```
load impedenza.mat


minf=400;
maxf=800;
f_axis=f_axis(minf:maxf);
Z_disp=Z_disp(minf:maxf);
s11db=s11db(minf:maxf);
s11=s11(minf:maxf);
rf=find(imag(Z_disp)>0);
rf=rf(1);

% Xin=0.5*(max(imag(Z_disp))+min(imag(Z_disp)));
% piresin=find(imag(Z_disp)<Xin);
% pires=f_axis(piresin);
% plot(f_axis,real(1./Z_disp));
% ylabel('Re{Z} [Ohm]');
% title('Input impedance - real part');
% hold;
% plot(f_axis,imag(1./Z_disp),'r');
% ylabel('Im{Z} [Ohm]');
% title('Input impedance - imaginary part');
% axis tight;
% grid on;
% legend('real part','imaginary part');
% plot(f_axis(rf),imag(Z_disp(rf)),'o');
% plot(f_axis(rf),real(Z_disp(rf)),'ro');
%
% figure
% plot(f_axis,abs(s11));
%
%
% fr=  f_axis(rf);
% rin= real(Z_disp(rf));
% pifaris=pires(1);
% xin=Xin(1);
% rin=real(Z_disp(piresin(1)));
% [feffi,feffii]=min(s11);
% feffin=feffii(1);
% mismatcheff=1-abs(s11(feffin))^2;

% figure
% GenerateSmith;
% plot((s11(400:800)),'b');

% here we choose the desired matching frequency
fdes = 686e6;
rf=find(f_axis==fdes);
```

```
Rin=real(Z_disp(rf));
Xin=imag(Z_disp(rf));
Imp=Z_disp;

%% L Matching calculation
[zmat,zmat1,zmat2,zmat3,bigvet,LU,LUT]=adattamento(Rin,Xin,Imp,50,fdes);
Zin=zmat;
%figure
s=(Zin-50)./(Zin+50);
% plot(f_axis,(abs(s)),'r')
% ylabel('|s11|');
% title('S11 parameter');
% axis tight;
% grid on;
% legend('without adaptation','With adaptation')


%% Loss calculation
loss = 10.*log10(1./(1-(abs(s)).^2));
sL = 0.7;
switchLoss = sL*4*(1);
insertionLoss=0;
antennaloss = 0;
total_loss = loss + insertionLoss + switchLoss + antennaloss;

figure
limitaxe = [10 7 5];
plot(f_axis,total_loss);
hold on;
plot([474000000 698000000 858000000], limitaxe);
% hold on
legend('Loss with adaptation');
ylabel('L [dB]');
xlabel('Frequency in Mhz');
%title(strcat('S11 parameter using matching circuit at ',num2str(f0/1000000),' Mhz resonant
frequency' ));
axis tight;
grid on;
XLIM([400000000 800000000])
YLIM([0 10])

% figure
% swr=(1+abs(s))./(1-abs(s));
% plot(f_axis,swr)
% xlabel('Frequency [GHz]')
% ylabel('SWR')
% grid on
% figure
% GenerateSmith;
% plot((s),'b');
```

```
function[zmat,zmat1,zmat2,zmat3,bigvet,LU,LUT]=adattamento(Rin,Xin,Imp,Zo,f)

if Rin<Zo
   text = 'Rin<zo'
   B=sqrt((Zo-Rin)/Rin)/Zo;
   X=sqrt(Rin*(Zo-Rin))-Xin;
   B1=-sqrt((Zo-Rin)/Rin)/Zo;
   X1=-sqrt(Rin*(Zo-Rin))-Xin;
   B2=+sqrt((Zo-Rin)/Rin)/Zo;
   X2=-sqrt(Rin*(Zo-Rin))-Xin;
   B3=-sqrt((Zo-Rin)/Rin)/Zo;
   X3=+sqrt(Rin*(Zo-Rin))-Xin;
   bigvet=[B,X;B1,X1;B2,X2;B3,X3];
   bigvet=[1/B,X;1/B1,X1;1/B2,X2;1/B3,X3];

   zmat=1./(j*B+1./(real(Imp)+j*(imag(Imp)+X)));
   zmat1=1./(j*B1+1./(real(Imp)+j*(imag(Imp)+X1)));
   zmat2=1./(j*B2+1./(real(Imp)+j*(imag(Imp)+X2)));
   zmat3=1./(j*B3+1./(real(Imp)+j*(imag(Imp)+X3)));

   LU=[0,0;0,0];
   char LUT

   X12=bigvet;
   if(X12(1,1)>0)
