Chapter 5

MATLAB APPLICATIONS: NUMERICAL SIMULATIONS OF DIFFERENTIAL EQUATIONS AND INTRODUCTION TO DYNAMIC SYSTEMS

5.1. Solution of Differential Equations and Dynamic Systems Fundamentals

To study real-world systems, one can use the MATLAB environment [1]. In particular, the dynamic systems are modeled using lumped-parameters and high-fidelity mathematical models given in the form of nonlinear differential (ordinary and partial) and difference equations [2 - 5]. These equations must be numerically or analytically solved, and the MATLAB environment offers the needed features. Then, the data-intensive analysis can be accomplished in MATLAB. The commonly used solvers to numerically solve ordinary nonlinear and linear differential equations are the `ode23`, `ode113`, `ode15S`, `ode23S`, `ode23T`, `ode23TB`, and `ode45` solvers. Below is the description of the `ode45` solver.

```
>> help ode45
 ODE45  Solve non-stiff differential equations, medium order method.
    [T,Y] = ODE45(ODEFUN,TSPAN,Y0) with TSPAN = [T0 TFINAL] integrates the
    system of differential equations y' = f(t,y) from time T0 to TFINAL with
    initial conditions Y0. Function ODEFUN(T,Y) must return a column vector
    corresponding to f(t,y). Each row in the solution array Y corresponds to
    a time returned in the column vector T. To obtain solutions at specific
    times T0,T1,...,TFINAL (all increasing or all decreasing), use
    TSPAN = [T0 T1 ... TFINAL].

    [T,Y] = ODE45(ODEFUN,TSPAN,Y0,OPTIONS) solves as above with default
    integration properties replaced by values in OPTIONS, an argument created
    with the ODESET function. See ODESET for details. Commonly used options
    are scalar relative error tolerance 'RelTol' (1e-3 by default) and vector
    of absolute error tolerances 'AbsTol' (all components 1e-6 by default).

    [T,Y] = ODE45(ODEFUN,TSPAN,Y0,OPTIONS,P1,P2...) passes the additional
    parameters P1,P2,... to the ODE function as ODEFUN(T,Y,P1,P2...), and to
    all functions specified in OPTIONS. Use OPTIONS = [] as a place holder if
    no options are set.

    ODE45 can solve problems M(t,y)*y' = f(t,y) with mass matrix M that is
    nonsingular. Use ODESET to set the 'Mass' property to a function MASS if
    MASS(T,Y) returns the value of the mass matrix. If the mass matrix is
    constant, the matrix can be used as the value of the 'Mass' option. If
    the mass matrix does not depend on the state variable Y and the function
    MASS is to be called with one input argument T, set 'MStateDependence' to
    'none'. ODE15S and ODE23T can solve problems with singular mass matrices.

    [T,Y,TE,YE,IE] = ODE45(ODEFUN,TSPAN,Y0,OPTIONS...) with the 'Events'
    property in OPTIONS set to a function EVENTS, solves as above while also
    finding where functions of (T,Y), called event functions, are zero. For
    each function you specify whether the integration is to terminate at a
    zero and whether the direction of the zero crossing matters. These are
    the three vectors returned by EVENTS: [VALUE,ISTERMINAL,DIRECTION] =
    EVENTS(T,Y). For the I-th event function: VALUE(I) is the value of the
    function, ISTERMINAL(I)=1 if the integration is to terminate at a zero of
    this event function and 0 otherwise. DIRECTION(I)=0 if all zeros are to
    be computed (the default), +1 if only zeros where the event function is
    increasing, and -1 if only zeros where the event function is
    decreasing. Output TE is a column vector of times at which events
```

```
occur. Rows of YE are the corresponding solutions, and indices in vector
IE specify which event occurred.

SOL = ODE45(ODEFUN,[T0 TFINAL],Y0...) returns a structure that can be
used with DEVAL to evaluate the solution at any point between T0 and
TFINAL. The steps chosen by ODE45 are returned in a row vector SOL.x.
For each I, the column SOL.y(:,I) contains the solution at SOL.x(I).
If events were detected, SOL.xe is a row vector of points at which events
occurred. Columns of SOL.ye are the corresponding solutions, and indices
in vector SOL.ie specify which event occurred. If a terminal event has
been detected, SOL.x(end) contains the end of the step at which the event
occurred. The exact point of the event is reported in SOL.xe(end).

Example
      [t,y]=ode45(@vdp1,[0 20],[2 0]);
      plot(t,y(:,1));
  solves the system y' = vdp1(t,y), using the default relative error
  tolerance 1e-3 and the default absolute tolerance of 1e-6 for each
  component, and plots the first component of the solution.

See also
    other ODE solvers:    ODE23, ODE113, ODE15S, ODE23S, ODE23T, ODE23TB
    options handling:     ODESET, ODEGET
    output functions:     ODEPLOT, ODEPHAS2, ODEPHAS3, ODEPRINT
    evaluating solution:  DEVAL
    ODE examples:         RIGIDODE, BALLODE, ORBITODE

NOTE:
  The interpretation of the first input argument of the ODE solvers and
  some properties available through ODESET have changed in this version
  of MATLAB. Although we still support the v5 syntax, any new
  functionality is available only with the new syntax. To see the v5
  help, type in the command line
  more on, type ode45, more off
```

The following examples illustrate the application of the MATLAB `ode45` solver.

MATLAB Illustrative Example.

The following set of two nonlinear differential equations, called the van der Pol equations,

$$\frac{dx_1(t)}{dt} = x_2,\ x_1(t_0) = x_{10},$$

$$\frac{dx_2(t)}{dt} = \mu\left[(1-x_1^2)x_2 - x_1\right],\ x_2(t_0) = x_{20},$$

has been used as an illustrative example to solve ordinary differential equations using different solvers over the last 18 years (the author integrated this MATLAB example into the engineering curriculum in 1985). Two m-files [1] to solve these differential equations are given below:

- MATLAB script with `ode15s` solver and plotting statements (file name: `vdpode.m`):

```
function  vdpode(MU)
%VDPODE  Parameterizable van der Pol equation (stiff for large MU).
%   For the default value of MU = 1000 the equation is in relaxation
%   oscillation, and the problem becomes very stiff. The limit cycle has
%   portions where the solution components change slowly and the problem is
%   quite stiff, alternating with regions of very sharp change where it is
%   not stiff (quasi-discontinuities). The initial conditions are close to an
%   area of slow change so as to test schemes for the selection of the
%   initial step size.
%
%   The subfunction J(T,Y,MU) returns the Jacobian matrix dF/dY evaluated
```

```
%   analytically at (T,Y). By default, the stiff solvers of the ODE Suite
%   approximate Jacobian matrices numerically. However, if the ODE Solver
%   property Jacobian is set to @J with ODESET, a solver calls the function
%   to obtain dF/dY. Providing the solvers with an analytic Jacobian is not
%   necessary, but it can improve the reliability and efficiency of
%   integration.
%
%   L. F. Shampine, Evaluation of a test set for stiff ODE solvers, ACM
%   Trans. Math. Soft., 7 (1981) pp. 409-420.
%
%   See also ODE15S, ODE23S, ODE23T, ODE23TB, ODESET, @.

%   Mark W. Reichelt and Lawrence F. Shampine, 3-23-94, 4-19-94
%   Copyright 1984-2002 The MathWorks, Inc.
%   $Revision: 1.18 $  $Date: 2002/04/08 20:04:56 $

if nargin < 1
  MU = 1000;      % default
end

tspan = [0; max(20,3*MU)];              % several periods
y0 = [2; 0];
options = odeset('Jacobian',@J);

[t,y] = ode15s(@f,tspan,y0,options,MU);

figure;
plot(t,y(:,1));
title(['Solution of van der Pol Equation, \mu = ' num2str(MU)]);
xlabel('time t');
ylabel('solution y_1');

axis([tspan(1) tspan(end) -2.5 2.5]);

% --------------------------------------------------------------------------
function dydt = f(t,y,mu)
dydt = [             y(2)
         mu*(1-y(1)^2)*y(2)-y(1) ];
% --------------------------------------------------------------------------

function dfdy = J(t,y,mu)
dfdy = [          0                  1
         -2*mu*y(1)*y(2)-1    mu*(1-y(1)^2) ];
```

- MATLAB script with a set of differential equations to be solved (file name: `vdp1000.m`):

```
function dydt = vdp1000(t,y)
%VDP1000  Evaluate the van der Pol ODEs for mu = 1000.
%
%   See also ODE15S, ODE23S, ODE23T, ODE23TB.

%   Jacek Kierzenka and Lawrence F. Shampine
%   Copyright 1984-2002 The MathWorks, Inc.
%   $Revision: 1.5 $  $Date: 2002/04/08 20:04:56 $

dydt = [y(2); 1000*(1-y(1)^2)*y(2)-y(1)];
```

Both files `vdpode.m` and `vdp1000.m` are in the particular MATLAB directory. Let these files be in the directory `cd c:\MATLAB6p5\toolbox\matlab\demos`. Then, to run these programs, we type in the Command Window

```
>> cd c:\MATLAB6p5\toolbox\matlab\demos
```

To perform numerical simulations, run the file `vdpode.m` by typing in the Command Window

```
>> vdpode
```

and pressing the Enter key. The resulting plot for the evolution of the state variable $x_1(t)$ is documented in Figure 5.1 (note that the initial conditions were assigned to be $x_0 = \begin{bmatrix} x_{10} \\ x_{20} \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$ and $\mu = 1000$). Please note that in the MathWorks `vdpode.m` file to solve ordinary differential equations, the solver `ode15s` is used, and the plotting statement is `plot(t,y(:,1))`.

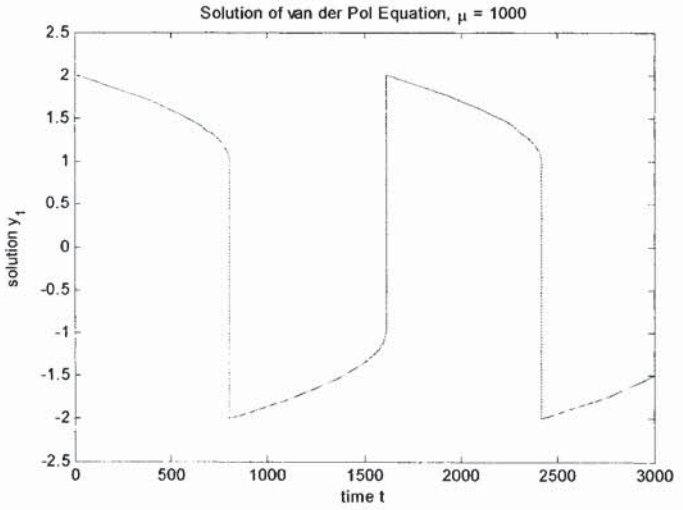


Figure 5.1. Dynamics of the state $x_1(t)$

The user can modify the file `vdpode.m`. For example, if we need to plot $x_1(t)$ and $x_2(t)$, as well as visualize the results plotting $x_1(t)$, $x_2(t)$ and t in three-dimensional plot (x_1, x_2, t), the following lines can be added to `vdpode.m` (the variable x_2 was divided by 100):

```
% Two-dimensional plot
plot(t,y(:,1),'-',t,y(:,2)/100,':');
xlabel('Time (seconds)');
title('Solution of van der Pol equation: x1 and x2/100');
pause
% 3-D plot w(y1,y2,t)
plot3(y(:,1),y(:,2)/100,t)
xlabel('x1'), ylabel('x2/100'), zlabel('time')
text(0,0,0,'0 Origin')
```

The resulting plots are illustrated in Figure 5.2.

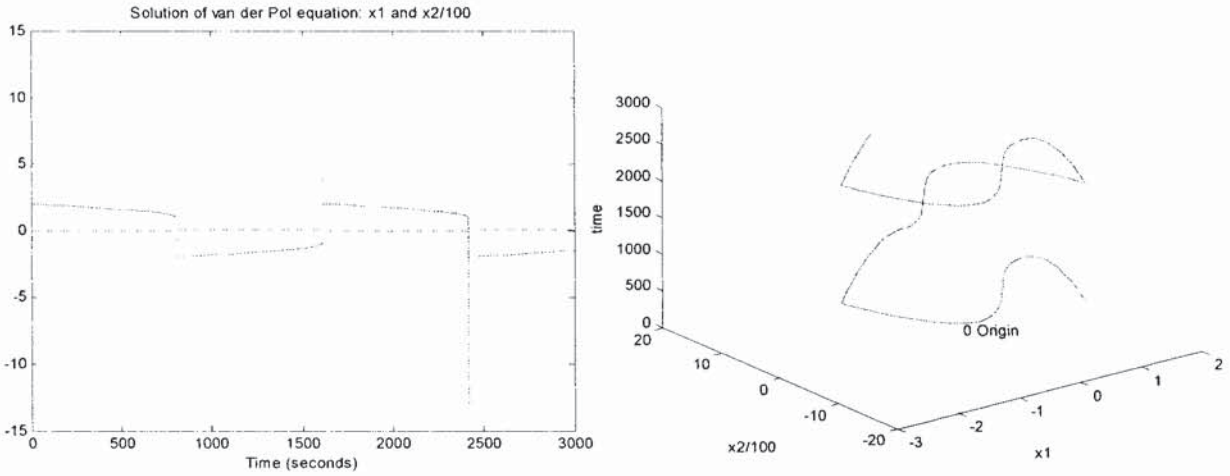


Figure 5.2. Evolution of the state variables using two- and tree-dimensional plots □

Example 5.1.1.

Numerically solve a system of highly nonlinear differential equations using the MATLAB `ode45` solver

$$\frac{dx_1(t)}{dt} = -15x_1 + 10|x_2| + 10x_1x_2x_3,\ x_1(t_0) = x_{10},$$

$$\frac{dx_2(t)}{dt} = -5x_1x_2 - \sin x_1 + x_2 - x_3,\ x_2(t_0) = x_{20},$$

$$\frac{dx_3(t)}{dt} = -5x_1x_2 + 10x_2\cos x_1 - 15x_3,\ x_3(t_0) = x_{30}.$$

Two m-files (`c5_1_1a.m` and `c5_1_1b.m`) are developed in order to numerically simulate this set of nonlinear differential equations. The initial conditions must be assigned, and let $x_0 = \begin{bmatrix} x_{10} \\ x_{20} \\ x_{30} \end{bmatrix} = \begin{bmatrix} 15 \\ -15 \\ 10 \end{bmatrix}$. The evolution of the state variables $x_1(t)$, $x_2(t)$, and $x_3(t)$ must be plotted. To illustrate the transient responses of $x_1(t)$, $x_2(t)$, and $x_3(t)$, the `plot` command is used. Three-dimensional graphics is also available and integrated in the first m-file. A three-dimensional plot is obtained using x_1, x_2, and x_3 as the variables by making use of `plot3`. Comments, which are not executed, appear after the % symbol. These comments explain particular steps in MATLAB scripts.

MATLAB script with `ode45` solver and plotting (two- and three-dimensional) statements using `plot` and `plot3` (`c5_1_1a.m`):

```
echo on; clear all
tspan=[0 3];          % initial and final time
y0=[15 -15 10]';      % initial conditions
[t,y]=ode45('c5_1_1b',tspan,y0);   %ode45 MATLAB solver
% Plot of the time history found by solving
% three differential equations assigned in the file c5_1_1b.m
plot(t,y(:,1),'--',t,y(:,2),'-',t,y(:,3),':');
xlabel('Time (seconds)');
title('Solution of Differential Equations: x1, x2 and x3');
pause
% 3-D plot w(y1,y2,y3)
```

```
plot3(y(:,1),y(:,2),y(:,3))
xlabel('x1'), ylabel('x2'), zlabel('x3')
text(15,-15,10,'x0 Initial')
text(0,0,0,'0 Origin')
v=axis
pause; disp('END')
```

MATLAB script with a set of differential equations to be solved (`c5_1_1b.m`):

```
function yprime = difer(t,y);
a11=-15; a12=10; a13=10; a21=-5; a22=-2; a31=-5; a32=10; a33=-15;
yprime=[a11*y(1,:)+a12*abs(y(2,:))+a13*y(1,:)*y(2,:)*y(3,:);...
a21*y(1,:)*y(2,:)+a22*sin(y(1,:))-y(2,:)+y(3,:);...
a31*y(1,:)*y(2,:)+a32*cos(y(1,:))*y(2,:)+a33*y(3,:)];
```

To calculate the transient dynamics and plot the transient dynamics, type in the Command window » `c5_1_1a` and press the Enter key. The resulting transient behavior and three-dimensional plot are documented in Figures 5.3 and 5.4.

