

LOUGHBOROUGH
UNIVERSITY OF TECHNOLOGY
LIBRARY

AUTHOR

BOSLEY, M J

COPY NO.

000326/02

VOL NO.

CLASS MARK

ARCHIVES COPY
FOR REFERENCE ONLY

**Simplification Of Linear Unsteady
State Models Of Chemical
Processes**

by

M. J. Bosley

SIMPLIFICATION OF LINEAR UNSTEADY-STATE MODELS
OF CHEMICAL PROCESSES

by

MICHAEL JOHN BOSLEY

A Doctoral Thesis submitted in partial fulfilment
of the requirements for the award of

DOCTOR OF PHILOSOPHY

of the

LOUGHBOROUGH UNIVERSITY OF TECHNOLOGY

OCTOBER, 1972.

Supervisor: Dr. F. P. Lees

Department of Chemical
Engineering.

© by Michael John Bosley.

Loughborough University Of Technology Library	
Date	Dec 72
Class	
Acc. No.	000326 / 02

SUMMARY

The need for simple and well understood mathematical models representative of the dynamics of large physical systems has long been recognised, and has recently attracted considerable attention.

This work was prompted by previous studies in the department showing that the responses of complex models of absorption columns could be represented equally well by much simpler models. The study has covered three areas:

- a) The reduction of order of transfer functions.
- b) The reduction of order of state-variable models.
- c) Associated topics and numerical techniques.

A survey has been carried out on methods for reducing state-variable models, or transfer functions, to lower order transfer functions. A number of schemes have been studied, including least-squares fitting in the frequency domain, the truncation of continued fractions, and the matching of moments. It has been shown that in certain situations the continued fraction and moments matching method are in fact identical.

Previous work for reducing the order of state variable models has been reviewed and two new methods have been proposed. Techniques based on modal analysis and least-squares fitting in the time domain have been

discussed. The method of moments has been extended to deal with state variable models: it has been shown that large multi-input - multi-output systems can easily be approximated by smaller models and produce responses which match acceptably those of the full systems. Similarly it has been shown that models can be reduced to give acceptable results by matching the frequency response of the reduced model to that of the full model.

Work on model simplification has involved the use of many numerical techniques. Efficient methods of computing the frequency response of large systems have been investigated and it has been shown that a considerable time saving can be effected by first transforming the model to its Jordan form. The determination of equivalent transfer functions from state variable models has been studied. The existing methods have been compared using large systems and a modified scheme proposed, allowing greater accuracy in determining transfer function coefficients with very little additional work.

ACKNOWLEDGEMENTS

In undertaking a research project of any length one must rely greatly on friends and colleagues, not only for providing assistance, but also a lively atmosphere in which to work. I thank them all sincerely.

I am particularly indebted to Dr. F. P. Lees, who besides suggesting what turned out to be a most interesting topic, has given valuable assistance and guidance throughout the project. My thanks are due also to Mr. H. W. Kropholler who has shown great interest in the work and given considerable help.

I would like to thank all other members of the University who assisted in any way with the work, particularly: Professor D. C. Freshwater for his support and interest; Dr. B. A. Buffham for a number of interesting discussions; Dr. R. M. Neale, now of Courtaulds Ltd, who cooperated on some of the work; the staff of the Computing Centre who have provided a fast and reliable service.

The work has been financed by the Science Research Council whom I acknowledge with thanks.

Finally, I would like to thank Julie who has helped in many ways, not the least of which was the preparation of this manuscript.

<u>SECTION</u>	<u>PAGE</u>
1 INTRODUCTION	1
2 COMPUTATIONAL TECHNIQUES USED	5
2.1 Computation of the time response of state variable models	5
2.1.1 The numerical solution	5
2.1.2 The analytic solution	6
2.1.3 Transformation of \underline{A} to its Jordan form	6
2.1.4 Use of the Jordan form in computing $e^{\underline{A}t}$	7
2.1.5 Jordan form of \underline{A} with complex or repeated eigenvalues	9
2.1.6 Solution for the step and impulse response	11
2.2 Derivation of transfer functions from state variable models	12
2.2.1 Davison's zero method	13
2.2.2 A proof of Davison's method	14
2.2.3 Choice of Γ in Davison's method	16
2.2.4 Checks on the stability of Davison's method	18
2.2.5 Leverrier's Algorithm	19
2.2.6 A modified algorithm	21
2.2.7 Numerical difficulties and an inverse algorithm	23
2.3 Computation of the frequency response of state variable models	25
2.3.1 A review of previous work	27
2.3.2 Frequency response of non- band plant matrices	28

<u>SECTION</u>	<u>PAGE</u>
2.3.3 Method 1 - General complex matrix inversion	29
2.3.4 Method 2 - Inversion of real matrices	29
2.3.5 Method 3 - Frequency response via the canonical form	30
2.3.6 Comparison of methods 1 - 3	33
2.4 Determination of the moments of state variable models	38
2.5 Computer system used	40
2.6 Nomenclature	41
3 THE SIMPLIFICATION OF TRANSFER FUNCTIONS	43
3.1 Characteristics of transfer functions	43
3.2 Classification of reduction methods	46
3.3 Simplification via the time response	50
3.3.1. Sinha and Pille's method	51
3.3.2. Sinha and Bereznai's method	52
3.4 Simplification via the frequency response	54
3.4.1. Meier and Luenberger's method	56
3.4.2. Levy's method	57
3.5 Dominant roots retention	59
3.6 Continued fraction expansion and truncation	61
3.7 Simplification via the moments	63
3.8 Illustrative example	67
3.9 Choice of model and criteria of fit	69
3.10 Nomenclature	71

<u>SECTION</u>	<u>PAGE</u>
4	RELATIONSHIPS BETWEEN THE CONTINUED FRACTION TRUNCATION AND MOMENTS MATCHING METHODS OF MODEL REDUCTION
	73
4.1	Relation 1
	73
4.1.1	A matrix expression for the inversion of continued fractions
	73
4.1.2	Moments of the general, poly- nomial transfer function
	75
4.1.3	A comparison of the two solutions
	77
4.2	Relation 2
	79
4.3	Relation 3
	80
4.4	A generalisation of the relation
	82
4.5	Nomenclature
	85
5	THE SIMPLIFICATION OF STATE VARIABLE MODELS
	86
5.1	Retention of the dominant modes
	86
5.1.1	Problem statement 1
	87
5.1.2	Ordering of the system eigen- values
	87
5.1.3	Problem statement 2
	88
5.1.4	The problem solution
	89
5.1.5	Nicholson's and Davison's method
	90
5.1.6	Marshall's method
	91
5.1.7	A comparison of the methods of Davison and Marshall
	92
5.1.8	Chidambara's methods
	94
5.1.9	Davison's modified models
	95
5.1.10	Analysis of the unretained variables
	96
5.1.11	An extension of Davison's method
	97

<u>SECTION</u>	<u>PAGE</u>
5.2 Least squares fitting in the time domain	98
5.3 Other methods of state variable reduction	101
5.4 Illustrative example	103
5.5 Nomenclature	109
6 REDUCTION OF ORDER OF STATE VARIABLE MODELS USING MOMENTS	111
6.1 Problem statement	111
6.2 Problem formulation	112
6.3 Problem solution	114
6.3.1 Exact fit	115
6.3.2 \underline{B}^* constrained to give correct initial rates	115
6.3.3 Least-squares solution of non-square data matrices	115
6.3.4 Constraint of \underline{B}^* to give correct steady state and moment weighting	117
6.4 Results	118
6.4.1. Illustrative example	120
6.5 Discussion of results	121
6.6 Nomenclature	124
7 THE REDUCTION OF STATE VARIABLE MODELS BY MATCHING THE FREQUENCY RESPONSE	125
7.1 Problem statement	125
7.2 Problem formulation	126
7.3 Problem solution	129
7.4 Results	130

<u>SECTION</u>	<u>PAGE</u>
7.4.1 Illustrative example	131
7.5 Discussion of results	132
7.6 Nomenclature	135
8 CONCLUSIONS AND SUGGESTIONS FOR FURTHER WORK	136
8.1 Davison's zero method	136
8.2 A modified form of the Leverrier algorithm	136
8.3 Frequency response computation of state variable models	137
8.4 Similarities between the moments and continued fraction methods	138
8.5 Reduction of state variable models by matching the moments	138
8.5.1 Introduction of a time delay into a state model	139
8.5.2 Application of the moments method to unstable systems	140
8.5.3 Fitting part of a response only	141
8.5.4 The approximate solution of partial differential equations using moments	141
8.6 Reduction of state variable models by matching the frequency response	142
8.7 Chen's state variable method	142
8.8 Closing remarks	143
9 REFERENCES	144

APPENDIX 1

- A1.1 A listing of the Levy program
- A1.2 Solution of the illustrative example by
Levy's method
- A1.3 A listing of the continued fraction program
- A1.4 Solution of the illustrative example by
Chen and Shieh's method
- A1.5 A listing of Lees' moments program
- A1.6 Solution of the illustrative example by
Lees' method

APPENDIX 2

- A2.1 A listing of the program for the reduction
of state variable models by matching the
moments
- A2.2 Model 1, an overdamped system, reduced by
matching the moments
- A2.3 Model 2, an oscillating model, reduced by
matching the moments
- A2.4 Model 3, an inverting model, reduced by
matching the moments
- A2.5 Model 4, a binary distillation column,
reduced by matching the moments

APPENDIX 3

- A3.1 A listing of the program for the reduction
of state variable models by matching the
frequency response

- A3.2 Model 1, an overdamped system, reduced by
 matching the frequency response
- A3.3 Model 4, a binary distillation column,
 reduced by matching the frequency response

TABLESPAGE

2.1	Values of zeroes for numerical example determined by Davison's method	16
2.2	Davison's estimation of zeroes for numerical example	17
2.3	Coefficients of the characteristic polynomial, for a 36 variable distill- ation problem	24
2.4	Coefficients of the 3rd state variable numerator of the 36th order distillation problem calculated by the Leverrier algorithm	26
2.5	Run times for stagewise processes by the stepping and inversion methods	28
2.6	Storage and run-times for methods 1,2, and 3	33
2.7	Timings for methods 1,2, and 3	34
2.8	Numerical comparisons of methods 1,2, and 3	37
3.1	(1,2) Transfer function parameters	47
3.2	Simple transfer function models	48
3.3	Parameters of modified (1,2) model in illustrative example	68
4.1	Routh array for Eq. (4.21)	81
4.2	Coefficients for the reduction of the seventh order model by the continued fractions and moments methods	84
7.1	Illustrative example - reduced order model	131

FIGURESPAGE

2.1	A block diagram for Davison's method	15
2.2	Two plateau effect of Davison's method	15
2.3	Stability checks on Davison's method	20
2.4	Mill Units/Frequency vs. order for systems with real eigenvalues	35
2.5	Start time in mill units vs. order for systems with real eigenvalues	36
3.1	Typical system step responses	44
3.2	Sinha and Bereznaï's error criteria	53
3.3	Step responses of illustrative model	68
5.1	Illustrative example - response $x_1(t)$	106
5.2	Illustrative example - response $x_2(t)$	107
5.3	Illustrative example - response $x_3(t)$	108

CHAPTER 1

Introduction

1. INTRODUCTION

Recent years have seen an increase in the size and complexity of many industrial processes and engineering operations. These range from the building of super-tankers and jumbo jets to the implementation of vast hydro and nuclear power schemes. The expansion of these industries and the advancing affluence of the civilized world is reflected in the growth of the process industries where the production of petroleum spirit is linked directly to the national economy. All engineering fields depend greatly upon human intervention and decision making, both in the design and operation of plants. In an attempt to eliminate errors there has been a trend to replace, or supplement, the operator by a control system, thus reducing the amount of human decision making. As modern processes become more complex, control technology must advance to satisfy the demands placed upon it.

In the 1950's thought was given to the uses to which computers might be put on process plants, and the 1960's saw the implementation of the first D.D.C. systems, replacing many analogue controllers with a single computer. This however, was only the first step. Once the computer was installed the way was open for realizing hitherto impossible advanced control strategies. However, before many computer systems can be operated effectively an accurate mathematical model

of the plant being controlled is necessary. The modelling of large plants has many problems.

Most processes are highly non-linear, some plant items are distributed parameter systems and lead to complex partial differential equations, while the flow of materials involve transport delays, or dead times. Even with the powerful computers available today such systems, if modelled accurately, could not be solved, let alone used for control purposes. Thus the models must be simplified in some way.

The first simplifications are often carried out at the modelling stage: non-linear systems are usually linearised about their steady states: distributed parameter systems are approximated by finite difference models: and time delays may be replaced by first order lags in series. The result is a set of linear ordinary differential equations.

Classically these equations have been transformed to the Laplace domain to give transfer functions relating one output to one input. For the operation of a single control loop, or to obtain one time or frequency response this is adequate, however, for the analysis of a complete plant many such transfer functions are required. An alternative approach is to convert all the differential equations to first order and to set up a state variable model relating all outputs to all inputs. This was always attractive, but not possible until the widespread appearance of large fast computers. Models of this type are in general use today, and for a complete

system analysis are used in preference to transfer functions.

The classic theorems of linear algebra apply to state variable models, and while theoretically simple to manipulate, are in practice more difficult. This difficulty lies with the model order. Consider a 36 plate distillation column: if it is conventionally modelled with a single equation for each composition and flow on a plate, 72 first order equations result, giving a plant matrix of order 72 with 5184 elements. To store this matrix requires, in most computers, 10 K of core store, and to operate on it considerably more. The modelling of an entire plant, or a distributed parameter system by finite difference methods, can lead to sets of 500 equations. Clearly with systems of that size the storage of the model is virtually impossible and the computational time taken in performing analysis is prohibitive. Two additional points affect the study of very large systems: although ostensibly an accurate model of the process, the very size can confuse and hinder analysis, furthermore, of the many states in the state vector, few may be of interest, the remainder being either unmeasurable or dummy variables. In the case of the distillation column referred to above, only input and output variables are usually of interest, whilst no use is made of the others states.

The above facts point to the necessity to be able

to reduce the size of state variable models and replace a model by a system which, although of lower order, maintains the characteristics of the original model. The advantages of doing this will be summarised briefly. Low order models:

- a) help the understanding of complex models.
- b) reduce computer storage.
- c) reduce the computational effort.
- d) eliminate the need to analyse unimportant states.

However, when reducing the system order it must be remembered that accuracy cannot be sacrificed to achieve a low order model, the results of which are meaningless.

In this thesis the problem of model simplification will be considered. The order reduction of transfer functions has been reviewed and a comparison made, whilst the more interesting reduction of state variable models has been considered in more depth, and two new methods proposed. Some of the numerical methods related to the study of linear systems have been examined.

CHAPTER 2

Computational techniques used

2. COMPUTATIONAL TECHNIQUES USED

Many of the methods described in subsequent chapters will require the use of the same computational techniques, such as the calculation of the time or frequency response. To avoid repetition, and to give a central record of methods used, all computational techniques, together with details of the computer system used, will be presented here.

2.1 COMPUTATION OF THE TIME RESPONSE OF STATE

VARIABLE MODELS

Two methods have been used to compute the time response of state variable models: the analytic solution and a numerical solution. The latter will be discussed only briefly, whilst the former will be considered in some depth as the theory forms the basis for a number of topics in this thesis. Time responses have been computed using both the given methods.

2.1.1 The numerical solution

There are many different numerical methods for solving differential equations, usually based on a truncated Taylor's series, and descriptions of them can be found in most texts on numerical methods (37). Runge-Kutta methods perform adequately for a wide class of problem. The particular method used is the Gill modification of the Runge-Kutta method (55). This routine has been used in preference to the basic

four-point method because it requires only 3/5 of the computer storage and minimizes rounding errors.

2.1.2 The analytic solution

The solution of

$$\dot{\underline{x}} = \underline{A}\underline{x} + \underline{B}\underline{u} \quad (2.1)$$

where A and B are both time invariant and u the input, varies with time is (100):

$$\underline{x}(t) = e^{\underline{A}t} \underline{x}(0) + \int_0^t e^{\underline{A}(t-\tau)} \underline{B}\underline{u}(\tau) d\tau \quad (2.2)$$

As all the states in Eq. (2.1) are linearised and only deviations about a steady state need be considered $\underline{x}(0) = 0$ and Eq. (2.2) reduces to

$$\underline{x}(t) = \int_0^t e^{\underline{A}(t-\tau)} \underline{B}\underline{u}(\tau) d\tau \quad (2.3)$$

The problem in Eq. (2.3) is the computation of the exponential matrix $e^{\underline{A}(t-\tau)}$. Buffham and Kropholler (19) have considered the many ways of computing this matrix. In this work it was decided to compute $e^{\underline{A}t}$ from the Jordan canonical form (120) as efficient eigenvalue and eigenvector routines were available (the Q.R. transform).

2.1.3 Transformation of A to its Jordan form

Any matrix A having distinct roots (real or complex) may be transformed into the diagonal matrix

$$\underline{\Lambda} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \quad (2.4)$$

where λ_i are the eigenvalues of \underline{A} . $\underline{\Lambda}$ is said to be the Jordan canonical form of \underline{A} . In general this transformation does not exist for systems having multiple eigenvalues, although similar ones do. The transformation is:

$$\underline{\Lambda} = \underline{U}^{-1} \underline{A} \underline{U} \quad (2.5)$$

where the unique, non-singular, transformation matrix may be computed in a number of ways (23), but is most readily constructed from the eigenvectors of \underline{A} in the following manner

$$\underline{U} = (\underline{u}_1 \mid \underline{u}_2 \mid \dots \mid \underline{u}_n) \quad (2.6)$$

where \underline{u}_i is the eigenvector corresponding to λ_i and is calculated from

$$(\underline{A} - \lambda_i \underline{I}) \underline{u}_i = \underline{0} \quad (2.7)$$

2.1.4. Use of the Jordan form in computing $e^{\underline{A}t}$

The solution of the homogeneous system

$$\dot{\underline{x}} = \underline{A} \underline{x} \quad \underline{x} = \underline{x}(0) \text{ at } t = 0 \quad (2.8)$$

$$\text{is } \underline{x}(t) = e^{\underline{A}t} \underline{x}(0) \quad (2.9)$$

Substituting $\underline{x} = \underline{U}\underline{y}$ into Eq. (2.8) gives

$$\underline{U}\dot{\underline{y}} = \underline{A}\underline{U}\underline{y} \quad \underline{y} = \underline{y}(0) \text{ at } t = 0 \quad (2.10)$$

$$\dot{\underline{y}} = \underline{U}^{-1} \underline{A}\underline{U}\underline{y} = \underline{\Lambda}\underline{y} \quad (2.11)$$

The solution to Eq. (2.11) is

$$\underline{y}(t) = e^{\underline{\Lambda}t} \underline{y}(0) \quad (2.12)$$

where

$$e^{\underline{\Lambda}t} = \begin{bmatrix} e^{\lambda_1 t} & & & \\ & e^{\lambda_2 t} & & \\ & & \ddots & \\ & & & e^{\lambda_n t} \end{bmatrix} \quad (2.13)$$

Thus Eq. (2.13) is easily computed as it possesses only scalar quantities on the diagonal. Eq. (2.12) is now transformed back to the original variables by substitution of $\underline{y} = \underline{U}^{-1}\underline{x}$.

$$\underline{x}(t) = \underline{U} e^{\underline{\Lambda}t} \underline{U}^{-1} \underline{x}(0) \quad (2.14)$$

The same method is used to solve the non-homogeneous system, Eq. (2.1).

2.1.5 Jordan form of A with complex or repeated eigenvalues

The method given above applies when the eigenvalues of \underline{A} are complex, however, complex arithmetic is involved but may be removed by use of a further transformation. This transformation is based on the fact that complex eigenvalues and eigenvectors must occur in conjugate pairs and that one complex eigenvalue contains all the information about that pair. The Jordan matrix with complex entries is

$$\underline{\Lambda} = \begin{bmatrix} \lambda_1 & & & & \\ & (\lambda_k + i \phi_k) & & & \\ & (\lambda_k - i \phi_k) & & & \\ & & \cdot & & \\ & & & \cdot & \\ & & & & \lambda_n \end{bmatrix} \quad (2.15)$$

and may be transformed to

$$\underline{\Lambda} = \begin{bmatrix} \lambda_1 & & & & \\ & \cdot & & & \\ & & \lambda_k & \phi_k & \\ & & -\phi_k & \lambda_k & \\ & & & \cdot & \\ & & & & \lambda_n \end{bmatrix} \quad (2.16)$$

by

$$\begin{bmatrix} 1 & & & & \\ & \cdot & & & \\ & & \cdot & & \\ & & & \frac{1}{2} & -i/2 \\ & & & \frac{1}{2} & i/2 \\ & & & & \cdot \\ & & & & & 1 \end{bmatrix}$$

and at the same time \underline{U} is transformed from

$$\underline{U} = (\underline{u}_1 | \dots | \underline{u}_k^R + i\underline{u}_k^C | \underline{u}_k^R - i\underline{u}_k^C | \dots | \underline{u}_n) \quad (2.17)$$

to

$$\underline{U} = (\underline{u}_1 | \dots | \underline{u}_k^R | \underline{u}_k^C | \dots | \underline{u}_n) \quad (2.18)$$

with this form of \underline{U} and $\underline{\Lambda}$, $e^{\underline{\Lambda}t}$ becomes

$$e^{\underline{\Lambda}t} = \begin{bmatrix} e^{\lambda_1 t} & & & \\ & \cdot & & \\ & & e^{\lambda_k t} \cos \phi_k t & e^{\lambda_k t} \sin \phi_k t \\ & & -e^{\lambda_k t} \sin \phi_k t & e^{\lambda_k t} \cos \phi_k t \\ & & & \cdot \\ & & & & e^{\lambda_n t} \end{bmatrix} \quad (2.19)$$

Clearly the forms given above for the complex eigenvalues are a general case, and Eqs. (2.4, 2.6, 2.13) are a special case of Eqs. (2.16, 2.18, 2.19).

When a matrix has multiple eigenvalues it may also have, though not necessarily, multiple eigenvectors. This leads to a singular \underline{U} . In this case transformations of the following form may be possible.

$$\underline{\Lambda} = \underline{P}^{-1} \underline{AP} = \begin{bmatrix} \lambda_1 & 1 & 0 \\ 0 & \lambda_1 & 1 \\ 0 & 0 & \lambda_1 \end{bmatrix} \quad (2.20)$$

$$e^{\underline{\Lambda}t} = \begin{bmatrix} e^{\lambda_1 t} & te^{\lambda_1 t} & \frac{1}{2}t^2 e^{\lambda_1 t} \\ 0 & e^{\lambda_1 t} & te^{\lambda_1 t} \\ 0 & 0 & e^{\lambda_1 t} \end{bmatrix} \quad (2.21)$$

where \underline{P} is composed of vectors, not eigenvectors, computed from

$$(\underline{A} - \lambda \underline{I}) \underline{p}_i = \underline{p}_{i-1} \quad i \neq 1 \quad (2.22)$$

and \underline{p}_1 is the eigenvector corresponding to λ . A detailed description of the above transformations is given by Ogata [100].

2.1.6 Solution for the step and impulse response

Eq. (2.3) for the impulse response, with $e^{\underline{\Lambda}t}$ and \underline{U} given by Eqs. (2.19, 2.18) for distinct eigenvalues is

$$\underline{x}(t) = \underline{U} e^{\underline{\Lambda}t} \underline{U}^{-1} \underline{B}u \quad (2.23)$$

Eqn. (2.3) for the step response is

$$\underline{x}(t) = \underline{U} \int_0^t e^{\underline{\Lambda}t} \underline{U}^{-1} \underline{B}u \, dt \quad (2.24)$$

where $\int_0^t e^{\underline{\Lambda}t} dt$ corresponding to $e^{\underline{\Lambda}t}$, Eqn. (2.19) is

$$\begin{bmatrix} a_1 & & & & \\ & \cdot & & & \\ & & b_k & c_k & \\ & & -c_k & b_k & \\ & & & \cdot & \\ & & & & a_n \end{bmatrix} \quad (2.25)$$

$$\text{and } a_i = \frac{e^{\lambda_i t} - 1}{\lambda_i} \quad (2.25a)$$

$$b_k = \frac{1}{\lambda_k^2 + \phi_k^2} (e^{\lambda_k t} (\phi_k \sin \phi_k t + \lambda_k \cos \phi_k t) - \lambda_k) \quad (2.25b)$$

$$c_k = \frac{1}{\lambda_k^2 + \phi_k^2} (e^{\lambda_k t} (\lambda_k \sin \phi_k t - \phi_k \cos \phi_k t) + \phi_k) \quad (2.25c)$$

2.2 DERIVATION OF TRANSFER FUNCTIONS FROM STATE VARIABLE MODELS

Two types of models are commonly used in modern control theory: the transfer function

$$G(s) = \frac{\sum_{i=0}^m b_i s^i}{\sum_{j=0}^n a_j s^j} \quad m \leq n \quad (2.26)$$

and the state variable model

$$\dot{\underline{x}} = \underline{A}\underline{x} + \underline{B}u \quad (2.1)$$

It is often necessary to make comparisons between these two models, which may be used to represent the same

system, and though of different form, the two can be related. Chen and Haas (23) have summarised the methods available to decompose a transfer function into state form. There is, however, no wholly accepted method of performing the reverse operation. Two methods have been investigated in depth: the Leverrier algorithm (78) , also called the Frame-Souriau-Faddeev algorithm, and a method of E. J. Davison (40). Method description, failings and suggested improvements follow.

2.2.1 Davison's zero method (40)

The state variable model, Eq. (2.1) in the frequency domain is written:

$$(s\mathbf{I} - \mathbf{A}) \mathbf{x}(s) = \mathbf{B}\mathbf{u}(s) \quad (2.27)$$

This equation for the impulse response (when $\mathbf{B}\mathbf{u}(s) = \mathbf{B}\mathbf{u}$) may be solved for the j th variable using Cramer's Rule

$$x_j(s) = \frac{|\mathbf{sI} - \mathbf{A}|_j}{|\mathbf{sI} - \mathbf{A}|} \quad (2.28)$$

where $|\mathbf{sI} - \mathbf{A}|_j$ is $|\mathbf{sI} - \mathbf{A}|$ with the j th column replaced by $\mathbf{B}\mathbf{u}$. Although not directly applicable to high order systems Cramer's Rule gives the basis for Davison's algorithm.

Eq. (2.28) requires $|\mathbf{sI} - \mathbf{A}|$, the characteristic equation of \mathbf{A} , which may be found by determining its eigenvalues. Davison makes $|\mathbf{sI} - \mathbf{A}|_j$ also an eigenvalue problem.

The zeroes of the j th variable are found by replacing the j th column in \underline{A} by $\Gamma \underline{b}$, where Γ is a large scalar to give

$$\underline{A}_j^\dagger = \begin{bmatrix} a_{11} & \dots & a_{1j-1} & \Gamma b_1 & a_{1j+1} & \dots & a_{1n} \\ a_{21} & \dots & a_{2j-1} & \Gamma b_2 & a_{2j+1} & \dots & a_{2n} \\ \vdots & & & & & & \\ a_{n1} & \dots & a_{nj-1} & \Gamma b_n & a_{nj+1} & \dots & a_{nn} \end{bmatrix} \quad (2.29)$$

The zeroes of the system are included in the eigenvalues of \underline{A}_j^\dagger . Any matrix of order n must have n eigenvalues whereas Eq. (2.28) need not have any zeroes, and has a maximum of $n-1$: the additional roots of \underline{A}_j^\dagger are extraneous and not system zeroes. These extraneous roots can be recognised by solving the problem at different values of Γ when the true zeroes maintain a constant value and the additional roots tend to infinity.

2.2.2 A proof of Davison's method

A number of different proofs of the Davison method have been given (43, 69, 119) but it is best understood by applying root locus theory.

Consider the negative feedback system shown in Fig. 2.1: it is well known that when the feedback gain is zero ($\Gamma = 0$), i.e. the open loop system, that the poles of $x_j(s)$ are given by the eigenvalues of the plant matrix \underline{A} , but when the loop is closed the poles of the closed loop system (the eigenvalues of \underline{A}_j^\dagger) migrate to

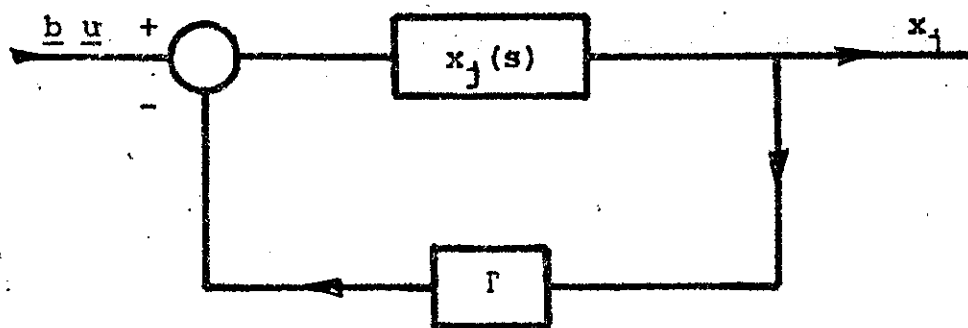


Figure 2.1 A block diagram for Davison's method

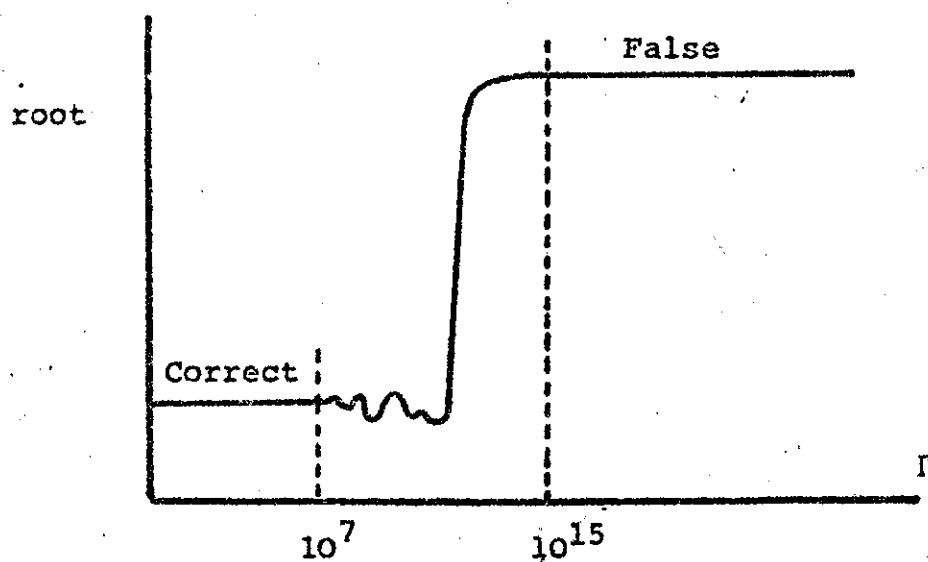


Figure 2.2 Two plateau effect of Davison's method

the open loop zeroes (the zeroes of $x_j(s)$) as Γ is increased to infinity. The negative feedback loop corresponds to the subtraction of the vector $\Gamma \underline{Bu}$ from the j th column of the matrix \underline{A} .

Whilst very easily programmed the method has a numerical problem associated with the choice of Γ .

2.2.3 Choice of Γ in Davison's method

The following example suffices to illustrate the problem (12).

$$\dot{\underline{x}} = \begin{bmatrix} 0 & -2 & -1 & -1 \\ 0 & -7 & -1 & -1 \\ 0 & 0 & -4 & -1 \\ 0 & 0 & 0 & -2 \end{bmatrix} \underline{x} + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Determine the transfer function corresponding to x_1 .

$$x_1(s) = \frac{(s+5)(s+1)(s+3)}{s^4+13s^3+50s^2+56s}$$

Table 2.1 shows the determined zeroes for variable 1 for different values of Γ .

Γ	Zeroes of system			extraneous root
10^3	-5.00375	-1.00225	-3.00300	0.996009×10^3
10^5	-5.00004	-1.00003	-3.00003	0.999960×10^5
10^7	-4.99999	-1.00000	-3.00000	10^7
10^9	-5.00802	-1.00000	-3.00000	10^9
10^{11}	-5.32843	-1.00000	-3.00000	10^{11}
10^{13}	-5.00000	-0.42857	-3.57143	10^{13}
10^{15}	-5.00000	-0.42857	-3.57143	10^{15}
actual zeroes	-5.00000	-1.00000	-3.00000	

Table 2.1 Values of zeroes for numerical example determined by Davison's method

Best results were obtained for Γ equal to 10^7 . It will however be noted from the table that two of the roots each exhibit two distinct values. Γ has been increased to 10^{27} without any further change in the value of the roots. Without any further information it is difficult to know which value to select. The same difficulty arose with other problems, including that given by Davison. Best results were always obtained when Γ was equal to 10^7 whereas Davison had recommended a value of 10^{15} . Typical results follow the pattern shown in Fig. 2.2 where two distinct plateaus exist.

The effect described above appears to be independent of problem size but dependent upon the particular computer and program used. This opinion is also held by Davison who has rerun the same problem on a different machine and obtained entirely satisfactory results for all values of Γ . His results are shown in Table 2.2. A number of rules have been developed to check that Γ has not moved into a region of instability (69).

Value of Γ	Zeros of system			Extraneous root
10^5	-1.00000	-3.00000	-5.00006	0.9996×10^5
10^7	-1.00000	-3.00000	-5.00000	10^7
10^9	-1.00000	-3.00000	-5.00000	10^9
10^{11}	-1.00000	-3.00000	-5.00000	10^{11}
10^{13}	-1.00000	-3.00000	-5.00000	10^{13}
10^{15}	-1.00000	-3.00000	-5.00000	10^{15}

Table 2.2 Davison's estimation of zeroes for numerical example c.f. Table 2.1

2.2.4 Checks on the stability of Davison's method

The zeroes are included in the roots of

$$|s\mathbf{I} - \mathbf{A}^\dagger| = 0$$

which for variable 2 of a third order problem are the roots of

$$\begin{vmatrix} s - a_{11} & -\Gamma b_1 & -a_{13} \\ -a_{21} & s - \Gamma b_2 & -a_{23} \\ -a_{31} & -\Gamma b_3 & s - a_{33} \end{vmatrix} = 0 \quad (2.30)$$

which may also be written

$$-\Gamma \begin{vmatrix} s - a_{11} & b_1 & -a_{13} \\ -a_{21} & b_2 & -a_{23} \\ -a_{31} & b_3 & s - a_{33} \end{vmatrix} + \begin{vmatrix} s - a_{11} & 0 & -a_{13} \\ -a_{21} & s & -a_{23} \\ -a_{31} & 0 & s - a_{33} \end{vmatrix} = 0 \quad (2.31)$$

The second determinant has a term s on the diagonal, hence it is not possible for this determinant to contribute to the constant term in the expansion of Eq. (2.31).

This constant is proportional to Γ and may be obtained in practice as the product of all the eigenvalues of \mathbf{A}^\dagger .

From Eq. (2.31), neglecting the second determinant, it follows that

$$-\Gamma K \prod_{i=1}^m (s + z_i) = 0 \quad (2.32)$$

where K is the system gain and z_i are the system zeroes, and the constant term in Eq. (2.32) is given by

$$-\Gamma K \prod_{i=1}^m z_i$$

which should equal the product of all the eigenvalues of \underline{A}^\dagger . Therefore

$$-\Gamma K \prod_{i=1}^m z_i = \prod_{i=1}^m z_i \cdot \prod_{j=1}^{n-m} e_j \quad (2.33)$$

where e_j are the extraneous roots. Since $\prod_{i=1}^m z_i$ is a constant in the system, for large Γ the system gain is given by

$$K = \frac{\prod_{j=1}^{n-m} e_j}{\Gamma} \quad (2.34)$$

which is, of course, also a constant.

The extraneous root product/ Γ and the eigenvalue product/ Γ provide a monitor on the choice of Γ . Their responses to different values of Γ are shown in Fig. 2.3. Best values of the zeroes are obtained when both curves are horizontal. On the computer system used at Loughborough best results were always given when Γ equalled 10^7 , but it must be stressed that before the method is extensively used Γ should be determined for a particular computer and program being used.

2.2.5 Leverrier's Algorithm (78)

This algorithm has been given by many people since it first appeared in 1840. Modified versions have been given by Faddeev (47), Frame (49), Ghani and Ackroyd (52), Marshall (84), Morgan (94), Rosenbrock (104), and Souriau (111). Bass (9) has discussed the history of its discovery.

The solution of Eq. (2.27) is

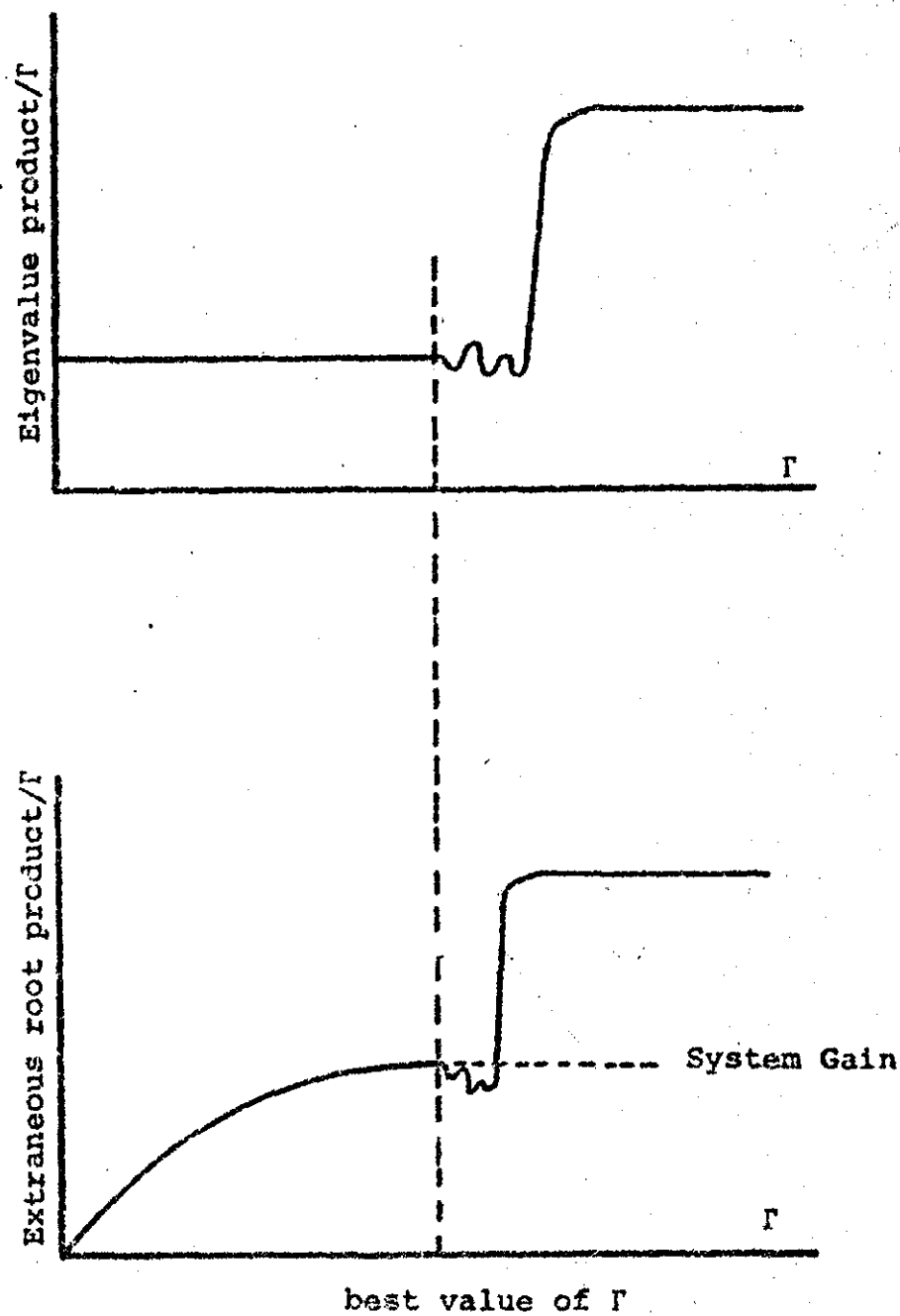


Figure 2.3 Stability checks on Davison's method

$$\underline{x}(s) = (s\underline{I} - \underline{A})^{-1} \underline{B}u \quad (2.35)$$

where

$$(s\underline{I} - \underline{A})^{-1} = \frac{\text{Adj}(s\underline{I} - \underline{A})}{|s\underline{I} - \underline{A}|} \quad (2.35a)$$

rearranging gives

$$|s\underline{I} - \underline{A}| \cdot \underline{I} = \text{Adj}(s\underline{I} - \underline{A}) \cdot (s\underline{I} - \underline{A}) \quad (2.36)$$

which may be written

$$(s^n - h_1 s^{n-1} \dots h_n) \underline{I} = (s^{n-1} \underline{I} + s^{n-2} \underline{R}_1 \dots + \underline{R}_{n-1})(s\underline{I} - \underline{A}) \quad (2.37)$$

where a comparison of the coefficients in s shows that the scalars, h_i , and the matrices, \underline{R}_j may be determined from the following recursive scheme.

$$\begin{aligned} \underline{A}_1 &= \underline{A} & h_1 &= \text{tr}(\underline{A}_1) & \underline{R}_1 &= \underline{A}_1 - h_1 \underline{I} \\ \underline{A}_2 &= \underline{A} \underline{R}_1 & h_2 &= \frac{1}{2} \text{tr}(\underline{A}_2) & \underline{R}_2 &= \underline{A}_2 - h_2 \underline{I} \\ &\vdots & & & & \\ &\vdots & & & & \\ &\vdots & & & & \\ \underline{A}_{n-1} &= \underline{A} \underline{R}_{n-2} & h_{n-1} &= \frac{1}{n-1} \text{tr}(\underline{A}_{n-1}) & \underline{R}_{n-1} &= \underline{A}_{n-1} - h_{n-1} \underline{I} \\ \underline{A}_n &= \underline{A} \underline{R}_{n-1} & h_n &= \frac{1}{n} \text{tr}(\underline{A}_n) & \underline{R}_n &= \underline{A}_n - h_n \underline{I} = \underline{0} \end{aligned} \quad (2.38)$$

In the absence of numerical error \underline{R}_n will be the null matrix.

2.2.6 A modified algorithm (15)

Essentially the same scheme has been used but

$|s\underline{I} - \underline{A}|$ has been computed from the eigenvalues of \underline{A} and $\text{Adj}(s\underline{I} - \underline{A})\underline{B}\underline{u}$ is evaluated rather than $\text{Adj}(s\underline{I} - \underline{A})$.
Let

$$\underline{x}(s) = \frac{\text{Adj}(s\underline{I} - \underline{A})\underline{B}\underline{u}}{|s\underline{I} - \underline{A}|}$$

$$= \frac{\begin{bmatrix} b_{10} + b_{11}s + \dots + b_{1n-1}s^{n-1} \\ \vdots \\ b_{n0} + b_{n1}s + \dots + b_{nn-1}s^{n-1} \end{bmatrix}}{a_0 + a_1s + a_2s^2 + \dots + a_ns^n} \quad (2.39)$$

where the denominator of Eq. (2.39) is the characteristic equation of \underline{A} and a_n is unity. Let the coefficients in the numerator vector, Eq. (2.39) be arranged into the following partitioned matrix

$$\begin{bmatrix} b_{10} & b_{11} & \dots & b_{1n-1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n0} & b_{n1} & \dots & b_{nn-1} \end{bmatrix} = (\underline{b}_0 \mid \underline{b}_1 \mid \dots \mid \underline{b}_{n-1})$$

and the vectors \underline{b}_i obtained from the following recursion

formula.

$$\underline{b}_{n-1} = \underline{Bu}$$

$$\underline{b}_{n-2} = \underline{Ab}_{n-1} + a_{n-1}\underline{Bu}$$

$$\text{or } \underline{b}_i = \underline{Ab}_{i+1} + a_{i+1}\underline{Bu} \quad i = n-2..0 \quad (2.41)$$

The proof of this scheme and the normal Leverrier algorithm have been given (97,105).

2.2.7 Numerical difficulties and an inverse algorithm(12)

The algorithms described above suffer from severe numerical difficulties in the evaluation of the polynomials. This is also true when double precision arithmetic is used, and in some cases triple length working must be used to evaluate the numerator accurately. This numerical inaccuracy is shown in Table 2.3 where column 2 shows the coefficients of the characteristic equation evaluated with the algorithm and column 4 shows the same coefficients determined from the eigenvalues (these results are computed from a 36th order model of a distillation column (69)). There is considerable difference, particularly in the low power of s .

It is, however, possible to solve the Leverrier algorithm from either end of the characteristic equation and relate the two solutions.

Table 2.3 Coefficients of the Characteristic Polynomial, for a 36 variable distillation problem

Number	Coefficient values		
	Leverrier algorithm		
	Using A	Using A ⁻¹	Using Eigen values
0	1.000 000 000	1.799 E32	1.000 000 000
1	1.723 878 716 E3	-2.853 E32	1.723 878 717 E3
2	1.404 607 824 E6	4.526 E32	1.404 607 824 E6
3	7.208 593 986 E8	-7.187 E32	7.208 593 988 E8
4	2.619 827 522 E11	1.142 E33	2.619 827 523 E11
5	7.186 753 624 E13	-1.815 E33	7.186 753 627 E13
6	1.548 789 462 E16	2.889 E33	1.548 789 463 E16
7	2.694 410 286 E18	-4.602 E33	2.694 410 287 E18
8	3.858 496 818 E20	7.337 E33	3.858 496 820 E20
9	4.614 704 668 E22	-1.170 E34	4.614 704 471 E22
10	4.660 257 222 E24	1.869 E34	4.660 257 225 E24
11	4.007 483 317 E26	-2.988 E34	4.007 483 321 E26
12	2.953 465 628 E28	4.782 E34	2.953 465 631 E28
13	1.874 615 830 E30	-7.659 E34	1.874 615 832 E30
14	1.028 419 199 E32	1.229 E35	1.028 419 201 E32
15	4.888 518 713 E33	-1.922 E35	4.888 518 717 E33
16	2.016 377 269 E35	5.185 E35	2.016 377 275 E35
17	7.220 894 099 E36	6.710 928 546 E36	7.220 894 045 E36
18	2.244 337 526 E38	2.252 555 231 E38	2.244 337 649 E38
19	6.046 961 334 E39	6.045 632 899 E39	6.046 959 099 E39
20	1.409 361 476 E41	1.409 387 162 E41	1.409 365 723 E41
21	2.832 967 649 E42	2.832 878 505 E42	2.832 881 983 E42
22	4.889 439 103 E43	4.890 980 604 E43	4.890 980 047 E43
23	7.245 471 570 E44	7.215 943 898 E44	7.215 944 002 E44
24	8.472 298 831 E45	9.039 736 094 E45	9.039 736 096 E45
25	2.047 E47	9.541 540 579 E46	9.541 540 600 E46
26	-2.025 E49	8.406 781 390 E47	8.406 789 408 E47
27	4.085 E51	6.114 268 754 E48	6.114 268 769 E48
28	-7.898 E53	3.622 329 790 E49	3.622 329 799 E49
29	1.531 E56	1.720 558 026 E50	1.720 558 031 E50
30	-2.972 E58	6.428 318 986 E50	6.428 319 004 E50
31	5.776 E60	1.845 364 696 E51	1.845 364 702 E51
32	-1.123 E63	3.949 589 620 E51	3.949 589 632 E51
33	2.187 E65	6.045 981 110 E51	6.045 981 131 E51
34	-4.261 E67	6.209 356 692 E51	6.209 356 714 E51
35	8.309 E69	3.809 739 393 E51	3.809 739 408 E51
36	-1.621 E72	1.050 105 483 E51	1.050 105 487 E51

If the inverse of \underline{A} exists then Eq. (2.37) above, may also be written

$$(s^n - t_1 s^{n-1} \dots t_n) \underline{I} = (s^{n-1} \underline{T}_0 + s^{n-2} \underline{T}_1 + \dots + \underline{T}_{n-1}) \cdot (s \underline{I} - \underline{A}) \quad (2.42)$$

where it may be shown that

$$\begin{aligned} t_i &= -h_{n-i}/h_n \\ \underline{R}_i &= h_n \underline{T}_{n-i-1} \underline{A}^{-1} \end{aligned} \quad (2.43)$$

Thus \underline{A} may be inverted and the normal program used to re-solve the problem from the opposite end of the polynomial. This gives a second set of coefficients for the characteristic equation based on the inverse problem. Column 3 in Table 2.3 shows these coefficients for the distillation problem. The computed results for the numerator polynomial for variable 3 are shown in Table 2.4. In each of these polynomials actually used for subsequent work the first 20 coefficients have been calculated using \underline{A} and the remainder using \underline{A}^{-1} .

2.3 Computation of the frequency response of state variable models

The frequency response may be computed from the state variable model, Eq. (2.1), in a number of different ways. Eq. (2.1) may be transformed into the Laplace domain

$$(s \underline{I} - \underline{A}) \underline{x}(s) = \underline{x}(0) \quad (2.44)$$

Table 2.4 Coefficients of the 3rd State Variable
numerator of the 36th order distillation
problem calculated by the Leverrier algorithm

Number	Characteristic equation calculated using Newton's identities		Characteristic equation calculated using eigenvalues	
	Using A	Using A ⁻¹	Using A	Using A ⁻¹
0	0-0	-1.047 276 037 E34	0-000 000 000	-1.591 215 960 E35
1	0-0	+1.605 399 219 E34	0-0	2.425 972 991 E35
2	0-0	-2.461 438 069 E34	0-0	-3.698 649 970 E35
3	0-0	+3.774 705 971 E34	0-0	5.638 986 249 E35
4	0-0	-5.789 890 479 E34	0-0	-8.597 248 596 E35
5	0-0	+8.882 915 830 E34	0-000 000 000	1.310 746 543 E36
6	-1.179 669 631 E10	-1.363 151 216 E35	-1.179 805 827 E10	-1.998 383 856 E36
7	-1.327 803 428 E13	+2.092 384 503 E35	-1.327 956 727 E13	3.046 774 159 E36
8	-7.028 589 371 E15	-3.212 580 382 E35	-7.029 400 843 E15	-4.645 185 006 E36
9	-2.330 977 704 E18	+4.933 874 120 E35	-2.331 246 823 E18	7.082 188 048 E36
10	-5.442 691 235 E20	-7.579 676 049 E35	-5.443 319 612 E20	-1.079 776 590 E37
11	-9.530 679 386 E22	+1.164 794 183 E36	-9.531 779 736 E22	1.646 276 355 E37
12	-1.301 599 356 E25	-1.790 571 518 E36	-1.301 749 629 E25	-2.510 003 514 E37
13	-1.423 186 300 E27	+2.753 509 384 E36	-1.423 350 615 E27	3.826 916 676 E37
14	-1.268 962 095 E29	-4.235 872 100 E36	-1.269 108 592 E29	-5.834 812 934 E37
15	-9.348 549 816 E30	+6.518 824 006 E36	-9.349 629 357 E30	8.896 267 276 E37
16	-5.744 386 384 E32	-1.003 695 856 E37	-5.745 049 128 E32	-1.356 414 685 E38
17	-2.963 624 563 E34	+1.542 911 312 E37	-2.963 967 698 E34	2.069 806 951 E38
18	-1.289 393 385 E36	-2.511 090 191 E37	-1.289 540 286 E36	-3.166 054 608 E38
19	-4.742 429 774 E37	-1.069 922 414 E37	-4.743 016 497 E37	4.332 957 892 E38
20	-1.475 861 173 E39	-1.532 429 487 E39	-1.475 954 141 E39	-2.208 776 935 E39
21	-3.880 759 571 E40	-3.873 539 606 E40	-3.882 728 680 E40	3.771 043 855 E40
22	-8.642 871 281 E41	-8.614 446 230 E41	-8.614 091 534 E41	-8.631 104 119 E41
23	-1.546 897 410 E43	-1.605 046 223 E43	-1.605 252 125 E43	-1.604 993 472 E43
24	-3.633 158 214 E44	-2.497 918 250 E44	-2.498 208 278 E44	-2.498 242 750 E44
25	1.892 548 459 E46	-3.221 444 084 E45	-3.221 718 343 E45	-3.221 810 564 E45
26	-4.354 967 227 E48	-3.408 585 349 E46	-3.410 967 196 E47	-3.408 979 699 E46
27	+8.430 448 017 E51	-2.921 894 255 E47	-2.881 954 645 E47	-2.922 231 485 E47
28	-1.646 998 349 E53	-1.997 387 696 E48	-2.813 501 778 E49	-1.997 618 321 E48
29	3.218 741 635 E55	-1.067 602 481 E49	1.545 995 314 E50	-1.067 725 739 E49
30	-6.295 433 219 E57	-4.352 601 567 E49	-3.352 506 519 E52	-4.353 104 103 E49
31	1.232 305 529 E59	-1.311 059 278 E50	6.782 629 889 E54	-1.311 210 647 E50
32	-2.414 156 850 E62	-2.793 562 455 E50	-1.374 085 700 E57	-2.793 884 989 E50
33	4.733 257 588 E64	-3.943 433 339 E50	2.783 721 051 E60	-3.943 888 634 E50
34	-9.287 328 903 E66	-3.280 251 742 E50	-5.639 511 027 E61	-3.280 630 469 E50
35	1.823 657 143 E69	-1.208 681 662 E50	1.142 510 347 E64	-1.208 821 213 E50

and then into the frequency domain by substitution of $s = i\omega$ to give

$$(i\omega \underline{I} - \underline{A}) \underline{x}(i\omega) = \underline{x}(0) \quad (2.45)$$

where $\underline{x}(i\omega)$ is a vector having real and imaginary parts.

Frequency response analysis requires the solution of Eq. (2.45) over a wide frequency range, or for many different values of ω . For high order systems the complex inversion involved in the many solutions may prove a heavy work load.

2.3.1 A review of previous work

Many of the stagewise problems encountered in chemical engineering give rise to plant matrices of special form. These are almost invariably band matrices and very often tri-diagonal. Woods (122) has solved Eq. (2.45) using a complex arithmetic matrix inversion routine, whereas Bollinger (10) and Lamb and Rippin (72) have utilized the band matrix structure to produce computationally more efficient techniques. Bollinger has used what is basically a Gaussian elimination on the band elements of the plant matrix. Lamb and Rippin have used a method which involves plate to plate calculations made up the stagewise plant being modelled.

Shunta and Luyben (108) have made a comparison of the stepping method and the general complex matrix inversion method for band matrix systems. Four different sized distillation columns were investigated: 6, 10, 20 and 30 plates for 65 values of frequency. Their results

are shown in Table 2.5.

No. of Trays	Stepping time (secs)	Inversion time (secs)
6	2.64	100.4
10	3.16	336.2
20	4.25	1891.3
30	5.37	5654

Table 2.5 Run times for stagewise processes by the
stepping and inversion methods

These results may be summarised by the equations

$$\text{Stepping time} = .11 n + 2$$

$$\text{Inversion time} = 1.45 n^{2.37}$$

where n is the number of trays in the column. Although the method of Bollinger was not run an estimate was made for the 75 plate column he investigated. This gave 800 seconds for the Bollinger method compared to 10 and 40,000 seconds respectively for the stepping and complex inversion methods.

It was thus shown, quite conclusively, that when it is possible to use the stepping technique much computational effort could be avoided, in addition to which the complex matrix inversion method has the disadvantage of requiring considerable core store.

2.3.2 Frequency response of non-band plant matrices

When computing the frequency response of general

systems, i.e. those with a plant matrix which is not band-structured, the efficient methods discussed above cannot be used and the general complex matrix inversion, or some alternative must be resorted to.

Three methods of computing the frequency response for all states in \underline{x} , and one when only some of the states are needed, have been investigated and compared [11].

2.3.3 Method 1 - General complex matrix inversion

Eq. (2.45)

$$(i\omega \underline{I} - \underline{A}) \underline{x} (i\omega) = \underline{x}(0)$$

may be solved using a general complex matrix inversion program, as was done by Shunta and Luyben [108] for each frequency considered. The routine used is based on the Crout factorization and has been described by Wilkinson [120].

2.3.4 Method 2 - Inversion of real matrices

An alternative to solving Eq. (2.45) directly, using complex arithmetic is to rearrange the equations so that real numbers only need be used. This may be done in a number of ways.

Pang and Johnson [101], working on a liquid-liquid extraction column have used the method of Lanczos [73]. If $(i\omega \underline{I} - \underline{A})^{-1}$ is known then the problem is effectively solved. Define two matrices \underline{Y} and \underline{Z} such that

$$i\underline{Y} + \underline{Z} = (i\omega \underline{I} - \underline{A})^{-1} \quad (2.46)$$

$$\text{and} \quad (i\omega \underline{I} - \underline{A})(i\underline{Y} + \underline{Z}) = \underline{I} \quad (2.47)$$

Multiplying out Eq. (2.47) and separating into real and imaginary parts

$$\begin{aligned}\omega \underline{Z} - \underline{A} \underline{Y} &= \underline{0} \\ -\underline{A} \underline{Z} - \omega \underline{Y} &= \underline{I}\end{aligned}\tag{2.48}$$

which may be solved to give:

$$\begin{aligned}\text{real part } \underline{Z} &= -(\underline{A} + \omega^2 \underline{A}^{-1}) \\ \text{Imaginary part } \underline{Y} &= -\omega(\underline{A} + \omega^2 \underline{A}^{-1})^{-1} \underline{A}^{-1}\end{aligned}\tag{2.49}$$

Thus the solution by this method involves the inversion of \underline{A} followed by the inversion of $(\underline{A} + \omega^2 \underline{A}^{-1})$ for each frequency considered.

A slightly different, and more efficient form is obtained by first multiplying Eq. (2.45) by the conjugate of $(i\omega \underline{I} - \underline{A})$:

$$\begin{aligned}(i\omega \underline{I} - \underline{A}) \underline{x}(i\omega) &= \underline{x}(0) \\ (\omega^2 \underline{I} + \underline{A}^2) \underline{x}(i\omega) &= -(i\omega \underline{I} + \underline{A}) \underline{x}(0)\end{aligned}\tag{2.50}$$

Eq. (2.50) requires the inversion of a matrix containing only real numbers but still requires a separate solution for each value of ω investigated. Results given later for method 2 are based on Eq. (2.50).

2.3.5 Method 3 - Frequency response via the canonical form

Eq. (2.45)

$$(i\omega \underline{I} - \underline{A}) \underline{x}(i\omega) = \underline{x}(0)$$

may be written in the canonical form by substituting Eq. (2.5) for \underline{A}

$$(\underline{U} i\omega \underline{U}^{-1} - \underline{U} \underline{A} \underline{U}^{-1}) \underline{x}(i\omega) = \underline{x}(0)\tag{2.51}$$

where $\underline{\Lambda}$ and \underline{U} are the general Jordan form and transformation matrices respectively. Eq. (2.51) may also be written

$$\underline{U}(i\omega \underline{I} - \underline{\Lambda})\underline{U}^{-1} \underline{x}(i\omega) = \underline{x}(0) \tag{2.52}$$

$$(i\omega \underline{I} - \underline{\Lambda})\underline{U}^{-1} \underline{x}(i\omega) = \underline{U}^{-1}\underline{x}(0) \tag{2.53}$$

This equation is readily solved without resort to matrix inversion for $(i\omega \underline{I} - \underline{\Lambda})^{-1}$ may be written out directly.

It was shown in section 2.1 that $\underline{\Lambda}$ has the general form

$$\underline{\Lambda} = \begin{bmatrix} \lambda_1 & & & & \\ & \lambda_2 & & & \\ & & \lambda_3 & \phi_3 & \\ & & -\phi_3 & \lambda_3 & \\ & & & & \lambda_4 & 1 & 0 \\ & & & & & \lambda_4 & 1 \\ & & & & & & \lambda_4 \end{bmatrix} \tag{2.54}$$

and thus $(i\omega \underline{I} - \underline{\Lambda}) =$

$$\begin{bmatrix} (i\omega - \lambda_1) & & & & \\ & (i\omega - \lambda_2) & & & \\ & & (i\omega - \lambda_3) & -\phi_3 & \\ & & \phi_3 & (i\omega - \lambda_3) & \\ & & & & (i\omega - \lambda_4) & -1 & 0 \\ & & & & & (i\omega - \lambda_4) & -1 \\ & & & & & & (i\omega - \lambda_4) \end{bmatrix} \tag{2.55}$$

$\frac{1}{i\omega - \lambda_1}$					
$\frac{1}{i\omega - \lambda_2}$					
	$\frac{i\omega - \lambda_3}{(i\omega - \lambda_3)^2 + \phi_3^2}$	$\frac{\phi_3}{(i\omega - \lambda_3)^2 + \phi_3^2}$			
	$\frac{-\phi_3}{(i\omega - \lambda_3)^2 + \phi_3^2}$	$\frac{i\omega - \lambda_3}{(i\omega - \lambda_3)^2 + \phi_3^2}$			
			$\frac{1}{i\omega - \lambda_4}$	$\frac{-1}{(i\omega - \lambda_4)^2}$	$\frac{1}{(i\omega - \lambda_4)^3}$
				$\frac{1}{i\omega - \lambda_4}$	$\frac{-1}{(i\omega - \lambda_4)^2}$
					$\frac{1}{i\omega - \lambda_4}$

The solution of Eq. (2.45) is then

$$\underline{x}(i\omega) = \underline{U}(i\omega \underline{I} - \underline{\Lambda})^{-1} \underline{U}^{-1} \underline{x}(0) \quad (2.57)$$

where $(i\omega \underline{I} - \underline{\Lambda})^{-1}$ is obtained by the above method without recourse to any inversion at all for the considered frequencies. Eq. (2.56) could alternatively have been found by taking the Laplace transform of Eqs. (2.13, 2.14, 2.19, 2.21), each of the forms of $\underline{\Lambda}$ considered earlier.

Eq. (2.57) is easily programmed and $\underline{x}(i\omega)$ can be found from Eq. (2.56) using only a small amount of complex arithmetic, and because the size of each block in the Jordan matrix is known $(i\omega \underline{I} - \underline{\Lambda})^{-1} \underline{x}(0)$ is best found without using matrix multiplication routines.

2.3.6 Comparison of methods 1 - 3

Methods 1 - 3 have each been applied to a series of problems (40) and the core store used and the time taken recorded. 33 different frequencies were considered. This data is shown in Table 2.6. (Times are shown in mill units, 1 mill $\sim \frac{1}{2}$ sec).

Matrix order	Method 1		Method 2		Method 3	
	store	time	store	time	store	time
10	4928	21	5184	16	11584	14
20	6848	95	6528	45	14400	33
30	9984	209	8640	82	18176	62
40	14272	409	11520	142	23872	111

Table 2.6 Storage and run-times for methods 1, 2 and 3

The time taken is obviously a function of how many frequencies are considered. For methods 2 and 3 some is required to start the sequence and the remaining time (and the time for method 1) is directly proportional to the number of frequencies considered. The run time/frequency is shown in Table 2.7 and has been plotted in Fig. 2.4.

Matrix order	Method 1		Method 2		Method 3	
	start time	time/freq	start time	time/freq	start time	time/freq
10	-	.64	2	.42	6	.24
20	-	2.88	5	1.21	14	.58
30	-	6.33	10	2.18	32	.91
40	-	12.39	20	3.70	65	1.39

Table 2.7 Timings for methods 1,2, and 3

The start time for each method is also shown in Table 2.7 and plotted in Fig. 2.5. From Figs. 2.4 and 2.5 the following time estimates for each of the methods have been calculated.

$$t_1 = .0055 FN^{2.13}$$

$$t_2 = .0139 N^{1.96} + .0116 FN^{1.56}$$

$$t_3 = .0185 N^{2.21} + .013 FN^{1.26}$$

where F is the number of frequencies considered and N is the system order.

