LINEAR SYSTEM REDUCTION USING CONTINUED FRACTION EXPANSIONS

A DISSERTATION

Submitted in partial fulfilment of the requirements for the award of the degree

of

MASTER OF ENGINEERING

in

ELECTRICAL ENGINEERING

(With Specialization in System Engineering and Operations Research)

By

PRAKASH CHANDRA PANT

DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITY OF ROORKEE
ROORKEE–247 667 (INDIA)

APRIL, 2000

CANDIDATE'S DECLARATION

I hereby declare that the work, which is being presented in the dissertation entitled **"Linear System Reduction Using Continued Fraction Expansions"** in the partial fulfillment of the requirements for the award of **Master of Engineering** in the **System Engineering & Operations Research**, submitted in the **Department of Electrical Engineering, University of Roorkee, Roorkee** is an authentic record of my own work carried out for the period from August-1999 to April - 2000 under the guidance of **Dr. Rajendra Prasad,** Asstt. Professor, Department of Electrical Engineering, University of Roorkee, Roorkee.

I have not submitted the matter embodied in this dissertation for the award of any other degree.

Dated: 27·04·2000

[illegible]

(PRAKASH CHANDRA PANT)

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Dated: 27-04-2000

[illegible]

Dr. Rajendra Prasad

Asstt. Professor,

Department of Electrical Engineering

University of Roorkee.

Roorkee-247667 (India)

ACKNOWLEDGEMENTS

I take this opportunity to express my sincere gratitude to **Dr. Rajendra Prasad** Asstt. Prof., **Electrical Engineering Department, University of Roorkee, Roorkee,** for encouraging me to undertake this dissertation as well as providing me all the necessary guidance and inspirational support throughout this dissertation, without which this dissertation would not have been in its present shape.

Special thanks to Prof. **M. K. Vasantha,** Prof. **A.K. Pant,** Prof. **H. O. Gupta,** . **Dr. S. Kumar**, Associate Prof., **Dr. (Ms.) Indra Gupta** Asstt. Prof. and Prof. **S.C. Saxena**, Head, **Department of Electrical Engineering, University of Roorkee, Roorkee** for encouragement, giving the moral support to do the work, and providing the excellent facilities of the lab., at all times, wholeheartedly, without which it was impossible for me to complete the work.

My sincere gratitude goes to Prof. and Head, Department of Mechanical and Industrial Engineering, for granting me permission to persue M.E. course.

I also take this opportunity to thank **Mr. Sanjay K. Jain,** Research Scholar, Electrical Engineering Department, for helping me in handling my computer with his extensive experience and knowledge which gave momentum and made the dissertation work roll fast. I am indebted to my classmates **Vinay Kumar Mangla, Vengala Chandramouly, Vijander Singh** and **Ms. Asha Rani,** for taking interest in discussing the problem and encouraging.

I also thank **Mr. Kalyan Singh** and **Mr. Rajendra Singh** Computer & Microprocessor lab. for helping me in the dissertation.

I thank all the members of Electrical Engineering Department for their help and encouragement at the hour of need.

I am grateful to the Mech. and Ind. Engg. Community for their kind cooperation and help. Further, I wish to thank to all those who helped me in this work directly or indirectly.

My sincere heartfelt gratitude to my mother for her encouragement and best wishes.

Finally, I wish to express my deep sense of gratitude to my wife Smt. Beena, whose deep affection and continuous encouragement has always been a source of inspiration to me in all my undertaking. I highly appreciate the patience of my daughter Prerna and son Maneesh for bearing all the hardships which came across during the course of this work.

[illegible]

Dated: 27·04·2000 **PRAKASH CHANDRA PANT**

ABSTRACT

Every physical system can be transformed into a mathematical model. The modeling procedure often leads to a comprehensive description of a process in the form of higher order differential equations, which are difficult to solve for analysis and design. These differential equations can be represented either in the form of a high order state space model in the form of first order differential equations or a high order transfer function.

Model order reduction is significant for better understanding a system's behavior, simplifying controller structures, and reducing system analysis and design complexity. In reduced order modeling the high order transfer function model or the high state space model is required to be reduced in such a way that the reduced order model retains the important properties of the original high order system.

In this dissertation continued fraction expansion (CFE) approach has been used for deriving the reduced order models using second Cauer form. The method is quite simple and attractive but it has a serious drawback of producing unstable (stable) models for stable (unstable) systems. To overcome this drawback a mixed method has been developed using Routh approximation, which has been extended for discrete time systems also. The method is having a unique feature, that it gives stable models for stable systems and matches the initial and final values of the responses. The methods discussed, has been tried to reduce single input single output (SISO) systems and extendable for multi-input multi-output (MIMO) systems.

Necessary computer programmes have been developed in 'C++' environment for CFE, bilinear transformation and computing α (Routh) quotients. The results of the original and the reduced models have been compared by plotting the unit step responses with the help of MATLAB software package.

CONTENTS

Chapter-1

INTRODUCTION

1.1 NECESSITY OF MODEL REDUCTION

Complex dynamic systems e.g. power system, communication system, transportation system, economic system etc. are frequently described by a large number of differential equations. When many such systems are interconnected, the resulting system size may be too large to be conveniently handled, even by the large computers. A typical example of this situation occurs in the dynamic-stability studies of modern interconnected power systems. Dynamic stability of power systems is defined as the stability under infinitesimal disturbances with the action of the regulating devices taken into account. Under dynamic conditions, the system equations are linear, but the total number of differential equations that describe the system performance increases rapidly with the increase in the number of interconnected machines. Therefore, the need for techniques to produce lower-order dynamic equivalents of higher-order systems is apparent.

In recent years, considerable attention has been given to the problem of approximating a linear dynamic system of high order by a model having a lower order. The time-domain methods of system simplification are usually based on the neglect of the non-dominant eigen values of the system or on the minimization of a functional of error between the output vectors of the reduced model and original system. On the other hand, the frequency-domain methods of system simplification usually consist of the determination of a transfer function whose frequency response is close to that of the system.

A large number of methods of reduction are based on the retention of the dominant poles of the system in the reduced model. The most important feature of these methods is that the reduced order model is always stable if high order system is stable. However most of these methods assume that the system is described in a state-vector form, and thus involve the computation of the eigen values and eigen vectors of the high order state matrix. This is computationally very cumbersome, and is known to fail when eigen values of the system are widely spread out.

Some of the reasons for using reduced order models of high order linear systems could be:

- The development of state space methods and optimal control techniques has made the design of a control system for high order multivariable system quite feasible when the order of the system becomes very high, special numerical techniques are required to carry out the calculations at a reasonable cost, on a digital computer. Hence the reduced order model reduces computational complexity, which leads to the saving in both the memory and time requirement of the computer.
- A system of uncomfortably high order poses difficulties in its analysis and synthesis. Hence in dealing with such a high order system is to approximate it by a low order system such that characteristics of the original system, e.g., time constant, damping ratio, natural frequency etc. are similar.

1.2 APPLICATION OF REDUCED ORDER MODELS

The reduced order models and reduction techniques have wide application in the control engineering field. A few are discussed below:

- Dynamic errors of high order systems can be calculated by using low order models.
- Transient response sensitivity of high order systems can be predicted by using low order models.

- Provides guidelines for on-line interactive modeling.
- In control system design and in designing reduced order estimators.
- And many other applications like, sub-optimal controls, adaptive controls etc.

1.3 STATEMENT OF MODEL REDUCTION PROBLEM

1.3.1 CONTINUOUS TIME SYSTEMS

Consider the linear time invariant dynamic system expressed in state space form as:

$$\left.\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned}\right\} \qquad \text{.......(1.1)}$$

where $x \in R^n,\ u \in R^q,\ y \in R^p;\ with\ \ p < n,\ q < n\ and\ \ p \le or \ge$ are state, control and output vectors; (A) $_{n*n}$, (B) $_{n*q}$, (C) $_{p*n}$ matrices. q is the number of inputs and p is the number of outputs.

Alternatively (1.1) can be expressed in frequency domain as:

$$\left.\begin{aligned} &Y(s) = G(s) * U(s) \\ &G(s) = C(sI - A)^{-1} B \ = \frac{N(s)}{D(s)} \end{aligned}\right\} \qquad \text{......(1.2)}$$

$$D(s) = \det(sI - A) = s^n + a_1 s^{n-1} + a_2 s^{n-2} + \ldots\ldots + a_{n-1}s + a_n \qquad \text{......(1.3)}$$

For single input single output (SISO) system (p = q = 1) and G(s) is the transfer function which can be put into the form:

$$G(s)=\frac{b_1 s^{n-1}+b_2 s^{n-2}+b_3 s^{n-3}+\dots\dots+b_{n-1}s+b_n}{s^n+a_1 s^{n-1}+a_2 s^{n-2}+\dots\dots+a_{n-1}s+a_n} \qquad \dots\dots(1.4)$$

In the time domain methods a reduced model of order r < n is given by the state equations:

$$\left.\begin{aligned} \dot{x}_r &= A_r x_r + B_r u \\ y_r &= C_r x_r \end{aligned}\right\} \qquad \dots\dots(1.5)$$

where, $x_r \in R^r,\ u \in R^q,\ y_r \in R^p;\ and\ A_r,\ B_r,\ C_r$ are constant matrices with dimensions compatible with $x_r, u\ and\ y_r$, such that for a specified set of inputs the reduced model response is a satisfactory approximation to the system response.

In the frequency domain, for SISO systems, the reduced model of order r < n is represented by:

$$G_r(s)=\frac{d_1 s^{r-1}+d_2 s^{r-2}+d_3 s^{r-3}+\dots\dots+d_{r-1}s+d_r}{s^r+e_1 s^{r-1}+e_2 s^{r-2}+\dots\dots+e_{r-1}s+e_r}=\frac{N_r(s)}{D_r(s)} \qquad \dots\dots(1.6)$$

1.3.2 *DISCRETE TIME SYSTEMS*

Dynamic systems [30] can be described by discrete state-space models or by input-output relationships in the z-transform domain. The purpose here is to extend the ideas and methods of simplification by the continued fraction expansion to discrete time systems. This extension will serve to unify the existing theoretical basis and will allow to present techniques, which are applicable to a wider class of system models. By reviewing the basic system properties of discrete time models in contrast to continuous-time models.

Let the discrete-time model be described by the difference equation:

$$\left.\begin{aligned} x[(k+1)T] &= Ax(kT) + Bu(kT) \\ y(kT) &= Cx(kT) \end{aligned}\right\} \qquad \text{.........(1.7)}$$

Where $x(kT) \in R^n$, $u(kT) \in R^m$, $Y(kT) \in R^p$ are the state, input and output vectors respectively. The A, B and C are coefficient matrices of appropriate dimensions. The input output description of (1.7) is given by the z- transform transfer function:

$$G(z) = C[zI - A]^{-1} B \qquad \text{.......(1.8)}$$

We recall that the use of the relation $z = e^{st}$ transforms the s-plane of the continuous time systems into the z-plane of the discrete systems. In particular:

- The left- half of the s-plane is mapped into the unit circle in the z-plane where the points $s = 0$ and $s = -\infty$ corresponds to $z = 1$ and $z = 0$ respectively.
- The right-half of the s-plane is mapped into the region outside the unit circle in the z-plane where the point $s = \infty$ corresponds to the point $z = \infty$.

Therefore, (1.7) can alternatively described for SISO system in the form of nth order transfer function:

$$G(z)=\frac{b_1 z^{n-1}+b_2 z^{n-2}+b_3 z^{n-3}+\dots\dots+b_{n-1}z+b_n}{z^n+a_1 z^{n-1}+a_2 z^{n-2}+\dots\dots+a_{n-1}z+a_n} \qquad \dots\dots\dots(1.9)$$

1.4 *MODEL REDUCTION USING CONTINUED FRACTION EXPANSION (CFE)*

The basic philosophy which give rise the derivation of simplified models by CFE are based on expanding the original higher order system using continued fraction expansion. As quotients descend lower and lower they have less and less significance as far as the overall system performance is concerned. Hence, truncating the continued fraction after some terms, and inverting the truncated CFE results in a reduced order model.

Consider the following rational transfer function:

$$G(s)=\frac{A_{2,n}s^{n-1}+A_{2,n-1}s^{n-2}+\dots\dots\dots+A_{24}s^3+A_{23}s^2+A_{22}s+A_{21}}{A_{1,n+1}s^n+A_{1,n}s^{n-1}+\dots\dots\dots+A_{14}s^3+A_{13}s^2+A_{12}s+A_{11}} \qquad \dots\dots(1.10)$$

where A_{ij} are constants.

Because a general control system is a low pass filter in nature, therefore in simplification, we take care of the steady state first and then the transient part. Hence we start Continued Fraction Expansion from the constant term, or arrange the polynomials in ascending power of 's'.