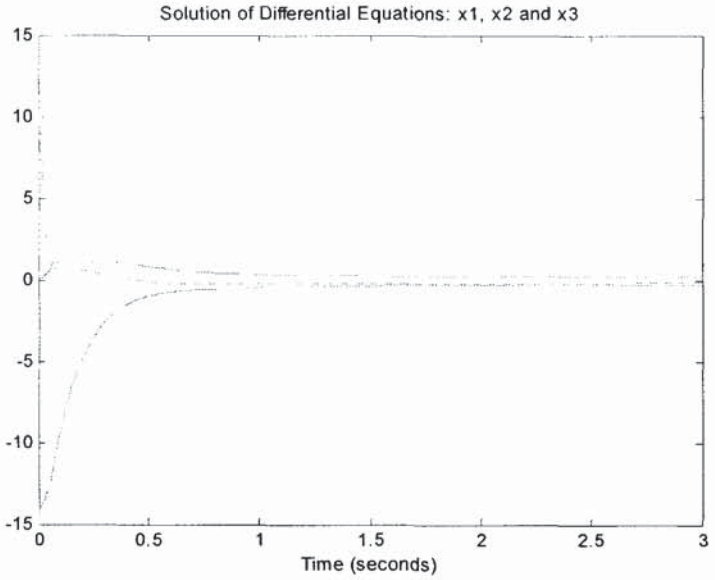


Figure 5.3. Evolution of the state variables, $x_0 = \begin{bmatrix} x_{10} \\ x_{20} \\ x_{30} \end{bmatrix} = \begin{bmatrix} 15 \\ -15 \\ 10 \end{bmatrix}$

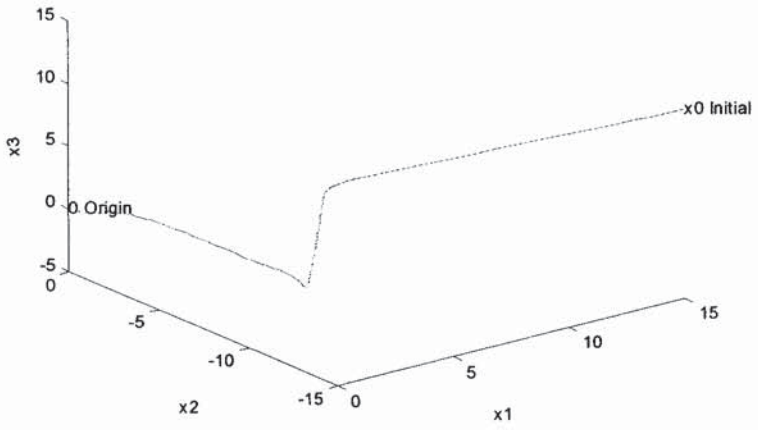


Figure 5.4. Three-dimensional plot □

A set of differential equations was assigned. However, to apply MATLAB, first mathematical models for real-word systems must be developed. Thus, numerical and analytical simulation and analysis of systems is a two-step process. Mathematical models depict the time-dependent mathematical relationships among the system's inputs, states, events, and outputs.

The Lagrange equations of motion, as well as Kirchhoff's and Newton's laws, can be used to develop mathematical models described by differential or difference equations. The real-world systems integrate many components, subsystems, and devices. Multivariable dynamic systems are studied with different levels of comprehensiveness. Consider the aircraft in Figure 5.5. In aircraft as well as in other flight vehicles (missiles, projectiles, rockets, spacecraft, etc.) and surface/undersea vehicles (ships, submarines, torpedoes, etc.), control surfaces are actuated by electromechanical actuators. Therefore, the actuator must be studied. These actuators are controlled by power amplifiers, and therefore the circuitry must be examined as well. Mechanical systems (rigid-body aircraft and actuators' torsional-mechanical dynamics) are modeled using Newtonian mechanics, the electromagnetics of electromechanical actuators are studied using Maxwell's equations, and the circuitry dynamics is usually modeled using Kirchhoff's laws [3, 4].

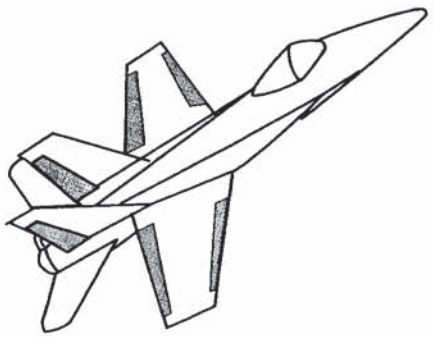


Figure 5.5. Aircraft

The aircraft outputs are the Euler angles θ, ϕ, and ψ. The reference inputs are the desired (assigned by the pilot or flight computer) Euler angles, which are denoted as r_θ, r_ϕ and r_ψ. For rigid-body aircraft, the longitudinal and lateral dynamics are modeled using the following state variables: the forward velocity v; the angle of attack α; the pitch rate q; the pitch angle θ; the sideslip angle β; the roll rate p; the yaw rate r; the roll angle ϕ; the yaw angle ψ. As was emphasized, the aircraft is controlled by displacing the control surfaces (right and left horizontal stabilizers, right and left leading- and trailing-edge flaps, right and left rudders). That is, a multi-input/multi-output dynamic system (e.g., aircraft, submarines, cars, etc.) must be simulated and analyzed in the MATLAB environment.

Having introduced the basics in flight control, the MATLAB demo offers a great number of illustrative examples which should be used. For example, the numerical simulations for the F-14 fighter are performed as illustrated in Figure 5.6.

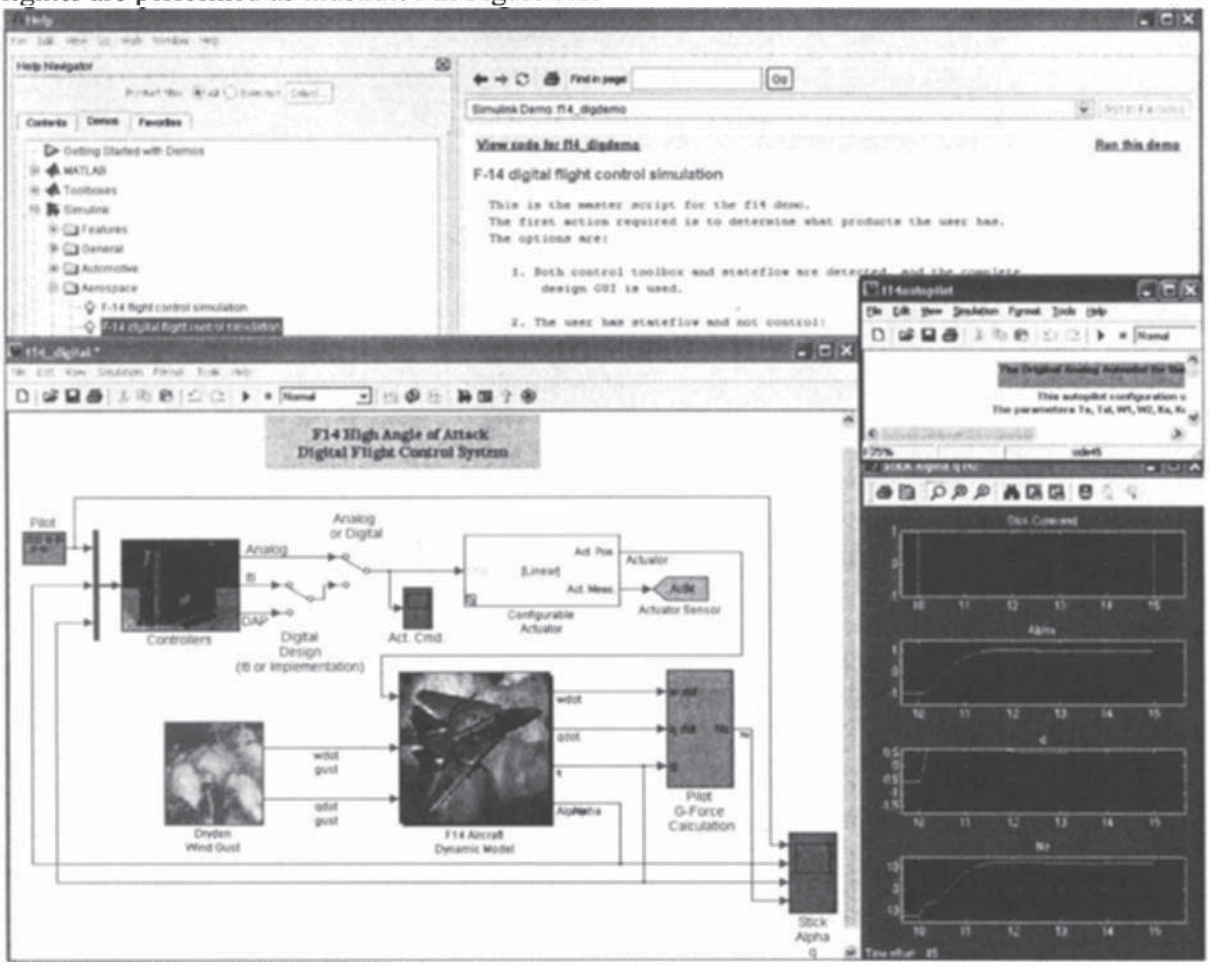


Figure 5.6. Simulations of the F-14 fighter using MATLAB demo

It was emphasized that the aircraft is controlled by changing the angular displacement of the flight control surfaces, and servo-systems are used to actuate ailerons, elevators, canards, flaps, rudders, stabilizers, tips, and other control surfaces. To deflect ailerons, canards, fins, flaps, rudders, and stabilizers, hydraulic and electric motors have been applied. A direct-drive control

surface *servo* driven by an electric motor is shown in Figure 5.7. Using the reference signal (the command angular displacement of the control surface), measured current in the phase windings i, mechanical angular velocity ω_{rm}, and actual mechanical angular displacement θ_{rm}, the controller develops signal-level signals which drive high-frequency switches of the power amplifier. The magnitude and frequency of the applied voltage to the phase winding is controlled by the PWM power amplifier (see Figure 5.7).

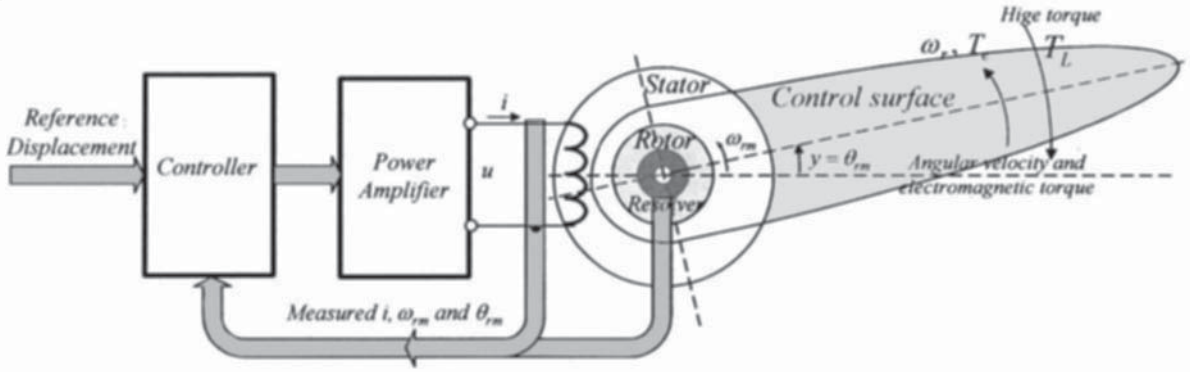


Figure 5.7. Fly-by-wire flight servo with electric motor and PWM power amplifier

The electromechanical flight servo-system integrates electromechanical motion devices (actuator and resolver) and power amplifier. These components must be modeled and then simulated and analyzed in the MATLAB environment. In fact, the analysis performed illustrates that the designer must develop accurate mathematical models integrating all components of complex multivariable real-world dynamic systems. The state and control variables must be defined, and differential equations (mathematical models) must be found with a minimum level of simplifications and assumptions. As the mathematical model is developed, the dynamic systems can be simulated and analyzed using MATLAB.

5.2. Mathematical Model Developments and MATLAB Applications

The equations to model the dynamics of mechanical systems can be straightforwardly found using Newton's second law of motion

$$\sum \vec{F}(\vec{x},t) = m\vec{a},$$

where $\vec{F}(\vec{x},t)$ is the vector sum of all forces applied to the body; $\vec{a}$ is the vector of acceleration of the body with respect to an inertial reference frame; m is the mass.

Hence, in the Cartesian system, or any other coordinate systems, the forces, acceleration, velocity, and displacement in one, two, or three dimensions (x, y, and z axes in the three-dimensional Cartesian system) are examined. One obtains

$$\sum \vec{F}(\vec{\mathbf{x}},t) = m\vec{a} = m\frac{d\vec{\mathbf{x}}^2}{dt^2} = m\begin{bmatrix} \dfrac{d\bar{x}^2}{dt^2} \\ \dfrac{d\bar{y}^2}{dt^2} \\ \dfrac{d\bar{z}^2}{dt^2} \end{bmatrix}.$$

Example 5.2.1.

Consider a body of mass m in the xy plane (two-dimensional systems). Derive the equations of motion assuming that the external force $\vec{F}_a$ is applied in the x direction ($\vec{F}_a(t,x) = 10v\cos 100t = 10\dfrac{dx}{dt}\cos 100t$) and the viscous friction force is $F_{fr} = B_v \dfrac{dx}{dt}$; B_v is the viscous friction coefficient.

Solution.

The free-body diagram is illustrated in Figure 5.8.

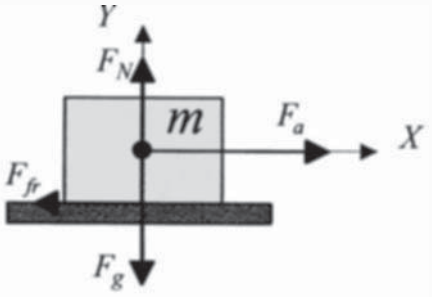


Figure 5.8. Free-body diagram

The *net* force acting in the x direction is found using the time-varying applied force $\vec{F}_a$ and the friction force $\vec{F}_{fr}$. In particular,

$$\sum \vec{F}_X = \vec{F}_a - \vec{F}_{fr}.$$

Hence, the second-order differential equation of motion in the x direction is

$$\vec{F}_a - \vec{F}_{fr} = ma_x = m\frac{d^2x}{dt^2}.$$

We obtain the following second-order differential equation to model the body dynamics in the x direction $\dfrac{d^2x}{dt^2} = \dfrac{1}{m}\left(F_a - B_v\dfrac{dx}{dt}\right)$. Using the velocity in the x direction $v = \dfrac{dx}{dt}$, a set of two first-order differential equations results, and

$$\frac{dx}{dt} = v,$$

$$\frac{dv}{dt} = \frac{1}{m}\left(F_a - B_v\frac{dx}{dt}\right) = \frac{1}{m}\left(10v\cos 100t - B_v v\right).$$

The sum of the forces acting in the y direction is

$$\sum \vec{F}_Y = \vec{F}_N - \vec{F}_g,$$

where $\vec{F}_g = mg$ is the gravitational force; $\vec{F}_N$ is the normal force (equal and opposite to the gravitational force, e.g., $\vec{F}_N = -\vec{F}_g$).

