System Dynamics (22.554 & 24.509)

III. The State Space Equations and Their Time Domain Solution

Introduction

Up to now we have addressed some of the mathematical preliminaries necessary for the study of dynamic systems from the state space viewpoint. Our goal is to be able to analyze general systems which may contain several inputs and outputs. For lumped-parameter models, it is always possible to describe such systems in terms of a coupled set of first-order differential (or difference) equations. Also, for convenience, we write the expressions in a specific way so that the nonlinear and variable coefficient terms can be isolated. This allows us to easily distinguish between linear stationary systems and those containing nonlinear and/or variable coefficient terms. Linear stationary systems allow analytic solutions and, therefore, they warrant special treatment.

In this section, we concentrate on three areas. Our first goal is to define precisely a general notation for the state variable treatment of dynamic systems. Once the basic notation and so-called ***standard form*** is established, the emphasis shifts to methods for putting general systems into standard form. In particular, we will look at specific techniques for the state space representation of nth order systems, for embedded statics (mixed algebraic and differential equations), and for the linearization of nonlinear/non-stationary systems. Finally, once it is clear that most systems can be modeled using the standard state space equations, we look at methods for the time domain solution of these equations. Our solution techniques cover both analytical methods (where appropriate) and numerical methods. This section of notes, when complete, should give the student a good background in the time domain simulation of dynamic systems.

The State Space Representation (Continuous Systems)

The general state variable form for continuous dynamic systems can be written as

$$\frac{d}{dt}\underline{x}(t) = \underline{\underline{A}}\,\underline{x}(t) + \underline{f}(\underline{x}, t) + \underline{g}(t) \tag{3.1}$$

where

$\underline{x}(t)$ = state vector (contains variables of interest)

$\underline{\underline{A}}$ = constant coefficient system matrix

$\underline{f}(\underline{x}, t)$ = vector containing the nonlinear and variable coefficient terms

$\underline{g}(t)$ = vector forcing function

If we collect the deterministic inputs in a vector, $\underline{u}(t)$, and assume that the desired outputs can be written as some linear combination of the state vector and input vector, then

$$\underline{y}(t) = \underline{\underline{C}}\,\underline{x}(t) + \underline{\underline{D}}\,\underline{u}(t) \tag{3.2}$$

where $\underline{\underline{C}}$ and $\underline{\underline{D}}$ are constant matrices that specify a particular relationship between $\underline{y}(t)$, $\underline{x}(t)$, and $\underline{u}(t)$. To complete this description, we note that, in general, the independent input, system output, and state vectors do not contain the same number of elements. To treat this, one can write $\underline{g}(t)$ as

$$\underline{g}(t) = \underline{\underline{B}}\underline{u}(t)$$

and allow the $\underline{\underline{B}}$, $\underline{\underline{C}}$, and $\underline{\underline{D}}$ matrix operators to be rectangular matrices (i.e. non-square). In the above treatment, the system matrix, $\underline{\underline{A}}$, is required to be square.

A particular element of eqn. (3.1) can be written as

$$\frac{d}{dt}x_i(t) = \sum_j^N a_{ij}x_j(t) + f_i(\underline{x}, \underline{u}, t) + \sum_k^M b_{ik}u_k(t) \tag{3.3}$$

and for eqn. (3.2), we have

$$y_\ell(t) = \sum_j^N c_{\ell j}x_j(t) + \sum_k^M d_{\ell k}u_k(t) \tag{3.4}$$

where we have an Nth order system with M inputs and P outputs (i.e. ℓ = 1, 2, ⋯ , P). Example 3.1 illustrates the use of eqns. (3.1) and (3.2) for a simple second-order system.

The most general form of eqn. (3.1) can be written as

$$\frac{d}{dt}\underline{x}(t) = \underline{f}(\underline{x}, \underline{u}, t) \tag{3.5}$$

where the vector function $\underline{f}(\underline{x}, \underline{u}, t)$, in this case, contains the full inter-relationship among the state variables and independent inputs. In this form, the desired system outputs are simply written as a general function of the inputs and state vector at each time point. Thus, the system outputs are simply denoted by the vector function $\underline{y}(\underline{x}, \underline{u}, t)$.

Equations (3.1) and (3.2) are usually used for linear stationary systems or for nonlinear or non-stationary systems that will be ***linearized*** about some state point given by $\underline{x}_r, \underline{u}_r$. The linear stationary form has been studied extensively over the years and a variety of very useful analytical tools are available for simulation and for further detailed study of these systems. Equation (3.5), however, is more general, but its use usually implies that only numerical simulation techniques will be used in the study of the system's dynamic behavior.

Example 3.1 Standard State Space Form

Problem Statement:

Put the following 2nd order system into standard state form.

$$\frac{d}{dt}x_1(t) = 7x_1(t) + 3x_2(t) + 4tx_1(t) + x_1(t)x_2(t) - u_1(t) + 2u_3(t)$$

$$\frac{d}{dt}x_2(t) = 9x_1(t) - 5x_2(t) - 3x_2^2(t) + 4u_2(t)$$

and

$$y(t) = x_1(t) + x_2(t) - 2u_2(t)$$

Problem Solution:

Putting this 2nd order, non-stationary, nonlinear system into standard state space form, gives

$$\frac{d}{dt}\underline{x}(t) = \underline{\underline{A}}\,\underline{x}(t) + \underline{f}(\underline{x},t) + \underline{\underline{B}}\,\underline{u}(t)$$

and $\underline{y} = \underline{\underline{C}}\,\underline{x} + \underline{\underline{D}}\,\underline{u}$

where $\underline{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$ $\quad \underline{u}(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \end{bmatrix}$ $\quad \underline{y}(t) = \begin{bmatrix} y(t) \end{bmatrix}$

$$\underline{\underline{A}} = \begin{bmatrix} 7 & 3 \\ 9 & -5 \end{bmatrix} \qquad \underline{f}(\underline{x},t) = \begin{bmatrix} 4tx_1(t) + x_1(t)x_2(t) \\ -3x_2^2(t) \end{bmatrix} \qquad \underline{\underline{B}} = \begin{bmatrix} -1 & 0 & 2 \\ 0 & 4 & 0 \end{bmatrix}$$

and $\underline{\underline{C}} = \begin{bmatrix} 1 & 1 \end{bmatrix}$ $\quad \underline{\underline{D}} = \begin{bmatrix} 0 & -2 & 0 \end{bmatrix}$

This case has two states, three inputs, and a single output variable.

Conversion to State Form

The above development is quite general and most systems of interest can be cast into this state space formulation. Before addressing the solution of these expressions, let's first look at how to put any system into this form. If we can do this, and if we can develop methods for the solution of eqn. (3.1) or eqn. (3.5), then we will have a general analysis procedure applicable to all dynamic systems. This is what we seek!

The goal here is to present a recipe for the conversion of certain classes of problems into state form. Three specific areas to be addressed are:

1. State space representation of nth order systems,
2. Treatment of mixed algebraic and differential equations, and
3. Linearization of nonlinear systems.

A sample illustration for working with each of these categories of problems is also given.

Conversion of nth Order Linear Differential Equation which Contains Derivatives of the Forcing Function

The following development is a specific recipe for performing the desired conversion. Since there is no unique choice of elements to include in the state vector, this particular algorithm is only one possible choice. However, it is quite useful for many applications and it can be modified as necessary to meet any special needs.

Given the following n^{th} order system,

$$\begin{aligned}\frac{d^n}{dt^n}y(t)+a_1\frac{d^{n-1}}{dt^{n-1}}y(t)+\quad\cdots\quad+a_{n-1}\frac{d}{dt}y(t)+a_n y(t)=\\ b_0\frac{d^n}{dt^n}u(t)+b_1\frac{d^{n-1}}{dt^{n-1}}u(t)+\quad\cdots\quad+b_{n-1}\frac{d}{dt}u(t)+b_n u(t)\end{aligned} \tag{3.6}$$

let

$$x_1(t)=y(t)-\beta_0 u(t)$$

$$x_2(t)=\frac{d}{dt}y(t)-\beta_0\frac{d}{dt}u(t)-\beta_1 u(t)=\frac{d}{dt}x_1(t)-\beta_1 u(t)$$

$$x_3(t)=\frac{d^2}{dt^2}y(t)-\beta_0\frac{d^2}{dt^2}u(t)-\beta_1\frac{d}{dt}u(t)-\beta_2 u(t)=\frac{d}{dt}x_2(t)-\beta_2 u(t)$$

or, in general, for the jth state variable

$$x_j(t)=\frac{d^{j-1}}{dt^{j-1}}y(t)-\beta_0\frac{d^{j-1}}{dt^{j-1}}u(t)-\beta_1\frac{d^{j-2}}{dt^{j-2}}u(t)-\quad\cdots\quad-\beta_{j-1}u(t)$$

or $$x_j(t)=\frac{d}{dt}x_{j-1}(t)-\beta_{j-1}u(t)\qquad\text{for } j>1 \tag{3.7}$$

In the above definition of the elements for the state vector, β_j is defined as

$$\begin{aligned}\beta_0&=b_0\\ \beta_1&=b_1-a_1\beta_0\\ \beta_2&=b_2-a_1\beta_1-a_2\beta_0\\ &\vdots\\ \beta_j&=b_j-a_1\beta_{j-1}-\quad\cdots\quad-a_{j-1}\beta_1-a_j\beta_0\end{aligned} \tag{3.8}$$

Using eqns. (3.7) and (3.8), one gets the following SISO state variable representation of eqn. (3.6):

$$\frac{d}{dt}\underline{x}(t) = \underline{\underline{A}}\,\underline{x}(t) + \underline{\underline{B}}\,u(t) \tag{3.9}$$

$$\underline{y}(t) = \underline{\underline{C}}\,\underline{x}(t) + \underline{\underline{D}}\,\underline{u}(t) \tag{3.10}$$

where

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \underline{\underline{B}} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix} \qquad \underline{\underline{C}} = \begin{bmatrix} 1 & 0 & 0 & \cdots \end{bmatrix} \qquad \underline{\underline{D}} = \beta_0 = b_0 \tag{3.11}$$

and

$$\underline{\underline{A}} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ & & \vdots & & \\ 0 & 0 & 0 & \cdots & 1 \\ -a_n & -a_{n-1} & -a_{n-2} & \cdots & -a_1 \end{bmatrix} \tag{3.12}$$

In the standard formulation for conversion of an nth order system, the state matrix will be an n×n square matrix, but the input $\underline{\underline{B}}$ matrix and output matrix $\underline{\underline{C}}$ are n×1 and 1×n vectors, respectively, and $\underline{\underline{D}}$ is a 1×1 scalar. Note also that this is a single-input single-output (SISO) system so that both $\underline{u}$ and $\underline{y}$ are 1×1 scalars and $\underline{x}$ is the n×1 state vector. Example 3.2 illustrates the use of the above general conversion algorithm.

Note that, for the special case where there are no derivatives of the forcing function (i.e. for the RHS a function of u(t) only), we have $\underline{\underline{B}} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix}^T$ and $\underline{\underline{D}} = 0$. This latter situation is actually the usual situation, and it is much easier to visualize and implement the conversion algorithm (since all the β_j's are zero except $\beta_n = 1$). Example 3.3 illustrates this less general, but very common case. It also highlights the importance of the state matrix in determining the solutions to linear stationary systems. We will focus further on the properties of the state matrix in later subsections.

Example 3.2 Conversion of 3rd Order System to State Form (General Case)

Problem Statement:

Given $\frac{d^3}{dt^3}y(t)+6\frac{d^2}{dt^2}y(t)-8\frac{d}{dt}y(t)+4y(t)=2\frac{d}{dt}u(t)+7u(t)$

convert this 3rd order linear stationary system to state form.

Problem Solution:

In general form, the given 3rd order equation can be written as

$$\frac{d^3}{dt^3}y(t)+a_1\frac{d^2}{dt^2}y(t)+a_2\frac{d}{dt}y(t)+a_3y(t)=b_0\frac{d^3}{dt^3}u(t)+b_1\frac{d^2}{dt^2}u(t)+b_2\frac{d}{dt}u(t)+b_3u(t)$$

where the explicit values of the a_j and b_j coefficients are determined via a one-to-one correspondence with the given ODE.

