CHAPTER

11

COMPUTER AIDED ANALYSIS OF NON-LINEAR CIRCUITS

11.1 INTRODUCTION

The solution to non-linear circuits generally takes one of three fomrs: time domain solutions of circuit differential equations, harmonic balance, or Volterra Series solutions. All of these techiques can also be applied to linear circuits which can be viewed as a sub-set of non-linear circuits. In all cases the source or sources must be explicitly represented as an input along with circuit models. The output is usually displayed in the form of a voltage or current plot as a function of time or as a spectral plot where the frequency content of the output signal is of interest. Since the output, in general, is not linearly related to the input, S-parameters (which consist of output to input siganl ratios) are not as useful a concept for non-linear circuits as for linear ciruits. Time domain solutions are often computationally slow since the results are determined by stepping time through small increments. A well known time domain circuit simulation is SPICE in its various realizations. The acronym stands for "Simulation Program with Integrated Circuit Emphasis. It was originally developed for assisting in the design of integrated circuits where timing and waveform shape are important.

This chapter will illustrate non-linear analysis techniques by applying them first to linear circuits where the solutions can be verified using techniques developed earlier. *Time domain* circuits can be divided into *two classes: circuits with and without reactive elements.* Both cases can include circuits with non-linear elements. Circuits with reactive elements are characterized by equations which include derivatives of variables whereas the equations characterizing non reactive circuits do not have derivatives. Each of these type problems can be further broken down into *two cases: circuits with and without delay elements* such as transmission lines, etc. Circuits without delay lines are sometimes referred to as being memoryless since their output at any point in time depends only on the state of the circuit (voltage and currents) at that point in time and does not depend upon the circuit state at past times. *Time domain solutions demonstrate transient* or start-up effects. Such techniques are important in the analysis of circuits where abrupt transistions occur and such techniques are used extensively in digital applications. Time domain techniques are important in the analysis and design of oscillators where one is interested in the start-up behavior of the circuit.

Usually circuits with elements that are highly non-linear must be analyzed using time domain techniques. Even simple looking non-linear circuits can have a sensitive dependence upon their initial conditions and demonstrate distinctly different output behavior. Such circuits are said to the chaotic and their behavior is part of the subject called *chaos*.

Harmonic balance is a popular non-linear technique because it generally is computationally fast. It provides a *steady state solution*. The harmonic balance techniques consists of partitioning the non-linear elements separately from the linear ones. Voltages and currents are represented as a finite sum of harmonic functions (sines and cosines). The process consists of assuming a partial solution at the interface between the non-linear and linear partitions. For example one might start by assuming that the current into the linear partition is zero and then calculate the implied voltage at the interface required by the linear part of the circuit. This voltage is easily calculated for any currents consisting of a sum of harmonic functions since the linear circuit response can be determined for each component and then summed. The resulting voltage must be converted to a time waveform and substituted into the non-linear element to determine the required current. The resulting current is then decomposed into harmonics and compared with the initial assumpition. If the two currents are sufficiently close together for each of the harmonic then the problem is complete and and the solution is given. If as occurs in general the two currents are not equal then a new guess for current is chosen and the process repeats.

A Volterra series approach consists of representing the solution to a non-linear circuit by a multi-dimensional correlation integral. In principle the integral is of infinite dimensions but in practice the dimension is limited to a small number. This approach can be though of as an extension of the impulse response approach to determining circuit output. The impulse response approach applies only to linear circuits and can be thought of as a Volterra series where only one dimensional integrals are involved.

Both Harmonic Balance and Volterra series are approaches which can be applied to *weakly non-linear circuits* where the non-linearity is not severe. There are many useful circuits which can be considered weakly-non-linear. Examples include mixers and frequency doublers. Time-Domain techniques are more applicable to strongly non-linear circuit problems such as switch transitions, oscillator start up, and situation where transient behavior is important. In all cases the results are very dependent upon the models for the non-linear elements which is made even more challenging as the frequency of operation is increased.

11.2 TIME DOMAIN

As indicated above time-domain solutions can be grouped into four categories which depend upon the circuit elements: (1) circuits with reactive but no delay line elements, (2.) circuits with no reactive or delay line elements, (3) circuit with no reactive elements but having delay line elements, and (4) circuits with both reactive and delay line elements. The time domain approach to circuits in category (1) are considered first. The circuit of figure 1 will be used as a starting point. This circuit is in fact a linear circuit with one reactive element. This fact will be instrcutive since the standard linear "steady state" solution can be solved and compared to the time-domain solution. The approach that will be used to analyze such problems is known as the *state variable* techniques. A state variable equation is of the following form:

$$\dot{\mathbf{Y}} = \mathbf{f}(t, \mathbf{Y}) + \mathbf{G}(t)$$

where $\dot{\mathbf{Y}}$, **Y**, and **G** are time function column vectors. The function **f** produces a column vector that depends on time "t" and the input vector **Y**. The vector **G**(t) is called the driving stimulus or forcing function and depends only on time, t. The equations is called a state variable equation because it relates the current state **f**(t,**Y**) and the forcing function **G**(t) to the change in state, $\dot{\mathbf{Y}}$. It is instructive to consider how state variable equations can be integrated, i.e., solved. To do this consider a one dimensional problem where the vectors have only one row. In this case, letting F(t,y)=f(t,y)+g(t), the state equation becomes

$$\dot{y} = F(t, y)$$

If the solution is known for a time t_k then a proceedure is needed to determine the solution for a time, $t_{k+1} = t_k + h$ which is an increment h away. The general approach uses a Taylor series expansion for y(t), i.e.,

$$y(t_k + h) = y(t_k) + \frac{\dot{y}(t_k)}{1!}h + \frac{\ddot{y}(t_k)}{2!}h^2 + \frac{\dddot{y}(t_k)}{3!}h^3 + \Lambda$$

The first derivative is known from the state equation. The second derivative and higher derivative can be found by using the multi-dimensional change rule for differentiation, i.e.,

$$\ddot{y}(t) = \frac{d}{dt}(\dot{y}) = \frac{d}{dt}[F(t,y)] = \frac{\partial F(t,y)}{\partial t} + \frac{\partial F(t,y)}{\partial y}\frac{dy}{dt} = \frac{\partial F(t,y)}{\partial t} + \frac{\partial F(t,y)}{\partial y}F(t,y)$$

and therefore

$$\ddot{y}(t_k) = \frac{\partial F(t_k, y_k)}{\partial t} + \frac{\partial F(t_k, y_k)}{\partial y}F(t_k, y_k)$$

In a similar manner the third derivative can be found to be

$$\dddot{y}(t) = \frac{d}{dt}(\ddot{y}) = \frac{d}{dt}\left(\frac{\partial F(t,y)}{\partial t} + \frac{\partial F(t,y)}{\partial y}F(t,y)\right) = \frac{\partial}{\partial t}\left(\frac{\partial F(t,y)}{\partial t} + \frac{\partial F(t,y)}{\partial y}F(t,y)\right) + \frac{\partial}{\partial t}\left(\frac{\partial F(t,y)}{\partial t} + \frac{\partial F(t,y)}{\partial y}F(t,y)\right)\frac{dy}{dt}$$

$$\dddot{y}(t) = \frac{\partial^2 F}{\partial t^2} + 2F\frac{\partial^2 F}{\partial t \partial y} + \frac{\partial F}{\partial t}\frac{\partial F}{\partial y} + \frac{\partial^2 F}{\partial y^2}F^2 + \left(\frac{\partial F}{\partial y}\right)^2 F$$

A technique known as the Runge-Kutta method calculates the derivatives by substituting values into the F function. The points for substitution are found by satisfying the Taylor series expansion to a specific order. In MATLAB there exists routines for solving sate variable differential equations. These can be applied to solve circuit problems.

MATLAB Program for solving state variable differential equations is given below. It is a modification of the ODE23.m file and has previsions for working with a circuit file. Circuit filenames have _RK for Runge_kutta and present state variable derivatives as well has plotting information. Modification include flags to control the working of the circuit file including the maintenance of solution history which is

required to solve problems involving transmission lines. *The modifications to the original program are shown in bold print.* New routine called JHU_RK23

```
function [tout, yout] = ode23(ypfun, t0, tfinal, y0, tol, trace)
%JHUODE23      Modification of MATLAB ODE23 to Solve differential equations,
%       involving circuits with transmission lines (delay lines).  Applies to
%       state variable representation of derivative where state variables also
%       depend on earlier values of the solution. Mainly involves calling the
%       state derivative function an extra time with an "update" flag set to
%       permit function to update history of the solution.  Also calls function at
%       completion so results can be plotted from function.
%       M.L. EDWARDS 9 Nov 1995
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
%       ODE23 integrates a system of ordinary differential equations using
%       2nd and 3rd order Runge-Kutta formulas.
%       [T,Y] = ODE23('yprime', T0, Tfinal, Y0) integrates the system of
%       ordinary differential equations described by the M-file YPRIME.M,
%       over the interval T0 to Tfinal, with initial conditions Y0.
%       [T, Y] = ODE23(F, T0, Tfinal, Y0, TOL, 1) uses tolerance TOL
%       and displays status while the integration proceeds.
%
%       INPUT:
%       F     - String containing name of user-supplied problem description.
%               Call: yprime = fun(t,y) where F = 'fun'.
%               t      - Time (scalar).
%               y      - Solution column-vector.
%               yprime - Returned derivative column-vector; yprime(i) = dy(i)/dt.
%       t0    - Initial value of t.
%       tfinal- Final value of t.
%       y0    - Initial value column-vector.
%       tol   - The desired accuracy. (Default: tol = 1.e-3).
%       trace - If nonzero, each step is printed. (Default: trace = 0).
%
%       OUTPUT:
%       T  - Returned integration time points (column-vector).
%       Y  - Returned solution, one solution column-vector per tout-value.
%
%       The result can be displayed by: plot(tout, yout).
%
%       See also ODE45, ODEDEMO.

%       C.B. Moler, 3-25-87, 8-26-91, 9-08-92.
%       Copyright (c) 1984-94 by The MathWorks, Inc.

% Initialization
clear global      %clears global variables used in "ypfun.m" to store solution history
pow = 1/3;
if nargin < 5, tol = 1.e-3; end
if nargin < 6, trace = 0; end

t = t0;
hmax = (tfinal - t)/16;
h = hmax/8;
y = y0(:);
chunk = 128;
tout = zeros(chunk,1);
yout = zeros(chunk,length(y));
k = 1;
tout(k) = t;
yout(k,:) = y.';

Ignore=feval(ypfun, t, y, 1);  % Passes result to ypfun so function can update
                                    % solution history --Update=1 flag set

if trace
   clc, t, h, y
end

% The main loop

while (t < tfinal) & (t + h > t)
   if t + h > tfinal, h = tfinal - t; end

   % Compute the slopes
```

```
   s1 = feval(ypfun, t, y); s1 = s1(:);
   s2 = feval(ypfun, t+h, y+h*s1); s2 = s2(:);
   s3 = feval(ypfun, t+h/2, y+h*(s1+s2)/4); s3 = s3(:);

   % Estimate the error and the acceptable error
   delta = norm(h*(s1 - 2*s3 + s2)/3,'inf');
   tau = tol*max(norm(y,'inf'),1.0);

   % Update the solution only if the error is acceptable
   if delta <= tau
      t = t + h;
      y = y + h*(s1 + 4*s3 + s2)/6;
      k = k+1;
      if k > length(tout)
         tout = [tout; zeros(chunk,1)];
         yout = [yout; zeros(chunk,length(y))];
      end
      tout(k) = t;
      yout(k,:) = y.';

      Ignore=feval(ypfun, t, y, 1);  % Passes result to ypfun so function can update
                                     % solution history --Update=1 flag set
   end
   if trace
      home, t, h, y
   end

   % Update the step size
   if delta ~= 0.0
      h = min(hmax, 0.9*h*(tau/delta)^pow);
   end
end

if (t < tfinal)
   disp('Singularity likely.')
   t
end

Ignore=feval(ypfun, t0, y0, 0, 1);     % Passes result to ypfun so function can plot
                                       % solution -- Plot=1 flag set

tout = tout(1:k);
yout = yout(1:k,:);
```

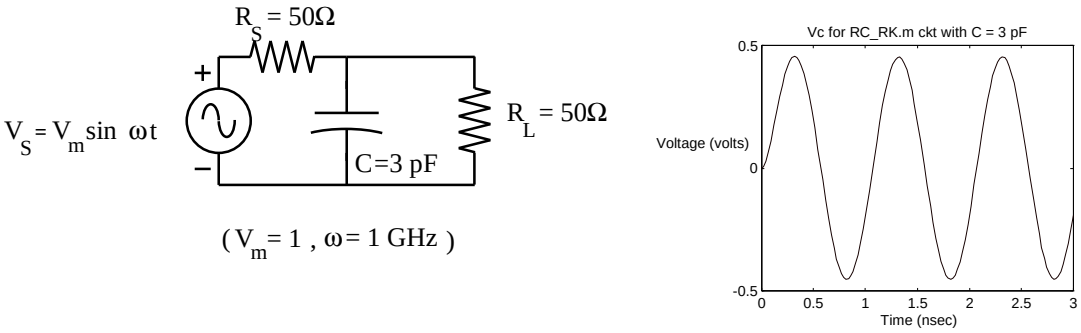


Figure 1. An example of a circuit with one reactive element and no delay elements.

