Table of Contents


```
clear; clc;tic
%  ---- Here we declare the namelist /INPUT/ that defines the search
 parameters.
%
%       It is useful to remember that
%
%
%          point(1)  defines the REAL value of the lower left hand co-
ordinate
%                    of the rectangle
%
%          point(2)  defines the IMAGINARY value of the lower left
 hand co-ordinate
%                    of the rectangle
%
%          step(1)   defines the length of the rectangle on the REAL
 axis.
%
%          step(2)   defines the length of the  rectangle on the
 IMAGINARY axis
%
%          droot_converged  We declare that a root has been found
 whenever
%                           ABS(f(z)) < droot_converged
%
%          npts      Number of points in per side of rectangle in the
 fixed
%                    integration quadrature
%
%
```

PEC Example

f = 1e12; omega = 2*pi*f; lambda = 3e8/f; % Material Properties ep1 = 12; ep2 = 4 ; ep3 = 2.1; ep4 = 1;

% EM constants mu0 = 4*pi*1e-7; ep0 = 8.854e-12;

% Propagations Constants k1 = omega*sqrt(mu0*ep0*ep1); k2 = omega*sqrt(mu0*ep0*ep2); k3 = omega*sqrt(mu0*ep0*ep3); k4 = omega*sqrt(mu0*ep0*ep4);

Open Dielectric

```
f = 1e12;
omega = 2*pi*f;
lambda = 3e8/f;
% Material Properties
ep1 = 1; % Air
ep2 = 9.7 ; % GaN/AlGaN layers combined
ep3 = 11; % Silicon base


% EM constants
mu0 = 4*pi*1e-7;
ep0 = 8.854e-12;

% Propagations Constants
k1 = omega*sqrt(mu0*ep0*ep1);
k2 = omega*sqrt(mu0*ep0*ep2);
k3 = omega*sqrt(mu0*ep0*ep3);

% point = -.20 -.20i;
% step = .40 + .40i;

% % For Dellnitz function with A B C T
% point = .95*k_air -.05*k_air*1i;
% % point = 0;
% step = .25*k_air + .12*k_air*1i;
```

Test Case 1

point = -2.2 - 1i*3.5; step = 2.5 + 1i*8;

Gold-film example

point = 1.9 - 1i*1.5; step = 0.4 + 1i*.20;

Wilkinson Polynomial

point = 5.5 - 1i*.5; step = 1.0 + 1i*1;

TLGF

```
point = 1.2*k1 - 1.2i*k1;
step = 1.04*k1 + 2.04i*k1;
```

```
% point = -20.3 -20.7i;
% step = 40.6 + 41.4i;

% Import from Namelist
% point  = -2.2 - 3.5*1i;
% step   = 5.0 + 8.0 *1i;

% point = -5 -5i;
% step = 10 + 10i;

%----------------------------------------------------------------------
```

Code paramters

```
%----------------------------------------------------------------------

% Convergence Criterion
droot_converged = 1e-14;

% Number of points on each side of the contour
% Increase it to ensure capturing all Riemann sheets
npts = 32768*2;

% Currently not used as splitting algorithm not implemented
maxboxes = 500;

% Limit the number of roots per box
% If this is greater, split
max_roots_per_box = 5;

% Maxium roots to be found by the routine
maxroots = 500;



%----------------------------------------------------------------------
```

ROOT COUNTING and CONTOUR INTEGRALS

```
%----------------------------------------------------------------------
% Count the number of roots within the rectangle now by examination
% of the change in the argument and also compute the integrals "s",
% defined in the paper, need to construct the matrix eigenvalue
% problem.
%
% The contour integral is computed through MATLAB's integral function
% using way-points to define the path of integration

[nroots, s] = countz (point, step, npts, maxroots);


%----------------------------------------------------------------------
```

BUILD HANKEL MATRICES

```
%---------------------------------------------------------------------
%
% Construct H and H1 from the computed contour integrals s_n
%

for k = 1 : nroots
    for l = 1 : nroots
        H1(k,l) = s(k + l);
        H(k,l) = s(k + l - 1 );
    end
end

%---------------------------------------------------------------------
```

SOLVE EIGENVALUE PROBLEM

```
%---------------------------------------------------------------------
% The eigenvalue problem looks like
%
% H - \lambda \time H1 = 0
%
% The eigenvalues, \lambda are the initial roots of the system
%
zinitial_roots = eig(H1,H,'qz');
%
% Check the quality of initial roots
%
zinitial_func = FZ(zinitial_roots);
%
```

USE NEWTON / HALLEY'S METHOD TO REFINE THE ROOTS

```
qpl = point;
%
qpt = point + step;
%
% Check if we have a converged solution at this stage
% for each root. If not, apply Newton's / Halley's method
%
for k = 1 : nroots
    %
    % If initially obtained roots are good enough
    %
    if(abs(zinitial_func(k)) < droot_converged)

        zfinal_roots(k) = zinitial_roots(k);
        zfinal_func(k)  = zinitial_func(k);
        %
```

```
        % Otherwise call Halley's method
        %
    else

        zfinal_roots(k) = newtzero(@FZ, zinitial_roots(k));
        %
        % Compute the value of the function at the converged root.
        %
        zfinal_func(k) = FZ(zfinal_roots(k));
    end
end
zfinal_func = zfinal_func';
zfinal_roots = zfinal_roots';
toc
```

```
Elapsed time is 0.200873 seconds.
```

Published with MATLAB® R2016a