      LU(1,1)=X12(1,1)/(2*pi*f);
      LUT(1,1)='L';
   else
      LU(1,1)=-1/(2*pi*f*X12(1,1));
      LUT(1,1)='C';
   end

   if(X12(1,2)>0)
      LU(1,2)=X12(1,2)/(2*pi*f)
      LUT(1,2)='L';
   else
      LU(1,2)=-1/(2*pi*f*X12(1,2))
      LUT(1,2)='C';
   end

   if(X12(2,1)>0)
      LU(2,1)=X12(2,1)/(2*pi*f);
      LUT(2,1)='L';
   else
      LU(2,1)=-1/(2*pi*f*X12(2,1));
      LUT(2,1)='C';
   end

   if(X12(2,2)>0)
```

```
        LU(2,2)=X12(2,2)/(2*pi*f)
        LUT(2,2)='L';
    else
        LU(2,2)=-1/(2*pi*f*X12(2,2))
        LUT(2,2)='C';
    end

    if(X12(3,1)>0)
        LU(3,1)=X12(3,1)/(2*pi*f)
        LUT(3,1)='L';
    else
        LU(3,1)=-1/(2*pi*f*X12(3,1))
        LUT(3,1)='C';
    end

    if(X12(3,2)>0)
        LU(3,2)=X12(3,2)/(2*pi*f)
        LUT(3,2)='L';
    else
        LU(3,2)=-1/(2*pi*f*X12(3,2))
        LUT(3,2)='C';
    end

    if(X12(4,1)>0)
        LU(4,1)=X12(4,1)/(2*pi*f)
        LUT(4,1)='L';
    else
        LU(4,1)=-1/(2*pi*f*X12(4,1))
        LUT(4,1)='C';
    end

    if(X12(4,2)>0)
        LU(4,2)=X12(4,2)/(2*pi*f)
        LUT(4,2)='L';
    else
        LU(4,2)=-1/(2*pi*f*X12(4,2))
        LUT(4,2)='C';
    end

else
     text = 'Rin>zo'
    B=(Xin+sqrt(Rin/Zo)*sqrt(Rin^2+Xin^2-Zo*Rin))/(Rin^2+Xin^2);
    X=1/B+Zo*Xin/Rin-Zo/(B*Rin);
    B1=(Xin-sqrt(Rin/Zo)*sqrt(Rin^2+Xin^2-Zo*Rin))/(Rin^2+Xin^2);
    X1=1/B1+Zo*Xin/Rin-Zo/(B1*Rin);
    zmat=j*X+1./(j*B+1./(Imp));
    zmat1=j*X1+1./(j*B1+1./(Imp));
    zmat2=0;
    zmat3=0;
    bigvet=[B,X;B1,X1];
```

```
    bigvet=[1/B,X;1/B1,X1]

    X12=bigvet;
    if(X12(1,1)>0)
       LU(1,1)=X12(1,1)/(2*pi*f);
       LUT(1,1)='L';
    else
       LU(1,1)=-1/(2*pi*f*X12(1,1));
       LUT(1,1)='C';
    end

    if(X12(1,2)>0)
       LU(1,2)=X12(1,2)/(2*pi*f)
       LUT(1,2)='L';
    else
       LU(1,2)=-1/(2*pi*f*X12(1,2))
       LUT(1,2)='C';
    end

    if(X12(2,1)>0)
       LU(2,1)=X12(2,1)/(2*pi*f);
       LUT(2,1)='L';
    else
       LU(2,1)=-1/(2*pi*f*X12(2,1));
       LUT(2,1)='C';
    end

    if(X12(2,2)>0)
       LU(2,2)=X12(2,2)/(2*pi*f)
       LUT(2,2)='L';
    else
       LU(2,2)=-1/(2*pi*f*X12(2,2))
       LUT(2,2)='C';
    end


end


end
```

```
function lesstimestep2(N,Nest,order)


load impedenza2.mat
nn = N; % N is the initial number of time steps (must be inferior to 90000)
Rs=50;  % Matching value: 50 ohm
V1= V(1:nn+1); % V1 is the truncated voltage time response
Vs1=Vs(1:nn+1);   % Vs is the source voltage (initial response)

%% Plot using marple approximation

V1m = V1;
%Filter the initial signal is possible
% t = (1:length(V1m)/10)*10*Dt;
% h = firls(42,[0 0.5 0.75 1],[1 1 0 0]);
%y = filter(h,1,V1m);
%V1m = y;

% Marple modelization
V1marple = polydata(ar(V1,order));

%Calculation of the signal for Nest+1 points.