Thus, using the recipe given previously, we have

$$x_j=\frac{d}{dt}x_{j-1}-\beta_{j-1}u \qquad \rightarrow \qquad x_1=y-\beta_0u=y$$

with $x_1=y-\beta_0u$

$$x_2=\frac{dy}{dt}-\beta_0\frac{du}{dt}-\beta_1u=\frac{dy}{dt}$$

$$x_3=\frac{d^2y}{dt^2}-\beta_0\frac{d^2u}{dt^2}-\beta_1\frac{du}{dt}-\beta_2u=\frac{d^2y}{dt^2}-2u$$

and $\beta_j=b_j-a_1\beta_{j-1}-\cdots-a_j\beta_0 \quad \rightarrow \quad \beta_0=b_0=0$

with $\beta_0=b_0$

$$\beta_1=b_1-a_1\beta_0=0$$

$$\beta_2=b_2-a_1\beta_1-a_2\beta_0=2$$

$$\beta_3=b_3-a_1\beta_2-a_2\beta_1-a_3\beta_0=7-6(2)=-5$$

Therefore
$$\frac{d}{dt}\begin{bmatrix}x_1(t)\\x_2(t)\\x_3(t)\end{bmatrix}=\begin{bmatrix}0&1&0\\0&0&1\\-4&8&-6\end{bmatrix}\begin{bmatrix}x_1(t)\\x_2(t)\\x_3(t)\end{bmatrix}+\begin{bmatrix}0\\2\\-5\end{bmatrix}u(t)$$

and
$$y(t)=\begin{bmatrix}1&0&0\end{bmatrix}\begin{bmatrix}x_1(t)\\x_2(t)\\x_3(t)\end{bmatrix}+\begin{bmatrix}0\end{bmatrix}u(t)$$

Example 3.3 Conversion of 3rd Order System to State Form (Usual Case)

Problem Statement:

Given $\frac{d^3}{dt^3}y(t)+a_1\frac{d^2}{dt^2}y(t)+a_2\frac{d}{dt}y(t)+a_3y(t)=u(t)$

convert this system to state form, noting that the RHS does not contain derivatives of u(t). Also show that the eigenvalues of the resultant system matrix are identical to the roots of the characteristic equation associated with the homogeneous solution to the given 3rd-order equation.

Problem Solution:

For this case, let's make the following substitutions:

$$x_1 = y \qquad x_2 = y' = \frac{d}{dt}x_1 = x_1' \qquad x_3 = y'' = \frac{d}{dt}x_2 = x_2'$$

or, in general,

$$x_j(t) = \frac{d}{dt}x_{j-1}(t) \qquad \text{with } x_1 = y$$

where this form is similar to eqn. (3.7) but it takes advantage of the fact that there are no derivatives of u(t) within the defining system equation (i.e., b_0 , b_1, ··· b_{n-1} are all zero).

Now, from the defining equation with $y''' = \frac{d}{dt}x_3 = x_3'$, we have

$$\frac{d}{dt}x_3 = -a_1x_3 - a_2x_2 - a_3x_1 + u$$

Therefore, the three equations for x_1', x_2', and x_3' can be put into matrix form, giving

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_3 & -a_2 & -a_1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}u$$

Also, since the solution of interest is y(t) (a single output), the general expression for the output equation reduces to

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix}u$$

Note that this case is very similar to that given in Example 3.2, except now, with only u(t) on the RHS of the defining equation, the input matrix within the state equation can always be written as $\underline{\underline{B}} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix}^T$.

In addressing the second part of the problem, recall from Section II that the characteristic equation associated with the homogeneous form of the given equation can be written as

$$\lambda^3 + a_1\lambda^2 + a_2\lambda + a_3 = 0$$

Now, with the state space form, let's look at the eigenvalues of the state matrix, or

$$\det\left(\underline{\underline{A}} - \lambda\underline{\underline{I}}\right) = \begin{vmatrix} -\lambda & 1 & 0 \\ 0 & -\lambda & 1 \\ -a_3 & -a_2 & -a_1-\lambda \end{vmatrix} = 0$$

Expanding along row 1 gives

$$\det\left(\underline{\underline{A}} - \lambda\underline{\underline{I}}\right) = -\lambda\left[\lambda^2 + a_1\lambda + a_2\right] - 1\left[a_3\right] = \lambda^3 + a_1\lambda^2 + a_2\lambda + a_3 = 0$$

and this is exactly the same result as above. Thus, the eigenvalues of the state matrix for a linear stationary system are identical to the roots of the characteristic equation of the original linear constant coefficient ODE. Thus, $\det\left(\underline{\underline{A}} - \lambda\underline{\underline{I}}\right) = 0$ is also referred to as the ***characteristic equation***. All this, however, is not really surprising, since we did expect to get the same dynamic behavior from both formulations!

Conversion of Mixed Algebraic and Differential Equations

Note that the general formulation as stated in eqn. (3.1) has only one derivative term in each equation. This is the ***standard form***. However, a linear system can have more than one derivative in each expression. For example, consider the following 2nd order system,

$$\frac{d}{dt}x_1 + \frac{d}{dt}x_2 + 4x_1 = u(t)$$

$$\frac{d}{dt}x_2 - 3x_1 - 4x_2 = 0$$

Putting these expressions into matrix form gives

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -4 & 0 \\ 3 & 4 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}u(t)$$

Thus, we see that a more general form for the state equations for linear stationary systems is

$$\underline{\underline{E}}\frac{d}{dt}\underline{x}(t) = \underline{\underline{A}}\,\underline{x}(t) + \underline{\underline{B}}\,\underline{u}(t) \tag{3.13}$$

Now to put this into standard form, we simply multiply by $\underline{\underline{E}}^{-1}$ (for $\underline{\underline{E}}$ not singular), giving

$$\frac{d}{dt}\underline{x} = \left(\underline{\underline{E}}^{-1}\underline{\underline{A}}\right)\underline{x} + \left(\underline{\underline{E}}^{-1}\underline{\underline{B}}\right)\underline{u} \tag{3.14}$$

where $\left(\underline{\underline{E}}^{-1}\underline{\underline{A}}\right)$ now becomes the system matrix.

Another common situation that occurs is to have a complex dynamic system that is described by a set of mixed algebraic and differential equations. Such a system is sometimes said to have ***embedded statics*** since the algebraic equations represent relationships that do not change with time. For example, if $dx_2/dt = 0$ in the above illustration, then $-3x_1 - 4x_2 = 0$, and we can solve for x_2 in terms of x_1 and reduce the system to a single ODE.

In general, systems with embedded statics can be reduced in order by the number of algebraic equations via a series of algebraic manipulations. A general procedure for doing this follows:

Step 1. Order the equations so that n_1 equations contain derivative terms and the next n_2 equations only have algebraic relationships. Note here that $n_1 + n_2$ is the total number of original equations.

Step 2. Write the original equations using partitioned matrices (with $\underline{g} = \underline{\underline{B}}\underline{u}$), giving

$$\begin{bmatrix} \underline{\underline{E}}_{11} & 0 \\ 0 & 0 \end{bmatrix} \frac{d}{dt} \begin{bmatrix} \underline{x}_d \\ \underline{x}_a \end{bmatrix} = \begin{bmatrix} \underline{\underline{C}}_{11} & \underline{\underline{C}}_{12} \\ \underline{\underline{C}}_{21} & \underline{\underline{C}}_{22} \end{bmatrix} \begin{bmatrix} \underline{x}_d \\ \underline{x}_a \end{bmatrix} + \begin{bmatrix} \underline{g}_d \\ \underline{g}_a \end{bmatrix}$$

where $\underline{x}_d =$ $n_1 \times 1$ vector containing elements that have derivative terms

$\underline{x}_a =$ $n_2 \times 1$vector containing elements that appear without derivatives

Step 3. Expand the matrix expression in Step 2, giving separate equations for $\underline{x}_d$ and $\underline{x}_a$,

$$\underline{\underline{E}}_{11} \frac{d}{dt} \underline{x}_d = \underline{\underline{C}}_{11} \underline{x}_d + \underline{\underline{C}}_{12} \underline{x}_a + \underline{g}_d$$

$$0 = \underline{\underline{C}}_{21} \underline{x}_d + \underline{\underline{C}}_{22} \underline{x}_a + \underline{g}_a$$

Step 4. Solve the algebraic expressions for $\underline{x}_a$ in terms of $\underline{x}_d$ and substitute into the differential equation,

$$\underline{x}_a = -\underline{\underline{C}}_{22}^{-1} \left(\underline{\underline{C}}_{21} \underline{x}_d + \underline{g}_a \right)$$

and $$\underline{\underline{E}}_{11} \frac{d}{dt} \underline{x}_d = \underline{\underline{C}}_{11} \underline{x}_d - \underline{\underline{C}}_{12} \underline{\underline{C}}_{22}^{-1} \underline{\underline{C}}_{21} \underline{x}_d - \underline{\underline{C}}_{12} \underline{\underline{C}}_{22}^{-1} \underline{g}_a + \underline{g}_d$$

Step 5. Finally, the new system written in standard form is

$$\frac{d}{dt} \underline{x}_d = \underline{\underline{A}} \underline{x}_d + \underline{w} \tag{3.15a}$$

where $\underline{\underline{A}} = \underline{\underline{E}}_{11}^{-1} \left(\underline{\underline{C}}_{11} - \underline{\underline{C}}_{12} \underline{\underline{C}}_{22}^{-1} \underline{\underline{C}}_{21} \right)$ and $\underline{w} = \underline{\underline{E}}_{11}^{-1} \left(\underline{g}_d - \underline{\underline{C}}_{12} \underline{\underline{C}}_{22}^{-1} \underline{g}_a \right)$ (3.15b)

As always, the best way to get a good understanding of how to use this procedure is to see a specific example problem. Example 3.4 gives a simple, but illustrative, demonstration for dealing with systems with embedded statics.

Example 3.4 Treating Mixed Algebraic and Differential Equations

Problem Statement:

Put the following system in standard state form:

$$\frac{d}{dt}x_1(t)+\frac{d}{dt}x_2(t)=-4x_1(t)+x_4(t)+x_5(t)+u(t)$$

$$\frac{d}{dt}x_2(t)=x_2(t)-5x_3(t)+3u(t)$$

$$0=x_3(t)+x_4(t)$$

$$0=x_1(t)-3x_2(t)+x_3(t)+7u(t)$$

$$0=x_1(t)-x_2(t)+x_5(t)$$

Problem Solution:

Let's write these equations in matrix form and partition them accordingly,

$$\begin{bmatrix} \begin{bmatrix}1&1\\0&1\end{bmatrix} & \begin{bmatrix}0&0&0\\0&0&0\end{bmatrix} \\ \begin{bmatrix}0&0\\0&0\\0&0\end{bmatrix} & \begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix}\end{bmatrix}\frac{d}{dt}\begin{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}\\ \begin{bmatrix}x_3\\x_4\\x_5\end{bmatrix}\end{bmatrix}=\begin{bmatrix}\begin{bmatrix}-4&0\\0&1\end{bmatrix} & \begin{bmatrix}0&1&1\\-5&0&0\end{bmatrix}\\ \begin{bmatrix}0&0\\1&-3\\1&-1\end{bmatrix} & \begin{bmatrix}1&1&0\\1&0&0\\0&0&1\end{bmatrix}\end{bmatrix}\begin{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}\\ \begin{bmatrix}x_3\\x_4\\x_5\end{bmatrix}\end{bmatrix}+\begin{bmatrix}\begin{bmatrix}1\\3\end{bmatrix}\\ \begin{bmatrix}0\\7\\0\end{bmatrix}\end{bmatrix}u$$

This is now in the form

$$\begin{bmatrix}\underline{\underline{E}}_{11} & 0\\ 0 & 0\end{bmatrix}\frac{d}{dt}\begin{bmatrix}\underline{x}_d\\ \underline{x}_a\end{bmatrix}=\begin{bmatrix}\underline{\underline{C}}_{11} & \underline{\underline{C}}_{12}\\ \underline{\underline{C}}_{21} & \underline{\underline{C}}_{22}\end{bmatrix}\begin{bmatrix}\underline{x}_d\\ \underline{x}_a\end{bmatrix}+\begin{bmatrix}\underline{b}_d\\ \underline{b}_a\end{bmatrix}u$$

which has solution

$$\frac{d}{dt}\underline{x}_d(t)=\underline{\underline{A}}\,\underline{x}_d(t)+\underline{w}u(t)$$

where $\underline{\underline{A}}=\underline{\underline{E}}_{11}^{-1}\left(\underline{\underline{C}}_{11}-\underline{\underline{C}}_{12}\underline{\underline{C}}_{22}^{-1}\underline{\underline{C}}_{21}\right)$

$$\underline{w}=\underline{\underline{E}}_{11}^{-1}\left(\underline{b}_d-\underline{\underline{C}}_{12}\underline{\underline{C}}_{22}^{-1}\underline{b}_a\right)$$

Explicit values for $\underline{\underline{A}}$ and $\underline{w}$ can be obtained by performing the matrix operations implied in the latter expressions.

Linearization of Nonlinear Systems

In general, all systems are nonlinear. However, almost all systems behave in a near linear manner over some range of operation. Unless the particular system is highly nonlinear (i.e. linear region is very small), linear analysis methods may be appropriate for studying the dynamic behavior of these systems about some reference operating point. The goal of this subsection is to develop a method for linearizing the general nonlinear state variable equations. The linearized expression for the system behavior can then be utilized with tools for the analysis of standard linear systems.

In practice, although not strictly nonlinear, the variable coefficient terms in the original equations are often treated in a manner similar to the nonlinear terms. This is done because most of the analytical techniques are applicable only for linear stationary systems.

