New Orleans JAZZED about Engineering Education

Paper ID #14411

Application of Micro Computer in Optimal Linearization of Nonlinear Systems

Dr. Alireza Rahrooh, Daytona State College

Alireza Rahrooh received B.S., M.S., and Ph.D. degrees in electrical engineering from University of Akron, Ohio in 1979, 1986, and 1990, respectively. He worked as an Electronic Engineer from 1979 to 1984. He was involved in conducting research for the Electrical Power Institute and NASA Lewis Research Center from 1984 to 1998. He was appointed to a faculty position in Electrical Engineering at Penn State University in 1988. In 1994, he joined the faculty of Engineering Technology at UCF till August of 2010 when he moved to Daytona State College. He has presented numerous papers at various conferences, is the author of more than 100 technical articles and recipient of 30 awards. His research interests include simulation, nonlinear dynamics, system identification and adaptive control. He is a member of ASEE, IEEE, Eta Kappa Nu, and Tau Beta Pi.

Dr. Walter W. Buchanan P.E., Texas A&M University

Walter W. Buchanan is a Professor at Texas A&M University. He is a Fellow and served on the Board of Directors of both ASEE and NSPE, is a past president of ASEE and the Massachusetts Society of Professional Engineers, and is a registered P.E. in six states. He is a past member of the Executive Committee of ETAC of ABET and is on the editorial board of the Journal of Engineering Technology.

Prof. Robert De La Coromoto Koeneke, Daytona State College

Robert Koeneke is an Associate Professor of Electrical Engineering Technology at Daytona State College. He received his B.S. in Electronics Engineering from Universidad Simon Bolivar in 1977 and his M.S. in Computer Science from Santa Clara University in 1982. His 34 years of professional career covers: teaching at undergraduate and graduate level, planning, developing and managing project in the areas of Telecommunications and Information Systems. His research interest includes embedded systems, digital programmable devices and computer communications. He is a member of IEEE, ASEE and ACM.

Application of Micro Computer in optimal Linearization of Nonlinear Systems

Abstract

This paper presents a computer-assisted method to generate accurate linear models of nonlinear systems with reduced biasing errors. The technique, which is based on finite difference methods, approximates partial derivatives of a Taylor series expansion of nonlinear state equations about a nominal operating point or trajectories. The matrices of the linear state-space representation of the nonlinear system can be determined using personal-computer software. It can be shown that positive and negative perturbations in the system inputs can result in more accurate linear model. The advantages of this approach are illustrated and discussed. The proposed techniques will be useful in motivating students to pursue a graduate degree in institutions where the limited budget will not allow purchasing costly modeling/simulation packages and software.

Introduction

Most standard control design techniques have been developed for linear systems. More importantly, any derived technique for a nonlinear system may not be applicable to other nonlinear systems due to their complicated dynamics. Thus, a good linear representation of these physical systems must be derived [1]. In practice, it is found that some behavior of nonlinear systems only occurs if they are driven into certain operation regions. For these systems, the linear model may give relatively accurate results over a wide range of operating conditions. However, there are numerous physical systems which have strong nonlinear characteristic. For these systems, a linearized model is valid only for a limited range of operation, and often only at the operating point at which the linearization is done. A particular method used to linearize a nonlinear system about an operating point is the "offset derivative". This technique makes use of the method of finite difference to approximate partial derivatives of a Taylor series expansion of the nonlinear state equations about a nominal operating point or trajectories. The difficulty encountered in using the method of finite difference is that small numerical errors can appear in partial-derivative approximations. These errors can cause large enough errors in the Jacobian matrices' elements that the eigenvalues of the system can be changed. Since, the trajectories have different slops in different directions about the operating points; these errors can be reduced or eliminated through positive and negative perturbation about the operating point. Once the full state linear model of the system has been determined, it may be possible to reduce the order of the linear models while still retaining the important dynamic characteristics of the system. This is desirable for two reasons. First, a lower order system is easier to handle mathematically; second, controls developed using optimal control techniques require that all specified states of the system be used to derive the optimal control. However, some of these states are not measurable and therefore cannot be used for control purposes. Furthermore, if all the states of a large-order system were measurable, the control derived would be too complex to implement. Once the linear models are generated, they should be evaluated as to how well the approximate the dynamics of the nonlinear system about the operating points for which they were generated. As an example this evaluation is made for the third order

Bernnan and Leake Jet Model [2] by comparing the full-state linear model and nonlinear engine model.