So first rewrite the polynomials in ascending order:

$$G(s)=\frac{A_{21}+A_{22}s+A_{23}s^2+\dots\dots\dots+A_{2,n-1}s^{n-2}+A_{2,n}s^{n-1}}{A_{11}+A_{12}s+A_{13}s^2+\dots\dots\dots+A_{1,n}s^{n-1}+A_{1,n+1}s^n} \qquad \dots\dots\dots(1.11)$$

The continued fraction proposed by Chen and Shieh [1], which is equivalent to a Taylor series expansion about s = 0 is [2] obtained as:

$$G(s)=\cfrac{1}{\frac{A_{11}}{A_{21}}+s\frac{A_{31}+A_{32}s+\dots\dots}{A_{21}+A_{22}s+\dots\dots}}$$

$$=\cfrac{1}{\frac{A_{11}}{A_{21}}+\cfrac{s}{\frac{A_{21}}{A_{31}}+s\frac{A_{41}+A_{42}s+\dots\dots}{A_{31}+A_{32}s+\dots\dots}}} \qquad \dots\dots\dots(1.12a)$$

$$=\cfrac{1}{h_1+\cfrac{s}{h_2+\cfrac{s}{h_3+\cfrac{s}{h_4+\frac{s}{\dots}}}}} \qquad \dots\dots\dots(1.12b)$$

where,

$A_{31} = A_{12} - h_1 A_{22}$ $\qquad$ $A_{41} = A_{22} - h_2A_{32}$

$A_{3,n-1} = A_{1n} - h_1 A_{2n}$ $\qquad$ $A_{4,n-1} = A_{2n} - h_2A_{3n}$

$A_{3,n} = A_{1,n+1}$ $\qquad$ $A_{4,n} = A_{2n+1}$

And

$h_1 = A_{11}/A_{21}$, $\qquad h_2 = A_{21}/A_{31}, \dots\dots,$ $\qquad h_i = A_{i,1}/A_{i+1,1}\dots\dots$

In order to find a r^{th} order reduced model it is necessary to keep the first 2r quotients in (1.12) and reconstruct $R_r(s)$ from it. Most important properties of the CFE are:

- It contains most of the essential characteristics of the original model in the first few terms.
- It converges faster than other series expansion.
- It does not require any knowledge of the model eigen spectrum.
- Since the denominator coefficients of the simplified model depend on both the numerator and denominator coefficients of the original model. Hence the continued fraction, like the time moments and Pade approximation, often gives unstable reduced models for stable full models.

Chapter-2

LITERATURE REVIEW

In any modeling/ model reducing task, two often conflicting factors prevail- "simplicity" and "accuracy". Therefore a great deal of experience is needed for a sound compromise between accuracy and simplicity. The common practice has been to work with simple and less accurate models. There are two motivations for this practice:

- The reduction of computational burden for system simulation, analysis and design; and,
- the simplification of control structures resulting from a simplified model.

It should be emphasized that these motivations are distinct in the sense that one does not necessarily imply the other.

The very old technique of model reduction is Aggregation Method [13]. The treatment of aggregation in modern time is probably due to Malinvaud (1956). Aggregation of large-scale linear time invariant systems is not merely a model reduction scheme but more importantly, a conceptual basis for other approximation techniques, including the modal aggregation (Davison, 1966, 1968) which retains the dominant modes of the original system.

One of the more popular methods for large-scale systems order reduction has been the "Continued Fraction" technique first introduced by Chen and Shieh

[1] and extended by many others. The original technique is based on a Taylor series expansion of the system's closed-loop transfer function about s = 0.

This method [1] has been modified and applied to multi input multi output systems and mixed with other methods by others (Chen and Haas,1968; Chuang, [2],1970; Chen and Shieh [4](1970). Various modifications and extensions to continued fraction expansion (CFE) have since been presented by Chen and Shieh [3,4]; Chen and Huang and many others. As pointed out by Wilson, Chen's results are probably the best that have been obtained, although the method is only applicable to single input single out put systems.

One of the better modifications of the original CFE is due to Chuang [2] which has carried out a Taylor series expansion for both s = 0 and s = ∞. This would, in effect, mean that the expansion begins from the constant term and then from the highest order term. Shieh and Goldman [5] have shown that a mixture of the first and second Cauer forms give good approximations. Bosley and Lees [6] have compared the step responses of Chen and Shieh's reduced model and original system and have found very little error. Shieh et al. have shown that the first, second and third Cauer form formulations for order reduction gives good approximations in the transient, steady state and overall region of the response curve respectively. Chen (1972,1974), Calfe and Healey (1974) have extended the method to the multivariable systems. Wright (1973), Davidson and Lucas (1974) have proposed general expansion in place of original Cauer's expansion.These later results, although sound, have less attractive computational features as mentioned by Parthasarthy and Singh (1975). A potentially attractive method from the view of stability and computational efforts is the mixed Cauer expansion of Shieh and Goldman [5].

The relationship between the inputs and outputs of multiport networks or multivariable control system is often expressed in terms of transfer function matrices. Expansion of transfer function matrices into a matrix continued fraction and the inversion of a matrix continued fraction to a transfer function matrix represents two basic operations in multiport network analysis and synthesis.

One difficulty with the CFE approach is that the stability of the model is not guaranteed, even though the original system is stable Chuang [2] modified the original techniques to have expansions about $s = 0$ and $s = \infty$ alternatively, thereby showing good agreement in both the transient and steady state region.

The very recent published papers are in the direction of model reduction include, reduction of unstable systems [28] and continuous time, time invariant stochastic systems [29].

REDUCTION OF CONTINUOUS TIME LINEAR SYSTEMS BY CONTINUED FRACTION EXPANSION

The model order reduction using CFE approach involves two basic operations viz., expansion and truncation. These operations for various Cauer forms are given below.

3.1 *CAUER FORMS OF CFE*

For the system whose transfer function is in the form of (1.10) can be expanded into the following three different Cauer form representation.

3.1.a. THE CAUER FIRST FORM

$$G(s) = \cfrac{1}{H_1 s + \cfrac{1}{H_2 + \cfrac{1}{H_3 s + \cfrac{1}{H_4 + \cfrac{1}{\ddots}}}}} \qquad \text{.....(3.1a)}$$

3.1.b. THE CAUER SECOND FORM:

$$G(s)=\cfrac{1}{h_1+\cfrac{1}{h_2/s+\cfrac{1}{h_3+\cfrac{1}{h_4/s+\cfrac{1}{\ddots}}}}} \qquad \text{........(3.1b)}$$

3.1.c. THE CAUER THIRD FORM:

$$G(s)=\cfrac{1}{h_1+H_1s+\cfrac{1}{h_2/s+H_2+\cfrac{1}{h_3+H_3s+\cfrac{1}{h_4/s+H_4+\cfrac{1}{\ddots}}}}} \qquad \text{......(3.1c)}$$

Equation (3.1c) is a combination of the Cauer first and second forms in such a way that if we let the h or H in (3.1c) approaches zero, then it will be identical with (3.1a) or (3.1b), respectively. The Cauer second form has been successfully applied by Chen and Shieh [4] to control system design and system identification.

Shieh and Goldman [5] developed the algorithm which is generalized Routh's algorithm for Cauer third form CFE , as follows:

3.2 EXPANSION BY GENERALIZED ROUTH'S ALGORITHM

Performing the long division in equation (1.11) we have

$$G(s) = \left[\frac{1}{\frac{A_{11}}{A_{21}} + \frac{A_{1,n+1}}{A_{2n}} s + \frac{(A_{12} - \frac{A_{11}A_{22}}{A_{21}} - \frac{A_{1,n+1}A_{21}}{A_{2n}})s + (A_{13} - \frac{A_{11}A_{23}}{A_{21}} - \frac{A_{1,n+1}A_{22}}{A_{2n}})s^2 + \ldots + (A_{1n} - \frac{A_{11}A_{2n}}{A_{21}} - \frac{A_{1,n+1}A_{2,n-1}}{A_{2n}})s^{n-1}}{A_{21} + A_{22}s + A_{23}s^2 + \ldots + A_{2n}s^{n-1}}} \right] \qquad \ldots\ldots\ldots(3.2)$$

define

$$h_p = \frac{A_{p,1}}{A_{p+1,1}}, \qquad p = 1,2,3,\ldots,n \qquad \ldots\ldots\ldots(3.3)$$

$$H_p = \frac{A_{p,n+2-p}}{A_{p+1,n+1-p}}, \qquad p = 1,2,3,\ldots,n \qquad \ldots\ldots\ldots(3.4)$$

where $h_p \neq 0$, $H_p \neq 0$, and substitute (3.3) and (3.4) into (3.2) and we have

$$G(s) = \left[\frac{1}{h_1 + H_1 s + \frac{(A_{12} - h_1 A_{22} - H_1 A_{21})s + (A_{13} - h_1 A_{23} - H_1 A_{22})s^2 + \ldots + (A_{1n} - h_1 A_{2n} - H_1 A_{2,n-1})s^{n-1}}{A_{21} + A_{22}s + A_{23}s^2 + \ldots + A_{2n}s^{n-1}}} \right] \qquad \ldots\ldots\ldots(3.5)$$

in which ($A_{12} - h_1 A_{22} - H_1A_{21}$), ($A_{13} - h_1 A_{23} - H_1A_{22}$),....,($A_{1n} - h_1\ {}_{2n} - H_1 A_{2,n-1}$) can be written as A_{31}, A_{32}, ..., $A_{3,n-1}$, respectively. Therefore we have:

$$G(s) = \cfrac{1}{h_1 + H_1 s + \cfrac{A_{31}s + A_{32}s^2 + \ldots\ldots + A_{3,n-1}s^{n-1}}{A_{21} + A_{22}s + \ldots\ldots + A_{2n}s^{n-1}}} \qquad \ldots\ldots\ldots(3.6)$$

Dividing again, we have the expression

$$G(s) = \cfrac{1}{h_1 + H_1 s + \cfrac{1}{h_2/s + H_2 + \cfrac{1}{h_3 + H_3 s + \cfrac{1}{h_4/s + H_4 + \cfrac{1}{\ddots}}}}} \qquad \ldots\ldots(3.7)$$

The quotients in expansion of (3.7) can be obtained by the following generalized Routh algorithm and the modified Routh array.

The coefficients in (1.10) can be expressed by the following double-subscript notation:

$$\begin{array}{lllll} A_{11} & A_{12} & A_{13}\ldots\ldots & A_{1n} & A_{1,n+1} \\ A_{21} & A_{22} & A_{23}\ldots\ldots & A_{2n} & \end{array} \qquad \ldots\ldots\ldots\ldots(3.8)$$

And the elements of third, fourth and subsequent rows can be evaluated from the following algorithm:

$A_{j,k} = A_{j-2,\, k+1} - h_{j-2}\, A_{j-1,\, k+1} - H_{j-2}\, A_{j-1,k}$,

$j=3,4,\ldots\ldots,n+1, \qquad k = 1,2,\ldots\ldots$ ....(3.9)

and

$$h_p = \frac{A_{p,1}}{A_{p+1,1}}, \qquad H_p = \frac{A_{p,n+2-p}}{A_{p+1,n+1-p}}, \qquad p = 1,2,3,\ldots,n \qquad \text{......(3.10)}$$

The complete array is

$A_{11} \quad A_{12} \quad A_{13} \; \ldots\ldots \; A_{1n} \quad A_{1,n+1}$

$h_1 = \frac{A_{11}}{A_{21}}$ $\qquad$ $H_1 = \frac{A_{1,n+1}}{A_{2n}}$

$A_{21} \quad A_{22} \quad A_{23} \; \ldots\ldots \; A_{2n}$

$h_2 = \frac{A_{21}}{A_{31}}$ $\qquad$ $H_2 = \frac{A_{2,n}}{A_{3,n-1}}$

$A_{31} \quad A_{32} \quad .. \quad A_{3,n-1}$

$h_3 = \frac{A_{31}}{A_{41}}$ $\qquad$ $H_3 = \frac{A_{3,n-1}}{A_{4,n-2}}$

$A_{41} \; \ldots\ldots \; A_{4,n-2}$

....

$$A_{n-1,1} \quad A_{n-1,2} \quad A_{n-1,3}$$

$$h_{n-1} = \frac{A_{n-1,1}}{A_{n,1}} \qquad H_{n-1} = \frac{A_{n-1,3}}{A_{n,2}}$$

$$A_{n,1} \quad A_{n,2}$$

$$h_n = \frac{A_{n,1}}{A_{n+1,1}} \qquad H_n = \frac{A_{n,2}}{A_{n+1,1}} \qquad \text{.......(3.11)}$$

$$A_{n+1,1}$$

The triangular pattern in the formulation of (3.11) is called the modified Routh array.

Equation (3.9) is generalized Routh algorithm. If all H are zero, (3.9) is simplified as:

$$A_{j,k} = A_{j-2,\,k+1} - h_{j-2} A_{j-1,\,k+1},$$

$$j = 3,4,\ldots..,n+1, \qquad k = 1,2,\ldots. \qquad \text{........(3.12)}$$

Equation (3.12) is a regular Routh algorithm which is commonly used to obtain the quotients of the Cauer second form. On the other hand, if all h are zero, (3.9) is simplified as:

$$B_{j,k} = B_{j-2,\,k+1} - H_{j-2} B_{j-1,\,k+1},$$

$$j = 3,4,\ldots.., \qquad k = 1,2,\ldots. \qquad \text{........(3.13)}$$

where

$$B_{1,i} = A_{1,\,n+2-i}, \qquad i = 1,2,\ldots,nH$$

and

$$B_{2,j} = A_{2,\,n+1-j}, \qquad j = 1,2,\ldots,n$$

Equation (3.13) is a regular Routh algorithm, which is used to evaluate the quotients of the Cauer first form. Either formulated pattern by the algorithms shown in (3.12) or (3.13) will be a zigzag pattern. It is noted that the elements $A_{j,k}$, $j = 3,4,\ldots$ and $k = 1,2,\ldots$, in (3.12) or (3.13) do not have the same values as those elements of (3.11).

3.3. CAUER MODIFIED FORM

One of the better modification of the original continued fraction expansion is carried out by Chuang (1970) which has carried out a Taylor series expansion for both $s = 0$ and $s = \infty$. This would in effect mean that the expansion begins from the constant and then from the highest-order term.

