3

Principles of Passive Electrical Circuit Modeling

Preview

In this chapter, we discuss issues relating to the modeling of simple passive electrical circuits consisting of sources, resistors, capacitors, and inductors only. The traditional approach to these types of systems is through either mesh equations or node equations. However, the resulting models are not in a state–space form, and they cannot easily be converted into a state–space form thereafter. We also discuss another technique which allows us to derive a state–space model directly, and we shall see why this approach is not commonly used: very often, the resulting equations contain either algebraic loops or structural singularities.

3.1 Introduction

A good selection of textbooks deal with passive electrical circuits and simulations thereof [3.1,3.2,3.3,3.4]. The most commonly used modeling principles are to express the circuit equations either through a special selection of *mesh equations* (expressed in terms of so–called loop currents using Kirchhoff's voltage law), or through a special selection of *node equations* (expressed in terms of so–called cutset voltages using Kirchhoff's current law). Let me explain the basic idea behind these two methods by means of the example shown in Fig.3.1.

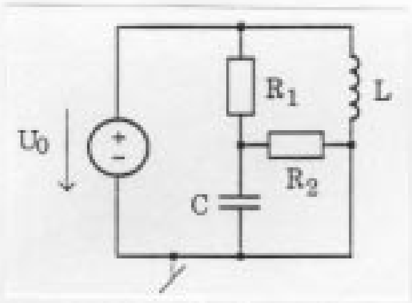

Figure 3.1. Example of a passive circuit

We have two elements that can store energy (the capacitor C and the inductor L), and we thus expect to obtain two state equations in the end.

3.2 Mesh Equations

Let me first discuss how the loop current approach (mesh equations, Kirchhoff's voltage law) can be used to generate a mathematical model for this circuit. Fig.3.2 shows the same circuit after the circuit has been "colored" by introducing a "tree".

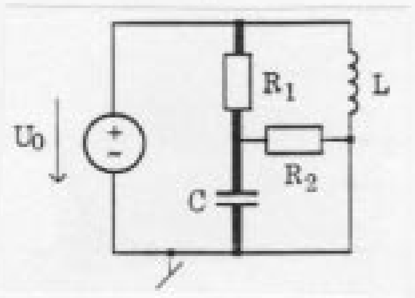

Figure 3.2. Passive circuit after selection of a tree

The "tree_branches" of the tree are those branches of the circuit that have been marked by bold lines (i.e., the branches containing the resistor R_1 and the capacitor C). Let me define what a tree is.

"A tree consists of a set of *connected tree_branches* such that the tree_branches alone don't form closed loops, and such

> that any addition of another branch to the tree would create a closed loop consisting of tree_branches only. The remaining branches of the circuit structure are called the *links* of the circuit."

A considerable freedom exists in the selection of tree_branches. However, some rules must be observed.

(1) Mesh equations cannot tolerate any independent current sources. Node equations cannot tolerate any independent voltage sources. If the circuit contains the wrong type of sources, they must be converted to equivalent sources of the other type. Fig.3.3 shows the conversion of independent sources.

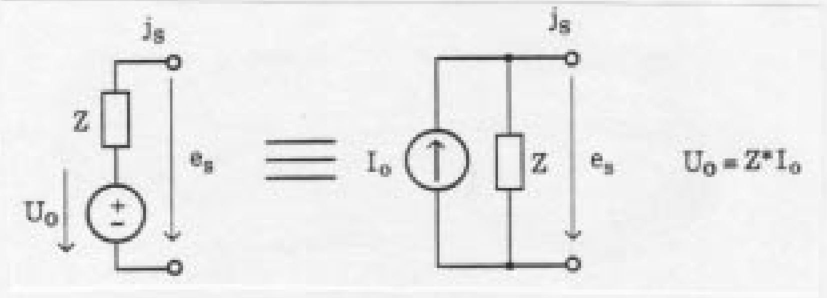

Figure 3.3. Conversion of independent sources

Furthermore, if the "wrong" sources are ideal sources (i.e., they have zero impedance associated with them), they must first be moved into other branches until the problem disappears.

(2) In the case of mesh equations, all voltage sources should be placed in *links*. In the case of node equations, all current sources should be placed in *branches*. In this way, they will appear only once in the resulting set of equations.

Using the following notation:

n_n	::=	number of circuit nodes
n_b	::=	number of circuit branches
n_l	::=	number of links
n_{tb}	::=	number of tree_branches
σ_i	::=	number of ideal current sources
σ_u	::=	number of ideal voltage sources
n_{ej}	::=	number of mesh equations
n_{ee}	::=	number of node equations

we can compute the number of tree_branches n_{tb} and the number of links n_l as follows:

$$n_{tb} = n_n - 1 \tag{3.1a}$$
$$n_l = n_b - n_{tb} \tag{3.1b}$$

and therefore, we can compute the number of equations that are needed for the two methods as:

$$n_{ee} = n_{tb} - \sigma_u \tag{3.2a}$$
$$n_{ej} = n_l - \sigma_i \tag{3.2b}$$

We usually select the technique that lets us get away with the smaller number of equations.

In our example, we have an ideal independent voltage source, thus mesh equations may be more convenient, i.e., we operate on Kirchhoff's voltage law rather than using Kirchhoff's current law.

It is useful to replace all passive circuit elements by impedances as shown in Fig.3.4 (i.e., we convert the circuit from the time domain to the frequency domain).

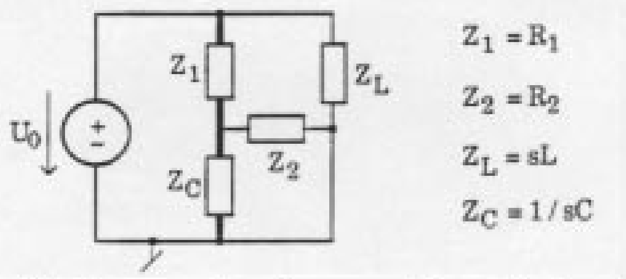

Figure 3.4. Frequency domain representation using impedances

Now, we introduce so–called *loop currents*, one for each link of the circuit. A *loop* is a generalized mesh. Except for the one link that it represents, it consists of tree_branches only. Fig.3.5 depicts the three loops of our circuit. Once the tree has been selected, the loops are fully determined. Notice that the short–circuit at the lower right corner of Fig.3.5 is drawn for convenience only and does not qualify for a link. The loop currents j_1, j_2, and j_3 are identical with the link currents i_1, i_2, and i_3 of Fig.3.6. The tree_branch currents i_4, and

i_5 are the directed sums of the loop currents that traverse the two tree branches.

