$$x_1^{(1)}(t) = x_2(t) \tag{3.69}$$

$$x_2^{(1)}(t) = -(b/J)x_2(t) + (a/J)x_3(t) - T_L(t)/J \tag{3.70}$$

$$x_3^{(1)}(t) = -(a/L)x_2(t) - [(R + R_0)/L]x_3(t) + (K_A/L)e(t) \tag{3.71}$$

and the output equations are

$$\theta(t) = x_1(t) \tag{3.72}$$

$$\theta^{(1)(t)} = x_2(t) \tag{3.73}$$

In matrix form, the state-equation and output equations are written as Eqs. (3.33) and (3.34), respectively, with the state-vector, $\mathbf{x}(t) = [x_1(t); x_2(t); x_3(t)]^T$ and the following coefficient matrices:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -b/J & a/J \\ 0 & -a/L & -(R+R_0)/L \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & -1/J \\ K_A/L & 0 \end{bmatrix};$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}; \quad \mathbf{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{3.74}$$

3.2 Linear Transformation of State-Space Representations

Since the state-space representation of a system is not unique, we can always find another state-space representation for the same system by the use of a *state transformation*. State transformation refers to the act of producing another state-space representation, starting from a given state-space representation. If a system is linear, the state-space representations also are linear, and the state transformation is a *linear transformation* in which the original state-vector is *pre-multiplied* by a *constant transformation matrix* yielding a new state-vector. Suppose $\mathbf{T}$ is such a *transformation matrix* for a linear system described by Eqs. (3.33) and (3.34). Let us find the new state-space representation in terms of $\mathbf{T}$ and the coefficient matrices, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$. The transformed state-vector, $\mathbf{x}'(t)$, is expressed as follows:

$$\mathbf{x}'(t) = \mathbf{T}\mathbf{x}(t) \tag{3.75}$$

Equation (3.75) is called a *linear state-transformation* with transformation matrix, $\mathbf{T}$. Note that for a system of order n, $\mathbf{T}$ must be a square matrix of size $(n \times n)$, because order of the system remains unchanged in the transformation from $\mathbf{x}(t)$ to $\mathbf{x}'(t)$. Let us assume that it is possible to transform the new state-vector, $\mathbf{x}'(t)$, back to the original state-vector, $\mathbf{x}(t)$, with the use of the following *inverse transformation*:

$$\mathbf{x}(t) = \mathbf{T}^{-1}\mathbf{x}'(t) \tag{3.76}$$

Equation (3.76) requires that the inverse of the transformation matrix, $\mathbf{T}^{-1}$, should *exist* (in other words, $\mathbf{T}$ should be *nonsingular*). Equation (3.70) is obtained by *pre-multiplying* both sides of Eq. (3.75) by $\mathbf{T}^{-1}$, and noting that $\mathbf{T}\ \mathbf{T}^{-1} = \mathbf{I}$. To find the transformed state-equation, let us differentiate Eq. (3.76) with time and substitute the result, $\mathbf{x}^{(1)}(t) = \mathbf{T}^{-1}\mathbf{x}'^{(1)}(t)$, along with Eq. (3.76), into Eq. (3.33), thereby yielding

$$\mathbf{T}^{-1}\mathbf{x}'^{(1)}(t) = \mathbf{A}\mathbf{T}^{-1}\mathbf{x}'(t) + \mathbf{B}\mathbf{u}(t) \tag{3.77}$$

Pre-multiplying both sides of Eq. (3.77) by $\mathbf{T}$, we get the transformed state-equation as follows:

$$\mathbf{x}'^{(1)}(t) = \mathbf{T}\mathbf{A}\mathbf{T}^{-1}\mathbf{x}'(t) + \mathbf{T}\mathbf{B}\mathbf{u}(t) \tag{3.78}$$