for ii= N+2:Nest+1
   V1m(ii) =0;
   for jj=2:order+1
      V1m(ii) = V1m(ii) - V1marple(jj)*V1m(ii+1-jj)/V1marple(1);
   end
   V1m(ii) = V1m(ii);
end

% Plot the initial and the predicted signal
figure;
plot(1:length(V1),V1);
hold on;
plot(length(V1)+1:length(V1m),V1m(length(V1)+1:end),'r');
grid on;
xlim([N-(Nest-N) Nest+1]);
legend('simulated response',strcat('response approximated with order',num2str(order)));
title('V');


%% FFT and calculation of the impedance of the antenna using marple
Vs1marple_fft =fft(Vs1,length(V));
V1marple_fft = fft(V1m,length(V));
Zmarple= -Rs./( 1 + Vs1marple_fft./V1marple_fft);

%% Calculation of the reflection coef using marple
s11marple = (Zmarple - Rs)./(Zmarple + Rs);
s11marple = s11marple(1:length(s11marple)/2);
s11marpledb = 20.*log10(abs(s11marple));
```

```
%% Calculation of the impedance and S11 without marple to make comparison
Vs1(nn+2:length(Vs))=0;
V1(nn+2:length(V))=0;
Vs1_fft= fft(Vs1);
V1_fft=fft(V1);
w = 2*pi.*f_axis;
Z1= -Rs./( 1 + Vs1_fft./V1_fft );

Z_disp = Z1; %% local variable
Z_disp(1) = 0; %% impedance at zero frequency is usually nonsense
s11 = (Z_disp - Rs)./(Z_disp + Rs);
s11 = s11(1:length(s11)/2,1);
s11db = 20.*log10(abs(s11));


%% Plot the results
figure
plot(f_axis, s11db);hold on; plot(f_axis, s11marpledb,'r');
legend('without approximation','using marple');
grid on;
xlabel('frequency in Ghz');
```

```
function adapCheb3 (Z_disp,N,f1,f2,f_axis,Rb,Lr1,Lar)


%Coded by Radwan CHARAFEDDINE and Zena FOURZOLI
%  for AAU 10 semester master thesis

%this function is used in order to generate a chebyshev impedance matching
%circuit of order N using the impedance Z_disp. Rb is the load to match f1
%and f2 are the edge in Mhz of the bandpass. Lr1 is a parameter to adjust,
%it is link to capacitors' value. It should be of the order of 1. Lar is
%the ripple attenuation in dB. The code used here after follows the theory
%of microwave filter impedance matching - matching network and coupling
%structures. You can refer to it for more information.



%% Uncomment the following if you want to test the matching circuit with a
%% constant impedance of 20+30i

% Z_disp = ones(length(Z_disp),1).*(20+i*30);

f1 = f1*1000000;
f2= f2*1000000;
%% Calculate de Chebychev coef for N
%Basic parameter initialization

minf=10;
maxf=1800;
f_axis=f_axis(minf:maxf);
Z_disp=Z_disp(minf:maxf);

rf1=find(f_axis==f1);
w1 = 2*pi*f1;
w2= 2*pi*f2;
w0 = w1+w2 -sqrt(square(w2-w1)+w1*w2);
w=  abs(w0/w1-w0/w2);
wprime = abs((2 - w0./(2*pi.*f_axis) - 1./(2 - w0./(2*pi.*f_axis))) / (2 - w0/w2 - 1./(2 - w0/w2)));
f0=  (w0/(2*pi))/1000000;
f0 = round(f0)*1000000
rf=find(f_axis==f0);
Imp= Z_disp(rf);

rf2=find(f_axis==f2);
Imp1 = Z_disp(rf1);
Imp2 = Z_disp(rf2);



%% formulas from microwave filters
```

```
Ra =real(Imp);
% Decrement value for information
delta = Ra./(abs(imag(Z_disp(rf2))+ imag(Z_disp(rf1)))/2)



%% Calculation of the Lar
% Uncomment it to test but it doesn t work. It has to work with the Gn
% calculation method 1 that doesn t work for the moment.