Using the state space representation, a general nonlinear non-stationary system can be written as

$$\frac{d}{dt}\underline{x}(t) = \underline{\underline{A}}\,\underline{x}(t) + \underline{f}(\underline{x},\underline{u},t) + \underline{\underline{B}}\,\underline{u}(t) \tag{3.16}$$

where all the nonlinear and variable coefficient terms are included within $\underline{f}(\underline{x},\underline{u},t)$. Now let's express the dependent state variable vector and independent forcing function vector as deviations from some reference condition, $\underline{x}_r$ and $\underline{u}_r$, giving

$$\underline{x}(t) = \underline{x}_r(t) + \delta\underline{x}(t) \tag{3.17a}$$

$$\underline{u}(t) = \underline{u}_r(t) + \delta\underline{u}(t) \tag{3.17b}$$

Substituting these expressions into eqn. (3.16) gives

$$\frac{d}{dt}\underline{x}_r(t) + \frac{d}{dt}\delta\underline{x}(t) = \underline{\underline{A}}\,\underline{x}_r(t) + \underline{\underline{A}}\,\delta\underline{x}(t) + \underline{f}(\underline{x}_r + \delta\underline{x}, \underline{u}_r + \delta\underline{u}, t) + \underline{\underline{B}}\,\underline{u}_r(t) + \underline{\underline{B}}\,\delta\underline{u}(t) \tag{3.18}$$

An approximation for the nonlinear, non-stationary terms can be given in the form of a first-order Taylor series expansion about the reference point. For the ith component of the vector, the first-order approximation gives

$$f_i(\underline{x},\underline{u},t) \approx f_i(\underline{x}_r,\underline{u}_r,t) + \sum_{j=1}^{N} \left.\frac{\partial f_i}{\partial x_j}\right|_{\underline{x}_r,\underline{u}_r} \delta x_j + \sum_{k=1}^{M} \left.\frac{\partial f_i}{\partial u_k}\right|_{\underline{x}_r,\underline{u}_r} \delta u_k \tag{3.19}$$

The last two summations simply represent matrix multiplication operations. Thus, eqn. (3.19) written in matrix notation gives

$$\underline{f}(\underline{x},\underline{u},t) = \underline{f}(\underline{x}_r + \delta\underline{x}, \underline{u}_r + \delta\underline{u}, t) \approx \underline{f}(\underline{x}_r,\underline{u}_r,t) + \underline{\underline{J}}_x(\underline{x}_r,\underline{u}_r)\delta\underline{x}(t) + \underline{\underline{J}}_u(\underline{x}_r,\underline{u}_r)\delta\underline{u}(t) \tag{3.20}$$

where $\underline{\underline{J}}_x(\underline{x}_r,\underline{u}_r)$ and $\underline{\underline{J}}_u(\underline{x}_r,\underline{u}_r)$ are referred to as Jacobian matrices evaluated at the reference point. These matrices can be represented symbolically as

$$\underline{\underline{J}}_x = \left[\frac{\partial f_i}{\partial x_j}\right] \quad \text{and} \quad \underline{\underline{J}}_u = \left[\frac{\partial f_i}{\partial u_k}\right] \tag{3.21}$$

For example, $\underline{\underline{J}}_x(\underline{x}_r, \underline{u}_r)$ written out in detail is given as

$$\underline{\underline{J}}_x(\underline{x}_r, \underline{u}_r) = \left.\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ & \vdots & & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}\right|_{\underline{x}_r, \underline{u}_r}$$

Now, getting back to the main development, substitution of eqn. (3.20) into (3.18) gives a very general representation of the approximate ***linearized*** system,

$$\frac{d}{dt}\delta\underline{x}(t) = \left[\underline{\underline{A}} + \underline{\underline{J}}_x(\underline{x}_r, \underline{u}_r)\right]\delta\underline{x}(t) + \left[\underline{\underline{B}} + \underline{\underline{J}}_u(\underline{x}_r, \underline{u}_r)\right]\delta\underline{u}(t) + \underline{g}(t) \tag{3.22}$$

where the additional forcing function, $\underline{g}(t)$, is given by

$$\underline{g}(t) = \underline{\underline{A}}\underline{x}_r + \underline{f}(\underline{x}_r, \underline{u}_r, t) + \underline{\underline{B}}\underline{u}_r - \frac{d}{dt}\underline{x}_r \tag{3.23}$$

This system is valid in some region around the reference point, $\underline{x}_r, \underline{u}_r$ -- but, if the system deviates significantly from the reference point, large errors can occur.

Equations (3.22) and (3.23) represent a linearization of the original nonlinear non-stationary system. The new system matrix for the linearized system becomes $\underline{\underline{A}} + \underline{\underline{J}}_x(\underline{x}_r, \underline{u}_r)$ and the new input matrix operator becomes $\underline{\underline{B}} + \underline{\underline{J}}_u(\underline{x}_r, \underline{u}_r)$. Note that these matrices are functions of the initial reference operating point, $\underline{x}_r, \underline{u}_r$. This is true since we have ***linearized*** about this condition.

One important aspect of the linearization process is the choice of the reference linearization point -- since the selection of this point is not always readily apparent. Unfortunately, there are no general rules that can be given as a guide here -- since every system is different and the best linearization point for a particular system may be highly problem dependent. However, there are two specific situations that are worthy of note, since they can lead to significant simplification of eqns. (3.22) and (3.23). In particular, if $\underline{x}_r$ and $\underline{u}_r$ are chosen so that they exactly satisfy the original nonlinear state equation, then the $\underline{g}(t)$ term given in eqn. (3.23) vanishes exactly. If, in addition to satisfying the state equation, the reference linearization point is also the initial operating point, $\underline{x}_r = \underline{x}_o$ and $\underline{u}_r = \underline{u}_o$, then, by definition, the initial condition for $\delta\underline{x}$ is simply the zero vector (i.e. the state vector written as a perturbation from reference is identically zero at time t = 0). If these two conditions apply, then eqn. (3.22) becomes

$$\frac{d}{dt}\delta\underline{x}(t) = \left[\underline{\underline{A}} + \underline{\underline{J}}_x(\underline{x}_r, \underline{u}_r)\right]\delta\underline{x}(t) + \left[\underline{\underline{B}} + \underline{\underline{J}}_u(\underline{x}_r, \underline{u}_r)\right]\delta\underline{u}(t) \quad \text{with} \quad \delta\underline{x}(0) = \begin{bmatrix}0 & 0 & \cdots\end{bmatrix}^T \tag{3.22a}$$

This latter expression is a special case, however, and it should only be used if appropriate for the system of interest.

We will have further occasion to work with the linearization of nonlinear systems. We will also see that the ***perturbation form*** given in eqn. (3.22a) is useful even if the original system is linear and stationary (since, by definition, the initial conditions for the new state vector, $\delta\underline{x}(t)$, are zero). For now, Example 3.5 gives a simple illustration of how to use the above methodology.

Example 3.5 Linearization of a 2nd Order Nonlinear System

Problem Statement:

Write a linear approximation for the single-input system described by

$$\underline{\underline{A}} = \begin{bmatrix} -1 & 2 \\ -1 & -3 \end{bmatrix} \qquad \underline{f} = \begin{bmatrix} x_2^2 \\ 0 \end{bmatrix} \qquad \underline{\underline{B}}\underline{u} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} u$$

Take the reference state and the initial condition as the point where the system is at equilibrium (i.e. $d\underline{x}_r/dt = 0$) with $\underline{x}_o = \underline{x}_r$ and $u_o = 0$.

Problem Solution:

Since we must expand the system around $\underline{x}_r, \underline{u}_r$, the first step requires that we identify the reference state. If the reference state is an equilibrium point, then one has

$$\frac{d}{dt}\underline{x}_r = \underline{0} = \underline{\underline{A}}\underline{x}_r + \underline{f}(\underline{x}_r, \underline{u}_r) + \underline{0}$$

or $\quad 0 = -x_{1r} + 2x_{2r} + x_{2r}^2 \quad$ and $\quad 0 = -x_{1r} - 3x_{2r}$

Rearranging these expressions gives

$$x_{1r} = -3x_{2r}$$

and

$$3x_{2r} + 2x_{2r} + x_{2r}^2 = 0$$

or $\quad (5 + x_{2r})x_{2r} = 0$

Therefore, the two states,

$$\underline{x}_{1r} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \text{and} \qquad \underline{x}_{2r} = \begin{bmatrix} 15 \\ -5 \end{bmatrix}$$

both satisfy the condition that $d\underline{x}_r/dt = 0$. Thus, there are two equilibrium states associated with this second-order nonlinear system.

Now for this particular nonlinear system, the Jacobian matrix associated with the state vector can be expressed as (note that $\underline{\underline{J}}_u = 0$ for this case)

$$\underline{\underline{J}}_x(\underline{x}_r, \underline{u}_r) = \left.\begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} \end{bmatrix}\right|_{\underline{x}_r, \underline{u}_r} = \left.\begin{bmatrix} 0 & 2x_2 \\ 0 & 0 \end{bmatrix}\right|_{\underline{x}_r, \underline{u}_r}$$

Therefore, for the reference state with $\underline{x}_r^T = \begin{bmatrix} 0 & 0 \end{bmatrix}$, the linearized system is

$$\frac{d}{dt}\delta\underline{x} = \begin{bmatrix} -1 & 2 \\ -1 & -3 \end{bmatrix}\delta\underline{x} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}\delta u$$

and for $\underline{x}_r^T = \begin{bmatrix} 15 & -5 \end{bmatrix}$, the linearized system is given as

$$\frac{d}{dt}\delta\underline{x} = \begin{bmatrix} -1 & -8 \\ -1 & -3 \end{bmatrix}\delta\underline{x} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}\delta u$$

where $\underline{x}_o = \underline{x}_r$, giving $\delta\underline{x}_o = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$ for both cases. These linear stationary systems represent linear approximations to the original system (about different reference states).

Some Additional Comments about Ex. 3.5: Although we have not discussed solution techniques as yet (see later in this section), it is interesting to elaborate a little on the current example. We noted above that the linear model is expected to give a good approximation to the behavior of the nonlinear system near the reference linearization point. Thus, we can get an idea of the behavior of the actual nonlinear system by looking at the eigenvalues of the state matrices for the two linear cases described above, as follows:

For $\underline{x}_r^T = \begin{bmatrix} 0 & 0 \end{bmatrix}$, we have

$$\begin{vmatrix} -1-\lambda & 2 \\ -1 & -3-\lambda \end{vmatrix} = (-1-\lambda)(-3-\lambda) + 2 = \lambda^2 + 4\lambda + 5 = 0$$

which leads to a pair of complex conjugate roots, $\lambda = -2 \pm j$, where we note that, since the real part is negative, the natural response has decaying sinusoidal behavior. Thus, this equilibrium point represents a region of stability in which small deviations from equilibrium tend to move back towards the equilibrium state.

For $\underline{x}_r^T = \begin{bmatrix} 15 & -5 \end{bmatrix}$, we have

$$\begin{vmatrix} -1-\lambda & -8 \\ -1 & -3-\lambda \end{vmatrix} = (-1-\lambda)(-3-\lambda) - 8 = \lambda^2 + 4\lambda - 5 = 0$$

which leads to a set of real distinct roots, $\lambda_1 = -5$ and $\lambda_2 = 1$. In this case, however, we have one positive root, which leads to growing exponential behavior. Thus, this equilibrium point is

surrounded by a region of instability in which small deviations from equilibrium tend to move away from the equilibrium state.

To illustrate this behavior, a short Matlab code was written to simulate the impulse response (see comments below) of both the above linear and nonlinear systems. In particular, Fig. 3.1 shows the impulse response for the system starting at the $\underline{x}_r^T = [0 \quad 0]$ point, and Fig. 3.2 displays the behavior to an impulse input if the system starts at $\underline{x}_r^T = [15 \quad -5]$. Note here that the response around these two initial states is significantly different and, in both cases, the linear approximation does a reasonable job in the vicinity of the reference point. However, for the unstable system, the linearized model tends to deviate fairly rapidly from the response of the nonlinear system. This is quite common for unstable systems, but it is not usually a concern since an unbounded output results from both cases (the only question concerns the rate at which this occurs). Also note that, if this was a real system (where all real systems have bounded output -- or they simply destroy themselves), then practical operation may have a controlled input so that the system operates well within the range of validity of the linearized models. Thus, in a controlled system, the linear model for both cases may be quite adequate.

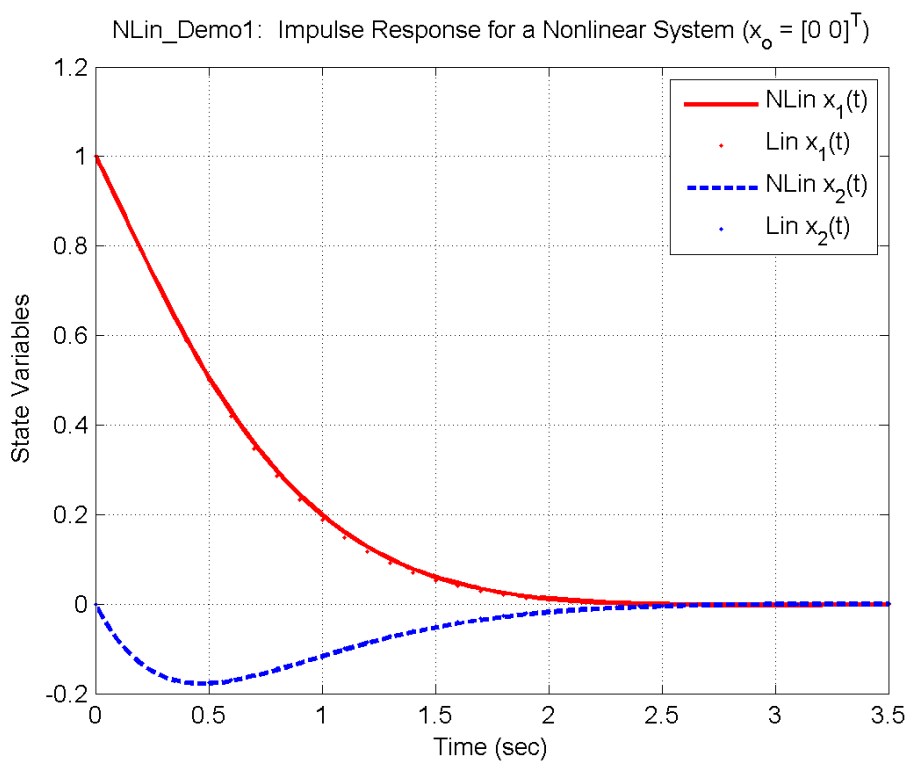


Fig. 3.1 Linear versus nonlinear solutions near the stable equilibrium point in Ex. 3.5.