The equations describing the circuit of figure 1 are shown below

$$i_C = C\frac{dV_C}{dt} \qquad i_{R_L} = \frac{V_C}{R_L} \qquad i_S = \frac{V_S - V_C}{R_S}$$

$$i_S = i_C + i_{R_L}$$

$$C\frac{dV_C}{dt} = -\left(\frac{1}{R_S}+\frac{1}{R_L}\right)V_C + \frac{V_S}{R_S}$$

$$\frac{dV_C}{dt} = \frac{1}{C}\left[-\left(\frac{1}{R_S}+\frac{1}{R_L}\right)V_C + \frac{V_S}{R_S}\right]$$

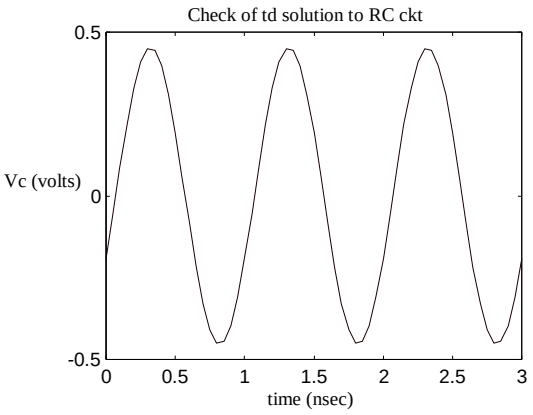
In this case the state equation has only one variable, V_C, and the derivative is computed in a function stored in the MATLAB file: RC.m. The script file RD_td.m calls the Runge-Kutta routine which solves the circuit.

From MATLAB command line >> **JHU_RK23('RC_RK',0,3e-9,0);**

```
function Vcdot=RC_RK(t,Vc,update,mkplot)
%function to solve RC ckt by time stepping
%funtion provides state derative for JHU_RK23.m
% - - - - - - - - - - - - - - - - - - - - - - -
%              (State variable)
%                    +Vc
%            ---Rs----------
%           |        |      |
%  (source) Vs       C      RL (load)
%           |        |      |
%            ---------------
%
%   Vs=Vmsin(wt)
% - - - - - - - - - - - - - - - - - - - - - - -
if nargin<3 update=0; end
if nargin<4 mkplot=0; end
global GLOBALT GLOBALVc

%  units
GHz=1e9;ns=1/GHz; pF=1e-12;
%
%  Component Values
Rs=50;RL=50;C=3*pF;f=1*GHz;
w=2*pi*f;Vm=1;
Vs=Vm*sin(w*t);

%    Circuit Equations
%  Ic=C(dVc/dt); IRL=Vc/RL; Is=(Vs-Vc)/Rs;
%  Is=Ic+IRL
%  C(dVc/dt)=-(1/Rs+1/RL)Vc+(1/Rs)Vs

Vcdot=(1/C)*(-(1/Rs+1/RL)*Vc+(1/Rs)*Vs);

if update
  GLOBALT= [GLOBALT ; t];
  GLOBALVc=[GLOBALVc;Vc];
end

if mkplot
  plot(GLOBALT/ns,GLOBALVc)
  xlabel('Time (nsec)'); ylabel('Voltage (volts)')
  title(['Vc for RC_RK.m ckt with C = ',num2str(C/pF),' pF'])
end
```

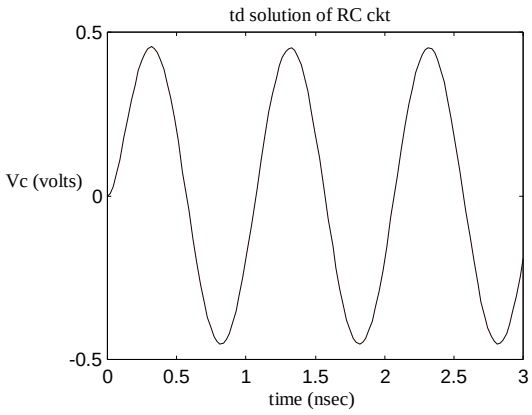
The results of the time-domain circuit can be checked by solving the circuit using AC circuit analysis techniques. The voltage can be converted to a phasor representation and the output voltage calculated as a phasor and then converted into it equivalent time representation. This has been accomplished for the circuit in figure 1 in a MATLAB file named: RC_chk.m shown below with the results. Note that the time-domain solution does not match the phasor solution for the first two cycles, but by the third cycle the solutions are in good agreement. This is an illustration of the transient effects accurately calculated with the time-domain approach. After the transients have stopped the steady state solution is obatined and the two output results begin to agree. This is an interesting case because the transient effects which often appear as amplitude difference show up for this probelm as phase variations.

```
% script file Vc=RC_chk
% to check solution of RC ckt obtained in
% time stepping procedure RC_td.m
% - - - - - - - - - - - - - - - - - - - - -
%                   +Vc
%            ---Rs----------
%           |        |      |
%  (source) Vs       Xc      RL (load)
%           |        |      |
%            --------------
%   Xc=-j1/wC
%   Vs=Vmsin(wt)=> Vs=Vm@angle(0) in phasor notation
% - - - - - - - - - - - - - - - - - - - - -
%  units
GHz=1e9; pF=1e-12;
%
%  Component Values
Rs=50;RL=50;C=3*pF;f=1*GHz;w=2*pi*f;ns=1e-9;Vm=1;
t=[0:.05*ns:3*ns];

Vs=Vm;  %phasor

Xc=-1/(w*C);
ZL=j*RL*Xc/(RL+j*Xc);
Zs= Rs+ZL;
Is=Vs/Zs;
Vc=Is*ZL;                       % phasor soltution
vc=abs(Vc)*sin(w*t+angle(Vc));  % converted to time soln
figure
plot(t/ns,vc);
xlabel('time (nsec)');
ylabel('Vc (volts)');
title('Check of td solution to RC ckt')
```



Phasor (Steady State) Solution

(a.)



Time-Domain Solution (Runge-Kutta)

(b.)

Attention is now turned to consider a circuit with three reactive elements as shown in the figure 2. The state variable will always be the current in an inductor and the voltage across a capacitor. The equations describing the circuit are shown below

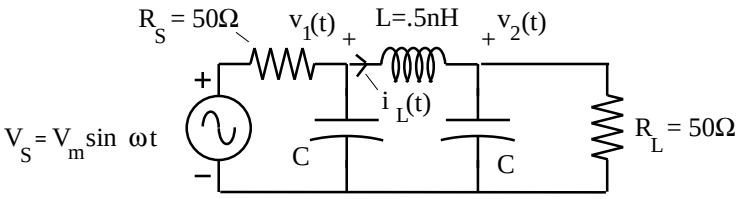


Figure 2. A circuit with three reactive elements.

$$v_1 - v_2 = L\frac{di_L}{dt} \qquad \frac{v_S - v_1}{R_S} = i_L + C\frac{dv_1}{dt} \qquad i_L = C\frac{dv_2}{dt} + \frac{v_2}{R_L}$$

Rearranging these equations gives

$$L\frac{di_L}{dt} = v_1 - v_2 \qquad R_S C\frac{dv_1}{dt} = -R_S i_L - v_1 + v_S \qquad R_L C\frac{dv_2}{dt} = R_L i_L - v_2$$

which can be represented in matrix terms as

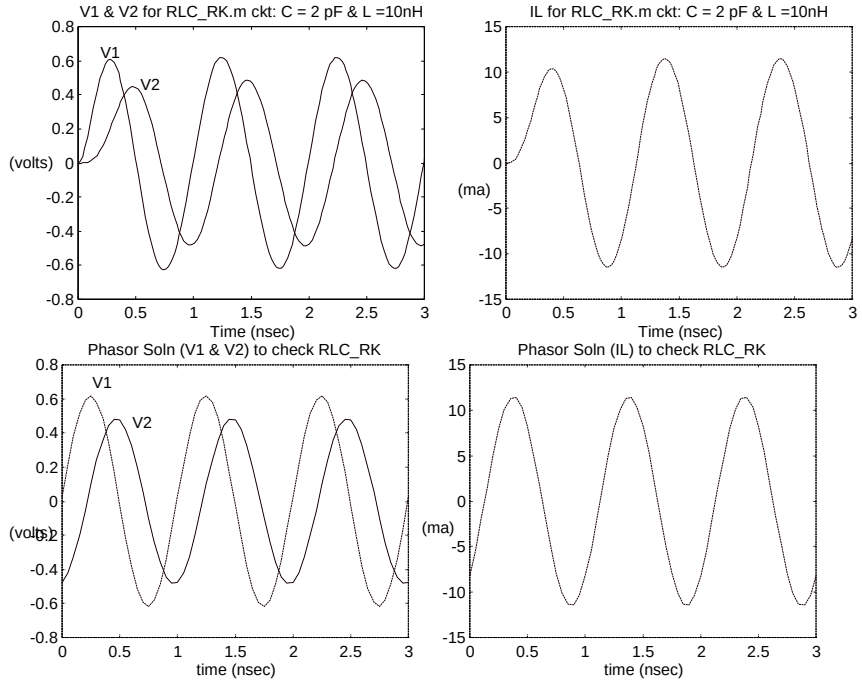
$$\begin{pmatrix} L & 0 & 0 \\ 0 & R_S C & 0 \\ 0 & 0 & R_L C \end{pmatrix} \frac{d}{dt}\begin{pmatrix} i_L \\ v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 & -1 \\ -R_S & -1 & 0 \\ R_L & 0 & -1 \end{pmatrix}\begin{pmatrix} i_L \\ v_1 \\ v_2 \end{pmatrix} + \begin{pmatrix} 0 \\ v_S \\ 0 \end{pmatrix}$$