The equation of motion in the y direction is

$$\vec{F}_N - \vec{F}_g = 0 = ma_y = m\frac{d^2y}{dt^2}.$$

□

Newton's second law of rotational motion is

$$\sum M = J\alpha = J\frac{d^2\theta}{dt^2},$$

where $\sum M$ is the sum of all moments about the center of mass of a body; J is the moment of inertia about the center of mass; α is the angular acceleration, $\alpha = \frac{d^2\theta}{dt^2}$; θ is the angular displacement.

In the next example we illustrate the application of the rotational Newtonian mechanics in the model developments.

Example 5.2.2.

Figure 5.9 illustrates a simple pendulum (point mass m) suspended by a massless unstretchable string of length l. Derive the equations of motion (mathematical model in the form of differential equations) assuming that the friction force is a linear function of the angular velocity (e.g., $T_f = B_m\omega$); B_m is the viscous friction coefficient.

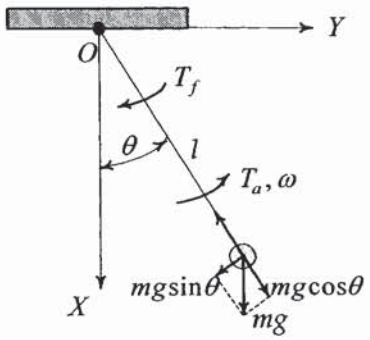


Figure 5.9. Simple pendulum

Solution.

The restoring force is $F_{rest} = -mg\sin\theta$.

Thus, the *net* moment about the pivot point O is

$$\sum M = T_{rest} + T_a - T_f = -mgl\sin\theta + T_a - B_m\omega,$$

where T_a is the applied torque.

We have the following second-order differential equation to model the pendulum motion

$$J\alpha = J\frac{d^2\theta}{dt^2} = -mgl\sin\theta + T_a - B_m\omega, \text{ or } \frac{d^2\theta}{dt^2} = \frac{1}{J}\left(-mgl\sin\theta + T_a - B_m\omega\right)$$

where J is the moment of inertial of the mass about the point O.

Taking note of $\frac{d\theta}{dt} = \omega$, one obtains a set of two first-order differential equations

$$\frac{d\omega}{dt} = \frac{1}{J}\left(-mgl\sin\theta + T_a - B_m\omega\right),$$

$$\frac{d\theta}{dt} = \omega.$$

The moment of inertia is $J = ml^2$. Thus, we have two linear differential equations

$$\frac{d\omega}{dt} = -\frac{B_m}{ml^2}\omega - \frac{g}{l}\sin\theta + \frac{1}{ml^2}T_a,$$

$$\frac{d\theta}{dt} = \omega.$$

This set of differential equations can be numerically simulated in MATLAB using two m-files as illustrated in the examples. The pendulum parameters (B_m, m, and l) should be assigned. The simulation of a simple pendulum is performed in Chapter 6 (see Example 6.1.2).

□

Example 5.2.3.

A body is suspended from a spring with coefficient k_s, and the viscous friction coefficient is B_v (see Figure 5.10). Derive the mathematical model in the form of differential equations.

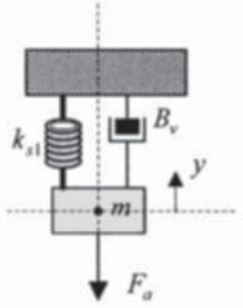


Figure 5.10.

Solution.

Using Newton's second law, one finds the resulting second-order differential equation

$$m\frac{d^2y}{dt^2} + B_v\frac{dy}{dt} + k_s y = F_a,$$

which models the body dynamics and, therefore, represents the mathematical model. □

The circuitry mathematical models to simulate and analyze the circuitry dynamics are found using Kirchhoff's voltage and current laws. The following examples illustrate the application of Kirchhoff's laws.

Example 5.2.4.

An electric circuit is documented in Figure 5.11. Find the mathematical model in the form of integro-differential equations.

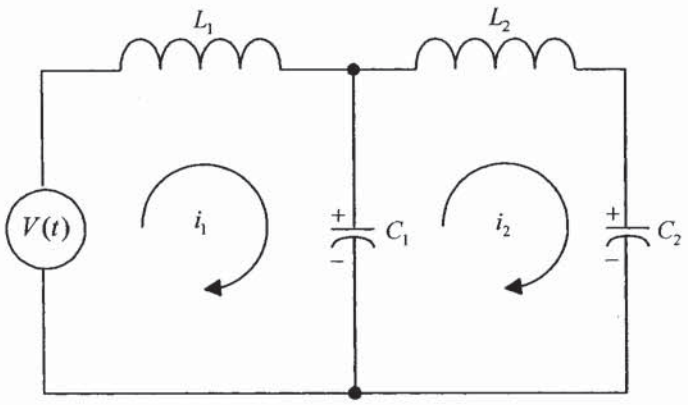


Figure 5.11. Two-mesh circuit

Solution.

Using Kirchhoff's law, we find

$$L_1\frac{di_1}{dt}+\frac{1}{C_1}\int(i_1-i_2)dt=V \text{ and } L_2\frac{di_2}{dt}-\frac{1}{C_1}\int(i_1-i_2)dt+\frac{1}{C_2}\int i_2dt=0.$$

Thus, the integro-differential equations to model the circuit are found. □

Example 5.2.5.

A *buck* (*step-down*) switching converter is documented in Figure 5.12 [3, 4]. In this converter, switch S, inductor L, and capacitor C have resistances. These resistances are denoted as r_s, r_L, and r_c. The resistive–inductive load with the *back emf* E_a is formed by the resistor r_a and inductor L_a. Derive the mathematical model and study the converter dynamics through numerical simulations.

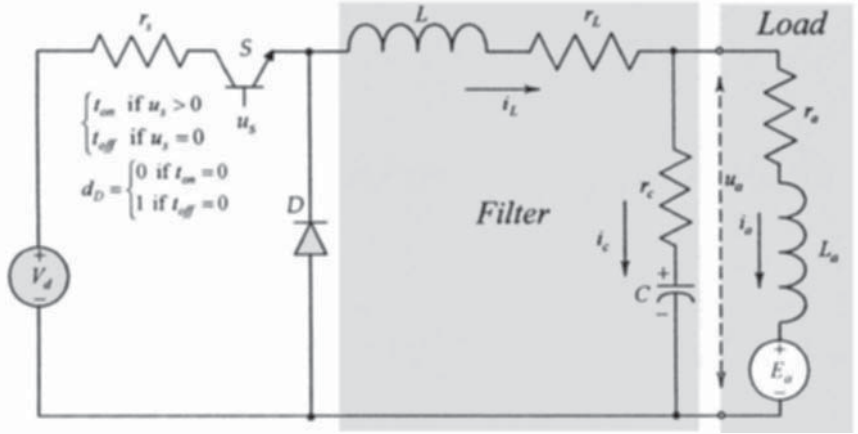


Figure 5.12. Switching converter

Solution.

The voltage is regulated by opening and closing the switch S. Thus, this switch, which is a high-frequency transistor, open and closed. Correspondingly, differential equations must be found for these two switch "states." One concludes that the voltage at the load is regulated by using a pulse-width-modulation (PWM) switching. For lossless switch (if $r_s=0$), the voltage

across the diode D is equal to the supplied voltage V_d when the switch is closed, and the voltage is zero if the switch is open. The voltage applied to the load u_a is regulated by controlling the switching *on* and *off* durations (t_{on} and t_{off}). The switching frequency is $\frac{1}{t_{on}+t_{off}}$. These *on* and *off* durations are controlled by u_s. If $u_s = 0$, the switch is closed, while if $u_s > 0$, the switch is open.

Making use of the duty ratio $d_D = \frac{t_{on}}{t_{on}+t_{off}}$, $d_D = \begin{cases} 0 \text{ if } t_{on} = 0 \\ 1 \text{ if } t_{off} = 0 \end{cases}$, $d_D \in [0 \quad 1]$, using the *averaging* concept, if $r_s = 0$ we have $u_{dN} = \frac{t_{on}}{t_{on}+t_{off}} V_d = d_D V_d$.

Using Kirchhoff's laws the following sets of differential equations are derived.

- Switch is closed:
  When the switch is closed, the diode is reverse biased. For t_{off}=0, $d_D = 1$, we have

$$\frac{du_C}{dt} = \frac{1}{C}(i_L - i_a),$$
$$\frac{di_L}{dt} = \frac{1}{L}\left(-u_C - (r_L + r_c)i_L + r_c i_a - r_s i_L + V_d\right),$$
$$\frac{di_a}{dt} = \frac{1}{L_a}\left(u_C + r_c i_L - (r_a + r_c)i_a - E_a\right).$$

- Switch is open:
  If the switch is open, the diode is forward biased. For $d_D = 0$ (t_{on}=0), we find

$$\frac{du_C}{dt} = \frac{1}{C}(i_L - i_a),$$
$$\frac{di_L}{dt} = \frac{1}{L}\left(-u_C - (r_L + r_c)i_L + r_c i_a\right),$$
$$\frac{di_a}{dt} = \frac{1}{L_a}\left(u_C + r_c i_L - (r_a + r_c)i_a - E_a\right).$$

When the switch is closed, the duty ratio is 1. In contrast, if the switch is open, the duty ratio is 0. Therefore, $d_D = \begin{cases} 0 \text{ if } t_{on} = 0 \\ 1 \text{ if } t_{off} = 0 \end{cases}$.

Assuming that the switching frequency is high, one uses the duty ratio to find the following augmented set of differential equations to model the converter transients. In particular,

$$\frac{du_C}{dt} = \frac{1}{C}(i_L - i_a),$$
$$\frac{di_L}{dt} = \frac{1}{L}\left(-u_C - (r_L + r_c)i_L + r_c i_a - r_s i_L d_D + V_d d_D\right),$$
$$\frac{di_a}{dt} = \frac{1}{L_a}\left(u_C + r_c i_L - (r_a + r_c)i_a - E_a\right).$$

Thus, the *buck* converter dynamics is modeled by a set of derived nonlinear differential equations. Our next step is to numerically solve these differential equations using MATLAB. By

making use of a set of differential equations, two m-files are written to perform simulations and visualize the converter dynamics. The following parameters are used:
r_s = 0.025 ohm, r_L = 0.05 ohm, r_c = 0.05 ohm, r_a = 2.5 ohm, C = 0.05 F, L = 0.005 H, L_a = 0.05 H, V_d = 40 V, E_a = 5 V and d_D = 0.5.

The initial conditions are assigned to be $\begin{bmatrix} u_{C0} \\ i_{L0} \\ i_{a0} \end{bmatrix} = \begin{bmatrix} 10 \\ 5 \\ -5 \end{bmatrix}$.

MATLAB script (c5_2_5a.m):

```
echo on; clear all
t0=0; tfinal=0.4; tspan=[t0 tfinal]; % initial and final time
y0=[10 5 -5]';                       % initial conditions
[t,y]=ode45('c5_2_5b',tspan,y0);     % ode45 MATLAB solver
% Plot of the time history found by solving
% three differential equations assigned in the file c5_2_5b.m
% 3-D plot using x1, x2 and x3 as the variables
plot3(y(:,1),y(:,2),y(:,3))
xlabel('x1'), ylabel('x2'), zlabel('x3')
text(10,5,-5,'x0 Initial')
v=axis
pause
% 3-D plot using x1, x3 and time as the variables
plot3(y(:,1),y(:,3),t)
xlabel('x1'), ylabel('x3'), zlabel('time')
text(10,-5,0,'x0 Initial')
v=axis
pause
% 2-D plots
subplot(2,2,1); plot(t,y);
xlabel('Time (seconds)');
title('Dynamics of the state variables');
subplot(2,2,2); plot(t,y(:,1),'-');
xlabel('Time (seconds)'); title('Voltage uc (x1), [V]');
subplot(2,2,3); plot(t,y(:,2),'--');
xlabel('Time (seconds)'); title('Current iL (x2), [A]');
subplot(2,2,4); plot(t,y(:,3),':');
xlabel('Time (seconds)'); title('Current ia (x3), [A]');
disp('END');
```

MATLAB script (c5_2_5b.m):

```
function yprime=difer(t,y);
% converter parameters
rs=0.025; rl=0.05; rc=0.05; ra=2.5; C=0.05; L=0.005; La=0.05;
% voltage applied, back emf and duty ratio
Vd=50; Ea=5; D=0.5;
% three differential equations to model a buck converter
yprime=[(y(2,:)-y(3,:))/C;...
(-y(1,:)-(rl+rc)*y(2,:)+rc*y(3,:)-rs*y(2,:)*D+Vd*D)/L;...
(y(1,:)+rc*y(2,:)-(rc+ra)*y(3,:)-Ea)/La];
```

Three-dimensional plots using $x_1(t)$, $x_2(t)$, and $x_3(t)$ as well as $x_1(t)$, $x_3(t)$, and t as the variables are shown in Figure 5.13. Figure 5.13 also illustrates two-dimensional plots reporting the transient dynamics for three state variables $u_C(t)$, $i_L(t)$, and $i_a(t)$ which are $x_1(t)$, $x_2(t)$, and $x_3(t)$. We conclude that we performed numerical simulations and visualized the results using plotting statements.

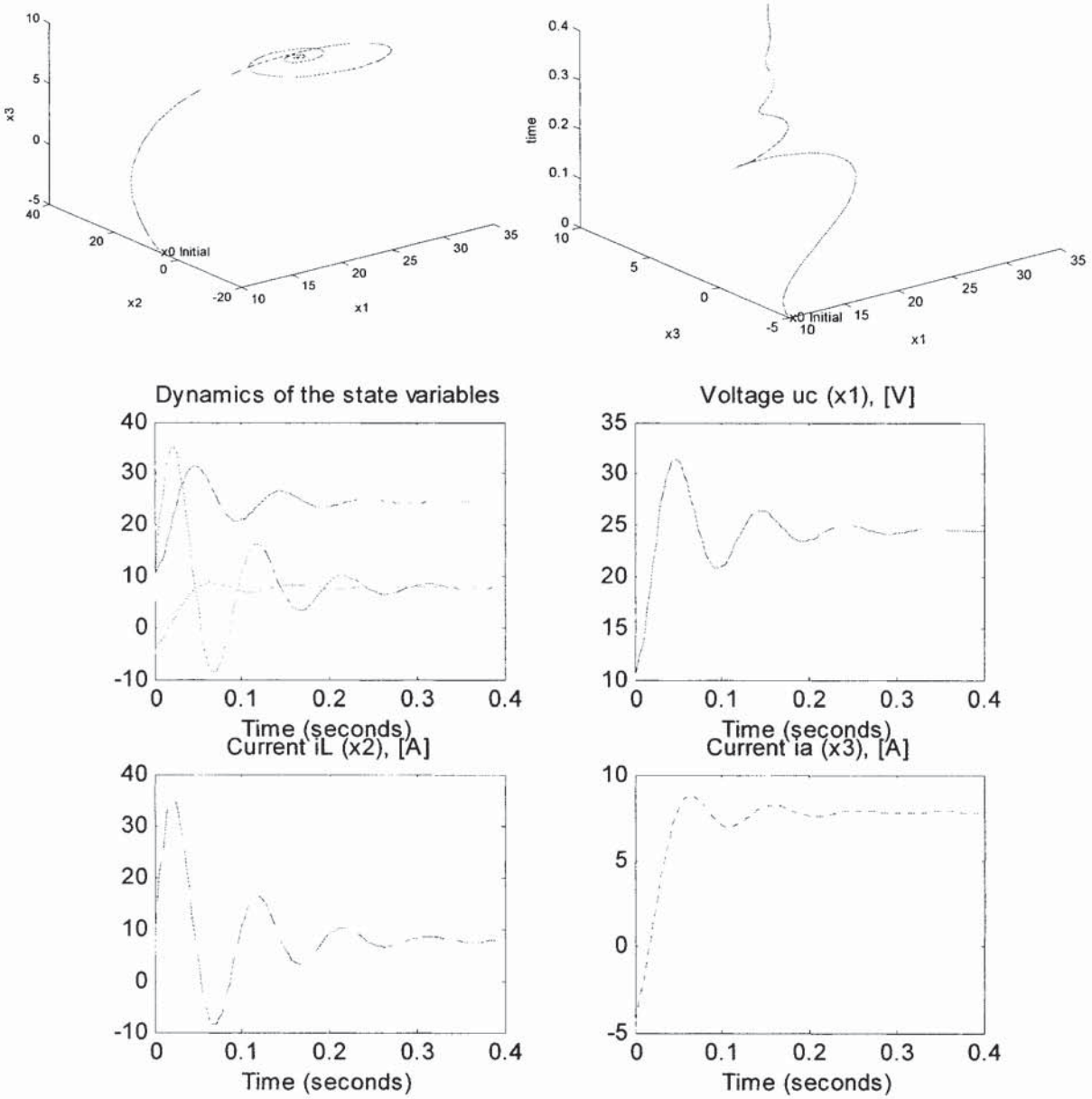


Figure 5.13. Evolution of the variables (in three dimensions) and transient dynamics

Making note of the output equation $u_a = u_C + r_c i_L - r_c i_a$, the voltage at the load terminal u_a can be plotted. We type in the Command Window

```
»rc=0.05;plot(t,y(:,1)+rc*y(:,2)-rc*y(:,3),'-');xlabel('Time (seconds)');title('Voltage ua, [V]');
```

The resulting plot for $u_a(t)$ is shown in Figure 5.14.

