnoldi method.

3.1 Mesh generation

We use the PDE toolbox in MATLAB to generate the mesh, for example,

```
pdecirc(0,0,1/2,'C1')
```

This command can generate the mesh on a circle centered at $(0, 0)$ with radius $1/2$. the command **'Export Mesh'** can be used to save the information of the mesh.

We may load the information of the mesh

```
p= mesh.p'; t = mesh.t(1:3,:)'; e= mesh.e(1:2,:)';
```

- In the Point matrix p, the first and second rows contain x- and y-coordinates of the points in the mesh.
- In the Edge matrix e, the first and second rows contain indices of the starting and ending point, the third and fourth rows contain the starting and ending parameter values, the fifth row contains the edge segment number, and the sixth and seventh row contain the left- and right-hand side subdomain numbers.

- In the Triangle matrix t, the first three rows contain indices to the corner points, given in counter clockwise order, and the fourth row contains the subdomain number.

Any other tools which can generate such information can also be used in the mesh generation part.

3.2 Construct the related matrices

We construct the stiffness matrix, mass matrix and weighted mass matrix with the following four subroutines:

- **'assemble'** (function [S, M, Mn]=assemble(mesh))

  Description: construct the stiffness matrix, mass matrix and weighted mass matrix

  Input: the matrix 'mesh'

  Output: mass matrices 'M', stiffness matrices 'S' and the weighted mass matrix 'Mn'

- **'quad_setup2'** (function [weight,place]=quad setup2())

  Description: get the quadrature weights and points on the reference element

  Input: no

  Output: quadrature weights 'weight' and quadrature points 'place'

- **'phihat'** (function [y,grady]=phihat(xhat))

  Description: gets all the basis function values and gradients at the quadrature points on the reference element

  Input: the position on the reference element 'xhat'

  Output: function values 'y' and gradients 'grrady' at 'xhat'

- **'Rindex'** (n=Rindex(x,index))

  Description: give the refractive index at 'x'

  Input: position 'x', index of the medium 'index'

  Output: refractive index 'n'

The physical element is mapped by an affine transformation onto the reference triangle $\{(x, y)|, 0 \leq x, y \leq 1, 0 \leq x + y \leq 1\}$, then the information on the physical element can be derived from the reference one. The information of the reference element is obtained in **'quad_setup2'** and **'phihat'**. In **'assemble'**, we construct the desired information on each physical element, then assemble it into big matrices.

3.3 Identify interior and boundary nodes

The subroutine **'intnode'** (function [Inode,Bnode]=intnode(mesh)) fulfills the task of finding the interior and boundary nodes.

Description: identify the interior and boundary nodes and store them

Input: the matrix 'mesh'

Output: interior nodes index 'Inode' and boundary nodes index 'Bnode'

The following are the codes **'intnode'**, the lines starting with % are comments.

```
function [Inode,Bnode]=intnode(mesh)
% find boundary points using boundary edge
Bnode = sort(unique([mesh.e(1,:) mesh.e(2,:)]))';
% find interior points using boundary points
Inode = setdiff((1:length(mesh.p)),Bnode)';
```

3.4 Compute the first Dirichlet eigenvalue for $-\triangle$

This subroutine **'Dirichlet'** (function lambda=DirichletEig(SS, MM)) gives the first Dirichlet eigenvalue for $-\triangle$.

Description: compute the first Dirichlet eigenvalue for $-\triangle$ on D

Input: the mass matrix 'MM' and stiffness matrix 'SS' after applying the boundary condition

Output: the first Dirichlet eigenvalue 'lambda'

```
function lambda=DirichletEig(SS, MM)
lambda=eigs(SS, MM, 1, 'SM');
```

'eigs' is used to compute the eigenvalues, we compute the Dirichlet eigenvalues by calling

```
lambda=DirichletEig(S(Inode,Inode), M(Inode,Inode));
```

S, M are the stiffness and mass matrices generated in **'assemble'**.

3.5 The matrices for the transmission eigenvalues problem

With the subroutine **'assemble'** and **'intnode'**, we are ready to construct the matrices for the transmission eigenvalues problem, this is the subroutine **'MixMethod'** ([A,B]=MixMethod(S, M, Mn, Inode, Bnode)), this code is clear with the definition of the matrices A and B (1.1).

Description: construct the matrices for the transmission eigenvalues problem

Input: the mass matrix M, stiffness matrix S, weighted mass matrix Mn, interior nodes index 'Inode' and boundary nodes index 'Bnode'

Output: matrices A and B in (1.1)

3.6 Solve the eigenvalue problem

Now we can solve the eigenvalue problem by the restarted Arnoldi method. This is the subroutine **'sptarnite'** (function k=sptarnite(A,B,lb,noe)),

Description: solve the eigenvalue problem by the restarted Arnoldi method

Input: matrices 'A' and 'B', left lower bound 'lb', number of eigenvalues 'noe'

Output: 'noe' eigenvalues

At beginning, the left bound is given by $lb = \lambda_0/sup_D(n)$ and the right bound is given by $rb = lb+1$. It calls **'sptarn'** to compute generalized eigenvalues. Complex eigenvalues are excluded and real eigenvalues are stored. Then the interval is shifted to right by one unit and **'sptarn'** is called again until 'noe' eigenvalues are found.

Here the subroutine **'test'** (function mark=test(D)) is used to determine weather to do the loop or not,

Description: determine weather to do the loop or not

Input: eigenvalues 'D'
Output: if no real eigenvalue in this interval, return 1, else return 0