Methods 1 and 2 have given identical results for all cases tried (up to order 40), however, method 3, although much faster than the others, especially when many

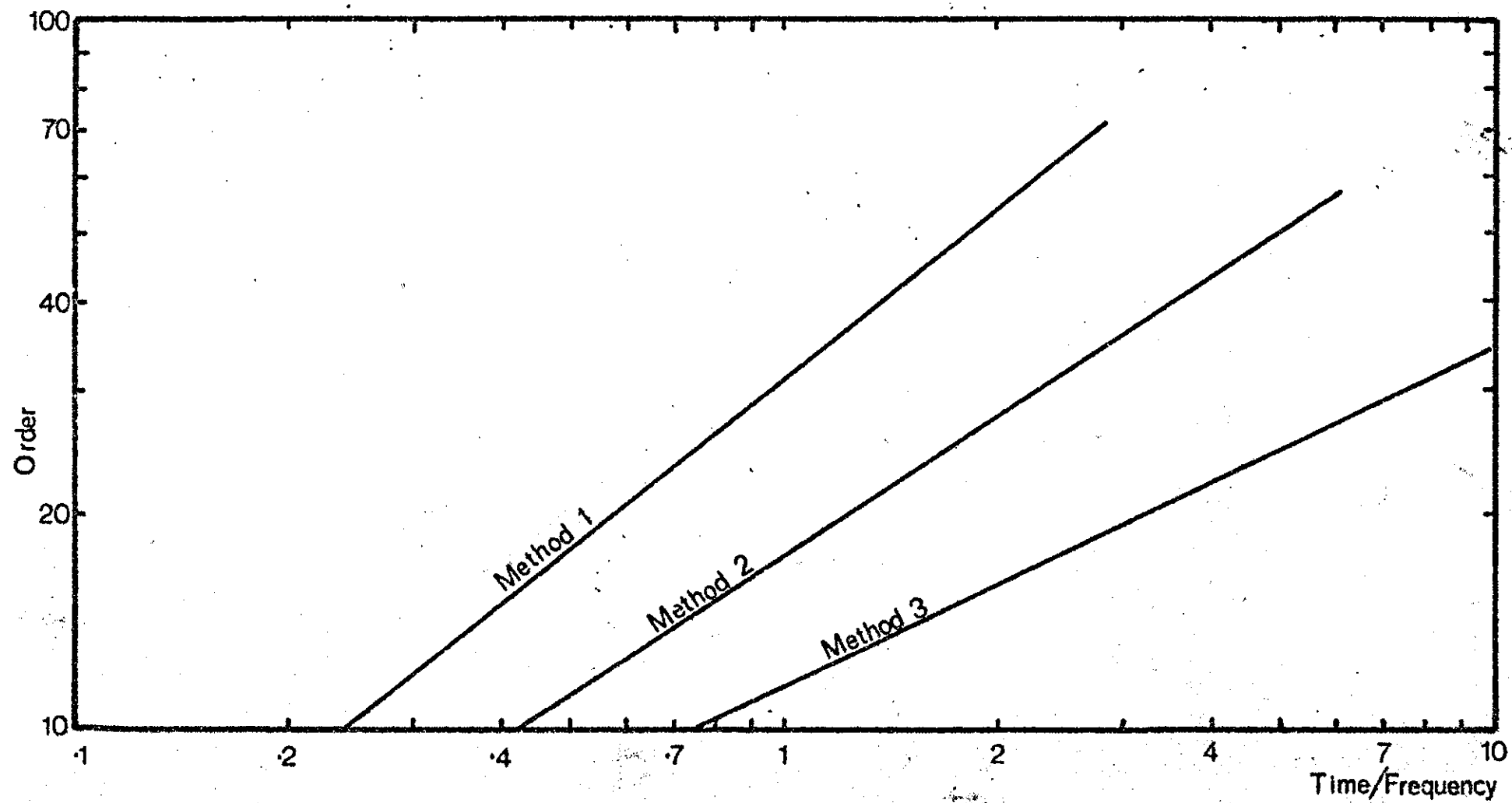


Figure 2.4 Mill Units/Frequency vs. order for system with real
eigenvalues

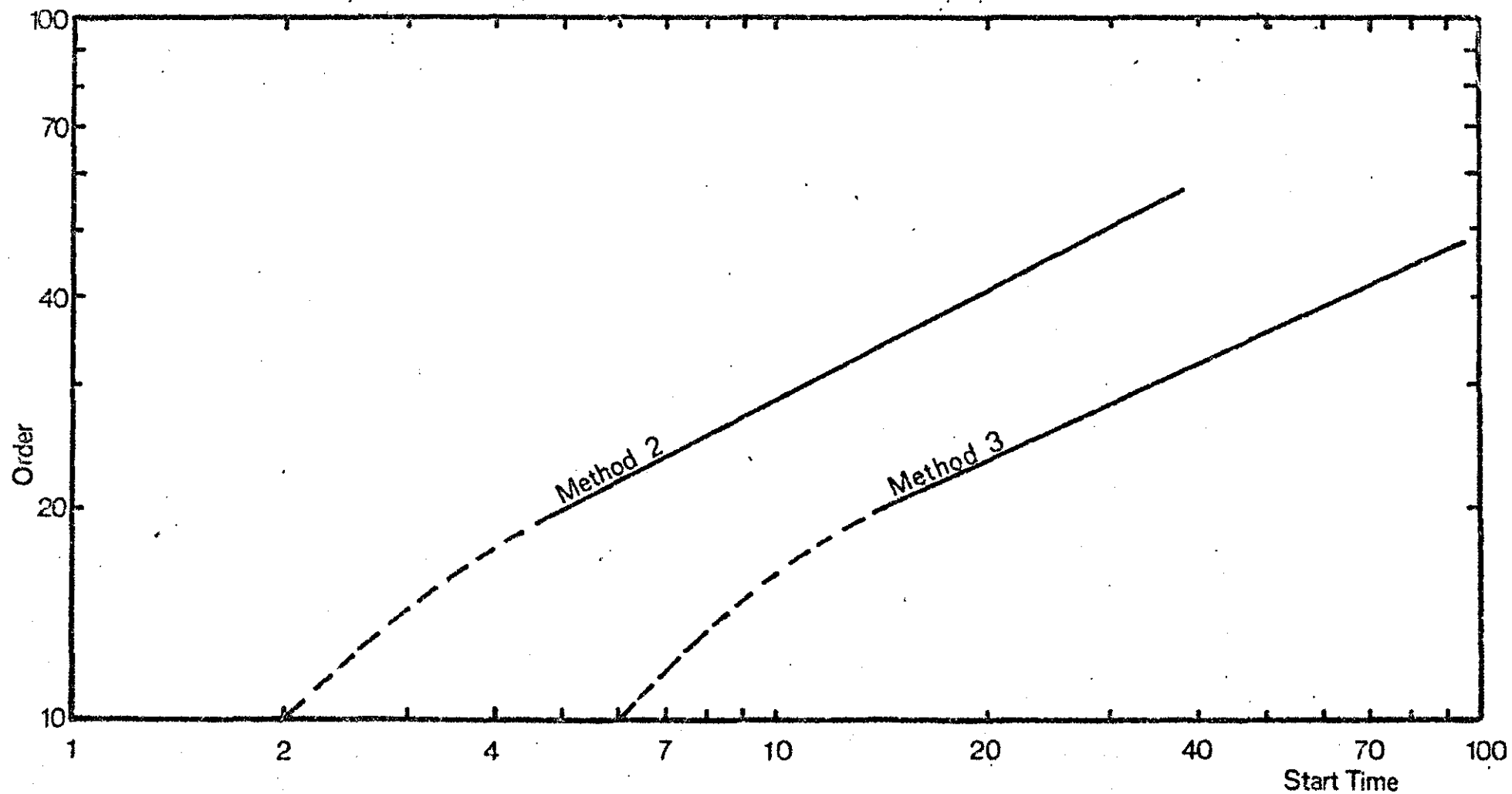


Figure 2.5 Start time in mill units vs. order for systems with real eigenvalues

frequencies are being investigated, has given results which show a numerical error, due to rounding effects when ill-conditioned systems with a wide spread of eigenvalues are being investigated. Table 2.8 shows this effect for a 10th order system where the largest and smallest eigenvalue are a factor of 10^4 different. Method 3 has, however, given satisfactory results for the well conditioned systems investigated.

Plant eigenvalues

-0.984150E 03 -0.328050E 03 -0.109350E 03 -0.364500E 02
 -0.121500E 02 -0.405000E 01 -0.135000E 01 -0.450000E 00
 -0.149999E 00 -0.495006E 01

Methods 1 and 2			Method 3		
State	real part	imaginary part	State	real part	imaginary part
1	0.374159E-04	-154691E-04	1	0.374016E-04	-154691E-04
2	0.147962E-04	-250930E-05	2	0.147750E-04	-450929E-05
3	0.503280E-05	-308198E-06	3	0.503187E-05	-308198E-06
4	0.167672E-05	-351739E-07	4	0.167668E-05	-351739E-07
5	0.558112E-06	-394004E-08	5	0.558111E-06	-394004E-08
6	0.185914E-06	-440868E-09	6	0.185914E-06	-440868E-09
7	0.619563E-07	-498459E-10	7	0.619563E-07	-498459E-10
8	0.206526E-07	-582361E-11	8	0.206526E-07	-582361E-11
9	0.316626E-07	-276529E-11	9	0.316626E-07	-276529E-11
10	0.101611E-02	-210220E-02	10	0.101611E-02	-210220E-02

Table 2.8 Numerical comparisons of methods
1, 2 and 3

2.4 DETERMINATION OF THE MOMENTS OF STATE VARIABLE MODELS

A response curve may be characterized by statistical parameters derived from it. The parameters usually calculated are the area under the impulse response curve, its mean, variance and skewness. These may alternatively be called the zeroth, first, second and third moments.

The i th unnormalised moment about the origin of an impulse response is defined:

$$M'_i = \int_0^{\infty} t^i f(t) dt \quad (2.59)$$

Moments may, if required, be normalized with respect to the zeroth moment (the area under the curve) and be taken about the mean. Expressions relating the normalized moment to unnormalized are given by Gibilaro and Lees (54). The Laplace transform of the same response $f(t)$ is defined

$$G(s) = \int_0^{\infty} e^{-st} f(t) dt \quad (2.60)$$

Differentiating Eq. (2.60) with respect to s gives

$$\frac{dG(s)}{ds} = - \int_0^{\infty} t e^{-st} f(t) dt \quad (2.61)$$

which in the limit as s approaches zero is $-M'_1$, the first moment given by Eq. (2.59); or more generally

$$\left[\frac{dG(s)}{ds^i} \right]_{s=0} = (-1)^i M'_i \quad (2.62)$$

Thus it is shown that the moments of the response of $f(t)$ may be derived directly from $G(s)$, the transfer function

giving that response (or the Laplace transform of the response).

Lees (75) and Gibilaro and Lees (54), extending the work of Paynter (103) have shown how application of Eq. (2.63) to low-order transfer functions gives relatively simple expressions relating the moments to the transfer function parameters. Similarly Kropholler (68) has applied Eq. (2.62) to the transformed state variable model, Eq. (2.44).

$$(s\mathbf{I} - \mathbf{A}) \mathbf{x}(s) = \mathbf{x}(\infty) \quad (2.44)$$

Repeated differentiation with respect to s and solving in the limit $s \rightarrow 0$ gives

$$\begin{aligned} \mathbf{M}'_{x_i} &= -i\mathbf{A}^{-1} \mathbf{M}'_{x_{i-1}} \quad i = 1, 2, \dots \\ \mathbf{M}'_{x_0} &= -\mathbf{A}^{-1} \mathbf{x}(0) \end{aligned} \quad (2.63)$$

where \mathbf{M}'_{x_i} is a vector containing the i th unnormalized moments of \mathbf{x} . Eq. (2.63) may alternatively be written

$$\mathbf{A} \mathbf{M}'_{x_i} + \mathbf{B} \phi_{j,i} = -\mathbf{M}'_{x_{i-1}} \quad (2.64)$$

where the elements ϕ_k of the input vector $\phi_{j,i}$ are given by

$$\begin{aligned} \phi_k &= 0 & k &\neq j & i &\geq 0 \\ \phi_k &= 1 & k &= j & i &= 0 \\ \phi_k &= 0 & k &= j & i &> 0 \end{aligned} \quad (2.65)$$

where j is the input to the system.

It has been shown that any number of moments may be computed, by repeated application of Eq. (2.63), with very little effort: one matrix inversion of \mathbf{A} being all

that is required.

2.5 COMPUTER SYSTEM USED

Most of the computational work included here was carried out on an ICL 1904A computer, although some of the initial work was done on a 1905 machine.

The 1904A has a core store of 128 K words (one word being 24 bits organized in 4 6-bit characters) and a magnetic drum (ICL 1964/1) of capacity 512 K words and a transfer rate of 25 K words/second. The store cycle time is 750 nanoseconds. Jobs have been run under the George II operating scheme.

The program language used throughout the work has been 1900 FORTRAN and where possible use has been made of existing routines supplied in the ICL scientific subroutine package, although a number of routines did not match their specification and gave considerable difficulties. Some ICL subroutines have been written in PLAN and run in a 15 bit address mode (compact) which means that stores higher than 32768 cannot be accessed and has hence placed a restriction of 32.5 K on most programs. In some programs this has resulted in a great deal of array movement and transfers to and from disc.

Discs of 200 K words/cartridge have been used, with a transfer rate of 52 K words/second. Tapes used had a transfer time of 10.5 K words/second. Graphs have been plotted on an ICL 1934 plotter with a step length of .005 inch.

2.6 NOMENCLATURE

<u>A</u>	plant matrix
<u>A</u> [†]	Davison's modified <u>A</u> matrix - defined by Eq. (2.22)
<u>a</u> _i	denominator coefficients of transfer function
<u>B</u>	input matrix
<u>b</u> _i	numerator coefficient of transfer functions
<u>b</u> _i	vector of numerator coefficients
<u>e</u> _i	extraneous root in Davison's method
G(s)	transfer function
<u>h</u> _i	defined by normalised coefficients of characteristic equation
<u>I</u>	identity matrix
<u>i</u>	$\sqrt{-1}$
<u>M'</u> _i	i th unnormalised moment about the origin
<u>M'</u> _{<u>x</u>_i}	vector of moments <u>M'</u> _i of <u>x</u>
<u>m</u>	order of numerator
<u>n</u>	system order
<u>P</u>	transformation matrix
<u>P</u> _i	partitioned vector of <u>P</u>
<u>p</u> _i	system pole
<u>R</u>	defined by Eqs. (2.36, 2.37)
<u>s</u>	Laplace operator
<u>T</u>	defined by Eq. (2.42)
<u>t</u> _i	defined by Eq. (2.42)
<u>t</u>	time
<u>U</u>	matrix of eigenvectors
<u>u</u> _i	eigenvector

\underline{u} forcing vector
 \underline{x} state vector
 \underline{Y} defined by Eq. (2.46)
 \underline{y} \underline{x} transformed by \underline{U}
 \underline{Z} defined by Eq. (2.46)
 z_{jk} kth zero for j the input
 (s) indicates Laplace transform - usually of
 vectors
 $(i\omega)$ indicates frequency transform - usually of
 vectors

Greek:

Γ constant
 $\underline{\Lambda}$ Jordan canonical form
 λ_k real part of eigenvalue
 τ time constant
 ϕ_k imaginary part of eigenvalue
 ω frequency

CHAPTER 3

The simplification of transfer functions

3. THE SIMPLIFICATION OF TRANSFER FUNCTIONS [14]

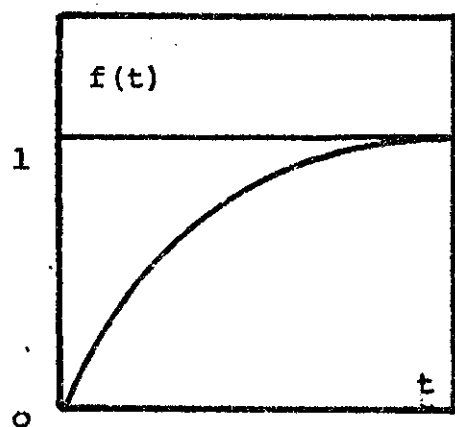
The problem of reducing the order of transfer functions has only relatively recently received attention, however, that of fitting a transfer function to experimentally generated plant data has been considered for much longer. The two problems are essentially the same. Reference will be made to a number of early methods for identifying plant data, whilst the more recent modelling techniques and specific methods for system order reduction will be discussed more fully.

Before describing simplification methods, consideration will be given to the type of response and the form of the models which are to be matched.

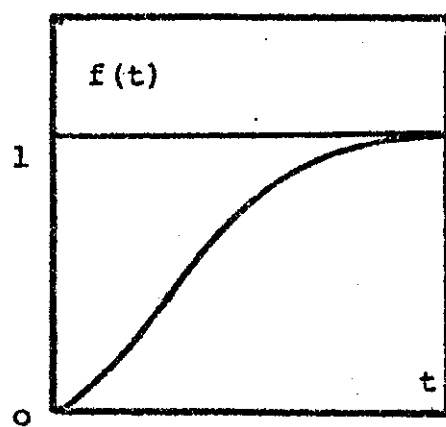
3.1 Characteristics of transfer functions

It is worthwhile considering the form of transient response, resulting from an input change, on chemical plants, as the models fitted have obviously been influenced by them. Figure 3.1 shows some of the commonly occurring step responses: they are the exponential, s-shaped, single-peak, oscillating, and inverting responses. Inspection of these responses shows that, although generated from high order models, they are similar to those given by second order systems. It is this fact that allows simplification to take place.

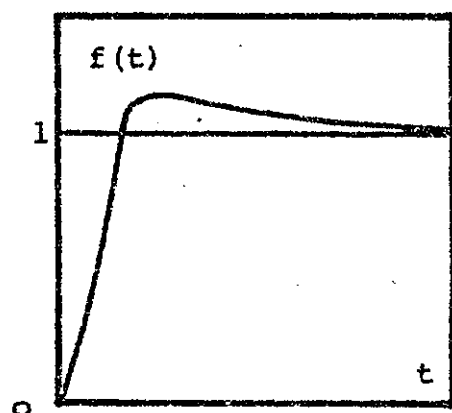
Classification of the impulse response is usually based on the mean, spread, and skewness of the curve, or its moments. Alternatively systems may be analysed via the frequency response, and classified according to the characteristics of the amplitude ratio and phase-lag.



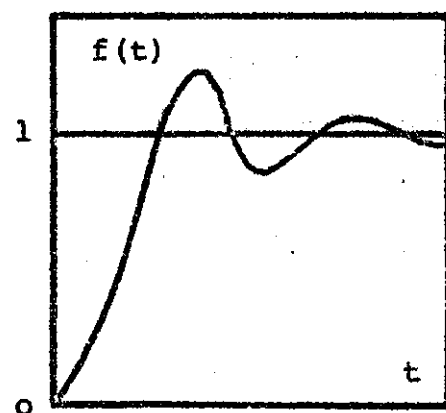
(a) exponential



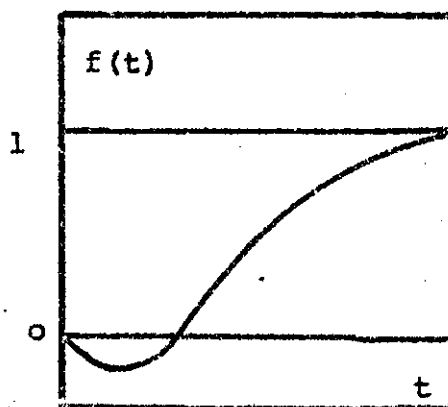
(b) s-shaped



(c) Single peak



(d) Oscillating



(e) Inverting

Figure 3.1 Typical System Step responses

Because of the fluid flows involved in chemical plants there is often a time lag between a system being forced and responding, and hence model builders have tended to include some degree of time delay.

The transfer function can take many forms, but probably the most general is:

$$G(s) = \frac{e^{-\tau s} \sum_{j=1}^m b_j s^j}{\sum_{i=1}^n a_i s^i} \quad m \leq n \quad (3.1)$$

Although this model itself is not very common the same model without the time delay has been widely used. Most processes may be represented by this model if the order of the numerator and denominator are carefully chosen. For model simplification however, it is the low order models which are of prime interest. Of particular importance are the (1,2) model

$$G(s) = \frac{K(1 + b_1 s)}{1 + a_1 s + a_2 s^2} \quad (3.2)$$

and the (2,3) model

$$G(s) = \frac{K(1 + b_1 s + b_2 s^2)}{1 + a_1 s + a_2 s^2 + a_3 s^3} \quad (3.3)$$

Models of this type, whilst retaining the same general form can represent many different responses. The type of response obtained may be defined in terms of the relations between the parameters, as shown in Table 3.1 for the (1,2) model. This model may also be written in the form:

$$G(s) = \frac{K(1 + \frac{2\alpha\zeta}{\omega_n} s)}{1 + \frac{2s\zeta}{\omega_n} + \frac{s^2}{\omega_n^2}} \quad (3.4)$$

Towill (115) has given the step and frequency responses of this model as functions of α and ζ . He has also studied the (2,3) and (3,4) models.

The transfer function may be further analysed in terms of the poles and zeroes. The general (m,n) model is

$$G(s) = \frac{K \prod_{j=1}^m (s - z_j)}{\prod_{i=1}^n (s - p_i)} \quad m \leq n \quad (3.5)$$

The characteristics of low order models in terms of the poles and zeroes have been described by Towill (115). Many other special models have been used and some of the more important are given in Table 3.2.

3.2 Classification of reduction methods

The determination of a low order transfer function which is equivalent to a higher order model involves both choice of model form and calculation of the parameters. Although interesting, many of the early modelling methods are not readily amenable to automatic computation, and are thus not particularly important. All of the satisfactory methods involve a large amount of arithmetic. However, some of the early graphical methods have been

Table 3.1 (1,2) Transfer function parameters

<u>Parameters</u>	<u>System</u>
$b_1 = 0$ $a_1^2 = 4a_2$ $a_1^2 > 4a_2$ $a_1^2 < 4a_2$	critically damped overdamped underdamped
$b_1 < 0$ $a_1^2 > 4a_2$ $a_1^2 < 4a_2$	inverting inverting and oscillatory
$\tau_2 < b_1 < \tau_1$ $1 + a_1s + a_2s^2 = (1 + \tau_1s)(1 + \tau_2s)$ all coefficients positive and not covered by above. a_1 or a_2 negative	side capacity combination of lead and lag terms unstable

Table 3.2 Simple transfer function models

Model	Model No.	No. of Parameters	Constraints	System	Reference
$\frac{Ke^{-\tau_1 s}}{1 + \tau_2 s}$	0	3	-	-	115
$\frac{Ke^{-\tau_1 s}}{(1 + \tau_2 s)(1 + \tau_3 s)}$	1	4	-	-	54
$\frac{Ke^{-\tau_1 s}}{(1 + \tau_2 s)^n}$	2	4	n need not be an integer	-	54
$\frac{Ke^{-\tau_1 s}}{1 + \frac{2\zeta s}{\omega_n} + \frac{1}{\omega_n^2} s^2}$	3	4	$\zeta < 1$ $\zeta = 1$ $\zeta > 1$	underdamped critically damped overdamped	76
$\frac{K(1 + \eta)}{1 + \tau_1 s} - \frac{\eta}{1 + \tau_2 s}$	4	4	$\frac{\tau_1}{\tau_2}, \frac{1 + \eta}{\eta} > 1$	inverting	60
$\frac{K(1 + \tau_1 s)}{(1 + \tau_2 s)(1 + \tau_3 s)}$	5	5	$\tau_3 < \tau_1 < \tau_2$	side capacity	81
$\frac{Ke^{-\tau_1 s}}{(1 + \frac{2\zeta s}{\omega_n} + \frac{1}{\omega_n^2} s^2)(1 + \tau_2 s)}$	6	4	$\zeta < 1$ $\zeta = 1$ $\zeta > 1$	underdamped critically damped overdamped	58

Table 3.2 : Cont'd ...

Model	Model No.	No. of Parameters	Constraints	System	References
$\frac{K e^{-\tau_1 s} (1 + \tau_2 s)}{(1 + \tau_3 s)(1 + \tau_4 s)} -$	7	5	$\tau_4 < \tau_2 < \tau_3$	side capacity	
$\frac{K(1 - \eta e^{-\tau_1 s})}{(1 + \tau_2 s)(1 + \tau_3 s)} -$	8	5	$\eta < 1$	distributed parameter system with distributed parameter forcing	58

updated and programs written (38).

Available techniques may be split broadly into two groups:

- a) models obtained by fitting parameters to data generalized by the complex model.
- b) operations directly involving the complex model.

3.3 SIMPLIFICATION VIA THE TIME RESPONSE

Methods based on fitting simple models to the computed time response of the full model are identical to those used in identifying real processes. The application of these methods is, however, easier in so far as there is no process noise to contend with.

There are a number of quite well known specific techniques for fitting particular predetermined models, such as first or second order systems and equal-stages-in-series systems, to step responses (46, 58, 61, 96). Fitting is normally effected by comparing the normalised response with standard sets of responses for different parameters and interpolating for the best fit. Dead time is matched by determining the displacement of the response on the time scale. Similar methods are available for fitting underdamped second order systems to oscillating responses (46, 61). The parameters of such systems can be found from the period between adjacent peaks and the peak-height decay ratio.

More general methods of fitting a transfer function to the step responses have been given, including the approximation of the response by a series of ramp functions

and time delays (96), and the derivation from the response of a continued fraction expansion of the fitted transfer function (107). The latter method will be discussed in Section 3.6.

There is a large literature on the fitting of simple models to impulse responses, the main technique being the matching of moments (18, 53, 62, 67). This will be considered later.

Since the digital computer has become widespread the methods described above have tended to be replaced by more numerate methods, usually relying upon a least-squares fit to some response. Sinha and co-workers (2, 110) have developed two methods for fitting low order models to the step response of the full system. A different criteria was used for each.

3.3.1 Sinha and Pille's method (110)

Sinha and Pille minimize the mean square error between samples of the two step responses taken over a given time interval:

$$\text{i.e. Minimize } J = \frac{1}{N} \sum_{i=1}^N (f_i(t) - f_i^*(t))^2 \quad (3.6)$$

Practically this means setting up a least squares problem, relating input to output data at different sampling points, the solution of which is the parameters of the pulse transfer function. This transform is then converted to the continuous time model. Weighting may be effected by neglecting data points in intervals where

a good fit is not required. The method is essentially the same as that given by Anderson (2) for simplifying state variable models although it has been modified, by using a recursive algorithm, to avoid repeated matrix inversion and the storage of vast amounts of data, thus giving computational advantages. Chidambara (31) has given a different least-squares method.

3.3.2 Sinha and Bereznai's method (102)

The method described above minimizes the error between the two curves at discrete points in time (see Fig. 3.2). This has the effect of producing large errors on the rapidly changing portion of the response and much smaller errors at the critical portions: the peak overshoot and steady state. Resulting from this bias, the large unimportant error is reduced at the expense of small errors at critical regions of the curve.

Sinha and Bereznai have proposed the alternative error criteria of placing an upper and lower bound, α , on the response throughout its length, thus reducing the emphasis on the rapidly changing portion of the transient response, to give a better overall approximation. The error criteria becomes

$$\text{Minimize } J = \text{Max}_{i=1,N} \cos \theta_i |f_i(t) - f_i^*(t)| \quad (3.7)$$

$$\theta_i = \tan^{-1} \frac{f_{i+1}(t) - f_i(t)}{2S} \quad (3.8)$$

where S is the uniform sampling interval. The minimization of the minimax error criteria is performed using

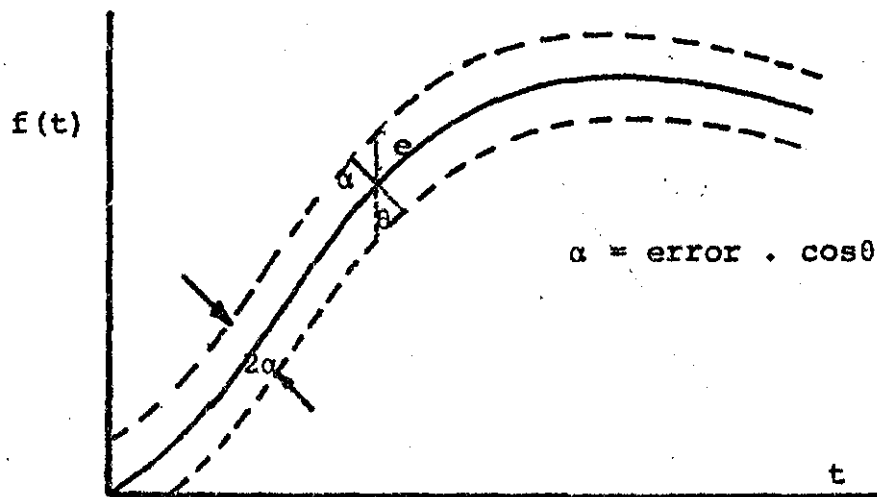
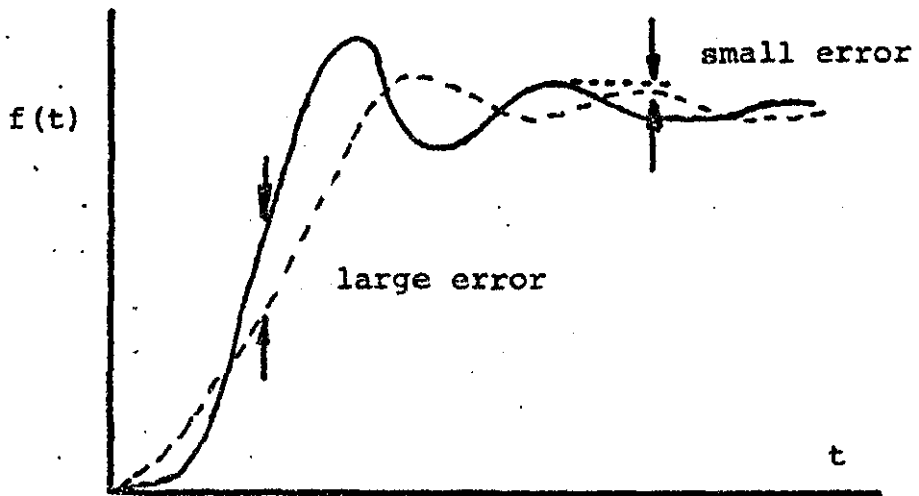


Figure 3.2 Sinha and Berenzai's error criteria

the Hooke and Jeeves pattern search. The start values for the model parameters in the pattern search are predicted by initially fitting a low order model to the response, using the classical methods described earlier. The optimum set of parameters are found and the model order increased by one until the optimum model of the required order is found.

3.4 SIMPLIFICATION VIA THE FREQUENCY RESPONSE.

As with identification in the time domain, many methods have been suggested in the past twenty years for obtaining a plant model from its frequency response. Some of these methods (many are graphical) are adequate, where low order models of a process are required and high accuracy is not necessary, but are not easily adapted to computational algorithms. Strobel (112) has listed the known methods.

Perhaps the best known method for obtaining a transfer function from the frequency response is that of Bode and Truxal (117) where corner frequencies are identified from the Bode plot. Other graphical methods are those of Ausman (8) and Linvill (80). Bode's method has been updated by Cowherd and Cadman (38) who give a computational algorithm for predicting each of the time constants in a lead-lag model iterating from the simple corner frequencies. A similar but more sophisticated approach has been outlined by Towill and Mehdi (116) in which dominant roots are monitored and a simple root substituted for all those neglected.

Dudnikov (45) has given an explicit method based on the expansion of the polynomial ratio transfer function into a continued fraction and the determination of the ensuing coefficients from a series of charts relating the real and imaginary parts. The method has been described by Naslin (96) .

Young (123) has shown how the coefficients of a transfer function with only denominator dynamics may be obtained by numerically differentiating the real and imaginary parts of the frequency response until a constant difference is obtained. Applying the same differentiation to the transfer function relates the differences to the coefficients. Process noise often means that a constant difference will not be obtained: in this case the best approximation to the experimental points is used.

Chen and Philip (26) have proposed a method related to the Bush decomposition of a polynomial. The transfer function is considered to be made up of a series of feedback loops, each loop increasing the function order by one. In turn each loop is made "open circuit" to give a lower order model which is identified from the Bode plot. The method may be programmed and is similar to that given by Chen and Knox (24) for identifying systems from the time response.

A number of schemes have been given which minimize the error between the given frequency response and that of the fitted model. Amongst the methods are the

Wiener-Lee decomposition [25] and those of Meier and Luenberger [87], Sumner [113], Levy [79], Kardashov [64] and Kalyaev [63].

3.4.1 Meier and Luenberger's method [81,87]

Analogies are drawn between the model reduction problem and the modelling of a Wiener filter. The reduced model of order m

$$G^*(s) = \sum_{k=1}^m \frac{r_k}{s + p_k} \quad (3.9)$$

is to be fitted to the frequency response, $G(s)$, by minimizing the response difference:

$$\text{i.e. Minimize } J = \frac{1}{2\pi i} \int_{-i\infty}^{i\infty} (G(s) - G^*(s))^2 ds \quad (3.10)$$

$$= \frac{1}{2\pi i} \int_{-i\infty}^{i\infty} \left[G(s) - \sum_{k=1}^m \frac{r_k}{s + p_k} \right]^2 ds \quad (3.11)$$

Clearly the necessary conditions for solution is that each of the partial derivatives $\frac{\partial J}{\partial r_k}$, $\frac{\partial J}{\partial p_k}$, $k = 1, m$ be equal to zero. Performing the above differentiation leads to sets of non-linear equations, the parameters of which are the system poles and residues. These equations can be solved by a number of numerical methods, although a degree of engineering judgement is required to select initial values to ensure rapid convergence.

3.4.2 Levy's method [79]

A further method for the least-squares fit in the complex plane has been given by Levy. However, unlike Meier and Luenberger his formulation leads to a set of linear equations.

Let the frequency dependent polynomial ratio to be fitted to the generated data $G(i\omega)$ be

$$G^*(i\omega) = \frac{B(i\omega)}{A(i\omega)} = \frac{b_0 + b_1(i\omega) + b_2(i\omega)^2 + \dots}{a_0 + a_1(i\omega) + a_2(i\omega)^2 + \dots} \quad (3.12)$$

The error in the fit at frequency ω_k is

$$\epsilon_k = G(i\omega_k) - \frac{B(i\omega_k)}{A(i\omega_k)} \quad (3.13)$$

Levy has overcome the problem of minimizing the sum of all ϵ_k^2 by multiplying Eq. (3.13) by $A(i\omega_k)$ and separating the right-hand side of the resulting equation into real and imaginary parts and squaring the absolute value.

$$|A(i\omega_k)\epsilon_k|^2 = R_k^2 + I_k^2 = f\{G(i\omega_k), a_0, a_1, a_2, \dots, b_0, b_1, b_2, \dots\} \quad (3.14)$$

An error function is now defined based on Eq. (3.14) summed over all r frequencies considered

$$J = \sum_{k=1}^r (R_k^2 + I_k^2) \quad (3.15)$$

The method of least-squares is then applied, with J being differentiated with respect to each of the parameters $b_0, b_1, b_2, \dots, a_1, a_2, \dots$ (a_1 being set to unity).

The method has been widely used and is capable of fitting some quite unusual frequency responses (23), but several difficulties have been noted in the method. Sanathanan and Koerner (106) found that the procedure described does not give a good fit if the frequency data spans several decades and have proposed instead, an iterative procedure in which Eq. (3.14) is modified by writing

$$\begin{aligned} \left| \frac{A(i\omega_k)_p \epsilon_k}{A(i\omega_k)_{p-1}} \right|^2 &= \left| \frac{A(i\omega_k)_p G(i\omega_k)}{A(i\omega_k)_{p-1}} - \frac{B(i\omega_k)_p}{A(i\omega_k)_{p-1}} \right|^2 \\ &= R_k^2 + I_k^2 \end{aligned} \quad (3.16)$$

where subscript p refers to the iteration number. The same method as used by Levy is then applied. In the first iteration $A(i\omega_k)_{p-1}$ is set to unity, which corresponds exactly to Levy's approach, but in succeeding iterations it is set equal to the previously calculated value. The authors give an example in which the iteratively computed parameters differ considerably from those obtained by the Levy method. These authors also draw attention to the numerical difficulties which may occur, since the least-squares estimations are often nearly singular.

Sumner (113) has made the same criticism of the Levy method but proposes a technique in which the error is normalised with respect to $|G(i\omega)|$ and the solution found using Davidon's method.

A further modification made to the method by Payne (102) is to incorporate certain constraints based on additional knowledge of the system, such as steady state gain (represented by coefficient b_0) and zero error to a ramp function. He notes that in the unconstrained system there is a tendency for poles to occur in the right hand half of the complex plane and so give unstable responses in systems known to be stable. He reports in fact that in fitting several hundred responses about 30% resulted in unstable systems when no constraints were used, but that the use of constraints reduced this to 1%.

Levy has also pointed out that the formulation of the problem does not permit the fitting of data that might possess a pole at the origin, but gives a solution to the problem.

The Levy technique is originally a technique for fitting rather than simplifying models, but in fact rarely fits a model of higher order than strictly necessary, so that it may be regarded as a simplification method also.

3.5 DOMINANT ROOTS RETENTION

Simple methods of reducing the order of transfer functions are often based on discarding the less important time constants. If the high order model is of the form

$$G(s) = \frac{\prod_{j=1}^m (1 + \tau_j s)}{\prod_{i=1}^n (1 + \tau_i s)} \quad m \leq n \quad (3.17)$$

it may be reduced by retaining only the dominant time constants in the following manner

$$G^*(s) = e^{-\tau s} \frac{\prod_{j=1}^{m-p} (1 + \tau_j s)}{\prod_{i=1}^{n-q} (1 + \tau_i s)} \quad m-p \leq n-q \quad (3.18)$$

where τ is calculated from the discarded time constants using the Matsubura equivalent time delay method (85).

$$\tau = \prod_{i=n-q+1}^n \tau_i - \prod_{j=m-p+1}^m \tau_j \quad (3.19)$$

Besides retaining the dominant modes the method matches the first moment of the two models and assumes that this adequately takes into account the neglected small time constants.

The method of dominant mode retention has been further developed for a computer solution by Nagarajan (95). The largest and smallest poles of the n order system are computed and if

$$\text{largest root} \geq \text{smallest root} \times K$$

the largest root is divided out of the characteristic equation, reducing it to order $n-1$. This procedure is repeated until a system of the desired order is obtained.

The best value of K, known as the range ratio, is reported to be 25. Once obtained, the parameters of the model of required order are modified to predict an "optimum model". The model is optimum in that the feedback error to a step input is minimised with respect to the feedback error of the full model to the same input. The method as described is rather restricting in that it is suitable for models with only denominator dynamics.

3.6 CONTINUED FRACTION EXPANSION AND TRUNCATION

A powerful method for the reduction of high order transfer functions is that developed by Chen and Shieh (27, 29) based on expanding the model into a continued fraction and truncating this to yield a lower order model. The property of the continued fraction is that it converges faster than other series expansions and furthermore contains most of the important system characteristics in the first few terms (74). The method is well suited to automatic computation. The transfer function

$$G(s) = \frac{b_0 + b_1s + b_2s^2 + \dots}{a_0 + a_1s + a_2s^2 + \dots} \quad (3.20)$$

may be expanded into the continued fraction

$$G(s) = \frac{1}{h_1 + \frac{1}{h_2/s + \frac{1}{h_3 + \frac{1}{h_4/s + \dots}}}} \quad (3.21)$$

where the coefficients h_i may be derived by application of the Routh algorithm [118]. Let Eq. (3.20) be written as

$$G(s) = \frac{A_{21} + A_{22}s + A_{23}s^2 + \dots}{A_{11} + A_{12}s + A_{13}s^2 + \dots} \quad (3.22)$$

and the following Routh array formed

$$\begin{array}{ccccccc} A_{11} & A_{12} & A_{13} & A_{14} & \dots & & \\ A_{21} & A_{22} & A_{23} & A_{24} & \dots & & \\ A_{31} & A_{32} & A_{33} & \dots & \dots & & \\ A_{41} & A_{42} & A_{43} & \dots & \dots & & \\ A_{51} & A_{52} & \dots & \dots & \dots & & \\ \dots & \dots & \dots & \dots & \dots & & \end{array}$$

where $A_{j,k} = A_{j-2,k+1} - h_{j-2} \cdot A_{j-1,k+1} \quad (3.23)$

and from the array the coefficients are formed such that:

$$h_i = \frac{A_{i,1}}{A_{i+1,1}} \quad (3.24)$$

Chen and Shieh have also shown how h_i may be derived from long division. Shieh, Chen and Huang [107] have shown how the coefficients of the continued fraction can be derived directly from a time or frequency response. The methods are basically the same as those of Chen and Philip [26] and Chen and Knox [24] discussed earlier.

Once Eq. (3.21) has been formed simplification is effected by prematurely truncating the fraction (after h_{2m} to obtain m order system), and from this reforming the reduced order transfer function. This may be done

by applying the Routh algorithm in reverse (28) or by other methods (30) .

Failure of this method occurs if at any time in the process coefficients $A_{i,1}$ and $A_{i,j}$ become equal to elements below them in the $i + 1$ row (in the simplest case $A_{11} = A_{21}$ and $A_{12} = A_{22}$). Neale has shown, however, that this can be overcome by changing the form of Eq. (2.21) (97).

The technique has the same drawback as Levy's method in that it is possible to produce a reduced model which has an unstable response even though the full model is stable. Chuang (35) has proposed that in this case a different form of continued fraction should be used. His method also gives an improved initial response.

Comparison of the poles of the original and reduced models shows that it is only the dominant roots, slightly modified, which are retained. *

The continued fraction simplification has also been explored by Gaisyenyuk (51) although the equations given are not general and refer only to a 4th order model. Different methods have been used for determining the continued fraction coefficients. Gaisyenyuk has used the method of Viskovatov, given by Khavonskii (66). However, Akin (1) has demonstrated the similarities between the two approaches.

3.7 SIMPLIFICATION VIA THE MOMENTS

A computationally more economic method is to identify a set of functions which are characteristic of the full

model and which can be calculated directly, without computation of the time or frequency responses, and to match these functions to the simple transfer function by a suitable choice of parameters in the latter. The principal method of this type is the matching of the moments of the impulse response.

There is quite a large literature on the fitting of simple models to experimentally determined moments (18, 53, 62, 67). This is straightforward in principle but tailing and noise may cause difficulties in fitting. The determination of low order from high order transfer functions by matching the lower moments of the impulse response was first suggested by Paynter (103).

The unnormalised moments M_i' of the impulse response, $f(t)$, taken about the origin are:

$$M_i' = \int_0^{\infty} t^i f(t) dt \quad i \geq 0 \quad (3.25)$$

The Laplace transform of the impulse response, which is the transfer function, is given by

$$G(s) = \int_0^{\infty} e^{-st} f(t) dt \quad (3.26)$$

Expanding the exponential term in Eq. (3.26) and applying the definition of moments, Eq. (3.25) provides a relationship between the transfer function and the moments

$$G(s) = M_0' - M_1' s + \frac{M_2' s^2}{2!} - \frac{M_3' s^3}{3!} + \dots \quad (3.27)$$

It is sometimes more convenient to use cumulants rather than moments. The relation between the transfer

function and the cumulants, C_i , is:

$$\ln\{G(s)\} = \ln(K) - C_1 s + \frac{C_2 s^2}{2!} - \frac{C_3 s^3}{3!} + \dots \quad (3.28)$$

This convenience derives from the fact that taking the logarithm of a lead-lag transfer function allows it to be expressed as a series rather than a ratio of terms.

The method used by Paynter is to expand both the full and simple transfer functions as polynomials and to match the cumulants using Eq. (3.28). A similar method has been given by Hsia [59]. An alternative approach used by Gibilaro and Lees [54] is to differentiate the transfer function with respect to s rather than expand it, utilizing the relation

$$\left[\frac{d^i G(s)}{ds^i} \right]_{s=0} = (-1)^i M'_i = (-1)^i \int_0^\infty t^i f(t) dt \quad (3.29)$$

This equation is derived directly from Eqs. (3.25 and 3.26) and gives a relation for the individual moments. Numerical values of the moments are calculated from the full model and are used to compute the values of the parameters in the simple model.

The method, as described, allows the reduction of a transfer function to a lower order, however it is often required to fit the low order model to one state in a state variable model. Lees [75, 76] has shown that the moments of the full model may be calculated directly from the state variable model. The method has been outlined in section 2.4. Direct computation of the moments by this means makes the moments method a powerful technique of model simplification.

Lees [77] has further extended the method to oscillatory and inverting systems, using models 3 and 4 of Table 3.2, and has given an algorithm for automatic selection of the model based on the fact that oscillatory systems have some negative moments while inverting systems have a normalised initial response which is negative.

The use of cumulants rather than moments has been investigated by Kropholler [68]. The relationship for individual cumulants is:

$$\left[\frac{d^i \ln G(s)}{ds^i} \right]_{s=0} = (-1)^i C_i \quad (3.30)$$

This is derived from Eq. (3.28) by differentiation. Kropholler has pointed out that in dealing with systems such as lead-lag models the use of cumulants has the advantage that it is possible to associate a part of each cumulant with numerator and a part with the denominator. He shows that cumulants may be obtained from the state variable model, not only by using Eq. (3.30) and the standard relations between moments and cumulants [65] but also from

$$C_i = (-1)^i (i-1)!! (\text{Tr}(\underline{A}^{-i}) - \text{Tr}(\underline{A}_j^{\dagger -i})) \quad (3.31)$$

where $\text{Tr}(\underline{A})$ is the trace of the plant matrix and $\underline{A}_j^{\dagger}$ is as defined by Eq. (2.29).

If the equations relating the moments, or cumulants, to the parameters of the simple model are implicit or if a larger number of moments are matched than there are parameters, the latter cannot be calculated directly and

the use of some form of minimization routine must be used. However, this usually presents little difficulty.

3.8 ILLUSTRATIVE EXAMPLE

The application of four of the main methods described is illustrated by the following example.

The simple transfer function

$$G(s) = \frac{b_0 + b_1 s}{a_0 + a_1 s + a_2 s^2} \quad (3.32)$$

is to be fitted to x_1 in the following seventh order model

$$\dot{\underline{x}} = \underline{A}\underline{x} + \underline{B}u \quad (3.33)$$

where

$$\underline{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -281350 & -3310975 & -2814271 & -853703 & -70342 & -4097 & -83364 \end{bmatrix}$$

$$\underline{B} = (0 \ 0 \ 0 \ 0 \ 0 \ 375000 \ -31333750)^T$$

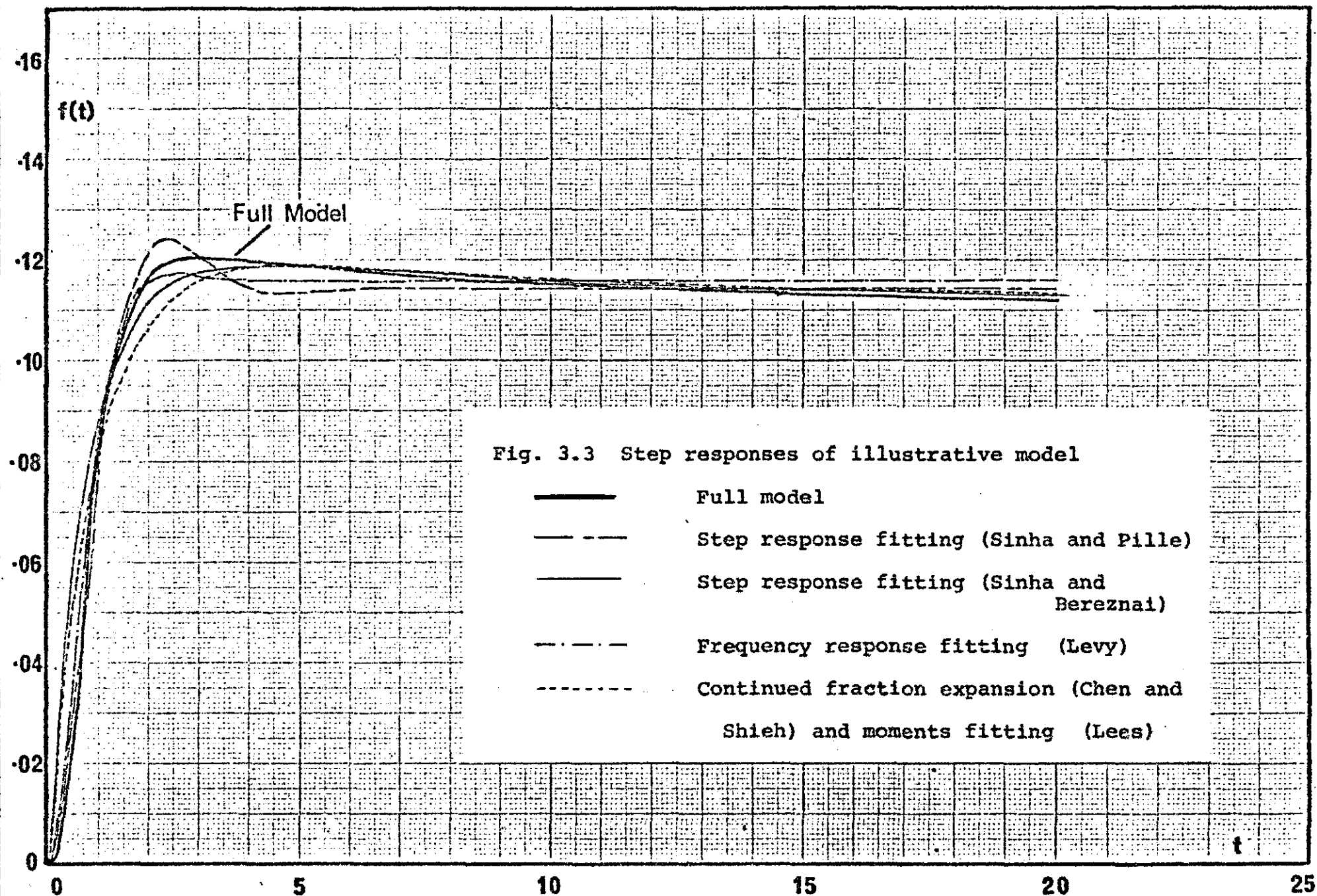
This model in its transfer function form has been considered by Sinha (109, 110). *

The values of the parameters in Eq. (3.32) obtained by various methods are shown in Table 3.3. The programs for each method and output from each computer solution are given in Appendix 1. The parameters given for the two step response fitting methods of Sinha are those quoted by that author (109, 110). The parameters for

Table 3.3 Parameters of modified (1,2) model in
illustrative example

Method	b_0	b_1	a_0	a_1	a_2
Step response fitting (Sinha and Pille)	0.3302	0	2.8886	2.0954	1
Step response fitting (Sinha and Bereznai)	0.01329	0.1536	0.1196	1.3456	1
Frequency response fitting (Levy, modified)	0.562087	0	4.83695	3.6764	1
Continued fraction expansion (Chen and Shieh)	0.0110442	0.129899	0.0993976	1.14643	1
Moments fitting (Lees)	0.0110436	0.129899	0.0993967	1.14643	1

Method	Steady State	System Poles
Sinha + Pille	.114311	$-1.4443 \pm i \cdot 0.0969456$
Sinha + Bereznai	.111120	$-.0956869 \quad -1.24991$
Levy	.116207	$-1.8382 \pm i \cdot 1.20746$
Chen + Shieh	.111112	$-.0944899 \quad -1.05194$
Full Model	.111111	See p. 67



the frequency response fitting were calculated by the method of Levy as modified by Sanathanan and Koerner. The parameters were also calculated by Chen and Shieh's continued fraction expansion and Lees' moments matching of the impulse response.

The step responses given in Table 3.3 are shown in Figure 3.3. The heavy solid line represents the response of the full model and the other lines the responses of the simplified models. The simple models given by the continued fraction and moments matching methods are identical (this will be discussed in Chapter 4). The step and frequency response fitting methods do not match the steady state exactly, but the errors are very small.

3.9 CHOICE OF MODEL AND CRITERIA OF FIT

Before applying any of the model simplification methods described consideration must be given to the form of the simple model required. Some techniques, such as those of Levy and Chen and Shieh, use general polynomial functions. The form of the model does not need to be specified in advance but the degree of reduction does. Other methods such as moments matching, do require prior choice of models, though a flexible set of models may be used with an algorithm to effect selection.

Some of the methods of simplification described involve the use of quite explicit matching criteria. Thus the step and frequency response matching methods minimize the sum of the errors squared, while the moments

matching methods match exactly, wherever possible, the lower moments of the impulse response. The continued fraction method, by contrast, is not based on any such criteria, although as will be shown in Chapter 4 it is equivalent to matching the low order model moments.

It is usually desirable that the steady state gain be matched and this presents little difficulty. The other main criteria are that the time or frequency response difference be minimized. There is no generally used method of checking the goodness of fit.

It is doubtful if much more can be said without knowing the use to which the model is to be put.

3.10 NOMENCLATURE

A	plant matrix
A_{ij}	coefficient in Chen's Routh array
a_i	denominator coefficient
b_i	numerator coefficient
C_i	Cumulant
$f(t)$	time response
$G(s)$	transfer function
$G(i\omega)$	frequency response
h_i	continued fraction coefficient
J	least-squares error
K	transfer function gain, or range ratio
M'_i	i th unnormalised moment about the origin
m	numerator order
N	number of samples in Sinha's method
n	denominator order
p_i	pole
r	number of frequencies fitted in Levy's method
S	sample interval in Sinha's method
s	Laplace operator
t	time
z_i	zero
Greek :	
α	transfer function coefficient or as defined by Fig. 3.2
ϵ	error
ζ	transfer function coefficient
θ_i	defined by Eq. (3.8)
τ	dead time
τ_i	transfer function time constant
ω	frequency

ω_n transfer function natural frequency

Superscript:

* reduced order model

T transposed matrix

CHAPTER 4

Relationships between the continued fraction
truncation and moments matching methods of
model reduction

4. RELATIONSHIPS BETWEEN THE CONTINUED FRACTION TRUNCATION AND MOMENTS MATCHING METHODS OF MODEL REDUCTION (13)

The reduction of a seventh order transfer function to a second order model was considered as an illustrative example in Section 3.8. The reduced order models predicted by the method of moments and continued fraction truncation were the same. Although apparently satisfying completely different criteria and following vastly different computational procedures three of the four parameters agreed in all six computed digits, whilst the remaining coefficient agreed to five significant figures. This unexpected agreement would suggest a previously unforeseen relationship between the two methods. This relation will be shown, in three different ways, and a generalization made which will be demonstrated numerically.

4.1 RELATION 1

4.1.1 A matrix expression for the inversion of continued fractions

A number of methods have been given for the inversion of a continued fraction (27,28,29,30) , i.e. deriving the transfer function

$$G^*(s) = \frac{b_0 + b_1s + b_2s^2 + \dots}{a_0 + a_1s + a_2s^2 + \dots} \quad (4.1)$$

from the known continued fraction coefficients, h_i .

A matrix expression has been given by Chen and Shieh [28].

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \dots\dots\dots \\ 0 & h_2 & 0 & 0 & \\ 0 & 1 & h_2 h_3 & 0 & \\ 0 & 0 & h_2 + h_4 & h_2 h_3 h_4 & \\ \vdots & & & & \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} h_1 & 0 & 0 & 0 & \dots\dots\dots \\ 1 & h_1 h_2 & 0 & 0 & \\ 0 & h_1 + h_3 & h_1 h_2 h_3 & 0 & \\ 0 & 1 & (h_1 h_2 + h_1 h_4 + h_3 h_4) & h_1 h_2 h_3 h_4 & \\ \vdots & & & & \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ \vdots \end{bmatrix} \quad (4.2)$$

For the simplified transfer function considered

$$G^*(s) = \frac{b_0 + b_1 s}{a_0 + a_1 s + a_2 s^2} \quad (4.3)$$

and noting that

$$\begin{aligned} b_2, b_3, \dots\dots\dots &= 0 \\ a_3, a_4, \dots\dots\dots &= 0 \\ a_2 &= 1 \end{aligned}$$

Eq. (4.2) may be written and partitioned as follows:

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & h_2 & 0 & 0 \\ \hline 0 & 0 & h_2 h_3 & 0 \\ 0 & 0 & h_2 + h_4 & h_2 h_3 h_4 \end{array} \right] \left[\begin{array}{c} a_0 \\ a_1 \\ 1 \\ 0 \end{array} \right] =$$

$$\left[\begin{array}{ccc|c} h_1 & 0 & 0 & 0 \\ 1 & h_1 h_2 & 0 & 0 \\ \hline 0 & h_1 + h_3 & h_1 h_2 h_3 & 0 \\ 0 & 1 & (h_1 h_2 + h_1 h_4 + h_3 h_4) & h_1 h_2 h_3 h_4 \end{array} \right] \left[\begin{array}{c} b_0 \\ b_1 \\ 0 \\ 0 \end{array} \right] \quad (4.4)$$

or

$$\left[\begin{array}{c|c} \underline{Q}_{11} & \underline{0} \\ \hline \underline{Q}_{21} & \underline{Q}_{22} \end{array} \right] \left[\begin{array}{c} \underline{a} \\ \underline{e} \end{array} \right] = \left[\begin{array}{c|c} \underline{R}_{11} & \underline{0} \\ \hline \underline{R}_{21} & \underline{R}_{22} \end{array} \right] \left[\begin{array}{c} \underline{b} \\ \underline{0} \end{array} \right] \quad (4.5)$$

4.1.2 Moments of the general polynomial transfer function

Consider the transfer function (where $a_n = 1$)

$$G(s) = \frac{b_0 + b_1 s + b_2 s^2 + \dots + b_{n-1} s^{n-1}}{a_0 + a_1 s + a_2 s^2 + \dots + s^n} \quad (4.6)$$

$$= \frac{\sum_{j=0}^{n-1} b_j s^j}{\sum_{i=0}^n a_i s^i} = \frac{B(s)}{A(s)} \quad (4.7)$$

Let Eq. (4.7) be written

$$G(s)A(s) = B(s) \quad (4.8)$$

and define

$$A_p(s) = \frac{d^p A(s)}{ds^p} \quad (4.9a)$$

$$\text{and } A_p(0) = [A_p(s)]_{s=0} \quad (4.9b)$$

and similarly for $B(s)$ and $G(s)$

It may be shown, by the Leibnitz theorem, that the p th differential of Eq.(4.8) is

$$p! \sum_{k=0}^p \frac{G_k(s) A_{p-k}(s)}{k! (p-k)!} = B_p(s) \quad (4.10)$$

$p = 0, 1, 2 \dots$

Letting $s = 0$ in Eq.(4.10), in accordance with the moments definition, gives a similar expression for the moments of Eq. (4.8).

$$p! \sum_{k=0}^p \frac{(-1)^k M'_k A_{p-k}(0)}{k! (p-k)!} = B_p(0)$$

$p = 0, 1, 2 \dots$

(4.11)

where M'_k is the k th unnormalised moment about the origin. However, expressions may also be written for the general differentials of the numerator and denominator polynomials, $B(s)$ and $A(s)$.

$$B_p(0) = p! b_p \quad (4.12a)$$

$$A_{p-k}(0) = (p-k)! a_{p-k} \quad (4.12b)$$

Substituting Eqs. (4.12) into Eq. (4.11) leads to

$$\sum_{k=0}^p \frac{(-1)^k M'_k a_{p-k}}{k!} - b_p = 0 \quad p = 0, 1, 2 \dots \quad (4.13)$$

which relates the transfer function coefficients to the moments, or vice versa, and if the moments are known allows the model parameters to be computed.

Eq. (4.13) is linear and is best expressed in matrix notation. For the second order model considered it becomes

$$\begin{bmatrix} M'_0/0! & 0 & -1 & 0 \\ -M'_1/1! & M'_0/0! & 0 & -1 \\ \hline M'_2/2! & -M'_1/1! & 0 & 0 \\ -M'_3/3! & M'_2/2! & 0 & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -M'_0/0! \\ M'_1/1! \end{bmatrix} \quad (4.14)$$

The right hand side of Eq. (4.14) is derived from the coefficient a_2 , which is unity in the second order transfer function, Eq. (4.3). The partitioning of the above equation should be noted.

4.1.3 A comparison of the two solutions

Eq. (4.5), the matrix formulation of the continued fraction inversion, may be rearranged to a form similar to the partitioned Eq. (4.14).

$$\begin{bmatrix} \underline{R}_{11}^{-1} \underline{Q}_{11} & -\underline{I} \\ \hline \underline{R}_{11}^{-1} \underline{Q}_{11} \underline{Q}_{22}^{-1} (\underline{Q}_{21} \underline{R}_{21} \underline{R}_{11}^{-1} \underline{Q}_{11}) & \underline{0} \end{bmatrix} \begin{bmatrix} \underline{a} \\ \underline{b} \end{bmatrix} = \begin{bmatrix} \underline{0} \\ -\underline{R}_{11}^{-1} \underline{Q}_{11} \underline{e} \end{bmatrix} \quad (4.15)$$

Eqs. (4.14) and (4.15) are now identical in format. From Eqs. (4.4) and (4.5) it is possible, after much tedious algebra, to write Eq. (4.15) in terms of the original coefficients, h_i .

$\frac{1}{h}$	0	-1	0	a_0	0
$\frac{-1}{h_1^2 h_2}$	$\frac{1}{h_1}$	0	-1	a_1	0
$\frac{h_1 + h_3}{h_1^3 h_2^2 h_3}$	$\frac{-1}{h_1^2 h_2}$	0	0	b_0	$\frac{-1}{h_1}$
$\frac{-(h_1^2 h_2 + h_1^2 h_4 + 2h_1 h_3 h_4 + h_3^2 h_4)}{h_1^4 h_2^3 h_3^2 h_4}$	$\frac{h_1 + h_3}{h_1^3 h_2^2 h_3}$	0	0	b_1	$\frac{1}{h_1^2 h_2}$

(4.16)

Eq. (4.14) and (4.16) are identical and allow the following expressions to be given:

$$M'_0 = \frac{1}{h_1} \quad (4.17a)$$

$$M'_1 = \frac{1}{h_1^2 h_2} \quad (4.17b)$$

$$M'_2 = \frac{2(h_1 + h_3)}{h_1^3 h_2^2 h_3} \quad ((4.17c)$$

$$M'_3 = \frac{6(h_1^2 h_2 + h_1^2 h_4 + 2h_1 h_3 h_4 + h_3^2 h_4)}{h_1^4 h_2^3 h_3^2 h_4} \quad (4.17d)$$

4.2 RELATION 2

The relations between the moments and h coefficients, Eq. (4.17), may be arrived at more directly but with a loss of clarity when trying to make extensions to a generalized case.

The moments of the second order model Eq. (4.3) may be written out from Eq. (4.13). They are:

$$M'_0 = \frac{b_0}{a_0} \quad (4.18a)$$

$$M'_1 = \frac{a_1 M'_0 - b_1}{a_0} \quad (4.18b)$$

$$M'_2 = \frac{2(a_1 M'_1 - M'_0)}{a_0} \quad (4.18c)$$

$$M'_3 = \frac{3a_1 M'_2 - 6M'_1}{a_0} \quad (4.18d)$$

Chen (27) has shown how Eq. (4.3) may be written in terms of the h coefficients.

$$G^*(s) = \frac{h_2 h_3 h_4 + (h_2 + h_4)s}{h_1 h_2 h_3 h_4 + (h_1 h_2 + h_1 h_4 + h_3 h_4)s + s^2} \quad (4.19)$$

thus

$$b_0 = h_2 h_3 h_4 \quad (4.20a)$$

$$b_1 = h_2 + h_4 \quad (4.20b)$$

$$a_0 = h_1 h_2 h_3 h_4 \quad (4.20c)$$

$$a_1 = h_1 h_2 + h_1 h_4 + h_3 h_4 \quad (4.20d)$$

Substitution of Eqs. (4.20) and (4.18) leads directly to the previously shown relationships, Eq. (4.17).

4.3 RELATION 3

The second order transfer function, Eq. (4.3) may be written as an infinite series in terms of its moments

$$G^*(s) = \frac{M'_0}{0!} - \frac{M'_1 s}{1!} + \frac{M'_2 s^2}{2!} - \frac{M'_3 s^3}{3!} + \dots \quad (4.21)$$

This may be considered to be a numerator, having a denominator of unity, and may, using the expressions given in Chapter 3 be written out into Chen's form of the Routh Array, (because Eq. (4.21) is an infinite series the Routh Array also has an infinite dimension, however, sufficient terms to calculate up to h_4 only are shown here)

1	0	0	0
M'_0	$-M'_1$	$M'_2/2!$	$-M'_3/3!$
$\frac{1}{M'_0} (M'_1)$	$\frac{-1}{M'_0} \frac{M'_2}{2!}$	$\frac{1}{M'_0} \frac{M'_3}{3!}$	
$-M'_1 + \frac{M'_0 M'_2}{M'_1 2!}$			
$\frac{-(1+M'_1)M'_2}{M'_0 2!} + \frac{M'_3}{3!}$			

Table 4.1 Routh array for Eq. (4.21)

Applying the definition for the coefficients

$$h_i = \frac{A_{i,1}}{A_{i+1,1}} \quad (4.22)$$

leads once again to expressions relating the moments to h_i , this time however, in terms of the moments.

$$h_1 = \frac{1}{M'_0} \quad (4.23a)$$

$$h_2 = \frac{M'^0_0{}^2}{M'_1} \quad (4.23b)$$

$$h_3 = \frac{M'^1_1{}^2}{-M'_0 M'^1_1{}^2 + \frac{M'^0_0{}^2 M'_2}{2!}} \quad (4.23c)$$

$$h_4 = \frac{-M'_0 M'_1{}^2 + M'_0{}^2 \frac{M'_2}{2!}}{-M'_1 \frac{(1+M'_1)M'_2}{2!} + \frac{M'_0 M'_1 M'_3}{3!}} \quad (4.23d)$$

4.4 A GENERALIZATION OF THE RELATION

It was shown numerically in section 3.8 that the two methods of model simplification gave the same results when reducing a seventh order model to second order, and it has been demonstrated above why this must be so. Thus it has been shown that the continued fraction method, besides satisfying its criteria of retaining the dominant terms of an expansion, also has the physical significance of matching exactly the lower moments of the impulse response of the full and reduced models. The question arises, however, is this a special case or will all order reductions give identical results?

The answer to the above question, as far as is known, is that it is not a special case and is true for all orders. It has however, been extremely difficult to prove this rigorously. The second order case has been demonstrated for simplicity but much of the tedious algebra has been omitted. The third order case, which has been analysed and proven, is considerably more complex. It has therefore been impossible to set up an inductive proof.

From the symmetry of the problem and the systematic nature of the equations (e.g. Eqs. (4.4), (4.14), and the

Routh Array), and from many manipulations of them it is apparent that such explicit relations will exist for all orders of reduction and the two methods, neglecting numerical errors, will always predict the same reduced model.

The generalization has been borne out by extending the illustrative problem of Chapter 3 and reducing it to all possible lower orders. The results are given in Table 4.2. The same model was predicted for all orders examined, including fourth order which is an unstable solution. In this investigation it was also shown the moments method as formulated in Eq. (4.14) is numerically unsound and leads to singular equations for higher order problems. This led Kropholler to propose an alternative moments formulation, upon which the results of Table 4.2 are based. This solution has been given by Bosley et al. (13).

<u>Denominator</u>								
Model Order	s^7	s^6	s^5	s^4	s^3	s^2	s^1	Const.
7	1	83.64	4097	70342	853703	2814270	3310875	281250
6	-	1	64.734	1189.94	15032.5	49951	58971.4	50.0.81
5	-	-	1	19.4511	306.276	1066.49	1289.69	109.787
* 4	-	-	-	1	-12.9594	-65.4426	-92.0323	-7.9178
3	-	-	-	-	1	4.08491	5.28328	.451958
2	-	-	-	-	-	1	1.14643	.0993976

<u>Numerator</u>								
Model Order		s^6	s^5	s^4	s^3	s^2	s^1	Const.
7		0	0	0	0	0	375000	31250
6		-	-.000715	.059776	-2.0716	-21.3603	6679.31	556.757
5		-	-	.000155	.0358564	-3.63307	146.08	12.1986
4		-	-	-	-.078537	1.56148	-10.4624	-.879755
3		-	-	-	-	-0495536	.598481	.0502175
2		-	-	-	-	-	.129899	.0110442

* unstable system

Table 4.2 Coefficients for the reduction of the seventh order model by the continued fractions and moments methods

4.5 NOMENCLATURE

A_{ij}	element in Chen's Routh Array
$A_p(s)$	defined by Eq. (4.9)
\underline{a}	defined by Eqs. (4.4) and (4.5)
a_i	transfer function coefficient
\underline{b}	defined by Eqs. (4.4) and (4.5)
b_i	transfer function coefficient
\underline{e}	defined by Eqs. (4.4) and (4.5)
$G(s)$	transfer function
h_i	continued fraction coefficient
\underline{I}	identity matrix
M'_k	kth unnormalized moment about the origin
n	system order
\underline{Q}_{ij}	defined by Eqs. (4.4) and (4.5)
\underline{R}_{ij}	defined by Eqs. (4.4) and (4.5)
s	Laplace operator

Superscript:

* reduced order system.