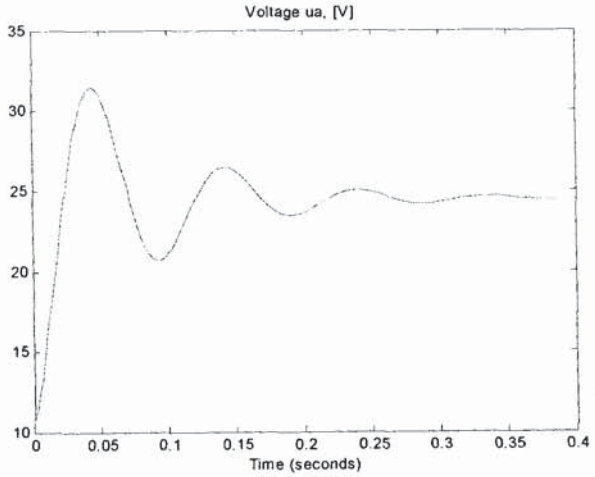


Figure 5.14. Transient dynamics for u_a, $u_a = u_C + r_c i_L - r_c i_a$ □

Example 5.2.6.

A one-quadrant *boost* (*step-up*) switching converter is documented in Figure 5.15 [3, 4]. The converter parameters are: $r_s = 0.025$ ohm, $r_L = 0.05$ ohm, $r_c = 0.05$ ohm, $r_a = 2.5$ ohm, $C = 0.05$ F, $L = 0.005$ H, $L_a = 0.05$ H, $V_d = 40$ V, $E_a = 5$ V and $d_D = 0.25$.

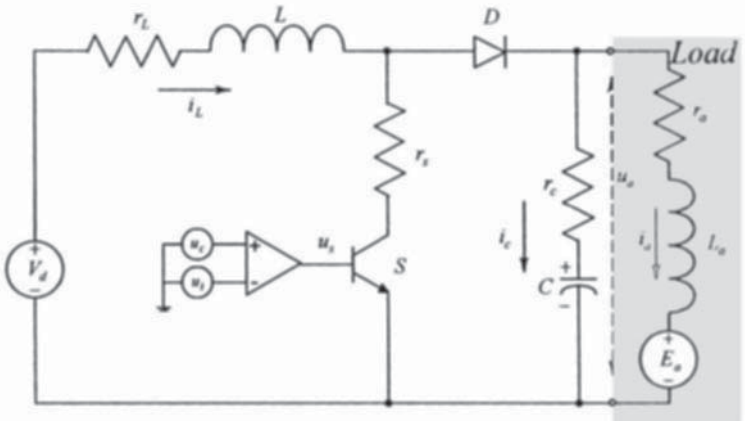


Figure 5.15. High-frequency switching converter

Solution.

When the switch is closed, the diode is reverse biased. Correspondingly, the following set of differential equations results:

$$\frac{du_C}{dt} = -\frac{1}{C} i_a, \quad \frac{di_L}{dt} = \frac{1}{L}\left(-\left(r_L + r_s\right) i_L + V_d\right), \quad \frac{di_a}{dt} = \frac{1}{L_a}\left(u_C - \left(r_a + r_c\right) i_a - E_a\right).$$

If the switch is open, the diode is forward biased due to the fact that the direction of the inductor current i_L does not change instantly. Hence, we have the differential equations

$$\frac{du_C}{dt} = \frac{1}{C}\left(i_L - i_a\right), \quad \frac{di_L}{dt} = \frac{1}{L}\left(-u_C - \left(r_L + r_c\right) i_L + r_c i_a + V_d\right), \quad \frac{di_a}{dt} = \frac{1}{L_a}\left(u_C + r_c i_L - \left(r_a + r_c\right) i_a - E_a\right).$$

Using the duty ratio d_D (which can vary from 0 to 1), we find the resulting model, and

$$\frac{du_C}{dt}=\frac{1}{C}\left(i_L-i_a-i_L d_D\right),$$

$$\frac{di_L}{dt}=\frac{1}{L}\left(-u_C-\left(r_L+r_c\right)i_L+r_c i_a+u_C d_D+\left(r_c-r_s\right)i_L d_D-r_c i_a d_D+V_d\right),$$

$$\frac{di_a}{dt}=\frac{1}{L_a}\left(u_C+r_c i_L-\left(r_a+r_c\right)i_a-r_c i_L d_D-E_a\right).$$

Our goal is to simulate the *boost* converter, and initial conditions and parameters must be assigned. Let the initial conditions be $\begin{bmatrix} u_{C0} \\ i_{L0} \\ i_{a0} \end{bmatrix}=\begin{bmatrix} 10 \\ 5 \\ -5 \end{bmatrix}$.

Taking note of the differential equations, the following m-files are written to solve differential equations with the assigned initial conditions, converter parameters, and duty ratio.

MATLAB script (c5_2_6a.m):

```
echo on; clear all
t0=0; tfinal=0.4; tspan=[t0 tfinal]; % initial and final time
y0=[10 5 -5]';                       % initial conditions
[t,y]=ode45('c5_2_6b',tspan,y0);     % ode45 MATLAB solver
% Plot of the time history found by solving
% three differential equations assigned in the file c5_2_5b.m
% 3-D plot using x1, x2 and x3 as the variables
plot3(y(:,1),y(:,2),y(:,3))
xlabel('x1'), ylabel('x2'), zlabel('x3')
text(10,5,-5,'x0 Initial')
v=axis
pause
% 3-D plot using x1, x3 and time as the variables
plot3(y(:,1),y(:,3),t)
xlabel('x1'), ylabel('x3'), zlabel('time')
text(10,-5,0,'x0 Initial')
v=axis
pause
% 2-D plots
subplot(2,2,1); plot(t,y);
xlabel('Time (seconds)');
title('Dynamics of the state variables');
subplot(2,2,2); plot(t,y(:,1),'-');
xlabel('Time (seconds)'); title('Voltage uc (x1), [V]');
subplot(2,2,3); plot(t,y(:,2),'--');
xlabel('Time (seconds)'); title('Current iL (x2), [A]');
subplot(2,2,4); plot(t,y(:,3),':');
xlabel('Time (seconds)'); title('Current ia (x3), [A]');
disp('END');
```

MATLAB script (c5_2_6b.m):

```
function yprime=difer(t,y);
% converter parameters
rs=0.025; rl=0.05; rc=0.05; ra=2.5; C=0.05; L=0.005; La=0.05;
% voltage applied, back emf and duty ratio
Vd=50; Ea=5; D=0.25;
% three differential equations to model a boost converter
yprime=[(y(2,:)-y(3,:)-y(2,:)*D)/C;...

(-y(1,:)-(rl+rc)*y(2,:)+rc*y(3,:)+y(1,:)*D+(rc-rs)*y(2,:)*D-
rc*y(3,:)*D+Vd)/L;...
(y(1,:)+rc*y(2,:)-(ra+rc)*y(3,:)-rc*y(2,:)*D-Ea)/La];
```

Two three-dimensional plots $[x_1(t), x_2(t), x_3(t)]$ and $[x_1(t), x_3(t), t]$ are illustrated in Figure 5.16. The converter transient dynamics for $x_1(t)$, $x_2(t)$, $x_3(t)$ are reported in Figure 5.16.

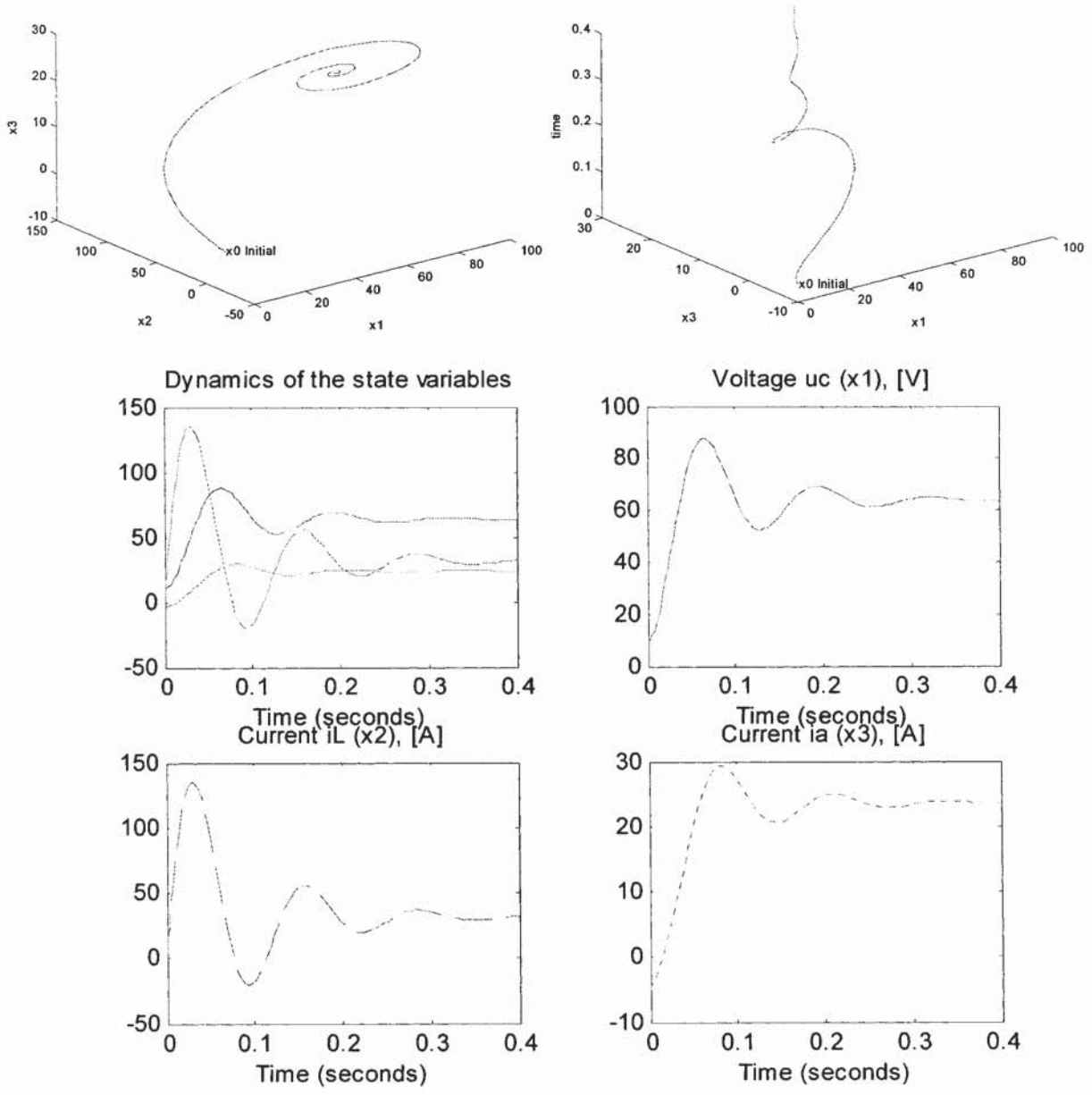


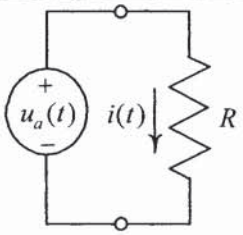
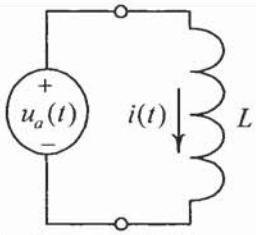
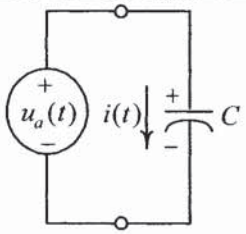
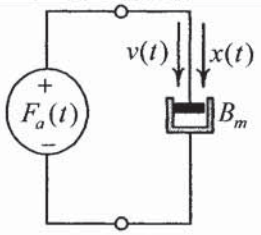
Figure 5.16. State variables evolution and dynamics of the converter

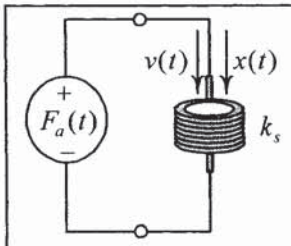
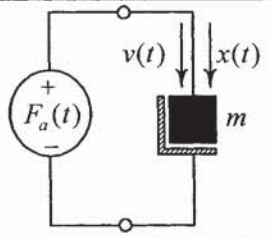
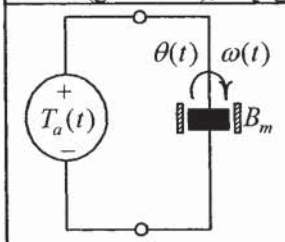
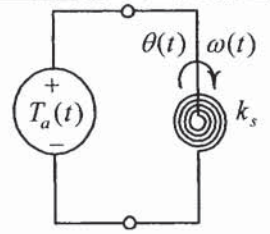
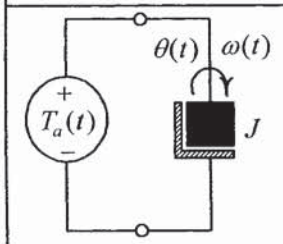
We conclude that numerical simulations and visualization were performed. In particular, the evolution of three state variables $u_C(t)$, $i_L(t)$, and $i_a(t)$ is documented, and the analysis can be performed. □

5.3. Modeling and Computing Using MATLAB

It was illustrated that differential equations result as one applies the fundamental laws to electrical and mechanical systems. It has been shown that the transient dynamics of electrical and mechanical systems are described by linear and nonlinear differential equations. To illustrate the similarity of results, and to visualize the results, the equations of motion for some electromechanical system elements are shown in Table 5.1 [3].