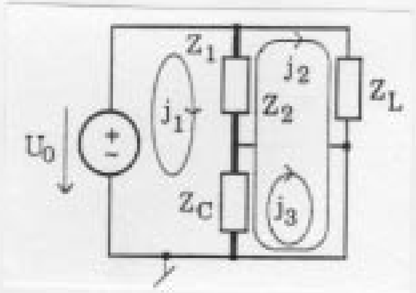

Figure 3.5. Circuit with tree and loop currents

$$U_0 = Z_1 * (j_1 - j_2) + Z_C * (j_1 - j_2 - j_3) \tag{3.3a}$$
$$0 = Z_L * j_2 + Z_1 * (j_2 - j_1) + Z_C * (j_2 + j_3 - j_1) \tag{3.3b}$$
$$0 = Z_2 * j_3 + Z_C * (j_3 + j_2 - j_1) \tag{3.3c}$$

The terms can be reordered as follows:

$$U_0 = (Z_1 + Z_C) * j_1 - (Z_1 + Z_C) * j_2 - Z_C * j_3 \tag{3.4a}$$
$$0 = -(Z_1 + Z_C) * j_1 + (Z_1 + Z_C + Z_L) * j_2 + Z_C * j_3 \tag{3.4b}$$
$$0 = -Z_C * j_1 + Z_C * j_2 + (Z_2 + Z_C) * j_3 \tag{3.4c}$$

which can be expressed using a matrix notation as:

$$\begin{pmatrix} U_0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} Z_1 + Z_C & -(Z_1 + Z_C) & -Z_C \\ -(Z_1 + Z_C) & Z_1 + Z_C + Z_L & Z_C \\ -Z_C & Z_C & Z_2 + Z_C \end{pmatrix} \cdot \begin{pmatrix} j_1 \\ j_2 \\ j_3 \end{pmatrix} \tag{3.5}$$

which can be abbreviated as:

$$\mathbf{e}_s = \mathbf{Z}_m * \mathbf{j}_l \tag{3.6}$$

$\mathbf{e}_s$ denotes the *source voltage vector*, $\mathbf{Z}_m$ denotes the *mesh impedance matrix*, and $\mathbf{j}_l$ denotes the *loop current vector.*

Somewhat more systematically, we can achieve the same result by starting off with two other matrices, namely the mesh incidence matrix and the branch impedance matrix. The *mesh incidence matrix* $\mathbf{\Phi}$, which in some texts is also called the *fundamental loop matrix,* is defined as a matrix that describes the circuit topology by coding the direction of the loop currents in the branches. Fig.3.6 illustrates the procedure.

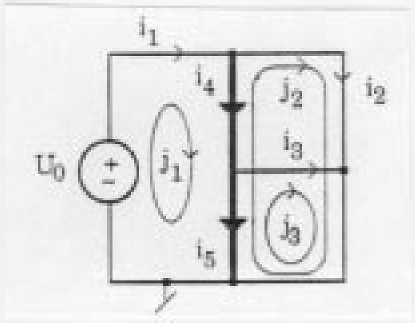

Figure 3.6. Circuit topology used for the mesh incidence matrix

This allows us to generate the following mesh incidence matrix:

$$\Phi = \begin{array}{c} \\ j1 \\ j2 \\ j3 \end{array} \begin{array}{c} \begin{array}{ccccc} i1 & i2 & i3 & i4 & i5 \end{array} \\ \left(\begin{array}{ccccc} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & -1 & -1 \\ 0 & 0 & 1 & 0 & -1 \end{array} \right) \end{array} \tag{3.7}$$

which contains +1 entries where the direction of a loop current corresponds with the direction of the branch current, it contains −1 entries where the loop current and the branch current have opposite directions, and it contains 0 entries for branches in which the loop current is not present.

The *branch impedance matrix* $\mathbf{Z}_b$ is defined as a diagonal matrix containing the individual branch impedances along the main diagonal:

$$\mathbf{Z}_b = \begin{array}{c} \\ i1 \\ i2 \\ i3 \\ i4 \\ i5 \end{array} \begin{array}{c} \begin{array}{ccccc} i1 & i2 & i3 & i4 & i5 \end{array} \\ \left(\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 0 & sL & 0 & 0 & 0 \\ 0 & 0 & R_2 & 0 & 0 \\ 0 & 0 & 0 & R_1 & 0 \\ 0 & 0 & 0 & 0 & 1/sC \end{array} \right) \end{array} \tag{3.8}$$

which we sometimes abbreviate as:

$$\mathbf{Z}_b = diag(0, sL, R_2, R_1, 1/sC) \tag{3.9}$$

We can now write all equations in a compact matrix form. Let us start with Kirchhoff's voltage law:

$$\mathbf{\Phi} \cdot \mathbf{u}_b = 0 \tag{3.10}$$

where $\mathbf{u}_b$ denotes the vector of voltages across each of the circuit branches. This can then be expressed as:

$$\mathbf{u}_b = \mathbf{Z}_b \cdot \mathbf{i}_b + \mathbf{u}_\sigma \tag{3.11}$$

where $\mathbf{i}_b$ denotes the vector of currents through each of the circuit branches, and $\mathbf{u}_\sigma$ denotes the vector of voltage sources in the circuit branches. We can now transform the vector of branch currents into the vector of loop currents as follows:

$$\mathbf{i}_b = \mathbf{\Phi}^T \cdot \mathbf{j}_l \tag{3.12}$$

Plugging the last three equations into each other, we find:

$$\mathbf{\Phi} \cdot \mathbf{Z}_b \cdot \mathbf{\Phi}^T \cdot \mathbf{j}_l = -\mathbf{\Phi} \cdot \mathbf{u}_\sigma \tag{3.13}$$

A comparison with eq(3.6) yields:

$$\mathbf{Z}_m = \mathbf{\Phi} \cdot \mathbf{Z}_b \cdot \mathbf{\Phi}^T \tag{3.14a}$$
$$\mathbf{e}_\sigma = -\mathbf{\Phi} \cdot \mathbf{u}_\sigma \tag{3.14b}$$

We can now evaluate *all* loop currents at once by computing:

$$\mathbf{j}_l = \mathbf{Z}_m{}^{-1} \cdot \mathbf{e}_\sigma \tag{3.15}$$

which we shall often abbreviate as:

$$\mathbf{j}_l = \mathbf{Z}_m \backslash \mathbf{e}_\sigma \tag{3.16}$$

using the slash operator ("/") to denote matrix division from the right, and the backslash operator ("\") to denote matrix division from the left. This is the notation used in MATLAB and in CTRL–C. We can then immediately find all branch currents using eq(3.12), and finally, we can find all branch voltages using eq(3.11). Notice, however, that the evaluation of eq(3.16) is more tricky than it seems at first sight since it involves the symbolic inversion of a polynomial matrix. Neither MATLAB nor CTRL–C can handle this type of matrix inversions.

3.3 Node Equations

Let me next discuss the alternative approach using node equations and Kirchhoff's current law. Since we now have a source of the "wrong" type, we first need to convert the circuit. Fig.3.7 shows how this is done.