CHAPTER 5

The simplification of state variable models

5. THE SIMPLIFICATION OF STATE VARIABLE MODELS

Compared with the classical theory of control the state representation of processes is a relatively recent development and has only been made possible by the availability of large fast computers. Consequently there has been proportionally less work on the order reduction of state models than on transfer functions.

Two completely different methods appeared about the same time: one based on the modal analysis of the system and the other on a least-squares fit in the time domain. Despite other methods being developed these approaches are still the backbone of the work and will be developed in some depth. Other methods, some resulting from extensions of these methods, will be discussed less fully. The classification given in Chapter 3 for methods of reduction and system responses applies equally to state variable models. An illustrative example will be given.

5.1 RETENTION OF THE DOMINANT MODES

A number of methods of reducing the order of state variable models by retention of the dominant modes have been proposed. They all, however, follow basically the same analysis and most of the theory was developed by a group at Cambridge led by Professor J.F. Coates. The workers in this group were Mann, Marshall, Nicholson and Davison. The work of Davison is the best known (39). Following the publication of Davison's work some lengthy correspondence appeared between Chidambara

and Davison [32, 33, 34, 42] in which little was added to the work (other than confusion).

5.1.1 Problem statement 1

The nth order system

$$\dot{\underline{x}} = \underline{A}\underline{x} + \underline{B}u \quad (5.1)$$

is to be reduced to the mth order system

$$\dot{\underline{x}}^* = \underline{A}^*\underline{x}^* + \underline{B}^*u \quad (5.2)$$

Eq. (5.1) can always be written in the form

$$\dot{\underline{x}} = \underline{U} \underline{\Lambda} \underline{U}^{-1} \underline{x} + \underline{B}u \quad (5.3)$$

where $\underline{\Lambda}$ is the Jordan canonical form and \underline{U} is the eigenvector matrix. In the following analysis it will be assumed that all eigenvalues are real and distinct and lie in the left hand half of the complex plane, hence $\underline{\Lambda}$ is diagonal, but much of the analysis can be extended to the general canonical form.

$\underline{\Lambda}$ can always be partitioned

$$\underline{\Lambda} = \begin{bmatrix} \underline{\Lambda}_1 & \\ & \underline{\Lambda}_2 \end{bmatrix} \quad (5.4)$$

where $\underline{\Lambda}_1$ contains the m dominant eigenvalues (i.e. those with the smallest modulus, or largest time constant), and $\underline{\Lambda}_2$ the remainder.

5.1.2 Ordering of the system eigenvalues

Of course in general $\underline{\Lambda}$ in Eq. (5.4) does not possess the system eigenvalues in ascending modulus order. Eqs. (5.1) and (5.3) can always be put into this form by

applying permutation matrices. Let $\underline{\Lambda}'$ contain the n eigenvalues in random order and $\underline{\Lambda}$ the same eigenvalues in modulus ascending order, and let \underline{P} be the permutation matrix that achieves that ordering.

$$\dot{\underline{x}} = \underline{U} \underline{\Lambda}' \underline{U}^{-1} \underline{x} + \underline{B} \underline{u} \quad (5.5)$$

$$\underline{P} \dot{\underline{x}} = \underline{P} \underline{U} \underline{P}^{-1} \underline{P} \underline{\Lambda}' \underline{P}^{-1} \underline{P} \underline{U}^{-1} \underline{P}^{-1} \underline{P} \underline{x} + \underline{P} \underline{B} \underline{u} \quad (5.6)$$

Let $\underline{y} = \underline{P} \underline{x}$ and $\underline{\Lambda} = \underline{P} \underline{\Lambda}' \underline{P}^{-1}$

therefore

$$\dot{\underline{y}} = (\underline{P} \underline{U} \underline{P}^{-1}) \underline{\Lambda} (\underline{P} \underline{U}^{-1} \underline{P}^{-1}) \underline{y} + \underline{P} \underline{B} \underline{u} \quad (5.7)$$

It may be noted that the states in \underline{x} have also been ordered. Hereafter it will be assumed that all systems have been graded into the form of Eq. (5.7).

5.1.3. Problem statement 2

$$\text{Eq. (5.1)} \quad \dot{\underline{x}} = \underline{A} \underline{x} + \underline{B} \underline{u} \quad (5.8)$$

by letting

$$\underline{x} = \underline{U} \underline{z} \quad \text{and} \quad \underline{V} \underline{U} = \underline{I} \quad (5.9)$$

may alternately be written

$$\dot{\underline{x}} = \underline{U} \underline{A} \underline{V} \underline{x} + \underline{B} \underline{u} \quad (5.10)$$

$$\dot{\underline{z}} = \underline{\Lambda} \underline{z} + \underline{V} \underline{B} \underline{u} \quad (5.11)$$

Let Eqs. (5.8) - (5.11) be partitioned in accordance with $\underline{\Lambda}$, Eq. (5.4) thus

$$\begin{bmatrix} \dot{\underline{x}}_1 \\ \dot{\underline{x}}_2 \end{bmatrix} = \begin{bmatrix} \underline{A}_1 & \underline{A}_2 \\ \underline{A}_3 & \underline{A}_4 \end{bmatrix} \begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \end{bmatrix} + \begin{bmatrix} \underline{B}_1 \\ \underline{B}_2 \end{bmatrix} \underline{u} \quad (5.12)$$

$$\begin{bmatrix} \dot{\underline{x}}_1 \\ \dot{\underline{x}}_2 \end{bmatrix} = \begin{bmatrix} \underline{U}_1 & \underline{U}_2 \\ \underline{U}_3 & \underline{U}_4 \end{bmatrix} \begin{bmatrix} \underline{\Lambda}_1 & \\ & \underline{\Lambda}_2 \end{bmatrix} \begin{bmatrix} \underline{V}_1 & \underline{V}_2 \\ \underline{V}_3 & \underline{V}_4 \end{bmatrix} \begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \end{bmatrix} + \begin{bmatrix} \underline{B}_1 \\ \underline{B}_2 \end{bmatrix} \underline{u} \quad (5.13)$$

$$\begin{bmatrix} \dot{\underline{z}}_1 \\ \dot{\underline{z}}_2 \end{bmatrix} = \begin{bmatrix} \underline{\Lambda}_1 & \\ & \underline{\Lambda}_2 \end{bmatrix} \begin{bmatrix} \underline{z}_1 \\ \underline{z}_2 \end{bmatrix} + \begin{bmatrix} \underline{V}_1 & \underline{V}_2 \\ \underline{V}_3 & \underline{V}_4 \end{bmatrix} \begin{bmatrix} \underline{B}_1 \\ \underline{B}_2 \end{bmatrix} \underline{u} \quad (5.14)$$

$$\begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \end{bmatrix} = \begin{bmatrix} \underline{U}_1 & \underline{U}_2 \\ \underline{U}_3 & \underline{U}_4 \end{bmatrix} \begin{bmatrix} \underline{z}_1 \\ \underline{z}_2 \end{bmatrix} \quad (5.15)$$

$$\begin{bmatrix} \underline{V}_1 & \underline{V}_2 \\ \underline{V}_3 & \underline{V}_4 \end{bmatrix} \begin{bmatrix} \underline{U}_1 & \underline{U}_2 \\ \underline{U}_3 & \underline{U}_4 \end{bmatrix} = \begin{bmatrix} \underline{I} & \underline{O} \\ \underline{O} & \underline{I} \end{bmatrix} \quad (5.16)$$

5.1.4 The problem solution

The impulse response for the full system may be written out from Eq. (5.13) as

$$\begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \end{bmatrix} = \begin{bmatrix} \underline{U}_1 & \underline{U}_2 \\ \underline{U}_3 & \underline{U}_4 \end{bmatrix} \begin{bmatrix} e^{\underline{\Lambda}_1 t} & \\ & e^{\underline{\Lambda}_2 t} \end{bmatrix} \begin{bmatrix} \underline{G}_1 \\ \underline{G}_2 \end{bmatrix} \underline{u} \quad (5.17)$$

where

$$\begin{bmatrix} \underline{G}_1 \\ \underline{G}_2 \end{bmatrix} = \begin{bmatrix} \underline{V}_1 & \underline{V}_2 \\ \underline{V}_3 & \underline{V}_4 \end{bmatrix} \begin{bmatrix} \underline{B}_1 \\ \underline{B}_2 \end{bmatrix} \quad (5.18)$$

and in particular for \underline{x}_1 as

$$\underline{x}_1 = \underline{U}_1 e^{\underline{\Lambda}_1 t} \underline{G}_1 \underline{u} + \underline{U}_2 e^{\underline{\Lambda}_2 t} \underline{G}_2 \underline{u} \quad (5.19)$$

Eq. (5.19) may be integrated to give the step response.

$$\underline{x}_1 = \underline{U}_1 e^{\underline{\Lambda}_1 t} \underline{\Lambda}_1^{-1} \underline{G}_1 \underline{u} + \underline{U}_2 e^{\underline{\Lambda}_2 t} \underline{\Lambda}_2^{-1} \underline{G}_2 \underline{u} - \underline{U}_1 \underline{\Lambda}_1^{-1} \underline{G}_1 \underline{u} - \underline{U}_2 \underline{\Lambda}_2^{-1} \underline{G}_2 \underline{u} \quad (5.20)$$

5.1.5 Nicholson's and Davison's method (98,39)

These methods although published separately are the same, however, Nicholson (98) gave the method as an appendix only, whilst Davison (39) has given a much more thorough treatment.

Consider the free systems associated with Eq. (5.12) and (5.14)

$$\begin{bmatrix} \dot{\underline{x}}_1 \\ \dot{\underline{x}}_2 \end{bmatrix} = \begin{bmatrix} \underline{A}_1 & \underline{A}_2 \\ \underline{A}_3 & \underline{A}_4 \end{bmatrix} \begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \end{bmatrix} \quad (5.21)$$

$$\begin{bmatrix} \dot{\underline{z}}_1 \\ \dot{\underline{z}}_2 \end{bmatrix} = \begin{bmatrix} \underline{\Lambda}_1 & \\ & \underline{\Lambda}_2 \end{bmatrix} \begin{bmatrix} \underline{z}_1 \\ \underline{z}_2 \end{bmatrix} \quad (5.22)$$

Solution of Eq. (5.22) is

$$\begin{bmatrix} \underline{z}_1 \\ \underline{z}_2 \end{bmatrix} = \begin{bmatrix} e^{\underline{\Lambda}_1 t} & \\ & e^{\underline{\Lambda}_2 t} \end{bmatrix} \begin{bmatrix} \underline{z}_1(0) \\ \underline{z}_2(0) \end{bmatrix} \quad (5.23)$$

However as $\underline{\Lambda}_2$ contains only large negative eigenvalues $e^{\underline{\Lambda}_2 t}$ may effectively be considered as zero. Rewriting Eq. (5.22)

$$\begin{bmatrix} \underline{z}_1 \\ \underline{z}_2 \end{bmatrix} = \begin{bmatrix} e^{\underline{\Lambda}_1 t} & \\ & \end{bmatrix} \begin{bmatrix} \underline{z}_1(0) \\ \underline{z}_2(0) \end{bmatrix} \quad (5.24)$$

or in particular $\underline{z}_2 = \underline{0}$ (5.25)

Retaining \underline{x}_1 from Eq. (5.21) gives

$$\dot{\underline{x}}_1 = \underline{A}_1 \underline{x}_1 + \underline{A}_2 \underline{x}_2 \quad (5.26)$$

and from Eqs. (5.15) and (5.25)

$$\underline{x}_2 = \underline{U}_3 \underline{U}_1^{-1} \underline{x}_1 \quad (5.27)$$

Substitute into Eq. (5.26) to give

$$\dot{\underline{x}}_1 = (\underline{A}_1 + \underline{A}_2 \underline{U}_3 \underline{U}_1^{-1}) \underline{x}_1 \quad (5.28)$$

$$\text{but if } \underline{x}^* = \underline{x}_1 \quad (5.29)$$

then

$$\underline{A}_{\text{DAV}}^* = \underline{A}_1 + \underline{A}_2 \underline{U}_3 \underline{U}_1^{-1} \quad (5.30)$$

In the above analysis the fact that Eqs. (5.12) and (5.14) are not free systems is ignored. Further a reduced order B matrix has not yet been derived.

The solution to Eq. (5.13) is

$$\begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \end{bmatrix} = \begin{bmatrix} \underline{U}_1 & \underline{U}_2 \\ \underline{U}_3 & \underline{U}_4 \end{bmatrix} \begin{bmatrix} e^{\underline{A}_1 t} & \\ & e^{\underline{A}_2 t} \end{bmatrix} \begin{bmatrix} \underline{G}_1 \\ \underline{G}_2 \end{bmatrix}$$

or in particular

$$\underline{x}_1 = \underline{U}_1 e^{\underline{A}_1 t} \underline{G}_1 u \quad (5.32)$$

c.f. the standard form

$$\underline{\dot{x}} = \underline{U} e^{\underline{A} t} \underline{U}^{-1} \underline{B} u \quad (5.33)$$

it may be concluded that the B matrix sought is

$$\underline{B}_{\text{DAV}}^* = \underline{U}_1 \underline{G}_1 \quad (5.34)$$

5.1.6 Marshall's method (82)

Marshall's approach is different from Davison's in that he assumes $\dot{\underline{z}}_2 = 0$ (5.35)

instead of $\underline{z}_2 = 0$ (5.36)

The basis for this is that the fast modes act and die quickly and thereafter play no part in the responses: they may therefore be approximated by a constant step. Setting $\dot{\underline{z}}_2$ to zero in Eq. (5.14) yields

$$\underline{0} = \underline{\Lambda}_2 \underline{z}_2 + \underline{G}_2 \underline{u} \quad (5.37)$$

$$\text{or } \underline{z}_2 = -\underline{\Lambda}_2^{-1} \underline{G}_2 \underline{u} \quad (5.38)$$

However from Eqs. (5.15) and (5.16) it may be shown

$$\underline{z}_2 = \underline{V}_3 \underline{x}_1 + \underline{V}_4 \underline{x}_2 \quad (5.39)$$

Equating Eqs. (5.38) and (5.39) and solving for \underline{x}_2 gives

$$\underline{x}_2 = -\underline{V}_4^{-1} \underline{V}_3 \underline{x}_1 - \underline{V}_4^{-1} \underline{\Lambda}_2^{-1} \underline{G}_2 \underline{u} \quad (5.40)$$

Writing Eq. (5.12) for \underline{x}_1 only

$$\dot{\underline{x}}_1 = \underline{A}_1 \underline{x}_1 + \underline{A}_2 \underline{x}_2 + \underline{B}_1 \underline{u} \quad (5.41)$$

substitute \underline{x}_2 from Eq. (5.40)

$$\dot{\underline{x}}_1 = (\underline{A}_1 - \underline{A}_2 \underline{V}_4^{-1} \underline{V}_3) \underline{x}_1 + (\underline{B}_1 - \underline{A}_2 \underline{V}_4^{-1} \underline{\Lambda}_2^{-1} \underline{G}_2) \underline{u}$$

c.f. the standard form (5.42)

$$\dot{\underline{x}}^* = \underline{A}^* \underline{x}^* + \underline{B}^* \underline{u} \quad (5.43)$$

it may be concluded

$$\underline{A}^*_{\text{MAR}} = \underline{A}_1 - \underline{A}_2 \underline{V}_4^{-1} \underline{V}_3 \quad (5.44)$$

$$\underline{B}^*_{\text{MAR}} = \underline{B}_1 - \underline{A}_2 \underline{V}_4^{-1} \underline{\Lambda}_2^{-1} \underline{G}_2 \quad (5.45)$$

5.1.7 A comparison of the methods of Davison and Marshall

Davison gives

$$\underline{A}^*_{\text{DAV}} = \underline{A}_1 + \underline{A}_2 \underline{U}_3 \underline{U}_1^{-1} \quad (5.46)$$

$$\underline{B}^*_{\text{DAV}} = \underline{U}_1 \underline{G}_1 \quad (5.47)$$

Marshall gives

$$\underline{A}^*_{MAR} = \underline{A}_1 - \underline{A}_2 \underline{V}_4^{-1} \underline{V}_3 \quad (5.48)$$

$$\underline{B}^*_{MAR} = \underline{B}_1 - \underline{A}_2 \underline{V}_4^{-1} \underline{\Lambda}_2^{-1} \underline{G}_2 \quad (5.49)$$

It is, however, by use of the relations given in Eqs. (5.12) and (5.16) possible to show for both models

$$\underline{A}^* = \underline{U}_1 \underline{\Lambda}_1 \underline{U}_1^{-1} \quad (5.50)$$

and that

$$\underline{B}^*_{MAR} = \underline{B}^*_{DAV} + \underline{A}^* \underline{U}_2 \underline{\Lambda}_2^{-1} \underline{G}_2 \quad (5.51)$$

Marshall (82) points out that his method always gives the correct steady state whilst that of Davison does not. Davison in reply (41) concedes this fact but points out the error may only be small and should be smaller as the reduced order is increased. In the same paper Davison shows that in some cases, whilst Marshall's method gives the correct steady states it does not give a good representation of the transient responses.

Fossard (48) has shown explicitly, both algebraically and numerically, what differences there are between the methods. The step responses for the solution, Davison's and Marshall's methods are:

$$\underline{x}^*_{\text{EXACT}} = \underline{U}_1 e^{\underline{\Lambda}_1 t} \underline{\Lambda}_1^{-1} \underline{G}_1 + \underline{U}_2 e^{\underline{\Lambda}_2 t} \underline{\Lambda}_2^{-1} \underline{G}_2 - \underline{U}_1 \underline{\Lambda}_1^{-1} \underline{G}_1 - \underline{U}_2 \underline{\Lambda}_2^{-1} \underline{G}_2 \quad (5.52)$$

$$\underline{x}^*_{\text{DAV}} = \underline{U}_1 e^{\underline{\Lambda}_1 t} \underline{\Lambda}_1^{-1} \underline{G}_1 - \underline{U}_1 \underline{\Lambda}_1^{-1} \underline{G}_1 \quad (5.53)$$

$$\begin{aligned} \underline{x}^*_{\text{MAR}} = \underline{U}_1 e^{\underline{\Lambda}_1 t} \underline{\Lambda}_1^{-1} \underline{G}_1 & - \underline{U}_1 \underline{\Lambda}_1^{-1} \underline{G}_1 - \underline{U}_2 \underline{\Lambda}_2^{-1} \underline{G}_2 - \\ & \underline{U}_1 e^{\underline{\Lambda}_1 t} \underline{U}_1^{-1} \underline{U}_2 \underline{\Lambda}_2^{-1} \underline{G}_2 \end{aligned} \quad (5.54)$$

Inspection of Eqs. (5.52) - (5.54) shows exactly why the two methods differ and why they both deviate from the solution. Davison's method cannot give the correct steady states due to the absence of the term $\underline{U}_2 \underline{\Lambda}_2 \underline{G}_2$. Similarly Marshall's method will have the correct steady state but a wrong transient portion caused by the inclusion of the term $\underline{U}_1 e^{\underline{\Lambda}_1 t} \underline{U}_1^{-1} \underline{U}_2 \underline{\Lambda}_2^{-1} \underline{G}_2$.

5.1.8 Chidambara's methods (32,33,34,43)

Following publication of Davison's method Chidambara questioned a number of points, the most valid of which was the steady state deviation. The letter was followed by a further eight in which Chidambara proposed two methods, Davison modified his method to give the correct steady state, and finally an alternative method was proposed by Davison.

Chidambara's first method (32), based upon the solution, Eq. (5.14), with the same assumption as used by Marshall ($\dot{\underline{z}} = 0$), leads to the set of equations:

$$\dot{\underline{z}}_1 = \underline{\Lambda}_1 \underline{z}_1 + \underline{G}_1 \underline{u} \quad (5.55a)$$

$$\underline{z}_2 = -\underline{\Lambda}_2^{-1} \underline{G}_2 \quad (5.55b)$$

$$\underline{x}^* = \underline{x}_1 = \underline{U}_1 \underline{z}_1 + \underline{U}_2 \underline{z}_2 \quad (5.55c)$$

the step response for which is

$$\underline{x}_{\text{CHD}}^* = \underline{U}_1 e^{\underline{\Lambda}_1 t} \underline{\Lambda}_1^{-1} \underline{G}_1 - \underline{U}_1 \underline{\Lambda}_1^{-1} \underline{G}_1 - \underline{U}_2 \underline{\Lambda}_2^{-1} \underline{G}_2 \quad (5.56)$$

This step response compares favourably with that of the solution, Eq. (5.52), however, the reduced model set, Eqs. (5.55) do not constitute an acceptable form of state variable model (32, Author's reply).

Chidambara's second method is identical to that of Marshall. This similarity has been shown by Fossard (48).

5.1.9 Davison's modified models (33,42)

In answer to Chidambara's criticism of his steady state errors Davison proposed two new methods. In order to do so it was necessary to relax the constraints on the form of the model and allow, like Chidambara, a more complex form.

The first form (33) was:

$$\dot{\underline{x}}_1 = \underline{A}^* \underline{x}_1 + \underline{B}^* \underline{u} \quad (5.57a)$$

$$\underline{x}^* = \underline{x}_1 + (\underline{A}^{*-1} \underline{B}^* - (\underline{A}^{-1} \underline{B})_1) \underline{u} \quad (5.57b)$$

where $[\underline{A}^{-1} \underline{B}]_1$ indicates the product $(\underline{A}^{-1} \underline{B})$ partitioned as before. Eqs. (5.57) merely adds the respective steady state errors to each state over its entire range.

The second form (42) was:

$$\dot{\underline{x}}_1 = \underline{A}^* \underline{x}_1 + \underline{B}^* \underline{u} \quad (5.58a)$$

$$\underline{x}^* = \underline{D} \underline{x}_1 \quad (5.58b)$$

when the diagonal matrix \underline{D} is

$$\underline{D} = \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_m \end{bmatrix} \quad (5.59)$$

and d_i is the ratio of the reduced steady state to the correct value of x_i . Clearly for this model to be adequate a different modifying matrix \underline{D} is required for each input.

5.1.10 Analysis of the unretained variables

It has been shown that with each of the methods

$$\underline{A}^* = \underline{U}_1 \underline{A}_1 \underline{U}_1^{-1} \quad (5.60)$$

clearly this implies that the partitioned eigenvector matrix \underline{U}_1 must be non-singular. As the system eigenvectors possess one arbitrary set this is not generally the case and the matrix \underline{U} must be partitioned to ensure that the determinant of \underline{U}_1 is large. Both Davison and Nicholson (42,98) advise that in order to ensure this, physically different variables must be retained (e.g. a pressure and temperature instead of two temperatures). However in satisfying this condition a variable of interest

may be lost.

e.g. Consider

$$\begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \quad \text{and} \quad \underline{x}^* = \underline{x}_1 \quad (5.61)$$

where variable x_4 may be of interest. In order to analyse such variables additional equations are required.

These are

$$\underline{x}_{2\text{DAV}} = \underline{U}_3 \underline{U}_1^{-1} \underline{x}_1 \quad (5.62)$$

$$\underline{x}_{2\text{MAR}} = -\underline{V}_4^{-1} \underline{V}_3 \underline{x}_1 - \underline{V}_4^{-1} \underline{\Lambda}_2^{-1} \underline{G}_2 \underline{u} \quad (5.63)$$

$$= \underline{x}_{2\text{DAV}} - \underline{V}_4^{-1} \underline{\Lambda}_2^{-1} \underline{G}_2 \underline{u} \quad (5.63a)$$

5.1.11 An extension of Davison's method (71)

The above methods are based upon retaining only the dominant modes and discarding the short time constants in the reduced model. The short time constants only contribute to the initial transient response and then play no part. The effect of neglecting these constants is to give a good overall approximation of the response at the expense of the early response. However, in some situations this may not be satisfactory (e.g. for control purposes this is the most important part of the response). Kuppurajulu and Elangovan (71) realising this have proposed that the eigenvalues be split into three groups

(not necessarily of equal size) any two of which can be neglected when working with a particular part of the curve only.

5.2 LEAST SQUARES FITTING IN THE TIME DOMAIN

In 1967 Anderson proposed a new method(2) which did not require the modal analysis of the unreduced system, but rather the full time response of all retained outputs for all inputs. Anderson has shown how his method applies to vector difference equations and stated that it may be extended to continuous time systems. This extension has been given by Chandrasekharan and Balakrishnan (20).

The nth order model, Eq. (5.1), is to be reduced to the mth order system

$$\dot{\underline{x}}^* = \underline{A}^* \underline{x}^* + \underline{B}^* \underline{u} \quad (5.64)$$

where chosen states from the vector \underline{x} make up the reduced state vector \underline{x}^* .

Eq. (5.64) for input \underline{u}_1 may be written out at intervals of a sampling interval T to give a set of K+1 equations

$$\begin{aligned} \underline{\dot{x}}_1^*(0) &= \underline{A}^* \underline{x}_1^*(0) + \underline{B}^* \underline{u}_1(0) \\ &\vdots \\ \underline{\dot{x}}_1^*(KT) &= \underline{A}^* \underline{x}_1^*(KT) + \underline{B}^* \underline{u}_1(KT) \end{aligned} \quad (5.65)$$

which may be partitioned to give

$$\begin{aligned} (\underline{\dot{x}}_1^*(0) \mid \dots \mid \underline{\dot{x}}_1^*(KT)) &= \underline{A}^* (\underline{x}_1^*(0) \mid \dots \mid \underline{x}_1^*(KT)) \\ &+ \underline{B}^* (\underline{u}_1(0) \mid \dots \mid \underline{u}_1(KT)) \end{aligned} \quad (5.66)$$

or

$$\dot{\underline{X}}_1^* = \underline{A}^* \underline{X}_1^* + \underline{B}^* \underline{U}_1 \quad (5.67)$$

Eq. (5.67) exists for all different system inputs, $\underline{u}_1 \dots \underline{u}_J$, and may be similarly partitioned to give

$$\begin{pmatrix} \dot{\underline{X}}_1^* & \dots & \dot{\underline{X}}_J^* \end{pmatrix} = \underline{A}^* \begin{pmatrix} \underline{X}_1^* & \dots & \underline{X}_J^* \end{pmatrix} + \underline{B}^* \begin{pmatrix} \underline{u}_1 & \dots & \underline{u}_J \end{pmatrix} \quad (5.68)$$

$$\text{or } \underline{Q} = \underline{A}^* \underline{R} + \underline{B}^* \underline{S} \quad (5.69)$$

$$= \begin{pmatrix} \underline{A}^* & \underline{B}^* \end{pmatrix} \begin{bmatrix} \underline{R} \\ \underline{S} \end{bmatrix} \quad (5.70)$$

$$\underline{Q} = \underline{T} \underline{W} \quad (5.71)$$

where \underline{Q} is order $m, (K+1)J$

$\underline{T} = \begin{pmatrix} \underline{A}^* & \underline{B}^* \end{pmatrix}$ is order $m, (m + J)$

$\underline{W} = \begin{bmatrix} \underline{R} \\ \underline{S} \end{bmatrix}$ is order $(m + J), (K + 1)J$

The matrices \underline{Q} and \underline{W} which are composed according to the above equations are built up from vector $\dot{\underline{x}}$ and \underline{x} for the selected output variables to be retained for all the inputs and times considered, and Eq. (5.71) is solved thus:

$$\begin{bmatrix} \underline{A}^{*T} \\ \underline{B}^{*T} \end{bmatrix} = \underline{W}^{T\dagger} \underline{Q}^T \quad (5.72)$$

where superscript \dagger indicates the generalized matrix pseudoinverse.

Some points must be considered in the use of Eq. (5.71). The sampling period T must be at least smaller than the smallest time constant and the overall period KT longer than the longest time constant. For most systems

these two points imply a large number of samples
(i.e. K is large) and

$$(m+J) \neq (K+1)J \quad (5.73)$$

therefore matrix \underline{W} is singular and the inverse does not exist. However a pseudoinverse does exist (86) which is the best possible solution to Eq. (5.71) in a least-squares sense. A good deal has been written about the geometrical interpretation of Eq. (5.71) in an attempt to clarify its meaning, however, this author believes that to discuss a least-squares solution is adequate and readily understood.

Nicholson has commented upon the large amount of core store that would be required for a system with a wide range of eigenvalues and a number of inputs, (99). Anderson has shown how this may be reduced by post-multiplying Eq. (5.71) by the transpose of \underline{W} (3). Anderson has also shown how the method can be modified if certain elements in \underline{A}^* or \underline{B}^* are known (4), and how weighting of the least-squares solution may be effected (5). This is done by merely including additional samples, some of which may be repeats, in the area in which emphasis is to be placed. In the same paper it is shown how small steady state errors may be eliminated.

Anderson has implemented all of the above modifications in the simplification of 19th and 31st order boiler models for both the discrete and continuous cases (6). These were compared to the same reductions by Nicholson's

method. Chandrasekharan and Balakrishnan (20) have compared the continuous formulation to Davison's reduction for a fifth order model. Their example will be used in Section 5.4.

5.3 OTHER METHODS OF STATE VARIABLE REDUCTION

There are in the literature a number of other methods of reducing the order of state variable models. Some of these are rigid mathematical extensions of the modal methods, whilst others are based on statistical analysis of the system step response, or engineering judgement applied either at the modelling or solution stage.

Mathematical extensions of the the modal methods have been made by Mitra (89,90,91,92,93) and Wilson (121). Both these methods minimise a performance index which measures the merits of retaining different sets of eigenvalues. A functional between the outputs of the full and reduced models is minimised, leading to a non-linear matrix equation which may be iteratively solved to give the reduced system parameters. In the case where there is only one input or output and the eigenvalues are specified the analysis leads to a linear problem. Chen (22) has noted that both methods may lead to steady state errors and that for a large plant the computations involved may be extremely complex.

Brown (17) has proposed a method, which unlike all others, minimizes the difference between the time rate of change of the reduced and full models, and not the time

response error (i.e. minimize $(\underline{\dot{x}} - \underline{\dot{x}}^*)^2$). However, the model fitted by Brown is not the normal state form but has time varying parameters, i.e. the model fitted is

$$\underline{\dot{x}}^* = \underline{A}^*(t) \underline{x}^* + \underline{B}^*(t) \underline{u}$$

The parameters are found from the covariance matrix of the combined state and input vectors in a system simulation to a random input. Other time varying solutions have been given by Graham and Strauss (57) and Freund (50).

A different statistical approach is that of Tether (114) who has indicated that work carried out on the minimum order of state variable models, can be extended to model simplification in the case of linear models. For continuous time systems the method is equivalent to determining the transfer function matrix or impulse response which has a finite number of terms in the series expansion equal to those in the full model expansion.

Methods have been developed by Aoki (7) and Kuo and Wei (70) which have been given the names aggregation and lumping. From the full model it is necessary to isolate states which behave similarly and are readily decoupled from each other. New states are added to the model which are direct linear combinations of the lumped and eliminated variables. Kuo and Wei have shown how lumping can be applied to monomolecular reactions and in particular the interconversion of butene isomers.

Coggan and Wilson (36) did a similar thing at the modelling stage and effectively lumped together five trays in a distillation column and modelled them as one.

Marshall (83) has developed a method different to all those above based on applying the Leverrier algorithm to the state equations to give the transfer function matrix. The relevant rational terms of this matrix are then reduced by neglecting unimportant roots to give a reduced order transfer matrix which can be converted back to the state equations. The effect of neglecting roots is followed on a visual display unit.

Chen has shown (21,22) how the continued fraction expansion and truncation may be extended to the multivariable case.

Finally, it has been shown in a recent paper by De Sarkar and Dharma Rao (44) how the geometric properties of the Lyapunov function may be utilized in reducing the order of state variable models.

5.4 ILLUSTRATIVE EXAMPLE

Some of the methods described earlier will be illustrated by the following example (20)*. The same example will be used in Chapters 6 and 7.

The fifth order model

* The problem posed requires that variables x_1, x_2, x_3 be analysed by the reduced model.

$$\begin{bmatrix} \dot{x}_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} -3.67 & 0 & -1.333 & 0 & 1.333 \\ 0 & -5.44 & 2.28 & -2.28 & 2.28 \\ -1.333 & 2.28 & -3.86 & 1.19 & -1.52 \\ 0 & -2.28 & 1.19 & -4.19 & 1.19 \\ 1.333 & 2.28 & -1.52 & -1.52 & -3.86 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} 9 \\ 6 \\ 6 \\ 3 \\ 6 \end{bmatrix}$$

is to be reduced to a third order model, the first three state variables being of interest. The following reductions were obtained

Davison's method:

$$\begin{bmatrix} \dot{x}_1 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} -1.006 & 0 & 0 \\ .6557 & -2.6719 & -.656193 \\ 1.3368 & -3.28096 & -2.34381 \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} 3.003 \\ 2.99785 \\ 9.0029 \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1.33066 & -.665832 & 1.33166 \\ -1.9985 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \\ x_5 \end{bmatrix}$$

Marshall's method:

$$\begin{bmatrix} \dot{x}_1 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} -1.006 & 0 & 0 \\ .6557 & -2.6719 & -.656193 \\ 1.3368 & -.328096 & -2.34381 \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} 4.20863 \\ 1.81897 \\ 5.99487 \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1.33066 & -.665832 & 1.33166 \\ -1.9985 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} 2.39403 \\ 3.59442 \end{bmatrix}$$

Anderson's method:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -3.51427 & 1.94234 & -2.14407 \\ - .003909 & -2.01141 & - .001475 \\ -1.36951 & .00966 & -2.46844 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 8.99372 \\ 5.9998 \\ 6.00166 \end{bmatrix}$$

The response for x_1 , x_2 and x_3 for full model and each of the above reduced models are shown in Figs. 5.1, 5.2, and 5.3 respectively.

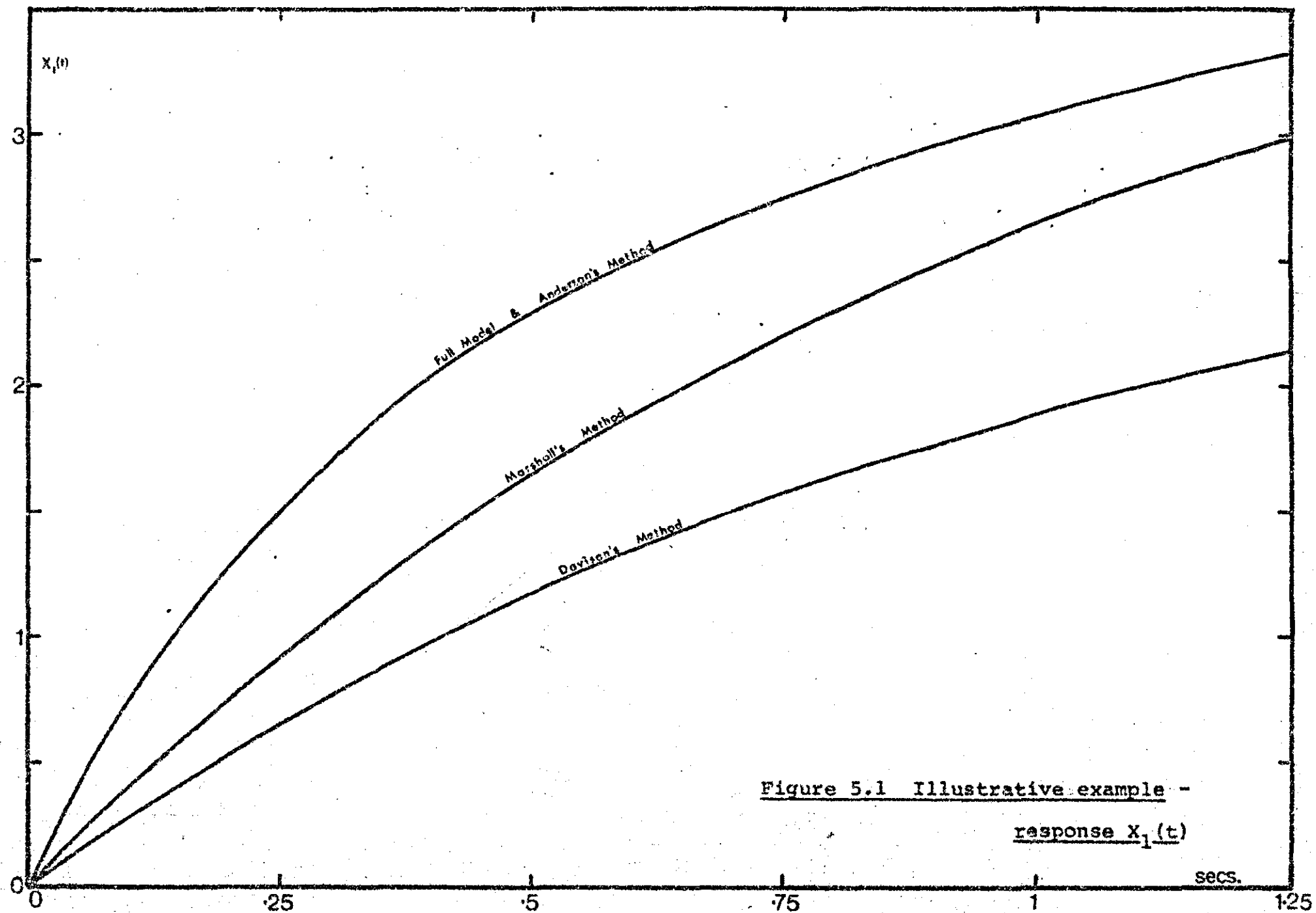


Figure 5.1 Illustrative example -

response $X_1(t)$

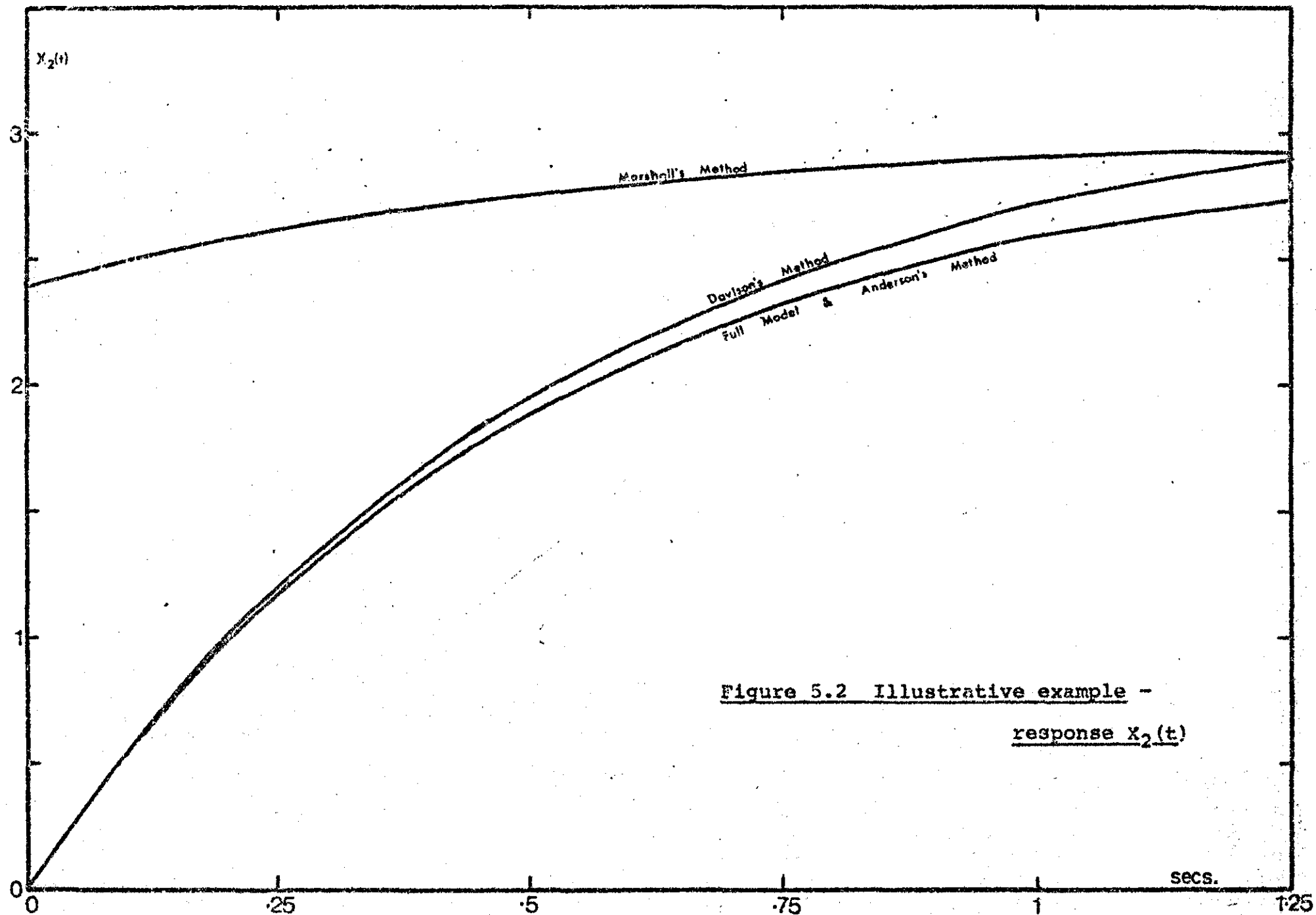
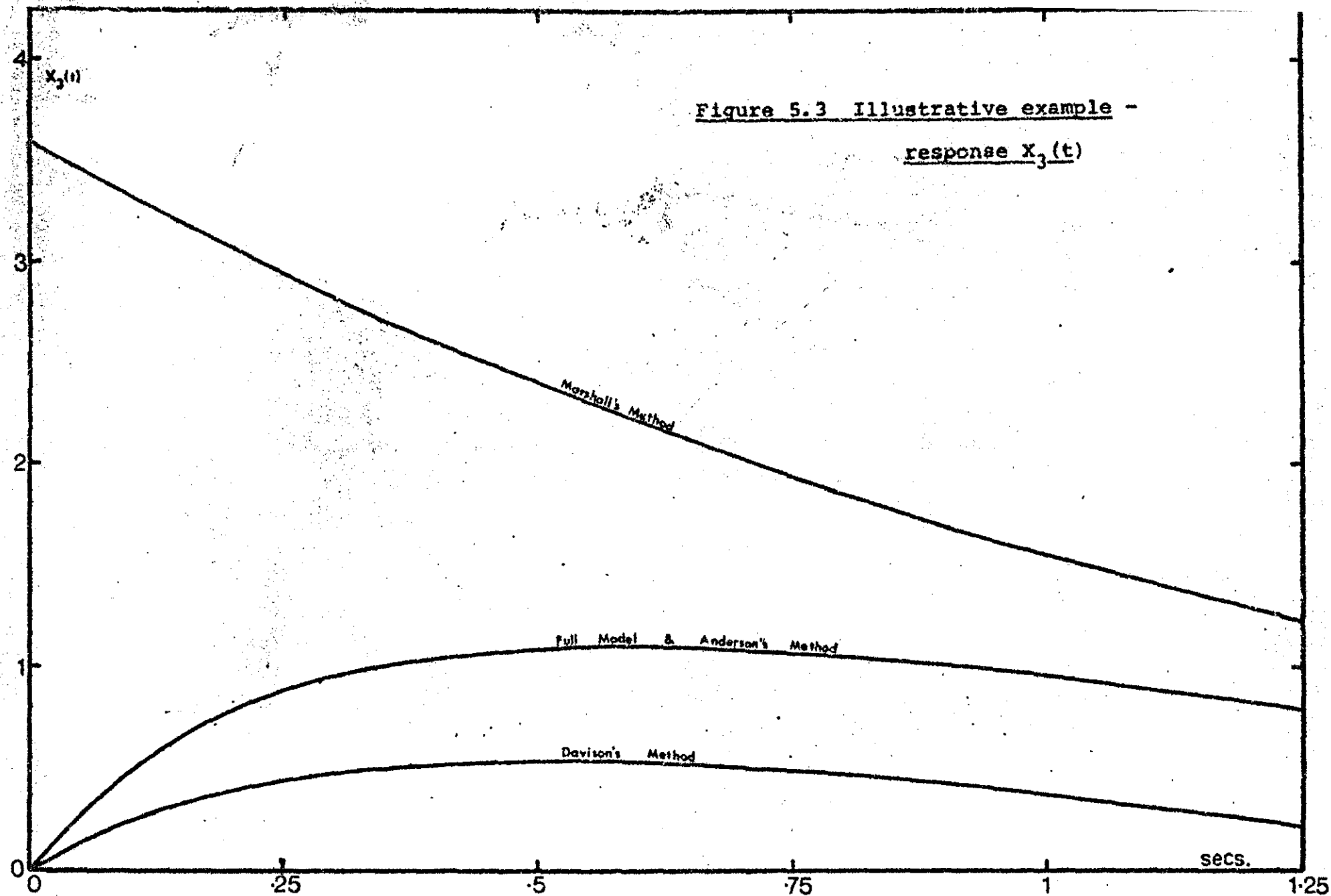


Figure 5.2 Illustrative example -
response $X_2(t)$



5.5 NOMENCLATURE

<u>A</u>	plant matrix
<u>B</u>	input matrix
<u>D</u>	Davison's modifying matrix, defined by Eq. (5.59)
d_i	reduced steady state/full steady state for variable i
<u>G</u>	defined by Eq. (5.18)
<u>I</u>	identity matrix
J	number of inputs
K	number of samples in Anderson's method
m	reduced system order
n	full system order
<u>P</u>	permutation matrix
<u>Q</u>	defined by Eqs. (5.68) and (5.69)
<u>S</u>	defined by Eqs. (5.68) and (5.69)
<u>T</u>	defined by Eqs. (5.70) and (5.71)
T	sampling interval in Anderson's method
t	time
<u>U</u>	matrix of eigenvectors or as defined by Eqs. (5.66) and (5.67) for Anderson's method
<u>u</u>	forcing vector
<u>V</u>	inverse of <u>U</u>
<u>W</u>	as defined by Eqs. (5.70) and (5.71)
<u>X</u>	as defined by Eqs. (5.66) and (5.67)
<u>x</u>	state vector
<u>y</u>	ordered state vector, <u>Px</u>
<u>z</u>	$\underline{U}^{-1}\underline{x}$

Greek:

$\underline{\Lambda}$ Jordan canonical form

$\underline{\Lambda}'$ ordered $\underline{\Lambda}$, \underline{PAP}^{-1}

Superscript:

* reduced system

† pseudoinverse

Subscript:

\underline{U}_{ij} indicates partitioning of \underline{U}

CHD Chidambara's model

DAV Davison's model

EXACT exact model

MAR Marshall's model

CHAPTER 6

The reduction of order of state variable models
using moments

6. THE REDUCTION OF ORDER OF STATE VARIABLE MODELS USING MOMENTS [16]

Lees (54,75,77) has shown how complex models may be simplified using the method of moments. This method has been discussed in depth in Chapter 3. However, the same method may easily be extended to the state-variable case and may be used for simplifying multi-input-multi-output systems. The derivation of moments from the state variable model has been described in Chapter 2.

6.1 PROBLEM STATEMENT

The nth order state variable model

$$\dot{\underline{x}} = \underline{A}\underline{x} + \underline{B}\underline{u} \quad (6.1)$$

is to be reduced to the mth order model

$$\dot{\underline{x}}^* = \underline{A}^*\underline{x}^* + \underline{B}^*\underline{u} \quad (6.2)$$

where

\underline{A} is the plant matrix (n,n)

\underline{A}^* is the reduced plant matrix (m,m)

\underline{u} is the forcing vector (order p)

\underline{B} is the input matrix (n,p)

\underline{B}^* is the reduced input matrix (m,p)

\underline{x} is the state vector (order n)

\underline{x}^* is the reduced state vector (order m).

In many models a large number of the states are not

required as outputs but contribute significantly to the responses of the outputs. These variables may be considered redundant. As an example, in a distillation column feed and product stream variables are important in the dynamic analysis of the column whilst compositions and temperatures on the intermediate plates are extraneous. The extraneous variables are considered to be redundant.

This method seeks to reduce the order of the problem by neglecting the redundant variables whilst maintaining the original responses of the non-redundant variables to all the plant inputs. The reduced state vector, \underline{x}^* , is thus composed from the elements of \underline{x} making use of the physical knowledge of the plant item. The order p of the forcing vector is not reduced, because it is assumed that all the inputs should be retained in the reduced model.

The matrices \underline{A}^* and \underline{B}^* must be found from the moments of the individual responses of all states to all inputs.

6.2 PROBLEM FORMULATION

The Laplace transform of Eq. (6.1) for an impulse is

$$s\underline{x}(s) = \underline{A}\underline{x}(s) + \underline{B}u \quad (6.3)$$

It was shown in Chapter 2 that the moments are given by differentiating this transform and setting s equal to zero, to give

$$\underline{A} \underline{m}_{j,i} + \underline{B} \underline{\phi}_{j,i} = -i \underline{m}_{j,i-1} \quad (6.4)$$

where $\underline{m}_{j,i}$ is the n th order vector of unnormalised i th moments about the origin of the state \underline{x} corresponding to the j th input and $\phi_{j,i}$ is given by

$$\phi_{j,i} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{for } i > 0$$

$$\phi_{j,i} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \quad \begin{array}{l} \text{(1 in the } j\text{th row)} \\ \text{for } i = 0 \end{array}$$

Similarly Eq. (6.4) may be written in terms of reduced matrices

$$\underline{A}^* \underline{m}_{j,i}^* + \underline{B}^* \phi_{j,i} = -i \underline{m}_{j,i-1}^* \quad (6.5)$$

If the moments of the full and reduced models are matched exactly, the elements of $\underline{m}_{j,i}^*$ are identical to those elements of $\underline{m}_{j,i}$ which correspond to the states retained in \underline{x}^* : if the match is not exact, the two sets of elements are only approximately equal.

Eq. (6.5) may be written for all the p forcing functions considered

$$\begin{array}{l} \underline{A}^* \underline{m}_{1,i}^* + \underline{B}^* \phi_{1,i} = -i \underline{m}_{1,i-1}^* \\ \vdots \\ \underline{A}^* \underline{m}_{p,i}^* + \underline{B}^* \phi_{p,i} = -i \underline{m}_{p,i-1}^* \end{array} \quad (6.6)$$

and augmented to give in partitioned form

$$\begin{aligned} \underline{A}^* (\underline{m}_{1,i}^* | \dots | \underline{m}_{p,i}^*) + \underline{B}^* (\underline{\phi}_{1,i} | \dots | \underline{\phi}_{p,i}) \\ = -i (\underline{m}_{1,i-1}^* | \dots | \underline{m}_{p,i-1}^*) \end{aligned} \quad (6.7)$$

$$\text{or } \underline{A}^* \underline{M}_i^* + \underline{B}^* \underline{\Phi}_i = \underline{S}_i^* \quad (6.8)$$

Eq. (6.8) may be written for all the I moments considered in addition to the zeroth ($i = 0, 1, \dots, I$)

$$\begin{aligned} \underline{A}^* \underline{M}_0^* + \underline{B}^* \underline{\Phi}_0 &= \underline{S}_0^* \\ \vdots \\ \underline{A}^* \underline{M}_I^* + \underline{B}^* \underline{\Phi}_I &= \underline{S}_I^* \end{aligned} \quad (6.9)$$

which may be similarly augmented to give

$$\underline{A}^* (\underline{M}_0^* | \dots | \underline{M}_I^*) + \underline{B}^* (\underline{\Phi}_0 | \dots | \underline{\Phi}_I) = (\underline{S}_0^* | \dots | \underline{S}_I^*) \quad (6.10)$$

$$\text{or } \underline{A}^* \underline{M}^* + \underline{B}^* \underline{\Phi} = \underline{S}^* \quad (6.11)$$

where \underline{M}^* and \underline{S}^* are order $(m, p(I+1))$

$\underline{\Phi}$ is order $(p, p(I+1))$

6.3 PROBLEM SOLUTION

It has been shown above that knowing the moments of the selected states for all inputs allows the matrices \underline{M}^* , $\underline{\Phi}$, and \underline{S}^* to be built up in Eq. (6.11), which may in turn be solved to give \underline{A}^* and \underline{B}^* . However it may be solved in a number of ways depending upon the number of inputs and the assumptions made in the solution.

6.3.1 Exact fit

Eq. (6.11) may be rewritten in partitioned form

$$\begin{bmatrix} \underline{A}^* & \underline{B}^* \end{bmatrix} \begin{bmatrix} \underline{M}^* \\ \underline{\Phi} \end{bmatrix} = \underline{S}^* \quad (6.12)$$

$$\text{or} \quad \underline{V} \underline{H} = \underline{S}^* \quad (6.13)$$

Rearranging

$$\underline{H}^T \underline{V}^T = \underline{S}^{*T} \quad (6.14)$$

Solving for \underline{V} in this manner produces a reduced model, the moments of which match exactly all fitted moments of the complex model, including the zeroth or steady state.

6.3.2 B* constrained to give correct initial rates

If it is assumed that the complex and simple models must have the same rate of change at time zero, then since the initial rate is given by \underline{B} , \underline{B}^* may be constructed from the elements of \underline{B} corresponding to the chosen states.

Eq. (6.11)

$$\underline{A}^* \underline{M}^* + \underline{B}^* \underline{\Phi} = \underline{S}^*$$

but if \underline{B}^* is known

$$\underline{A}^* \underline{M}^* = \underline{S}^* - \underline{B}^* \underline{\Phi} \quad * \quad \underline{T} \quad (6.15)$$

$$\text{or} \quad \underline{M}^{*T} \underline{A}^{*T} = \underline{T}^T \quad (6.16)$$

Solving (6.16) matches not only the moments but also the actual rates of change of each variable.

6.3.3. Least-squares solution of non-square data matrices

The following discussion is based on section 6.3.1.

but applies equally to section 6.3.2.

Assuming \underline{H} is a square matrix Eq. (6.14) may be solved to give

$$\begin{bmatrix} \underline{A}^* & \underline{B}^* \end{bmatrix}^T = \underline{V}^T = (\underline{H}^T)^{-1} \underline{S}^{*T} \quad (6.17)$$

Consider the dimensions of the matrix \underline{H}

$$\underline{H} = \begin{bmatrix} \underline{M}^* \\ \underline{\Phi} \end{bmatrix} \quad (6.18)$$

\underline{M}^* is order $(m, p(I+1))$

$\underline{\Phi}$ is order $(p, p(I+1))$

To apply Eq. (6.17) \underline{H} must be square and the following equality exists:

$$m = pI \quad (6.19)$$

Eq. (6.19) shows that the model can only be reduced to an order equal to the product of the number of moments fitted and the number of inputs. This rule will however lead to conflicting results: e.g. a 9th order model with 5 inputs (39) by matching three moments and the zeroth must be "reduced" to a 15th order model.

Clearly the rule, Eq. (6.19) cannot in general apply.

It has been shown above that in general the matrix \underline{H} is not square and an alternative method is required. The linear least-squares method of Golub [56] has been used. However as with any least-squares method unequal weighting of variables can occur. This weighting can take three forms:

- a) high moments tend to be numerically dominant to lower moments, or vice versa.
- b) states may have a different order of moment; e.g. a flow variable would normally be greater in value than a composition.
- c) the same state may have a different order of moments for different inputs.

It is important that each of these weightings is eradicated and in particular that the correct steady state is maintained.

6.3.4 Constraint of B* to give correct steady state and moment weighting

It is usually desirable that on forcing the simplified model it reaches the correct steady state. The steady state is given by setting $i = 0$ in Eq. (6.5).

$$\underline{A}^* \underline{m}_{j,o}^* + \underline{B}^* \phi_{j,o} = \underline{0} \quad (6.20)$$

or for all inputs

$$\underline{B}^* (\phi_{1,o}, \dots, \phi_{p,o}) = -\underline{A}^* (\underline{m}_{1,o}^*, \dots, \underline{m}_{p,o}^*) \quad (6.21)$$

or

$$\underline{B}^* \underline{\phi}_o = -\underline{A}^* \underline{M}_o^* \quad (6.22)$$

and $\underline{\phi}_o$ is an identity matrix.

Eq. (6.22) may be substituted into Eq. (6.11) to give

$$\underline{A}^* (\underline{M}^* - \underline{M}_o^*) = \underline{S}^* \quad (6.23)$$

or $\underline{A}^* \underline{F} = \underline{S}^*$

and \underline{A}^* is given by solving

$$\underline{F}^T \underline{A}^{*T} = \underline{S}^{*T} \quad (6.24)$$

Eq. (6.24) has been solved one column of \underline{A}^{*T} and \underline{S}^{*T} at a time, thus minimising the least-squares error for each variable singly, and avoiding unequal weighting, the rows of \underline{F}^T being each time normalised with respect to the corresponding element in \underline{S}^T , thus avoiding weighting for high moments and different inputs. Once \underline{A}^* is calculated \underline{B}^* may be found from Eq. (6.22).

The use of the substitution discussed above and solution of this pair of equations ensures that the reduced model maintains the correct steady state.

6.4 RESULTS

Each of the solution schemes discussed above have been investigated.

The exact solution of section 6.3.1 was found to give good results when reducing a 6th order model to 3rd order with one input. However, when reducing the size of large models the constraint on the order to which a system could be reduced, Eq. (6.19) as discussed in Section 6.3.3 was found to be too restricting. The solution scheme was therefore discarded.

The solution of section 6.3.2. was similarly rejected. It was found that the initial rate of change was matched for a very brief period only, and thereafter the reduced model found its own rate of change. Further-

more this scheme would not correctly fit the steady state as the two requirements conflict. The least-square solution of Section 6.3.4 performed well and four examples will be given. The following is a summary of the numerical runs presented.

1. Reduction of a 12th order model with 2 inputs, the responses of which are overdamped, to a 4th order model by matching up to the 4th moment.
2. Reduction of 12th order model with 2 inputs, the responses of which are oscillatory, to a 4th order model by matching up to the 4th moment.
3. Reduction of a 12th order model with 2 inputs, the responses of which are inverting, to a 4th order model by matching up to the 4th moment.
4. Reduction of an 11th order model of a binary distillation column with 2 inputs to a 4th order model by matching up to the 4th moment.

The numerical data and the time responses of these models can be found in Appendix 2. The program used can also be found there. The above models will be referred to as Models 1-4 respectively.

6.4.1 Illustrative example

The fifth order model used as an example in Chapter 5 has been reduced to a third order model by matching the zeroth and first three moments using the method of Section 6.3.4, although because of the small dimensions involved it constitutes an exact fit. Details of the full and reduced models are shown below. Time responses are not shown as those of both the full and reduced models were virtually coincident throughout their entire length.

Reduced Model

$$\underline{A}^* = \begin{bmatrix} -2.88298 & 1.18666 & -1.86676 \\ - .000764 & -2.01513 & - .000752 \\ -1.52746 & .198686 & -2.53747 \end{bmatrix}$$

$$\underline{B}^* = \begin{bmatrix} 8.72778 \\ 5.99897 \\ 6.06793 \end{bmatrix}$$

Details of the full model A and B are given in Chapter 5.

Eigenvalues:

Full Model	-1	-2	-3	-5	-9
Reduced Model	-1.01276	-2.01556	-4.40716		

Moments

State	0	1	2	3
1	4.18353	3.20679	5.99491	17.6494
2	2.97534	1.47595	1.46442	21.795
3	.105975	-1.77303	-4.89153	-16.2368

(full and reduced models have the same moments)

6.5 DISCUSSION OF RESULTS

Method 6.3.4 has been found to be the most reliable method and to give the most acceptable results. All results presented in Appendix 2 are based upon this method. Methods 6.3.1 and 6.3.2. have been presented as background to the method.

It may be seen from the data and figures given that the reduced systems retain the important characteristics of the full system. There is usually a very close fit between the actual moments and those fitted to the responses (usually to three figures in the fourth moment). Further the eigenvalues of \underline{A}^* are well representative of the eigenvalues of \underline{A} and may well be the dominant values from the latter. Analysis of the figures shows that there is good agreement between the actual and the fitted time response. In the case of oscillating and inverting systems the peak heights of the reduced system tend to be less than that of the full.

It may be seen from those figures which show the response of the last variable in a series that the reduced order curve tends to oscillate about the full model zero time steady state. These variables being the last in a series tend to be characterised by a large inherent time delay, or its equivalent in lags in series. The process of order reduction involves the elimination of many of these lags, and therefore the reproduction of the time delay is not entirely satisfactory. The oscillation of the initial part of the response about the base line is rather similar in behaviour to that of the Padé approximation to a time delay.

$$G(s) = \frac{1 - .5\tau s}{1 + .5\tau s}$$

The moments matched in the examples are the zeroth and the next four. Experience in the use of the method suggests this is usually the best choice, although good results are often obtained matching one less moment.

Important characteristics of this method of simplification are that the correct steady states are obtained, that the reduction is effected according to a definite criterion, namely the matching of moments, and that there are no restrictions on the inputs and outputs which can be retained. This does not appear to be true of methods based on modal analysis. There is no restriction on the order of the reduced model, though, as with other methods, the latter cannot be expected to represent well the full model if the order is reduced

too low. The method has definite computational advantages over other methods in that a complete time or frequency response is not needed (this may be prohibitive with very large systems), nor are the eigenvalues and vectors required. It is needed only to invert the full plant matrix to obtain the system moments.

The method cannot be applied to systems where the output to a step response is not bounded to a steady-state value, but increases indefinitely, and where the moments are therefore infinite.

It has been shown that the method of moments may be used to reduce the order of state variable models and that the reduced order models give acceptable responses. The method is not presented as a substitute for existing methods but rather as a possible alternative.

6.6 NOMENCLATURE

\underline{A} plant matrix
 \underline{B} input matrix
 \underline{F} defined by Eq. (6.23) and (6.24)
 $G(s)$ transfer function
 \underline{H} defined by Eqs. (6.12) and (6.13)
 I number of moments fitted
 i i th moment
 j j th input
 \underline{H}^* defined by Eq. (6.10) and (6.11)
 $\underline{m}_{j,i}$ vector of i th moments of \underline{x} for the j th input
 m reduced system order
 n full system order
 p number of inputs to system
 \underline{S}^* defined by Eqs. (6.10) and (6.11)
 s Laplace operator
 \underline{T} defined by Eq. (6.15)
 \underline{u} forcing vector
 \underline{V} defined by Eqs. (6.12) and (6.13)
 \underline{x} state vector

Greek:

τ time delay
 $\underline{\Phi}$ defined by Eqs. (6.10) and (6.11)
 $\phi_{j,i}$ defined by Eq. (6.4)

General:

$\underline{x}(s)$ Laplace transform of \underline{x}
 \underline{M}_i^* partitioned matrix from \underline{M}^*

CHAPTER 7

The reduction of state variable models by matching
the frequency response

7. THE REDUCTION OF STATE VARIABLE MODELS BY MATCHING THE FREQUENCY RESPONSE

A number of methods were presented in Chapter 3 for fitting a low order transfer function to a higher order model. Many of these methods were based on minimizing the error in the frequency domain between the two models, usually by a least-squares approach. Frequency methods have not, however, been used for reducing the order of state variable models.

The method of Levy [79] is one of the best frequency methods for reducing the size of a transfer function and an attempt has been made to extend the analysis to the multi-dimensional case. Whilst theoretically attractive no acceptable results have been obtained, in addition to which, for very high order problems a considerable amount of computational effort is required in the solution of the many simultaneous equations. The method was discarded and an alternative sought. One such feasible method is presented here.

7.1 PROBLEM STATEMENT

The n th order state variable model with p inputs

$$\dot{\underline{x}} = \underline{A}\underline{x} + \underline{B}\underline{u} \quad (7.1)$$

is to be reduced to the m th order model with p inputs

$$\dot{\underline{x}}^* = \underline{A}^*\underline{x}^* + \underline{B}^*\underline{u} \quad (7.2)$$

by matching the frequency response of selected states

where:

\underline{x} is the n th order state vector

\underline{x}^* is the m th order reduced state vector

\underline{A} is the (n,n) plant matrix

\underline{A}^* is the (m,m) reduced plant matrix

\underline{B} is the (n,p) input matrix

\underline{B}^* is the (m,p) reduced input matrix

\underline{u} is the p th order forcing vector.

The method is similar to that discussed in Chapter 6, for reducing state variable order by matching the moments, in that variables whose responses are of no interest are considered to be extraneous and redundant. System order reduction is effected by neglecting redundant variables whilst maintaining the original responses of the non-redundant variables to all inputs. It is considered desirable that the reduced order model should maintain all the original inputs. In this case the reduced model, or the matrices \underline{A}^* and \underline{B}^* , must be found from the full model frequency response.

7.2 PROBLEM FORMULATION

Eq. (7.1) may be transformed into the Laplace domain

$$s\underline{x}(s) = \underline{A} \underline{x}(s) + \underline{B} \underline{u}(s) \quad (7.3)$$

and by letting $s = i\omega$ into the frequency domain

$$i\omega \underline{x}(i\omega) = \underline{A} \underline{x}(i\omega) + \underline{B} \underline{u}(i\omega) \quad (7.4)$$

Let \underline{u}_j be the unit vector selecting the j th input to the system, with the unity scalar being in the j th position in the vector.

Eq. (7.4) may be written for each of the p inputs

$$\begin{aligned} i\omega \underline{x}_1(i\omega) &= \underline{A} \underline{x}_1(i\omega) + \underline{B} \underline{u}_1(i\omega) \\ &\vdots \\ i\omega \underline{x}_p(i\omega) &= \underline{A} \underline{x}_p(i\omega) + \underline{B} \underline{u}_p(i\omega) \end{aligned} \quad (7.5)$$

and augmented to give in partitioned form

$$\begin{aligned} i\omega (\underline{x}_1(i\omega) | \dots | \underline{x}_p(i\omega)) &= \underline{A} (\underline{x}_1(i\omega) | \dots | \underline{x}_p(i\omega)) + \\ &\quad \underline{B} (\underline{u}_1(i\omega) | \dots | \underline{u}_p(i\omega)) \end{aligned} \quad (7.6)$$

However $(\underline{u}_1(i\omega) | \dots | \underline{u}_p(i\omega))$ is a square diagonal matrix and post-multiplying Eq. (7.6) by its inverse gives each of the frequency responses normalised with respect to its input, and using the transformation

$$\underline{X}(i\omega) = (\underline{x}_1(i\omega) | \dots | \underline{x}_p(i\omega)) (\underline{u}_1(i\omega) | \dots | \underline{u}_p(i\omega))^{-1} \quad (7.7)$$

gives

$$i\omega \underline{X}(i\omega) = \underline{A} \underline{X}(i\omega) + \underline{B} \quad (7.8)$$

where $\underline{X}(i\omega)$ is a (n,p) matrix with complex elements.