Table 5.1. Basic Elements of Electromechanical Systems

Electromechanical System	Variables Used	Equation
 Resistance, R [ohm]	Applied voltage $u_a(t)$ [V] Current $i(t)$ [A]	$u_a(t) = Ri(t)$ $i(t) = \frac{1}{R}u_a(t)$
 Inductance, L [H]	Applied voltage $u_a(t)$ [V] Current $i(t)$ [A]	$u_a(t) = L\frac{di(t)}{dt}$ $i(t) = \frac{1}{L}\int_{t_0}^{t} u_a(\tau)d\tau$
 Capacitance, C [F]	Applied voltage $u_a(t)$ [V] Current $i(t)$ [A]	$u_a(t) = \frac{1}{C}\int_{t_0}^{t} i(\tau)d\tau$ $i(t) = C\frac{du_a(t)}{dt}$
 Translational damper, B_v [N-sec]	Applied force $F_a(t)$ [N] Linear velocity $v(t)$ [m/sec] Linear position $x(t)$ [m]	$F_a(t) = B_m v(t)$ $v(t) = \frac{1}{B_m}F_a(t)$ and $F_a(t) = B_m v(t) = B_m\frac{dx(t)}{dt}$ $x(t) = \frac{1}{B_v}\int_{t_0}^{t} F_a(\tau)d\tau$

 Translational spring, k_s [N]	Applied force $F_a(t)$ [N] Linear velocity $v(t)$ [m/sec] Linear position $x(t)$ [m]	$F_a(t) = k_s x(t)$ $x(t) = \frac{1}{k_s} F_a(t)$ and $v(t) = \frac{dx(t)}{dt} = \frac{1}{k_s}\frac{dF_a(t)}{dt}$ $F_a(t) = k_s \int_{t_0}^{t} v(\tau)d\tau$
 Mass (grounded), m [kg]	Applied force $F_a(t)$ [N] Linear velocity $v(t)$ [m/sec] Linear position $x(t)$ [m]	$F_a(t) = m\frac{dv}{dt} = m\frac{d^2x(t)}{dt^2}$ $v(t) = \frac{1}{m}\int_{t_0}^{t} F_a(\tau)d\tau$
 Rotational damper, B_m [N-m-sec/rad]	Applied torque $T_a(t)$ [N-m] Angular velocity $\omega(t)$ [rad/sec] Angular displacement $\theta(t)$ [rad]	$T_a(t) = B_m\omega(t)$ $\omega(t) = \frac{1}{B_m} T_a(t)$ and $T_a(t) = B_m\omega(t) = B_m\frac{d\theta(t)}{dt}$ $\theta(t) = \frac{1}{B_m}\int_{t_0}^{t} T_a(\tau)d\tau$
 Rotational spring, k_s [N-m-sec/rad]	Applied torque $T_a(t)$ [N-m] Angular velocity $\omega(t)$ [rad/sec] Angular displacement $\theta(t)$ [rad]	$T_a(t) = k_s\theta(t)$ $\theta(t) = \frac{1}{k_s} T_a(t)$ and $\omega(t) = \frac{d\theta(t)}{dt} = \frac{1}{k_s}\frac{dT_a(t)}{dt}$ $T_a(t) = k_s\int_{t_0}^{t} \omega(\tau)d\tau$
 Rotational mass (grounded), J [kg-m^2]	Applied torque $T_a(t)$ [N-m] Angular velocity $\omega(t)$ [rad/sec] Angular displacement $\theta(t)$ [rad]	$T_a(t) = J\frac{d\omega}{dt} = J\frac{d^2\theta(t)}{dt^2}$ $\omega(t) = \frac{1}{J}\int_{t_0}^{t} T_a(\tau)d\tau$

The similarity of equations of motion is evident as one compares the derived dynamics, which is given by the corresponding differential equations. Consider the translational and rotational (torsional) mechanical systems shown in Figure 5.17.

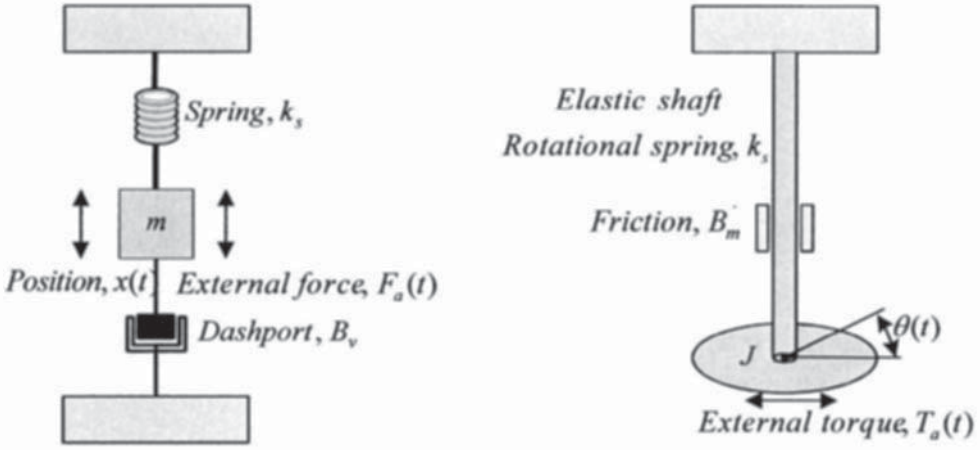


Figure 5.17. Translational and torsional mechanical systems

From Newton's second law, the second-order differential equations of translational and torsional dynamics are found to be

1. Translational dynamics: $$m\frac{d^2x}{dt^2} + B_v\frac{dx}{dt} + k_s x = F_a(t),$$
2. Torsional dynamics: $$J\frac{d^2\theta}{dt^2} + B_m\frac{d\theta}{dt} + k_s\theta = T_a(t),$$

where $F_a(t)$ and $T_a(t)$ are the time-varying applied force and torque; B_v and B_m are the viscous friction coefficients; k_s is the translational and rotational (elasticity) spring coefficient.

Consider two *RLC* circuits illustrated in Figure 5.18.

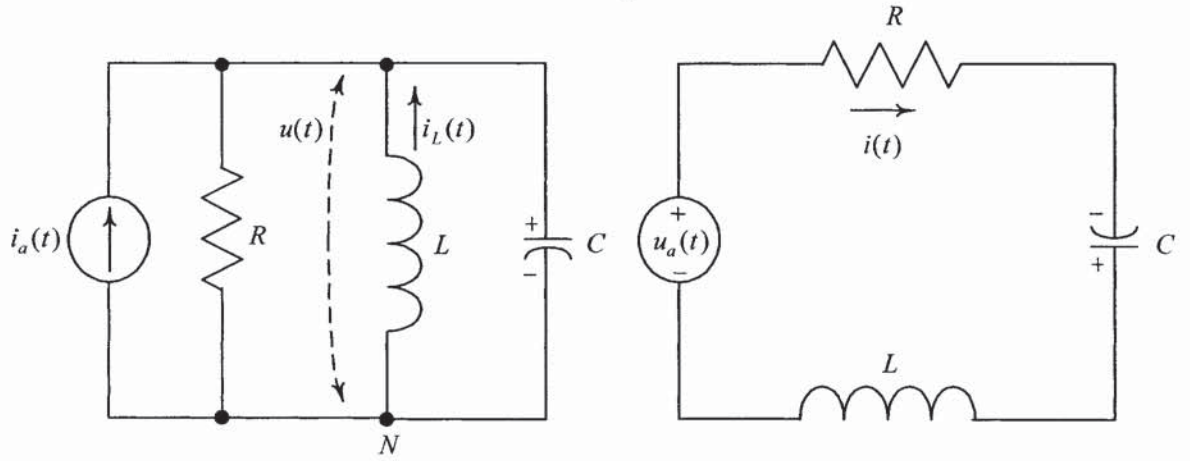


Figure 5.18. Parallel and series *RLC* circuits

The energy is stored in the inductor and the capacitor. The integro-differential equation (an integral as well as a derivative of the dependent variables appears) for the parallel circuit is obtained by summing the currents away from the top node

$$\frac{u}{R}+\frac{1}{L}\int_{t_0}^{t}u(\tau)d\tau-i_L(t_0)+C\frac{du}{dt}=i_a(t),$$

while the integro-differential equation for the series circuit is found by summing the voltages around the closed path. In particular,

$$Ri+\frac{1}{C}\int_{t_0}^{t}i(\tau)d\tau-v_C(t_0)+L\frac{di}{dt}=u_a(t).$$

By differentiating these equations with respect to time and using the fact that $i_L(t_0)$ and $v_C(t_0)$ are constants, we have

$$C\frac{d^2u}{dt^2}+\frac{1}{R}\frac{du}{dt}+\frac{1}{L}u=\frac{di_a}{dt},\ \text{or}\ \frac{d^2u}{dt^2}+\frac{1}{RC}\frac{du}{dt}+\frac{1}{LC}u=\frac{1}{C}\frac{di_a}{dt},$$

and

$$L\frac{d^2i}{dt^2}+R\frac{di}{dt}+\frac{1}{C}i=\frac{du_a}{dt},\ \text{or}\ \frac{d^2i}{dt^2}+\frac{R}{L}\frac{di}{dt}+\frac{1}{LC}i=\frac{1}{L}\frac{du_a}{dt}.$$

Parallel and series *RLC* circuits lead to the second-order differential equations. It is evident that these linear differential equations can be numerically modeled in MATLAB using the `ode` solvers that were illustrated.

It was shown that the mechanical systems and the *RLC* circuits considered are modeled by the second-order differential equations. The analytical solution of linear differential equations with constant coefficients can be easily derived. The general solution of the second-order linear differential equations is found by using the roots of the characteristic equation. The damping coefficient ξ and the resonant frequency ω_0 are given by

$$\xi=\frac{1}{2RC},\ \omega_0=\frac{1}{\sqrt{LC}}\ \text{and}\ \xi=\frac{R}{2L},\ \omega_0=\frac{1}{\sqrt{LC}}$$

for the parallel and series *RLC* circuits (for mechanical systems, $\xi=\frac{B_m}{2\sqrt{k_s m}}$ and $\omega_0=\sqrt{\frac{k_s}{m}}$).

We write the differential equation as

$$\frac{d^2x}{dt^2}+2\xi\frac{dx}{dt}+x=f(t)$$

to find three possible solutions examining the characteristic equation

$$s^2+2\xi s+\omega_0^2=(s-s_1)(s-s_2)=0.$$

This characteristic equation was found by making use the Laplace operator $s=\frac{d}{dt}$. Furthermore, $s^2=\frac{d^2}{dt^2}$. The characteristic roots (eigenvalues) are given as

$$s_{1,2}=-\xi\pm\sqrt{\xi^2-\omega_0^2}.$$

Case 1. If $\xi^2 > \omega_0^2$, the real distinct characteristic roots s_1 and s_2 result.

The general solution is $x(t) = ae^{s_1 t} + be^{s_2 t} + c_f$, where coefficients a and b are obtained using the initial conditions, c_f is the solution due to the *forcing* function f (for the *RLC* circuits f is $i_a(t)$ and $u_a(t)$).

Case 2. For $\xi^2 = \omega_0^2$, the characteristic roots are real and identical, e.g.,

$s_1 = s_2 = -\xi$.

The solution of the second-order differential equation is given as

$x(t) = (a+b)e^{-\xi t} + c_f$.

Case 3. If $\xi^2 < \omega_0^2$, the complex distinct characteristic roots are found as

$s_{1,2} = -\xi \pm j\sqrt{\omega_0^2 - \xi^2}$.

Hence, the general solution is

$$x(t) = e^{-\xi t}\left[a\cos\left(\sqrt{\omega_0^2 - \xi^2}\,t\right) + b\sin\left(\sqrt{\omega_0^2 - \xi^2}\,t\right)\right] + c_f = e^{-\xi t}\sqrt{a^2+b^2}\cos\left[\left(\sqrt{\omega_0^2 - \xi^2}\,t\right) + \tan^{-1}\left(\frac{-b}{a}\right)\right] + c_f.$$

Example 5.3.1.

For the series *RLC* circuit, find the analytical solutions. Derive and plot the transient response due to the unit step input with initial conditions. Assign the following parameters: R = 0.4 ohm, L = 0.5 H, C = 2 F, a = 2 and b = – 2.

Solution.

For the series *RLC* circuit, the following differential equation was obtained:

$$\frac{d^2 i}{dt^2} + \frac{R}{L}\frac{di}{dt} + \frac{1}{LC}i = \frac{1}{L}\frac{du_a}{dt}.$$

The characteristic equation is $s^2 + \frac{R}{L}s + \frac{1}{LC} = 0$. Therefore, the characteristic roots are

$$s_1 = -\frac{R}{2L} - \sqrt{\left(\frac{R}{2L}\right)^2 - \frac{1}{LC}} \text{ and } s_2 = -\frac{R}{2L} + \sqrt{\left(\frac{R}{2L}\right)^2 - \frac{1}{LC}}.$$

If $\left(\frac{R}{2L}\right)^2 > \frac{1}{LC}$, then the characteristic roots are real and distinct.

If $\left(\frac{R}{2L}\right)^2 = \frac{1}{LC}$, then the characteristic roots are real and identical.

If $\left(\frac{R}{2L}\right)^2 < \frac{1}{LC}$, then the characteristic roots are complex.

Making use of the assigned values for R, L, and C, one concludes that the underdamped series dynamics are given by

$$x(t) = e^{-\xi t}\left[a\cos\left(\sqrt{\omega_0^2 - \xi^2}\,t\right) + b\sin\left(\sqrt{\omega_0^2 - \xi^2}\,t\right)\right] + c_f,$$

where $\xi = \frac{R}{2L} = 0.4$ and $\omega_0 = \frac{1}{\sqrt{LC}} = 1$.

In the Command Window we type the following statements:

```
>> t=0:.01:15; a=2; b=-2; cf=1; e=0.4; w0=1;
>> x=exp(-e*t).*(a*cos(sqrt(w0^2-e^2)*t)+b*sin(sqrt(w0^2-e^2)*t))+cf;plot(t,x)
```

The resulting circuitry dynamics are documented in Figure 5.19.

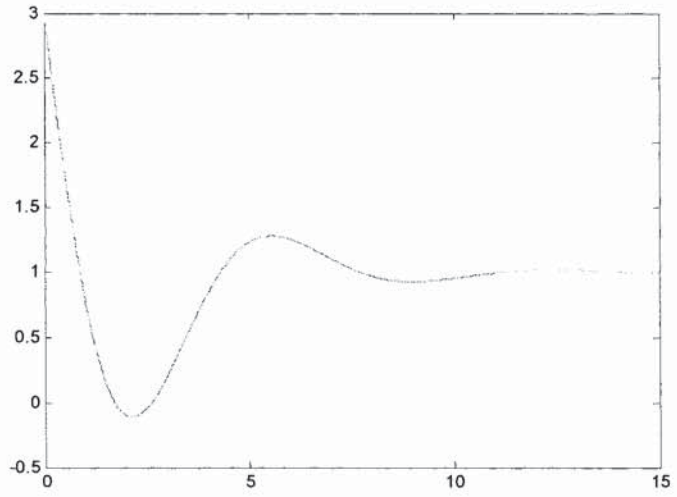


Figure 5.19. Circuitry dynamics due to the unit step input and initial conditions □

We have used Newton's and Kichhoff's laws to find the differential equations to perform the analysis of mechanical systems and electric circuits. Mathematical models of electromechanical systems can be derived integrating differential equations found for electrical and mechanical subsystems. Furthermore, the application of MATLAB was illustrated to perform numerical simulations. It must be emphasized that the MATLAB environment can be used to derive the analytical solution as demonstrated by the following example.

Example 5.3.2.
Analytically solve the third-order differential equation

$$\frac{d^3x}{dt^3}+2\frac{dx}{dt}+3x=4f$$

using the Symbolic Math Toolbox.