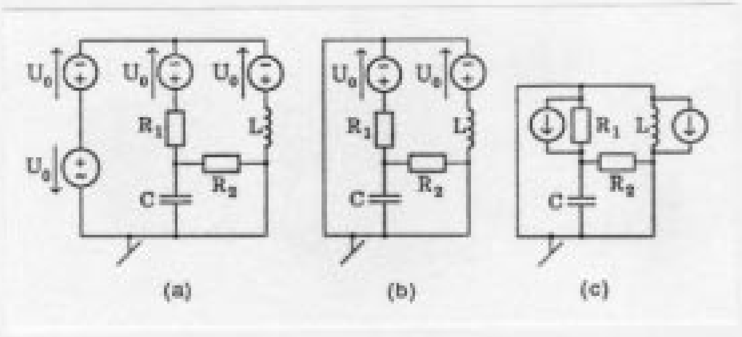

Figure 3.7. Conversion of the voltage source

Since the "wrong" source is ideal, we start by moving the source into other branches. This is easily accomplished by compensating the source with an equivalent source of reverse polarity as shown in Fig.3.7a and Fig.3.7b. Fig.3.7b is equivalent to the original circuit in every respect except for the potential at the additional top node. Now, we can convert the voltage sources to equivalent current sources as shown in Fig.3.7c. This circuit is again equivalent to the previous ones *except for the internal characteristics of the sources.* Consequently, the voltage across and the current through the inductor L and the resistor R_1 are no longer the same as before. In fact, the inductor has been short-circuited altogether. Since these "modifications" affect about half of our original circuit, this approach may not be sensible for the given problem. However, if we wish to determine the voltage across the capacitor only, this approach works perfectly well.

Instead of continuing with this example, let me demonstrate this technique by means of a slightly different example. Fig.3.8 shows another passive circuit.

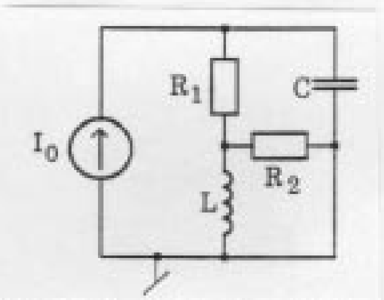

Figure 3.8. Another passive circuit

Fig.3.9 demonstrates the steps needed to prepare the circuit for the formulation of node equations using Kirchhoff's current law.

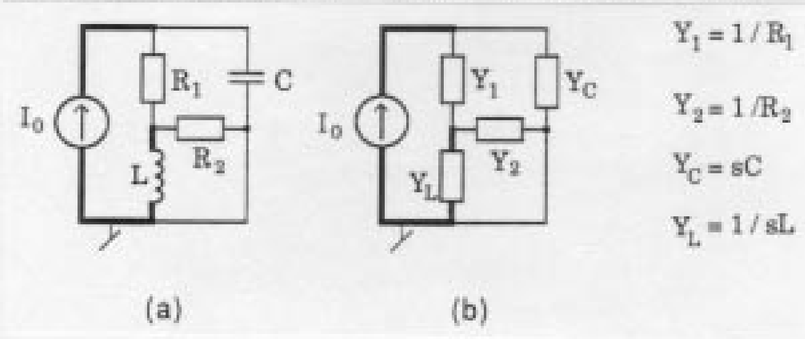

Figure 3.9. Preparation of the circuit for node equations

Fig.3.9a shows the selection of the tree which now should contain the current source. Every node of the circuit must be reached by the tree. It is usually a good idea to build the tree as a *star* with the center at the ground node (reference node). For this purpose, it is often necessary to introduce additional fictitious tree_branches (tree_branches with zero admittance). Fig.3.9b shows the conversion of the circuit from the time domain to the frequency domain, now using *admittances* rather than *impedances*.

Then, we introduce so-called *cutset potentials*, one for each tree_branch of the circuit. A *cutset* is a generalized node. Except for the one tree_branch that it represents, it cuts through links only. Fig.3.10a depicts the three cutsets of our circuit. Once the tree has been selected, the cutsets are fully determined. The cutset potentials e_1, e_2, and e_3 are identical with the node potentials at the nodes in which the tree_branches end. If every tree_branch connects one node of the circuit with the reference node (as in our example), the cutset potentials are also identical with the voltages across the tree branches u_1, u_2, and u_3 of Fig.3.10b. The link voltages u_4, and u_5 are the directed sums of the cutset potentials that cut through the two links.

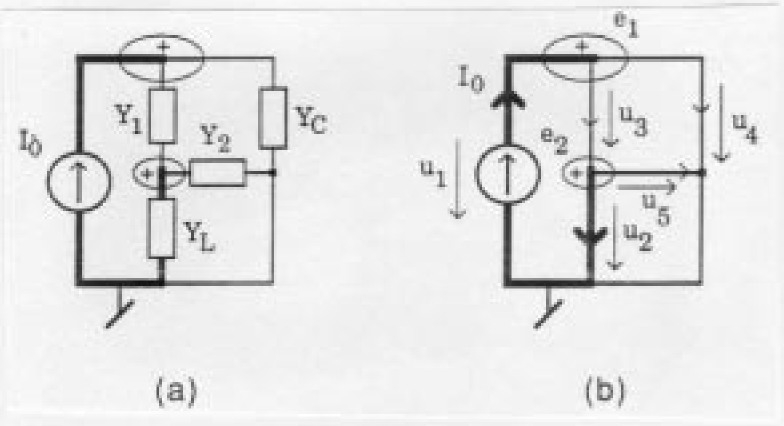

Figure 3.10. Introducing cutset voltages

Fig.3.10a shows the introduction of cutset voltages and their polarities. Fig.3.10b places direction conventions on all branch voltages.

Using the circuit as shown in Fig.3.10a, we can immediately proceed to generate circuit equations by applying Kirchhoff's current law to every cutset of the tree:

$$I_0 = Y_1 * (e_1 - e_2) + Y_C * e_1 \tag{3.17a}$$
$$0 = Y_L * e_2 + Y_1 * (e_2 - e_1) + Y_2 * e_2 \tag{3.17b}$$

which can be reordered as:

$$I_0 = (Y_1 + Y_C) * e_1 - Y_1 * e_2 \tag{3.18a}$$
$$0 = -Y_1 * e_1 + (Y_1 + Y_2 + Y_L) * e_2 \tag{3.18b}$$

which can further be written in a matrix notation as:

$$\begin{pmatrix} I_0 \\ 0 \end{pmatrix} = \begin{pmatrix} Y_1 + Y_C & -Y_1 \\ -Y_1 & Y_1 + Y_2 + Y_L \end{pmatrix} \cdot \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} \tag{3.19}$$

which can be abbreviated as:

$$\mathbf{j}_\sigma = \mathbf{Y}_n * \mathbf{e}_l \tag{3.20}$$

where $\mathbf{j}_\sigma$ denotes the *source current vector*, $\mathbf{Y}_n$ denotes the *node admittance matrix*, and $\mathbf{e}_l$ denotes the *cutset voltage vector*.

