Appendix

Appendix 3: MatLab Programs for Chapter 3

Program List (3.1): Computation of the input immittance of a transmission line with arbitrary length

```
% TEM Line Input impedance computation
%
%                   ZL+jZ0*tan(beta*L)
%           Zin=Z0 ------------------
%                  Z0+jZL*tan(beta*L)
%
% Note that phase=beta*L; varies between 0 and pi
% or beta*L=(w/vp)*L where vp is the phase velocity\newline
%
% Input
%      ZL: Load termination\newline
%      Z0: Characteristic Impedance
%      phase=beta*L=w*tou; we sweep over w*tou axis\newline
% The phase is being scanned
%
% Output
%       Zin=Rin(w)+jXin(w)
% Program TEM Line: Computation of input impedance
ZL=0.1  % INPUT: Normalized with respect to 50 ohm
Z0=1;   % INPUT: Normalized with respect to 50 ohms
N=501;  % Sampling number
delta=pi/N; % Step size for sweeping over w*tou axis
phase=0.0;% phase=w*tou axis

% Quarter wavelength equivalent resonance circuit components
L=2/pi; % Normalized inuctance with respect to tou
C=L;    % Normalized Capacitance
R=1/ZL; % Resistance at resonance frequency: w*tou=pi/2
j=sqrt(-1);
for i=1:N
     Nin=ZL+j*tan(phase); % Numerator of Zin(w*tou)
     Din=Z0+j*ZL*tan(phase);% Denominator of Zin(w*tou)
     Zin=Nin/Din; % TEM Line input imedance
     Rin(i)=real(Zin);
     Xin(i)=imag(Zin);
     fi(i)=phase;% array for plot purpose
 % Equivalent parallel resonance circuit impedance computations
```

```
    Yres=(1/L/phase/j)+j*phase*C+ZL;% Admittance
    Zres=1/Yres;% Impedance of the resonance circuit
    Rres(i)=real(Zres);
    Xres(i)=imag(Zres);
    Yin=1/Zin;
    Gin(i)=real(Yin);
    Bin(i)=imag(Yin);
    phase=phase+delta;
 end
 figure
 plot(fi,Rin,fi,Rres); % plots of real parts
 figure
 plot(fi,Xin,fi,Xres); % plots of imaginary parts
```

Appendix 4: MatLab Programs for Chapter 4

Program List 4.1a. MatLab Program developed for Example 4.1A

```
% Main_DPS_LoadedTRL_Example_4_1A.m
 clear; clc; close all;
 % This program designs a simple loaded transmission line digital phase
 % shifter with PIN-Diodes using the design
   steps given by Algorithm 3.1
 % This program is written by BS Yarman on May 20, 2018; Vanikoy.
 % Inputs:
   % Desired phase shifts: DEL-FI
   % Actual Design Frequency f0 (Center Frequency)
   % Normalized angular Frequency w0
   % (it is usually selected as unity)
   % Port normalization number R
   % (It is usually selected as R=50 ohms)
   % z0: Normalized Characteristic impedance of the transmission line
   % (usually, z0=Z0/R
   % Note: Z0 is selected as the port normalization number R=50 ohms.
   % Then, normalized characteristic impedance of (4.5) is z0=1
   % vp: Propagation velocity of the transmission line medium.
   % ----------------------------------------------------------------
 % Output:
 % L=physical length of TRL
 % CDa: Actual Diode capacitance
 % ----------------------------------------------------------------
 % Inputs:
      Del_Teta=-45
      f0a=3e9
      w0=1
      R=50; Z0=R; z0=Z0/R;
      eps_r=3.4
  % ----------------------------------------------------------------
  % Computational Steps:
```

```
% Step 1. Compute XB from (4.13) by setting XA=0
XB=2*tand(Del_Teta/2);
% Step 2. Employing (4.10), compute the physical length of the
transmission line.
Teta_T0=(atand(2/XB));
if Teta_T0<0
   Teta_T0=180+Teta_T0;
end
mu=tand(Teta_T0); % See equation (4.6b)
Teta_Rad=Teta_T0*pi/180
tau=Teta_Rad/2/pi/f0a, % actual delay length of TRL
% -----------------------------------------------------------------
% Compute the normalized delay length tau_n of TRL
tau_n=2*pi*f0a*tau/w0;
% -----------------------------------------------------------------
v0=3e8; % Free Space Velocity of Propagation
vp=v0/sqrt(eps_r)
length=tau*vp, % Actual physical length of TRL
% Note: If the transmission line is realized using microstrip line,
% then v_p is computed employing (4.16).
% Step 3. Compute the normalized value of CD.
CD=abs(1/w0/XB), % Normalized Capacitanceof the PIN-Diode [D]
% Step 4. Compute the actual value of CDa
CDa=CD/2/pi/f0a/R,
   % Actual value of the revere biased capacitance [D]
% -----------------------------------------------------------------
w1=0;  w2=2; N=10001;
DW=(w2-w1)/(N-1);
w=w1;
% -----------------------------------------------------------------
W=zeros(1,N); DEL_FIA=zeros(1,N);
RO21A_Ar=zeros(1,N); RO21B_Ar=zeros(1,N);
FI21A_Ar=zeros(1,N); FI21B_Ar=zeros(1,N);
for i=1:N
   W(i)=w*f0a;
% [ Zin ] =Zin_Loaded_TRL_X(w,z0,tau_n, XB);
[ RO21B,FI21B ] = SPAR_Loaded_TRL(w, w0, XB, tau_n);
[ RO21A,FI21A ] = SPAR_Loaded_TRL(w, w0, 0, tau_n);
FI21A_Ar(i)=FI21A; FI21B_Ar(i)=FI21B;
RO21A_Ar(i)=20*log10(RO21A); RO21B_Ar(i)=20*log10(RO21B);
DEL_FIA(i)=FI21B-FI21A;
%
  w=w+DW;
end
%
figure
plot(W,DEL_FIA,W,FI21A_Ar,W,FI21B_Ar)
xlabel('Normalized Angular Frequency w')
ylabel('Phase Shift DEL-FI=FI21B-FI21A')
title ('Phase Performance of Loaded TRL for DEL-FI=FI21B-FI21A')
legend('DEL-FI','FI21A','FI21B')
%
figure
```

```
plot(W,RO21A_Ar,W,RO21B_Ar)
xlabel('Normalized Angular Frequency w')
ylabel('TPG of State-A and State-B in dB')
title ('Gain Performance of LL-DPS ')
legend('TPG for State-A','TPG for State-B')
```

Program List 4.1b. MatLab Program developed for Example 4.1B

```
% Main_DPS_LoadedTRL_Example_3_1B.m
clear; clc; close all;
% This program designs a simple loaded transmission line digital phase
% shifter with PIN-Diodes using the designsteps given by Algorithm 3.1
% This program is written by BS Yarman on May 20, 2018; Vanikoy.
% Inputs:
    % Desired phase shifts: DEL-FI
    % Actual Design Frequency f0 (Center Frequency)
    % Normalized angular Frequency w0
    % (it is usually selected as unity)
    % Port normalization number R
    % (It is usually selected as R=50 ohms)
    % z0: Normalized Characteristic impedance of the transmission line
    % (usually, z0=Z0/R
    % Note: Z0 is selected as the port normalization number R=50 ohms.
    % Then, normalized characteristic impedance of (4.5) is z0=1
    % vp: Propagation velocity of the transmission line medium.
% ----------------------------------------------------------------
% Output:
% L=physical length of TRL
% CDa: Actual Diode capacitance
% ----------------------------------------------------------------
% Inputs:
       Del_Teta=-90
       f0a=3e9
       w0=1
       R=50; Z0=R; z0=Z0/R;
       eps_r=3.4
% ----------------------------------------------------------------
% Computational Steps:
% Step 1. Compute XB from (4.13) by setting XA=0
XB=2*tand(Del_Teta/2);
% Step 2. Employing (4.10), compute thephysical length of the
transmission line.
Teta_T0=(atand(2/XB));
if Teta_T0<0
    Teta_T0=180+Teta_T0;
end
mu=tand(Teta_T0); % See equation (4.6b)
Teta_Rad=Teta_T0*pi/180
tau=Teta_Rad/2/pi/f0a, % actual delay length of TRL
% ---------------------------------------------------------------
```

```
% Compute the normalized delay length tau_n of TRL
tau_n=2*pi*f0a*tau/w0;
% -------------------------------------------------------------
v0=3e8; % Free Space Velocity of Propagation
vp=v0/sqrt(eps_r)
length=tau*vp, % Actual physical length of TRL
% Note: If the transmission line is realized using microstrip line,
% then v_pis computed employing (4.16).
% Step 3. Compute the normalized value of CD.
CD=abs(1/w0/XB), % Normalized Capacitance of the PIN-Diode [D]
% Step 4. Compute the actual value of CDa
CDa=CD/2/pi/f0a/R, % Actual value of the revere biased capacitance [D]
%-------------------------------------------------------------

w1=0; w2=2; N=10001;
DW=(w2-w1)/(N-1);
w=w1;
% -------------------------------------------------------------
W=zeros(1,N); DEL_FIA=zeros(1,N);
RO21A_Ar=zeros(1,N); RO21B_Ar=zeros(1,N);
FI21A_Ar=zeros(1,N); FI21B_Ar=zeros(1,N);
for i=1:N
    W(i)=w*f0a;
% [ Zin ] =Zin_Loaded_TRL_X(w,z0,tau_n, XB);
[ RO21B,FI21B ] = SPAR_Loaded_TRL(w, w0, XB, tau_n);
[ RO21A,FI21A ] = SPAR_Loaded_TRL(w, w0, 0, tau_n);
FI21A_Ar(i)=FI21A; FI21B_Ar(i)=FI21B;
RO21A_Ar(i)=20*log10(RO21A); RO21B_Ar(i)=20*log10(RO21B);
DEL_FIA(i)=FI21B-FI21A;
%
    w=w+DW;
end
%
figure
plot(W,DEL_FIA,W,FI21A_Ar,W,FI21B_Ar)
xlabel('Normalized Angular Frequency w')
ylabel('Phase Shift DEL-FI=FI21B-FI21A')
title ('Phase Performance of Loaded TRL for DEL-FI=FI21B-FI21A')
legend('DEL-FI','FI21A','FI21B')
%
figure
plot(W,RO21A_Ar,W,RO21B_Ar)
xlabel('Normalized Angular Frequency w')
ylabel('TPG of State-A and State-B in dB')
title ('Gain Performance of LL-DPS ')
legend('TPG for State-A','TPG for State-B')
```

Program List 4.1C. MatLab Program developed for Example 4.1C

```
%Main_ISLL_DPS_with_Inductors_Example_4_1C.m
% This program designs a simple loaded linephase shifter with series
% inductive load. In this configuration,when the PIN diodes are
on, they
```

```
% are shorted. Then, the line teta_T0 isloaded with the
%normalized inductor L.
% If the diodes are off, the diode reverse biased capacitor
CD resonates
% with the loading inductor L. In this case, XA=w*L-1/(w*CD)
 or at w=w0,
% CD=1/(w0^2*L).
clear; clc; close all;
% This program designs a simple loaded transmission line digital phase
% shifter with ideal switches using the design steps given
by Algorithm 3.1
% This program is written by BS Yarman on May 20, 2018; Vanikoy.
 % Inputs:
    % Desired phase shifts: DEL-FI
    % Actual Design Frequency f0 (Center Frequency)
    % Normalized angular Frequency w0
    % (it is usually selected as unity)
    % Port normalization number R
    % (It is usually selected as R=50 ohms)
    % z0: Normalized Characteristic impedance of the transmission line
    % (usually, z0=Z0/R
    % Note: Z0 is selected as the port normalization number R=50 ohms.
    % Then, normalized characteristic impedance of (4.5) is z0=1
    % vp: Propagation velocity of the transmission line medium.
% ---------------------------------------------------------------
% Output:
% L=physical length of TRL
% CDa: Actual Diode capacitance
% ---------------------------------------------------------------
% Inputs:
       Del_Teta=45
       f0a=3e9
       w0=1.
       R=50; Z0=R; z0=Z0/R;
       eps_r=3.4
% ---------------------------------------------------------------
% Computational Steps:
% Step 1. Compute XB from (4.13) by setting XA=0
XB=2*tand(Del_Teta/2);
% Step 2. Employing (4.10), compute the physical length
of the transmission line.
Teta_T0=(atand(2/XB));
if Teta_T0<0
    Teta_T0=180+Teta_T0;
end
mu=tand(Teta_T0); % See equation (4.6b)
Teta_Rad=Teta_T0*pi/180
tau=Teta_Rad/2/pi/f0a, % actual delay length of TRL
% ---------------------------------------------------------------
% Compute the normalized delay length tau_n of TRL
tau_n=2*pi*f0a*tau/w0;
% ---------------------------------------------------------------
v0=3e8; % Free Space Velocity of Propagation
```

```
vp=v0/sqrt(eps_r)
length=tau*vp, % Actual physical length of TRL
% Note: If the transmission line is realized using microstrip line,
% then vp is computed employing (4.16).
% Step 3. Compute the normalized value of Inductor L:
% Note that XB=w0*L, L=XB/w0;
L=XB/w0, % Normalized value of the series load inductor
% Step 4. Compute the actual value La of the inductor
L (Normalized value)
La=L*R/2/pi/f0a, % Actual value of the series load inductor
CD=1/w0/w0/L,
   % Normalized value of the reverse biased diode capacitance
CDa=CD/R/2/pi/f0a,%Actual value of the reverse biased diode
capacitance
% ---------------------------------------------------------------

w1=0; w2=2; N=10001;
DW=(w2-w1)/(N-1);
w=w1;
% ---------------------------------------------------------------
W=zeros(1,N); DELFIA = zeros(1,N);
RO21A_Ar = zeros(1,N);RO21B_Ar=zeros(1,N);
FI21A_Ar=zeros(1,N); FI21B_Ar=zeros(1,N);
for i=1:N
    W(i)=w*f0a;
% [ Zin ] =Zin_Loaded_TRL_X(w,z0,tau_n, XB);
XA_w=w*L-1/w/CD;
XB_w=w*L;
[ RO21B,FI21B ] = SPAR_Loaded_TRL(w, w0, XB_w, tau_n);
[ RO21A,FI21A ] = SPAR_Loaded_TRL(w, w0, XA_w, tau_n);
FI21A_Ar(i)=FI21A; FI21B_Ar(i)=FI21B;
RO21A_Ar(i)=20*log10(RO21A); RO21B_Ar(i)=20*log10(RO21B);
DEL_FIA(i)=FI21B-FI21A;
%
    w=w+DW;
end
%
figure
plot(W,DEL_FIA,W,FI21A_Ar,W,FI21B_Ar)
xlabel('Normalized Angular Frequency w')
ylabel('Phase Shift DEL-FI=FI21B-FI21A')
title ('Phase Performance of Loaded TRL for DEL-FI=FI21B-FI21A')
legend('DEL-FI','FI21A','FI21B')
%
figure
plot(W,RO21A_Ar,W,RO21B_Ar)
xlabel('Normalized Angular Frequency w')
ylabel('TPG of State-A and State-B in dB')
title ('Gain Performance of LL-DPS')
legend('TPG for State-A','TPG for State-B')
```

Program List 4.2. Computation of Scattering Parameters for Loaded Transmission Line

```
function [ RO21,FI21 ] = SPAR_Loaded_TRL(w, w0, X, tau_n)
 % This function generates the scattering parameters of the loaded
 % transmission line [X]-[TRL]-[X]
 % Inputs:
 % w: Normalized angular frequency
 % w0: Normalized angular frequency
 % X: Normalized series load of TRL
 % tau_n: normalized length of TRL
 % where Teta_T0 is the length of the TRL in degree.
 % Output:
 % S11: Input Reflection Coefficient of the loaded TRL: [X-TRL-X]
 % S21: Transfer Scattering Parameter of the loaded TRL
 % RO11=abs(S11)
 % FI11=Phase of S11=atan2d(X11,R11)
 % RO21=abs(S21)
 % FI21=Phase of S21=atan2d(X21,R21)
 % ---------------------------------------------------------------
 j=sqrt(-1);
 % Step 1: Compute mu=tan(w0*tau_n)
 mu=tan(w0*tau_n);
 % Step 2: Compute length of TRL as Teta=w*tau_n
 Teta=w*tau_n;
 % Step 3: Compute S21T
 S21T=cos(Teta)-j*sin(Teta);
 % Step 4: Compute S11L and S21L
 gL=2+j*X;
 S11L=j*X/gL;
 S21L=2/gL;
 %---------------------------------------------------------------
 \% Step 5: Compute S11 and S21 of X-TRL-X
 \% S21=[S21L^2 ][S21T ]/(1-S21T^2 S11L^2)
 %---------------------------------------------------------------
 S11=j*(2*X-mu*X*X)/(2*(1-mu*X)+j*(2*mu+2*X-mu*X*X));
 R11=real(S11); X11=imag(S11); FI11=atan2d(X11,R11);
 S21=S21L*S21L*S21T/(1-S21T*S21T*S11L*S11L);
 R21=real(S21); X21=imag(S21); FI21=atan2d(X21,R21);
 %---------------------------------------------------------------
 % Step 6: Generate {RO11, F11},{RO21,FI21}
 % RO11=abs(S11);
 RO21=abs(S21);
 % error1=RO21*RO21-(1-RO11*RO11);
 % error2=2*FI21-(180+2*FI11);
 end
```

Program List 4.3. Main program for Example 4.2

```
% Main_SLL_DPS_Example_4_2.m
 clc
 close all
 clear
```

```
% This program designs a Series Loaded-Line DPS with Impedance
 Transforming
% Lines
% This program is developed by BS Yarman, Vanikoy, June 1, 2018.
% Inputs:
% R: Termination resistors
% and normalization numbers for the scattering parameters
% ZT: Characteristic impedance of the transforming line ?_T
% f0a: Actual Design Frequency
% DEl-Teta0: Desired Phase Shift between the switching states
% which is specified at the design frequency f0a
% w0: Normalized Angular Frequency at f0a
% CDa: Actual Reverse Biased Capacitance CDa of the switching Diodes
% Computational Steps
% Step 1a: Normalize ZT
% zT=ZT/R
% Step 1b: Normalize CDa
% CD=R*?0a*CDa=R X 2 X ? X f0a X CDa
% Step 1c: Compute the major design parameter ?
% beta=1/2 tan((Del-Teta0)/2)
% Step 2: Compute the normalized load reactance XL
% XL=-1/(w0*CD)
% Step 3:Compute the coefficients {a,b,c} of the quadratic
  equation (4.24a)
% as in (4.26)
% a=(zT^2*beta+XL)
% b=-(4*beta*XL-zT*beta*XL+zT-zT^2)
% c=(4*zT*beta+zT*XL)
% Step 4:Compute the discriminant of the quadratic equation
  as in(4.25b)
% Discriminant=b^2-4ac>0
% Step 5:Check if Discriminant>0.If yes, continue with the following
  steps.
% If not, vary the design parameters w0 or CD until you end up with a
  % positive discriminant.
% Step 6: Compute the loading reactance XA of State-A
% XA(1,2)=(-b+/-sqrt(Discriminant))/2a
% In this step, prefer the positive value for XA to make
  the centerline as
% short as possible. If both values of XA(1,2) is negative,
  then prefer
% the smallest value of |X_A(1,2)| to minimize the length of the
  centerline.
% Step 7: Compute electrical length of the line transformer Teta_T
% Teta_T=atand[XA]+k1*pi
% In this step, if XA is positive set k1=0.
% If XA is negative, then set k1=1.
% Step 8: Compute the electrical lengths Teta_0A and Teta_0B
  as in (4.29)
% Teta_(0A)=atand(2/XA)+k2*pi>0
% Teta_(0B)=atand(2/XB)+k3*pi>0
% Step 9: Compute the mismatched lengths ?T01 and ?T02 of the
  centerline
```

```
% as in (4.30)
% Teta_T0,1=sqrt(Teta_0A*Teta_0B)
% Teta_T0,2=(Teta_(0_A)+Teta_(0_B))/2
% Step 10: Analyze the phase shift performance of the designed Series
% Loaded-Line-Digital Phase Shifter (SLL-DPS) using MatLab both for
% Teta(T0,1) and Teta(T0,2), plot the results and identify the
  best value
% for Teta_T0 out of {Teta(T0,1),Teta(T0,2)}.

%----------------------------------------------------------------
R=50, ZT=50, Z0=50,
f0a=3e9,
CDa=2.8e-12,
w0=1.
Del_Teta=22.5,
%----------------------------------------------------------------
% Step 1a: Normalize ZT
zT=ZT/R,
% Step 1b: Normalize CDa
w0a=2*pi*f0a,
% C_D=R?_0a C_Da=RX2X?f_0aXC_Da
CD=R*w0a*CDa,
% Step 1c: Compute the major design parameter ?=1/2 tan((??_0)/2)
% beta=(1/2)* tan((Teta_0)/2)
beta=tand(Del_Teta/2)/2,
% Step 2: Compute the normalized load reactance XL
XL=-1/w0/CD,
% Step 3: Compute the coefficients {a,b,c} of the quadratic equation
  (4.24a) as in (4.26)
a=zT*zT*beta+XL,
b=-(4*beta*XL-zT*beta*XL+zT-zT*zT),
c=(4*zT*beta+zT*XL),
% Step 4: Compute the discriminant_a of the quadratic equation
  as in (4.25b)
% ?_a=b^2-4ac?0
Discriminant=b*b-4*a*c,
% Step 5: Check if ?_a>0.
%If yes, continue with the following steps.
% If not, vary the design parameters ?_0 orC_D until you end up with
  positive discriminant ?_a.
if Discriminant<0
    Attention='Negative Discriminator: Change CDa or w0'
end
if Discriminant>0
% Step 6a: Compute the loading reactance X_A of State-A
XA1=(-b-sqrt(Discriminant))/2/a,
XA2=(-b+sqrt(Discriminant))/2/a,
if XA1>0
    XA=XA1,
    ROA1=9999999;
    if XA2>0
        if XA1>XA2
            XA=XA2,
```

```
            ROA2=9999999;
        end
        if XA2<XA1
            XA=XA1,
        end
    end
end
if XA1<0
    ROA1=abs(XA1);
end
if XA2<0
   ROA2=abs(XA2)
end
if ROA1>ROA2
    XA=XA2
end
if ROA2>ROA1
    XA=XA1
end
% Step 6b: Compute XB from XA
XB=(XA-4*beta)/(1+beta*XA),
% Step 7: Compute electrical length of the line transformer Teta_T
if XA>0
Teta_T=atand(XA)
end
if XA<0
    Teta_T=atand(XA)+180,
end
% Step 8: Compute the electrical lengths Teta_0A and Teta_0B
  as in (4.29)
if XA<0
Teta_0A=atand(XA)+180,
end
if XA>0
 Teta_0A=atand(XA),
end
% Teta_0A=atand(2/XA)+k2*pi>0
% Teta_0B=atand(2/XB)+k3*pi>0
if XB<0
Teta_0B=atand(XB)+180,
end
if XB>0
 Teta_0B=atand(XB),
end
%
% Step 9: Compute the mismatched lengths Teta_(T0,1) and Teta_(T0,2) of
%the centerline as in (4.30)
% Teta_T01=sqrt(Teta_0A*Teta_0B)
Teta_T01=sqrt(Teta_0A*Teta_0B),
Teta_T02=(Teta_0A+Teta_0B)/2,
% Select the short line length:
if Teta_T01>Teta_T02
    Teta_T0=Teta_T02
```

```
end
if Teta_T02>Teta_T01
    Teta_T0=Teta_T01
end
%-------------------------------------------------------------
% In order to use our analysis tool, compute the normalized
  delay length
tau_n=Teta_T0/w0
%-------------------------------------------------------------
end; % End of positive Discriminant loop
%-------------------------------------------------------------
w1=0.9; w2=1.1; N=10001;
DW=(w2-w1)/(N-1);
w=w1;
%-------------------------------------------------------------
W=zeros(1,N); DEL_FIA=zeros(1,N);
RO21A_Ar=zeros(1,N); RO21B_Ar=zeros(1,N);
FI21A_Ar=zeros(1,N); FI21B_Ar=zeros(1,N);
for i=1:N
    W(i)=w*f0a;
% [ Zin ] =Zin_Loaded_TRL_X(w,z0,tau_n, XB);
[ RO21B,FI21B ] = SPAR_Loaded_TRL(w, w0, XB, tau_n);
[ RO21A,FI21A ] = SPAR_Loaded_TRL(w, w0, XA, tau_n);
FI21A_Ar(i)=FI21A; FI21B_Ar(i)=FI21B;
RO21A_Ar(i)=20*log10(RO21A); RO21B_Ar(i)=20*log10(RO21B);
DEL_FIA(i)=FI21B-FI21A;
%
    w=w+DW;
end
%
figure
plot(W,DEL_FIA,W,FI21A_Ar,W,FI21B_Ar)
xlabel('Normalized Angular Frequency w')
ylabel('Phase Shift DEL-FI=FI21B-FI21A')
title ('Phase Performance of Loaded TRL for DEL-FI=FI21B-FI21A')
legend('DEL-FI','FI21A','FI21B')
%
figure
plot(W,RO21A_Ar,W,RO21B_Ar)
xlabel('Normalized Angular Frequency w')
ylabel('TPG of State-A and State-B in dB')
title ('Gain Performance of LL-DPS')
legend('TPG for State-A','TPG for State-B')
 % -------------------------------------------------------------
 if Discriminant<0
    Attention='Discriminant is negative change CDa or w0'
end
```