Linearization Mathematics

Represent a nonlinear system by the following vector-matrix state equations:

$d\mathbf{X}(t)/dt = \mathbf{X}'(t) \equiv f[\mathbf{X}(t), \mathbf{U}(t), t]$ (1)

$\mathbf{Y}(t) \equiv g[\mathbf{X}(t), \mathbf{U}(t), t]$

Where $\mathbf{X}(t)$ represents the nx1 state vector, $\mathbf{U}(t)$ the px1 input vector, $f[\mathbf{X}(t), \mathbf{U}(t), t]$ denotes an nx1 function, $g[\mathbf{X}(t), \mathbf{U}(t), t]$ represents 1xk function, and Y(t) is 1xK vector. In general f and g are a function of state vector and the input vector. The state variables $\mathbf{X}(t)$ cannot change instantaneously with time. While output variables $\mathbf{Y}(t)$ may. It should be clear that no single linear model can accurately represent the system because of generally wide operating range and nonlinear characteristics. Thus, linear model must be derived at various conditions particularly, if the Jacobian, **J**, is experiencing a large variation in time and space throughout the operating envelope [3].

Let the nominal operating trajectory and the output be $\mathbf{X_o}$, $\mathbf{Y_o}$ respectively, which correspond to the nominal input $\mathbf{U_o}$ and some fixed initial conditions. Expanding the nonlinear state equation (1) into a Taylor series about $\mathbf{X}(t) = \mathbf{X_o}$ yields

$$\boldsymbol{X_i'(t) = f_i(X_O, U_O, t) + \sum_{j=1}^{n} \frac{\partial f_i(X,U,t)}{\partial X_j} \Big|_{xo,uo} (X_j - X_{oj}) + \sum_{j=1}^{p} \frac{\partial f_i(X,U,t)}{\partial U_j} \Big|_{xo,uo} (U_j - U_{oj}) + HOT} \quad (2)$$

Let

$$\boldsymbol{\Delta X_i = X_i - X_{oi} \qquad i = 1, 2, \ldots, n}$$

$$\boldsymbol{\Delta U_j = U_j - U_{oi} \qquad j = 1, 2, \ldots, p}$$

Then since

$$\boldsymbol{\Delta X_i' = X_i' - X_{oi}'}$$

$$\boldsymbol{X_{oi}' = f_i(X_O, U_O)}$$

If all the higher order terms (HOT) are negligible then, equation (2) can be written as

$$\boldsymbol{\Delta X_i'(t) = \sum_{j=1}^{n} \frac{\partial f_i(X,U)}{\partial X_j} \Big|_{xo,uo} \Delta X_j + \sum_{j=1}^{p} \frac{\partial f_i(X,U)}{\partial U_j} \Big|_{xo,uo} \Delta U_j}$$

At an operating point the system is assumed to be time invariant: thus, the above equation may be written in vector-matrix form

$$\Delta \boldsymbol{X}' = \underline{\boldsymbol{J}}\Delta \boldsymbol{X} + \underline{\boldsymbol{G}}\Delta \boldsymbol{U} \tag{3}$$

