Problem Set #9

In this assignment, you are to develop a MATLAB code to implement the CT/LN vector basis functions on (straight-sided) triangular cells to find the resonant wavenumbers for a cavity resonator with perfect electric walls (like Problem Sets 4 and 8). In this assignment, however, you are to solve the vector Helmholtz equation

$$\nabla \times \left(\nabla \times \bar{H} \right) = k_0^2 \, \bar{H}$$

for the TM-to-z polarization ($E_z, \bar{H}_t$) subject to the boundary condition

$$\hat{n} \times \left(\nabla \times \bar{H} \right) = 0$$

You may adapt the code from problem set 4 (or 8) or create a new one from scratch. Mesh generator codes are provided for rectangular and coaxial regions. These produce a mesh file ('cylfil.txt') containing the following data:

```
First line:                 nnodes,ncells,nedges,ninnernodes');
Lines 2 – nnodes+1:         x,y coordinates');
The next 'ncells' lines:    pceton(1:ncells,1:3)');
The next 'ncells' lines:    pcetoe(1:ncells,1:3)');
The next 'ncells' lines:    er(1:ncells)');
The next 'nedges' lines:    pedtoc(1:nedges,1:2)');
The next 'nedges' lines:    pedton(1:nedges,1:2)');
```

where 'nnodes' 'ncells' and 'nedges' are the number of nodes, cells, and edges in the mesh, and 'ninnernodes' is the number of interior nodes (which you probably don't need, since you will be working with the *H*-field as the primary unknowns and do not need to use Dirichlet boundary conditions). The pointer array 'pceton' is the conventional cell-to-node pointer, while 'pcetoe' is the cell-to-edge pointer. Pointers 'pedtoc' and 'pedton' point from edge indices to cell and node indices, respectively.

Note: You may not need to use all these arrays! But they are there if you need them.

The mesh generators produce triangular cells with the local edges numbered so that the edge opposite node 1 is edge 1, and so on.

The major differences between this code and those used in Problem Sets 4 and 8 is that here the unknowns "live" on mesh edges, and you will need to use the 'pcetoe' pointer to fill the global systems. You will also need some global convention for orienting the basis functions on a particular edge, such as from smaller node index to larger node index. Expressions for the element matrix entries may be found in equations (9.184)–(9.187), but feel free to derive your own matrix entries or even use quadrature to compute them.

Turn in:

(1) A complete program listing.

(2) A table presenting results for the 4 lowest TM resonant wavenumbers for a coaxial cavity with PEC walls with inner radius $a = 1.0$ and outer radius $b = 4.0$. One of these results should be for a coaxial mesh with 5 layers of uniform thickness and 12 nodes/cells on the inner conductor. Provide at least 3 different sets of results, and specify the details of the mesh used for each case.

(3) An analysis of the convergence rate of the resonant wavenumbers obtained by this approach.

(4) A description of how the number of zero eigenvalues obtained is related to the number of nodes in the mesh. (The zero eigenvalues should be below 1.0e-06 in magnitude and should be easy to distinguish from the desired results.)

(5) Some discussion on the relative accuracy of this approach compared to those of Problem Set #4 and Problem Set #8.

As an aid to debugging your code, the following output data was obtained for a rectangular cavity of dimension 1.0 by 0.5 divided into a mesh of 10 cells by 5 cells by 'rectmeshvec.m'[1]. The mesh has 66 corner nodes, 100 cells, and 165 edges.

Output:

(65 zero eigenvalues)
66 7.0226640619736
67 8.9322492206326
68 11.375543006254
69 12.75008506974
70 14.011979890708
71 14.024976591894
72 15.883771000977
… (other wavenumbers up to number 165 = 60.0)

Hint: For rectangular cavities, the exact wavenumbers are: $k_{mn} = \sqrt{\left(\frac{m\pi}{a}\right)^2 + \left(\frac{n\pi}{b}\right)^2}$,

where m and n are nonzero for the TM case.

[1] Note: Program 'rectmeshvec.m' actually produces nodes for 6-node triangles instead of 3-node triangles, and includes the mid-side nodes in the node list. These nodes are not used in the pointer arrays and you may ignored their presence. However the number of nodes and additional (x,y) coordinates listed in the mesh file may confuse you so don't rely on that number to determine the actual number of "corner" nodes. Program 'coaxmeshvec.m' does not have this problem!