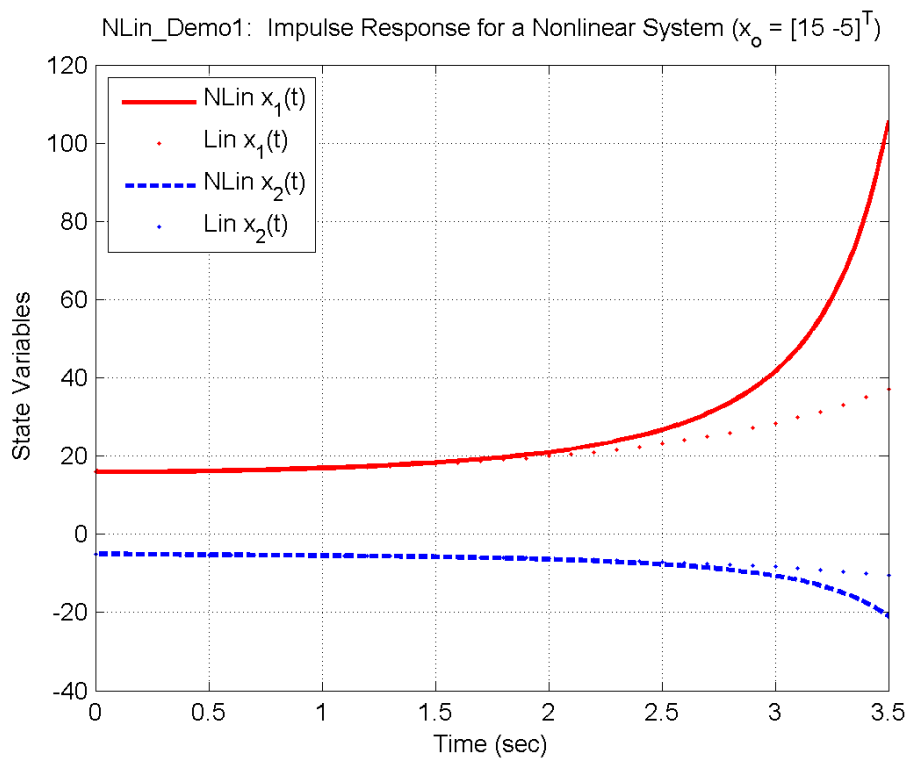


Fig. 3.2 Linear versus nonlinear solutions near the unstable equilibrium point in Ex. 3.5.

For completeness, the Matlab code that generated Figs. 3.1 and 3.2 is given in Table 3.1. However, we will not discuss any of the details of this code here. Another example problem (see Example 3.6) that focuses on different solution methods is discussed later in this section. Once this problem has been studied in some detail, it is recommended that the student refer back to the current example to obtain a better understanding of the Matlab code given in Table 3.1.

Table 3.1 Listings of Matlab m-file nlin_demo1.m.

```
%
%   NLIN_DEMO1.M  MATLAB demo that compares the linear and nonlinear
%                    solutions for a simple nonlinear system
%
%   A simple second-order nonlinear system is simulated using the ode45 routine to
%   numerically integrate the nonlinear state equations.  The nonlinear system has
%   two equilibrium points and these are treated as both the initial condition and
%   the reference point for a linear approximation to the nonlinear system.
%
%   The system has no physical interpretation, since the equations were simply "made
%   up" to present a simple example for illustration purposes.  It is interesting,
%   however, since one of the equilibrium points is in a region of stability and the
%   other one is an unstable equilibrium point (where deviations from equilibrium
%   continually move away from the equilibrium state). The linearized systems do
%   quite well in both cases near the linearization point, but the one that uses
%   the unstable equilibrium as the reference state tends to deviate significantly as
%   the system moves away from the reference state.
%
```

```
%   File prepared by J. R. White, UMass-Lowell (last update: Feb. 2020)
%

      clear all,   close all,   nfig = 0;
%
%   anonymous function for ode45
      ftx = @(t,x) [-x(1) + 2*x(2) + x(2)*x(2);
                    -x(1) - 3*x(2)              ];
%
%   let's focus on the xe = [0 0]^T equilibrium point first
%
%   numerical solution of nonlinear system (unit impulse input)
      timeo = 0;  timef = 3.5;  xo1 = [0 0]'; b = [1 0]';  xo1p = xo1 + b;
      options = odeset('RelTol',1.0e-6);
      [tn1,xn1] = ode45(ftx,[timeo,timef],xo1p,options);
%
%   linear solution (unit impulse input)
      Nt1 = 36;  tl1 = linspace(timeo,timef,Nt1);
      A1 = [-1 2;-1 -3];  B = [1 0]';  C = eye(2);  D = [0 0]';
      sys1 = ss(A1,B,C,D);  dx1 = impulse(sys1,tl1);
      xl1 = zeros(Nt1,2);
      for i = 1:2  % add reference point and deviation variable to get state vector
        xl1(:,i) = xo1(i) + dx1(:,i);
      end
%
%   compare linear and nonlinear solutions
      nfig = nfig+1;   figure(nfig)
      plot(tn1,xn1(:,1),'r-',tl1,xl1(:,1),'r.', ...
           tn1,xn1(:,2),'b--',tl1,xl1(:,2),'b.','LineWidth',2),grid
      title('NLin\_Demo1:  Impulse Response for a Nonlinear System (x_o = [0 0]^T)')
      ylabel('State Variables'), xlabel('Time (sec)')
      legend('NLin x_1(t)','Lin x_1(t)','NLin x_2(t)','Lin x_2(t)')
%
%   now let's focus on the xe = [15 -5]^T equilibrium point
%
%   numerical solution of nonlinear system (unit impulse input)
      timeo = 0;  timef = 3.5;  xo2 = [15 -5]'; b = [1 0]';  xo2p = xo2 + b;
      options = odeset('RelTol',1.0e-6);
      [tn2,xn2] = ode45(ftx,[timeo,timef],xo2p,options);
%
%   linear solution (unit impulse input)
      Nt2 = 36;  tl2 = linspace(timeo,timef,Nt2);
      A2 = [-1 -8;-1 -3];  B = [1 0]';  C = eye(2);  D = [0 0]';
      sys2 = ss(A2,B,C,D);  dx2 = impulse(sys2,tl2);
      xl2 = zeros(Nt2,2);
      for i = 1:2   % add reference point and deviation variable to get state vector
        xl2(:,i) = xo2(i) + dx2(:,i);
      end
%
%   compare linear and nonlinear solutions
      nfig = nfig+1;   figure(nfig)
      plot(tn2,xn2(:,1),'r-',tl2,xl2(:,1),'r.', ...
           tn2,xn2(:,2),'b--',tl2,xl2(:,2),'b.','LineWidth',2),grid
      title('NLin\_Demo1:  Impulse Response for a Nonlinear System (x_o = [15 -5]^T)')
      ylabel('State Variables'), xlabel('Time (sec)')
      legend('NLin x_1(t)','Lin x_1(t)','NLin x_2(t)', ...
             'Lin x_2(t)','Location','NorthWest')
%
%  end of demo
```

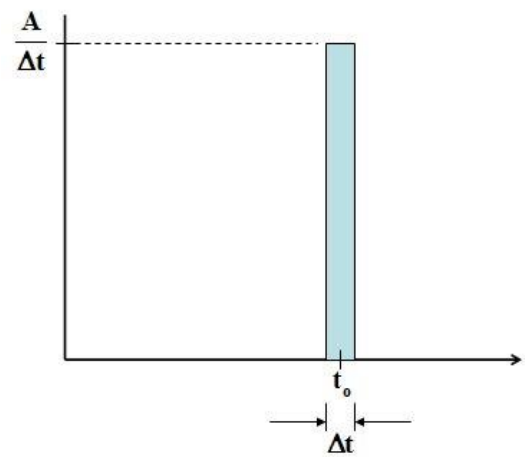
Comments on the Impulse Input (mentioned in Ex. 3.5)

Dynamic systems are often characterized by studying their response to certain types of inputs. One of the most common inputs is the **unit impulse**. This is actually an idealized function that can never be achieved exactly, but it is, nevertheless, an important test input that is used to characterize many systems via simulation. The output of a system due to a unit impulse input is referred to as the **impulse response** of that system.

We define an impulse with strength A via the sketch shown below. As the interval width associated with the rectangular pulse decreases to zero, the height increase to infinity. However, even though the height becomes infinite, the area within the rectangle remains constant at a value of A, where $A = (A/\Delta t)(\Delta t)$. If A has a value of unity, then the rectangle has unit area, and we refer to this as the unit impulse. This can be expressed using a formal mathematical notation as shown below.

An impulse of strength A is given by

$$A\delta(t - t_o) = \lim_{\Delta t \to 0} \frac{A}{\Delta t} \Delta t$$



and the integral over any interval that contains t_o is

$$\int_0^\infty A\delta(t - t_o)dt = A$$

where A = 1 for the unit impulse.

Now, the importance of this function can be illustrated nicely within the context of Ex. 3.5. If the input function is the unit impulse, then $u(t) = \delta(t)$, which implies that $t_o = 0$. We first write out the defining equations for the system,

$$\frac{d}{dt} x_1(t) = -x_1(t) + 2x_2(t) + x_2^2(t) + \delta(t)$$

$$\frac{d}{dt} x_2(t) = -x_1(t) - 3x_2(t)$$

and integrate them over the interval from $t = 0^-$ to $t = 0^+$, where 0^- and 0^+ represent very small times just before and after t = 0, respectively. It is during this infinitesimal interval, from $t = 0^-$ to $t = 0^+$, that the unit impulse is applied. Doing this gives

$$\int_{0^-}^{0^+} d\left(x_1(t)\right) = -\int_{0^-}^{0^+} x_1(t)dt + 2\int_{0^-}^{0^+} x_2(t)dt + \int_{0^-}^{0^+} x_2^2(t)dt + \int_{0^-}^{0^+} \delta(t)dt$$

$$\int_{0^-}^{0^+} d\left(x_2(t)\right) = -\int_{0^-}^{0^+} x_2(t)dt - 3\int_{0^-}^{0^+} x_2(t)dt$$

Since the integration interval is infinitesimally small, the only integrals that lead to a nonzero contribution are the ones on the left hand side (LHS) of the equations that contain the exact derivatives, and the single term on the RHS containing the impulse function. But, based on the

definition of the unit impulse from above, we know that the area under the unit impulse function is simply unity. Thus, formal integration of the above equations gives

$$x_1(0^+) - x_1(0^-) = 1 \qquad \text{or} \qquad x_1(0^+) = x_1(0^-) + 1$$

$$x_2(0^+) - x_2(0^-) = 0 \qquad \text{or} \qquad x_2(0^+) = x_2(0^-)$$

With the notation used here, the state evaluated at $t = 0^-$ represents the **real initial conditions** for the system. However, after application of the unit impulse input at t = 0, the **effective initial conditions**, given by the state evaluated at $t = 0^+$, are altered to include the effect of the impulse input. Thus, we see that **the impulse input manifests itself as an initial condition on the system**. This latter statement is extremely important, since it effectively tells us how to treat and interpret systems with impulse inputs -- **they are treated as unforced systems with modified initial conditions!**

The above manipulations for the case of an impulse input can also be generalized for the system written in general state space form. If we write the single-input state equations as

$$\frac{d}{dt}\underline{x} = \underline{\underline{A}}\,\underline{x} + \underline{f}(\underline{x}, t) + \underline{b}\,\delta(t) \qquad \text{with} \qquad \underline{x}(0^-) = \underline{x}_o$$

and integrate over the interval from $t = 0^-$ to $t = 0^+$, as before, we get

$$\int_{0^-}^{0^+} d\,\underline{x} = \int_{0^-}^{0^+} \underline{\underline{A}}\,\underline{x}\,dt + \int_{0^-}^{0^+} \underline{f}(\underline{x}, t)\,dt + \int_{0^-}^{0^+} \underline{b}\,\delta(t)\,dt$$

or

$$\underline{x}(0^+) = \underline{x}(0^-) + \underline{b}$$

Thus, **systems with impulse inputs are treated as unforced systems with the effective initial conditions given by $\underline{x}_o + \underline{b}$**. Careful study of the Matlab code in Table 3.1 shows that this is exactly what was done for the simulations in Ex. 3.5, and this is what we will do in all future problems involving impulse inputs.