The J and G matrices are Jacobian matrices defined by

$$\underline{J} = \begin{bmatrix} \frac{\partial f_1}{\partial X_1} & \frac{\partial f_1}{\partial X_2} & \cdots & \frac{\partial f_1}{\partial X_n} \\ \frac{\partial f_2}{\partial X_1} & \frac{\partial f_2}{\partial X_2} & \cdots & \frac{\partial f_2}{\partial X_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_n}{\partial X_1} & \frac{\partial f_n}{\partial X_2} & \cdots & \frac{\partial f_n}{\partial X_n} \end{bmatrix} \qquad \underline{\boldsymbol{G}} = \begin{bmatrix} \frac{\partial f_1}{\partial U_1} & \frac{\partial f_1}{\partial U_2} & \cdots & \frac{\partial f_1}{\partial U_p} \\ \frac{\partial f_2}{\partial U_1} & \frac{\partial f_2}{\partial U_2} & \cdots & \frac{\partial f_2}{\partial U_p} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_n}{\partial U_1} & \frac{\partial f_n}{\partial U_2} & \cdots & \frac{\partial f_n}{\partial U_p} \end{bmatrix}$$

Or in more compact form:

$$\boldsymbol{J_{ij}} = \frac{\boldsymbol{\partial f_i}}{\boldsymbol{\partial X_j}} \qquad \begin{matrix} \boldsymbol{i = 1, 2, \ldots, n} \\ \boldsymbol{j = 1, 2, \ldots, p} \end{matrix}$$

(4)

$$\boldsymbol{G_{ij}} = \frac{\boldsymbol{\partial f_i}}{\boldsymbol{\partial U_j}} \qquad \begin{matrix} \boldsymbol{i = 1, 2, \ldots, n} \\ \boldsymbol{j = 1, 2, \ldots, p} \end{matrix}$$

Thus, the nonlinear system has been linearized at a nominal operating point. The output equation of (1) may be linearized using the preceding technique by assuming small perturbations about the operating point, i.e.

$$\Delta \boldsymbol{Y_i} = \boldsymbol{Y_i} - \boldsymbol{Y_{oi}}$$

With ***i= 1, 2,……, k***

The output Taylor series expansion is:

$$\boldsymbol{Y_i(t)} = \boldsymbol{g_i(X_O, U_O, t)} + \sum_{j=1}^{k} \frac{\partial g_i(X,U,t)}{\partial X_j} \Big|_{xo,uo} \boldsymbol{(X_j - X_{oj})} + \sum_{j=1}^{p} \frac{\partial g_i(X,U,t)}{\partial U_j} \Big|_{xo,uo} \boldsymbol{(U_j - U_{oj})} + \boldsymbol{HOT} \tag{5}$$

Since $\Delta \boldsymbol{Y_{oi}} = \boldsymbol{g_i(X_O, U_O)}$

Now, if all the higher order terms (HOT) are small compared with the order terms then, equation (5) can be written as

$$\Delta \boldsymbol{Y_i} = \sum_{\boldsymbol{j=1}}^{\boldsymbol{k}} \frac{\boldsymbol{\partial g_i(X,U,t)}}{\boldsymbol{\partial X_j}} \Big|_{\boldsymbol{xo,uo}} \Delta \boldsymbol{X_j} + \sum_{\boldsymbol{j=1}}^{\boldsymbol{p}} \frac{\boldsymbol{\partial g_i(X,U)}}{\boldsymbol{\partial U_j}} \Big|_{\boldsymbol{xo,uo}} \Delta \boldsymbol{U_j}$$

At an operating point the system is assumed to be time invariant: thus, the above equation may be written in vector-matrix form.

$$\Delta \boldsymbol{Y} = \underline{\boldsymbol{C}} \Delta \boldsymbol{X} + \underline{\boldsymbol{D}} \Delta \boldsymbol{U} \tag{6}$$