Program List 4.4. Computation of Scattering Parameters for PLL-DPS

```
function [ RO21,FI21,RO11,FI11 ] = SPAR_Parallel_Loaded_TRL(w, w0, Y,
   tau_n)
% This function generates the scattering parameters of the loaded
```

```
% transmission line [X]-[TRL]-[X].
% Developed by BS Yarman on June 10, 2018,Philadelphia,
                          Pennsylvania, USA
 % Inputs:
    % w: Normalized angular frequency
    % w0: Normalized angular frequency
    % Y: Normalized Parallel load of TRL
    % tau_n: normalized length of TRL
    % where Teta_T0 is the length of the TRL in degree.
 % Output:
    % S11: Input Reflection Coefficient of the loaded TRL: [X-TRL-X]
    % S21: Transfer Scattering Parameter of the loaded TRL
    % RO11=abs(S11)
    % FI11=Phase of S11=atan2d(X11,R11)
    % RO21=abs(S21)
    % FI21=Phase of S21=atan2d(X21,R21)
    % ----------------------------------------------------------------
j=sqrt(-1);
% Step 1: Compute mu=tan(w0*tau_n)
mu=tan(w0*tau_n);
% Step 2: Compute length of TRL asTeta=w*tau_n
Teta=w*tau_n;
% Step 3: Compute S21T
S21T=cos(Teta)-j*sin(Teta);
% Step 4: Compute S11L and S21L
gL=2+j*Y;
S11L=(-1)*j*Y/gL;
S21L=2/gL;
%-----------------------------------------------------------------
% Step 5: Compute S11 and S21 of Y-TRL-Y
% S21=[S21L^2 ][S21T ]/(1-S21T^2S11L^2)
%-----------------------------------------------------------------
S11=(-1)*j*(2*Y-mu*Y*Y)/(2*(1-mu*Y)+j*(2*mu+2*Y-mu*Y*Y));
R11=real(S11); X11=imag(S11); FI11=atan2d(X11,R11);
S21=S21L*S21L*S21T/(1-S21T*S21T*S11L*S11L);
R21=real(S21); X21=imag(S21); FI21=atan2d(X21,R21);
%-----------------------------------------------------------------
% Step 6: Generate {RO11, F11},{RO21,FI21}
RO11=abs(S11);
RO21=abs(S21);
% error1=RO21*RO21-(1-RO11*RO11);
% error2=2*FI21-(180+2*FI11);
  end
```

Program List 4.5. Main_PLL_DPS_Example_4_3.m

```
% Main_PLL_DPS_Example_4_3.m
clc
close all
clear
% This program designs a Parallel Loaded-Line DPS with
% Impedance Transforming Lines
```

```
% This program is developed by BS Yarman,Philladelphia,
  USA June 10, 2018.
% Inputs:
% R: Termination resistors
% and normalization numbers for the scattering parameters
% ZT: Characteristic impedance of the transforming line ZT
% f0a: Actual Design Frequency
% DEl-Teta0: Desired Phase Shift between the switching states
% which is specified at the design frequency f0a
% w0: Normalized Angular Frequency at f0a
% CDa: Actual Reverse Biased Capacitance CDa of the switching Diodes
% Computational Steps
% Step 1a: Normalize ZT
% zT=ZT/R
% Step 1b: Normalize CDa
% CD=R*w0a*CDa=RX2XpiXf0aXCDa
% Step 1c: Compute the major design parameter beta
% beta=1/2 tan((Del-Teta0)/2)
% Step 2: Compute the normalized load susceptance YL=w*CD
% YL=w*CD
% Step 3:Compute the coefficients {aY,bY,cY} as in (4.40g)
% aY=1+beta*YL;
% bY=3*beta;
% cY=4*beta*YL+1
% Step 4:DELY=bY*bY-4*aY*cY;
%
% Step 5:Check if DELY>0.If yes, continue with the following steps.
% If not, vary the design parameters w0 or CD until you end up with a
% positive discriminant.
% Step 6: Compute the loading reactance XA of State-A
% YA(1,2)=(-bY+/-sqrt(DELY))/2aY
% % Step 8: Compute the electrical lengths Teta_0A and Teta_0B
  as in (4.29)
% Teta_(0A)=atand(2/YA)+kA*pi>0
% Teta_(0B)=atand(2/YB)+kB*pi>0
% Step 9: Compute the mismatched lengths?T01 and Teta_T02 of the
  centerline
% as in (4.30)
% Teta_T0,1=sqrt(Teta_0A*Teta_0B)
% Teta_T0,2=(Teta_(0_A)+Teta_(0_B))/2
% Step 10: Analyze the phase shift performance of the designed Series
% Loaded-Line-Digital Phase Shifter (SLL-DPS) using MatLab both for
% Teta(T0,1) and Teta(T0,2), plot the results and identify the
  best value
% for Teta_T0 out of {Teta(T0,1),Teta(T0,2) }.
%----------------------------------------------------------------
R=50, ZT=50, Z0=50,
f0a=3e9,
CDa=2.8e-12,
w0=1,
Del_Teta=-22.5,% in degree
%----------------------------------------------------------------
% Step 1a: Normalize ZT
```

```
zT=ZT/R,
% Step 1b: Normalize CDa
w0a=2*pi*f0a,
% CD=R*w0a*CDa=RX2XpiXf0aXCDa
CD=R*w0a*CDa,
% Step 1c: Compute the major design parameter
% beta=(1/2)* tan((Teta_0)/2)
beta=tand(Del_Teta/2)/2,
% Step 2: Compute the normalized load susceptance YL
YL=w0*CD,
% Step 3: Compute the coefficients{aY,bY,cY}
aY=1+beta*YL,
bY=+3*beta,
cY=4*beta*YL+1,
% Step 4: Compute the discriminant DELY
DELY=bY*bY-4*aY*cY,
% Step 5: Check if ?_a>0.
%If yes, continue with the following steps.
% If not, vary the design parameters CD until you end up with
                          positive discriminant DELY.
if DELY<0
    Attention='Negative Discriminator: ChangeCDa'
end
if DELY>0
% Step 6a: Compute the loading reactance YA of State-A
YA1=(-bY-sqrt(DELY))/2/aY,
YA2=(-bY+sqrt(DELY))/2/aY,
ROA2=abs(YA2),
ROA1=abs(YA1)
if YA1>0
    YA=YA1,
    ROA1=9999999;
    if YA2>0
        if YA1>YA2
            YA=YA2,
            ROA2=9999999;
        end
        if YA2<YA1
            YA=YA1,
        end
    end
end
if YA1<0
    ROA1=abs(YA1);
end
if YA2<0
   ROA2=abs(YA2)
end
if ROA1>ROA2
    YA=YA2
end
if ROA2>ROA1
    YA=YA1
```

```
end
% Step 6b: Compute YB from YA
YB=(YA+4*beta)/(1-beta*YA),
YB2=(YL*YA-1)/(YA+YL),
% Step 7: Compute electrical length of the line transformer Teta_T
if YA<0
Teta_T=-atand(1/YA)
end
if YA>0
    Teta_T=-atand(1/YA)+180,
end
% Step 8: Compute the electrical lengths Teta_0A and Teta_0B
                          as in (4.29)
if YA<0
Teta_0A=atand(2/YA)+180,
end
if YA>0
 Teta_0A=atand(2/YA),
end
\% Teta_0A=atand(2/YA)+kA*pi>0
\% Teta_0B=atand(2/YB)+kB*pi>0
if YB<0
Teta_0B=atand(2/YB)+180,
end
if YB>0
 Teta_0B=atand(1/YB),
end
%
% Step 9: Compute the mismatched lengths Teta_(T0,1) and Teta_(T0,2) of
%the centerline as in (4.30)
% Teta_T01=sqrt(Teta_0A*Teta_0B)
Teta_T01=sqrt(Teta_0A*Teta_0B),
Teta_T02=(Teta_0A+Teta_0B)/2,
% Select the short line length:
if Teta_T01>Teta_T02
    Teta_T0=Teta_T02
end
if Teta_T02>Teta_T01
    Teta_T0=Teta_T01
end
%----------------------------------------------------------------
% In order to use our analysis tool, compute the normalized delay
                          lengths
% as
tau_n=pi*Teta_T0/w0/180
tau_nT=pi*Teta_T/w0/180
%----------------------------------------------------------------
end; % End of positive Discriminant loop
%----------------------------------------------------------------
w1=0.9; w2=1.1; N=10001;
DW=(w2-w1)/(N-1);
w=w1;
%----------------------------------------------------------------
```

```
W=zeros(1,N); DEL_FIA=zeros(1,N);
RO21A_Ar=zeros(1,N); RO21B_Ar=zeros(1,N);
FI21A_Ar=zeros(1,N); FI21B_Ar=zeros(1,N);
for i=1:N
    W(i)=w*f0a;
    YL=w*CD;
    muT=tan(w*tau_nT);
    YB=(YL+muT)/(1-YL*muT);
    YA=-1/muT;
[ RO21B,FI21B,RO11B,FI11B ] = SPAR_Parallel_Loaded_TRL(w, w0, YB,
   tau_n);
[ RO21A,FI21A,RO11A,FI11A ] = SPAR_Parallel_Loaded_TRL(w, w0, YA,
   tau_n);
FI21A_Ar(i)=FI21A; FI21B_Ar(i)=FI21B;
RO21A_Ar(i)=20*log10(RO21A); RO21B_Ar(i)=20*log10(RO21B);
DEL_FIA(i)=FI21B-FI21A;
%
    w=w+DW;
end
%
figure
plot(W,DEL_FIA,W,FI21A_Ar,W,FI21B_Ar)
xlabel('Normalized Angular Frequency w')
ylabel('Phase Shift DEL-FI=FI21B-FI21A')
title ('Phase Performance of PLL-DPS with TRL Transformer: Teta-T')
legend('DEL-FI','FI21A','FI21B')
%
figure
plot(W,RO21A_Ar,W,RO21B_Ar)
xlabel('Normalized Angular Frequency w')
ylabel('TPG of State-A and State-B in dB')
title ('Gain Performance of PLL-DPS with Transfermer Teta-T')
legend('TPG for State-A','TPG for State-B')
 %-----------------------------------------------------------------
 if DELY<0
    Attention='Discriminant is negative change CDa or w0'
end
```

Program List 4.6. Main_PLL_DPS_Example_4_4.m

```
% Main_PLL_DPS_Example_4_4.m
clc
close all
clear
%-----------------------------------------------------------------
%Design of PLL-DPS with Parallel Resonant Circuits
% This algorithm designs a PLL-DPS with parallel resonant circuits.
%In this circuit, we employed PIN diodes as switching elements
%with reverse biased capacitors CD.
% However, the same configuration can be implemented using MOS devices
% perhaps as MMIC.
% Inputs:
```

```
% f0a: Actual Center Frequency (ACF) of the designy
% w0: Normalized Angular Frequency (NAF) of the design
% Del_Teta: Digital Phase Shift at w0
% CD: Reverese biase capacitance
% Computational Steps:
% -----------------------------------------------------------------
% Inputs:
R=50;
f0a=3e9
w0=1
Del_Teta=-22.5,
CDa=1006e-15,
% Step 1a: Compute the major design parameter
beta_Y=tand(Del_Teta/2),
% Step 1b: Compute the normalized value of CD
w0a=2*pi*f0a,
CD=w0a*R*CDa,
% Step 2: Compute the effective capacitor C of State-A
C=-2*(beta_Y)/w0,
% Step 3: Compute Normalized value of the auxiliary or
  tuning capacitor CA
 CA=(C+sqrt(C*C+4*CD*C))/2,
% Step 4: Compute the actual value of CA
  CAa=CA/2/pi/f0a/R,
% Step 5: Compute the switch capacitor C_T
  CT=CA*CD/(CA+CD),
% Step 6a: Compute the normalized value of the shunt inductor LA from
% the resonance condition of State-B
LA=1/w0/w0/CT,
% Step 6b: Compute the actual value of LALAa=R*LA/2/pi/f0a,
% Step 7: Compute the length of the centralline Teta_T0
 Teta_T0=atand(2/w0/C)
 if Teta_T0<0
     Teta_T0=Teta_T0+180
 end
%
%-----------------------------------------------------------------
w1=0.5; w2=2.1; N=10001;
DW=(w2-w1)/(N-1);
w=w1;
tau_n=pi*Teta_T0/w0/180
%-----------------------------------------------------------------
W=zeros(1,N); DEL_FIA=zeros(1,N);
RO21A_Ar=zeros(1,N); RO21B_Ar=zeros(1,N);
FI21A_Ar=zeros(1,N); FI21B_Ar=zeros(1,N);
for i=1:N
   W(i)=w*f0a;
YA=(w*w*LA*CA-1)/w/w/LA;
YB=(w*w*LA*CT-1)/w/w/LA;
[ RO21B,FI21B,RO11B,FI11B ] = SPAR_Parallel_Loaded_TRL(w, w0, YB,
   tau_n);
[ RO21A,FI21A,RO11A,FI11A ] = SPAR_Parallel_Loaded_TRL(w, w0, YA,
   tau_n);
```

```
FI21A_Ar(i)=FI21A; FI21B_Ar(i)=FI21B;
RO21A_Ar(i)=20*log10(RO21A); RO21B_Ar(i)=20*log10(RO21B);
DEL_FIA(i)=FI21B-FI21A;
%
    w=w+DW;
end
%
figure
plot(W,DEL_FIA,W,FI21A_Ar,W,FI21B_Ar)
xlabel('Normalized Angular Frequency w')
ylabel('Phase Shift DEL-FI=FI21B-FI21A')
title ('Phase Performance of PLL-DPS with L//C Shunt Loads')
legend('DEL-FI','FI21A','FI21B')
%
figure
plot(W,RO21A_Ar,W,RO21B_Ar)
xlabel('Normalized Angular Frequency w')
ylabel('TPG of State-A and State-B in dB')
title ('Gain Performance of LL-DPS')
legend ('TPG for State-A','TPG for State-B')
```

Program List 4.7. Main_PLL_DPS_Example_4_5.m

```
% Main_PLL_DPS_Example_4_5.m
clc
close all
clear
%-----------------------------------------------------------------
%Design of Perfectly Matched PLL-DPS with effective loading inductor L
% in State-B
% This algorithm designs a Perfectly Matched PLL-DPS with loaded with
% effective inductor in State-B.
% In this circuit, we employed PIN diodes as switching elements
% with reverse biased capacitors CD.
% However, the same configuration can be implemented using MOS devices
% perhaps as MMIC.
% Inputs:
% f0a: Actual Center Frequency (ACF) of the design
% w0: Normalized Angular Frequency (NAF) of the design
% Del_Teta: Digital Phase Shift at w0 which must be a negative quantity
% CD: Reverese biase capacitance
% Computational Steps:
% ----------------------------------------------------------------
% Inputs:
R=50;
f0a=3e9
w0=1
Del_Teta=-22.5,
CDa=1006e-15,
% Step 1a: Compute the major design parameter
beta_Y=tand(Del_Teta/2),
```

```
% Step 1b: Compute the normalized value of CD
w0a=2*pi*f0a,
CD=w0a*R*CDa,
% Step 2: Compute the effective inductor L of State-B
L=-1/beta_Y/w0/2,
if L<0
    L=-L
end
% Step 3: Compute Normalized value of the auxiliary or tuning
  capacitor CA
Discriminant=1+4*w0*w0*L*CD
CA=(1+sqrt(Discriminant))/2/w0/w0/L,
% Step 4: Compute the actual value of CA
CAa=CA/2/pi/f0a/R,
% Step 5: Compute the switch capacitor C_T
CT=CA*CD/(CA+CD),
% Step 6a: Compute the normalized value of the shunt inductor LA from
% the resonance condition of State-B
LA=1/w0/w0/CA,
% Step 6b: Compute the actual value of LA
LAa=R*LA/2/pi/f0a,
% Step 7: Compute the length of the central line Teta_T0
% Teta_T0=-atand(1/beta_Y)
Teta_T0=-atand(2*w0*L)+180
% ----------------------------------------------------------------
w1=0.5; w2=2.1; N=10001;
DW=(w2-w1)/(N-1);
w=w1;
tau_n=pi*Teta_T0/w0/180
% ----------------------------------------------------------------
W=zeros(1,N); DEL_FIA=zeros(1,N);
RO21A_Ar=zeros(1,N); RO21B_Ar=zeros(1,N);
FI21A_Ar=zeros(1,N); FI21B_Ar=zeros(1,N);
for i=1:N
   W(i)=w*f0a;
YA=(w*w*LA*CA-1)/w/LA;
YB=(w*w*LA*CT-1)/w/LA;
[ RO21B,FI21B,RO11B,FI11B ] = SPAR_Parallel_Loaded_TRL(w, w0, YB,
   tau_n);
[ RO21A,FI21A,RO11A,FI11A ] = SPAR_Parallel_Loaded_TRL(w, w0, YA,
   tau_n);
FI21A_Ar(i)=FI21A; FI21B_Ar(i)=FI21B;
RO21A_Ar(i)=20*log10(RO21A); RO21B_Ar(i)=20*log10(RO21B);
DEL_FIA(i)=FI21B-FI21A;
%
    w=w+DW;
end
%
figure
plot(W,DEL_FIA,W,FI21A_Ar,W,FI21B_Ar)
xlabel('Normalized Angular Frequency w')
ylabel('Phase Shift DEL-FI=FI21B-FI21A')
title ('Phase Performance of PLL-DPS with L//C Shunt Loads')
```

```
legend('DEL-FI','FI21A','FI21B')
%
figure
plot(W,RO21A_Ar,W,RO21B_Ar)
xlabel('Normalized Angular Frequency w')
ylabel('TPG of State-A and State-B in dB')
title ('Gain Performance of LL-DPS')
legend('TPG for State-A','TPG for State-B')
```

Program List 5.1: Main_Lowpass_T_with_Classical_Formulas.m

```
% Main_Lowpass_T_with_Classical_Formulas.m
% December 7, 2018
% Developed by BS Yarman, Vanikoy, Istanbul, Turkey
% It should be noted that the formulas used in this program is valid
   for the values of teta between 0 and 90 degree.
% Exact 90 is not possible
clc, clear
close all
teta=input('Enter negative value for the phase
shift teta=')
while teta==0
 stop= 'Attention teta=0. Phase can never be zero degree. Therefore,
    change teta and re-run the program again.'
    break
end
 while teta==-180
   stop= 'Attention teta=-180. Phase can never be -180 degree.
   Therefore,  change teta and re-run the program again.'
   break
end
while teta>0
stop1='Attention: teta is positive. For a Lowpass Symmterric
T-Section  teta must be negative quantity.'
stop2='Please enter a negative value for teta and re-run the program'
     break
end
% Phase teta is proper. Then, start Computations
if teta>-180
if teta<0
w0=1;
mu=tand(teta)
eta=tand(90-teta)
[ L1,L2,L,C] = mu_Based_Components_of_Lowpass_T
( w0,teta )
%  L=(1-sqrt(1+mu*mu))/w0/mu;
%  C=2*L/(1+w0*w0*L*L);
% ------------------------------------------------
[ La,Ca ] = Components_of_Lowpass_T_Section
( w0, teta )
Error_L=La-L
Error_C=Ca-C
% ------------------------------------------------
```

```
w=0;N=1000;w1=0;w2=2;DW=(w2-w1)/N;
FRI(1:(N+1))=zeros;
for j=1:N+1
WA(j)=w;
% -----------------------------------------------
[ S11,S21,RO11,F11,RO21,F21 ] = S_Par_Lowpass_T
( w,La,Ca );
  F21A(j)=F21;
  RO21A(j)=RO21;
% -----------------------------------------------
     w=w+DW;
end
% -----------------------------------------------
% Phase of S21
figure
plot(WA,F21A)
title('Phase variation of a "Lowpass T-Section" with classical
    formulation')
legend('F21A with classical computations')
xlabel('Normalized Angular Frequency')
ylabel('Phase of S21')
% Amplitude of S21
figure
plot(WA,RO21A)
title('Amplitude variation of a "Lowpass T-Section" with classical
    formulation')
legend('RO21A:Computations with classical formulation')
xlabel('Normalized Angular Frequency')
ylabel('Amplitude of S21')
% -----------------------------------------------
end
end
```

Program List 5.2: function Components_of_Lowpass_T_Section

```
function [ L,C ] = Components_of_Lowpass_T_Section
( w0, teta )
% This function generates the component values for a lowpass T-Section
%   Developed by BS Yarman: December 6, 2018; Vanikoy, Istanbul, Turkey
% Developed by BS Yarman. December 7, 2018, Vanikoy, Istanbul, Turkey
% Inputs:
%       teta: Negative value of teta
%       w0: Normalized angular frequency
% Outputs:
%       L: Series arm inductor
%       C: Shunt arm capacitor
% -----------------------------------------------
eta=tand(90-teta);
L=(eta+sqrt(1+eta*eta))/w0;
C=2*L/(1+w0*w0*L*L);
End
```

Program List 5.3: function S_Par_Lowpass_T (w,L,C)

```
function [ S11,S21,RO11,F11,RO21,F21 ] = S_Par_
Lowpass_T ( w,L,C )
%This function generates the S-Parameters of Lowpass T Section from the
     computed
% series arm inductor L and the shunt capacitor C
%  Developed by BS Yarman: December 7, 2018, Vanikoy, Istanbul
% ----------------------------------------------
dr=2*(1-w*w*L*C);
dx=w*(2*L+C-w*w*L*L*C);
j=sqrt(-1);
D=dr+j*dx;
N11= j*((1+w*w*L*L)*w*C-2*w*L);
S11=N11/D; R11=real(S11); X11=imag(S11);F11=atan2d
(X11,R11);
S21=2/D;   R21=real(S21); X21=imag(S21);F21=atan2d
(X21,R21);
RO11=abs(S11);
RO21=abs(S21);
End
```

Program List 5.4: For a Lowpass Symmetrical-T Section $\mu = \tan(\theta)$ based component values

```
function [ L1,L2,La_mu,Ca_mu ] = mu_Based_
Components_of_Lowpass_T( w0,teta )
% In this function component values of a symmetric Lowpass T is
    computed
%   Input phase teta is defined as a negative angle mu=tand(teta);
% Solution of equation (5.13e) for inductor L(1,2)
L1=(1/w0)*(1/mu + sqrt(1+1/mu^2 ));
L2=(1/w0)*(1/mu - sqrt(1+1/mu^2 ));
% ----------------------------------------------
if L1>0
     La_mu=L1;
end
if L2>0
     La_mu=L2;
end
if teta==-90
     L1=1/w0; L2=1/w0;
     La_mu=1/w0;
end
% ----------------------------------------------
Ca_mu=2*La_mu/(1+w0*w0*La_mu*La_mu);
End
```

Program List 5.5: Main_Alternative_Lowpass_T.m

```
% Main_Alternative_Lowpass_T.m
% Developed by BS Yarman
% December 7, 2018, Vanikoy, Istanbul, Turkey
```

```
% Computations with alternative formulas
% December 7, 2018
% ------------------------------------------------
clc; close all
% Inputs:
teta=input('T-Section Phase Shift Teta in Degree=')
w0=0.5;
% Alternative way to generate inductor L
L=tand(teta/2)/w0;
C=2*L/(1+w0*w0*L*L);
% ------------------------------------------------
w=0;N=1000;w1=0;w2=2;DW=(w2-w1)/N;
FRI(1:(N+1))=zeros;
for j=1:N+1
     WA(j)=w;
% ------------------------------------------------
[S11,S21,RO11,F11,RO21,F21 ] = S_Par_Lowpass_T
( w,L,C );
 F21A(j)=F21;
 RO21A(j)=RO21;
% ------------------------------------------------
 F11A(j)=F11;
 RO11A(j)=RO11;
% ------------------------------------------------
 w=w+DW;
end
% ------------------------------------------------
% Phase of S21
figure
plot(WA,F21A)
title('Phase variation F21A of a Lowpass
T-Section')
legend('Using alternative formulas-F21')
xlabel('Normalized Angular Frequency')
ylabel('Phase of S21')
% Amplitude of S21
figure
plot(WA,RO21A)
title('Amplitude variation RO21A of a Lowpass
T-Section')
legend('Using alternative formulas-RO21')
xlabel('Normalized Angular Frequency')
ylabel('Amplitude of S21')
% ------------------------------------------------
figure
plot(WA,F11A)
title('Phase variation F11A of a Lowpass
T-Section')
legend('Using alternative formulas-F11')
xlabel('Normalized Angular Frequency')
ylabel('Phase of S11')
% Amplitude of S11
figure
```

```
plot(WA,RO11A)
 title('Amplitude variation RO11A of a Lowpass
 T-Section')
legend('Using alternative formulas-RO11')
xlabel('Normalized Angular Frequency')
ylabel('Amplitude of S11')
```

