3

Synthesis of analogue circuits

We have studied how (network) functions, systems of differential equations, matrices and other mathematical abstractions can represent physical reality. We will now look at this from the other side – can we construct physical objects that correspond to a given abstraction?

We will first look at the relatively simple problem of constructing passive circuits that correspond to a given network function. By this time, you are of course aware that the function itself should have certain characteristics if it is to correspond to a realisable network. In the special but all-important case of the design of filters to meet specific requirements, we have to go one step backwards to obtain a system function that ensures such compliance, before we can obtain a realisation. We will see how the so-called "modern filter design" techniques accomplish this task through mathematical approximations of the desired performance characteristics. These techniques, even though traditionally called "modern", have been about for more than a century!

We will then look at a few standard methods for the realisation of functional blocks such as filters using active circuits.

The treatment up to this has been limited to linear circuits. We have seen that the state space approach allows us to model non-linear systems as well as linear systems. As far as circuits are concerned, we still need mechanisms for the modelling of non-linear elements, even the simplest of them; the diode. We will now step two steps back and investigate some elementary methods for the modelling of non-linear elements.

We also need at least some idea about the sensitivity of network characteristics to changes in component values. Tellegan's Theorem is an unexpected result that allows us to investigate the sensitivity of circuit performance to parameter changes quite economically. We will look at how this helps us in the design of robust systems.

3.1   Passive circuits

We already know that a network can be represented by a network function. We have also studied how this can be done. We are now going to explore how a network can be synthesised, given its network function. We will first look at one port networks, defined by either impedance or admittance functions and later go on to investigate two port networks.

Let us recall some of the common properties of passive RLC functions.

They are real, rational functions.
They do not have poles (or zeros) on the right-half s-plane, nor do they have multiple order poles on the $j\omega$ axis.
The resulting matrices are symmetric.

Let us now look at each of these types of networks. The LC network functions have these additional properties:

They are simple, that is there are no higher order poles or zeros.
All poles and zeros lie on the $j\omega$ axis.
Poles and zeros alternate.
The origin and infinity are always critical frequencies, that is, there will be either a pole or a zero at both the origin and at infinity.
The multiplicative constant is positive.

The properties of RC network functions are:

The poles and zeros of an RC driving point function lie on the non-positive real axis.
They are simple.
Poles and zeros alternate.
The slopes of impedance functions are negative, those of admittance functions are positive.

Foster and Cauer first proposed the realisation of these networks in various canonical forms, in the 1920s. We will now examine each of these forms. There are four canonical forms, relating to the realisation of LC, RC and RL networks as follows:

	1st Foster form	2nd Foster form	1st Cauer form	2nd Cauer form
Partial fraction expansion of impedance function	✔			
Partial fraction expansion of admittance function		✔		
Continued fraction expansion about the point at infinity			✔	
Continued fraction expansion about the origin				✔

Considering the pole-zero properties of LC network functions, we can assume that the impedance function will have a zero or pole at the origin and pairs of complex conjugate poles and zeros. The admittance function will have a pole or zero at the origin. These functions may be expanded as partial fractions to yield realisations of Foster-form networks. They may also be subjected to continued fraction expansion to yield Cauer-form networks.

We will consider realisations of each of these forms.

3.1.1   Realisation of LC driving point functions

First Foster form – Impedance functions of LC networks

$$Z(s) = \frac{H(s^2 + \omega_1^{\ 2})(s^2 + \omega_3^{\ 2}). \ . \ .(s^2 + \omega_m^{\ 2})}{(s^2 + \omega_2^{\ 2})(s^2 + \omega_4^{\ 2}). \ . \ . \ \ (s^2 + \omega_r^{\ 2})}$$
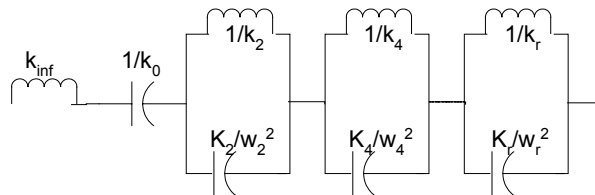
where $0 \leq \omega_1^{\ 2} \leq \omega_2^{\ 2} \leq \omega_3^{\ 2} \leq. \ . \ ..$
and $m = r \pm 1$
By partial fraction expansion we can write:

$$Z(s) = k_\infty s + \frac{k_0}{s} + \sum_{p=2,4,...}^{r} \frac{k_p s}{s^2 + \omega_p^{\ 2}}, \quad k_\infty, k_0 \geq 0, \ k_p > 0$$

This corresponds to the following circuit realisation:

Let us now consider a few examples.

Example 1

We will consider the following simple driving point impedance function, where ω₁ > 0 and m = r – 1:
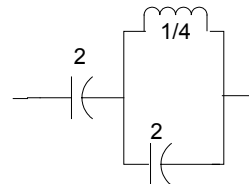
$$Z(s) = \frac{s^2 + 1}{s(s^2 + 2)}$$

By partial fraction expansion we get: This corresponds to the circuit shown:

$$Z(s) = \frac{A}{s} + \frac{Bs}{s^2 + 2} = \frac{A(s^2 + 2) + Bs^2}{s(s^2 + 2)}$$
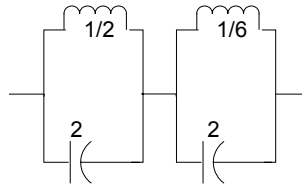
$$\therefore A + B = 1, \quad 2A = 1$$
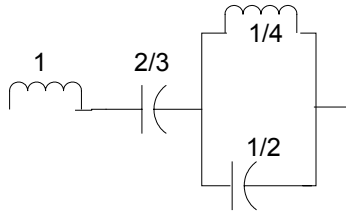
$$A = \frac{1}{2}, B = \frac{1}{2}$$

Example 2 $\omega_1 = 0$ and m = r – 1:

$$Z(s) = \frac{(s^2 + 0)(s^2 + 2)}{s(s^2 + 1)(s^2 + 3)} = \frac{s(s^2 + 2)}{(s^2 + 1)(s^2 + 3)}$$

$$= \frac{As}{s^2 + 1} + \frac{Bs}{s^2 + 3} = \frac{s^3(A + B) + s(3A + B)}{(s^2 + 1)(s^2 + 3)}$$

$$\therefore A + B = 1, \quad 3A + B = 2$$

$$\therefore A = \tfrac{1}{2}, \ B = \tfrac{1}{2}$$

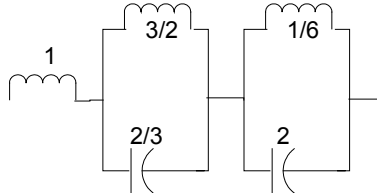$$\therefore Z(s) = \frac{s}{2(s^2 + 1)} + \frac{s}{2(s^2 + 3)}$$

Example 3 $\omega_1 > 0$ and m = r + 1:

$$Z(s) = \frac{(s^2 + 1)(s^2 + 3)}{s(s^2 + 2)} = s + \frac{3}{2s} + \frac{s}{2(s^2 + 2)}$$

Example 4

$$Z(s) = \frac{(s^2 + 0)(s^2 + 2)(s^2 + 4)}{s(s^2 + 1)(s^2 + 3)}$$

$$= \frac{s(s^2 + 2)(s^2 + 4)}{(s^2 + 1)(s^2 + 3)} = s + \frac{3s}{2(s^2 + 1)} + \frac{s}{2(s^2 + 3)}$$

Second Foster form – Admittance functions of LC networks
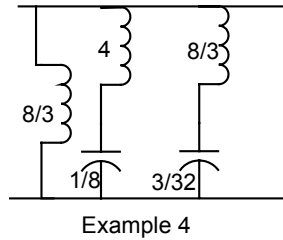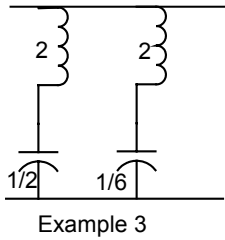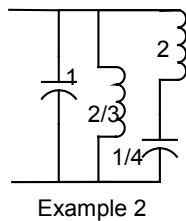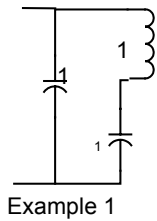
We will consider the same examples as before:

1. $Z(s) = \frac{s^2 + 1}{s(s^2 + 2)} \Rightarrow Y(s) = \frac{s(s^2 + 2)}{s^2 + 1} = s + \frac{s}{s^2 + 1}$

2. $Z(s) = \frac{(s^2 + 0)(s^2 + 2)}{s(s^2 + 1)(s^2 + 3)} \Rightarrow Y(s) = \frac{s(s^2 + 1)(s^2 + 3)}{(s^2 + 0)(s^2 + 2)} = s + \frac{3}{2s} + \frac{s}{2(s^2 + 2)}$

3. $Z(s) = \frac{(s^2 + 1)(s^2 + 3)}{s(s^2 + 2)} \Rightarrow Y(s) = \frac{s(s^2 + 2)}{(s^2 + 1)(s^2 + 3)} = \frac{s}{2(s^2 + 1)} + \frac{s}{2(s^2 + 3)}$
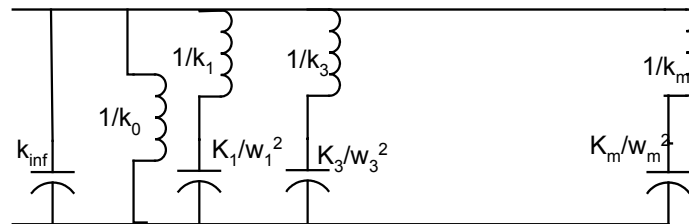
4.

$$Z(s) = \frac{(s^2 + 0)(s^2 + 2)(s^2 + 4)}{s(s^2 + 1)(s^2 + 3)} \Rightarrow$$

$$Y(s) = \frac{s(s^2 + 1)(s^2 + 3)}{(s^2 + 0)(s^2 + 2)(s^2 + 4)} = \frac{3}{8s} + \frac{s}{4(s^2 + 2)} + \frac{3s}{8(s^2 + 4)}$$

The realisations of the four functions are as shown below:

Example 1   Example 2

Example 3   Example 4

We can obtain a general realisation for the admittance function Y(s) as follows:
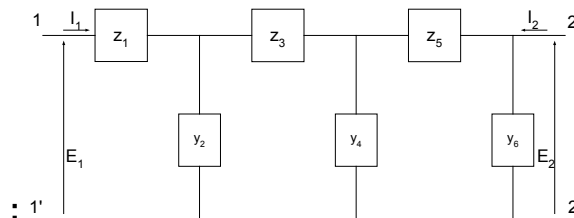
$$Y(s) = k_\infty s + \frac{k_0}{s} + \sum_{p=1,3,\ldots}^{m} \frac{k_p s}{s^2 + \omega_p^2}, \qquad k_\infty, k_0 \geq 0, \; k_p > 0$$

Realisation of LC network functions using First Cauer form

The first Cauer form is generated by the continued fraction expansion of a reactance function about the point at infinity.

Consider a ladder network as shown [Note that the series arms are impedances while the shunt arms are admittances]

To compute the driving point impedance $Z = \dfrac{E_1}{I_1}$, we start at the other end:

The impedance of the last term $z_6 = \dfrac{1}{y_6}$

The impedance of the last two arms is $z_5 + \dfrac{1}{y_6}$

The impedance of the last three arms is $\dfrac{1}{y_4 + \dfrac{1}{z_5 + \dfrac{1}{y_6}}}$

Proceeding in this manner, we can write

$$Z = z_1 + \cfrac{1}{y_2 + \cfrac{1}{z_3 + \cfrac{1}{y_4 + \cfrac{1}{z_5 + \cfrac{1}{y_6}}}}}$$

The driving point admittance may also be written in a similar manner, where the first term is an admittance, the next an impedance, etc. We will now consider each of the four examples used to illustrate the Foster forms.

Example 1 $\qquad Z(s) = \dfrac{s^2 + 1}{s(s^2 + 2)}, \qquad Y(s) = \dfrac{s(s^2 + 2)}{s^2 + 1}$

Starting with the impedance function Z(s) that has a zero at infinity, we proceed as follows:

$$Z(s) = \cfrac{1}{\dfrac{s^3 + 2s}{s^2 + 1}} = \cfrac{1}{s + \dfrac{s}{s^2 + 1}} = \cfrac{1}{s + \cfrac{1}{\dfrac{s^2 + 1}{s}}} = \cfrac{1}{s + \cfrac{1}{s + \dfrac{1}{s}}}$$

Either of these will give us the following circuit realisation:
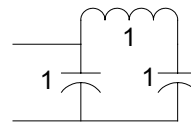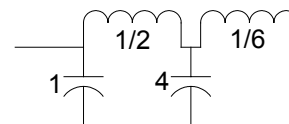
If we use the admittance function, we get:

$$Y(s) = \cfrac{s^3 + 2s}{s^2 + 1} = s + \cfrac{s}{s^2 + 1} = s + \cfrac{1}{\dfrac{s^2 + 1}{s}} = s + \cfrac{1}{s + \dfrac{1}{s}}$$

Example 2 $\qquad Z(s) = \dfrac{s(s^2 + 2)}{(s^2 + 1)(s^2 + 3)}, \qquad Y(s) = \dfrac{(s^2 + 1)(s^2 + 3)}{s(s^2 + 2)}$

$$Y(s) = \cfrac{s^4 + 4s^2 + 3}{s^3 + 2s} = s + \cfrac{2s^2 + 3}{s^3 + 2s} = s + \cfrac{1}{\dfrac{s^3 + 2s}{2s^2 + 3}} = s + \cfrac{1}{s/2 + \dfrac{s/2}{2s^2 + 3}}$$

This leads to the realisation:

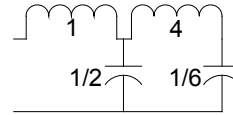$$= s + \cfrac{1}{s/2 + \cfrac{1}{\dfrac{2s^2 + 3}{s/2}}} = s + \cfrac{1}{s/2 + \cfrac{1}{4s + \dfrac{3}{s/2}}}$$

Example 3 $Z(s) = \dfrac{(s^2+1)(s^2+3)}{s(s^2+2)}$, $Y(s) = \dfrac{s(s^2+2)}{(s^2+1)(s^2+3)}$

Note that this is the reciprocal of the function considered in example 2.

$$Z(s) = \frac{s^4+4s^2+3}{s^3+2s} = s + \cfrac{1}{s/2 + \cfrac{1}{4s + \cfrac{3}{s/2}}}$$
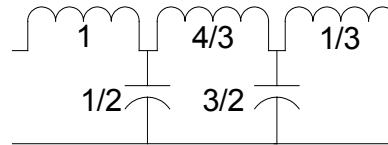
This leads to the realisation:

Example 4 $Z(s) = \dfrac{s(s^2+2)(s^2+4)}{(s^2+1)(s^2+3)}$, $Y(s) = \dfrac{(s^2+1)(s^2+3)}{s(s^2+2)(s^2+4)}$

$$Z(s) = \frac{s^5+6s^3+8s}{s^4+4s^2+3} = s + \cfrac{1}{\dfrac{s^4+4s^2+3}{2s^3+5s}} = s + \cfrac{1}{s/2 + \cfrac{1}{\dfrac{2s^3+5s}{3s^2/2+3}}}$$

It can be realised as:

$$= s + \cfrac{1}{s/2 + \cfrac{1}{4s/3 + \cfrac{1}{\dfrac{3s^2/2+3}{s}}}} = s + \cfrac{1}{s/2 + \cfrac{1}{4s/3 + \cfrac{1}{3s/2 + \cfrac{1}{s/3}}}}$$


Realisation of LC network functions using Second Cauer form

The second Cauer form is generated by the continued fraction expansion of a reactance function about the origin. This too results in a realisation where the series arms are impedances while the shunt arms are admittances. We will consider the same examples as before. However, in this realisation the series arms contain capacitors while the shunt arms have inductors.

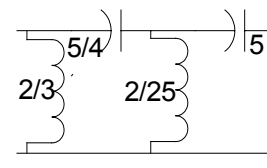Example 1 $Z(s) = \dfrac{s^2+1}{s(s^2+2)}$, $Y(s) = \dfrac{s(s^2+2)}{s^2+1}$

$$Z(s) = \frac{1+s^2}{2s+s^3} = \frac{1}{2s} + \cfrac{1}{\dfrac{4}{s}+2s} = \frac{1}{2s} + \cfrac{1}{\dfrac{4}{s} + \cfrac{1}{\dfrac{1}{2s}}}$$
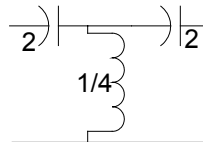
Example 2 $Z(s) = \dfrac{s(s^2+2)}{(s^2+1)(s^2+3)}$, $Y(s) = \dfrac{(s^2+1)(s^2+3)}{s(s^2+2)}$

$$Y(s) = \frac{3+4s^2+s^4}{2s+s^3} = \frac{3}{2s} + \frac{(5/2)s^2+s^4}{2s+s^3} = \frac{3}{2s} + \cfrac{1}{\dfrac{2s+s^3}{(5/2)s^2+s^4}} = \frac{3}{2s} + \cfrac{1}{\dfrac{4/5}{s} + \dfrac{(1/5)s}{(5/2)+s}}$$

$$= \frac{3}{2s} + \cfrac{1}{\dfrac{4/5}{s} + \cfrac{1}{\dfrac{(5/2)+s}{(1/5)s}}} = \frac{3}{2s} + \cfrac{1}{\dfrac{4/5}{s} + \cfrac{1}{\dfrac{25/2}{s}+5s}} = \frac{3}{2s} + \cfrac{1}{\dfrac{4/5}{s} + \cfrac{1}{\dfrac{25/2}{s} + \dfrac{1}{1/5}{s}}}$$
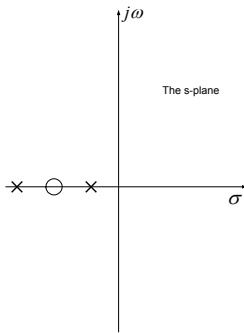

Please work out the other two examples.

3.1.2 RC Driving point functions

We will start with the properties of RC driving point functions that we are already familiar with. We have noted that:

- The poles and zeros are simple.
- They all lie on the non-positive σ- axis.
- Poles and zeros alternate.
- The critical frequency of the smallest magnitude is a pole.

A typical pole-zero plot on the s-plane is as shown. With these properties, the form of the impedance function may be represented by:

$$Z(s) = \frac{H(s - s_2)(s - s_4)\ldots(s - s_m)}{(s - s_1)(s - s_3)\ldots(s - s_n)}$$

As the smallest critical frequency corresponds to a pole and as poles and zeros alternate,

$$0 \le s_1 < s_2 < s_3 \ldots$$

The following table gives the possible forms of the expansion of such a function.

First Foster form: Expansion of Z(s)	$Z(s) = k_0 + \dfrac{k_1}{s - s_1} + \dfrac{k_3}{s - s_3} \ldots + \dfrac{k_n}{s - s_n}$
Second Foster form: Expansion of Y(s)	$Y(s) = k_\infty s + k_0 + \dfrac{k_2 s}{s - s_2} + \dfrac{k_4 s}{s - s_4} + \ldots + \dfrac{k_m s}{s - s_m}$
First Cauer form: Continued fraction expansion about infinity	$Z(s) = a_1 + \cfrac{1}{b_2 s + \cfrac{1}{a_3 + \cfrac{1}{b_4 s + \cfrac{1}{a_5 + \ldots}}}}$
Second Cauer form: Continued fraction expansion about the origin.	$Z(s) = \dfrac{1}{a_1 s} + \cfrac{1}{b_2 + \cfrac{1}{a_3 s + \cfrac{1}{b_4 + \cfrac{1}{a_5 s + \ldots}}}}$

There are four possible combinations that we need to study. We will consider one example from each of these:

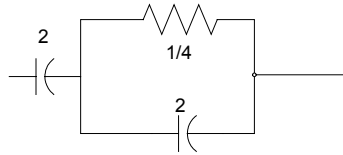|	m = n –1 (Order of numerator is one less than that of denominator)		m = n (Both numerator and denominator are of the same order)	
$s_1 = 0$	Example 1	$Z(s) = \dfrac{(s+1)}{s(s+2)}$	Example 2	$Z(s) = \dfrac{(s+1)(s+3)}{s(s+2)}$
$s_1 > 0$	Example 3	$Z(s) = \dfrac{(s+2)}{(s+1)(s+3)}$	Example 4	$Z(s) = \dfrac{(s+2)(s+4)}{(s+1)(s+3)}$

First Foster form:

Example 1

$$Z(s) = \frac{(s+1)}{s(s+2)} = \frac{A}{s} + \frac{B}{s+2} = \frac{A(s+2) + Bs}{s(s+2)}$$

$$\therefore A + B = 1, \quad 2A = 1$$

$$\therefore A = 1/2, \ B = 1/2$$

$$Z(s) = \frac{1/2}{s} + \frac{1/2}{(s+2)}$$

Example 2

$$Z(s) \quad = \frac{(s+1)(s+3)}{s(s+2)} = \frac{s^2 + 4s + 3}{s^2 + 2s}$$

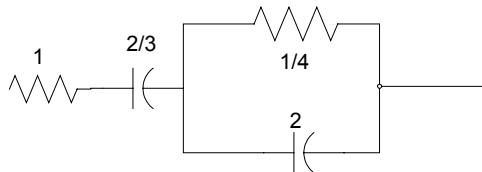$$= 1 + \frac{2s+3}{s(s+2)} = 1 + \frac{A}{s} + \frac{B}{s+2}$$

$$= 1 + \frac{A(s+2) + Bs}{s(s+2)}$$

$$A + B = 2$$
$$2A = 3$$
$$A = 3/2, \ B = 1/2$$

$$Z(s) = 1 + \frac{3/2}{s} + \frac{1/2}{s+2}$$

Example 3

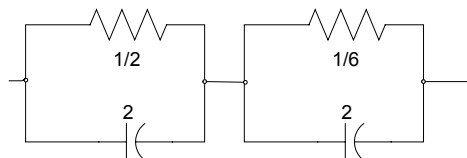$$Z(s) \quad = \frac{(s+2)}{(s+1)(s+3)} = \frac{A}{s+1} + \frac{B}{s+3}$$

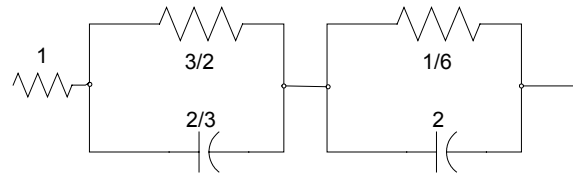$$\frac{A(s+3) + B(s+1)}{(s+1)(s+3)}$$

$$A + B = 1$$
$$3A + B = 2$$
$$A = 1/2, \ B = 1/2$$

$$Z(s) = \frac{1/2}{s+1} + \frac{1/2}{s+3}$$

Example 4

$$Z(s) \quad = \frac{(s+2)(s+4)}{(s+1)(s+3)} = \frac{s^2+6s+8}{s^2+4s+3}$$

$$= 1 + \frac{2s+5}{(s+1)(s+3)} = 1 + \frac{A}{s+1} + \frac{B}{s+3}$$

$$= 1 + \frac{A(s+3)+B(s+1)}{(s+1)(s+3)}$$

$A + B = 2$

$3A + B = 5$

$A = 3/2,\ B = 1/2$

$$Z(s) = 1 + \frac{3/2}{(s+1)} + \frac{1/2}{(s+3)}$$

Second Foster form:

Example 1

$$Z(s) = \frac{(s+1)}{s(s+2)}$$

$$Y(s) = \frac{s(s+2)}{(s+1)} = \frac{s^2+2s}{s+1} = s + \frac{s}{s+1}$$
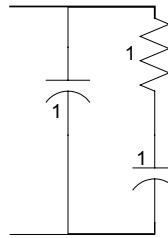
Example 2

$$Z(s) = \frac{(s+1)(s+3)}{s(s+2)}$$

$$Y(s) \quad = \frac{s(s+2)}{(S+1)(S+3)} = \frac{As}{s+1} + \frac{Bs}{s+3}$$

$$= \frac{As(s+3)+Bs(s+1)}{(s+1)(s+3)}$$

$A + B = 1$

$3A + B = 2$

$A = 1/2.\ B = 1/2$
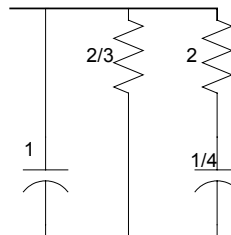
$$Y(s) = \frac{s/2}{s+1} + \frac{s/2}{s+3}$$
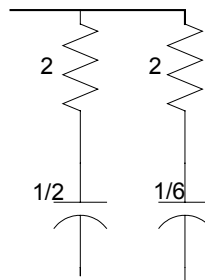
Example 3

$$Z(s) = \frac{(s+2)}{(s+1)(s+3)}$$

$$Y(s) \quad = \frac{(s+1)(s+3)}{(s+2)} = \frac{s^2+4s+3}{s+2}$$

$$= s + \frac{2s+3}{s+2} = s + \frac{3}{2} + \frac{s/2}{s+2}$$

Example 4

$$Z(s) = \frac{(s+2)(s+4)}{(s+1)(s+3)}$$

$$Y(s) = \frac{(s+1)(s+3)}{(s+2)(s+4)} = \frac{3}{8} + \frac{7s/4 + 5s^2/8}{(s+2)(s+4)}$$
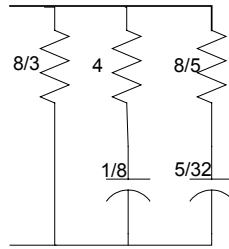
$$= \frac{3}{8} + \frac{As}{s+2} + \frac{Bs}{s+4} = \frac{3}{8} + \frac{As(s+4) + Bs(s+2)}{(s+2)(s+4)}$$

$$A + B = 5/8$$

$$4A + 2B = 7/4 \Rightarrow 2A + B = 7/8$$

$$A = 1/4, \; B = 3/8$$

$$Y(s) = \frac{3}{8} + \frac{s/4}{s+2} + \frac{3s/8}{s+4}$$

First Cauer form:

Example 1

$$Z(s) = \frac{(s+1)}{s(s+2)} = \frac{1}{\frac{s(s+2)}{s+1}} = \frac{1}{s + \frac{s}{s+1}}$$

$$= \frac{1}{s + \frac{1}{\frac{s+1}{s}}} = \frac{1}{s + \frac{1}{1 + \frac{1}{s}}}$$

Example 2

$$Z(s) = \frac{(s+1)(s+3)}{s(s+2)} = \frac{s^2 + 4s + 3}{s^2 + 2s}$$

$$= 1 + \frac{2s+3}{s^2+2s} = 1 + \frac{1}{\frac{s^2+2s}{2s+3}} = 1 + \frac{1}{s/2 + \frac{s/2}{2s+3}}$$

$$= 1 + \frac{1}{s/2 + \frac{1}{\frac{2s+3}{s/2}}} = 1 + \frac{1}{s/2 + \frac{1}{4 + \frac{3}{s/2}}}$$

$$= 1 + \frac{1}{s/2 + \frac{1}{4 + \frac{1}{s/6}}}$$
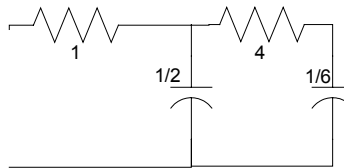
Example 3

$$Z(s) = \frac{(s+2)}{(s+1)(s+3)} = \frac{1}{\frac{s^2+4s+3}{s+2}} = \frac{1}{s + \frac{2s+3}{s+2}}$$

$$= \frac{1}{s + \frac{1}{\frac{s+2}{2s+3}}} = \frac{1}{s + \frac{1}{1/2 + \frac{1/2}{2s+3}}} = \frac{1}{s + \frac{1}{1/2 + \frac{1}{4s+6}}}$$

$$= \frac{1}{s + \frac{1}{1/2 + \frac{1}{4s + \frac{1}{1/6}}}}$$

Example 4

$$Z(s) = \frac{(s+2)(s+4)}{(s+1)(s+3)} = \frac{s^2+6s+8}{s^2+4s+3} = 1 + \frac{2s+5}{s^2+4s+3}$$

$$= 1 + \cfrac{1}{\cfrac{s^2+4s+3}{2s+5}} = 1 + \cfrac{1}{s/2 + \cfrac{3s/2+3}{2s+5}}$$

$$= 1 + \cfrac{1}{s/2 + \cfrac{1}{\cfrac{2s+5}{3s/2+3}}} 1 + \cfrac{1}{s/2 + \cfrac{1}{4/3 + \cfrac{1}{3s/2+3}}}$$

$$= 1 + \cfrac{1}{s/2 + \cfrac{1}{4/3 + \cfrac{1}{3s/2 + \cfrac{1}{1/3}}}}$$
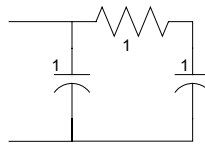

Second Cauer form:

Example 1

$$Z(s) = \frac{(s+1)}{s(s+2)} = \frac{1+s}{2s+s^2} = \frac{1}{2s} + \frac{s/2}{2s+s^2}$$

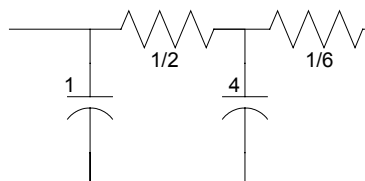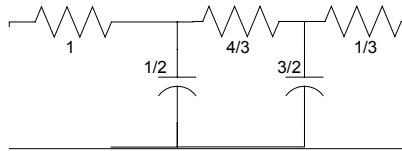$$= \frac{1}{2s} + \cfrac{1}{\cfrac{2s+s^2}{s/2}} = \frac{1}{2s} + \cfrac{1}{4 + \cfrac{s^2}{s/2}}$$

$$= \frac{1}{2s} + \cfrac{1}{4 + \cfrac{1}{\cfrac{1}{2s}}}$$
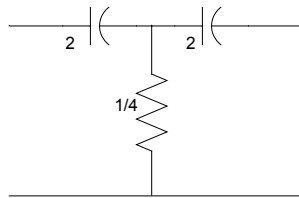
Example 2

$$Z(s) = \frac{(s+1)(s+3)}{s(s+2)} = \frac{3+4s+s^2}{2s+s^2} = \frac{1}{2s/3} + \frac{5s/2+s^2}{2s+s^2}$$

$$= \frac{1}{2s/3} + \cfrac{1}{\cfrac{2s+s^2}{5s/2+s^2}} = \frac{1}{2s/3} + \cfrac{1}{4/5 + \cfrac{s^2/5}{5s/2+s^2}}$$

$$= \frac{1}{2s/3} + \cfrac{1}{4/5 + \cfrac{1}{\cfrac{5s/2+s^2}{s^2/5}}} = \frac{1}{2s/3} + \cfrac{1}{4/5 + \cfrac{1}{25/2s + \cfrac{1}{1/5}}}$$
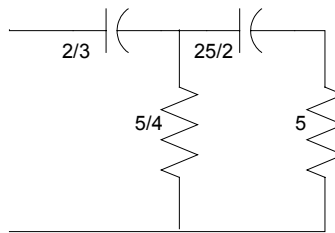
3.1.3 RL Driving point functions

The easiest way to start is to make use of the duality of RC and RL network functions. We used the following general form of the impedance of an RC function:

$$Z(s) = \frac{H(s-s_2)(s-s_4)\ldots(s-s_m)}{(s-s_1)(s-s_3)\ldots(s-s_n)}$$

We now consider a function of the same form as the admittance function of an RL network:

$$Y(s) = \frac{H(s-s_2)(s-s_4)\ldots(s-s_m)}{(s-s_1)(s-s_3)\ldots(s-s_n)}$$

As the smallest critical frequency corresponds to a pole and as poles and zeros alternate,

$$0 \le s_1 < s_2 < s_3 \ldots$$

There are four possible combinations that we need to study. An example of each is given below:

	m = n −1 (Order of numerator is one less than that of denominator)		m = n (Both numerator and denominator are of the same order)	
$s_1 = 0$	Example 1	$Y(s) = \dfrac{(s+1)}{s(s+2)}$	Example 2	$Y(s) = \dfrac{(s+1)(s+3)}{s(s+2)}$
$s_1 > 0$	Example 3	$Y(s) = \dfrac{(s+2)}{(s+1)(s+3)}$	Example 4	$Y(s) = \dfrac{(s+2)(s+4)}{(s+1)(s+3)}$

Their realisations follow the same pattern as in the case of the RC network functions.

3.1.4 Synthesis of RLC circuits

Uot of the many procedures available in classical design for the synthesis of RLC circuits, we will consider the Brune Synthesis, which starts off with what is called the Foster Preamble.

Foster Preamble
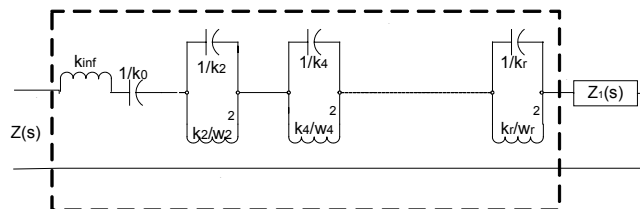
Step 1: Reactance reduction

We have to realise a given positive real function Z(s).

If the function contains poles on the imaginary axis, we can expand this in partial fraction form to yield:

$$Z(s) = k_\infty s + \frac{k_0}{s} + \sum \frac{k_r s}{s^2 + \omega^2} + Z_1(s)$$

The first part of this can be realised as a Foster network, while $Z_1(s)$ is still a positive real function.

The first part of the network has poles only on the $j\omega$ axis, and its real part is zero. The remainder function, that is, $Z_1(s)$ is still positive real, for its real part is still positive and all its poles are on the non-positive (left) half of the s-plane.

$Z_1(s)$ is a minimum reactance function. It is of a lower order than $Z(s)$

Step 2: Susceptance reduction

$Z_1(s)$ will not have poles on the $j\omega$ axis, but it may have zeros on the $j\omega$ axis. This may be so even if the original function $Z(s)$ did not have zeros on the $j\omega$ axis, for they may be introduced during the removal of poles on the $j\omega$ axis in step 1.

We will now remove them, by partial fraction expansion of the admittance function $Y_1(s) = 1/Z_1(s)$.

$$Y_1(s) = 1/Z_1(s) = k_\infty s + \frac{k_0}{s} + \sum \frac{k_r s}{s^2 + \omega^2} + Y_2(s)$$

The first part of this may be realised as:
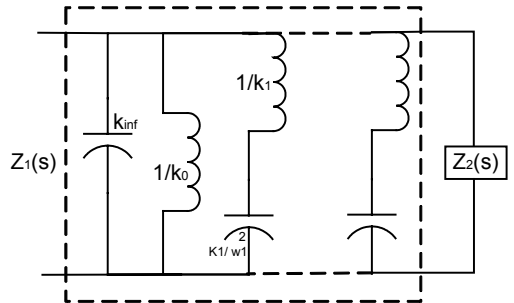
Step 3: Reactance reduction

We may now find that even though $Z_1(s)$ was minimum reactive, $Z_2(s)$ is no longer so, for $j\omega$ axis poles may have been introduced during the removal of $j\omega$ axis zeros. We now repeat Step 1, and then Step 2, repeatedly until the remainder is both minimum reactive and minimum susceptive.

This may be illustrated by a flow chart as follows:
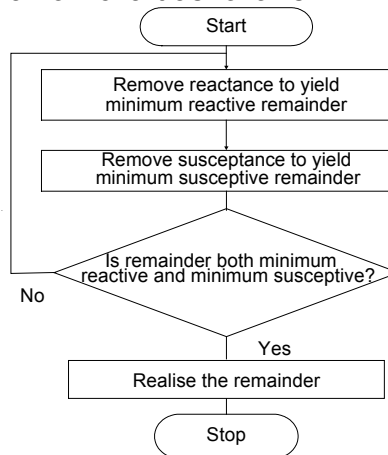
Note: Steps 1 and 2 may be interchanged. We should choose the sequence that gives rise to a simpler realisation.

Example $$Z(s) = \frac{12s^5 + 8s^4 + 32s^3 + 14s^2 + 8s + 2}{4s^5 + 4s^4 + 10s^3 + 9s^2 + 4s + 1}$$

We will first check whether either the numerator or the denominator contains a quadratic factor that can be removed:

$$
\begin{array}{r|l}
& 3s/2 \\ \hline
8s^4 + 14s^2 + 2 & 12s^5 + 32s^3 + 8s \\
& 12s^5 + 21s^3 + 3s \\ \hline
& 11s^3 + 5s
\end{array}
$$

$$
\begin{array}{r|l}
& 8s/11 \\ \hline
11s^3 + 5s & 8s^4 + 14s^2 + 2 \\
& 5s^4 + 40s^2/11 \\ \hline
& 114s^2/11 + 2
\end{array}
$$

$$
\begin{array}{r|l}
& 121s/114 \\ \hline
114s^2/11 + 2 & 11s^3 + 5s \\
& 11s2 + 121s/57 \\ \hline
& 164s/57
\end{array}
$$

$$
\begin{array}{r|l}
& \dfrac{114 \times 57\ s}{11 \times 164} \\ \hline
164s/57 & 114s^2/11 + 2 \\
& 114s^2/11 \\ \hline
& 2
\end{array}
$$

$$
\begin{array}{r|l}
& 82s/57 \\ \hline
2 & 164s/57 \\
& 164s/57 \\ \hline
& 0
\end{array}
$$

We will now repeat this with the denominator polynomial:

$$
\begin{array}{r|l}
& 2s/7 \\ \hline
14s^4 + 9s^2 + 1 & 4s^5 + 10s^3 + 4s \\
& 4s^5 + 18s^3/7 + 2s/7 \\ \hline
& 52s^3/7 + 26s/7
\end{array}
$$

$$
\begin{array}{r|l}
& 49s/28 \\ \hline
52s^3/7 + 26s/7 & 14s^4 + 9s^2 + 1 \\
& 14s^4 + 7s^2 \\ \hline
& 2s^2 + 1
\end{array}
$$

$$
\begin{array}{r|l}
& 26s/7 \\ \hline
2s^2 + 1 & 52s^3/7 + 26s/7 \\
& 52s^3/7 + 26s/7 \\ \hline
& 0
\end{array}
$$

The remainder is zero, and the last non-zero remainder is ($2s^2+1$), indication a pair of jω axis poles at s = + j / √2 and – j / √2

This means that we can remove a partial fraction of the form: $\dfrac{s}{s^2 + \dfrac{1}{2}}$

$$
\begin{array}{r|l}
& 4s^3 + 14s^4 + 8s + 2 \\ \hline
s^2 + 1/2 & 4s^5 + 14s^4 + 10s^3 + 9s^2 + 4s + 1 \\
& 4s5 + \qquad 2s3 \\ \hline
& 14s^4 + 8s^3 + 9s^2 + 4s + 1 \\
& 14s^4 \qquad + 7s^2 \\ \hline
& 8s^3 + 2s^2 + 4s + 1 \\
& 8s^3 \qquad + 4s \\ \hline
& 2s^2 \qquad + 1 \\
& 2s^2 \qquad + 1 \\ \hline
\end{array}
$$

($s^2 + ½$) ($4s^3 + 14s^4 + 8s + 2$) = $4s^5 + 14s^4 + 10s^3 + 9s^2 + 4s + 1$

$$\therefore Z(s) = \frac{12s^5 + 8s^4 + 32s^3 + 14s^2 + 8s + 2}{4s^5 + 4s^4 + 10s^3 + 9s^2 + 4s + 1}$$

$$= \frac{12s^5 + 8s^4 + 32s^3 + 14s^2 + 8s + 2}{(s^2 + \frac{1}{2})(4s^3 + 14s^2 + 8s + 2)}$$

$$= \frac{6s^5 + 4s^4 + 16s^3 + 7s^2 + 4s + 1}{(s^2 + \frac{1}{2})(2s^3 + 7s^2 + 4s + 1)}$$

$$= \frac{As}{s^2 + \frac{1}{2}} + \frac{Bs^3 + Cs^2 + Ds + E}{(2s^3 + 7s^2 + 4s + 1)}$$

We can compute the values of A, B, C, D and E by equating coefficients:

$$B = 6, \quad 2A + C = 4, \quad 7A + D + B/2 = 16, \quad 4A + E + C/2 = 7, \quad a + D/2 = 4, \quad E/2 = 1$$

This gives: $\quad A = 1, \; B = 6, \; C = 2, \; D = 6, \; E = 2$

$$\therefore Z_1(s) = Z(s) - \frac{s}{s^2 + \frac{1}{2}} = \frac{6s^3 + 2s^2 + 6s + 2}{2s^3 + 7s^2 + 4s + 1}$$

We will now consider Y_1(s):

$$Y_1(s) = \frac{2s^3 + 7s^2 + 4s + 1}{6s^3 + 2s^2 + 6s + 2} = \frac{2s^3 + 7s^2 + 4s + 1}{2(s^2 + 1)(3s + 1)} = \frac{s}{s^2 + 1} + \frac{s + \frac{1}{2}}{3(s + \frac{1}{3})} = \frac{s}{s^2 + 1} + Y_2$$
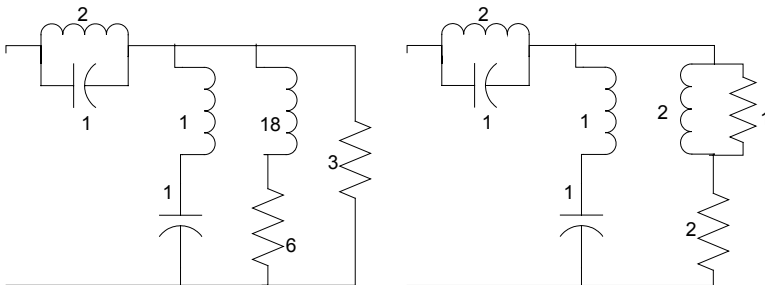
We can now write either

$$Y_2 = \frac{s + \frac{1}{2}}{3(s + \frac{1}{3})} = \frac{1}{3} + \frac{\frac{1}{18}}{s + \frac{1}{3}}$$

or,

$$Z_2 = \frac{1}{Y_2} = \frac{3s + 1}{s + \frac{1}{2}} = 2 + \frac{s}{s + \frac{1}{2}}$$

These two alternatives lead to the following realisations:

Brune synthesis

Brune synthesis starts with the Foster preamble, to remove reactive and susceptive components to yield a minimum reactive and minimum susceptive function. This would be of the form:

$$Z(s) = \frac{a_n s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0}{b_n s^n + b_{n-1} s^{n-1} + \ldots + b_1 s + b_0}$$

$$where \ a_n, b_n, a_0, b_0 \neq 0$$

We then proceed to remove a "minimum resistance" from the remainder to obtain a minimum resistive function. As the real part of $Z(j\omega)$ is a function of ω, care has to be exercised to remove only a minimum resistance, for otherwise, we will be left with a non-positive real function that will not be realisable.

$$Z_1(s) = Z(s) - R_1$$

At the frequency at which the resistance is a minimum, we now have a pure reactive network.

$$Z_1(j\omega_1) = X_1(j\omega_1) = jX_1$$

We now remove this inductance from the network.

$$L_1 = \frac{X_1}{\omega_1}. \qquad Z_2(s) = Z_1(s) - L_1 s$$

Since $Z(s)$ had no poles or zeros on the $j\omega$ axis, the removal of $L_1 s$ from $Z_1(s)$ to yield $Z_2(s)$ has introduced a zero at $s_1 = \pm j\omega_1$ to $Z_2(s)$. We now proceed to remove this:

$$Y_3(s) = \frac{1}{Z_3(s)} = \frac{1}{Z_2(s)} - \frac{\dfrac{s}{L_2}}{s^2 - s^2{}_1}$$

As we saw in the Foster realisation, the removed branch is a shunt element consisting of an inductance and a capacitance in series.
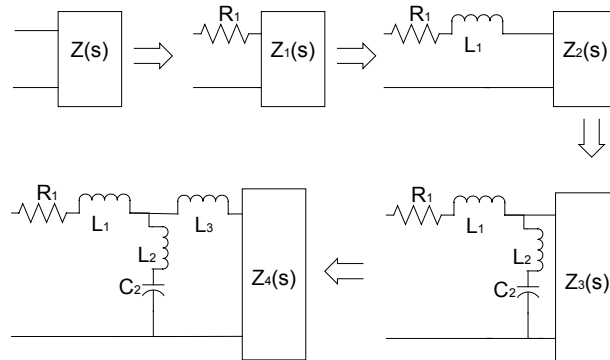
$Z_3(s)$ now has a pole at infinity, which is removed to yield $Z_4(s)$.

$Z_4(s)$ is of the same form as $Z(s)$, except that it is of order (n-2) instead of n. We have completed one cycle of the Brune cycle.

There are two possible cases to be considered in going through this cycle. One is the possibility that L_1 is negative, leaving $Z_2(s)$ positive real, and the other is when L_1 is positive, leaving $Z_2(s)$ non-positive real.

In either case, we will end up with the structure shown at the end of the Brune cycle, with either L_1 or L_3 negative.