The state variable derivative and Runge-Kutta solution are implemented in the Matlab function RLC.m and script file RLC_td.m, respecively. This solution is also compared with a phasor (steady state) solution (RLC_chk.m) and the results for both are shown below.

```
function IVdot=RLC_RK(t,IV,update,mkplot)
% function to solve RLC ckt
% IV= IVvector (vector of current/voltage state variables)
% Call from JHU_RK23(...)
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
%           (V1,V2,IL=State Variables)
%
%                 V1+       +V2
%            ---Rs------L-----------
%           |        | -IL-> |      |
% (source)  Vs       C       C      RL(load)
%           |        |       |      |
%            ----------------------
%
%   Vs=Vmsin(wt)
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
if nargin<3 update=0; end
if nargin<4 mkplot=0; end

% units
GHz=1e9;ns=1/GHz; pF=1e-12; nH=1e-9;
Rs=50;RL=50;L=10*nH;C=2*pF;f=1*GHz;w=2*pi*f;
Vm=1;Vs=Vm*sin(w*t);

%Circuit Equations
% IL=C(dV2/dt)+V2/RL
% (Vs-V1)/Rs=IL+CdV1/dt
% V1-V2=L(dIL/dt)

%
% (L  0  0)     (IL)   (0     1    -1  )(IL)   (0 )
% (0 RsC 0) d/dt(V1) = (-Rs  -1     0  )(V1) + (Vs)
% (0  0  C)     (V2)   (1     0  -1/RL )(V2)   (0 )
%
```

```
%               IV=[IL V1 V2]';

A=[L 0  0; 0  Rs*C 0;0  0    C  ];
B=[0 1 -1;-Rs  -1  0;1  0  -1/RL];


IVdot=inv(A)*B*IV+inv(A)*[0;Vs;0];

global GLOBALT GLOBALIL GLOBALV1 GLOBALV2

if update
  GLOBALT= [GLOBALT;t];
  GLOBALIL=[GLOBALIL;IV(1)];
  GLOBALV1=[GLOBALV1;IV(2)];
  GLOBALV2=[GLOBALV2;IV(3)];
end

if mkplot
  plot(GLOBALT/ns,GLOBALIL*1000)
  xlabel('Time (nsec)'); ylabel('(ma)')
  title(['IL for RLC_RK.m ckt: C = ',num2str(C/pF),' pF & L =',num2str(L/nH), 'nH'])
  figure
  plot(GLOBALT/ns,GLOBALV1,GLOBALT/ns,GLOBALV2)
  xlabel('Time (nsec)'); ylabel('(volts)')
  title(['V1 & V2 for RLC_RK.m ckt: C = ',num2str(C/pF),' pF & L =',num2str(L/nH), 'nH'])
end
```

```
%Script file V2=RLC_chk
% to check solution of circuit defined by RLC_RK.m
% and solve via Runge-Kutta time stepping JHU_RK23
% - - - - - - - - - - - - - - - - - - -
%units
GHz=1e9; ns=1e-9;pF=1e-12;nH=1e-9;

%Component Values
Rs=50;RL=50;C=2*pF;L=10*nH;f=1*GHz; w=2*pi*f; Vm=1;
t=[0:.05*ns:3*ns];

Vs=Vm;      % phasor

Xc=-1/(w*C);
XL=w*L;
ZL=j*RL*Xc/(RL+j*Xc);
ZLprime=j*XL+ZL;
Z1=j*Xc*ZLprime/(ZLprime+j*Xc);
Zs=Rs+Z1;
Is=Vs/Zs;
V1=Is*Z1;
IL=V1/ZLprime;
ILt=abs(IL)*sin(w*t+angle(IL));
V1t=abs(V1)*sin(w*t+angle(V1));
V2=V1*ZL/(ZLprime);              %load phasor voltage
V2t=abs(V2)*sin(w*t+angle(V2));  %conveted to time soln
plot(t/ns,ILt*1000)
xlabel('time (nsec)')
ylabel('(ma)')
title('Phasor Soln (IL) to check RLC_RK ')
figure
plot(t/ns,V1t,t/ns,V2t)
xlabel('time (nsec)')
ylabel('(volts)')
title('Phasor Soln (V1 & V2) to check RLC_RK ')
```



The circuit illustrated in figure 3 illustrates a circuit with no reactive elements but it does have one non linear element, a diode. In this case the diode is assumed to have a resistance which either equals a small value, $R_{\min}$ or a large value, $R_{\max}$ depedning uput whether $v_d > 0$ or $v_d < 0$. Since the equations describing this circuit do not involve any terms with derivatives then the solution is an algebraic exercise which is illustrated by the MATLAB script file RD.m.

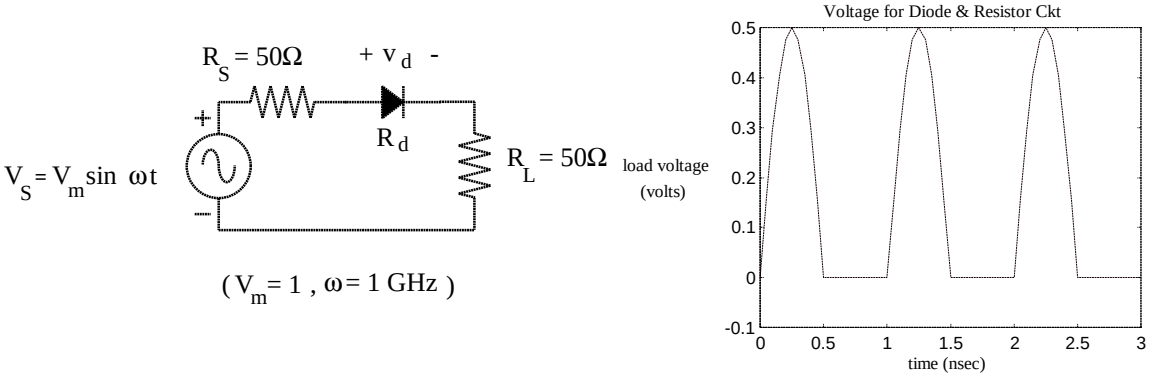


Figure 3. Example of a circuit with a non-linear element but no reactive elements

```
%script file RD.m to illustrate
%non-linear element without reactive elements

%            +  Vd -
%   ----Rs----->|---
%  |                |
%  Vs               RL
%  |                |
%   ----------------

%units
  GHz=1e9; ns=1e-9;
 Rs=50; RL=50;Rmin=1e-6; Rmax=1e6;
 f=1*GHz; w=2*pi*f;Vm=1;
 t=[0:.05*ns:3*ns];
```

```
 Vs=Vm*sin(w*t);

% Diode characteristic=variable resistance, Rd,equals
% "Rmin" when forward biased, "Rmax" when back biased
% Rd=Rmin+(Rmax-Rmin)(Vd<0) and since Vd<0 when Vs<0

  Rd=Rmin+(Rmax-Rmin)*(Vs<0);
  Is=Vs./(Rs+Rd+RL);
  VRL=Is*RL;

plot(t/ns,VRL)
xlabel('time (nsec)');
ylabel('load voltage (volts)');
title('Voltage for Diode & Resistor Ckt')
```

Figure 4 illustrates a circuit with both a non-linear element and a reactive element. Because of the derivative the circuit is solved using the Runge-Kutta time domain. Various values for the capacitor are substitued to see their effect on the solution. Note that the low capacitance case is very similar to the previous solution since for small capacitance the circuit is equivalent to the previous one.

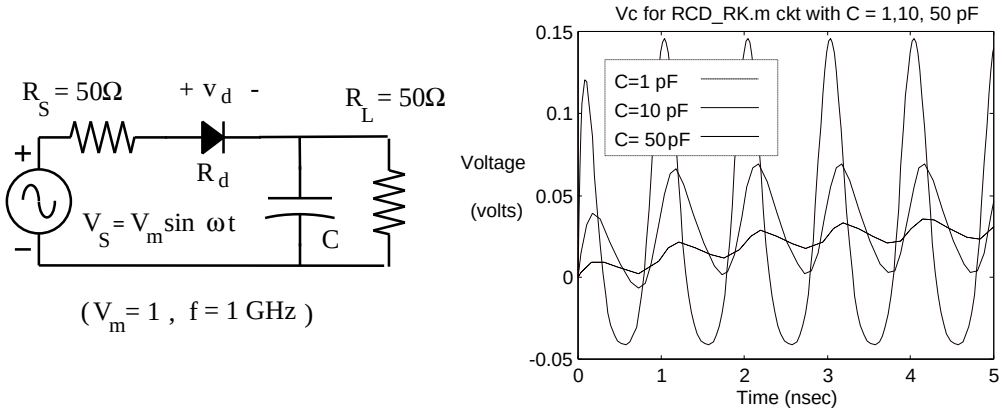


Figure 4. Example of a circuit

```
function Vcdot=RCD_RK(t,Vc,update,mkplot)
% Illustrates non-linear element with a reactive element
% funtion provides state derative for JHU_RK23.m
% - - - - - - - - - - - - - - - - - - - - - - - - - - - -
%                        (State variable)
%                    + Vd -    +Vc
%           .---Rs---->|------.-------.
%           |                 |       |
%  (source) Vs                C       RL (load)
%           |                 |       |
%           '-----------------'-------'
%   Vs=Vmsin(wt)
% - - - - - - - - - - - - - - - - - - - - - - - - - - - -
if nargin<3 update=0; end
if nargin<4 mkplot=0; end
```

```
global GLOBALT GLOBALVc GLOBALTMP
if GLOBALTMP==[] GLOBALTMP=0;end % Used to preserve last non-linear solution
if mkplot GLOBALTMP=[]; end      % to Diode IV curve for next starting point
                                 % Clears Tmp storage when solution completed
%units
 GHz=1e9; ns=1e-9;pF=1e-12;
%Circuit parameters
 k=2; Io=.001;
 Rs=50; RL=50;Rmin=1e-6; Rmax=1e6;
 f=1*GHz; w=2*pi*f;Vm=1;C=50*pF;

 Vs=Vm*cos(w*t);

% Diode modelled as Id=Io*(exp(k*Vd)-1)

% Id=C(dVc/dt)+Vc/RL
% Vs=Rs*Id+Vd+Vc
% Id=Io*(exp(k*Vd)-1)
% F(Vd)=Rs*Io*exp(k*Vd)+Vd-Rs*Io+Vc-Vs=0
% Solve for Vd using Newton's method with X=Vd
% Iterate Xn+1=Xn - F(Xn)/F'(Xn) until abs(F(Xn))<1e-4 =>Vd=Xn
% Id=(Vs-Vd-Vc)/Rs
% dVc/dt=(Id-Vc/RL)/C
% - - - - - - Routine for solving non-linear IV Diode relationship - - - -
 X=GLOBALTMP;
 FX=Rs*Io*exp(k*X)+X-Rs*Io+Vc-Vs;
 while abs(FX)>1e-6;
   FX=Rs*Io*exp(k*X)+X-Rs*Io+Vc-Vs;
   FpX=Rs*Io*k*exp(k*X)+1;
   X=X-FX/FpX;
 end
 GLOBALTMP=X;
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 Id=(Vs-X-Vc)/Rs;
 Vcdot=(Id-Vc/RL)/C ;



if update
  GLOBALT= [GLOBALT ; t];
  GLOBALVc=[GLOBALVc;Vc];
end

if mkplot
  plot(GLOBALT/ns,GLOBALVc)
  xlabel('Time (nsec)'); ylabel('Voltage (volts)')
  title(['Vc for RC_RK.m ckt with C = ',num2str(C/pF),' pF'])
end
```

Alternative Diode Model can also be used

```
% Diode characteristic=variable resistance, Rd,equals
% "Rmin" when forward biased, "Rmax" when back biased
% Rd=Rmin+(Rmax-Rmin)(Vd<0) and since Vd<0 when Vs-Vc<0

  Rd=Rmin+(Rmax-Rmin)*((Vs-Vc)<0);

% Id=C(dVc/dt)+Vc/RL
% Vs=Rs*Id+Rd*Id+Vc
% Id=(Vs-Vc)/(Rs+Rd)
% CdVc/dt=-Vc/RL + (Vs-Vc)/(Rs+Rd)

 Vcdot=(-Vc/RL + (Vs-Vc)/(Rs+Rd))/C ;
```

The circuit in figure 5 illustrates one with two sources and a non linear element. One source designated as the RF source is a sinewave signal. The other designated as the LO source is a squarewave signal. Since there are non reactive elements in the circuit it can be solve by algebraic techniques. which are implemented in the MATLAB file RD2Vp.m.