Analytic Solution of Linear Stationary Systems

Now that we know how to put systems into state form, the next logical step is to address the time domain solution of the state equations. Linear stationary systems represent a special class of problems because they allow a formal analytical solution written in closed form. This subsection treats this important class of dynamic systems.

Our approach to developing analytic solutions is to emphasize the analogy between the scalar problem (single equation) and the matrix or the multivariable problem. The scalar case is treated to review the mathematics involved. Then, by analogy, the matrix case is solved using a generalization of the basic methodology. We will see that the matrix exponential function that was defined in Section II plays an important role in developing analytical solutions for linear stationary systems.

Homogeneous Linear Stationary Systems

To begin, let's concentrate on an LTI system which has no independent forcing function [i.e. u(t) = 0] and only a single state variable. This results in a scalar ***homogeneous system***.

Scalar Case

$$\frac{d}{dt}x(t) = ax(t) \qquad \text{and} \qquad x(0) = x_0 \tag{3.24}$$

Assume a solution of the form

$$x(t) = be^{\lambda t} \tag{3.25}$$

Substitution into the original equation gives

$$b\lambda e^{\lambda t} = abe^{\lambda t} \qquad \text{or} \qquad \lambda = a$$

Now evaluating $x(t) = be^{\lambda t}$ at t = 0 gives that $x(0) = b = x_0$. Therefore, the final solution is

$$x(t) = x_0 e^{at} \tag{3.26}$$

This same approach can also be taken for a general homogeneous LTI system written in state form.

Matrix Case

$$\frac{d}{dt}\underline{x}(t) = \underline{\underline{A}}\,\underline{x}(t) \qquad \text{and} \qquad \underline{x}(0) = \underline{x}_0 \tag{3.27}$$

By analogy to the scalar case, assume a solution of the form

$$\underline{x}(t) = e^{\underline{\underline{\lambda}}t}\,\underline{b} \tag{3.28}$$

Substitution into the original equation gives

$$\underline{\underline{\lambda}}e^{\underline{\underline{\lambda}}t}\,\underline{b} = \underline{\underline{A}}e^{\underline{\underline{\lambda}}t}\,\underline{b} \qquad \text{or} \qquad \underline{\underline{\lambda}} = \underline{\underline{A}}$$

Now, using the initial conditions gives, $\underline{x}(0) = \underline{b} = \underline{x}_0$. Therefore, the final solution is

$$\underline{x}(t) = e^{\underline{\underline{A}}t}\,\underline{x}_0 \tag{3.29}$$

This latter result can also be cast into several other useful forms. For example, if we evaluate the initial condition at some arbitrary $t = t_0$ instead of t = 0, eqn. (3.29) becomes

$$\underline{x}(t) = e^{\underline{\underline{A}}(t-t_0)}\,\underline{x}(t_0) \tag{3.30}$$

Or writing $\Delta t = t - t_0$, we have

$$\underline{x}(t_0 + \Delta t) = e^{\underline{\underline{A}}\Delta t}\,\underline{x}(t_0)$$

and, substituting t for t_0, we see that

$$\underline{x}(t + \Delta t) = e^{\underline{\underline{A}}\Delta t}\,\underline{x}(t) \tag{3.31}$$

is also a valid form of the solution. In fact, this latter form is particularly useful since $e^{\underline{\underline{A}}\Delta t}$ is simply a constant matrix for Δt = constant. Letting $\underline{\underline{G}} = e^{\underline{\underline{A}}\Delta t}$, eqn. (3.31) can be written as

$$\underline{x}(t+\Delta t) = \underline{\underline{G}}\,\underline{x}(t) \tag{3.32}$$

Thus the evaluation of $\underline{x}(t+\Delta t)$ for any time requires only repeated matrix-vector multiplication. This recursive formulation can also be written in discrete form as $\underline{x}_{k+1} = \underline{\underline{G}}\,\underline{x}_k$.

In the technical literature, the matrix exponential, $e^{\underline{\underline{A}}t}$, is sometimes referred to as the ***state transition matrix***. To see how this term applies, consider the following development.

In general, the solution to the homogeneous state equation, $\frac{d}{dt}\underline{x}(t) = \underline{\underline{A}}\,\underline{x}(t)$, can be written as

$$\underline{x}(t) = \underline{\underline{\Phi}}(t)\underline{x}(0) \tag{3.33}$$

where $\underline{\underline{\Phi}}(t)$ is the state transition matrix which is the unique solution of

$$\frac{d}{dt}\underline{\underline{\Phi}}(t) = \underline{\underline{A}}\,\underline{\underline{\Phi}}(t) \qquad \text{with} \qquad \underline{\underline{\Phi}}(0) = \underline{\underline{I}} \tag{3.34}$$

To show that the solution to eqn. (3.34) is indeed a solution to the original system of equations, consider the following:

$$\underline{x}(0) = \underline{\underline{\Phi}}(0)\underline{x}(0) = \underline{\underline{I}}\,\underline{x}(0) = \underline{x}(0)$$

and $$\frac{d}{dt}\underline{x}(t) = \frac{d}{dt}\left(\underline{\underline{\Phi}}(t)\underline{x}(0)\right) = \frac{d}{dt}\left(\underline{\underline{\Phi}}(t)\right)\underline{x}(0) = \underline{\underline{A}}\,\underline{\underline{\Phi}}(t)\underline{x}(0) = \underline{\underline{A}}\,\underline{x}(t)$$

For a homogeneous linear time-invariant system we know (from above) that $\underline{x}(t) = e^{\underline{\underline{A}}t}\underline{x}_0$, and we have just argued that $\underline{x}(t) = \underline{\underline{\Phi}}(t)\underline{x}(0)$. Therefore, the state transition matrix is simply $e^{\underline{\underline{A}}t}$. Thus, we see from eqn. (3.33) that the state transition matrix simply specifies a ***transformation of the initial conditions*** to give the state at any time t.

We already know several properties of the state transition matrix, $\underline{\underline{\Phi}} = e^{\underline{\underline{A}}t}$. For example,

1. $\frac{d}{dt}e^{\underline{\underline{A}}t} = \underline{\underline{A}}e^{\underline{\underline{A}}t} = e^{\underline{\underline{A}}t}\underline{\underline{A}}$
2. $\int e^{\underline{\underline{A}}t}dt = \underline{\underline{A}}^{-1}e^{\underline{\underline{A}}t} = e^{\underline{\underline{A}}t}\underline{\underline{A}}^{-1}$
3. $e^{\underline{\underline{A}}(t+\tau)} = e^{\underline{\underline{A}}t}e^{\underline{\underline{A}}\tau}$
4. Letting τ = -t, gives $e^{\underline{\underline{A}}(t+\tau)} = e^{\underline{\underline{A}}t}e^{-\underline{\underline{A}}t} = \underline{\underline{I}}$ or $\left[e^{\underline{\underline{A}}t}\right]^{-1} = e^{-\underline{\underline{A}}t}$
5. $e^{(\underline{\underline{A}}+\underline{\underline{B}})t} = e^{\underline{\underline{A}}t}e^{\underline{\underline{B}}t}$ if $\underline{\underline{A}}\,\underline{\underline{B}} = \underline{\underline{B}}\,\underline{\underline{A}}$

Note also that these properties can be written in terms of $\underline{\underline{\Phi}}(t)$. For example, properties #3 and #4 can be written as

$$\underline{\underline{\Phi}}(t+\tau) = e^{\underline{\underline{A}}(t+\tau)} = \underline{\underline{\Phi}}(t)\underline{\underline{\Phi}}(\tau)$$

and

$$\underline{\underline{\Phi}}^{-1}(t) = \left[e^{\underline{\underline{A}}t}\right]^{-1} = e^{-\underline{\underline{A}}t} = \underline{\underline{\Phi}}(-t)$$

Thus, the student should be aware that the terminology and manipulation of the so-called state transition matrix, $\underline{\underline{\Phi}}(t)$, is nothing more than working with the familiar matrix exponential. Although the notation associated with the state transition matrix is often used in the literature, we will simply use the term ***matrix exponential*** in the bulk of these notes.

Non-Homogeneous Linear Stationary Systems

Returning to the solution of linear stationary systems, we now address the solution of non-homogeneous systems.

Scalar Case

$$\frac{d}{dt}x(t) - ax(t) = bu(t) \tag{3.35}$$

Multiplying by the integrating factor, $e^{-\int a dt} = e^{-at}$, gives

$$e^{-at}\left(\frac{d}{dt}x(t) - ax(t)\right) = \frac{d}{dt}\left(e^{-at}x(t)\right) = e^{-at}bu(t)$$

Now, integrating this expression between t_0 and t gives

$$\int_{t_0}^{t} d\left(e^{-a\tau}x(\tau)\right) = \int_{t_0}^{t} e^{-a\tau}bu(\tau)d\tau$$

$$e^{-at}x(t) - e^{-at_0}x(t_0) = \int_{t_0}^{t} e^{-a\tau}bu(\tau)d\tau$$

Finally, multiplying through by e^{at} and rearranging gives

$$x(t) = e^{a(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{a(t-\tau)}bu(\tau)d\tau \tag{3.36}$$

If $t_0 = 0$, this becomes

$$x(t) = e^{at}x_0 + \int_{0}^{t} e^{a(t-\tau)}bu(\tau)d\tau \tag{3.37}$$

The standard procedure for checking the correctness of a solution [such as eqn. (3.37)] is to substitute it into the original differential equation. This case is no different, but one must be careful when differentiating integral terms that have variable limits of integration. The usual technique for doing this is called ***Leibnitz's Rule***, which can be stated as follows:

If $F(t)=\int_{a(t)}^{b(t)}\phi(x,t)dx$, then with suitable conditions on a(t), b(t), and $\phi(x,t)$ [i.e. that they are well behaved with no discontinuities], we have

$$\frac{d}{dt}F(t)=\int_{a(t)}^{b(t)}\frac{\partial}{\partial t}\phi(x,t)dx+\phi\left[b(t),t\right]\frac{d}{dt}b(t)-\phi\left[a(t),t\right]\frac{d}{dt}a(t) \tag{3.38}$$

Now, substituting eqn. (3.37) into eqn. (3.35) and using Leibnitz's Rule, we have

$$\frac{d}{dt}x(t)=ae^{at}x_0+a\int_0^t e^{a(t-\tau)}bu(\tau)d\tau+e^{a(t-t)}bu(t)=ax(t)+bu(t)$$

which shows that eqn. (3.37) is indeed the proper solution to the ODE given in eqn. (3.35).

Matrix Case

$$\frac{d}{dt}\underline{x}(t)-\underline{\underline{A}}\,\underline{x}(t)=\underline{\underline{B}}\,\underline{u}(t) \tag{3.39}$$

By analogy to the scalar case, let's multiply this expression by the integrating factor, $e^{-\underline{\underline{A}}t}$, giving

$$e^{-\underline{\underline{A}}t}\left[\frac{d}{dt}\underline{x}(t)-\underline{\underline{A}}\,\underline{x}(t)\right]=\frac{d}{dt}\left[e^{-\underline{\underline{A}}t}\underline{x}(t)\right]=e^{-\underline{\underline{A}}t}\underline{\underline{B}}\,\underline{u}(t)$$

Integrating this between t_0 and t, one has

$$\int_{t_0}^{t}d\left[e^{-\underline{\underline{A}}\tau}\underline{x}(\tau)\right]=\int_{t_0}^{t}e^{-\underline{\underline{A}}\tau}\underline{\underline{B}}\,\underline{u}(\tau)d\tau$$

$$e^{-\underline{\underline{A}}t}\underline{x}(t)-e^{-\underline{\underline{A}}t_0}\underline{x}(t_0)=\int_{t_0}^{t}e^{-\underline{\underline{A}}\tau}\underline{\underline{B}}\,\underline{u}(\tau)d\tau$$

Finally, multiplying through by $e^{\underline{\underline{A}}t}$ and rearranging gives

$$\underline{x}(t)=e^{\underline{\underline{A}}(t-t_0)}\underline{x}(t_0)+\int_{t_0}^{t}e^{\underline{\underline{A}}(t-\tau)}\underline{\underline{B}}\,\underline{u}(\tau)d\tau \tag{3.40}$$

If $t_0 = 0$, this becomes

$$\underline{x}(t)=e^{\underline{\underline{A}}t}\underline{x}_0+\int_0^t e^{\underline{\underline{A}}(t-\tau)}\underline{\underline{B}}\,\underline{u}(\tau)d\tau \tag{3.41}$$

As a check on this solution we can differentiate eqn. (3.41), giving

$$\frac{d}{dt}\underline{x}(t)=\underline{\underline{A}}e^{\underline{\underline{A}}t}\underline{x}_0+\underline{\underline{A}}\int_0^t e^{\underline{\underline{A}}(t-\tau)}\underline{\underline{B}}\,\underline{u}(\tau)d\tau+\underline{\underline{B}}u(t)=\underline{\underline{A}}\,\underline{x}(t)+\underline{\underline{B}}\,\underline{u}(t)$$

As a final note, one should be aware that these expressions can also be written in terms of the state transition matrix, where

$$\underline{\underline{\Phi}}(t)=e^{\underline{\underline{A}}t} \qquad \text{and} \qquad \underline{\underline{\Phi}}(t-\tau)=e^{\underline{\underline{A}}(t-\tau)}$$

For example, eqn. (3.41) written with $\underline{\underline{\Phi}}(t)$ becomes

$$\underline{x}(t)=\underline{\underline{\Phi}}(t)\underline{x}_0+\int_0^t\underline{\underline{\Phi}}(t-\tau)\underline{\underline{B}}\,\underline{u}(\tau)d\tau \tag{3.42}$$

Linear Time-Varying Systems

To complete our discussion of the analytic solution of the linear state equations, we really should look briefly at the case where the system matrix is a function of the independent time variable. However, after a relatively tedious development, the bottom line is that, in the general case, ***a closed form solution in terms of the matrix exponential is not possible for time varying systems***. This is because, for the general case, $\underline{\underline{A}}(t)$ and $\int_{t_0}^{t} \underline{\underline{A}}(\tau)d\tau$ do not commute. Thus, such a development would only serve as an illustration of why the matrix exponential solution technique cannot be used for general time-varying systems. Therefore, we will bypass this proof and refer the interested student to the literature for further discussion of analytic solutions to time varying systems (see the textbook ***Modern Control Engineering*** by Ogata, for example).