As before, we can achieve the same result more systematically by starting off with two other matrices, namely the node incidence matrix, and the branch admittance matrix. The *node incidence matrix* $\mathbf{\Psi}$, which is sometimes also called the *fundamental cutset matrix*, is defined as a matrix that describes the circuit topology by recording the direction of the cutset voltages relative to the direction of the branch voltages. This procedure is illustrated in Fig.3.10b which allows us to generate the following node incidence matrix:

$$\mathbf{\Psi} = \begin{array}{c} \\ e1 \\ e2 \end{array} \begin{array}{c} \begin{array}{ccccc} u1 & u2 & u3 & u4 & u5 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 & 1 \end{pmatrix} \end{array} \tag{3.21}$$

The nod incidence matrix contains $+1$ entries where the direction of a cutset voltage corresponds with the direction of the branch voltage, it contains -1 elements where the cutset voltage and the branch voltage have opposite directions, and it contains 0 entries for branches in which the cutset voltage is not present.

The *branch admittance matrix* $\mathbf{Y}_b$ is defined as a diagonal matrix containing the individual branch admittances along the main diagonal:

$$\mathbf{Y}_b = diag(0, 1/sL, 1/R_1, sC, 1/R_2) \tag{3.22}$$

We can again write all equations in a compact matrix form. Let us start with Kirchhoff's current law:

$$\mathbf{\Psi} \cdot \mathbf{i}_b = 0 \tag{3.23}$$

where $\mathbf{i}_b$ denotes the vector of currents through each of the circuit branches. This can then be expressed as:

$$\mathbf{i}_b = \mathbf{Y}_b \cdot \mathbf{u}_b + \mathbf{i}_\sigma \tag{3.24}$$

where $\mathbf{u}_b$ denotes the vector of voltages across each of the circuit branches, and $\mathbf{i}_\sigma$ denotes the vector of current sources in the circuit branches. We can now transform the vector of branch voltages into the vector of cutset voltages as follows:

$$\mathbf{u}_b = \mathbf{\Psi}^T \cdot \mathbf{e}_l \tag{3.25}$$

Plugging the last three equations into each other, we find:

$$\mathbf{\Psi} \cdot \mathbf{Y}_b \cdot \mathbf{\Psi}^T \cdot \mathbf{e}_l = -\mathbf{\Psi} \cdot \mathbf{i}_\sigma \tag{3.26}$$

A comparison with eq(3.20) yields:

$$\mathbf{Y}_n = \mathbf{\Psi} \cdot \mathbf{Y}_b \cdot \mathbf{\Psi}^T \tag{3.27a}$$
$$\mathbf{j}_\sigma = -\mathbf{\Psi} \cdot \mathbf{i}_\sigma \tag{3.27b}$$

We can now evaluate *all* cutset voltages at once by computing:

$$\mathbf{e}_l = \mathbf{Y}_n \backslash \mathbf{j}_\sigma \tag{3.28}$$

We can then immediately find all branch voltages using eq(3.25), and finally, we can find all branch currents using eq(3.24). Notice, however, that also the evaluation of eq(3.28) is more tricky than it seems since it again involves the symbolic inversion of a polynomial matrix.

3.4 Disadvantages of Mesh and Node Equations

We have not answered the question yet how these techniques can help us to come up with a set of *first order differential equations*, i.e., our *state–space model*. Let us return once more to the original circuit example, and the set of equations as formulated in eq(3.3a-c). In order to derive a state–space description, we need to transform these equations back to the time domain:

$$U_0 = R_1(j_1 - j_2) + \frac{1}{C}\int_0^t (j_1 - j_2 - j_3)d\tau \tag{3.29a}$$

$$0 = L\frac{dj_2}{dt} + R_1(j_2 - j_1) + \frac{1}{C}\int_0^t (j_2 + j_3 - j_1)d\tau \tag{3.29b}$$

$$0 = R_2 j_3 + \frac{1}{C}\int_0^t (j_3 + j_2 - j_1)d\tau \tag{3.29c}$$

In order to come up with state equations, we need to get rid of the integral terms. This can be achieved by *differentiating* all three equations once:

$$\frac{dU_0}{dt} = R_1\left(\frac{dj_1}{dt} - \frac{dj_2}{dt}\right) + \frac{1}{C}(j_1 - j_2 - j_3) \tag{3.30a}$$

$$0 = L\frac{d^2 j_2}{dt^2} + R_1\left(\frac{dj_2}{dt} - \frac{dj_1}{dt}\right) + \frac{1}{C}(j_2 + j_3 - j_1) \tag{3.30b}$$

$$0 = R_2\frac{dj_3}{dt} + \frac{1}{C}(j_3 + j_2 - j_1) \tag{3.30c}$$

We realize that several bad things have happened.

(1) In our equations, the term $\frac{dU_0}{dt}$ suddenly appears. This is a derivative of an input signal. We certainly don't want to operate on such a signal, and yet, we shall have a hard time getting rid of it again.

(2) A second derivative term appeared in our equations, which does not fit into our state–space description, and which needs to be reduced to two first order terms (by means of a technique that we shall discuss in Chapter 5).

(3) These equations seem to describe a *fourth order system* while we know that the order of our system can certainly not be higher than two. Consequently, linear dependencies must exist between some of the derivative terms in these equations.

This discussion clearly demonstrates that circuit equations are not geared towards the generation of a state–space model. We can draw two possible conclusions:

(1) The methodology demonstrated above is not adequate to generate a state–space model, and thus, we need to come up with a different modeling methodology that will allow us to generate the requested state–space model directly, or:

(2) State–space models are not the right approach to describe electrical circuits, and thus, we need to come up with a different simulation methodology that will allow us to perform simulation runs using the above generated circuit equations directly.

Both argumentations have their pro's and con's, and thus, we shall proceed along both avenues.

3.5 State–Space Models

Let me begin with the first alternative. A good technique to come up with state equations directly is the following. We start by introducing variables for every single current and voltage in the circuit. This is shown in Fig.3.11 for our original circuit example.

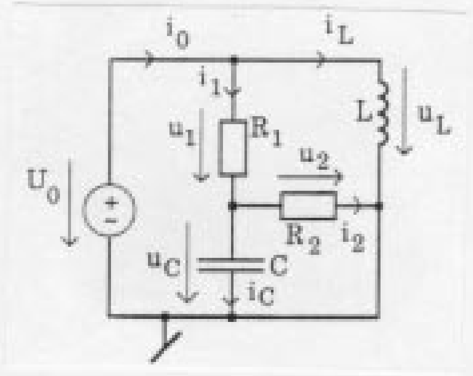

Figure 3.11. Passive circuit with all variables named

In our example, we have introduced the following nine unknowns: u_1, u_2, u_C, u_L, i_1, i_2, i_C, i_L, and i_0.