% hh = 10^(ripple/10);
% dd= sinh(asinh(sqrt(1/(hh-1)))/N);
% ee = dd - 2*delta*sin(pi/(2*N));
% s11max = cosh(N*asinh(ee))/cosh(N*asinh(dd))
% Lar = 10*log10(1/(1-s11max^2))


%% Calculation of Gn coefficients: Method 1
% Uncomment it to test but it doesn t work

% D = (dd/(delta*sin(pi/2*N)))-1;
%
% g(1) = 1;
% g(2) = 1/(delta);
%
% teta = 2*pi/N;
% for r=1:N-1
%     num = ((sin(r*teta)^2) *(cos(r*teta)^2)) + (( cos(r*teta)^2 + (D^2)*(sin(r*teta))^2)*
(sin(teta)^2)* (delta^2));
%     k(r)=sqrt( num/(sin((2*r-1)*teta)*sin((2*r+1)*teta)));
% end
%
% for ii=3:N+1
%     g(ii)= 1/(g(ii-1)*k(ii-2)^2);
% end
% g(N+2) = 1/(D*delta*g(N+1));
% g


%% Calculation of Gn coefficients: Method 2
%
beta = log(coth(Lar/17.37));
lambda = sinh(beta/(2*N));
for ii=1:N
   a(ii)= sin(((2*ii-1)*pi)/(2*N));
   b(ii) = lambda^2 +(sin((ii*pi)/N)^2);
end

g(1)=1;
g(2)= 2*a(1)/lambda;
for ii=2:N
```

```
   g(ii+1)=(4*a(ii-1)*a(ii))/(b(ii-1)*g(ii));
end

if(mod(N,2)==0) % impaire
   g(N+2) =1;
else  % paire
   g(N+2) = (coth(beta/4))^2;
end
g


%% Conversion to pass band filter :method 1
% j = sqrt(-1);
% for ii=3:N+2
%      %% pair it is a capacitor => equivalent is capacitor in parallel with coil
%    if(mod(ii,2)==0)
%        Cp = g(ii)/(w.*w0)
%        zc = 1./(j.*Cp.*2*pi.*f_axis);
%        size(zc)
%        Lp = w./(w0.*g(ii))
%        zl = j.*2*pi.*f_axis.*Lp;
%        zm(ii,:) = 1./(1./zc + 1./zl );
%
%    else    % it is a coil => equivalent is capacitor in serie with coil
%         Cs = w./(w0.*g(ii))
%         Ls = g(ii)/(w.*w0)
%        zl = (j.*2*pi.*f_axis.*Ls);
%        zc = 1./(j.*2*pi.*f_axis.*Cs);
%        zm(ii,:) = zc + zl;
%    end
% end

%% Calculation of the total impedance for method 1
%
% zm = zm';
%
% Z= 1./ (1./Z_disp + 1./zm(:,1));
%
% for ii=2:N
%    if(mod(ii,2)==0) % impair -> parallel impedance
%    Z= 1./ (1./Z + 1./zm(:,ii));
%    else % pair -> serie impedance
%    Z = Z + zm(:,ii);
%    end
% end

%% Conversion to band pass filter : method 2
Lr = ones(1,N+2).*Lr1*1e-9;
Cr = zeros(1,N);
```

```
for ii=1:N
   Cr(ii)= 1/(Lr(ii+1)*w0^2);
end
K = zeros(1,N+1);
K(1) = sqrt((Ra*w0*Lr(2)*w)/(g(1)*g(2)));
for ii=2:N
   K(ii)= w*w0*sqrt(Lr(ii+1)*Lr(ii+2)/(g(ii+1)*g(ii+2)));
end