One summary note is in order before leaving our discussion of analytic solutions in the time domain. We have seen that linear stationary systems have analytical closed form solutions and that linear non-stationary systems (in general) do not. However, it should be clear that solutions to realistic problems (with several unknowns or state variables) must be determined via computer implementation, whether we are dealing with stationary or non-stationary systems. The analytic solutions for linear stationary systems are extremely important, but they can be generated by hand for only very low-order systems. Thus, no matter what form the system takes, computer implementation for realistic engineering problems will always be required. An illustration of this analytical solution scheme for an LTI system is given in Example 3.6 at the end of this section.

Discretization of Linear Stationary Systems

As just discussed, the computer implementation of any potential solution scheme is usually a prerequisite for the analysis and simulation of realistic systems. In general there are two good approaches used in computer simulations of high-order dynamic systems. These include the discretization of the analytical solution scheme for linear stationary systems and general numerical integration techniques that can be adapted to any system of interest (i.e. non-stationary and non-linear). In this subsection we treat the analytical discretization process and defer the discussion of numerical integration techniques for the next subsection.

The basic goal of the discretization process is to convert the standard continuous linear stationary state equations to discrete form. That is, given

$$\frac{d}{dt}\underline{x}(t) = \underline{\underline{A}}\,\underline{x}(t) + \underline{\underline{B}}\,\underline{u}(t) \tag{3.43}$$

convert the continuous system into a difference equation of the form

$$\underline{x}\left[(k+1)T\right] = \underline{\underline{G}}(T)\underline{x}(kT) + \underline{\underline{H}}(T)\underline{u}(kT) \tag{3.44}$$

where T is the sampling period and $\underline{\underline{G}}$ and $\underline{\underline{H}}$ are constant matrices that may be a function of the choice of T. For convenience, eqn. (3.44) is often written simply as

$$\underline{x}_{k+1} = \underline{\underline{G}}\,\underline{x}_k + \underline{\underline{H}}\,\underline{u}_k \tag{3.45}$$

These expressions [both eqns. (3.44) and (3.45)] represent a discrete recurrence relationship that can be evaluated quite easily on the computer if the $\underline{\underline{G}}$ and $\underline{\underline{H}}$ matrices are known. Our goal here is to derive explicit expressions for these matrices.

In order to accomplish this transformation, we must make an assumption concerning the behavior of $\underline{u}(t)$. In fact, the only assumption in the development below is that $\underline{u}(t)$ can be written as a ***piecewise constant function*** of time. Therefore, $\underline{u}(t) \to \underline{u}_k$, where it is implied that the forcing function is constant over the kth interval. This approximation is often referred to as a ***zero-order hold***. Note that a similar development, where $\underline{u}(t)$ varies linearly over the interval, is referred as a ***first-order hold***. For general applications, an assumption of a constant or linear variation of the input over a single interval is not really very restrictive, since the sampling time T can be chosen to be sufficiently small so that these representations of $\underline{u}(t)$ are indeed good approximations. If $\underline{u}(t)$ is a slowly varying function, then the choice of sampling time is made so that the series form of the matrix exponential converges rapidly.

To derive the discrete form for the zero-order hold approximation, recall that the solution to the continuous linear stationary problem is

$$\underline{x}(t) = e^{\underline{\underline{A}}(t-t_0)}\underline{x}(t_0) + e^{\underline{\underline{A}}t}\int_{t_0}^{t} e^{-\underline{\underline{A}}\tau}\underline{\underline{B}}\underline{u}(\tau)d\tau$$

We are interested in evaluating this expression over a single sampling interval, T. Thus, letting $t = (k+1)T$ and $t_0 = kT$, one has

$$\underline{x}_{k+1} = e^{\underline{\underline{A}}T}\underline{x}_k + \int_{kT}^{(k+1)T} e^{\underline{\underline{A}}[(k+1)T-\tau]}\underline{\underline{B}}\underline{u}(\tau)d\tau$$

Now if we introduce a new variable η such that $\tau = \eta + kT$ or $\eta = \tau - kT$ with

$$d\eta = d\tau, \qquad (k+1)T - \eta - kT = T - \eta, \qquad \text{and} \qquad \underline{u}(\eta + kT) = \underline{u}(kT) = \underline{u}_k$$

one has

$$\underline{x}_{k+1} = e^{\underline{\underline{A}}T}\underline{x}_k + \left[\int_0^T e^{\underline{\underline{A}}(T-\eta)}d\eta\right]\underline{\underline{B}}\underline{u}_k$$

where we have used the zero-order hold approximation that $\underline{u}(t) \approx \underline{u}_k$ over the interval.

Performing the integral, we have

$$\int_0^T e^{\underline{\underline{A}}(T-\eta)}d\eta = e^{\underline{\underline{A}}T}\int_0^T e^{-\underline{\underline{A}}\eta}d\eta = e^{\underline{\underline{A}}T}\left[-\underline{\underline{A}}^{-1}e^{-\underline{\underline{A}}\eta}\right]_0^T$$

$$= -\underline{\underline{A}}^{-1}e^{\underline{\underline{A}}T}\left[e^{-\underline{\underline{A}}T} - \underline{\underline{I}}\right] = \underline{\underline{A}}^{-1}\left[e^{\underline{\underline{A}}T} - \underline{\underline{I}}\right]$$

and the final result is

$$\underline{x}_{k+1} = e^{\underline{\underline{A}}T}\underline{x}_k + \underline{\underline{A}}^{-1}\left[e^{\underline{\underline{A}}T} - \underline{\underline{I}}\right]\underline{\underline{B}}\underline{u}_k \tag{3.46}$$

where we see that this expression is in the final form that we desire. By comparison with eqn. (3.45), one can write explicit expressions for the $\underline{\underline{G}}$ and $\underline{\underline{H}}$ matrices as

$$\underline{\underline{G}} = e^{\underline{\underline{A}}T} \tag{3.47}$$

and $$\underline{\underline{H}} = \underline{\underline{A}}^{-1}\left[e^{\underline{\underline{A}}T} - \underline{\underline{I}}\right]\underline{\underline{B}} \tag{3.48}$$

Another form for $\underline{\underline{H}}$ may be more appropriate if one plans to use the infinite series expansion for generating a numerical result for the matrix exponential. Recall that

$$\underline{\underline{G}} = e^{\underline{\underline{A}}T} = \underline{\underline{I}} + \underline{\underline{A}}T + \frac{1}{2!}\left(\underline{\underline{A}}T\right)^2 + \frac{1}{3!}(\underline{\underline{A}}T)^3 + \quad \cdots$$

If $\underline{\underline{H}}$ is expanded in a similar manner, one has

$$\underline{\underline{H}} = \underline{\underline{A}}^{-1}\left[\underline{\underline{I}} + \underline{\underline{A}}T + \frac{1}{2!}\left(\underline{\underline{A}}T\right)^2 + \frac{1}{3!}(\underline{\underline{A}}T)^3 + \quad \cdots \quad - \underline{\underline{I}}\right]\underline{\underline{B}}$$

$$= T\left[\underline{\underline{I}} + \frac{1}{2!}\underline{\underline{A}}T + \frac{1}{3!}\left(\underline{\underline{A}}T\right)^2 + \quad \cdots\right]\underline{\underline{B}} \;=\; T\underline{\underline{Q}}\,\underline{\underline{B}} \tag{3.49}$$

where $$\underline{\underline{Q}} = \left[\underline{\underline{I}} + \frac{1}{2!}\underline{\underline{A}}T + \frac{1}{3!}\left(\underline{\underline{A}}T\right)^2 + \quad \cdots\right] \tag{3.50}$$

Note that $\underline{\underline{Q}}$ is similar to $\underline{\underline{G}}$ and they both can be generated within the same subroutine with little additional effort. Thus, $\underline{\underline{H}}$ can be found via eqn. (3.49) which avoids the computation of the inverse of $\underline{\underline{A}}$. If Sylvester's theorem is used to compute $\underline{\underline{G}}$, then $\underline{\underline{H}}$ can be found by either eqn. (3.48) or (3.49) depending on the choice of the programmer. The evaluation of eqn. (3.48) requires some software to determine $\underline{\underline{A}}^{-1}$ and the use of eqn. (3.49) requires the evaluation of a truncated series expansion involving the system matrix.

In either case, once the $\underline{\underline{G}}$ and $\underline{\underline{H}}$ matrices have been generated, the numerical simulation of the system can proceed using the recursion relation given in eqn. (3.45). Note that this method gives an **exact solution** (independent of the sampling period T) assuming that $\underline{\underline{G}}$ and $\underline{\underline{H}}$ can be given exactly (or to any desired degree of accuracy) and that $\underline{u}(t)$ is indeed constant within interval T. Thus, the discrete solution of eqn. (3.45) is the same as the continuous solution to eqn. (3.43) if the piecewise constant assumption for $\underline{u}(t)$ is exact. An example of this discrete technique is given in Example 3.6 at the end of this section.

Numerical Integration of the State Equations

An alternate solution technique to using the matrix exponential approach is to simply integrate the defining state equations. There are several numerical techniques for doing this (see any good numerical methods textbook), but the details of these schemes are outside the scope of this course. Our goal will be to give a simple illustration that introduces the proper terminology and describes the basic concept behind all numerical integration methods. With these basics, the student should be able to use available "canned" software for the numerical integration of a set of realistic system equations. The student can gain further insight and understanding of the several methods that are available with further self-study (as desired).

The advantage of numerical integration methods over the discretization technique discussed in the previous subsection is that they can easily handle time-varying system matrices, $\underline{\underline{A}}(t)$. In addition, relatively complicated nonlinear problems can also be treated in a general way. These advantages, plus the efficiency of the many current state-of-the-art integration schemes that are available, make these methods preferable to the discrete matrix exponential approach for high-order linear stationary systems and for all nonlinear or non-stationary models.

As a brief introduction to numerical integration, we will look at the two simplest methods available; the Euler and Modified Euler methods. Although these methods present the basic concepts, they are quite inefficient relative to some of the more advanced techniques such as adaptive Runge-Kutta (RK) methods and the adaptive multistep methods that use the Adams-Bashforth-Moulton integration formulas. Again, since our goal here is simply an introduction to the basic concepts, the student is referred to other references for more information on the various numerical integration schemes available. Thus, one is cautioned that the methods presented here are intended only as an illustration of the basic methodology, and that these techniques are not normally implemented as derived (since there are better schemes available).

<u>Euler Method (Scalar Case)</u>

As a starting point, consider the general first-order differential equation

$$\frac{d}{dt}x(t) = \underbrace{a(t)x(t) + b(t)u(t)}_{\text{(linear case)}} = \underbrace{f(x,u,t)}_{\text{(nonlinear case)}} \tag{3.51}$$

In the Euler method, one assumes that the right hand side of eqn. (3.51) is constant over some small time interval $\Delta t = t_{k+1} - t_k$. With this condition, integrating eqn. (3.51) gives

$$\int_{t_k}^{t_{k+1}} \frac{d}{dt}x(t)dt = f(x_k, u_k, t_k)\int_{t_k}^{t_{k+1}} dt$$

or

$$x_{k+1} = x_k + \Delta t\left\{f(x_k, u_k, t_k)\right\} \tag{3.52}$$

Thus, the problem has been converted to a recurrence relation for x_{k+1} in terms of x_k and the function evaluated with the solution at time t_k. This is a relatively crude approximation but, if Δt is chosen sufficiently small, this simple scheme can give reasonable results (and it is certainly easy to conceptualize).