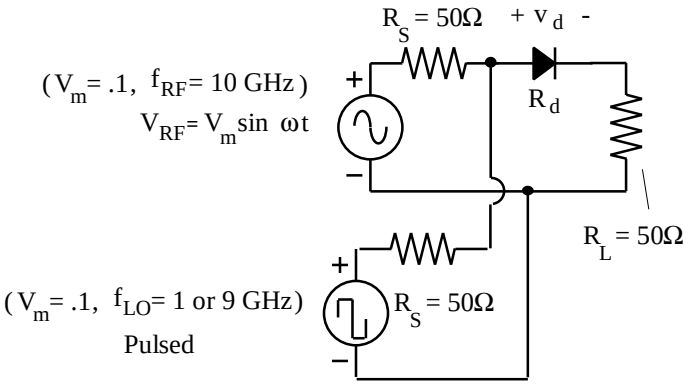
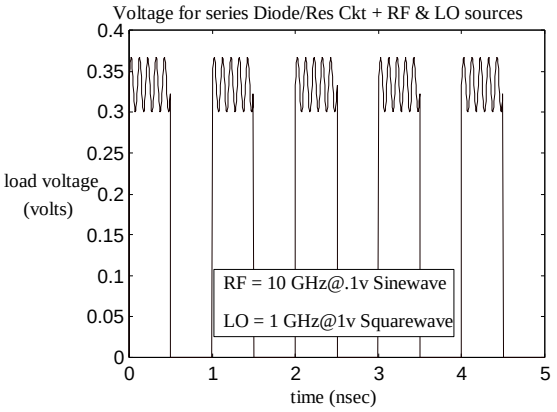
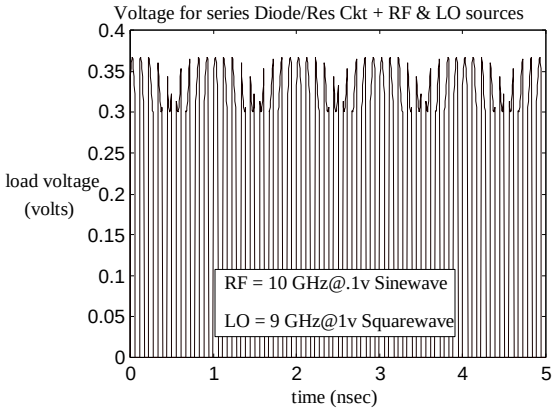


Figure 5. Example of a non-linear circuit with no reactive elements. Local Oscillator signal is a squarewave



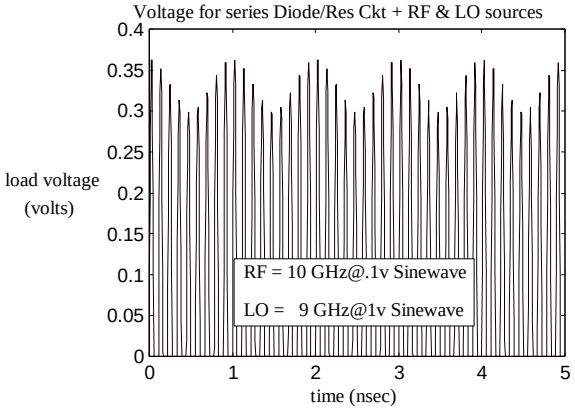


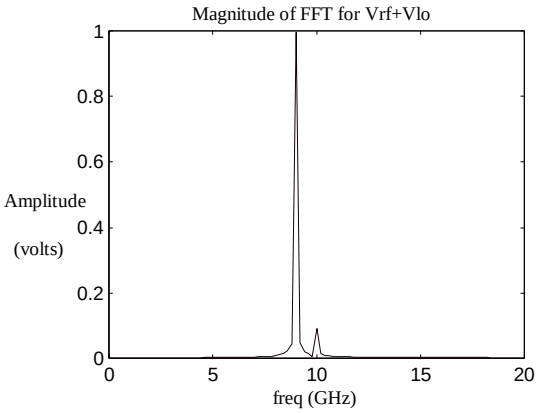
```
%script file RD2Vp.m to illustrate
%non-linear element without reactive elements
%LO square wave
%                  +V1       +VL
%   ----Rrf--.-------->|-----
%  | +       |               |
%  Vrf       |               RL
%  |         |               |
%   ---------|---.-----------
%            |   |
%            |   |
%   ----Rrf--'   |
%  |+            |
%  Vlo           |
%  |             |
%   -------------'
%
%units
 GHz=1e9; ns=1e-9;
 Rrf=50; Rlo=50;RL=50;Rmin=1e-6; Rmax=1e6;
 frf=10*GHz; wrf=2*pi*frf;Vmrf=.1;
 flo=1*GHz; wlo=2*pi*flo;Vmlo=1;
 t=[0:.005*ns:5*ns];
 Vrf=Vmrf*sin(wrf*t);
 Vlo=Vmlo*((sin(wlo*t)>0)-(sin(wlo*t)<0));

% (Vrf-V1)/Rrf+(Vlo-V1)/Rlo=V1/(Rd+RL)
% VL=RLV1/(Rd+RL)
% First equation becomes
% VLo/RLo+Vrf/Rrf=V1Y  where Y=1/RLo+1/Rrf+1/(Rd+RL)
% V1=(VLo/RLo+Vrf/Rrf)/Y and VL=RLV1/(Rd+RL)

% Diode characteristic=variable resistance, Rd,equals
% "Rmin" when forward biased, "Rmax" when back biased
% Rd=Rmin+(Rmax-Rmin)((V1-VL)<0) and since (V1-VL)<0 when (Vrf+Vlo)<0

 Rd=Rmin+(Rmax-Rmin)*((Vrf+Vlo)<0);
 Y=(1/Rrf+1/Rlo)*ones(size(t))+1./(RL+Rd);
 V1=(Vlo/Rlo+Vrf/Rrf)./Y;
 VL=(RL./(Rd+RL)).*V1;

plot(t/ns,VL)
xlabel('time (nsec)');
ylabel('load voltage (volts)');
title('Voltage for series Diode/Res Ckt + RF & LO sources')
```

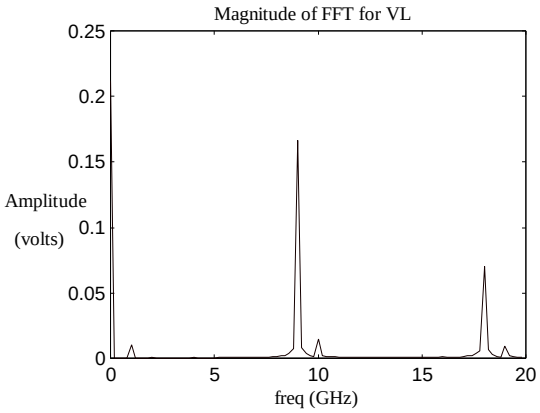
In the previous circuit the the local oscillator signal is changed to be a sinewave. The frequncy content of the output is also analyzed by taking an FFT of the time output. Note that the output consists of a sum and difference components.

```
%script file RD2Vq.m to illustrate
%non-linear element without reactive elements LO sine wave
%                 +V1       +VL
%   ----Rrf--.--------->|-----
%  | +       |               |
%  Vrf       |               RL
%  |         |               |
%   ---------|---.-----------
%            |   |
%   ----Rrf--'   |
%  |+            |
%  Vlo           |
%  |             |
%   -------------'
%
%units
 GHz=1e9; ns=1e-9;
 Rrf=50; Rlo=50;RL=50;Rmin=1e-6; Rmax=1e6;
 frf=10*GHz; wrf=2*pi*frf;Vmrf=.1;
 flo=9*GHz; wlo=2*pi*flo;Vmlo=1;
 deltat=.005*ns; tf=5*ns;t=[0:deltat:tf];
 Vrf=Vmrf*sin(wrf*t);
 Vlo=Vmlo*sin(wlo*t);

% (Vrf-V1)/Rrf+(Vlo-V1)/Rlo=V1/(Rd+RL)
% VL=RLV1/(Rd+RL)
% First equation becomes
% VLo/RLo+Vrf/Rrf=V1Y  where Y=1/RLo+1/Rrf+1/(Rd+RL)
% V1=(VLo/RLo+Vrf/Rrf)/Y and VL=RLV1/(Rd+RL)

% Diode characteristic=variable resistance, Rd,equals
% "Rmin" when forward biased, "Rmax" when back biased
% Rd=Rmin+(Rmax-Rmin)((V1-VL)<0) and since (V1-VL)<0 when (Vrf+Vlo)<0

 Rd=Rmin+(Rmax-Rmin)*((Vrf+Vlo)<0);
 Y=(1/Rrf+1/Rlo)*ones(size(t))+1./(RL+Rd);
 V1=(Vlo/Rlo+Vrf/Rrf)./Y;
 VL=(RL./(Rd+RL)).*V1;

plot(t/ns,VL), xlabel('time (nsec)');
ylabel('load voltage (volts)');
ti
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% Calculate and display Fourier transform, First of Vrf+Vlo and then for Vd

 fs=1/deltat; N=length(t)-1; deltaf=fs/N;
 freq=[0:deltaf:fs/2];
 FT=fft(Vrf+Vlo); ps=abs(FT(1:N/2))/(N/2); % scaled
 figure
 Nplot=100;
 plot(freq(1:Nplot)/GHz,ps(1:Nplot));
 xlabel('freq (GHz)');
 ylabel('Amplitude (volts)');
 title('Magnitude of FFT for Vrf+Vlo')

 FT=fft(VL); ps=abs(FT(1:N/2))/(N/2); % scaled
 figure'  plot(freq(1:Nplot)/GHz,ps(1:Nplot));
 xlabel('freq (GHz)');
 ylabel('Amplitude (volts)');
```

```
title('Magnitude of FFT for VL')
```

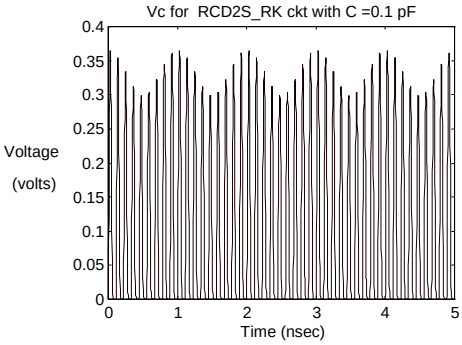


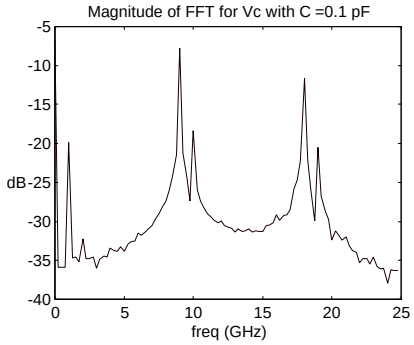


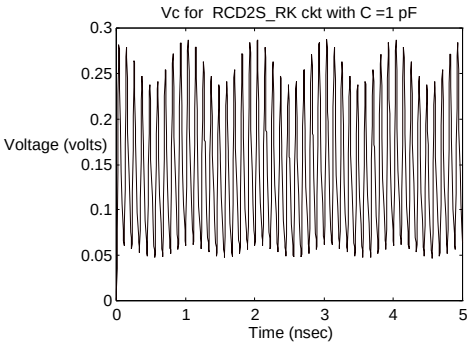


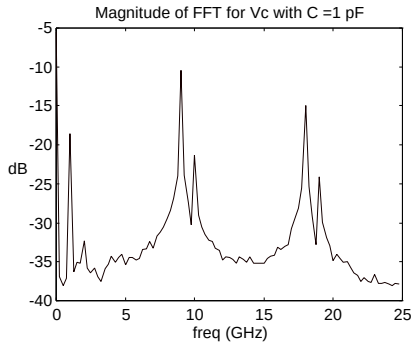

```
function Vcdot=RCD2s_RK(t,Vc,update,mkplot)
% to calculate the state derivative of a circuit
% with a non-linear element and reactive elements
% where RF and LO are both sine waves  Solve via JHU_RK23
%
%                 +V1       +VL
%   ----Rrf--.--------->|--.-----.
%  | +       |            |     |
%  Vrf       |            C     RL
%  |         |            |     |
%   ---------|---.--------'-----'
%            |   |
%            |   |
%   ----Rrf--'   |
%  |+            |
%  Vlo           |
%  |             |
%   -------------'
%
if nargin<3 update=0; end
if nargin<4 mkplot=0; end
global GLOBALT GLOBALVc
```