The **C** and **D** matrices are input Jacobian matrices defined by

$$\underline{\boldsymbol{C}} = \begin{bmatrix} \frac{\partial g_1}{\partial X_1} & \frac{\partial g_1}{\partial X_2} & \cdots & \frac{\partial g_1}{\partial X_n} \\ \frac{\partial g_2}{\partial X_1} & \frac{\partial g_2}{\partial X_2} & \cdots & \frac{\partial g_2}{\partial X_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial g_k}{\partial X_1} & \frac{\partial g_k}{\partial X_2} & \cdots & \frac{\partial g_k}{\partial X_n} \end{bmatrix} \quad \underline{\boldsymbol{D}} = \begin{bmatrix} \frac{\partial g_1}{\partial U_1} & \frac{\partial g_1}{\partial U_2} & \cdots & \frac{\partial g_1}{\partial U_n} \\ \frac{\partial g_2}{\partial U_1} & \frac{\partial g_2}{\partial U_2} & \cdots & \frac{\partial g_2}{\partial U_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial g_k}{\partial U_1} & \frac{\partial g_k}{\partial U_2} & \cdots & \frac{\partial g_k}{\partial U_n} \end{bmatrix}$$

Or in more compact form

$$\boldsymbol{C_{ij}} = \frac{\boldsymbol{\partial g_i}}{\boldsymbol{\partial X_j}} \qquad \begin{matrix} \boldsymbol{i = 1, 2, \ldots, k} \\ \boldsymbol{j = 1, 2, \ldots, n} \end{matrix}$$

$$\boldsymbol{D_{ij}} = \frac{\boldsymbol{\partial g_i}}{\boldsymbol{\partial U_p}} \qquad \begin{matrix} \boldsymbol{i = 1, 2, \ldots, k} \\ \boldsymbol{j = 1, 2, \ldots, n} \end{matrix} \tag{7}$$

The approximating of matrix elements of the linearized model, equations (4) and (7) is accomplished by a finite difference method using **PC-MATLAB**. The program only requires the desired operating point, inputs, perturbation size, and the nonlinear model state equations. Then, the individual states are varied one at a time while the others are held constant. This is done to derive the **J** and **C** matrices. To generate **G** and **D** the inputs are varied one at a time while holding the others constant including the states. The biasing errors in the finite-difference calculations of **J** may be reduced by perturbing the state in the positive direction from the operating point to generate **J+** and **C+**. The, the states are perturbing in the negative direction from the operating point to derive **J-** and **C-** matrices. Finally, the **J+** and **J-** matrices and **C+** and **C-** matrices are averaged to obtained the **J** and **C** matrices. The perturbation size must be within the linear region of the operating point, and it must be larger than the iteration tolerance for implicit calculations.

The larger perturbation size ensures easier calculations of the partial derivative as long as it does not exceed the linearity limit about the operating point.

Another important potential source of error due the linearization process is inability of the linear model to converge to the expected steady-state value. Thus, some modification of the **G** matrix is necessary to eliminate this steady-state error. Note that at steady-state $\Delta \boldsymbol{X}' = \mathbf{0}$ which results in

$$\underline{\boldsymbol{J}}\Delta \boldsymbol{X} + \underline{\boldsymbol{G}}\Delta \boldsymbol{U} = \mathbf{0}$$

then

$$\Delta \boldsymbol{X} = -\underline{\boldsymbol{J}}^{-1}\underline{\boldsymbol{G}}\Delta \boldsymbol{U} = \mathbf{0} \qquad (8)$$

$\Delta \boldsymbol{X}$ is calculated by making small perturbations in the inputs. Then, the term $(-\underline{\boldsymbol{J}}^{-1}\underline{\boldsymbol{G}})$ can be evaluated by $\Delta \boldsymbol{X}/\Delta \boldsymbol{U}$, since **J** is known, the modified $\underline{\mathbf{G}}_{\mathbf{m}}$ is

$$\underline{\mathbf{G}}_{\mathbf{m}} = -\underline{\boldsymbol{J}}\,(-\underline{\boldsymbol{J}}^{-1}\underline{\boldsymbol{G}}) \qquad (9)$$

For the matrix **D**, inserting equation (8) in equation (6) results in

$$\Delta \boldsymbol{Y} - \underline{\boldsymbol{C}}(-\underline{\boldsymbol{J}}^{-1}\underline{\boldsymbol{G}}\Delta \boldsymbol{U}) = \underline{\boldsymbol{D}}_{\boldsymbol{m}}\,\Delta \boldsymbol{U} \qquad (10)$$

