Supporting Information

Stacked Al/Ag Anode for Short Circuit Protection in ITO Free Top-Emitting Organic Light-Emitting Diodes

Min Qian,[abc] Xiao-Bo Shi,[a] Jie Ma,[a] Jian Liang,[a] Yuan Liu,[a] Zhao-Kui Wang,[a]* and Liang-Sheng Liao[a]*

[a]Jiangsu Key Laboratory for Carbon-Based Functional Materials & Devices, Institute of Functional Nano & Soft Materials (FUNSOM), Soochow University, Suzhou, Jiangsu 215123, China

[b]Microelectronics Department, Soochow University, Suzhou, Jiangsu 215006, China

[c]WENZHENG College, Soochow University, Suzhou, Jiangsu 215104, China

Address all correspondence to the authors. Email: lsliao@suda.edu.cn; zkwang@suda.edu.cn

MATLAB Calculation Program for The Reflectance of Multi-Layer Optical Film Based on Transfer Matrix Method

```
clear all;
hold on;
n0= 1.7      %2.08%; %
n_substrate=1.52; %1.52;
%d_metal=100;
n_cappinglayer=1.89;%
%d_cappinglayer=60;
nn=0;
%d_metal2=20;

  d_metal2=44;                % anode:glass/Al/Ag/org     cathode: org/Ag(M1)/Al(M2)/capping/Ag(M3)/substrate
  d_metal3=0;
y=56;
for d_metal1=y:1:y
      x=0;
for d_cappinglayer=x:5:x %+250
%for lamda=624:10:624
%for lamda=528:10:528
for lamda=380:5:780
%for theta_out=0:5:60
for theta_out=0:1:0

       nn=nn+1;
          % n=[0.04-i*1.93 1.7 0.05-i*2.87];
        %n_metal=n(1)+(nn-1)*((0.035-i*2.69)/300) ;       %modify the     refractive index of Ag
     epsilon_metal1=LD(lamda*1e-9,'Ag','LD');
     %epsilon_metal1=nk_Sm(lamda);

       n_metal1=sqrt(epsilon_metal1);
       n_metal1=n_metal1';

       n_metal_real(nn)=real(n_metal1);
       n_metal_imag(nn)=imag(n_metal1);


       theta=theta_out*3.1415926/180;     % emerging angle = incidence angle in incdidence dielectric
```

```
COStheta0=sqrt(1-(sin(theta)*n_substrate/n0)^2)    %%%%incidence angle
theta0=acos(COStheta0);                                      %

COStheta1=sqrt(1-(sin(theta0)*n0/n_metal1)^2)    %%%%

phase_metal1=(2*pi*n_metal1*d_metal1)/lamda*COStheta1;

M_metal1=[cos(phase_metal1) i*sin(phase_metal1)/n_metal1/COStheta1;
      i*n_metal1*COStheta1*sin(phase_metal1) cos(phase_metal1)];

%=============================
epsilon_metal2=LD(lamda*1e-9,'Al','LD');
n_metal2=sqrt(epsilon_metal2);
n_metal2=n_metal2';
%d_metal2=y-d_metal1;
%d_metal2=0;



COStheta2=sqrt(1-(sin(theta)*n0/n_metal2)^2)    %%%%

phase_metal2=(2*pi*n_metal2*d_metal2)/lamda*COStheta2;
M_metal2=[cos(phase_metal2) i*sin(phase_metal2)/n_metal2/COStheta2;
i*n_metal2*COStheta2*sin(phase_metal2) cos(phase_metal2)];
  %=====================================


COStheta3=sqrt(1-(sin(theta)*n0/n_cappinglayer)^2)    %%%%

phase_cappinglayer=(2*pi*n_cappinglayer*d_cappinglayer)/lamda*COStheta3;
M_cappinglayer=[cos(phase_cappinglayer) i*sin(phase_cappinglayer)/n_cappinglayer/COStheta3;
      i*n_cappinglayer*COStheta3*sin(phase_cappinglayer) cos(phase_cappinglayer)];

%===========================================================
epsilon_metal3=LD(lamda*1e-9,'Ag','LD');
n_metal3=sqrt(epsilon_metal3);
n_metal3=n_metal3';
%d_metal3=15;

COStheta4=sqrt(1-(sin(theta)*n0/n_metal3)^2)    %%%%

phase_metal3=(2*pi*n_metal3*d_metal3)/lamda*COStheta4;
M_metal3=[cos(phase_metal3) i*sin(phase_metal3)/n_metal3/COStheta4;
i*n_metal1*COStheta4*sin(phase_metal3) cos(phase_metal3)];
```

```
        M=(M_metal1*M_metal2*M_cappinglayer)*M_metal3*[1;n_substrate];
        %M=(M_metal1*M_metal2*M_cappinglayer)^3*[1;n_substrate];


        B=M(1,1);
        C=M(2,1);
        temp=(n0*B-C)/(n0*B+C);

        R(nn)=(temp*temp');
        phase(nn)=angle(temp);
      % phase_nd(nn)=2*pi*2*

        temp1=n0*B+C;
        T(nn)=4*n0*n_substrate/(temp1*temp1');

        A(nn)=1-R(nn)-T(nn);

        theta_array(nn)=theta_out;

end
        lamda_array(nn)=lamda;
end
        d_cappinglayer_arry(nn)=d_cappinglayer;
end
        d_metal_array(nn)=d_metal1;
end

%plot(d_cappinglayer_arry,T);
hold on
%plot(lamda_array,T,'g');
%plot(lamda_array,A,'b');
%figure;
%plot(lamda_array,phase,'b');
plot(lamda_array,R,'r');
%plot(theta_array,R,'r');
%plot(theta_array,phase,'b');

%plot(d_metal_array,T); %,'r');
%
%figure;
%plot(lamda_array,n_metal_real);
```

```
%figure;
%plot(lamda_array,n_metal_imag);
zoom on;
%a=lamda_array';
%b=R';
%data=[a,b];
%command1=['save dbralqag150.txt data -ascii'];
%eval(command1);
```