We can write the transformed state-equation Eq. (3.78) in terms of the new coefficient matrices $\mathbf{A}'$, $\mathbf{B}'$, $\mathbf{C}'$, $\mathbf{D}'$, as follows:

$$\mathbf{x}'^{(1)}(t) = \mathbf{A}'\mathbf{x}'(t) + \mathbf{B}'\mathbf{u}(t) \tag{3.79}$$

where $\mathbf{A}' = \mathbf{T}\mathbf{A}\mathbf{T}^{-1}$, and $\mathbf{B}' = \mathbf{T}\mathbf{B}$. Similarly, substituting Eq. (3.76) into Eq. (3.34) yields the following transformed output equation:

$$\mathbf{y}(t) = \mathbf{C}'\mathbf{x}'(t) + \mathbf{D}'\mathbf{u}(t) \tag{3.80}$$

where $\mathbf{C}' = \mathbf{C}\mathbf{T}^{-1}$, and $\mathbf{D}' = \mathbf{D}$.

There are several reasons for transforming one state-space representation into another, such as the utility of a particular form of state-equations in control system design (the *controller* or *observer companion form*), the requirement of transforming the state variables into those that are *physically meaningful* in order to implement a control system, and sometimes, the need to *decouple* the state-equations so that they can be easily solved. We will come across such state-transformations in the following chapters.

Example 3.8

We had obtained two different state-space representations for the same electrical network in Examples 3.4 and 3.5. Let us find the state-transformation matrix, $\mathbf{T}$, which transforms the state-space representation given by Eqs. (3.42) and (3.43) to that given by Eqs. (3.51) and (3.52), respectively. In this case, the original state-vector is $\mathbf{x}(t) = [i_2(t); i_2^{(1)}(t) - R_3e(t)/\{L(R_1 + R_3)\}]^T$, whereas the transformed state-vector is $\mathbf{x}'(t) = [LC\{e(t) - i_2(t)(R_1R_3 + R_1R_2 + R_2R_3)/R_3 - Li_2^{(1)}(t)(R_1 + R_3)/R_3\}; i_2(t)L(R_1 + R_3)/R_3]^T$. The state-transformation matrix, $\mathbf{T}$, is of size (2×2). From Eq. (3.69), it follows that

$$\begin{bmatrix} LC\{e(t) - i_2(t)(R_1R_3 + R_1R_2 + R_2R_3)/R_3 - Li_2^{(1)}(t)(R_1 + R_3)/R_3\} \\ i_2(t)L(R_1 + R_3)/R_3 \end{bmatrix}$$

$$= \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \begin{bmatrix} i_2(t) \\ i_2^{(1)}(t) - R_3e(t)/\{L(R_1 + R_3)\} \end{bmatrix} \tag{3.81}$$

where T_{11}, T_{12}, T_{21}, and T_{22} are the *unknown* elements of **T**. We can write the following *two* scalar equations out of the matrix equation, Eq. (3.81):

$$LCe(t) - LCi_2(t)(R_1R_3 + R_1R_2 + R_2R_3)/R_3 - L^2Ci_2^{(1)}(t)(R_1 + R_3)/R_3$$
$$= T_{11}i_2(t) + T_{12}[i_2^{(1)}(t) - R_3e(t)/\{L(R_1 + R_3)\}] \tag{3.82}$$

$$i_2(t)L(R_1 + R_3)/R_3 = T_{21}i_2(t) + T_{22}[i_2^{(1)}(t) - R_3e(t)/\{L(R_1 + R_3)\}] \tag{3.83}$$

Equating the coefficients of $i_2(t)$ on both sides of Eq. (3.82), we get

$$T_{11} = -LC(R_1R_3 + R_1R_2 + R_2R_3)/R_3 \tag{3.84}$$

Equating the coefficients of $e(t)$ on both sides of Eq. (3.82), we get

$$T_{12} = -L^2C(R_1 + R_3)/R_3 \tag{3.85}$$

Note that the same result as Eq. (3.85) is obtained if we equate the coefficients of $i_2^{(1)}(t)$ on both sides of Eq. (3.82). Similarly, equating the coefficients of corresponding variables on both sides of Eq. (3.83) we get

$$T_{21} = L(R_1 + R_3)/R_3 \tag{3.86}$$

and

$$T_{22} = 0 \tag{3.87}$$

Therefore, the required state-transformation matrix is

$$\mathbf{T} = \begin{bmatrix} -LC(R_1R_3 + R_1R_2 + R_2R_3)/R_3 & -L^2C(R_1 + R_3)/R_3 \\ L(R_1 + R_3)/R_3 & 0 \end{bmatrix} \tag{3.88}$$

With the transformation matrix of Eq. (3.88), you may verify that the state-space coefficient matrices of Example 3.5 are related to those of Example 3.4 according to Eqs. (3.79) and (3.80).

