Apéndice A Código en Matlab para simulación de viga en Cantilever.

```
% =============== Opciones de ingreso por usuario ==================
% [Sim=1][Data=2][Both=3]
outputMode=1;
%*************** Importacion de datos experimentales***************
if (outputMode~=1)
    load cp13ct167vac_2.csv;
    w_data=cp13ct167vac_2(50:2000,1);
    amp_data= cp13ct167vac_2(50:2000,2);
    phase_data_scaled = phase_data - (max(phase_data)-.05);
    % Normalize Amplitude to 1, Shift phase to 0 degrees
    % (offsetted slightly to account for nosie)
    amp_data_scaled = amp_data/(max(amp_data)-.00015);
    phase_data_scaled = phase_data - (max(phase_data)-.05);
end;
 % ************************* Simulación **************************
if (outputMode~=2)
% ============= Nombre de Variables=============
    % List of Variables:
    % Vrf = Drive Voltage
    % k = Spring constant
    % w0 = (Half) resonant frequency
    % R = Tip radius of probe
    % z0 = Separation distance of cantilevers
    % H = Probe tip height
    % c_angle = Half cone anlge
    % Q = Quality factor
    % drive_w1 = (Constant) modulating frequency

% -------- Potencia de Entrada ------------
```

```
    Vrf = 2;

% ------ Distancia de Separación-----------
    z0 = 3.8e-6;

% ------- Propiedades de la prueba cantilever---------

    k1 = .06;

    w0_C1 = 2*pi*(2*6.375e3);

    H = 15e-6;

    R = 30e-9;

    c_angle = (pi/180)*8;

    Q1 = 4000;

    drive_w1 = w0_C1+(2*pi*10);

    % Initial Values for probe amplitudes and phase

    z1 = 10e-9;

    z1_phi = -(pi/180)*90;

    % ----- Propiedades de la Viga----------

   k2 = 3.33 ;

   w0_C2 = 2*pi*(2*83.9175)e3;

   Q2 = 6200;

   %Initial Values for amplitude and phase

   z2 = 1e-9;

   z2_phi = 0;

   % ----- Error inicial---------

   z1_err = 1;

   z2_err = 1;



   % =================Principal ======================

   span = 800;

   start_freq = (2*83.715e3);

   num_steps = 500;
```

```
for i = 1:1:num_steps

    j = 0;

    w(i) = 2*pi*(start_freq+(i*span/num_steps));


    while(z1_err>1e-15 & z2_err > 1e-15)

        % calc dfdz

        phase_shift=0;

        F_probe = -find_F(R,z0,H,c_angle,z2,z2_phi,Vrf);

        dfdz_probe = -find_dfdz(R,z0,H,c_angle,Vrf,z2,z2_phi);

        F_test = -find_F(R,z0,H,c_angle,z1,z1_phi,Vrf);

        dfdz_test=-find_dfdz(R,z0,H,c_angle,Vrf,z1,z1_phi);


        % calc shifted resonant frequencies

        w0_C1_eff = find_new_w0(w0,C1,dfdz_probe,k1);

        w0_C2_eff = find_new_w0(w0_C2,dfdz_test,k2);

        %calc new z1,z2

        z1_new = (F_probe/k1-dfdz_probe)...

            *find_Az(drive_w1,w0_C1,w0_C1_eff,Q1);

        z2_new = (F_test/(k2-dfdz_test))...