MATLAB Calculation Program for

The LD Model of Dielectric Function for Various Metals

```
function varargout = LD(lambda,material,model)


% LD : Lorentz-Drude and Drude model for the dielectric constant of metals
%
%***********************************************************************
%

%***********************************************************************
%    DESCRIPTION:
%    This function computes the complex dielectric constant (i.e. relative
%    permittivity) of various metals using either the Lorentz-Drude (LD) or
%    the Drude model (D). Please note that the LD model provides a better
%    fit with the exact values, therefore it is the default choice.
%
%***********************************************************************
%
%    USAGE: epsilon = LD(lambda,material,model)
%
%         OR: [epsilon_Re epsilon_Im] = LD(lambda,material,model)
%
%         OR: [epsilon_Re epsilon_Im N] = LD(lambda,material,model)
```

```
%
%
%     WHERE: "epsilon_Re" and "epsilon_Im" are respectively the real and
%                imaginary parts of the dielectric constant "epsilon", and "N"
%                is the complex refractive index.
%
%
%     INPUT PARAMETERS:
%
%           lambda      ==> wavelength (meters) of light excitation on material.
%                                 Accepts either vector or matrix inputs.
%
%           material ==>        'Ag' = silver
%                                   'Al' = aluminum
%                                   'Au' = gold
%                                   'Cu' = copper
%                                   'Cr' = chromium
%                                   'Ni' = nickel
%                                   'W'    = tungsten
%                                   'Ti' = titanium
%                                   'Be' = beryllium
%                                   'Pd' = palladium
%                                   'Pt' = platinum
%
%           model        ==> Choose 'LD' or 'D' for Lorentz-Drude or Drude model.
%
%     REFERENCES:
%
%     [1] Rakic et al., Optical properties of metallic films for vertical-
%           cavity optoelectronic devices, Applied Optics (1998)
%     [2] E. Palik, Handbook of Optical Constants of Solids,
%           Academic Press (1997)
%
%*******************************************************************


if nargin < 3, model = 'LD'; end    % Lorentz-Drude model by default
if nargin < 2, return; end



%*******************************************************************
% Physical constants
%*******************************************************************
twopic = 1.883651567308853e+09; % twopic=2*pi*c where c is speed of light
```

```
omegalight = twopic*(lambda.^(-1)); % angular frequency of light (rad/s)
invsqrt2 = 0.707106781186547;    % 1/sqrt(2)
ehbar = 1.519250349719305e+15; % e/hbar where hbar=h/(2*pi) and e=1.6e-19


%**********************************************************************
% Lorentz-Drude model parameters for dispersive medium [1]
%**********************************************************************
% N.B. Gamma and omega values are in eV, while f is adimensional.ÎÞÁ¿¸Ù


switch material
    case 'Ag'
        % Plasma frequency
        omegap = 9.01*ehbar;
        % Oscillators' strenght
        f =         [0.845 0.065 0.124 0.011 0.840 5.646];
        % Damping frequency of each oscillator
        Gamma = [0.048 3.886 0.452 0.065 0.916 2.419]*ehbar;
        % Resonant frequency of each oscillator
        omega = [0.000 0.816 4.481 8.185 9.083 20.29]*ehbar;
        % Number of resonances
        order = length(omega);


    case 'Al'
        omegap = 14.98*ehbar;
        f =         [0.523 0.227 0.050 0.166 0.030];
        Gamma = [0.047 0.333 0.312 1.351 3.382]*ehbar;
        omega = [0.000 0.162 1.544 1.808 3.473]*ehbar;
        order = length(omega);


    case 'Au'
        omegap = 9.03*ehbar;
        f =         [0.760 0.024 0.010 0.071 0.601 4.384];
        Gamma = [0.053 0.241 0.345 0.870 2.494 2.214]*ehbar;
        omega = [0.000 0.415 0.830 2.969 4.304 13.32]*ehbar;