Shieh and Goldman [5] have shown that a mixture of the first and second form give good approximations for both the transient and steady state responses.

The Cauer modified form is

$$G_m(s) = \cfrac{1}{h_1' + \cfrac{s}{H_1' + \cfrac{1}{h_2' + \cfrac{s}{H_2' + \cfrac{1}{\ddots}}}}} \qquad \ldots\ldots(3.14)$$

The transfer function $G_m(s)$ is expanded into Cauer type CFE about $s = 0$ and $s = \infty$. h_1', h_2' ,……., and H_1',H_2' ,…….., are evaluated by modified Routh array [7].

$$\begin{array}{llllll}
 & a_{11} & a_{12}\ldots\ldots\ldots a_{1,n-1} & a_{1n} & 1 & \\
h_1' = \dfrac{a_{11}}{b_{11}} & & & & & \\
 & b_{11} & b_{12}\ldots\ldots\ldots b_{1,n-1} & b_{1n} & & \\
 & & & & & H_1' = b_{1,n} \\
 & a_{21} & a_{22}\ldots\ldots\ldots a_{2,n-1} & 1 & & \\
h_2' = \dfrac{a_{21}}{b_{21}} & & & & & \\
 & b_{21} & b_{22}\ldots\ldots\ldots b_{2,n-1} & & & \\
 & \ldots\ldots & \ldots\ldots & \ldots\ldots & & H_2' = b_{2,n-1} \\
 & a_{n1} & 1 & & & \\
h_n' = \dfrac{a_{n1}}{b_{n1}} & & & & & \\
 & b_{n1} & & & & \\
 & & & H_n' = b_{n1} & &
\end{array}$$

where

$a_{j+1,k} = a_{j,k+1} - h_j'\, b_{j,k+1}$ $\qquad$ $j = 1,2,\ldots.n - 1$

$b_{j+1,k} = b_{j,k} - H_j'\, a_{j+1,k}$ $\qquad$ $k = 1,2,\ldots.n - j$

and

$$h_j' = \frac{a_{j,1}}{b_{j,1}} , \qquad H_j' = \frac{b_{j,n+1-j}}{a_{j+1,n+1-j}} , \qquad j = 1,2....n$$

3.4 CONTINUED FRACTION INVERSION

If the quotients of a Continued Fraction of the Cauer third form are given, or all h and H are known, what is the corresponding transfer function ? This is the problem of Continued Fraction Inversion.

Derivation of the transfer function, directly from the continued fraction representation, is tedious and involves many multiplication's. It is desirable to have computerized algorithm to implement the continued fraction inversion. Various simple algorithm for developing the entire Routh array and the corresponding transfer function from the known continued fraction coefficients, are being described below.

Method no. 1:

The algorithm was given by Shieh and Goldman [5].

From (3.11) it is noted that

$A_{n,1} = h_n A_{n+1,1}$

$A_{n-1,1} = h_{n-1} A_{n,1} = h_{n-1} h_n A_{n+1,1}$

$A_{n-2,1} = h_{n-2} A_{n-1,1} = h_{n-2} h_{n-1} A_{n+1,1}$

....

$A_{31} = h_3 A_{41} = h_3 h_4 \ldots.. h_n A_{n+1,1}$

$A_{21} = h_2 A_{31} = h_2 h_3 \ldots\ldots h_n A_{n+1,1}$

$A_{11} = h_1 A_{21} = h_1 h_2 \ldots\ldots h_n A_{n+1,1}$ ....(3.14)

and

$A_{n,2} = H_n A_{n+1,1}$

$A_{n-1,3} = H_{n-1} A_{n,2} = H_{n-1} H_n A_{n+1,1}$

....

$A_{2,n} = H_2 A_{3,n-1} = H_2 H_3 \ldots H_n A_{n+1,1}$

$A_{1,n} = H_1 A_{2n} = H_1 H_2 \ldots H_n A_{n+1,1}$ ....(3.15)

Equation (3.14) and (3.15) can be written as the following general equation.

Let $A_{n+1,1} = 1$, then

$A_{j,i} = \prod_{p=j}^{n} h_p, \qquad p = j, j+1, \ldots, n$ ....(3.16)

And

$A_{j,n+2-j} = \prod_{p=j}^{n} H_p, \qquad p = j, j+1, \ldots, n$ ....(3.17)

Where j is the row number in the modified Routh array. The intermediate terms can be evaluated from (3.12), starting from the element in the last row of the modified Routh array and ending up at the elements in the first row. Or if we substitute j = n + 1 and k = 1, then (3.12) yields:

$$A_{n+1,1} = A_{n-1,2} - h_{n-1} A_{n,2} - H_{n-1} A_{n-1} \qquad \text{........(3.18)}$$

Likewise, if we rearrange the order of (3.18), we have

$$A_{n-1,2} = A_{n+1,1} + h_{n-1} A_{n,2} + H_{n-1} A_{n-1}$$

If we perform the same procedures on other elements, we have

$$A_{n-2,2} = A_{n,1} + h_{n-2} A_{n-1,2} + H_{n-2} A_{n-1,1}$$

$$A_{n-2,3} = A_{n,2} + h_{n-2} A_{n-1,3} + H_{n-2} A_{n-1,2}$$

$$\ldots\ldots \quad \ldots\ldots \quad \ldots\ldots \quad \ldots\ldots$$

$$A_{1n} = A_{3,n-1} + h_1 A_{2n} + H_1 A_{2,n-1} \qquad \text{........(3.19)}$$

The general form for (3.19) is

$$A_{j,k} = A_{j+2,k-1} - h_j A_{j+1,k} + H_j A_{j+1,k-1},$$

$$j = n-1, n-2, \ldots, 1, \; k = 2, 3, \ldots, n+1-j \qquad \text{.........(3.20)}$$

Equations (3.17), (3.18), and (3.19) are used to obtain the continued fraction inversion.

Method no. 2:

The algorithm was given by Rao and Lamba [8].

Let the truncated transfer function be represented by

$$G(s) = \cfrac{1}{H_1 + \cfrac{1}{\cfrac{H_2}{s} + \cfrac{1}{H_3 + \cfrac{1}{\ddots + \cfrac{1}{\cfrac{H_{2n}}{s}}}}}}$$

........(3.21)

where H_1 through H_{2n} are called " partial coefficients."

The continued fraction inversion of (3.21) can be represented as:

$$G(s)=\frac{A_{21} + A_{22}s + A_{23}s^2 + A_{24}s^3 + \dots + A_{2,n-1}s^{n-2} + A_{2,n}s^{n-1}}{A_{11} + A_{12}s + A_{13}s^2 + A_{14}s^3 + \dots + A_{1,n}s^{n-1} + s^n} \qquad ......(3.22)$$

The constants A_{11}-$A_{1,n}$ and A_{21}-$A_{2,n}$ are to be evaluated from the partial coefficients H_1-H_{2n}. It is easy to see that for the type of continued fraction form shown in (3.21), the coefficient of s^n in (3.22) is always equal to unity and that the order of the numerator cannot be greater than (n -1).

The first and second rows of Routh table (3.23) are written by copying the coefficients of the denominator and numerator of (3.22), respectively. The subsequent rows are developed by using Routh algorithm.

Routh table:

A_{11}	A_{12}	A_{13}	...	...	$A_{1,n}$	1
A_{21}	A_{22}	A_{23}	...	...	$A_{2,n}$	0
A_{31}	A_{32}	A_{33}	...	...	1	

$$\begin{array}{llllll}
A_{41} & A_{42} & A_{43} & \ldots & \ldots & 0 \\
A_{51} & A_{52} & A_{53} & \ldots & 1 & \\
A_{61} & A_{62} & A_{63} & \ldots & 0 & \\
\ldots & \ldots & \ldots & & \ldots & \\
\ldots & \ldots & \ldots & & & \\
A_{2n-1,1} & 1 & & & & \\
A_{2n,1} & 1 & & & & \\
1 & & & & &
\end{array} \qquad \ldots\ldots\ldots(3.23)$$

If the last two elements of the first rows of the Routh table are as shown then last element of the first column is equal to unity.

That is

$$A_{2n+1,1} = 1. \qquad \ldots\ldots\ldots(3.24)$$

The partial coefficients are related to the first column of Routh table by the following equations.

$$H_p = \frac{A_{p,1}}{A_{p+1,1}}, \qquad p = 2n, 2n-1, \ldots, 1, \qquad H_p \neq 0. \qquad \ldots\ldots\ldots(3.25)$$

Once we know the partial coefficients $H_1 - H_{2n}$, the first column of (3.23) can be evaluated with the help of (3.24) and (3.25), starting from the bottom of the table.

The recursive relationship between the other elements of the Routh table is given by

$$A_{j-2,k+1} = A_{j,k} + \frac{A_{j-2,1}A_{j-1,k+1}}{A_{j-1,1}} \qquad \text{........(3.26)}$$

for

$j = 2n, 2n-1, \ldots, 3$

$k = 1,2, \ldots, n-1$

with

$A_{2,k} = 0$

$A_{3,k} = 0,$ for $k \geq n+1$

$A_{4,k} = 0$

$A_{5,k} = 0,$ for $k \geq n$

....

....

$A_{2n,k} = 0$

$A_{2n+1,k} = 0,$ for $k \geq 2$ ....(3.27a)

And

$A_{1,n+1} = A_{3,n} = A_{5,n-1} = \ldots = A_{2n-1,2} = 1$ ....(3.27b)

With the help of (3.26) and (3.27), all the elements of the Routh table other than first column are evaluated. The transfer function corresponding to the continued fraction expansion (3.21) can be written from the first two rows of the Routh table (3.23).

Method no. 3:

The following algorithm was given by Parthasararathy & H. Singh [9].In method no. 2 Rao and Lamba have proposed a simple algorithm for developing the Routh array and thereby converting a continued fraction into a rational transfer function. This technique is limited to case where the continued fraction is in the Cauer second form. But there are many cases where the continued fraction expansion is available only in the Cauer first form.

This method gives an algorithm for inverting a continued fraction given in the Cauer first form in the light of the method presented by Rao and Lamba[8].

Let the transfer function be represented by equation (3.22), which generates the continued fraction in Cauer first form represented by equation(3.1a), where H_1 to H_{2n} are known as the " partial coefficients."

Without loss of generality the coefficient of s^n in (3.22) can always be taken as unity and numerator is at least one degree lower than the denominator. The problem is to evaluate the constants A_{12} to $A_{1,n+1}$ and A_{21} to $A_{2,n}$. From the knowledge of the partial coefficients H_1 to H_{2n} and construct the transfer function of the form shown in (3.22).

$$\begin{array}{cccccc}
1 & A_{12} & A_{13} \ldots & \ldots & A_{1,n} & A_{1,n+1} \\
A_{21} & A_{22} & A_{23} \ldots & \ldots & A_{2,n} & 0 \\
A_{31} & A_{32} & A_{33} \ldots & \ldots & A_{3,n} & \\
A_{41} & A_{42} & A_{43} \ldots & \ldots & 0 &
\end{array}$$

$$\begin{array}{llll} \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ A_{2n-1,1} & A_{2n-1,2} & & \\ A_{2n,1} & 0 & & \\ A_{2n+1,1} & & & \end{array} \qquad \text{.........(3.28)}$$

It is to be noted that

$$A_{11} = 1 \qquad \text{.........(3.29)}$$

And the end elements of all the odd rows will be the constant equal to $A_{2n+1,1}$ (the last element of the array) and those of the even rows will be zeros.

The authors later [12] pointed out that the generalized algorithm given by Chao K.S. et al. based on a backward expansion of the Routh array, can be applied as well to the inversion of the continued fraction in Cauer first form which is represented by (3.1a).

3.5 ILLUSTRATIVE EXAMPLES

The method of reducing the large scale transfer function by continued fraction expansion can be illustrated by taking examples as below. The algorithm for deducing the reduced order model is explained in the following steps.

Step 1:

Find out the coefficient h_1, h_2, h_3,.....for Cauer second form by the developed computer programme as given in appendix.

Step 2:

Substituting these values of h_1, h_2, h_3,.....in the Cauer second form as given in (3.1b). After performing the algebraic calculation the required reduced order model R(s) is obtained.

This method is illustrated with the help of following example:

Example 3.5.1

The original system [11] is described as:

$$G(s) = \frac{8s^2 + 6s + 2}{s^3 + 4s^2 + 5s + 2}$$

Coefficients for Cauer second form are computed as -

$h_1 = 1$, $h_2 = -2$, $h_3 = 0.5$, $h_4 = 0.22$

With the help of equation (3.1b) the following **second order reduced model** is obtained.

$$R_2(s) = \frac{-1.78s^2 - 0.22s}{s^2 - 0.68s - 0.22s}$$

The negative coefficients in the denominator of $R_2(s)$ shows that the poles fall in the right half of s-plane, which explains that the reduced order system $R_2(s)$ is unstable.

Example 3.5.2:

The original system [14] is described as:

$$G(s) = \frac{28s^3 + 496s^2 + 1800s + 2400}{2s^4 + 36s^3 + 204s^2 + 360s + 240}$$

Coefficients for Cauer second form are computed as -

$h_1 = 0.1$, $h_2 = 13.33333$, $h_3 = -0.695876$, $h_4 = -1.350645$

With the help of equation (3.1b) the following **second order reduced smodel** is obtained.

$$R_2(s) = \frac{11.979355s + 12.528619}{s^2 + 2.1377495s + 1.2528619}$$

The comparison of unit step responses of G(s) and R_2(s) is shown in fig. (3.5.2).