The combination of L_1, L_2 and L_3 may be realised with a coupled transformer as shown:
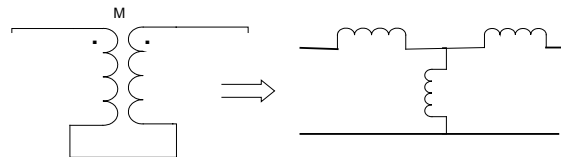
The complete synthesis procedure is illustrated in the following flow chart:
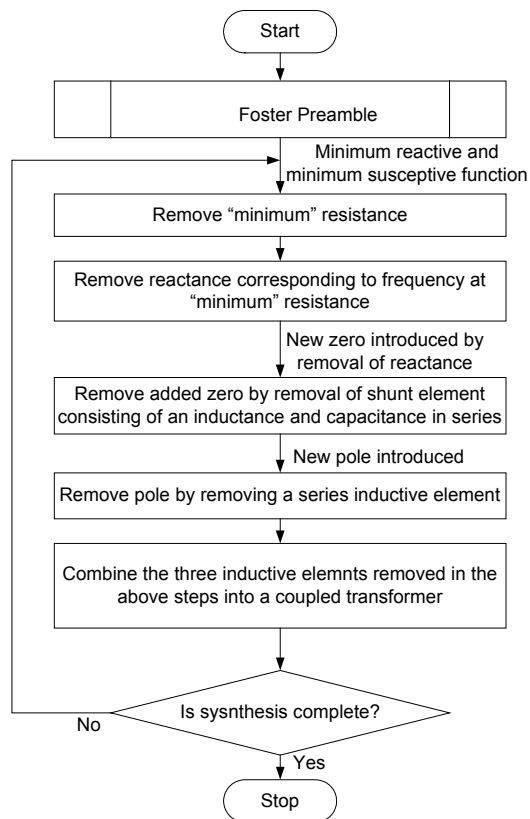
3.1.5 Insertion power function and the refection coefficient

Let us consider an LC filter. As there are no resistance elements, there will be no filter loss in such a filter. Let us see what happens when we connect such a lossless LC filter between a source and a load.
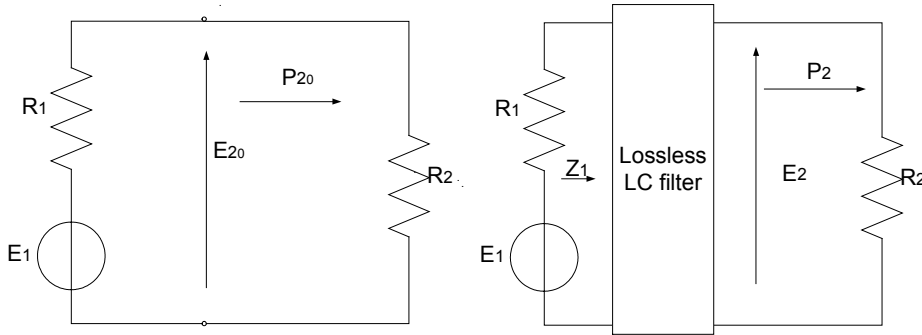
The figure shows the voltage and power distribution before and after the insertion of a lossless LC filter.

Voltage insertion function $\left| \dfrac{E_{2_0}}{E_2} \right| = e^{\alpha}$

Power insertion function $\left| \dfrac{P_{2_0}}{P_2} \right| = \left| \dfrac{E_{2_0}}{E_2} \right|^2 = e^{2\alpha}$

We also have: $[P_{2_0}] = \dfrac{E_1^{\,2}}{(R_1 + R_2)^2} R_2$

For [P$_{20}$] to be a maximum,

$$\frac{\partial P_{2_0}}{\partial R_2} = 0$$

$$-2R_2(R_1 + R_2)^{-3} + (R_1 + R_2)^{-2} = 0$$

$$2R_2 = (R_1 + R_2)$$

$$R_2 = R_1$$

$$[P_{2_0}]_{\max} = \frac{E_1^{\,2}}{4R_1}$$

The maximum value that P_2 can take is only the maximum value of P_{20}

$$\therefore [e^{2\alpha}]_{\min.} = \frac{P_{2_0}}{[P_{2_9}]_{\max.}} = \frac{E_1^{\,2} R_2}{(R_1 + R_2)^2} \cdot \frac{4R_1}{E_1^{\,2}} = \frac{4R_1 R_2}{(R_1 + R_2)^2} \leq 1$$

It is of course obvious that this has to be less than or equal to unity, for the transmitted power cannot be more than the available power. This is also known as the transmission ratio.

The fractional power rejected by the load is used to define the reflection coefficient $\Gamma(s)$.

$$e^{2\alpha} = \frac{P_{2_0}}{P_2} = \frac{R_2}{(R_1 + R_2)^2} \frac{\left|(Z_1 + R_1)^2\right|}{R_1}$$

$$\therefore \frac{(R_1 + R_2)^2}{R_2} e^{2\alpha} = \frac{\left|(Z_1 + R_1)^2\right|}{R_1}$$

The reflection coefficient is defined as

$$\Gamma(s) = \frac{Z_1(s) - R_1}{Z_1(s) + R_1}$$

From the above, we get

$$\Gamma(s)\Gamma(-s) = 1 - e^{-2\alpha} \frac{4R\, R_2}{(R_1 + R_2)^2}$$

3.1.6 Butterworth filters
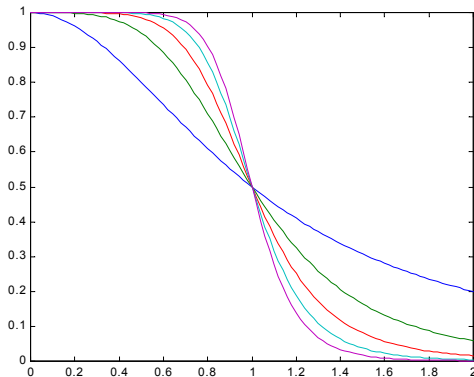
The figure shows a plot of the function $\dfrac{1}{1 + \Omega^{2n}}$ for different values of n from 1 to 5.

This general form is used in the specification of the Butterworth filter to denote the attenuation of power over the frequency range of interest. In general, we consider the y-axis as representing power or the square of the voltage, and the x-axis as representing the normalised frequency. We also usually consider the range of frequencies up to the half-power point ($\Omega=1$) as the pass band, and the region $\Omega > 1$ as the stop band.

Example:

Assume that we want to design a filter with a pass band of 0 – 5 MHz. and an attenuation of at least 20 dB at 7.5 MHz.

We define the normalised frequency as: $\Omega = \dfrac{\omega}{\omega_0}$

At 7.5 MHz., Ω = 1.5.

$$-20 = 10\log_{10}\frac{1}{1+(1.5)^{2n}}$$

$$\frac{1}{1+(1.5)^{2n}} = 10^{-2}$$

$$(1.5)^{2n} = 99$$

$$2n\ln 1.5 = \ln 99$$

$$2n\,(0.405) = 4.595$$

$$2n = \frac{4.595}{0.405} = 11.35$$

$$n = 5.675 \Rightarrow 6$$

$$2n = 12$$

Now let us see what would happen if we had tighter specifications for the pass band. Instead of 3dB attenuation (corresponding to half-power), let us assume that we wanted the attenuation in the pass band to be limited to 0.25 dB. To accommodate this, we use a slightly different form of the general function defining the filter characteristics: $\dfrac{1}{1+\varepsilon^2\Omega^{2n}}$

At $\Omega=1,$

$$-0.25 = 10\log_{10}\frac{1}{1+\varepsilon^2}$$

$$\log_{10}\frac{1}{1+\varepsilon^2} = -0.025$$

$$(1+\varepsilon^2) = 10^{0.025}$$

$$\varepsilon^2 = 1.05925 - 1 = 0.05925$$

$$-20 = 10\log_{10}\frac{1}{1+0.0(1.5)^{2n}}$$

$$\frac{1}{1+0.05925(1.5)^{2n}} = 10^{-2}$$

$$0.05925(1.5)^{2n} = 99$$

$$2n\ln 1.5 = \ln\frac{99}{0.05925} = \ln 1670$$

$$2n\,(0.405) = 7.421$$

$$2n = \frac{7.421}{0.405} = 18.3$$

$$n = 9.15 \Rightarrow 10$$

$$2n = 20$$

We see that with the tighter specifications, we need a much more complex filter with more components.

Further example: Low pass filter, with the following specifications:

Pass band	0 – 20 kHz.
Pass band tolerance	3 dB
Load and source impedance	600 Ω
High frequency attenuation	≥ 40 dB at 100 kHz.

$$|M(\omega)|^2 = \frac{1}{1+\varepsilon^2 \Omega^{2n}}$$

At $\Omega = 1$:

$$3 = 10 \log_{10}(1+\varepsilon^2)$$

$$\therefore \varepsilon = 1$$

For an attenuation of 40 dB at 100 kHz.,

At $\Omega = \dfrac{100}{20} = 5$:

$$40 = 10 \log_{10}(1+\Omega^{2n}) = 10 \log_{10}(1+5^{2n})$$

$$1 + 5^{2n} = 10000$$

$$5^{2n} = 9999$$

$$2n \ln 5 = \ln 9999$$

$$2n * 1.609 = 9.210$$

$$n = 2.86 \Rightarrow 3$$

$$|M(\omega)|^2 = \frac{1}{1+\varepsilon^2 \Omega^{2n}} = \frac{1}{1+\Omega^6},$$

Substituting $s = j\Omega$:

$$\Gamma(s)\Gamma(-s) = 1 - \frac{4R_1 R_2}{(R_1+R_2)^2} \frac{1}{1-s^6}$$

As R_1 and R_2 are given as 600 Ω,

$$\Gamma(s)\Gamma(-s) = 1 - \frac{4R_1 R_2}{(R_1+R_2)^2} \frac{1}{1-s^6} = 1 - \frac{1}{1-s^6} = \frac{-s^6}{1-s^6} = \frac{-s^6}{(1-s^{3^2})} = \frac{-s^6}{(1-s^3)(1+s^3)}$$

$$= \frac{-s^6}{(1-s)(1+s+s^2)(1+s)(1-s+s^2)} = \frac{(s^3)}{(1+s)(1+s+s^2)} \frac{(-s^3)}{(1-s)(1-s+s^2)}$$

Let us chose $\Gamma(s) = \dfrac{s^3}{(1+s)(1+s+s^2)}$

But $\Gamma(s) = \dfrac{Z(s)-1}{Z(s)+1}$,

where $Z(s)$ *is the normalised impedance*

$$\Gamma(s)Z(s) + \Gamma(s) = Z(s) - 1$$

$$Z(s)[\Gamma(s)-1] = -[\Gamma(s)+1]$$

$$Z(s) = \frac{1+\Gamma(s)}{1-\Gamma(s)} = \frac{1+\dfrac{s^3}{1+2s+2s^2+s^3}}{1-\dfrac{s}{1+2s+2s^2+s^3}} = \frac{1+2s+2s^2+2s^3}{1+2s+2s^2}$$

This may now be synthesised. Using continued fraction expansion:

$$2s^2 + 2s + 1 \overline{\big)\, 2s^3 + 2s^2 + 2s + 1} \quad \cfrac{s}{}$$

	s			2s
2s² + 2s + 1	2s³ + 2s² + 2s + 1		s + 1	2s² + 2s + 1
	2s³ + 2s² + s			2s² + 2s
	s + 1			1

$$\frac{1 + 2s + 2s^2 + 2s^3}{1 + 2s + 2s^2} = s + \cfrac{1}{2s + \cfrac{1}{s + \cfrac{1}{1}}}$$
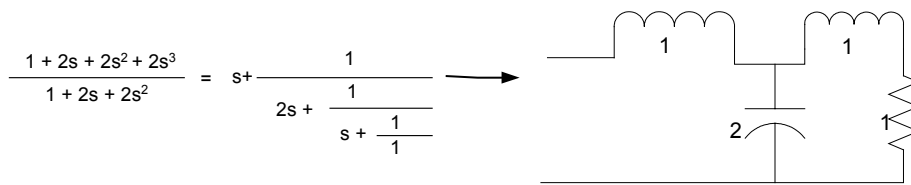
Now we need to de-normalise this, to obtain the final implementation:

$R_0 = 600\ \Omega$
$\omega_0 = 20\ 000 \times 2\ \pi$

The corresponding transformations are: $R \Rightarrow R \times R_0$
$L \Rightarrow L \times R_0\ /\ \omega_0$
$C \Rightarrow C\ /\ (R_0 \times \omega_0)$
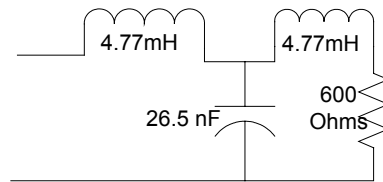
3.1.7 Chebyschev filters

In principle, this (and other mathematically defined filters such as the elliptic filter) is similar to the Butterworth filter in that we define a mathematical function approximating the characteristics that we desire.

Here, we define $|M(\omega)|^2 = \dfrac{1}{1 + \varepsilon^2\,C_n^{\ 2}(\Omega)}$

Where $C_n(\Omega) = \begin{cases} \cos(n\cos^{-1}\Omega),\ for\ |\Omega| \le 1 \\ \cosh(n\cosh^{-1}\Omega),\ for\ |\Omega| > 1 \end{cases}$

We have $C_0\ (\Omega) = 1$
and $C_1\ (\Omega) = \Omega$

We can also obtain a recursive expression for higher order values of C_n as:

$$C_n(\Omega) = 2\Omega C_{n-1}(\Omega) - C_{n-2}(\Omega)$$

The shape of the function generated by this approximation is as shown:

We will consider the same example as before: As the pass band tolerance is 3 dB, we again have $\varepsilon = 1$.

$$40 = 10 \log (1 + C_n^{\,2}(5))$$

$$C_n^{\,2}(5) = 9999$$

$$C_n(5) = 99.99 = \cosh (n \cosh^{-1} 5)$$

$$n = \frac{\cosh^{-1} 99.99}{\cosh^{-1} 5} = \frac{5.298}{2.312} = 2.29 \Rightarrow 3$$

We can find $C_n(\Omega)$ using the recursive algorithm given earlier, and we get

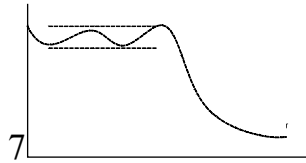$$C_2(\Omega) = 2\Omega C_1(\Omega) - C_0(\Omega) = 2\Omega^2 - 1$$

$$C_3(\Omega) = 2\Omega C_2(\Omega) - C_1(\Omega) = 2\Omega(2\Omega^2 - 1) - \Omega = 4\Omega^3 - 3\Omega$$

$$|M(\omega)|^2 = \frac{1}{1 + \varepsilon^2 C_n^{\,2}(\Omega)}$$

$$= \frac{1}{1 + (4\Omega^3 - 3\Omega)^2}$$

Substituting $s = j\Omega$;

$$\Gamma(s)\Gamma(-s) = 1 - \frac{1}{1 - 9s^2 - 24s^4 - 16s^6}$$

This has to be separated into two parts, and then we can proceed as before. In this case, we will have to resort to numerical methods to factorise the expression into two parts. As before, we will end up with a filter with three components. However, for filters with tighter specifications, the Chebyschev approximation yields lower degree implementations than the Butterworth approximation.

Pole-zero patterns

In the Butterworth filter, the poles lie on a semi-circle centered on the origin, on the left half of the s-plane. All the zeros are at infinity. In the Chebyshev filter, we bring the poles closer to the $j\omega$ axis in an attempt to increase the cut-off slope. However, this introduces a ripple in the pass band. The reverse is attempted in the Bessel filter, where the poles are moved further away from the circular locus. The elliptic filter has its poles closer to the axis as in the Chebyshev filter, but also has the zeros moved closer to the real axis, from infinity. This filter has ripples in the cut-off band as well.

3.1.8 Transformation to obtain high-pass characteristics

Once you have obtained a low-pass filter, it is quite simple to get a high-pass filter by transformation. Consider the first low-pass Butterworth filter we designed:

Pass band	0 – 20 kHz.
Pass band tolerance	3 dB
Load and source impedance	600 Ω
High frequency attenuation	$\geq$ 40 dB at 100 kHz.

$$\frac{1 + 2s + 2s^2 + 2s^3}{1 + 2s + 2s^2}$$
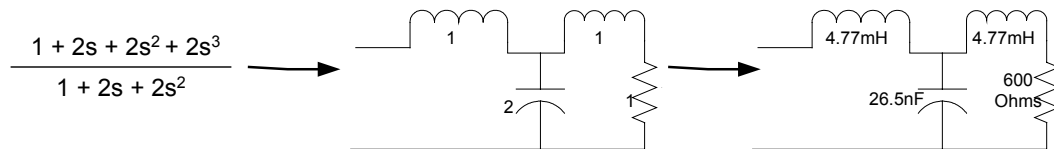
The filter itself is only the LC section, the terminal resistance being the load resistance. This is illustrated below, where both the source and load resistances are shown:
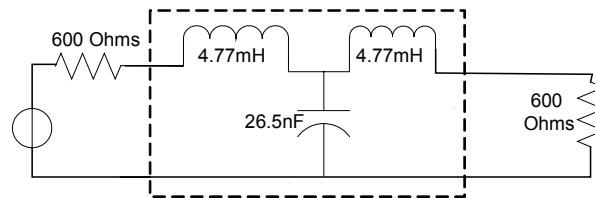
Starting with the normalised function, we can obtain a high pass filter merely by the transformation s $\Rightarrow$ 1/s. The transformation is invariant at Ω = 1, while all the other frequencies are inverted. In the case under study, we had:

$$\left|M(\omega)\right|^2 = \frac{1}{1 + \varepsilon^2 \Omega^{2n}} = \frac{1}{1 + \Omega^6},$$

Substituting $s = j\Omega$:

$$\Gamma(s)\Gamma(-s) = 1 - \frac{1}{1 - s^6}$$

This would transform to:

$$\Gamma(s)\Gamma(-s) = 1 - \frac{1}{1 - s^{-6}} = \frac{1}{1 - s^6} = \frac{1}{(1-s)(1+s+s^2)(1+s)(1-s+s^2)} = \frac{1}{(1+s)(1+s+s^2)} \frac{1}{(1-s)(1-s+s^2)}$$

Chose $\Gamma(s) = \dfrac{1}{(1+s)(1+s+s^2)}$

But $\Gamma(s) = \dfrac{Z(s) - 1}{Z(s) + 1}$,

$$\therefore Z(s) = \frac{1+\Gamma(s)}{1-\Gamma(s)} = \frac{1 + \dfrac{1}{1+2s+2s^2+s^3}}{1 - \dfrac{1}{1+2s+2s^2+s^3}} = \frac{2+2s+2s^2+s^3}{2s+2s^2+s^3}$$

By continued fraction expansion, we have:

$$\begin{array}{r} 1/s \\ \hline 2s + 2s^2 + s^3 \overline{\smash{\big)}\, 2 + 2s + 2s^2 + s^3} \\ \underline{2 + 2s + s^2} \\ s^2 + s^3 \end{array}$$

$$\begin{array}{r} 2/s \\ \hline s + s^2 \overline{\smash{\big)}\, 2 + 2s + s^3} \\ \underline{2 + 2s} \\ s^2 \end{array}$$

$$\frac{2 + 2s + 2s^2 + s^3}{2s + 2s^2 + s^3} = 1/s + \cfrac{1}{2/s + \cfrac{1}{1/s + 1}}$$
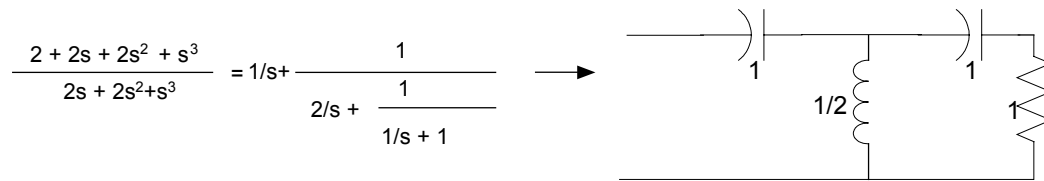
De-normalising will then yield:
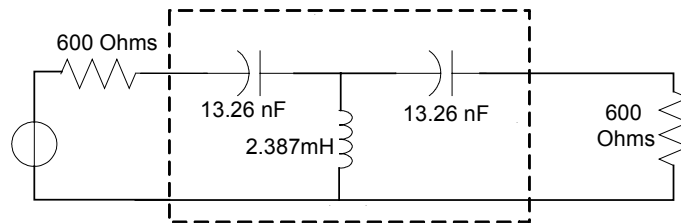
This filter will have the following specifications, they being the inverse of those of the original low pass filter:

Pass band	20 kHz and above.
Pass band tolerance	3 dB
Load and source impedance	600 Ω
Low frequency attenuation	$\geq$ 40 dB at 4 kHz.

The transformation could have been obtained by operating directly on Z(s) (instead of on the insertion power characteristic) or even by transforming each of the filter elements. The filter elements transform as follows:

Low-pass prototype (normalised)	Low-pass	High-pass	Band-pass	Band-stop
L	L / W_c	$1 / L W_c$	$L / (BW)$ $(BW) / W_c^2 L$	$(BW)L / W_c^2$ $1 / (BW)L$
C	C / W_c	$1 / C W_c$	$(BW) / W_c^2 C$ $C / (BW)$	$1 / (BW)C$ $(BW)C / W_c^2$

Note that the above table allows for de-normalisation for frequency only, and does not take impedance normalisation into account.

3.1.9 Band pass and band stop filters

It is possible to obtain band pass and band stop characteristics by frequency transformation of low pass filters, as we found with high pass filters. However, it is much more convenient to construct these as cascade and parallel combinations of a low pass and a high pass filter as this would give us independent control of both the cut-off frequency and the roll-off characteristics on either side. We illustrate this, using ideal filter characteristics in the following diagram:
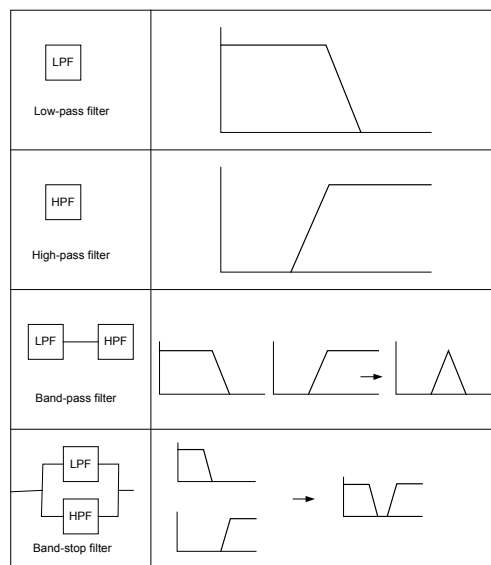
3.2 Active filters

The filters that we have studied so far have all been passive filters – that is, they consisted only of passive components. The Butterworth and Chebyschev filters were both passive reactive filters and had only capacitors and inductors.

With the advent of LSI and VLSI, it became necessary to look at the possibility of implementing a filter circuit within an IC. It is possible to implement resistors and capacitors using film technology (even though implementing resistors is comparatively more difficult), but there is no way to construct an inductor. This necessitated the development of techniques to realise networks with inductors using only resistors and capacitors. Active circuits such as gyrators and negative impedance converters are used for this purpose.

NICs and gyrators are implemented using operational amplifiers. Operational amplifiers may also be used directly to implement transfer functions representing filter characteristics, using their high gaim, high input impedance and low output impedance characteristics. There are however limitations on their frequency spectrum, and filters operating at high frequencies, such as in communication circuits are still implemented using inductors.