Where the modified **Dm** is calculated by making small changes in inputs while holding all the states constant such that equation (10) is satisfied. Once the matrices are properly determined, the linearity of these matrices must first be confirmed. This is normally done by generating a $\underline{\boldsymbol{J}}_1$matrix with one perturbation size and then a $\underline{\boldsymbol{J}}_2$ matrix with different perturbation size. If the eigenvalues of both $\underline{\boldsymbol{J}}_1$ and $\underline{\boldsymbol{J}}_2$ are within the frequency range of interest, then both $\underline{\boldsymbol{J}}_1$ and $\underline{\boldsymbol{J}}_2$ are reasonably valid. This indicates that both perturbation sizes are within the linear region about the operating point. Second, the linear models should be evaluated to see how well they approximate the dynamics of the nonlinear system about the operating point from which they are generated. This is accomplished by comparing the simulation results of both linear and nonlinear models using a very accurate numerical integration algorithm. For a given input the linear model should maintain the dynamics of the nonlinear model, and the difference between the steady-state errors of both model responses should be a small as possible. This was done for a nonlinear system given in the next section.

Nonlinear Jet Engine Model

As an example of the linear-model generation procedure, the nonlinear third-order Bernnan and Leaks engine model described by the following nonlinear differential equation[2] is considered.

$$\mathbf{T_3} = \mathbf{0.64212} + \mathbf{0.35788}\boldsymbol{N}^2$$

$$\boldsymbol{W}_3' = \mathbf{1.3009}\boldsymbol{N} - \mathbf{0.13982}\left[\boldsymbol{P}_4^2 - \sqrt{(\boldsymbol{P}_4 + \mathbf{0.41688}\boldsymbol{N} - \mathbf{0.0899}\boldsymbol{PN})}\right]$$

$$P_4' = W_f' \ (0.93586P_4/P_\rho + 31.486) + 21.435W_3'T_3 - 53.864\ P_4^2/P_\rho$$

$$P_\rho' = 37.78\ W_3' - 38.448\ W_3' + 0.66849\ W_f'$$

$$N' = (1.258/N)\left(\frac{P_4^2}{P_\rho} - W_3'N^2\right) \quad (11)$$

Where P_4 is Combustor Pressure, P_ρ is Combustor density, N is Rotor Speed, W_3' is Compressor Discharge Mass Flow, T_3 is Compressor discharge Temperature, W_f' is fuel input rate. The system is normalized about $W_f' = 1.0$ with initial conditions:

$P_4 = 0.5831$, P_ρ =1.77504, N =0.54589

States:

$X_1 = P_4$, $X_2 = P\rho$, $X_3 = N$

And input:

$U = W_f'$

The resulting linear model from the ± 0.3 percent perturbation with $X_O = [1\ 1\ 1]^T$ as the operating point is:

$$\begin{bmatrix} X_1' \\ X_2' \\ X_3' \end{bmatrix} = \begin{bmatrix} -112.266 & 52.925 & 42.259 \\ -48.1501 & 0 & 47.443 \\ 2.8377 & -1.258 & -4.096 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} + \begin{bmatrix} 32.422 \\ 0.668 \\ 0 \end{bmatrix} [U] \quad (12)$$

Therefore:

$$J = \begin{bmatrix} -112.266 & 52.925 & 42.259 \\ -48.1501 & 0 & 47.443 \\ 2.8377 & -1.258 & -4.096 \end{bmatrix}, \text{ and } G = \begin{bmatrix} 32.422 \\ 0.668 \\ 0 \end{bmatrix}$$

