Project Tutorial for ELEN 100

Aleksandar I. Zečević
Dept. of Electrical Engineering
Santa Clara University

Project 1: Design of Passive Filters

In this project we will study some of the basic concepts involved in filter design. Although practical filters are almost always built using op amps (so-called *active filters*), the necessary calculations can be quite complicated. For that reason, in the following we will consider the *passive* low-pass filter shown in Fig. 1.

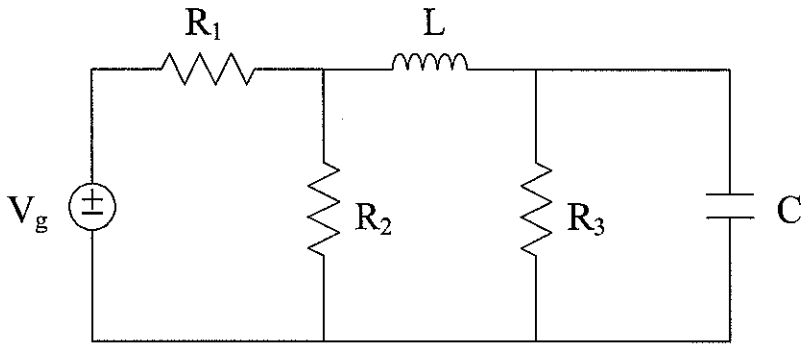

Fig. 1. A passive low-pass filter.

Our objective will be to choose the circuit elements in such a way that the transfer function has poles at $\omega_1 = 2,500\,rad/s$ and $\omega_2 = 9,000\,rad/s$. In other words, we will require the transfer function to have the form

$$H(j\omega) = K\,\frac{1}{(1+j\omega/a)(1+j\omega/b)} \tag{1}$$

where $a = 2,500$ and $b = 9,000$ (with no special demands on K).

The first step in the design process will be to obtain an expression for $H(j\omega)$ in terms of the circuit elements. This type of analysis can get quite messy, so we need to use all available shortcuts. In this particular case we can replace $\vec{V}_g$, R_1 and R_2 with a Thevenin equivalent, and then lump the components into two impedances. The resulting circuit is shown in Fig. 2

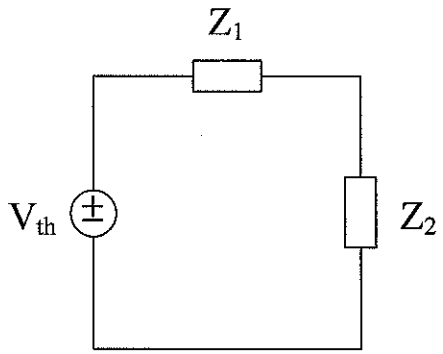

Fig. 2. The simplified circuit.

and its elements are given as

$$\begin{aligned} \vec{V}_{th} &= R_2\vec{V}_g/(R_1+R_2) \\ R_{th} &= R_1R_2/(R_1+R_2) \\ Z_1 &= R_{th}+j\omega L \\ Z_2 &= R_3/(1+j\omega CR_3) \end{aligned} \tag{2}$$

Since this circuit is basically a voltage divider, its output voltage is easily computed as

$$\vec{V}_0 = \frac{Z_2}{(Z_1+Z_2)}\,\vec{V}_{th} \tag{3}$$

Multiplying the numerator and denominator by $(1+j\omega CR_3)$ and recalling (2), we further obtain

$$H(j\omega) \equiv \frac{\vec{V}_0}{\vec{V}_g} = \frac{R_3R_2}{(R_1+R_2)} \cdot \frac{1}{(j\omega)^2LCR_3 + j\omega(L+CR_3R_{th}) + (R_3+R_{th})} \tag{4}$$

We should note at this point that the desired transfer function (1) can be expressed as

$$H(j\omega) = \frac{1}{(j\omega)^2/(ab) + j\omega(a+b)/ab + 1} \tag{5}$$

In order to equate this with (4), we need to factor out the term (R_3+R_{th}), which produces

$$H(j\omega) = K\,\frac{1}{(j\omega)^2LCR_3/(R_3+R_{th}) + j\omega(L+CR_3R_{th})/(R_3+R_{th}) + 1} \tag{6}$$

where

$$K = \frac{R_3R_2}{(R_1+R_2)(R_3+R_{th})} \tag{7}$$

Now it is simply a matter of matching the coefficients next to the corresponding powers of $j\omega$, which produces the following two equations:

$$\frac{1}{ab} = \frac{LCR_3}{(R_3+R_{th})} \tag{8}$$

$$\frac{a+b}{ab} = \frac{(L+CR_3R_{th})}{(R_3+R_{th})} \tag{9}$$

Since (8) and (9) represent two equations in four unknowns (L, C, R_3 and R_{th}), we can make life a bit easier by picking L and C in advance and solving for R_3 and R_{th} only. As an illustration, let us set $L = 1\,H$ and $C = 0.1\,\mu F$, since these are nice, round numbers which simplify the calculation. Substituting the values for a, b, L and C into (8) and (9), we now obtain

$$R_3 + R_{th} = 2.25R_3 \tag{10}$$

and

$$5.111\times10^{-4}(R_3+R_{th}) = 1 + 10^{-7}R_3R_{th} \tag{11}$$

From the first equation it follows that $R_{th} = 1.25R_3$. Substituting this into (11) we obtain a quadratic equation in R_3

$$1.25 \times 10^{-7} R_3^2 - 1.15 \times 10^{-3} R_3 + 1 = 0 \tag{12}$$

This equation has two real solutions, both of which are physically reasonable:

$$R_3 = 8,227\,\Omega \quad \Longrightarrow \quad R_{th} = 10,283\,\Omega \tag{13}$$

and

$$R_3 = 972\,\Omega \quad \Longrightarrow \quad R_{th} = 1,215\,\Omega \tag{14}$$

If we pick (13) and recall that

$$R_{th} = \frac{R_1 R_2}{R_1 + R_2} \tag{15}$$

we can set $R_1 = R_2 = 20,566\,\Omega$, which completes the design. The frequency response for the obtained circuit is shown in Fig. 3, and the corresponding Matlab code is provided in Appendix 1.

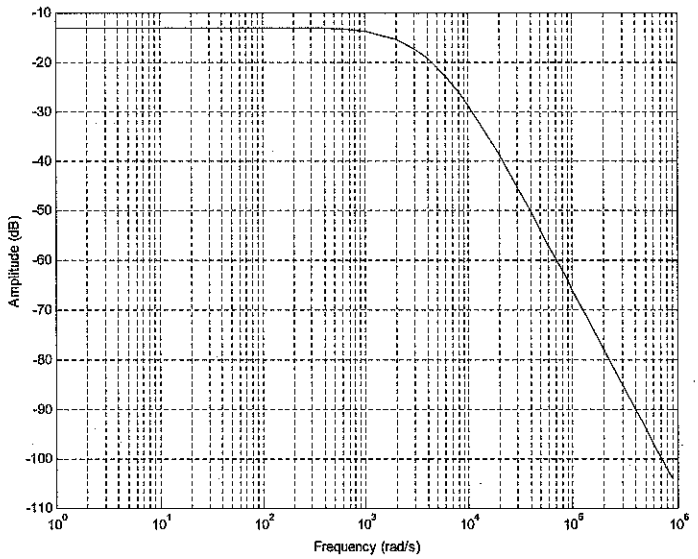

Fig. 3. Frequency response of the designed filter.

From a practical standpoint, we should note that the design we obtained can never be realized perfectly. For one thing, all components have tolerances, so we won't be able to get the exact values that were computed. In addition, some of the values we chose are not standard, so we may have to settle for the closest available ones. With that in mind, it is necessary to evaluate how variations in the element values affect the frequency characteristic.

To set up the problem for such an analysis in Matlab, let us first observe that the nodal equations for this circuit are

$$\begin{aligned}
&1)\ \ \vec{I}_g + \vec{I}_{R1} = 0 \qquad \Rightarrow \qquad \vec{V}_1 = \vec{V}_g \\
&2)\ \ -\vec{I}_{R1} + \vec{I}_{R2} + \vec{I}_L = 0 \\
&3)\ \ -\vec{I}_L + \vec{I}_{R3} + \vec{I}_C = 0
\end{aligned} \qquad \begin{aligned}
&\vec{I}_{R1} = (\vec{V}_1 - \vec{V}_2)/R_1 \\
&\vec{I}_{R2} = \vec{V}_2/R_2 \\
&\vec{I}_{R3} = \vec{V}_3/R_3 \\
&\vec{I}_L = (\vec{V}_2 - \vec{V}_3)/j\omega L \\
&\vec{I}_C = j\omega C \vec{V}_3 \\
&\vec{I}_g = ?
\end{aligned} \tag{16}$$

In matrix form, this becomes

$$\begin{bmatrix} 1 & 0 & 0 \\ -1/R_1 & (1/R_1 + 1/(j\omega L) + 1/R_2) & -1/(j\omega L) \\ 0 & -1/(j\omega L) & (1/R_3 + 1/(j\omega L) + j\omega C) \end{bmatrix} \begin{bmatrix} \vec{V}_1 \\ \vec{V}_2 \\ \vec{V}_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{17}$$