3.2.1 Active filters using gyrators

A gyrator is a device that converts a load impedance in to an input impedance proportional to its inverse. Assuming a voltage controlled current source (VCCS), we can represent a gyrator as follows. [The constant of proportionality (g) between the voltage and the current is called its gyration ration.]
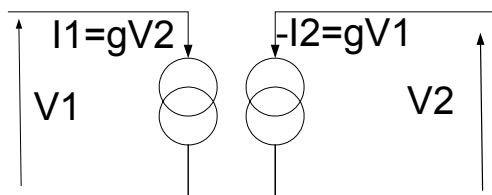
Representing this circuit with its (ABCD) parameters, we have:

$$V_1 = AV_2 + BI_2$$

$$I_1 = CV_2 + DI_2$$

$$A = \left.\left|\frac{V_1}{V_2}\right|\right|_{I_2=0} = 0, \qquad B = \left.\left|\frac{V_1}{I_2}\right|\right|_{V_2=9} = -\frac{1}{g}, \qquad C = \left.\left|\frac{I_1}{V_2}\right|\right|_{I_2=0} = g, \qquad D = \left.\left|\frac{I_1}{I_2}\right|\right|_{V_2=0} = 0$$

This gives:

$$V_1 = -g^{-1}I_2$$
$$I_1 = gV_2$$
or
$$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} 0 & g^{-1} \\ g & 0 \end{bmatrix} \begin{bmatrix} V_2 \\ -I_2 \end{bmatrix}$$

Symbolically, a gyrator is represented by:
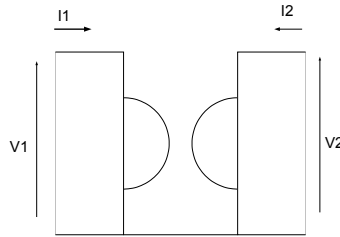
We will now show that this circuit represents an inversion of the impedance.
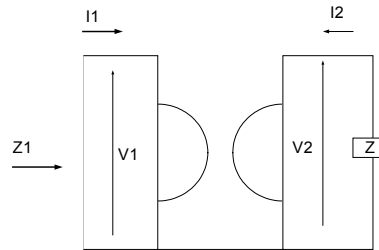
$$Z_1 = \frac{V_1}{I_1} = \frac{-g^{-1}I_2}{gV_2} = \frac{1}{-g^2\dfrac{V}{I_2}} = \frac{1}{g^2 Z}$$

Example
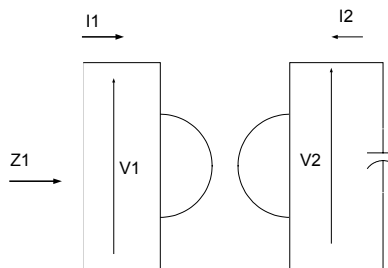
If we have a capacitance C at the output, then,

$$Z = \frac{1}{Cs}$$

$$Z_1 = \frac{1}{g^2 Z} = \frac{Cs}{g^2}$$

This is the same as the impedance of an inductance L = C/g^2.

To obtain the equivalence for a series inductance, we have to connect two gyrators in cascade:
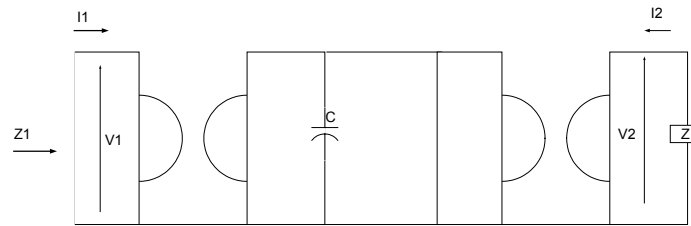
If the terminal impedance is Z, the impedance looking in at the intermediate port is $1/g^2Z$, as before. This now appears in parallel with the impedance of the capacitance C to give $\frac{1}{Cs+g^2Z}$

With this at the output port of the first gyrator, the input impedance is:

$$Z_1 = \frac{1}{g^2\,\dfrac{1}{Cs+g^2Z}} = \frac{Cs+g^2Z}{g^2} = \frac{Cs}{g^2}+Z$$

This is equivalent to a series inductance of C/g^2, followed by the original impedance across the final output port.

Let us now see how we can implement a complete LC filter without the use of inductors. We will consider a circuit derived in an earlier lecture::
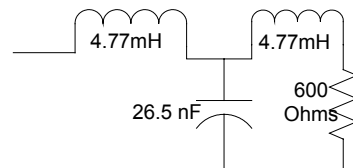
This may be realised as:
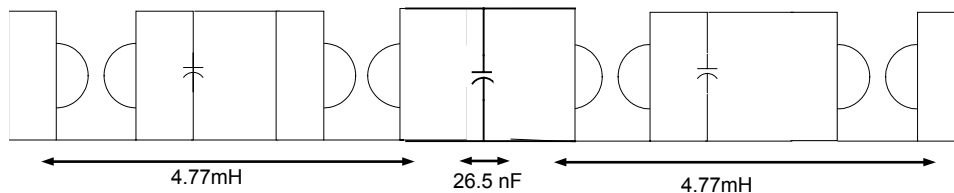
Realisation of the gyrator

The following circuit is a possible realisation of a gyrator using operational amplifiers. It approximates an ideal gyrator as $R_0 \to 0$.
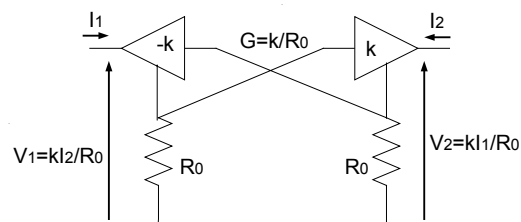
3.2.2 Active filters using NICs

Another device used in the design of active filters is the negative impedance converter (NIC).

An ideal NIC is a two-port network, which converts a load impedance to an input impedance, which is proportional to the negative of the load impedance.

The following figures illustrate voltage and current controlled NICs

Voltage controlled NIC

Current controlled NIC
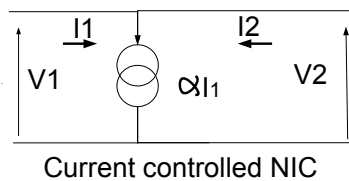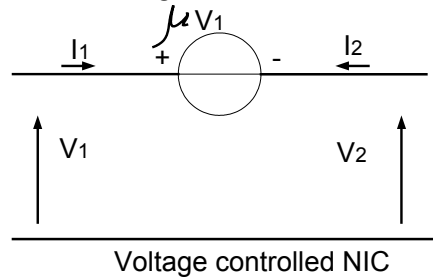
Their ABCD parameters are:

Voltage controlled NIC (VNIC):

$$\begin{bmatrix} \dfrac{1}{1-\mu} & 0 \\ 0 & 1 \end{bmatrix}$$

Current controlled NIC (INIC)

$$:\begin{bmatrix} 1 & 0 \\ 0 & \dfrac{1}{1-\alpha} \end{bmatrix}$$

Realisation of an INIC (The ration k= R_1/R_2 = 1/(α-1))
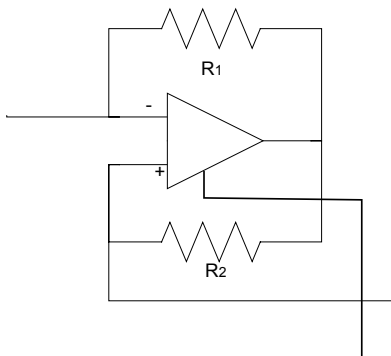
3.2.3 The state-variable filter

We have earlier (in Chapter 2) seen how to obtain a state-space representation of a transfer function, and how such a system may be realised using a series of integrators and summers. This approach may be used to implement a filter using operational amplifiers. An operational amplifier with capacitive feedback functions as an integrator while resistive feedback produces a summer.

We will see how to connect two integrators and a summer to obtain a second order filter realisation.
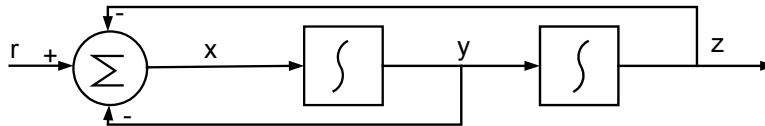
We have the relationships:

$$x = r - y - z, \qquad y = \frac{x}{s}, \qquad z = \frac{y}{s}$$

Rearranging to get the transfer functions between r and the other variables:

$$x = r - \frac{x}{s} - \frac{x}{s^2}$$

$$r = x\left[1 + \frac{1}{s} + \frac{1}{s^2}\right] = x\left[\frac{s^2 + s + 1}{s^2}\right]$$

$$\frac{x}{r} = \frac{s^2}{s^2 + s + 1}$$

$$\frac{y}{r} = \frac{s}{s^2 + s + 1}$$

$$\frac{z}{r} = \frac{1}{s^2 + s + 1}$$

We see that the outputs x, y and z correspond to standard second order high-pass, band-pass and low-pass filter responses. The coefficients of the terms in s and s^2 may be adjusted simply by changing the gain and integrator time constants as desired. We have also seen before that any higher order transfer function may be broken down into a series of second order terms and possibly one first order term. This fact is used to implement higher order filters.

3.2.4 Switched capacitor filters

Even though it is possible to build in resistors into integrated circuits, they do present difficulties. It is advantageous if we can dispense with their use and use only capacitors. It is also difficult to achieve great accuracy in the construction of individual resistors and capacitors (typical accuracies are in the order of 30%) whereas accuracies of almost three orders of magnitude lower (about 0.05%) are

possible if we only require ratios between capacitors, rather than their absolute values. These are the motivations for the development of switched capacitor filters. We will first try to understand how switched capacitors can simulate a resistor.
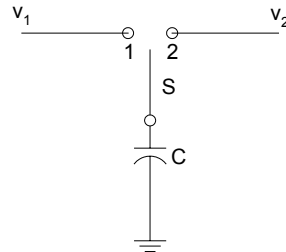
The figure shows a capacitor C connected to a switch S that has two positions 1 and 2, connected to two terminals maintained at voltages v_1 and v_2. For simplicity, we will assume that all connections are ideal and without resistance, so that the capacitor charges up instantly to the relevant voltage upon connection to either of the two terminals.

When S is at 1, the capacitor charges up to voltage v_1, so that the charge on the capacitor is Cv_1. If the switch is now thrown over to 2, the charge changes to Cv_2, that is, a charge of $C(v_1 - v_2)$ is delivered from 1 to 2 when the switch is thrown from 1 to 2, assuming that $v_1 > v_2$.

Let us now assume that the switch toggles between 1 and 2, at a frequency of f Hz. Then, the charge transfer per unit time (that is, the current flow) is

$$i = C(v_1 - v_2)f$$

Comparing this with the current flow through a resistor R connected between 1 and 2, we can write:

$$R = 1/Cf$$

We have a simulated resistor, whose value may be changed at will simply by changing the clock frequency! The other advantage we were looking for, that of only requiring ratios between capacitors rather than their absolute values, becomes apparent when we use this simulated resistor to construct an integrator using an operational amplifier. It is best to use ratios close to unity, for greater accuracy.
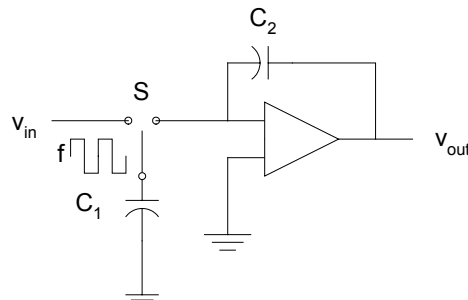
The above integrator has a natural frequency of C_1f/C_2 or a time constant of C_2/C_1f.

3.3 Modern trends in circuit analysis and design

In this section, we will briefly examine some of the more modern trends in circuit analysis and design, especially those techniques that attempt to automate the process through the use of computers.

We are familiar with nodal and mesh analysis of electric circuits. We will begin our study with an attempt to formulate the system equations automatically, so as to enable a computer implementation of the equation formulation stage. We have already seen how matrix analysis would allow us to solve a set of equations, once they have been formulated.

We also need to study how to model non-linear circuit elements. We will look at the simpler models for diodes and transistors as examples of modelling.

A variety of software is available for circuit simulation. One of the most widely used among them is the Spice family, developed in the 1970s at the University of California at Berkley. Different implementations of this basic software are available from a number of suppliers. We will take a very brief look at pSpice, which runs under the Windows environment.

We will then examine how slight differences in parameter values would affect the performance of a circuit. This study is facilitated by the application of Tellegan's theorem, which is a rather unexpected result that leads to a great simplification of the problems involved in the study of sensitivity.

Finally, we examine the application of AI techniques such as genetic algorithms in automatic design. These would take us beyond the mere application of computers to replicate what we would otherwise do manually, to totally new methods of looking at the issues encountered in circuit design. It has been reported that such techniques generate new topologies, rather than merely optimising the parameter values of a standard design.

3.3.1 Automatic equation formulation – modified nodal analysis

In the first instance, before we can think of equation formulation, we need to be able to present the data (the topology of the circuit, the nature and the numerical values of the components, etc.) in a machine-amenable form. The simplest (from the machine point of view) that we can think of is a list of components and sources giving the nodes to which each one of them is connected, along with the value of the element. From a user point of view, the simplest would probably be a schematic diagram (which will give the topological information) along with the value of each element. Graphical User Interface (GUI) software that generates a list (of the first type) from a schematic diagram is now freely available, so we will assume that the input is a list, which is more easily understood by a machine.

We now have to make a decision between nodal analysis and mesh analysis. In manual equation formulation involving comparatively small circuits with only a few components, you would have noticed that sometimes it is simpler to use nodal analysis while at other times, it is simpler to use mesh analysis. This generally depends on the number of independent node-pairs and the number of independent loops present in the circuit. For simple circuits it is easy to figure this out by inspection of a schematic diagram, especially when it is planer. However, for the automatic analysis of complex circuits, mesh analysis will involve the construction of a tree and the identification of tree-links that are associated with each loop current, while the identification of node pairs is very simple. We only need to identify one node as the reference node, and then each of the other nodes along with the reference node will form an independent node-pair.

There are other reasons for the choice of nodal analysis.

> Most active circuits are more easily modelled using a current source (see section 3.3.3 below) than a voltage source, and this leads naturally to nodal formulation of equations.

> In automatic equation formulation, it is a great advantage to be able to modify an existing description easily, after the addition or removal of a branch or element. This can be very easily accomplished in nodal analysis.

If we adopt this approach, we will have two types of equations to describe the circuit:

> Kirchoff's Current Law (KCL) equations (one for each node, except for the reference node)

> Branch constitutive equations (one for each branch)

We will consider a simple circuit consisting of an ideal current source i_s and three resistances R_1, R_2 and R_3 connected as shown:
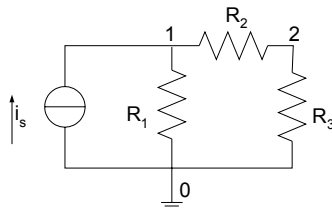
KCL Equations: Node 1: $\qquad i_s + i_{01} + i_{21} \quad = 0$

Node 2: $\qquad i_{12} + i_{02} \quad\quad = 0$

Branch equations: $\quad$ R$_1$: $\quad i_{10} = -i_{01} = v_1 / R_1$

R$_2$: $\quad i_{12} = -i_{21} = (v_1 - v_2) / R_2$

R$_3$: $\quad i_{20} = -i_{02} = v_2 / R_3$

By substituting the second set of equations into the first set, we can write:

$$i_s - \frac{v_1}{R_1} - \frac{v_1 - v_2}{R_2} = 0$$

$$\frac{v_1 - v_2}{R_2} - \frac{v_2}{R_3} = 0$$

To solve for v_1 and v_2, we will rewrite these equations after rearranging as:

$$v_1\left[\frac{1}{R_1} + \frac{1}{R_2}\right] + v_2\left[-\frac{1}{R_2}\right] = i_s$$

$$v_1\left[-\frac{1}{R_2}\right] + v_2\left[\frac{1}{R_2} + \frac{1}{R_3}\right] = 0$$

If we write $G_i = 1/R_i$, the equations become:

$$\begin{bmatrix} (G_1 + G_2) & -G_2 \\ -G_2 & (G_2 + G_3) \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} i_s \\ 0 \end{bmatrix}$$

In general, we can write $Gv = i$, where **G** is the conductance matrix, **v** is the node voltage vector and **i** is the vector of current sources. As we saw in the example, the conductance matrix $G = \begin{bmatrix} g_{jk} \end{bmatrix}$ and the right-hand-side vector $i = \begin{bmatrix} i_k \end{bmatrix}$ are formed as follows:

g_{jj} is equal to the sum of all conductances connected to node j, and
g_{jk} is equal to negative sum of all conductances connected between nodes j and k
i_k is equal to the sum of all independent currents flowing into node k.

Note that we have considered only resistive branches and independent current sources in this analysis. How would we accommodate voltage sources? Let us modify the example circuit we considered earlier by the addition of a real voltage source as shown.
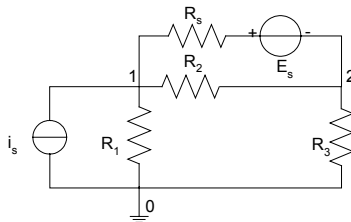
The new equations are:
$$\begin{bmatrix} (G_1 + G_2 + G_s) & -(G_2 + G_s) \\ -(G_2 + G_s) & (G_2 + G_3 + G_s) \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} i_s + E_s G_s \\ 0 \end{bmatrix}$$

where $G_s = 1/R_s$. We could obtain this either by working through the KCL and branch constitutive equations as before, or more simply by replacing the voltage source with its Norton equivalent:
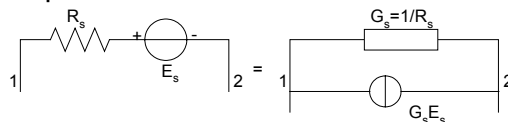
However, we will run into difficulties if we were to allow for ideal voltage sources, with $R_s = 0$ or $G_s \rightarrow \infty$. One way out of this difficulty is to introduce a gyrator, where a current source is connected to one port, producing a voltage across the other as shown:

An ideal current source i. equal in magnitude to E, connected through a gyrator, appears as an ideal voltage source E. If we use this concept with the circuit that we considered earlier (assuming R_s to be zero) we will end up with the following circuit:

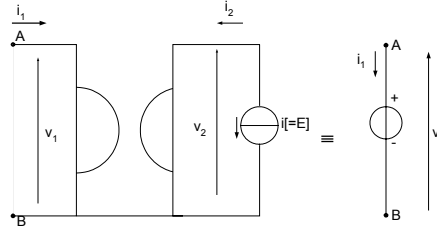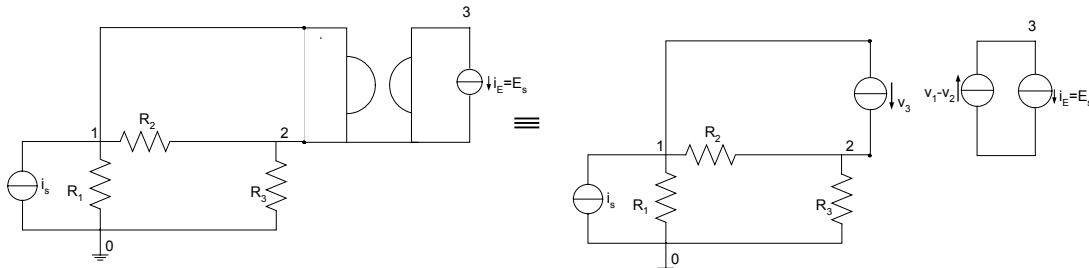We now have an additional node (node 3) and the equations describing the system are:

$$\begin{bmatrix} (G_1 + G_2) & -G_2 & 1 \\ -G_2 & (G_2 + G_3) & -1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} i_s \\ 0 \\ E_s \end{bmatrix}$$

Note that v_3 that appears in the voltage vector is not a voltage, but a (pseudo) current, while E_s appears in the current vector.

In the modified nodal analysis, we dispense with the circuitous route of connecting a gyrator and instead, hypothesise a current through the voltage source as shown below:
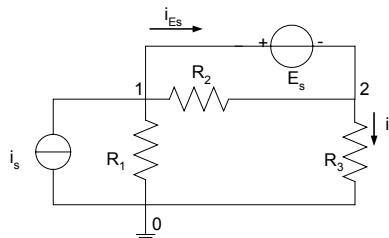
The equations describing the system are identical (except for the slight change in notation) to those obtained earlier:

$$\begin{bmatrix} (G_1 + G_2) & -G_2 & 1 \\ -G_2 & (G_2 + G_3) & -1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ i_{Es} \end{bmatrix} = \begin{bmatrix} i_s \\ 0 \\ E_s \end{bmatrix}$$

The last equation however is not a nodal equation, but a branch constitutive equation. The figure also shows a current i_3 through the branch connecting node 2 to the reference node 0. This has been introduced to illustrate how we can specify a current if it is required as an output variable. As we have already accepted the idea of including branch constitutive equations among our circuit description, there should be no difficulty in adding another such equation if required. The enhanced set of equations now becomes:

$$\left[\begin{array}{cccc|cc} (G_1 + G_2) & -G_2 & | & 1 & 0 \\ -G_2 & G_2 & | & -1 & 1 \\ \hline 1 & -1 & | & 0 & 0 \\ 0 & G_3 & | & 0 & -1 \end{array}\right] \left[\begin{array}{c} v_1 \\ v_2 \\ \hline i_{Es} \\ i_3 \end{array}\right] = \left[\begin{array}{c} i_s \\ 0 \\ \hline E_s \\ 0 \end{array}\right]$$

Note that with the introduction of the last equation defining i_3, the second equation has also changed. However, the first two equations are still nodal equations while the last two are branch constitutive equations. This method of formulating circuit descriptive equations is known as modified nodal analysis.

3.3.2 Contribution of individual elements towards the conductance matrix – The stamp of an element

In the last section, we said that nodal analysis (and by implication, modified nodal analysis) facilitates the addition and / or removal of branches or elements. This becomes clear when we examine the formation of the modified nodal equations. We will first look at the formulation of the conductance matrix **G**, shown at the top left hand corner of the LHS and the current vector [**i**] at the top of the RHS of the composite modified nodal analysis (MNA) equations.

We will consider the circuit that was used as an example in the previous section, and write down the contribution of each component to the overall system descriptive equations. We should remember that in the final formulation, we postulated a current i_3 from node 2 to the reference node, so that the resistance R_3 will appear only among the branch constitutive equations.
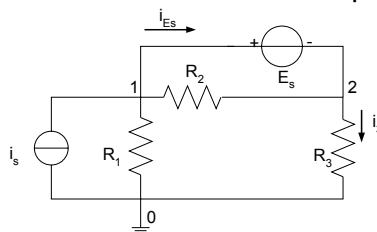
Contribution of Resistance R_1 (Conductance G_1):
    Add G_1 to g_{11}, subtract G_1 from g_{12}

Contribution of Resistance R_2 (Conductance G_2):
    Add G_2 to g_{11} and g_{22}, subtract G_2 from g_{12} and g_{21}.

Contribution of Ideal current source i_s:

> Add i_s to i_{11} [In this case the current source is connected between the reference node 0 and node 1. Hence it generates only one entry. Otherwise, there would be a $-i_s$ at the position corresponding to the other node.]

We call the affect of each element its **stamp**.