        order = length(omega);


    case 'Cu'
        omegap = 10.83*ehbar;
        f =         [0.575 0.061 0.104 0.723 0.638];
        Gamma = [0.030 0.378 1.056 3.213 4.305]*ehbar;
        omega = [0.000 0.291 2.957 5.300 11.18]*ehbar;
        order = length(omega);


    case 'Cr'
        omegap = 10.75*ehbar;
        f =       [0.168 0.151 0.150 1.149 0.825];
        Gamma = [0.047 3.175 1.305 2.676 1.335]*ehbar;
        omega = [0.000 0.121 0.543 1.970 8.775]*ehbar;
        order = length(omega);


    case 'Ni'
        omegap = 15.92*ehbar;
        f =       [0.096 0.100 0.135 0.106 0.729];
        Gamma = [0.048 4.511 1.334 2.178 6.292]*ehbar;
        omega = [0.000 0.174 0.582 1.597 6.089]*ehbar;
        order = length(omega);


    case 'W'
        omegap = 13.22*ehbar;
        f =       [0.206 0.054 0.166 0.706 2.590];
        Gamma = [0.064 0.530 1.281 3.332 5.836]*ehbar;
        omega = [0.000 1.004 1.917 3.580 7.498]*ehbar;
        order = length(omega);


    case 'Ti'
        omegap = 7.29*ehbar;
        f =       [0.148 0.899 0.393 0.187 0.001];
        Gamma = [0.082 2.276 2.518 1.663 1.762]*ehbar;
        omega = [0.000 0.777 1.545 2.509 1.943]*ehbar;
        order = length(omega);


    case 'Be'
        omegap = 18.51*ehbar;
        f =       [0.084 0.031 0.140 0.530 0.130];
        Gamma = [0.035 1.664 3.395 4.454 1.802]*ehbar;
        omega = [0.000 0.100 1.032 3.183 4.604]*ehbar;
        order = length(omega);
    case 'Pd'
        omegap = 9.72*ehbar;
        f =       [0.330 0.649 0.121 0.638 0.453];
        Gamma = [0.008 2.950 0.555 4.621 3.236]*ehbar;
        omega = [0.000 0.336 0.501 1.659 5.715]*ehbar;
        order = length(omega);


    case 'Pt'
        omegap = 9.59*ehbar;
        f =       [0.333 0.191 0.659 0.547 3.576];
        Gamma = [0.080 0.517 1.838 3.668 8.517]*ehbar;
        omega = [0.000 0.780 1.314 3.141 9.249]*ehbar;
        order = length(omega);


    otherwise
        error('ERROR! Not a valid choice of material in input argument.')
end


%*************************************************************************
% Drude model (intraband effects)
%*************************************************************************


epsilon_D = ones(size(lambda)) - ((f(1)*omegap^2) *(omegalight.^2 + i*Gamma(1)*omegalight).^(-1));


%*************************************************************************
% Lorentz model (interband effects)
%*************************************************************************


switch model
    case 'D'     % Drude model
        epsilon = epsilon_D;

    case 'LD'    % Lorentz-Drude model
        epsilon_L = zeros(size(lambda));
        % Lorentzian contributions
        for k = 2:order
            epsilon_L = epsilon_L + (f(k)*omegap^2)*(((omega(k)^2)*ones(size(lambda)) - omegalight.^2) -
i*Gamma(k)*omegalight).^(-1);
        end

        % Drude and Lorentz contributions combined
        epsilon = epsilon_D + epsilon_L;

    otherwise
        error('ERROR! Invalid option. Choose "LD" or "D"')
end


%**********************************************************************
% Output variables
%**********************************************************************


switch nargout
    case 1 % one output variable assigned
        varargout{1} = epsilon;

    case 2 % two output variables assigned

        % Real part of dielectric constant
        varargout{1} = real(epsilon);

        % Imaginary part of dielectric constant
        varargout{2} = imag(epsilon);

    case 3 % three output variables assigned

        % Real part of dielectric constant
        varargout{1} = real(epsilon);

        % Imaginary part of dielectric constant
        varargout{2} = imag(epsilon);

        % Complex refractive index [2]: N = n + i*k
        varargout{3}   =   invsqrt2*(sqrt(sqrt((varargout{1}).^2   +   (varargout{2}).^2)   +   varargout{1})   +
i*sqrt(sqrt((varargout{1}).^2 +...
            (varargout{2}).^2) - varargout{1}));

    otherwise
    error('Invalid number of output variables; 1,2 or 3 output variables.')
end
```