Example 3.5.3:

The original system [15] is described as:

$$G(s) = \frac{8169.13375s^3 + 50664.96749s^2 + 9984.32343s + 500}{100s^4 + 10520s^3 + 52101s^2 + 10105s + 500}$$

Coefficients for Cauer second form are computed as -

$h_1 = 1$, $h_2 = 4.1433$, $h_3 = 0.029912$, $h_4 = 19.039186$

With the help of equation (3.1b) the following **second order reduced model** is obtained.

$$R(s) = \frac{23.182486s + 2.3596099}{s^2 + 23.751986s + 2.3596099}$$

The comparison of unit step responses of G(s) and $R_2(s)$ is shown in fig. (3.5.3.)

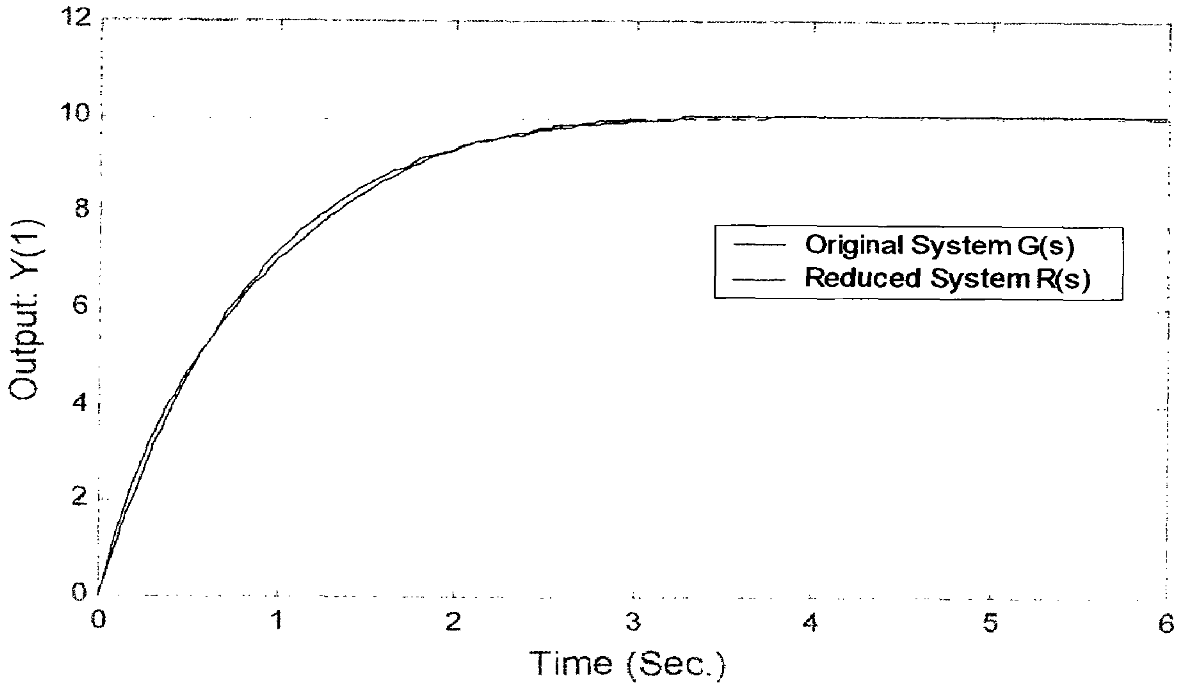

Fig 3.5.2 Comparison of unit step responses

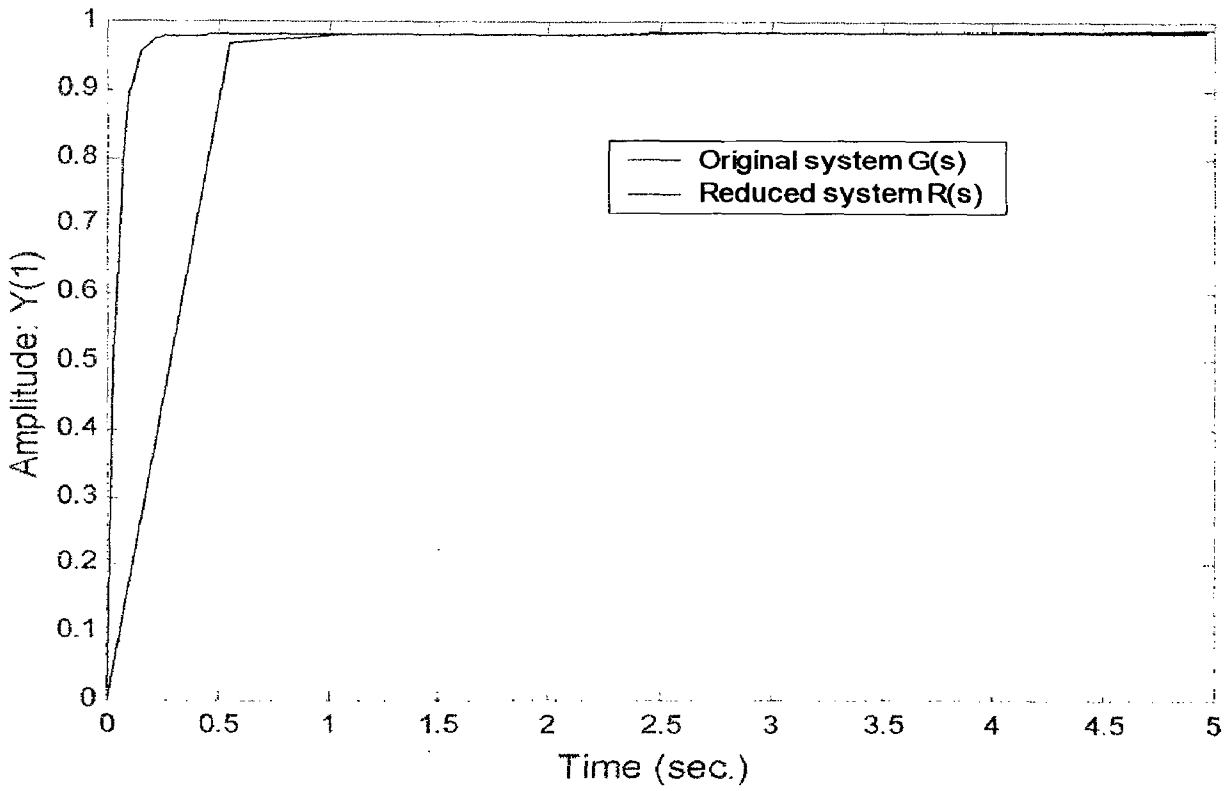

Fig 3.5.3 Comparison of unit step responses

STABILITY BASED REDUCED ORDER MODEL USING CONTINUED FRACTION EXPANSION

4.1 ROUTH APPROXIMATION METHOD

Hutton and Friedland [16] introduced this well-known method for obtaining stable models of asymptotically stable continuous time systems. The Routh approximation is a novel method for reducing the order, based on the idea of truncating the well-known Routh table used to determine stability. The Routh approximants can be computed by a finite recursive algorithm that is suited for programming on a digital computer.

Here the reciprocal transfer function $\hat{G}(s) = \frac{1}{s}G(\frac{1}{s})$ is expanded in the alpha – beta canonical form that is truncated after r terms and reciprocated back to give a desired model of order r.

A higher order SISO system may be described by the nth-order transfer function as:

$$G(s) = \frac{b_1 s^{n-1} + b_2 s^{n-2} + b_3 s^{n-3} + \ldots\ldots + b_n}{a_0 s^n + a_1 s^{n-1} + a_2 s^{n-2} + \ldots\ldots\ldots + a_n} \qquad \ldots\ldots\ldots\ldots(4.1)$$

It is required to reduce the form (4.1) to r th order given by -

$$R_r(s) = \frac{b_1' s^{r-1} + b_2' s^{r-2} + b_3' s^{r-3} + \dots\dots + b_r'}{a_0' s^r + a_1' s^{r-1} + a_2' 2s^{r-2} + \dots\dots\dots + a_r'} \qquad \dots\dots\dots\dots(4.2)$$

Alpha – Beta Expansion

Hutton and Friedland introduced this expansion for a transfer function of the form (4.1) that is asymptotically stable can always be expanded in the following canonical form [13]:

$$\begin{aligned} G(s) &= \beta_1 F_1(s) + \beta_2 F_1(s) F_2(s) + \cdots + \beta_n F_1(s) F_2(s) \cdots F_n(s) \\ &= \sum_{i=1}^{n} \beta_i \prod_{j=1}^{i} F_j(s) \end{aligned} \qquad \dots\dots\dots(4.3)$$

where β_i,(i = 1,2, ...,n) are constants and the F_r (s) for r = 2,3, ...,n are defined by the continued fraction expansions.

$$F_r(s) = \cfrac{1}{\alpha_r s + \cfrac{1}{\alpha_{r+1} s + \cfrac{1}{\alpha_{r+2} s + \cfrac{1}{\ddots \; \alpha_{n-1} s + \cfrac{}{\alpha_n s}}}}} \qquad \dots\dots(4.4)$$

For $F_1(s)$, definition (4.4) is modified slightly, the first term of the continued fraction expansion is $(1+\alpha_1 s)$ instead of $\alpha_1 s$.

$$F_1(s) = \frac{1}{1+\alpha_1 s}$$ ....(4.5)

The canonical form (4.3) is referred to as the alpha–beta expansion of G(s) and plays a fundamental role in the theory of Routh approximations.

The n parameters α_i , (i =1,2 --------,n) appearing in the alpha – beta expansion can be computed by using the classical Routh table in the following fashion:

Alpha (Routh)Table

	$a_0^0 = a_0$ $a_0^1 = a_1$	$a_2^0 = a_2$ $a_2^1 = a_3$	$a_4^0 = a_4$ $a_4^1 = a_5$	$a_6^0 = a_6$..
$\alpha_1 = a_0^0 / a_0^1$	$a_0^2 = a_2^0 - \alpha_1 a_2^1$	$a_2^2 = a_4^0 - \alpha_1 a_4^1$	$a_4^2 = a_6^0 - \alpha_1 a_6^1$	....
$\alpha_2 = a_0^1 / a_0^2$	$a_0^3 = a_2^1 - \alpha_2 a_2^2$	$a_2^3 = a_2^1 - \alpha_2 a_4^2$	....	
$\alpha_3 = a_0^2 / a_0^3$	$a_0^4 = a_2^2 - \alpha_3 a_2^3$	$a_2^4 = a_4^2 - \alpha_3 a_4^3$	......	
$\alpha_4 = a_0^3 / a_0^4$	$a_0^5 = a_2^3 - \alpha_4 a_2^4$	....		
$\alpha_5 = a_0^4 / a_0^5$	....			
$\alpha_6 = a_0^5 / a_0^6$	....			

The first two rows of the table are formed from the coefficients of the denominator of G(s).

The β_i parameters can be similarly obtained using the coefficients of the numerator.

b_j, j =1,2--------,n , as shown in (4.1).

Beta (Routh) Table

	$B_0^1=b_1$ $B_0^2=b_2$	$b_2^1=b_3$ $b_2^2=b_4$	$b_4^1=b_5$ $b_4^2=b_6$	....
$\beta_1 = b_0^1/a_0^1$	$B_0^3=b_2^1-\beta_1 a_2^1$	$b_2^3=b_4^1-\beta_1 a_4^1$	......	
$\beta_2 = b_0^2/a_0^2$	$B_0^4=b_2^2-\beta_2 a_2^2$	$b_2^4=b_4^2-\beta_2 a_4^2$	....	
$\beta_3 = b_0^3/a_0^3$	$B_0^5=b_2^3-\beta_3 a_2^3$	......		
$\beta_4 = b_0^4/a_0^4$	$B_0^6=b_2^4-\beta_4 a_2^4$			
$\beta_5 = b_0^5/a_0^5$	......			
$\beta_6 = b_0^6/a_0^6$	......			

The recursive formula to compute the entries of alpha and beta tables can be obtained from the following :

$$a_0^{i+1} = a_2^{i-1} - \alpha_i a_2^i$$

$$a_2^{i+1} = a_4^{i-1} - \alpha_i a_4^i$$

$$\ldots \quad \ldots \quad \ldots$$

$$a_{n-i-2}^{i+1} = a_{n-i}^{i-1} - \alpha_i a_{n-i}^i \qquad i = 1,2,\ldots\ldots,n-1 \qquad \ldots\ldots. (4.6)$$

for (n -i) odd, the last equation in (4.6) is replaced by

$$a^{i+1}_{n-i-1} = a^{i-1}_{n-i+1} \qquad \text{......... (4.7)}$$

The α_i are marginal entries given by

The coefficients β_i appearing in the canonical form can also obtained by use of a tabular algorithm as shown in beta table. The first two rows of the beta table are obtained from the coefficients of the numerator of G(s).The remaining entries are computed from entries in the Routh table, using the following recursive formula :

$$\beta_i = \frac{b_0^i}{a_0^i} \qquad i = 1,2,\ldots\ldots\ldots\ldots,n \qquad \text{........(4.8)}$$

and

$$b^{i+2}_{j-2} = b^i_j - \beta_i a^i_j \ ,$$

$j = 2,4,\ldots\ldots,n - i$ for n - i even.