The stamp of resistance R_1 is thus:

	v_1	v_2	RHS
Node 1	G_1	-	-
Node 2	-	-	-

The stamp of resistance R_2 is:

	v_1	v_2	RHS
Node 1	G_2	$-G_2$	-
Node 2	$-G_2$	G_2	-

The stamp of Ideal current source i_s is:

	v_1	v_2	RHS
Node 1	-	-	i_s
Node 2	-	-	-

We will now consider the rest of the MNA equations, arising from the stamp of the ideal voltage source (with postulated current i_{Es}) and the stamp of the branch current (we have introduced i_3 as an output variable). As noted earlier, these generate branch constitutive equations and not nodal equations.

The stamp of the ideal voltage source E_s is:

	v_1	v_2	i_{Es}	i_3	RHS
Node 1			1		
Node 2			-1		
Branch with voltage source	1	-1			E_s
Branch with current i_3					
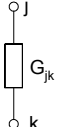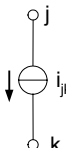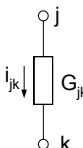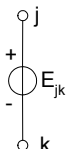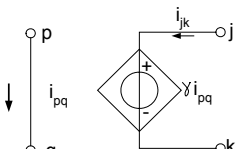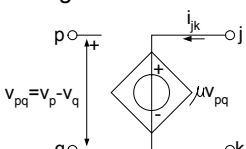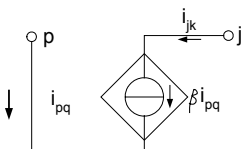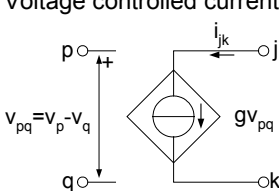
The stamp of the branch current i_3 is:

	v_1	v_2	i_{Es}	i_3	RHS
Node 1					
Node 2				1	
Branch with voltage source					
Branch with current i_3		G_3		-1	

Combining all the stamps, we can now complete the MNA equations as:

$$
\begin{bmatrix}
(G_1 + G_2) & -G_2 & | & 1 & 0 \\
-G_2 & G_2 & | & -1 & 1 \\
---------- & & |---------- & \\
1 & -1 & | & 0 & 0 \\
0 & G_2 & | & 0 & -1
\end{bmatrix}
\begin{bmatrix}
v_1 \\
v_2 \\
-- \\
i_{Es} \\
i_3
\end{bmatrix}
=
\begin{bmatrix}
i_s \\
0 \\
-- \\
E_s \\
0
\end{bmatrix}
$$

The following is a summary of the stamps of different elements:

Conductance connected between nodes j and k

	v_j	v_k	RHS
Node j	G_{jk}	$- G_{jk}$	
Node k	$- G_{jk}$	G_{jk}	

Current source between j and k

	v_j	v_k	RHS
Node j			$-i_{jk}$
Node k			i_{jk}

Branch current as an output variable

	v_j	v_k	i_{jk}	RHS
Node j			1	
Node k			-1	
Branch jk	G_{jk}	$-G_{jk}$	-1	

Ideal voltage source

	v_j	v_k	i_{jk}	RHS
Node j			1	
Node k			-1	
Branch jk	1	-1		E_{jk}

Current controlled voltage source

	v_j	v_k	i_{jk}	i_{pq}	RHS
Node j			1		
Node k			-1		
Branch jk	1	-1		γ	

Voltage controlled voltage source

	v_j	v_k	i_{jk}	v_p	v_q	RHS
Node j			1			
Node k			-1			
Branch jk	1	-1		$-\mu$	μ	

Current controlled current source

	v_j	v_k	i_{pq}	RHS
Node j			β	
Node k			$-\beta$	

Voltage controlled current source

	v_j	v_k	v_p	v_q	RHS
Node j			g	$-g$	
Node k			$-g$	g	

Example

We will consider the case of an ideal operational amplifier as a special case of a voltage controlled voltage source (VCVS). The configuration and the stamp of a VCVS are:
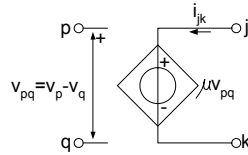
	v_j	v_k	i_{jk}	v_p	v_q	RHS
Node j			1			
Node k			-1			
Branch jk	1	-1		$-\mu$	μ	

We assume the following characteristics of an ideal operational amplifier:

> Very high input impedance
> Very high gain

Comparing the configuration of the operational amplifier with the VCVS, we make the following identification:
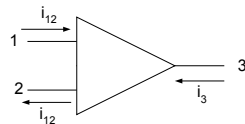
Node p	Node 1
Node q	Node 2
Node j	Node 3
Node k	Reference node
Branch current i_{jk}	i_3

Then the stamp becomes:

	v_3	v_0	I_3	v_1	v_2	RHS
Node 3			1			
Ref. Node			-1			
Branch 3	1	-1		$-\mu$	μ	

Deleting the rows and columns corresponding to the reference node, we have:

	v_3	I_3	v_1	v_2	RHS
Node 3		1			
Branch 3	1		$-\mu$	μ	

We may omit the entry in column v_3 as $1 << \mu$, thus eliminating that column. Finally, we can eliminate columns v_1 and v_2 by adding them together, giving:

	I_3	RHS
Node 3	1	

We will now use this reduced stamp to study a circuit with an ideal operational amplifier.
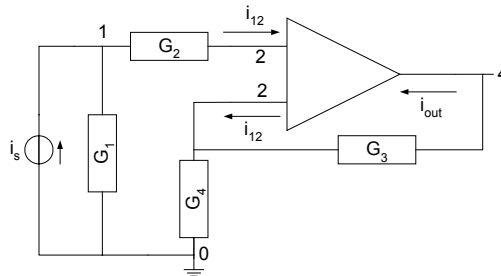
Using the insight obtained from the previous exercise, we have named both the inputs of the operational amplifier with the same name, as node 2. We will now write down the stamps for each of the elements and then combine them together.

Current source i_s			v_1	RHS	
		Node 1		i_s	

Conductance G_1			v_1	RHS	
		Node 1	G_1		

Conductance G_2			v_1	v_2	RHS	
		Node 1	G_2	$-G_2$		
		Node 2	$-G_2$	G_2		

Conductance G_3			V_2	v_4	RHS	
		Node 2	G_3	$-G_3$		
		Node 4	$-G_3$	G_3		

Conductance G_4			v_2	RHS	
		Node 2	G_4		

Operational amplifier			i_{out}	RHS	
		Node 4	1		

Combining these stamps we have:

	v_1	v_2	v_4	i_{out}	RHS
Node 1	G_1+G_2	$-G_2$			i_s
Node 2	$-G_2$	$G_2+G_3+G_4$	$-G_3$		
Node 4		$-G_3$	G_3	1	

This corresponds to the matrix equation:

$$\begin{bmatrix} G_1 + G_2 & G_2 & 0 & 0 \\ -G_2 & G_1 + G_3 + G_4 & -G_3 & 0 \\ 0 & -G_3 & G_3 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_4 \\ i_{out} \end{bmatrix} = \begin{bmatrix} i_s \\ 0 \\ 0 \end{bmatrix}$$

If we considered the two inputs to the operational amplifier as two different nodes (say Node 2 and Node 2'), both at the voltage v_2, we would have split the second nodal equation into two, giving:

	v_1	v_2	v_4	i_{out}	RHS
Node 1	G_1+G_2	$-G_2$			i_s
Node 2	$-G_2$	G_2			
Node 2'		G_3+G_4	$-G_3$		
Node 4		$-G_3$	G_3	1	

3.3.3 Modelling of non-linear elements – diodes and transistors

There have been various attempts to model non-linear elements such as diodes, and the early models were designed for clarity of insight and ease of use in manual analysis and design. We will look at the piece-wise linear model of a diode, for large signals as well as small signals, as an example of these techniques. Later, other models (known as companion models) have evolved facilitating computer modelling of circuits containing non-linear elements. We will develop the companion model of a diode and use it to obtain the model of a junction transistor.

Piece-wise linear model of a diode

The figure illustrates the characteristics of a real diode and of an ideal diode.

Real diode   Ideal diode
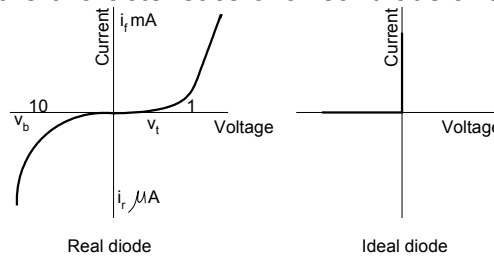
Note that in the case of the real diode, the positive and negative current and voltage axes are scaled differently, to allow significant characteristics to be meaningfully displayed. v_t is the threshold voltage while v_b is the breakdown voltage. It is clear that we can construct an approximate characteristic of a real diode using a combination of ideal diodes. We will consider four regions:

       The conducting region – forward biased, at voltages above the threshold
       The region below the threshold voltage
       The reverse biased region, before breakdown
       The breakdown region

The figure shows the reconstructed characteristics, and how two ideal diodes are connected to obtain it. [The affect of r_r on the forward biased region below the threshold is insignificant.]

Piece-wise linear
diode characteristic

Piece-wise linear model of a real
diode using two ideal diodes
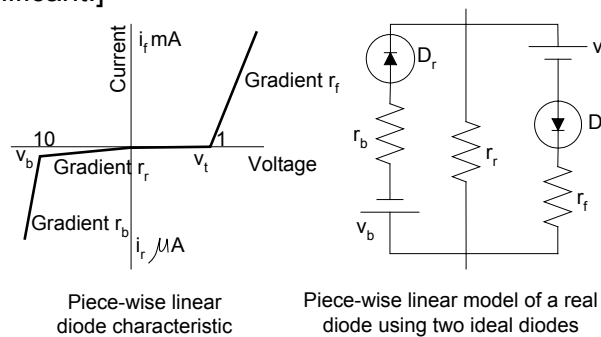
The dc bias in each direction provides the offsets while the resistances determine the gradients.

The small signal model

The above model cannot accurately model the small-signal behaviour of a diode. By the small-signal behaviour we mean the changes in conduction for small changes in the applied voltage, in the forward bias or normal operating condition. This is especially true for operation in the non-linear region (close to the knee).

We will illustrate the development of a small signal model using a typical example. Consider a diode with the characteristics as shown, connected across a dc source of 1V, through a resistor of 10 Ω:
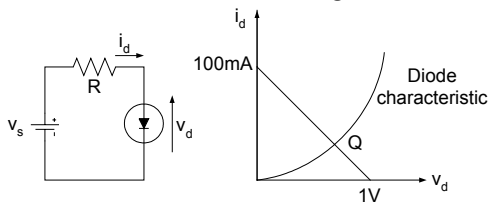
We may obtain the operating point Q by drawing the load line (connecting the two extreme points corresponding to zero current and zero voltage across the diode). The load line and the characteristic curve cross each other at the operating point Q, in this case at approximately (0.65V, 35 mA). Let us now add a small ac excitation (of, say, peak magnitude 0.1V) and observe the operation of the diode:
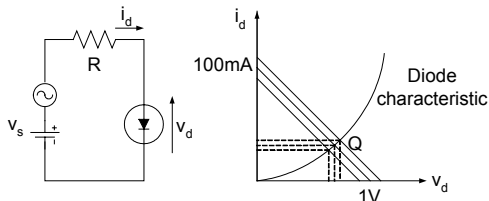
We can again construct the extreme operating points, corresponding to the positive and negative peaks of the ac excitation voltage. We notice that the current through the diode varies between approximately 32 mA and 38 mA while the voltage across it varies between approximately 0.63 V and 0.67 V. This behaviour may conveniently be represented by a resistance equal in value to the gradient of the characteristic curve at the operating point Q, $\dfrac{\Delta v}{\Delta i}\Big|_Q$ [In this case, approximately 0.04 / 0.006 = 6.67Ω.] The value of the forward resistance r_f computed as above would vary with the operating point Q, set by the dc bias voltage. We may compute the reverse resistance r_r and the breakdown resistance r_b in a similar manner.

Companion model of a diode

The graphical methods described above have been used for a long time, but principally in manual computation, usually involving only one non-linear element. It is difficult to implement when more than one such element exists in a circuit. The obvious way out, which also is suitable for automatic computation, is to

devise an iterative algorithm that, hopefully, will converge to the desired solution. Newton's method is such an algorithm.

Let us consider the non-ideal (or real) diode described by the equation:

$$i_d = I_s(e^{\lambda v_d} - 1)$$
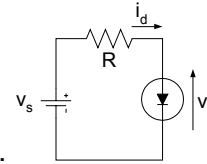
connected to a voltage source through a resistor as before.

We are interested in determining the exact operating point corresponding to (v_d, i_d), starting with a first guess $(v_d{}^0, i_d{}^0)$ through an iterative process. As a first approximation, we will use only the first term of a Taylor series expansion of the expression for the diode current i_d, about the starting point:

$$i_d \approx i_d{}^0 + \left[\frac{\partial i_d}{\partial v_d}\right]_{v_d = v_d{}^0} \left(v_d - v_d{}^0\right)$$

This is a straight line passing through $(v_d{}^0, i_d{}^0)$, tangential to the diode characteristic. However, we do not still know v_d, so the best we can do is to get to the intersection of this line with the load line, and consider the value of v at that point as the next updated value, $v_d{}^1$. We are now in a position to obtain $i_d{}^1$, by projecting $v_d{}^1$ to the characteristic curve.
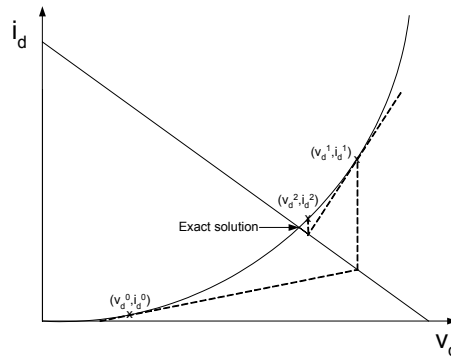
Now, we can start again from the newly determined approximate solution $(v_d{}^1, i_d{}^1)$ until we reach the solution to the desired degree of accuracy.
[The tangents (linearised characteristic curves, at the points where they are evaluated) may be calculated using the relationship $i_d = I_s(e^{\lambda v_d} - 1)$. We can write

$$\left[\frac{\partial i_d}{\partial v_d}\right]_{v_d = v_d{}^0} = \left[I_s \lambda e^{\lambda v_d{}^0}\right]$$

$$\therefore i_d \approx I_s(e^{\lambda v_d{}^0} - 1) + I_s \lambda e^{\lambda v_d{}^0}(v_d - v_d{}^0) \approx I_s(e^{\lambda v_d{}^0} - 1) + I_s \lambda e^{\lambda v_d{}^0} \Delta v_d{}^0$$

where

$$\Delta v_d{}^0 = v_d{}^1 - v_d{}^0$$

Solving this equation along with that of the load line will enable us to solve for $v_d{}^1$.]

We can generalise this expression for the mth iteration as:

$$i_d^{m+1} \approx i_d^{m} + \left[\frac{\partial i_d}{\partial v_d}\right]_{v_d = v_d^{m}} \left(v_d^{m+1} - v_d^{m}\right)$$

Let us now look at this equation in an abstract manner, without reference to its origin. If we denote $\left[\frac{\partial i_d}{\partial v_d}\right]_{v_d = v_d^{m}}$ by G_d^{m}, we can write:

$$i_d^{m+1} \approx i_d^{m} + G_d^{m}(v_d^{m+1} - v_d^{m}) = (i_d^{m} - G_d^{m}v_d^{m}) + G_d^{m}v_d^{m+1}$$

This corresponds to the following circuit model:
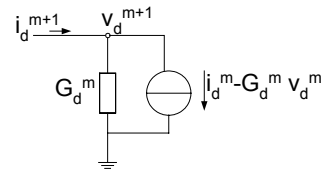
You have to remember that this is only an electrical simulation of the iterative equation that we derived for the solution of the non-linear problem. However, we can use this as a model for the diode. We call it a companion model.

The companion model may be connected in place of the diode in the original circuit.
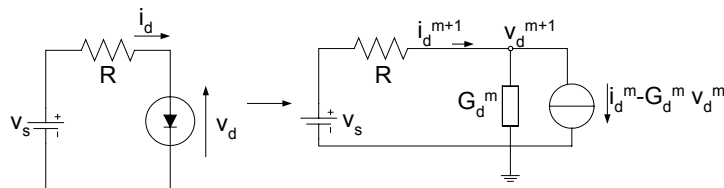
We start the solution process by making a first guess of the diode voltage v_d^{0}. At any stage, the algorithm is as follows:

Given v_d^{m}, calculate i_d^{m} using the (known) relationship $i_d^{m} = I_s(e^{\lambda v_d^{m}} - 1)$

Calculate G_d^{m} from the relationship $G_d^{m} = \left[\frac{\partial i_d}{\partial v_d}\right]_{v_d = v_d^{m}} = \lambda I_s (e^{\lambda v_d^{m}} - 1)$

Using the (numerical) values of the current source $(i_d^{m} - G_d^{m} v_d^{m})$ and the conductance (G_d^{m}), we can solve the companion network by nodal analysis for v_d^{m+1}. We now repeat the process until we reach a solution to an acceptable degree of accuracy [usually determined by the difference between v_d^{m} and v_d^{m+1}]

The Ebers-Moll model of a transistor

We will now attempt to derive a companion model for a transistor, using that of the diode as a starting point. The simplest transistor model is the Ebers-Moll model described below. There are two possible configurations for a Bipolar Transistor. The NPN and the PNP configurations are assembled with two PN junctions back to back, the type depending on whether the common junction is an N-region or a P-region. We will consider an NPN transistor in the following discussion.

It would be incorrect to treat a transistor as consisting of just two diodes connected back to back, for its operation depends on the junction being formed on a single crystal. In addition, the separation between the two junctions has to be rather small for it to work satisfactorily. The figure shows an NPN transistor.
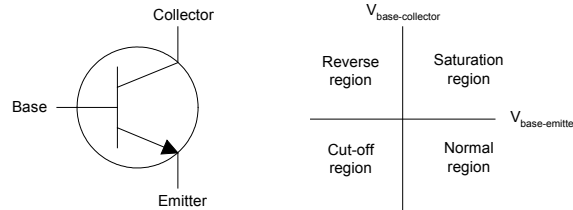
We need to take its actual mode of operation into consideration in deriving a model to represent the transistor. The base-emitter junction is forward biased in normal operation while the base- collector junction is reverse biased. This is illustrated in the chart shown above.

In the normal (or forward) operating region, the base-emitter junction is forward biased so that the emitter injects a steady stream of charge carriers into the base region. As the base-collector junction is reverse biased, only a very small (intrinsic) current will flow through this junction. However, due to the narrowness of the base region, most of the charge carriers injected by the emitter find their way to the collector. This means that the collector current is almost independent of the base-collector voltage, as long as the junction remains reverse biased. On the other hand, the emitter current (and hence the collector current) can be readily controlled by the base-emitter voltage. We can model this phenomenon as a current controlled current source.
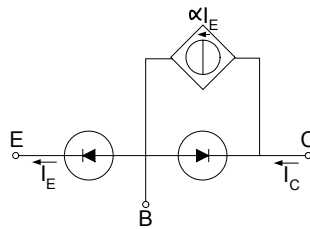
We can now replace the diodes with their companion models to obtain the companion model of the transistor.

There are many variations of the Ebers-Moll model. A simplified model will omit the reverse biased base-collector diode, as its contribution is comparatively small. On the other hand, a more comprehensive model will include another CCCS to take account of its contribution. These variations are shown below.
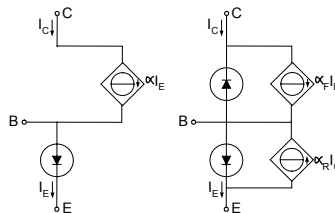
More accurate (and more elaborate) Ebers-Moll models that take account of configurational series resistances and depletion capacitances are in use.

Example
We will consider the analysis of a simple transistor amplifier as an example.

Quiescent state:
set $i_s=0$

$i_s=0$

$i_s=0$
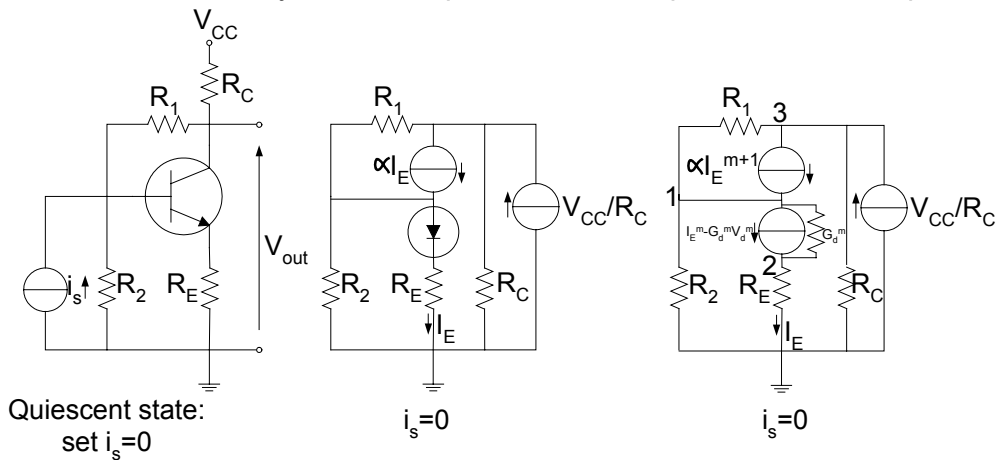
The figure shows a simple transistor amplifier. We will consider its operation at the quiescent state, with $i_s = 0$. The second figure shows the transistor replaced by a simplified Ebers-Moll model while in the last figure; its companion model replaces the diode. We can now write down the equations pertaining to the operation of the circuit by first writing down the stamps of each element.