Solution.
Using the `dsolve` solver, we type in the Command Window

```
>> x=dsolve('D3x+2*Dx+3*x=4*f')
```

The resulting solution is

```
x =
4/3*f+C1*exp(-t)+C2*exp(1/2*t)*cos(1/2*11^(1/2)*t)+C3*exp(1/2*t)*sin(1/2*11^(1/2)*t)
```

Using the `pretty` function, we find

```
>> pretty(x)
                                                    1/2
  4/3 f + C1 exp(-t) + C2 exp(1/2 t) cos(1/2 11     t)

                                          1/2
        + C3 exp(1/2 t) sin(1/2 11        t)
```

Thus, the solution is

$$x(t)=\frac{4}{3}f+c_1e^{-t}+c_2e^{0.5t}co\left(\tfrac{1}{2}\sqrt{11}t\right)+c_3e^{0.5t}\sin\left(\tfrac{1}{2}\sqrt{11}t\right).$$

Using the initial conditions, the unknown constants are found. As an example, let us assign the following initial conditions $\left(\frac{d^2x}{dt^2}\right)_0 = 5, \left(\frac{dx}{dt}\right)_0 = 0$ and $x_0 = -5$. We have

```
>> x=dsolve('D3x+2*Dx+3*x=4*f','D2x(0)=5','Dx(0)=0','x(0)=-5'); pretty(x)

                                                                    1/2
  4/3 f + (- 4/5 f - 2) exp(-t) + (- 8/15 f - 3) exp(1/2 t) cos(1/2 11    t)

                                  1/2                          1/2
        - 1/165 (16 f + 15) 11    exp(1/2 t) sin(1/2 11    t)
```

Hence, $c_1 = -\frac{4}{5}f - 2$, $c_2 = -\frac{8}{15}f - 3$, and $c_3 = -\frac{1}{165}(16f + 15)\sqrt{11}$.

If the forcing function is time-varying, the analytical solution of

$$\frac{d^3x}{dt^3} + 2\frac{dx}{dt} + 3x = 4f(t)$$

is found as

```
>> x=dsolve('D3x+2*Dx+3*x=4*f(t)'); pretty(x)

       /     /
      |     |
  4/55 |11   |  exp(t) f(t) dt exp(- 3/2 t)
      |     |
       \     /
          /
         |                                  1/2
       + |  -3 exp(- 1/2 t) f(t) 11    %1 - 11 exp(- 1/2 t) f(t) %2 dt %2
         |
          /
            /                                                              \
           |                                  1/2                          |
         + |  3 exp(- 1/2 t) f(t) 11    %2 - 11 exp(- 1/2 t) f(t) %1 dt %1|
           |                                                               |
            /                                                              /
        exp(1/2 t) + C1 exp(-t) + C2 exp(1/2 t) %2 + C3 exp(1/2 t) %1
                  1/2
  %1 := sin(1/2 11    t)

                  1/2
  %2 := cos(1/2 11    t)
```

Letting $f(t) = \sin(t)$ and assuming $\left(\frac{d^2x}{dt^2}\right)_0 = 5, \left(\frac{dx}{dt}\right)_0 = 0$ and $x_0 = -5$, we have

```
>> x=dsolve('D3x+2*Dx+3*x=4*sin(t)','D2x(0)=5','Dx(0)=0','x(0)=-5'); pretty(x)

                                                                        1/2
  2/5 %2 sin(%4) - 2/5 %2 sin(%3) - 2/5 %1 cos(%4) - 2/55 %1 sin(%3) 11

                                 1/2                          1/2
        + 2/55 %1 sin(%4) 11     + 2/55 %2 cos(%4) 11

                                 1/2                          1/2
        - 4/55 %2 sin(%3) 11     - 4/55 %2 sin(%4) 11

                                 1/2                          1/2
        + 4/55 %1 cos(%4) 11     + 4/55 %1 cos(%3) 11     + 2/5 %1 cos(%3)

                                                                  1/2
        - 2/5 cos(t) + 2/5 sin(t) - 2/55 %2 cos(%3) 11     - 8/5 exp(-t)

                                 13   1/2
        - 3 exp(1/2 t) %2 - exp(-- 11    exp(1/2 t) %1
                                 55
                  1/2
  %1 := sin(1/2 11    t)
```

```
                    1/2
%2 := cos(1/2 11     t)

                      1/2
%3 := 1/2 (-2 + 11    ) t

                      1/2
%4 := 1/2 (2 + 11     ) t
```

Thus, the Symbolic Math Toolbox allows us to find the analytical solutions for differential equations.

□

Example 5.3.3.

Consider the series *RLC* circuit given in Figure 5.20. Find the analytical solution using MATLAB. Plot the circuitry dynamics assigning circuitry parameters and setting initial conditions.

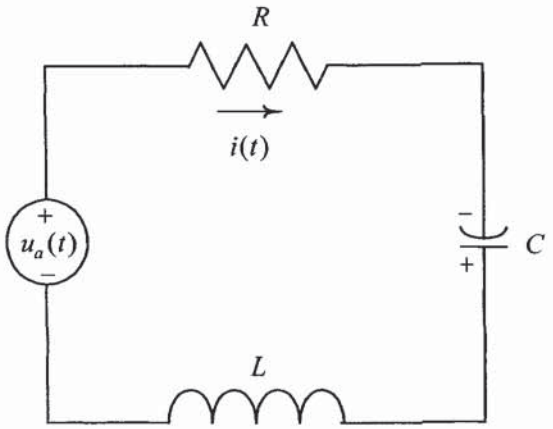


Figure 5.20. Series *RLC* circuit

Solution.

The state and control (*forcing* function) variables are used in the development of the mathematical model. Using the voltage across the capacitor and the current through the inductor as the state variables, and the supplied voltage $u_a(t)$ as the control, we have the following set of first-order differential equations:

$$C\frac{du_C}{dt} = i\,,\ L\frac{di}{dt} = -u_C - Ri + u_a(t)\,.$$

Hence, we have

$$\frac{du_C}{dt} = \frac{1}{C}i\,,\ \frac{di}{dt} = \frac{1}{L}\left(-u_C - Ri + u_a(t)\right).$$

The analytical solution is found using the Symbolic Math Toolbox. In particular, for time-varying $u_a(t)$, we obtain

```
>> [V,I]=dsolve('DV=I/C','DI=(-V-R*I+Va(t))/L')

V =
-(-C*int(diff(Va(t),t)*exp(1/2/L*t*R)*exp(-1/2/C/L*t*(R^2*C^2-4*C*L)^(1/2)),t)*exp(-
1/2/L*t*R+1/2/C/L*t*(R^2*C^2-
4*C*L)^(1/2))+C*int(diff(Va(t),t)*exp(1/2/L*t*R)*exp(1/2/C/L*t*(R^2*C^2-
4*C*L)^(1/2)),t)*exp(-1/2/L*t*R-1/2/C/L*t*(R^2*C^2-4*C*L)^(1/2))-C1*exp(-
```

```
1/2/L*t*R+1/2/C/L*t*(R^2*C^2-4*C*L)^(1/2))*(R^2*C^2-4*C*L)^(1/2)-C2*exp(-1/2/L*t*R-
1/2/C/L*t*(R^2*C^2-4*C*L)^(1/2))*(R^2*C^2-4*C*L)^(1/2))/(R^2*C^2-4*C*L)^(1/2)

I =
-1/2*(int(diff(Va(t),t)*exp(1/2/L*t*R)*exp(-1/2/C/L*t*(R^2*C^2-
4*C*L)^(1/2)),t)*R*C*exp(-1/2/L*t*R+1/2/C/L*t*(R^2*C^2-4*C*L)^(1/2))-
int(diff(Va(t),t)*exp(1/2/L*t*R)*exp(1/2/C/L*t*(R^2*C^2-4*C*L)^(1/2)),t)*R*C*exp(-
1/2/L*t*R-1/2/C/L*t*(R^2*C^2-
4*C*L)^(1/2))+int(diff(Va(t),t)*exp(1/2/L*t*R)*exp(1/2/C/L*t*(R^2*C^2-
4*C*L)^(1/2)),t)*exp(-1/2/L*t*R-1/2/C/L*t*(R^2*C^2-4*C*L)^(1/2))*(R^2*C^2-
4*C*L)^(1/2)+exp(-1/2/L*t*R+1/2/C/L*t*(R^2*C^2-
4*C*L)^(1/2))*int(diff(Va(t),t)*exp(1/2/L*t*R)*exp(-1/2/C/L*t*(R^2*C^2-
4*C*L)^(1/2)),t)*(R^2*C^2-4*C*L)^(1/2)-2*Va(t)*(R^2*C^2-4*C*L)^(1/2)+C1*R*exp(-
1/2/L*t*R+1/2/C/L*t*(R^2*C^2-4*C*L)^(1/2))*(R^2*C^2-4*C*L)^(1/2)+C1*R^2*C*exp(-
1/2/L*t*R+1/2/C/L*t*(R^2*C^2-4*C*L)^(1/2))-4*C1*L*exp(-1/2/L*t*R+1/2/C/L*t*(R^2*C^2-
4*C*L)^(1/2))+C2*R*exp(-1/2/L*t*R-1/2/C/L*t*(R^2*C^2-4*C*L)^(1/2))*(R^2*C^2-
4*C*L)^(1/2)-C2*R^2*C*exp(-1/2/L*t*R-1/2/C/L*t*(R^2*C^2-4*C*L)^(1/2))+4*C2*L*exp(-
1/2/L*t*R-1/2/C/L*t*(R^2*C^2-4*C*L)^(1/2)))/(R^2*C^2-4*C*L)^(1/2)
```

If the applied voltage (forcing function) is constant $u_a(t) = const$, we have

```
>> [V,I]=dsolve('DV=I/C','DI=(-V-R*I+Va)/L')

V =
C1*exp(-1/2/L*t*R+1/2/C/L*t*(R^2*C^2-4*C*L)^(1/2))
+C2*exp(-1/2/L*t*R-1/2/C/L*t*(R^2*C^2-4*C*L)^(1/2))

I =
-1/2*(-2*Va*C+C1*R*exp(-1/2/L*t*R+1/2/C/L*t*(R^2*C^2-4*C*L)^(1/2))*C
+C1*(R^2*C^2-4*C*L)^(1/2)*exp(-1/2/L*t*R+1/2/C/L*t*(R^2*C^2-4*C*L)^(1/2))
+C2*R*exp(-1/2/L*t*R-1/2/C/L*t*(R^2*C^2-4*C*L)^(1/2))*C
-C2*(R^2*C^2-4*C*L)^(1/2)*exp(-1/2/L*t*R-1/2/C/L*t*(R^2*C^2-4*C*L)^(1/2)))/C
```

Assigning the values of the resistance, inductance, and capacitance to be $R = 50$ ohm, $L = 0.25$ H, and $C = 0.01$ F, for $u_a(t) = 10$ V, we obtain

```
>> % [V,I]=dsolve('DV=I/C','DI=(-V-R*I+Va)/L')
>> R=50, L=0.25, C=0.01, Va=10, [V,I]=dsolve('DV=I/0.01','DI=(-V-50*I+10)/0.25')

R =
    50
L =
    0.2500
C =
    0.0100
Va =
    10

V =
1/2*C1*exp(-20*(5+2*6^(1/2))*t)
-5/24*C1*6^(1/2)*exp(20*(-5+2*6^(1/2))*t)
+5/24*C1*6^(1/2)*exp(-20*(5+2*6^(1/2))*t)+1/2*C1*exp(20*(-5+2*6^(1/2))*t)-
1/120*C2*6^(1/2)*exp(20*(-5+2*6^(1/2))*t)+1/120*C2*6^(1/2)*exp(-20*(5+2*6^(1/2))*t)
I =
-10/(5+2*6^(1/2))/(-5+2*6^(1/2))-1/24*(5*C1*6^(1/2)*exp(20*(-5+2*6^(1/2))*t)
-5*C1*6^(1/2)*exp(-20*(5+2*6^(1/2))*t)+12*C2*exp(-20*(5+2*6^(1/2))*t)
+5*C2*6^(1/2)*exp(20*(-5+2*6^(1/2))*t)
-5*C2*6^(1/2)*exp(-20*(5+2*6^(1/2))*t)
+12*C2*exp(20*(-5+2*6^(1/2))*t))/(5+2*6^(1/2))/(-5+2*6^(1/2))
```

where the constants $C1$ and $C2$ must be found by using the initial conditions.

The initial conditions can be easily incorporated. Assigning the initial conditions to be $[25, -5]^T$, we obtain

```
>> % [V,I]=dsolve('DV=I/C','DI=(-V-R*I+Va)/L'); R=50; L=0.25; C=0.01; Va=10;
>> [V,I]=dsolve('DV=I/0.01','DI=(-V-50*I+10)/0.25', 'V(0)=25, I(0)=-5')

V =
-5/2*exp(-20*(5+2*6^(1/2))*t)+11/12*6^(1/2)*exp(20*(-5+2*6^(1/2))*t)-
11/12*6^(1/2)*exp(-20*(5+2*6^(1/2))*t)-5/2*exp(20*(-5+2*6^(1/2))*t)

I =
-10/(5+2*6^(1/2))/(-5+2*6^(1/2))-1/24*(50*6^(1/2)*exp(20*(-5+2*6^(1/2))*t)-
50*6^(1/2)*exp(-20*(5+2*6^(1/2))*t)+180*exp(-20*(5+2*6^(1/2))*t)
```

```
+180*exp(20*(-5+2*6^(1/2))*t))/(5+2*6^(1/2))/(-5+2*6^(1/2))
```

The derived expressions can be simplified using the `simplify` function. In particular, `simplify(V)` and `simplify(I)` are used as demonstrated here:

```
>> V_simplify=simplify(V), I_simplify=simplify(I)

V_simplify =
-5/2*exp(-20*(5+2*6^(1/2))*t)+11/12*6^(1/2)*exp(20*(-5+2*6^(1/2))*t)-
11/12*6^(1/2)*exp(-20*(5+2*6^(1/2))*t)-5/2*exp(20*(-5+2*6^(1/2))*t)

I_simplify =
5/12*(-24-5*6^(1/2)*exp(20*(-5+2*6^(1/2))*t)+5*6^(1/2)*exp(-20*(5+2*6^(1/2))*t)-
18*exp(-20*(5+2*6^(1/2))*t)-18*exp(20*(-5+2*6^(1/2))*t))/(5+2*6^(1/2))/(-5+2*6^(1/2))
```

The derived expression for $u_C(t)$ is

$$u_C(t) = -\frac{5}{2}e^{-20(5+2\sqrt{6})t} + \frac{11}{12}\sqrt{6}e^{20(-5+2\sqrt{6})t} - \frac{11}{12}\sqrt{6}e^{-20(5+2\sqrt{6})t} - \frac{5}{2}e^{20(-5+2\sqrt{6})t}.$$

Figure 5.21 documents the plots for $u_C(t)$ and $i(t)$ which are found using the following statement:

```
t=0:0.001:3;
I_simplify=5/12*(-24-5*6^(1/2)*exp(20*(-5+2*6^(1/2))*t)
+5*6^(1/2)*exp(-20*(5+2*6^(1/2))*t)-18*exp(-20*(5+2*6^(1/2))*t)
-18*exp(20*(-5+2*6^(1/2))*t))/(5+2*6^(1/2))/(-5+2*6^(1/2));
plot(t,I_simplify); hold;
V_simplify=-5/2*exp(-20*(5+2*6^(1/2))*t)
+11/12*6^(1/2)*exp(20*(-5+2*6^(1/2))*t)
-11/12*6^(1/2)*exp(-20*(5+2*6^(1/2))*t)-5/2*exp(20*(-5+2*6^(1/2))*t);
plot(t,V_simplify)
```