$j = 2,4, \ldots\ldots, n-i-1$ for n-i odd

$i = 1,2,\ldots\ldots,n-2.$ ......(4.9)

The r^th Routh reduced model using alpha-beta expansion $R_r(s)$ for the full model G(s) is found by truncating the expansion (4.3) and rearranging the retained terms as a rational transfer function.Truncating the continued fraction (4.4) after the r^th term and denoting it by $g_{j,r}(s)$, the reduced model transfer function $R_r(s)$ is similar to (4.3):

$$R_r(s)=\sum_{i=1}^{r}\beta_i\prod_{j=1}^{i}g_{j,r}(s)$$ ....(4.10)

Where

$$g_{j,r}(s)=\cfrac{1}{\alpha_j s+\cfrac{1}{\alpha_{j+1}s+\cfrac{\ddots}{\alpha_{i-1}s+\cfrac{1}{\alpha_i s}}}+}$$ ....(4.11)

Denote the numerator and denominator of $R_r(s)$ by $N_r(s)$ and $D_r(s)$, respectively,defined below:

$N_1(s) = \beta_1$, $D_1(s) = 1+ \alpha_1 s$.

$N_2(s) = \beta_2 + \alpha_2\beta_1 s$, $D_2(s) = 1+ \alpha_2 s + \alpha_1\alpha_2 s^2$

$N_3(s) = (\beta_1 + \beta_3) + \alpha_3\beta_2 s + \alpha_2\alpha_3\beta_1 s^2$, $D_3(s) = 1 + (\alpha_1+\alpha_3) s +\alpha_2\alpha_3 s^2 +\alpha_1\alpha_2\alpha_3 s^3$

............(4.12)

....

And in general

$N_r(s)=\alpha_r sN_{r-1}(s)+N_{r-2}(s) + \beta_r$ ....(4.13)

$D_r(s)=\alpha_r sD_{r-1}(s)+D_{r-2}(s)$ ....(4.14)

For, r =1,2,.........,and

$N_{-1}(s) = N_0(s) = 0, \qquad D_{-1}(s) = D_0(s) = 1$ ....(4.15)

The relation (4.13) – (4.15) along with the α-β tables are sufficient to find a rth –order reduced model. Hutton and Friedland (1975) mention that this reduced model preserves high-frequency characteristics, and for control application it is preferable to use reciprocal transfer function defined by -

$$\hat{G}(s) = \frac{1}{s} G(\frac{1}{s}) = \frac{b_n s^{n-1} + \ldots\ldots\ldots b_2 s + b_1}{a_n s^n + \ldots\ldots\ldots\ldots a_n s + a_0} \qquad \ldots\ldots\ldots\ldots(4.16)$$

Which, if compared with (4.1), is simply G(s) with a_i, b_j coefficients reversing their orders.

4.2 ROUTH - HURWITZ ARRAY METHOD

A more direct approach to derive the Routh approximants of the transfer function was suggested by Krishnamurthyand Seshadri [17,18]. The proposed technique consists of obtaining the numerator and the denominator polynomial of the reduced order model respectively from the numerator and the denominator polynomial of the system by forming the Routh - Hurwitz stability arrays for the numerator and the denominator polynomial of G (s) = N (s) /D (s) where,

$N(s) = n_{11}\, s^m + n_{21}\, s^{m-1} + n_{12}\, s^{m-2} + n_{22}\, s^{m-3} + \ldots\ldots$ ....(4.17a)

$D(s) = d_{11}\, s^n + n_{21}\, s^{n-1} + n_{12}\, s^{n-2} + n_{22}\, s^{n-3} + \ldots\ldots$ ....(4.17b)

By using the conventional Routh algorithm given by

$$\left.\begin{aligned} n_{ij} &= n_{i-2,j+1} - (n_{i-2,1} * n_{i-1,j+1})/(n_{i-1,1}) \\ d_{ij} &= d_{i-2,j+1} - (d_{i-2,1} * d_{i-1,j+1})/(d_{i-1,1}) \end{aligned}\right\} \quad \text{........(4.18)}$$

for i ≥ 3 and $1 \leq j \leq$ [(m – i +3) / 2 or [(n – i +3) / 2] where [*] stands for the integral part of the quantity and m & n are degrees of the numerator and denominator polynomials respectively. The stability arrays are given below.

Numerator stability Array				Denominator stability Array			
n_{11}	n_{12}	n_{13}	n_{14}	d_{11}	d_{12}	d_{13}	d_{14}
n_{21}	n_{22}	n_{23}	n_{24}	d_{21}	d_{22}	d_{23}	d_{24}
n_{31}	n_{32}	n_{33}		d_{31}	d_{32}	d_{33}	
n_{41}	n_{42}	n_{43}		d_{41}	d_{42}	d_{43}	
....	....	....		....	....	....	
$n_{m,1}$				$d_{n,1}$			
$n_{m+1,1}$				$d_{n+1,1}$			

A reduced order r ≤ n can easily be constructed with (m + 2 - r)th and (m + 3 - r)th rows of numerator stability array and (n + 1 - r)th and (n + 2 - r)th rows of denominator stability and is given by:

$$R(s) = \frac{n_{m+2-r,1}\, s^{r-1} + n_{m+3-r,1}\, s^{r-2} + \ldots\ldots}{d_{n+1-r,1}\, s^{r} + d_{n+2-r,1}\, s^{r-1} + \ldots\ldots\ldots\ldots} \qquad \ldots\ldots(4.19)$$

For r > (m+1), the first two rows of numerator stability array should be used for numerator polynomial while, for r = 1 last two rows should be used.

4.3 MIXED METHOD USING CAUER SECOND FORM OF CONTINUED FRACTION EXPANSION

The major problem of continued fraction expansion technique [1,5] often leading to unstable models, has been solved by many authors [1,19,20]. Shamash [21] has shown that for systems dominated by large magnitude poles, such methods [1,19,20,21] may approximate the poles with small moduli and thus show poor matching during transient period.

A computationally attractive algorithm for SISO systems is developed that overcomes these drawbacks. Comparison of the original system and reduced order system are made through examples. In this method, the denominator polynomial of the reduced model is found by the Routh Approximation method [16,18] and the numerator polynomial by matching the quotients of the second form of continued fraction expansion.

Consider the nth order system transfer function G(s) and its rth order reduced equivalent R(s) be represented as:

$$G(s) = \frac{\sum_{j=1}^{n} a_{2,j} s^{j-1}}{D(s)}$$

$$= \cfrac{1}{h_1 + \cfrac{1}{\cfrac{h_2}{s} + \cfrac{1}{h_3 + \cfrac{1}{\cfrac{h_4}{s} + \ddots}}}} \qquad \text{.......(4.20)}$$

$$R(s) = \frac{\sum_{j=1}^{r} b_{2,j} s^{j-1}}{D_r(s)} \qquad \text{.........(4.21)}$$

where

$$D(s) = \sum_{j=1}^{n+1} a_{1,j} s^{j-1} \qquad \text{........(4.22)}$$

and $D_r(s)$ is the denominator polynomial of degree r. Equation (3.1b) gives the second Cauer CFE of the original system transfer function G(s). The first 2r quotients h_i are retained and truncated continued fraction is inverted to obtain the rth order reduced model $R_r(s)$.

The procedure, as proposed here, is to find denominator polynomial $D_r(s)$ in equation (4.21) using the Routh approximation method discussed at (4.10) and the numerator polynomial $b_{2,,j}$ matching the second Cauer CFE quotients with the $D_r(s)$ quotients. The method consists of the following steps:

Step 1:

The denominator polynomial D(s) is first written with inverted coefficients as $\hat{D}(s)$, form the alpha (Routh) table as given at (4.1.[illegible]) and find the α_I (i = 1,2,3,......,r). The D_r (s) polynomial quotients are determined with the help of equation (4.14).

Step 2:

Evaluate the quotients h_i , (i = 1,2,3, ...r) of second Cauer CFE quotient in equation (4.20) by using the developed computer programme on the basis of Routh algorithm equation (3.12).

Step 3:

Match the quotients h_i with D_r (s) coefficients to determine the numerator coefficients N_r(s).

4.4 ILLUSTRATIVE EXAMPLES

The above given algorithm is illustrated with the help of following examples-

Example 4.4.1

Consider the original transfer function as in example (3.5.1)

Step 1:

For second order reduced model (r = 2), the values of α_1 and α_2 are evaluated as:

α_1 = 0.4

α_2 = 1.3888

Step 2:

With equation (4.14) and above values $D_2(s)$ is evaluated:

$$D_2(s) = s^2 + 1.38885s + 0.5555 \qquad \text{.........(4.23)}$$

Step 3:

The second cauer quotients h_i , i = 1,2 are evaluated and matched with $D_2(s)$ in equation (4.23), to evaluate $N_2(s)$ as:

$h_1 = 1$

$h_2 = -2$

$N_2(s) = 1.111212s + 0.5555$

Hence, **second order reduced model** is written as -

$$R_2(s)= \frac{1.111212s + 0.5555}{s^2 + 1.38885s + 0.5555}$$

Comparison of unit step responses of G(s) and $R_2(s)$ plotted with the help of mat lab. is shown in Fig. (4.4.1). The same model when reduced by Cauer second form CFE in chapter-3 gave unstable model.

Example 4.4.2:

Consider the original transfer function [17]

$$G(s) = \frac{N(s)}{D(s)}$$

N (s) = $35s^7 + 1086s^6 + 13285s^5 + 80402s^4 + 23837s^3 + 511812s^2 + 482964s + 194480$

D(s) = $s^8 + 33s^7 + 437s^6 + 3017s^5 + 11870s^4 + 27470s^3 + 37492s^2 + 28880s + 9600$

Step 1:

For third order reduced model (r = 3), therefore, values of α_1 and α_2 and α_3 are evaluated as -

α_1 = 0.332410

α_2 = 1.018311

α_3 = 1.728900

Step 2:

With equation (4.14) and above values, $D_3(s)$ is evaluated -

$$D_3(s) = s^3 + 2.06131s^2 + 1.7605579s + 0.585227 \qquad \text{.........(4.24)}$$

Step 3:

The second cauer quotients h_i , i = 1,2,3 are evaluated and matched coefficients of $D_3(s)$ equation (4.24), to evaluate $N_3(s)$ as -

$h_1 = 0.049362$

$h_2 = 38.589317$

$h_3 = 0.453857$

$N_3(s) = 26.657933s^2 + 29.442228s + 11.85582$

Hence, **third order reduced model** is written as -

$$R_2(s) = \frac{26.657933s^2 + 29.442228s + 11.85582}{s^3 + 2.06131s^2 + 1.7605579s + 0.585227}$$

Comparison of unit step responses of G(s) and $R_3(s)$ plotted with the help of mat lab. is shown in figure (4.4.2).