We can now go ahead and write four branch equations and five mesh and/or node equations. In general, we add linearly independent equations until we have as many equations as we have variables in the circuit. For our example, the following set of equations is found:

$$u_1 = R_1 * i_1 \tag{3.31a}$$
$$u_2 = R_2 * i_2 \tag{3.31b}$$
$$u_L = L * \frac{di_L}{dt} \tag{3.31c}$$
$$i_C = C * \frac{du_C}{dt} \tag{3.31d}$$
$$U_0 = u_1 + u_C \tag{3.31e}$$
$$u_C = u_2 \tag{3.31f}$$
$$u_L = u_1 + u_2 \tag{3.31g}$$
$$i_0 = i_1 + i_L \tag{3.31h}$$
$$i_1 = i_2 + i_C \tag{3.31i}$$

Now, we need to *solve* these equations for the appropriate variables. We start by remembering that our goal is to come up with a set of first order differential equations. Therefore, we solve eq(3.31c) and eq(3.31d) for the derivative terms. Consequently, our two state variables will be the current through the inductor i_L and the voltage across the capacitor u_C. We can mark these two variables as "solved" in our list of variables (by crossing them out from the list of unknowns. We place the two derivative terms in eq(3.31c) and eq(3.31d) in square brackets (meaning that we want to solve for these variables), and we underline the two state variables in all other equations where ever they occur.

Now, we can proceed with either of two philosophies.

(1) We can look for equations which have only one unknown left. We need to solve that equation for this one unknown, or we won't use the equation at all.

(2) We can look for variables that occur in one equation only. We must use that equation to evaluate the unknown, otherwise we won't evaluate that variable at all.

With each unknown found, we proceed in the same way. We cross it out from the list of unknowns, we place it in square brackets in the equation that we plan to use for its evaluation, and we underline it in all other equations. (Of course, all input variables, such as U_0 in our example, are known right away, and can thus be underlined in all equations from the beginning.) We proceed until all variables have been crossed out, and until all equations have been used up. In our example, this algorithm leads to the following *unique* solution:

$$u_1 = R_1 * [i_1] \tag{3.32a}$$

$$u_2 = R_2 * [i_2] \tag{3.32b}$$

$$u_L = L * [\frac{di_L}{dt}] \tag{3.32c}$$

$$i_C = C * [\frac{du_C}{dt}] \tag{3.32d}$$

$$U_0 = [u_1] + u_C \tag{3.32e}$$

$$u_C = [u_2] \tag{3.32f}$$

$$[u_L] = u_1 + u_2 \tag{3.32g}$$

$$[i_0] = i_1 + i_L \tag{3.32h}$$

$$i_1 = i_2 + [i_C] \tag{3.32i}$$

which can now be rearranged as follows:

$$i_1 = u_1/R_1 \tag{3.33a}$$
$$i_2 = u_2/R_2 \tag{3.33b}$$
$$\frac{di_L}{dt} = u_L/L \tag{3.33c}$$
$$\frac{du_C}{dt} = i_C/C \tag{3.33d}$$
$$u_1 = U_0 - u_C \tag{3.33e}$$
$$u_2 = u_C \tag{3.33f}$$
$$u_L = u_1 + u_2 \tag{3.33g}$$
$$i_0 = i_1 + i_L \tag{3.33h}$$
$$i_C = i_1 - i_2 \tag{3.33i}$$

Since most CSSL's allow us to specify auxiliary *algebraic equations* in addition to the *state equations*, and since they usually provide for an *equation sorter*, the above set of equations represents a perfectly good CSSL dynamic model description.

3.6 Algebraic Loops

Let us now consider the slightly modified circuit depicted in Fig.3.12:

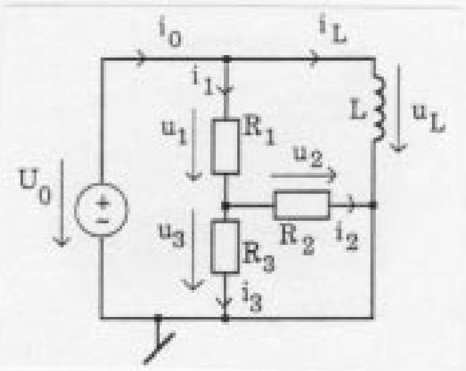

Figure 3.12. Another passive circuit with all variables named

In this example, we have introduced the following nine unknowns: u_1, u_2, u_3, u_L, i_1, i_2, i_3, i_L, and i_0.

We can go ahead and write four branch equations and five mesh and/or node equations as before.

$$u_1 = R_1 * i_1 \tag{3.34a}$$
$$u_2 = R_2 * i_2 \tag{3.34b}$$
$$u_3 = R_3 * i_3 \tag{3.34c}$$
$$u_L = L * \frac{di_L}{dt} \tag{3.34d}$$
$$U_0 = u_1 + u_3 \tag{3.34e}$$
$$u_3 = u_2 \tag{3.34f}$$
$$u_L = u_1 + u_2 \tag{3.34g}$$
$$i_0 = i_1 + i_L \tag{3.34h}$$
$$i_1 = i_2 + i_3 \tag{3.34i}$$

We try to solve these equations for the appropriate variables using the previously introduced recipe. However this time, the solution is not unique. After marking the first three equations, we are stuck:

$$u_1 = R_1 * i_1 \tag{3.35a}$$
$$u_2 = R_2 * i_2 \tag{3.35b}$$
$$u_3 = R_3 * i_3 \tag{3.35c}$$
$$u_L = L * [\frac{di_L}{dt}] \tag{3.35d}$$
$$U_0 = u_1 + u_3 \tag{3.35e}$$
$$u_3 = u_2 \tag{3.35f}$$
$$[u_L] = u_1 + u_2 \tag{3.35g}$$
$$[i_0] = i_1 + i_L \tag{3.35h}$$
$$i_1 = i_2 + i_3 \tag{3.35i}$$

At this point, all remaining equations contain at least two unknowns, and all remaining unknowns appear in at least two equations. We now have to make a *choice*. We can do this in an arbitrary fashion. For example, we could decide to solve eq(3.35e) for variable u_3. From then on, everything else will follow as before, and we obtain the following set of equations:

$$[u_1] = R_1 * i_1 \tag{3.36a}$$
$$u_2 = R_2 * [i_2] \tag{3.36b}$$
$$u_3 = R_3 * [i_3] \tag{3.36c}$$
$$u_L = L * [\frac{di_L}{dt}] \tag{3.36d}$$
$$U_0 = u_1 + [u_3] \tag{3.36e}$$
$$u_3 = [u_2] \tag{3.36f}$$
$$[u_L] = u_1 + u_2 \tag{3.36g}$$
$$[i_0] = i_1 + i_L \tag{3.36h}$$
$$[i_1] = i_2 + i_3 \tag{3.36i}$$

However, the fact that we had to make a choice invariably results in an *algebraic loop.* Let us rearrange the equations, and let us try to recognize the resulting algebraic loop.

$$u_1 = R_1 * i_1 \tag{3.37a}$$
$$i_2 = u_2 / R_2 \tag{3.37b}$$
$$i_3 = u_3 / R_3 \tag{3.37c}$$
$$\frac{di_L}{dt} = u_L / L \tag{3.37d}$$
$$u_3 = U_0 - u_1 \tag{3.37e}$$
$$u_2 = u_3 \tag{3.37f}$$
$$u_L = u_1 + u_2 \tag{3.37g}$$
$$i_0 = i_1 + i_L \tag{3.37h}$$
$$i_1 = i_2 + i_3 \tag{3.37i}$$

In order to compute u_3 from eq(3.37e), we need knowledge of u_1. However, in order to compute u_1 from eq(3.37a), we need knowledge of i_1. In order to compute i_1 from eq(3.37i), we need knowledge of i_3. Finally, in order to compute i_3 from eq(3.37c), we need knowledge of u_3 which closes the *algebraic loop.*

Unfortunately, algebraic loops are extremely common in electrical circuits, and this is the most serious drawback of the above outlined technique.