Example 3.9

For a linear, time-invariant state-space representation, the coefficient matrices are as follows:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ -3 & -1 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \mathbf{C} = [1 \quad 2]; \quad \mathbf{D} = [0 \quad 0] \tag{3.89}$$

If the state-transformation matrix is the following:

$$\mathbf{T} = \begin{bmatrix} -1 & 1 \\ -1 & -1 \end{bmatrix} \tag{3.90}$$

let us find the transformed state-space representation. The first thing to do is to check whether **T** is singular. The determinant of **T**, $|\mathbf{T}| = 2$. Hence, **T** is nonsingular and its inverse can be calculated as follows (for the definitions of *determinant*, *inverse*, and other matrix operations see Appendix B):

$$\mathbf{T}^{-1} = \mathrm{adj}(\mathbf{T})/|\mathbf{T}| = (1/2)\begin{bmatrix} -1 & 1 \\ -1 & -1 \end{bmatrix}^T = (1/2)\begin{bmatrix} -1 & -1 \\ 1 & -1 \end{bmatrix} \tag{3.91}$$

Then the transformed state coefficient matrices, $\mathbf{A}'$, $\mathbf{B}'$, $\mathbf{C}'$, $\mathbf{D}'$, of Eqs. (3.79) and (3.80) are then calculated as follows:

$$\mathbf{A}' = \mathbf{TAT}^{-1} = (1/2)\begin{bmatrix} -1 & 1 \\ -1 & -1 \end{bmatrix}\begin{bmatrix} 1 & 2 \\ -3 & -1 \end{bmatrix}\begin{bmatrix} -1 & -1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1/2 & 7/2 \\ -3/2 & -1/2 \end{bmatrix} \tag{3.92}$$

$$\mathbf{B}' = \mathbf{TB} = \begin{bmatrix} -1 & 1 \\ -1 & -1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ -1 & -1 \end{bmatrix} \tag{3.93}$$

$$\mathbf{C}' = \mathbf{CT}^{-1} = (1/2)[\,1 \quad 2\,]\begin{bmatrix} -1 & -1 \\ 1 & -1 \end{bmatrix} = [\,1/2 \quad -3/2\,] \tag{3.94}$$

$$\mathbf{D}' = \mathbf{D} = [\,0 \quad 0\,] \tag{3.95}$$

It will be appreciated by anyone who has tried to invert, multiply, or find the determinant of matrices of size larger than (2×2) by hand, that doing so can be a tedious process. Such calculations are easily done using MATLAB, as the following example will illustrate.

Example 3.10

Consider the following state-space representation of the linearized longitudinal dynamics of an aircraft depicted in Figure 2.25:

$$\begin{bmatrix} v^{(1)}(t) \\ \alpha^{(1)}(t) \\ \theta^{(1)}(t) \\ q^{(1)}(t) \end{bmatrix} = \begin{bmatrix} -0.045 & 0.036 & -32 & -2 \\ -0.4 & -3 & -0.3 & 250 \\ 0 & 0 & 0 & 1 \\ 0.002 & -0.04 & 0.001 & -3.2 \end{bmatrix}\begin{bmatrix} v(t) \\ \alpha(t) \\ \theta(t) \\ q(t) \end{bmatrix} + \begin{bmatrix} 0 & 0.1 \\ -30 & 0 \\ 0 & 0 \\ -10 & 0 \end{bmatrix}\begin{bmatrix} \delta(t) \\ \mu(t) \end{bmatrix} \tag{3.96}$$