The eigenvalues resulting from using different perturbation sizes is tabulated in table (1). Also, figure (1) shows the simulation of this linear model compared to the actual nonlinear simulation. Adams-Bashforth two step (**AB-2**) integration method [4] ($x_{n+1} = x_n + 0.5 * h[3x_n' - x_{n-1}']$) with stepsize of h=0.002 sec is used to simulate the linear model and nonlinear model. Also,

Table (1) Eigenvalues variation of linearized Brennan and Leake engine

Eigenvalues	**Perturbation**					
	0.3% of operating point			**0.3% of operating point**		
	± 0.3	+0.3	-0.3	± 0.5	+0.5	-0.5
λ_1	-81.24	-82.03	-80.34	-81.24	-82.17	-80.26
λ_2	-32.33	-31.19	-33.51	-32.33	-31.17	-33.33
λ_3	-3.15	-3.14	-3.17	-3.15	-3.14	-3.16

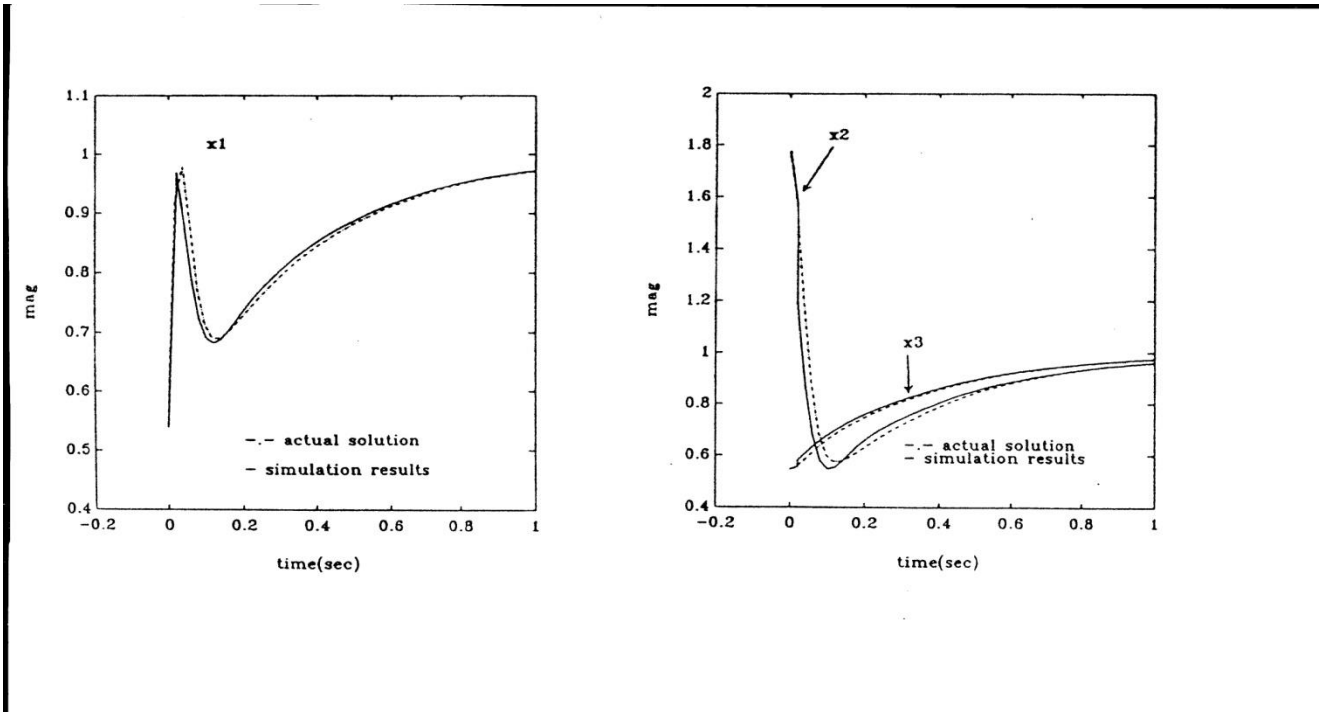

Figure (1) Simulation of linear and nonlinear engine model using **AB-2** with h=0.002 sec.