3.7 Structural Singularities

Let us now look at yet another circuit as depicted in Fig.3.13.

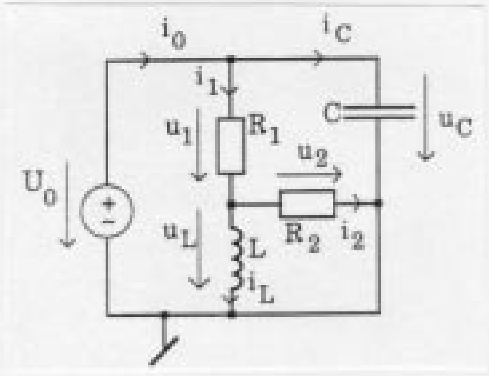

Figure 3.13. Yet another passive circuit with all variables named

In this example, we have introduced the following nine unknowns: u_1, u_2, u_C, u_L, i_1, i_2, i_C, i_L, and i_0.

We go ahead and write four branch equations and five mesh and/or node equations as before.

$$u_1 = R_1 * i_1 \tag{3.38a}$$
$$u_2 = R_2 * i_2 \tag{3.38b}$$
$$u_L = L * \frac{di_L}{dt} \tag{3.38c}$$
$$i_C = C * \frac{du_C}{dt} \tag{3.38d}$$
$$U_0 = u_1 + u_L \tag{3.38e}$$
$$u_L = u_2 \tag{3.38f}$$
$$u_C = u_1 + u_2 \tag{3.38g}$$
$$i_0 = i_1 + i_C \tag{3.38h}$$
$$i_1 = i_2 + i_L \tag{3.38i}$$

We try to solve these equations for the appropriate variables using the same recipe. However this time, the problem is *overspecified*. This situation is frequently referred to as a *structural singularity*. The term stems from the equivalent situation as it occurs in mechanical system modeling.

We are stuck after the first three equations have been utilized:

$$u_1 = R_1 * i_1 \tag{3.39a}$$
$$u_2 = R_2 * i_2 \tag{3.39b}$$
$$u_L = L * [\frac{di_L}{dt}] \tag{3.39c}$$
$$i_C = C * [\frac{du_C}{dt}] \tag{3.39d}$$
$$U_0 = u_1 + u_L \tag{3.39e}$$
$$u_L = u_2 \tag{3.39f}$$
$$u_C = u_1 + u_2 \tag{3.39g}$$
$$[i_0] = i_1 + i_C \tag{3.39h}$$
$$i_1 = i_2 + i_L \tag{3.39i}$$

At this point, we have no equation left to compute i_C. Obviously, we cannot use u_C as a state variable since we need that equation to compute i_C. We thus must revise our strategy, solve eq(3.39d) for i_C rather than for u_C, and continue from there. Let us see what happens now:

$$u_1 = R_1 * i_1 \tag{3.40a}$$
$$u_2 = R_2 * i_2 \tag{3.40b}$$
$$u_L = L * [\frac{di_L}{dt}] \tag{3.40c}$$
$$[i_C] = C * \frac{du_C}{dt} \tag{3.40d}$$
$$U_0 = u_1 + u_L \tag{3.40e}$$
$$u_L = u_2 \tag{3.40f}$$
$$[u_C] = u_1 + u_2 \tag{3.40g}$$
$$[i_0] = i_1 + i_C \tag{3.40h}$$
$$i_1 = i_2 + i_3 \tag{3.40i}$$

At this point, we are left with an algebraic loop as before.

3.8 Disadvantages of State–Space Models

In the last example, we could not avoid leaving one of the differential terms on the right hand side of the equal sign. Whenever we face the situation of having no equation left to solve for a particular unknown, or being left with an unused equation that contains only known variables (assuming that we didn't choose linearly dependent equations right from the beginning), we are confronted with a *degenerate system* (a so-called structural singularity). In our example, the system really is of first order and not of second order, and the additional differentiator is a *true differentiator* which cannot be eliminated from the circuit. Such systems do not have a state-space model. The best that we can hope for (and in a linear system, we can always achieve this) is to move the differentiation operator out of the integration loop into the output equation. We then end up with a *generalized state-space model* of the form:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{3.41a}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}(s)\mathbf{u} \tag{3.41b}$$

Algebraic loops and structural singularities are serious problems that make the derivation of state-space models for electrical circuits difficult if not impossible. For this reason, the approach is not commonly used in today's conventional circuit simulators. However, the approach has its beauties, and, if successful, can speed up the run-time execution of the resulting model quite dramatically. Several techniques exist to reduce algebraic loops and structural singularities automatically, and we shall discuss some of these techniques in Chapter 5 of this text. However, whether or not algebraic loops and/or structural singularities can *always* be removed in an automated (i.e. algorithmic) fashion, is still an unanswered question. Therefore, the development of such algorithms must be considered open research.

3.9 Summary

We have introduced the two standard techniques used in the analysis of electrical circuitry, the mesh equation approach and the node equation approach. More details on the implementation of these techniques in modern circuit simulators (such as SPICE) will be presented in Chapter 6 of this text. Details of the numerical techniques required to simulate these types of models (using *implicit differentiation*) are discussed in the second volume. We have also shown an alternative approach to circuit analysis, an approach that leads directly to a state-space description. However, we have shown that this route is quite problematic due to algebraic loops and structural singularities which occur frequently in electrical circuits. More details about these problems and how we deal with them are presented in Chapter 5 of this text, together with a tool (DYMOLA) which helps us automate the discussed algorithms.

References

[3.1] P. R. Bélanger, E. L. Adler, and N. C. Rumin (1985), *Introduction to Circuits with Electronics: An Integrated Approach*, Holt, Rinehart and Winston, Inc., New York.

[3.2] Leonard S. Bobrow (1981), *Elementary Linear Circuit Analysis*, Holt, Rinehart and Winston, Inc., New York.

[3.3] Lawrence P. Huelsman (1984), *Basic Circuit Theory*, Second Edition, Prentice–Hall, Englewood Cliffs, N.J.

[3.4] David E. Johnson, John L. Hilburn, and John R. Johnson (1978), *Basic Electric Circuit Analysis*, Prentice–Hall, Englewood Cliffs, N.J.