K(N+1) = sqrt((Rb*w0*Lr(N+1)*w)/(g(N+1)*g(N+2)));

M = zeros(1,N+1);
M(1) = K(1)*sqrt(1+(w0*Lr(1)/Ra)^2)/w0;
for ii =2:N
   M(ii) = K(ii)/w0;
end
M(N+1) = K(N+1)/w0*sqrt(1+(w0*Lr(N+2)/Rb)^2);

L = zeros(1,N+2);

L(1) = Lr(1)-M(1);
Lp(1) = Lr(1);
Me(1) = (M(1)+ ((Lr(1)-M(1))*w0^2*M(1)*Lp(1)/Ra^2))/(1+(w0*Lr(1)/Ra)^2);
Lp(2) = Lr(2) + M(1)-Me(1);

for ii=3:N
   Lp(ii) = Lr(ii);
end
Lp(N+2) = Lr(N+2);
Me(N) = (M(N+1)+ ((Lr(N+2)-
M(N+1))*w0^2*M(N+1)*Lp(N+2)/Rb^2))/(1+(w0*Lr(N+2)/Rb)^2);
Lp(N+1) = Lr(N+1) + M(N+1) -Me(N);
L(2) = Lr(2)- Me(1)-M(2);

for ii = 3:N+1
  L(ii) = Lr(ii) -M(ii-1) - M(ii);
end

L(N+1) = Lr(N+1) - Me(N) -M(N);

L(N+2) = Lr(N+2) -M(N+1);

L
Cr
M


%% Calculation of the total impedance for method 2

 j = sqrt(-1);
```

```
for ii = 1: N
zl(:,ii) = j.*L(ii).*2*pi.*f_axis  ;%+ insertionLoss(f_axis,L(ii),1);
zm(:,ii) = j.*M(ii).*2*pi.*f_axis  ;%+ insertionLoss(f_axis,M(ii),1);
zc(:,ii)= 1./( j.*Cr(ii).*2*pi.*f_axis); % + insertionLoss(f_axis,Cr(ii),2)) ;
%zc(:,ii) = 1./(1./zc(:,ii) + 1./insertionLoss(f_axis,Cr(ii),2));
end

zl(:,N+2) = j.*L(N+2).*2*pi.*f_axis ;% +insertionLoss(f_axis,L(N+2),1);

zl(:,N+1) = j.*L(N+1).*2*pi.*f_axis;% + insertionLoss(f_axis,L(N+1),1);
zm(:,N+1) = j.*M(N+1).*2*pi.*f_axis ;%+ insertionLoss(f_axis,M(N+1),1);

if(imag(Imp)>0)
react = 'C = '
serieReactance = 1./(w0.*imag(Imp));
num2str(serieReactance)
serieReactance= serieReactance';
 serieImp = 1./(serieReactance.*2*pi.*f_axis.*j);
else
react = 'L = '
serieReactance = -(imag(Imp))./w0;
num2str(serieReactance)
serieReactance= serieReactance';
serieImp = (serieReactance.*2*pi.*f_axis.*j);
end

serieImp = -serieImp';
Zs1 = Z_disp + serieImp;
Zs = Zs1 + zl(:,1);
Z = 1./(1./Zs+ 1./zm(:,1));
for ii = 1:N
    Z = 1./(1./(Z + zl(:,ii+1) + zc(:,ii)) + 1./zm(:,ii+1));
end
Z = Z + zl(:,N+2);

%% Uncomment to Plot impedance
% figure
% plot(f_axis,real(Z));
% hold on
% plot(f_axis,imag(Z),'r');
% legend('Real(Z)','Imag(Z)');
% ylabel('Z');
% xlabel('Frequency in Mhz');
% title(strcat('S11 parameter using matching circuit at ',num2str(f0/1000000),' Mhz resonant
frequency' ));
% axis tight;
% grid on;
% XLIM([400000000 800000000])
```

```
%% Calculation of tranfert function
% Something is wrong in there...