Let $\underline{X}(i\omega)$ be separated into its real and imaginary parts

$$\underline{X}(i\omega) = \underline{r} + i \underline{\phi}$$

where clearly \underline{r} and $\underline{\phi}$ are both (n,p) real matrices. With Eq. (7.9) substituted Eq. (7.8) becomes, for a particular frequency ω_k

$$i\omega_k(\underline{r}_k + i \underline{\phi}_k) = \underline{A}(\underline{r}_k + i \underline{\phi}_k) + \underline{B} \quad (7.10)$$

Eq. (7.10) can also be written for the reduced model

$$i\omega_k(\underline{r}_k^* + i \underline{\phi}_k^*) = \underline{A}^*(\underline{r}_k^* + i \underline{\phi}_k^*) + \underline{B}^* \quad (7.11)$$

where \underline{r}_k^* and $\underline{\phi}_k^*$ are (m,p) matrices containing the real and imaginary parts of the frequency response of the vector \underline{x}^* corresponding to the selected states from \underline{x} , and if an exact fit is obtained they will agree exactly to the equivalent elements in \underline{r}_k and $\underline{\phi}_k$, and for an approximate fit there will only be approximate agreement.

Eq. (7.11) may be separated into its real and imaginary parts to give:

$$\omega_k \underline{r}_k^* = \underline{A}^* \underline{\phi}_k^* \quad (7.12)$$

$$-\omega_k \underline{\phi}_k^* = \underline{A}^* \underline{r}_k^* + \underline{B}^* \quad (7.13)$$

\underline{B}^* , the input matrix, affects the gain of the system and is always obtainable from the full system steady states, which are generally known. The steady states are given at zero frequency.

If $\omega_0 = 0$

$$\text{then from Eq. (7.13)} \quad \underline{B}^* = -\underline{A}^* \underline{r}_0^* \quad (7.14)$$

Substituting into Eq. (7.13) gives

$$-\omega_k \underline{\phi}_k^* = \underline{A}^* (\underline{r}_k^* - \underline{r}_0^*) \quad (7.15)$$

Eq. (7.15) relates the plant matrix to its real and imaginary parts and can be written for all K fitted frequencies.

$$\begin{aligned} -\omega_1 \underline{\phi}_1^* &= \underline{A}^*(\underline{r}_1^* - \underline{r}_0^*) \\ &\vdots \\ -\omega_K \underline{\phi}_K^* &= \underline{A}^*(\underline{r}_K^* - \underline{r}_0^*) \end{aligned} \quad (7.16)$$

which may be augmented and written in partitioned form

$$-(\omega_1 \underline{\phi}_1^* \mid \dots \mid \omega_K \underline{\phi}_K^*) = \underline{A}^*(\underline{r}_1^* - \underline{r}_0^* \mid \dots \mid \underline{r}_K^* - \underline{r}_0^*) \quad (7.17)$$

or

$$\underline{\phi}^* = \underline{A}^* \underline{R}^* \quad (7.18)$$

where \underline{R}^* and $\underline{\phi}^*$ are (m, pK) matrices.

7.3 PROBLEM SOLUTION

It has been shown above that the matrices $\underline{\phi}^*$ and \underline{R}^* can be built up from the known frequency responses of the full system for the selected states of \underline{x} retained in \underline{x}^* . Eq. (7.18) may be solved to give \underline{A}^* and Eq. (7.14) to give the input matrix \underline{B}^* , based on ensuring that the model has the correct steady states. Eq. (7.18) may be solved in the form:

$$\underline{R}^{*T} \underline{A}^{*T} = \underline{\phi}^{*T} \quad (7.19)$$

The method is similar to that of Anderson (2) in that to ensure an adequate fit for all inputs and selected outputs a vast amount of data will be required in fitting over many frequencies. Thus in general the matrices \underline{R}^* and $\underline{\phi}^*$ will not be square. Clearly the solution problem is very similar to that of Anderson and also the moments state variable order reduction, and the same method has been used for solving Eq. (7.19): the linear least-squares method of Golub [56].

As with the moments method steps have been taken to minimise the effect of unequal weighting in the problem. Unequal weighting is due to:-

- a) low frequencies having a larger (numerically) frequency response than the high frequencies.
- b) some system outputs being numerically dominant.
- c) outputs responding dissimilarly to different inputs.

The effect of weighting in Eq. (7.19) has been eliminated by solving \underline{A}^{*T} and $\underline{\phi}^{*T}$ one column at a time thus minimizing the error for each variable singly, the rows of \underline{R}^{*T} being each time normalised with respect to the corresponding scalar element in $\underline{\phi}^{*T}$, thus avoiding weighting for different frequencies and inputs. This is the same procedure as employed for the moments problem.

7.4 RESULTS

A 12th and 11th order problem, each with two inputs,

have been reduced to fourth order models by matching the frequency response in the manner outlined above. The problems have been described in Chapter 6. They are the system of overdamped stirred tanks and the binary distillation column. Computational details and the time responses for each of these systems can be found in Appendix 3. A listing of the program used is also given in Appendix 3.

The illustrative example considered in Chapter 5 has also been analysed and will be given here.

7.4.1 Illustrative example

The 5th order problem, described in Chapter 5 has been reduced to a third order model. The reduced model and computational details are shown in Table 7.1. The time responses for the retained variables are not given as the full and reduced model responses, for all variables, were coincident throughout the entire fitted range.

Table 7.1. Illustrative example-reduced order model

Variables retained	1	2	3
Frequencies fitted:			
.0001	.0002	.0004	.0007
.001	.002	.004	.007
.01	.02	.04	.07
.1	.2	.4	.7
1	2	4	7
10	20	40	70
100	200	400	700

Table 7.1, Cont'd ...

$$\underline{A}^* = \begin{bmatrix} -3.1485 & 1.4866 & -2.0978 \\ -0.0008 & -2.0154 & -0.0025 \\ -1.4657 & 0.1295 & -2.4863 \end{bmatrix}$$

$$\underline{B}^* = \begin{bmatrix} 8.9711 \\ 6.0002 \\ 6.0101 \end{bmatrix}$$

Eigenvalues= -1.0320, -2.0173, -4.6009

7.5 DISCUSSION OF RESULTS

It may be seen from the responses given in Appendix 3 that there is in general good agreement between the full and reduced order model responses. Two points may be noted in particular:

- a) the steady state is always correctly fitted.
- b) inherent system dead times are not always fitted closely. This has been commented upon in Chapter 6 and what was written there applies. However the responses in Appendix 3 show that two variables with large dead times have been very closely fitted and are a better fit than given by the moments match. Other responses are typical of the moments match in that they oscillate about the dead time.

Analysis of the full and reduced system eigenvalues shows that the latter are well representative of the former.

The method has associated with it a number of

computational difficulties - these are all connected with the frequency range over which the model is to be fitted. Practice has shown that it is very difficult to find out over what range a model should be fitted. Clearly the fit should start near the steady state frequency. The difficulty is accentuated by the fact that each variable requires a different frequency range. This is particularly true of variables which have a large inherent time delay. One method of selecting a range is to relate the system time constants to a frequency. A large and small eigenvalue were selected and multiplied by 2π to give a start and finish frequency. This does however, involve a degree of judgement in selecting the eigenvalues. Before stating this as a criterion in selecting the frequency range more experience is necessary and a lot more models need to be fitted. Of course an alternative is to blanket fit, i.e. use a large number of frequencies. This, however, involves the same difficulty as experienced by Anderson (2) in that the arrays Φ^* and P^* become very large and the procedure computationally inefficient. It is thought that the good fit of the time delays referred to above is due to fitting over a wide range.

It has been shown that state variable models may be reduced in order by matching the frequency responses if care is taken in choosing the frequency ranges. It has the advantage over modal methods that the eigenanalysis

is not required and that variables of interest can be retained in the reduced model without an additional equation. It has the disadvantage however, that the complete frequency response is required, which may in the case of very large systems prove impractical.

7.6 NOMENCLATURE

<u>A</u>	plant matrix
<u>B</u>	input matrix
<u>i</u>	$\sqrt{-1}$
<u>K</u>	number of frequencies fitted
<u>m</u>	reduced system order
<u>n</u>	full system order
<u>p</u>	number of inputs
<u>R</u>	defined by Eqs. (7.17) and (7.18)
<u>r_k</u>	real part of <u>x</u> (i ω_k)
<u>s</u>	Laplace operator
<u>u</u>	forcing vector
<u>u_i</u>	unit vector for the <i>i</i> th input
<u>X</u> (i ω)	defined by Eq. (7.7)

Greek:

<u>Φ</u>	defined by Eqs. (7.17) and (7.18)
<u>ϕ_k</u>	imaginary part of <u>x</u> (i ω_k)
<u>ω</u>	frequency

Superscript:

*	reduced system
T	transposed matrix

Transforms:

<u>x</u> (s)	Laplace transform of <u>x</u>
<u>x</u> (i ω)	frequency transform of <u>x</u>

CHAPTER 8

Conclusions and suggestions for further work

8. CONCLUSIONS AND SUGGESTIONS FOR FURTHER WORK

Each of the main points raised in previous chapters will be concluded here. A detailed discussion can be found with each separate topic.

8.1 DAVISON'S ZERO METHOD

Use of the Davison method for determining the zeroes of state variable models has shown that there is a difficulty concerned with the choice of the constant Γ . The value, 10^{15} , recommended by Davison has been shown to be inapplicable to some computer systems, and moreover if this value is used there may be significant errors in the predicted zeroes. If Γ is increased from 10^3 to 10^{30} and an individual zero monitored it is found to exhibit two distinct plateaux and a region of instability. The first plateau always gives the correct zero (at Loughborough when $\Gamma = 10^7$). As high a value of Γ as possible should be used before the unstable region is reached. To assist in noting this region two methods of monitoring the system stability have been developed.

8.2 A MODIFIED FORM OF THE LEVERRIER ALGORITHM

It has been shown that use of the Leverrier algorithm to find the numerator polynomials of the transfer function matrix can lead to numerical errors when systems of high order are analysed. These errors are not apparent when working with systems up to about tenth order. The algorithm has, however, been reformulated in terms of the inverse plant matrix, when the resulting

polynomial coefficients are simply related to those derived from the standard algorithm. Clearly the inverse problem suffers from the same numerical difficulties as the forward, but it has been shown that solving one problem from either end of the characteristic equation gives two sets of coefficients which may be combined to give one acceptable set: some coefficients being drawn from the forward, and some from the reverse problem.

8.3 FREQUENCY RESPONSE COMPUTATION OF STATE VARIABLE MODELS

A comparison has been made between the different methods of computing the frequency response of a state variable model. It has been shown that where an efficient eigenvalue routine exists, and the response is required over many frequencies, definite savings in computational time can be made by first transforming the model to the Jordan canonical form thus eliminating the need to invert a matrix, possibly complex, for each frequency considered. The method does however, require more core store than others. Solution by this method can sometimes lead to numerical inaccuracies in systems with a wide spread of eigenvalues. These errors may possibly be eliminated by placing double precision working in critical parts of the program.

An area needing further investigation is the performance of the method when analysing systems with

complex or multiple eigenvalues. To date only real systems have been examined.

8.4 SIMILARITIES BETWEEN THE MOMENTS AND CONTINUED FRACTION METHODS

It was shown numerically in Chapter 3 that these two methods when applied to a seventh order model predicted the same second order model, and it was demonstrated in Chapter 4 why this must be so. Thus the continued fraction method satisfies not only its own criterion but also has the physical significance of matching the lower moments of the impulse response. It is believed that the second order model is not a special case but that relationships of the form shown in Chapter 4 exist for all orders of model, provided that the moments method is applied to a model of the type fitted by Chen's method. It has however, at the time of writing, been impossible to rigorously prove this.

8.5 REDUCTION OF STATE VARIABLE MODELS BY MATCHING THE MOMENTS

The moments method of reducing the order of transfer functions has been successfully extended to the order reduction of state variable models. The reduced model retains the main characteristics of the full one, although in the case of systems with large inherent time delays the reduced model does not always closely fit the full one over the initial response. This

deviation is thought to be an inevitable outcome of simplification. The method has the advantage over some others in that there is a free choice of variables retained, and further, that neither the time or frequency response, nor an eigen-analysis are required. The reduced models represent well the full models for the inputs investigated, impulse and step functions. Other forcings of the reduced model have not been examined.

There are arising from the moments work a number of possible extensions.

8.5.1 Introduction of a time delay into a state model

One failure of the state variable model is its inability to adequately represent a time delay. The usual method of introducing a delay is to model it with a series of first order lags. This does, of course, increase the order of the model proportionally. It is proposed that it may be possible to include a delay based on the moments state variable simplification method.

The effect of including a delay in a model is to shift the response to the right on the time scale. Moments computed about the mean of an impulse response, other than the first, are obviously unaffected by such a shift and remain constant. Thus if the vectors of system moments for each input are computed, as described in Chapter 2, and then normalised about the mean the effect of including a time delay is to add it

to the first moment only. The vectors may then be converted back to unnormalised moments about the origin and formulated as described in Chapter 6. The problem is solved and \underline{A}^* , which may or may not be reduced, is the plant matrix with the time delay included.

8.5.2 Application of the moments method to unstable systems

One criticism of the moments method is that it cannot be used to reduce systems which may possess a pole on, or to the right of, the complex axis. It is thought that application of the shifting theorem may remedy this by moving the eigenvalues so that they all lie in the left half of the complex plane.

By moving the eigenvalues an amount α and substituting

$$\underline{\dot{y}} = e^{\alpha t} \underline{x} \quad (8.1)$$

the non stable system

$$\underline{\dot{x}} = \underline{A}x + \underline{B}u \quad (8.2)$$

may be transformed to the stable system

$$\underline{\dot{y}} = (\underline{A} + \alpha \underline{I})\underline{y} + e^{\alpha t} \underline{B}u \quad (8.3)$$

where the input has been modified to take account of the shift. Such a scheme for reduction is only possible if the system will respond accurately to a response other than the fitted impulse.

8.5.3 Fitting part of a response only

It was reported in Chapter 5 that by carefully grouping, and selecting, the system eigenvalues, Davison's method can be used to fit accurately only that portion of a response which is of interest, other parts being less closely fitted. It is proposed that such a fit be effected using the moments method, by making use of a generalized Laplace transform

$$F(s) = \int_{T_1}^{T_2} e^{-st} f(t) dt \quad (8.4)$$

where the only difference to the normal transform is the integration interval.

If an analysis identical to that of the moments method, for either a transfer function or state variable model, is carried out, it is found that moments corresponding to the finite integration interval can be computed quite simply from the moments of the infinite response, the cut off points, and the full model parameters. It is proposed that the simplified model be fitted to these "finite moments".

8.5.4 The approximate solution of partial differential equations using moments

If the moments method can be used to approximate transfer functions (the transform of an ordinary differential equation (ODE)) can it also be used to approximate the transform of a partial differential equation (PDE)?

Whilst PDE s are usually difficult to solve their solutions often gives rise to the standard responses, varying quite simply with the space parameter. The moments will also vary with this parameter in a similar manner. Clearly if the distribution of moments is known, they can be found at any point and a simple model fitted to them, thus giving a time solution at that point in space. As a PDE may be transformed to an ODE the distribution of moments in space is generally known.

8.6 REDUCTION OF STATE VARIABLE MODELS BY MATCHING THE FREQUENCY RESPONSE

It has been shown that a state variable model can be successfully reduced to a lower order by matching the frequency response of the reduced model to that of the full model. The method is straightforward but does require the storage of a lot of numerical data. Like the moments method the fit to inherent system time delays is not always satisfactory.

More work is required to find over what frequency range a particular model should be fitted and also whether or not the reduced model responds well to inputs other than step and impulse.

8.7 CHEN'S STATE VARIABLE METHOD

It has not been possible to investigate Chen's state variable method and therefore little can be concluded. As the author had, however, reached a similar point in his work some time ago but discarded

the method because of the difficulty of handling multiple inputs, other than by a least-squares method, it would be interesting to know how Chen has overcome the problem. Further, in view of the similarity between the continued fraction and moments transfer function work it would be of interest to know what, if any, comparisons can be made between the state variable methods.

8.8 CLOSING REMARKS

This work has shown that linear mathematical models can be successfully approximated by models of lower order, not only using rigid mathematical methods, some of which are alien to the engineer, but by making use of principles more readily understood.

There is still considerable interest in the field of model simplification and there remains a great deal more work still to be done, particularly in the area of non-linear and time varying systems. What is certain is that it is a fascinating topic!

CHAPTER 9

References

1. Akin, J.E., A note on control system model simplification, Int. J. Control, 14, 5, 989-993, (1971).
2. Anderson, J.H., Geometrical approach to the reduction of dynamical systems, Proc. IEE 114, 7, 1014-1018, (1967).
3. Anderson, J.H., Computational difficulty associated with the reduction of dynamic systems, Electronics Letters, 3, 10, 469 (1967).
4. Anderson, J.H., Reduction of dynamical systems when certain elements of the reduced system matrices are specified, Electronics Letters 3, 11, 504 - 505, (1967).
5. Anderson, J.H., Adjustment of the responses of reduced dynamical systems, Electronics Letters 4, 4, 75 - 76, (1968).
6. Anderson, J.H. and Kwan, H.W., The systematic reduction of complex process models; Two case studies, Int. Conf. on Computer Aided Design, Southampton, 15 - 18 April, (1969).
7. Aoki, M., Control of large scale dynamic systems by aggregation, IEEE. Trans. Automatic Control, AC-13, 3, (1968).
8. Ausman, I.S., Amplitude frequency response analysis and synthesis of unfactored transfer functions, ASME Trans., paper 62-NA-98, (1963).

9. Bass, R.W. and Gura, I., High order system design via state-space considerations, JACC preprints, p 311 Rensselaer Polytechnic, New York (1965).
10. Bollinger, R.E., Predicting fractionator dynamics using a frequency domain solution technique, Symp. on Computer and Mathematics for Engineering applications in industry, St. Louis (1968).
11. Bosley, M.J., Computation of frequency response from the state equations, Loughborough Univ. Chem. Eng. J., 5, 20 - 22 (1970).
12. Bosley, M.J., Kropholler, H.W., Lees, F.P. and Neale, R.M., The determination of transfer functions from state variable models, with comments by E.J. Davison and R.S. Morgan. Automatica 8, 213 - 218, (1972).
13. Bosley, M.J., Kropholler, H.W. and Lees, F.P., On the relation between the continued fraction expression and moments matching methods of model reduction, Submitted to Int. J. Control.
14. Bosley, M.J. and Lees, F.P., A survey of derivation of simple transfer functions from high order state variable models, Automatica, in press.
15. Bosley, M.J. and Lees, F.P., Methods for the reduction of high order state variable models to simple transfer functions models, IFAC Symp. on Digital Simulation of Continuous Processes, Győr, Hungary, paper B11, (1971).
16. Bosley, M.J. and Lees, F.P. The reduction of the order of state variable models using the method of moments, Submitted to Chem. Eng. Sci.

17. Brown, R.F., Reduction of order of linear state equations, *Electronics Letters* 4, 19, 416 - 417, (1968).
18. Buffham, B.A. and Gibilaro, L.G., A generalization of the tank in series mixing model, *AIChE Journal* 14, 5, 805 - 806, (1968).
19. Buffham, B.A. and Kropholler, H.W., Evaluation of the exponential matrix, Symp. on On-line computer methods relevant to Chemical Engineering, Univ. Nottingham, Sept. (1971).
20. Chandrasekharan, P.C. and Balakrishnan, K.V., Reduction of order of linear dynamic systems, *J. Inst. Engineering (India), Electronics and Telecommunications* 52, pt. EII, 1, 43 - 45 (1971).
21. Chen, C.F., A matrix cauer model in multivariable systems analysis and design, 4th UKAC Control Convention, Multivariable control system design and applications, 66 - 70, Manchester, 1 - 3 Sept., (1972).
22. Chen, C.F. Model reduction of multivariable control systems by means of matrix continued fractions, 5th IFAC World Congress, Paris, preprint 35.1, 12 - 17 June, (1972).
23. Chen, C.F. and Haas, I.J., Elements of Control Systems Analysis, Prentice-Hall, Englewood Cliffs, N.J. (1968).
24. Chen, C.F. and Knox, A.E., An identifier, *Int. J. Electronics*, 24, 6, 521 - 533, (1968).

25. Chen, C.F. and Philip, B.L., Graphical determination of transfer function coefficients of a system from its frequency response, IEEE. Trans. on Applications and Industry 82, 42 - 45, (1963).
26. Chen, C.F. and Philip, B.L., Accurate determination of complex root transfer functions from frequency response data, IEEE Trans. Automatic Control AC-10, 356 - 358, (1965).
27. Chen, C.F. and Shieh, L.S., A novel approach to linear model simplification, Int. J. Control 8, 6, 561 - 570, (1968).
28. Chen, C.F. and Shieh, L.S., Continued fraction inversion by Routh's Algorithm, IEEE Trans. Circuit Theory CT-16, 2, 197 - 202, (1969).
29. Chen, C.F. and Shieh, L.S., An algebraic method for control system design, Int. J. Control 11, 5, 717 - 739, (1970).
30. Chen, C.T., A formula and algorithm for continued fraction inversion, Proc. IEEE 57, 10, 1780 - 1781, (1969).
31. Chidambara, M.R., Two simple techniques for the simplification of large dynamic systems, Proc. Joint Automatic Control Conf., Boulder, Colorado, 669 - 674, (1969).
32. Chidambara, M.R. and Davison, E.J., On a method for simplifying linear dynamic systems, IEEE Trans. Automatic Control, AC-12, 119 - 121, (1967).

33. Chidambara, M.R. and Davison, E.J., Further remarks on simplifying linear dynamic systems, IEEE Trans. Automatic Control AC-12, 213 - 214 (1967).
34. Chidambara, M.R. and Davison, E.J. Further comments on a method for simplifying linear dynamic systems, IEEE Trans. Automatic Control AC-12, 799 - 780, (1967).
35. Chuang, S.C., Application of continued fraction method for modelling transfer functions to give more accurate initial transient response, Electronics Letters 6, 26, 861 - 863, (1971).
36. Coggan, G.C. and Wilson, J.A., Approximate models and the elimination of bias in on-line estimation of industrial processes, Preprints B.C.S./Inst. Ch.E. Symp. On-line computer methods relevant to Chemical Engineering, 136 - 150, 22 Sept., (1971).
37. Conte, S.D., Elementary Numerical Analysis, McGraw Hill Book Co., New York, (1965).
38. Cowherd, W.F. and Cadman, T.W., Numerical prediction of transfer functions, Instruments and Control Systems 42, 5, 109 - 113, (1969).
39. Davison, E.J., A method for simplifying linear dynamic systems, IEEE Trans. Automatic Control AC-11, 1, 93 - 101, (1966).
40. Davison, E.J., The numerical solution of large systems of linear differential equations, AIChE, Journal 14, 1, 46 - 50, (1968).
41. Davison, E.J., The simplification of large linear systems, Control 12, 418 - 419, (1968).

42. Davison, E.J., A new method for simplifying large linear dynamic systems, IEEE Trans. Automatic Control VAC-13, 214 - 215, (1968).
43. Davison, E.J., On the calculation of zeroes of a linear constant system, IEEE Trans. Circuit Theory CT-18, 1, 183 - 184, (1971).
44. De Sarkar, A.K. and Dharma Rao, N., Dynamic system simplification and an application to power system stability studies, Proc. IEE 119, 7, 904 - 910, (1972).
45. Dudnikov, E.E., *Automatika i Telemekhanika* 20, 5, 576, (1959).
46. Evans, W.R. Control system dynamics, McGraw-Hill Book Co., New York, (1954).
47. Faddeev, D.K. and Faddeeva, V.N., Computational methods of Linear algebra, Freeman, London, (1963)
48. Fossard, A.J., Sur la simplification des modeles lineaires, *Automatisme*, 15, 4, 141 - 147, (1970).
49. Frame, J.S., Matrix functions and applications, Pt. IV. Matrix function and constituent matrices, IEEE Spectrum 1, 123 - 131, (1964).
50. Freund, E., The reduction of multivariable systems Proc. 5th Hawai Int. Conf. on System Science, Honolulu, Hawai, 10 - 12, 11-13 Jan, (1972).
51. Gaisyenyuk, B.S., The reduction of the order of a transfer function in automatic systems (in russian), *Izvestiyn Vyshikh Uchebnikh Zavvedyeni-Electro-mechanika* 11, 1241 - 1246, (1969).

52. Ghani, F. and Ackroyd, M.H., Computing transfer functions from state space matrices, *Electronic Letters*, 7, 17, 487 - 489, (1971).
53. Gibilaro, L.G., Models for mixing in stirred vessels, Ph.D Thesis, Loughborough University of Technology, (1967).
54. Gibilaro, L.G. and Lees, F.P., The reduction of complex transfer function models to simple models using the method of moments, *Chem. Eng. Sci.* 24, 85 - 93, (1969).
55. Gill, S., A process for the step by step integration of differential equations in an automatic digital computing machine, *Proc. Camb. Phil. Soc.*, 47, 96 - 108, (1950).
56. Golub, G., Numerical methods for solving linear least-squares problems. *Numerische Mathematik* 7, ISS3, 206 - 216, (1965).
57. Graham, E.U. and Strauss, J.C., Simplification of dynamic system models using parameter estimation, *Proc. 2nd Princeton Conf. on Information Systems and Science*.
58. Harriott, P., Process Control, McGraw-Hill Book Co., New York, (1964).
59. Hsia, T.C., On the simplification of linear systems *IEEE Trans. Automatic Control* AC-17, 372 - 374, (1972).
60. Iinoya, K. and Altpeter, R.J., Inverse responses in process control, *Ind. Eng. Chem.* 54, 7, 39 - 43, (1962).

61. Johnson, E.F., Automatic process control, McGraw-Hill Book Co., New York, (1962).
62. Johnson, J.L., Fan, L.T. and Wu, Y.S., Comparison of moments, s-plane and frequency response methods for analyzing pulse testing data from flow systems, Ind. Eng. Chem. Proc. Des. Dev. 10, 425 - 431, (1971).
63. Kalyaev, A.V., Computing the transient response in linear systems by the method of reducing the order of the differential equation, *Automatika i Telemekhanika* 20, 9, 1141 - 1150, (In English), (1959).
64. Kardashov, A.A., An analysis of the quality of an automatic control system by the method of lowering the order of the differential equation, *Automation and Remote Control* 24, 8, 978 - 988, (1963).
65. Kendall, M.G. and Stuart, A., The advanced theory of statistics, Vol. 1, 67 - 71, Charles Griffin, London, (1958).
66. Khovanskii, A.N., Application of continued fractions and their generalization to problems of approximate analysis, Technical-Theoretical Literature Publishers, Moscow, (1956).
67. Klinkenberg, A., Distribution of residence times in a cascade of mixed vessels with backmixing, Ind. Eng. Chem. Fund. 5, 283 - 285, (1966).
68. Kropholler, H.W., The determination of relative variance and other moments for generalized flow networks or system transfer functions, Ind. Eng. Chem. Fund. 9, 3, 329 - 333, (1970).

69. Kropholler, H.W. and Neale, R.M., Determination of the zeroes of a transfer function, 2nd IFAC Symp on Multivariable Technical Control Systems, Dusseldorf, (1971).
70. Kuo, J.C.W. and Wei, J., A lumping analysis in monomolecular reaction systems - Analysis of the exactly and approximately lumpable systems (2 papers), I & EC Fund. 8, 1, 114 - 133, (1969).
71. Kuppurajulu, A. and Elangovan, S., System analysis by simplified models, IEEE Trans. Automatic Control AC-15, 234 - 237, (1970).
72. Lamb, D.E. and Rippin, D.W.T., A theoretical study of the dynamics and control of binary distillation columns, 53rd annual AIChE meeting, Washington, (1960).
73. Lanczos, C., Applied analysis, Pitman, London, (1957).
74. Lapidus, L., Digital computation for chemical engineers, 346 - 354, McGraw-Hill Book Co., New York, (1962).
75. Lees, F.P., The determination of the moments of the impulse response of chemical processes from the basic transformed equations. Chem. Eng. Sci. 24, 1607 - 1613, (1969).
76. Lees, F.P., Unsteady state models of gas absorption columns, Ph.D. Thesis, Loughborough University of Technology, (1969).

77. Lees, F. P., The derivation of simple transfer function models of oscillating and inverting processes from the basic transformed equations using the method of moments, Chem. Eng. Sci. 26, 1179 - 1186, (1971).
78. Leverrier, U.J.J., Sur les variations seculaire des elements des orbites, J. Math., (1840).
79. Levy, E.C., Complex curve fitting, IRE Trans. Automatic Control AC-4, 37 - 43, (1959).
80. Linrill, J.G., The selection of network functions to approximate prescribed frequency characteristics, Tech. Dept. 145, MIT Research Lab. of Electronics, Cambridge, Mass., (1950).
81. Luenberger, D.G. and Meier, L., Approximation of linear constant systems, Joint Automatic Control Conf., Seattle, Wash., Aug 17 - 19, 728 - 735, (1966).
82. Marshall, S. A., An approximate method for reducing the order of a linear system, Control 10, 642 - 643, (1966).
83. Marshall, S. A., The reduction of the order of linear dynamical systems, Inst. Meas. Control/ United Simulation Council meeting on Reduction of system models to usable dimensions, CEGB London, 16 Nov, (1971).
84. Marshall, S.A., Remarks on computing the zeroes of large systems, IEEE Trans. Automatic Control AC-17, 261, (1972).

85. Matsubura, M., On the equivalent deadtime, IEEE Trans. Automatic Control AC-10, 464 - 466, (1965).
86. Mayne, D., An algorithm for the calculation of the pseudo-inverse of a singular matrix, Computer J. 9, 312 - 317, (1966).
87. Meier, L. and Luenberger, D.G., Approximation of linear constant systems, IEEE Trans. Automatic Control AC-12, 585 - 588, (1967).
88. Melsa, I. L., Computer programs for computational assistance in the study of linear control theory, McGraw Hill Book Co., New York, (1970).
89. Mitra, D., On the reduction of complexity of linear dynamical models, UKAEA, Rep. R520, Winfrith, (1967).
90. Mitra, D., On the reduction of complexity of linear dynamical models - computer studies, UKAEA Rep. R535, Winfrith, (1967).
91. Mitra, D., The reduction of complexity of linear time invariant dynamical systems, IFAC World Congress, Warsaw, 19 - 33, (1969).
92. Mitra, D., W matrix and the geometry of model eigenvalue and reductions, Proc. IEE 116, 6, 1101 - 1106, (1969).
93. Mitra, D., Analytical results on the use of reduced models in the control of linear dynamical systems, Proc. IEE 116, 8, 1439 - 1444, (1969).
94. Morgan, B.S., Sensitivity analysis and synthesis of multivariable systems, IEEE Trans. Automatic Control AC-11, 3, 506 - 512, (1966).

95. Nagarajan, R., Optimum reduction of large dynamic systems, Int. J. Control 14, 6, 1169 - 1174, (1971).
96. Naslin, P., Dynamics of linear and non-linear systems, Blackie, London, (1965).
97. Neale, R.M., The dynamics of multistage systems, Ph.D Thesis, Loughborough University of Technology (1971).
98. Nicholson, H., Dynamic optimisation of a boiler, Proc. IEE 111, 8, 1479 - 1499, (1964).
99. Nicholson, H. and Anderson, J.H., Geometrical approach to reduction of dynamical systems, Proc. IEE 115, 2, 361 - 363, (1968).
100. Ogata, S., State Space Analysis of Control Systems Prentice-Hall, Englewood Cliffs, N.J., (1967).
101. Pang, K.H. and Johnson, A.I., The determination of theoretical frequency response of some stagewise processes, Can. J. Chem. Eng. 47, 477 - 482, (1969).
102. Payne, P.A., An improved technique for transfer function synthesis from frequency response data, IEEE Trans. Automatic Control AC-15, 480 - 483, (1970).
103. Paynter, H.M., On an analogy between stochastic processes and monotone dynamic systems, in G. Malker (ed.) Regelungstechnik moderne theorien und ihre verwendbarkeit, R. Oldesbourg, Verlag, 243 - 250, (1956).
104. Rosenbrock, H.H., Transfer matrix of linear dynamic systems, Electronics Letters 1, 95 - 96, (1965).

105. Rosenbrock, H.H., State Space and multivariable theory, Nelson, London, (1970).
106. Sanathanan, C.K. and Koerner, J., Transfer function synthesis as a ratio of two complex polynomials, IEEE Trans. Automatic Control AC-8, 56 - 58, (1963).
107. Shieh, L.S., Chen, C.F., and Huang, C.J., Simple methods for identifying linear systems from frequency and time response data, Int. J. Control 13, 6, 1027 - 1039, (1971).
108. Shunta, J.P. and Luyben, W.L., Comparison of stepping and general complex matrix inversion techniques in calculating the frequency response of binary distillation columns, Ind. Eng. Chem. Fund. 8, 4, 838 - 840, (1969).
109. Sinha, N.K. and Bereznoi, G.T., Optimum approximation of high-order systems by low-order models, Int. J. Control 14, 5, 951 - 959, (1971).
110. Sinha, N.K. and Pille, W., A new method for the reduction of dynamic systems, Int. J. Control 14, 1, 111 - 118, (1971).
111. Souriau, J. M., Une methode pour la decomposition spectrale et l'inversion des matrices, C. R. Acad. Sci. 227, 1010 - 1011, (1948).
112. Strobel, H., On a new method of determining the transfer function by simultaneous evaluation of the real and imaginary parts of the measured frequency response, 3rd IFAC Conf., London, paper 1F, (1966).

113. Sumner, A., A method of fitting rational transfer functions to frequency responses, Proc. Inst. Mech. Engrs. 179, Pt. 3H, 132 - 137, (1964-65).
114. Tether, A.J., Construction of minimal linear state-variable models from finite input-output data, IEEE. Trans. Automatic Control AC-15, 427 - 436, (1970).
115. Towill, D.R., Transfer function techniques for control engineers, Iliffe, London, (1970).
116. Towill, D.R. and Mehdi, Z., Prediction of the transient response sensitivity of high order linear systems using low order models, Measurement and Control 3, T1 - T9, (1970).
117. Truxal, J.G., Control system synthesis, McGraw-Hill Book Co., New York, (1955).
118. Wall, H.S., Analytic theory of continued fractions, 182, Van Nostrand, New York, (1948).
119. White, G.W.T., The calculation of pole-zero patterns for process control problems, Trans. Soc. Instrument. Tech., 18, 3, 118 - 122, (1966).
120. Wilkinson, J.H., The algebraic eigenvalue problem, Oxford Univ. Press, (1965).
121. Wilson, D.A., Optimum solution of model reduction problems, Proc. IEE 117, 6, 1161 - 1165, (1970).
122. Woods, R.M., The frequency response of multicomponent distillation columns, Trans. Inst. Chem. Engrs. 45, T190 - T195, (1967).
123. Young, J.F., Determination of transfer function coefficients, Control, 10, 5, 246 - 248, (1966).

APPENDIX 1

A1.1 A listing of the Levy program.

```

MASTER LEVY WITH S/K MODS
REAL IMAGT(200)
REAL IANDA(40),IMAG(200)
COMPLEX CZ
DIMENSION WL(200),S(40),W(200),P(200),T(41),U(40),A(1600)
DIMENSION B(40),X(40),AA(1600)
DIMENSION RT(200)
EQUIVALENCE (AA(1),IMAGT(1)),(AA(201),RT(1))
COMMON NOPRINT

```

```

*****
C
C THIS PROGRAM READS IN FREQUENCY RESPONSE DATA AND FITS TO IT A
C TRANSFER FUNCTION (POLYNOMIAL RATIO) BY A LEAST SQUARES TYPE
C METHOD. THE PROGRAM IS BASED ON THE LEVY METHOD,REF...

```

```

C
C SANATHANAN & KOERNER,IEEE TRANS V. AC-8,P.56,JAN 196

```

```

C DATA REQUIRED
C IRUN=NUMBER OF DIFFERENT ORDER TRANSFER FUNCTION TO BE FITTED
C TO EACH FREQUENCY RESPONSE.
C NF=NUMBER OF FREQUENCIES CONSIDERED
C MODE...FREQUENCY RESPONSE DATA MAY EITHER BE READ IN AS THE REAL
C AND IMAGINARY PARTS (MODE=0) OR AS THE AMPLITUDE RATIO AND PHASE
C DIFFERENCE(MODE=1)
C F=FREQUENCY
C R=REAL PART(MODE=0) OR AMPLITUDE RATIO(MODE=1)
C IMAG=IMAGINARY PART (MODE=0) OR THE PHASE DIFFERENCE(MODE=1)
C N=ORDER OF THE NUMERATOR POLYNOMIAL.
C D=ORDER OF DENOMINATOR POLYNOMIAL
C NIT=NUMBER OF ITERATIONS TO BE MADE

```

```

C PROGRAM WRITTEN BY:
C N.J. BOSLEY,
C DEPT. CHEMICAL ENGINEERING,
C LOUGHBOROUGH UNIVERSITY OF TECHNOLOGY.

```

```

*****
C
C THIS SECTION READS IN DATA ,CONVERTS IT TO THE REQUIRED FORM
C AND WRITES A DATA CHECK

```

```

C READ(1,61)IRUN
C READ(1,61)NF
C READ(1,61)MODE
C READ(1,62)(W(I),R(I),IMAG(I),I=1,NF)
C NOPRINT=0
C WRITE(2,72)
C IF(MODE.EQ.0)GO TO 120
C NOPRINT=1

```

```

3001 CALL LEVY4(R,IMAG,W,NF)
120 CONTINUE
C WRITE(2,18)
C DO 74 I=1,NF
74 WRITE(2,75)I,R(I),IMAG(I),W(I)
C WRITE(2,2050)
C DO 2000 KIR=1,IRUN
C READ(1,61)M,N,NIT
C WRITE(2,2001)M,N

```

```

      DO 1 K=1,NF
1    WL(K)=1
C
C    THIS SECTION COMPUTES THE SUMMATIONS S,T,U, AND LAMDA
C
      DO 110 ITER=1,NIT
        J=-1
        DO 2 I=1,2*N+1
          J=-J
          S(I)=0
          DO 3 K=1,NF
3          S(I)=S(I)+W(K)**(I-1)*R(K)+WL(K)
            IF(J)4,5,6
          4 T(I)=0
            DO 7 K=1,NF
7          T(I)=T(I)+W(K)**(I-1)*IMAG(K)+WL(K)
            GO TO 2
          5 LAMDA(I),U(I)=0
            DO 8 K=1,NF
              LAMDA(I)=LAMDA(I)+W(K)**(I-1)*WL(K)
              U(I)=U(I)+(R(K)+R(K)+IMAG(K)+IMAG(K))*W(K)**(I-1)+WL(K)
            2 CONTINUE
C
C    THE METHOD SETS UP THE MATRIX EQUATION
C                                     AB=C
C    THIS SECTION SETS UP THE A MATRIX
C
      NA=N+M+1
      DO 14 I=1,NA+NA
14     A(I)=0
        J2=-1
        DO 15 I=1,M+1
          J2=-J2
          IF(J2.EQ.-1)K=I+1
          IF(J2.EQ.1)K=I
          J1=-1
          IF(J2)16,5,17
17     DO 23 J=1,M+1,2
          J1=-J1
          A(I+NA+(J-1))=J1+LAMDA(K)
23     K=K+2
          GO TO 15
16     DO 19 J=2,M+1,2
          J1=-J1
          A(I+NA+(J-1))=J1+LAMDA(K)
19     K=K+2
15     CONTINUE
        J2=1
        K=3
        DO 25 I=M+2,NA
          IF(J2.LT.0)K=K+2
          K1=K
          J2=-J2
          J1=-1
          IF(J2)21,5,22
21     DO 23 J=M+2,NA,2
          J1=-J1
          A(I+NA+(J-1))=U(K)+J1
23     K=K+2
          GO TO 25
22     DO 24 J=M+3,NA,2
          J1=-J1
          A(I+NA+(J-1))=U(K)+J1

```

```

24 K=K+2
25 K=K1
   J1=-1
   DO 26 J=1,M+1
     J1=-J1
     K=I+1
     IF(J1)28,5,29
26 J2=-1
   DO 27 J=M+2,NA,2
     J2=-J2
     A(I+NA+(J-1))=J2+T(K)
     IF(J.EQ.NA)GO TO 27
     A(I+NA+J)=J2+S(K+1)
27 K=K+2
   GO TO 26
28 J2=-1
   DO 30 J=M+2,NA,2
     A(I+NA+(J-1))=J2+S(K)
     IF(J.EQ.NA)GO TO 30
     A(I+NA+J)=-J2+T(K+1)
     J2=-J2
30 K=K+2
29 CONTINUE
   J1=-1
   K1=1
   DO 31 I=M+2,NA
     K1=K1+1
     K=K1
     J1=-J1
     IF(J1)32,5,86
86 J2=-1
   DO 87 J=1,M+1,2
     J2=-J2
     A(I+NA+(J-1))=J2+T(K)
     IF(J.EQ.M+1)GO TO 87
     A(I+NA+J)=-J2+S(K+1)
87 K=K+2
   GO TO 31
32 J2=-1
   DO 88 J=1,M+1,2
     J2=-J2
     A(I+NA+(J-1))=J2+S(K)
     IF(J.EQ.M+1)GO TO 88
     A(I+NA+J)=J2+T(K+1)
88 K=K+2
31 CONTINUE

```

```

C
C   THIS SECTION SETS UP THE C VECTOR
C

```

```

   J1=-1
   K=1
   DO 33 J=1,M+1
     J1=-J1
     IF(J1)34,5,35
35 C(I)=S(K)
   GO TO 33
34 C(I)=T(K)
33 K=K+1
   DO 40 I=M+2,NA
36 C(I)=0
   K=2
   DO 40 I=M+3,NA,2
     C(I)=U(K)

```

```

10 K=K+2
C
C THIS SECTION SOLVES THE MATRIX EQUATION & STORES THE PARAMETERS
C IN ARRAY X.
C
NA1=NA+NA
IN=1
CALL F4=CSL(A,B,NA,NA1,NA,IN,X,D,1D,IT,AA,S,T)
IF(IT)3A,37,38
3A WRITE(2,39)
GO TO 5
37 WRITE(2,40)
GO TO 5
C
C THIS SECTION WRITES OUT THE PARAMETERS, COMPUTES THE EFFECT OF FI
C AND CALCULATES THE WEIGHING FUNCTION TO BE USED ON THE
C NEXT ITERATION
C
32 WRITE(2,41)ITER
WRITE(2,42)
WRITE(2,43)(X(I),I=1,M+1)
WRITE(2,44)
N1=1
WRITE(2,45)N1,(X(I),I=M+2,NA)
DO 60 J=1,NF
J=NA+1
IF(N.F0.1)GO TO 101
CZ=CMPLX(X(J),X(NA)*W(K))
IF(N.F0.2)GO TO 100
DO 71 I=1,N-2
J=J+1
71 CZ=CZ*CMPLX(0.,W(K))+X(J)
100 CONTINUE
CZ=CZ*CMPLA(0.,W(K))+1
60 W(K)=1/(REAL(CZ)+REAL(CZ)+AIMAG(CZ)* AIMAG(CZ))
CALL F5VY2(W,NF,NA,N,M,X,RT,IMAGT,R,IMAG,WL)
110 CONTINUE
5 WRITE(2,2002)
GO TO 2000
101 CZ=CMPLX(1.,W(K)*X(NA))
GO TO 60
2000 CONTINUE
STOP
61 FORMAT(310)
62 FORMAT(350,0)
30 FORMAT(///21H A MATRIX IS SINGULAR)
40 FORMAT(///28H A MATRIX IS ILL-CONDITIONED)
41 FORMAT(32H PARAMETERS DERIVED ON ITERATION,13)
42 FORMAT(///62H NUMERATOR COEFFICIENTS STARTING WITH ZEROth IN ASCEN-
ding order///)
43 FORMAT(24H16,6)
44 FORMAT(///68H DENOMINATOR COEFFICIENTS STARTING WITH THE ZEROth IN
ascending order///)
18 FORMAT(57H0 REAL IMAGINARY FREQUEN
cy)
72 FORMAT(16H1FULL MODEL DATA/)
75 FORMAT(13,3(4X,E14,5))
2001 FORMAT(24H REDUCED ORDERS REQUIRED/12H NUMERATOR =,13/
1 12H DENOMINATOR =,13)
2002 FORMAT(27H TRY USING DIFFERENT ORDERS)
2050 FORMAT(1H1)
END

```

```

SUBROUTINE LEVY2(W,NF,NA,N,M,X,RT,IMAGT,R,IMAG,WL)
REAL IMAG(NF),IMAGT(NF)
COMPLEX ENUM,DENOM,ERROR
DIMENSION W(NF),X(NA),RT(NF),R(NF),WL(NF)

```

```

C
C THIS SEGMENT GIVEN THE PARAMETERS OF THE POLYNOMIAL COMPUTES AND
C WRITES THE FREQUENCY RESPONSE AND COMPUTES FROM THIS THE ERROR
C BETWEEN THE ORIGINAL DATA AND THAT GIVEN BY THE FITTED MODEL
C

```

```

WRITE(2,6)
WRITE(2,2)
WRITE(2,1)
TOTERR=0
DO 3 I=1,NF
  IF(N.F0.0)GO TO 10
  A=X(I+1)*W(K)
  A1=X(M)
  ENUM=CMPLX(A1,A)
  I=M
  IF(M.F0.1)GO TO 11
  DO 4 I1=1,M-1
    I=M-I1
  4 ENUM=ENUM+CMPLX(0.,W(K))+X(I)
11 CONTINUE
  IF(N.F0.1)GO TO 12
  A=X(NA)*W(K)
  A1=X(MA-1)
  DENOM=CMPLX(A1,A)
  I=MA-1
  IF(N.F0.2)GO TO 13
  DO 5 I1=N-2
    I=I-I1
  5 DENOM=DENOM+CMPLX(0.,W(K))+X(I)
13 CONTINUE
  DENOM=DENOM+CMPLX(0.,W(K))+1
  ERROR=CMPLX(R(K),IMAG(K))*DENOM-ENUM
  TOTERR=TOTERR+WL(K)*(REAL(ERROR)*REAL(ERROR)+AIMAG(ERROR)
  1 *AIMAG(ERROR))
  IMAGT(K)=AIMAG(ENUM/DENOM)
  RT(K)=REAL(ENUM/DENOM)
  3 WRITE(2,7)K,RT(K),IMAGT(K),W(K)
  WRITE(2,20)TOTERR
RETURN
10 ENUM=CMPLX(X(1),0.)
GO TO 11
12 DENOM=CMPLX(X(NA),0.)
GO TO 13
1  FORMAT(50H0          REAL          IMAGINARY          FREQUENCY
  1)
2  FORMAT(66H CALCULATED RESULTS FROM LEVY POLYNOMIAL RATIO)
4  FORMAT(////)
7  FORMAT(13,3(4X,E14.5))
20  FORMAT(////33H TOTAL ERROR IN COMPLEX FITTING =,E16.6,////)
END

```



```

SUBROUTINE LEVY4(X,Y,W,NO)
DIMENSION X(NO),Y(NO),W(NO)
COMMON NOPRINT

```

```

THIS SEGMENT GIVEN THE FREQUENCY RESPONSE IN THE FORM OF AMPLITUDE
RATIO AND PHASE DIFFERENCE CONVERTS IT TO THE EQUIVALENT REAL
AND IMAGINARY PARTS

```

```

IF(NOPRINT.EQ.1)GO TO 20
WRITE(2,1)

```

```

1 FORMAT(52H0          GAIN          ANGLE          FREQUENCY

```

```

1Y)

```

```

20 CONTINUE

```

```

DO 10 I=1,NO

```

```

IF(NOPRINT.EQ.1)GO TO 21

```

```

WRITE(2,4)I,X(I),Y(I),W(I)

```

```

4 FORMAT(13,4X,E14.5,4X,E14.5,4X,E14.5)

```

```

21 CONTINUE

```

```

Y(I)=Y(I)*3.14159/180.

```

```

YS=Y(I)

```

```

XS=X(I)

```

```

X(I)=XS*COS(YS)

```

```

Y(I)=XS*SIN(YS)

```

```

10 CONTINUE

```

```

RETURN

```

```

END

```

APPENDIX 1

A1.2 Solution of the illustrative example
by Levy's method.

A SAMPLE OUTPUT FROM LEVY'S METHOD

FULL MODEL DATA

	REAL	IMAGINARY	FREQUENCY
1	0.11111E 00	0.25370E-04	0.10000E-02
2	0.11125E 00	0.23803E-03	0.10000E-01
3	0.11164E 00	0.38540E-03	0.20000E-01
4	0.11299E 00	0.15396E-03	0.40000E-01
5	0.11460E 00	-0.90075E-03	0.60000E-01
6	0.11722E 00	-0.47805E-02	0.10000E 00
7	0.11884E 00	-0.17074E-01	0.20000E 00
8	0.11811E 00	-0.23051E-01	0.25000E 00
9	0.11293E 00	-0.39484E-01	0.40000E 00
10	0.94466E-01	-0.65838E-01	0.70000E 00
11	0.69786E-01	-0.82524E-01	0.10000E 01
12	0.27421E-01	-0.88571E-01	0.15000E 01
13	-0.37641E-02	-0.76069E-01	0.20000E 01
14	-0.27200E-01	-0.40969E-01	0.30000E 01
15	-0.26751E-01	-0.18628E-01	0.40000E 01
16	-0.16767E-01	-0.15017E-02	0.40000E 01

REDUCED ORDERS REQUIRED

NUMERATOR = 1

DENOMINATOR = 2

PARAMETERS DERIVED ON ITERATION 1

NUMERATOR COEFFICIENTS STARTING WITH ZEROth IN ASCENDING ORDER

0.11599E 00 -0.10512E-01

DENOMINATOR COEFFICIENTS STARTING WITH THE ZEROth IN ASCENDING ORDER

0.10000E 01 0.75870E 00 0.20357E 00

CALCULATED RESULTS FROM LEVY POLYNOMIAL RATIO

	REAL	IMAGINARY	FREQUENCY
1	0.11599E 00	-0.98515E-04	0.10000E-02
2	0.11599E 00	-0.98513E-03	0.10000E-01
3	0.11597E 00	-0.19701E-02	0.20000E-01
4	0.11591E 00	-0.39394E-02	0.40000E-01
5	0.11581E 00	-0.59068E-02	0.60000E-01
6	0.11548E 00	-0.98327E-02	0.10000E 00
7	0.11395E 00	-0.19553E-01	0.20000E 00
8	0.11281E 00	-0.24335E-01	0.25000E 00
9	0.10791E 00	-0.38199E-01	0.40000E 00
10	0.92001E-01	-0.62440E-01	0.70000E 00
11	0.69758E-01	-0.79653E-01	0.10000E 01
12	0.28269E-01	-0.88457E-01	0.15000E 01

13	-0.44342E-02	-0.76982E-01	0.20000E 01
14	-0.28656E-01	-0.40483E-01	0.30000E 01
15	-0.27223E-01	-0.17973E-01	0.40000E 01
16	-0.16803E-01	-0.21201E-02	0.60000E 01

TOTAL ERROR IN COMPLEX FITTING = 0.270211E-03

PARAMETERS DERIVED ON ITERATION 2

NUMERATOR COEFFICIENTS STARTING WITH ZEROth IN ASCENDING ORDER

0.116005E 00 -0.942490E-02

DENOMINATOR COEFFICIENTS STARTING WITH THE ZEROth IN ASCENDING ORDER

0.100000E 01 0.754798E 00 0.212374E 00

CALCULATED RESULTS FROM LEVY POLYNOMIAL RATIO

	REAL	IMAGINARY	FREQUENCY
1	0.11601E 00	-0.96985E-04	0.10000E-02
2	0.11600E 00	-0.96984E-03	0.10000E-01
3	0.11599E 00	-0.19396E-02	0.20000E-01
4	0.11593E 00	-0.38784E-02	0.40000E-01
5	0.11583E 00	-0.58157E-02	0.60000E-01
6	0.11552E 00	-0.96875E-02	0.10000E 00
7	0.11407E 00	-0.19268E-01	0.20000E 00
8	0.11298E 00	-0.23993E-01	0.25000E 00
9	0.10829E 00	-0.37747E-01	0.40000E 00
10	0.92846E-01	-0.62118E-01	0.70000E 00
11	0.70798E-01	-0.79813E-01	0.10000E 01
12	0.28669E-01	-0.89238E-01	0.15000E 01
13	-0.67778E-02	-0.77321E-01	0.20000E 01
14	-0.28490E-01	-0.39763E-01	0.30000E 01
15	-0.26369E-01	-0.17479E-01	0.40000E 01
16	-0.15880E-01	-0.23127E-02	0.60000E 01

TOTAL ERROR IN COMPLEX FITTING = 0.264640E-03

PARAMETERS DERIVED ON ITERATION 3

NUMERATOR COEFFICIENTS STARTING WITH ZEROth IN ASCENDING ORDER

0.116012E 00 -0.940043E-02

DENOMINATOR COEFFICIENTS STARTING WITH THE ZEROth IN ASCENDING ORDER

0.100000E 01 0.755037E 00 0.212638E 00

CALCULATED RESULTS FROM LEVY POLYNOMIAL RATIO

	REAL	IMAGINARY	FREQUENCY
1	0.11601E 00	-0.96994E-04	0.10000E-02
2	0.11601E 00	-0.96992E-03	0.10000E-01
3	0.11599E 00	-0.19397E-02	0.20000E-01
4	0.11593E 00	-0.38787E-02	0.40000E-01
5	0.11584E 00	-0.58161E-02	0.60000E-01
6	0.11553E 00	-0.96833E-02	0.10000E 00
7	0.11407E 00	-0.19270E-01	0.20000E 00
8	0.11298E 00	-0.23996E-01	0.25000E 00
9	0.10829E 00	-0.37751E-01	0.40000E 00
10	0.02850E-01	-0.62127E-01	0.70000E 00
11	0.20793E-01	-0.79826E-01	0.10000E 01
12	0.28647E-01	-0.89241E-01	0.15000E 01
13	-0.48000E-02	-0.77300E-01	0.20000E 01
14	-0.28477E-01	-0.39729E-01	0.30000E 01
15	-0.26340E-01	-0.17462E-01	0.40000E 01
16	-0.15855E-01	-0.23175E-02	0.60000E 01

TOTAL ERROR IN COMPLEX FITTING = 0.264626E-03

PARAMETERS DERIVED ON ITERATION 4

NUMERATOR COEFFICIENTS STARTING WITH ZEROth IN ASCENDING ORDER

0.116012E 00 -0.939950E-02

DENOMINATOR COEFFICIENTS STARTING WITH THE ZEROth IN ASCENDING ORDER

0.100000E 01 0.755037E 00 0.212649E 00

CALCULATED RESULTS FROM LEVY POLYNOMIAL RATIO

	REAL	IMAGINARY	FREQUENCY
1	0.11601E 00	-0.96993E-04	0.10000E-02
2	0.11601E 00	-0.96991E-03	0.10000E-01
3	0.11599E 00	-0.19397E-02	0.20000E-01
4	0.11593E 00	-0.38787E-02	0.40000E-01
5	0.11584E 00	-0.58161E-02	0.60000E-01
6	0.11553E 00	-0.96832E-02	0.10000E 00
7	0.11407E 00	-0.19270E-01	0.20000E 00
8	0.11298E 00	-0.23996E-01	0.25000E 00
9	0.11829E 00	-0.37751E-01	0.40000E 00
10	0.92851E-01	-0.62127E-01	0.70000E 00
11	0.70794E-01	-0.79827E-01	0.10000E 01
12	0.28647E-01	-0.89242E-01	0.15000E 01
13	-0.48012E-02	-0.77300E-01	0.20000E 01
14	-0.28477E-01	-0.59728E-01	0.30000E 01
15	-0.26339E-01	-0.17461E-01	0.40000E 01
16	-0.15854E-01	-0.23175E-02	0.60000E 01

TOTAL ERROR IN COMPLEX FITTING = 0.264626E-03

PARAMETERS DERIVED ON ITERATION 5

NUMERATOR COEFFICIENTS STARTING WITH ZEROth IN ASCENDING ORDER

0.116012E 00 -0.939946E-02

DENOMINATOR COEFFICIENTS STARTING WITH THE ZEROth IN ASCENDING ORDER

0.100000E 01 0.755037E 00 0.212650E 00

CALCULATED RESULTS FROM LEVY POLYNOMIAL RATIO

	REAL	IMAGINARY	FREQUENCY
1	0.11601E 00	-0.96993E-04	0.10000E-02
2	0.11601E 00	-0.96991E-03	0.10000E-01
3	0.11599E 00	-0.19397E-02	0.20000E-01
4	0.11593E 00	-0.38787E-02	0.40000E-01
5	0.11584E 00	-0.58161E-02	0.60000E-01

6	0.11553E 00	-0.96832E-02	0.10000E 00
7	0.11407E 00	-0.19270E-01	0.20000E 00
8	0.11298E 00	-0.23996E-01	0.25000E 00
9	0.10829E 00	-0.37751E-01	0.40000E 00
10	0.02851E-01	-0.62127E-01	0.70000E 00
11	0.70794E-01	-0.79827E-01	0.10000E 01
12	0.28647E-01	-0.89242E-01	0.15000E 01
13	-0.43012E-02	-0.77300E-01	0.20000E 01
14	-0.28477E-01	-0.39728E-01	0.30000E 01
15	-0.26339E-01	-0.17461E-01	0.40000E 01
16	-0.15854E-01	-0.23175E-02	0.40000E 01

TOTAL ERROR IN COMPLEX FITTING = 0.264626E-03.

TRY USING DIFFERENT ORDERS

APPENDIX 1

A1.3 A listing of the continued fraction
program.


```

MASTER CHEN/SHIEH BY ROUTH ARRAY
DIMENSION ZP(150),ENUM(20),DENOM(20),H(30),A(400),D(20)
DIMENSION INT(20),ITS(20)
COMMON /STAR/ISTAB
COMMON C(1296),BU(36),IMP

```

```

*****
C
C THIS PROGRAM REDUCES AN NTH. ORDER TRANSFER FUNCTION TO ALL
C FEASIBLE LOWER ORDER MODELS BY THE METHOD OF TRUNCATED
C CONTINUED FRACTIONS.
C REFERENCE:

```

```

C CHEN C.F. & SHIEH L.S.,
C INT. J. CONTROL 1970 V.11 N.5 717-739

```

```

C THE MODEL MAY BE INPUT EITHER AS THE POLES AND ZEROES OR AS
C NUMERATOR AND DENOMINATOR POLYNOMIALS. THE MODEL PARAMETERS,
C ITS TIME RESPONSE(STEP OR IMPULSE), AND THE CHEN & SHIEH W
C COEFFICIENTS ARE OUTPUT FOR THE FULL MODEL. REDUCED ORDER MODEL
C PARAMETERS AND RESPONSE ARE OUTPUT. IF THE PREDICTED MODEL IS
C UNSTABLE THAT CYCLE IS TERMINATED AND A LOWER ORDER MODEL SOUGHT.

```

```

C DATA REQUIRED

```

```

C MODE=0 INPUT IS IN POLE/ZERO FORM AND THE FOLLOWING DATA IS NEEDED

```

```

C NPO=NUMBER OF POLES

```

```

C NZC= NUMBER OF COMPLEX POLES

```

```

C NZZ= NUMBER OF ZEROES

```

```

C NZC= NUMBER OF COMPLEX ZEROES

```

```

C CONST= TRANSFER FUNCTION GAIN

```

```

C ZD CONTAINS THE POLES AND ZEROES. THESE ARE INPUT SEPERATLY, ZEROES
C FIRST FOLLOWED BY POLES. COMPLEX ROOTS MUST BE THE FIRST IN EACH
C SERIES AND AS THEY ARE IN CONJUGATE PAIRS IN THE FORM -

```

```

C REAL IMAG REAL IMAG REAL REAL.....

```

```

C MODE=1 INPUT IS AS NUMERATOR/DENOMINATOR POLYNOMIALS AND THE
C FOLLOWING DATA IS NEEDED:

```

```

C NND= NUMBER OF DENOMINATOR COEFFICIENTS

```

```

C NNE= NUMBER OF NUMERATOR COEFFICIENTS

```

```

C THE NUMBER OF COEFFICIENTS IS ONE GREATER THAN THE ORDER

```

```

C A CONTAINS THE COEFFICIENTS, DENOMINATOR FOLLOWED BY NUMERATOR
C IN ASCENDING ORDER OF S (STARTING WITH THE CONSTANT)

```

```

C DATA NEEDED BY BOTH INPUTS

```

```

C THAX= MAXIMUM TIME FOR RESPONSE

```

```

C DT= TIME RESPONSE INTERVAL

```

```

C PROGRAM WRITTEN BY:

```

```

C M.J. ROSLEY,

```

```

C DEPT. CHEMICAL ENGINEERING,

```

```

C LOUGHBROUGH UNIVERSITY OF TECHNOLOGY,

```

```

C ENGLAND.