Program List 5.5: Main_Highpass_T_with_Classical_Formulas.m

```
% Main_Highpass_T_with_Classical_Formulas.m
% January 6, 2019
% Developed by BS Yarman, Vanikoy, Istanbul, Turkey
clc, clear
close all
teta=input('Enter positive value for the phase shift
teta=')
w0=1;
[ L,C ] = AlternativeComponents_of_Highpass_T_
Section( w0, teta );
[ La,Ca,C1,C2 ] = mu_Based_Components_of_Highpass_
T( w0, teta )
Error_C=C-Ca
Error_L=L-La
% -----------------------------------------------
w=0;N=5000;w1=0;w2=2;DW=(w2-w1)/N;
FRI(1:(N+1))=zeros;
for j=1:N+1
     WA(j)=w;
% -----------------------------------------------
[ S21,RO21,F21 ] = S_Par_Highpass_T ( w,L,C );
   F21A(j)=F21;
   RO21A(j)=RO21;
% -----------------------------------------------

     w=w+DW;
end
% -----------------------------------------------
% Phase of S21
figure
plot(WA,F21A)
title('Phase variation of a "Highpass T-Section" ')
legend('F21A with classical computations')
xlabel('Normalized Angular Frequency')
ylabel('Phase of S21')
% Amplitude of S21
figure
plot(WA,RO21A)
title('Amplitude variation of a "Highpass
 T-Section" ')
legend('RO21A')
xlabel('Normalized Angular Frequency')
ylabel('Amplitude of S21')
```

Program List 5.6: function mu_Based_Components_of_Highpass_T

```
function [ La,Ca,C1,C2 ] = mu_Based_Components_of_
Highpass_T( w0, teta )
% This function generates the component values for
a lowpass T-Section
% Developed by BS Yarman: December 6, 2018;
  Vanikoy, Istanbul, Turkey
% Developed by BS Yarman. December 7, 2018, Vanikoy,
  Istanbul, Turkey
% Inputs:
%       teta: Positive angle at w0
%       w0: Normalized angular frequency
% Outputs:
%       C: Series arm Capacitor
%       L: Shunt arm Inductor
% -----------------------------------------------
mu=tand(teta);
if teta==90
     C1=1/w0;
     C2=1/w0;
     Ca=1/w0;
end
C1=(1/w0/mu)*(1+sqrt(mu*mu+1));
if C1>0
     Ca=C1;
end
C2=(1/w0/mu)*(1-sqrt(mu*mu+1));
if C2>0
     Ca=C2;
end
La=(1/w0/w0)*(1+w0*w0*Ca*Ca)/2/Ca;

End
```

Program List 5.7: AlternativeComponents_of_Highpass_T_Section

```
function [ L,C ] = AlternativeComponents_of_
Highpass_T_Section( w0, teta )
% This function generates the component
 values for a lowpass T-Section
% Developed by BS Yarman: December 6, 2018;
Vanikoy, Istanbul, Turkey
% Developed by BS Yarman. December 7, 2018,
Vanikoy, Istanbul, Turkey
% Inputs:
%       teta: Positive angle at w0
%       w0: Normalized angular frequency
% Outputs:
%       C: Series arm Capacitor
%       L: Shunt arm Inductor
% -----------------------------------------------
C=(1/w0)/tand(teta/2);
```

```
L=(1/w0/w0)*(1+w0*w0*C*C)/2/C;
end
```

Program List 5.8: Main_Lowpass_PI_Section.m

```
% Main_Lowpass_PI_Section.m
% January 12, 2019
% Developed by BS Yarman, Vanikoy, Istanbul, Turkey
% It should be noted that the formulas used in this program is valid
  for the values of teta between 0 and -180 degree.
% Exact 90 is possible
clc, clear
close all
teta=input('Enter negative value for the phase
 shift teta=')
while teta==0
 stop= 'Attention teta=0. Phase can never be zero
 degree. Therefore, change teta and re-run the
  program again.'
 break

end
while teta==-180
    stop= 'Attention teta=-180. Phase can never be
    -180 degree. Therefore, change teta and re-run
     the program again.'
  break
end
while teta>0
stop1='Attention: teta is positive. For a Lowpass
 Symmetric PI-Section teta must be negative
  quantity.'
stop2='Please enter a negative value for teta and
 re-run the program'
   break
end
% Phase teta is proper. Then, start Computations
if teta>-180
if teta<0
w0=input('Enter the normalized angular frequency
 w0=')
 mu=tand(teta)
 eta=tand(90-teta)
% ------------------------------------------------
[ Ca,La,error ] = eta_Based_Components_of_Lowpass_
PI( w0,teta )
% ------------------------------------------------
w=0;N=1000;w1=0;w2=2;DW=(w2-w1)/N;
FRI(1:(N+1))=zeros;
for j=1:N+1
    WA(j)=w;
% ------------------------------------------------
[ S11,S21,RO11,F11,RO21,F21 ] = S_Par_Lowpass_PI
```

```
( w,La,Ca );
  F21A(j)=F21;
  RO21A(j)=RO21;
% -----------------------------------------------
  w=w+DW;
end
% -----------------------------------------------
% Phase of S21
figure
plot(WA,F21A)
title('Phase variation of a "Lowpass PI-Section"
 with classical formulation')
legend('F21A with classical computations')
xlabel('Normalized Angular Frequency')
ylabel('Phase of S21')
% Amplitude of S21
figure
plot(WA,RO21A)
title('Amplitude variation of a "Lowpass PI-
Section" with classical formulation')
legend('RO21A:Computations with classical
 formulation')
xlabel('Normalized Angular Frequency')
ylabel('Amplitude of S21')
end
end
```

Program List 5.9: function eta_Based_Components_of_

```
Lowpass_PI
function [ C,L,error ] = eta_Based_Components_of_
Lowpass_PI( w0,teta )
% In this function teta must be a negative quantity
% In this function component values of a symmetric
 Lowpass pi is computed
%   Input phase teta is defined as a negative
 quantity
if teta>=0
    stop='Attention: teta is positive. It must be a
     negative quantity'
    C='teta is positive. It must be negative'
    L='teta is positive. It must be negative'
    error='teta is positive. It must be negative'
end
if teta<0
eta=tand(90-teta);
% ------------------------------
C=(1/w0)*(eta + sqrt(1+eta*eta ));
L=2*C/(1+w0*w0*C*C);
C1=1/w0*tand(abs(teta/2));
error=norm(C-C1);
end
% -----------------------------------------------
end
```

Program List 5.10: function S_Par_Lowpass_PI

```
( w,L,C )
function [ S11,S21,RO11,F11,RO21,F21 ] = S_Par_
Lowpass_PI( w,L,C )
% This function generates the S-Parameters of
 Lowpass T Section from the
% computed series arm inductor L and the shunt
 capacitor C
%  Developed by BS Yarman: December 7, 2018,
 Vanikoy, Istanbul
% S11=-((1+w0^2 L^2 )jw0C-2jw0L)/(2(1-w0^2 LC)+jw0
(2L+C-w0^2 L^2 C) )
% S_21=2/(2(1-w0^2 LC)+jw0(2L+C-w0^2 L^2 C) )=2/
(d_r+jd_x )=ρ_21 e^(jθ_21 (w0))
% ---------------------------------------------------
dr=2*(1-w*w*L*C);
dx=w*(2*C+L-w*w*C*C*L);
j=sqrt(-1);
D=dr+j*dx;
N11= j*((1+w*w*C*C)*w*L-2*w*C);
S11=N11/D; R11=real(S11); X11=imag(S11);F11=atan2d
(X11,R11);
S21=2/D;   R21=real(S21); X21=imag(S21);F21=atan2d
(X21,R21);
RO11=abs(S11);
RO21=abs(S21);
  end
```

Program List 5.11: Main_Highpass_PI_Section.m

```
% Main_Highpass_PI_Section.m
% January 18, 2019
% Developed by BS Yarman, Vanikoy, Istanbul, Turkey
% It should be noted that the formulas used in this
program is valid for
% the values of teta between 0 and +180 degree.
% Exact 90 is possible
clc, clear
close all
teta=input('Enter positive value for the phase
shift teta=')
while teta==0
 stop= 'Attention teta=0. Phase can never be zero
 degree. Therefore, change teta and re-run the
 program again.'
    break
 end
  while teta==180
    stop= 'Attention teta=180. Phase can never be
    -180 degree. Therefore, change teta and re-run
    the program again.'
    break
```

```
 end
while teta<0
stop1='Attention: teta is negative. For a Highpass
Symmetric PI-Section teta must be positive
quantity.'
stop2='Please enter a positive value for teta and
re-run the program'
     break
end
% Phase teta is proper. Then, start Computations
if teta<180
if teta>0
w0=input('Enter the normalized angular frequency
w0=')
mu=tand(teta)
eta=tand(90-teta)
% -------------------------------------------------
[ Ca,La,error ] = eta_Based_Components_of_Highpass_
PI( w0,teta )
% -------------------------------------------------
w=0;N=1000;w1=0;w2=2;DW=(w2-w1)/N;
FRI(1:(N+1))=zeros;
for j=1:N+1
     WA(j)=w;
% -------------------------------------------------
[ S11,S21,RO11,F11,RO21,F21 ] = S_Par_Highpass_PI
(w,La,Ca );
   F21A(j)=F21;
   RO21A(j)=RO21;
% -------------------------------------------------
   w=w+DW;
end
% -------------------------------------------------
% Phase of S21
figure
plot(WA,F21A)
title('Phase variation of a "highpass PI-Section"
 with classical formulation')
legend('F21A with classical computations')
xlabel('Normalized Angular Frequency')
ylabel('Phase of S21')
% Amplitude of S21
figure
plot(WA,RO21A)
title('Amplitude variation of a "Highpass
PI-Section" with classical formulation')
legend('RO21A:Computations with classical
formulation')
xlabel('Normalized Angular Frequency')
ylabel('Amplitude of S21')
% -----------------------------------------------------
end
end
```

Program List 5.12: Function eta_Based_Components_of_Highpass_PI

```
function [ C,L,error ] = eta_Based_Components_of
Highpass_PI( w0,teta )
% In this function teta must be a positive quantity
% In this function component values of a symmetric
% Highpass pi is computed
% Input phase teta is defined as a positive quantity
if teta<=0
   stop='Attention: teta is positive. It must be a
   negative quantity'
   C='teta is positive. It must be positive'
   L='teta is positive. It must be positive'
   error='teta is positive. It must be negative'
end
if teta>0
eta=-tand(90+teta);
% -------------------------------
L=(1/w0)*(1/tand(teta/2));
C=(1/w0/w0)*(1+w0*w0*L*L)/2/L;
L1=(1/w0)*(eta + sqrt(1+eta*eta ));
error=norm(L-L1);
end
% ---------------------------------------------------
End
```

Program List 5.13: Function S_Par_Highpass_PI

```
function [ S11,S21,RO11,F11,RO21,F21 ] = S_Par_Highpass_PI( w,L,C )
% This function generates the S-Parameters of
% Lowpass T Section from the
% computed series arm inductor L and the shunt capacitor C
% Developed by BS Yarman: December 7, 2018, Vanikoy, Istanbul
% S11=-((1+w0^2 L^{}2 )jw0C-2j?L)/(2(1-w0^2 LC)+jw0
(2L+C-w0^2 L^2 C) )
% S_21=2/(2(1-w0^2 LC)+jw0(2L+C-w0^2 L^2 C) )=2/
(d_r+jd_x )=ρ_21 e^(jθ_21 (w0))
% ---------------------------------------------------
dr=1-w*w*(2*L*C+L*L);
dx=2*w*L*(1-w*w*L*C);
j=sqrt(-1);
D=dr+j*dx;
N11=-(1+w*w*L*(L-2*C));
S11=N11/D; R11=real(S11); X11=imag(S11);F11=atan2d
(X11,R11);
S21=-j*2*w*w*w*L*L*C/D;   R21=real(S21); X21=imag
(S21);F21=atan2d(X21,R21);
RO11=abs(S11);
RO21=abs(S21);

End
```

Appendix 6: MatLab Programs for Chapter 6

Program List 6.1: Main_Lowpass_TSection_DPS.m

```
% Main Program: Main_Lowpass_TSection_DPS.m
% February 20, 2019
% Developed by BS Yarman, Vanikoy, Istanbul
% Enter Positive value
%-------------------------------------------------------------------
clc; close all
% Inputs:
teta=input('Alternative Formulas to Design Lowpass T Section. Enter
positive value for Teta in Degree=')
w0=1;
% Alternative way to generate inductor L
L=tand(teta/2)/w0;\% C=
2*L/(1+w0*w0*L*L);
%-------------------------------------------------------------------
% Component Values
CD1=1/(w0^2*L);
CD2=CD1;
DEN=2;
Delta=sqrt(C*C+4*C*CD2);
CA=(C)/DEN+Delta/DEN;
CT=CA*CD2/(CA+CD2);
LA=1/w0/w0/CT;
j=sqrt(-1);
w=0;N=1000;w1=0;w2=2;DW=(w2-w1)/N;
FRI(1:(N+1))=zeros;
for i=1:N+1
    WA(i)=w;
%----------------
% State A:
% ----------------
za=j*w*L;
ya=1/j/w/LA+j*w*CA;
 [ S11a,S21a,RO11a,F11a,RO21a,F21a ] = S_Par_LPT_DPS (za,ya);
   F21A(i)=F21a;
   RO21A(i)=RO21a;
   F11A(i)=F11a;
   RO11A(i)=RO11a;
% ------------------
% State-B
% ------------------
  zb=j*w*L+1/j/w0/CD1;
  yb=1/j/w/LA+j*w*CT;
 [ S11b,S21b,RO11b,F11b,RO21b,F21b ] = S_Par_LPT_DPS (zb,yb);
   F21B(i)=F21b;
   RO21B(i)=RO21b;
   F11B(i)=F11b;
   RO11B(i)=RO11b;
%-------------------------------------------------------------------
  w=w+DW;
```

```
end
%-------------------------------------------------------------------
Plot_State_AB_LPT_DPS(WA,F21A,RO21A, F11A,RO11A,F21B,RO21B, F11B,RO11B)
```

Program List 6.2: S_Par_LPT_DPS

```
function [ S11,S21,RO11,F11,RO21,F21 ] = S_Par_LPT_DPS (z,y)
%This function generates the S-Parameters of a Lowpass T Section
% Phase Shifter from the series arm impedance Z(jw) and
% the shunt arm admittance Y(jw)
% computed series arm inductor L and the shunt capacitor C
% Developed by BS Yarman: Feb 20, 2019, Vanikoy, Istanbul
% See Equations (5.9)
%-------------------------------------------------------------------
D=z*z*y+2*z*y+2*z+y+2;
S11=((1-z*z)*y-2*z)/D;
S21=2/D;
R11=real(S11); X11=imag(S11);F11=atan2d(X11,R11);
R21=real(S21); X21=imag(S21);F21=atan2d(X21,R21);
RO11=abs(S11);
RO21=abs(S21);
end
```

Program List 6.3: function Plot_State_AB_LPT_DPS

```
function Plot_State_AB_LPT_DPS(WA,F21A,RO21A, F11A,RO11A,
F21B,RO21B, F11B,RO11B)
figure
plot(WA,F21A,WA,F21B)
title('State A and State B: Phase variations F21A and F21B of
a Lowpass T-Section')
legend('F21A','F21B')
xlabel('Normalized Angular Frequency')
ylabel('Phase of S21A and S21B')
% Amplitude of S21
figure
plot(WA,RO21A,WA,RO21B)
title('State-A and State-B: Amplitude variation RO21A and RO21B of a
Lowpass T-Section')
legend('RO21A','RO21B')
xlabel('Normalized Angular Frequency')
ylabel('Amplitude of S21A and S21B')
%-------------------------------------------------------------------
figure
plot(WA,F11A, WA, F11B)
title('State-A and State-B: Phase variation F11A and F11B of a
Lowpass T-Section')
legend('F11A')
xlabel('Normalized Angular Frequency')
ylabel('Phase of S11A and S11B')
% Amplitude of S11
figure
plot(WA,RO11A, WA, RO11B)
```

```
title('State-A and State-B: Amplitude variation RO11A and RO11B of a
Lowpass T-Section')
legend('RO11A','RO11B')
xlabel('Normalized Angular Frequency')
ylabel('Amplitude of S11A and S11B')

end
```

Program List 6.4: Main Program: Main_Example_6_3.m

```
% Main Program: Main_Example_6_3.m
% February 22, 2019
% Developed by BS Yarman, Vanikoy, Istanbul
% This program evaluates the lossy performance of a LPT-DPS for a
specified
% center actual frequency f0 in Hz.
%-------------------------------------------------------------------
clc; close all
% Inputs:
teta=input('Alternative Formulas to Design Lowpass T Section. Enter
positive value for Teta in Degree=')
% Inputs
f0a=input('Enter the actual center frequency in Hz f0a =')
w0=input('At f0a, enter the normalized angular frequency w0 =')
R=input('Enter the normaliziation Resistor R =')
%-------------------------------------------------------------------
% Compute the normalized element values of LPT-DPS
% Alternative way to generate inductor L
L=tand(abs(teta)/2)/w0;
C=2*L/(1+w0*w0*L*L);
%-------------------------------------------------------------------
% Component Values and their related resistive losses:
CD1=1/(w0^2*L); % See equation (6.2d) of Chapter 6
% ASSUMPTION 1:
% It is assumed that the series loss Ron of a forward biased diode is
% equal to the on channel resistor of an CMOS Switch
% [see Equation (6.1) of Chapter 6].
[ RF1,rf1 ] =Channel_Resistance_of_a_CMOS(CD1,R,f0a);
% ASSUMPTION 2:
% Reverse biased resistive loss of a diode is the "Percent_RVS"
amount of
% its reverse baised impedance at w0
Percent_RVS=100;
rr1=1/w0/CD1/Percent_RVS;
CD2=CD1;
 [ RF2,rf2 ] =Channel_Resistance_of_a_CMOS(CD2,R,f0a);
rr2=1/w0/CD2/Percent_RVS;
%
DEN=2;
Delta=sqrt(C*C+4*C*CD2);
CA=(C)/DEN+Delta/DEN;
CT=CA*CD2/(CA+CD2);
LA=1/w0/w0/CT;
```

```
%-------------------------------------------------------------------
% Loss Computations for both State-A and State-B:
% Assumption 3: Loss of an inductor is Percent_L amount of its impedance
% value at w0.
% Assumption 4: Connectivity loss of an inductor is "Percent_S"amount of
% its impedance value at w0.
% Assumption 5: Conductive loss of a Capacitor is "Percent_C" amount of
its
% admittance value at w0.
Percent_S=100;
Percent_L=10;
Percent_C=100;
%-------------------------------------------------------------------
% Resistive loss of the series arms in State-A:
rL=w0*L/Percent_L;
rs=w0*L/Percent_S;
ra=rL+rf1+rs;
%-------------------------------------------------------------------
% Conductive loss of the Shunt arm in State-A:
rLA=w0*LA/Percent_L;
GCA=w0*CA/Percent_C;
GA=GCA+rLA/(rLA*rLA+w0*w0*LA*LA)+(w0*w0*rf2*CA*CA)
/(1+w0*w0*rf2*rf2*CA*CA);
%-------------------------------------------------------------------
% Resistive loss of the series arms in State-B:
rb=rL+rr1+rs;
%-------------------------------------------------------------------
% Conductive loss of the Shunt arm in State-:
%-------------------------------------------------------------------
GB=GCA+rLA/(rLA*rLA+w0*w0*LA*LA)+(w0*w0*rf2*CT*CT)
/(1+w0*w0*rr2*rr2*CT*CT);
%-------------------------------------------------------------------
j=sqrt(-1);
w=0;N=2000;w1=0;w2=2;DW=(w2-w1)/N;
FRI(1:(N+1))=zeros;
for i=1:N+1
    WA(i)=w;
% ----------------
% State A:
% ----------------
za=ra+j*w*L;
ya=GA+1/j/w/LA+j*w*CA;
 [ S11a,S21a,RO11a,F11a,RO21a,F21a ] = S_Par_LPT_DPS (za,ya);
    F21A(i)=F21a;
    RO21A(i)=20*log10(RO21a);
    F11A(i)=F11a;
    RO11A(i)=20*log10(RO11a);
% ------------------
% State-B
% ------------------
  zb=rb+j*w*L+1/j/w0/CD1;
  yb=GB+1/j/w/LA+j*w*CT;
  [ S11b,S21b,RO11b,F11b,RO21b,F21b ] = S_Par_LPT_DPS (zb,yb);
```

```
    F21B(i)=F21b;
    RO21B(i)=20*log10(RO21b);
    F11B(i)=F11b;
    RO11B(i)=20*log10(RO11b);
%---------------------------------------------------------------------
DEL_FI21(i)=F21A(i)-F21B(i);
    w=w+DW;
 end
%---------------------------------------------------------------------
Plot_State_AB_LPT_DPS(WA,F21A,RO21A, F11A,RO11A,F21B,
RO21B,F11B,RO11B,DEL_FI21)
```

Program List 6.5: Channel_Resistance_of_a_CMOS

```
function [ Ron,ron ] = Channel_Resistance_of_a_CMOS(Coff,R,f0)
% This function generates the on forward biased channel resistance of a
% 0.180um CMOS switch manufactured by TSMC
% February 22, 2019, Vanikoy, Istanbul, Turkey
% Developed by BS Yarman
% Inputs:
% Coff: Normalized value of the Inductor L
% f0=Actual operating frequency (Normalization frequency)
% R: Actual terminatons resistors (Normalization Resistor)
% Output:
% Ron: Actual value of the channel resistor
% ron: Normalized Value of the channel resistor
%---------------------------------------------------------------------
[ Coff_Actual ] =ActualValues_of_a_Capacitor(Coff,R,f0);
% Ron (?) x C_D1 (Farad)=672 x 10^(-15)
Ron=672e-15/Coff_Actual;
ron=Ron/R;

end
```

Program List 6.6: function ActualValues_of_a_Capacitor

```
function [ C_actual ] = ActualValues_of_a_Capacitor(C,R,f0)
% February 22, 2019, Vanikoy, Istanbul, Turkey
% Developed by BS Yarman
% Inputs:
% C: Normalized value of the capacitor C
% f0=Actual operating frequency (Normalization frequency)
% R: Actual terminatons resistors (Normalization Resistor)
% Output:
% C_actual: Actual value of the capacitor
%---------------------------------------------------------------------
C_actual=C/2/pi/f0/R;
end
```

Program List 6.7: function ActualValues_of_an_Inductor

```
function [ L_actual ] =ActualValues_of_an_Inductor(L,R,f0)
```

```
% February 22, 2019, Vanikoy, Istanbul, Turkey
% Developed by BS Yarman
% Inputs:
% L: Normalized value of the Inductor L
% f0=Actual operating frequency (Normalization frequency)
% R: Actual terminatons resistors (Normalization Resistor)
% Output:
% L_actual: Actual value of the Inductor
%-----------------------------------------------------------------------
L_actual=L*R/2/pi/f0;

end
```

Appendix 7: MatLab Programs for Chapter 7

Program List 7.1: Main Program: Main_Lowpass_PI_Section_DPS.m

```
% Main Program: Main_Lowpass_PI_Section_DPS.m
% February 24, 2019
% Developed by BS Yarman, Vanikoy, Istanbul
% Enter Positive value
% ---------------------------------------------------------------------
clc; close all
% Inputs:
teta=input('Alternative Formulas to Design Lowpass T Section. Enter
positive value for Teta in Degree=')
% Inputs w0=1;
ra=0;rb=0; GA=0;GB=0;
% ---------------------------------------------------------------------
% Alternative way to generate inductor L
C=tand(teta/2)/w0;
L=2*C/(1+w0*w0*C*C);
% ---------------------------------------------------------------------
% Component Values
CD1=1/(w0^2*L);
CD2=CD1;
Delta=sqrt(C*C+4*C*CD2);
CA=C/2+Delta/2;
CT=CA*CD2/(CA+CD2);
LA=1/w0/w0/CT;
j=sqrt(-1);
w=0;N=2000;w1=0;w2=2;DW=(w2-w1)/N;
FRI(1:(N+1))=zeros;
for i=1:N+1
    WA(i)=w;
% ----------------
% State A:
% ----------------
za=ra+j*w*L;
ya=GA+1/j/w/LA+j*w*CA;
 [ S11a,S21a,RO11a,F11a,RO21a,F21a ] = S_Par_LP_PI_DPS (za,ya);
   F21A(i)=F21a;
   RO21A(i)=20*log10(RO21a);
```

```
   F11A(i)=F11a;
   RO11A(i)=20*log10(RO11a);
% ------------------
% State-B
% ------------------
  zb=rb+j*w*L+1/j/w0/CD1;
  yb=GB+1/j/w/LA+j*w*CT;
  [ S11b,S21b,RO11b,F11b,RO21b,F21b ] = S_Par_LP_PI_DPS (zb,yb);
    F21B(i)=F21b;
    RO21B(i)=20*log10(RO21b);
    F11B(i)=F11b;
    RO11B(i)=20*log10(RO11b);
% -------------------------------------------------------------------
DEL_FI21(i)=F21A(i)-F21B(i);
    w=w+DW;
end
% -------------------------------------------------------------------
Plot_State_AB_LP_PI_DPS(WA,F21A,RO21A, F11A,RO11A,F21B
RO21B, F11B,RO11B,DEL_FI21)
```