Current source αI_E^{m+1}				
		v_1	v_3	RHS
	Node 1			αI_E^{m+1}
	Node 3			$- \alpha I_E^{m+1}$

Current source $I_E^m - G_d^m V_d^m$				
		v_1	v_2	RHS
	Node 1			$-(I_E^m - G_d^m V_d^m)$
	Node 2			$(I_E^m - G_d^m V_d^m)$

Current source $V_{CC}/R_C = V_{CC} G_C$		
	v_3	RHS
Node 3		$V_{CC} G_C$

Resistance R_1 = Conductance G_1			
	v_1	v_3	RHS
Node 1	G_1	$-G_1$	
Node 3	$- G_1$	G_1	

Resistance R_2 = Conductance G_2		
	v_1	RHS
Node 1	G_2	

Resistance R_E = Conductance G_E		
	v_2	RHS
Node 2	G_E	

Resistance R_C = Conductance G_C		
	v_3	RHS
Node 3	G_C	

Conductance G_d^m			
	v_1	v_2	RHS
Node 1	G_d^m	$- G_d^m$	
Node 2	$- G_d^m$	G_d^m	

We can now collect these stamps together to form the system equations:

$$\begin{bmatrix} G_1 + G_2 + G_d{}^m & -G_d{}^m & -G_1 \\ -G_d{}^m & G_E + G_d{}^m & 0 \\ -G_1 & 0 & G_1 + G_C \end{bmatrix} \begin{bmatrix} V_1{}^{m+1} \\ V_2{}^{m+1} \\ V_3{}^{m+1} \end{bmatrix} = \begin{bmatrix} \alpha I_E{}^{m+1} - (I_E{}^m - G_d{}^m V_d{}^m) \\ (I_E{}^m - G_d{}^m V_d{}^m) \\ -\alpha I_E{}^{m+1} + V_{CC} G_C \end{bmatrix}$$

In order to solve these equations we need to invoke the following relationships:

$$V_d{}^m = V_1{}^m - V_2{}^m$$
$$I_E{}^{m+1} = I_E{}^m - G_d{}^m V_d{}^m + G_d{}^m V_d{}^{m+1}$$

Substitution of these and rearrangement will lead to:

$$\begin{bmatrix} G_1 + G_2 + (1-\alpha) G_d{}^m & -(1-\alpha) G_d{}^m & -G_1 \\ -G_d{}^m & G_E + G_d{}^m & 0 \\ -G_1 + \alpha G_d{}^m & -\alpha G_d{}^m & G_1 + G_C \end{bmatrix} \begin{bmatrix} V_1{}^{m+1} \\ V_2{}^{m+1} \\ V_3{}^{m+1} \end{bmatrix} = \begin{bmatrix} (\alpha - 1)(I_E{}^m - G_d{}^m (V_1{}^m - V_2{}^m)) \\ I_E{}^m - G_d{}^m (V_1{}^m - V_2{}^m) \\ -\alpha(I_E{}^m - G_d{}^m (V_1{}^m - V_2{}^m)) + V_{CC} G_C \end{bmatrix}$$

To start the solution, we may guess a value for $(V_1{}^0 - V_2{}^0)$ and use this to calculate $I_E{}^0$ using the relationship

$$i_E{}^0 = I_s \left(e^{\lambda(v_1{}^0 - V_2{}^0)} - 1 \right)$$

We can then solve for $[V_1{}^1 \ V_2{}^1 \ V_3{}^1]^T$, and then continue until convergence.

Having obtained the quiescent operating point, we can introduce any desired signal i_s (which we originally set to zero) and study the (transient) behaviour of the amplifier.

The Gummel-Poon Model
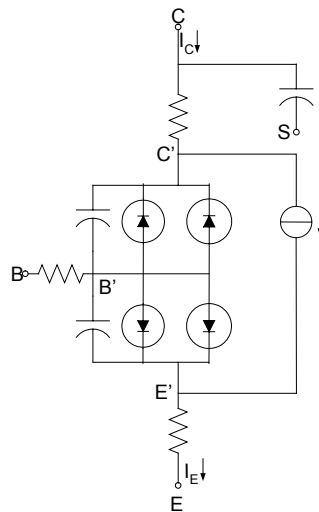
The figure shows a simplified version of the Gummel-Poon model that closely resembles the physical layout of the real transistor. It accounts for the series

resistances, depletion capacitances, the leakage paths and the capacitance to the substrate layer. More accurate models are in use in the commercial implementations of modelling software. Detailed procedures for the measurement of the parameters have been laid down, making the model to be of real practical use.

It can be seen that simplifying assumptions would lead to the Gummel-Poon model collapsing to the Ebers-Moll model.

3.3.4 Modelling transient behaviour – capacitors and inductors

When considering the automatic formulation of circuit equations, we confined ourselves to the treatment of conductance elements only. We are now in a position to expand this to cover capacitors and inductors.

You will recall that when we derived the companion model of a diode, we wrote down a difference equation describing the (approximate) behaviour of the system and then set up an electrical simulation of this equation. We can follow the same procedure in solving circuits containing capacitors and inductors. It is quite easy to write down the circuit equations containing such elements using the modified nodal analysis technique that was described. However, the equations will no longer be algebraic equations (unlike in the case of circuits containing only resistors), but will be differential equations. Our strategy is to convert the differential equations to (approximate) difference equations and then to set up electrical analogies of these systems of equations.

As before, we will call the resulting model a companion network.
There are a number of alternative approximations available for deriving difference equations from differential equations. We will consider the simplest of these, the backward Euler formula.

Consider a time dependent variable x.

We can write $$\left.\frac{dx}{dt}\right|_{t=t^{n+1}} \approx \frac{x^{n+1} - x^n}{t^{n+1} - t^n}$$

Note that the superscript n refers to the instant of time. (When deriving the companion model for the diode, we used the superscript m to denote the iteration count.) We can now use this to model the behaviour (the voltage – current relationship) of a capacitor:

$$i = C\frac{dv}{dt}$$

Using the backward Euler formula, $i^{n+1} = C\frac{v^{n+1} - v^n}{t^{n+1} - t^n} = \frac{C}{T}v^{n+1} - \frac{C}{T}v^n$

Now we will set up an electrical analogue that simulates this approximate relationship as shown below:
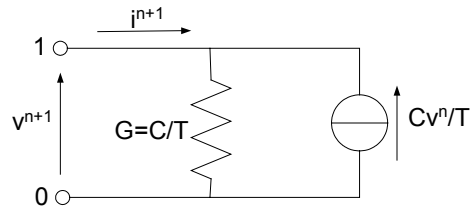
We will now consider a simple RC network connected across a battery and see how this model would operate.
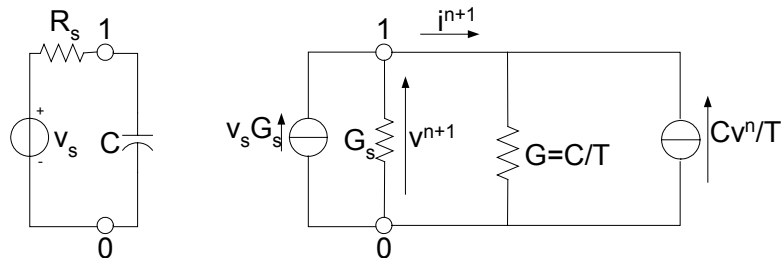
The stamps of the different elements (the equivalent current source i^{n+1}, current source Cv^{n+1}/T), conductance G_s (= $1/R_s$) and conductance C/T would be:

Equivalent current source v_sG_s			
		v^{n+1}	RHS
	Node 1		v_sG_s

Current source Cvn/T			
		v^{n+1}	RHS
	Node 1		Cv^n/T

Conductance Gs			
		v^{n+1}	RHS
	Node 1	G_s	

Conductance G=C/T			
		v^{n+1}	RHS
	Node 1	C/T	

The resulting equation is: $$\left[G_s + \frac{C}{T} \right]\left[v^{n+1} \right] = \left[v_s G_s + \frac{Cv^n}{T} \right]$$

We have a recurrent relationship between v^n (the voltage across the capacitor at time t = nT) and v^{n+1}. We can solve it for all n, knowing v^0.

The case of an inductor may be treated in a similar manner. The defining relationship in this case is:

$$v = L\frac{di}{dt}$$

Using the backward Euler formula

$$v^{n+1} = L\frac{i^{n+1} - i^n}{t^{n+1} - t^n} = \frac{L}{T}i^{n+1} - \frac{C}{T}i^n$$

Earlier, we noted that a gyrator could be used to transform a capacitor into an inductor. Using this, we may model the inductor by a capacitor connected through a gyrator as shown below: This of course gives rise to an additional node.
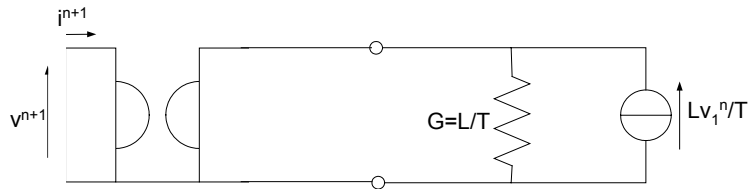
3.3.5 Dynamic behaviour of non-linear circuits

We have now studied, separately, the modelling of non-linear circuit elements and the modelling of time dependent circuits (dynamic behaviour of circuits), and are in a position to consider the dynamic behaviour of non-linear circuits. This would involve both the techniques we used, that of iteration and of discretisation, and thus the use of two superscripts, one each to denote the iteration count and the time instant.

We will consider an example chosen from two of the examples used before. Consider the transistor amplifier shown, where a capacitor C_E is connected across the emitter resistor R_E. We first replace this capacitor with its companion network, where the superscript n refers to the time ($t^n = nT$). In the next figure, the npn transistor is replaced by its companion network, m representing the iteration count. Double superscripts are used to indicate dependence on both the time instant and the iteration.

In solving the system equations, we start with the (known) initial conditions (at n = 0) and an initial first guess of $v_d^{0,0}$ ($= v_1^{0,0} - v_2^{0,0}$). The equations are solved iteratively until convergence, and then we go to the next time step at n=1. At this point, we can use the final values obtained at time step 0 as the initial guess for $v_d^{0,1}$, and continue as before.

The stamps for each element of the circuit are shown in the table.

V_{CC}

R_1 R_C

V_{out}

i_s R_2 R_E C_E

V_{CC}

R_1 R_C

V_{out}

i_s^n R_2 R_E C_E/T $C_E v^n/T$

G_1 3

$\alpha I_E^{n,m+1}$

1

$I_E^{n,m} - G_d^{n,m} V_d^{n,m}$ $G_d^{n,m}$

2

i_s^n G_2 G_E C_E/T $C_E v^n/T$ G_C

$V_{CC} G_C$

Current source I_s^n			
		$v_1^{n,m+1}$	RHS
	Node 1		I_s^n

Current source $\alpha I_E^{n,m+1}$				
		$v_1^{n,m+1}$	$v_3^{n,m+1}$	RHS
	Node 1			$\alpha I_E^{n,m+1}$
	Node 3			$-\alpha I_E^{n,m+1}$

Current source $\alpha I_E^{n,m} - G_d^{n,m} v_d^{n,m}$ $=\alpha I_E^{n,m} - G_d^{n,m} (v_1^{n,m} - v_2^{n,m})$				
		$v_1^{n,m+1}$	$v_2^{n,m+1}$	RHS
	Node 1			$-[\alpha I_E^{n,m} - G_d^{n,m} (v_1^{n,m} - v_2^{n,m})]$
	Node 2			$[\alpha I_E^{n,m} - G_d^{n,m} (v_1^{n,m} - v_2^{n,m})]$

Current source $C_E v^n/T$			
		$v_2^{n,m+1}$	RHS
	Node 2		$C_E v^n/T$

Current source $v_{CC} G_C$			
		$v_3^{n,m+1}$	RHS
	Node 3		$V_{CC} G_C$

Conductance $G_1 = 1/R_1$				
		$v_1^{n,m+1}$	$v_3^{n,m+1}$	RHS
	Node 1	G_1	$-G_1$	
	Node 3	$-G_1$	G_1	

Conductance $G_2 = 1/R_2$			
		$v_1^{n,m+1}$	RHS
	Node 1	G_2	

Conductance $G_d^{n,m}$				
		$v_1^{n,m+1}$	$v_2^{n,m+1}$	RHS
	Node 1	$G_d^{n,m}$	$-G_d^{n,m}$	
	Node 2	$-G_d^{n,m}$	$G_d^{n,m}$	

Conductance C_E/T			
		$v_2^{n,m+1}$	RHS
	Node 2	C_E/T	

Conductance G_C			
		$v_3^{n,m+1}$	RHS
	Node 3	G_C	

Conductance G_E			
		$v_2^{n,m+1}$	RHS
	Node 2	G_E	

The resulting equations are:

$$
\begin{bmatrix}
G_1 + G_2 + G_d^{\,n,m} & -G_d^{\,n,m}. & -G_1 \\
-G_d^{\,n,m} & G_E + G_d^{\,n,m} + C_E/T & 0 \\
-G_1 & 0 & G_1 + G_C..
\end{bmatrix}
\begin{bmatrix}
v_1^{\,n,m+1`} \\
v_2^{\,n.m+1} \\
v_3^{\,n.m+1}
\end{bmatrix}
=
$$

$$
\begin{bmatrix}
i_s^{\,n} + \alpha\, i_E^{\,n.m+1} - \left[\alpha\, i_E^{\,n,m} - G_d^{\,n.m}(v_1^{\,n,m`} - v_2^{\,n.m}) \right] \\
\left[\alpha\, i_E^{\,n,m} - G_d^{\,n.m}(v_1^{\,n,m`} - v_2^{\,n.m}) \right] + C_E v^n T \\
-\alpha\, i_E^{\,n.m+1} + V_{CC} G_c
\end{bmatrix}
$$

As before, we need to use the relationship

$$
I_E^{\,n,m+1} = I_E^{\,n,m} - G_d^{\,n,m}(v_1^{\,n,m} - v_2^{\,n,m}) + G_d^{\,n,m}(v_1^{\,n,m+1} - v_2^{\,n,m+1})
$$

in order to eliminate $I_E^{\,n,m+1}$ from these equations. Making the above substitution and rearranging, we get:

$$
\begin{bmatrix}
G_1 + G_2 + (1-\alpha)G_d^{\,n,m} & -(1-\alpha)G_d^{\,n,m}. & -G_1 \\
-G_d^{\,n,m} & G_E + G_d^{\,n,m} + C_E/T & 0 \\
-G_1 + \alpha G_d^{\,n,m} & -\alpha G_d^{\,n,m} & G_1 + G_C..
\end{bmatrix}
\begin{bmatrix}
v_1^{\,n,m+1`} \\
v_2^{\,n.m+1} \\
v_3^{\,n.m+1}
\end{bmatrix}
=
$$

$$
\begin{bmatrix}
i_s^{\,n} + \left[(1-\alpha)G_d^{\,n.m}(v_1^{\,n,m`} - v_2^{\,n.m}) \right] \\
\left[\alpha\, i_E^{\,n,m} - G_d^{\,n.m}(v_1^{\,n,m`} - v_2^{\,n.m}) \right] + C_E v^n T \\
-\alpha\, i_E^{\,n.m} + \alpha G_d^{\,n,m}(v_1^{\,n,m} - v_2^{\,n,m}) + V_{CC} G_c
\end{bmatrix}
$$

The complete solution procedure is as follows:

1. Start with the known initial conditions, at n = 0 (time t = 0)
2. Start at iteration count zero, with m = 0
3. Iterate until system converges, using the exponential diode model as before
4. Go to the next time step, n = n+1
5. Go back to step 2

We may use this technique of combining different approaches (Newton-like methods to solve non-linear problems, Euler-like methods to computer dynamic variations, modified nodal analysis to formulate the system equations) to attack any complex circuit problem. However, manual solution becomes difficult as the complexity increases, and methods with better stability properties may be needed for each of the first two algorithms.

Newton's method is sometimes modified by using either accelerators (for faster convergence) or retarders (for improved stability) while the Simpson's formula is sometimes used in place of Euler's.

3.3.6 Spice

Spice is a comprehensive circuit-modelling package, and is the most popular of such software packages. It was initially developed at the Department of Electrical Engineering and Computer Science, University of California at Berkley in the 1970s, and stands for Simulation Program Integrated Circuits Especially. Many versions of Spice are now available, from the University of California at Berkley (spice 3) as well as from commercial vendors (PSpice from Orcad / Cadence, Hspice from Avanti etc.)

Even though there is constant development and upgrading, the general principles of circuit simulation software are those that we studied in the previous sections, namely,

> Circuit definition using a list (circuit capture from a schematic is now commonly available)
> Equation formulation using modified nodal analysis
> Non-linear element simulation by iteration
> Discretisation in the time domain
> Use of efficient algorithms for the solution of system equations, including solution of sparse systems

However, these are mostly transparent to the user. Sophisticated analysis modes are now available including

> dc,
> ac small signal,
> transient,
> pole-zero,
> distortion,
> sensitivity,
> noise,

and   temperature

3.3.7 Tellegen's theorem, sensitivity and robust design

Sensitivity calculations are important for economic circuit design. If the overall performance of a circuit is very much dependent on the value of a particular component, it is obvious that tight control has to be exercised over its value and we will be justified in using a low-tolerance, high-value component. On the other hand, if the circuit has a low sensitivity to the value of a component, it is possible to use a cheap component for that application.

Analytically, the obvious method of calculating the sensitivity (of an output variable) to changes in parameter values would be through its partial derivative. However, this is possible only in the case of linear circuits, where a closed form solution is available. In the majority of cases that we encounter in practice, it would be necessary to carry out a simulation study, after imposing a small

perturbation on the value of each parameter. This is a rather tiresome process, and any innovation to circumscribe it is welcome.

It is fortunate that such a method does actually exist through the application of Tellegen's theorem, though it is difficult to visualise the connection between sensitivity analysis and Tellegen's theorem at first glance.

Tellegen's theorem

Tellegen's theorem may be considered in two stages, the simple theorem and the extended theorem. Let us consider the simple theorem, which is almost a restatement of Kirchhoff's Current Law, first

Consider an interconnected network of n nodes and m branches. The figure shows the jth node, connected to nodes 1 to n (any self-loops connecting the jth node to itself, and multiple connections have been omitted for convenience, but their treatment is trivial)
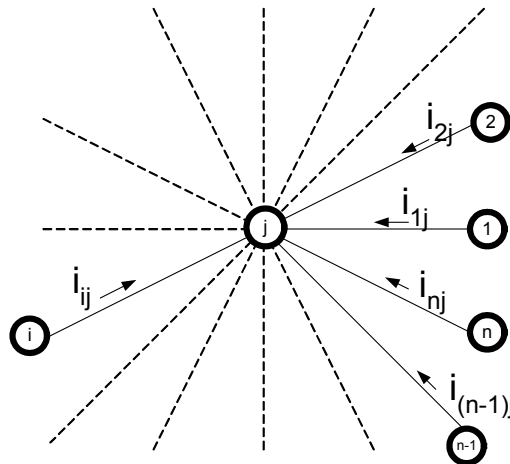
We can write down an expression for the summation of the voltage – current product of all branches incident at node j as:

$$\sum\nolimits_{j} = \sum_{i=1}^{n} v_{ij}\, i_{ij} = \sum_{i=1}^{n} (v_i - v_j)\, i_{ij}$$

The sum of the voltage – current products over all the branches of the circuit may be written down in terms of $\sum\nolimits_{j}$ by recognising that if this were to be summed over all j from 1 to n, it would include each voltage – current product term twice over.

$$\sum_{1}^{m} v_{ij} i_{ij} = \frac{1}{2} \sum_{j=1}^{n} \left[\sum\nolimits_{j} \right] = \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{n} (v_i - v_j) i_{ij}$$

[Note: Double subscripts refer to branch voltages and currents while single subscripts refer to node voltages]

Noting that $i_{ij} = -i_{ji}$, we can write:

$$\sum_{1}^{m} v_{ij} i_{ij} = \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{n} (v_i - v_j) i_{ij} = \frac{1}{2} \left[\sum_{i=1}^{n} v_i \sum_{j=1}^{n} i_{ij} + \sum_{j=1}^{n} v_j \sum_{i=1}^{n} i_{ji} \right]$$

We note that the second summation in each of the two terms within the square brackets is identically zero, by KCL. Thus, the total sum (of the voltage – current product over all branches of the circuit) is zero.

We will now effect a slight change of notation (for brevity) by using single subscripts to denote branch voltages and currents, and write:

$$\sum_{k=1}^{m} v_k i_k = 0$$

This is Tellegen's (simple) theorem, which states that the sum of the product of branch voltages and currents in a circuit is always zero,

Tellegen's (extended) theorem defines a relationship between the branch currents and voltages of two networks possessing the same structure. Let us consider two such networks $N \, and \, \hat{N}$ as shown:
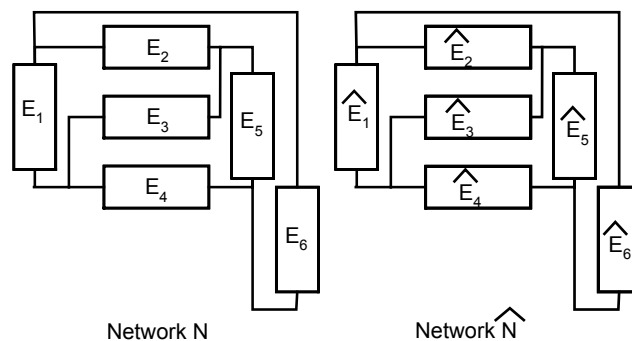
Network N   Network $\hat{N}$

The networks have the same structure (topology), but the elements in the two networks can be totally different. For example, element E_1 of network N can be a resistor while the corresponding element $\hat{E}_1 \, of \, \hat{N}$ may be a capacitor (or even a current or voltage source.) Even the constraint on the two networks being of the same structure is not very restrictive, for elements are allowed to be short circuits and open circuits. However, the currents and voltages of corresponding elements should be marked consistently, that is, the directions on one network should be the same as on the other. This is illustrated in the figure:

Element in Network N   Element in Network $\hat{N}$
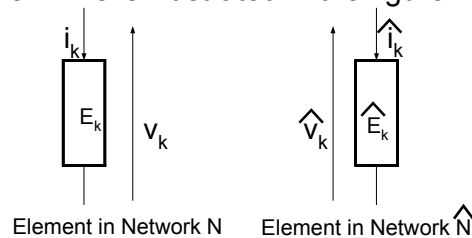
Let us look again at the expression for the sum of voltage – current products we derived earlier:

$$\sum_{1}^{m} v_{ij} i_{ij} = \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{n} (v_i - v_j) i_{ij} = \frac{1}{2} \left[\sum_{i=1}^{n} v_i \sum_{j=1}^{n} i_{ij} + \sum_{j=1}^{n} v_j \sum_{i=1}^{n} i_{ji} \right]$$

Note that the sum of the currents at a node is still zero, even if the voltage reference is changed, as long as the number of nodes remains unchanged. This leads us directly to Tellegen's (extended) theorem relating the currents and voltages of the two networks $N \, and \, \hat{N}$:

$$\sum_{k=1}^{m} v_k \hat{i}_k = 0, \qquad\qquad \sum_{k=1}^{m} \hat{v}_k i_k = 0$$

Unlike Tellegen's (simple) theorem (which can be intuitively justified as a statement of the conservation instantaneous power), this is rather unexpected result.

Sensitivity analysis using Tellegen's theorem

We will consider a simple resistive network driven by an ideal voltage source, and where we are interested in the sensitivity of the voltage across one pair of nodes, to changes in the values of the resistors in the network.