            *find_Az(w(i),w0_c2,w0_C2_eff,Q2);



        %calc phase of z1,z2

        z1_phi = find_Phi(drive_w1,w0_C1,w0_C1_eff,Q1);

        z2_phi = find_Phi(w(i),wo_C2,w0_C2_eff,Q2);

        % calc error

        z1_err = abs(z1_new-z1);

        z2_err = abs(z2_new-z2);

        % update new z1,z2 values

        z1 = abs(z1_new);

        z2 = abs(z2_new);
```

```
            %if solution values don´t converge,
            % break and take last value


            j = j + 1;

            if j>30

                e =1;

                break

            end

        end;

        % Data collection

        Amp_z1(i) = z1;

        Amp_z2(i) = z2;

        Phase_z1(i)=  z1_phi;

        Phase_z2(i) = z2_phi;

        Resonance_C1(i)= w0_C1_eff;

        Force(i)= F_probe;


        %Reset error tolerances

        z1_err = 100;

        z2_err = 100;

    end

end;

% ================== Grafica de simulación (1) =======================

if outputMode ==1

    %Plot amplitude and phase of probe and test cantilever

 figure

    subplot(2,1,1);

    plot(w/pi/2000,(Amp_z1));

    title('Amplitude');

    legend ('Probe [m]');
```

```
    subplot(2,1,2);
    plot(w/pi/2000,(Amp_z2));
    legend('Test [m]');
    xlabel('Frequency [kHz]');

    figure
    subplot(2,1,1);
    plot(w/pi/2000,(180/pi)*(Phase_z1));
    title('Phase');
    legend('Probe [degrees]');
    subplot(2,1,2);
    plot(w/pi/2000,(180/pi)*Phase_z2);
    legend ('Test [degrees]');
    xlabel('Frequency [kHz]');

    % Plot intermediate probe cantilever values
    figure
    subplot(2,2,1);
    plot(w/pi/2000,Force_Grad);
    title ('Gradient [N/m]');
    subplot(2,2,2);
    plot(w/pi/2000,(Resonance_C1-w0_C1)/2/pi);
    title('Resonance Shift [Hz]');
    subplot(2,2,4);
    plot(w/pi/2000,Forces);
    title('Force [N]');

end
% ================== Graficación de datos (2) =======================
if outputMode == 2
    % Plot experiment data only
    figure
```

```
    subplot (2,1,1);

    plot(w_data,amp_data);

    ylabel('Amplitude [V]');

    subplot(2,1,2);

    plot(w_data,phase_data);

    ylabel('Phase [Degrees]');

    xlabel('Frequency [kHz]');

end

% =============== Graficación de ambos modos (3) ====================

if outputMode ==3

    Amp_z1_scaled = Amp_z1/max(Amp_z1);

    Phase_z1_scaled = (180/pi)*((Phase_z1)-max((Phase_z1)));

    figure

    subplot (2,1,1);

    plot(2*w_data,amp_data_scaled,w/pi/2000,Amp_z1_scaled,'m');

    legend('Experimental Data','Simulated Data');

    ylabel('Normalized Amplitude');

    subplot(2,1,2);

    plot(2*w_data,phase_data_scaled,w/pi/2000,Phase_z1_scaled,'m');

    xlabel('Frequency [kHz]');

    ylabel ( 'Offsetted Phase');

end

% ********************************************************************
% ************* Simulador de Respuesta, Lista de Funciones***********
% ********************** Función: find_F ****************************
% ********************************************************************

%List of variables

% Vrf = Drive Voltage

% R = Tip Radius of probe

% z0 = Separation distance of Cantilevers
```

```
% H = Probe Tip Height

% theta = Half cone angle

% z = Current probe cantilever amplitude response

% z_phi = Current probe cantilever phase response


function F = find_F(R,z0,H,theta,z,z_phi,Vrf)

epsilon = 8.85e-12;

k = 1/(log(tan(theta/2)));

F = (pi*epsilon*Vrf^2*(1/4)*(R^2/(z0^2+z0*R)+k^2*(-log((z0+R)/H)-
1+(R/sin(theta))/(z0+R))))-(pi*epsilon*Vrf^2*(1/8)*(z)*(-
R^2*(2*z0+R)/(z0^2+z0*R)^2+k^2*(-1/(z0+R)-
(R/sin(theta))/(z0+R)^2))*cos(z_phi));

function dfdz = find_dfdz (R,z0,H.theta,Vrf,z,z_phi)

epsilon = 8.85e-12;

k = 1/(log(tan(theta/2)));

A = (pi*epsilon*Vrf^2*(-R^2*(z0^2+z0*R)/(z0^2+z0*R)^2+k^2*(-1/(z0+R)-
R/sin(theta)/(z0+R)^2)));

B = (pi*epsilon*Vrf^2*(-2*R^2*((z0^2+z0*R)^2-
(2*z0+R)^2/(z0^2+z0*R)^3+k^2*(1/(z0+R)^2+2*R/sin(theta)/(z0+R)^3)));

dfdz=(1/4)*A-(1/8)*B*z*cos(z_phi);

function Az = find_Az(w,w0,w0_new,Q);

Az= Q/((Q^2*(1-w^2/w0_new^2)^2+(w^2*w0^2/w0_new^4))^.5);

function Phi = find_Phi(w,w0,w0_new,Q)

Phi = -1*atan2((w*w0/Q/w0_new^2),(1-w^2/w0_new^2));

function new_w0 = find_new_w0(w0,dfdz,k);

new_w0=w0*(1-dfdz*(.5/k));

end
```