Conclusion

It is critical that computer usage be integrated into problems which involve the application of basic concepts in engineering. A method to derive and validate linear models from a nonlinear digital simulation discussed in this paper can be viewed as an appropriate use of the computer. To derive a good linear model at an operating point, accurate partial derivatives must be obtained. When using the finite-difference method to approximate partial derivatives, small errors in the calculations can occur. These errors are bias errors, which must be minimized or eliminated if possible. This can be accomplished by perturbing the state in both positive and negative directions about an operating point and averaging the resulting partials. For simulations in which differential

equations are solved explicitly, this method results in repeatability of system eigenvalues for different perturbation sizes about the operating point. For simulations in which the differential equations are solved implicitly, exact repeatability of system eigenvalues for different perturbation sizes is difficult to obtain because of the iterative nature of the solutions. Thus, for a given perturbation size the linear model predicts the average steady-state value that the nonlinear simulation would give for the same perturbation size in the plus and minus direction about the operating point.

The accuracy of the matrices was checked by showing that linearity holds for different perturbation sizes. Once the accuracy of the matrices was assured, the ability of the linear models to approximate the nonlinear system at an operating point was investigated. This was done by comparing a similar transient run with the nonlinear and linear simulations.

Finally, the methods used in this paper to generate linear models from a nonlinear digital simulation are general. The application presented is for the third order Bernnan and Leake engine model, but the method is not restricted to that simulation. The problem of bias errors in finite-difference approximations of partial derivative, again are not limited to Bernnan and Leake engine model simulation. They are universal and must be dealt with whenever deriving linear models or comparing transients run with the nonlinear and linear simulations. The technique and the simulation algorithm is very useful in teaching undergraduate and graduate control systems courses.

References

1. Norman S. Nise *Control Systems Engineering*, Sixth Edition, , Wiley publishing Company, Inc., 2011.
2. Bernnan, T. C. and Leake, *Simplified Simulation Models for Control Studies of Turbojet Engines*, Technical Report No. EE-757, Department of Electrical Engineering, University of Notre Dame, 2007.
3. Kuo, B. C., *Automatic Control Systems*, Prentice Hall, Englewood Cliffs, NJ 2012.
4. Lambert, J. D., *Computational Methods in Ordinary, Differential Equations*, Wiley, NY, 1973.

