```
function cavityTMCTLN
% compute propagation constants for waveguide cross section modeled
 with
% triangular cells
%
% September 23, 2018 A. F. Peterson
% Modified November 28, 2018 by Caitlyn Caggia

global xy pceton pcetoe er pedtoc pedton icell;

nnodes = dlmread('cylfil.txt','', [0,0,0,0]);
ncells = dlmread('cylfil.txt','', [0,1,0,1]);
nedges = dlmread('cylfil.txt','', [0,2,0,2]);
% ninner = dlmread('cylfil.txt','', [0,3,0,3]);

xy=dlmread('cylfil.txt','', [1,1,nnodes,2]);

nstart=nnodes + 1;
nend=nstart + ncells - 1;
pceton=dlmread('cylfil.txt','', [nstart,1,nend,3]);

nstart=nend + 1;
nend=nstart + ncells - 1;
pcetoe=dlmread('cylfil.txt','', [nstart,1,nend,3]);

nstart=nend + 1;
nend=nstart + ncells - 1;
er=dlmread('cylfil.txt','', [nstart,1,nend,1]);

nstart=nend + 1;
nend=nstart + nedges - 1;
pedtoc=dlmread('cylfil.txt','', [nstart,1,nend,2]);

nstart=nend + 1;
nend=nstart + nedges - 1;
pedton=dlmread('cylfil.txt','', [nstart,1,nend,2]);

% initialize variables
nunks = nedges;
W=zeros(nunks);
Y=zeros(nunks);

% loop through the cells, filling global matrix one cell at a time

for icell=1:ncells

    %    compute 3 by 3 element matrix for cell 'icell'
    [eleS, eleT] = elemat(icell);

    %    check direction (points from lower global to higher)
    n1=pceton(icell,1); n2=pceton(icell,2); n3=pceton(icell,3);
    if n2>n3
```

```
        eleS(1,:) = -1*eleS(1,:);
        eleS(:,1) = -1*eleS(:,1);
        eleT(1,:) = -1*eleT(1,:);
        eleT(:,1) = -1*eleT(:,1);
    end
    if n3>n1
        eleS(2,:) = -1*eleS(2,:);
        eleS(:,2) = -1*eleS(:,2);
        eleT(2,:) = -1*eleT(2,:);
        eleT(:,2) = -1*eleT(:,2);
    end
    if n1>n2
        eleS(3,:) = -1*eleS(3,:);
        eleS(:,3) = -1*eleS(:,3);
        eleT(3,:) = -1*eleT(3,:);
        eleT(:,3) = -1*eleT(:,3);
    end

    %    add contributions from cell 'icell' to global matrices
    for ii=1:3
        for jj=1:3
            ig=pcetoe(icell,ii); % 'ig' is the global edge for 'ii'
            jg=pcetoe(icell,jj);  % 'jg' is the global edge for 'jj'
            W(ig,jg) = W(ig,jg) + er(icell)*eleS(ii,jj);
            Y(ig,jg) = Y(ig,jg) + er(icell)*eleT(ii,jj);
        end
    end
end

%  write results to file 'eigfil.txt'
fid = fopen('eigfil.txt', 'wt');
E = sort(eig(W,Y)); % use [V,E] = eig(W,Y) to get eigenvectors as well

str = 'TM resonant wavenumbers: ';
fprintf(fid,'%s \n',str);
for ii=1:nunks
    reaE=real(sqrt(E(ii)));
    imaE=imag(sqrt(E(ii)));
    fprintf(fid,'%6d %15.14g %15.14g\n',ii, reaE,imaE);
end
end

%
 ----------------------------------------------------------------------

function [eleS,eleT] = elemat(icell)
%  elemat: construct the element matrix for the contributions of
%          basis and testing functions of the form
%               S = grad Bm dot grad Bn
%                           and
%               T = Bm times Bn
%          over a triangular cell

global pceton xy;
```

```
eleS(3,3)=0;  eleT(3,3)=0;
n1=pceton(icell,1);
n2=pceton(icell,2);
n3=pceton(icell,3);

x(1)=xy(n1,1);  y(1)=xy(n1,2);
x(2)=xy(n2,1);  y(2)=xy(n2,2);
x(3)=xy(n3,1);  y(3)=xy(n3,2);

b(1)=y(2)-y(3);
b(2)=y(3)-y(1);
b(3)=y(1)-y(2);

c(1)=x(3)-x(2);
c(2)=x(1)-x(3);
c(3)=x(2)-x(1);

w(1) = sqrt((x(3)-x(2))^2 + (y(3)-y(2))^2);
w(2) = sqrt((x(3)-x(1))^2 + (y(3)-y(1))^2);
w(3) = sqrt((x(2)-x(1))^2 + (y(2)-y(1))^2);

Area = abs(b(3)*c(1) - b(1)*c(3))*0.5;

for m=1:3
    for n=1:3

        m1 = circlemath(m+1); m2 = circlemath(m+2);
        n1 = circlemath(n+1); n2 = circlemath(n+2);

        eleS(m,n) = (w(m)*w(n))/(4*Area^3)*(b(m1)*c(m2) -
 b(m2)*c(m1))...
            *(b(n1)*c(n2) - b(n2)*c(n1));

        addsum = 0;
        for i=1:2
            for j = 1:2

                beta = 1/24;
                if circlemath(m+i) == circlemath(n+j), beta =
 1/12; end

                alpha = -1;
                if i == j, alpha = 1; end

                m3i = circlemath(m+3-i); n3j = circlemath(n+3-j);
                addsum = addsum +
 alpha*beta*(b(m3i)*b(n3j)+c(m3i)*c(n3j));

            end
        end

        eleT(m,n) = w(m)*w(n)/(2*Area)*addsum;
```

```
    end
end

end

function [newindex] = circlemath(index)

newindex = mod((index-1), 3) + 1;

end
```

Published with MATLAB® R2018a