Program List 7.2: function S_Par_LP_PI_DPS

```
function [ S11,S21,RO11,F11,RO21,F21 ] = S_Par_LP_PI_DPS (z,y)
% This function generates the S-Parameters of a Lowpass T Section
% Phase Shifter from the series arm impedance Z(jw) and
% the shunt arm admittance Y(jw)
% computed series arm inductor L and the shunt capacitor C
% Developed by BS Yarman: Feb 20, 2019, Vanikoy, Istanbul
% See Equations (5.9)
% -----------------------------------------------------------------
% D=zy^2+2zy+2y+z+2 of S11 = N11/D and S21=2/D
% N11=z(1-y^2)-2y of S11 = N11/D
D=z*y*y+2*z*y+2*y+z+2;
S11=((1-y*y)*z-2*y)/D;
S21=2/D;
R11=real(S11); X11=imag(S11);F11=atan2d(X11,R11);
R21=real(S21); X21=imag(S21);F21=atan2d(X21,R21);
RO11=abs(S11);
RO21=abs(S21);

end
```

Program List 7.3: Main Program: Main_Example_7_2.m

```
% Main Program: Main_Example_7_2.m
% February 24, 2019
% Developed by BS Yarman, Vanikoy, Istanbul
% This program evaluates the lossy performance of a LPT-DPS for a
  specified
% center actual frequency f0 in Hz.
% ------------------------------------------------------------------
clc; close all
```

```
% Inputs:
teta=input('Alternative Formulas to Design Lowpass T Section. Enter
positive value for Teta in Degree=')
% Inputs
f0a=input('Enter the actual center frequency in Hz f0a =')
w0=input('At f0a, enter the normalized angular frequency w0 =')
R=input('Enter the normaliziation Resistor R =')
% -------------------------------------------------------------------
% Compute the normalized element values of LPT-DPS
% Alternative way to generate inductor L
C=tand(abs(teta)/2)/w0;
L=2*C/(1+w0*w0*C*C);
% --------------------------------------------------------------------
% Component Values and their related resistive losses:
CD1=1/(w0^2*L);  % See equation (6.2d) of Chapter 6
% ASSUMPTION 1:
% It is assumed that the series loss Ron of a forward biased diode is
% equal to the on channel resistor of an CMOS Switch
% [see Equation (6.1) of Chapter 6].
[ RF1,rf1 ] =Channel_Resistance_of_a_CMOS(CD1,R,f0a);
% ASSUMPTION 2:
% Reverse biased resistive loss of a diode is the "CrDEL_D: Cronicel
  Delta"
% amount of its reverse baised impedance at w0.
CrDel_D=100;
rr1=1/w0/CD1/CrDel_D;
CD2=CD1;
[ RF2,rf2 ] =Channel_Resistance_of_a_CMOS(CD2,R,f0a);
rr2=1/w0/CD2/CrDel_D;
%
DEN=2;
Delta=sqrt(C*C+4*C*CD2);
CA=(C)/DEN+Delta/DEN;
CT=CA*CD2/(CA+CD2);
LA=1/w0/w0/CT;
% --------------------------------------------------------------------
% Loss Computations for both State-A and State-B:
% Assumption 3: Loss of an inductor is CrDEL_L amount of its impedance
% value at w0.
% Assumption 4: Connectivity loss of an inductor is "CrDEL_S" amount of
% its impedance value at w0.
% Assumption 5: Conductive loss of a Capacitor is "CrDEL_C" amount of
% its admittance value at w0.
CrDel_S=100;
CrDEL_L=10;
CrDel_C=100;
% ---------------------------------------------------------------------
% Resistive loss of the series arms in State-A:
rL=w0*L/CrDEL_L;
rs=w0*L/CrDel_S;
ra=rL+rf1+rs;
% -------------------------------------------------------------------
% Conductive loss of the Shunt arm in State-A:
```

```
rLA=w0*LA/CrDEL_L;
GCA=w0*CA/CrDel_C;
GA=GCA+rLA/(rLA*rLA+w0*w0*LA*LA)+(w0*w0*rf2*CA*CA)/(1+w0*w0*rf2*rf2
*CA*CA);
% ------------------------------------------------------------------
% Resistive loss of the series arms in State-B: rb=rL+rr1+rs;
% -------------------------------------------------------------------
% Conductive loss of the Shunt arm in State-:
% --------------------------------------------------------------------
GB=GCA+rLA/(rLA*rLA+w0*w0*LA*LA)+(w0*w0*rf2*CT*CT)/(1+w0*w0*rr2*rr2*CT*
    CT);
% -------------------------------------------------------------------
j=sqrt(-1);
w=0;N=2000;w1=0;w2=2;DW=(w2-w1)/N;
FRI(1:(N+1))=zeros;
for i=1:N+1
    WA(i)=w;
% ----------------
% State A:
% ----------------
za=ra+j*w*L;
ya=GA+1/j/w/LA+j*w*CA;
[ S11a,S21a,RO11a,F11a,RO21a,F21a ] = S_Par_LP_PI_DPS (za,ya);
  F21A(i)=F21a; RO21A(i)=20*log10(RO21a);
  F11A(i)=F11a;
  RO11A(i)=20*log10(RO11a);
  % ------------------
  % State-B
  % ------------------
  zb=rb+j*w*L+1/j/w0/CD1;
  yb=GB+1/j/w/LA+j*w*CT;
  [ S11b,S21b,RO11b,F11b,RO21b,F21b ] = S_Par_LP_PI_DPS (zb,yb);
  F21B(i)=F21b;
  RO21B(i)=20*log10(RO21b);
  F11B(i)=F11b;
  RO11B(i)=20*log10(RO11b);
% ---------------------------------------------------------------
DEL_FI21(i)=F21A(i)-F21B(i);
      w=w+DW;
end
% -----------------------------------------------------------------
Plot_State_AB_LP_PI_DPS(WA,F21A,RO21A, F11A,RO11A,F21B,RO21B, F11B,
    RO11B,DEL_FI21)
[CD1a] = ActualValues_of_a_Capacitor(CD1,R,f0a)
[CAa] = ActualValues_of_a_Capacitor(CA,R,f0a)
[LAa] = ActualValues_of_an_Inductor(LA,R,f0a)
[La] = ActualValues_of_an_Inductor(L,R,f0a)
% ------------------------------------------------------------------
ra_actual=ra*R
rb_actual=rb*R
% ------------------------------------------------------------------
GA_actual=GA/R; RA_actual=1/GA_actual
GB_actual=GB/R; RB_actual=1/GB_actual
```

Program List 7.4: function Plot_State_AB_LP_PI_DPS

```
function Plot_State_AB_LP_PI_DPS(WA,F21A,RO21A, F11A,RO11A,F21B,RO21B,
F11B,RO11B,DEL_FI21)
figure
plot(WA,F21A,WA,F21B,WA,DEL_FI21)
title('State A and State B: Phase variations F21A and F21B of a
Lowpass PI-Section') legend('F21A','F21B','DEL-FI21')
xlabel('Normalized Angular Frequency')
ylabel('Phase of S21A and S21B')
% Amplitude of S21
figure
plot(WA,RO21A,WA,RO21B)
title('State-A and State-B: Amplitude variation RO21A and RO21B of a
Lowpass PI-Section')
legend('RO21A in dB','RO21B in dB')
xlabel('Normalized Angular Frequency')
ylabel('Amplitude of S21A and S21B in dB')
%  -------------------------------------------------------------
figure
plot(WA,F11A, WA, F11B)
title('State-A and State-B: Phase variation F11A and F11B of a Lowpass
PI-Section') legend('F11A')
xlabel('Normalized Angular Frequency')
ylabel('Phase of S11A and S11B')
% Amplitude of S11
figure
plot(WA,RO11A, WA, RO11B)
title('State-A and State-B: Amplitude variation RO11A and RO11B of a
Lowpass PI-Section')
legend('RO11A in dB','RO11B in dB')
xlabel('Normalized Angular Frequency')
ylabel('Amplitude of S11A and S11B in dB')

end
```

Appendix 8: MatLab Programs for Chapter 8

Program List 8.1. Main_Highpass_TSection_DPS.m

```
% Main Program: Main_Highpass_TSection_DPS.m
% February 20, 2019
% Developed by BS Yarman, Vanikoy, Istanbul
% Enter Positive value
%----------------------------------------------------------------------
 clc; close all
 % Inputs:
 teta=input('Alternative Formulas to Design Lowpass T Section. Enter
                              positive value for Teta in Degree=')
 % Inputs
 w0=1;
 ra=0;rb=0; GA=0;GB=0;
 %---------------------------------------------------------------------
 % Computation of the ideal component values of an Highpass T-Section
```

```
C=tand(90-teta/2)/w0;
L=(1+w0*w0*C*C)/2/C;
%--------------------------------------------------------------------
% Component Values
CD1=C;
CD2=CD1;
% Delta=1+4*w0*w0*L*CD2
% CA=(1+sqrt(Delta))/2/w0/w0/L
Delta=1+4*w0*w0*L*CD2;
CA=(1+sqrt(Delta))/2/w0/w0/L;
% HPT State-A: D1 is revered biased; D2 is reversed biased
CT=CA*CD2/(CA+CD2);
LA=1/w0/w0/CA;
j=sqrt(-1);
w=0;N=2000;w1=0;w2=2;DW=(w2-w1)/N;
FRI(1:(N+1))=zeros;
for i=1:N+1
    WA(i)=w;
% ----------------
% State A:
% ----------------
za=ra+1/j/w/CD1;
ya=GA+1/j/w/LA+j*w*CT;
[ S11a,S21a,RO11a,F11a,RO21a,F21a ] = S_Par_HPT_DPS ( za,ya );
   F21A(i)=F21a;
   RO21A(i)=20*log10(RO21a);
   F11A(i)=F11a;
   RO11A(i)=20*log10(RO11a);
% ------------------
% State-B
% ------------------
 zb=rb;
 yb=GB+1/j/w/LA+j*w*CA;
 [ S11b,S21b,RO11b,F11b,RO21b,F21b ] = S_Par_HPT_DPS ( zb,yb );
   F21B(i)=F21b;
   RO21B(i)=20*log10(RO21b);
   F11B(i)=F11b;
   RO11B(i)=20*log10(RO11b);
%---------------------------------------------------------------------
DEL_FI21(i)=F21A(i)-F21B(i);
    w=w+DW;
end
%---------------------------------------------------------------------
Plot_State_AB_HPT_DPS(WA,F21A,RO21A, F11A,RO11A,F21B,RO21B, F11B,RO11B
   ,DEL_FI21)
```

Program List 8.2. S_Par_HPT_DPS

```
function [ S11,S21,RO11,F11,RO21,F21 ] = S_Par_HPT_DPS ( z,y )

 % This function generates the S-Parameters ofa Lowpass T Section
 % Phase Shifter from the series arm impedance Z(jw) and
```

```
% the shunt arm admittance Y(jw)
% computed series arm inductor L and the shunt capacitor C
% Developed by BS Yarman: Feb 20, 2019,Vanikoy, Istanbul
% See Equations (5.9)
%------------------------------------------------------------------
D=z*z*y+2*z*y+2*z+y+2;
S11=((1-z*z)*y-2*z)/D;
S21=2/D;
R11=real(S11); X11=imag(S11);F11=atan2d(X11,R11);
R21=real(S21); X21=imag(S21);F21=atan2d(X21,R21);
RO11=abs(S11);
RO21=abs(S21);

 end
```

Program List 8.3. `function Plot_State_AB_HPT_DPS`

```
function Plot_State_AB_HPT_DPS(WA,F21A,RO21A, F11A,RO11A,F21B,RO21B,
    F11B,RO11B,DEL_FI21)
 figure
 plot(WA,F21A,WA,F21B,WA,DEL_FI21)
 title('State A and State B: Phase variations F21A and F21B
                            of a Highpass T-Section')
 legend('F21A','F21B','DEL-FI21')
 xlabel('Normalized Angular Frequency')
 ylabel('Phase of S21A and S21B')
 % Amplitude of S21
 figure
 plot(WA,RO21A,WA,RO21B)
 title('State-A and State-B: Amplitude variation RO21A and RO21B
                            of a Highpass T-Section')
 legend('RO21A in dB','RO21B in dB')
 xlabel('Normalized Angular Frequency')
 ylabel('Amplitude of S21A and S21B in dB')
 %------------------------------------------------------------------
 figure
 plot(WA,F11A, WA, F11B)
 title('State-A and State-B: Phase variation F11A and F11B
                            of a Highpass T-Section')
 legend('F11A')
 xlabel('Normalized Angular Frequency')
 ylabel('Phase of S11A and S11B')
 % Amplitude of S11
 figure
 plot(WA,RO11A, WA, RO11B)
 title('State-A and State-B: Amplitude variation RO11A and RO11B
                            of a Highpass T-Section')
 legend('RO11A in dB','RO11B in dB')
 xlabel('Normalized Angular Frequency')
 ylabel('Amplitude of S11A and S11B in dB')

 end
```

Program List 8.4. Main Program:Main_Example_8_3.m

```
% Main Program: Main_Example_8_3.m
% March 3, 2019
% Developed by BS Yarman, Vanikoy, Istanbul
% This program evaluates the lossy performance of a Highpass
                          T-Section DPS
% for a specified actual center frequency f0 in Hz.
% -------------------------------------------------------------------
clc; close all
% Inputs:
teta=input('Design of a Lossy Highpass T Section DPS. Enter positive
                                       value for Teta in Degree=')
% Inputs
f0a=input('Enter the actual center frequency in Hz f0a =')
w0=input('At f0a, enter the normalized angular frequency w0 =')
R=input('Enter the normaliziation Resistor R=')
%--------------------------------------------------------------------
% Compute the normalized element values of LPT-DPS
% Compute the ideal component values C & L of a higpass T-Section
C=tand(90-teta/2)/w0;
L=(1+w0*w0*C*C)/2/C;
%---------------------------------------------------------------------
% Compute the unknown Component Values
CD1=C;
CD2=CD1;
Delta=1+4*w0*w0*L*CD2;
CA=(1+sqrt(Delta))/2/w0/w0/L;
%
% HPT State-A: D1 is revered biased; D2 is reversed biased
CT=CA*CD2/(CA+CD2);
LA=1/w0/w0/CA;
%------------------------------------------------------------------
% Component Values and their related resistive losses:
CD1=C; % See equation (6.2d) of Chapter 6
% ASSUMPTION 1:
% It is assumed that the series loss Ron of a forward biased diode is
% equal to the on channel resistor of an CMOS Switch
% [see Equation (6.1) of Chapter 6].
[ RF1,rf1 ] =Channel_Resistance_of_a_CMOS( CD1,R,f0a );
% ASSUMPTION 2:
% Reverse biased resistive loss of a diode is the "Percent_RVS"
                         amount of its reverse baised impedance at w0
CrDel_D=100;
rr1=1/w0/CD1/CrDel_D;
CD2=CD1;
[ RF2,rf2 ] =Channel_Resistance_of_a_CMOS( CD2,R,f0a );
rr2=1/w0/CD2/CrDel_D;
%
%--------------------------------------------------------------------
% Loss Computations for both State-A and State-B:
% Assumption 3: Loss of an inductor is Percent_L amount of
its impedance
```

```
% value at w0.
% Assumption 4: Connectivity loss of an inductor is "Percent_S"
                           amount of its impedance value at w0.
% Assumption 5: Conductive loss of a Capacitor is "Percent_C" amount
                                 of its admittance value at w0.
CrDel_S=100;
CrDEL_L=10;
CrDel_C=100;
%--------------------------------------------------------------------
% Resistive loss of the series arms in State-A:
rL=w0*L/CrDEL_L;
rs=w0*L/CrDel_S;
ra=rr1;
%---------------------------------------------------------------------
% Conductive loss of the Shunt arm in State-A:
rLA=w0*LA/CrDEL_L;
GCA=w0*CA/CrDel_C;
GA=GCA+rLA/(rLA*rLA+w0*w0*LA*LA)+(w0*w0*rr2*CT*CT)/(1+w0*w0*rr2*rr2*
CT*CT);
%----------------------------------------------------------------------
% Resistive loss of the series arms in State-B:
rb=rf1;
%----------------------------------------------------------------------
% Conductive loss of the Shunt arm in State-:
%----------------------------------------------------------------------
GB=GCA+rLA/(rLA*rLA+w0*w0*LA*LA)+(w0*w0*rf2*CA*CA)/(1+w0*w0*rf2*rf2
*CA*CA);
%----------------------------------------------------------------------
j=sqrt(-1);
w=0;N=2000;w1=0;w2=2;DW=(w2-w1)/N;
FRI(1:(N+1))=zeros;
for i=1:N+1
    WA(i)=w;
% ----------------
% State A:
% ----------------
za=ra+1/j/w/CD1;

ya=GA+j*w*(CT/(1+w*w*rr2*rr2*CT*CT)-LA/(rLA*rLA+w*w*LA*LA));
[ S11a,S21a,RO11a,F11a,RO21a,F21a ] = S_Par_HPT_DPS ( za,ya );
   F21A(i)=F21a;
   RO21A(i)=20*log10(RO21a);
   F11A(i)=F11a;
   RO11A(i)=20*log10(RO11a);
% ------------------
% State-B
% ------------------
 zb=rb;
 yb=GB++j*w*(CA/(1+w*w*rr2*rr2*CA*CA)-LA/(rLA*rLA+w*w*LA*LA));
 [ S11b,S21b,RO11b,F11b,RO21b,F21b ] = S_Par_HPT_DPS ( zb,yb );
   F21B(i)=F21b;
   RO21B(i)=20*log10(RO21b);
   F11B(i)=F11b;
```

```
   RO11B(i)=20*log10(RO11b);
%----------------------------------------------------------------------
DEL_FI21(i)=F21A(i)-F21B(i);
    w=w+DW;
end
%----------------------------------------------------------------------
Plot_State_AB_HPT_DPS(WA,F21A,RO21A, F11A,RO11A,F21B,RO21B,F11B,RO11B,
   DEL_FI21)
[ CD1a ] = ActualValues_of_a_Capacitor( CD1,R,f0a )
[ CAa ] = ActualValues_of_a_Capacitor( CA,R,f0a )
[ LAa ] =ActualValues_of_an_Inductor( LA,R,f0a )
[ La ] =ActualValues_of_an_Inductor( L,R,f0a )
%----------------------------------------------------------------------
ra_actual=ra*R
rb_actual=rb*R
%----------------------------------------------------------------------
GA_actual=GA/R; RA_actual=1/GA_actual
GB_actual=GB/R; RB_actual=1/GB_actual
```

Program List 8.5. Channel_Resistance_of_a_CMOS

```
function [ Ron,ron ] = Channel_Resistance_of_a_CMOS( Coff,R,f0 )
% This function generates the on forward biased channel
  resistance of a
% 0.180um CMOS switch manufactured by TSMC
% February 22, 2019, Vanikoy, Istanbul, Turkey
% Developed by BS Yarman
% Inputs:
% Coff: Normalized value of the Inductor L
% f0=Actual operating frequency (Normalization frequency)
% R: Actual terminatons resistors (Normalization Resistor)
% Output:
% Ron: Actual value of the channel resistor
% ron: Normalized Value of the channel resistro
%--------------------------------------------------------------------
[ Coff_Actual ] =ActualValues_of_a_Capacitor( Coff,R,f0 );
% Ron (?)×C_D1 (Farad)=672×10^(−15)
Ron=672e-15/Coff_Actual;
ron=Ron/R;

end
```

Program List 8.6. ActualValues_of_a_Capacitor

```
function [ C_actual ] = ActualValues_of_a_Capacitor( C,R,f0 )
% February 22, 2019, Vanikoy, Istanbul,Turkey
% Developed by BS Yarman
% Inputs:
% C: Normalized value of the capacitor C
% f0=Actual operating frequency(Normalization frequency)
% R: Actual terminatons resistors(Normalization Resistor)
% Output:
% C_actual: Actual value of the capacitor
```

```
 %------------------------------------------------------------------
 C_actual=C/2/pi/f0/R;
end
```

Program List 8.7. ActualValues_of_an_Inductor

```
function [ L_actual ]=ActualValues_of_an_Inductor( L,R,f0 )
 % February 22, 2019, Vanikoy, Istanbul,Turkey
 % Developed by BS Yarman
 % Inputs:
 % L: Normalized value of the Inductor L
 % f0=Actual operating frequency (Normalization frequency)
 % R: Actual terminatons resistors (Normalization Resistor)
 % Output:
 % L_actual: Actual value of the Inductor
 %------------------------------------------------------------------
 L_actual=L*R/2/pi/f0;
end
```

Appendix 9: MatLab Program for Chapter 9

In this appendix, we present the MatLab programs specifically developed for the chapter under consideration.

Program List 9.1. Main Program Main_SS-DPS_Example1.m

```
 % Main Program Main_SS-DPS_Example1.m
 % January 20, 2018, Vanikoy
 % Developed by BS Yarman
 % This program computes the element values of SS-DPS for D_Teta=45
   degree
 % at w0=1.
 % Inputs:
 %   Del_Teta_w0
 % Output:
 %   Del_Teta_w
 % -----------------------------------------------------------------
 clc; close all;clear
 Del_Teta_w0=input('Del_Teta_w0 in degree=')
 N=1001;
 w1=1e-8;w2=2; DW=(w2-w1)/(N-1);
 w=w1;
 % Component values:
 % -----------State-A:--------------------------------------------
 Lp1=tand(Del_Teta_w0/4)
 Cp2=Lp1
 % -----------State-B:--------------------------------------------
 Cp1=1/tand(Del_Teta_w0/4)
 Lp2=Cp1
 % -----------------------------------------------------------------
 %
 for i=1:N
```

```
    W(i)=w;
    Teta_B(i)=2*atand(1/(w*Cp1));
    Teta_A(i)=-2*atand(w*Lp1);
    Del_Teta_w=2*(atand(1/(w*Cp1))+atand(w*Lp1));
    Del_Teta(i)=Del_Teta_w;
    w=w+DW;
end
figure
plot(W,Del_Teta)
title ('Phase Variation of SS-DPS')
xlabel('Normalized Angular Frequency w')
ylabel('Teta-B-Teta-A)')
% -------------------------------------------------------------------
figure
plot(W,Teta_B, W,Teta_A)
legend('Teta-B','Teta-A')
title('Teta-B and Teta-A versus W')
xlabel('Normalized Angular Frequency')
ylabel('Teta-B, Teta-A')
```

Program List 9.2. Main Program Main_SS_DPS_Example2.m

```
% Main Program Main_SS_DPS_Example2.m
% January 21, 2018, Vanikoy
% Developed by BS Yarman
% This program computes the normalized element values of SS-DPS
% for D_Teta=45,90,135 and 180 and plots the results
% -------------------------------------------------------------------
clc; close all;clear
w0=1;
Del_Teta_1=45;Del_Teta_2=90; Del_Teta_3=180; Del_Teta_4=135;
[ Lp145,Cp145,Lp245,Cp245,W,Del_Teta_45 ] = SS_DPS_Net_Phase_Shift(w0,
   Del_Teta_1);
[ Lp190,Cp190,Lp290,Cp2,W,Del_Teta_90 ] = SS_DPS_Net_Phase_Shift(w0,
   Del_Teta_2);
[ Lp1180,Cp1180,Lp2180,Cp2180,W,Del_Teta_180 ] =
   SS_DPS_Net_Phase_Shift(w0,Del_Teta_3);
[ Lp1135,Cp1135,Lp2135,Cp2135,W,Del_Teta_135 ] =
   SS_DPS_Net_Phase_Shift(w0,Del_Teta_4);
figure
plot(W,Del_Teta_45,W,Del_Teta_90,W,Del_Teta_135,W,Del_Teta_180);
legend('SS-DPS 45','SS-DPS 90','SS-DPS135','SS-DPS 180')
%-------------------------------------------------------------------
z(1)=abs((45-50.29)/45)*100; y(1)=50.29;
z(2)=abs((90-97.15)/90)*100; y(2)=97.15;
z(3)=abs((135-139.8)/135)*100;y(3)=139.8;
z(4)=abs((180-180)/180)*100; y(4)=180;
Teta(1)=45;Teta(2)=90;Teta(3)=135;Teta(4)=180;
x=Teta;
%-------------------------------------------------------------------
figure
plot(Teta,z)
title('Phase Perturbation eps-teta versus Teta over an octave
```

```
                                                        0.6<w<1.2')
xlabel('Teta in Degree')
ylabel('Perturbation in Percent (%)')
%-------------------------------------------------------------------
% 3D Plot of Perturbation
figure
plot3(x,y,z)
```