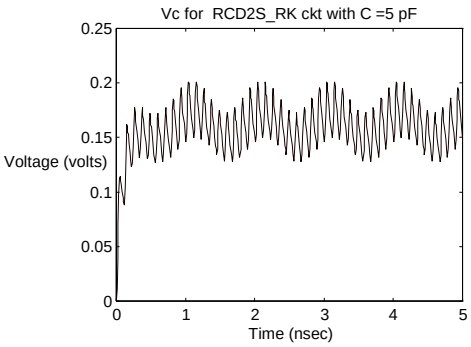
```
%units
 GHz=1e9; ns=1e-9; pF=1e-12;
 Rrf=50; Rlo=50;RL=50;Rmin=1e-6; Rmax=1e6;
 frf=10*GHz; wrf=2*pi*frf;Vmrf=.1;
 flo=9*GHz; wlo=2*pi*flo;Vmlo=1;
 Vrf=Vmrf*sin(wrf*t);
 Vlo=Vmlo*sin(wlo*t);
 C=1*pF;

% (Vrf-V1)/Rrf+(Vlo-V1)/Rlo=CdVc/dt+Vc/RL
% (V1-Vc)/Rd=(Vrf-V1)/Rrf+(Vlo-V1)/Rlo
% First equation becomes
% V1=(VLo/RLo+Vrf/Rrf+Vc/Rd)/Y   where Y=1/RLo+1/Rrf+1/Rd
% dVc/dt=(Vrf/Rrf+Vlo/Rlo-V1(1/Rrf+1/Rlo)-Vc/RL)/C

% Diode characteristic=variable resistance, Rd,equals
% "Rmin" when forward biased, "Rmax" when back biased
% Rd=Rmin+(Rmax-Rmin)((V1-Vc)<0)

 Rd=Rmin+(Rmax-Rmin)*((Vrf+Vlo)<0);
 Y=1/Rlo+1/Rrf+1/Rd;
 V1=(Vlo/Rlo+Vrf/Rrf+Vc/Rd)/Y;
 Vcdot=(Vrf/Rrf+Vlo/Rlo-V1*(1/Rrf+1/Rlo)-Vc/RL)/C;

if update
  GLOBALT= [GLOBALT ; t];
  GLOBALVc=[GLOBALVc;Vc];
end

if mkplot
  plot(GLOBALT/ns,GLOBALVc)
  xlabel('Time (nsec)'); ylabel('Voltage (volts)')
  title(['Vc for  RCD2S_RK ckt with C =',num2str(C/pF),' pF'])

% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% Calculate and display Fourier transform of Vd
 deltat=.005*ns;tf=max(GLOBALT);
 uniformt=[1*ns:deltat:tf];           %uniform time steps delayed 1ns for
 uniformV=interp1(GLOBALT,GLOBALVc,uniformt');    %interprolating values from Vc

 fs=1/deltat; N=length(uniformt)-1; deltaf=fs/N;
 freq=[0:deltaf:fs/2];
 FT=fft(uniformV); ps=10*log10((abs(FT(1:N/2))/(N/2))); % scaled
 figure
 Nplot=100;
 plot(freq(1:Nplot)/GHz,ps(1:Nplot));
 xlabel('freq (GHz)');
 ylabel('dB');
 title(['Magnitude of FFT for Vc with C =',num2str(C/pF),' pF'])
end
```

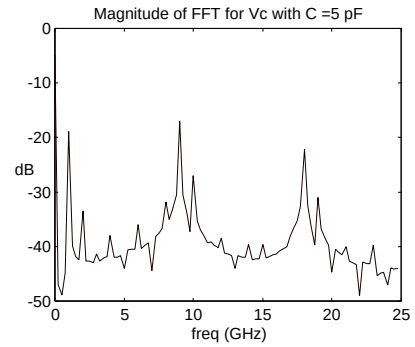




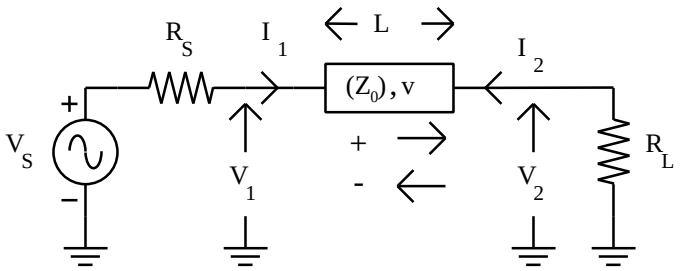








Transmission Lines (Delay Lines)



$$V_1 = V_1^+ + V_1^- \qquad V_2 = V_2^+ + V_2^-$$
$$Z_o I_1 = V_1^+ - V_1^- \qquad -Z_o I_2 = V_2^+ - V_2^-$$

$$2V_1^+ = V_1 + Z_o I_1 \qquad 2V_2^+ = V_2 - Z_o I_2$$
$$2V_1^- = V_1 - Z_o I_1 \qquad 2V_2^- = V_2 + Z_o I_2$$

$$V_2^+ = V_1^+(t - L/v)$$
$$V_1^- = V_2^-(t - L/v)$$

$$V_2(t) - Z_o I_2(t) = V_1(t - L/v) + Z_o I_1(t - L/v)$$

$$V_1(t) - Z_o I_1(t) = V_2(t - L/v) + Z_o I_2(t - L/v)$$

$$I_1 = \frac{1}{R_S}(V_S - V_1)$$

$$I_2 = -\frac{1}{R_L}V_2$$

Initial Conditions: $I_1(t) = I_2(t) = V_1(t) = V_2(t) = 0 \; for \; t < 0$

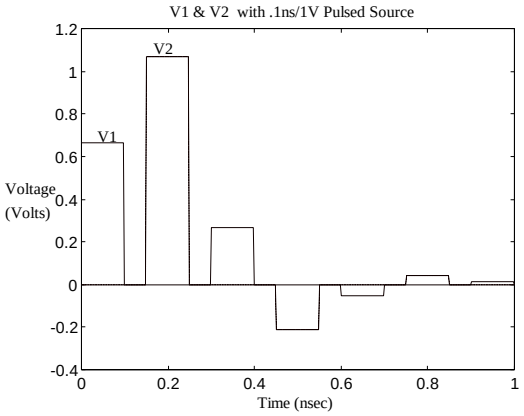
```
function DelayOut=Delay(t_hist,v_hist,t)
% DelayOut=Delay(t_history,v_history,t)
```

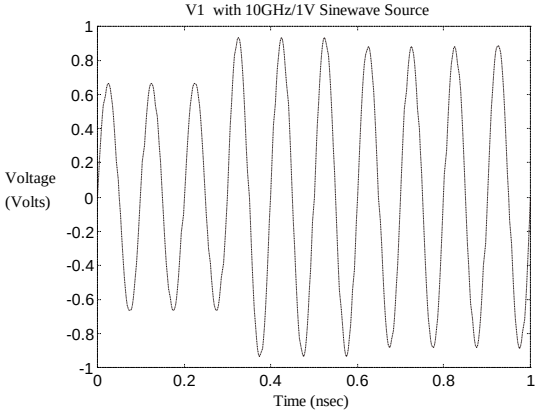
```
% function to provide delayed output for solving
% time domain transmission line problems.  t_hist and v_hist
% are column vectors.  If t<t_hist(1) or t_hist=v_hist=[]
% then DelayOut=0 otherwise interpolation occurs

[m,n]=size(t_hist); [p,q]=size(v_hist); [r,s]=size(t);


if m==0 & n==0 & p==0 & q==0;         %check for empty matrice (occurs at calculation
start-up)
   DelayOut=0;
else
   if (n~=1) | (q~=1) | (s~=1);
     error('*** Input to Delay function must be column vectors ***')
   else
     if m~=p;
       error('*** delay column vectors must be of same length ***')
     else
       if t<t_hist(1);
         DelayOut=0;
       else
         DelayOut=interp1(t_hist,v_hist,t);
       end
     end
   end
end
```

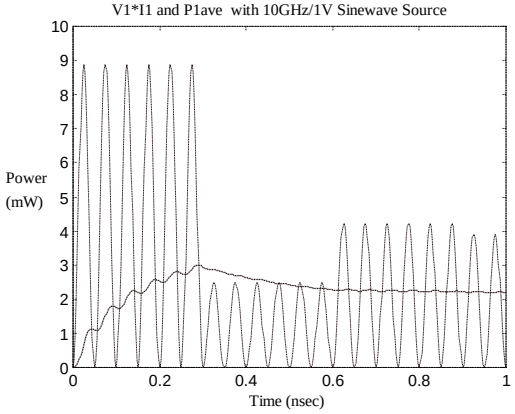
```
function [deltat,Nsteps,X]=Setup(SigPeriod,Ns,Nd,Tdelay,Tf,U)
%function sets parameters for time stepping with Tlin's
Tmin=min(Tdelay);
Tmax=max(Tdelay);
deltat=min(Tmin/Nd,SigPeriod/Ns);
Nsteps=ceil(Tf/deltat);
NumVar=max(size(U));
Nneg=ceil(Tmax/deltat);
X=[deltat*[-Nneg:1:-1]',zeros(Nneg,NumVar)];
```

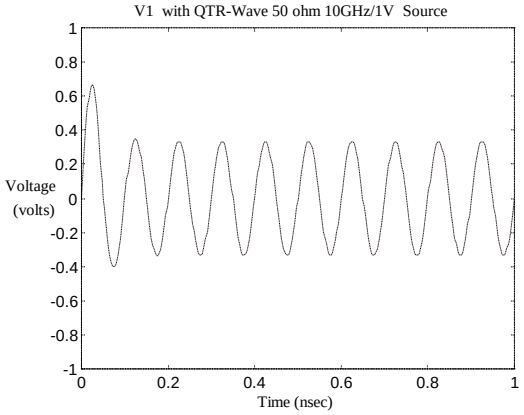
```
% script file:  tlin_tda.m
% for calculating time domain solution of transmission line with source and load
%
%               I1    --------    I2
```

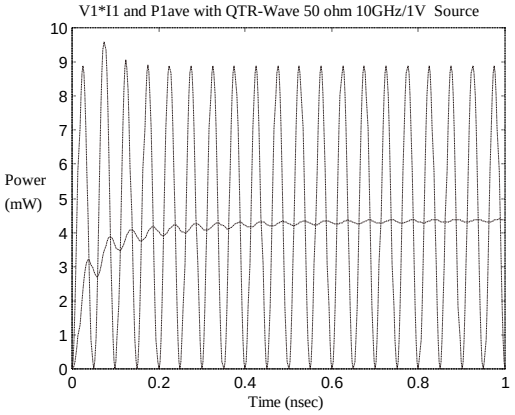
```
%       ----Rs---->---|Za,va,La|---<------
%      |      +V1      --------     +V2   |
%      Vs                                 RL
%      |                                  |
%      gnd                               gnd
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
%          Junction Equations                |    * * *Node Matrices * *       |
%    V1(t)-Za*I1(t)=V2(t-La/va)+Za*I2(t-La/va) |   (1 -Za)(V1) (V2A+Za*I2A)      |
%    V1(t)+Rs*I1(t)=Vs(t)                    |   (1 +Rs)(I1)=(   Vs     )      |
%                                            |                                 |
%    V2(t)-Za*I2(t)=V1(t-La/va)+Za*I1(t-La/va) |   (1 -Za)(V2) (V1A+Za*I1A)      |
%    V2(t)+RL*I2(t)=0                        |   (1 +RL)(I2)=(   0      )      |
%                                            |  where V2A=V2(t-La/va), etc     |
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
%units
GHz=1e9; ns=1e-9;Vo=1;f=10e9;w=2*pi*f;Tf=1*ns;

% * * * *Circuit* * * * * *
Rs=25;RL=200;   %Source/Load Parameters

% Tlin: Char Impedance; Length (cm); prop vel (cm/sec);Time delay (sec)
Za=50; La=3; va=2e10; Ta=La/va;

U1=inv([1 -Za;1 Rs]);
U2=inv([1 -Za;1 RL]);
U=[U1,zeros(size(U2));zeros(size(U1)),U2];

% * * * *Set up Step Size, etc. * * * *
%[Time,IVvar]<==[SigPeriod,Sigsamples,Delaysamples,{Set of delay times],Tf,Umatrix]
Ns=40;
[deltat,Nsteps,X]=setup(1/f,Ns,4,[Ta],Tf,U);

% * * * *Step thru Calulation * * *
for M=0:Nsteps   % Each time step "M" determines volt/currents from previous values
  t=deltat*M;
  VIA= interp1(X(:,1),X(:,2:max(size(U))+1),t-Ta); % X=[t.V1,I1,V2,I2] delayed by Ta
  V1A=VIA(1);  I1A=VIA(2); V2A=VIA(3);   I2A=VIA(4);

% Vs=Vo*sin(w*t);    %Sinewave Source
  Vs=Vo*(t<.1*ns);   %SquareWave Source

  Source=[[V2A+Za*I2A;Vs];[V1A+Za*I1A;Vs]];   % calculate voltage/current
  P=U*Source;
  X=[X;[t,P']];   % appends time & converts from column to row and addes to bottom of X
end
  Nneg=nnz(X(:,1)<0);Npos=[Nneg+1:length(X(:,1))];    % Steps with positive time values
  T=1e9*X(Npos,1); V1=X(Npos,2);  I1=X(Npos,3);   V2=X(Npos,4);  I2=X(Npos,5);

% * * * * *Plot Routine * * * * *

Pwr=V1.*I1;
AvePwr=WinAve(Pwr',Ns);
plot(T,V1); title('V1  with QTR-Wave sqrt(5000) ohm 10GHz/1V  Source');
axis([0 1e9*Tf -1 1]);xlabel('Time (nsec)'); ylabel('Voltage (volts)')
figure
plot(T,Pwr*1000,T,AvePwr*1000); title('Pin & Pinave with QTR-Wave/sqrt(5000) ohm Line');
axis([0 1e9*Tf 0 10]);xlabel('Time (nsec)'); ylabel('Power (mW)')
```