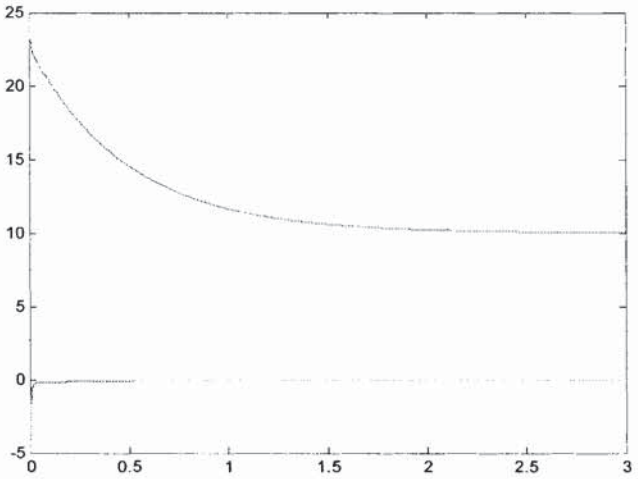


Figure 5.21. Dynamics for $u_C(t)$ and $i(t)$ □

The state-space modeling concept is widely used in simulation and analysis. The state, control (forcing function), and output variables are used. The state-space techniques are commonly applied in simulation and analysis of dynamic systems in the MATLAB environment. Mathematical models of dynamic systems are found in the form of linear and nonlinear differential equations. In general, a set of n first-order linear ordinary differential equations with n states $x \in \mathbb{R}^n$ and m controls (forcing functions) $u \in \mathbb{R}^m$ is written as [4]

$$\frac{dx}{dt}=\begin{bmatrix}\frac{dx_1}{dt}\\ \frac{dx_2}{dt}\\ \vdots\\ \frac{dx_{n-1}}{dt}\\ \frac{dx_n}{dt}\end{bmatrix}=\begin{bmatrix}a_{11} & a_{12} & \cdots & a_{1n-1} & a_{1n}\\ a_{21} & a_{22} & \cdots & a_{2n-1} & a_{2n}\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ a_{n-11} & a_{n-12} & \cdots & a_{n-1n-1} & a_{n-1n}\\ a_{n1} & a_{n2} & \cdots & a_{nn-1} & a_{nn}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ \vdots\\ x_{n-1}\\ x_n\end{bmatrix}+\begin{bmatrix}b_{11} & b_{12} & \cdots & b_{1m-1} & b_{1m}\\ b_{21} & b_{22} & \cdots & b_{2m-1} & b_{2m}\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ b_{n-11} & b_{n-12} & \cdots & b_{n-1m-1} & b_{n-1m}\\ b_{n1} & b_{n2} & \cdots & b_{nm-1} & b_{nm}\end{bmatrix}\begin{bmatrix}u_1\\ u_2\\ \vdots\\ u_{m-1}\\ u_m\end{bmatrix}=Ax+Bu,$$

where $A\in\mathbb{R}^{n\times m}$ and $B\in\mathbb{R}^{n\times m}$ are the matrices of coefficients.

The output equation is expressed as

$y= Hx + Du,$

where $H\in\mathbb{R}^{b\times n}$ and $D\in\mathbb{R}^{b\times m}$ are the matrices of coefficients.

Nonlinear multivariable dynamic systems are modeled by a set of n first-order nonlinear differential equations

$$\frac{dx}{dt}=F(t,x,u),\ x(t_0)=x_0,$$

where t is the time; $F(t,x,u)$ is the nonlinear function.

In the first section of this chapter we considered the aircraft. The aircraft outputs are the Euler angles, and the fighter is controlled by deflecting the control surfaces. The multi-input (eight control surfaces) - multi-output (three Euler angles θ, ϕ, and ψ to be controlled) nature is obvious. The pilot assigns the desired Euler angles r_θ, r_ϕ, and r_ψ (pedal and stick reference commands). Using the errors between the reference vector $r=\begin{bmatrix}r_\theta\\ r_\phi\\ r_\psi\end{bmatrix}$ and output vector $y=\begin{bmatrix}\theta\\ \phi\\ \psi\end{bmatrix}$, as defined by $e=r-y=\begin{bmatrix}r_\theta\\ r_\phi\\ r_\psi\end{bmatrix}-\begin{bmatrix}\theta\\ \phi\\ \psi\end{bmatrix}$, the controller $u=\Pi(e,x)$ calculates the control inputs (control surface deflections). The aircraft outputs (θ, ϕ and ψ) can be obtained by using the state variables (v, α, q, θ, β, p, r, ϕ and ψ).

Figure 5.22 shows the block diagram representation of the multivariable aircraft with nine states $x\in\mathbb{R}^9$ ($v, \alpha, q, \theta, \beta, p, r, \phi, \psi$), eight control surfaces $u\in\mathbb{R}^8$ (right and left horizontal stabilizers, right and left leading- and trailing-edge flaps, right and left rudders), three outputs $y\in\mathbb{R}^3$ (θ, ϕ, ψ), and three reference inputs $r\in\mathbb{R}^3$ (r_θ, r_ϕ, r_ψ) [4].

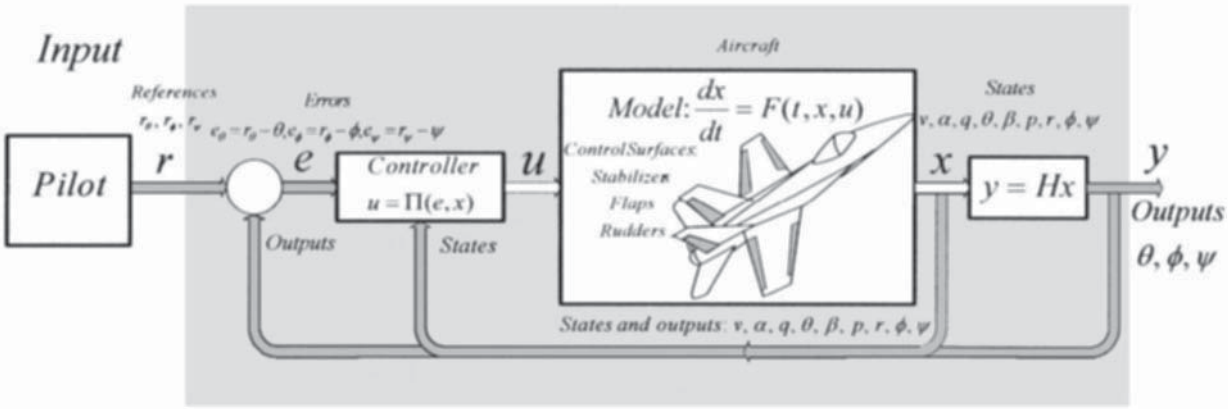


Figure 5.22. Block diagram representation of a multi-input/multi-output aircraft

The functional block diagram of nonlinear multivariable dynamic systems (n states, m controls, b reference inputs, and b outputs), which are described by

state-space equation $\dfrac{dx}{dt} = F(t,x,u), x(t_0) = x_0$

output equation $y = Hx$,

is illustrated in Figure 5.23.

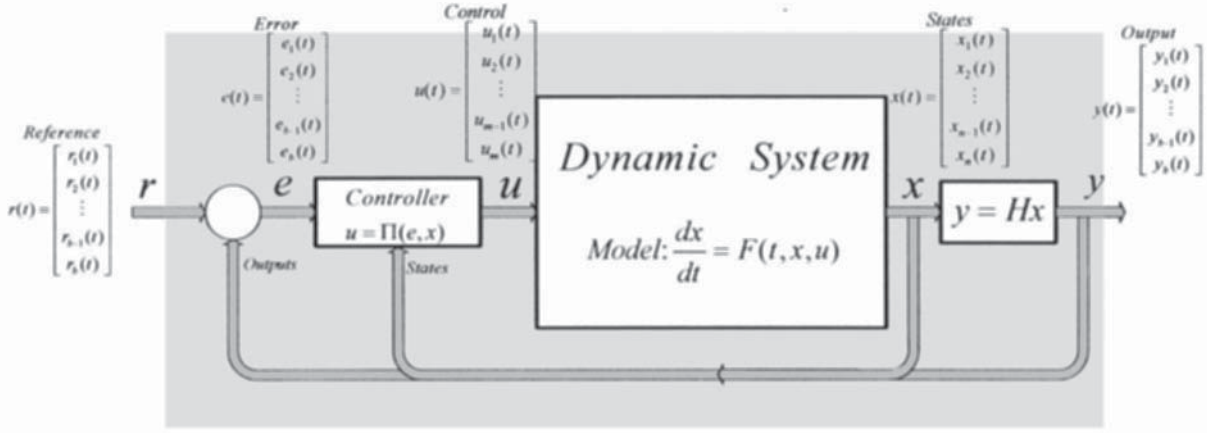


Figure 5.23. Functional block diagram of multi-input/multi-output dynamic systems

Example 5.3.4.
Consider the aircraft described by the sate-space differential equations
$\dot{x} = Ax + Bu$, $y = Hx + Du$,

$$\frac{dx}{dt}=\begin{bmatrix}\frac{dx_1}{dt}\\ \frac{dx_2}{dt}\\ \frac{dx_3}{dt}\\ \frac{dx_4}{dt}\\ \frac{dx_5}{dt}\\ \frac{dx_6}{dt}\end{bmatrix}=Ax+Bu=A\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\\ x_5\\ x_6\end{bmatrix}+Bu,\ y=Hx+Du=H\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\\ x_5\\ x_6\end{bmatrix}+Du$$

where

$$A=\begin{bmatrix}-20 & -150 & -1000 & -5000 & -12500 & -15000\\ 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0\end{bmatrix},\ B=\begin{bmatrix}1\\ 0\\ 0\\ 0\\ 0\\ 0\end{bmatrix},\ H=\begin{bmatrix}0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}\text{ and } D=[0]$$

Perform numerical simulations using the `lsim` MATLAB solver.

Solution.

The description of the `lsim` solver is given below.

```
>> help lsim
 LSIM  Simulate time response of LTI models to arbitrary inputs.

    LSIM(SYS,U,T) plots the time response of the LTI model SYS to the
    input signal described by U and T.  The time vector T consists of
    regularly spaced time samples and U is a matrix with as many columns
    as inputs and whose i-th row specifies the input value at time T(i).
    For example,
           t = 0:0.01:5;    u = sin(t);    lsim(sys,u,t)
    simulates the response of a single-input model SYS to the input
    u(t)=sin(t) during 5 seconds.

    For discrete-time models, U should be sampled at the same rate as SYS
    (T is then redundant and can be omitted or set to the empty matrix).
    For continuous-time models, choose the sampling period T(2)-T(1) small
    enough to accurately describe the input U.  LSIM issues a warning when
    U is undersampled and hidden oscillations may occur.

    LSIM(SYS,U,T,X0) specifies the initial state vector X0 at time T(1)
    (for state-space models only).  X0 is set to zero when omitted.

    LSIM(SYS1,SYS2,...,U,T,X0) simulates the response of multiple LTI
    models SYS1,SYS2,... on a single plot.  The initial condition X0
    is optional.  You can also specify a color, line style, and marker
    for each system, as in
       lsim(sys1,'r',sys2,'y--',sys3,'gx',u,t).

    Y = LSIM(SYS,U,T) returns the output history Y.  No plot is drawn on
    the screen.  The matrix Y has LENGTH(T) rows and as many columns as
    outputs in SYS.  For state-space models,
       [Y,T,X] = LSIM(SYS,U,T,X0)
    also returns the state trajectory X, a matrix with LENGTH(T) rows
    and as many columns as states.

    For continuous-time models,
       LSIM(SYS,U,T,X0,'zoh')  or  LSIM(SYS,U,T,X0,'foh')
    explicitly specifies how the input values should be interpolated
```

```
    between samples (zero-order hold or linear interpolation).  By
    default, LSIM selects the interpolation method automatically based
    on the smoothness of the signal U.

    See also GENSIG, STEP, IMPULSE, INITIAL, LTIMODELS.
```

Using this description, we download four matrices, e.g.,

```
A=[-15 -150 -1100 -4500 -12500 -15000;
     1    0     0     0      0      0;
     0    1     0     0      0      0;
     0    0     1     0      0      0;
     0    0     0     1      0      0;
     0    0     0     0      1      0];
B=[1 0 0 0 0 0]';
H=[0 0 0 0 0 1];
D=[0];
```

enter the simulation duration as 5 seconds, assign the step input (command), and letting the initial conditions for six variables be $[1\ 5\ 0\ -5\ -10\ -20\ ]^T$

```
t=0:.01:5; u=ones(size(t)); x0=[0 10 50 100 -50 -10];
```

Typing in the Command Window

```
>> [y,x]=lsim(A,B,H,D,u,t,x0); plot(t,x)
```

and pressing the Enter key, the transient dynamics of the aircraft result. Figure 5.24 represents the aircraft's state variable behavior. If one needs to plot the output transient, it can be done using the following statement:

```
>> plot(t,y)
```

The resulting plot is illustrated in Figure 5.24.

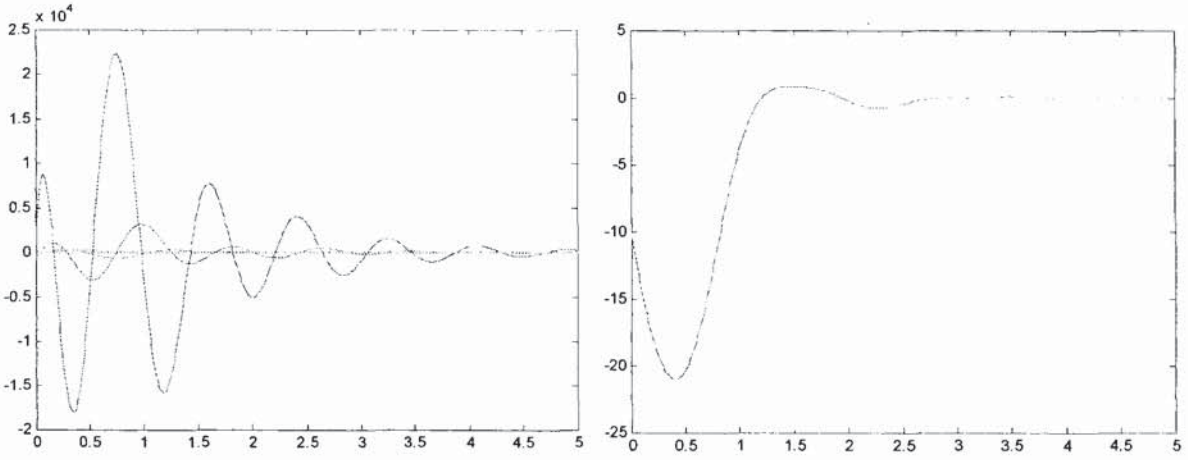


Figure 5.24. State variables and output evolutions due to step input and initial conditions

The "system" methodology is illustrated for the example under consideration. In particular, we use the `help lsim`. The user can define the "system" with six state variables, one control, and one output. Then, using the `lsim` solver, we numerically simulate the "systems" and find the output responses due to the initial conditions and unit step input.

```
States={'x1' 'x2' 'x3' 'x4' 'x5' 'x6'}
Control={'u'}
Output={'y'}
A=[-15 -150 -1100 -4500 -12500 -15000;
     1    0     0     0      0      0;
```

```
     0     1     0     0     0     0;
     0     0     1     0     0     0;
     0     0     0     1     0     0;
     0     0     0     0     1     0];
B=[1 0 0 0 0 0]';
H=[0 0 0 0 0 1];
D=[0];
System=ss(A,B,H,D,'statename',States,'inputname',Control,'outputname',Output);
System
t=0:.01:5; u=ones(size(t)); x0=[0 10 50 100 -50 -10];
lsim(System,u,t);plot(t,y); pause
step(System); % step response with zero initial conditions
```

The following system description results in the Command Window

```
States =
    'x1'    'x2'    'x3'    'x4'    'x5'    'x6'

Control =
    'u'

Output =
    'y'

a =
              x1          x2          x3          x4          x5          x6
   x1        -15        -150       -1100       -4500  -1.25e+004   -1.5e+004
   x2          1           0           0           0           0           0
   x3          0           1           0           0           0           0
   x4          0           0           1           0           0           0
   x5          0           0           0           1           0           0
   x6          0           0           0           0           1           0

b =
       u
   x1  1
   x2  0
   x3  0
   x4  0
   x5  0
   x6  0

c =
       x1  x2  x3  x4  x5  x6
   y    0   0   0   0   0   1

d =
       u
   y   0

Continuous-time model.
```

The resulting dynamics are documented in Figure 5.25.