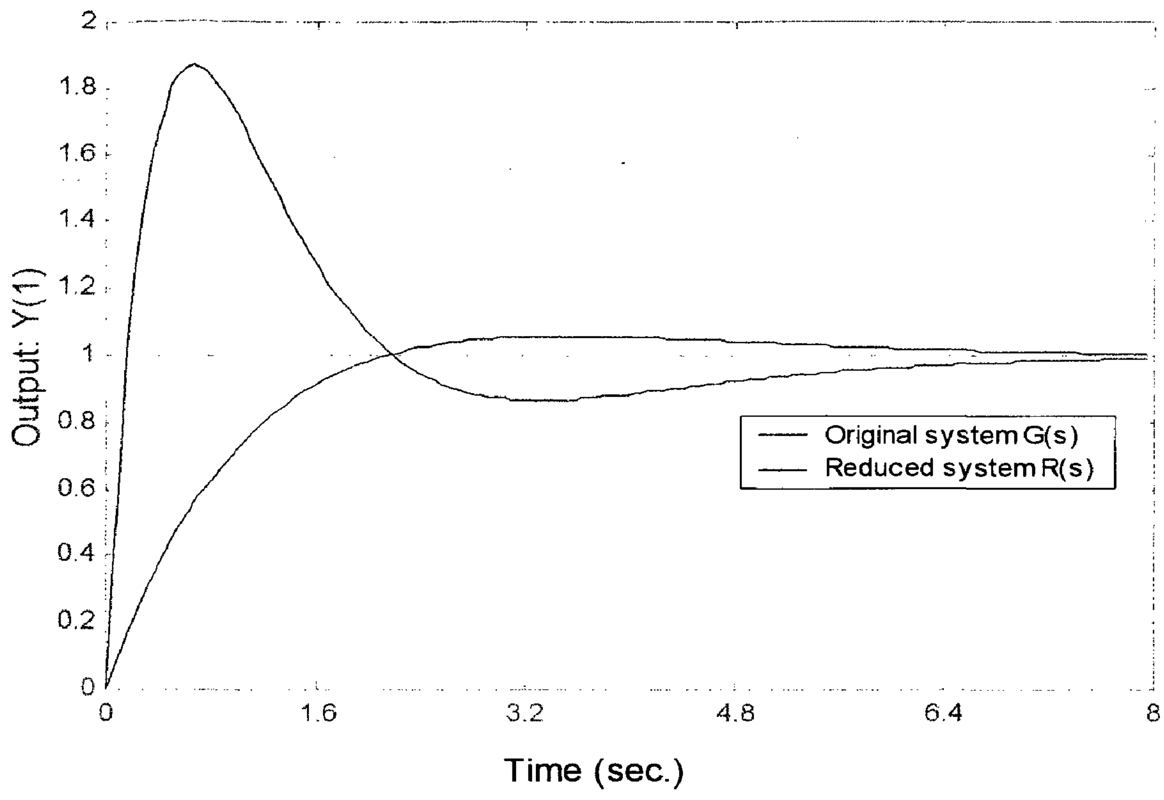

Fig 4.4.1 Comparison of unit step responses

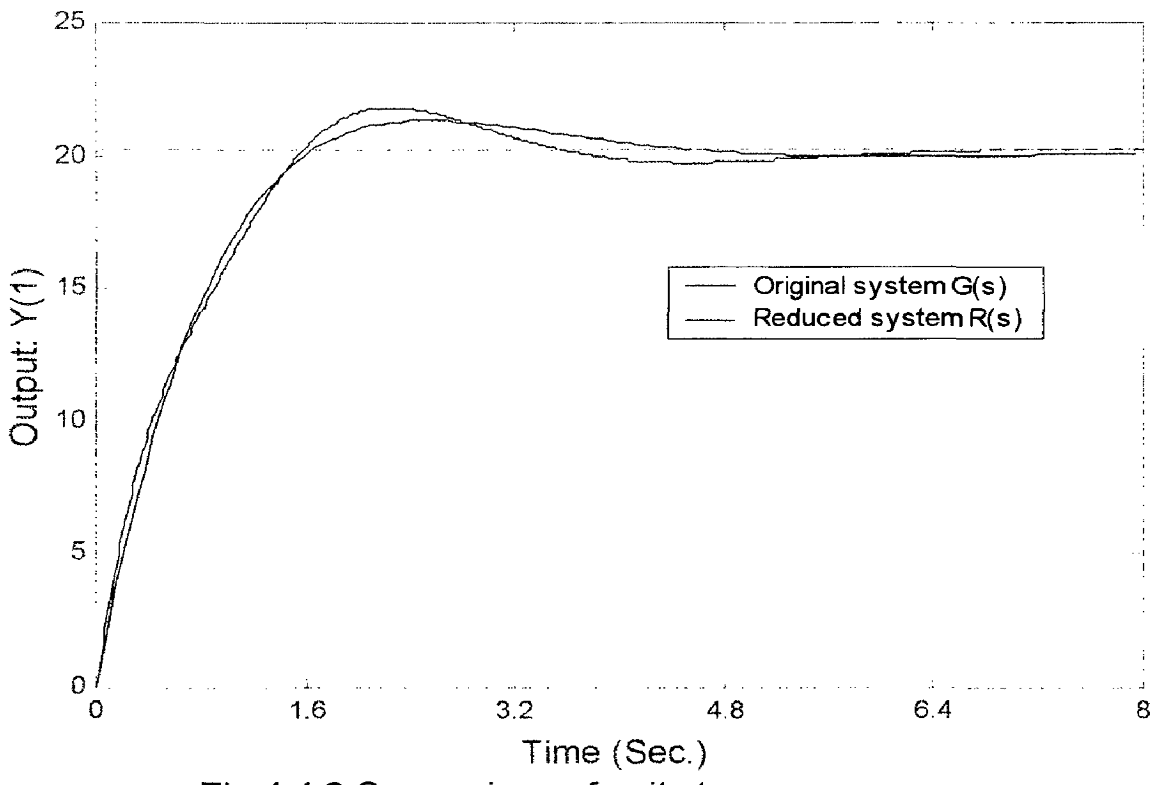

Fig 4.4.2 Comparison of unit step responses

Chapter-5

REDUCTION OF DISCRETE TIME SYSTEMS USING CONTINUED FRACTION EXPANSION

5.1 *INTRODUCTION*

For discrete time system also the same arguments as for continuous systems hold as far as the need for reduced order modeling is concerned. Moreover, the fast development and usage of small digital computers and the processor in the design and implementation of control system have increased the importance of reduced order modeling methods for discrete systems. In this chapter the CFE technique for continuous systems are extended to the discrete time systems.

The use of bilinear transformation [25] plays an important role to extend CFE methods for continuous time systems to reduce discrete time systems i.e., z-transfer function in the ' w' domain.

5.2 *MATHEMATICAL FORMULATION*

Let the nth order discrete time stable SISO system $G_o(z) = N_o(z) / D_o(z)$ and its rth order model $R_r(z) = N_r(z) / D_r(z)$ be

given by:

$$\left.\begin{aligned} N_0(z) &= a_0 + a_1 z + a_2 z^2 + \dots\dots + a_{n-1} z^{n-1} \\ D_0(z) &= b_0 + b_1 z + b_2 z^2 + \dots\dots + b_n z^n \end{aligned}\right\} \qquad \dots\dots\dots(5.\ 1)$$

$r < n$

$$\left.\begin{aligned} N_r(z) &= a_0^* + a_1^* z + a_2^* z^2 + ...a_{r-1}^* z^{r-1} \\ D_r(z) &= b_0^* + b_1^* z + b_2^* z^2 + ...b_r^* z^r \end{aligned}\right\} \qquad ....(5.2)$$

The direct substitution of the bilinear transformation [25],

z = (1 + w)/(1 – w) in G_0 (z) gives G_0 (w) in w-domain which can be simplified as:

$$G_0(z) = \Big|_{z=(1+w)/(1-w)} = G_0(w) = k + G_1(w) \qquad ....(5.3)$$

where $k = G_0(w)\Big|_{w \to \infty}$ ....(5.4)

and

$$G_1(w) = \frac{f_0 + f_1 w + f_2 w^2 + ... + f_{n-1} w^{n-1}}{g_0 + g_1 w + g_2 w^2 + ...g_n w^n} \qquad ......(5.5)$$

G_1 (w) can be simplified by continued fraction expansion method. This gives an rth order reduced model R_1 (w) in w-domain of G_1(w), which can be written as:

$$R_r(w) = k + R_{1r}(w) \qquad ....(5.6)$$

R_r (w) is then transformed back to the 'z' domain by using the reverse bilinear transformation w = (z – 1)/(z +1) as:

$$R_r(z) \;=\; R_r(w)\Big|_{w\;=\;(z-1)/(z+1)} \qquad\qquad ....(5.7)$$

The reduced order model obtained by the above steps gives a stable model if, the original system is stable. But the initial value of the step response of reduced order model will not be zero even if the initial value of the original system is zero. To overcome this drawback the numerator and denominator polynomials of the original system are expressed as a function of the bilinear transformed variable 'w', i.e. by writing N_0 (z) & D_0(z) as:

$$\left.\begin{aligned} N_0(z)\Big|_{z=(1+w)\;/(1-w)} &= N(w)/(1-w)^{\,n-1} = 0 \\ D_0(z)\Big|_{z=(1+w)\;/(1-w)} &= D(w)/(1-w)^{\,n\ 1} = 0 \end{aligned}\right\} \qquad ....(5.8)$$

Which gives

$$G(w) = \frac{N(w)}{D(w)} = \frac{c_0 + c_1 w + c_2 w^2 + ... + c_{n-1} w^{n-1}}{d_0 + d_1 w + d_2 w^2 + ... + d_n w^n} \qquad ....(5.9)$$

In this case the rank of G(w) remains the same as that of G_0 (z)and

$$G(w)\Big|_{w\to\infty} = k \;=\; 0 \qquad ....(5.10)$$

Hence the step response of G_0(z) and reduced order model will match at t = 0. G(w) can be simplified by continuous time methods to give an r^{th} order model R_r (w) which by using the reverse bilinear transformation separately in the numerator and denominator may be converted in the 'z' domain as in equation (5.8).

$$\left.\begin{aligned} N_r(w)\Big|_{w=(z-1)/(z+1)} &= N_r(z)/(z+1)^{r-1} = 0 \\ D_r(w)\Big|_{w=(z-1)/(z+1)} &= D_r(z)/(z+1)^{r} \quad = 0 \end{aligned}\right\} \qquad \text{.......(5.11)}$$

The above scheme is used in developing the following method for the reduction of discrete time SISO systems.

5.3 *MIXED METHOD USING ROUTH APPROXIMATION*

The complete scheme of the method is carried out in the following steps:

Step 1:

Convert G(z) to G(w) by using bilinear transformation as in (5.8).

Step 2:

Compute second Cauer form quotients CFE, h_i (i=1,2,...r) of D(w) by developed computer programme given in appendix.

Step 3:

Compute α_i , i =1,2,....r, of D(w) by alpha (Routh) table as given in chapter-4 then $D_r(w)$ is obtained with equation (4.14).

Step 4:

Obtain $N_r(w)$ by matching the quotients h_i with the coefficients of $D_r(w)$.

Step 5:

Obtain $R_r(z)$ by applying the reverse bilinear transformation in R(w).

Step 6:

To remove any steady state error between the original system and model output (for a unit step input) multiply the numerator of the reduced order model with gain correction factor k_g obtained as:

$$k_g = \frac{G(z)}{R(z)} \bigg|_{z=1} \qquad \text{.......(5.12)}$$

5.4 *ILLUSTRATIVE EXAMPLES*

The above given algorithm is illustrated with the help of following examples-

Example 5.4.1:

Consider the 8th order system [26]:

$G(z) = N(z) / D(z)$

$N(z) = 1.682z^7 + 1.116z^6 - 0.21z^5 + 0.152z^4 - 0.516z^3 - 0.262z^2 + 0.044z - 0.006$

$D(z) = 8z^8 - 5.046z^7 - 3.348z^6 + 0.63z^5 - 0.456z^4 + 1.548z^3 + 0.786z^2$
$0.132z + 0.018$

Step 1:

By applying the bilinear transformation, G(w) is obtained as:

$G(w) = N(w)/D(w)$, where

$N(w) = -1.525879e\text{-}05w^7 + 4.879944w^6 + 27.856186w^5 + 54.127853w^4 + 62.176041w^3 + 46.255997w^2 + 18w + 2$

$D(w) = 8w^8 + 78.64w^7 + 292.928w^6 + 526.818w^5 + 584.144w^4 + 400.24w^3$
$+ 139.232w^2 + 16w + 2$

Step 2:

Second Cauer quotients h_i, for r=2 of G(w) are evaluated as -

$h_1 = 1$

$h_2 = -1$

Step 3:

From D(w) of G(w), compute α_i (for r=2) as -

$\alpha_1 = 0.125$

$\alpha_2 = 0.1793681$

Compute D_2 (w) by equation (4.14).

$D_2(w) = W^2 + 0.1793681w + 0.022421$

Step 4:

Match h_i with coefficients of D_2 (w), to obtain N_2 (w).

$N_2(w) = 0.201789w + 0.022421$

Express, $R_2(w) = \dfrac{N_2(w)}{D_2(w)} = \dfrac{0.201789w + 0.022421}{w^2 + 0.1793681w + 0.022421}$

Step 5:

Convert $R_2(w)$ in $R_2(z)$ by reverse bilinear transformation.

$$R_2(z) = \frac{N_2(z)}{D_2(z)} = \frac{0.22421z - 0.179368}{1.2017891z^2 - 1.955158z + 0.8430529}$$

Step 6:

Compute the correction factor k_g by equation (5.12) and multiply in $N_2(z)$.

We have, $k_g = 2$

The corrected $N_2(z)$ is obtained as -

$N_2(z) = 0.44842z - 0.358736$

The **final reduced order model** is expressed as -

$$R_2(z) = \frac{0.44842z - 0.358736}{1.2017891z^2 - 1.955158z + 0.8430529}$$

The unit step responses of $G(z)$ and $R_2(z)$ is compared as shown in figure (5.4.1)

Example 5.4.2

Consider the 5th order system [27]:

$G(z) = N(z) / D(z)$,where

$N(z) = 3z^4 - 8.886z^3 + 10.0221z^2 - 5.091975z + 0.981112$

$D(z) = z^5 - 3.7z^4 + 5.47z^3 - 4.037z^2 + 1.4856z - 0.2173$

Step 1:

By applying the bilinear transformation G(w) is obtained as:

$G(w) = N(w)/D(w)$,where

$N(w) = 27.981188w^4 + 15.663599w^3 + 3.842474w^2 + 0.487501w + 0.025238$

$D(w) = 15.909901w^5 + 11.989698w^4 + 3.5302w^3 + 0.532201w^2 + 0.0367w + 0.0013$

Step 2:

Compute second Cauer quotients h_i for (r = 2) of G(w) as -

$h_1 = 0.05151$

$h_2 = 2.177754$

Step 3:

From D(w) of G(w), compute α_i (for r=2) as -

$\alpha_1 = 0.035422$

$\alpha_2 = 0.090138$

Compute $D_2(w)$ by equation (4.15).

$D_2(w) = W^2 + 0.090138w + 0.00319286$

Step 4:

Match h_i with coefficients of D_2 (w), to obtain N_2 (w).

$N_2(w) = 1.1973514w + 0.061984$

Express, $R_2(w) = \dfrac{N_2(w)}{D_2(w)} = \dfrac{1.1973514w + 0.061984}{w^2 + 0.090138w + 0.00319286}$

Step 5:

Convert $R_2(w)$ in $R_2(z)$ by reverse bilinear transformation.

$$R_2(z) = \frac{N_2(z)}{D_2(z)} = \frac{1.2593354z - 1.1353674}{1.09333308z^2 - 1.9936144z + 0.9130548}$$

Step 6:

Compute the correction factor k_g by equation (5.12) and multiply in $N_2(z)$.

We have, $k_g = 1.9999$

The corrected $N_2(z)$ is obtained as -

$N_2(z) = 2.5186424z - 2.2707092$

The **final reduced order model** is expressed as -

$$R_2(z) = \frac{2.5186424z - 2.2707092}{1.0933308z^2 - 1.9936144z + 0.9130548}$$

The unit step responses between G(Z) and R_2(Z) is as shown in Fig. (5.4.2).