```

```

*****
C
C DETERMINE THE INPUT MODE

```

```

C READ(1,1)MODE

```

```

C IF(MODE.EQ.1)GO TO 2

```

```

C
C   THIS SECTION READS IN DATA IN POLE/ZERO FORM AND CONVERTS IT TO
C   NUMERATOR AND DENOMINATOR POLYNOMIALS USING SUBROUTINE FUDDLE
C
C   READ(1,1)NP,NZ,NPC,NZC
C
C   READ IN ZEROS
C
C   IF(NZ.EQ.0)GO TO 100
C   READ(1,3)(ZP(I),I=1,NZ)
100 CONTINUE
C   READ(1,3)CONST
C   NN=NZ+1
C   CALL FUDDLE(ENUM,NZC,NZ,ZP,H,NN)
C   H=NO
C   NN=NP+1
C   L=2+NN-1
C   DO 5 J=1,NZ+1
C   A(2+L+(J-1))=ENUM(J)+CONST
C
C   READ IN POLES
C
C   READ(1,3)(ZP(I),I=1,NP)
C   CALL FUDDLE(DENOM,NPC,NP,ZP,H,NN)
C   DO 4 I=1,NN
C   A(1+L+(J-1))=DENOM(J)
C   DO 6 I=NZ+2,NN
C   A(2+L+(J-1))=0
C   GO TO 7
C
C   THIS SECTION READS IN DATA IN POLYNOMIAL FORM
C
C   2 READ(1,1)NM,NV
C   V=NM-1
C   L=2+NM-1
C   READ(1,3)(A(1+L+(J-1)),J=1,NN)
C   READ(1,3)(A(2+L+(J-1)),J=1,NM)
C   DO 14 J=NM+1,NN
C   A(2+L+(J-1))=0
C
C   DATA IS NOW IN A FORM TO RUN THE METHOD. PARAMETERS ARE SET
C   FOR USE IN THE SUBROUTINES AND TIME DATA IS READ IN.
C
C   7 READ(1,3)DT,TMAX
C   DO 13 J=1,NN
C   A(1+L+(J-1))=A(1+L+(J-1))/A(1+L+(NM-1))
C   A(2+L+(J-1))=A(2+L+(J-1))/A(1+L+(NM-1))
C   DO 13 I=3,L
C   13 A(I+L+(J-1))=0
C   WRITE(2,8)N
C   IREF=0
C   IPRINT=1
C   NNR=NN
C   K=0
C   DO 15 I=1,2
C   DO 15 J=1,NN
C   K=K+1
C   15 A(K)=A(I+L+(J-1))
C   12 N1=1+NNR
C   N2=2+(NNR-1)
C
C   BACKPOUTH FORMS REDUCED MODEL PARAMETERS FROM THE H COEFFICIENT
C

```

```

CALL BACKROUTH(N1,N2,NNP,A,H,IPRINT)
C
C IF AN UNSTABLE SYSTEM IS PREDICTED THIS SECTION IS MISSED
C OTHERWISE THE TIME RESPONSE IS GIVEN
C
IF(ISTAR.EQ.0)GO TO 30
WRITE(2,31)
GO TO 10
30 CONTINUE
K2=NNP+NNR
K1=2*K2+7+NNR
N12=N+1
IF(IREP.EQ.1)GO TO 10
K=0
DO 16 I=1,2
DO 16 J=1,NN
C=K+1
16 A(I+L*(J-1))=H(K)
DO 17 I=2,L
DO 17 J=1,NN
17 A(I+L*(J-1))=0
C
C COMPUTE THE H COEFFICIENTS
C
CALL ROUTHAREAV(N1,N2,NN,A,H)
C
C REDUCE THE SYSTEM ORDER AND RETURN TO BACKROUTH TO
C COMPUTE THE PARAMETERS.
C
IPRINT=0
IREP=1
10 NNP=NNP-1
C=NNP-1
L=2+NNP-1
IF(NNP.LT.3)STOP
WRITE(2,11)N
GO TO 12
STOP
1 FORMAT(4I0)
2 FORMAT(20F0.0)
3 FORMAT(23H1FULL MODEL OF ORDER .13,////)
11 FORMAT(///26H REDUCED MODEL OF ORDER .13,////)
31 FORMAT(24H THIS SYSTEM IS UNSTABLE)
END

```

```

SUBROUTINE FUDDLE(C,KC,KT,ZP,A,M1)
DIMENSION R(3),ZP(M1),C(M1),A(M1)

```

THIS SEGMENT ORGANISES THE MULTIPLYING OUT OF A TRANSFER FUNCTION IN POLE/ZERO FORM TO A RATIO OF TWO POLYNOMIALS

```

IF(KT.EQ.0)C(1)=1
K = 1
IF(KC.EQ.0) GO TO 4
2 K = K+2
IF(K.NE.2) GO TO 3
A(1),C(1) = ZP(K-1)**2 + ZP(K)**2
A(2),C(2) = -2.0*ZP(K-1)
A(3),C(3) = 1.0

```

```

M,IPT = 3
4 IF(K.EQ.KC) GO TO 1
GO TO 2
7 A(1) = ZP(K-1)**2 + ZP(K)**2
Z(2) = -2.0+ZP(K-1)
B(K) = 1.0
N=3
IPT = M+N-1
CALL FIDDLY(A,B,C,M,N,IPT)
M = IPT
DO 5 I=1,M
5 A(I) = C(I)
GO TO 4
1 K=N+1
IF(K.GT.KT) RETURN
IF(K.EQ.1) GO TO 7
B(1) = -ZP(K)
B(2) = 1.0
N=2
IPT=M+N-1
CALL FIDDLY(A,B,C,M,N,IPT)
M = IPT
DO 8 I=1,M
8 A(I) = C(I)
GO TO 4
7 A(1),C(1) = -ZP(K)
A(2),C(2) = 1.0
M,IPT = 2
GO TO 4
END

```

```

SUBROUTINE FIDDLY(A,B,C,M,N,IPT)
DIMENSION A(M),B(N),C(IPT)

```

THIS SEGMENT MULTIPLIES TWO POLYNOMIALS A,B OF LENGTH M,N TO FORM A THIRD POLYNOMIAL C OF LENGTH IPT, WHERE IPT IS SET TO M+N-1 BY THE USER

```

DO 1 I=1,IPT
1 C(I) = 0.0
IC = 0
2 JC = JC+1
IO = IC+1
IF(JC.GT.IPT) RETURN
IA = 0
3 IA = IA+1
IF(IA.GT.M) GO TO 2
IB = 0
4 IB=IB+1
IF(IB.GT.N) GO TO 3
IF((IA+IB).NE.IP) GO TO 4
C(JC) = A(IA)*B(IB) + C(JC)
GO TO 3
END

```

```

SUBROUTINE BACKPOUTH(N1,N2,NNR,A,H,IPRINT)
DIMENSION A(N1),H(N2)
COMMON /STAR/ISTAR
C
C THIS SEGMENT GIVEN THE CHEN& SHIEH H COEFFICIENTS FORMS THE
C CORRESPONDING ROUTH ARRAY FROM WHICH THE TRANSFER FUNCTION
C COEFFICIENTS ARE FOUND
C
ISTAR=0
L=2+NNR-1
NNY=NNR-1
IF(IPRINT.EQ.1)GO TO 8
DO 1 I=1,N1
1 A(I)=0
A(L)=1
DO 2 K=1,L-1
I=L-K
2 A(I)=H(K)+A(I+1)
I=L-1
L1=1
L2=-1
DO 3 K=1,L-2
I=L-1
L2=L2
I=(L2.EQ.1)L1=L1+1
DO 3 J=2,L1
3 A(I+L+(J-1))=A(I+2+L+(J-2))+H(I)*A(I+1+L+(J-1))
GO TO 8
DO 4 I=1,2
DO 4 J=1,NNR
4 A(I+L+(J-1))=A(I+L+(J-1))/A(1+L*(NNR-1))
X CONTINUE
WRITE(2,6)(A(1+L+(J-1)),J=1,NNR)
WRITE(2,7)(A(2+L+(J-1)),J=1,NNR)
DO 13 J=1,NNR
13 IF(A(1+L+(J-1)).LT.0)ISTAR=1
RETURN
6. FORMAT(40H DENOMINATOR STARTING WITH ZEROth POWER,/(1X,4E16.6)
7. FORMAT(////37H NUMERATOR STARTING WITH ZEROth POWER,/(
1 (1X,4E16.6))
END

```

```

SUBROUTINE BUSHANDINT(N,A,K1,D,DT,TMAX,X,Y,C1,K2,INT,ITS,N2,N12)
DIMENSION A(K1),D(N),X(N12),Y(N),C1(K2),INT(N),ITS(N)
COMMON C(1296),BU(36),IMP
C
C THIS SEGMENT GIVEN THE ROUTH ARRAY TAKES THE TRANSFER FUNCTION
C (THE FIRST TWO LINES) AND PUTS INTO THE BUSH CANONICAL FORM. THE
C STATE VARIABLE MODEL IS THEN USED TO GIVE THE TIME RESPONSE
C USING SUBROUTINE MATINGR
C
DO 1 I=1,N+N
1 C(I)=0
DO 2 I=1,N-1
BU(I)=0
2 C(I+I+1)=1
BU(N)=1
DO 3 J=1,N
3 C(N+N+(J-1))=-A(1+N2*(J-1))

```

```

      DO 4 J=1,N
      A(1)=A(2+N2+(J-1))
C
C   THE STEP RESPONSE IS GIVEN IF IMP=0 AND THE IMPULSE IF IMP =1.
C   IF IRUNG = 0 THE ANALYTIC SOLUTION IS GIVEN, OR IF
C   IRUNG = 1 THE RUNGE KUTTA INTEGRATION.
C
      IMP=0
      IRUNG=0
      N1=N+1
      N2=N+N
      N3=N2+7*N
      CALL MATINGR(IRUNG,N,DT,TMAX,N1,N2,N3,X,Y,C1,C,INT,ITS,A,ITS,
1      A,D,C)
      RETURN
      END

```

```

      SUBROUTINE ROUTHAPRAY(N1,N2,NN,A,H)
      DIMENSION A(N1),H(N2)
C
C   THIS SEGMENT ,GIVEN THE TRANSFER FUNCTION NUMERATOR AND
C   DENOMINATOR, FORMS THE ROUTH ARRAY AND COMPUTES THE
C   OPEN & SHIF H COEFFICIENTS.
C
      L=2+NN-1
      L2=1
      L1=NN-1
      DO 2 I=3,L
      L2=-L2
      IF(L2.EQ.1) L1=L1-1
      DO 2 J=1,L1
      A(I+L+(J-1))=A(I-2+L+J)-A(I-2)/A(I-1)+A(I-1+L*J)
      WRITE(2,6)
      DO 3 I=1,N2
      A(I)=A(I)/A(I+1)
      WRITE(2,4)I,H(I)
      RETURN
      A FORMAT(////41H H PARAMETERS CALCULATED FROM ROUTH ARRAY,/,10X
1  24H NO. COEFFICIENT/)
      A FORMAT(12X,12,6X,E16.6)
      END

```

```

      SUBROUTINE MATINGR(IRUNG,N,DT,TMAX,N1,N2,N3,X,Y,M,MINV,INT,ITS
1AT,IC,REINT,Z,F,FT)
      REAL M(N2),MINV(N2)
      DIMENSION INT(N),ITS(N),X(N1),Y(N),AT(N3),IC(N),REINT(N1)
      DIMENSION Z(N1),F(N),FT(N)
      COMMON A(1296),BU(36),IMP
      IVS=0
      DO 600 I=1,N*N
      A(I)=A(I)
      IF(IRUNG.EQ.1)GO TO 700
      IF(N.IT. 3)GO TO 900
C
C   COMPUTE EIGENVALUES

```

```

C
CALL EODIRHESSE(N,A(1),INT(1))
CALL EODRHESSE(N,A(1),ITS(1),X(1),V(1),AT(1),IVS)
IEQ=0
C
C TEST FOR EQUAL ROOTS
C
DO 5 I=1,N-1
DO 5 J=I+1,N
IF (ABS(X(I)-X(J)).GT..1E-06)GO TO 5
IF (V(I).EQ.-V(J).AND.ABS(V(I)).GT..1E-09)GO TO 5
WRITE(2,7)
IEQ=1
GO TO 000
5 CONTINUE
C
C TEST FOR COMPLEX EIGENVALUES
C
DO 3 I=1,N
IF (ABS(V(I)).LT..1E-09)GO TO 4
IC(I)=1
GO TO 3
4 IC(I)=0
3 CONTINUE
C
C COMPUTE EIGENVECTORS
C
NROZ=N+N+7*N
CALL E4ORVS(N,NROZ,A,M,X,Y,AT)
CALL E4PACK(N,A,M,V,INT)
C
C COMPUTE INVERSE EIGENVECTORS
C
DO 9 I=1,N
DO 9 J=1,N
MINV(I+N*(J-1))=0.0
DO 10 I=1,N
MINV(I+N*(I-1))=1.0
IM=1
NA=N+N
DO 020 I=1,N+N
020 AT(N1+I)=M(I)
CALL E4SOLVE (M,MINV,N,NA,NA,IN,D,ID,IT,REINT)
DO 031 J=1,N+N
031 A(I)=AT(N1+J)
CALL HTM4(MINV,BU,Z,N,N,1)
000 CONTINUE
IF (IEQ.EQ.1)GO TO 700
IF (IMO)8,520,521
520 WRITE(2,522)
523 CONTINUE
DO 2003 I=1,N
2003 AT(N1+I)=F(I)
WRITE(2,2000)
T=-DT
11 T=T+DT
LI=1
IF (J4P)8,12,13
C
C THIS SECTION CALCULATES THE IMPULSE RESPONSE
C
13 DO 14 I=1,N
IF (LI)101,8,100

```

```

100 IF(1C(I).EQ.1)GO TO 15
   F(I)=7(I)*EXP(X(I)*T)
   GO TO 14
15 R=EXP(V(I)*T)
   S=V(I)*T
   F(I)=7(I)*R+COS(S)+Z(I+1)*R*SIN(S)
   F(I+1)=-Z(I)*R*SIN(S)+Z(I+1)*R+COS(S)
101 LI=-LI
14 CONTINUE
   CALL HTM4(M,F,FT,N,N,1)

C
C   WRITE TIME RESPONSE
C
   FET=0
   DO 2004 I=1,N
2004 FET=FET+FT(I)*AT(N1+I)
   WRITE(2,2002)T,FET
   IF(T-TMAX)11,11,8

C
C   THIS SECTION COMPUTES THE STEP RESPONSE
C
12 DO 20 I=1,N
   IF(LI)102,8,103
103 IF(1C(I).EQ.1)GO TO 21
   F(I)=7(I)*(EXP(X(I)*T)-1)/V(I)
   GO TO 20
21 R=EXP(V(I)*T)
   V1=V(I)
   S=V(I)*T
   P=V(I)+V(I)+X(I)*X(I)
   F(I)=7(I)*(( V1*SIN(S)+X(I)+COS(S))+R-X(I))/P+Z(I+1)*((X(I)+SIN(S)
1)- V1+COS(S))*R+V1)/P
   F(I+1)=-Z(I)*((X(I)+SIN(S)- V1+COS(S))*R+V1 )/P+Z(I+1)*(( V1+SIN
1(S)+X(I)+COS(S))+R-X(I))/P
102 LI=-LI
20 CONTINUE
   GO TO 14
   STOP
521 WRITE(2,524)
   GO TO 523

C
C   THIS SECTION DOES THE RUNGE KUTTA INTEGRATION
C
700 WRITE(2,702)
   DO 601 I=1,N*N
601 A(I)=M(I)
   IRUNG=1
   M1=N+1
   IF(IRUNG.EQ.1)GO TO 710
   WRITE(2,522)
   DO 704 I=2,M1
704 X(I)=0
   GO TO 715
710 WRITE(2,524)
   DO 711 I=2,M1
711 X(I)=RU(I-1)
715 H=DT
   X(1)=0
   IF(IRUNG.EQ.0)GO TO 8
   INIT=0
   WRITE(2,2000)
705 CALL F4RUNG(M1,INIT,H,X,Z,Y)
   INIT=1

```



```

      FET=0
      DO 2001 I=2,M1
2001  FET=FET+X(I)*F(I-1)
      WRITE(2,2002)X(1),FET
      IF(X(1).LE.TMAX)GO TO 705
      CONTINUE
      RETURN
      7  FORMAT(///18H EQUAL ROOTS FOUND)
      10  FORMAT(4X,13.5X,E10.4)
      201  FORMAT(3(3X,13.5X,E17.6))
      522  FORMAT(////11H STEP INPUT////)
      524  FORMAT(////17H IMPULSE RESPONSE////)
      702  FORMAT(24H1RUNGE-KUTTA INTEGRATION///)
2000  FORMAT(25H          TIME          FT/)
2002  FORMAT(2E16.6)
      END

```

```

SUBROUTINE F4DERV(M,Y,DY)
  DIMENSION Y(M),DY(M)

```

```

C THIS SEGMENT COMPUTES THE PATES FOR THE RUNGE KUTTA INTEGRATIO
C

```

```

      COMMON A(1206),BU(36),IMP
      N=M-1
      DO 1 I=1,N
      IF(IMP.EQ.1)GO TO 3
      DY(I+1)=BU(I)
      GO TO 4
      2 DY(I+1)=0
      4 DO 1 J=1,N
      SUM=A(I+4*(J-1))+Y(J+1)
      1 DY(I+1)=SUM+DY(I+1)
      RETURN
      END

```

APPENDIX 1

A1.4 Solution of the illustrative example
by Chen and Shieh's method.

A SAMPLE OUTPUT FROM THE CHEN & SHIEH METHOD

FULL MODEL OF ORDER 7

DENOMINATOR STARTING WITH ZEROth POWER

0.281250E 06	0.331087E 07	0.281427E 07	0.853703E 06
0.703420E 05	0.409700E 04	0.836400E 02	0.100000E 01

NUMERATOR STARTING WITH ZEROth POWER

0.312500E 05	0.375000E 06	0.000000E 00	0.000000E 00
0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00

N PARAMETERS CALCULATED FROM ROUTH ARRAY

NO.	COEFFICIENT
1	0.900000E 01
2	-0.487329E 00
3	-0.367168E-01
4	0.617229E 00
5	-0.253380E 02
6	-0.179453E 00
7	0.604446E 03
8	-0.289834E-01
9	-0.176205E 03
10	0.786918E-01
11	-0.524950E 05
12	-0.869436E-03
13	0.785369E 05
14	0.714678E-03

REDUCED MODEL OF ORDER 6

DENOMINATOR STARTING WITH ZEROth POWER

0.401081E 04	0.589714E 05	0.499510E 05	0.150325E 05
0.118904E 04	0.647340E 02	0.100000E 01	

NUMERATOR STARTING WITH ZEROth POWER

0.556757E 03	0.667931E 04	-0.213603E 02	-0.207160E 01
0.597757E-01	-0.714678E-03	0.000000E 00	

REDUCED MODEL OF ORDER 5

DENOMINATOR STARTING WITH ZEROth POWER

0.109787E 03	0.128969E 04	0.106649E 04	0.306276E 03
0.196511E 02	0.100000E 01		

NUMERATOR STARTING WITH ZEROth POWER

0.121936E 02	0.146080E 03	-0.363307E 01	0.358564E-01
0.156757E-03	0.000000E 00		

REDUCED MODEL OF ORDER 4

DENOMINATOR STARTING WITH ZEROth POWER

-0.701750E 01	-0.920323E 02	-0.654426E 02	-0.129594E 02
0.100000E 01			

NUMERATOR STARTING WITH ZEROth POWER

-0.879759E 00	-0.104264E 02	0.156148E 01	-0.785370E-01
0.000000E 00			

THIS SYSTEM IS UNSTABLE

REDUCED MODEL OF ORDER 3

DENOMINATOR STARTING WITH ZEROth POWER

0.451956E 00	0.528328E 01	0.408491E 01	0.100000E 01
--------------	--------------	--------------	--------------

NUMERATOR STARTING WITH ZEROth POWER

0.502175E-01 0.598481E 00 -0.495536E-01 0.000000E 00

REDUCED MODEL OF ORDER 2

DENOMINATOR STARTING WITH ZEROth POWER

0.003976E-01 0.114643E 01 0.100000E 01

NUMERATOR STARTING WITH ZEROth POWER

0.110442E-01 0.129899E 00 0.000000E 00

APPENDIX 1

A1.5 A listing of Lees' moments program .

```

MASTED MOMENTS METHOD FOR TRANSFER FUNCTION REDUCTION
DIMENSION X(20),G(40),H(40),A(400),D(20),E(20),W(20)
REAL MOM(20)

```

```

*****
THIS PROGRAM COMPUTES THE MOMENTS OF A GIVEN MATHEMATICAL
MODEL AND FITS TO THEM A TRANSFER FUNCTION OF THE USER'S CHOICE.
THE MODEL PARAMETERS ARE FITTED TO THE MOMENTS USING ROSENBRACK'S
HILLCLIMB.

```

```

DATA REQUIRED:

```

```

SUBROUTINE DETMOM4(NM,MOM)

```

```

THIS SUPPLIES THE NM MOMENTS OF THE MODEL, IN ADDITION TO THE
ZEROTH, AND STORES THEM IN THE ARRAYMOM. THIS ARRAY MUST NOT
BE DIMENSIONED GREATER THAN 20.

```

```

SUBROUTINE MODEL(M,X,G)

```

```

THIS SEGMENT, GIVEN THE CURRENT VALUE OF THE MODEL PARAMETERS
STORED IN ARRAY X COMPUTES THE MOMENTS OF THE FITTED TRANSFER
FUNCTION.

```

```

X=START VALUE FOR MODEL PARAMETERS.

```

```

G=LOWER CONSTRAINT ON MODEL PARAMETERS.

```

```

H=UPPER CONSTRAINT ON MODEL PARAMETERS.

```

```

PROGRAM WRITTEN BY

```

```

M.J. BOSLEY,
DEPT. CHEMICAL ENGINEERING,
LOUGHBOROUGH UNIVERSITY OF TECHNOLOGY

```

```

*****

CALL DETMOM4(NM,MOM)
I=NM
DO 4 I=1,M
  READ(1,2)X(I),G(I),H(I)
  NM1=NM+1
  NA=NM*(I+1)
  NG=2*N
  BJ=-1
  WRITE(2,20)
  CALL RMOVE(MOM(2),W(1),NM)
  CALL RMS6D(M,M,200,1,BJ,F,Y,G,H,W,NM1,NA,NG,A,D,E)
  WRITE(2,40)(X(I),I=1,M)
2X CALL MODEL (M,X,G)
24 WRITE(2,25)MOM(1),(G(I),I=1,NM)
  WRITE(2,30)F
  STOP
2 FORMAT(3G0.0)
20 FORMAT(1H1)
25 FORMAT(////21H SIMPLE MODEL MOMENTS,/,4E16.6)
30 FORMAT(////27H ERROR IN MATCHING MOMENTS=,F16.6)
40 FORMAT(////26H SIMPLE MODEL PARAMETERS =,4E16.6)
END

```

```

SUBROUTINE DETMON4(NM,MOM)
  DIMENSION A(400),B(100),AINV(400)
  REAL MOM(20)
  INTEGER OUTPUT

  THIS SEGMENT EVALUATES THE MOMENTS OF THE STATE VARIABLE MODEL:

      
$$X' = AX + BU$$


  AND STORES THEM IN ARRAY MOM.

  DATA REQUIRED:
  N=ORDER OF SYSTEM.
  IP= NUMBER OF INPUTS INTO SYSTEM.
  NM=NUMBER OF MOMENTS WHICH ARE TO BE FITTED IN ADDITION TO THE
  ZEROth. THIS MUST EQUAL THE NUMBER OF PARAMETERS IN THE
  MODEL WHICH IS BEING FITTED.
  A = PLANT MATRIX
  B = INPUT MATRIX.
  INPUT= SELECTED INPUT.
  OUTPUT = SELECTED OUTPUT.

  READ(1,1)N,IP,NM
  READ(1,2)((A(I+N*(J-1)),J=1,N),I=1,N)
  READ(1,2)((B(I+N*(J-1)),J=1,IP),I=1,N)
  READ(1,1)INPUT,OUTPUT
  N2=N+N
  NIP=N+IP
  WRITE(2,22)
  CALL WRITEMAT(N,N,N2,A)
  WRITE(2,23)
  CALL WRITEMAT(N,IP,NIP,B)
  DO 20 I=1,N2
20  AINV(I)=0
  DO 21 I=1,N
21  AINV(I+N*(I-1))=1
  CALL EASOLVE(A,AINV,N,N2,N2,1,D,IP,IT,MOM)
  IF(IT)3,4,3
  4  CALL EDUMNT(N,1,N,AINV(1),B(1+N*(INPUT-1)),A(1),0,NRR)
  CALL EDMASUS(A(1),DUM,A(1),N,-1,NRR)
  DO 6 I=2,NM+1
  Z=-I+1
  CALL EDUMNT(N,1,N,AINV(1),A(1+N*(I-2)),A(1+N*(I-1)),0,NRR)
  6  CALL EDMASUS(Z,DUM,A(1+N*(I-1)),N,0,NRR)
  DO 7 I=1,NM+1
  7  MOM(I)=A(OUTPUT+N*(I-1))
  WRITE(2,8)OUTPUT,INPUT,(MOM(I),I=1,NM+1)
  DO 10 I=1,NM+1
  10  B(I)=MOM(I)/MOM(1)
  DO 11 I=1,2
  11  MOM(I)=B(I)
  MOM(3)=B(3)-B(2)*B(2)
  MOM(4)=B(4)-3*B(3)+B(2)+2*B(2)**3
  WRITE(2,12)(MOM(I),I=1,NM+1)
  RETURN
  5  WRITE(2,9)
  STOP
  1  FORMAT(3I0)
  2  FORMAT(4G6D,0)
  3  FORMAT(////20H MOMENTS FOR OUTPUT ,I3,I3H WITH INPUT ,I3,//4E16,6)
  4  FORMAT(////22H PLANT MATRIX SINGULAR)

```



```

12 FORMAT(/19H NORMALISED MOMENTS,/,4E16.6)
22 FORMAT(////13H PLANT MATRIX)
23 FORMAT(////13H INPUT MATRIX)
END

```

```

SUBROUTINE CALXGH(N,M,IT,F,X,G,H,MOM)
REAL MOM(N)
DIMENSION X(N),G(N),H(N),CALCMOM(20)

```

THIS SEGMENT FORMULATES AN ERROR SQUARED OBJECTIVE FUNCTION.

```

C CALL MODEL (N,X,CALCMOM)
C F=0
C DO 4 I=1,N
C H(1)=X(I)+1
C F=F+(1 - CALCMOM(I)/MOM(I))**2
C RETURN
C END

```

```

SUBROUTINE WRITEMAT(N,M,NM,A)
DIMENSION A(NM)

```

THIS SEGMENT WRITES OUT AN N . M MATRIX.

```

C WRITE(2,1)
C DO 2 I=1,N
2 WRITE(2,3)1,(A(I+N*(J-1)),J=1,M)
C RETURN
C FORMAT(//)
C FORMAT(/6H ROW =,I3,4E16.6/(9X,4E16.6))
C END

```

```

SUBROUTINE MODEL (N,X,CALCMOM)
DIMENSION X(M),CALCMOM(4)
REAL M1,M2,M3

```

THIS SEGMENT EVALUATES THE FIRST 3 NORMALISED MOMENTS OF THE (1,2) POLYNOMIAL RATIO TRANSFER FUNCTION CORRESPONDING TO THE CURRENT MODEL PARAMETERS STORED IN ARRAY X. THE CALCULATED MOMENTS ARE STORED IN ARRAY CALCMOM.

```

C M1=X(1)
C M2=X(2)
C M3=X(3)
C M1=M1-M1
C M2=2*M2*M1-2*M2
C M3=3*M1*M2-6*M2*M1
C CALCMOM(1)=M1
C CALCMOM(2)=M2-M1*M1
C CALCMOM(3)=M3-3*M2*M1+2*M1**3
C RETURN

```

APPENDIX 1

A1.6 Solution of the illustrative example
by Lees' method.

A SAMPLE OUTPUT FROM THE MOMENTS METHOD

PLANT MATRIX

ROW = 1	0.000000E 00 0.000000E 00	0.100000E 01 0.000000E 00	0.000000E 00 0.000000E 00	0.000000E 00
ROW = 2	0.000000E 00 0.000000E 00	0.000000E 00 0.000000E 00	0.100000E 01 0.000000E 00	0.000000E 00
ROW = 3	0.000000E 00 0.000000E 00	0.000000E 00 0.000000E 00	0.000000E 00 0.000000E 00	0.100000E 01
ROW = 4	0.000000E 00 0.100000E 01	0.000000E 00 0.000000E 00	0.000000E 00 0.000000E 00	0.000000E 00
ROW = 5	0.000000E 00 0.000000E 00	0.000000E 00 0.100000E 01	0.000000E 00 0.000000E 00	0.000000E 00
ROW = 6	0.000000E 00 0.000000E 00	0.000000E 00 0.000000E 00	0.000000E 00 0.100000E 01	0.000000E 00
ROW = 7	-0.281250E 04 -0.703420E 05	-0.331087E 07 -0.409700E 04	-0.281427E 07 -0.836400E 02	-0.853703E 04

INPUT MATRIX

ROW = 1	0.000000E 00
ROW = 2	0.000000E 00
ROW = 3	0.000000E 00
ROW = 4	0.000000E 00
ROW = 5	0.000000E 00
ROW = 6	0.375000E 04
ROW = 7	-0.313337E 08

MOMENTS FOR OUTPUT 1 WITH INPUT 1
 0.111111E 00 -0.253333E 01 -0.282007E 01 -0.960490E 02

NORMALISED MOMENTS
 0.100000E 01 -0.228000E 00 -0.254326E 02 -0.881825E 03

MPLE MODEL PARAMETERS = 0.117647E 02 0.115387E 02 0.103341E 02

MPLE MODEL MOMENTS

0.100000E 01 -0.228000E 00 -0.254326E 02 -0.881825E 03

ROR IN MATCHING MOMENTS= 0.235283E-13

APPENDIX 2

A2.1 A listing of the program for the
reduction of state variable models
by matching the moments.

```

MASTER MOMENTS REDUCTION LEAST SQUARES2
REAL MOM(400),MO(200)
DIMENSION AINV(1681),X(41),Y(41),Q(1681),S(800),R(2000),INT(41)
DIMENSION KX(41),FT(41),B(400)
DIMENSION TEXT(10)
EQUIVALENCE (MOM(1),X(1)),(MOM(42),Y(1)),(MOM(83),FT(1))
COMMON A(1681),BU(41),IMP

```

```

C *****
C
C THIS PROGRAM REDUCES THE NTH ORDER STATE VARIABLE MODEL WITH
C IP INPUTS

```

$$X' = AX + BU$$

```

C TO THE MTH ORDER STATE VARIABLE MODEL WITH IP INPUTS

```

$$X'^* = A^*X^* + B^*U$$

```

C BY USING THE METHOD OF MOMENTS WITH EQUATIONS SOLVED
C BY THE LINEAR LEAST SQUARES METHOD AND THE B* MATRIX BUILT TO
C CORRECT STEADY STATE VALUES

```

DATA REQUIRED

```

C N= ORDER OF FULL MODEL

```

```

C IP= NUMBER OF INPUTS INTO THE MODEL

```

```

C M= ORDER TO WHICH THE MODEL IS TO BE REDUCED

```

```

C NM= NUMBER OF MOMENTS TO BE MATCHED IN ADDITION TO THE ZEROth

```

```

C IMP = PARAMETER FOR SYSTEM TIME RESPONSE. SET TO 0 FOR
C STEP AND TO 1 FOR IMPULSE

```

```

C A= PLANT MATRIX

```

```

C B= FORCING MATRIX

```

```

C KX IS AN MTH ORDER ARRAY CONTAINING THE ELEMENTS OF X WHICH ARE
C TO BE RETAINED IN THE REDUCED MODEL

```

```

C TMAX= MAX. TIME OVER WHICH MODEL IS TO BE INTEGRATED

```

```

C DT=INTEGRATION INTERVAL

```

```

C N.B. THE PRODUCT (NM+1)*IP MUST NOT BE LESS THAN THE REDUCED ORDER

```

RESULTS OUTPUT

```

C THE COMPLETE MATRIX ,THE EIGENVALUES AND EIGENVECTORS,

```

```

C THE TIME RESPONSE AND MOMENTS FOR EACH INPUT ;ALL IS

```

```

C WRITTEN OUT FOR BOTH THE FULL AND REDUCED MODELS.

```

```

C THE LEAST SQUARES ERRORS ARE WRITTEN AND ALSO

```

```

C PROGRAM AND MATHEMATICAL ERRORS.

```

```

C PROGRAM WRITTEN BY

```

```

M.J. BOSLEY

```

```

DEPT. CHEMICAL ENGINEERING,

```

```

LOUGHBOROUGH UNIVERSITY OF TECHNOLOGY,
ENGLAND.

```

```

C READ IN SYSTEM ORDERS AND FULL PLANT MATRIX

```

```

C READ(1,1)M,N,IP,NM,IMP

```

```

C READ(1,2)((A(I+N*(J-1)),J=1,N),I=1,N)

```

```

C READ(1,2)DT,TMAX

```

```

C COMPUTE PARAMETERS FOR USE IN PROGRAM

```

```

C IV=0

```

```

IPFAD=3
IRUNG=0
EDS=.1E-10
NIP=N+IP
NL=NM+1
NMP=IP*N4
NO=M*NMP
NP=IP*NMP
MIP=N+IP
M2=M*M
M22=2*M2
IF=1
WRITE(2,110)

```

```

USE MATINGR FOR THE FULL SYSTEM

```

```

28 N1=N+1
N2=N*N
N3=N2+7*N
N5=N*N4
DO 4 I=1,N2
4 AINV(I)=A(I)

```

```

MATINGR WRITES OUT THE PLANT MATRIX, COMPUTES THE EIGENVALUES
AND EIGENVECTORS AND THEN COMPUTES THE TIME RESPONSE OF THE
SYSTEM ANALYTICALLY (IRUNG=0) OR BY RUNGE KUTTE (IRUNG=1)

```

```

5 CALL MATINGR(IRUNG,N,DT,TMAX,N1,N2,N3,X,Y,Q,A,INT,KX,R,KX,R,R,BU,
1 FT,IX,IREAD)
GO TO(6,10,8,10),IE

```

```

READ IN AND WRITE OUT B MATRIX

```

```

6 READ(1,2)((R(I+N*(J-1)),J=1,IP),I=1,N)
WRITE(2,112)
CALL WRITEMAT(N,IP,NIP,B)
IF=2
8 IF(IE.EQ.3) IE=4

```

```

COMPUTE THE TIME RESPONSE FOR EACH FORCING FUNCTION

```

```

DO 10 K=1,IP
WRITE(2,115)K
DO 11 I=1,N
11 BU(I)=R(I+N*(K-1))
GO TO 5
10 CONTINUE
DO 12 I=1,N2
12 A(I)=AINV(I)
IF(IE.EQ.4) GO TO 9

```

```

READ IN AND WRITE OUT VARIABLES OF INTEREST

```

```

READ(1,1)(KX(I),I=1,M)
WRITE(2,1000)(KX(I),I=1,M)

```

```

PUT ARRAYS TO ZERO FOR USE IN THE PROGRAM

```

```

DO 13 I=1,NO
Q(I)=0
13 S(I)=0
DO 14 I=1,NR
14 R(I)=0

```

```

      M7=1
30  IN=1

C
C   COMPUTE THE INVERSE PLANT MATRIX FOR USE IN
C   DETERMINING THE MOMENTS
C
      DO 15 I=1,N2
15  AINV(I)=0
      DO 16 I=1,N
16  AINV(I+N*(I-1))=1
      CALL F4SOLVE(A,AINV,N,N2,N2,IN,D,ID,IT,BU)
      IF(IT)17,18,18

C
C   DETERMINE THE MOMENTS FOR EACH FORCING FUNCTION IN TURN
C   USING SUBROUTINE DETMOM
C
18  DO 19 NF=1,IP
      DO 20 I=1,N
20  BU(I)=B(I+N*(NF-1))
      CALL DETMOM2(MOM,N,NM,AINV,NF,N5,N2)
      IF(MZ.EQ.3)GO TO 19

C
C   FOR THE FULL SYSTEM THIS SEGMENT SETS UP THE ARRAYS Q AND S AND
C   STORES THE ZERO TH MOMENTS ,OR STEADY STATES, IN M0
C
      DO 21 J1=1,N4
      J=NF+IP*(J1-1)
      DO 21 I=1,M
      I2=KX(I)
21  O(I+M*(J-1))=MOM(I2+N*(J1-1))
      DO 22 J1=1,NM
      J=NF+IP+J1
      DO 22 I=1,M
      I2=KX(I)
22  S(I+M*(J-1))=-J1*MOM(I2+N*(J1-1))
      DO 24 I=1,M
      I2=KX(I)
24  M0(I+M*(NF-1))=MOM(I2)
19  CONTINUE
      IF(MZ.EQ.3)GO TO 550

C
C   FOR THE FULL SYSTEM SET UP THE ARRAY R
C
      DO 23 I=1,IP
23  R(I+IP*(I-1))=1

C
C   PREPARE ARRAYS FOR THE LEAST SQUARES SOLUTION TAKING
C   ACCOUNT OF THE STEADY STATE CONDITIONS HELD IN M0
C
      CALL FPMUNT(M,NMP,IP,M0(1),R(1),A(1),0,NRR)
      CALL FPHASUS(A(1),Q(1),A(1),NQ,2,NRR)
      CALL TRANMAT(A,Q,M,NMP)
      CALL TRANMAT(S,A,M,NMP)
      CALL FMOVE(Q(1),AINV(1),NQ)
      CALL FMOVE(A(1),R(801),NQ)

C
C   SOLVE THE LEAST SQUARES PROBLEM ONE COLUMN AT A TIME,
C   EACH ROW BEING NORMALISED. THE EUCLID NORMS ARE COMPUTED BEFORE
C   AND AFTER SOLVING. THE COLUMN BY COLUMN SOLUTION IS STORED IN S
C
      DO 500 J=1,M
      DO 501 I=1,NMP
      RO=R(800+I+NMP*(J-1))

```



```

      IF(RQ.EQ.0)GO TO 505
      A(I)=1
      DO 502 K=1,M
      K2=I+NMP*(K-1)
502  Q(K2)=AINV(K2)/RQ
      GO TO 501
505  A(I)=0
      DO 506 K=1,M
      K2=I+NMP*(K-1)
506  Q(K2)=AINV(K2)
501  CONTINUE
      EUC=0
      DO 302 I=1,NMP
302  EUC=EUC+A(I)*A(I)
      EUC=SQRT(EUC)
301  CONTINUE
      M7=2
      CALL ILSQ(NMP,M,1,NQ,NMP,M,M22,Q,A,BU,INT,EPS,IT,R)
      IF(IT)25,200,17
200  CONTINUE
      NE=1+M*(J-1)
500  CALL FMOVE(BU(1),S(NE),M)

```

C THE REDUCED PLANT MATRIX A* IS FORMED FROM S AND THE B* MATRIX
 C FROM A* BY MULTIPLYING THE ZEROth MOMENTS.B* IS WRITTEN OUT
 C

```

      CALL TRANMAT(S,A,M,M)
      WRITE(2,26)
      CALL FPMUNT(M,IP,M,A(1),M0(1),B(1),0,NRR)
      CALL FPMASUS(B(1),DUM,B(1),MIP,-1,NRR)
      WRITE(2,27)
      CALL WRITEMAT(M,IP,MIP,B)

```

C SET UP PARAMETERS TO RETURN TO MATINGR FOR THE REDUCED SYSTEM
 C

```

      N=M
      IX=0
      IC=3
      IPFAD=4
      GO TO 28
9  M7=3
      WRITE(2,66)
      GO TO 30
550  CONTINUE
9999  CONTINUE
      STOP
17  WRITE(2,32)M2
      IF(M2.NE.2)STOP
      WRITE(2,2020)IT
      WRITE(2,2021)(INT(I),I=IT+1,N)
      STOP
25  WRITE(2,33)
      STOP
1  FORMAT(20I0)
2  FORMAT(40F0.0)
110  FORMAT(16H FULL MODEL DATA//// )
31  FORMAT(//22H LEAST SQUARES ERROR =,E16.6)
26  FORMAT(15H REDUCED SYSTEM//)
27  FORMAT(////27H FORCING FUNCTION MATRIX B*//)
66  FORMAT(////22H REDUCED MODEL MOMENTS//)
32  FORMAT(////7H MATRIX,13,12H IS SINGULAR)
33  FORMAT(////17H DIMENSIONS WRONG)
112  FORMAT(////26H FORCING FUNCTION MATRIX B//)

```

```

115 FORMAT(///3RH TIME RESPONSE FOR FORCING FUNCTION      ,13//)
300 FORMAT(////18H RHS EUCLID. NORMS)
303 FORMAT(////19H RHS EUCLID NORM =,F16.6)
531 FORMAT(1H1)
1000 FORMAT(////23H VARIABLES OF INTEREST=,30I3)
466 FORMAT(A8)
2020 FORMAT(14H MATRIX RANK =,I4)
2021 FORMAT(18H USELESS COLUMNS =,20I3)
END

```

```

SUBROUTINE LLSQ(M,N,L,MN,ML,NL,N2L,A,B,X,IPIV,EPS,IER,AUX)
DIMENSION A(MN),B(ML),X(NL),IPIV(N),AUX(N2L)

```

```

C THIS SEGMENT SOLVES THE LINEAR LEAST SQUARES PROBLEM, I.E.
C MINIMIZES THE EUCLID NORM OF B-AX, WHERE A IS (M,N) MATRIX
C M GE. N, AND B IS (N,L) MATRIX. THIS PROGRAM IS GIVEN IN DETAIL
C IN THE IBM 360 SCIENTIFIC SUBROUTINES MANUAL. THE METHOD HAS
C BEEN GIVEN BY
C G.GOLUB, NUMERISCHE MATHEMATIK, VOL7, ISS.3(1965), 206-216
C

```

```

C IF(M-N)30,1,1
1 PIV=0
  IEND=0
  DO 4 K=1,N
    IPIV(K)=K
    H=0
    IST=IEND+1
    IEND=IEND+M
    DO 2 I=IST,IEND
2 H=H+A(I)*A(I)
    AUX(K)=H
    IC(H-PIV)4,4,3
3 PIV=H
  KPIV=K
4 CONTINUE
  IC(PIV)31,31,5
5 SIG=SQRT(PIV)
  TOL=SIG*ABS(EPS)
  LM=1*M
  IST=-M
  DO 21 K=1,N
    IST=IST+M+1
    IEND=IST+M-K
    IC(KPIV-K
    IC(I)8,8,6
6 H=AUX(K)
  AUX(K)=AUX(KPIV)
  AUX(KPIV)=H
  ID=1+M
  DO 7 I=IST,IEND
    J=I+ID
    H=A(I)
    A(I)=A(J)
7 A(J)=H
8 IF(K-1)11,11,9
9 SIG=0
  DO 10 I=IST,IEND
10 SIG=SIG+A(I)*A(I)
  SIG=SQRT(SIG)

```

```

      IF(SIG-TOL)32,32,11
11  H=A(IST)
      IF(H)12,13,13
12  SIG=-SIG
13  IPIV(KPIV)=IPIV(K)
      IPIV(K)=KPIV
      BETA=H+SIG
      A(IST)=BETA
      BETA=1/(SIG*BETA)
      J=N+K
      AUX(J)=-SIG
      IF(K-N)14,19,19
14  PIV=0
      ID=0
      JST=K+1
      KPIV=JST
      DO 18 J=JST,N
      ID=ID+M
      H=0
      DO 15 I=IST,IEND
      IT=I+ID
15  H=H+A(I)*A(IT)
      H=BETA*H
      DO 16 I=IST,IEND
      IT=I+ID
16  A(IT)=A(IT)-A(I)*H
      IT=IST+ID
      H=AUX(J)-A(IT)*A(IT)
      AUX(J)=H
      IF(H-PIV)18,18,17
17  PIV=H
      KPIV=J
18  CONTINUE
19  DO 21 J=K,LM,M
      H=0
      IFND=J+M-K
      IT=IST
      DO 20 I=J,IFND
      H=H+A(IT)*B(I)
20  IT=IT+1
      H=BETA*H
      IT=IST
      DO 21 I=J,IFND
      B(I)=B(I)-A(IT)*H
21  IT=IT+1
      IFR=0
      I=N
      LN=L+N
      PIV=1/AUX(2*N)
      DO 22 K=N,LN,N
      X(K)=PIV*B(I)
22  I=J+M
      IF(N-1)26,26,23
23  JST=(N-1)+M+N
      DO 25 J=2,N
      JST=JST-M-1
      K=N+N+1-J
      PIV=1/AUX(K)
      KST=K-N
      ID=IPIV(KST)-KST
      IKT=2-J
      DO 25 K=1,L
      H=R(KST)

```

```

    IST=IST+N
    IEND=IST+J-2
    II=JST
    DO 24 I=IST,IEND
    II=II+M
24  H=H-A(II)*X(I)
    I=IST-1
    II=I+10
    X(I)=X(II)
    X(II)=PIV*H
25  KST=KST+M
26  IST=N+1
    IEND=0
    DO 29 J=1,L
    IEND=IEND+M
    H=0
    IC(M-N)29,20,27
27  DO 28 I=IST,IEND
28  H=H+B(I)*B(I)
    IST=IST+M
29  AIX(J)=H
    RETURN
30  IER=-2
    RETURN
31  IER=-1
    RETURN
32  IER=K-1
    RETURN
    END

```

```

SUBROUTINE NETMOM2(Q,N,NM,AINV,IP,N1,N2)
  DIMENSION Q(N1),AINV(N2)
  COMMON A(1681),BU(41),IMP

```

THIS SEGMENT COMPUTES THE MOMENTS (ZEROth UP TO THE NM MOMENT)
OF THE SYSTEM

$$X' = AX + B$$

```

DO 1 I=1,N
Q(I)=0
DO 1 J=1,N
1  Q(I)=-AINV(I+N*(J-1))*BU(J)+Q(I)
    J=1
DO 2 K=2,NM+1
DO 2 I=1,N
Q(I+N*(K-1))=0
DO 2 J=1,N
2  Q(I+N*(K-1))=-(K-1)*AINV(I+N*(J-1))*Q(J+N*(K-2))+Q(I+N*(K-1))
    WRITE(2,3)IP
DO 4 J=1,N
4  WRITE(2,5)I,(Q(I+N*(J-1)),J=1,NM+1)
    RETURN
3  FORMAT(///30H MATRIX OF MOMENTS FOR FORCING FUNCTION,I3//)
5  FORMAT(/6H ROW =,I3,4E16.6/(9X,4E16.6))
11 FORMAT(////19H NORMALISED MOMENTS)
    END

```

```

SUBROUTINE MATINGR(IRUNG,N,DT,TMAX,N1,N2,N3,X,Y,M,MINV,INT,ITS,
1      AT,IC,REINT,Z,F,FT,IX,IREAD)
REAL M(N2),MINV(N2)
DIMENSION INT(N),ITS(N),X(N1),Y(N1),AT(N3),IC(N),REINT(N1)
DIMENSION Z(N1),F(N),FT(N)
COMMON A(1681),BU(41),IMP

```

THIS SEGMENT COMPUTES THE TIME RESPONSE OF THE SYSTEM

$X' = AX + B$
 BY THE ANALYTICAL METHOD OR (IF IRUNG=1 OR EQUAL ROOTS ARE
 FOUND) BY THE RUNGE KUTTA METHOD

```

ICOU=0
IF(IX.NE.0)GO TO 2000
WRITE(2,502)
DO 500 I=1,N
500 WRITE(2,501)I,(A(I+N*(J-1)),J=1,N)
IVS=0
DO 600 I=1,N*N
600 M(I)=A(I)
IF(N.GE.3)GO TO 2001
IX=2
RETURN

```

COMPUTE EIGENVALURS

```

2001 CONTINUE
CALL EPPDIRHESSE(N,A(1),INT(1))
CALL EPPQRHESSE(N,A(1),ITS(1),X(1),Y(1),AT(1),IVS)
WRITE(2,504)
WRITE(2,503)
DO 203 I=1,N
203 WRITE(2,202)I,X(I),Y(I)
IFQ=0

```

TEST FOR EQUAL ROOTS

```

DO 5 I=1,N-1
DO 5 J=I+1,N
IF(ABS(X(I)-X(J)).GT..1E-06)GO TO 5
IF(Y(I).EQ.-Y(J).AND.ABS(Y(I)).GT..1E-09)GO TO 5
WRITE(2,7)
IFQ=1
IX=2
RETURN
5 CONTINUE

```

TEST FOR COMPLEX EIGENVALUES

```

IF(IRUNG.EQ.1)GO TO 4000
DO 3 I=1,N
IF (ABS(Y(I)).LT..1E-09)GO TO 4
IF(I)=1
GO TO 3
4 IC(I)=0
3 CONTINUE

```

COMPUTE EIGENVECTORS

```

NBOZ=N+N+7*N
CALL E4QRV5(N,NBOZ,A,M,X,Y,AT)

```

```

      CALL F4BACK(N,A,M,Y,INT)
      WRITE(2,505)
      DO 506 I=1,N
      WRITE(2,501)I,(M(I+N*(J-1)),J=1,N)
506 CONTINUE

      COMPUTE INVERSE EIGENVECTORS

      DO 9 I=1,N
      DO 9 J=1,N
      9 MINV(I+N*(J-1))=0.0
      DO 10 I=1,N
      10 MINV(I+N*(J-1))=1.0
      IU=1
      NA=N*N
      DO 980 I=1,N*N
      980 AT(NI+1)=M(I)
      CALL F4SOLVE (M,MINV,N,NA,NA,IN,D,IO,IT,REINT)
      WRITE(2,1000)
      DO 1001 I=1,N
      1001 WRITE(2,501)I,(MINV(I+N*(J-1)),J=1,N)
      DO 981 I=1,N*N
      981 M(I)=AT(NI+1)
      IX=1
4000 CONTINUE
      IF (IRUNG.EQ.1) IX=2
      RETURN
2000 IF (IX.EQ.2) GO TO 700
      CALL UTM4(MINV,BU,2,N,N,1)
      900 CONTINUE
      IF (IEQ.EQ.1) GO TO 700
      IF (IMP) 8,520,521
      520 WRITE(2,522)
      523 CONTINUE
      T=-DT
      11 T=T+DT
      LI=1
      IF (IMP) 8,12,13

      THIS SECTION CALCULATES THE IMPULSE RESPONSE

      13 DO 14 I=1,N
      IF (LI) 101,8,100
      100 IF (IC(I).EQ.1) GO TO 15
      F(I)=Z(I)*EXP(X(I)*T)
      GO TO 14
      15 R=EXP(X(I)*T)
      S=Y(I)*T
      F(I)=Z(I)*R+COS(S)+Z(I+1)*R*SIN(S)
      F(I+1)=-Z(I)*R*SIN(S)+Z(I+1)*R*COS(S)
      101 LI=-LI
      14 CONTINUE
      CALL UTM4(M,F,FT,N,N,1)

      WRITE TIME RESPONSE

      ICOU=ICOU+1
      WRITE(2,17)T,ICOU
      WRITE(2,300)
      WRITE(IREAD)T,(FT(I),I=1,N)
      WRITE(2,301)((I,FT(I)),I=1,N)
      NTIME=NINT((T-TMAX)/DT)
      IF (NTIME) 11,8,8

```

THIS SECTION COMPUTES THE STEP RESPONSE

```

12  DO 20 I=1,N
    IF(LL)102,8,103
103  IF(IC(I).EQ.1)GO TO 21
    F(I)=7(I)*(EXP(X(I)*T)-1)/X(I)
    GO TO 20
21  R=EXP(X(I)*T)
    Y1=Y(I)
    S=Y(I)*T
    P=Y(I)*Y(I)+X(I)*X(I)
    F(I)=7(I)*(( Y1*SIN(S)+X(I)*COS(S))*R-X(I))/P+Z(I+1)*((X(I)*SIN(S)
1) - Y1*COS(S))*R+Y1)/P
    F(I+1)=-Z(I)*((X(I)*SIN(S)- Y1*COS(S))*R+Y1 )/P+Z(I+1)*(( Y1*SIN
1(S)+X(I)*COS(S))*R-X(I))/P
102  LI=-LI
20  CONTINUE
    GO TO 14
    STOP
521  WRITE(2,524)
    GO TO 523

```

THIS SECTION DOES THE RUNGE KUTTA INTEGRATION

```

700  WRITE(2,702)
    DO 601 I=1,N*N
601  A(I)=M(I)
    I=UNG=1
    M1=N+1
    IF(IMP.EQ.1)GO TO 710
    WRITE(2,522)
    DO 704 I=2,M1
704  X(I)=0
    GO TO 711
710  WRITE(2,524)
    DO 711 I=2,M1
711  X(I)=RU(I-1)
715  H=DT
    X(1)=0
    IF(IRUNG.EQ.0)GO TO 8
    INIT=0
705  CALL F4RUNG(M1,INIT,H,X,Z,Y)
    INIT=1
    ICOU=ICOU+1
    WRITE(2,17)X(1),ICOU
    WRITE(2,300)
    WRITE(IREAD)(X(I),I=1,N+1)
    WRITE(2,301)((I,X(I+1)),I=1,N)
    NTIME=NINT((X(1)-TMAX)/DT)
    IF(NTIME)705,8,8
8  CONTINUE
    RETURN
7  FORMAT(///18H EQUAL ROOTS FOUND)
505  FORMAT(///23H MATRIX OF EIGENVECTORS/)
504  FORMAT(///19H SYSTEM EIGENVALUES/)
503  FORMAT(35H VARIABLE      REAL      IMAG)
202  FORMAT(3X,I3,2(4X,E13.6))
502  FORMAT(///9H A MATRIX/)
501  FORMAT(/6H ROW =,I3,4E16.6/(9X,4E16.6))
17  FORMAT(///7H TIME =,E10.4,I4,/)
19  FORMAT(4X,I3,5X,E10.4)
300  FORMAT(64H VARIABLE      FT      VARIABLE      FT      VARIABLE

```

```

1      FT)
301  FORMAT(3(3X,13,E17.6))
522  FORMAT(////11H STEP INPUT////)
524  FORMAT(////17H IMPULSE RESPONSE////)
1000  FORMAT(///31H MATRIX OF INVERSE EIGENVECTORS/)
702  FORMAT(24H1RUNGE-KUTTA INTEGRATION//)
      END

```

```

SUBROUTINE F4DERV(M,Y,DY)
  DIMENSION Y(M),DY(M)
  COMMON A(1681),BU(41),IMP

```

THIS SEGMENT COMPUTES THE RATES X' OF
 $X' = AX + BU$
 FOR A RUNGE KUTTA INTEGRATION

```

      N=M-1
      DO 1 I=1,N
        IF(IMP.EQ.1)GO TO 3
        DY(I+1)=BU(I)
        GO TO 4
      3  DY(I+1)=0
      4  DO 1 J=1,N
        SUM=A(I+N*(J-1))*Y(J+1)
      1  DY(I+1)=SUM+DY(I+1)
      RETURN
      END

```

```

SUBROUTINE TRANMAT(A,AT,M,N)
  DIMENSION A(M,N),AT(N,M)

```

THIS SEGMENT PUTS THE TRANSPOSE OF A(M,N) INTO AT

```

      DO 1 I=1,M
      DO 1 J=1,N
      1  AT(J,I)=A(I,J)
      RETURN
      END

```

```

SUBROUTINE WRITEMAT(N,M,NM,A)
  DIMENSION A(NM)

```

THIS SEGMENT WRITES THE MATRIX A(N,M)

```

      WRITE(2,1)
      DO 2 I=1,N
      2  WRITE(2,3)I,(A(I+N*(J-1)),J=1,M)
      RETURN
      1  FORMAT(//)
      3  FORMAT(/6H ROW =,I3,4E16.6/(9X,4E16.6))
      END

```


APPENDIX 2

A2.2 Model 1, an overdamped system,
reduced by matching the moments.

Contents

- 1. Full and reduced model data.
- 2. Step responses as detailed below.

Figure	Input	state x_i	reduced state x_i^*
A2.1	1	1	1
A2.2	1	3	2
A2.3	1	7	3
A2.4	1	12	4
A2.5	2	3	2
A2.6	2	7	3
A2.7	2	12	4

State x_1 was not forced by input 2.

AN OVERDAMPED SYSTEM OF STIRRED TANKS

FULL MODEL DATA

A MATRIX

ROW = 1	-0.100000E 01	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 2	0.200000E 01	-0.200000E 01	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 3	0.000000E 00	0.100000E 02	-0.100000E 02	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 4	0.000000E 00	0.000000E 00	0.130000E 02	-0.130000E 02
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 5	0.000000E 00	0.000000E 00	0.000000E 00	0.300000E 01
	-0.300000E 01	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 6	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.800000E 01	-0.800000E 01	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 7	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.500000E 01	-0.500000E 01	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 8	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.110000E 02	-0.110000E 02
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 9	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.600000E 01
	-0.600000E 01	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 10	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.120000E 02	-0.120000E 02	0.000000E 00	0.000000E 00
ROW = 11	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.900000E 01	-0.900000E 01	0.000000E 00
ROW = 12	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.140000E 02	-0.140000E 02

SYSTEM EIGENVALUES

VARIABLE	REAL	IMAG
1	-0.100000E 01	0.000000E 00
2	-0.200000E 01	0.000000E 00
3	-0.300000E 01	0.000000E 00
4	-0.500000E 01	0.000000E 00
5	-0.600000E 01	0.000000E 00
6	-0.130000E 02	0.000000E 00
7	-0.800000E 01	0.000000E 00
8	-0.120000E 02	0.000000E 00
9	-0.100000E 02	0.000000E 00
10	-0.110000E 02	0.000000E 00
11	-0.900000E 01	0.000000E 00
12	-0.140000E 02	0.000000E 00

FORCING FUNCTION MATRIX B

ROW = 1	0.100000E 01	0.000000E 00
ROW = 2	0.000000E 00	0.200000E 01
ROW = 3	0.000000E 00	0.000000E 00
ROW = 4	0.000000E 00	0.000000E 00
ROW = 5	0.000000E 00	0.000000E 00
ROW = 6	0.000000E 00	0.000000E 00
ROW = 7	0.000000E 00	0.000000E 00
ROW = 8	0.000000E 00	0.000000E 00
ROW = 9	0.000000E 00	0.000000E 00
ROW = 10	0.000000E 00	0.000000E 00
ROW = 11	0.000000E 00	0.000000E 00
ROW = 12	0.000000E 00	0.000000E 00

VARIABLES OF INTEREST= 1 3 7 12

MATRIX OF MOMENTS FOR FORCING FUNCTION 1

ROW = 1	0.100000E 01	0.100000E 01	0.200000E 01	0.600000E 01
	0.240000E 02			

ROW = 2	0.100000E 01 0.465000E 02	0.150000E 01	0.350000E 01	0.112500E 02
ROW = 3	0.100000E 01 0.514584E 02	0.160000E 01	0.382000E 01	0.123960E 02
ROW = 4	0.100000E 01 0.555621E 02	0.167692E 01	0.407799E 01	0.133371E 02
ROW = 5	0.100000E 01 0.805691E 02	0.201026E 01	0.541816E 01	0.187552E 02
ROW = 6	0.100000E 01 0.910627E 02	0.213526E 01	0.595197E 01	0.209872E 02
ROW = 7	0.100000E 01 0.111158E 03	0.233526E 01	0.688608E 01	0.251189E 02
ROW = 8	0.100000E 01 0.121019E 03	0.242617E 01	0.732720E 01	0.271172E 02
ROW = 9	0.100000E 01 0.141827E 03	0.259283E 01	0.819147E 01	0.312129E 02
ROW = 10	0.100000E 01 0.152951E 03	0.267617E 01	0.863750E 01	0.333723E 02
ROW = 11	0.100000E 01 0.169155E 03	0.278728E 01	0.925690E 01	0.364579E 02
ROW = 12	0.100000E 01 0.180163E 03	0.285871E 01	0.966528E 01	0.385291E 02

MATRIX OF MOMENTS FOR FORCING FUNCTION 2

OW = 1	0.000000E 00 0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
OW = 2	0.100000E 01 0.150000E 01	0.500000E 00	0.500000E 00	0.750000E 00
OW = 3	0.100000E 01 0.187440E 01	0.600000E 00	0.620000E 00	0.936000E 00
OW = 4	0.100000E 01 0.221382E 01	0.676923E 00	0.724142E 00	0.110311E 01
OW = 5	0.100000E 01 0.554816E 01	0.101026E 01	0.139765E 01	0.250076E 01
OW = 6	0.100000E 01 0.711381E 01	0.113526E 01	0.168146E 01	0.313130E 01
OW = 7	0.100000E 01 0.106823E 02	0.133526E 01	0.221556E 01	0.446064E 01
OW = 8	0.100000E 01 0.125498E 02	0.142617E 01	0.247487E 01	0.513560E 01

ROW = 9	0.100000E 01 0.169755E 02	0.159283E 01	0.300581E 01	0.663851E 01
ROW = 10	0.100000E 01 0.194621E 02	0.167617E 01	0.328517E 01	0.745980E 01
ROW = 11	0.100000E 01 0.233231E 02	0.178728E 01	0.368234E 01	0.868725E 01
ROW = 12	0.100000E 01 0.260469E 02	0.185871E 01	0.394787E 01	0.953322E 01

REDUCED SYSTEM

FORCING FUNCTION MATRIX B*

ROW = 1	0.100000E 01	0.941393E-11
ROW = 2	-0.999544E-03	0.133023E 01
ROW = 3	0.377982E-04	-0.357424E 00
ROW = 4	-0.420106E-04	0.499981E 00

A MATRIX

ROW = 1	-0.100000E 01	-0.151433E-10	0.813149E-11	-0.240211E-11
ROW = 2	0.133123E 01	-0.956635E 00	-0.507506E 00	0.133915E 00
ROW = 3	-0.357461E 00	0.169465E 01	-0.896009E 00	-0.441213E 00
ROW = 4	0.500023E 00	-0.179194E 01	0.444438E 01	-0.315243E 01

SYSTEM EIGENVALUES

VARIABLE	REAL	IMAG
1	-0.170239E 01	-0.118812E 01
2	-0.170239E 01	0.118812E 01
3	-0.100000E 01	0.000000E 00
4	-0.160029E 01	0.000000E 00

REDUCED MODEL MOMENTS

MATRIX OF MOMENTS FOR FORCING FUNCTION 1

ROW = 1	0.100000E 01 0.240000E 02	0.100000E 01	0.200000E 01	0.600000E 01
ROW = 2	0.100000E 01 0.514777E 02	0.159876E 01	0.382505E 01	0.124025E 02
ROW = 3	0.100000E 01 0.111246E 03	0.233385E 01	0.688675E 01	0.251415E 02
ROW = 4	0.100000E 01 0.180290E 03	0.285738E 01	0.966491E 01	0.385445E 02

MATRIX OF MOMENTS FOR FORCING FUNCTION 2

ROW = 1	-0.224993E-21 -0.819209E-10	-0.267703E-11	-0.629873E-11	-0.194394E-10
ROW = 2	0.100000E 01 0.186512E 01	0.598057E 00	0.626728E 00	0.926176E 00
ROW = 3	0.100000E 01 0.106671E 02	0.133298E 01	0.221703E 01	0.447652E 01
ROW = 4	0.100000E 01 0.260848E 02	0.185654E 01	0.394723E 01	0.954102E 01

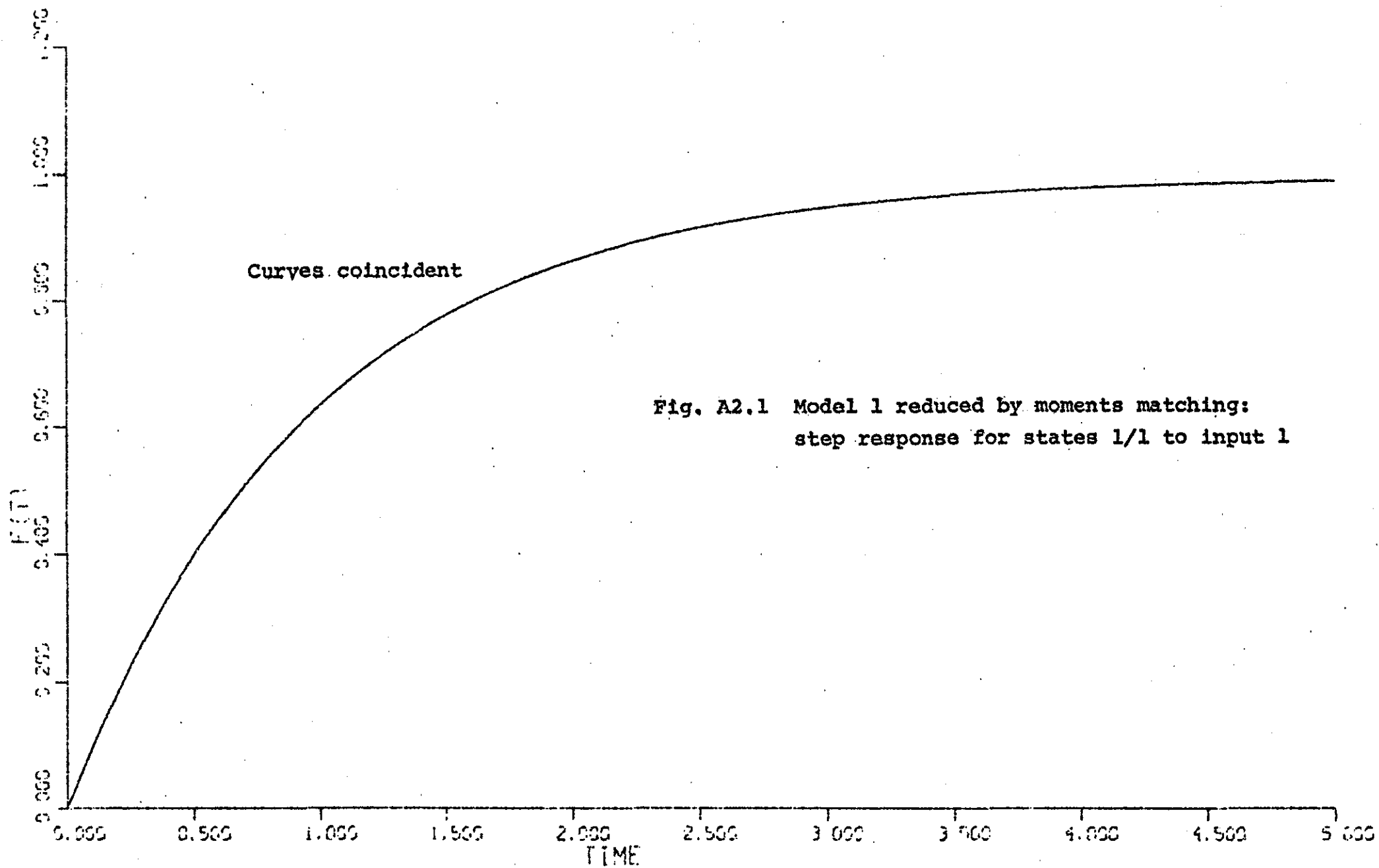


Fig. A2.1 Model 1 reduced by moments matching:
step response for states 1/1 to input 1

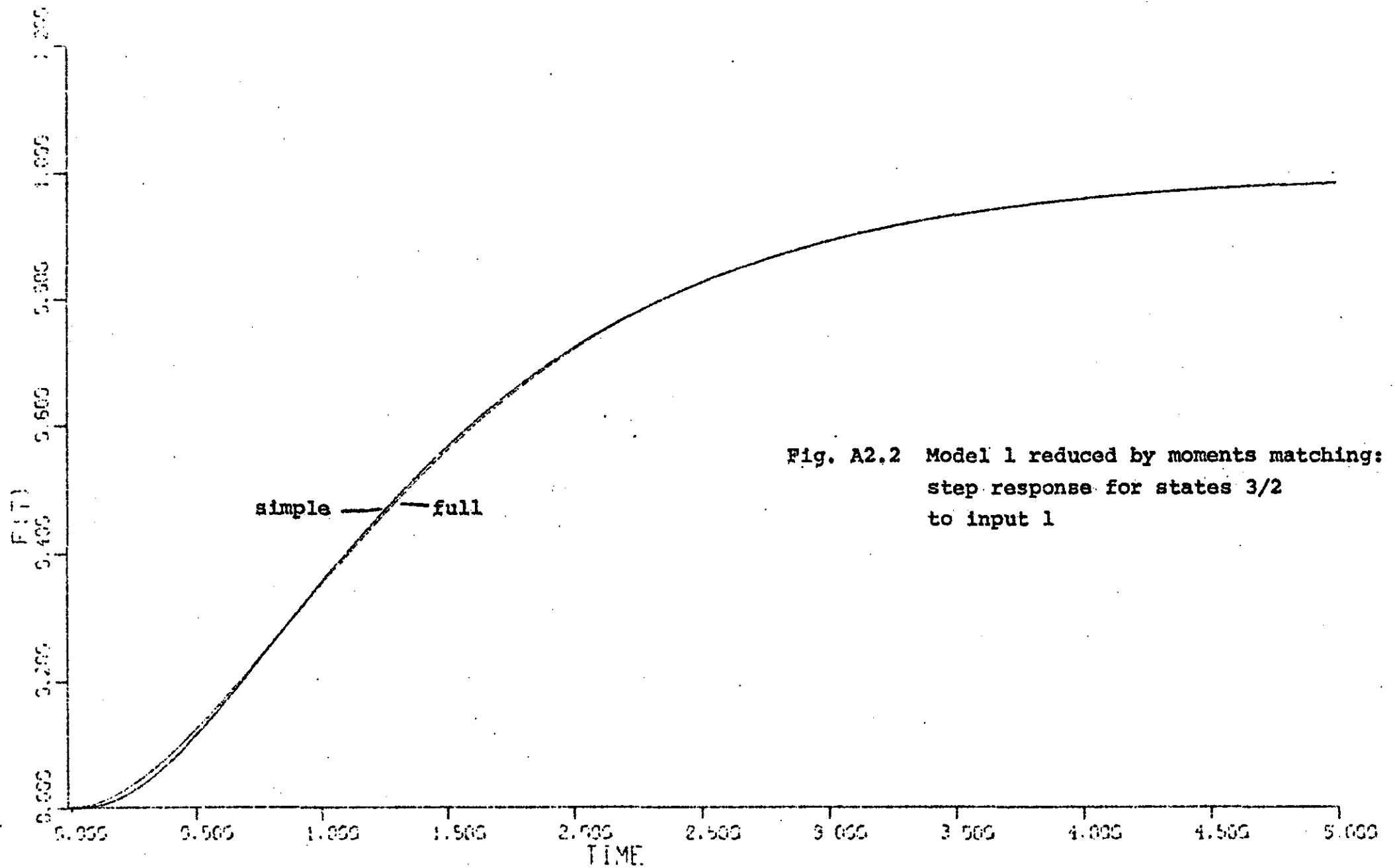


Fig. A2.2 Model 1 reduced by moments matching:
step response for states 3/2
to input 1