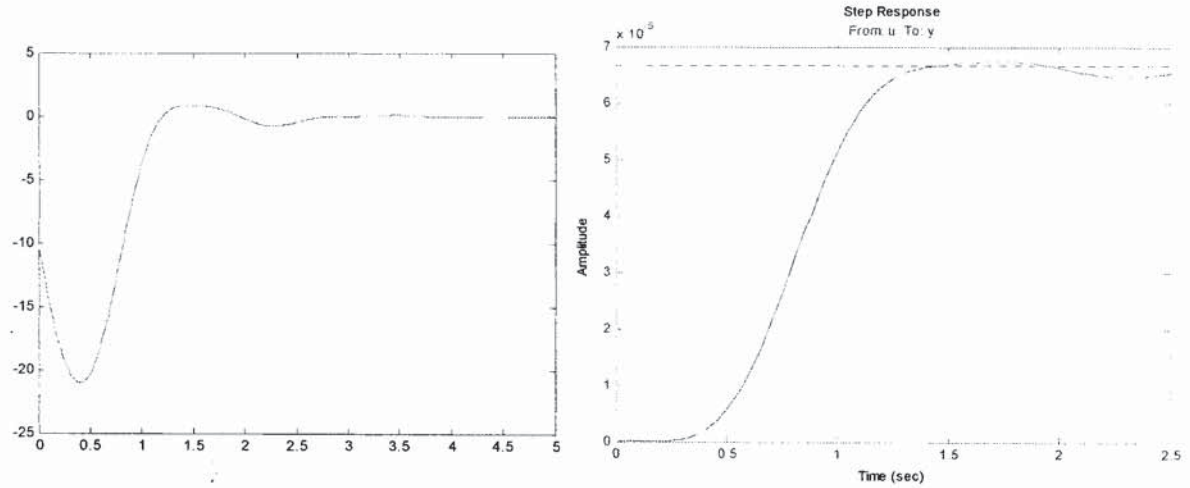


Figure 5.25. Output dynamics due to initial conditions and unit step input

□

Example 5.3.5.

Using the state-space concept, develop the state-space model of the series *RLC* circuit illustrated in Figure 5.26. The voltage across the capacitor is the circuit output. Find matrices A, B, H, and D of the state-space model and the output equation. Perform numerical simulations in MATLAB assigning the following parameters: $R = 1$ ohm, $L = 0.1$ H, and $C = 0.5$ F.

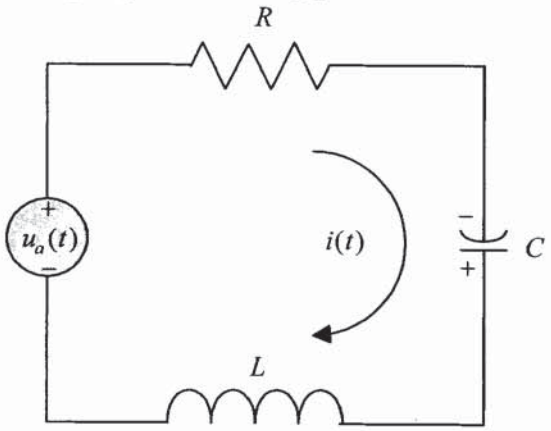


Figure 5.26. *RLC* circuit

Solution.

Using Kirchhoff's law, which gives the following equations

$$C\frac{du_C}{dt} = i \text{ and } L\frac{di}{dt} = -u_C - Ri + u_a(t),$$

a set of two first-order differential equations to model the circuitry dynamics is found to be

$$\frac{du_C}{dt} = \frac{1}{C}i,$$

$$\frac{di}{dt} = \frac{1}{L}\left(-u_C - Ri + u_a(t)\right).$$

The state and control variables are denoted as

$x_1(t) = u_C(t)$, $x_2(t) = i(t)$ and $u(t) = u_a(t)$.

Thus, we have

$$\frac{dx}{dt} = \begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{bmatrix} = \begin{bmatrix} \frac{du_C}{dt} \\ \frac{di}{dt} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{C} \\ -\frac{1}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} u_C \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} u_a = Ax + Bu.$$

The matrices of coefficients are found to be $A = \begin{bmatrix} 0 & \frac{1}{C} \\ -\frac{1}{L} & -\frac{R}{L} \end{bmatrix}$ and $B = \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix}$.

The voltage across the capacitor is the output. Hence, $y(t) = u_C(t)$. The output equation is

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} u_C \\ i \end{bmatrix} = [0]u_a = Hx + Du,$$

where $H = \begin{bmatrix} 1 & 0 \end{bmatrix}$ and $D = \begin{bmatrix} 0 \end{bmatrix}$.

The simulation is performed assigning R = 1 ohm, L = 0.1 H, and C = 0.5 F. Correspondingly, we find the numerical values for matrices to be

$$A = \begin{bmatrix} 0 & \frac{1}{C} \\ -\frac{1}{L} & -\frac{R}{L} \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ -10 & -10 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} = \begin{bmatrix} 0 \\ 10 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 0 \end{bmatrix}, \text{ and } D = [0].$$

To perform numerical simulations, we download these four matrices:

```
>> A=[0 2; -10 -10], B=[0 10]', H=[1 0], D=[0]
```

These matrices are stored in the memory, and the following matrices are displayed:

```
A =
     0     2
   -10   -10
B =
     0
    10
H =
     1     0
D =
     0
```

Assigning the simulation duration to be 2 seconds, assigning the step input (command) with the magnitude 10, and letting the initial conditions be $[-5\ -10\ ]^T$, we type

```
>> t=0:.001:2; u=10*ones(size(t)); x0=[-5 -10];
```

Typing in the Command Window

```
>> [y,x]=lsim(A,B,H,D,u,t,x0); plot(t,x)
```

and pressing the Enter key, the transient dynamics of the circuit result. Figure 5.27 represents the states variable behavior. If we need to plot the output transient (y) and the input command (u), the user types the following statements:

```
>> plot(t,y), hold, plot(t,u,'+')
```

or

```
>> plot(t,y,t,u,'+')
```

The plots are shown in Figure 5.27.

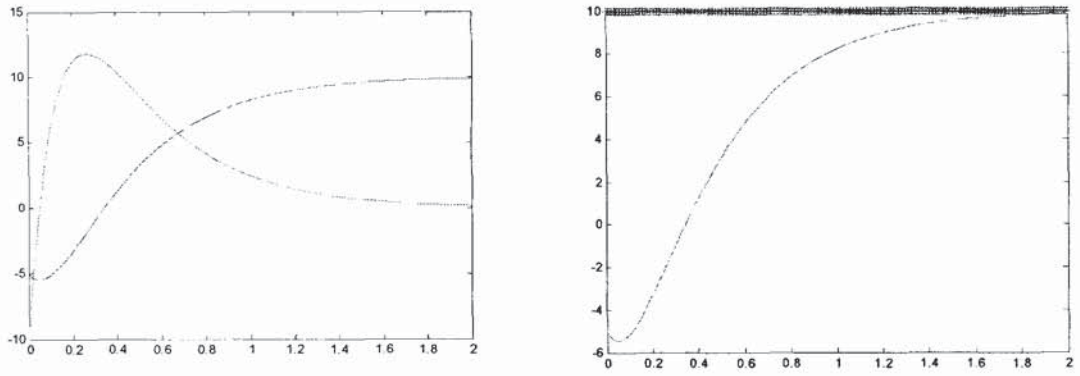


Figure 5.27. State variables and output behavior due to input function and initial conditions

□

Example 5.3.6. Mathematical model of permanent-magnet direct-current motors

Develop a mathematical model and build an *s*-domain block diagram for permanent-magnet DC motors. A schematic diagram of a permanent-magnet DC machine (motor and generator operation) is illustrated in Figure 5.28 [3]. Perform numerical simulations for a permanent-magnet DC motor in MATLAB assigning the following motor parameters: $r_a = 1$ ohm, $k_a = 0.1$, $L_a = 0.01$ H, $B_m = 0.005$ and $J = 0.001$ kg-m^2.

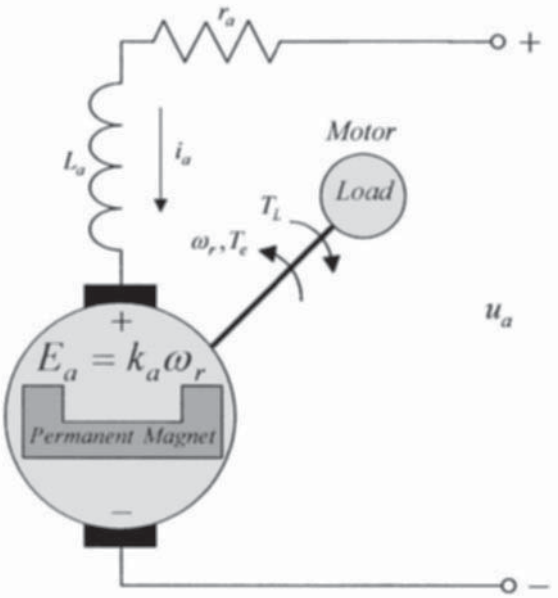


Figure 5.28. Schematic diagram of a permanent-magnet DC motor

Solution.

The flux, established by the permanent magnets, is constant. Applying Kirchhoff's voltage and Newton's second laws, the differential equations for permanent-magnet DC motors are derived using the motor representation documented in Figure 5.28. Denoting the *back emf*

and *torque* constants as k_a, we have the following differential equations describing the armature winding and *torsional-mechanical* dynamics [3, 4]:

$$\frac{di_a}{dt} = -\frac{r_a}{L_a} i_a - \frac{k_a}{L_a}\omega_r + \frac{1}{L_a} u_a, \quad \text{(motor circuitry dynamics)}$$

$$\frac{d\omega_r}{dt} = \frac{k_a}{J} i_a - \frac{B_m}{J}\omega_r - \frac{1}{J} T_L. \quad \text{(\textit{torsional-mechanical} dynamics)}$$

Augmenting these two first-order differential equations, in the state-space form we have

$$\begin{bmatrix} \frac{di_a}{dt} \\ \frac{d\omega_r}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{r_a}{L_a} & -\frac{k_a}{L_a} \\ \frac{k_a}{J} & -\frac{B_m}{J} \end{bmatrix} \begin{bmatrix} i_a \\ \omega_r \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} \\ 0 \end{bmatrix} u_a - \begin{bmatrix} 0 \\ \frac{1}{J} \end{bmatrix} T_L.$$

An *s*-domain block diagram of permanent-magnet DC motors is developed and shown in Figure 5.29 (this block-diagram is of particular importance if we use SIMULINK).

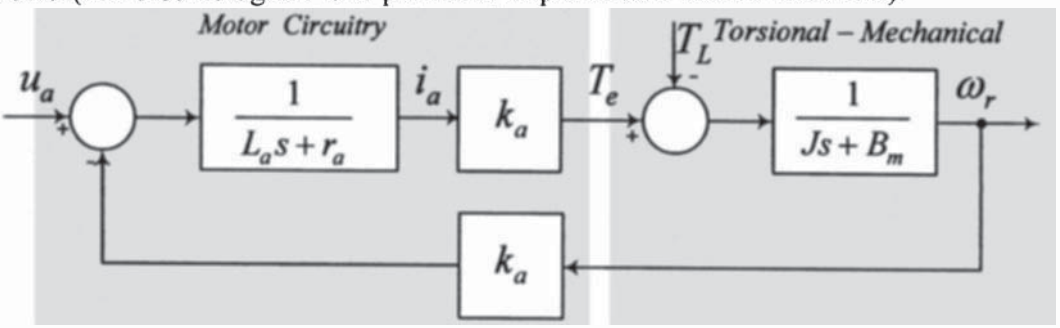


Figure 5.29. Block diagram of permanent-magnet DC motors

As assigned, the simulation is performed assuming the following parameters: $r_a = 1$, $k_a = 0.1$, $L_a = 0.01$, $B_m = 0.005$ and $J = 0.001$. Taking note that the motor output is the angular velocity, we use the following matrices:

$$A = \begin{bmatrix} -100 & -10 \\ 100 & -5 \end{bmatrix}, \quad B = \begin{bmatrix} 100 \\ 0 \end{bmatrix}, \quad H = \begin{bmatrix} 0 & 1 \end{bmatrix} \text{ and } D = [0].$$

To perform numerical simulations, we download these matrices:

```
>> A=[-100 -10; 100 5], B=[100 0]', H=[0 1], D=[0]
```

Assigning the simulation duration to be 1 second, letting the applied voltage be 10 V (u_a = 10 V), and setting the initial conditions for two variables to be $[0 \;\; 0]^T$ ($i_a = 0$ A and $\omega_r = 0$ rad/sec), we type in the Command Window

```
>> t=0:.001:1; u=10*ones(size(t)); x0=[0 0];
>> [y,x]=lsim(A,B,H,D,u,t,x0);
>> plot(t,x); xlabel('Time (seconds)'); title('Angular velocity and current');
```

The motor state variables are plotted in Figure 5.30, e.g., two states $i_a(t)$ and $\omega_r(t)$ are documented. The simulation results illustrate that the motor reaches the angular velocity 200 rad/sec within 1 sec.

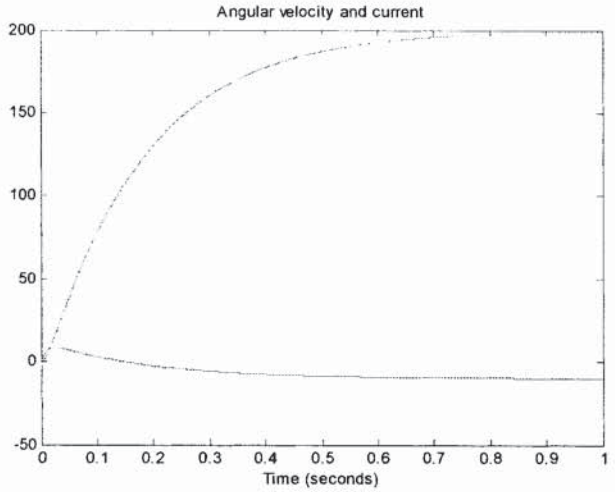


Figure 5.30. Motor state variables dynamics: behavior of $i_a(t)$ and $\omega_r(t)$ states

The simulation of permanent-magnet DC motors is also reported in Examples 6.2.2 and 6.2.6 using SIMULINK. □

REFERENCES

1. *MATLAB 6.5 Release 13,* CD-ROM, MathWorks, Inc., 2002.
2. Kuo, B. C., *Automatic Control Systems*, Prentice Hall, Englewood Cliffs, NJ, 1995.
3. Lyshevski, S. E., *Electromechanical Systems, Electric Machines, and Applied Mechatronics*, CRC Press, Boca Raton, FL, 2000.
4. Lyshevski, S. E., *Control Systems Theory with Engineering Applications*, Birchauser, Boston, MA, 2002.
5. Ogata, K., *Modern Control Engineering*, Prentice-Hall, Upper Saddle River, NJ, 2001.