Without loss of generality, we will call the branch containing the source, branch 1 and the branch (with the hypothetical infinite resistance), the voltage across which interest us, branch 2. We will assume that there are (m-2) other branches within the network of resistances R_1, R_2, ..., R_{m-2}
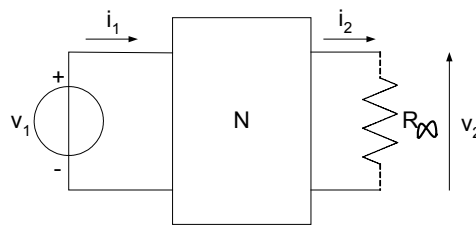
We will denote by *N* the complete network, including the source and output branches, and assume that there is another network, of the same structure, denoted by $\hat{N}$. We shall now invoke Tellegen's (extended) theorem:

$$\sum_{k} v_k \hat{i}_k = 0, \qquad\qquad \sum_{k} \hat{v}_k i_k = 0$$

Now, let the resistance R_i in the network N change its value by a small amount δR_i, resulting in changes in all the voltages and currents in the network, denoted by $\delta v_k, \delta i_k$. We will further assume that all the other resistances remain unchanged. Invoking Tellegen's theorem again, we have:

$$\sum_{k=1}^{m} (v_k + \delta v_k)\hat{i}_k = 0 \qquad \sum_{k=1}^{m} \hat{v}_k (i_k + \delta i_k) = 0$$

From these two sets of equations, we have:

$$\sum_{k=1}^{m} \delta v_k \hat{i}_k = 0 \qquad \sum_{k=1}^{m} \hat{v}_k \delta i_k = 0$$

$$\therefore \sum_{k=1}^{m} (\delta v_k \hat{i}_k - \hat{v}_k \delta i_k) = 0$$

$$ie., (\delta v_1 \hat{i}_1 - \hat{v}_1 \delta i_1) + (\delta v_2 \hat{i}_2 - \hat{v}_2 \delta i_2) + (\delta v_3 \hat{i}_3 - \hat{v}_3 \delta i_3) + . \ . \ . + (\delta v_m \hat{i}_m - \hat{v}_m \delta i_m) = 0$$

We will now consider the conditions under which some of the terms of the above expression can be made identically equal to zero. We will consider them in three groups:

Consider the first term, $(\delta v_1 \hat{i}_1 - \hat{v}_1 \delta i_1) = 0$

As v_1 does not change, δv_1 is zero. We can make the first term identically zero by making $\hat{v}_1$ zero.

Now consider the other terms, other than the second (output branch) and the ith term (corresponding to the branch where the resistance has changed):

$$(\delta v_j \hat{i}_j - \hat{v}_j \delta i_j) = 0, \ j \neq 1, 2, i$$

We have:

$$\delta v_j = \frac{\partial v_j}{\partial R_j}\delta R_j + \frac{\partial v_j}{\partial i_j}\delta i_j$$

$$As \ \delta R_j = 0 \ for \ j \neq i,$$

$$\delta v_j = \frac{\partial v_j}{\partial i_j}\delta i_j = R_j \delta i_j$$

and
$$\hat{v}_j = \hat{R}_j \hat{i}_j$$

Therefore $(\delta v_j \hat{i}_j - \hat{v}_j \delta i_j) = (R_j \delta i_j)\hat{i}_j - (\hat{R}_j\hat{i}_j)\delta i_j = (R_j - \hat{R}_j)\delta i_j \hat{i}_j$

This can be made identically zero by making $\hat{R}_j = R_j$

With these assumptions ($\hat{v}_1 = 0$, $\hat{R}_j = R_j \ for \ j \neq i$), the equation becomes:

$$(\delta v_2 \hat{i}_2 - \hat{v}_2 \delta i_2) + (\delta v_i \hat{i}_i - \hat{v}_i \delta i_i) = 0$$

As the original network N has branch 2 on open circuit, δi_2 is zero. Also,

$$\delta v_i = \frac{\partial v_i}{\partial R_i}\delta R_j + \frac{\partial v_i}{\partial i_i}\delta i_i = i_i \delta R_i + R_i \delta i_i$$

Substitution of these values yields:

$$\delta v_2 \,\hat{i}_2 + ((i_i \,\delta R_i + R_i \,\delta i_i)\hat{i}_i - \hat{v}_i \,\delta i_i) = 0$$

$$\delta v_2 \,\hat{i}_2 + i_i \,\hat{i}_i \,\delta R_i + R_i \,\delta i_i \,\hat{i}_i - \hat{v}_i \,\delta i_i = 0$$

Substituting $R_i \,\hat{i}_i \;\; for \;\; \hat{v}_i$ [by setting $\hat{R}_i = R_i$], we get:

$$\delta v_2 \,\hat{i}_2 + i_i \,\hat{i}_i \,\delta R_i + R_i \,\delta i_i \,\hat{i}_i - R_i \,\hat{i}_i \,\delta i_i = 0$$

$$ie., \;\; \delta v_2 \,\hat{i}_2 + i_i \,\hat{i}_i \,\delta R_i = 0$$

Finally, we set $\hat{i}_2 = 1$. This gives us

$$\delta v_2 + i_i \hat{i}_i \,\delta R_i = 0$$

$$\frac{\delta v_2}{\delta R_i} = -i_i \,\hat{i}_i$$

We have obtained a remarkable result. By choosing the network $\hat{N}$ in a particular manner, we have shown that the sensitivity of the output voltage v_2 to changes in each of the resistance values of the network N may be obtained by the solution of just two networks, that of N and $\hat{N}$
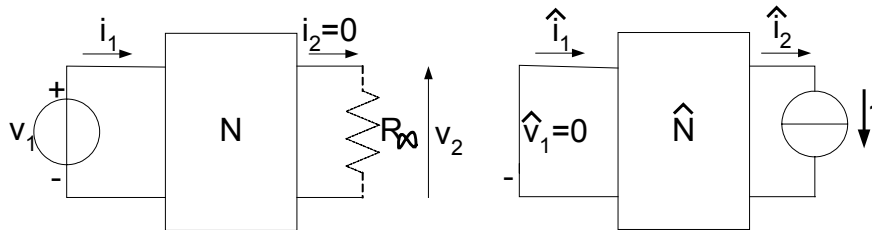
The new network $\hat{N}$ is obtained from the original network N by:

> Replacing a voltage source by a short circuit, (and a current source by an open circuit),
> Replacing the voltage output (open circuit) by a unit current source, (and a current output by a unit voltage source),
> Keeping all resistors unchanged.

Such a network is known as the adjoint network.

By a similar reasoning, we can also show that the sensitivities in terms of the conductances G_i are given by
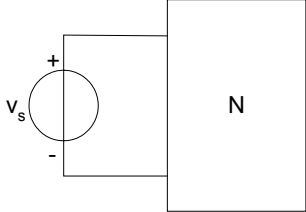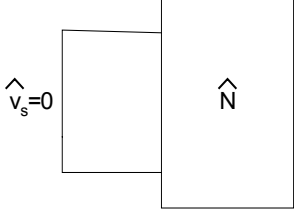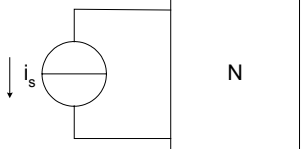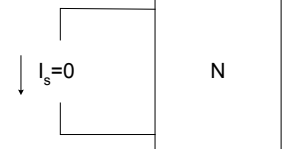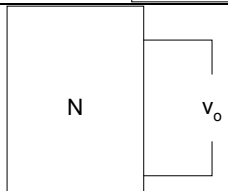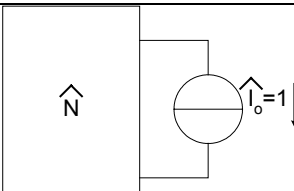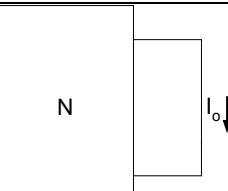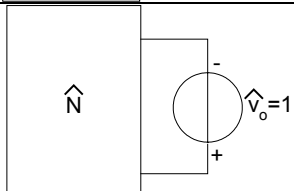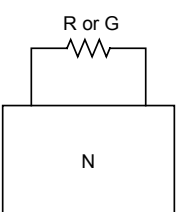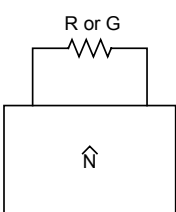
$$\frac{\delta v_2}{\delta G_i} = v_i \,\hat{v}_i$$

In the limit, these become the partial derivatives, and we have the following relationships [including the current sensitivities]

$$\frac{\partial v_2}{\partial R_i} = -i_i\,\hat{i}_i, \qquad \frac{\partial v_2}{\partial G_i} = v_i\,\hat{v}_i$$

$$\frac{\partial i_2}{\partial R_i} = -i_i\,\hat{i}_i, \qquad \frac{\partial i_2}{\partial G_i} = v_i\,\hat{v}_i$$

The following figure shows the definition of the adjoint network for different types of sources and terminations, for networks containing only resistive/conductive elements. We will reserve the word adjoint network for networks conforming to this definition only.

	Network model	Adjoint network model
Voltage source		
Current source		
Voltage outpu		
Current output		
Resistance / Conductance		

We now need to extend this mechanism to cover other network elements such as capacitors and inductors. Before attempting that, it would help to note that it

could be extended from the consideration of branches to cover two-port (and multi-port) networks.

Consider a two-port network N along with its adjoint network, as shown:
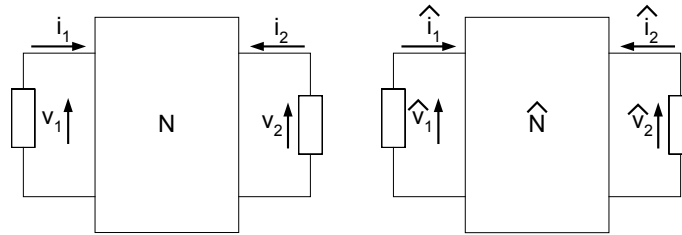
We are interested in finding a description for the adjoint network $\hat{N}$ of N, where N is described by its conductance matrix G, such that:

$$\begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} i_1 \\ i_2 \end{bmatrix}$$

Let $\hat{N}$ be described by its conductance matrix $\hat{G}$ defined by

$$\begin{bmatrix} \hat{g}_{11} & \hat{g}_{12} \\ \hat{g}_{21} & \hat{g}_{22} \end{bmatrix} \begin{bmatrix} \hat{v}_1 \\ \hat{v}_2 \end{bmatrix} = \begin{bmatrix} \hat{i}_1 \\ \sigma \\ \hat{i}_2 \end{bmatrix}$$

Invoking Tellegen's theorem, we have as before:

$$\therefore \sum_{k=1}^{m} (\delta v_k \, \hat{i}_k - \hat{v}_k \, \delta i_k) = 0$$

Assuming that the products corresponding to the internal branches are individually made equal to zero, we have:

$$(\delta v_1 \, \hat{i}_1 - \hat{v}_1 \, \delta i_1) + (\delta v_2 \, \hat{i}_2 - \hat{v}_2 \, \delta i_2) = 0$$

$$ie., \begin{bmatrix} \hat{i}_1 & \hat{i}_2 \end{bmatrix} \begin{bmatrix} \delta v_1 \\ \delta v_2 \end{bmatrix} - \begin{bmatrix} \hat{v}_1 & \hat{v}_2 \end{bmatrix} \begin{bmatrix} \delta i_1 \\ \delta i_2 \end{bmatrix} = 0$$

$$ie., \begin{bmatrix} \hat{I} \end{bmatrix}^T [\delta V] - \begin{bmatrix} \hat{V} \end{bmatrix}^T [\delta I] = 0$$

$$ie., \begin{bmatrix} \hat{I} \end{bmatrix}^T [\delta V] - \begin{bmatrix} \hat{V} \end{bmatrix}^T [G][\delta V] = 0$$

$$ie., \left\{ \begin{bmatrix} \hat{I} \end{bmatrix}^T - \begin{bmatrix} \hat{V} \end{bmatrix}^T [G] \right\} [\delta V] = 0$$

For this condition to be satisfied we have:

$$\left\{ \begin{bmatrix} \hat{I} \end{bmatrix}^T - \begin{bmatrix} \hat{V} \end{bmatrix}^T [G] \right\} = 0$$

$$ie., \begin{bmatrix} \hat{I} \end{bmatrix}^T = \begin{bmatrix} \hat{V} \end{bmatrix}^T [G]$$

$$\therefore \left[\hat{I}\right] = [G]^T \left[\hat{V}\right]$$
$$But \ \left[\hat{I}\right] = \left[\hat{G}\right]\left[\hat{V}\right]$$
$$\therefore \left[\hat{G}\right] = [G]^T$$

If N is described by its conductance matrix G, then its adjoint network $\hat{N}$ is described by a conductance matrix G^T. This result may be extended to multi-port networks.

If we have a network described by the hybrid vector relationship

$$
\begin{bmatrix}
G_{11} & | & \alpha_{12} \\
--- & |------ & \\
\mu_{21} & | & r_{22}
\end{bmatrix}
\begin{bmatrix}
V_1 \\
-- \\
I_2
\end{bmatrix}
=
\begin{bmatrix}
I_1 \\
-- \\
V_2
\end{bmatrix}
$$

instead of the simple relationship $Gv=i$, then, it can be shown that its adjoint network is described by the matrix equation

$$
\begin{bmatrix}
[G_{11}]^T & | & -[\mu_{21}]^T \\
------- & |----- & \\
-[\alpha_{12}]^T & | & [r_{22}]^T
\end{bmatrix}
\begin{bmatrix}
\hat{V}_1 \\
-- \\
\hat{I}_2
\end{bmatrix}
=
\begin{bmatrix}
\hat{I}_1 \\
-- \\
\hat{V}_2
\end{bmatrix}
$$

Application 1: Current controlled current source:
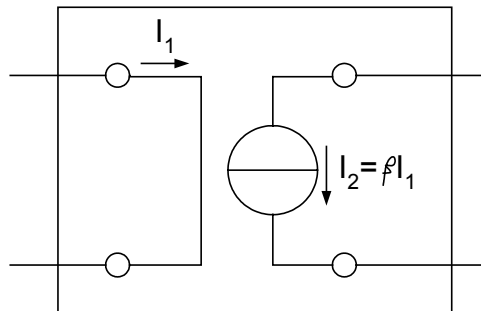
This , considered as a two-port network is represented by the pair of equations:

$$I_2 = \beta I_1$$
$$V_1 = 0$$

This nay be written as:

$$
\begin{bmatrix}
g_{22} & | & \alpha_{21} \\
--- & |---- & \\
\mu_{12} & | & r_{11}
\end{bmatrix}
\begin{bmatrix}
V_2 \\
-- \\
I_1
\end{bmatrix}
=
\begin{bmatrix}
I_2 \\
-- \\
V_1
\end{bmatrix}
$$

where $g_{22} = 0$, $\alpha_{12} = \beta$, $\mu_{12} = 0$, $r_{11} = 0$, giving us

$$\begin{bmatrix} 0 & | & \beta \\ -\,-\,- & | & -\,-\,-\,- \\ 0 & | & 0 \end{bmatrix} \begin{bmatrix} V_2 \\ -\,- \\ I_1 \end{bmatrix} = \begin{bmatrix} I_2 \\ -\,- \\ V_1 \end{bmatrix}$$

The adjoint network is described by:

$$\begin{bmatrix} 0 & | & 0 \\ -\,-\,- & | & -\,-\,-\,- \\ -\beta & | & 0 \end{bmatrix} \begin{bmatrix} \hat{V}_2 \\ -\,- \\ \hat{I}_1 \end{bmatrix} = \begin{bmatrix} \hat{I}_2 \\ -\,- \\ \hat{V}_1 \end{bmatrix}$$

The corresponding network is:
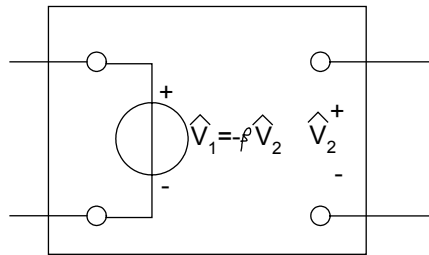
Application 2: Gyrator

A gyrator is defined as:

$$V_1 = -g^{-1} I_2$$
$$I_1 = g V_2$$

or

$$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} -g^{-1} & 0 \\ 0 & g \end{bmatrix} \begin{bmatrix} I_2 \\ V_2 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I_2 \\ V_2 \end{bmatrix}$$

for a gyrator constant g = 1
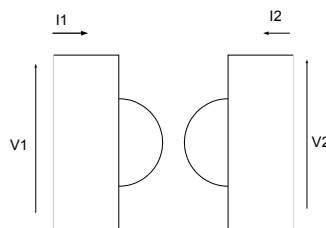
The adjoint network would then be represented by:

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{V}_2 \\ \hat{I}_2 \end{bmatrix} = \begin{bmatrix} \hat{V}_1 \\ \hat{I}_1 \end{bmatrix}$$

This is the same as the original network

Sensitivity in the complex frequency domain

The analysis for resistive networks can be extended to cover sensitivity to changes in impedances and admittances (in the frequency domain), but it should be remembered that the sensitivities obtained are for changes in the impedances (or admittances) and not for changes in capacitance or inductance. Appropriate allowances have to be made to obtain sensitivities in terms of capacitance or inductor values.

Time-dependent sensitivity – Treatment of capacitors etc

The treatment so far was restricted to networks consisting of resistors, and so had no need to consider their variation with time. This was retained in the extension to impedances and admittances in the complex frequency domain, for still the variables retained their constant characteristics, for consideration was limited to one frequency. If we want to consider the time-domain characteristics of circuits containing capacitors, we encounter a different problem, that of time dependence, for both the currents and voltages in such circuits are functions of time.

We will consider the case of a network with a simple time-invariant capacitor connected as shown, and its adjoint network.
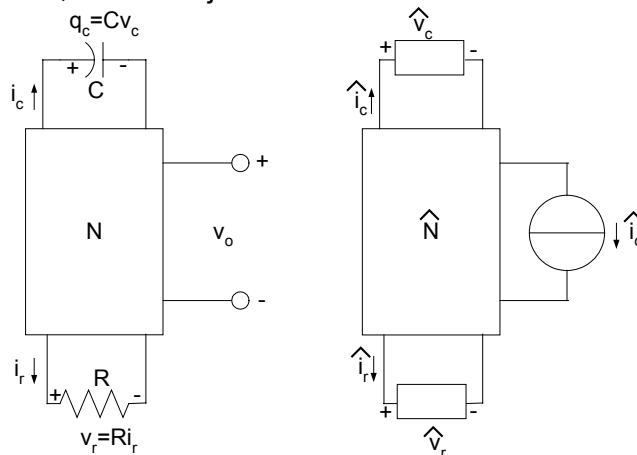
We will consider the time interval from zero to t_f, and attempt to derive the sensitivity of the output voltage v_o at time t_f, to changes in C and R. In the pure resistive case, we defined the adjoint network of a voltage output as a unit current source. Here, we will define it as a unit impulse, applied at time t_f.

$$\hat{i}_o(t) = \delta(t_f - t)$$

Let us now consider the Tellegen's result applied to the two networks:

$$\sum_{k=1}^{m} \left[\delta v_k \hat{i}_k - \delta i_k \hat{v}_k \right] = 0$$

As before, assuming that the terms within N and $\hat{N}$ cancel themselves out, we have:

$$\left[\delta v_r \, \hat{i}_r - \delta i_r \, \hat{v}_r \right] + \left[\delta v_o \, \hat{i}_o - \delta i_o \, \hat{v}_o \right] + \left[\delta v_c \, \hat{i}_c - \delta i_c \, \hat{v}_c \right] = 0$$

Substituting for $\hat{i}_o(t) = \delta(t_f - t)$

and

$$i_c = \frac{dq}{dt} = C \frac{dv_c}{dt} + v_c \frac{dC}{dt}$$

$$\left[((R + \delta R)(i_r + \delta i_r) - Ri_r)\hat{i}_r - \delta i_r (R\hat{i}_r) \right] + \left[\delta v_o \, \delta(t_f - t_0) - (0)\hat{v}_c \right] +$$

$$\left[\delta v_c \hat{i}_c - (\delta C \frac{dv_c}{dt} + \delta v_c \frac{dC}{dt})\hat{v}_c \right] = 0$$

Neglecting second order terms and rearranging,

$$\delta R \, i_r \, \hat{i}_r + \delta v_o \, \delta(t_f - t_0) + \delta v_c \, \hat{i}_c - \frac{d}{dt}\left[\delta C \, v_c + \delta v_c \, C \right]\hat{v}_c = 0$$

Integrating over the time span t_0 to t_f,

$$\int_{t_0}^{t_f} \delta R \, i_r \hat{i}_r \, dt +$$

$$\int_{t_0}^{t_f} \delta v_o \, \delta(t_f - t_0) \, dt +$$

$$\int_{t_0}^{t_f} \delta v_c \, \hat{i}_c \, dt -$$

$$\int_{t_0}^{t_f} \frac{d}{dt}\left[\delta C \, v_c + \delta v_c \, C \right]\hat{v}_c \, dt = 0$$

This gives us [Comments are shown against each term]:

$$\delta R \int_{t_0}^{t_f} i_r \hat{i}_r \, dt + \qquad\qquad [a \; convolution \; \mathrm{int}egral]$$

$$\delta v_o(t_f) + \qquad\qquad [the \; impulse \; function \; extracts \; the \; value \; at \, t_f]$$

$$\int_{t_0}^{t_f} \delta v_c \hat{i}_c dt -$$

$$\left\{ \hat{v}_c v_c \delta C + \hat{v}_c C \, \delta v_c \right]_{t_0}^{t_f} - \int_{t_0}^{t_f} (v_c \delta C + C \delta v_c) \, d\hat{v}_c \right\} \; [\mathrm{int}egration \; by \; parts]$$

$$= 0$$

We can make some of the terms in this expression zero [we attempt to make as many of them as possible to be identically zero by making appropriate assumptions about the boundary conditions and about elements of the adjoint network] by making the following assumptions:

$$\hat{v}_c(t_f) = 0 \qquad [A \ boundary \ condition]$$

$$\hat{C} = -C$$

$$\hat{R} = R \qquad [As \ before]$$

Making $\hat{v}_c(t_f) = 0$ will ensure that the first term of the last row becomes zero. To see the sense of making $\hat{C} = -C$, let us look at the entry on the third row and the last item on the last row:

$$\int_{t_0}^{t_f} \delta v_c \hat{i}_c \, dt - \left\{ -\int_{t_0}^{t_f} (C \delta v_c) \, d\hat{v}_c \right\} \qquad = \int_{t_0}^{t_f} \left[\hat{i}_c + C \frac{d\hat{v}_c}{dt} \right] \delta v_c \, dt$$

If $\hat{C} = -C$, we would have $\hat{i}_c = -C \frac{d\hat{v}_c}{dt}$, so that this integral becomes identically zero. This leaves us with the equation:

$$\delta R \int_{t_0}^{t_f} i_r \hat{i}_r \, dt + \delta v_o(t_f) + \delta v_0(t_f) + \left[\hat{v}_c(t_0) v_c(t_0) \delta C + \hat{v}_c(t_0) C \delta v_c(t_0) \right] + \int_{t_0}^{t_f} (v_c \delta C) \, d\hat{v}_c = 0$$

Rearranging, we get:

$$-\delta R \left[\int_{t_0}^{t_f} i_r \hat{i}_r \, dt \right] - \delta C \left[v_c(t_0) \hat{v}_c(t_0) + \int_{t_0}^{t_f} v_c \, d\hat{v}_c \right] - \hat{v}_c(t_0) C \delta v_c(t_0) = \delta v_0(t_f)$$

To get the sensitivity of v_o (t_f) to changes in *R and C*, we consider the limit $\delta R \to 0$ and $\delta C \to 0$:

$$\frac{\partial v_o(t_f)}{\partial R} = -\int_{t_0}^{t_f} i_r \hat{i}_r \, dt - \hat{v}_c(t_0) C \frac{\partial v_c(t_0)}{\partial R}$$

Similarly, we can write down the sensitivity of v_o (t_f) to changes in *C* as:

$$\frac{\partial v_o(t_f)}{\partial C} = -v_c(t_0) \hat{v}_c(t_0) - \int_{t_0}^{t_f} v_c \, d\hat{v}_c - \hat{v}_c(t_0) C \frac{\partial v_c(t_0)}{\partial C}$$

Integrating the second term on the RHS by parts and substituting the assumed boundary conditions [$\hat{v}_c(t_f) = 0$] gives us:

$$\frac{\partial v_o(t_f)}{\partial C} = \int_{t_0}^{t_f} v_c \hat{v}_c \, dt + \hat{v}_c(t_0) C \frac{\partial v_c(t_0)}{\partial C}$$

Robust design

The concept of robust design is also related to that of sensitivity. A system is said to be robust if it is still capable of meeting the performance requirements in an adverse environment. This can be interpreted as an environment full of noise, so that the inputs and / or parameter values are corrupted with noise. To illustrate the concept, we will consider the simplest possible case, that of a single-valued performance criterion, in a system with only one parameter.
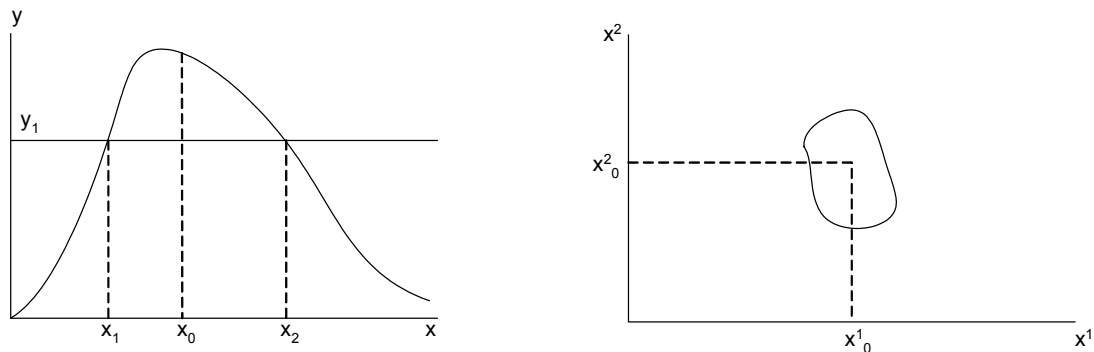
This is illustrated in the first figure, where the criterion y should be more than y_1 for acceptability. The parameter x (nominal value x_0) may lie anywhere between x_1 and x_2, and the system would still pass the acceptance test. The allowable

range of x is dependent upon the sensitivity of y to x at x_0, that is, on $\left[\dfrac{\partial y}{\partial x} \right]_{x=x_0}$.