Program List 9.3. Main Program:Main_SS_DPS_Example3.m

```
% Main Program: Main_SS_DPS_Example3.m
% January 25, 2018, Vanikoy
% Developed by BS Yarman
% This program computes the scattering parameters for Example 3.
%
% -----------------------------------------------------------------
clc; close all;clear
% Basic Design
Teta1=45; foa=10e9;R=100
mu1=tand(Teta1/2);
L1=mu1;C1=L1;
Teta2=135;
mu2=tand(Teta2/2);
C2=1/mu2;
L2=C2;
%----------------
% Actual Element Values
R=100; f0a=10e9;
[ L1a ] = Actual_Inductor(L1,R,f0a);
[ L2a ] = Actual_Inductor(L2,R,f0a);
[ C1a ] = Actual_Capacitor(C1,R,f0a);
[ C2a ] = Actual_Capacitor(C2,R,f0a);
Lagging_Section=[L1a C1a]
Leading_Section=[L2a C2a]
```

Program List 9.4. Main Program:Main_SS_DPS_Example4.m

```
% Main Program: Main_SS_DPS_Example4.m
clc; clear; close all
% This program developed by BS Yarman, January 30, 2018, Vanikoy
% -----------------------------------------------------------------
% Inputs:
% Enter target phase shift:
Teta=45;
% For the 0.18u process, select the optimum OFF mode capacitor Coff1
   for S1
Coff1a=90e-15;
% Select normalization resistor R
R=100;
% Specify the actual center frequency
f0a=10e9;
% Enter vector K=[k2 k4]; where k2 is the multiplier for Coff2a and
% k4 is the multiplier for Coff4a
```

```
K=[0.9 0.6];
k2=K(1); k4=K(2);
% ------------------------------------------------------------------
% This program implements the practical design algorithm.
% Part I: Basic Symmetrical Lagging & Leading Section (SLLS) Design:
[ L1,C1, L1a,C1a,L2,C2,L2a,C2a ] = Basic_SLLS(Teta,f0a,R);
% ------------------------------------------------------------------
% Part II: Design of NMOS switches for 0.18u process technology
%
% Design of S1:
% Select optimim choice for Coff1a:
[ Coff1, Ron1, Ron1a ] = NMOS_Switch_Design(Coff1a,R,f0a);
% Design of S2:
Coff2a_max=C2a; Coff2a=k2*Coff2a_max;
[ Coff2, Ron2, Ron2a ] = NMOS_Switch_Design(Coff2a,R,f0a);
% Design Design of S3:
Coff3a=Coff1a;
[ Coff3, Ron3, Ron3a ] = NMOS_Switch_Design(Coff3a,R,f0a);
% Design of S4:
Coff4a_max=C1a; Coff4a=k4*Coff4a_max;
[ Coff4, Ron4, Ron4a ] = NMOS_Switch_Design(Coff4a,R,f0a);
% Part III: Computation of the elements of SSS-DPS
Lp1=L1/(1+L1*Coff1);
Cp1=C2-Coff2;
%
Lp2=L2/(1+L2*Coff3);
Cp2=C1-k4*Coff4;
% Part IV: Computations of actual elements
[ Lp1a ] = Actual_Inductor(Lp1,R,f0a);
[ Lp2a ] = Actual_Inductor(Lp2,R,f0a);
%
[ Cp1a ] = Actual_Capacitor(Cp1,R,f0a);
[ Cp2a ] = Actual_Capacitor(Cp2,R,f0a);
Normalized_Component_Values= [Lp1 Cp1 Lp2 Cp2]
Actual_Component_Values=[Lp1a Cp1a Lp2a Cp2a]
```

Program List 9.5. function Basic_SLLS

```
function [ L1,C1, L1a,C1a,L2,C2,L2a,C2a ] = Basic_SLLS(Teta,f0a,R)
% This function designs a basic symmetrical Lagging & Leading Lattice
   Section
%   Developed by BS Yarman January 30, 2018
% Basic SLLS Design
%   Inputs:
%       Teta: Target Phase Shift in Degree
%       f0a:  Actual center frequency of the design
%       R:    Normalization resistor (It may be chosen as 100 ohms)
%   Output:
%       L1,C1, L2, C2:  Normalized element values of Basic SLLS
%       L1a,C1a,L2a,C2a:Actal element values of Basic SLLS
mu=tand(Teta/4);
% Normalized element values
L1=mu;C1=L1;
```

```
C2=1/L1;L2=C2;
%----------------
% Actual Element Values
[ L1a ] = Actual_Inductor(L1,R,f0a);
[ L2a ] = Actual_Inductor(L2,R,f0a);
[ C1a ] = Actual_Capacitor(C1,R,f0a);
[ C2a ] = Actual_Capacitor(C2,R,f0a);
End
```

Program List 9.6. function NMOS_Switch_Design

```
function [ Coffn, Ron, Rona ] = NMOS_Switch_Design(Coffa,R,f0a)
% This function designs an NMOS switch with normalized element values
% Develeoped by BS Yarman, January 29, 2018
%   Inputs:
%       Coffa: optimum value of the OFF State Capacitor of NMOS
        in Farad
%       R: Normalization Resistor (For symmetrical Lattice it is
        100 ohms)
%       f0a: Actual Center Frequency
%   Outputs:
%       Coffn: Normalized OFF State Capacitor
%       Ron: Normalized ON State resistor
%       Rona: Actual ON state resistor.
% Step 1: Normalization
Coffn=2*pi*f0a*R*Coffa;
% Computation of the normalized value of the channel resistor Ron1
% when NMOS is ON state
Rona=672*1e-15/Coffa;
Ron=672*1e-15/Coffa/R;

end
```

Program List 9.7. function Actual_Inductor

```
function [ La ] = Actual_Inductor(Ln,R,f0a)
La=Ln*R/2/pi/f0a;

end
```

Program List 9.8. function Actual_Capacitor

```
function [ Ca ] = Actual_Capacitor(Cn,R,f0a)
Ca=Cn/R/2/pi/f0a;

end
```

Program List 9.9. Main Program:Main_SS_DPS_Example5.m

```
% Main Program: Main_SS_DPS_Example5.m
clc; clear;close all
% This program developed by BS Yarman, February 5, 2018, Vanikoy
```

```
% ----------------------------------------------------------------
% Inputs:
% Enter target phase shift:
Teta=45;
% For the 0.18u process, select the optimum OFF mode capacitor Coff1
   for S1
Coff_opt=90e-15;
% Select normalization resistor R
R=100;
% Specify the actual center frequency
f0a=10e9;
% Enter vector K=[k2 k4]; where k2 is the multiplier for Coff2a and
% k4 is the multiplier for Coff4a
K=[1 1];
k2=K(1); k4=K(2);
% ----------------------------------------------------------------
[ NCV,ACV,Ron,Coffn,Rona,Coffa ] = SSS_DPS_Design(Teta, R, f0a,
   Coff_opt,K)

Ron1=Ron(1); Ron4=Ron(4); Lp1=NCV(1); Cp1=NCV(2); Lp2=NCV(3); Cp2=NCV
   (4);
Coff2=Coffn(2); Coff3=Coffn(3);
% ----------------------------------------------------------------
w1=0; w2=2;N=101; dw=(w2-w1)/(N-1);
w=w1;
for i=1:N
    W(i)=w;
[ FI21B,ILB,VSWR ] = SSS_DPS_State_B(w,Ron1,Ron4,Lp1,Lp2,Cp1,Cp2,
                     Coff2,Coff3);
Phase_B(i)=FI21B;
Insertion_Loss_B(i)=ILB;
w=w+dw;
end
figure
plot(W,Phase_B)
xlabel('Normalized Angular Frequency w')
ylabel(' Phase of State-B in Degree')
title('Phase versus normalized angular frequency for State-B')
figure
plot(W,Insertion_Loss_B)
xlabel('Normalized Angular Frequency w')
ylabel(' Gain of State-B in Degree')
title('Gain versus normalized angular frequency for State-B')
```

Program List 9.10. function SSS_DPS_State_B

```
function [ FI21B,ILB,VSWR ] = SSS_DPS_State_B(w,Ron1,Ron4,Lp1,Lp2,Cp1,
   Cp2,Coff2,Coff3)
% Simple Single Symmetrical Lattice Digital Phase Shifter in State-B
% Developed by BS Yarman, February 4, 2018, Vanikoy
% Computations for State-B: SSS Lattice is a Leading Symmetrical
   Section
% % ----------------------------------------------------------------
```

```
%   Inputs:
%       Lp1,Lp2
%       Cp1,Cp2
%       Ron1, Ron4 (Normalized values)
%       Coff2,Coff3 (Normalized values)
%   Output:
%       FI21B: Phase of S21B in degree
%       ILB:   Insertion Loss in dB. ILB=20log10 ( |S21B| ) in dB
% -----------------------------------------------------------------
DaB=Ron1*Ron1+w*w*Lp1*Lp1;
RaB=w*w*Ron1*Lp1/DaB;
XaB=w*Ron1*Ron1*Lp1/DaB-1/w/(Cp1+Coff2);
zaB=complex(RaB,XaB);
%
DbB=1+(w*Ron4*Cp2)*(w*Ron4*Cp2);
RbB=Ron4/DbB;
XbB=Lp2/(1-w*w*Lp2*Coff3)-w*Ron4*Ron4*Cp2/DbB;
zbB=complex(RbB,XbB);
% -----------------------------------------------------------------
[ FI21B,VSWR,ILB ] = S_Par_SLS(zaB,zbB);

end
```

Program List 9.11. S_Par_SLS

```
function [ FI21B,VSWR,ILB ] = S_Par_SLS(zaB,zbB)
% This function generates the scattering parameters of a symmetrical
% Lattice defined by means of its series and cross arm impedances zaB
   and
% zbB
% This function is developed by BS Yarman, Feb 5, 2018
%   Inputs:
%       Complex series arm impedance zaB
%       Complex cross arm impedance zbB
%   Output:
%       FI21B: Phase of S21B
%       ILB: 20log10( |S21B| ) insertion loss in dB
%       VSWR: Voltage Standing Wave Ratio
% -----------------------------------------------------------------
S11B=(zaB*zbB-1.0)/(zaB*zbB+zaB+zbB+1.0);
S21B=(zbB-zaB)/(zaB*zbB+zaB+zbB+1.0); R21B=real(S21B);X21B=imag(S21B);
ro21B=abs(S21B);
ILB=20*log10(ro21B);
FI21B=atan2d(X21B,R21B);
ro11B=abs(S11B);
VSWR=(1+ro11B)/(1-ro11B);

end
```

Program List 9.12. function SSS_DPS_Component_Values

```
function [ NCV,ACV,RON,COFF, RONA, COFFA  ] = SSS_DPS_Component_Values
   (Teta,R, Coffa,f0a)
```

```
% This function generates the estimated component values of an SSS-DPS
% Unit.
% In this function, Cp1 & Cp2 "ComponentValue-Control vector
  K=[k2 k4]"
% is automatically generated from the selected optimum off state
  capacitor
% Coff_opt and from the basic component values of basic symmetrical
% LC sections. That is from C1 and C2.k2=Coff_opt/C2, k4=Coff_opt/C1
  and
% Cp1=C2-k2*C2=C2(1-k2), Cp2=C1(1-k4).
% In this function the purpose is that "to the optimum value of switch
% capacitors when possible.
% Developed By B.S. Yarman, January 29, 2018
% Inputs:
% Teta: Target phase Shift in degree at w0=1.
% R : Normalization resistor for S-Parameters. It may be R=100 ohm.
% Coff: Optimum OFF state capacitor of NMOS switch in Farad
% fo0 : Actual Center Frequency in Hz.
% K : OFF State Capacitor Control Vector with two enrees K=[k2 k4]
% Outputs:
% (Lp1,Cp1): Series arm inductor and capacitor
% (Lp2,Cp2): Cross arm inductor and capacitor
% NCV: Normalized Component Values as a vector [Lp1, Cp1, Lp2, Cp2]
%
% ----------------------------------------------------------------
% Part I: Define Basic SSS-DPS Cell
[ L1,C1,L2,C2 ] = Basic_SSS_DPS_Design(Teta);
 % Compute the normalized value of the actual optimum OFF state
                              capacitor
Coffn=2*pi*f0a*Coffa*R;
 %----------------------------------------------------------------
 % Part II: Generate the switch parameters for the optimum values of
 % OFF state capacitor Coff.
 % k2=K(1); k4=K(2);
 % ---- Design Switch S1:------------------------------------------
[ Coff1, Ron1, ~ ] = NMOS_Switch_Design(Coffa,R,f0a);
 % ---- Switch S2:-------------------------------------------------
Coff2_max=C2;
 % Compute the control number k2 automatically
k2=Coffn/Coff2_max;
Coff2=k2*Coff2_max;
[ Coff2a ] = Actual_Capacitor(Coff2,R,f0a);
[ ~, Ron2, ~ ] = NMOS_Switch_Design(Coff2a,R,f0a);
% ---- Design Switch S3: ----------------------------
Coff3=Coff1;
Ron3=Ron1;
 % ---- Design Switch S4: ---------
Coff4_max=C1;
k4=Coffn/Coff4_max;
Coff4=k4*Coff4_max;
[ Coff4a ] = Actual_Capacitor(Coff4,R,f0a);
[ ~, Ron4, ~ ] = NMOS_Switch_Design(Coff4a,R,f0a);
 %----------------------------------------------------------------
```

```
 % Part III: Compute the series armcomponents values:
Cp1=C2-Coff2;
 % Check S2 if it is okay:
 if Cp1<0
  % k2=1; % if Cp1 is negative, then set C2=Coff2, k2=1 case.
    Cp1=0;
    Coff2=C2;
    [ Coff2a ] = Actual_Capacitor(Coff2,R,f0a);
    [ Coff2, Ron2, Ron2a ] = NMOS_Switch_Design(Coff2a,R,f0a);
 end
Lp1=L1/(1+L1*Coff1);
 % Part IV: Compute the cross arm component values
Cp2=C1-Coff4;
 if Cp2<0
     % k4=1; % if Cp2 is negative, then set C1=Coff4, k4=1 case.
    Cp2=0;
    Coff4=C1;
    [ Coff4a ] = Actual_Capacitor(Coff4,R,f0a);
    [ Coff4, Ron4, Ron4a ] = NMOS_Switch_Design(Coff4a,R,f0a);
 end
Lp2=L2/(1+L2*Coff3);
 % Computation of Actual Compenet Values ACV
[ Lp1a ] = Actual_Inductor(Lp1,R,f0a);
[ Cp1a ] = Actual_Capacitor(Cp1,R,f0a);
 %
[ Lp2a ] = Actual_Inductor(Lp2,R,f0a);
[ Cp2a ] = Actual_Capacitor(Cp2,R,f0a);
 %-----------------------------------------------------------------
NCV=[Lp1 Cp1 Lp2 Cp2];
ACV=[Lp1a Cp1a Lp2a Cp2a];
 %-----------------------------------------------------------------
% Generate the ON state resistor vector:
RON=[Ron1 Ron2 Ron3 Ron4]; RONA=R*RON;
COFF=[Coff1 Coff2 Coff3 Coff4]; COFFA= Actual_Capacitor(COFF,R,f0a);

 end
```

Program List 9.13. function UnevenTeta_Basic_SLLS

```
 function [ L1,C1, L1a,C1a,L2,C2,L2a,C2a ] = UnevenTeta_Basic_SLLS(
   Del_Teta,TetaB,f0a,R)
 % This function designa a basic symmetrical Lagging & Leading
   Lattice Section
 % Developed by BS Yarman Feb 15, 2018
 % Basic SLLS Design
 % Inputs:
    % Del_Teta: Target Phase Shift in Degree
    % TetaB: Phase Shift of Leading Symmetrical Section
    % f0a: Actual center frequency of the design
    % R: Normalization resistor (It may be chosen as 100 ohms)
 % Output:
     % L1,C1, L2, C2: Normalized element values of Basic SLLS
     % L1a,C1a,L2a,C2a:Actal element values of Basic SLLS
```

```
%-----------------------------------------------------------------
% Note: TetaA is the absolute value of the phase of "State-A".
% Stae-A phase is the lagging state which yields a negative phase.
% Teta=TetaB+TetaA. Therefore, it is given by
% TetaA=Teta-TetaB where TetaB is a positive leading phase.
% TetaA/2 and TetaB/2 must vary between 0 and 90 degree. Therefore
  TetaA &
% TetaB must be less than 180 degree.
%-----------------------------------------------------------------
% Step 1: For Specified Del_Teta and TetaB Determine TetaA
TetaA=Del_Teta-TetaB;
% Step 2: Generate Major Design Parameters (MDP) for the Basic
  Design muA=tand(TetaA/2);
muB=tand(TetaB/2);
% Step 3: Compute the Normalized Component Values (NCV) of the
  Basic Design
L1=muA;C1=L1;
C2=1/muB;L2=C2;
%------------------------------------------------------------------
% Step 3: Compute the Actaul Component Values (ACV) of the Basic
                    Design
[ L1a ] = Actual_Inductor( L1,R,f0a);
[ L2a ] = Actual_Inductor( L2,R,f0a);
[ C1a ] = Actual_Capacitor(C1,R,f0a);
[ C2a ] = Actual_Capacitor(C2,R,f0a);
%-----------------------------------------------------------------

end
```

Program List 9.14. function UnevenPhase_DPS_Component_Values

```
function [ Lp1,Cp1,Lp2,Cp2,RON,COFF,RONA, COFFA  ] =
   UnevenPhase_DPS_Component_Values(Teta,TetaB, R, Coffa,f0a)
% This function generates the estimatedcomponent values of an SSS-DPS
% Unit.
% Developed By B.S. Yarman, Feb 7, 2018
% Inputs:
   % Teta: Target phase Shift in degree at w0=1.
   % TetaB: Un-evenly distributed phase of state-B at w=1
   % TetaA: Unevenly distributed phase of State-A. TetaA=Teta-TetaB
   % R: Normalization resistor for S-Parameters. It may be R=100 ohm.
   % Coffa: Actual Optimum value of the "OFF state capacitor" of NMOS
     switch in Farad
   % f0a : Actual Center Frequency in Hz.
% K : OFF State Capacitor Control Vector with two entrees K=[k2 k4]
% Note: k2=1, k4=1 corresponds to the ideal values of switch
     capacitor.
% Ideal situation may yield big values. Therefore, k2 and k2
     must be equal or
% less than 1.Cp1=C2-k2*Coff_max>0.k2 may be selected as
     k2=C2/Coff_opt
% Similarly Cp2=C1-k4*Coff4_max,Coff4_max=C1. k4=Coff_opt/Coff4_max
%
```

```
 % Outputs:
     % (Lp1,Cp1): Series arm inductor and capacitor
     % (Lp2,Cp2): Cross arm inductor and capacitor
 %-----------------------------------------------------------------
 % Part I: Define Basic SSS-DPS Cell
 [ L1,C1,~,~,L2,C2,~,~ ]=UnevenTeta_Basic_SLLS(Teta,TetaB,f0a,R);
  % Compute the normalized value of the optimum OFF state capacitance
  of
  % NMOS
 Coffn=2*pi*f0a*Coffa*R;
 %-----------------------------------------------------------------
 % Part II: Generate the switch parameters for the optimumvalues of
 % OFF state capacitor Coff.
 % k2=K(1); k4=K(2);
 % ---- Design Switch S1:---------------------------------------
Coff1a=Coffa;
[ Coff1, Ron1, Ron1a ] = NMOS_Switch_Design(Coff1a,R,f0a);
 % ---- Design Switch S2:---------------------------------------
 % Set the maximum OFF state capacitance ofS2:
Coff2_max=C2;
k2=Coffn/Coff2_max;
Coff2=k2*Coff2_max;
[ Coff2a ] = Actual_Capacitor(Coff2,R,f0a);
[ ~, Ron2, Ron2a ] = NMOS_Switch_Design(Coff2a,R,f0a);
 % ---- Design Switch S3:--------------------------------------
Coff3=Coff1;
Coff3a=Coff1a;
Ron3=Ron1;
Ron3a=Ron1a;
% ---- Design Switch S4:---------------------------------------
Coff4_max=C1;
k4=Coffn/Coff4_max;
Coff4=k4*Coff4_max;
[ Coff4a ] = Actual_Capacitor(Coff4,R,f0a);
Ron4=672*1e-15/Coff4a/R;
[ Coff2, Ron4, Ron4a ] = NMOS_Switch_Design(Coff4a,R,f0a);
 %-----------------------------------------------------------------
 % Part III: Compute the series arm components values:
Cp1=C2-Coff2;
 if Cp1<0
    Cp1=0;k2=0;end
Lp1=L1/(1+L1*Coff1);
% Part IV: Compute the cross arm component values
Cp2=C1-Coff4;
 if Cp2<0
    Cp2=0; k4=0; end
Lp2=L2/(1+L2* Coff3);
%-----------------------------------------------------------------
% Generate the ON state resistor vector:
RON=[Ron1 Ron2 Ron3 Ron4];
RONA=[Ron1a Ron2a Ron3a Ron4a];
COFF=[Coff1 Coff2 Coff3 Coff4];
COFFA=[Coff1a Coff2a Coff3a Coff4a];
 end
```

```
% Main Program: Main_SS_DPS_Example6.m

clc; clear;close all
% This program is developed by BS Yarman, February 7, 2018, Vanikoy
% ---------------------------------------------------------------
% Inputs:
% Enter target phase shift:
Del_Teta=input('Enter Del_Teta=')
% For the 0.18u process, select the optimum OFF mode capacitor Coff1
for S1
Coff_opt=90e-15;
% Select normalization resistor R
R=100;
% ---------------------------------------------------------------
TetaB=input('Enter TetaB=')
% Specify the actual center frequency
f0a=input('Enter Actual Center Frequency f0a=')
%
% Computational Steps:
% Step 1: Design SSS_DPS for unevenly distributed phase-shift
  Del_Teta,
% TetaB and TetaA. User specify Del_Teta and TetaB. TetaA=Del_TetaB>0
%
[ Lp1,Cp1,Lp2,Cp2,Ron,Coffn,RONA, COFFA  ] =
   UnevenPhase_DPS_Component_Values(Del_Teta,TetaB, R, Coff_opt,f0a)
%
Ron1=Ron(1); Ron2=Ron(2); Ron3=Ron(3); Ron4=Ron(4);
Coff1=Coffn(1); Coff2=Coffn(2); Coff3=Coffn(3); Coff4=Coffn(4);
% ---------------------------------------------------------------
w1=1e-9; w2=2;N=10001; dw=(w2-w1)/(N-1);
w=w1;
for i=1:N
    W(i)=w;
[ FI21A,GainA,VSWRA ] = SSS_DPS_State_A(w,Ron2,Ron3,Lp1,Lp2,Cp1,Cp2,
   Coff1,Coff4);
[ FI21B,GainB,VSWRB ] = SSS_DPS_State_B(w,Ron1,Ron4,Lp1,Lp2,Cp1,Cp2,
   Coff2,Coff3);
Phase_A(i)=FI21A;
Phase_B(i)=FI21B;
GainA_dB(i)=GainA;
GainB_dB(i)=GainB;
Phase_Shift(i)=FI21B-FI21A;
w=w+dw;
end
Plot_3S_DPS(W,Phase_A,Phase_B,Phase_Shift,GainA_dB,GainB_dB)
NCV=[Lp1 Cp1 Lp2 Cp2]
[ Lp1a ] = Actual_Inductor(Lp1,R,f0a);
[ Cp1a ] = Actual_Capacitor(Cp1,R,f0a);
[ Lp2a ] = Actual_Inductor(Lp2,R,f0a);
[ Cp2a ] = Actual_Capacitor(Cp2,R,f0a);
L_3SDPS=[Lp1a Lp2a]
C_3SDPS=[Cp1a Cp2a]
```