Modified Euler Method (Scalar Case)

A somewhat better estimate for integrating the defining equation over some specified Δt is to assume that f(x,u,t) varies linearly over the given step. This is equivalent to the so-called trapezoidal rule. Doing this gives the following recurrence relation,

$$x_{k+1} = x_k + \frac{\Delta t}{2}\left\{f(x_k, u_k, t_k) + f(x_{k+1}, u_{k+1}, t_{k+1})\right\} \tag{3.53}$$

The problem with eqn. (3.53) is that we need x_{k+1} to calculate f_{k+1}, but f_{k+1} is needed to calculate x_{k+1}, etc. A solution to this circular dilemma is to use the Euler method to get a first guess for x_{k+1} and f_{k+1} and then use this guess to get a better estimate of x_{k+1}. This crude ***Predictor-Corrector scheme*** can be summarized as follows;

Predictor Step $\quad x'_{k+1} = x_k + \Delta t f_k \qquad (3.54)$

Corrector Step $\quad x_{k+1} = x_k + \frac{\Delta t}{2}(f_k + f'_{k+1})$

where $\quad f_k = f(x_k, u_k, t_k) \quad$ and $\quad f'_{k+1} = f(x'_{k+1}, u_{k+1}, t_{k+1})$

with x'_{k+1} representing a first estimate (prediction) of the value for the dependent variable at t_{k+1} and x_{k+1} being the best estimate (correction) for this time point.

The modified Euler method is a simple, but illustrative example of a predictor-corrector numerical integration scheme. Other refinements such as adaptive control/updating of the integration step, Δt, can be done by addressing the error in the predictor and corrector steps. In fact, in most current algorithms, one only inputs a desired tolerance level, and a variable integration step is computed that will satisfy the specified error criterion.

The above numerical integration schemes for the scalar problem can be easily generalized for solution of large systems of equations. In this case we desire the solution of a general system of first order equations,

$$\frac{d}{dt}\underline{x}(t) = \underline{\underline{A}}(t)\underline{x}(t) + \underline{\underline{B}}(t)\underline{u}(t) = \underset{\text{(nonlinear case)}}{\underline{f}(\underline{x}, \underline{u}, \underline{t})} \tag{3.55}$$

$$\text{(linear case)}$$

By analogy to the scalar problem, one has the following expressions:

Euler Method (Matrix Case)

$$\underline{x}_{k+1} = \underline{x}_k + \Delta t \underline{f}_k \qquad \text{where} \qquad \underline{f}_k = \underline{f}(\underline{x}_k, \underline{u}_k, \underline{t}_k) \tag{3.56}$$

Modified Euler Method (Matrix Case)

Predictor Step $\quad \underline{x}'_{k+1} = \underline{x}_k + \Delta t \underline{f}_k \qquad (3.57)$

Corrector Step $\quad \underline{x}_{k+1} = \underline{x}_k + \frac{\Delta t}{2}(\underline{f}_k + \underline{f}'_{k+1})$

where $\quad \underline{f}_k = \underline{f}(\underline{x}_k, \underline{u}_k, \underline{t}_k) \quad$ and $\quad \underline{f}'_{k+1} = \underline{f}(\underline{x}'_{k+1}, \underline{u}_{k+1}, \underline{t}_{k+1})$

Thus we see that numerical integration is a relatively simple task to perform on the computer (for both scalar and matrix first-order ordinary differential equations). State-of-the-art methods are more complicated than the methods described here, but they are similar in basic structure.

The most important part of the above simple schemes is the choice of an appropriate time interval, Δt. For the solution of several state equations, the choice of Δt becomes even more important and difficult, since some state variables may vary rapidly while others are slowly varying functions of time. The more advanced schemes usually address this concern by automatically adjusting Δt as needed to meet user specified accuracy requirements (this is why they are called advanced). Thus, ***Adaptive Predictor-Corrector*** methods are the norm for performing realistic time domain simulations.

Example 3.6 illustrates how to use the numerical integration scheme specific to Matlab for the simulation of a simple 2nd order system. This example also shows that the numerical scheme gives exactly the same results as the analytical solution schemes (both continuous and discrete).

Well, we have finally come to the end of this section of notes. Our main goals here were to develop a notation for the standard state space representation of dynamic systems, to address ways to convert general systems to state form, and to look at both analytical and numerical solution schemes for solving the standard state equations. At this point, we have (hopefully) completed these goals and the student should have access to a variety of tools for the time domain solution of general dynamic systems. There are certainly more subjects to be treated in later sections. However, the ability to simulate the time domain behavior of systems for arbitrary input forcing functions, $\underline{u}(t)$, is a major first step in understanding dynamic systems. Hopefully, we have successfully taken this first step.

Example 3.6 An Example Illustrating Various Solution Schemes

The most general representation for any 2nd order linear stationary system is given as

$$\frac{d^2y}{dt^2} + a_1\frac{dy}{dt} + a_2 y = b_0\frac{d^2u}{dt^2} + b_1\frac{du}{dt} + b_2 u$$

Using the general recipe for converting nth order systems into state form, one has

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -a_2 & -a_1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \beta_0 u$$

where $\beta_0 = b_0$, $\beta_1 = b_1 - a_1 b_0$, and $\beta_2 = b_2 - a_1 b_1 + a_1^2 b_0 - a_2 b_0$

Also, $x_1 = y - \beta_0 u$ and $x_2 = \frac{d}{dt}x_1 - \beta_1 u$

where $x_1(0)$ and $x_2(0)$ represent the initial conditions for the system.

As a specific example, consider the simple series RLC circuit shown in Fig. 3.3.

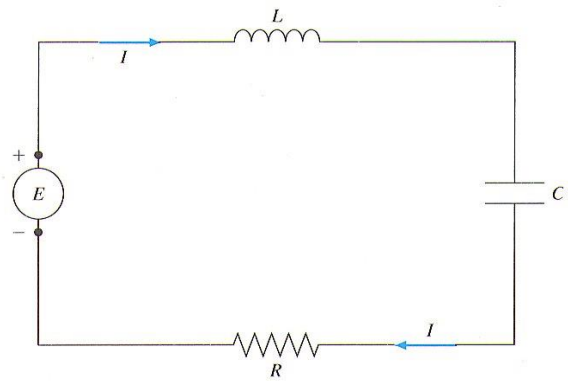


Fig. 3.3 RLC circuit for Example 3.6.

From Kirchhoff's voltage law (KVL), the sum of the voltage drops around a closed loop must be zero, or

$$E = E_L + E_C + E_R$$

where E(t) is the applied voltage and the voltage drops across the individual components are given by

$$E_L = L\frac{dI}{dt} \qquad E_C = \frac{1}{C}\int I dt \qquad E_R = RI$$

where I(t) is the current in the loop. Substituting these relationships into the balance equation gives

$$E(t) = L\frac{d}{dt}I(t) + \frac{1}{C}\int_0^t I(\tau)d\tau + RI(t)$$

This combined differential and integral balance equation can be put into standard differential form by differentiating each term, or

$$L\frac{d^2}{dt^2}I(t) + R\frac{d}{dt}I(t) + \frac{1}{C}I(t) = \frac{d}{dt}E(t)$$

Comparing this system to the general representation for a 2nd order LTI system, we have

$$a_1 = \frac{R}{L} \qquad a_2 = \frac{1}{LC} \qquad b_0 = 0 \qquad b_1 = \frac{1}{L} \qquad b_2 = 0$$

Thus, the standard state space representation for this specific system is

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{R}{L} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \frac{1}{L} \\ -\frac{R}{L^2} \end{bmatrix}u \qquad \text{and} \qquad y = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0]u$$

where $x_1 = I$ and $x_2 = \frac{d}{dt}x_1 - \frac{1}{L}u$

and u(t) is the input voltage, E(t), and y(t) is the resultant current, I(t), versus time.

For specificity in the numerical solutions, let's assume that the initial conditions are all zero with no applied voltage. Also, for the reference case, let R = 100 ohms, L = 0.1 henry, and C = 0.001 farad. Finally, for simplicity, let's look at the case of a step input in E(t) of 10 volts. Thus, for this case, u(t) = 10 volts for all t > 0. The sensitivity of the response, I(t), to changes in these parameters can also be addressed.

Concerning the initial conditions of interest here, we note that the step change in E(t) at $t = 0^+$ manifests itself as an implied initial condition (IC) on the derivative of the current. To see this, one can evaluate the basic KVL equation at $t = 0^+$, or

$$E(0^+) = L\left.\frac{dI}{dt}\right|_{0^+} + \frac{1}{C}\int_{0^-}^{0^+} I(\tau)d\tau + RI(0^+)$$

But, since the current can't change instantaneously, the last two terms are zero, giving

$$\left.\frac{dI}{dt}\right|_{0^+} = \frac{1}{L}E(0^+)$$

Thus, the formal ICs for this simple RLC circuit with a step change in voltage are

$$I(0^+) = 0 \qquad \text{and} \qquad I'(0^+) = \frac{1}{L}E(0^+)$$

and based on the definition of the state vector, these give

$$x_1(0^+) = I(0^+) = 0 \qquad \text{and} \qquad x_2(0^+) = I'(0^+) - \frac{1}{L}E(0^+) = 0$$

Thus, for this problem, the proper ICs for the state vector are $\underline{x}(0) = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$.

To illustrate the various solution techniques discussed above, let's perform a time domain simulation [i.e. determine y(t) for a given u(t)] for this system using three different methods:

A. Continuous analytical solution using the closed form representation of $e^{\underline{\underline{A}}t}$
B. Discretization of the LTI system
C. Numerical integration of the state equations

All three solution schemes were implemented in Matlab. This is true even for the analytical solution where we took advantage of Matlab's ability to easily evaluate matrix equations. The solutions are implemented within the **ltidemo1.m** file. A listing of the **ltidemo1.m** file is given in Table 3.2. A brief overview of each method follows:

<u>Solution Method A</u>

The analytical solution for the state response versus time for a LTI system with a scalar **constant input forcing function** can be written as

$$\underline{x}(t) = e^{\underline{\underline{A}}t}\underline{x}_0 + e^{\underline{\underline{A}}t}\left[\int_0^t e^{-\underline{\underline{A}}\tau}d\tau\right]\underline{b}u$$

This expression can be simplified as follows:

$$\underline{x}(t) = e^{\underline{\underline{A}}t}\underline{x}_0 + e^{\underline{\underline{A}}t}\left[-\underline{\underline{A}}^{-1}e^{-\underline{\underline{A}}\tau}\right]_0^t \underline{b}u$$

$$\underline{x}(t) = e^{\underline{\underline{A}}t}\underline{x}_0 + e^{\underline{\underline{A}}t}\left[\underline{\underline{A}}^{-1}\left(\underline{\underline{I}} - e^{-\underline{\underline{A}}t}\right)\right]\underline{b}u \;=\; e^{\underline{\underline{A}}t}\underline{x}_0 + \underline{\underline{A}}^{-1}\left(e^{\underline{\underline{A}}t} - \underline{\underline{I}}\right)\underline{b}u$$

or $$\underline{x}(t) = e^{\underline{\underline{A}}t}\underline{x}_0 + \underline{\underline{A}}^{-1}e^{\underline{\underline{A}}t}\underline{b}u - \underline{\underline{A}}^{-1}\underline{b}u \;=\; e^{\underline{\underline{A}}t}\left(\underline{x}_0 + \underline{\underline{A}}^{-1}\underline{b}u\right) - \underline{\underline{A}}^{-1}\underline{b}u$$

where, in the last manipulation, we used that fact that $\underline{\underline{A}}^{-1}$ and $e^{\underline{\underline{A}}t}$ commute.

To implement this last expression, we used Sylvester's Theorem for the case of distinct roots to evaluate $e^{\underline{\underline{A}}t}$. This required that the eigenvalues be computed from

$$\left|\underline{\underline{A}} - \lambda\underline{\underline{I}}\right| = \begin{vmatrix} -\lambda & 1 \\ -\dfrac{1}{LC} & -\dfrac{R}{L} - \lambda \end{vmatrix} = \lambda^2 + \frac{R}{L}\lambda + \frac{1}{LC} = 0$$

and that the matrix manipulations in the above expression, including the inverse operation, be performed. All these operations were completed directly within Matlab as shown in the first part of **ltidemo1.m** (see Table 3.2). The solutions for the nominal set of parameters given above and for the case where R = 10 ohms are plotted in Figs. 3.4 and 3.5, respectively. Note that R = 100 ohms gives an overdamped response and an underdamped solution is obtained for R = 10 ohms. This behavior can be explained simply by computing the eigenvalues for the two cases.

<u>Solution Method B</u>

The discrete solution of the LTI state equations is incorporated as part of Matlab's Control Toolbox. This toolbox has a series of functions for the analysis of linear stationary systems using the same terminology developed in these notes. In the recent versions of Matlab, an object-oriented approach to working with systems is implemented. In particular, Matlab defines an LTI state-space object via knowledge of the $\underline{\underline{A}}$, $\underline{\underline{B}}$, $\underline{\underline{C}}$, and $\underline{\underline{D}}$ matrices from the standard state-space representation. For this situation the **ss** command is used as follows (see Matlab's help file for function **ss**):

sys = ss(A,B,C,D)

This creates an LTI object named **sys** that can be manipulated within several other Matlab functions. For the case of a unit step input, for example, one has

[y,t,x] = step(sys) or **[y,t,x] = step(sys,t)**

where **y** and **x** contain the output and state vector time domain responses in a 3-D array format. The number of rows in **y** and **x** is determined by the number of time points in the equally-spaced time vector **t**. The **y** array has as many columns as outputs and **x** has as many columns as states (i.e. the order of the system). Finally, the number of pages (the length of the 3rd dimension) is equal to the number of inputs to the system. Thus, there is a 2-D matrix for each system input.