[3.5] Granino A. Korn (1966), *Random–Process Simulation and Measurements*, McGraw–Hill, New York.

Homework Problems

[H3.1] Choosing between Mesh and Node Equations

Fig.H3.1 shows a simple passive circuit. The circuit contains one *dependent current source*. The current i_4 is at all times proportional to the voltage v_3. The proportionality factor is $4\ A\ V^{-1}$.

Determine how many mesh equations (node equations) will be needed to describe this circuit.

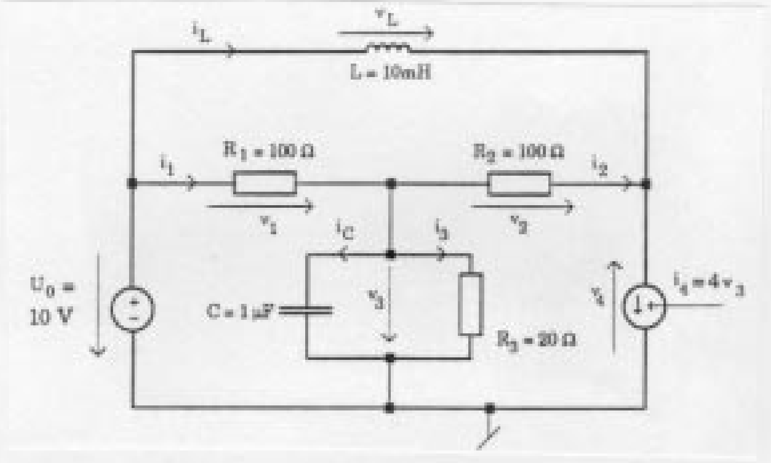

Figure H3.1. Circuit diagram of a simple passive circuit

[H3.2] Mesh Equations

For the circuit of Fig.H3.1, replace all storage elements by equivalent impedances, choose an appropriate tree, and determine a consistent set of mesh equations. Present these equations in a matrix form.

[H3.3] Node Equations

For the circuit of Fig.H3.1, reduce the independent voltage source to a set of current sources, replace all storage elements by equivalent admittances, choose an appropriate tree, and determine a consistent set of node equations. Present these equations in a matrix form.

[H3.4] CSSL Model

For the circuit of Fig.H3.1, use the state–space modeling approach to determine a consistent set of simulation equations. Code these equations in any CSSL, and simulate the system over 50 *μsec*.

[H3.5] Linear State–Space Model

From the simulation equations of hw(H3.4), eliminate all auxiliary variables, and write the resulting state equations in a matrix form of the type

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} + \mathbf{b} \cdot u \qquad (H3.5a)$$
$$y = \mathbf{C} \cdot \mathbf{x} + \mathbf{d} \cdot u \qquad (H3.5b)$$

where the output vector consists of the variables v_3 and i_C, and where the single input is the independent voltage source U_0.

Use CTRL–C (or MATLAB) to determine the eigen–modi of this system (which are the eigenvalues of the **A** matrix). Choose the final time of the simulation to be minus three times the inverse of the slower of the two eigen modi.

Simulate a step response of this system using CTRL–C (MATLAB) directly. For this purpose, it is necessary to create a *time base*, i.e., a vector of time values at which we wish to sample the simulated trajectories. This is accomplished with the statement:

```
[>   t = 0 : tmx/1000 : tmx;
```

which generates a vector of length 1001 containing numbers that are equidistantly spaced between 0.0 and *tmx*. This vector contains the communication points. Next, we need to generate the input signal sampled at the communication points. Since we want to simulate a step response, we can create a vector of length 1001 each element of which is 1.0. This can be achieved with the statement:

```
[>   u = U0 * ONES(t);
```

Finally, we can apply an initial condition, and simulate the system over time, using the CTRL–C statements:

```
[>   x0 = [v30; iL0];
[>   SIMU('IC', x0)
[>   y = SIMU(a, b, c, d, u, t);
```

The MATLAB solution looks very similar.

Determine analytically the steady–state value of the output vector to step input. (At steady–state, all derivatives have died out, and thus, $\dot{\mathbf{x}}_{ss} = 0$. Compare the analytically found value with the numerically found value as a means to validate your simulation.

[H3.6] Structural Singularity

Fig.H3.6 shows another simple passive circuit. Use the state–space modeling technique to determine a set of simulation equations. You will notice that this circuit exhibits a structural singularity.

Prove that the two inductive currents are linearly dependent on each other, and thus, do not qualify as two separate state variables.

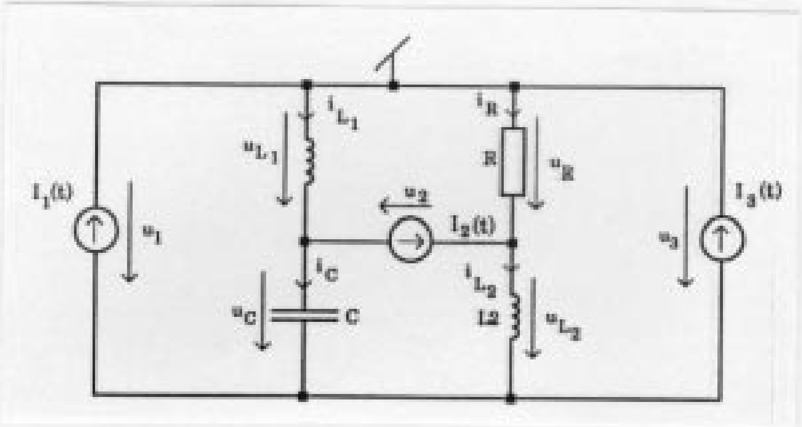

Figure H3.6. Circuit diagram of another simple passive circuit

[H3.7] Passive Filter

Fig.H3.7 shows a simple passive filter.

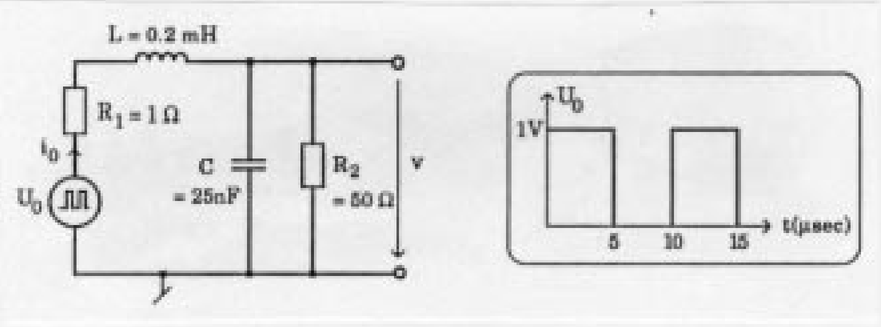

Figure H3.7. Circuit diagram of a simple passive filter

Use the state–space modeling approach to derive a set of first order differential equations to describe this system. Use any CSSL to simulate the system over 50 *μsec*, and display the resulting trajectories for u_0, i_0, and u_C. Use the step–function approach to generate the square–wave voltage source U_0.