Program List 9.15. Main Program:Main_SSS_DPS_Example7.m

```
% Main Program: Main_SSS_DPS_Example7.m
clc; clear;close all
% This program is developed by BS Yarman, February 15, 2018, Vanikoy
% ------------------------------------------------------------------
% Inputs:
% Enter target phase shift:
Del_Teta=input('Enter a positive phase-shift for Del_Teta=')
% For the 0.18u process, select the optimum OFF mode capacitor
Coff1 for S1 Coff_opt=90e-15;
% Select normalization resistor R
R=100;
%-----------------------------------------------------------------
TetaB=input('Enter a positive phase-shift for TetaB=')
% Specify the actual center frequency
w0=input('Enter Center Frequency w0=')
f0a=input('Enter Actual Center Frequency f0a=')
%
% Computational Steps:
% Step 1: Design SSS_DPS for unevenly distributed phase-shift Del_Teta,
% TetaB and TetaA. User specify Del_Teta and TetaB. TetaA=TetaA=
   Del_Teta-TetaB>0
%
[ Lp1,Cp1,Lp2,Cp2,Ron,Coffn,RONA, COFFA  ] =
   UnevenPhase_DPS_Component_Values(w0, Del_Teta,TetaB, R, Coff_opt,
   f0a)
%
Ron1=Ron(1); Ron2=Ron(2); Ron3=Ron(3); Ron4=Ron(4);
Coff1=Coffn(1); Coff2=Coffn(2); Coff3=Coffn(3); Coff4=Coffn(4);
% ------------------------------------------------------------------
w1=-5; w2=5;N=10001; dw=(w2-w1)/(N-1);
w=w1;
for i=1:N
    W(i)=w;
[ FI21A,GainA,VSWRA ] = SSS_DPS_State_A(w,Ron2,Ron3,Lp1,Lp2,Cp1,Cp2,
   Coff1,Coff4);
[ FI21B,GainB,VSWRB ] = SSS_DPS_State_B(w,Ron1,Ron4,Lp1,Lp2,Cp1,Cp2,
   Coff2,Coff3);
Phase_A(i)=FI21A;
Phase_B(i)=FI21B;
GainA_dB(i)=GainA;
GainB_dB(i)=GainB;
Phase_Shift(i)=FI21B-FI21A;
w=w+dw;
end
Plot_3S_DPS(W,Phase_A,Phase_B,Phase_Shift,GainA_dB,GainB_dB)
NCV=[Lp1 Cp1 Lp2 Cp2]
[ Lp1a ] = Actual_Inductor(Lp1,R,f0a);
[ Cp1a ] = Actual_Capacitor(Cp1,R,f0a);
[ Lp2a ] = Actual_Inductor(Lp2,R,f0a);
[ Cp2a ] = Actual_Capacitor(Cp2,R,f0a);
L_3SDPS=[Lp1a Lp2a]
C_3SDPS=[Cp1a Cp2a]
```

Program List 9.16. Main Program:Main_SSS_DPS_Example7.m

```
% Main Program: Main_SSS_DPS_Example7.m
clc; clear;close all
% This program is developed by BS Yarman,February 15, 2018, Vanikoy
%---------------------------------------------------------------------
% Inputs:
% Enter target phase shift:
Del_Teta=input('Enter a positive phase-shift for Del_Teta=')
% For the 0.18u process, select the optimumOFF mode capacitor
  Coff1 for S1
Coff_opt=90e-15;
% Select normalization resistor R
R=100;
%---------------------------------------------------------------------
TetaB=input('Enter a positive phase-shift for TetaB=')
% Specify the actual center frequency
w0=input('Enter Center Frequency w0=')
f0a=input('Enter Actual Center Frequency f0a=')
%
% Computational Steps:
% Step 1: Design SSS_DPS for unevenly distributed phase-shift Del_Teta,
% TetaB and TetaA. User specify Del_Teta and TetaB.
  TetaA=TetaA=Del_Teta-TetaB>0
%
[ Lp1,Cp1,Lp2,Cp2,Ron,Coffn,RONA, COFFA  ] =
    UnevenPhase_DPS_Component_Values(w0, Del_Teta,TetaB, R, Coff_opt,
    f0a)
%
Ron1=Ron(1); Ron2=Ron(2); Ron3=Ron(3); Ron4=Ron(4);
Coff1=Coffn(1); Coff2=Coffn(2); Coff3=Coffn(3); Coff4=Coffn(4);
%---------------------------------------------------------------------
w1=-5; w2=5;N=10001; dw=(w2-w1)/(N-1);
w=w1;
for i=1:N
    W(i)=w;
[ FI21A,GainA,VSWRA ] = SSS_DPS_State_A(w,Ron2,Ron3,Lp1,Lp2,Cp1,Cp2,
    Coff1,Coff4);
[ FI21B,GainB,VSWRB ] = SSS_DPS_State_B(w,Ron1,Ron4,Lp1,Lp2,Cp1,Cp2,
    Coff2,Coff3);
Phase_A(i)=FI21A;
Phase_B(i)=FI21B;
GainA_dB(i)=GainA;
GainB_dB(i)=GainB;
Phase_Shift(i)=FI21B-FI21A;
w=w+dw;
end
Plot_3S_DPS(W,Phase_A,Phase_B,Phase_Shift,GainA_dB,GainB_dB)
NCV=[Lp1 Cp1 Lp2 Cp2]
[ Lp1a ] = Actual_Inductor(Lp1,R,f0a);
[ Cp1a ] = Actual_Capacitor(Cp1,R,f0a);
[ Lp2a ] = Actual_Inductor(Lp2,R,f0a);
[ Cp2a ] = Actual_Capacitor(Cp2,R,f0a);
```

```
L_3SDPS=[Lp1a Lp2a]
C_3SDPS=[Cp1a Cp2a]
```

Program List 9.17. Main Program:Main_SSS_DPS_Example6.m

```
% Main Program: Main_SSS_DPS_Example6.m
clc; clear;close all
% This program is developed by BS Yarman,February 15, 2018, Vanikoy
%-----------------------------------------------------------------
% Inputs:
% Enter target phase shift:
Del_Teta=input('Enter Del_Teta=')
% For the 0.18u process, select the optimum OFF mode capacitor
  Coff1 for S1
Coff_opt=90e-15;
% Select normalization resistor R
R=100;
%-----------------------------------------------------------------
TetaB=input('Enter TetaB=')
% Specify the normalized and actual center frequency
w0=input('Enter w0=')
f0a=input('Enter Actual Center Frequency f0a=')
%
% Computational Steps:
% Step 1: Design SSS_DPS for unevenly distributed phase-shift Del_Teta,
% TetaB and TetaA. User specify Del_Teta and TetaB. TetaA=Del_TetaB>0
%
[ Lp1,Cp1,Lp2,Cp2,Ron,Coffn,RONA, COFFA  ] =
   UnevenPhase_DPS_Component_Values(w0, Del_Teta,TetaB, R, Coff_opt,
   f0a);
%
Ron1=Ron(1); Ron2=Ron(2); Ron3=Ron(3); Ron4=Ron(4);
Coff1=Coffn(1); Coff2=Coffn(2); Coff3=Coffn(3); Coff4=Coffn(4);
% ----------------------------------------------------------------
w1=-5; w2=5;N=10001; dw=(w2-w1)/(N-1);
w=w1;
for i=1:N
    W(i)=w;
[ FI21A,GainA,VSWRA ] = SSS_DPS_State_A(w,Ron2,Ron3,Lp1,Lp2,Cp1,Cp2,
   Coff1,Coff4);
[ FI21B,GainB,VSWRB ] = SSS_DPS_State_B(w,Ron1,Ron4,Lp1,Lp2,Cp1,Cp2,
   Coff2,Coff3);
Phase_A(i)=FI21A;
Phase_B(i)=FI21B;
GainA_dB(i)=GainA;
GainB_dB(i)=GainB;
Phase_Shift(i)=FI21B-FI21A;
w=w+dw;
end
Plot_3S_DPS(W,Phase_A,Phase_B,Phase_Shift,GainA_dB,GainB_dB)
NCV=[Lp1 Cp1 Lp2 Cp2]
[ Lp1a ] = Actual_Inductor(Lp1,R,f0a);
[ Cp1a ] = Actual_Capacitor(Cp1,R,f0a);
[ Lp2a ] = Actual_Inductor(Lp2,R,f0a);
```

```
[ Cp2a ] = Actual_Capacitor(Cp2,R,f0a);
L_3SDPS=[Lp1a Lp2a]
C_3SDPS=[Cp1a Cp2a]
```

Program List 9.18. Main_Negative_FI21B.m

```
% Main_Negative_FI21B.m
% This program generates the componentvalues of a 3S-DPS employing
% lossy switches. Therefore, effect of thelossy switches is
  minimized on
% the component values.
% This program is developed by BS Yarman,on April 10, 2018, Vanikoy,
% Istanbul.
% Algorithm to design 3S-DPS with arbitrary selection of
% State-B Phase TetaB in degree
% Inputs:
% w0: Normalized angular frequency,
% TetaA=FI21A(w0): Phase of State-A at w=w0,in degree
% TetaB=FI21B(w0): Phase of State-B at w=w0,in degree
% C_off1a: Off-Mode Capacitor of Switch 1 (S1,)
% C_off2a: Off-Mode Capacitor of Switch 2 (S2),
% C_off4a: Off-Mode Capacitor of Switch 4 (S4),
% R_a: Actual Normalization Resistor
% Algorithm to design 3S-DPS with arbitrary selection of State-B
  Phase ?_B
% Inputs:
% w0: Normalized angular frequency,
% TetaA=FI21A(w0): Phase of State-A at w=w0,
% TetaB=FI21B(w0): Phase of State-B at w=w0,
% C_off1a: Off-Mode Capacitor of Switch 1 (S1),
% C_off2a: Off-Mode Capacitor of Switch 2 (S2),
% C_off4a: Off-Mode Capacitor of Switch 4 (S4),
% R_a: Actual Normalization Resistor
%-----------------------------------------------------------------
clc, clear, close all
%-----------------------------------------------------------------
TetaB=input('Enter Negative values for TetaB in degree=')
TetaA=input('Enter positive value of TetaA in degree=')
f0a=input('Enter Actual Center Freuency f0a=')
w0=input('Enter Normalized Angular Center Frequency w0=')
Coffa=input('Enter Coffa=')
Coff1a=Coffa;
Coff2a=Coffa;
Coff3a=Coffa;
Coff4a=Coffa;
Ra=100;
%-----------------------------------------------------------------
% Computational Steps:
% Step-1: Normalized the actual capacitances
wa=2*pi*f0a;
Coff1=wa*Ra*Coff1a;
Coff2=wa*Ra*Coff2a;
Coff3=wa*Ra*Coff3a;
```

```
Coff4=wa*Ra*Coff4a;
%-----------------------------------------------------------------
% Step-2: Compute the actual ON-State channel resistors and
% their normalized values
Ron1a=672e-15/Coff1a;
Ron2a=672e-15/Coff2a;
Ron3a=672e-15/Coff3a;
Ron4a=672e-15/Coff4a;
% Normalized on-channel resistors
Ron1=Ron1a/Ra;
Ron2=Ron2a/Ra;
Ron3=Ron3a/Ra;
Ron4=Ron4a/Ra;
%-----------------------------------------------------------------
% Step-3: Design of Reference State-A using Lagging Section
  (L1 and C1).
% Compute the major design parameters:muA,L1, Lp1 and Cp2 as follows.
  muA=tand(TetaA/2);
L1=muA/w0;
C1=L1;
Lp1=L1/(1+w0*w0*L1*Coff1);
%-----------------------------------------------------------------
% Step-4: Compute RaB and beta as follows.
RaBD=Ron1*Ron1+w0*w0*Lp1*Lp1;
RaB=(w0*w0*Ron1*Lp1*Lp1)/RaBD;
beta=(w0*Ron1*Ron1*Lp1)/RaBD;
%-----------------------------------------------------------------
% Step-5: Solve equation (57)to determine XaB
gammaB=tand(TetaB);
% It should be noted that tand(FI21)=tand[FI21(+/-)180)].
  Therefore, solution
% Xab may yield either FI21B or FI21B (+/-)180.
% gammaB=abs(gammaB);
Discriminant=1-gammaB*gammaB*(RaB*RaB-1);
XaB1=(1+sqrt(Discriminant))/(gammaB);
XaB2=(1-sqrt(Discriminant))/(gammaB);
if XaB1<0;XaB=XaB1;end
if XaB2<0;XaB=XaB2;end
%-----------------------------------------------------------------
% Step-6: Compute Cp1 as in (63)
Cp1=1/w0/(beta-XaB)-Coff2;
if Cp1<0; Cp1=0; C2=Coff2;end
% Design Switch 2:
if Cp1==0
Coff2a=Coff2/2/pi/f0a/Ra;
Ron2a=672e-15/Coff2a;
Ron2=Ron2a/Ra;
end
%-----------------------------------------------------------------
% Step 7: Compute RbB and XbB as in (64)
DenRaB=RaB*RaB+XaB*XaB;
RbB=RaB/DenRaB;
XbB=-XaB/DenRaB;
```

```
%------------------------------------------------------------------
% Step 8a: Compute Cp2 as in (65c)
Cp2a=(1/w0/Ron4)*sqrt((Ron4-RbB)/RbB);
% Step 8b: Check if Cp2 is negative
if (Ron4-RbB)<0
    Cp2a=0;
end
if Cp2a<0
    Cp2a=0;
end
if Cp2a==0
    Coff4=C1;
    Coff4a=Coff4/2/pi/f0a/Ra;
    Ron4a=672e-15/Coff4a;
    Ron4=Ron4a/Ra;
end
%------------------------------------------------------------------
% Step 8b
Cp2b=C1-Coff4;
if Cp2b<0
    Cp2b=0;
end
if Cp2b==0
    Coff4=C1;
    Coff4a=Coff4/2/pi/f0a/Ra;
    Ron4a=672e-15/Coff4a;
    Ron4=Ron4a/Ra;
end
%------------------------------------------------------------------
Cp2=(Cp2b+Cp2a)/2;
%------------------------------------------------------------------
% Step 9: Compute Lp2
Lp2=XbB/(1+w0*XbB*Coff3);
%------------------------------------------------------------------
% Step 10: Compute the actual component values.
[ Lp1a ] = Actual_Inductor(Lp1,Ra,f0a);
[ Cp1a ] = Actual_Capacitor(Cp1,Ra,f0a);
[ Lp2a ] = Actual_Inductor(Lp2,Ra,f0a);
[ Cp2a ] = Actual_Capacitor(Cp2,Ra,f0a);
%
%------------------------------------------------------------------
% Step 11: Normalized and actual component values of 3S-DPS in
 vector form.
NCV=[Lp1 Cp1 Lp2 Cp2]
L_3SDPS=[Lp1a Lp2a]
C_3SDPS=[Cp1a Cp2a]
%------------------------------------------------------------------
% Step 12: Plot the results
w1=-5; w2=5;N=10001; dw=(w2-w1)/(N-1);
w=w1;
for i=1:N
    W(i)=w;
```

```
[ FI21A,GainA,VSWRA ] = SSS_DPS_State_A(w,Ron2,Ron3,Lp1,Lp2,Cp1,Cp2,
    Coff1,Coff4);
[ FI21B,GainB,VSWRB ] = SSS_DPS_State_B(w,Ron1,Ron4,Lp1,Lp2,Cp1,Cp2,
    Coff2,Coff3);
Phase_A(i)=FI21A;
Phase_B(i)=FI21B;
GainA_dB(i)=GainA;
GainB_dB(i)=GainB;
Phase_Shift(i)=FI21B-FI21A;
w=w+dw;
end
Plot_3S_DPS(W,Phase_A,Phase_B,Phase_Shift,GainA_dB,GainB_dB)
%-----------------------------------------------------------------
```

Program List 9.19. Main_SSS_DPS_Example7C.m

```
% Main_SSS_DPS_Example7C.m
% This program generates the component values of a 3S-DPS employing
    the
% lossy switches for Example 7. Therefore, effect of the lossy
    switches is
% minimized on the computations.
% This program is developed by BS Yarman, on April 12, 2018, Vanikoy,
% Istanbul.
% Algorithm 4 to design 3S-DPS with arbitrary selection of
% State-B Phase at w0 is designated by TetaB=FI21B(w0)>0 which is
    positive
% in degree
% Inputs:
% w0: Normalized angular frequency,
% TetaA=FI21A(w0)>0: Phase of State-A at w=w0,in degree
% TetaB=FI21B(w0)>0: Phase of State-B at w=w0,in degree
% C_off1a: Off-Mode Actual Capacitor of Switch 1 (S1),
% C_off2a: Off-Mode Actual Capacitor of Switch 2 (S2),
% C_off4a: Off-Mode Actual Capacitor of Switch 4 (S4),
% Ra=100 ohm: Actual Normalization Resistor
% Algorithm-4 to design 3S-DPS with arbitrary selection of State-B
    Phase
% FI21B(w0)=TetaB>0
% ---------------------------------------------------------------
clc, clear, close all
% ---------------------------------------------------------------
TetaB=-10
TetaA=55
f0a=8e9
w0=1.
%
% Note: In this program KFLAG=-1 results in good design. In other
    words,
% XaB is negative. gammaB=tand(TetaB)>0
KFLAG=+1
Coff1a=25e-15;
Coff2a=90e-15;
```

```
Coff3a=90e-15;
Coff4a=90e-15;
Ra=100;
% ------------------------------------------------------------------
% Computational Steps:
% Step-1: Normalized the actual capacitances
wa=2*pi*f0a;
Coff1=wa*Ra*Coff1a;
Coff2=wa*Ra*Coff2a;
Coff3=wa*Ra*Coff3a;
Coff4=wa*Ra*Coff4a;
% ------------------------------------------------------------------
% Step-2a: Compute the actual ON-State channel resistors and
% their normalized values
Ron1a=672e-15/Coff1a;
Ron2a=672e-15/Coff2a;
Ron3a=672e-15/Coff3a;
Ron4a=672e-15/Coff4a;
% Step 2b: Normalized on-channel resistors
Ron1=Ron1a/Ra;
Ron2=Ron2a/Ra;
Ron3=Ron3a/Ra;
Ron4=Ron4a/Ra;
% ------------------------------------------------------------------
% Step-3: Design of Reference State-A using Lagging Section
          (L1 and C1).
% Compute the major design parameters: muA,L1,C1 and Lp1 as follows.
muA=tand(TetaA/2);
L1=muA/w0;
C1=L1;
Lp1=L1/(1+w0*w0*L1*Coff1);
% ------------------------------------------------------------------
% Step-4: Compute RaB and beta as follows.
RaBD=Ron1*Ron1+w0*w0*Lp1*Lp1;
RaB=(w0*w0*Ron1*Lp1*Lp1)/RaBD;
beta=(w0*Ron1*Ron1*Lp1)/RaBD;
% ------------------------------------------------------------------
% Step-5: Solve equation (9.61)to determine XaB
% Note that TetaB=FI21B(w0)
% Step 5a: Compute gammaB
gammaB=tand(TetaB);
% It should be noted that tand(FI21)=tand[FI21(+/-)180)]. Therefore,
   solution
% Xab may yield either FI21B or FI21B (+/-)180.
% gammaB=abs(gammaB);
% Step 5b: Discriminant
Discriminant=1-gammaB*gammaB*(RaB*RaB-1);
% Step 5c: Compute XaB1
XaB1=(1+sqrt(Discriminant))/(gammaB);
% Step 5d: Compute Xab2
XaB2=(1-sqrt(Discriminant))/(gammaB);
% ------------------------------------------------------------------
% Step 5e:  Select negative XaB<0
```

```
if KFLAG==-1
if XaB1<0;XaB=XaB1;end
if XaB2<0;XaB=XaB2;end
end
% ----------------------------------------------------------------
% Computations with positive XaB>0
if KFLAG==+1
if XaB1>0;XaB=XaB1;end
ifxsxs XaB2>0;XaB=XaB2;end
end
% ----------------------------------------------------------------
% Step-6: Compute Cp1 as in (9.63)
Cp1=1/w0/(beta-XaB)-Coff2
if Cp1<0; Cp1=0;
C2=Coff2;end
% Design Switch 2:
if Cp1==0
Coff2a=Coff2/2/pi/f0a/Ra;
Ron2a=672e-15/Coff2a;
Ron2=Ron2a/Ra;
end
% ----------------------------------------------------------------
% Step 7: Compute RbB and XbB as in (9.64)
DenRaB=RaB*RaB+XaB*XaB;
% Step 7a: Compute RbB
RbB=RaB/DenRaB;
% Step 7b: Compute XbB
XbB1=-XaB/DenRaB
XbB2=-1/XaB
XbB=input('Enter XbB=')
% ----------------------------------------------------------------
% Step 8a:  Compute Cp2 as in (9.65c)
Cp2_I=(1/w0/Ron4)*sqrt((Ron4-RbB)/RbB)
% Step 8b:  Check if Cp2 is negative
if (Ron4-RbB)<0;Cp2_I=0;end
if Cp2_I<0;Cp2_I=0;end
% ----------------------------------------------------------------
% Step 8b
Cp2_II=C1-Coff4
if
Cp2_II<0;Cp2_II=0;end
% if Cp2b==0
%     Coff4=C1;
%     Coff4a=Coff4/2/pi/f0a/Ra;
%     Ron4a=672e-15/Coff4a;
%     Ron4=Ron4a/Ra;
% end
% ----------------------------------------------------------------
Cp2=input('Enter Cp2=')
if Cp2==0
    Coff4=C1;
    Coff4a=Coff4/2/pi/f0a/Ra;
    Ron4a=672e-15/Coff4a;
```

```
    Ron4=Ron4a/Ra;
end
% ----------------------------------------------------------------
% Step 9: Compute Lp2
Lp2=XbB/(1+w0*XbB*Coff3);
% ----------------------------------------------------------------
% Step 10: Compute the actual component values.
[ Lp1a ] = Actual_Inductor(Lp1,Ra,f0a);
[ Cp1a ] = Actual_Capacitor(Cp1,Ra,f0a);
[ Lp2a ] = Actual_Inductor(Lp2,Ra,f0a);
[ Cp2a ] = Actual_Capacitor(Cp2,Ra,f0a);
%
% ----------------------------------------------------------------
% Step 11: Normalized and actual component values of 3S-DPS in vector
   form.
NCV=[Lp1 Cp1 Lp2 Cp2]
L_3SDPS=[Lp1a Lp2a]
C_3SDPS=[Cp1a Cp2a]
% ----------------------------------------------------------------
% Step 12: Plot the results
w1=0; w2=2;N=10001; dw=(w2-w1)/(N-1);
w=w1;
for i=1:N
    W(i)=w;
[ FI21A,GainA,VSWRA ] = SSS_DPS_State_A(w,Ron2,Ron3,Lp1,Lp2,Cp1,Cp2,
   Coff1,Coff4);
[ FI21B,GainB,VSWRB ] = SSS_DPS_State_B(w,Ron1,Ron4,Lp1,Lp2,Cp1,Cp2,
   Coff2,Coff3);
Phase_A(i)=FI21A;
Phase_B(i)=FI21B;
GainA_dB(i)=GainA;
GainB_dB(i)=GainB;
Phase_Shift(i)=FI21B-FI21A;
w=w+dw;
end
Plot_3S_DPS(W,Phase_A,Phase_B,Phase_Shift,GainA_dB,GainB_dB)
% ----------------------------------------------------------------
```