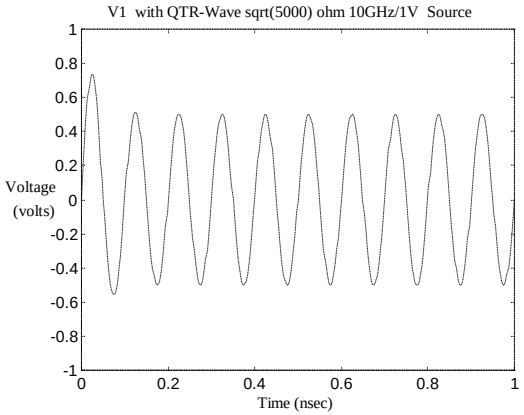


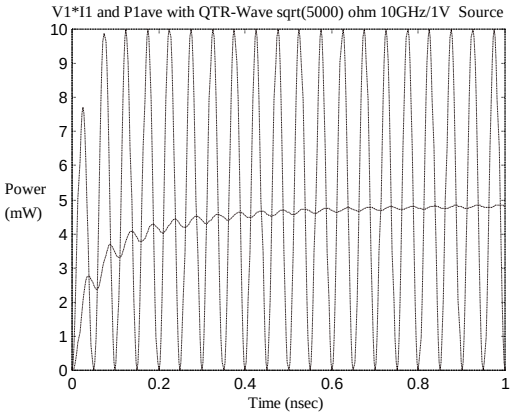


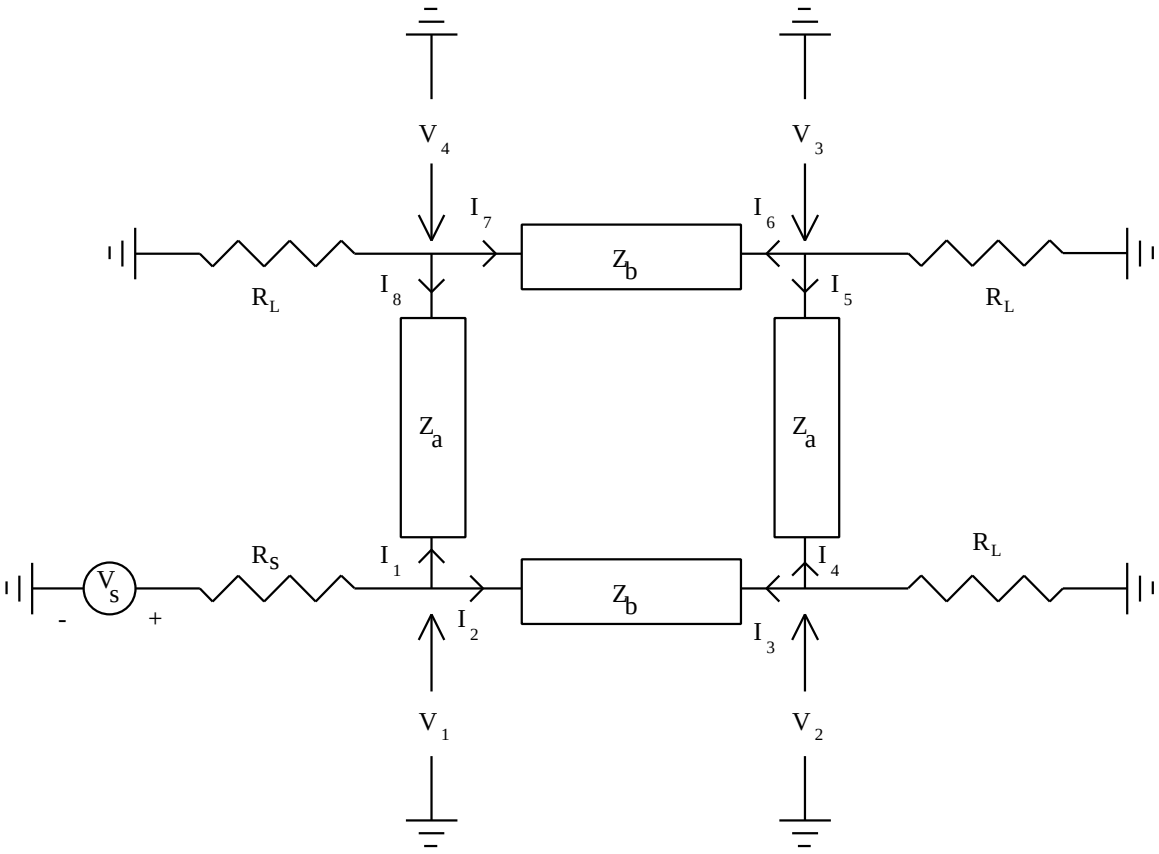












Notation: $V_n = V_n(t), \quad I_n = I_n(t)$

$$V_1 + R_s I_1 + R_s I_2 = V_s$$
$$V_1 - Z_a I_1 = V_4(t - T_a) + Z_a I_8(t - T_a)$$
$$V_1 - Z_b I_2 = V_2(t - T_b) + Z_b I_3(t - T_b)$$

$$V_2 + R_L I_3 + R_L I_4 = 0$$
$$V_2 - Z_a I_4 = V_3(t - T_a) + Z_a I_5(t - T_a)$$
$$V_2 - Z_b I_3 = V_1(t - T_b) + Z_b I_2(t - T_b)$$

$$V_3 + R_L I_5 + R_L I_6 = 0$$
$$V_3 - Z_a I_5 = V_2(t - T_a) + Z_a I_4(t - T_a)$$
$$V_3 - Z_b I_6 = V_4(t - T_b) + Z_b I_7(t - T_b)$$

$$V_4 + R_L I_7 + R_L I_8 = 0$$
$$V_4 - Z_a I_8 = V_1(t - T_a) + Z_a I_1(t - T_a)$$
$$V_4 - Z_b I_7 = V_3(t - T_b) + Z_b I_6(t - T_b)$$

$$\begin{pmatrix}1 & R_s & R_s\\ 1 & -Z_a & 0\\ 1 & 0 & -Z_b\end{pmatrix}\begin{pmatrix}V_1\\ I_1\\ I_2\end{pmatrix}=\begin{pmatrix}V_s\\ V_4(t-T_a)\\ V_2(t-T_b)\end{pmatrix}+\begin{pmatrix}0\\ Z_aI_8(t-T_a)\\ Z_bI_3(t-T_b)\end{pmatrix}$$

$$\begin{pmatrix}1 & R_L & R_L\\ 1 & -Z_a & 0\\ 1 & 0 & -Z_b\end{pmatrix}\begin{pmatrix}V_2\\ I_4\\ I_3\end{pmatrix}=\begin{pmatrix}0\\ V_3(t-T_a)\\ V_1(t-T_b)\end{pmatrix}+\begin{pmatrix}0\\ Z_aI_5(t-T_a)\\ Z_bI_2(t-T_b)\end{pmatrix}$$

$$\begin{pmatrix}1 & R_L & R_L\\ 1 & -Z_a & 0\\ 1 & 0 & -Z_b\end{pmatrix}\begin{pmatrix}V_3\\ I_5\\ I_6\end{pmatrix}=\begin{pmatrix}0\\ V_2(t-T_a)\\ V_4(t-T_b)\end{pmatrix}+\begin{pmatrix}0\\ Z_aI_4(t-T_a)\\ Z_bI_7(t-T_b)\end{pmatrix}$$

$$\begin{pmatrix}1 & R_L & R_L\\ 1 & -Z_a & 0\\ 1 & 0 & -Z_b\end{pmatrix}\begin{pmatrix}V_4\\ I_8\\ I_7\end{pmatrix}=\begin{pmatrix}0\\ V_1(t-T_a)\\ V_3(t-T_b)\end{pmatrix}+\begin{pmatrix}0\\ Z_aI_1(t-T_a)\\ Z_bI_6(t-T_b)\end{pmatrix}$$

$$U_1=\begin{pmatrix}1 & R_s & R_s\\ 1 & -Z_a & 0\\ 1 & 0 & -Z_b\end{pmatrix}^{-1}\quad U_2=\begin{pmatrix}1 & R_L & R_L\\ 1 & -Z_a & 0\\ 1 & 0 & -Z_b\end{pmatrix}^{-1}\quad U_3=\begin{pmatrix}1 & R_L & R_L\\ 1 & -Z_a & 0\\ 1 & 0 & -Z_b\end{pmatrix}^{-1}\quad U_4=\begin{pmatrix}1 & R_L & R_L\\ 1 & -Z_a & 0\\ 1 & 0 & -Z_b\end{pmatrix}^{-1}$$

$$\begin{pmatrix}V_1\\ I_1\\ I_2\end{pmatrix}=U_1\cdot\begin{pmatrix}V_s\\ V_4(t-T_a)\\ V_2(t-T_b)\end{pmatrix}+U_1\cdot\begin{pmatrix}0\\ Z_aI_8(t-T_a)\\ Z_bI_3(t-T_b)\end{pmatrix}$$

$$\begin{pmatrix}V_2\\ I_4\\ I_3\end{pmatrix}=U_2\cdot\begin{pmatrix}0\\ V_3(t-T_a)\\ V_1(t-T_b)\end{pmatrix}+U_2\cdot\begin{pmatrix}0\\ Z_aI_5(t-T_a)\\ Z_bI_2(t-T_b)\end{pmatrix}$$

$$\begin{pmatrix}V_3\\ I_5\\ I_6\end{pmatrix}=U_3\begin{pmatrix}0\\ V_2(t-T_a)\\ V_4(t-T_b)\end{pmatrix}+U_3\begin{pmatrix}0\\ Z_aI_4(t-T_a)\\ Z_bI_7(t-T_b)\end{pmatrix}$$

$$\begin{pmatrix}V_4\\ I_8\\ I_7\end{pmatrix}=U_4\cdot\begin{pmatrix}0\\ V_1(t-T_a)\\ V_3(t-T_b)\end{pmatrix}+U_4\cdot\begin{pmatrix}0\\ Z_aI_1(t-T_a)\\ Z_bI_6(t-T_b)\end{pmatrix}$$

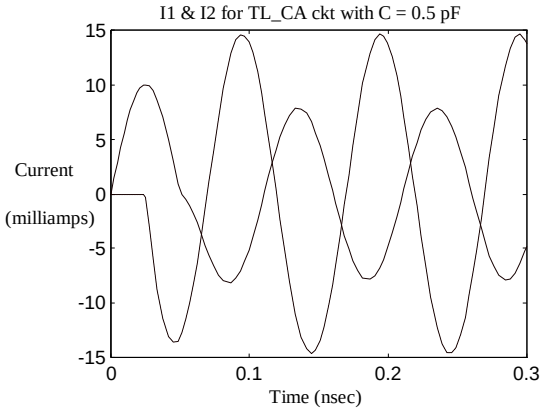
$$TV_1I_1I_2V_2I_4I_3V_3I_5I_6V_4I_8I_7$$