Since the frequency is a variable, (17) can be rewritten more conveniently as

$$(G_1 + j\omega G_2 + \frac{1}{j\omega} G_3)\vec{V} = b \tag{18}$$

where

$$G_1 = \begin{bmatrix} 1 & 0 & 0 \\ -1/R_1 & (1/R_1 + 1/R_2) & 0 \\ 0 & 0 & 1/R_3 \end{bmatrix}, \tag{19}$$

$$G_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & C \end{bmatrix} \tag{20}$$

and

$$G_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1/L & -1/L \\ 0 & -1/L & 1/L \end{bmatrix} \tag{21}$$

are *fixed* matrices. As usual, $\vec{V}$ represents the vector of unknown voltages, and

$$b = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. \tag{22}$$

Given this formulation, we can use the Matlab functions in Appendix 2 to simulate the frequency response (the program automatically generates sets of random element values). The results of such a simulation are shown in Fig. 4.

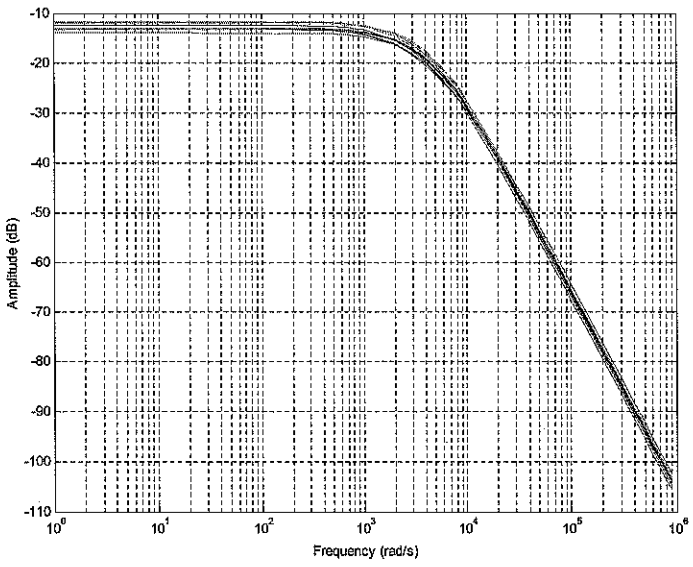

Fig. 4. The frequency response with random variations of element values.

Project 2: Transient Analysis and Simulation

The main objective of this project is to study how the transient response of a circuit can be simulated in Matlab and SPICE. As an illustration we will use the simple circuit in Fig. 5, in which $R_1 = R_2 = 2\,\Omega$, $R_3 = 1\,\Omega$, and $C = 0.5\,F$.

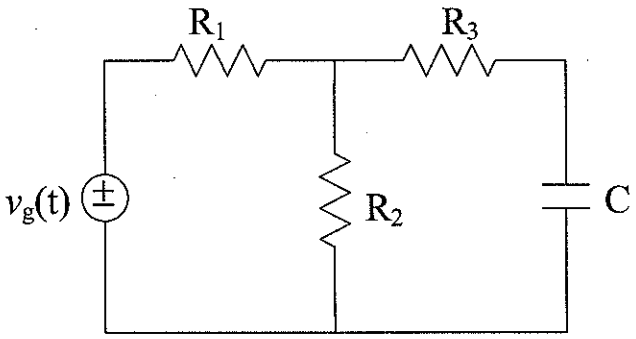

Fig. 5. A test circuit for transient analysis.

If we assume that $v_g(t)$ is a unit step function and that $v_C(0) = 0$, nodal analysis in the Laplace domain produces the following equations

$$\begin{aligned} &1)\quad -I_{R1}(s) + I_{R2}(s) + I_{R3}(s) = 0 \\ &2)\quad -I_{R3}(s) + I_C(s) = 0 \end{aligned} \qquad \begin{aligned} I_{R1}(s) &= (V_g(s) - V_1(s))/R_1 \\ I_{R2}(s) &= V_1(s)/R_2 \\ I_{R3}(s) &= (V_1(s) - V_2(s))/R_3 \\ I_C(s) &= sCV_2(s) \end{aligned} \tag{23}$$

After substituting the element values, (23) can be rewritten in matrix form as

$$\begin{bmatrix} 2 & -1 \\ -1 & (1+s/2) \end{bmatrix} \begin{bmatrix} V_1(s) \\ V_2(s) \end{bmatrix} = \begin{bmatrix} 1/2s \\ 0 \end{bmatrix} \tag{24}$$