zelem(:,1)= serieImp;
zelem(:,2)= zl(:,1);
zelem(:,3)=zm(:,1);
zelem(:,4)= zl(:,1);

for ii=6:3:(6+(N-1)*3)
    zelem(:,ii-1) = zc(:,((ii/3)-1));
    zelem(:,ii) = zm(:,ii/3);
    zelem(:,ii+1) = zl(:,((ii/3)+1));
end

Zeq1= 1./(1./zelem(:,9) + 1./(zelem(:,8) + zelem(:,7)));
Zeq2=  1./(1./zelem(:,6) + 1./(zelem(:,4) + zelem(:,5)));
Zeq3=  1./(1./zelem(:,6) + 1./(zelem(:,8) + zelem(:,7)));

aa =  1./(1./zelem(:,9) + 1./Zeq3);
bb = aa + zelem(:,10);
cc = 1./(1./zelem(:,9) + 1./zelem(:,10));
dd = cc + zelem(:,7) + zelem(:,8);
ee = 1./(1./zelem(:,6) + 1./Zeq1);
ff = ee + zelem(:,4) + zelem(:,5);
gg = 1./(1./zelem(:,3) +1./Zeq2);
hh=  2*gg + zelem(:,2);
H = cc.*ee.*gg.*bb./(dd.*ff.*hh.*aa);

figure;
plot(f_axis,20.*log10(abs(H)));
grid on;

caca = 1./zelem(:,1) + 1./zelem(:,2) - ((1./zelem(:,2)).*gg./hh);
Zin = 1./caca;
pipi = (1./zelem(:,10)).*(aa./bb-1);
Zout = 1./pipi;




%% Uncomment to see the transfert figure
% figure
% plot(f_axis,20*log10(abs(Vs)));
% ylabel('20log10(|H|)');
% xlabel('Frequency in Mhz');
% axis tight;
% grid on;
%XLIM([400000000 2000000000])
```

```
%% Transfert function and S11 parameter plots

S11 = (Z - Rb)./ (Z + Rb);
s11db = 20*log10(abs(S11));

%% Lar theory plot
% epsilone = 10^(Lar/10)-1;
% lat = zeros(1,length(wprime));
% lat(1,1:2) = 10*log10(1+epsilone.*cos(N.*acos(wprime(1:2))).^2);
% lat(1,3:end) = 10*log10(1+epsilone.*cosh(N.*acosh(wprime(3:end))).^2);
% figure;
% plot(wprime,lat);
% xlim([0 3]);

%% Smith chart plot
% figure
% GenerateSmith;
% plot(S11(460:800),'b');


%% Loss calculation

T = H.*H.*(Zin./Zout);
insertionloss = 10.*log10(1./(1-abs(T).^2));
figure;
plot(f_axis,insertionloss);
grid on;
loss = 10.*log10(1./(1-(abs(S11)).^2));
sL = 0.7;
switchLoss = sL*2*(1);
insertionloss=0;
antennaloss = 0;
total_loss = loss + insertionloss + switchLoss + antennaloss;

%% Loss plot

figure
limitaxe = [10 7 5];
plot(f_axis,total_loss);
hold on;
% maximum allowed loss by specifications
plot([474000000 698000000 858000000], limitaxe);
% hold on
legend('Loss with adaptation');
ylabel('L [dB]');
xlabel('Frequency in Mhz');
%title(strcat('S11 parameter using matching circuit at ',num2str(f0/1000000),' Mhz resonant
frequency' ));
axis tight;
legend('total loss','maximum allowed loss');
```

```
grid on;
XLIM([400000000 800000000])
YLIM([0 10])

%% S11 plot
figure
limitaxe = ones(1,1:800).*-3;
plot(f_axis,s11db);
hold on
plot(f_axis,limitaxe);
legend('S11 with adaptation');
ylabel('|s11|');
xlabel('Frequency in Mhz');
%title(strcat('S11 parameter using matching circuit at ',num2str(f0/1000000),' Mhz resonant
frequency' ));
axis tight;
grid on;
XLIM([400000000 800000000])
YLIM([-1+min(s11db) 0])
s11db2 = s11db(1:900);
bd1 = find((s11db2>-3.2) & (s11db2<-2.8))
bd = bd1(end)-bd1(1);
text(450000000,-2,strcat('Bandwidth:',num2str(bd)));

%% S11 without adaptation - uncomment to plot
% figure
% refl = (Z_disp- Rb)./(Z_disp+Rb);
% refldb = 20*log10(abs(refl));
% plot(f_axis,refldb,'r');
% legend('S11 without adaptation');
% ylabel('|s11|');
% xlabel('Frequency in Mhz');
% title(strcat('S11 parameter using matching circuit at ',num2str(f0/1000000),' Mhz resonant
frequency' ));
% axis tight;
% grid on;
% XLIM([400000000 900000000])
% YLIM([-2 1])
```