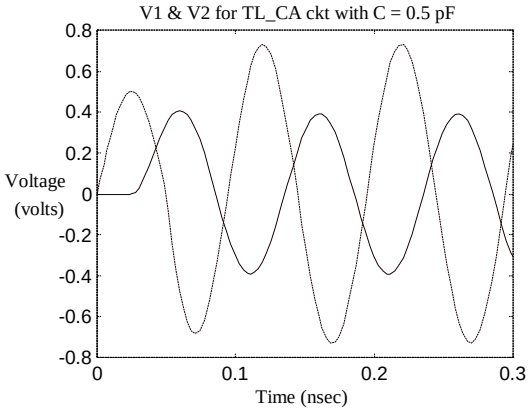
$$X_{1,1}X_{1,2}X_{13},X_{1,4}X_{1,5}X_{1,6}X_{1,7}X_{1,8}X_{1,9}X_{1,10}X_{1,11}X_{1,12}X_{1,13}$$

```
[t,V2]=>>JHUODE23('TL_C_RK',t0,tf,V20);
```

```
function V2dot=TL_C_RK(t,V2,update,mkplot)
% to investigate an ideal diode and transmision line.
%
%                I1                  I2      +V3
%        .---Rs-->--(TL=Za,La,va)--<---.------.
%        |        +V1             +V2   |      |
%        Vs                             C      RL
%        |                              |      |
```

```
%       gnd                              gnd   gnd
%- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% ( 1 -Za )( V1 ) ( V2(t-Ta)+Za*I2(t-Ta) )                                  |
% ( 1 +Rs )( I1 )=(          Vs          )                                  |
%                                                                           |
% ( 1 -Za )( V2 ) ( V1(t-Ta)+Za*I1(t-Ta)              )                    |
% ( 1 +RL )( I2 )=(           0          -RL*C*dV2/dt )                    |
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
if nargin<3 update=0; end
if nargin<4 mkplot=0; end
global GLOBALT GLOBALV1 GLOBALI1 GLOBALV2 GLOBALI2

%units
GHz=1e9;ns=1e-9;pF=1e-12;f=10*GHz; w=2*pi*f;
Za=50;La=.5;va=2e10;Ta=La/va;RL=50;Rs=50;C=.5*pF;Vm=1;

%Delayed current/voltage values
  V1A=delay(GLOBALT,GLOBALV1,t-Ta);  I1A=delay(GLOBALT,GLOBALI1,t-Ta);
  V2A=delay(GLOBALT,GLOBALV2,t-Ta);  I2A=delay(GLOBALT,GLOBALI2,t-Ta);
  Vs=Vm*sin(w*t);

%New current/voltage values
  IV=inv([1 -Za;1 +Rs])*[V2A+Za*I2A;Vs];
  V1=IV(1);  I1=IV(2);
  I2=(V2-V1A-Za*I1A)/Za;
  V2dot=-(V2+RL*I2)/(RL*C);

if update
  GLOBALT= [GLOBALT ; t];  GLOBALV1=[GLOBALV1;V1];  GLOBALI1=[GLOBALI1;I1];
  GLOBALV2=[GLOBALV2;V2];  GLOBALI2=[GLOBALI2;I2];end

if mkplot
  plot(GLOBALT/ns,GLOBALV1,GLOBALT/ns,GLOBALV2)
  xlabel('Time (nsec)'); ylabel('Voltage (volts)')
  title(['V1 & V2 for TL_CA ckt with C = ',num2str(C/pF),' pF'])
  figure'   plot(GLOBALT/ns,1000*GLOBALI1,GLOBALT/ns,1000*GLOBALI2)
  xlabel('Time (nsec)'); ylabel('Current (milliamps)')
  title(['I1 & I2 for TL_CA ckt with C = ',num2str(C/pF),' pF'])
end
```

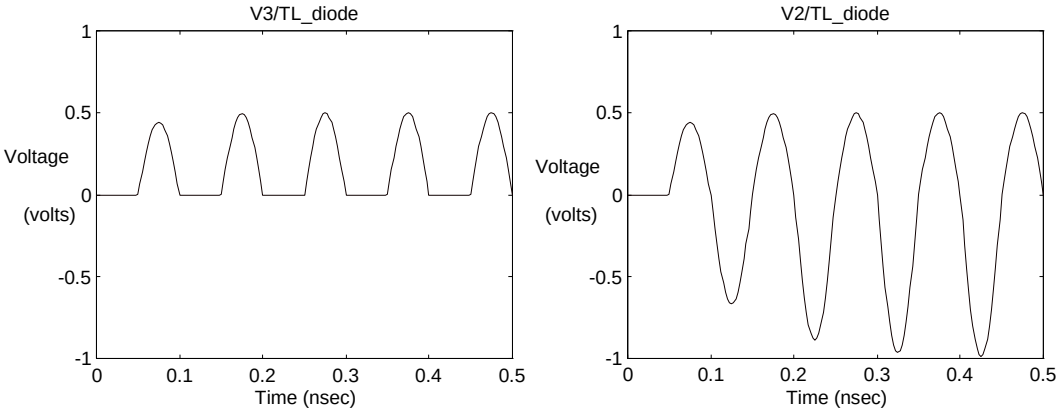


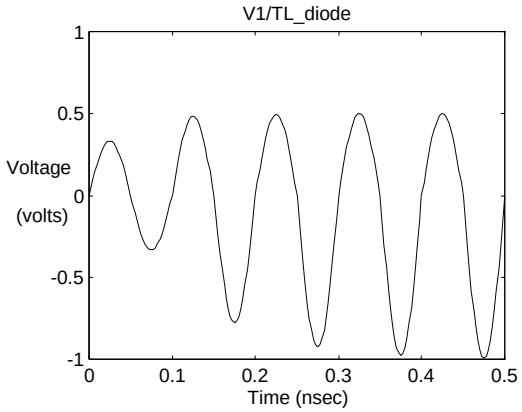


Transmission Line and Diode

```
%                I1                 I2      +V3
%        .---Rs-->--(TL=Z1,L1,v1)-<---->|-----.

%        |        +V1            +V2         |
%        Vs                                  RL
%        |                                   |
%       gnd                                 gnd
%- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% (1 -Za)(V1) (V2(t-Ta))   (I2(t-Ta))      |
% (1 +Rs)(I1)=(  Vs    )+Za(  0     )      |
%                                          |   Rd=Rshort+(Ropen-Rshort)*(V2>0)
% (1 -Za  0)(V2) (V1(t-Ta))   (I1(t-Ta))   |   Solve both cases and choose V2>0
% (1 +Rd -1)(I2)=(  0     )+Za(  0     )   |
```

```
% (0  RL  1)(V3) (  0     )   (  0     )
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
%units
GHz=1e9;ns=1e-9;Vo=1;f=10*GHz;w=2*pi*f;Tf=.5*ns;

% * * * *Circuit* * * * * *
Rs=100;RL=100;Rdshort=0;Rdopen=1e9;
% Tlin:Char Imped;Length (cm);vel (cm/s);Delay (s)
Za=50;La=1;va=2e10;Ta=La/va;

U1=inv([1 -Za;1 Rs]);
U2short=inv([1 -Za  0;1 Rdshort  -1;0  RL  1]);
U2open =inv([1 -Za  0;1 Rdopen   -1;0  RL  1]);
Ushort=[U1,zeros(2,3);zeros(3,2),U2short];
Uopen =[U1,zeros(2,3); zeros(3,2),U2open ];

% * * * *Set up Step Size, etc. * * * *
%[Time,IVvar]<==[SigPeriod,Sigsamples,Delaysamples,{Set of delay times],Tf,Umatrix]
Ns=40;
[deltat,Nsteps,X]=setup(1/f,Ns,4,[Ta],Tf,Ushort);

% * * * *Step thru Calulation * * *
for M=0:Nsteps   % for each time step "M" determine volt/currents from previous
  t=deltat*M;     % X=[t.V1,I1,V2,I2,V3]

  VIA= interp1(X(:,1),X(:,2:6),t-Ta);
  V1A=VIA(1);  I1A=VIA(2); V2A=VIA(3);   I2A=VIA(4);

  Vs=Vo*sin(w*t);   % Source
  P1=U1*[V2A;Vs]+U1*[Za*I2A;0];     % calculate voltage/current

    P2short=U2short*[V1A;0;0]+U2short*[Za*I1A;0;0];
    P2open =U2open*[V1A;0;0 ]+U2open*[Za*I1A;0;0 ];
    on=(P2open(1)>0);     off=(P2open(1)<=0);

  P2=on*P2short+off*P2open;
  P=[t,[P1;P2]'];
  X=[X;P];
end

  Nneg=nnz(X(:,1)<0);Npos=[Nneg+1:length(X(:,1))];    % Steps with positive time values
  T=1e9*X(Npos,1);V1=X(Npos,2); I1=X(Npos,3); V2=X(Npos,4); I2=X(Npos,5); V3=X(Npos,6);

%* * * * * *Plotting * * * * * * *
Vmax=1;
plot(T,V1);title('V1/TL_diode');axis([0 Tf/ns -Vmax Vmax]);
xlabel('Time (nsec)');ylabel('Voltage (volt)');
figure' plot(T,V2);title('V2/TL_diode');axis([0 Tf/ns -Vmax Vmax]);
xlabel('Time (nsec)');ylabel('Voltage (volt)');
figure' plot(T,V3);title('V3/TL_diode');axis([0 Tf/ns -Vmax Vmax]);
xlabel('Time (nsec)');ylabel('Voltage (volt)');
```

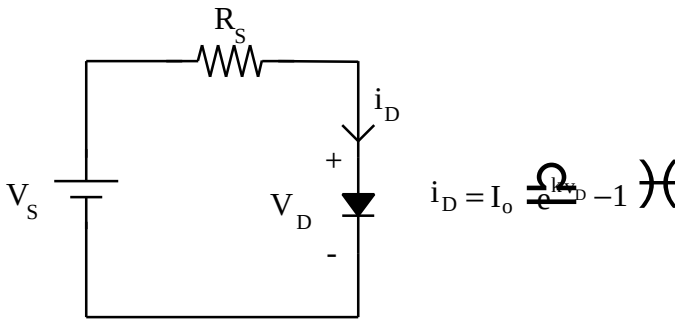




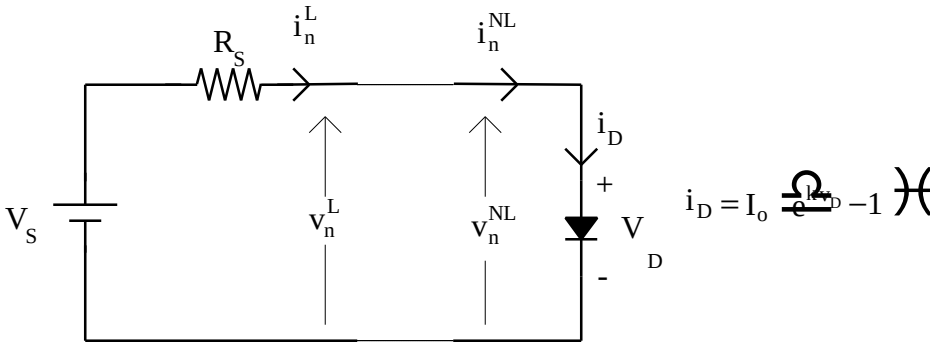
HARMONIC BALANCE

The harmonic balance technique is based upon describing time dependent signals as a finite Fourier series and applying Kirchhoff's Laws at the junction between non-linear elements and the linear elements of a circuit. An iterative approach is employed and convergence is determined by comparing the closeness of each of the Fourier components. The technique is illustrated by several examples which are solved in detail with subsequent examples increasing in complexity. MATLAB is used in a manual fashion to make the process explicit while eliminating unnecesary tedium. It should be clear to the reader that the process easily lends its self to automation.