Program List 9.20. Main_SSS_DPS_Example8.m

```
% Main_SSS_DPS_Example8.m
% This program is written by BS Yarman on April 28, 2018
% Vanikoy, Istanbul
clc,close all
% Inputs:
Ra=100; % Actual Normalization Resistor (ANR)
f0a=8e9, % Actual Center Frequency (ACF) in Hertz.
w0=1.0, % Normalized Angular Center Frequency (NACF)
% ------------------------------------------------------
Coff1a=25e-15, % Actual OFF-MODE Capacitor of S1 in Farad
Coff2a=90e-15, % Actual OFF-MODE Capacitor of S2 in Farad
Coff3a=90e-15, % Actual OFF-MODE Capacitor of S3 in Farad
Coff4a=90e-15, % Actual OFF-MODE Capacitor of S4 in Farad
```

```
% -------------------------------------------------------
TetaA=55, % Phase of State-A: FI21A=-TetaA
TetaB=10, % Phase of State-B: FI21B=-TetaB
% Del_Teta=FI21B-FI21A=-35-(-55)=-35+80
% -----------------------------------------------------------------
% Step 1: Compute muA and muB:
 muA=tand(TetaA/2)
 muB=tand(TetaB/2)
% -----------------------------------------------------------------
% Step 2: Compute the normalized value of Coff1 and Ron1
% ----------------------------------
Coff1=2*pi*f0a*Ra*Coff1a
Ron1a=672e-15/Coff1a, Ron1=Ron1a/Ra
% ----------------------------------
Coff2=2*pi*f0a*Ra*Coff2a
Ron2a=672e-15/Coff2a, Ron2=Ron2a/Ra
% -----------------------------------
Coff3=2*pi*f0a*Ra*Coff3a
Ron3a=672e-15/Coff1a, Ron3=Ron3a/Ra
% -----------------------------------
Coff4=2*pi*f0a*Ra*Coff4a
Ron4a=672e-15/Coff4a, Ron4=Ron4a/Ra
% -----------------------------------
% Step 3: Compute Lp1
Lp1=muA/(1+w0*w0*muA*Coff1)
% Step 4: Compute eta=w0*Ron1*Ron1*Lp1/(Ron1*Ron1+Lp1*Lp1)
eta=w0*Ron1*Ron1*Lp1/(Ron1*Ron1+Lp1*Lp1)
% Step 5: Check if eta>muB
if eta>muB
    attention='Design Parameters are GOOD'
end
    if eta<muB
    attention='Design Parameters are NO GOOD. Go back to Input-step
    and reduce coff1a or increase w0'
    end

if eta>muB
% Step 6: Compute the imaginary part X_bB of the cross-arm impedance
   Z_bB as in (9.66g)
% and compute  the imaginary part X_aB of the series-arm impedance
   Z_aB as in (9.66k).
XbB=-1/muB
XaB=-1/XbB
% Step 7: Compute the cross-arm inductor Lp2 as in (9.66i)and series-
   arm
% capacitors Cp1:
% Step 7a: Cross-Arm Inductors Lp2:
Lp2=XbB/(1+w0*XbB*Coff3)
% Check if Lp2 is positive. If not re-design S3
if Lp2<0
    Coff3_max=1/w0/abs(XbB)
    Coff3=input('Re-Design S3 Coff3=')
    Coff3a=Coff3/2/pi/f0a/Ra
```

```
    Ron3a=672e-15/Coff3a
    Ron3=Ron3a/Ra
end
% Step 7b: Compute Series-Arm capacitors Cp1:
Cp1=1/w0/(eta-muB)-Coff2
% Check if Cp1 is positive. If not Set Cp1=0 and re-design S2
if Cp1<0
    Cp1=0
    Cp1a=0
    Attention='Cp1 is negative. Therefore S2 is re-designed
    Coff2=1/w0/(eta-XaB)'
    Coff2=1/w0/(eta-XaB)
    Coff2a=Coff2/2/pi/f0a/Ra
    Ron2a=672e-15/Coff2a
    Ron2=Ron2a/Ra
end
% Step 8: Compute the realizable value of the cross-arm capacitor Cp2
Cp2=muA-Coff4
if Cp2<0
    Cp2=0
    Cp2a=0
    Attention='Cp2 is negative. Therefore S4 is re-designed Coff4=muA'
    Coff4_max=muA
    Coff4=input('Enter new normalized value for Coff4=')
    Coff4a=Coff4/2/pi/f0a/Ra
Ron4a=672e-15/Coff4a, Ron4=Ron4a/Ra
end
% ----------------------------------------------------------------
% Step 9: Electric Performance Analysis
w1=0; w2=2;N=10001; dw=(w2-w1)/(N-1);
w=w1;
for i=1:N
    W(i)=w;
    Fa(i)=w*f0a;
[ FI21A,GainA,VSWRA ] = SSS_DPS_State_A(w,Ron2,Ron3,Lp1,Lp2,Cp1,Cp2,
   Coff1,Coff4);
[ FI21B,GainB,VSWRB ] = SSS_DPS_State_B(w,Ron1,Ron4,Lp1,Lp2,Cp1,Cp2,
   Coff2,Coff3);
Phase_A(i)=FI21A;
Phase_B(i)=FI21B;
GainA_dB(i)=GainA;
GainB_dB(i)=GainB;
Phase_Shift(i)=FI21B-FI21A;
w=w+dw;
end
% ----------------------------------------------------------------
Plot_3S_DPS(W,Phase_A,Phase_B,Phase_Shift,GainA_dB,GainB_dB)
figure
plot(Fa,Phase_Shift)
title('DEL-FI=45 at F=8 GHz')
xlabel('Actual Frequencies')
ylabel('Phase-Shift=FI21B-FI21A')
legend('Del-FI=45 Degree')
```

```
% ----------------------------------------------------------------
figure
plot(Fa,GainB_dB,Fa,GainA_dB)
title('DEL-FI=45 at F=8 GHz')
legend('GainB','GainA')
xlabel('Actual Frequencies')
ylabel('GainB and GainA')
% ----------------------------------------------------------------

NCV=[Lp1 Cp1 Lp2 Cp2]
[ Lp1a ] = Actual_Inductor(Lp1,Ra,f0a);
[ Cp1a ] = Actual_Capacitor(Cp1,Ra,f0a);
[ Lp2a ] = Actual_Inductor(Lp2,Ra,f0a);
[ Cp2a ] = Actual_Capacitor(Cp2,Ra,f0a);
L_3SDPS=[Lp1a Lp2a]
C_3SDPS=[Cp1a Cp2a]
% ----------------------------------------------------------------
end
% ----------------------------------------------------------------
```

Appendix 10: Program Lists for Chapter 10

Program List 10.1. Main_Example_10_1.m

```
% Main Program: Main_Example_10_1.m
% March 5, 2019
% Developed by BS Yarman, Vanikoy, Istanbul
% This program evaluates the lossy performance of a Highpass T-Section
DPS
% for a specified actual center frequency f0 in Hz.
%-------------------------------------------------------------------
clc; close all
% Inputs:
Teta_A=input('Design of 360 Degree T Section DPS. Enter Phase of
State-A (Lowpass-T): Teta-A in Degree=')
Teta_B=input('Design of 360 Degree T Section DPS. Enter Phase of
    State-B(Highpass-T): Teta-B in Degree=')
% Inputs
f0a=input('Enter the actual center frequency in Hz f0a =')
w0=input('At f0a, enter the normalized angular frequency w0 =')
R=input('Enter the normaliziation Resistor R =')
%------------------------------------------------------------------
% Compute the normalized element values of T-360 Degree-DPS
LL=tand(abs(Teta_A)/2)/w0;
CL=(2*LL)/(1+w0*w0*LL*LL);
%--------------------------------------------------------------------
%Ideal Highpass T-Section: See Equations (10.2a) and (10.2b)
CH=tand(90-Teta_B/2)/w0;
LH=(1+w0*w0*CH*CH)/2/CH/w0/w0;
```

```
[ CD1,L1,LA,CA,CT,CAa,LAa,CD1a,L1a,CTa ] = T360_DPS(Teta_A, Teta_B, f0a
    ,w0,R);
%-----------------------------------------------------------------------
j=sqrt(-1);
w=0;N=2000;w1=0;w2=5;DW=(w2-w1)/N;
FRI(1:(N+1))=zeros;
WA(1:(N+1))=zeros;
DEL_FI21(1:N+1)=zeros;
for i=1:N+1
WA(i)=w;
% ----------------
% State A:
% ----------------
za=j*w*L1;
ya=j*w*CA+1/j/w/LA;
[ S11a,S21a,RO11a,F11a,RO21a,F21a ] = S_Par_T_Section (za,ya);
    F21A(i)=F21a;
    RO21A(i)=20*log10(RO21a);
    F11A(i)=F11a;
    RO11A(i)=20*log10(RO11a);
% ------------------
% State-B
% ------------------
zb=j*w*L1+1/j/w/CD1;
yb=j*w*CT+1/j/w/LA;
   [ S11b,S21b,RO11b,F11b,RO21b,F21b ] = S_Par_T_Section (zb,yb);
   F21B(i)=F21b;
   RO21B(i)=20*log10(RO21b);
   F11B(i)=F11b;
   RO11B(i)=20*log10(RO11b);
% ---------------------------------------------------------------------
 DEL_FI21(i)=F21B(i)-F21A(i);
     w=w+DW;
end
% --------------------------------------------------------------------
Plot_State_AB_T360_DPS(WA,F21A,RO21A, F11A,RO11A,F21B,RO21B, F11B,RO11B
    ,DEL_FI21)
```

Program List 10.2. Main_Example_10_2.m

```
% Main Program: Main_Example_10_2.m
% March 3, 2019
% Developed by BS Yarman, Vanikoy, Istanbul
% This program evaluates the lossy performance of a Highpass T-Section
DPS
% for a specified actual center frequency f0 in Hz.
% ----------------------------------------------------------------------
clc; close all
% Inputs:
Teta_A=input('Design of 360 Degree T Section DPS. Enter Phase of
State-A(Lowpass-T): Teta-A in Degree=')
Teta_B=input('Design of 360 Degree T Section DPS. Enter Phase of
State-B(Highpass-T): Teta-B in Degree=')
```

```
% Inputs
f0a=input('Enter the actual center frequency in Hz f0a =')
w0=input('At f0a, enter the normalized angular frequency w0 =')
R=input('Enter the normaliziation Resistor R =')
% ---------------------------------------------------------------------
% Compute the normalized element values of T-360 Degree-DPS
[ CD1,L1,LA,CA,CT,CAa,LAa,CD1a,L1a,CTa ] = T360_DPS(Teta_A, Teta_B, f0a
    ,w0,R)
% ---------------------------------------------------------------------
% ASSUMPTION 1:
% It is assumed that the series loss Ron of a forward biased diode is
% equal to the on channel resistor of an CMOS Switch
% [see Equation (6.1) of Chapter 6].
[ RF1,rf1 ] =Channel_Resistance_of_a_CMOS(CD1,R,f0a);
% ASSUMPTION 2:
% Reverse biased resistive loss of a diode is the "Percent_RVS" amount
of
% its reverse baised impedance at w0
CrDel_D=100;
rr1=1/w0/CD1/CrDel_D;
CD2=CD1;
[ RF2,rf2 ] =Channel_Resistance_of_a_CMOS(CD2,R,f0a);
rr2=1/w0/CD2/CrDel_D;
%
% ---------------------------------------------------------------------
% Loss Computations for both State-A and State-B:
% Assumption 3: Loss of an inductor is Percent_L amount of its impedance
% value at w0.
% Assumption 4: Connectivity loss of an inductor is "Percent_S"
amount of
% its impedance value at w0.
% Assumption 5: Conductive loss of a Capacitor is "Percent_C" amount
of its
% admittance value at w0.
CrDel_S=100;
CrDEL_L=10;
CrDel_C=100;
% ---------------------------------------------------------------------
% Resistive loss of the series arms in State-A:
rL1=w0*L1/CrDEL_L;
rLA=w0*LA/CrDEL_L;
rs=w0*L1/CrDel_S;
ra=rf1;
% ---------------------------------------------------------------------
% Conductive loss of the Shunt arm in State-A:
GCA=w0*CA/CrDel_C;
% GCA=0;GA=0;
GA=GCA+rLA/(rLA*rLA+w0*w0*LA*LA);
% ---------------------------------------------------------------------
% Resistive loss of the series arms in State-B:
rb=rr1;
% ---------------------------------------------------------------------
% Conductive loss of the Shunt arm in State-B:
```

```
% -----------------------------------------------------------------------
GB=GCA+rLA/(rLA*rLA+w0*w0*LA*LA)+(w0*w0*rr2*CT*CT)/(1+w0*w0*rr2*rr2*CT*
    CT);
% -----------------------------------------------------------------------
j=sqrt(-1);
w=0;N=2000;w1=0;w2=2;DW=(w2-w1)/N;
FRI(1:(N+1))=zeros;
WA(1:(N+1))=zeros;
DEL_FI21(1:N+1)=zeros;
for i=1:N+1
     WA(i)=w;
% ----------------
% State A:
% ----------------
%za=ra+j*w*L1;
za=ra+j*w*L1;
%ya=GA+j*w*CA+1/j/w/LA;
ya=GA+j*w*(CA/(1+w*w*rf2*rf2*CA*CA)-LA/(rLA*rLA+w*w*LA*LA));
[ S11a,S21a,RO11a,F11a,RO21a,F21a ] = S_Par_T_Section (za,ya);
    F21A(i)=F21a;
    RO21A(i)=20*log10(RO21a);
    F11A(i)=F11a;
    RO11A(i)=20*log10(RO11a);
% ------------------
% State-B
zb=rb+j*w*L1+1/j/w/CD1;
yb=GB+j*w*(CT/(1+w*w*rr2*rr2*CT*CT)-LA/(rLA*rLA+w*w*LA*LA));
[ S11b,S21b,RO11b,F11b,RO21b,F21b ] = S_Par_T_Section (zb,yb);
    F21B(i)=F21b;
    RO21B(i)=20*log10(RO21b);
    F11B(i)=F11b;
    RO11B(i)=20*log10(RO11b);
% -----------------------------------------------------------------------
DEL_FI21(i)=F21B(i)-F21A(i);
     w=w+DW;
end
% -----------------------------------------------------------------------
Plot_State_AB_T360_DPS(WA,F21A,RO21A, F11A,RO11A,F21B,RO21B, F11B,RO11B
    ,DEL_FI21)
% -----------------------------------------------------------------------
ra_actual=ra*R
rb_actual=rb*R
% -----------------------------------------------------------------------
GA_actual=GA/R; RA_actual=1/GA_actual
GB_actual=GB/R; RB_actual=1/GB_actual
```

Program List 10.3. Function T360_DPS

```
function [ CD1,L1,LA,CA,CT,CAa,LAa,CD1a,L1a,CTa ] = T360_DPS(Teta_A,
    Teta_B, f0a,w0,R)
% This function generates the element values of an ideal 360 degree
Simple
% T-Section based Digital Phase Shifter
```

```
% Inputs:
% Teta_A: Desired phase shift of the Lowpass Based T-Section DPS
% Teta_B: Desired phase shift of the Highpass Based T-Section DPS
% f0a: Actual centre frequency
% w0: Normalized angular frequency. It is selected as w0=1
% R: Port normalization number. It is usually, selected as R=50 ohms
% Outputs:
% CD1: Reverse Biased diode capacitance of the series arms.
% L1: Series arm inductor
% LA: Shunt arm inductor
% ---------------------------------------------------------------------
% Ideal Lowpass T-Section: See Equations (10.1a) and (10.1b)
LL=tand(abs(Teta_A)/2)/w0;
CL=(2*LL)/(1+w0*w0*LL*LL);
% ---------------------------------------------------------------------
% Ideal Highpass T-Section: See Equations (10.2a) and (10.2b)
CH=tand(90-Teta_B/2)/w0;
LH=(1+w0*w0*CH*CH)/2/CH;
% ---------------------------------------------------------------------
% State-A: Series arm component computations
L1=LL;
CD1=CH/(1+w0*w0*L1*CH);
% ---------------------------------------------------------------------
% State B: Computation of CA: See Equation (10.4) & (10.5)
A=w0*w0*LH;
B=-(w0*w0*LH*CL+1);
CD2=CD1;
C=B*CD2;
Delta=B*B-4*A*C;
CA=(-B+sqrt(Delta))/2/A;
% ---------------------------------------------------------------------
LA=1/(CA-CL)/w0/w0;
CT=CA*CD2/(CA+CD2);
% ---------------------------------------------------------------------
CAa=CA/2/pi/f0a/R;
CD1a=CD1/2/pi/f0a/R;
LAa=R*LA/2/pi/f0a;
L1a=R*L1/2/pi/f0a;
CTa=CT/2/pi/f0a/R;


end
```

Program List 10.4. Main_Example_10_3.m

```
% Main Program: Main_Example_10_3.m
% March 11, 2019
% Developed by BS Yarman, Vanikoy, Istanbul
% This program evaluates the lossy performance of a Highpass T-Section
DPS
% for a specified actual center frequency f0 in Hz.
% ---------------------------------------------------------------------
clc; close all
```

```
% Inputs:
Teta_A=input('Design of 360 Degree T Section DPS. Enter Phase of
State-A(Lowpass-T): Teta-A in Degree=')
Teta_B=input('Design of 360 Degree T Section DPS. Enter Phase of
State-B(Highpass-T): Teta-B in Degree=')
% Inputs
f0a=input('Enter the actual center frequency in Hz f0a =')
w0=input('At f0a, enter the normalized angular frequency w0 =')
R=input('Enter the normaliziation Resistor R =')
% -------------------------------------------------------------------
% Compute the normalized element values of T-360 Degree-DPS
[ CD1,L1,LA,CA,CT,CAa,LAa,CD1a,L1a,CTa ] = T360_DPS(Teta_A, Teta_B, f0a
    ,w0,R)
% -------------------------------------------------------------------
% ASSUMPTION 1:
% It is assumed that the series loss Ron of a forward biased diode is
% equal to the on channel resistor of an CMOS Switch
% [see Equation (6.1) of Chapter 6].
[ RF1,rf1 ] =Channel_Resistance_of_a_CMOS(CD1,R,f0a);
% ASSUMPTION 2:
% At w0, quality factor for inductors and capacitors
QL=20;
QC=20;
% -------------------------------------------------------------------
CD2=CD1;
[ RF,rf ] =Channel_Resistance_of_a_CMOS(CD1,R,f0a);
% -------------------------------------------------------------------
% Series Arm Losses
rf1=rf;
rL1=(w0*L1)/QL;
ra=rL1+rf1;
% Shunt arm losses
rf2=rf;
rLA=(w0*LA)/QL;
GCA=(w0*CA)/QC;
GCD1=(w0*CD1)/QC;
% -------------------------------------------------------------------
j=sqrt(-1);
w=0;N=2000;w1=0;w2=2;DW=(w2-w1)/N;
FRI(1:(N+1))=zeros;
WA(1:(N+1))=zeros;
DEL_FI21(1:N+1)=zeros;
for i=1:N+1
     WA(i)=w;
% ---------------
% State A:
% ---------------
%za=ra+j*w*L1;
za=ra+j*w*L1;
% Computation of shunt arm admittance ya
% ZTA=r_f2+1/(GCA+jw*CA)
ZTA=rf+1/(GCA+j*w*CA);
YTA=1/ZTA;
```

```
ya=1/(rLA+j*w*LA)+YTA;
[ S11a,S21a,RO11a,F11a,RO21a,F21a ] = S_Par_T_Section (za,ya);
    F21A(i)=F21a;
    RO21A(i)=20*log10(RO21a);
    F11A(i)=F11a;
    RO11A(i)=20*log10(RO11a);
% ------------------
% State-B
zb=(rL1+j*w*L1)+1/(GCD1+j*w*CD1);
ZTB=1/(GCA+j*w*CA)+1/(GCD1+j*w*CD1);
YTB=1/ZTB;
yb=1/(rLA+j*w*LA)+YTB;
  [ S11b,S21b,RO11b,F11b,RO21b,F21b ] = S_Par_T_Section (zb,yb);
      F21B(i)=F21b;
      RO21B(i)=20*log10(RO21b);
      F11B(i)=F11b;
      RO11B(i)=20*log10(RO11b);
% ----------------------------------------------------------------------
DEL_FI21(i)=F21B(i)-F21A(i);
     w=w+DW;
end
% --------------------------------------------------------------------
Plot_State_AB_T360_DPS(WA,F21A,RO21A, F11A,RO11A,F21B,RO21B, F11B,RO11B
    ,DEL_FI21)
% ----------------------------------------------------------------------
ra_actual=ra*R
rb_actual=rb*R
% ----------------------------------------------------------------------
GA_actual=GA/R; RA_actual=1/GA_actual
GB_actual=GB/R; RB_actual=1/GB_actual
```

Program List 10.5. Function S_Par_T_Section

```
function [ S11,S21,RO11,F11,RO21,F21 ] = S_Par_T_Section (z,y)
%This function generates the S-Parameters of a T Section
% Phase Shifter from the series arm impedance Z(jw) and
% the shunt arm admittance Y(jw)
% ----------------------------------------------------------------------
% Developed by BS Yarman: Feb 20, 2019, Vanikoy, Istanbul
% See Equations (5.9)
% ----------------------------------------------------------------------
D=z*z*y+2*z*y+2*z+y+2;
S11=((1-z*z)*y-2*z)/D;
S21=2/D;
R11=real(S11); X11=imag(S11);F11=atan2d(X11,R11);
R21=real(S21); X21=imag(S21);F21=atan2d(X21,R21);
RO11=abs(S11);
RO21=abs(S21);
end
```

Program List 10.6. function Plot_State_AB_T360_DPS

```
function Plot_State_AB_T360_DPS(WA,F21A,RO21A, F11A,RO11A,F21B,RO21B,
    F11B,RO11B,DEL_FI21)
figure
plot(WA,F21A,WA,F21B,WA,DEL_FI21)
title('State A and State B: Phase variations F21A and F21B of a
360-T-Section')
legend('F21A','F21B','DEL-FI21')
xlabel('Normalized Angular Frequency')
ylabel('Phase of S21A and S21B')
% Amplitude of S21
figure
plot(WA,RO21A,WA,RO21B)
title('State-A and State-B: Amplitude variation RO21A and RO21B of a
360-T-Section')
legend('RO21A in dB','RO21B in dB')
xlabel('Normalized Angular Frequency')
ylabel('Amplitude of S21A and S21B in dB')
% -----------------------------------------------------------------
figure
plot(WA,F11A, WA, F11B)
title('State-A and State-B: Phase variation F11A and F11B of a
360-T-Section')
legend('F11A')
xlabel('Normalized Angular Frequency')
ylabel('Phase of S11A and S11B')
% Amplitude of S11
figure
plot(WA,RO11A, WA, RO11B)
title('State-A and State-B: Amplitude variation RO11A and RO11B of a
360-T-Section')
legend('RO11A in dB','RO11B in dB')
xlabel('Normalized Angular Frequency')
ylabel('Amplitude of S11A and S11B in dB')

end
```

Appendix 11: MatLab Programs for Chapter 11

Program List 11.1. Main_Example_11_1.m

```
% Main Program: Main_Example_11_1.m
% March 26, 2019
% Developed by BS Yarman, Vanikoy, Istanbul
% This program evaluates the ideal performance of a Highpass
    PI-Section DPS
% for a specified actual center frequency f0 in Hz.
% ---------------------------------------------------------------------
clc; close all
% Inputs:
Teta_A=input('Design of 360 Degree PI Section DPS. Enter Phase of
State-A(Lowpass-PI): Teta-A in Degree=')
```

```
Teta_B=input('Design of 360 Degree PI Section DPS. Enter Phase of
State-B(Highpass-PI): Teta-B in Degree=')
% Inputs
f0a=input('Enter the actual center frequency in Hz f0a =')
w0=input('At f0a, enter the normalized angular frequency w0 =')
R=input('Enter the normaliziation Resistor R =')
% ----------------------------------------------------------------------
% Compute the normalized element values of PI-360 Degree-DPS
% Ideal Lowpass PI-Section: See Equations (11.1a) and (11.1b)
% CL=(1/w0)*tan(?A/2)>0
CL=(1/w0)*tand(Teta_A/2);
% L_L=L=(2C_L)/(1+?_0^2 C_L^2 )>0
LL=2*CL/(1+w0*w0*CL*CL);
L=LL;
% ----------------------------------------------------------------------
% Ideal Highpass PI-Section: See Equations (11.2a) and (11.2b)
% Ideal Highpass T-Section: See Equations (11.2a) and (11.2b)
% LH=(1/w0)*cotan(?B/2)>0
LH=(1/w0)*cotd(Teta_B/2);
% CH=(1/(?0^2 ))((1+?0^2 LH^2)/(2LH^2 ))>0
CH=(1/w0/w0)*(1+w0*w0*LH*LH)/2/LH;
% Compute ideal element values of 360 Degree PI Section Digital Phase
Shifter
[CD1,L,LA,CA,CT,CAa,LAa,CD1a,La,CTa] = PI_360_DPS(Teta_A, Teta_B, f0a,
    w0,R);
% ----------------------------------------------------------------------
j=sqrt(-1);
w=0;N=2000;w1=0;w2=5;DW=(w2-w1)/N;
FRI(1:(N+1))=zeros;
WA(1:(N+1))=zeros;
DEL_FI21(1:N+1)=zeros;
for i=1:N+1
     WA(i)=w;
% ----------------
% State A:
% ----------------
za=j*w*L;
ya=j*w*CA+1/j/w/LA;
[ S11a,S21a,RO11a,F11a,RO21a,F21a ] = S_Par_PI_Section ( za,ya );
    F21A(i)=F21a;
    RO21A(i)=20*log10(RO21a);
    F11A(i)=F11a;
    RO11A(i)=20*log10(RO11a);
% ------------------
% State-B
% ------------------
zb=j*w*L+1/j/w/CD1;
yb=j*w*CT+1/j/w/LA;
[ S11b,S21b,RO11b,F11b,RO21b,F21b ] = S_Par_PI_Section ( zb,yb );
    F21B(i)=F21b;
    RO21B(i)=20*log10(RO21b);
    F11B(i)=F11b;
    RO11B(i)=20*log10(RO11b);
```

```
% ----------------------------------------------------------------------
DEL_FI21(i)=F21B(i)-F21A(i);
    w=w+DW;
end
% ----------------------------------------------------------------------
Plot_State_AB_PI_360_DPS(WA,F21A,RO21A, F11A,RO11A,F21B,RO21B, F11B,
    RO11B,DEL_FI21)
```