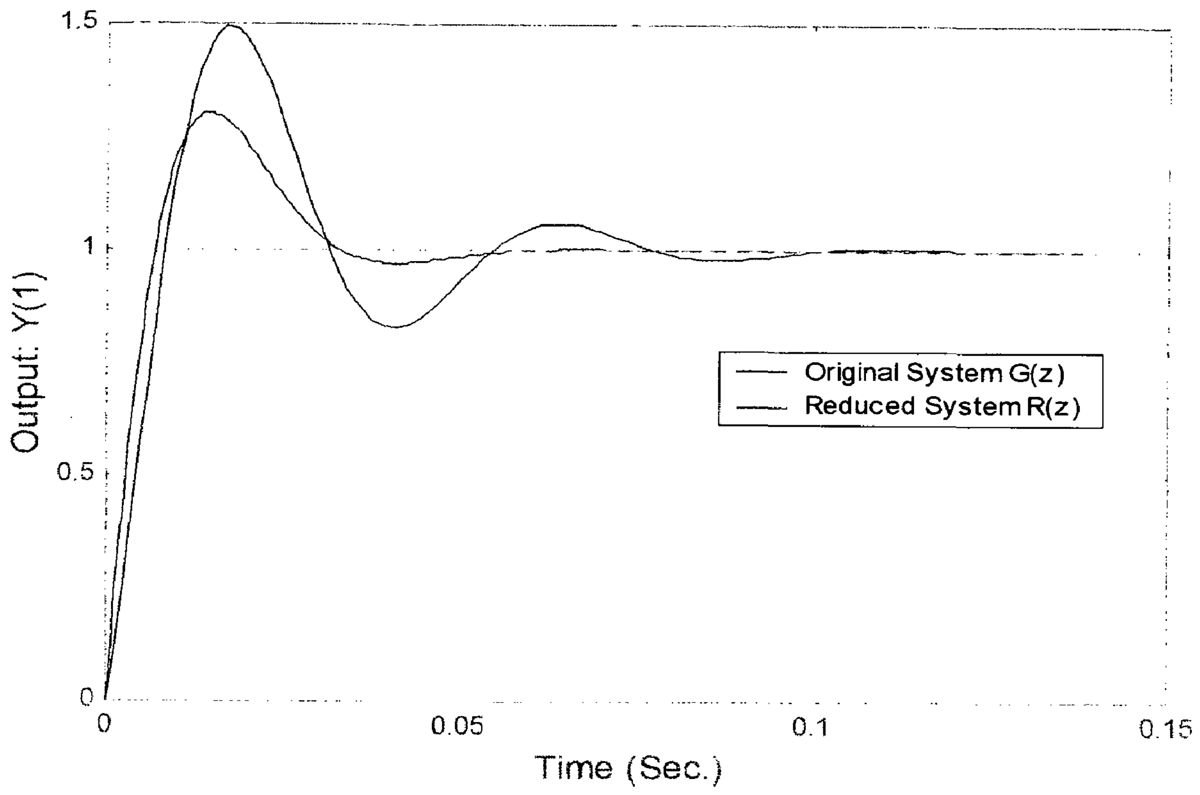

Fig 5.4.1 Comparison of unit step responses

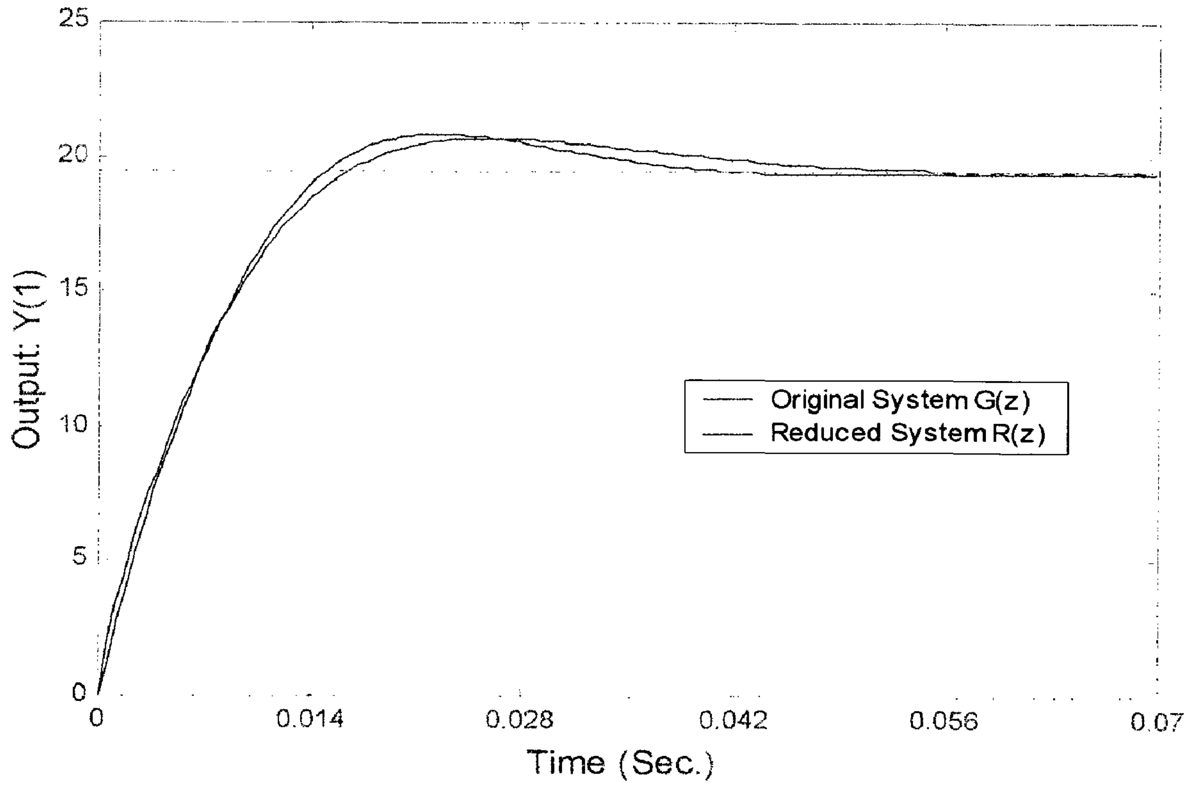

Fig 5.4.2 Comparison of unit step responses

Chapter -6

CONCLUSIONS AND FUTURE SCOPE OF WORK

Reduction of linear dynamic systems has been an active area of research during the last few decades. The work included here in deals with the reduction of linear dynamic systems using continued fraction expansions. This concluding chapter is primarily devoted to summarising the main contributions of the work done and scope of future work in this field.

The continued fraction expansion approach is an algebraic method for deriving the reduced order models using various Cauer forms. The method is quite attractive and simple but it has a serious drawback of producing unstable (stable) models for stable (unstable) system, this has been shown in chapter-3.To overcome this drawback a mixed method has been proposed in chapter - 4 which has been extended for discrete time systems in chapter-5. The mixed method utilizes the advantages of continued fraction expansion, bilinear transformation and Routh approximation. The unique feature of this method is that it gives stable models for stable systems and matches the initial and final values of the responses. The methods included here are illustrated for SISO systems only however, these are extendable for multiinput - multioutput (MIMO) systems.

The future scope of the work include extension of these methods for MIMO systems. The other possibility may be developing new mixed methods using Cauer first and third forms of CFE coupled with other stability based reduction methods. These methods can also be tried for the design of controller.

REFERENCES

1. **Chen C.F.** and **Shieh L.S.**, "A novel approach to linear model simplification", Int. J. Control Vol. 8, No. 6, pp 561-570, 1968. Also available in Proc. Joint Joint Automatic Control Conference, Michigan, pp 454-461, 1968.

2. **Chaung S.C,** "Application of Continued Fraction for Modeling transfer functions to give More accurate Initial transient response", Electronic Letters, Vol. 6, No. 6, pp. 861-863, 1970.

3. **Shieh L.S. Chen C.F**. and **Huang C.J.**, " Simple methods for Identifying linear systems for frequency or time response data", Int. J. Control, Vol. 13, pp 1027-1039, 1971.

4. **Chen C.F.** and **Shieh L.S.**, "An algebraic method for control system design," Int. J. Control, Vol. 11, No. 5, pp 717-739, 1970.

5. **Shieh L.S., Goldman M.J.,** "Continued fraction expansion and inversion of the Cauer third form", IEEE Transactions on circuits and systems, Vol. CAS-21, No. 3, May 1974.

6. **Bosley M.J.** and **Lees F.P.**, " A survey of simple transfer function derivations from high order state variable models", Automatic Control, Vol. 8, pp 765-775, 1972.

7. **Parthasarthy R., et. al.**" On model reduction by modified Cauer form", IEEE Transaction on Automatic Control Vol. AC-28, No. 4, pp 523-527, April 1983.

8. **Rao Vittal S., Lamba S.S.,** "A note on Continued fraction inversion by Routh's Algorithm", Technical notes and Correspondence, IEEE transaction on Automatic control, June 1974.

9. **Parthasarthy R., Singh H.,** " On continued fraction inversion by Routh's Algorithm", IEEE Trans. on Automatic Control, Vol. AC-20, pp. 278-279, April 1975.

10. **Wall H.S.,** " Analytical theory of continued fractions", Princeton N.J.: Van Nostrand, 1948.

11. **Prasad R.**, "Analysis and design of control systems using reduced order models", Ph. D. Thesis, Deptt. of Elect. Engg.,U.O.R., Roorkee, Feb. 1989.

12. **Partarsarthy R., Singh H.,** "On continued fraction Inversion by Routh's algorithm", Tech. Notes and correspondence, IEEE. Trans. On Auto. Control, June 1976.

13. **Jamshidi M.,** "Large-Scale Systems Modeling and Control Series", Vol. 9, North Holland, New York, Amsterdam, Oxford, 1983.

14. **Shamash Y.**, " Model reduction using the Routh stability criterion and the Pade approximation technique", Int. J. Control, Vol. 21, No. 3, pp 475-484, 1975.

15. **Shamash Y.**, "Stable biased reduced order models using the Routh method of reduction", Int. J. control, Vol. 11 No. 5, pp 641-654, 1980.

16. **Hutton M.F., Friedland B.,** " Routh approximants for reducing order of linear time invariant systems", IEEE Trans. Automatic control , Vol. Ac-20, pp 329-337.1975.

17. **Krishnamurthy V., Sheshadri V.,** "A simple and direct method of reducing the order of linear time invariant systems by Routh approximation in the frequency domain", IEEE Trans. Automatic Control, Vol. Ac-21, No. 5, pp 797-799, Oct 1976.

18. **Krishnamurthy V., Sheshadri V.,** "Model reduction using Routh stability criterion", IEEE Trans. Automatic control, Vol. AC-23, No. 4, pp 729-731, Aug. 1978.

19. **Shieh L.S.** and **Goldman M.J.,** " A mixed form for linear system reduction", IEEE Trans. Syst. Man. Cyber., Vol. SMC-4, No. 6, pp 584-588, Nov. 1974.

20. **Shieh L.S.** and **Y.J. Wei**, " A mixed method for multivariable system reduction", IEEE Trans. Automatic Contr., Vol. AC-20, No. 3, pp 429-432, June 1975.

21. **J. Pal**, " System reduction by a mixed method", ibid, Vol. AC-25, No. pp 973-976, 1980.

22. **Chen T.C., Chang C.Y.,** and **Han K.W.,** "Model using the stability equation method and the continued fraction method", Int. J. Contr., Vol. 32, No. 1, pp 81-94, 1980.

23. **Parthasarthy R.** and **Jaysimha K.N.,** "System reduction using stability equation method and modified Cauer continued fraction", Proc. IEEE, Vol. 70, No. 10, pp 1234-1236, Oct. 1982.

24. **Shamash Y.,** "Truncation method of reduction: A viable alternative", Electron Lett., Vol. 17, No. 2,pp 97-99, Jan 1981.

25. **Davies A.C.**, "Bilinear transformation of polynomials", IEEE transactions on circuits and systems, Nov. 1974.

26. **Bistritz Y.**, "A discrete Routh stability method for discrete system modeling", System and control letters, vol.2, no. 2 pp. 83-87, 1982.

27. **Farsi M.**, **et. al.**, "Stable reduced order models for discrete time systems", Proc. IEEE,vol. 133, Pt. D No. 3, pp.137- 140, May 1986.

28. **Yang J., et. al**. "Model reduction of unstable systems", Int. J. systems science, vol. 24, no. 12, 2407- 2414,1993.

29. **Wang Zidong, Unbehaucn H.**, "Model reduction based on regional pole and covariance equivalent realizations", IEEE trans. On automat. Control, vol. 44, no. 10, Oct. 1999.