Appendix

The following personal computer software is written in MATLAB to linearized nonlinear systems about a given nominal operating point using finite differences method.

```
function [a,b] = linearize (F,x0,u0,pert)
 %  THIS PROGRAM LINEARIZED THE SYSTEM DESCRIBED IN F ABOUT THE POINT
 % XXXXX. IT
 %  RETURNES THE LINEAR  a  AND  b  MATRICES SUCH AS THAT xdot = a*x + b*u IS
% LINEAR. Pert
%  IS THE RECENT PERTURBATION ABOUT THE OPERATING POINT.
%  TIME IS ASSUMED TO BE INVARIENT IN   F. NOTE THAT THE PERTURBATION IS
 % DONE IN BOTH
%  POSITIVE AND NEGATIVE DIRECTION.
```

```
%  F  MUST BE OF THE FORM
%
 %              Function Xprime = Function_name (u,x)

%   WITH   F = 'function-name' IN THE CALL TO BE LINEARIZED. Function_name IS THE
%  NAME YOU
 %  ASSIGN TO THE FUNCTION.
%  THE FUNCTION MUST RETURN THE DERVATIVE AT THE POINT.
%   WHERE  u  IS THE INPUT AND  x  IS THE STATE OF THE SYSTEM. SEE THE FILE
% JETS3.M AS AN
%   EXAMPLE (THE FUNCTION MUST BE .M FILE). TO RUN AN EXAMPLE USING
%JET3.M, AT THE
%  MATLAB PROMPT TYPE:
%  x0=[1;1;1]
%  u0= 1
%  pert=0.003
%  [a,b]=lineriz('jet3' , x0,u0,pert)

%  CALCUATE PERTURBATIONS AND DERIVATIVE AT OPERATING POINT %
%
delx =x0. * pert;
delx =u0. * pert;
xd= feval (f,u0,x0);
%%
% RETURB  X  IN POSITIVE DIRECTION %
%%
for  i=1 : length (x0),
  for  J=1 : length (x0),
     x=  x0;
     for  k=1 : length (x0),
        if k ==  j,
           if  delx (k) == 0
              dx=eps;
           else
              dx=delx(k);
            end
            x(k)   =  x(k)  +  dx;
            end
            xt  =   feval (F,u0,x);
            if   delx (j)   == 0,
```

```
            dx=eps;
         else
            dx=delx(j);
         end
          xp(I,j)  =  (xt(i)  -  xd(i) )  /  dx;
        end
    end
 %%
 % PERTURB  X   IN NEGATIVE DIRECTION  %
%%
%%
for  i=1 : length (x0),
  for  J=1 : length (x0),
    x=  x0;
    for  k=1 : length (x0),
      if k ==  j,
        if  delx (k) == 0
           dx=eps;
        else
           dx=delx(k);
         end
         x(k)  =  x(k)  +  dx;
         end
         xt  = feval (F,u0,x);
         if  delx (j)  == 0,
            dx=eps;
         else
            dx=delx(j);
         end
        xm (i,j)  =  - (xt(i) -  xd(i) )  /  dx;
      end
  end
%%
% PERTURB  U  IN POSITIVE DIRECTION %
for  i=1 : length (x0),
  for  J=1 : length (u0),
    u=  u0;
    for  k=1 : length (u0),
      if k ==  j,
        if  delu (k) == 0
```

```
            du=eps;
          else
            du=delu(k);
          end
          u(k)   =  u(k)  +  du;
          end
          end

          ut  =   feval (F,u,x0);
          if   delu(j)   == 0,
               du=eps;
          else
               du=delu(j);
          end
          up(i,j)   =   (ut(i)   -   xd(i) )  /  du;
        end
    end
 %%
 %  PERTURB  U  IN NEGATIVE DIRECTION  %
for  i=1 : length (x0),
  for  J=1 : length (u0),
    u=  u0;
    for  k=1 : length (u0),
      if k ==  j,
        if  delu (k) == 0
           du=eps;
         else
           du=delu(k);
          end
          u(k)   =  u(k)  +  du;
          end
          end
          ut  =   feval (F,u,x0);
          if   delu(j)   === 0,
               du=eps;
          else
               du=delu(j);
          end
          um(i,j)   =   (ut(i)   -   xd(i) )  /  delu (j);
        end
```

```
    end
% CALCULATE LINEARIZED SYSTEM (JACOBIAN ) %
a= (xp+xm) / 2;
b= (up+um) / 2;
function yprime = jet3(u,y)
% NONLINEAR 3RD ORDER NORMALIZED BERNNAN AND LEAK TURBOFAN JET ENGINE MODEL
% yprime IS THE DERIVATIVE OF STATE VARIABLE. FUNCTIONS IN THIS FORM ARE VERY
 % USEFUL IN FUNCTION EVALUATION USING MATLAB.
wf=u;
T3=0.64212 + 0.35788*y(3, : )^2;
w3 = 1.3009*y(3, : ) – 0.139825*y (1, : )
w3= w3 – 0.13982*sqrt(y(1,: )^2 + 0.41688*(3, : )^2 -0.899*y(1,: ) *y(3,: )
yp1= (0.93586* y(1,: ) /y(2,: ) + 31.486) *wf+ 21.435*w3*t3 - 53.86* (y(1,: )
yp2= 37.78*w3 – 38.488*y(1,: ) + 0.66849*wf;
yp3 = 1.258* (y(1,: )^2/y(2,: ) – w3*y (3,: )^2)/y(3,: );
yprime = [yp1;yp2;yp3];
```