In the first form given above, the sample period and the number of time points in the **t** vector is determined automatically by Matlab. In the second form, the **t** vector is specified by the user and passed into Matlab's **step** function. Note that Matlab has similar functions, **impulse** and **lsim**, respectively, if the time domain input function is a unit impulse or a general input vector.

For the specific RLC circuit of interest in this example, the appropriate calling sequence is

sys = ss(A,10*B,C,D); [y,t,x] = step(sys,t);

since there is only a single input quantity and its strength is 10 units (and the default is for a unit step input). The Matlab m-file that implements the step function "effectively" calls two other m-files. The first one, **c2d**, converts the continuous state space system into a discrete state space representation by computing the $\underline{\underline{G}}$ and $\underline{\underline{H}}$ matrices given in eqns. (3.47) and (3.48) using Matlab's built-in matrix exponential routine, **expm**, with a sampling time determined by the constant interval width in vector **t**. It then calls **ltitr** (Linear Time Invariant Time Response), which simply implements the recursive expression given in eqn. (3.45).

As expected, the solutions using Method B are identical with those from Method A. The student is referred to the second part of **ltidemo1.m** in Table 3.2 for the details of the Matlab implementation (i.e. a single line of code…), and to Figs. 3.4 and 3.5 for displays of the output responses using Method B.

<u>Solution Method C</u>

The last method to be highlighted in this example is the numerical integration technique implemented with Matlab's **ode45** routine. This routine applies an adaptive step control algorithm by obtaining error estimates using two Runge Kutta (RK) predictions of different order. The **ode45** function uses a 4th and 5th order RK set for high accuracy. An overview of how to use the **ode45** routine in Matlab can be obtained by simply typing **help ode45** at the Matlab prompt (this procedure is the best way to learn how to use any Matlab function). In the current case, the call to **ode45** should resemble the following,

[t,x] = ode45 (ftx,[t1,t2],xo,options);

where **t1** and **t2** are the initial and final integration times, **xo** represents the initial condition of the state vector, and **options** is an ODE options structure within Matlab that allows the user to control several options within the ODE numerical integration routine (see function **odeset** for the various options that can be adjusted). The default options are satisfactory for many cases, except for possible control of the user-specified error tolerance. The outputs from **ode45** includes a time vector containing the discrete time points where the state is evaluated, and the state vector in a matrix whose columns are the various elements of the state versus time. Note here that the time vector is determined automatically within **ode45** and it is usually not evenly spaced because of the adaptive step control algorithm that is used. The user can, on option, include an equally spaced time vector, **t**, instead of the two-element vector, **[t1,t2]**, to control this aspect of the **ode45** solution, as desired.

The function name, **ftx**, in general, is a function handle to the function routine or anonymous function that evaluates the right hand side of the state equations at each time point. The user supplies this routine, and it can be specialized for any particular case of interest. For the current RLC series circuit simulation with a step input, this function is so simple that it was implemented within a single-line anonymous function just before the call to **ode45**. Here the form of the anonymous function is

ftx = @(t,x) A*x + B*U

where the **A** and **B** matrices and inputs **U** must be pre-defined. If, in some other case, the state equations or the inputs are more complicated, then a separate function file would be written, and the function handle, **ftx**, would simply point to this function file.

The results from Method C, as expected, are identical to those from the other techniques. This can be seen by comparing the results in Figs. 3.4 and 3.5. In the current example, Method B, the discrete solution method, is clearly the easiest to use since the problem fits exactly into the LTI class of problems, and Matlab's **ss**, **step**, **impulse** and **lsim** routines were designed to easily treat LTI systems. For nonlinear or variable coefficient systems, the numerical integration technique with the **ode45** routine would probably be the method of choice.

Table 3.2 Listing of the ltidemo1.m Matlab file.

```
%
%   LTIDEMO1.M  MATLAB sample that demonstrates various solution schemes for
%          Linear Time Invariant (LTI) systems.  DEMO1 in this series focuses
%          on time domain solutions.
%
%   Three solutions schemes are demonstrated:
%     1. Analytical solution using continuous form of matrix exponential
%     2. Discretization of LTI system using matrix exponential with sampling time T
%     3. Numerical integration of the state equations
%
%   The model of interest involves a simple RLC electrical circuit with an applied
%   voltage
%     Li'' + Ri' + i/C = e'(t)
%
%   This can be put into standard state space form as follows:
%      d |x1|   |   0      1 ||x1|   |  1/L   |
%      - |  | = |            ||  | + |        |u
%      dt|x2|   |-(1/LC) -R/L||x2|   |-(R/L^2)|
%
%               |x1|
%      y = [1 0]|  | + [0]u        with y = i(t) and u = e(t)
%               |x2|
%   and
%      x1 = y   and x2 = x1'-(1/L)u
%
%   File prepared by J. R. White, UMass-Lowell (last update: Feb. 2020)
%

      clear all,   close all,   nfig = 0;
%
%   define coefficients for specific problem
      L = 0.1;                % inductance (henry)
      Ca = 0.001;             % capacitance (farad)
      RR = [100 10 0];        % various resistances (ohms)
      ir = menu('Choose R value in RLC circuit', ...
                'R = 100 ohms (over damped response)', ...
                'R = 10 ohms (under damped response)', ...
                'R = 0 ohms  (undamped response)');
      R = RR(ir);
%
%   create state space model
      A = [0 1;-1/(L*Ca) -R/L];
      B = [1/L -R/L^2]';     C = [1 0];     D = 0;
%
%   set integration time, time vector, and initial conditions
      t1 = 0;   t2 = 0.25;   nt = 251;   t = linspace(t1,t2,nt);   xo = [0 0]';
```

```
%
%   set strength of input step function
      us = 10;
%
%   Part A  Continuous Analytical Solution
%
%   Note: This solution uses Sylvester's theorem for distinct roots to construct
%   the continuous matrix exponential form of the solution.  This will have to be
%   modified if the roots are repeated. Also the solution here is for the specific
%   case of a step input to a 2nd order LTI system.
%
%   find eigenvalues of state matrix (use roots command for this simple 2x2 case)
      pp = [1 R/L 1/(L*Ca)];    rr = roots(pp);
%
%   determine matrix/vector constants in final equations
      aibu = inv(A)*B*us;   xxo = xo+aibu;
      AA1 = (A-rr(2)*eye(size(A)))/(rr(1)-rr(2));
      AA2 = (A-rr(1)*eye(size(A)))/(rr(2)-rr(1));
%
%   determine terms with time behavior
      tb1 = exp(rr(1)*t);   tb2 = exp(rr(2)*t);
%
%   construct state vector versus time
      xa = zeros(2,nt);   xa(:,1) = xo;
      ya = zeros(1,nt);   ya(:,1) = C*xo + D*us;
      for k = 2:nt
        xa(:,k) = (AA1*tb1(k)+AA2*tb2(k))*xxo - aibu;
        ya(:,k) = C*xa(:,k) + D*us;
      end
%
%   note that this problem only has one input, thus
      xa = xa';   % store final state vector in matrix with jth column being xj(t)
      ya = ya';   % store final output vector in matrix with jth column being yj(t)
%
%   Part B  Discrete Solution of LTI System
%
%   simulate time domain response
      sys = ss(A,B*us,C,D);    [yb,tb,xb] = step(sys,t);
%
%   Part C  Numerical Solution of State Space System
%
%   Note: Since the state matrices and u(t) are constants for this case, we will use
%   a simple anonymous function to evaluate the derivatives of the state for use by
%   the ODE45 ode solver.
      U = us;   options = odeset('RelTol',1.0e-6);
      ftx = @(t,x) A*x + B*U;
      [tn,xc] = ode45(ftx,[t1,t2],xo,options);
      xct = xc';    ntn = length(tn);   yc = zeros(1,ntn);
      for k = 1:ntn
        yc(:,k) = C*xct(:,k) + D*U;
      end
      yc = yc';   % store final output vector in matrix with jth column being yj(t)
%
%   Plot all three solutions (current in milliamps versus time)
      nfig = nfig+1;   figure(nfig)
      plot(t,1000*ya,'r',tb,1000*yb,'g.',tn,1000*yc,'b.','LineWidth',2),grid
      title(['LTIDemo1:  Step Response for a Simple RLC Circuit (R = ', ...
           num2str(R),' ohms)'])
      ylabel('Current (ma)'), xlabel('Time (sec)')
      legend('Continuous Analytical Solution','Discrete Solution', ...
             'Numerical Solution')
%
%  end of demo
```

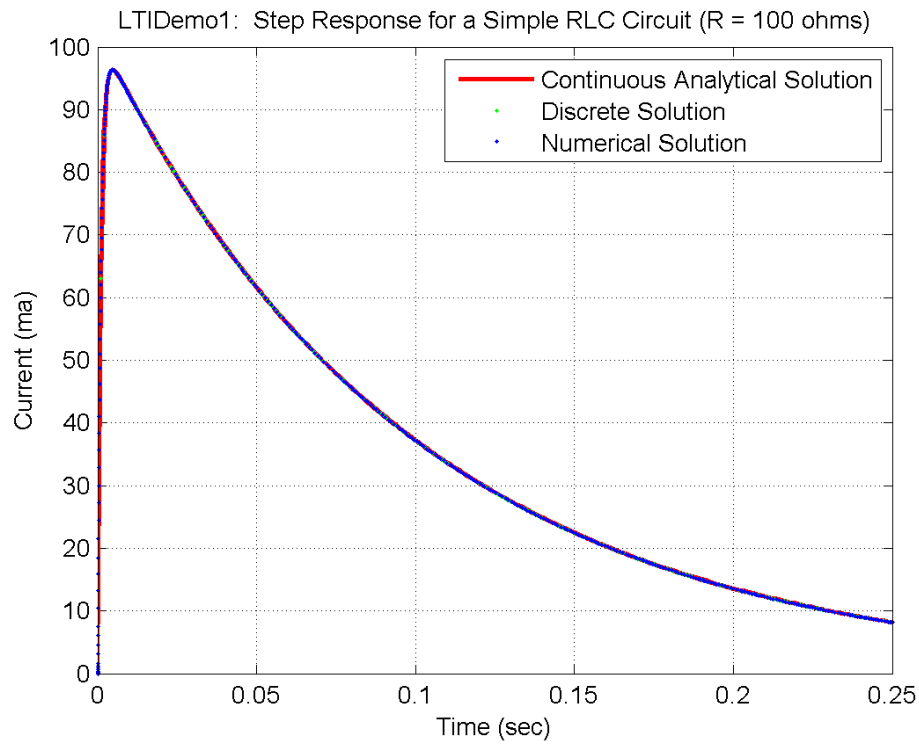


Fig. 3.4 Various solutions for a step input to an RLC circuit (R = 100 ohms).

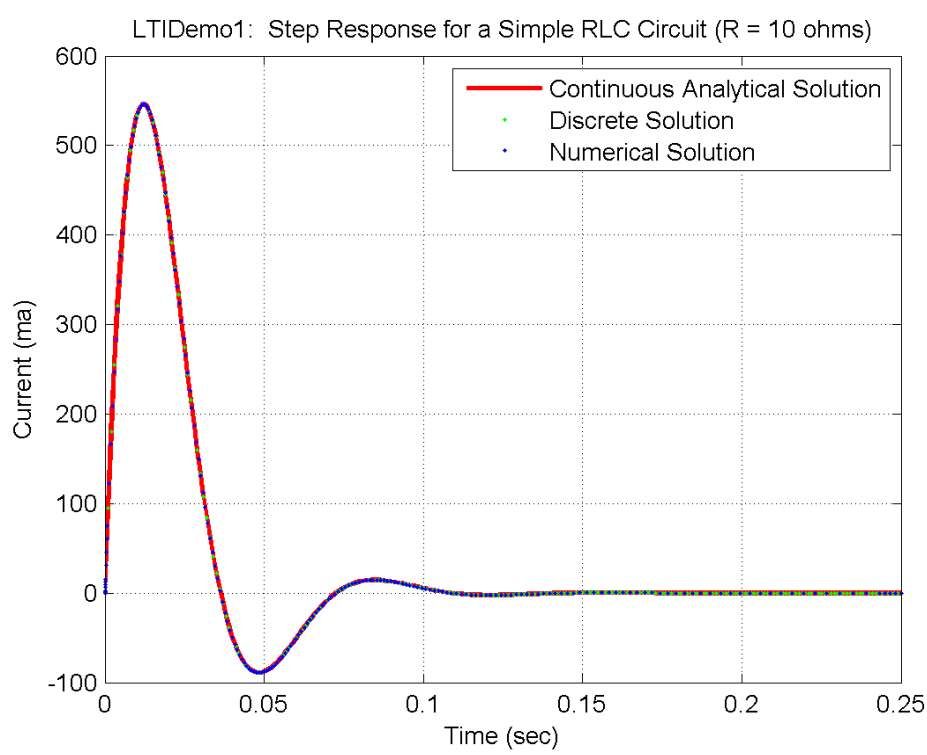


Fig. 3.5 Various solutions for a step input to an RLC circuit (R = 10 ohms).