30. **Mahmoud M.S., Singh M.G.,** "Large Scale Systems Modelng", vol. 3,Pergamon Press, England, First edition 1981.

```
//*****************************************************************************************//
//********************************* SOURCE CODE FOR ****************************//
//*************************** BILINEAR TRANSFORMATION ***********************//
//*****************************************************************************************//

#include<iostream.h>
#include<fstream.h>
#include<conio.h>
#include<math.h>
#include<stdio.h>
void main(void)
{
        int i,j,k,n,l;
        float a[20],b[20];
        char* str;
        cout<<"\n Enter output file for results\n";
        cin>>str;
        ofstream outfile(str,ios::out);
        cout<<"\n BILINEAR TRANSFORMATION OF POLYNOMIALS";
        cout<<"\n ENTER THE POLYNOMIAL DEGREE n: ";
        cin>>n;
        cout<<"\n ENTER THE COEFFICIENT IN DECREASING ORDER \n";
        outfile<<"\n THE COEFFICIENT IN DECREASING ORDER \n";

        l=n+1;
        for(j=1;j<=l;j++)
        {
                cout<<"\n a["<<l-j<<"] = ";
        cin>>a[j];
        outfile<<"\n a["<<l-j<<"] = "<<a[j]<<endl;
}
b[1] = a[1];
for(j=1;j<=n;j++)
{
        k=l+1-j;

        for(i=2;i<=k;i++)
        {
                b[i]=a[i]+b[i-1];
        }

        for(i=2;i<=k;i++)
        {
                a[i]=b[i];
        }
}
cout<<"\n  TRANSFORMED POLYNOMIAL ";
outfile<<"\n  TRANSFORMED POLYNOMIAL ";
cout<<"\n HIGHEST POWER COEFFICIENT FIRST";
outfile<<"\n HIGHEST POWER COEFFICIENT FIRST";
for(j=1;j<=l;j++)
{
        i=l+1-j;
        b[j]=a[i];
}
for(j=2;j<=l;j++)
{
        a[j]=b[j]*pow(2,(j-1));
}
```

```
        for(j=1;j<=n;j++)
        {
                k=l+1-j;
                for(i=2;i<=k;i++)
                {
                        b[i] = a[i]-b[i-1];
                }
                for(i=2;i<=k;i++)
                {
                        a[i]=b[i];
                }
        }
        cout<<"\n";
        outfile<<"\n";
        for(j=l;j>0;j--)
        {
                cout<<"\t"<<b[j];
                outfile<<"\t"<<b[j];
        }
        getch();
}
```

```
//*****************************************************************************************//
//***************************** SOURCE CODE FOR *********************************//
//*************CONTINUED CAUER FRACTION EXPANSION *******************//
//*****************************************************************************************//

#include<iostream.h>
#include<fstream.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main()
 {
        float a[20][20],h[20],H[20];
        int i,j,k,n,r;
        ofstream outfile("output.dat",ios::out);
        clrscr();
        cout<<"Enter the order of the transfer function n:";
        cin>>n;
        cout<<"Enter the order of the reduced order system r:";
        cin>>r;
        cout<<"Enter the numerator coefficients:"<<endl;
        outfile<<"The numerator coefficients:"<<endl;
        for(i=0;i<=n-1;i++)
        {
                cout<<"a2["<<i<<"] = ";
                cin>>a[2][i+1];
                outfile<<"a2["<<i<<"] = "<<a[2][i+1]<<endl;
        }
        cout<<"Enter the denominator coefficients:"<<endl;
        outfile<<"The denominator coefficients:"<<endl;
```

```
for(i=0;i<=n;i++)
{
          cout<<"a1["<<i<<"] = ";
         cin>>a[1][i+1];
         outfile<<"a1["<<i<<"] = "<<a[1][i+1]<<endl;
}
h[1] = a[1][1]/a[2][1];
H[1] = a[1][n+1]/a[2][n];

for(j=3;j<=n+1;j++)  //n+1
{
         for(k=1;k<=n+2-j;k++)
         {
                  a[j][k] = a[j-2][k+1] - H[j-2]*a[j-1][k];
         }
         H[j-1] = a[j-1][n+3-j]/a[j][n+2-j];
}

clrscr();
cout<<"\t"<<"Coefficients for the Cauer First Form are as follows:"<<endl;
outfile<<"\t"<<"Coefficients for the Cauer First Form are as
             follows:"<<endl;
for(i=1;i<=n;i++)
 {
          cout<<"\t\t"<<"H["<<i<<"] = "<<H[i]<<endl;
         outfile<<"\t\t"<<"H["<<i<<"] = "<<H[i]<<endl;
 }

cout<<"\t"<<"Coefficients for the Cauer Third Form are as follows:"<<endl;
outfile<<"\t"<<"Coefficients for the Cauer Third Form are as
             follows:"<<endl;
```

```
for(j=3;j<=n+1;j++)
{
        for(k=1;k<=n+2-j;k++)
        {
                a[j][k] = (a[j-2][k+1]) - (h[j-2]*a[j-1][k+1]) - (H[j-2]*a[j-1][k]);
        }
        h[j-1] = a[j-1][1]/a[j][1];
        H[j-1] = a[j-1][n+3-j]/a[j][n+2-j];
}

for(i=1;i<=n;i++)
{
        cout<<"\t\t"<<"h["<<i<<"] = "<<h[i]<<"\t"<<"H["<<i<<"] =
              "<<H[i]<<endl;
        outfile<<"\t\t"<<"h["<<i<<"] = "<<h[i]<<"\t"<<"H["<<i<<"] =
              "<<H[i]<<endl;
}

for(j=3;j<=2*n+1;j++)
{
        for(k=1;k<=n;k++)
        {
                a[j][k] = ((a[j-1][1] * a[j-2][k+1]) - (a[j-2][1] * a[j-1][k+1]))
                           /a[j-1][1];
        }
        if(j%2==0)
        a[j][n+2-j/2] = 0;
}
cout<<"\t"<<"Coefficients for the Cauer Second Form are as
      follows:"<<endl;
```

```
        outfile<<"\t"<<"Coefficients for the Cauer Second Form are as
                follows:"<<endl;
         for(j=1;j<=2*r;j++)
         {
                h[j] = a[j][1]/a[j+1][1];
                cout<<"\t"<<"\t"<<"h["<<j<<"] = "<<h[j]<<endl;
                outfile<<"\t\t"<<"h["<<j<<"] = "<<h[j]<<endl;
         }
        outfile.close();
        getch();
}
```

```
//*****************************************************************************************//
//********************************* SOURCE CODE FOR ****************************//
//*********ROUTH APPROXIMATION FOR ALPHA COEFFICIENTS ***********//
//****************************************************************************************//

#include<stdio.h>
#include<iostream.h>
#include<conio.h>
#include<math.h>
#include<ctype.h>
#include<dos.h>
#include<graphics.h>
#define M 10
#include<stdlib.h>
void routh();

FILE *fp1;
void main()
{
        char c,t;
        int i;
        int gdriver=DETECT,gmode;
        fp1=fopen("output.dat","w");
        textbackground(3);
        textcolor(BLACK);
        initgraph(&gdriver,&gmode,"c:\\tc\\bgi");

        clrscr();
        routh();
        setgraphmode(gmode);
        cleardevice();
        setbkcolor(7);

}


typedef struct
{

        int n;
        double x[M];

}poli;

void tini_1_dim(double a[M]);
void tini_2_dim(double a[M][M]);

void routh()
{
        int i,j,n;

        double A[M],B[M],a[M][M],b[M][M],alpha[M],beta[M];
        tini_1_dim(A);
        tini_1_dim(B);
        tini_1_dim(alpha);
        tini_1_dim(beta);
        tini_2_dim(a);
        tini_2_dim(b);
        clrscr();

        cout<<"\n\n\t\tENTER ORDER OF DENOMINATOR : ";
        cin>>n;
```

```
cout<<"\n\t\tEnter Coefficients  of Denominator in Decending Powers of s";
cout<<endl;
for(i=0;i<=n;i++)
{
        cout<<"\t\t";
        cin>>B[i];
}

/*cout<<"\n\t\tEnter Coefficients of Numerator in Decending Powers of s";
cout<<endl;
for(i=n;i>=1;i--)
{
        cout<<"\t\t";
        cin>>A[i];
}   */

clrscr();
int k;
i=0;
for(j=0;j<=n;j+=2)
{
        k=n-j;
        if(k<0)
                b[i][j]=0;
        else
                b[i][j]=B[k];
}
++i;
for(j=0;j<=n;j+=2)
{
         k=n-j-1;
```

```
          if(k<0)
                b[i][j]=0;
           else
                 b[i][j]=B[k];
    }
    for(i=1;i<n;i++)
    {
         alpha[i]=b[i-1][0]/b[i][0];
         for(j=0;j<n;j+=2)
         {
              b[i+1][j]=b[i-1][j+2]-alpha[i]*b[i][j+2];
         }
    }
    b[n][0]=1;
    alpha[n]=b[n-1][0];
    cout<<"\n\n\t\tAlpha Table...............:"<<endl;
    fprintf(fp1,"\n\t\Routh's Approximation:");
    fprintf(fp1,"\n\n\t\Alpha Table..........:\n");
    for(i=1;i<=n;i++)
    {
         printf("\t\t%lf\n",alpha[i]);
         fprintf(fp1,"\t\t%lf\n",alpha[i]);
    }
    gotoxy(50,23);puts("Hit Any Key To Continue..");
    getch();
    clrscr();

}


void tini_2_dim(double x[M][M])
```

```
{
          int i,j;
          for(i=0;i<M;i++)
          {
                    for(j=0;j<M;j++)
                    {
                              x[i][j]=0;
                     }
          }
}



void tini_1_dim(double alpha[M])
{
          int i,j;
          for(i=0;i<M;i++)
          {
                    alpha[i]=0;
          }
}

void impengy()
{
          int i,j,k,s,t,n;
          float a[M][M][2],b[M][M][2],alpha[M],beta[M],imp[M],A[M][M][2],B[M][M][2];

          for(i=0;i<M;i++)
          {
                    for(j=0;j<M;j++)
```

```
        {
                for(k=0;k<2;k++)
                {
                        a[i][j][k]=0;  A[i][j][k]=0;
                        b[i][j][k]=0;  B[i][j][k]=0;
                }
        }
}
for(i=0;i<M;i++)
{
        alpha[i]=0;
        beta[i]=0;
        imp[i]=0;
}


printf("\n\n\tenter order of the System\n");
scanf("%d",&n);
printf("\n\tEnter den coff in ascending powers of s\n");
for(i=1;i<=n+1;i++)
{
        printf("a[1][%d]\t",i);
        scanf("%f",&a[1][i][0]);
}
 printf("\n\tEnter num coff in ascending powers of s\n");
 for(i=1;i<=n;i++)
 {
        printf("b[1][%d]\t",i);
        scanf("%f",&b[1][i][0]);
 }
  alpha[1]=a[1][1][0]/a[1][2][0];
```

```
beta[1]=b[1][1][0]/a[1][2][0];
 t=2;
for(i=1;i<=n;i+=2)
{
        a[1][i][1]=a[1][t][0];
        b[1][i][1]=a[1][t][0];
        t+=2;
}
for(i=2;i<=n;i++)
{
        for(j=1;j<=n;j++)
        {
                if(j%2)
                {
                        a[i][j][0]=a[i-1][j+1][0];
                 }
                 else
                 {
                        a[i][j][0]=a[i-1][j+1][0]-alpha[i-1]*a[i-1][j+2][0];
                  }
          }
          alpha[i]=a[i][1][0]/a[i][2][0];
          t=2;
          for(j=1;j<=n;j+=2)
          {
                a[i][j][1]=a[i][t][0];
                t+=2;
           }
 }
/* N table*/
 for(i=2;i<=n;i++)
```

```
{
      for(j=1;j<=n;j++)
      {
              if(j%2)
              {
                      b[i][j][0]=b[i-1][j+1][0];
              }
              else
              {
                      b[i][j][0]=b[i-1][j+1][0]-beta[i-1]*a[i-1][j+2][0];
              }
      }
      beta[i]=b[i][1][0]/a[i][2][0];
      t=2;
      for(j=1;j<=n;j+=2)
      {
              b[i][j][1]=a[i][t][0];
              t+=2;
      }

    }
    clrscr();

    printf("\n\n\t\tN table\n\n");
    fprintf(fp1,"\n\n\t\tN table\n\n");
    for(i=1;i<=n;i++)
    {
            for(k=0;k<=1;k++)
            {
                    for(j=1;j<=n;j++)
                    {
                        printf("%f\t",b[i][j][k]);
                        fprintf(fp1,"%f\t",b[i][j][k]);
                }
                printf("\n");
                fprintf(fp1,"\n");
        }
        printf("\n");
        fprintf(fp1,"\n");
}
getch();
printf("\nEnter order of reduced system");

scanf("%d",&k);
for(i=1; i<=n;i++)
{
        A[2][i][0] = a[n-k+2][i][0];
        A[2][i][1] = a[n-k+2][i][1];
        A[1][i][1] = a[n-k+1][i][1];
        B[2][i][0] = b[n-k+2][i][0];
        B[2][i][1] = b[n-k+2][i][1];
        B[1][i][1] = b[n-k+1][i][1];
}
for(i=1;i<=k;i+=2)
{
        A[1][i+1][0] = A[1][i][1];
        B[1][i+1][0] = B[2][i][0];
}
A[1][1][0] = A[1][1][1]*alpha[1];
B[1][1][0] = A[1][1][1]*beta[1];
for(i=3;i<=k;i++)
{
        A[1][i][0] = A[2][i-1][0]+alpha [1]*A[1][i+1][0];
        B[1][i][0] = B[2][i-1][0]+beta[1]*A[1][i+1][0];
}
getch();
clrscr();
printf("REDUCED ORDER SYSTEM\n");
printf("\n\n\tNUM IN ASCENDING POWERS OF s\n\n\t");
fprintf(fp1,"\n\n\tREDUCED ORDER NUMERATOR
        POLYNOMIAL'S COEFFICIENTS \n\n\t IN ASCENDING
        POWERS OF s\n");
for(i=1;i<=k;i++)
{
        printf("%f\t",B[1][i][0]);
        fprintf(fp1,"%f\t",B[1][i][0]);
}
printf("\n\n\tDEN IN ASCENDING POWS OF s\n\n\t");
fprintf(fp1,"\n\n\t REDUCED ORDER DENOMINATOR
         POLYNOMIAL'S COEFFICIENTS\n\n\t IN ASCENDING
         POWERS OF s\n");

for(i=1;i<=k+1;i++)
{
        printf("%f\t",A[1][i][0]);
        fprintf(fp1,"%f\t",A[1][i][0]);
}
getch();
fclose(fp1);
```