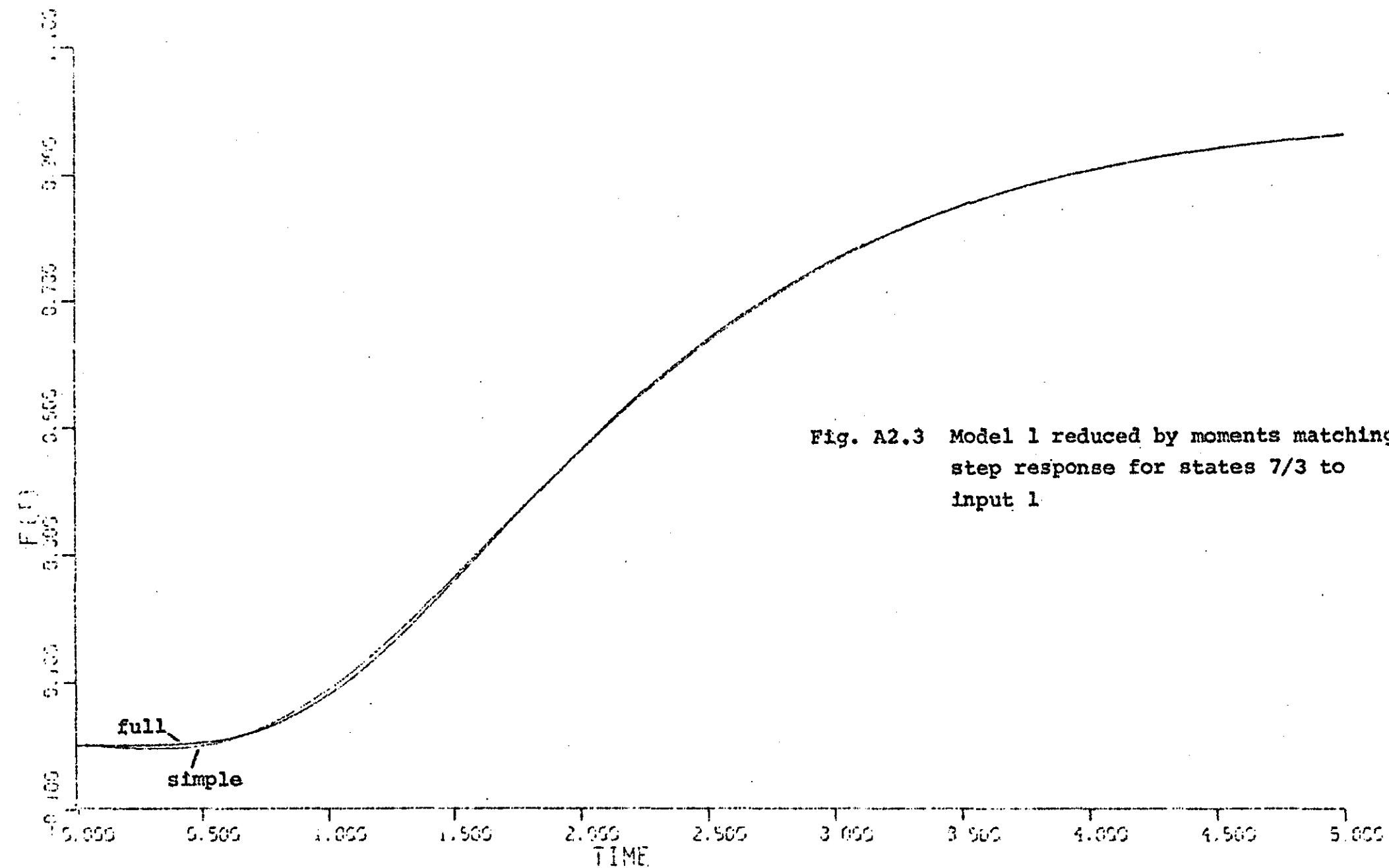


Fig. A2.3 Model 1 reduced by moments matching:
step response for states 7/3 to
input 1

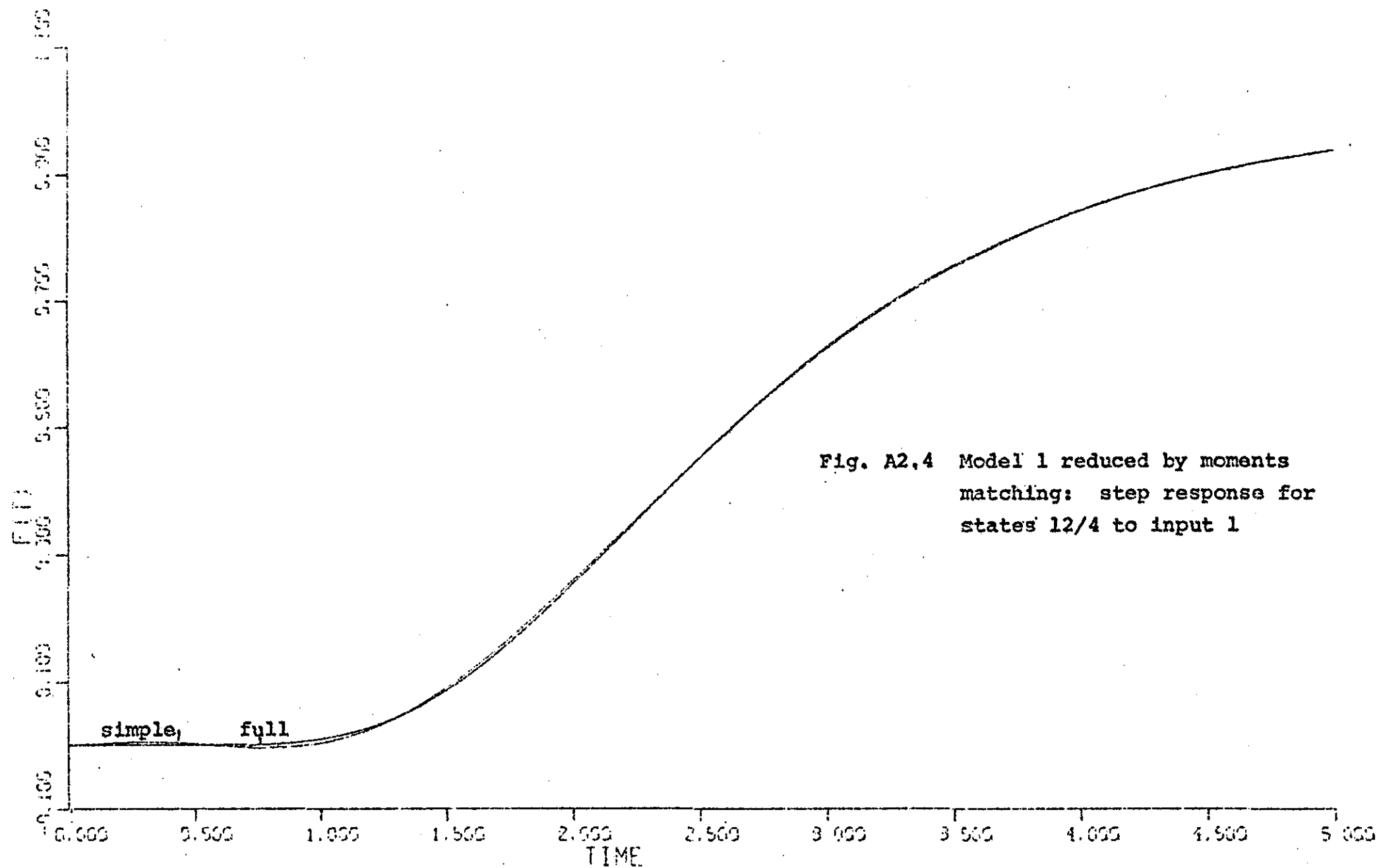


Fig. A2.4 Model 1 reduced by moments
matching: step response for
states 12/4 to input 1

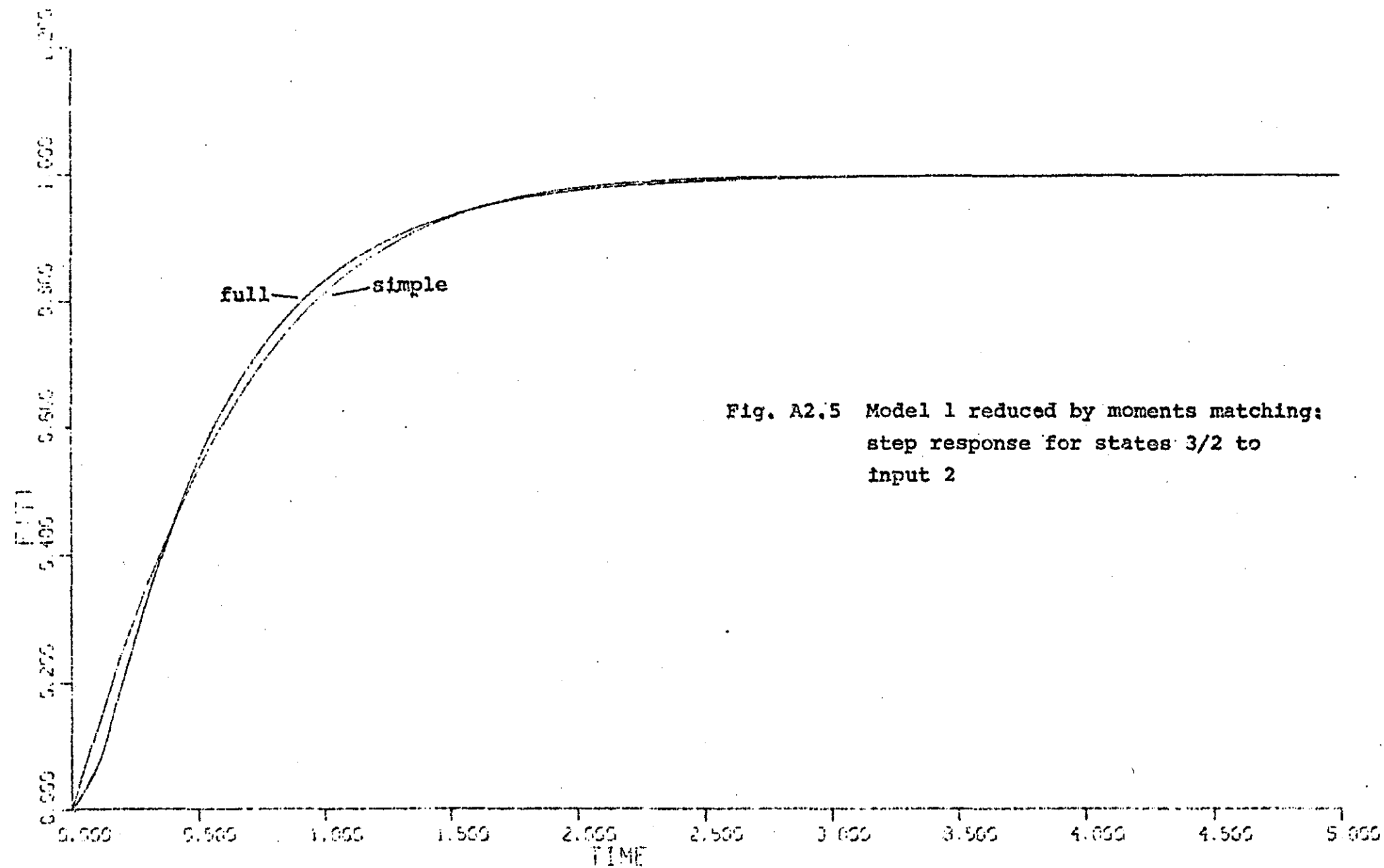


Fig. A2.5 Model 1 reduced by moments matching:
step response for states 3/2 to
input 2

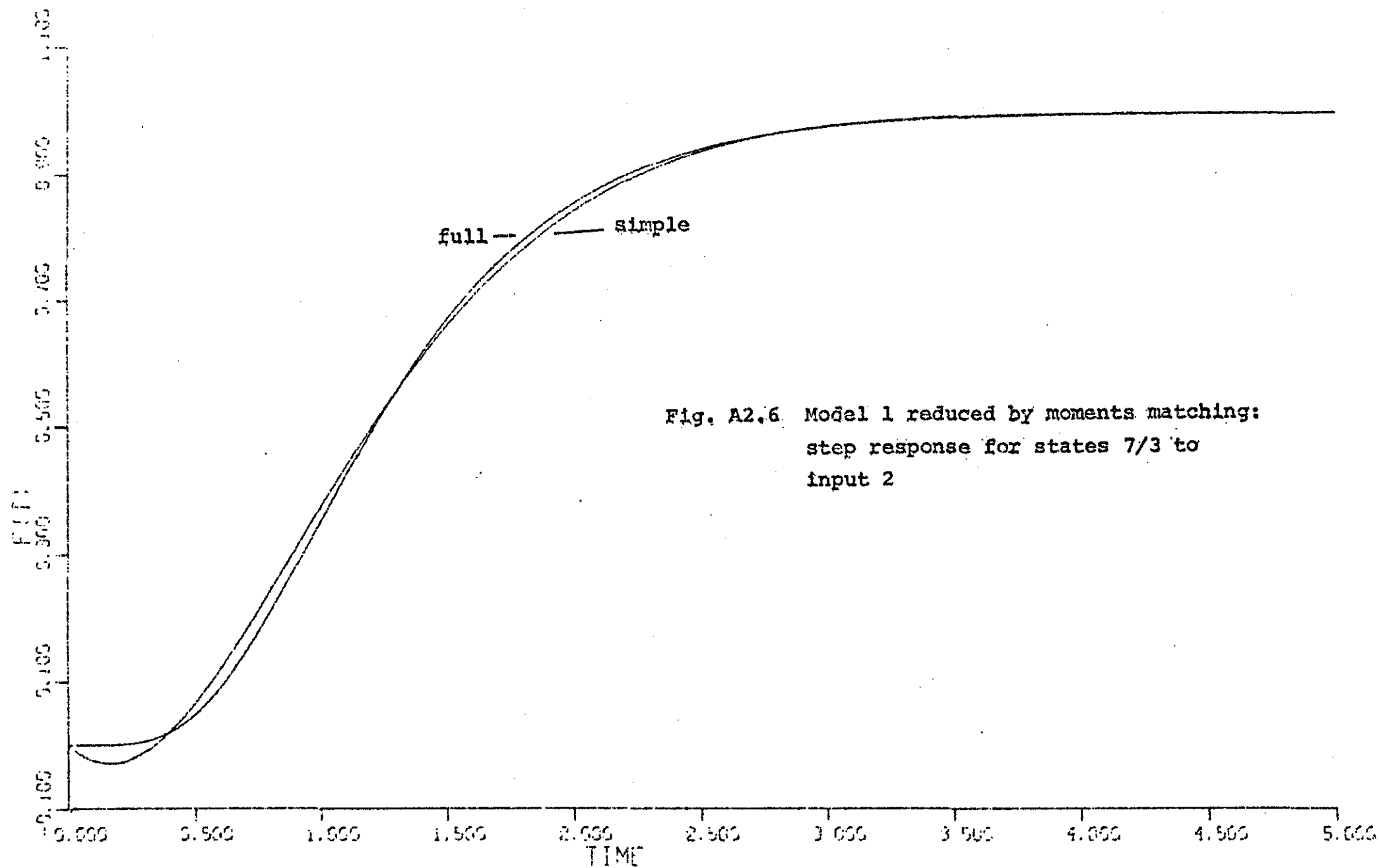


Fig. A2.6 Model 1 reduced by moments matching:
step response for states 7/3 to
input 2

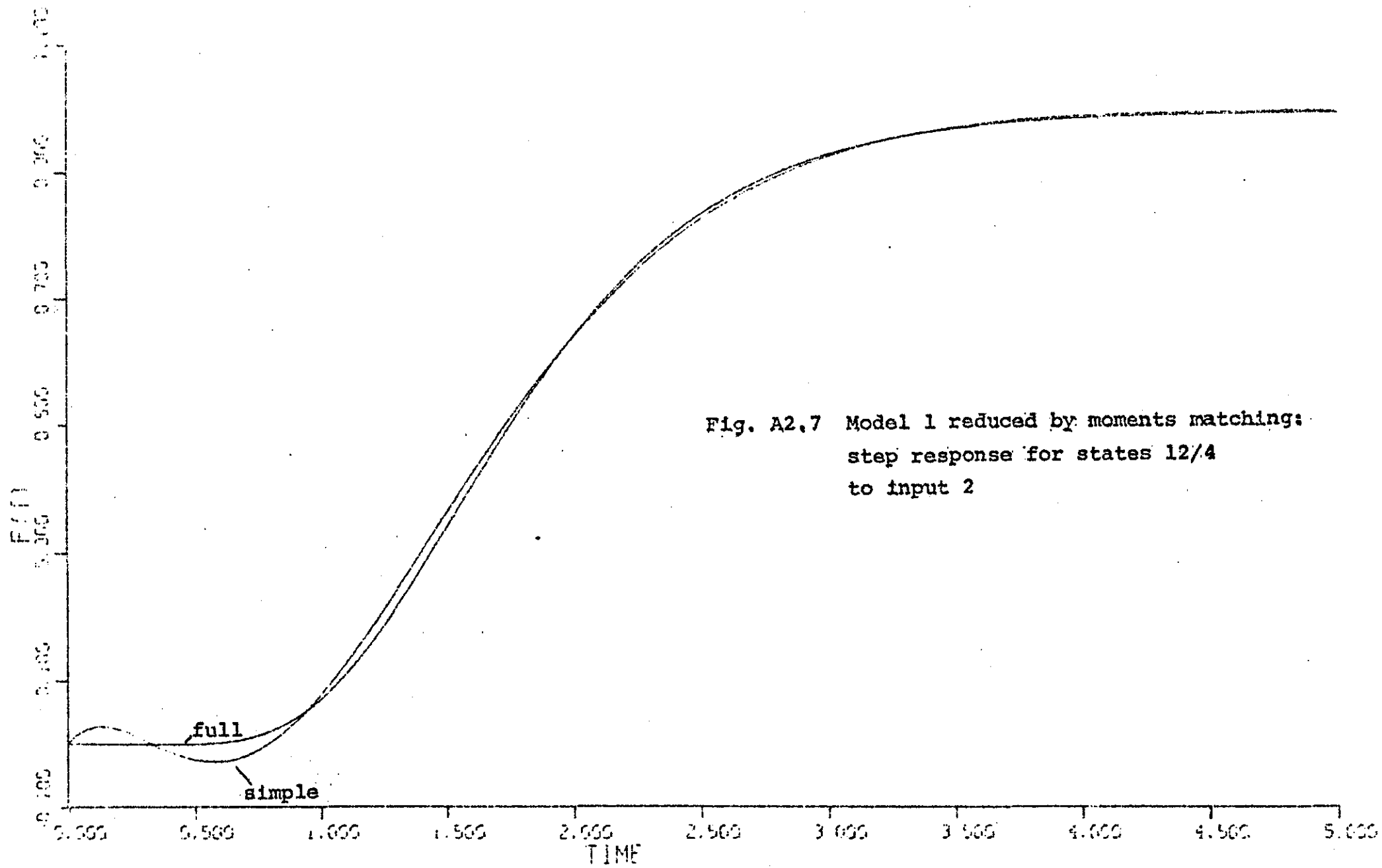


Fig. A2.7 Model 1 reduced by moments matching:
step response for states 12/4
to input 2

APPENDIX 2

A2.3 Model 2, an oscillating model, reduced
by matching the moments.

Contents

1. Full and reduced model data.
2. Step responses as detailed below.

Figure	Input	state x_i	reduced state x_i^*
A2.8	1	2	1
A2.9	1	5	2
A2.10	1	9	3
A2.11	1	12	4
A2.12	2	2	1
A2.13	2	5	2
A2.14	2	9	3
A2.15	2	12	4

N OSCILLATORY MODEL

ULL MODEL DATA

MATRIX

OW = 1	-0.400000E 00	-0.100000E 01	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
OW = 2	0.100000E 01	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
OW = 3	0.000000E 00	0.100000E 02	-0.100000E 02	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
OW = 4	0.000000E 00	0.000000E 00	0.100000E 01	-0.800000E 00
	-0.100000E 01	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
OW = 5	0.000000E 00	0.000000E 00	0.000000E 00	0.100000E 01
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
OW = 6	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.800000E 01	-0.800000E 01	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
OW = 7	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.100000E 01	-0.120000E 01	-0.100000E 01
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
OW = 8	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.100000E 01	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
OW = 9	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.600000E 01
	-0.600000E 01	0.000000E 00	0.000000E 00	0.000000E 00
OW = 10	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.120000E 02	-0.120000E 02	0.000000E 00	0.000000E 00
OW = 11	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.900000E 01	-0.900000E 01	0.000000E 00
OW = 12	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.140000E 02	-0.140000E 02

SYSTEM EIGENVALUES

VARIABLE	REAL	IMAG
1	-0.200000E 00	-0.972796E 00
2	-0.200000E 00	0.972796E 00
3	-0.400000E 00	-0.916515E 00
4	-0.400000E 00	0.916515E 00
5	-0.600000E 00	-0.800000E 00
6	-0.600000E 00	0.800000E 00
7	-0.600000E 01	0.000000E 00
8	-0.100000E 02	0.000000E 00
9	-0.800000E 01	0.000000E 00
10	-0.120000E 02	0.000000E 00
11	-0.900000E 01	0.000000E 00
12	-0.140000E 02	0.000000E 00

FORCING FUNCTION MATRIX B

1W = 1	0.100000E 01	0.100000E 01
2W = 2	0.000000E 00	0.000000E 00
3W = 3	0.000000E 00	0.100000E 01
4W = 4	0.000000E 00	0.000000E 00
5W = 5	0.000000E 00	0.000000E 00
6W = 6	0.000000E 00	0.000000E 00
7W = 7	0.000000E 00	0.000000E 00
8W = 8	0.000000E 00	0.000000E 00
9W = 9	0.000000E 00	0.000000E 00
10W = 10	0.000000E 00	0.000000E 00
11W = 11	0.000000E 00	0.000000E 00
12W = 12	0.000000E 00	0.000000E 00

DEGREES OF INTEREST= 2 5 9 12

MATRIX OF MOMENTS FOR FORCING FUNCTION 1

1W = 1	0.000000E 00	-0.100000E 01	-0.800000E 00	0.304000E 01
	0.176640E 02			

OW = 2	0.100000E 01 0.130944E 02	0.400000E 00	-0.168000E 01	-0.441600E 01
OW = 3	0.100000E 01 0.111384E 02	0.500000E 00	-0.158000E 01	-0.489000E 01
OW = 4	0.000000E 00 0.651600E 02	-0.100000E 01	-0.260000E 01	0.450000E 01
OW = 5	0.100000E 01 -0.229896E 02	0.130000E 01	-0.190000E 01	-0.162900E 02
OW = 6	0.100000E 01 -0.313491E 02	0.142500E 01	-0.114375E 01	-0.167189E 02
OW = 7	0.000000E 00 0.844256E 02	-0.100000E 01	-0.525000E 01	-0.946875E 01
OW = 8	0.100000E 01 -0.170535E 03	0.262500E 01	0.315625E 01	-0.211064E 02
OW = 9	0.100000E 01 -0.183243E 03	0.279167E 01	0.408681E 01	-0.190630E 02
OW = 10	0.100000E 01 -0.189217E 03	0.287500E 01	0.456597E 01	-0.179215E 02
OW = 11	0.100000E 01 -0.196408E 03	0.298611E 01	0.522955E 01	-0.161783E 02
OW = 12	0.100000E 01 -0.200683E 03	0.305754E 01	0.566634E 01	-0.149641E 02

ATRIX OF MOMENTS FOR FORCING FUNCTION 2

OW = 1	0.000000E 00 0.176640E 02	-0.100000E 01	-0.800000E 00	0.504000E 01
OW = 2	0.100000E 01 0.130944E 02	0.400000E 00	-0.168000E 01	-0.441600E 01
OW = 3	0.110000E 01 0.111386E 02	0.510000E 00	-0.157800E 01	-0.488940E 01
OW = 4	0.000000E 00 0.678360E 02	-0.110000E 01	-0.278000E 01	0.466200E 01
OW = 5	0.110000E 01 -0.244822E 02	0.139000E 01	-0.155400E 01	-0.169590E 02
OW = 6	0.110000E 01 -0.331814E 02	0.152750E 01	-0.117212E 01	-0.173985E 02
OW = 7	0.000000E 00 0.880832E 02	-0.110000E 01	-0.569500E 01	-0.103856E 02
OW = 8	0.110000E 01 -0.180424E 03	0.284750E 01	0.346187E 01	-0.220208E 02

OW = 9	0.110000E 01 -0.193614E 03	0.303083E 01	0.447215E 01	-0.197847E 02
OW = 10	0.110000E 01 -0.199792E 03	0.312250E 01	0.499257E 01	-0.185366E 02
OW = 11	0.110000E 01 -0.207184E 03	0.324472E 01	0.571362E 01	-0.166320E 02
OW = 12	0.110000E 01 -0.211558E 03	0.332329E 01	0.618838E 01	-0.153060E 02

REDUCED SYSTEM

FORCING FUNCTION MATRIX B*

OW = 1	0.253834E 00	0.343120E 00
OW = 2	-0.232904E 00	-0.158635E 00
OW = 3	0.199987E-01	0.149887E-01
OW = 4	-0.130779E-01	-0.987845E-02

MATRIX

OW = 1	0.639027E 00	-0.112362E 01	0.170541E 01	-0.147465E 01
OW = 2	0.975595E 00	-0.618674E 00	0.763401E 00	-0.887418E 00
OW = 3	-0.700986E-01	0.214656E 00	0.281422E 01	-0.297878E 01
OW = 4	0.450720E-01	-0.120148E 00	0.425444E 01	-0.416629E 01

SYSTEM EIGENVALUES

VARIABLE	REAL	IMAG
1	-0.122771E 00	-0.779903E 00
2	-0.122771E 00	0.779903E 00
3	-0.543084E 00	-0.593351E 00
4	-0.543084E 00	0.593351E 00

REDUCED MODEL MOMENTS

MATRIX OF MOMENTS FOR FORCING FUNCTION 1

NU = 1	0.100000E 01 0.114394E 02	0.477618E 00	-0.159913E 01	-0.482914E 01
NU = 2	0.100000E 01 -0.290927E 02	0.140752E 01	-0.124992E 01	-0.166804E 02
NU = 3	0.100000E 01 -0.190234E 03	0.289651E 01	0.465293E 01	-0.177878E 02
NU = 4	0.100000E 01 -0.205976E 03	0.316240E 01	0.628821E 01	-0.132074E 02

MATRIX OF MOMENTS FOR FORCING FUNCTION 2

NU = 1	0.100000E 01 0.144397E 02	0.341604E 00	-0.176996E 01	-0.410945E 01
NU = 2	0.110000E 01 -0.196169E 02	0.130606E 01	-0.177118E 01	-0.167253E 02
NU = 3	0.110000E 01 -0.188688E 03	0.294892E 01	0.400568E 01	-0.209550E 02
NU = 4	0.110000E 01 -0.208157E 03	0.324137E 01	0.567836E 01	-0.168717E 02

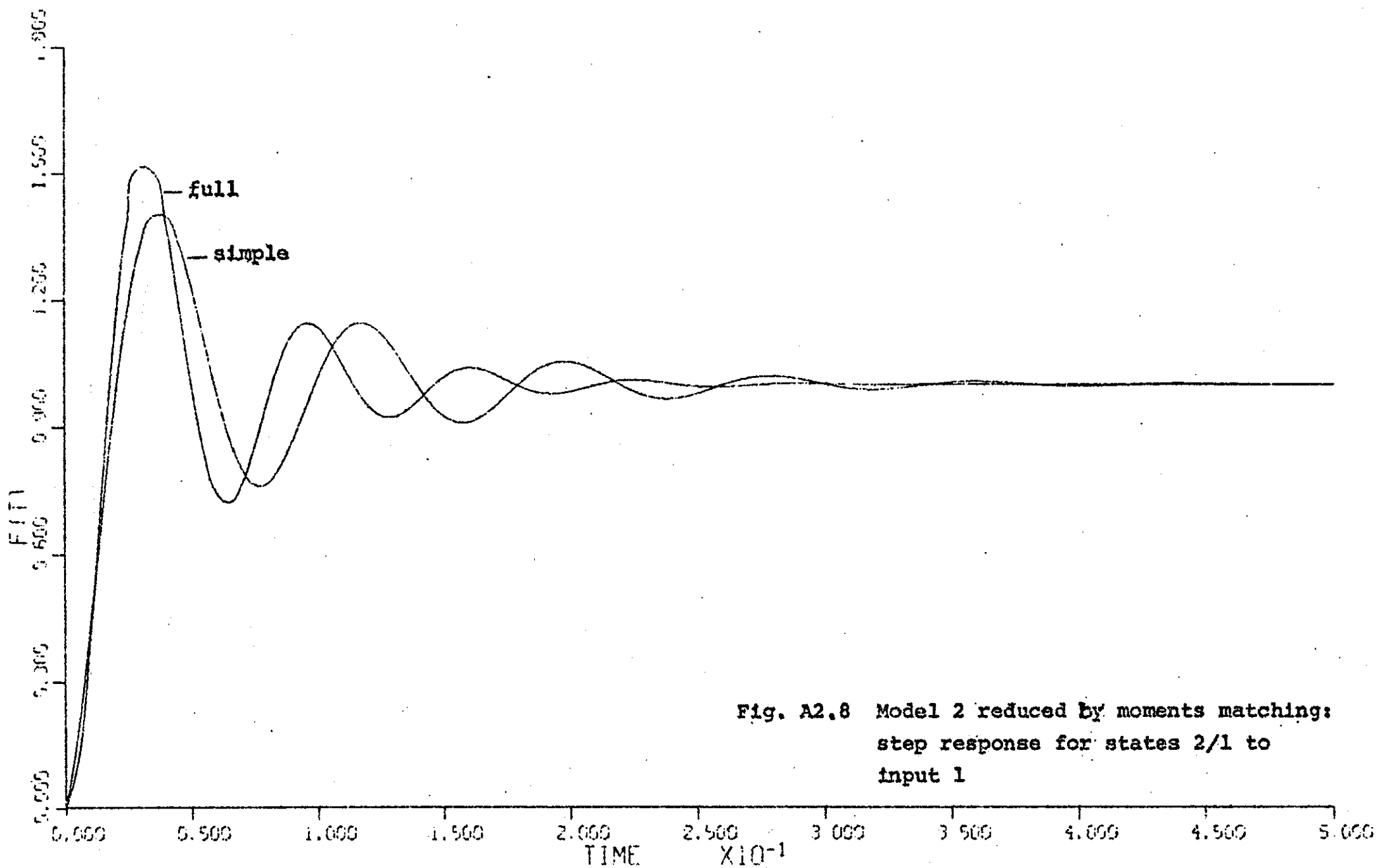


Fig. A2.8 Model 2 reduced by moments matching:
step response for states 2/1 to
input 1

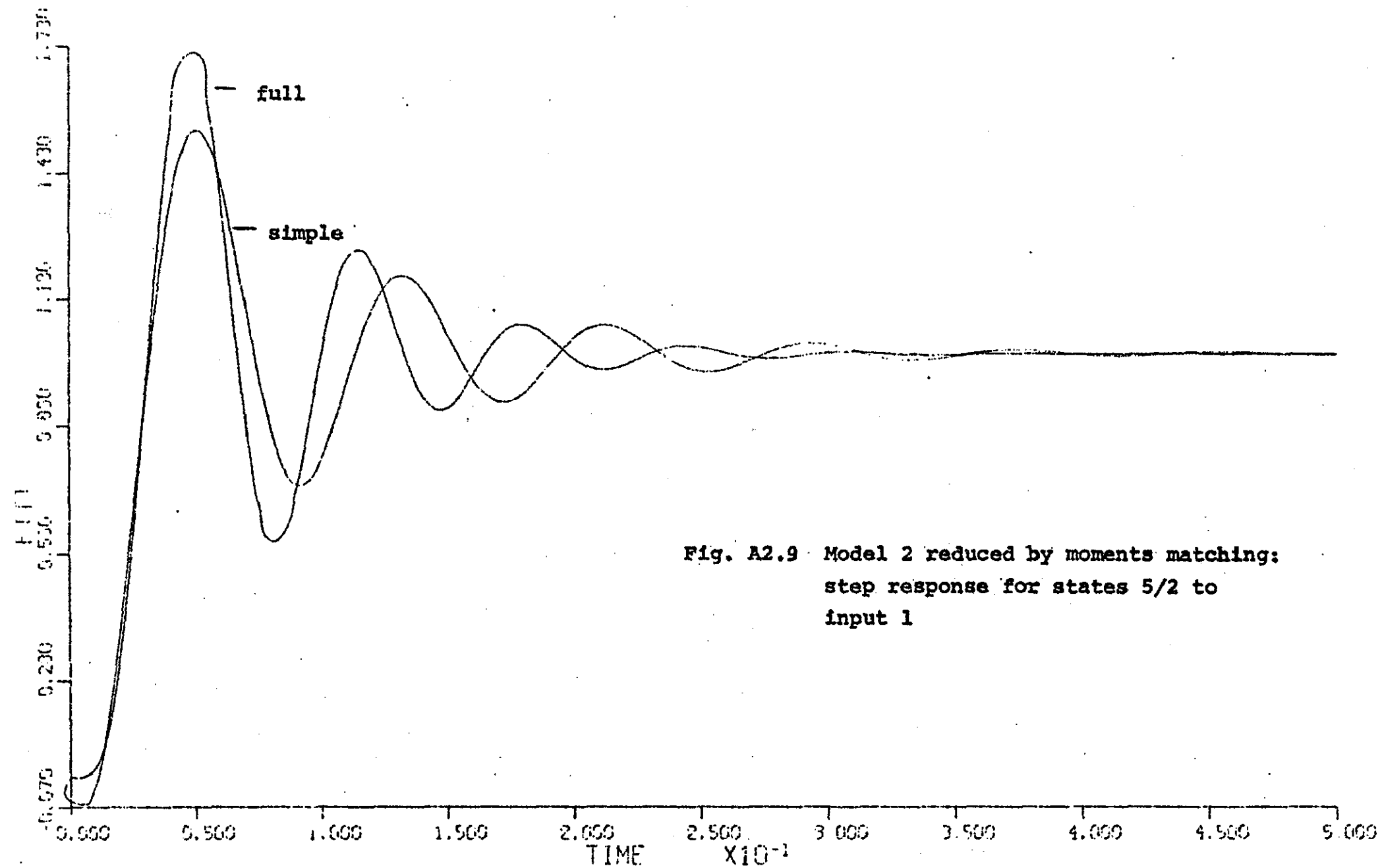


Fig. A2.9 Model 2 reduced by moments matching:
step response for states 5/2 to
input 1

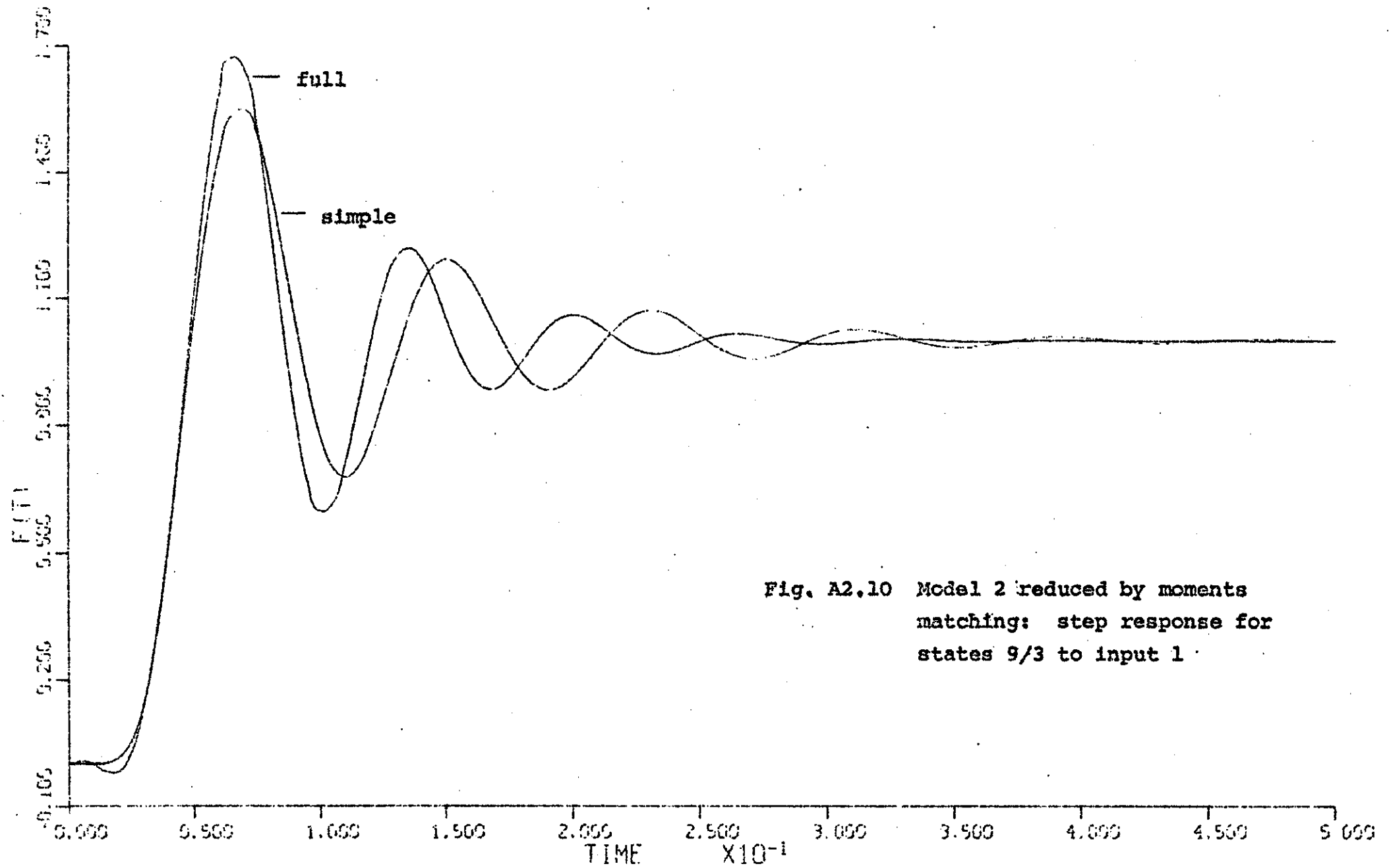


Fig. A2.10 Model 2 reduced by moments
matching: step response for
states 9/3 to input 1

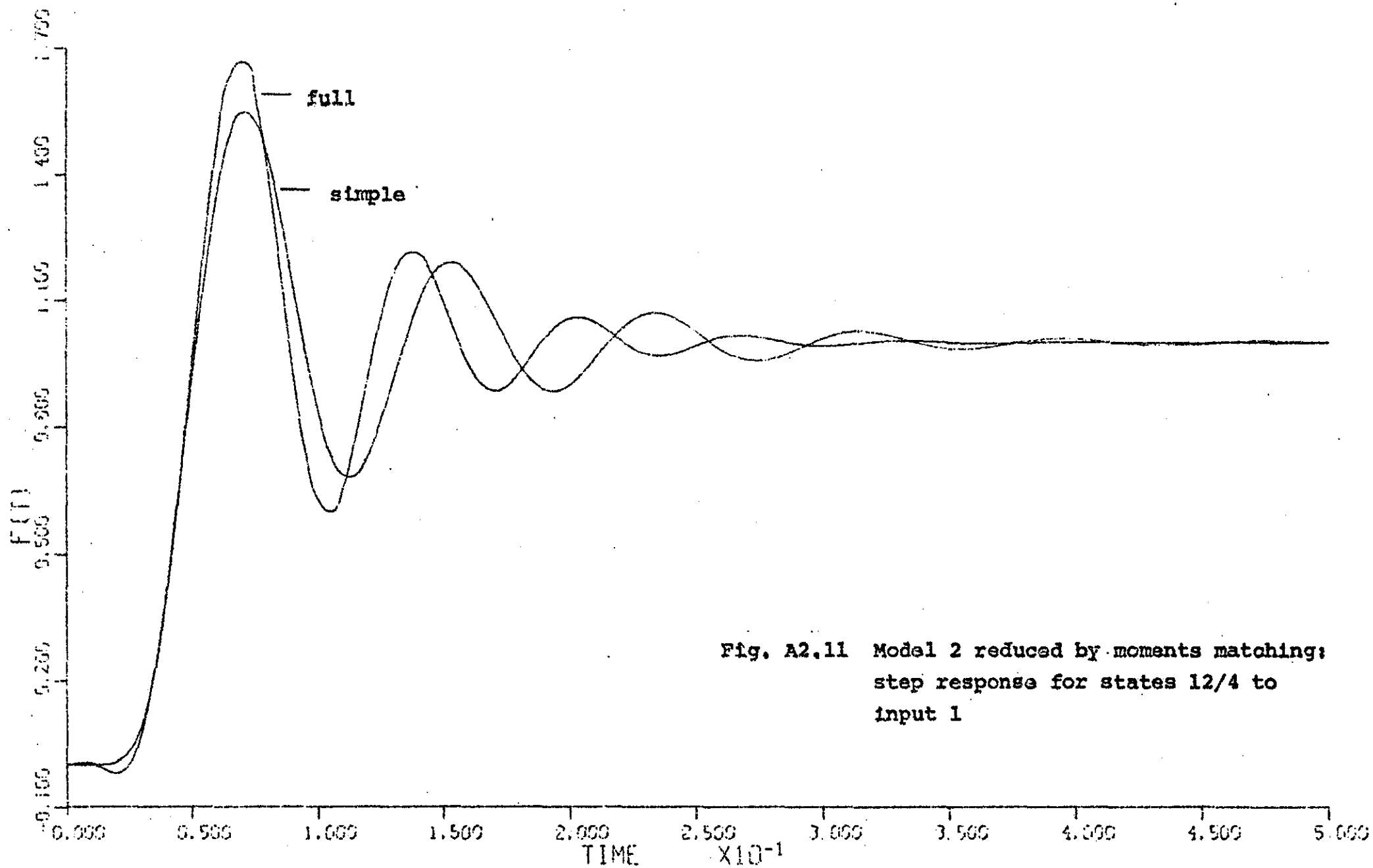


Fig. A2.11 Model 2 reduced by moments matching:
step response for states 12/4 to
input 1

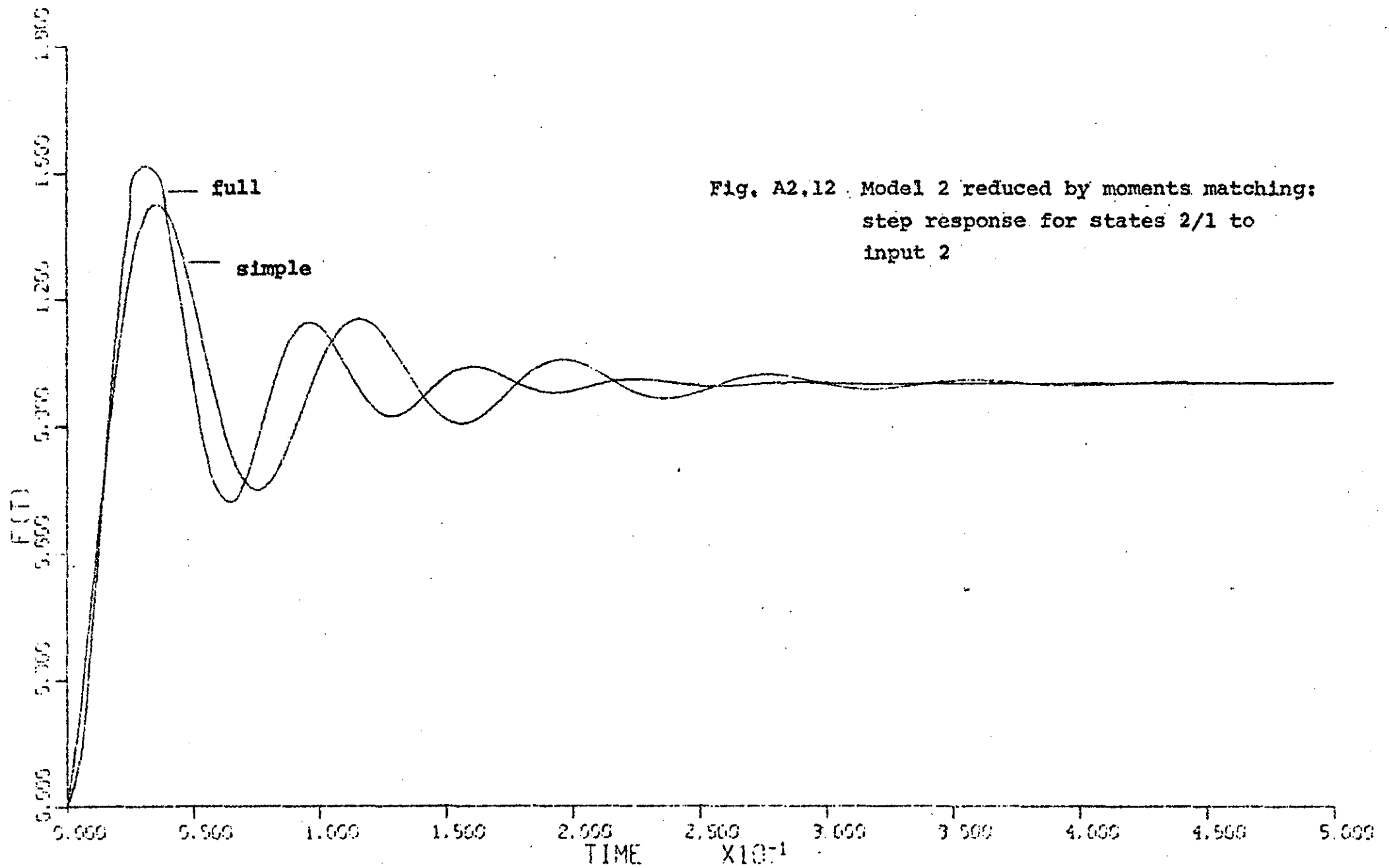


Fig. A2.12 Model 2 reduced by moments matching:
step response for states 2/1 to
input 2

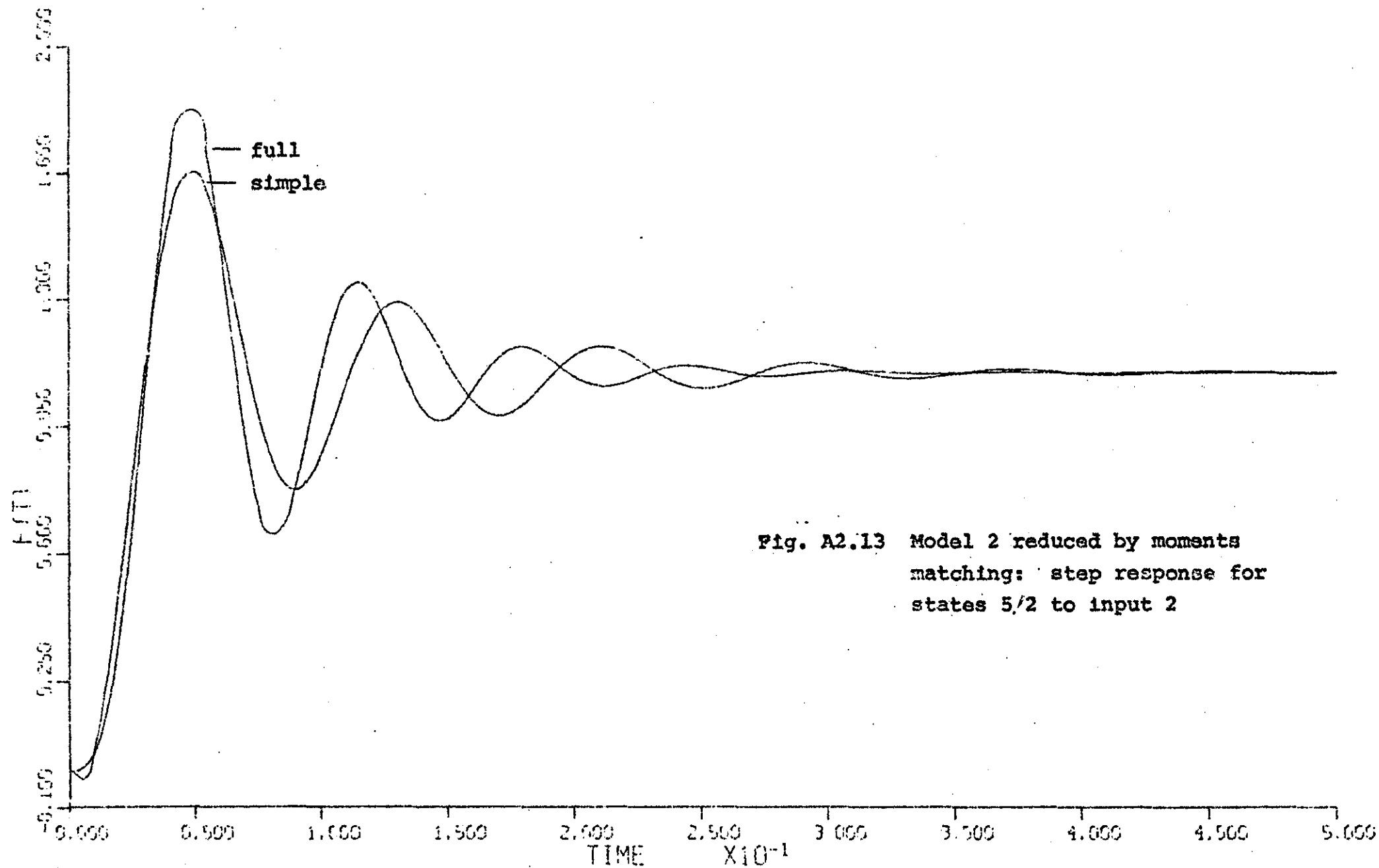


Fig. A2.13 Model 2 reduced by moments
matching: step response for
states 5/2 to input 2

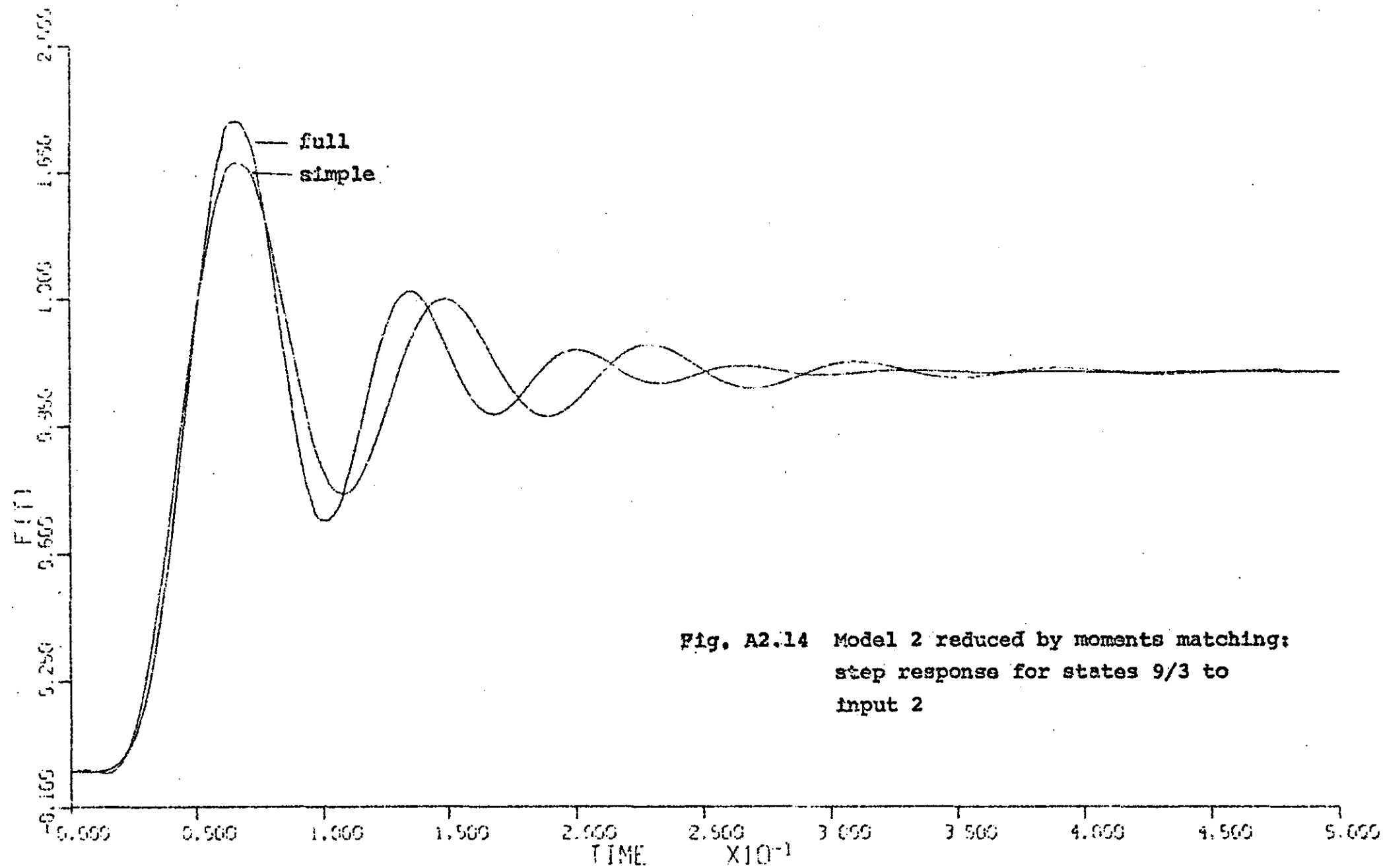


Fig. A2.14 Model 2 reduced by moments matching:
step response for states 9/3 to
input 2

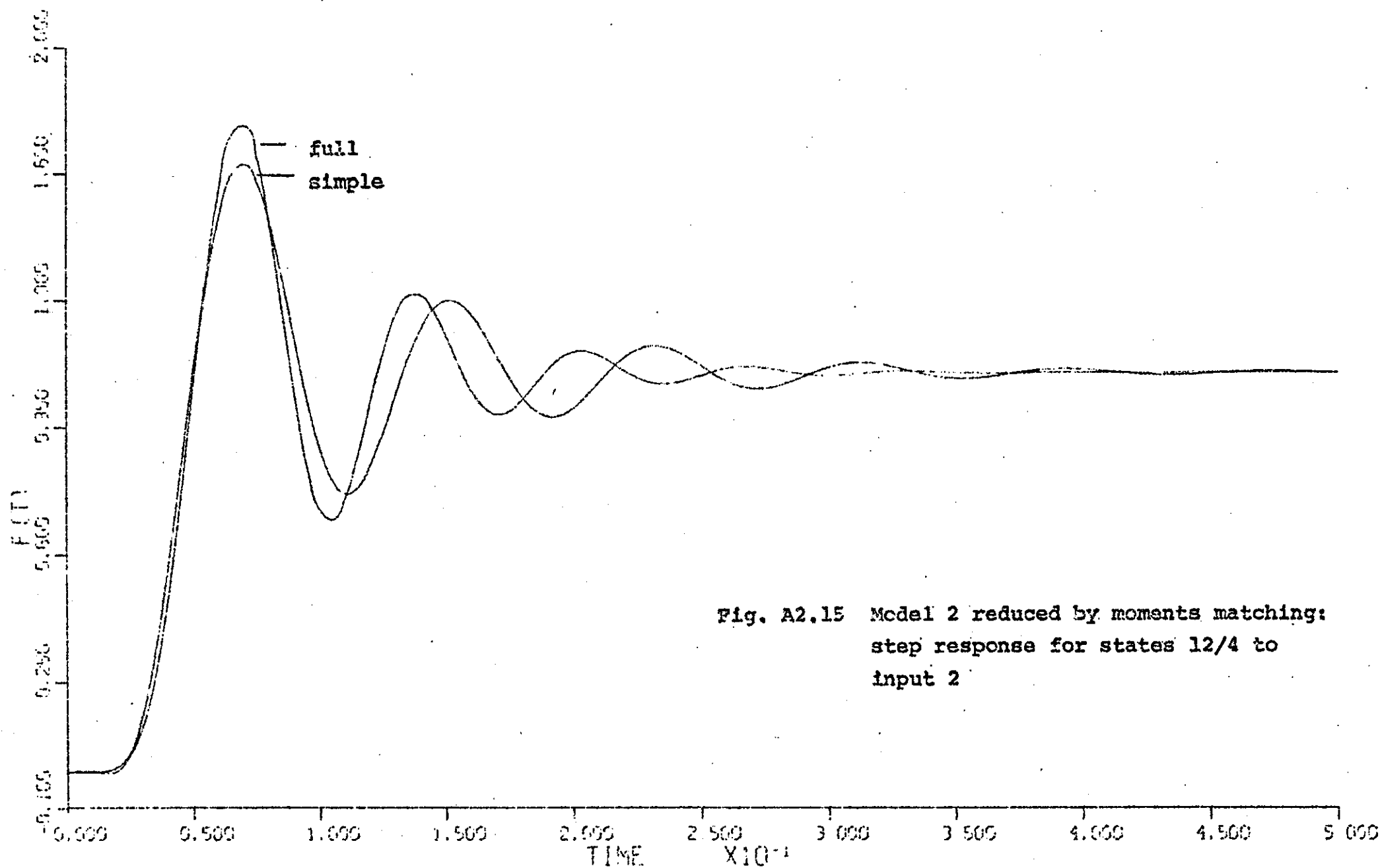


Fig. A2.15 Model 2 reduced by moments matching:
step response for states 12/4 to
input 2

APPENDIX 2

A2.4 Model 3, an inverting model, reduced
by matching the moments.