Since this is a 2×2 matrix, we can invert it analytically, obtaining $V_1(s)$ and $V_2(s)$ as

$$V_1(s) = \frac{1+s/2}{2s(s+1)} = \frac{1/2}{s} - \frac{1/4}{s+1} \tag{25}$$

and

$$V_2(s) = \frac{1}{2s(s+1)} = \frac{1/2}{s} - \frac{1/2}{s+1} \tag{26}$$

The explicit expressions for $v_1(t)$ and $v_2(t)$ are now easily found to be

$$v_1(t) = \frac{1}{2} - \frac{1}{4}e^{-t} \tag{27}$$

and

$$v_2(t) = \frac{1}{2} - \frac{1}{2}e^{-t} \tag{28}$$

respectively.

It is important to recognize that this approach is not practical for large circuits. One of the main reasons for this is that some of the elements in the matrix are not numbers, but rather functions of s. This type of "symbolic" inversion is virtually impossible to perform for large matrices. In such cases the only alternative is *numerical simulation.*

As a preparatory step for such a simulation, it is necessary to describe the circuit as a combination of differential and algebraic equations (so-called DAEs). In our case, this implies rewriting the nodal equations as

$$\begin{aligned}
&1)\ \ i_g(t)+i_{R1}(t)=0 \qquad \Rightarrow \qquad v_1(t)=v_g(t)\\
&2)\ \ -i_{R1}(t)+i_{R2}(t)+i_{R3}(t)=0\\
&3)\ \ -i_{R3}(t)+i_C(t)=0
\end{aligned}
\qquad
\begin{aligned}
&i_{R1}(t)=(v_1(t)-v_2(t))/R_1\\
&i_{R2}(t)=v_2(t)/R_2\\
&i_{R3}(t)=(v_2(t)-v_3(t))/R_3\\
&i_C(t)=C\dot{v}_3(t)\\
&i_g(t)=\ ?
\end{aligned}
\qquad (29)$$

These equations can be expressed in matrix form in the following way

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -C \end{bmatrix} \begin{bmatrix} \dot{v}_1(t) \\ \dot{v}_2(t) \\ \dot{v}_3(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -1/R_1 & (1/R_1+1/R_2+1/R_3) & -1/R_3 \\ 0 & -1/R_3 & 1/R_3 \end{bmatrix} \begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \end{bmatrix} - \begin{bmatrix} v_g(t) \\ 0 \\ 0 \end{bmatrix} \qquad (30)$$

Note that the first two equations in (30) are purely algebraic (there are no derivatives in them), while the last one is differential.

In order to solve system (30), we first need to define matrix

$$M = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -C \end{bmatrix} \qquad (31)$$

and then enter the following three lines in Matlab:

$M = [0\ \ 0\ \ 0;\ 0\ \ 0\ \ 0;\ 0\ \ 0\ \ -0.5];$

options=odeset('mass', M);

$[t, x]$=ode23t(@transient2, [0 5], x_0, options);

The function transient2.m is provided in Appendix 3, and the corresponding plots of $v_2(t)$ and $v_3(t)$ are shown in Fig. 6 (these are voltages accross R_2 and C, respectively).

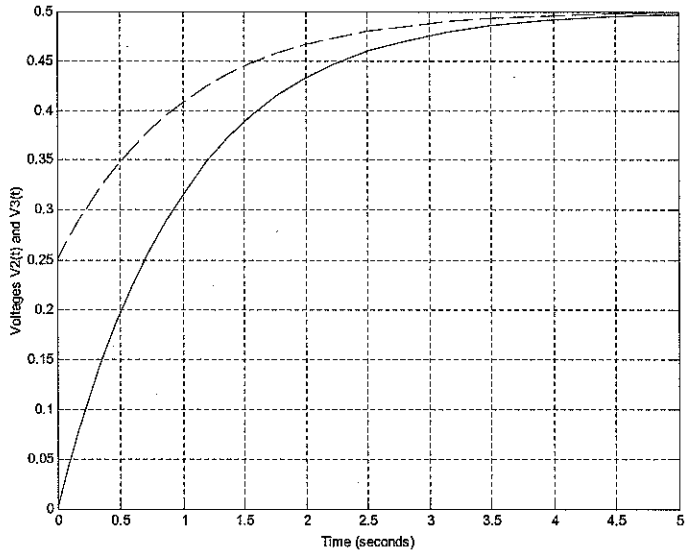

Fig. 6. The transient responses for the circuit in Fig. 5.

It is easily verified that these plots match the analytic expressions obtained in (27) and (28). We should also point out that SPICE uses a very similar approach in executing transient analysis.