The iteration technique will be illustrated considring a DC problem. This can be thought of as a harmonic balance solution where only the DC term of the Finite Fourier Series is considered. The circuit is illustrated below. The circuit is partitioned into linear an non-linear subcircuits. This partitioning is illustrated in the next figure.



$$i_D = I_o\left(e^{k \cdot v_D} - 1\right)$$



$$i_n^L = \frac{V_s - v_n^L}{R_S}$$

$$i_n^{NL} = i_n^L$$

Letting $i_D = i_n^{NL}$ and $v_D = v_n^{NL}$ in the above current -voltage relationship for the diode and solving for v_n^{NL} yeilds:

$$v_n^{NL} = \frac{1}{k}\ln\left(\frac{i_n^{NL}}{I_o} + 1\right)$$

$$Error(n) = v_n^{NL} - v_n^L$$

If Error=0 then the iteration has converged and the answer satisfies Kirckoff's Laws and is the physical solution to the non-linear problem. In practice the error only has to have an absolute value that is sufficiently small. If the after the "n th" iteration the error is too large then an "n+1" is started by defining a new linear voltage to apply to the linear part of the circuit, i.e.,

$$v_{n+1}^L = v_n^L + C \cdot Error(n) with (0 \le C \le 1)$$

The process is illustrated with a diode where

$$\boxed{\begin{aligned} V_S &= 3volts \\ R_S &= 50\Omega \\ k &= 2 \\ I_o &= 1ma = .001a \end{aligned}}$$

by assuming a starting voltage of

$$v_0^L = 0volts$$

$$i_0^L = \frac{3-0}{50} = .06a$$

$$i_0^{NL} = i_0^L = .06\ a$$

$$v_0^{NL} = \frac{1}{k}\ln\left(\frac{i_0^{NL}}{I_o}+1\right) = 2.0554\text{volts}$$

$$Error(0) = v_0^{NL} - v_0^L = 2.0554 - 0 = 2.0554$$

If the weighting factor C=.5 is used then

$$v_1^L = v_0^L + C \cdot Error(0) = 0 + .5(2.0554) = 1.0277volts$$

The process now repeats. These results and their continuation is shown in table 1a. Table 1a, b, c illustrate the iteration technique where different starting voltages are used. The same weighting factor "C=.5" is used in all threee cases.

Table 1 Iterative Technique with different starting voltages {0, 3, .-3}
and constant weigthing factor C=.5

(Table 1a: $v_0^L = 0volts$)

$n \Rightarrow$	0	1	2	3	4	5	6
v_n^L	0	1.0277	1.4388	1.5876	1.6378	1.6541	1.6594
$i_n^L = i_n^{NL}$	0.0600	0.0394	0.0312	0.0282	0.0272	0.0269	0.0268
v_n^{NL}	2.0554	1.8500	1.7363	1.6879	1.6705	1.6646	1.6627
$v_n^{NL} - v_n^L$	2.0554	0.8223	0.2975	0.1003	0.0327	0.0105	0.0034

(Table 1b: $v_0^L = 3volts$)

$n \Rightarrow$	0	1	2	3	4	5	6
v_n^L	3.0000	1.5000	1.6085	1.6446	1.6563	1.6601	1.6613
$i_n^L = i_n^{NL}$	0	0.0300	0.0278	0.0271	0.0269	0.0268	0.0268
v_n^{NL}	0	1.7170	1.6807	1.6680	1.6638	1.6625	1.6621
$v_n^{NL} - v_n^L$	-3.0000	0.2170	0.0722	0.0234	0.0075	0.0024	0.0008

(Table 1c: $v_0^L = -3volts$)

$n \Rightarrow$	0	1	2	3	4	5	6
v_n^L	-3.0000	-0.3011	0.9007	1.3906	1.5708	1.6322	1.6523
$i_n^L = i_n^{NL}$	0.1200	0.0660	0.0420	0.0322	0.0286	0.0274	0.0270
v_n^{NL}	2.3979	2.1025	1.8804	1.7511	1.6936	1.6724	1.6653
$v_n^{NL} - v_n^L$	5.3979	2.4036	0.9797	0.3605	0.1228	0.0402	0.0130

Table 2: Iterative Technique with different weighting factors {.25, .5, .75}
and constant staring voltage of 0 volts

Table 2a: C=.25

$n \Rightarrow$	0	1	2	3	4	5	6
v_n^L	0	0.5139	0.8762	1.1287	1.3026	1.4212	1.5013
$i_n^L = i_n^{NL}$	0.0600	0.0497	0.0425	0.0374	0.0339	0.0316	0.0300
v_n^{NL}	2.0554	1.9632	1.8861	1.8244	1.7769	1.7418	1.7166

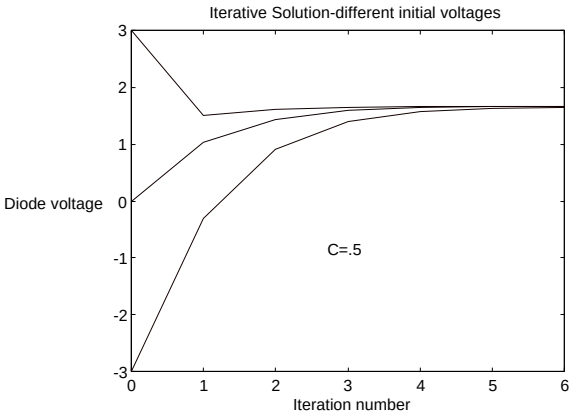
$v_n^{NL} - v_n^L$	2.0554	1.4493	1.0099	0.6957	0.4743	0.3206	0.2152

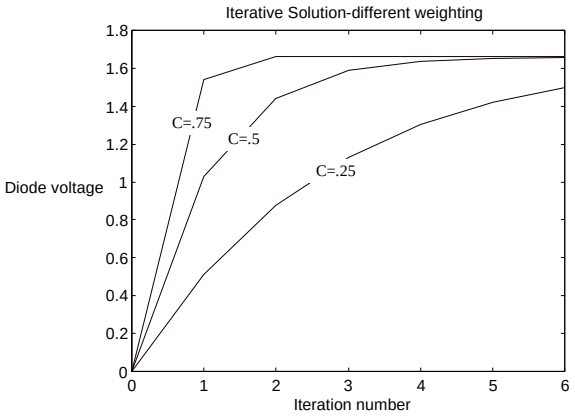
Table 2b: C=.5

$n \Rightarrow$	0	1	2	3	4	5	6
v_n^L	0	1.0277	1.4388	1.5876	1.6378	1.6541	1.6594
$i_n^L = i_n^{NL}$	0.0600	0.0394	0.0312	0.0282	0.0272	0.0269	0.0268
v_n^{NL}	2.0554	1.8500	1.7363	1.6879	1.6705	1.6646	1.6627
$v_n^{NL} - v_n^L$	2.0554	0.8223	0.2975	0.1003	0.0327	0.0105	0.0034

Table 2c: C=.75

$n \Rightarrow$	0	1	2	3	4	5	6
v_n^L	0	1.5416	1.6629	1.6618	1.6619	1.6619	1.6619
$i_n^L = i_n^{NL}$	0.0600	0.0292	0.0267	0.0268	0.0268	0.0268	0.0268
v_n^{NL}	2.0554	1.7034	1.6615	1.6619	1.6619	1.6619	1.6619
$v_n^{NL} - v_n^L$	2.0554	0.1618	-0.0015	0.0000	0.0000	0.0000	0.0000






```
function Out=DC_HB(iL0,N,C)
% DC_HB(I0,N,C) illustrates DC Harmonic Balance solution
% to series diode and Dc voltage
% N=number of iterations, C=convergence factor 0<C<1
%               iL        iNL
```

```
%    .------Rs-->---    -->-.
%    | +           +   +    | +
%   ---           vL   vNL  \/ Diode  iD=I0*(exp(k*vD)-1)
%    -  Vs                  --
%    |             -   -    |
%     --------------    ----'
%
%   iL0=initial linear current assumption
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Rs=50; Vs=3; k=2; I0=.001;
IVvar=zeros(4,N);
IVvar(1,1)=iL0;
for n=1:N
  iL=IVvar(1,n);
  vL=Vs-Rs*iL;
  vNL=vL;
  iNL=I0*(exp(k*vNL)-1);
  delta_i=iNL-iL;
  if n<N IVvar(1,n+1)=iL+C*delta_i;end
  Out=[Out,[iL;vL;iNL;delta_i]];
end
```

```
% Script file  RCD_HB.m
% to illustrate Harmonic Balance Soln to
% Resistor,Diode, Capacitor circuit
%
%          + Vd -    +Vc
%   ---Rs---->|--------------
%  |                 |       |
%  Vs                C       RL
%  |                 |       |
%   -------------------------

%units
 GHz=1e9; ns=1e-9;pF=1e-12;
 Rs=50; RL=50;Rmin=1e-6; Rmax=1e6;
 fs=1*GHz; ws=2*pi*fs;Vm=1;C=50*pF;
%
% Is=Vs/Rs    .-- C ---.  iL(f)      iNL(f)
%  .-----.----|        |--<--o     o-->---.
%  |     |     -- RL --'     +     +       |  +
%  /\    Rs                 vL(f) vNL(f) \/  vD
%  Is    |                                 --
%  |     |                   -     -       |
%  '-----'-------------------o     o-------'
%       Lin Sub-Ckt                Non-Lin Sub-Ckt
%
%      I1             I2
%    o->-- Zseries ---<-o
%  +    |                +    ( V1 )   ( Zshunt     Zshunt      ) ( I1 )
%  V1   Zshunt           V2    (    ) = (                        ) (    )
%  -    |                 -    ( V2 )   ( Zshunt  Zshunt+Zseries ) ( I2 )
%    o------------------o

%Find Y matrix for linear sub-circuit for series impedance

NH=5;
Tol=1e-4; %In absolute terms
```

```
Z0=[1  1; 1  2]*Rs;             %DC impedance matrix
Zarray=Z0 ;                     %Store as sub-array of Zarray
for n=1:NH
  Xc=-1/(n*ws*C);
  Zseries=RL*(j*Xc)/(RL+j*Xc);
  Zshunt=Rs;
  Z=[Zshunt Zshunt;Zshunt (Zshunt+Zseries)];
  Zarray(2*n+1:2*n+2,2*n+1:2*n+2)=Z;
end

%Source represented as a phasor Is=(Vm/Rs)*cos(ws*t);
Is=zeros(NH+1,1) ;

Is(2,1)=Vm/Rs;
IL=zeros(NH+1,1);
I=zeros(2*(NH+1),1);
even=2*[1:NH+1];
odd=even-1;
I(odd)=Is;

for mm=1:10

  I(even)=IL;
  V=Zarray*I;
  VL=V(even);
  Vs=V(odd);

  %Form voltage time waveform
  Ns=10;
  Npercycle=2*NH*Ns;
  Ncycle=5;
  deltat=1/(fs*Npercycle);
  t=[1:Ncycle*Npercycle]*deltat;
  w=ws*[0:NH]';
  nwt=w*t;
  Ntime=Ncycle*Npercycle;
  AL=VL*ones(1,Ntime);
  VLt=AL.*exp(j*nwt);
  vL=real(sum(VLt));
%plot(t/ns,vL); title('vL')

  Io=.001;k=2;
  iL=Io*(exp(k*vL)-1);


  FFTiL=fft(iL)*2/Ntime; FFTiL(1)=FFTiL(1)/2 ; %normalization different for DC term
  Freq=[0:length(FFTiL)-1]/max(t);
%  plot(Freq(1:50)/GHz,abs(FFTiL(1:50)))
  Nhar=Ncycle*[0:NH]+1;
  INL=FFTiL(Nhar); INL=INL(:);


  Error=INL+IL;
 % Terr=sum(abs(Error))
 % Terr<Tol
  IL=IL-.5*Error;

end
Terr=sum(abs(Error))
As=Vs*ones(1,Ntime);
Vst=As.*exp(j*nwt);
vs=real(sum(Vst));

plot(t/ns,vs-vL);title('vL')
```

Cap =1, 10, 50 pF Compare with Time-Domain RK solution