Contents

1. Full and reduced model data.
2. Step responses as detailed below.

Figure	Input	state x_i	reduced state x_i^*
A2.16	1	1	1
A2.17	1	3	2
A2.18	1	5	3
A2.19	1	8	4
A2.20	2	1	1
A2.21	2	3	2
A2.22	2	5	3
A2.23	2	8	4

AN INVERTING MODEL

FULL MODEL DATA

A MATRIX

ROW = 1	-0.167500E 02	0.112500E 02	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 2	-0.192500E 02	0.127500E 02	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 3	0.000000E 00	0.900000E 01	-0.900000E 01	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 4	0.000000E 00	0.000000E 00	0.800000E 01	-0.800000E 01
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 5	0.000000E 00	0.000000E 00	0.000000E 00	0.700000E 01
	-0.700000E 01	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 6	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.600000E 01	-0.600000E 01	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 7	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.750000E 01	-0.750000E 01	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 8	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.650000E 01	-0.650000E 01
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 9	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.500000E 01
	-0.500000E 01	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 10	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.550000E 01	-0.550000E 01	0.000000E 00	0.000000E 00
ROW = 11	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.850000E 01	-0.850000E 01	0.000000E 00
ROW = 12	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.450000E 01	-0.450000E 01

SYSTEM EIGENVALUES

VARIABLE	REAL	IMAG
1	-0.100000E 01	0.000000E 00
2	-0.300000E 01	0.000000E 00
3	-0.900000E 01	0.000000E 00
4	-0.649976E 01	0.000000E 00
5	-0.600018E 01	0.000000E 00
6	-0.700019E 01	0.000000E 00
7	-0.549992E 01	0.000000E 00
8	-0.749991E 01	0.000000E 00
9	-0.500002E 01	0.000000E 00
10	-0.800002E 01	0.000000E 00
11	-0.450000E 01	0.000000E 00
12	-0.850000E 01	0.000000E 00

FORCING FUNCTION MATRIX B

ROW = 1	0.100000E 01	0.100000E 01
ROW = 2	0.100000E 01	0.100000E 01
ROW = 3	0.000000E 00	0.100000E 01
ROW = 4	0.000000E 00	0.000000E 00
ROW = 5	0.000000E 00	0.100000E 01
ROW = 6	0.000000E 00	0.000000E 00
ROW = 7	0.000000E 00	0.000000E 00
ROW = 8	0.000000E 00	0.100000E 01
ROW = 9	0.000000E 00	0.000000E 00
ROW = 10	0.000000E 00	0.000000E 00
ROW = 11	0.000000E 00	0.100000E 01
ROW = 12	0.000000E 00	0.000000E 00

VARIABLES OF INTEREST= 1 3 5 8

MATRIX OF MOMENTS FOR FORCING FUNCTION 1

ROW = 1	-0.500000E 00	-0.100000E 01	-0.233333E 01	-0.733333E 01
	-0.297778E 02			

ROW = 2	-0.833333E 00 -0.417284E 02	-0.144444E 01	-0.329630E 01	-0.102963E 02
ROW = 3	-0.833333E 00 -0.468435E 02	-0.153704E 01	-0.363786E 01	-0.115089E 02
ROW = 4	-0.833333E 00 -0.533570E 02	-0.164120E 01	-0.404816E 01	-0.130270E 02
ROW = 5	-0.833333E 00 -0.619155E 02	-0.176025E 01	-0.455109E 01	-0.149774E 02
ROW = 6	-0.833333E 00 -0.736285E 02	-0.189914E 01	-0.518414E 01	-0.175695E 02
ROW = 7	-0.833333E 00 -0.842192E 02	-0.201025E 01	-0.572020E 01	-0.198576E 02
ROW = 8	-0.833333E 00 -0.982508E 02	-0.213846E 01	-0.637819E 01	-0.228014E 02
ROW = 9	-0.833333E 00 -0.119996E 03	-0.230512E 01	-0.730024E 01	-0.271815E 02
ROW = 10	-0.833333E 00 -0.143015E 03	-0.245664E 01	-0.819356E 01	-0.316507E 02
ROW = 11	-0.833333E 00 -0.159370E 03	-0.255468E 01	-0.879466E 01	-0.347547E 02
ROW = 12	-0.833333E 00 -0.196196E 03	-0.273986E 01	-0.100124E 02	-0.414297E 02

MATRIX OF MOMENTS FOR FORCING FUNCTION 2

ROW = 1	-0.500000E 00 -0.297778E 02	-0.100000E 01	-0.233333E 01	-0.733333E 01
ROW = 2	-0.833333E 00 -0.417284E 02	-0.144444E 01	-0.329630E 01	-0.102963E 02
ROW = 3	-0.722222E 00 -0.468431E 02	-0.152469E 01	-0.363512E 01	-0.115080E 02
ROW = 4	-0.722222E 00 -0.533543E 02	-0.161497E 01	-0.403886E 01	-0.130226E 02
ROW = 5	-0.579365E 00 -0.619037E 02	-0.169774E 01	-0.452393E 01	-0.169614E 02
ROW = 6	-0.579365E 00 -0.735853E 02	-0.179430E 01	-0.512202E 01	-0.175224E 02
ROW = 7	-0.579365E 00 -0.841298E 02	-0.187155E 01	-0.562110E 01	-0.197709E 02
ROW = 8	-0.425519E 00 -0.980623E 02	-0.193701E 01	-0.621711E 01	-0.226403E 02

ROW = 9	-0.425519E 00 -0.119547E 03	-0.202211E 01	-0.702595E 01	-0.268559E 02
ROW = 10	-0.425519E 00 -0.142168E 03	-0.209948E 01	-0.778940E 01	-0.311046E 02
ROW = 11	-0.307872E 00 -0.158183E 03	-0.213570E 01	-0.829192E 01	-0.340312E 02
ROW = 12	-0.307872E 00 -0.193927E 03	-0.220412E 01	-0.927152E 01	-0.402122E 02

REDUCED SYSTEM

FORCING FUNCTION MATRIX B*

ROW = 1	0.129231E 01	0.299442E 00
ROW = 2	0.233058E 00	0.439966E 00
ROW = 3	-0.638803E 00	0.619053E 00
ROW = 4	-0.687154E 00	0.270035E 00

A MATRIX

ROW = 1	-0.235112E 02	0.217581E 02	-0.691083E 01	0.810264E 00
ROW = 2	-0.539242E 01	0.839650E 01	-0.553048E 01	0.649105E 00
ROW = 3	0.122183E 02	-0.594858E 01	-0.181652E 01	-0.332415E 00
ROW = 4	0.124299E 02	-0.117367E 02	0.690159E 01	-0.344738E 01

SYSTEM EIGENVALUES

VARIABLE	REAL	IMAG
1	-0.173383E 02	0.000000E 00
2	-0.107454E 01	-0.115700E 01
3	-0.107454E 01	0.115700E 01
4	-0.891212E 00	0.000000E 00

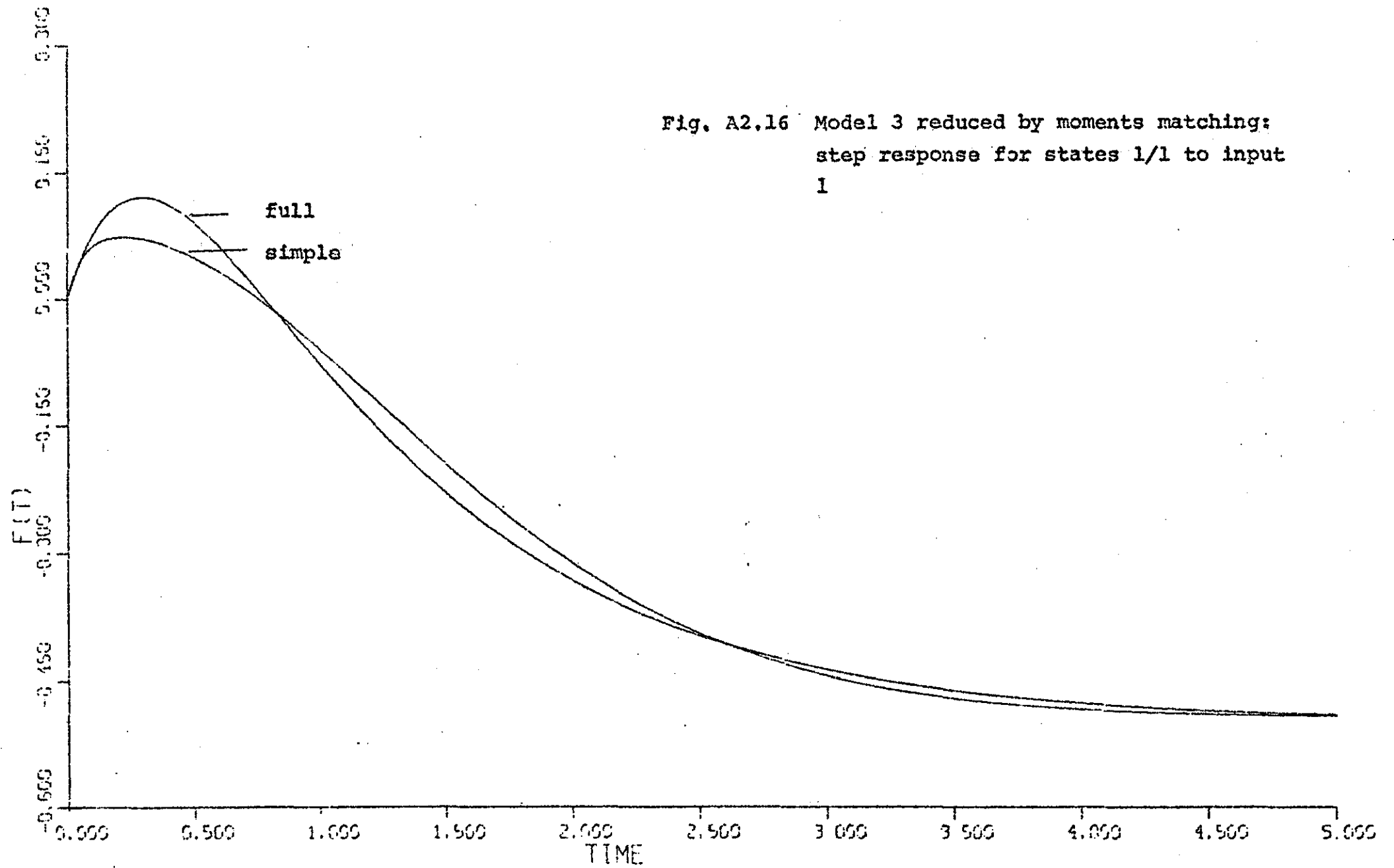
REDUCED MODEL MOMENTS

MATRIX OF MOMENTS FOR FORCING FUNCTION 1

ROW = 1	-0.500000E 00 -0.299120E 02	-0.998913E 00	-0.234292E 01	-0.725229E 01
ROW = 2	-0.833333E 00 -0.470196E 02	-0.153541E 01	-0.365194E 01	-0.113904E 02
ROW = 3	-0.833333E 00 -0.619848E 02	-0.175858E 01	-0.456583E 01	-0.148669E 02
ROW = 4	-0.833333E 00 -0.981999E 02	-0.213670E 01	-0.639478E 01	-0.226981E 02

MATRIX OF MOMENTS FOR FORCING FUNCTION 2

ROW = 1	-0.500000E 00 -0.292204E 02	-0.999762E 00	-0.236222E 01	-0.720834E 01
ROW = 2	-0.722222E 00 -0.459781E 02	-0.152434E 01	-0.367806E 01	-0.113285E 02
ROW = 3	-0.579365E 00 -0.608702E 02	-0.169737E 01	-0.456706E 01	-0.148089E 02
ROW = 4	-0.425519E 00 -0.968126E 02	-0.193661E 01	-0.626182E 01	-0.225185E 02



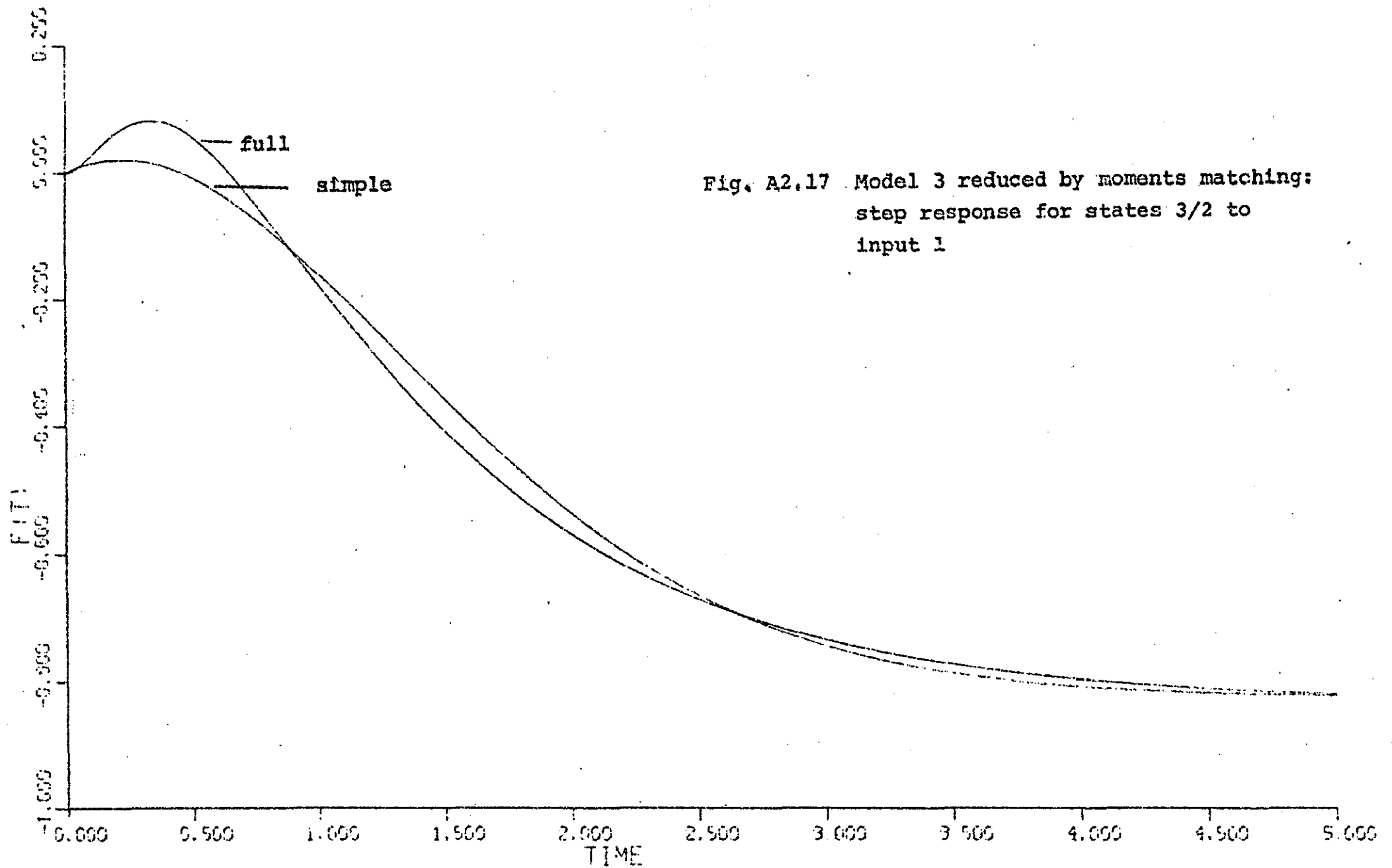
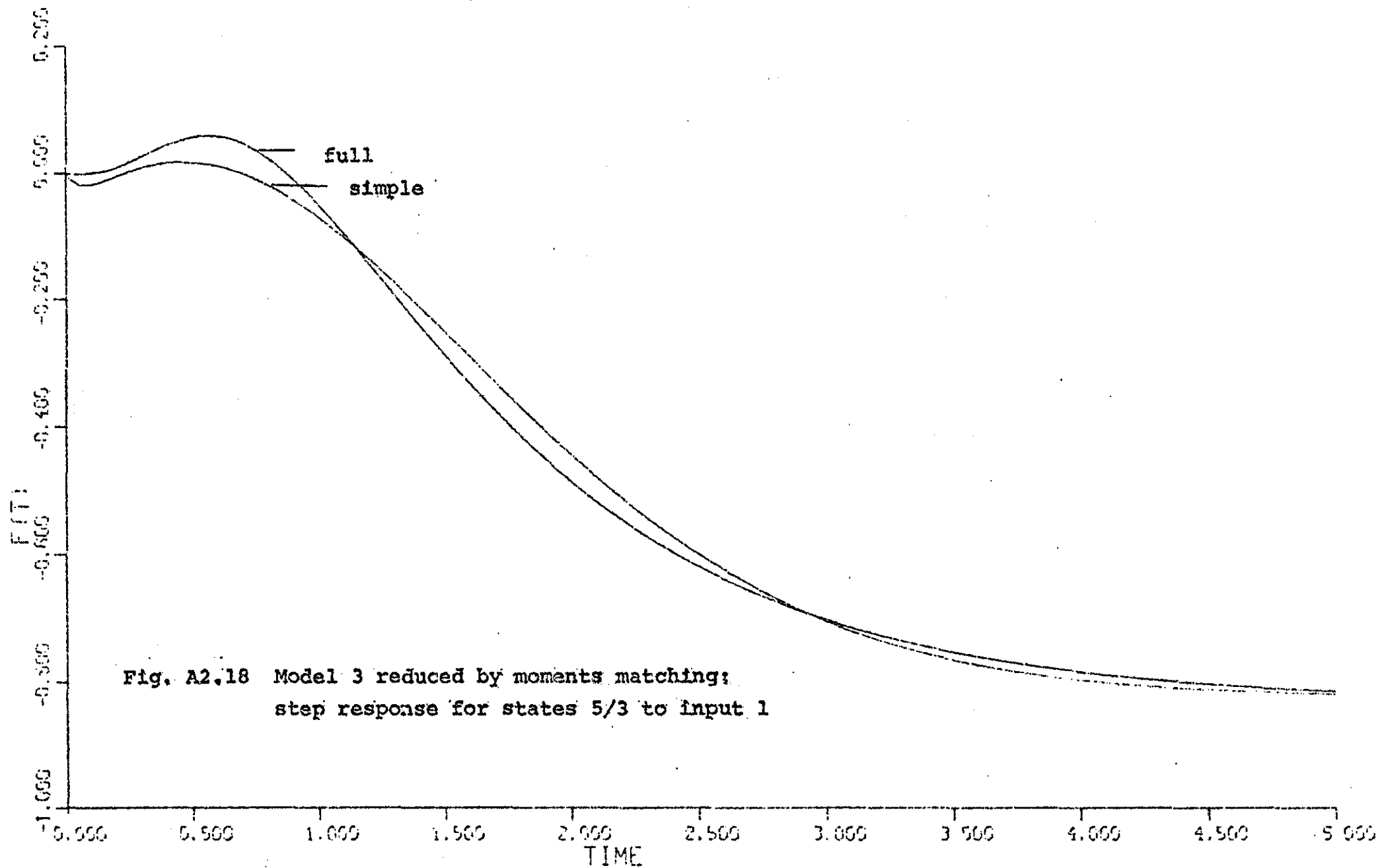
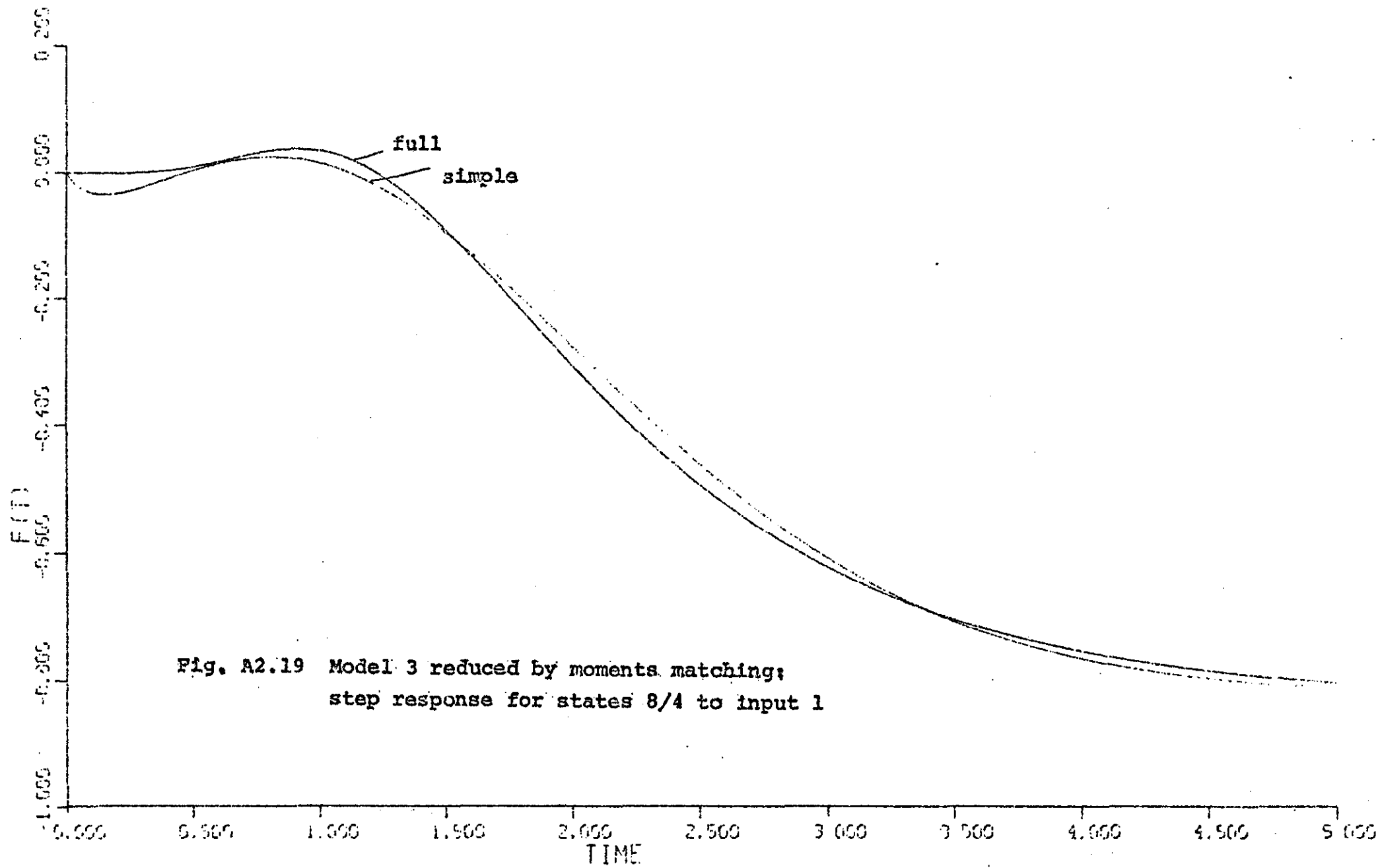
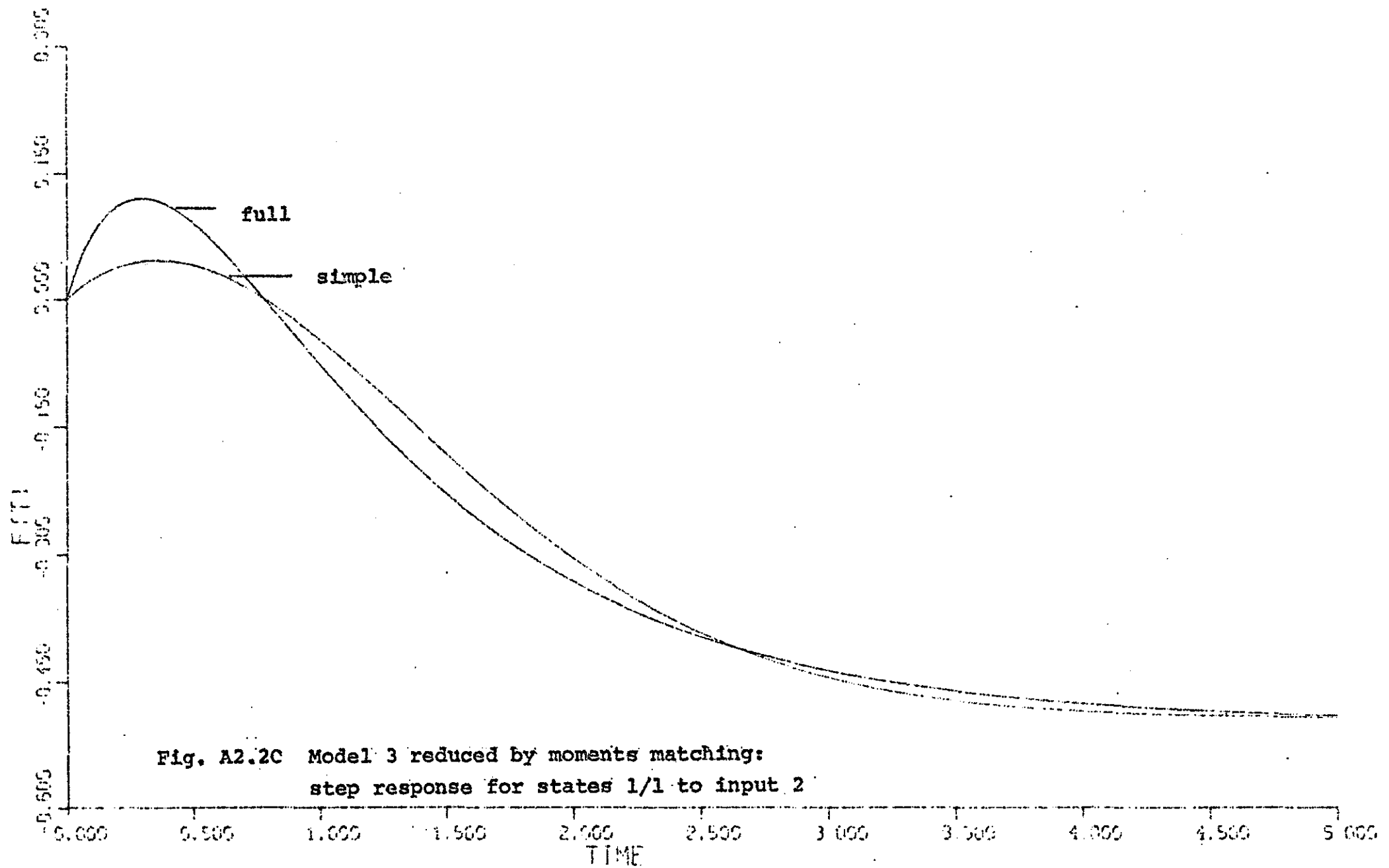


Fig. A2.17 Model 3 reduced by moments matching:
step response for states 3/2 to
input 1







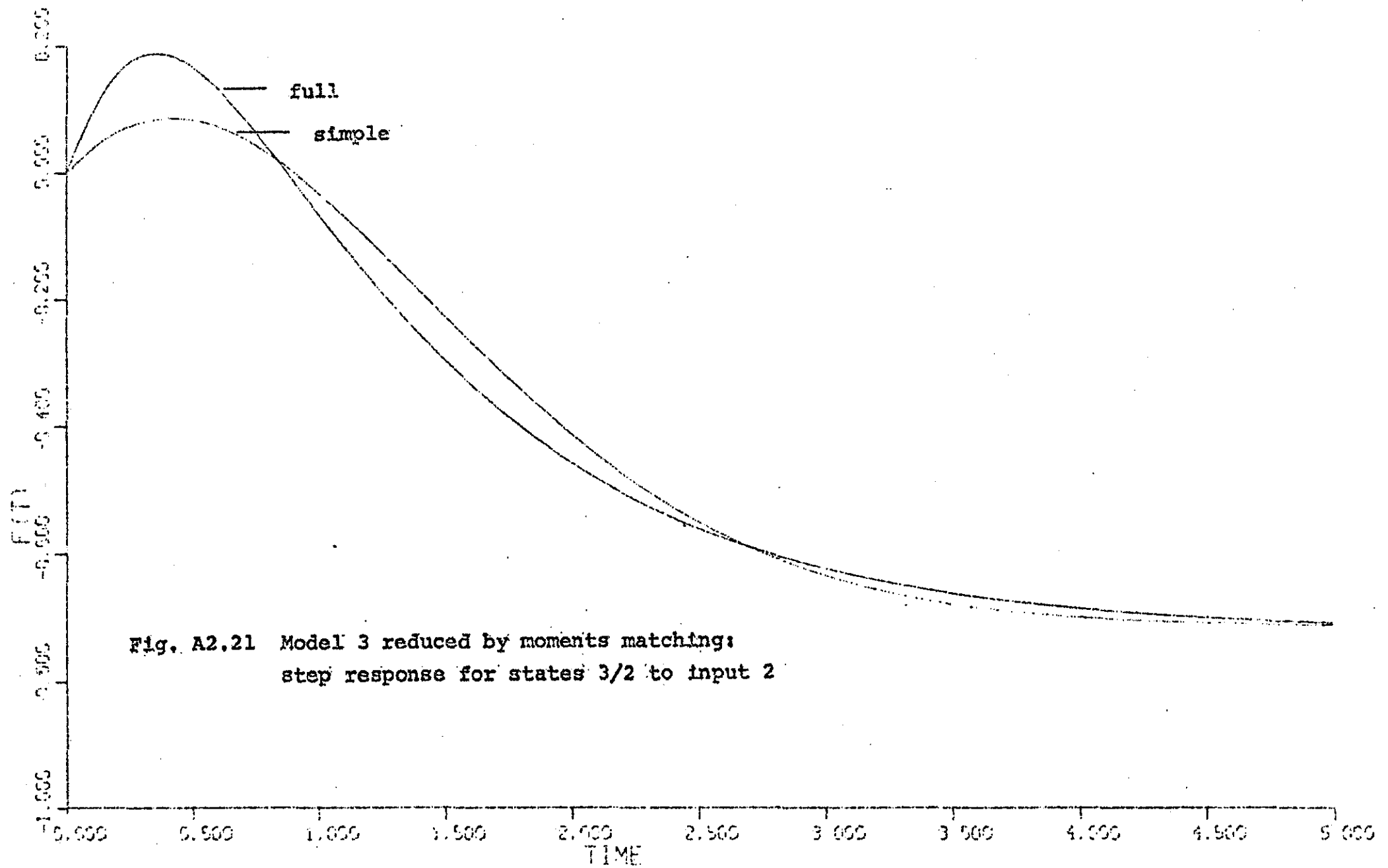


Fig. A2.21 Model 3 reduced by moments matching:
step response for states 3/2 to input 2

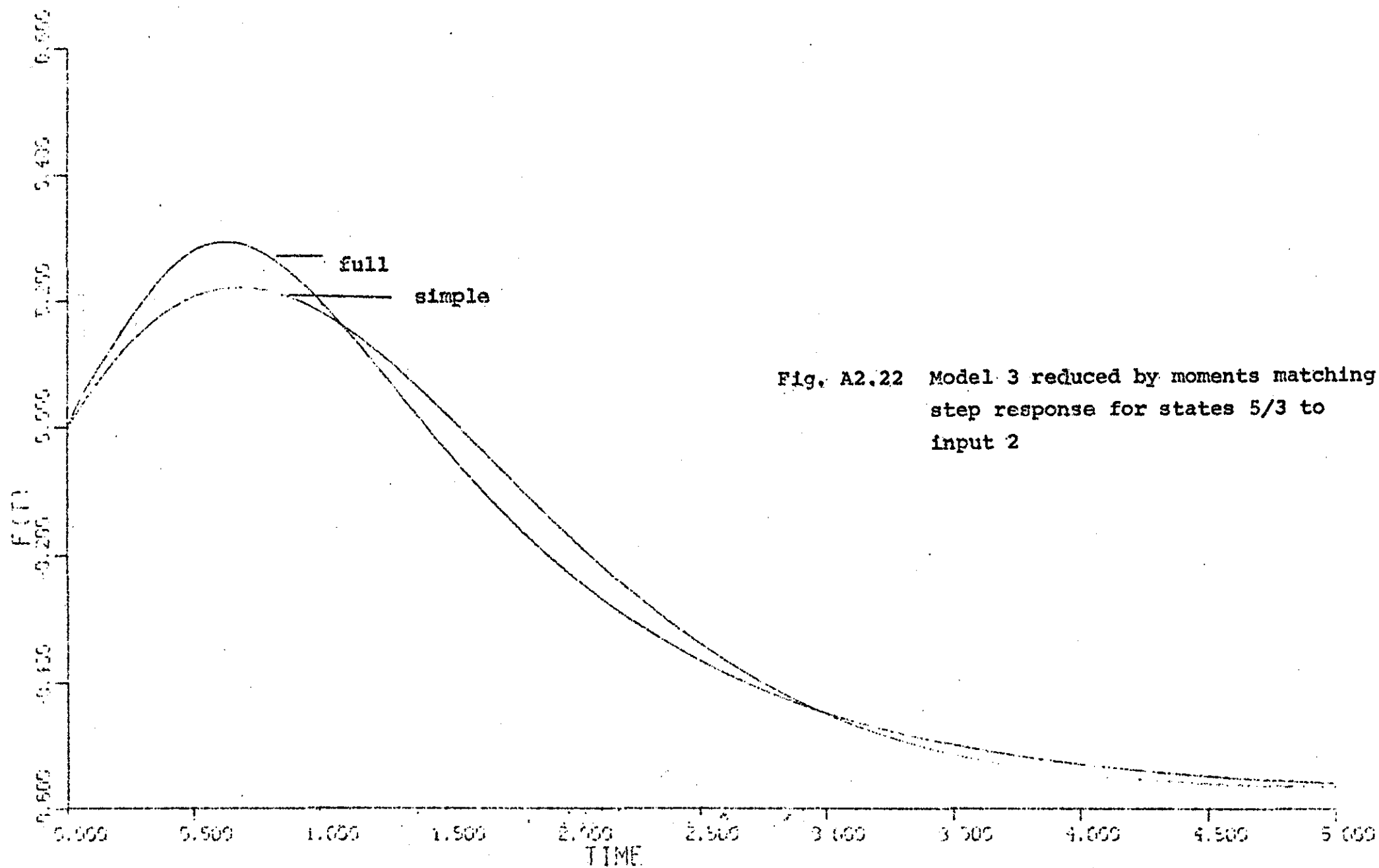


Fig. A2.22 Model 3 reduced by moments matching:
step response for states 5/3 to
input 2

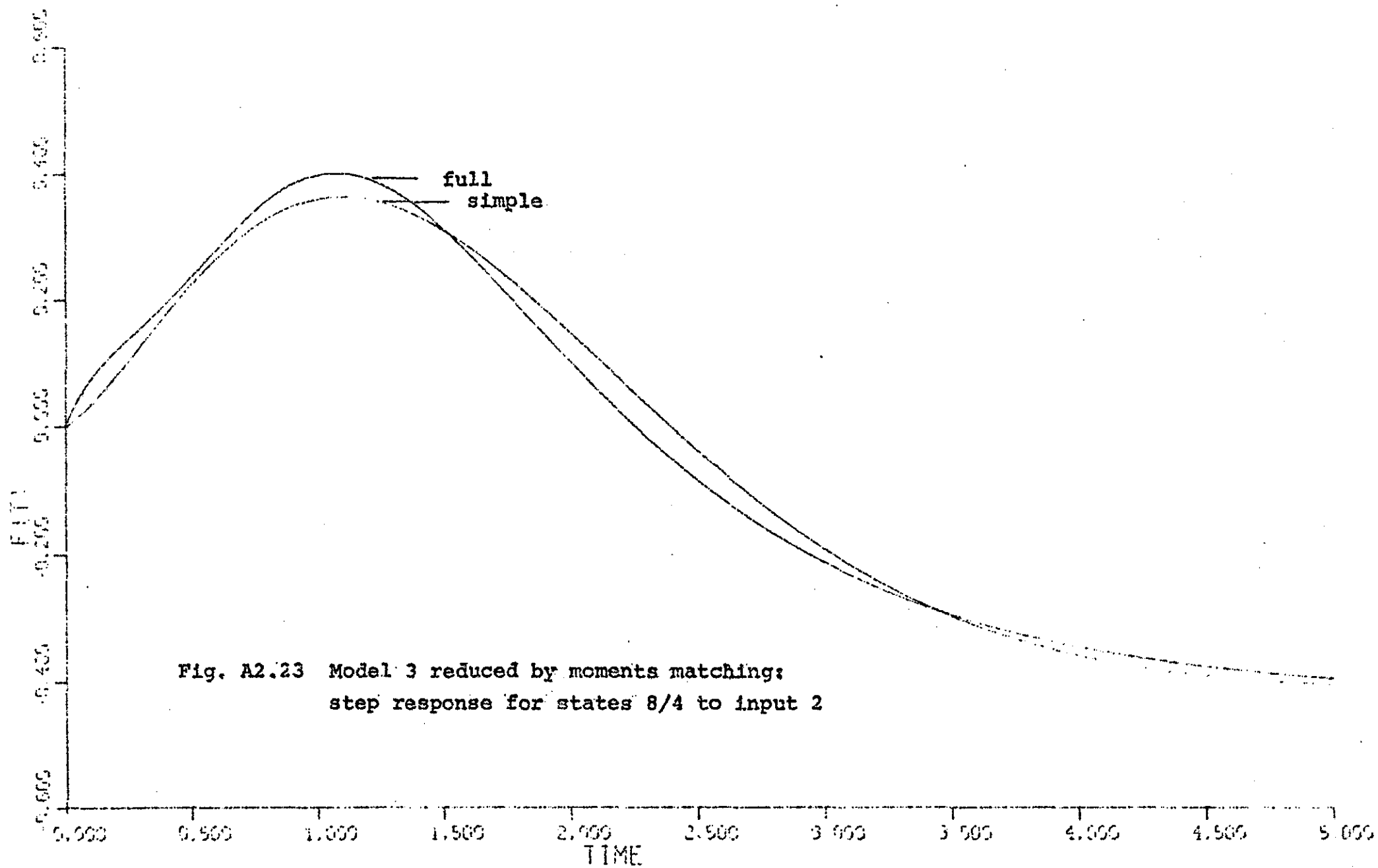


Fig. A2.23 Model 3 reduced by moments matching:
step response for states 8/4 to input 2

APPENDIX 2

A2.5 Model 4, a binary distillation column, reduced by matching the moments.

Contents

1. Full and reduced model data.
2. Step responses as detailed below.

Figure	Input	state x_i	reduced state x_i^*
A2.24	1	1	1
A2.25	1	3	2
A2.26	1	8	3
A2.27	1	11	4
A2.28	2	1	1
A2.29	2	3	2
A2.30	2	8	3
A2.31	2	11	4

A BINARY DISTILLATION COLUMN

FULL MODEL DATA

A MATRIX

ROW = 1	-0.140000E-01 0.000000E 00 0.000000E 00	0.430000E-02 0.000000E 00 0.000000E 00	0.000000E 00 0.000000E 00 0.000000E 00	0.000000E 00 0.000000E 00 0.000000E 00
ROW = 2	0.950000E-02 0.000000E 00 0.000000E 00	-0.138000E-01 0.000000E 00 0.000000E 00	0.460000E-02 0.000000E 00 -0.300000E-03	0.000000E 00 0.000000E 00 0.000000E 00
ROW = 3	0.000000E 00 0.000000E 00 0.000000E 00	0.950000E-02 0.000000E 00 0.000000E 00	-0.141000E-01 0.000000E 00 -0.500000E-03	0.630000E-02 0.000000E 00 0.000000E 00
ROW = 4	0.000000E 00 0.110000E-01 0.000000E 00	0.000000E 00 0.000000E 00 0.000000E 00	0.950000E-02 0.000000E 00 -0.800000E-03	-0.158000E-01 0.000000E 00 0.000000E 00
ROW = 5	0.000000E 00 -0.312000E-01 0.000000E 00	0.000000E 00 0.150000E-01 0.000000E 00	0.000000E 00 0.000000E 00 -0.800000E-03	0.950000E-02 0.000000E 00 0.000000E 00
ROW = 6	0.000000E 00 0.202000E-01 0.000000E 00	0.000000E 00 -0.352000E-01 0.000000E 00	0.000000E 00 0.220000E-01 -0.800000E-03	0.000000E 00 0.000000E 00 0.000000E 00
ROW = 7	0.000000E 00 0.000000E 00 0.000000E 00	0.000000E 00 0.202000E-01 0.000000E 00	0.000000E 00 -0.422000E-01 -0.800000E-03	0.000000E 00 0.280000E-01 0.000000E 00
ROW = 8	0.000000E 00 0.000000E 00 0.370000E-01	0.000000E 00 0.000000E 00 0.000000E 00	0.000000E 00 0.202000E-01 -0.600000E-03	0.000000E 00 -0.482000E-01 0.000000E 00
ROW = 9	0.000000E 00 0.000000E 00 -0.572000E-01	0.000000E 00 0.000000E 00 0.420000E-01	0.000000E 00 0.000000E 00 -0.300000E-03	0.000000E 00 0.202000E-01 0.000000E 00
ROW = 10	0.000000E 00 0.000000E 00 0.202000E-01	0.000000E 00 0.000000E 00 -0.483000E-01	0.000000E 00 0.000000E 00 0.000000E 00	0.000000E 00 0.000000E 00 0.000000E 00
ROW = 11	0.255000E-01 0.000000E 00 0.000000E 00	0.000000E 00 0.000000E 00 0.255000E-01	0.000000E 00 0.000000E 00 -0.185000E-01	0.000000E 00 0.000000E 00 0.000000E 00

SYSTEM EIGENVALUES

VARIABLE	REAL	IMAG
----------	------	------

1	-0.960340E-01	0.000000E 00
2	-0.505295E-01	0.000000E 00
3	-0.700745E-01	0.000000E 00
4	-0.247194E-01	0.000000E 00
5	-0.337243E-01	0.000000E 00
6	-0.823316E-02	0.000000E 00
7	-0.200682E-01	0.000000E 00
8	-0.489846E-03	0.000000E 00
9	-0.139995E-01	0.000000E 00
10	-0.325079E-02	0.000000E 00
11	-0.173768E-01	0.000000E 00

FORCING FUNCTION MATRIX B

ROW = 1	0.000000E 00	0.000000E 00
ROW = 2	-0.300000E-05	0.300000E-04
ROW = 3	-0.500000E-05	0.500000E-04
ROW = 4	-0.500000E-05	0.500000E-04
ROW = 5	-0.500000E-05	0.500000E-04
ROW = 6	-0.500000E-05	0.500000E-04
ROW = 7	-0.500000E-05	0.500000E-04
ROW = 8	-0.200000E-04	0.500000E-04
ROW = 9	-0.400000E-04	0.400000E-04
ROW = 10	-0.200000E-04	0.200000E-04
ROW = 11	0.460000E-03	0.460000E-03

VARIABLES OF INTEREST= 1 3 8 11

MATRIX OF MOMENTS FOR FORCING FUNCTION 1

ROW = 1	-0.574695E-02 -0.268102E 13	-0.129206E 02	-0.534743E 05	-0.328209E 09
ROW = 2	-0.187110E-01 -0.842360E 13	-0.407304E 02	-0.168093E 06	-0.103128E 10
ROW = 3	-0.436445E-01 -0.195232E 14	-0.947117E 02	-0.389839E 06	-0.239040E 10

ROW = 4	-0.687112E-01 -0.303097E 14	-0.147609E 03	-0.605634E 06	-0.371144E 10
ROW = 5	-0.605829E-01 -0.260902E 14	-0.127626E 03	-0.521677E 06	-0.319505E 10
ROW = 6	-0.821884E-01 -0.347806E 14	-0.170613E 03	-0.695711E 06	-0.425949E 10
ROW = 7	-0.756661E-01 -0.313015E 14	-0.153885E 03	-0.626275E 06	-0.383352E 10
ROW = 8	-0.545821E-01 -0.218370E 14	-0.107573E 03	-0.436992E 06	-0.267445E 10
ROW = 9	-0.292622E-01 -0.112396E 14	-0.554612E 02	-0.224952E 06	-0.137658E 10
ROW = 10	-0.126521E-01 -0.474880E 13	-0.234569E 02	-0.950506E 05	-0.581615E 09
ROW = 11	-0.496005E-03 -0.105196E 14	-0.501687E 02	-0.210147E 06	-0.128816E 10

ATRIX OF MOMENTS FOR FORCING FUNCTION 2

OW = 1	0.762414E-02 0.307406E 13	0.151469E 02	0.615120E 05	0.376483E 09
OW = 2	0.248228E-01 0.965834E 13	0.475425E 02	0.193227E 06	0.118284E 10
OW = 3	0.557639E-01 0.223845E 14	0.109905E 03	0.447717E 06	0.274132E 10
OW = 4	0.837728E-01 0.347511E 14	0.170249E 03	0.694927E 06	0.425572E 10
OW = 5	0.715961E-01 0.299128E 14	0.146418E 03	0.598105E 06	0.366315E 10
OW = 6	0.954440E-01 0.398760E 14	0.195185E 03	0.797291E 06	0.488323E 10
OW = 7	0.866859E-01 0.358870E 14	0.175726E 03	0.717530E 06	0.439471E 10
OW = 8	0.615672E-01 0.250359E 14	0.122669E 03	0.500577E 06	0.306588E 10
OW = 9	0.324124E-01 0.128861E 14	0.631838E 02	0.257654E 06	0.157802E 10
OW = 10	0.139696E-01 0.544443E 13	0.267139E 02	0.108862E 06	0.666722E 09
OW = 11	0.546292E-01 0.120611E 14	0.606530E 02	0.241397E 06	0.147708E 10

REDUCED SYSTEM

FORCING FUNCTION MATRIX B*

ROW = 1	-0.995775E-06	0.371608E-05
ROW = 2	-0.500885E-06	0.420928E-04
ROW = 3	-0.235026E-04	0.409633E-04
ROW = 4	0.447200E-03	0.477588E-03

A MATRIX

ROW = 1	-0.335520E-02	0.159182E-03	0.207708E-03	0.365755E-05
ROW = 2	-0.177024E-01	0.180244E-02	0.419014E-03	-0.612056E-03
ROW = 3	-0.259250E-01	0.701277E-02	-0.330329E-02	-0.567334E-03
ROW = 4	0.553710E-01	-0.698167E-02	0.811376E-02	-0.184876E-01

SYSTEM EIGENVALUES

VARIABLE	REAL	IMAG
1	-0.185537E-01	0.000000E 00
2	-0.487669E-03	0.000000E 00
3	-0.991276E-03	0.000000E 00
4	-0.331092E-02	0.000000E 00

REDUCED MODEL MOMENTS

MATRIX OF MOMENTS FOR FORCING FUNCTION 1

ROW = 1	-0.574695E-02 -0.268128E 13	-0.129234E 02	-0.533596E 05	-0.327523E 09
ROW = 2	-0.436445E-01 -0.195330E 14	-0.947365E 02	-0.388827E 06	-0.238557E 10
ROW = 3	-0.545821E-01 -0.218496E 14	-0.107602E 03	-0.435824E 06	-0.266905E 10
ROW = 4	-0.496005E-03 -0.105215E 14	-0.501805E 02	-0.209679E 06	-0.128546E 10

MATRIX OF MOMENTS FOR FORCING FUNCTION 2

ROW = 1	0.762414E-02 0.305954E 13	0.151531E 02	0.612539E 05	0.374252E 09
ROW = 2	0.557639E-01 0.222845E 14	0.109958E 03	0.445439E 06	0.272451E 10
ROW = 3	0.615672E-01 0.249251E 14	0.122730E 03	0.497948E 06	0.304689E 10
ROW = 4	0.546292E-01 0.120046E 14	0.606776E 02	0.240343E 06	0.146822E 10

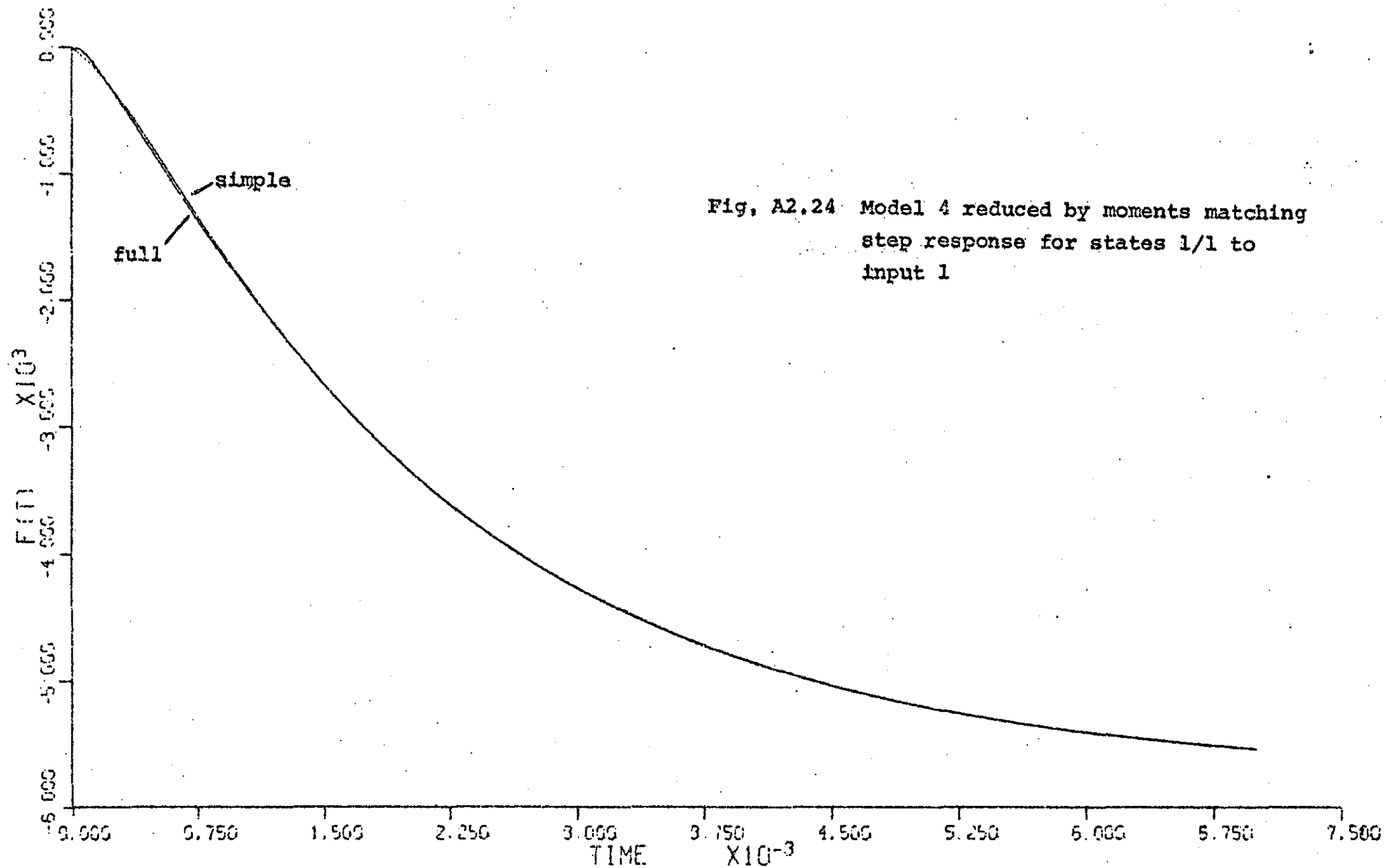


Fig. A2.24 Model 4 reduced by moments matching
step response for states 1/1 to
input 1

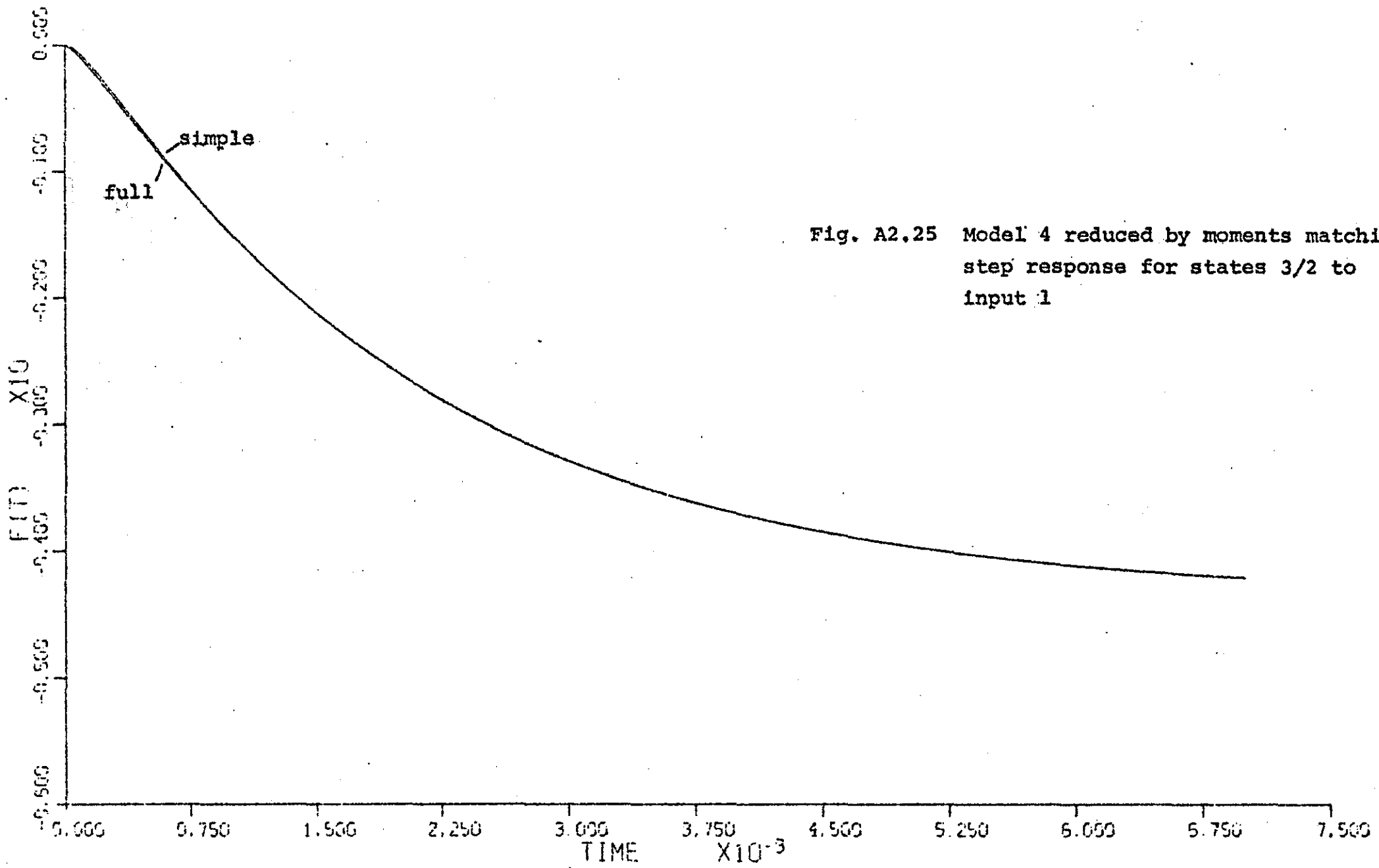


Fig. A2.25 Model 4 reduced by moments matching:
step response for states 3/2 to
input 1

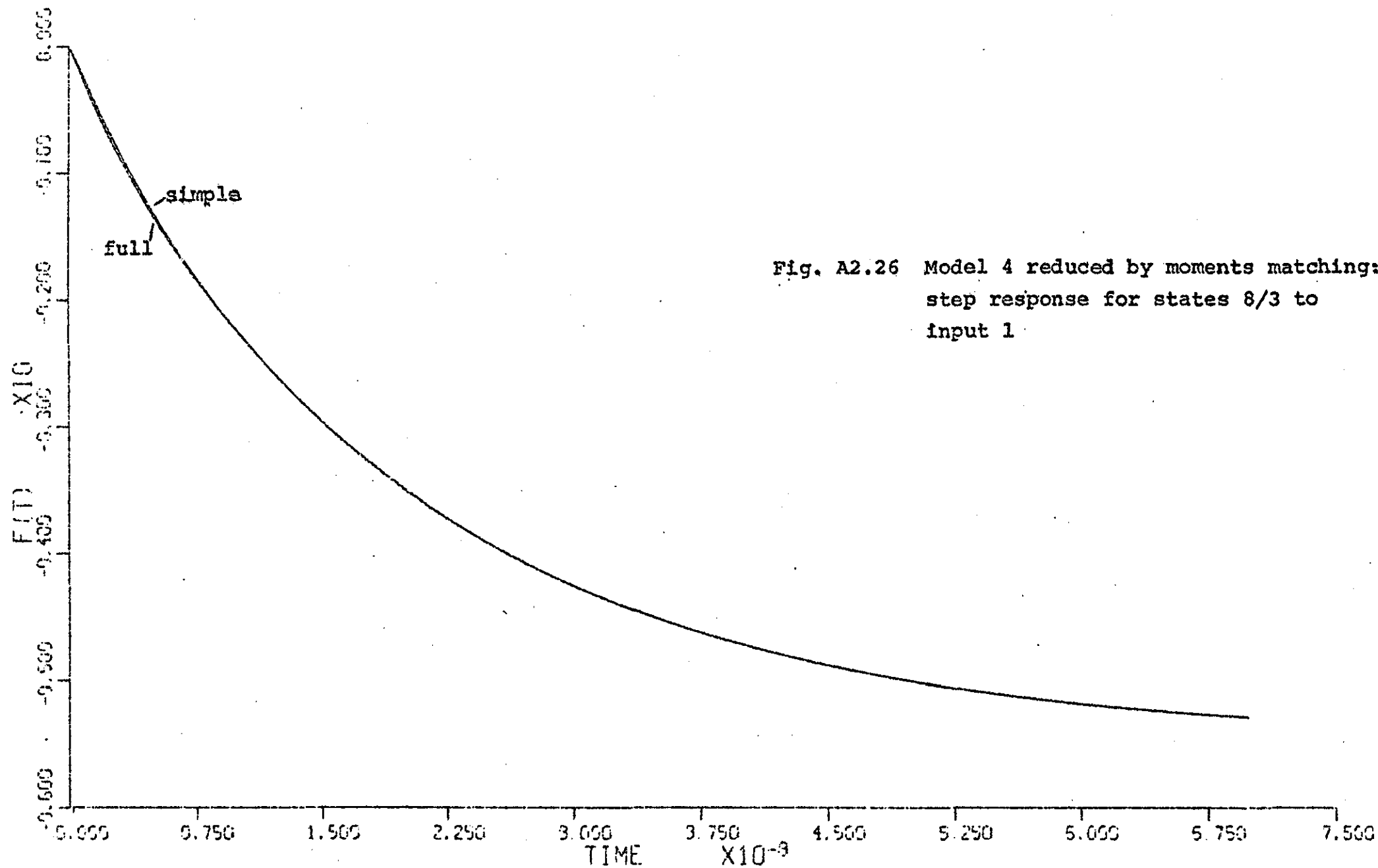
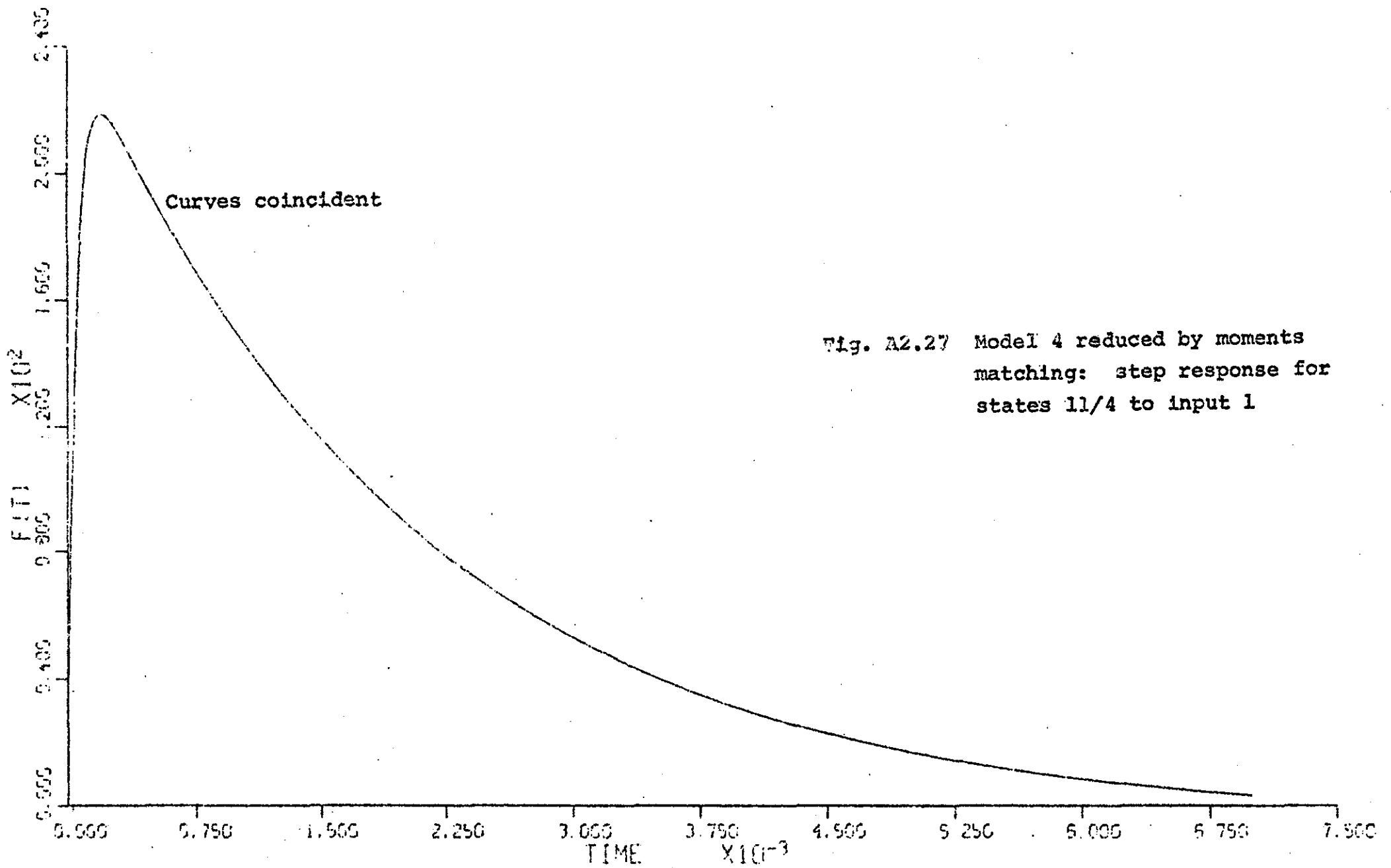


Fig. A2.26 Model 4 reduced by moments matching:
step response for states 8/3 to
input 1



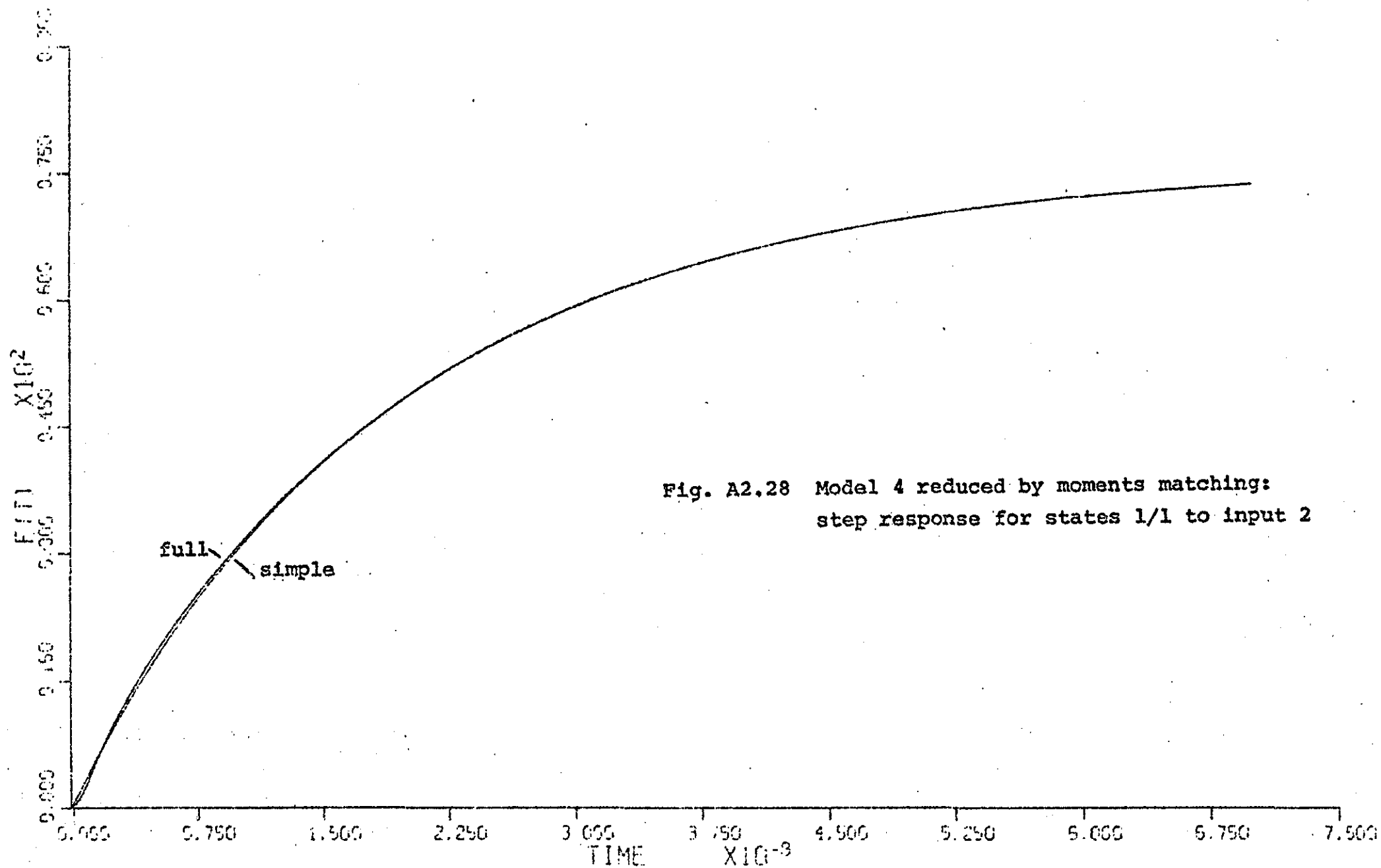
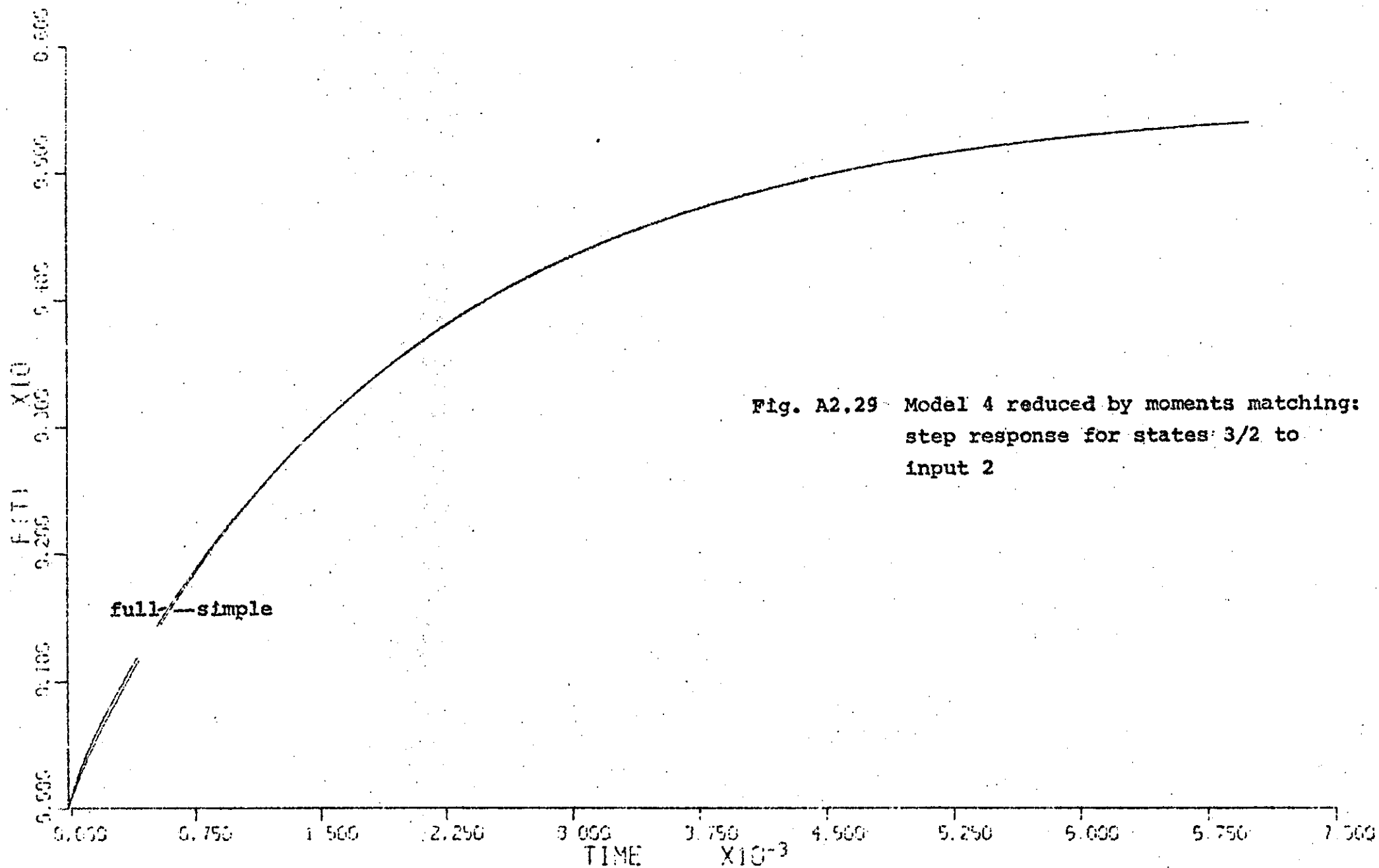


Fig. A2.28 Model 4 reduced by moments matching:
step response for states 1/1 to input 2



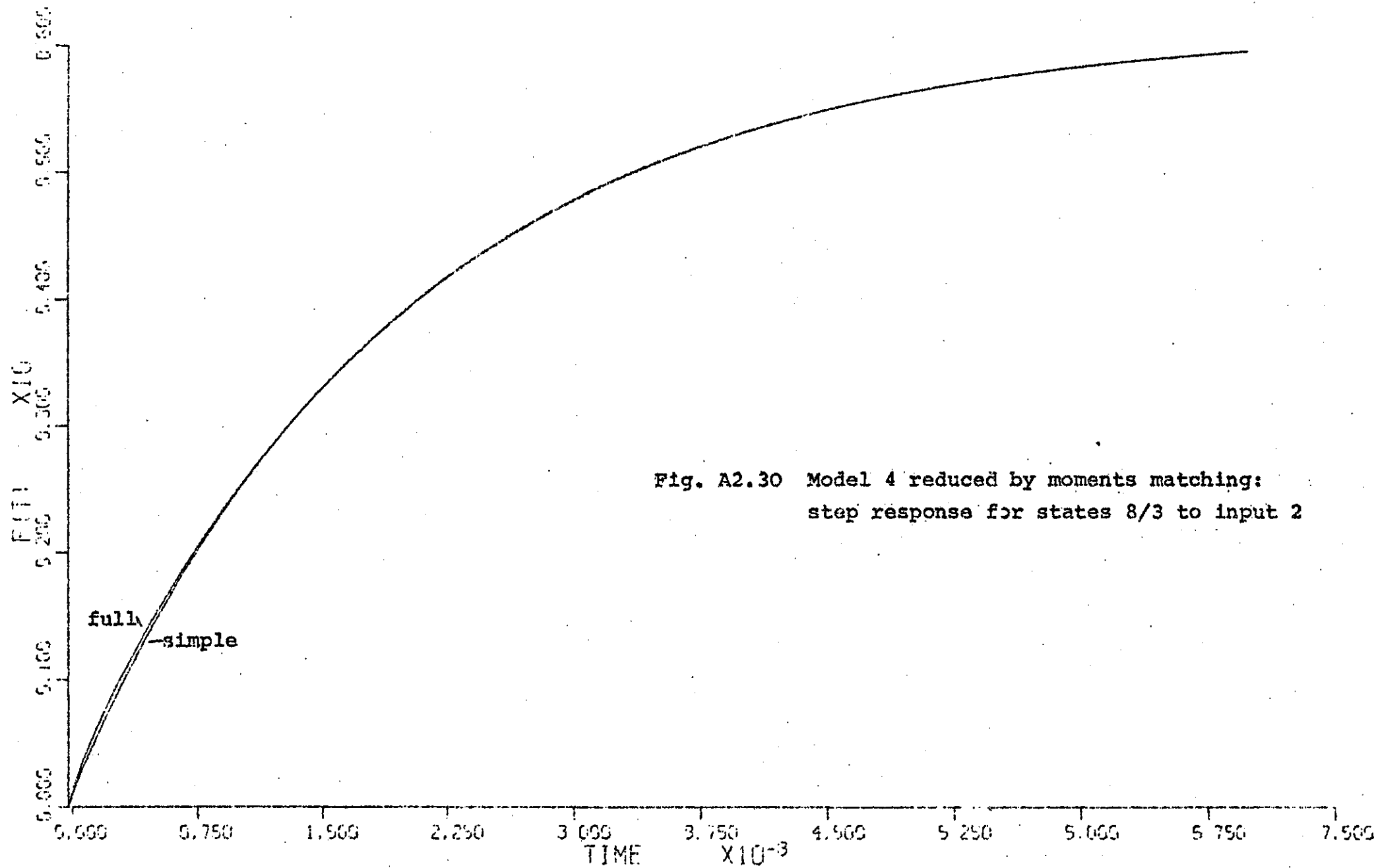


Fig. A2.30 Model 4 reduced by moments matching:
step response for states 8/3 to input 2

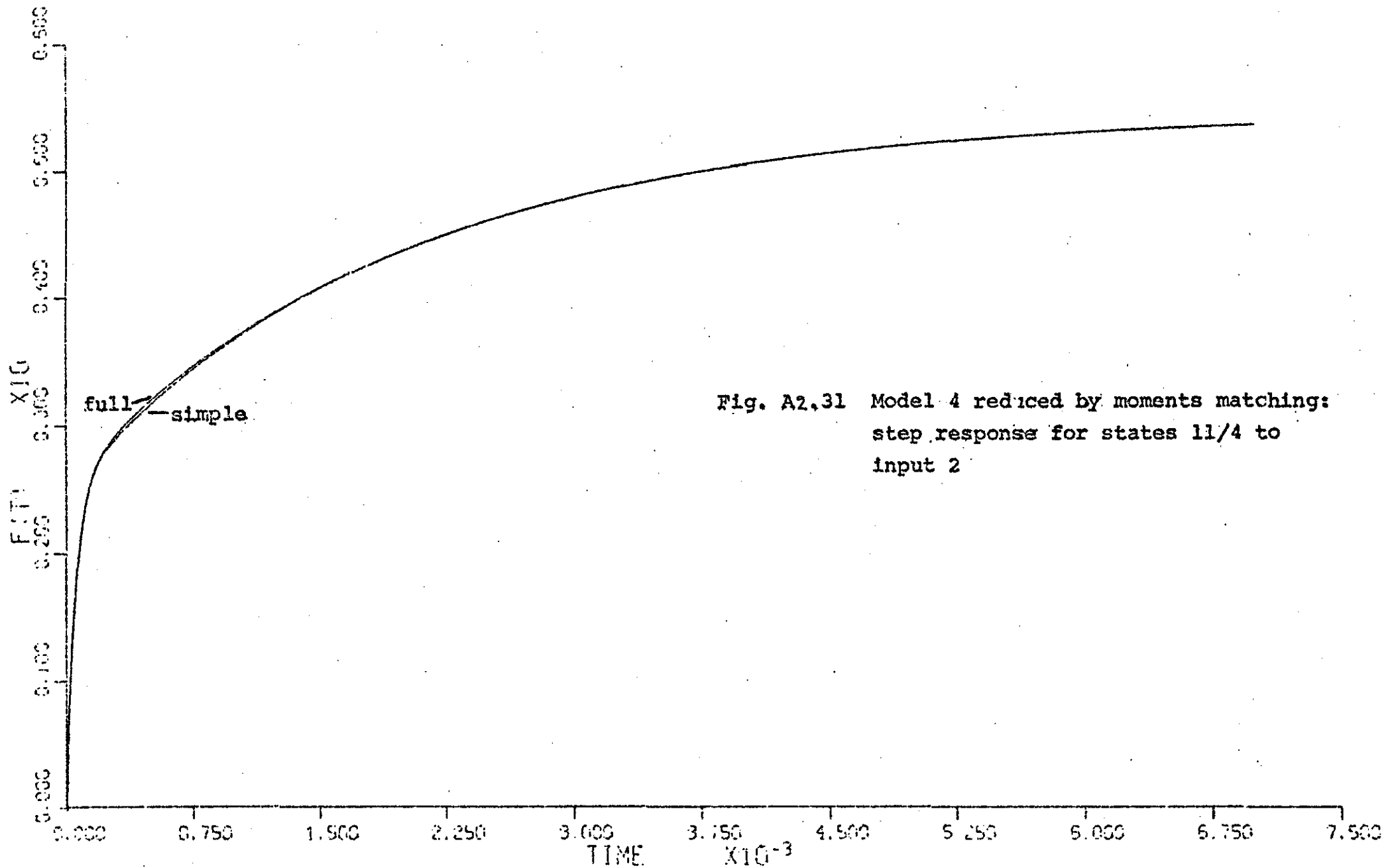


Fig. A2.31 Model 4 reduced by moments matching:
step response for states 11/4 to
input 2

APPENDIX 3

A3.1 A listing of the program for the
reduction of state variable models by
matching the frequency response.

```

MASTER FREQUENCY RESPONSE REDUCTION,TYPE 2.
INTEGER P,DIXSTART
DIMENSION WORK3(400),WORK4(400),WORK5(800)
DIMENSION W(50),      R(400),IR(20),WORK1(400),WORK2(400)
DIMENSION HOLD(20),IQ(20),IPIV(20),WORK6(400),WORK7(800)
COMMON /TRESP/A(1681),BU(41),IMP
COMMON RO(400)
DEFINE FILE 3(600,100,L,NIL)
DEFINE FILE4(600,100,L,NIL)

```

```

*****
THIS PROGRAM REDUCES THE N TH. ORDER MODEL

$$X' = AX + BU$$

TO AN EQUIVALENT M TH. ORDER MODEL BY MATCHING THE FREQUENCY
RESPONSES OF ALL INPUTS AND SELECTED OUTPUTS OF THE FULL AND
REDUCED MODELS.