Program List 11.2. Main_Example_11_2.m

```
% Main Program: Main_Example_11_2.m
% March 27, 2019
% Developed by BS Yarman, Vanikoy, Istanbul
% This program evaluates the lossy performance of a Highpass PI-Section
     DPS
% for a specified actual center frequency f0 in Hz.
% ----------------------------------------------------------------------
% In this program we used the approximate explicit formulas to
    compute the
% losses for both State-A and State-B
% ----------------------------------------------------------------------
clc; close all
% Inputs:
Teta_A=input('Design of 360 Degree PI Section DPS. Enter Phase of
    State-A(Lowpass-PI): Teta-A in Degree=')
Teta_B=input('Design of 360 Degree PI Section DPS. Enter Phase of
    State-B(Highpass-PI): Teta-B in Degree=')
% Inputs
f0a=input('Enter the actual center frequency in Hz f0a =')
w0=input('At f0a, enter the normalized angular frequency w0 =')
R=input('Enter the normaliziation Resistor R =')
% ----------------------------------------------------------------------
% Compute the normalized element values of PI-360 Degree-DPS
[ CD1,L1,LA,CA,CT,CAa,LAa,CD1a,L1a,CTa ] = PI_360_DPS( Teta_A, Teta_B,
    f0a,w0,R )
% ----------------------------------------------------------------------
% ASSUMPTION 1:
% It is assumed that the series loss Ron of a forward biased diode is
% equal to the on channel resistor of an CMOS Switch
% [see Equation (6.1) of Chapter 6].
[ RF1,rf1 ] =Channel_Resistance_of_a_CMOS( CD1,R,f0a );
% ASSUMPTION 2:
% Reverse biased resistive loss of a diode is the "Percent_RVS"
    amount of
% its reverse baised impedance at w0
CrDel_D=10;
rr1=1/w0/CD1/CrDel_D;
CD2=CD1;
[ RF2,rf2 ] =Channel_Resistance_of_a_CMOS( CD2,R,f0a );
rr2=1/w0/CD2/CrDel_D;
%
% ----------------------------------------------------------------------
% Loss Computations for both State-A and State-B:
```

```
% Assumption 3: Loss of an inductor is Percent_L amount of its impedance
% value at w0.
% Assumption 4: Connectivity loss of an inductor is "Percent_S"
    amount of
% its impedance value at w0.
% Assumption 5: Conductive loss of a Capacitor is "Percent_C" amount
    of its
% admittance value at w0.
CrDel_S=10;
CrDEL_L=10;
CrDel_C=10;
% ----------------------------------------------------------------------
% Resistive loss of the series arms in State-A:
rL1=w0*L1/CrDEL_L;
rLA=w0*LA/CrDEL_L;
rs=w0*L1/CrDel_S;
ra=rf1;
% ----------------------------------------------------------------------
% Conductive loss of the Shunt arm in State-A:
GCA=w0*CA/CrDel_C;
% GCA=0;GA=0;
GA=GCA+rLA/(rLA*rLA+w0*w0*LA*LA);
% ----------------------------------------------------------------------
% Resistive loss of the series arms in State-B:
rb=rr1;
% ----------------------------------------------------------------------
% Conductive loss of the Shunt arm in State-B:
% ----------------------------------------------------------------------
GB=GCA+rLA/(rLA*rLA+w0*w0*LA*LA)+(w0*w0*rr2*CT*CT)/(1+w0*w0*rr2*rr2*CT*
    CT);
% ----------------------------------------------------------------------
j=sqrt(-1);
w=0;N=2000;w1=0;w2=2;DW=(w2-w1)/N;
FRI(1:(N+1))=zeros;
WA(1:(N+1))=zeros;
DEL_FI21(1:N+1)=zeros;
for i=1:N+1
     WA(i)=w;
% ----------------
% State A:
% ----------------
%za=ra+j*w*L1;
za=ra+j*w*L1;
%ya=GA+j*w*CA+1/j/w/LA;
ya=GA+j*w*(CA/(1+w*w*rf2*rf2*CA*CA)-LA/(rLA*rLA+w*w*LA*LA));
[ S11a,S21a,RO11a,F11a,RO21a,F21a ] = S_Par_PI_Section (za,ya);
    F21A(i)=F21a;
    RO21A(i)=20*log10(RO21a);
    F11A(i)=F11a;
    RO11A(i)=20*log10(RO11a);
% ------------------
% State-B
zb=rb+j*w*L1+1/j/w/CD1;
```

```
yb=GB+j*w*(CT/(1+w*w*rr2*rr2*CT*CT)-LA/(rLA*rLA+w*w*LA*LA));
  [ S11b,S21b,RO11b,F11b,RO21b,F21b ] = S_Par_PI_Section (zb,yb);
  F21B(i)=F21b;
  RO21B(i)=20*log10(RO21b);
  F11B(i)=F11b;
  RO11B(i)=20*log10(RO11b);
% ----------------------------------------------------------------------
DEL_FI21(i)=F21B(i)-F21A(i);
  w=w+DW;
end
% ----------------------------------------------------------------------
Plot_State_AB_PI_360_DPS(WA,F21A,RO21A, F11A,RO11A,F21B,RO21B, F11B,
    RO11B,DEL_FI21)
% ----------------------------------------------------------------------
ra_actual=ra*R
rb_actual=rb*R
% ----------------------------------------------------------------------
GA_actual=GA/R; RA_actual=1/GA_actual
GB_actual=GB/R; RB_actual=1/GB_actual
```

Program List 11.3. Main_Example_11_3.m

```
% Main Program: Main_Example_11_3.m
% March 27, 2019
% Developed by BS Yarman, Vanikoy, Istanbul
% This program evaluates the lossy performance of a 360 PI-Section DPS
% for a specified actual center frequency f0 in Hz.
% ----------------------------------------------------------------------
clc; close all
% Inputs:
Teta_A=input('Design of 360 Degree PI Section DPS. Enter Phase of
    State-A(Lowpass-PI): Teta-A in Degree=')
Teta_B=input('Design of 360 Degree PI Section DPS. Enter Phase of
    State-B(Highpass-PI): Teta-B in Degree=')
% Inputs
f0a=input('Enter the actual center frequency in Hz f0a =')
w0=input('At f0a, enter the normalized angular frequency w0 =')
R=input('Enter the normaliziation Resistor R =')
% ----------------------------------------------------------------------
% Compute the normalized element values of PI-360 Degree-DPS
[CD1,L1,LA,CA,CT,CAa,LAa,CD1a,L1a,CTa] = PI_360_DPS(Teta_A, Teta_B, f0a
    ,w0,R)
% ----------------------------------------------------------------------
% ASSUMPTION 1:
% It is assumed that the series loss Ron of a forward biased diode is
% equal to the on channel resistor of an CMOS Switch
% [see Equation (6.1) of Chapter 6].
[RF1,rf1] = Channel_Resistance_of_a_CMOS(CD1,R,f0a);
% ASSUMPTION 2:
% At w0, quality factor for inductors and capacitors
QL=75;
QC=75;
% ----------------------------------------------------------------------
```

```
CD2=CD1;
[RF,rf] =Channel_Resistance_of_a_CMOS(CD1,R,f0a);
% ----------------------------------------------------------------------
% Series Arm Losses
rf1=rf;
rL1=(w0*L1)/QL;
 ra=rL1+rf1;
% Shunt arm losses
rf2=rf;
rLA=(w0*LA)/QL;
GCA=(w0*CA)/QC;
GCD1=(w0*CD1)/QC;
% ----------------------------------------------------------------------
j=sqrt(-1);
w=0;N=2000;w1=0;w2=2;DW=(w2-w1)/N;
FRI(1:(N+1))=zeros;
WA(1:(N+1))=zeros;
DEL_FI21(1:N+1)=zeros;
for i=1:N+1
   WA(i)=w;
% ----------------
% State A:
% ----------------
%za=ra+j*w*L1;
za=ra+j*w*L1;
% Computation of shunt arm admittance ya
% ZTA=rf2+1/(GCA+jwCA)
ZTA=rf+1/(GCA+j*w*CA);
YTA=1/ZTA;
ya=1/(rLA+j*w*LA)+YTA;
[S11a,S21a,RO11a,F11a,RO21a,F21a] = S_Par_PI_Section (za,ya);
    F21A(i)=F21a;
    RO21A(i)=20*log10(RO21a);
    F11A(i)=F11a;
    RO11A(i)=20*log10(RO11a);
% ------------------
% State-B
zb=(rL1+j*w*L1)+1/(GCD1+j*w*CD1);
ZTB=1/(GCA+j*w*CA)+1/(GCD1+j*w*CD1);
YTB=1/ZTB;
yb=1/(rLA+j*w*LA)+YTB;
[S11b,S21b,RO11b,F11b,RO21b,F21b] = S_Par_PI_Section (zb,yb);
   F21B(i)=F21b;
   RO21B(i)=20*log10(RO21b);
   F11B(i)=F11b;
   RO11B(i)=20*log10(RO11b);
% ----------------------------------------------------------------------
     DEL_FI21(i)=F21B(i)-F21A(i);
     w=w+DW;
end
% ----------------------------------------------------------------------
Plot_State_AB_PI_360_DPS(WA,F21A,RO21A, F11A,RO11A,F21B,RO21B, F11B,
    RO11B,DEL_FI21)
```

```
% -------------------------------------------------------------------
zb0=(rL1+j*w0*L1)+1/(GCD1+j*w0*CD1);
rb=real(zb0);
%
ra_actual=ra*R
rb_actual=rb*R
% -------------------------------------------------------------------
% GA_actual=GA/R; RA_actual=1/GA_actual
% GB_actual=GB/R; RB_actual=1/GB_actual
```

Program List 11.4. function S_Par_PI_Section

```
function [ S11,S21,RO11,F11,RO21,F21 ] = S_Par_PI_Section (z,y)
%This function generates the S-Parameters of a PI Section
% Phase Shifter from the series arm impedance Z(jw) and
% the shunt arm admittance Y(jw)
% -------------------------------------------------------------------
% Developed by BS Yarman: March 26, 2019, Vanikoy, Istanbul
% See Equations (5.9)
% -------------------------------------------------------------------
% N=z(1-y^2)-2y
N=z*(1-y*y)-2*y;
% D= zy^2+2zy+2y+z+2
D=z*y*y+2*z*y+2*y+z+2;
S11=N/D;
S21=2/D;
R11=real(S11); X11=imag(S11);F11=atan2d(X11,R11);
R21=real(S21); X21=imag(S21);F21=atan2d(X21,R21);
RO11=abs(S11);
RO21=abs(S21);
end
```

Program List 11.5. function PI_360_DPS

```
function [ CD1,L,LA,CA,CT,CAa,LAa,CD1a,La,CTa ] = PI_360_DPS(Teta_A,
    Teta_B, f0a,w0,R)
% This function generates the element values of an ideal 360 degree
    Simple
% PI-Section based Digital Phase Shifter
% Developed by BS Yarman on March 26, 2019, Vanikoy, Istanbul
% Inputs:
% Teta_A: Desired phase shift of the Lowpass Based PI-Section DPS
% Teta_B: Desired phase shift of the Highpass Based PI-Section DPS
% f0a: Actual centre frequency
% w0: Normalized angular frequency. It is selected as w0=1
% R: Port normalization number. It is usually, selected as R=50 ohms
% Outputs:
% CD1: Reverse Biased diode capacitance of the series arms.
% L: Series arm inductor
% LA: Shunt arm inductor
% CA: Shunt arm capacitor
% -------------------------------------------------------------------
% Ideal LowpassPI-Section: See Equations (11.1a) and (11.1b)
```

```
% CL=(1/w0)*tan(?A/2)>0
CL=(1/w0)*tand(Teta_A/2);
% L_L=L=(2C_L)/(1+?_0^2 C_L^2)>0
LL=2*CL/(1+w0*w0*CL*CL);
% ---------------------------------------------------------------------
% Ideal Highpass T-Section: See Equations (11.2a) and (11.2b)
% LH=(1/w0)*cotan(?B/2)>0
LH=(1/w0)*cotd(Teta_B/2);
% CH=(1/(?0^2))((1+?0^2 LH^2)/(2LH^2))>0
CH=(1/w0/w0)*(1+w0*w0*LH*LH)/2/LH;

% ---------------------------------------------------------------------
% State-A: Series arm component computations
L=LL;
CD1=CH/(1+w0*w0*L*CH);
% ---------------------------------------------------------------------
% State B: Computation of CA: See Equation (11.4) & (11.5)
A=w0*w0*LH;
B=-(w0*w0*LH*CL+1);
CD2=CD1;
C=B*CD2;
Delta=B*B-4*A*C;
CA=(-B+sqrt(Delta))/2/A;
% ---------------------------------------------------------------------
LA=1/(CA-CL)/w0/w0;
% LA=1/(?02̂)/(CA-CL))>0
CT=CA*CD2/(CA+CD2);
% ---------------------------------------------------------------------
%
CAa=CA/2/pi/f0a/R;
CD1a=CD1/2/pi/f0a/R;
LAa=R*LA/2/pi/f0a;
La=R*L/2/pi/f0a;
CTa=CT/2/pi/f0a/R;
end
```

Program List 11.6. function Plot_State_AB_PI_360_DPS

```
function Plot_State_AB_PI_360_DPS(WA,F21A,RO21A, F11A,RO11A,F21B,RO21B,
      F11B,RO11B,DEL_FI21)
figure
plot(WA,F21A,WA,F21B,WA,DEL_FI21)
title('State A and State B: Phase variations F21A and F21B of a
     360-PI-Section')
legend('F21A','F21B','DEL-FI21')
xlabel('Normalized Angular Frequency')
ylabel('Phase of S21A and S21B')
% Amplitude of S21
figure
plot(WA,RO21A,WA,RO21B)
title('State-A and State-B: Amplitude variation RO21A and RO21B of a
     360-PI-Section')
legend('RO21A in dB','RO21B in dB')
```

```
xlabel('Normalized Angular Frequency')
ylabel('Amplitude of S21A and S21B in dB')
% ---------------------------------------------------------------------
figure
plot(WA,F11A, WA, F11B)
title('State-A and State-B: Phase variation F11A and F11B of a
    360-PI-Section')
legend('F11A')
xlabel('Normalized Angular Frequency')
ylabel('Phase of S11A and S11B')
% Amplitude of S11
figure
plot(WA,RO11A, WA, RO11B)
title('State-A and State-B: Amplitude variation RO11A and RO11B of a
    360-PI-Section')
legend('RO11A in dB','RO11B in dB')
xlabel('Normalized Angular Frequency')
ylabel('Amplitude of S11A and S11B in dB')

end
```

Appendix 12: MatLab Programs for Chapter 12

Program List 12.1. Main_Example_12_1.m

```
% Main Program: Main_Example_12_2.m
% March 27, 2019
% Developed by BS Yarman, Vanikoy,Istanbul
% This program evaluates the lossy performance of a 360 PI-Section DPS
% for a specified actual center frequency f0 in Hz.
%---------------------------------------------------------------------
 clc; close all
 % Inputs:
 Teta_A=0;
 Teta_B=input('Design of 180 Degree Highpass PI Section DPS. Enter
                Phase of State-B(Highpass-PI): Teta-B in Degree=')
 % Inputs
 f0a=input('Enter the actual center frequency in Hz f0a =')
 w0=input('At f0a, enter the normalized angular frequency w0 =')
 R=input('Enter the normaliziation Resistor R =')
 %------------------------------------------------------------------
 % Compute the normalized element values of PI-360 Degree-DPS
 [CD1,L1,LA,CA,CT,CAa,LAa,CD1a,L1a,CTa] = PI_360_DPS(Teta_A, Teta_B,
 f0a,w0,R)
 %------------------------------------------------------------------
 % ASSUMPTION 1:
 % It is assumed that the series loss Ron of a forward biased diode is
 % equal to the on channel resistor of an CMOS Switch
 % [see Equation (6.1) of Chapter 6].
 [RF1,rf1] =Channel_Resistance_of_a_CMOS(CD1,R,f0a);
 % ASSUMPTION 2:
 % At w0, quality factor for inductors and capacitors
 QL=10;
```

```
QC=10;
%------------------------------------------------------------------
CD2=CD1;
[RF,rf] =Channel_Resistance_of_a_CMOS(CD1,R,f0a);
%-------------------------------------------------------------------
% Series Arm Losses
rf1=rf;
rL1=(w0*L1)/QL;
ra=rL1+rf1;
% Shunt arm losses
rf2=rf;
rLA=(w0*LA)/QL;
GCA=(w0*CA)/QC;
GCD1=(w0*CD1)/QC;
%------------------------------------------------------------------
j=sqrt(-1);
w=0;N=2000;w1=0;w2=2;DW=(w2-w1)/N;
FRI(1:(N+1))=zeros;
WA(1:(N+1))=zeros;
DEL_FI21(1:N+1)=zeros;
for i=1:N+1
    WA(i)=w;
% ----------------
% State A:
% ----------------
%za=ra+j*w*L1;
za=ra+j*w*L1;
% Computation of shunt arm admittance ya
% ZTA=rf2+1/(GCA+jwCA)
ZTA=rf+1/(GCA+j*w*CA);
YTA=1/ZTA;
ya=1/(rLA+j*w*LA)+YTA;
[S11a,S21a,RO11a,F11a,RO21a,F21a] = S_Par_PI_Section (za,ya);
   F21A(i)=F21a;
   RO21A(i)=20*log10(RO21a);
   F11A(i)=F11a;
   RO11A(i)=20*log10(RO11a);
% ------------------
% State-B
zb=(rL1+j*w*L1)+1/(GCD1+j*w*CD1);
ZTB=1/(GCA+j*w*CA)+1/(GCD1+j*w*CD1);
YTB=1/ZTB;
yb=1/(rLA+j*w*LA)+YTB;
[S11b,S21b,RO11b,F11b,RO21b,F21b] = S_Par_PI_Section (zb,yb);
   F21B(i)=F21b;
   RO21B(i)=20*log10(RO21b);
   F11B(i)=F11b;
   RO11B(i)=20*log10(RO11b);
%-----------------------------------------------------------------
DEL_FI21(i)=F21B(i)-F21A(i);
    w=w+DW;
end
%------------------------------------------------------------------
```

```
Plot_State_AB_PI_360_DPS(WA,F21A,RO21A, F11A,RO11A,F21B,RO21B, F11B,
   RO11B,DEL_FI21)
%-------------------------------------------------------------------
zb0=(rL1+j*w0*L1)+1/(GCD1+j*w0*CD1);
rb=real(zb0);
%
ra_actual=ra*R
rb_actual=rb*R
%-------------------------------------------------------------------
% GA_actual=GA/R; RA_actual=1/GA_actual
% GB_actual=GB/R; RB_actual=1/GB_actual
```

Program List 12.2. Main_Example_12_2.m

```
% Main Program: Main_Example_12_2.m
% March 27, 2019
% Developed by BS Yarman, Vanikoy, Istanbul
% This program evaluates the lossy performance of a 360 PI-Section DPS
% for a specified actual center frequency f0 in Hz.
%-----------------------------------------------------------------
clc; close all
% Inputs:
Teta_A=0;
Teta_B=input('Design of 180 DegreeHighpass PI Section DPS. Enter
             Phase of State-B(Highpass-PI): Teta-B in Degree=')
% Inputs
f0a=input('Enter the actual center frequency in Hz f0a =')
w0=input('At f0a, enter the normalized angular frequency w0 =')
R=input('Enter the normaliziation Resistor R=')
%-----------------------------------------------------------------
% Compute the normalized element values of PI-360 Degree-DPS
[CD1,L1,LA,CA,CT,CAa,LAa,CD1a,L1a,CTa] = PI_360_DPS(Teta_A, Teta_B,
   f0a,w0,R)
%-----------------------------------------------------------------
% ASSUMPTION 1:
% It is assumed that the series loss Ron of a forward biased diode is
% equal to the on channel resistor of anCMOS Switch
% [see Equation (6.1) of Chapter 6].
[RF1,rf1] =Channel_Resistance_of_a_CMOS(CD1,R,f0a);
% ASSUMPTION 2:
% At w0, quality factor for inductors and capacitors
QL=10;
QC=10;
%------------------------------------------------------------------
CD2=CD1;
[RF,rf] =Channel_Resistance_of_a_CMOS(CD1,R,f0a);
%-------------------------------------------------------------------
% Series Arm Losses
rf1=rf;
rL1=(w0*L1)/QL;
 ra=rL1+rf1;
% Shunt arm losses
rf2=rf;
```

```
rLA=(w0*LA)/QL;
GCA=(w0*CA)/QC;
GCD1=(w0*CD1)/QC;
%-------------------------------------------------------------------
j=sqrt(-1);
w=0;N=2000;w1=0;w2=2;DW=(w2-w1)/N;
FRI(1:(N+1))=zeros;
WA(1:(N+1))=zeros;
DEL_FI21(1:N+1)=zeros;
for i=1:N+1
    WA(i)=w;
% ----------------
% State A:
% ----------------
%za=ra+j*w*L1;
%za=ra+j*w*L1;
% Computation of shunt arm admittance ya
% ZTA=rf2+1/(GCA+jwCA)
ZTA=rf+1/(GCA+j*w*CA);
YTA=1/ZTA;
ya=1/(rLA+j*w*LA)+YTA;
[S11a,S21a,RO11a,F11a,RO21a,F21a] = S_Par_PI_Section (za,ya);
   F21A(i)=F21a;
   RO21A(i)=20*log10(RO21a);
   F11A(i)=F11a;
   RO11A(i)=20*log10(RO11a);
% ------------------
% State-B
zb=(rL1+j*w*L1)+1/(GCD1+j*w*CD1);
ZTB=1/(GCA+j*w*CA)+1/(GCD1+j*w*CD1);
YTB=1/ZTB;
yb=1/(rLA+j*w*LA)+YTB;
 [ S11b,S21b,RO11b,F11b,RO21b,F21b ] = S_Par_PI_Section (zb,yb);
   F21B(i)=F21b;
   RO21B(i)=20*log10(RO21b);
   F11B(i)=F11b;
   RO11B(i)=20*log10(RO11b);
%-----------------------------------------------------------------
DEL_FI21(i)=F21B(i)-F21A(i);
    w=w+DW;
end
%-----------------------------------------------------------------
Plot_State_AB_PI_360_DPS(WA,F21A,RO21A, F11A,RO11A,F21B,RO21B,F11B,
   RO11B,DEL_FI21)
%-----------------------------------------------------------------
zb0=(rL1+j*w0*L1)+1/(GCD1+j*w0*CD1);
rb=real(zb0);
%
ra_actual=ra*R
rb_actual=rb*R
%-----------------------------------------------------------------
% GA_actual=GA/R; RA_actual=1/GA_actual
% GB_actual=GB/R; RB_actual=1/GB_actual
```

Program List 12.3. Function S_Par_PI_Section

```
function [ S11,S21,RO11,F11,RO21,F21 ] = S_Par_PI_Section (z,y)
% This function generates the S-Parameters of a PI Section
% Phase Shifter from the series arm impedance Z(jw) and
% the shunt arm admittance Y(jw)
%-------------------------------------------------------------------
% Developed by BS Yarman: March 26,2019,Vanikoy,Istanbul
% See Equations (5.9)
%--------------------------------------------------------------------
% N=z(1-y^2)-2y
N=z*(1-y*y)-2*y;
% D= zy^2+2zy+2y+z+2
D=z*y*y+2*z*y+2*y+z+2;
S11=N/D;
S21=2/D;
R11=real(S11);X11=imag(S11);F11=atan2d(X11,R11);
R21=real(S21);X21=imag(S21);F21=atan2d(X21,R21);
RO11=abs(S11);
RO21=abs(S21);


end
```

Program List 12.4. Function PI_360_DPS

```
function [CD1,L,LA,CA,CT,CAa,LAa,CD1a,La,CTa]
           = PI_360_DPS(Teta_A, Teta_B, f0a,w0,R)
% This function generates the element valuesof an ideal 360 degree
   Simple
% PI-Section based Digital Phase Shifter
% Developed by BS Yarman on March 26,2019,Vanikoy,Istanbul
% Inputs:
         % Teta_A: Desired phase shift of the Lowpass Based
                      PI-Section DPS
         % Teta_B: Desired phase shift of the Highpass Based
                      PI-Section DPS
         % f0a: Actual centre frequency
         % w0: Normalized angular frequency. It is selected
                      as w0=1
         % R: Port normalization number. It is
                      usually, selected as R=50 ohms
% Outputs:
         % CD1: Reverse Biased diode
                 capacitance of the series arms.
         % L: Series arm inductor
         % LA: Shunt arm inductor
         % CA: Shunt arm capacitor
%---------------------------------------------------------------
% Ideal LowpassPI-Section: See Equations(11.1a) and (11.1b)
% CL=(1/w0)*tan(?A/2)>0
CL=(1/w0)*tand(Teta_A/2);
```

```
 % L_L=L=(2C_L)/(1+?_0^2C_L^2)>0
LL=2*CL/
 (1+w0*w0*CL*CL);
% ----------------------------------------------------------------
% Ideal Highpass T-Section: See Equations(11.2a) and (11.2b)
% LH=(1/w0)*cotan(?B/2)>0
LH=(1/w0)*cotd(Teta_B/2);
% CH=(1/(?0^2))((1+?0^2 LH^2)/(2LH^2))>0
CH=(1/w0/w0)*(1+w0*w0*LH*LH)/2/LH;
%----------------------------------------------------------------
% State-A: Series arm component computations
L=LL;
CD1=CH/(1+w0*w0*L*CH);
%-----------------------------------------------------------------
% State B: Computation of CA: See Equation (11.4 & 11.5)
A=w0*w0*LH;
B=-(w0*w0*LHCL+1);
CD2=CD1;
C=B*CD2;
Delta=B*B-4*A*C;
CA=(-B+sqrt(Delta))/2/A;
%-------------------------------------------------------------------
LA=1/(CA-CL)/w0/w0;
% LA=1/(?0^2)/(CA-CL))>0
CT=CA*CD2/(CA+CD2);
%--------------------------------------------------------------------
CAa=CA/2/pi/f0a/R;
CD1a=CD1/2/pi/f0a/R;
LAa=R*LA/2/pi/f0a;
La=R*L/2/pi/f0a;
CTa=CT/2/pi/f0a/R;

end
```