The second figure illustrates a slightly more complex situation where the two parameters (x_1 and x_2) are required to lie within the space enclosed by the closed curve shown. This may be extended to a hyper-space of n-dimentions, defined by n parameters.

3.3.8 Automatic design and the use of Artificial Intelligence

Automatic design of analogue circuits has been theoretically possible for some time now, using the algorithms that we have studied in the previous sections for the analysis of such circuits. It should be possible to design a circuit to meet given requirements by the repeated application of an automatic circuit analysis programme (such as Spice) and a suitably defined performance criterion, merely by "closing the loop". However, it has been found that the computational burden of such an algorithm is unbearably high, and that it is not possible to obtain a satisfactory solution within reasonable limits of time and effort.

An alternative approach using genetic algorithms has been suggested recently, and satisfactory results have been reported for the design of analogue circuits, notably for the design of filter circuits.

In fact, designs superior to those obtained by experienced design engineers using conventional design techniques have been reported as arising from the use of these techniques. There are some variations among the methods reported in the literature and we will discuss some of them below.

Genetic algorithms

Generic algorithms refer to a class of search techniques that try to emulate the natural phenomenon of evolution. It was first proposed by Holland in the 1960s as a means for the study of the phenomenon of adaptation in nature. He also intended to import the mechanisms of natural adaptation to computer systems. However, our intention here is to use it for the solution of a specific problem; that of searching for an optimal or semi-optimal solution to a stated problem in circuit design.

Genetic algorithms are particularly suited for the solution of problems where the search domain is very large. This is the case with analogue circuit design, for even with self-imposed limitations such as the use of resistors, capacitors and inductors only as circuit elements, and on the number of components to be used, the combinations of connectivity patterns as well as of parameter values is almost infinite.

Chromosomes and genes

Borrowing concepts and vocabulary from evolutionary genetics, each potential solution is described by a chromosome, consisting of a number of genes. Traditionally, a chromosome consists of a fixed number of genes, and the genes themselves are coded in binary form, even though the literature contains instances where variable length chromosomes and non-binary coded genes have been used.

Population and fitness

A population, again by analogy with evolutionary biology, consists of a number of chromosomes representing different individuals or different possible solutions. Each member of the population will be associated with a fitness value, corresponding to how well it meets the desired requirements.

Parents, children and succeeding generations

The GA (genetic algorithm) works through the creation of successive generations, each fitter than its predecessor, by a process of constructing children (of a new generation) from parents (of the older generation). This is accomplished through a process of selection, crossover and mutation.

The complete process is illustrated in the following chart.

```
┌─────────────────────────────┐
│ Start with a randomly generated │
│ population of n l-bit chromosomes. │
└─────────────────────────────┘
                │
        ┌───────────────────────┐
        │ Repeat for required number │
        │ of iterations (or generations) │
        └───────────────────────┘
                │
        ┌───────────────────────┐
        │ Calculate the fitness f(x) │
        │ of each chromosome x │
        └───────────────────────┘
                │
        ┌───────────────────────┐
        │ Repeat until n │
        │ chromosomes (a new │
        │ generation) are │
        │ created │
        └───────────────────────┘
                │
        ┌───────────────────────┐
        │ Select a pair of chromosomes (the │
        │ probability of being chosen is │
        │ proportional to fitness) to be "parents" │
        │ of a new chromosome │
        └───────────────────────┘
                │
        ┌───────────────────────┐
        │ Form a pair of children from the pair of │
        │ parent chromosomes. The children are │
        │ formed either by crossover (with specified │
        │ crossover probability) or, if no crossover │
        │ occurs, by replication of the parents. │
        └───────────────────────┘
                │
        ┌───────────────────────┐
        │ Subject the two new chromosomes to │
        │ mutation, with specified probability of │
        │ mutation. │
        └───────────────────────┘
                │
        ┌───────────────────────┐
        │ Place the "children" in a new │
        │ generation of chromosomes │
        └───────────────────────┘

                One
                Generation
                │
        ┌───────────────────────┐
        │ Select the fittest chromosome as the │
        │ required solution │
        └───────────────────────┘
                │
            (  Stop  )
```

The above is a generic genetic algorithm that could be used for the solution of problems in any domain. We need to make a number of decisions if we are to apply it to the solution of a particular problem, in a particular domain. The first, and perhaps the most difficult, is how to present the problem in a suitable manner.

Two different approaches to passive analogue circuit design using genetic algorithms have been reported, both with some measure of success. In one approach, both the topology of the network as well as its parameter values (the values of the resistors, capacitors and inductors) have been optimised together in one process, through a suitable coding mechanism. In the other, only the topology is represented in the GA, and the optimal parameter values for each configuration is obtained using conventional optimisation methods. [This seems a better idea, for the optimisation of parameter values can be handled more

economically by conventional methods.] These two processes are repeated until a satisfactory solution is obtained. There are many other variations in these algorithms, and different workers have reported different procedures. For example, in some algorithms, a chromosome from a child generation will only replace a member of the parent generation having a lower fitness than itself, while in others there is no such check.

The other major decision is about the fitness function. The fitness function should represent how well the individual (chromosome) meets the design requirements. These may be specified either in the frequency domain or in the time domain, or may even be a combination of the two. As it is possible to meet tighter specifications with higher order circuits with more components, there will be a tendency to increase the number of components indefinitely to achieve a better fit. This is of course counter productive, as both size and costs will escalate with increasing number of components. The fitness function should be designed to take this into account by punishing designs using large numbers of components.

In a typical implementation, the fitness function has been multiplied by a penalty function p(n) of the number of components n, where it has been defined as:

$$p(n) = \frac{1}{1 + a^{(n - n\,\text{max})}}$$

The figure shows plots of p(n) for a = 10 for nmax = 5 and 10. It illustrates how the penalty function decreases rapidly as n passes nmax.
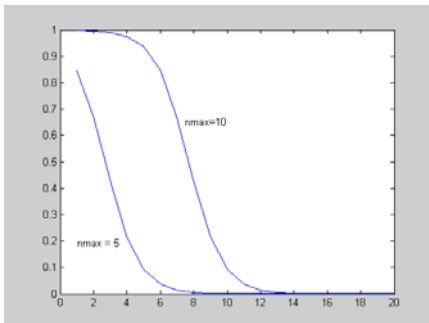
An example

Before we consider the problem of analogue circuit design with all its intricacies, we will consider a simpler example to try to understand how a genetic algorithm works.

Let us consider the problem of maximising the function

$$f(x) = abs\left[\frac{\sin x}{1 + x^2}\right]$$

over the range $-16 < x < 16$

We will first try to gain an insight into the problem by examining how this function behaves over the range of interest. The figure shows a plot of f(x) verses x.
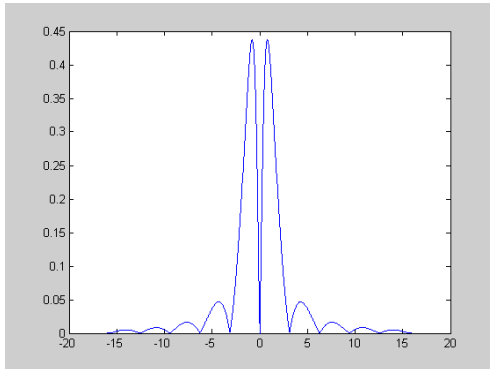
Note that f(x) is a continuous function, and that we need to divide the search space into fairly small segments. As we are interested in representing each solution (that is, each value of x) as a chromosome made up of binary valued genes, one possibility is to construct a chromosome consisting of (say) ten genes, so that we divide the range (-16,16) into 1024 segments. If we consider the number represented by the ten bits as j, then x would be (j-512)/32, for $0 < j \le 1024$.

We will attempt to implement a genetic algorithm to solve this problem using MATLAB.

Define the function to be optimised (maximised):
To do this we first type "edit" in the MATLAB command window. This opens a new window for editing. In this window we can create an m file defining f(x) as follows:

function[y]=f(x)
y=abs(sin(x)/(1.+x^2));

This is then saved as the m file f.m. Let us try this out (as we already have a plot of the function) by substituting different values for x:

>> f(-5.)

ans =

   0.0369

>> f(0)

ans =

   0

```
>> f(1)

ans =

    0.4207
```

We will now create a (random) population (say) of 100 to start the algorithm. As each solution is represented by a binary string of 10 bits (corresponding to 1024), the initial population may be represented by a 100 x 10 random matrix of ones and zeros. In more general terms, the stating population corresponds to a (n x m) matrix, where n is the population size and m is the length of each chromosome.

We will again invoke "edit" from the MATLAB command window and create a new file as follows:

```
                function[y]=starting_population(n,m)
y=round(rand(n,m));
```

The MATLAB command "rand" generates a matrix (of given size) of random numbers in the range (0,1.0) while "round" rounds off a given number to the nearest integer. Together, they generate a matrix of random ones and zeros as required.

Let us try it out (for a starting population of 10, to save space!)

```
>> sp=starting_population(10,10)

sp =

    1   0   0   0   1   0   1   0   0   0
    0   0   0   1   0   0   0   0   0   0
    1   1   1   0   1   0   0   0   1   0
    0   1   0   1   1   0   0   0   1   0
    0   0   1   1   0   0   1   0   0   1
    0   1   1   0   0   0   0   1   0   0
    1   1   1   1   1   1   0   1   1   1
    1   0   1   0   1   0   1   1   0   0
    1   1   0   0   0   0   1   0   1   0
    1   0   0   1   1   1   1   1   1   1
```

Each row corresponds to one individual (one chromosome) of the starting population [Note that re-running the programme will generate a different set of values.] We may write an m file to extract any chromosome we wish from the total population. The following will extract the jth chromosome:

```
function[y]=chromosome(population,j)
[n,m]=size (population);
y=zeros(m);
y=population(j,:);
```

We will try this out for a few chromosomes:

> chromosome(sp,1)

ans =

   1   0   0   0   1   0   1   0   0   0

>> chromosome(sp,5)

ans =

   0   0   1   1   0   0   1   0   0   1

In terms of decimal values (in the range 0 – 1023), our starting population corresponds to:

     552, 64, 930, 354, 201, 388, 1015, 684, 778, 639

We need to be able to do this conversion automatically. To do this, we will first create a vector of binary weights and then multiply the vector of binary values in each chromosome by the weights:

```
        function[y]=weights(n)
y=ones(n,1);
for j=1:n,
                y(j)=y(j)*2^(n-j);
end
```

        >> weights(10)

ans =

   512
   256
   128
    64
    32
    16
     8
     4
     2
     1

We can now obtain the required decimal values by multiplying the binary sequence corresponding to any chromosome by the weight vector.

For example, for the fifth chromosome:

```
>> chromosome(sp,5)

ans =

   0   0   1   1   0   0   1   0   0   1

>> chromosome (sp,5)*weights(10)

ans =

  201
```

To obtain all the values:

```
>> x=zeros(10,1);
>> for j=1:10,
x(j)=chromosome(sp,j)*weights(10);
    end
>> x

x =

        552
        64
        930
        354
        201
        388
        1015
        684
        778
        639
```

Now we proceed to evaluate the 'goodness' or 'fitness' of each chromosome, using the function 'f' defined at step 1. We write all these into a single function 'goodness' as follows:

```
        f function[y]=goodness(funct,population)
n=size(population,1);
m=size(population,2);
x=zeros(m,1);
y=zeros(m,1);
for j= 1:m
                x(j)=(chromosome(population,j)*weights(n)-512.)/32.;
                y(j)=feval(funct,(x(j)));
end
```

Calling it from Matlab (the function is f and the population is sp) we obtain:

y=goodness('f',sp)

y =

    0.3703
    0.0050
    0.0028
    0.0384
    0.0030
    0.0418
    0.0000
    0.0264
    0.0128
    0.0439

The next step in our flowchart is to find a pair of chromosomes to be the parents, where the probability of being so chosen is proportional to the fitness computed as above. One way of doing this is to scale the fitness factors so that they add up to one, arrange them to form cumulative sums on a straight line from zero to one, get a random number in the range (0,1) and select the corresponding chromosome as a parent. The cumulative sum may be formed as follows:

 cumy=cumsum(y)/sum(y)

cumy =

    0.6801
    0.6893
    0.6944
    0.7650
    0.7705
    0.8473
    0.8474
    0.8958
    0.9193
    1.0000

The following function will select a parent from the population in the manner described above:

r=rand;
parent=find (r<cumy);
p=parent(1);

Try this out (you may get different answers, as it is a random process. However, over a large number of trials, the probability of selecting a particular parent would depend on its fitness or goodness factor.) We will now use these two segments of code in a function to create an array of parents, of the same length as the original population:

```
function[y]=select_parents(goodness_coef)
n=length(goodness_coef);
normalised_cu_goodness=zeros(n,1);
normalised_cu_goodness=cumsum(goodness_coef)/sum(goodness_coef);
for j=1:n
    r=rand;
    parent=find (r<normalised_cu_goodness);
    y(j)=parent(1);
end
```

If we run this function with input equal to the goodness coefficients, we should get a selection for parents:

> select_parents(y)

ans =

   1   6   1   1   1   6   1   2   1   6

We will run it again (and again), and get different selections, due to the random nature of the selection process.

>> select_parents(y)

ans =

   1   4   1   1   1   1   2   1   1   1

>> select_parents(y)

ans =

  10   1   1   1   8   4   1   1   6   1

Note that chromosome 1 which has the highest fitness is selected most often as a parent.

The next step is to form a pair of children from a pair of parents. The process we use is called "crossover" We will consider how this is done using the first two parents selected, that is parent 10 and parent 1.

Parent 10: 1   0   0   1   1   1   1   1   1   1
Parent 1 : 1   0   0   0   1   0   1   0   0   0

The crossover will be affected at a randomly chosen point p1.

```
>> p1=floor((m-1)*rand)+1;
>> p1

p1 =

    2
```

The first child is formed by taking the first two genes from parent 10 and the balance from parent 1, while taking the first two genes from parent 1 and the balance from parent 10 forms the second child.

Child 1 : 1   0   0   0   1   0   1   0   0   0
Child 2 : 1   0   0   1   1   1   1   1   1   1

[In this particular case, the two children happen to be the same as the two parents!. This is not so in general.]

We will consider the next pair of parents, to get the third and fourth children. The parents are selected from the list:

  10   1   1   1   8   4   1   1   6   1

We have already considered parents 10 and 1. The next pair is 1 and 1. This will yield two children with the same chromosomes as parent 1, whatever the point of crossover.

Child 3 : 1   0   0   0   1   0   1   0   0   0
Child 4 : 1   0   0   0   1   0   1   0   0   0

The next pair of children will be formed by crossover between parents 8 and 4.

Parent 8: 1   0   1   0   1   0   1   1   0   0
Parent 4: 0   1   0   1   1   0   0   0   1   0

The crossover point is given by:
```
>> p1=floor((m-1)*rand)+1;
>> p1
p1 =
    6
```

This gives us:

```
Child 5: 1   0   1   0   1   0   0   0   1   0
Child 6: 0   1   0   1   1   0   1   1   0   0
```

The next set of parents are again 1 and 1, giving identical children:

```
Child 7 : 1   0   0   0   1   0   1   0   0   0
Child 8 : 1   0   0   0   1   0   1   0   0   0
```

The last set of children is from parents 6 and 1:
The crossover point is 3 (using of the random process described earlier).

```
Parent 6: 0   1   1   0   0   0   0   1   0   0
Parent 1: 1   0   0   0   1   0   1   0   0   0
```

The children are:

```
Child 9:        0   1   1   0   1   0   1   0   0   0
Child 10:  1   0   0   0   0   0   0   1   0   0
```

The new population thus is:

```
1   0   0   0   1   0   1   0   0   0
1   0   0   1   1   1   1   1   1   1
1   0   0   0   1   0   1   0   0   0
1   0   0   0   1   0   1   0   0   0
1   0   1   0   1   0   0   0   1   0
0   1   0   1   1   0   1   1   0   0
1   0   0   0   1   0   1   0   0   0
1   0   0   0   1   0   1   0   0   0
0   1   1   0   1   0   1   0   0   0
1   0   0   0   0   0   0   1   0   0
```

Note that the fittest individual in the original population (chromosome 1) has now produced six children with identical genes, while the others have declined. One more iteration will produce nine individuals (chromosomes) with this combination of genes and only one other.

We will now write the last few instructions into an m file to create a new population. We will do this in two stages; stage 1 will create two children from two specified parents while stage two will create a total population. To create two children from parent1 and parent2 we will use the function parents_to_children:

```
function [y]=parents_to_children(population,parent1,parent2)
    m=size(population,2);
    child1=zeros(m,1);
```

```
    child2=zeros(m,1);
    pointer=floor(((m-1)*rand)+1);
    child1=[population(parent1,1:pointer),population(parent2,pointer+1:m)];
    child2=[population(parent2,1:pointer),population(parent1,pointer+1:m)];
    y=[child1 ; child2];
```

We will now use this in a new function 'generation' to create a new generation, replacing the old population with its children:

```
function[y]=generation(population,selection)
    m=size(population,2);
    n=floor((size(population,1))/2);
    children=zeros(size(population));
    children=population;
    for j=1:n,
        k=2*j;
        parent1=selection(k-1);
        parent2=selection(k);
        children(k-1:k,:)=parents_to_children(population,parent1,parent2);
        end
    y=children;
```

Try out the above sequence of operations and see how the 'goodness' improves over the generations. However, the improvements stop after some time, and the population settles down at a sub-optimum.

Mutation

Mutation, with a low probability of occurrence, is used to overcome this phenomenon, just as in nature. We can use the following function to achieve this:

```
function[y]=mutate(population,probability)
[n,m]=size(population);
y=population;
for j=1:n
    for k=1:m
        if (rand < probability)
            y(j,k)=1-y(j,k);
        end
    end
end
```

Let us try this on the population we last had:

Before mutation:

```
Population (np): 1   0   0   0   1   0   1   0   0   0
1   0   0   1   1   1   1   1   1   1
1   0   0   0   1   0   1   0   0   0
1   0   0   0   1   0   1   0   0   0
1   0   1   0   1   0   0   0   1   0
0   1   0   1   1   0   1   1   0   0
1   0   0   0   1   0   1   0   0   0
1   0   0   0   1   0   1   0   0   0
0   1   1   0   1   0   1   0   0   0
1   0   0   0   0   0   0   1   0   0
```

We can mutate this (with a selected probability, say 0.02) and observe the resulting population:
>> npm=mutate(np,0.02)

npm =

```
1   0   0   0   1   0   0   0   0   0
1   0   0   1   1   1   1   1   1   1
1   0   0   0   1   0   1   0   0   0
1   0   0   0   1   0   1   0   0   0
1   0   1   0   1   0   0   0   1   1
0   1   0   1   1   0   1   1   0   0
1   0   0   0   1   0   1   0   0   0
1   0   0   0   1   0   1   0   0   0
0   1   1   0   1   0   1   0   0   0
1   0   0   0   0   0   0   1   0   0
```

Let us look at the 'goodness' factors before and after mutation:

Before mutation:

>> gnp=goodness('f',np)

gnp =

```
    0.3703
    0.0439
    0.3703
    0.3703
    0.0353
    0.0445
    0.3703
    0.3703
    0.0446
    0.1228
```

>> gnpm=goodness('f',npm)

gnpm =

   0.4207
   0.0439
   0.3703
   0.3703
   0.0344
   0.0445
   0.3703
   0.3703
   0.0446
   0.1228

The average 'goodness' has increased marginally from 0.2143 to 0.2192, but more importantly, that of the best chromosome has improved from 0.3703 to 0.4207. We have of course to remember that this is based on a random process, and that these coefficients are equally likely to decrease due to mutation. However, taken in combination with crossover, it tends to search out global optima over a large number of iterations.

We will now incorporate all the separate functions into a single algorithm:

```
function[y]=genetic_algorithm(population_size,chromosome_length,mutation_pro
bability,funct,itterations)
    p=starting_population(population_size,chromosome_length);
    for j=1:itterations
        g=goodness(funct,p);
        s=select_parents(g);
        p=generation(p,s);
        p=mutate(p,mutation_probability);
    end;
    y=p;
```

This algorithm is suitable as a a genetic algorithm for the solution of any problem, provided a chromosome consisting of a finite number of bits and a fitness function can be defined to represent the problem at hand.