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} v(t) \\ \alpha(t) \\ \theta(t) \\ q(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} \delta(t) \\ \mu(t) \end{bmatrix} \tag{3.97}$$

where the *elevator deflection*, $\delta(t)$ (Figure 2.25) and *throttle position*, $\mu(t)$ (not shown in Figure 2.25) are the two inputs, whereas the change in the *pitch angle*, $\theta(t)$, and the *pitch-rate*, $q(t) = \theta^{(1)}(t)$, are the two outputs. The state-vector selected to represent the dynamics in Eqs. (3.96) and (3.97) is $\mathbf{x}(t) = [v(t); \alpha(t); \theta(t); q(t)]^T$, where $v(t)$ represents a change in the *forward speed*, and $\alpha(t)$ is the change in the *angle of attack*. All the changes are measured from an initial equilibrium state of the aircraft given by $\mathbf{x}(0) = \mathbf{0}$. Let us transform the state-space representation using the following transformation matrix:

$$\mathbf{T} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ -3.5 & 1 & 0 & -1 \\ 0 & 0 & 2.2 & 3 \end{bmatrix} \tag{3.98}$$

You may verify that $\mathbf{T}$ is nonsingular by finding its determinant by hand, or using the MATLAB function *det*. The state-transformation can be easily carried out using the intrinsic MATLAB functions as follows:

```
>>A=[-0.045 0.036 -32 -2; -0.4 -3 -0.3 250; 0 0 0 1; 0.002 -0.04 0.001
  -3.2]; <enter>

>>B=[0 0.1;-30 0;0 0;-10 0]; C=[0 0 1 0; 0 0 0 1]; D=zeros(2,2); <enter>

>>T=[1 -1 0 0; 0 0 -2 0; -3.5 1 0 -1; 0 0 2.2 3]; <enter>

>>Aprime=T*A*inv(T), Bprime=T*B, Cprime=C*inv(T) <enter>

Aprime =

     -4.3924     -77.0473     -1.3564     -84.4521
     0           -0.7333      0           -0.6667
     4.4182      40.0456      1.3322      87.1774
     0.1656      -2.6981      0.0456      -2.4515

Bprime =
       30.0000       0.1000
       0             0
       -20.0000      -0.3500
       -30.0000      0

Cprime =
        0      -0.5000     0      0
        0      0.3667      0      0.3333
```

and $\mathbf{D}'$ is, of course, just $\mathbf{D}$. The transformed state coefficient matrices can be obtained in one step by using the MATLAB Control System Toolbox (CST) command *ss2ss*. First, a state-space LTI object is created using the function *ss* as follows:

```
>> sys1=ss(A,B,C,D) <enter>

a =
                 x1          x2          x3      x4
        x1       -0.045      0.036       -32     -2
        x2       -0.4        -3          -0.3    250
```

```
        x3      0         0        0      1
        x4      0.002    -0.04     0.001 -3.2

  b =
                u1        u2
        x1      0         0.1
        x2     -30        0
        x3      0         0
        x4     -10        0

  c =
                x1    x2    x3    x4
        y1      0     0     1     0
        y2      0     0     0     1

  d =
                u1    u2
        y1      0     0
        y2      0     0

Continuous-time model.
```

Then, the function *ss2ss* is used to transform the LTI object, *sys1*, to another state-space representation, *sys2*:

```
>>sys2 = ss2ss (sys1,T) <enter>

a =
                x1          x2          x3         x4
        x1     -4.3924     -77.047     -1.3564    -84.452
        x2      0          -0.73333     0         -0.66667
        x3      4.4182      40.046      1.3322     87.177
        x4      0.1656     -2.6981      0.0456    -2.4515

b =
                u1          u2
        x1      30          0.1
        x2      0           0
        x3     -20         -0.35
        x4     -30          0

c =
                x1          x2          x3         x4
        y1      0          -0.5         0          0
        y2      0           0.36667     0          0.33333

d =
                u1    u2
        y1      0     0
        y2      0     0

Continuous-time model.
```