[H3.8] Connecting Power to an Unloaded Power Line

Fig.H3.8a shows an unloaded and initially unenergized power line that must be brought on–line.

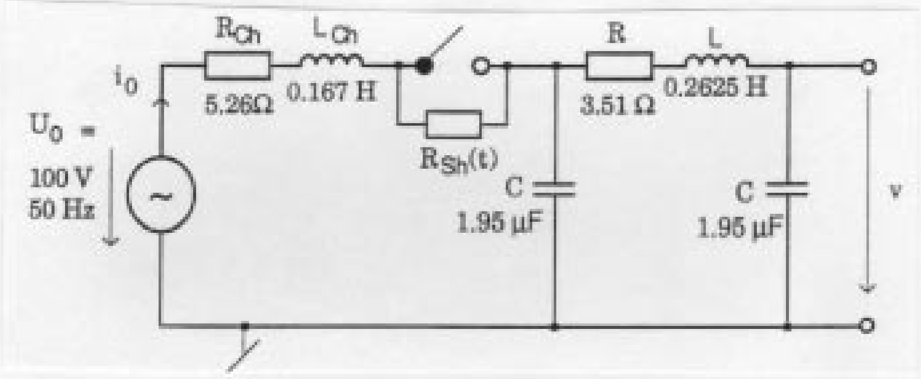

Figure H3.8a. Circuit diagram of a power line

The AC voltage source is connected to the power line which has a characteristic impedance of $R_{Ch} + j \cdot L_{Ch}$. The power line itself is represented through a single π–element.

The aim of this experiment is to determine how we can minimize the overshoot on the power line. For this purpose, we shall connect the voltage source to the power line (by closing the switch) at different phase angles. We shall try seven different phase angles equidistantly spaced between 0° and 180°.

Also, we want to try whether a shunt resistor may help suppress the overshoot on the power line. For this purpose, we introduce a time–varying shunt resistor R_{Sh} as shown in Fig.H3.8b.

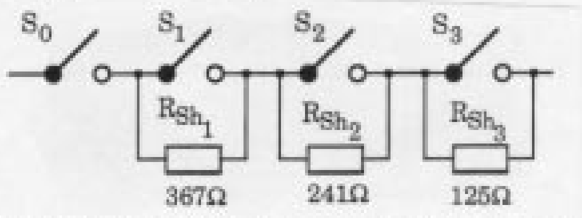

Figure H3.8b. Circuit diagram of the shunt resistor

The shunt resistor R_{Sh} is placed in parallel with the main switch. It contains itself four switches which are being closed at various time instants: the switch S_0 closes at time zero, S_1 closes after 3.5 *msec*, S_2 closes after 9 *msec*, and S_3 closes after 18 *msec*.

Simulate the system with and without shunt resistor for all seven phase angles, and plot u_2 on two separate graphs (seven curves per graph). Use the state-space approach for modeling. The shunt resistor is easiest coded as a Fortran subroutine.

[H3.9]* Resonance Circuit

Fig.H3.9 shows a simple resonance circuit.

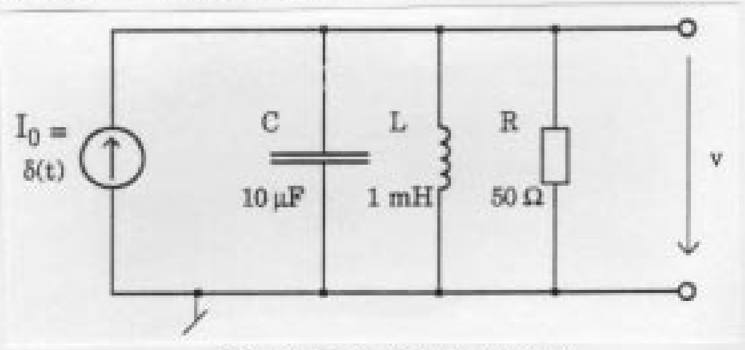

Figure H3.9. Resonance circuit

We want to analyze the response of this circuit to a Dirac current impulse:

$$i_0(t) = \delta(t) \qquad (H9.3a)$$

For this purpose, we perform two separate simulation experiments.

In the first experiment, we approximate the Dirac impulse by a rectangular pulse of length δt. Choose the height of the pulse such that the identity:

$$\int_{-\infty}^{\infty} \delta(t)\, dt = 1 \qquad (H3.9b)$$

is preserved. Repeat this experiment for three different values of δt, namely: $\delta t = 6$ *μsec*, $\delta t = 0.6$ *μsec*, and $\delta t = 0.06$ *μsec*. Simulate the system for a duration of $t_{max} = 2$ *msec*. Use a fixed step integration algorithm with the step size $\Delta t = 0.01 \cdot \delta t$ up to the time δt, and a much increased step size of $\Delta t = 60$ *μsec* thereafter. Plot $v(t)$ on one graph for all three simulation runs. Use the state-space approach, and formulate the model in any of the CSSL's. Some languages (such as ACSL) don't provide for fixed step algorithms. In that case, simply set the communication interval accordingly. In ACSL, you can set the communication interval initially in a CINTERVAL declaration:

$$\textbf{cinterval}\ cint = 2.0E-5$$

This declares the variable *cint*. Modify *cint* immediately in the INITIAL section to:

$$cint = 0.01 * width$$

and declare a discrete event of type *change* to happen at time *width* ($= \delta t$):

schedule *change* **.at.** *width*

This statement will force ACSL to enter a DISCRETE block (to be coded as part of the DYNAMIC block, but outside the DERIVATIVE block) by the name of *change* in which you can modify the communication interval:

```
DISCRETE change
   cint = 60.0E − 6
END $ "of DISCRETE change"
```

The current I_0 can, of course, be modified simultaneously.

In the second experiment, we shall notice that this is a linear system which can be written in the form:

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} + \mathbf{b} \cdot u \qquad (H3.9c)$$

$$y = \mathbf{c}' \cdot \mathbf{x} + d \cdot u \qquad (H3.9d)$$

where the single input u is the current source $i_0(t)$, and the single output y is the voltage $v(t)$. We can further notice that any such system can be analytically solved. The analytical solution is:

$$y(t) = \mathbf{c}' \exp(\mathbf{A}t)\mathbf{x}_0 + \mathbf{c}' \int_0^t \exp(\mathbf{A}(t-\tau))\mathbf{b}u(\tau)d\tau \qquad (H3.9e)$$

and since $u(t) = \delta(t)$, we can use the sifting property of the Dirac distribution to evaluate the integral. This allows us to reformulate the given problem which has a Dirac input and no initial condition as another equivalent problem which has no input but a non–vanishing initial condition. Determine what the equivalent initial condition has to be in terms of the matrices $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}'$, and d. Simulate this modified problem, and compare the results to those obtained from the previous simulations. It may be easiest to simulate this linear time–invariant problem in CTRL–C (or MATLAB) directly.