NOTE: Since the step function is *discontinuous*, it cannot be handled numerically, and we must include a finite (although very small) rise time. Keep in mind also that the PULSE function in SPICE is *inherently periodic*. If you want to simulate a unit step, you must set the period to be *larger* than the overall simulation time.

Appendix 1

```
function F=freqresp4(G,C,L,b,w)

mag=zeros(3,1);
% We need to initialize vector mag, in order not
% to confuse Matlab.

for k=1:length(w)
   omega=w(k);
   % Omega is the next frequency in vector w.

   A=G+i*omega*C+(1/(i*omega))*L;
   x=A\b;
   % Vector x contains the solution of the node voltage
   % equations in complex form.

   mag=[mag abs(x)];
   % The function abs(x) computes the magnitudes of all voltages.
   % These magnitudes are stored as a new column in matrix mag.
end

% Each column of matrix mag  now contains voltage magnitudes computed
% at a particular frequency. Note, however, that the first column of
% this matrix is essentially redundant. It is a vector of zeros that
% serves only as a place holder.

V3=mag(3,2:length(w)+1);
% Since we are interested only in V3, we will extract row 3 of matrix
% mag (and ignore the first column).

F=20*log10(V3);
```

Appendix 2

```
function F=freqresp5(QX,b,w,num)
% QX is a vector that contains all the nominal element values.
% Resistors are listed first (as Q(1),Q(2) and Q(3)), then
% capacitors (Q(4)), and finally inductors (Q(5)). num is the
% number of different cases that we wish to simulate.

for s=1:num
   Q=variation(QX);
% This function produces random variations of the element
% values, which are within 20% of the nominal ones.

   G=zeros(3,3); C=zeros(3,3); L=zeros(3,3);
   % This line saves us some time, since we now need to enter
   % only the nonzero elements in G, C and L.
   G(1,1)=1;G(2,1)=-1/Q(1);G(2,2)=1/Q(1)+1/Q(2);G(3,3)=1/Q(3);
   C(3,3)=Q(4);
   L(2,2)=1/Q(5);L(2,3)=-1/Q(5);L(3,2)=-1/Q(5);L(3,3)=1/Q(5);
% For each new set of element values, matrices G, C and L must
% be recomputed.

   mag=zeros(3,1);
% Vector mag is initialized.

   for k=1:length(w)
      omega=w(k);
   % Omega is the next frequency in w.

      A=G+i*omega*C+(1/(i*omega))*L;
      x=A\b;
   % Vector x contains the solution of the node voltage
   % equations in complex form.

      mag=[mag abs(x)];
   % The magnitudes of all voltages as stored as a new column in
   % matrix mag.
   end

   V3=mag(3,2:length(w)+1);
% Since we are interested only in V3, we will extract row 3 of matrix
% mag (and ignore the first column).

   FX=20*log10(V3);
   F=[F;FX];
```

```
% Each row of vector F corresponds to a different set of element values.
% You can plot the all at once, using the command semilogx(w,F).
end



function Q=variation(R)
% This function produces random variations of the elements
% of vector R, which are within 20% of the nominal ones.

m=length(R);
g=ones(1,m)+0.5*rand(1,m);
% Function rand(1,m) generates a random vector of size 1xm,
% whose elements are between 0 and 1. The elements of vector
% g are therefore random numbers with values between 1 and 1.5.

Q=0.8*(R.*g);
% Each element of R is multiplied by a random number between 1 and 1.5.
% The additional multiplication by 0.8 secures that the elements of Q
% satisfy: 0.8*R(i)<Q(i)<1.2*R(i). In other words, the elements of Q are
% within 20% of the values in R.
```

Appendix 3

```
function F=transient2(t,x)
% This function provides the right-hand side of the
% differential equation for the Matlab solver.

F=[0;0;0];
% Vector F is initialized.

R1=2; R2=2; R3=1;
% These are the resistor values for our circuit. If you want
% to perform a simulation with different R1, R2 and R3, all you
% need to do is change this line.

if (t>=0)&(t<=1e-6)
   h=1e6*t;
else
   h=1;
end
% h(t) represents an approximation of the step function,
% with a rise time of 1 microsecond.

F(1)=x(1)-h;
F(2)=-(1/R1)*x(1)+(1/R1+1/R2+1/R3)*x(2)-(1/R3)*x(3);
F(3)=-(1/R3)*x(2)+(1/R3)*x(3);
% This is the right-hand side of the DAE. It is written in terms of R1, R2
% and R3, so that you don't need to rewrite the code every time your
% element values change.
```