```

```

OUTPUT:
FULL AND REDUCED SYSTEMS ARE WRITTEN TOGETHER WITH THEIR
RESPECTIVE TIME AND FREQUENCY RESPONSES AND EIGENANALYSES.
INFORMATION IS GIVEN ON THE SELECTED VARIABLES AND THE QUALITY
OF FIT OBTAINED IN THE LEAST SQUARES SOLUTION.

```

```

DATA REQUIRED:
M REDUCED MODEL ORDER.
N FULL MODEL ORDER.
ID NUMBER OF INPUTS.
P NUMBER OF FREQUENCIES MATCHED.
W FREQUENCIES.
A PLANT MATRIX.
DT INTERVAL FOR THE TIME RESPONSE.
TMAX MAXIMUM TIME FOR TIME RESPONSE.
G INPUT MATRIX.
IP SELECTED OUTPUTS.

```

```

PROGRAM WRITTEN BY:
      M.J. BOSLEY,
      DEPT. CHEM. ENG.
      LOUGHBOROUGH UNIVERSITY OF TECHNOLOGY,
      ENGLAND.

```

```

*****
READ INPUT PARAMETERS, FREQUENCIES, PLANT MATRIX, AND TIME
RESPONSE PARAMETERS.

```

```

DO 9900 KBOZ=1,2
READ(1,1)M,N,IP,P
READ(1,2)(W(K),K=1,P)
READ(1,2)((A(I+N*(J-1)),J=1,N),I=1,N)
READ(1,2)DT,TMAX

```

```

COMPUTE PARAMETERS FOR USE IN PROGRAM.

```

```

EPS=.1E-10
DIXSTART=0
IDP=IP+P

```

```

MIPP=M*IPP
MIP=M*IP
M2=M*M
M22=2*M2
IF=1
IPFAD=5
IMP=1

```

SET UP PARAMETERS FOR USE IN COMPUTING TIME RESPONSE.

```
WRITE(2,9998)
```

```
53 CONTINUE
```

```

IV=0
IRUNG=0
N1=N+1
N2=N*N
N3=N2+7*N
NIP=N*IP
N2IP=2*NIP
KNIP=NIP
CALL FMOVE(A(1),WORK1(1),N2)

```

COMPUTE THE EIGENANALYSIS.

```

500 CALL MATINGR(IRUNG,N,DT,TMAX,N1,N2,N3,WORK2,WORK3,WORK5,A,IQ,IR,
1      WORK7,IR,WORK7,WORK7,BU,WORK4,IX,IREAD)
GO TO(501,502,506,502),IE

```

READ INPUT MATRIX.

```

501 READ(1,2)((B(I+N*(J-1)),J=1,IP),I=1,N)
WRITE(2,4)
CALL WRITEMAT(N,IP,NIP,B)
IC=2
506 CONTINUE
IF=4

```

COMPUTE THE TIME RESPONSE FOR EACH INPUT IN TURN.

```

DO 502 K=1,IP
DO 505 I=1,N
505 BU(I)=B(I+N*(K-1))
502 CONTINUE
CALL FMOVE(WORK1(1),A(1),N2)

```

COMPUTE THE SYSTEM FREQUENCY RESPONSE AND WRITE IT OUT.

```

CALL FREQRESP2(N,IP,NIP,N2,P,DIXSTART,A,WORK1,B,WORK2,WORK6,WORK3,
1      W,WORK4,N1,HOLD)
IF(DIXSTART.GT.0)GO TO 9999

```

READ THE SELECTED VARIABLES.

```
READ(1,1)(IR(I),I=1,M)
```

SET UP ARRAYS FOR THE LEAST SQUARES SOLUTION.

```

CALL SETUP(M,IP,MIPP,P,N1,WORK4,WORK1,WORK3,W,IR,N)
WRITE(2,300)
WRITE(2,301)(IR(I),I=1,M)
DO 120 I=1,MIPP
120 WORK4(I)=WORK3(I)

```

```

C SOLVE THE LEAST SQUARES PROBLEM ONE COLUMN AT A TIME
C NORMALISING W.R.T. THE R.H.S. VARIABLE TO EFFECT EQUAL WEIGHTING
C TO ALL INPUTS AND VARIABLES. THE EUCLID NORMS BEFORE AND
C AFTER SOLUTION ARE WRITTEN OUT TO ESTIMATE THE QUALITY OF FIT.
C
DO 100 K=1,M
EUC=0
DO 110 I=1,IPP
J=I+IPP*(K-1)
IF(WORK1(J).EQ.0)GO TO 102
QHELD=WORK1(J)
WORK1(J)=1
DO 101 L=1,M
J1=I+IPP*(L-1)
101 WORK3(J1)=WORK4(J1)/QHELD
GO TO 110
102 WORK1(J)=0
DO 350 L=1,M
J1=I+IPP*(L-1)
350 WORK3(J1)=WORK4(J1)
110 EUC=EUC+WORK1(J)*WORK1(I)
EUC=SQRT(EUC)
CALL LISQ(IPP,M,1,MIPP,IPP,M,M22,WORK3,WORK1,HOLD,IPIV,EPS,IER,
WORK5)
IF(IER.EQ.0)GO TO 205
WRITE(2,206)IER
WRITE(2,2021)(IPIV(I),I=IER+1,M)
STOP
205 CONTINUE
100 CALL FMOVE(HOLD(1),A(1+M*(K-1)),M)
310 CONTINUE
CALL TRANMAT(A,WORK1,M,M)
CALL FMOVE(WORK1(1),A(1),M2)
C
C WRITE OUT THE REDUCED SYSTEM AND SET UP PARAMETERS TO ENABLE
C THE TIME AND FREQUENCY RESPONSES TO BE COMPUTED,AS FOR
C THE FULL SYSTEM.
C
WRITE(2,5)
DO 200 I=1,M
I1=IR(I)
DO 200 J=1,IP
200 WORK1(I+M*(J-1))=R0(I1+N*(J-1))
CALL FPMUNT(M,IP,M,A(1),WORK1(1),R(1),0,NRR)
CALL FPMASUS(B(1),DUM,B(1),MIP,-1,NRR)
WRITE(2,4)
CALL WRITEMAT(M,IP,MIP,B)
DIYSTART=IPP
N=M
IF=3
GO TO 53
4000 CONTINUE
STOP
1 FORMAT(20I0)
2 FORMAT(40F0.0)
3 FORMAT(18H1FULL ORDER SYSTEM,///, 9H MATRIX A)
4 FORMAT(///19H MATRIX B)
5 FORMAT(///19H REDUCED MODEL DATA)
10 FORMAT(///14H LEAST SQUARES/)
50 FORMAT(///19H SYSTEM EIGENVALUES/)
52 FORMAT(2(5X,E16.6))
103 FORMAT(///18H RHS EUCLID NORM =,E16.6)
104 FORMAT(///22H LEAST SQUARES ERROR =,E16.6)

```

```

206 FORMAT(20H STOP AT LEASE SQUARE,RANK =,I3)
300 FORMAT(////22H VARIABLES OF INTEREST)
301 FORMAT(19I6)
350 FORMAT(////24H NUMBER OF FREQUENCIES =,I3)
504 FORMAT(////40H TIME RESPONSE FOR FORCING FUNCTION      ,I3//)
9908 FORMAT(16H FULL MODEL DATA)
2021 FORMAT(18H USELESS COLUMNS =,20I3)
END

```

```

SUBROUTINE FREQRESP2(N,IP,NIP,N2,P,DIXSTART,A,A2,B,AB,STORE,RH,W,
1      REINT,N1,HOLD)
  INTEGER P,DIXSTART
  DIMENSION A(N2),A2(N2),B(NIP),AB(NIP),STORE(N2),RH(N2),W(P)
  DIMENSION HOLD(N)
  DIMENSION REINT(N1)
  COMMON R0(400)

```

THIS SEGMENT COMPUTES THE REAL AND IMAGINARY PARTS OF THE MODEL
 $X' = AX + BU$
 FOR ALL INPUTS AND OUTPUTS AND WRITES IT TO DISC FILES.

```

CALL FMOVE(A(1),STORE(1),N2)
CALL FMOVE(B(1),R0(1),NIP)
M7=50
CALL F4SOLVE(STORE,R0,N,N2,NIP,1,0,10,IT,REINT)
CALL FPMASUS(R0(1),DUM,R0(1),NIP,-1,NRR)
CALL FPMUMT(N,N,N,A(1),A(1),A2(1),0,NRR)
CALL FPMUMT(N,IP,N,A(1),B(1),AB(1),0,NRR)
CALL FPMASUS(AB(1),DUM,AB(1),NIP,-1,NRR)
CALL FMOVE(A2(1),STORE(1),N2)
M7=100
DO 7 K=1,P
DO 2 J=1,N
2 A2(J+N*(I-1))=A2(I+N*(I-1))+W(K)*W(K)
CALL FMOVE(AB(1),RH(1),NIP)
IN=1
IC=3
6 M7=M7+1
CALL F4SOLVE(A2,RH,N,N2,NIP,IN,0,10,IT,REINT)
IC(IT)3.6.4
4 DO 5 J=1,IP
L=1+N*(J-1)
CALL FMOVE(RH(L),HOLD(1),N)
L=DIXSTART+J+IP*(K-1)
5 WRITE(10,L)HOLD
IF(IC EQ.4)GO TO 7
Q=-W(K)
CALL FMOVE(B(1),RH(1),NIP)
CALL FPMASUS(Q,DUM,RH(1),NIP,0,NRR)
IN=2
IC=4
GO TO 6
7 CALL FMOVE(STORE(1),A2(1),N2)
RETURN
3 WRITE(2,1)M7
STOP
1 FORMAT(//5H M7 =,I3)
100 FORMAT(4E16.6)
101 FORMAT(14)

```



```
200 FORMAT(////28H REAL PART AT ZERO FREQUENCY)
END
```

```
SUBROUTINE SETUP(M,IP,MIPP,P,N1,HOLD,Q,S,W,IR,N)
INTEGER P
DIMENSION HOLD(MIPP),Q(MIPP),S(MIPP),W(P),IR(N)
COMMON RQ(400)
```

```
THIS SEGMENT READS BACK FROM DISC THE REAL AND IMAGINARY
PARTS FOR EACH VARIABLE AND INPUT AND SETS UP ARRAYS PRIOR
TO SOLVING THE LEAST SQUARES PROBLEM.
```

```
IPD=P+IP
DO 4 K=1,P
DO 4 J1=1,IP
J=J1+IP*(K-1)
READ(3,J)HOLD
DO 3 I=1,M
3 S(I+M*(J-1))=HOLD(IR(I))-RQ(IR(I)+N*(J1-1))
READ(4,J)HOLD
DO 4 I=1,M
4 Q(I+M*(J-1))=-HOLD(IR(I))+W(K)
CALL TRANMAT(Q,HOLD,M,IPP)
CALL FMOVE(HOLD(1),Q(1),MIPP)
CALL TRANMAT(S,HOLD,M,IPP)
CALL FMOVE(HOLD(1),S(1),MIPP)
RETURN
END
```

```
SUBROUTINE WRITEFRFQ(N,IP,P,N1,N2,N2IP,HOLD,WORK,W,DIXSTART,WORK7)
```

```
INTEGER DIXSTART
```

```
INTEGER P
```

```
DIMENSION WORK7(N2IP)
```

```
DIMENSION HOLD(N1),WORK(N2IP),W(P)
```

```
DIMENSION TEXT1(4),TEXT2(4)
```

```
DATA TEXT1(1),TEXT2(1)/32H
```

```
REAL
```

```
IMAG,
```

```
1 32H
```

```
GAIN
```

```
LAG/
```

```
THIS SEGMENT READS BACK FROM DISC THE REAL AND IMAGINARY PARTS
FOR EACH VARIABLE AND INPUT, AND FROM IT COMPUTES THE
EQUIVALENT FREQUENCY RESPONSE. THE WHOLE OF THIS DATA
IS THEN WRITTEN OUT.
```

```
WRITE(2,1)
```

```
IX=IP/3
```

```
IX=IP-3+IX
```

```
IX=IX
```

```
IF(IX.GT.0)IZ=IX+1
```

```
DO 101 K=1,P
```

```
J1=0
```

```
DO 12 J1=1,IP
```

```
L=J1+IP*(K-1)+DIXSTART
```

```
READ(3,1)HOLD
```

```
J1=J1+1
```

```
DO 11 I=1,N
```

```

11 WORK(I+N*(JL-1))=HOLD(I)
   READ(4,1)HOLD
   JL=JL+1
   DO 12 I=1,N
12 WORK(I+N*(JL-1))=HOLD(I)
   DO 104 J=1,JP
   JB=2*J-1
   DO 104 I=1,N
   KP=I+N*(JR-1)
   KI=I+N*JR
   ZR=WORK(KP)
   ZI=WORK(KI)
   AR=SQRT(ZR*ZR+ZI*ZI)
   IF(ZR.NE.0.OR.ZI.NE.0)GO TO 200
   PHASE=0
   GO TO 201
200 CONTINUE
   PHASE=ATAN2(ABS(ZI),ABS(ZR))*180/3.14159
   IF(ZI.LT.0.AND.ZR.GT.0)PHASE=PHASE
   IF(ZI.LT.0.AND.ZR.LT.0)PHASE=180-PHASE
   IF(ZI.GT.0.AND.ZR.LT.0)PHASE=180+PHASE
   IF(ZI.GT.0.AND.ZR.GT.0)PHASE=360-PHASE
201 CONTINUE
   WORK7(KP)=AR
104 WORK7(KI)=PHASE
   WRITE(2,3)W(K)
   DO 8 J1=1,12
   IST=3*J1-2
   IEND=IST+2
   IF(J1.EQ.12)IEND=IST+IV-1
   WRITE(2,6)(I,I=IST,IEND)
   WRITE(2,7)
   WRITE(2,21)(TEXT1,I=IST,IEND)
   DO 8 I=1,N
8 WRITE(2,9)I,(WORK(I+N*(J-1)),J=2*IST-1,2*IEND)
   DO 101 J1=1,12
   IST=3*J1-2
   IEND=IST+2
   IF(J1.EQ.12)IEND=IST+IV-1
   WRITE(2,6)(I,I=IST,IEND)
   WRITE(2,7)
   WRITE(2,21)(TEXT2,I=IST,IEND)
   DO 101 I=1,N
101 WRITE(2,9)I,(WORK7(I+N*(J-1)),J=2*IST-1,2*IEND)
   RETURN
1 FORMAT(19H1FREQUENCY RESPONSE)
3 FORMAT(////12H FREQUENCY =,E16.6)
6 FORMAT(//17H FORCING FUNCTION,4X,3(12,29X))
7 FORMAT(//6H STATE)
9 FORMAT(2X,12,4X,6(4X,E12.6))
100 FORMAT(14)
21 FORMAT(15A8)
END

```

SUBROUTINE LLSO(M,N,L,MN,ML,NL,N2L,A,B,X,IPIV,EPS,IER,AUX)
 DIMENSION A(MN),B(ML),X(NL),IPIV(N),AUX(N2L)

C
 C THIS SEGMENT SOLVES THE LINEAR LEAST SQUARES PROBLEM, I.E.
 C MINIMIZES THE EUCLID NORM OF B-AX WHERE A IS THE (M,N) MATRIX

M GE, N AND R IS THE (N,L) MATRIX. THIS PROGRAM IS GIVEN IN DETAIL
IN THE IBM 360 SCIENTIFIC SUBS. MANUAL.

REFERENCE:

G GOLUP. NUMERISCHE MATHEMATIK, VOL 7, ISS. 3 (1965), 206-216.

```

      IC(M-N)30,1,1
1  PIV=0
   IFND=0
   DO 4 K=1,N
     IPIV(K)=K
     H=0
     IST=IFND+1
     IFND=IFND+M
     DO 2 I=IST,IEND
2  H=H+A(I)+A(I)
     AUX(K)=H
     IC(H-PIV)4,4,3
3  PIV=H
     KPIV=K
4  CONTINUE
     IC(PIV)31,31,5
5  SIG=SQRT(PIV)
     TOL=SIG*ABS(EPS)
     LM=I-M
     IST=-M
     DO 21 K=1,N
       IST=IST+M+1
       IFND=IST+M-K
       I=KPIV-K
       IC(I)8,8,6
6  H=AUX(K)
     AUX(K)=AUX(KPIV)
     AUX(KPIV)=H
     ID=I+M
     DO 7 I=IST,IEND
       J=I+ID
       H=A(I)
       A(I)=A(J)
7  A(J)=H
8  IC(K-1)11,11,9
9  SIG=0
     DO 10 I=IST,IEND
10  SIG=SIG+A(I)+A(I)
     SIG=SQRT(SIG)
     IC(SIG-TOL)32,32,11
11  H=A(IST)
     IC(H)12,13,13
12  SIG=-SIG
13  IPIV(KPIV)=IPIV(K)
     IPIV(K)=KPIV
     BETA=H+SIG
     A(IST)=BETA
     BETA=1/(SIG+BETA)
     J=N+K
     AUX(J)=-SIG
     IC(K-N)14,19,19
14  PIV=0
     ID=0
     JST=K+1
     KPIV=JST
     DO 18 J=JST,N
       ID=ID+M
       H=0

```

```

DO 15 I=IST,IEND
  I=I+10
15 H=H+A(I)*A(I)
  H=BETA+H
DO 16 I=IST,IEND
  I=I+10
16 A(I)=A(I)-A(I)+H
  I=IST+10
  H=AUX(I)-A(I)+A(I)
  AUX(J)=H
  IC=(H-PIV)18,18,17
17 PIV=H
  KPIV=J
18 CONTINUE
19 DO 21 J=K,LM,M
  H=0
  IFND=J+M-K
  I=IST
  DO 20 I=J,IFND
  H=H+A(I)*B(I)
20 I=I+1
  H=BETA+H
  I=IST
  DO 21 I=J,IEND
  B(I)=B(I)-A(I)+H
21 I=I+1
  IC=0
  I=N
  LN=1+N
  PIV=1/AUX(2+N)
  DO 22 K=N,LN,N
  X(K)=PIV+B(I)
22 I=I+M
  IC=(N-1)26,26,23
23 JCT=(N-1)*M+N
  DO 25 J=2,N
  JCT=JCT-M-1
  K=N+M+1-J
  PIV=1/AUX(K)
  KCT=K-N
  ID=IDIV(KST)-KST
  IST=2-J
  DO 25 K=1,L
  H=B(KCT)
  IST=IST+N
  IFND=IST+J-2
  I=IST
  DO 24 I=IST,IEND
  I=I+M
24 H=H-A(I)*X(I)
  I=IST-1
  I=I+10
  X(I)=V(I)
  X(I)=PIV+H
25 KCT=KCT+M
26 ICT=N+1
  IFND=0
  DO 29 J=1,L
  IFND=IFND+M
  H=0
  IC=(M-N)29,29,27
27 DO 28 I=IST,IEND
28 H=H+B(I)*B(I)

```

```

      ICT=IST+M
20  AUX(J)=H
      RETURN
30  ICR=-2
      RETURN
31  ICR=-1
      RETURN
32  ICR=K-1
      RETURN
      END
      SUBROUTINE MATINGR(IRUNG,N,DT,TMAX,N1,N2,N3,X,Y,M,MINV,INT,ITS,
1      AT,IC,REINT,Z,F,FT,IX,IREAD)
      REAL M(N2),MINV(N2)
      DIMENSION INT(N),ITS(N),X(N1),Y(N1),AT(N3),IC(N),REINT(N1)
      DIMENSION Z(N1),F(N),FT(N)
      COMMON /TRESP/A(1681),BU(41),JMP
      ICN=0
      IF(IX.NE.0)GO TO 2000
      WRITE(2,502)
      DO 500 I=1,N
500  WRITE(2,501)I,(A(I+N*(J-1)),J=1,N)
      IVS=0
      DO 600 I=1,N*N
600  M(I)=A(I)
      IF(N.GE.3)GO TO 2001
      IX=2
      RETURN
C
C      COMPUTE EIGENVALUES
C
2001  CONTINUE
      CALL EPPDIRHESSE(N,A(1),INT(1))
      CALL EPPORHESSE(N,A(1),ITS(1),X(1),Y(1),AT(1),IVS)
      WRITE(2,504)
      WRITE(2,503)
      DO 203 I=1,N
203  WRITE(2,202)I,X(I),Y(I)
      IFO=0
      RETURN
C
C      TEST FOR EQUAL ROOTS
C
      DO 5 I=1,N-1
      DO 5 J=I+1,N
      IF(ABS(X(I)-X(J)).GT..1E-06)GO TO 5
      IF(Y(I).EQ.-Y(J).AND.ABS(Y(I)).GT..1E-09)GO TO 5
      WRITE(2,7)
      IFO=1
      IX=2
      RETURN
5  CONTINUE
C
C      TEST FOR COMPLEX EIGENVALUES
C
      IF(IRUNG.EQ.1)GO TO 4000
      DO 3 I=1,N
      IF(ABS(Y(I)).LT..1E-09)GO TO 4
      IC(I)=1
      GO TO 3
4  IC(I)=0
3  CONTINUE
C
C      COMPUTE EIGENVECTORS

```

```

C
  NRO7=N*N+7*N
  CALL F4QSVS(N,NRO7,A,M,X,Y,AT)
  CALL F4BACK(N,A,M,V,INT)
  WRITE(2,505)
  DO 506 I=1,N
  WRITE(2,501)I,(M(I+N*(J-1)),J=1,N)
506 CONTINUE

C
C   COMPUTE INVERSE EIGENVECTORS
C
  DO 9 I=1,N
  DO 9 J=1,N
  9 MINV(I+N*(J-1))=0.0
  DO 10 I=1,N
  10 MINV(I+N*(I-1))=1.0
  IX=1
  NA=N*N
  DO 980 I=1,N*N
  980 AT(N1+I)=M(I)
  CALL F4SOLVE (M,MINV,N,NA,NA,IN,D,TD,IT,REINT)
  WRITE(2,1000)
  DO 1001 I=1,N
  1001 WRITE(2,501)I,(MINV(I+N*(J-1)),J=1,N)
  DO 981 I=1,N*N
  981 M(I)=AT(N1+I)
  IX=1
  6000 CONTINUE
  IF(TRYING.EQ.1)IX=2
  RETURN
  2000 IF(IX.EQ.2)GO TO 700
  CALL UTM4(MINV,BU,7,N,N,1)
  900 CONTINUE
  IF(TEQ.EQ.1)GO TO 700
  IF(IMP)R,520,521
  520 WRITE(2,522)
  523 CONTINUE
  T=-DT
  11 T=T+DT
  LI=1
  IF(IMP)R,12,13

C
C   THIS SECTION CALCULATES THE IMPULSE RESPONSE
C
  13 DO 14 I=1,N
  IF(LI)101,8,100
  100 IF(TC(I).EQ.1)GO TO 15
  F(I)=7(I)*EXP(X(I)*T)
  GO TO 14
  15 R=EXP(X(I)*T)
  S=Y(I)+T
  F(I)=7(I)*R*COS(S)+Z(I+1)*R*SIN(S)
  F(I+1)=-Z(I)*R*SIN(S)+Z(I+1)*R*COS(S)
  101 LI=LI+1
  14 CONTINUE
  CALL UTM4(M,F,FT,N,N,1)

C
C   WRITE TIME RESPONSE
C
  ICOU=ICOU+1
  WRITE(2,17)T,ICOU
  WRITE(2,300)
  WRITE(READ)T,(FT(I),I=1,N)

```

```
WRITE(2,301)((I,FT(I)),I=1,N)
```

```
NTIME=NINT((T-TMAX)/DT)
```

```
IF(NTIME)11,8,8
```

THIS SECTION COMPUTES THE STEP RESPONSE

```
12 DO 20 I=1,N
   IF(11)102,8,103
103 IF(10(I).EQ.1)GO TO 21
   F(I)=7(I)*(EXP(X(I)+T)-1)/X(I)
   GO TO 20
21 R=EXP(X(I)+T)
   Y1=V(I)
   S=Y(I)+T
   P=Y(I)+V(I)+X(I)+X(I)
   F(I)=7(I)*((Y1+SIN(S)+X(I)+COS(S))*R-X(I))/P+Z(I+1)*((X(I)+SIN(S)
1)-Y1+COS(S))*R+V1)/P
   F(I+1)=-Z(I)*((X(I)+SIN(S)-Y1+COS(S))*R+V1)/P+Z(I+1)*((Y1+SIN
1(P)+X(I)+COS(S))*R-X(I))/P
102 LI=-11
20 CONTINUE
   GO TO 14
STOP
521 WRITE(2,524)
   GO TO 523
```

THIS SECTION DOES THE RUNGE KUTTA INTEGRATION

```
700 WRITE(2,702)
   DO 601 I=1,N*N
601 A(I)=M(I)
   IRUNG=1
   M1=N+1
   IF(IRUNG.EQ.1)GO TO 710
   WRITE(2,522)
   DO 704 I=2,M1
704 X(I)=0
   GO TO 711
710 WRITE(2,524)
   DO 711 I=2,M1
711 X(I)=9H(I-1)
715 H=DT
   X(1)=0
   IF(IRUNG.EQ.0)GO TO 8
   INIT=0
705 CALL F4PUNG(M1,INIT,H,X,Z,Y)
   INIT=1
   ICOUNT=ICOUNT+1
   WRITE(2,17)X(1),ICOUNT
   WRITE(2,300)
   WRITE(1,READ)(X(I),I=1,N+1)
   WRITE(2,301)((I,X(I+1)),I=1,N)
   NTIME=NINT((X(1)-TMAX)/DT)
   IF(NTIME)705,8,8
8 CONTINUE
   RETURN
7 FORMAT(///18H EQUAL ROOTS FOUND)
505 FORMAT(///23H MATRIX OF EIGENVECTORS/)
504 FORMAT(///19H SYSTEM EIGENVALUES/)
503 FORMAT(35H VARIABLE REAL IMAG)
202 FORMAT(3X,I3,2(4X,E13.6))
502 FORMAT(///9H A MATRIX/)
501 FORMAT(/6H ROW #,I3,4E16.6/(9X,4E16.6))
```

```

17 FORMAT(///7H TIME =,E10.4,I4.//)
19 FORMAT(4X,I3,5X,E10.4)
300 FORMAT(64H VARIABLE FT VARIABLE FT VARIABLE
1 FT)
301 FORMAT(3(3X,I3,E17.6))
522 FORMAT(///11H STEP INPUT///)
524 FORMAT(///17H IMPULSE RESPONSE///)
1000 FORMAT(///31H MATRIX OF INVERSE EIGENVECTORS/)
702 FORMAT(24H1RUNGE-KUTTA INTEGRATION///)
END

```

```

SUBROUTINE F4DERV(M,Y,DY)
DIMENSION Y(M),DY(M)
COMMON /TRESP/A(1681),BU(41),IMP

```

THIS SEGMENT COMPUTES THE RATE OF CHANGE OF THE MODEL

$$X' = AX + B$$
FOR USE IN THE RUNGE KUTTA INTEGRATION.

```

N=M-1
DO 1 I=1,N
IS(IMP,EQ.1)GO TO 3
DY(I+1)=BU(I)
GO TO 4
3 DY(I+1)=0
4 DO 1 J=1,N
SUM=A(I+N*(J-1))+Y(J+1)
1 DY(I+1)=SUM+DY(I+1)
RETURN
END

```

```

SUBROUTINE WRITEMAT(N,M,NM,A)
DIMENSION A(NM)

```

THIS SEGMENT WRITES OUT THE (N,M) MATRIX A.

```

WRITE(2,1)
DO 2 I=1,N
2 WRITE(2,3)I,(A(I+N*(J-1)),J=1,M)
RETURN
1 FORMAT(//)
3 FORMAT(/6H ROW =,I3,7E16.6/(9X,7E16.6))
END

```

```

SUBROUTINE TRANMAT(A,AT,M,N)
DIMENSION A(N,N),AT(N,M)

```

THIS SEGMENT PUTS THE TRANSPOSE OF THE (M,N) MATRIX A INTO AT.

```

DO 1 I=1,M
DO 1 J=1,N

```



```
1  AT(I,T)=A(I,J)  
   RETURN  
   END
```

APPENDIX 3

A3.2 Model 1, an overdamped system, reduced
by matching the frequency response.

Contents

1. Full and reduced model data.
2. Step responses as detailed below.

Figure	Input	State x_i	reduced state x_i^*
A3.1	1	1	1
A3.2	1	3	2
A3.3	1	7	3
A3.4	1	12	4
A3.5	2	3	2
A3.6	2	7	3
A3.7	2	12	4

State x_1 was not forced by input 2

A SYSTEM OF OVERDAMPED STIRRED TANKS

FULL MODEL DATA

A MATRIX

ROW = 1	-0.100000E 01	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 2	0.200000E 01	-0.200000E 01	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 3	0.000000E 00	0.100000E 02	-0.100000E 02	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 4	0.000000E 00	0.000000E 00	0.130000E 02	-0.130000E 02
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 5	0.000000E 00	0.000000E 00	0.000000E 00	0.300000E 01
	-0.300000E 01	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 6	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.800000E 01	-0.800000E 01	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 7	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.500000E 01	-0.500000E 01	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 8	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.110000E 02	-0.110000E 02
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 9	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.600000E 01
	-0.600000E 01	0.000000E 00	0.000000E 00	0.000000E 00
ROW = 10	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.120000E 02	-0.120000E 02	0.000000E 00	0.000000E 00
ROW = 11	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.900000E 01	-0.900000E 01	0.000000E 00
ROW = 12	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.000000E 00	0.000000E 00
	0.000000E 00	0.000000E 00	0.140000E 02	-0.140000E 02

SYSTEM EIGENVALUES

VARIABLE	REAL	IMAG
1	-0.100000E 01	0.000000E 00
2	-0.200000E 01	0.000000E 00
3	-0.300000E 01	0.000000E 00
4	-0.500000E 01	0.000000E 00
5	-0.600000E 01	0.000000E 00
6	-0.130000E 02	0.000000E 00
7	-0.800000E 01	0.000000E 00
8	-0.120000E 02	0.000000E 00
9	-0.100000E 02	0.000000E 00
10	-0.110000E 02	0.000000E 00
11	-0.900000E 01	0.000000E 00
12	-0.140000E 02	0.000000E 00

MATRIX B

ROW = 1	0.100000E 01	0.000000E 00
ROW = 2	0.000000E 00	0.200000E 01
ROW = 3	0.000000E 00	0.000000E 00
ROW = 4	0.000000E 00	0.000000E 00
ROW = 5	0.000000E 00	0.000000E 00
ROW = 6	0.000000E 00	0.000000E 00
ROW = 7	0.000000E 00	0.000000E 00
ROW = 8	0.000000E 00	0.000000E 00
ROW = 9	0.000000E 00	0.000000E 00
ROW = 10	0.000000E 00	0.000000E 00
ROW = 11	0.000000E 00	0.000000E 00
ROW = 12	0.000000E 00	0.000000E 00

VARIABLES OF INTEREST

1 3 7 12

FREQUENCIES FITTED

0.100000E-01	0.200000E-01	0.300000E-01	0.400000E-01
0.700000E-01	0.100000E 00	0.200000E 00	0.300000E 00
0.400000E 00	0.500000E 00	0.600000E 00	0.700000E 00
0.800000E 00	0.900000E 00	0.100000E 01	0.130000E 01

0.150000E 01	0.170000E 01	0.200000E 01	0.230000E 01
0.270000E 01	0.300000E 01		

REDUCED MODEL DATA

MATRIX B

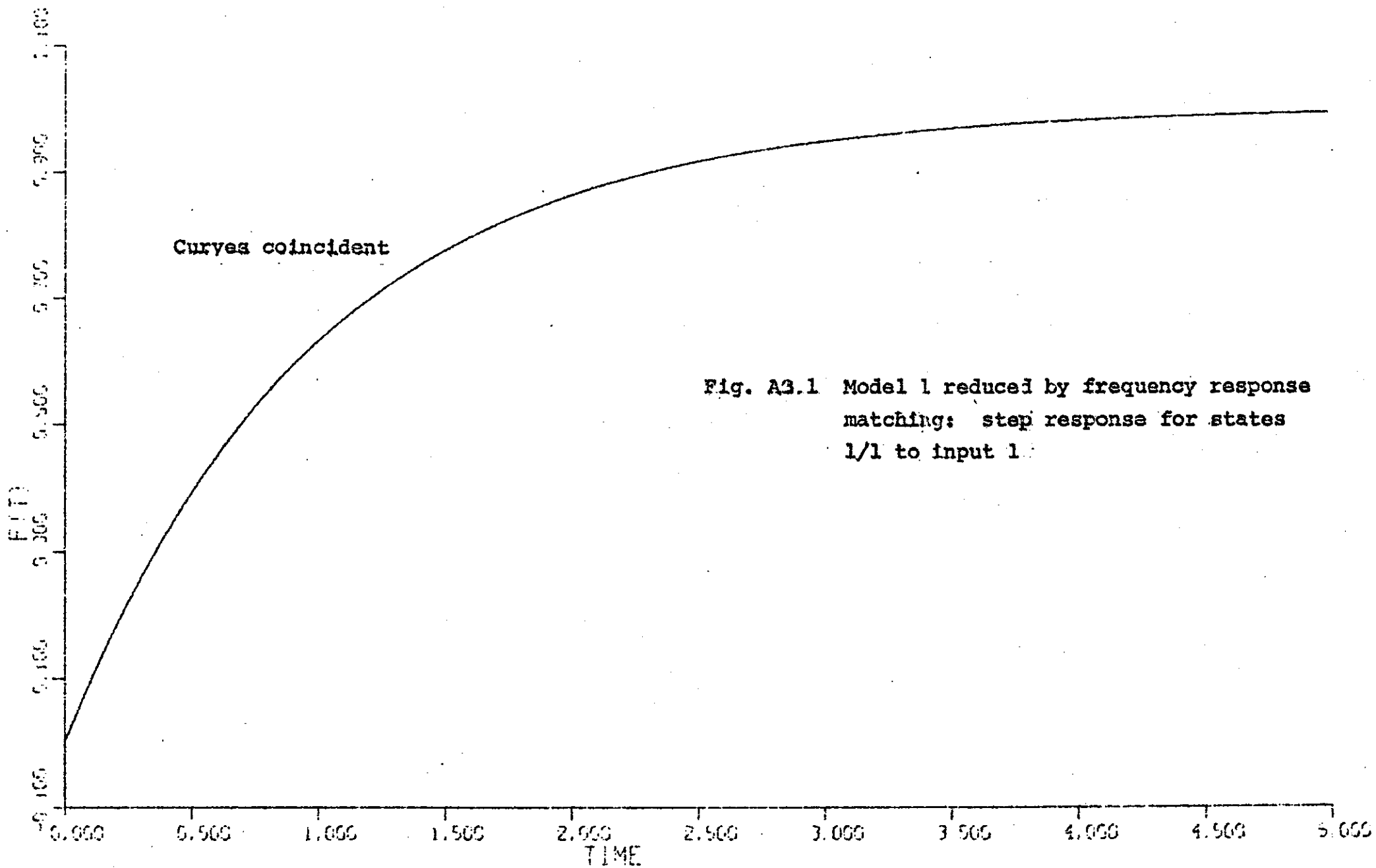
ROW = 1	0.100000E 01	-0.156210E-07
ROW = 2	-0.103582E 00	0.168969E 01
ROW = 3	0.103047E 00	-0.692271E 00
ROW = 4	-0.156227E 00	0.877878E 00

A MATRIX

ROW = 1	-0.100000E 01	0.321287E-07	-0.312381E-07	0.147304E-07
ROW = 2	0.179327E 01	-0.152594E 01	-0.227168E 00	0.634203E-01
ROW = 3	-0.795318E 00	0.218220E 01	-0.106511E 01	-0.424822E 00
ROW = 4	0.103410E 01	-0.221478E 01	0.435237E 01	-0.301547E 01

SYSTEM EIGENVALUES

VARIABLE	REAL	IMAG
1	-0.100000E 01	0.000000E 00
2	-0.169860E 01	0.000000E 00
3	-0.195396E 01	-0.121151E 01
4	-0.195396E 01	0.121151E 01



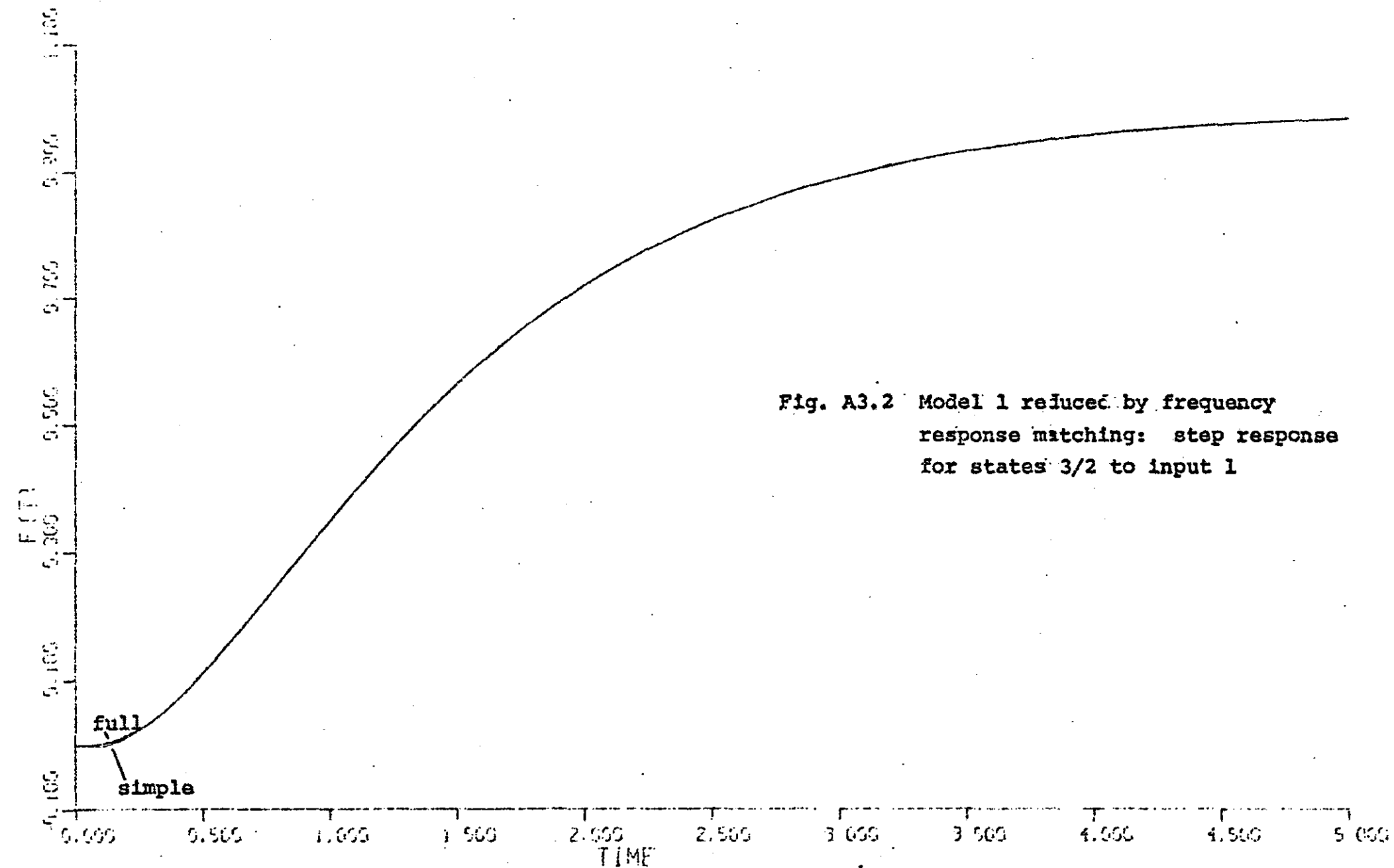


Fig. A3.2 Model 1 reduced by frequency
response matching: step response
for states 3/2 to input 1

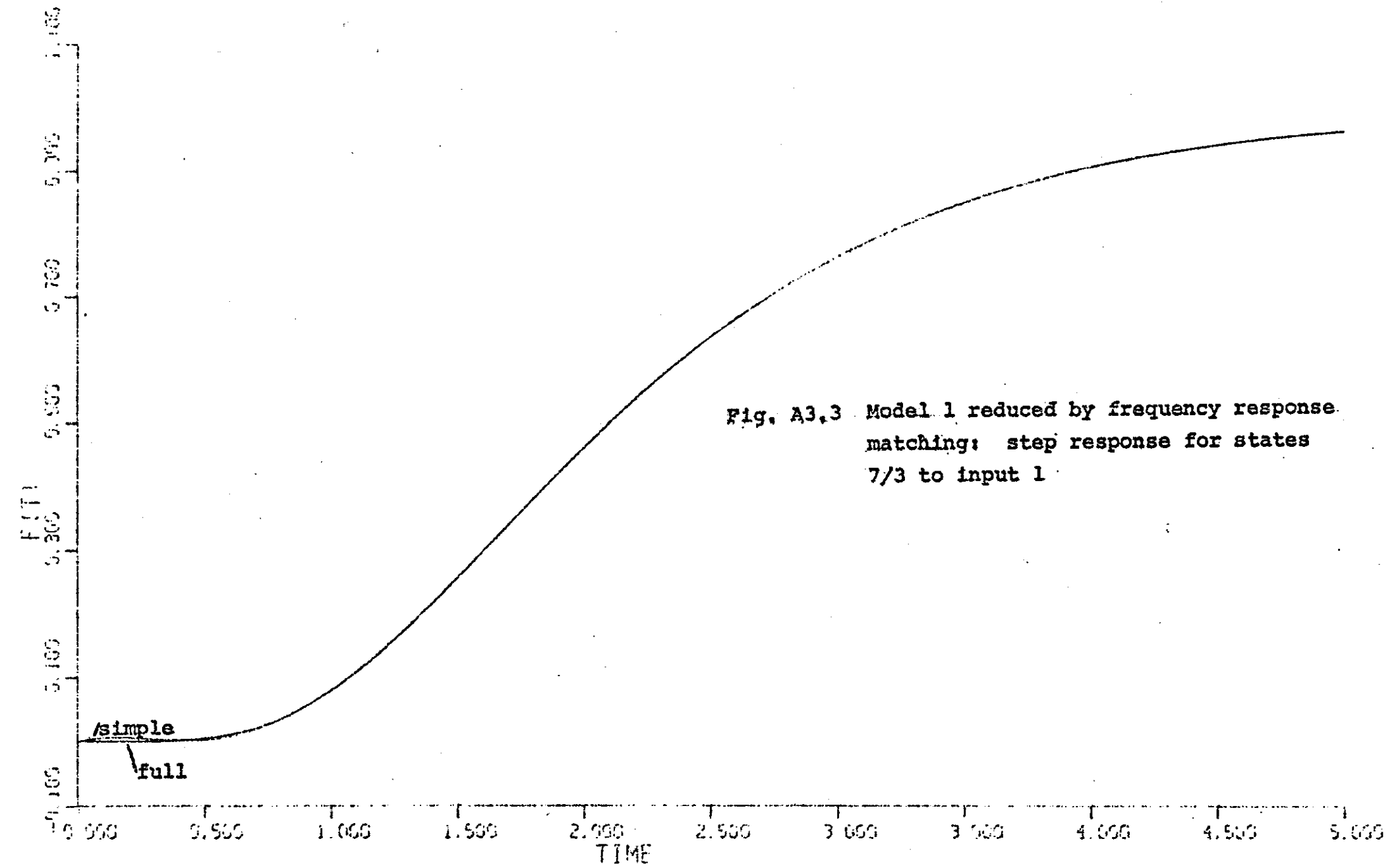


Fig. A3.3 Model 1 reduced by frequency response matching: step response for states 7/3 to input 1

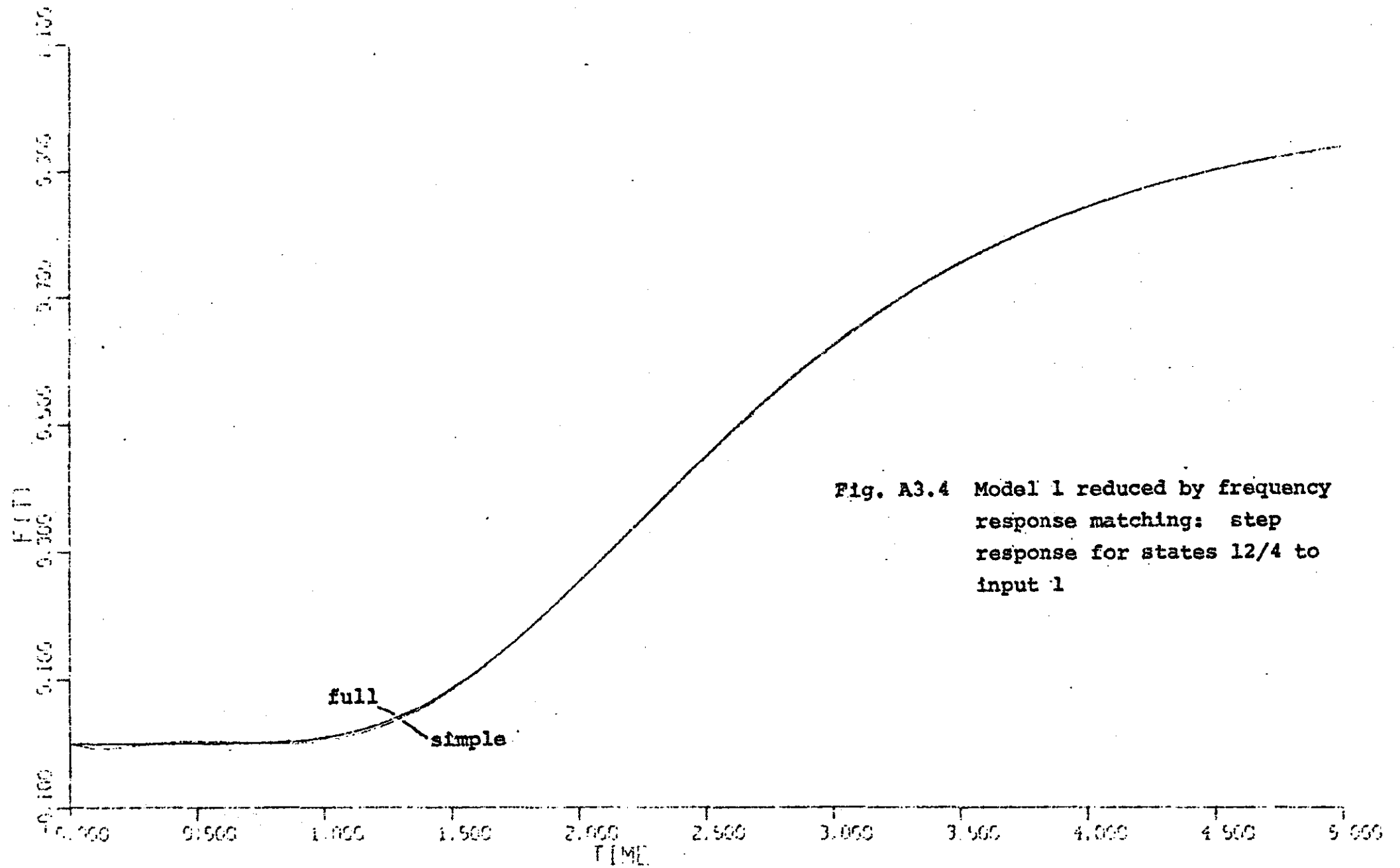


Fig. A3.4 Model 1 reduced by frequency response matching: step response for states 12/4 to input 1

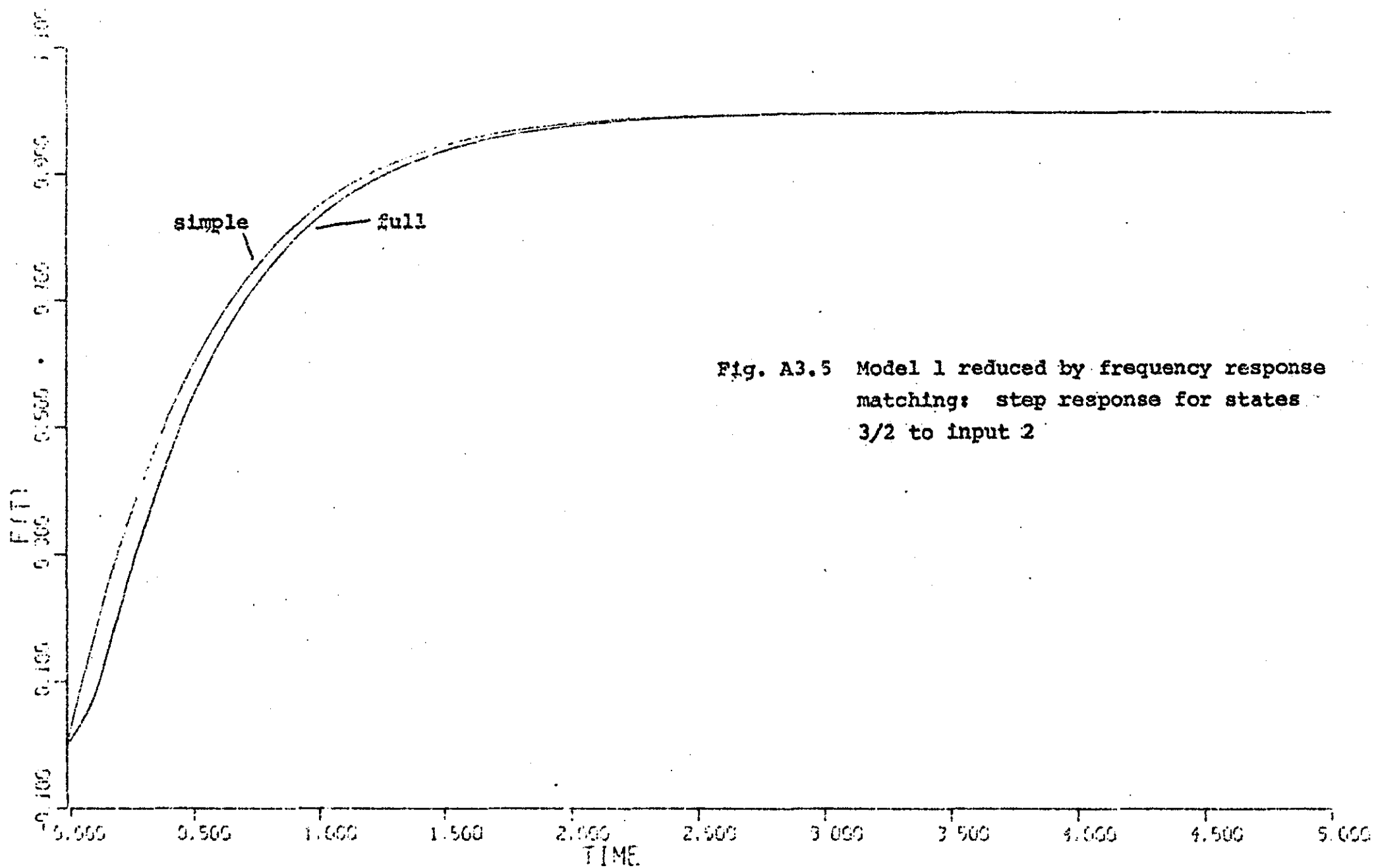


Fig. A3.5 Model 1 reduced by frequency response matching: step response for states 3/2 to input 2

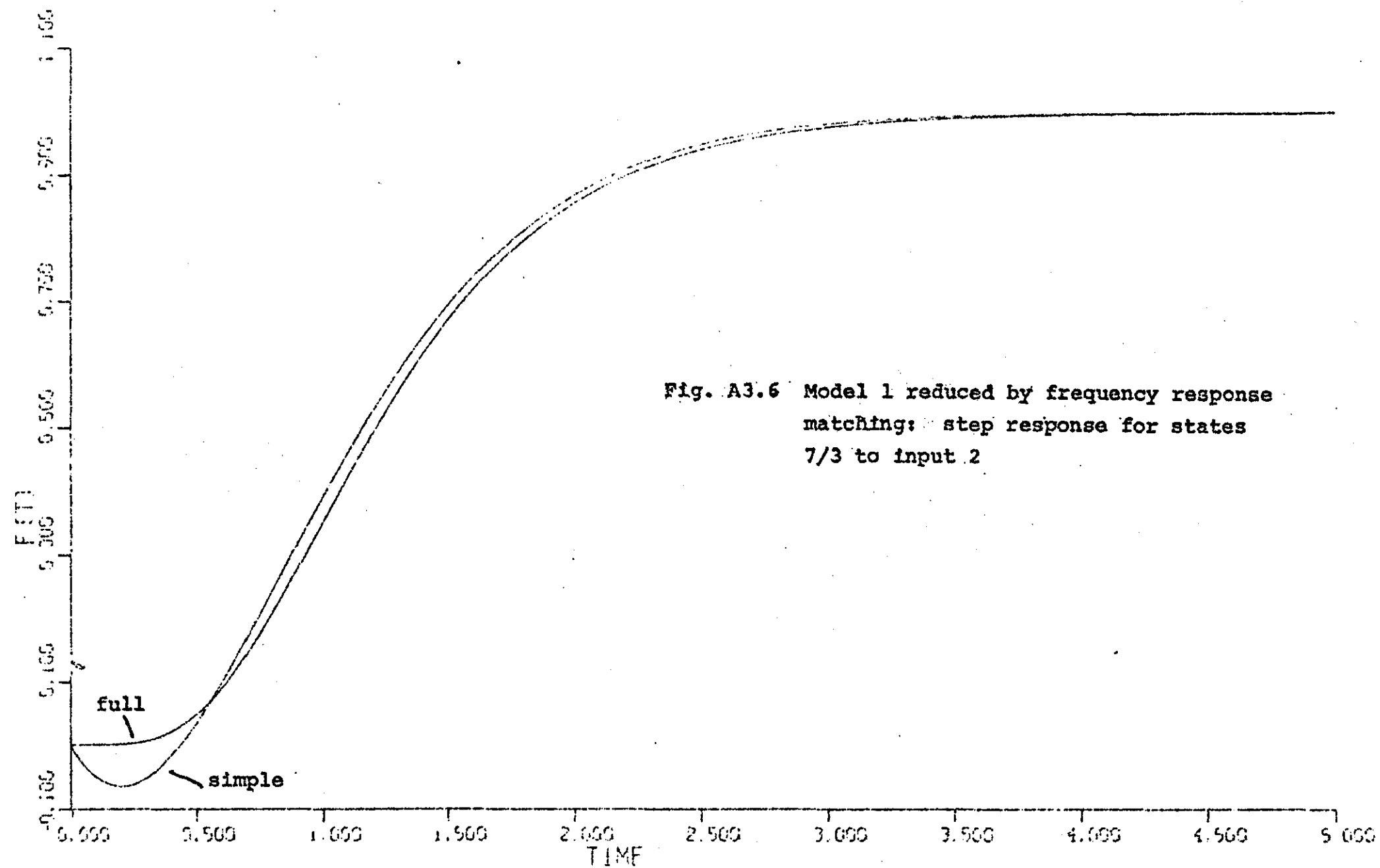


Fig. A3.6 Model 1 reduced by frequency response matching: step response for states 7/3 to input 2

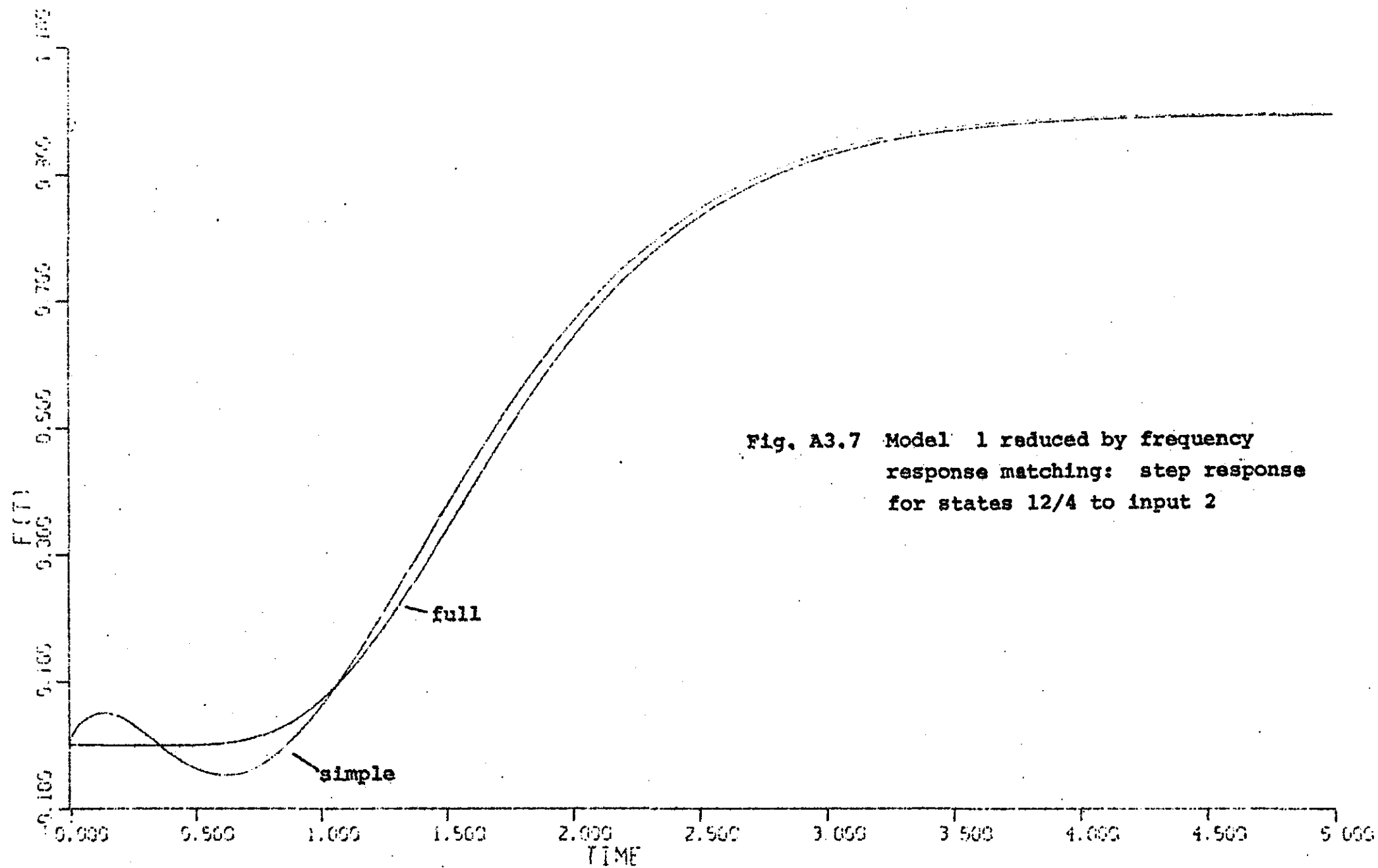


Fig. A3.7 Model 1 reduced by frequency
response matching: step response
for states 12/4 to input 2

APPENDIX 3

A3.3 Model 4, a binary distillation column,
reduced by matching the frequency response.

Contents

1. Full and reduced model data.
2. Step responses as detailed below.

Figure	Input	State x_i	reduced state x_i^*
A3.8	1	1	1
A3.9	1	3	2
A3.10	1	8	7
A3.11	1	11	4
A3.12	2	1	1
A3.13	2	3	2
A3.14	2	8	3
A3.15	2	11	4

A BINARY DISTILLATION COLUMN

FULL MODEL DATA

A MATRIX

ROW = 1	-0.140000E-01 0.000000E 00 0.000000E 00	0.430000E-02 0.000000E 00 0.000000E 00	0.000000E 00 0.000000E 00 0.000000E 00	0.000000E 00 0.000000E 00 0.000000E 00
ROW = 2	0.950000E-02 0.000000E 00 0.000000E 00	-0.138000E-01 0.000000E 00 0.000000E 00	0.460000E-02 0.000000E 00 -0.300000E-03	0.000000E 00 0.000000E 00 0.000000E 00
ROW = 3	0.000000E 00 0.000000E 00 0.000000E 00	0.930000E-02 0.000000E 00 0.000000E 00	-0.141000E-01 0.000000E 00 -0.500000E-03	0.630000E-02 0.000000E 00 0.000000E 00
ROW = 4	0.000000E 00 0.110000E-01 0.000000E 00	0.000000E 00 0.000000E 00 0.000000E 00	0.950000E-02 0.000000E 00 -0.800000E-03	-0.158000E-01 0.000000E 00 0.000000E 00
ROW = 5	0.000000E 00 -0.312000E-01 0.000000E 00	0.000000E 00 0.150000E-01 0.000000E 00	0.000000E 00 0.000000E 00 -0.800000E-03	0.950000E-02 0.000000E 00 0.000000E 00
ROW = 6	0.000000E 00 0.202000E-01 0.000000E 00	0.000000E 00 -0.352000E-01 0.000000E 00	0.000000E 00 0.220000E-01 -0.800000E-03	0.000000E 00 0.000000E 00 0.000000E 00
ROW = 7	0.000000E 00 0.000000E 00 0.000000E 00	0.000000E 00 0.202000E-01 0.000000E 00	0.000000E 00 -0.422000E-01 -0.800000E-03	0.000000E 00 0.280000E-01 0.000000E 00
ROW = 8	0.000000E 00 0.000000E 00 0.370000E-01	0.000000E 00 0.000000E 00 0.000000E 00	0.000000E 00 0.202000E-01 -0.600000E-03	0.000000E 00 -0.482000E-01 0.000000E 00
ROW = 9	0.000000E 00 0.000000E 00 -0.572000E-01	0.000000E 00 0.000000E 00 0.420000E-01	0.000000E 00 0.000000E 00 -0.300000E-03	0.000000E 00 0.202000E-01 0.000000E 00
ROW = 10	0.000000E 00 0.000000E 00 0.202000E-01	0.000000E 00 0.000000E 00 -0.483000E-01	0.000000E 00 0.000000E 00 0.000000E 00	0.000000E 00 0.000000E 00 0.000000E 00
ROW = 11	0.255000E-01 0.000000E 00 0.000000E 00	0.000000E 00 0.000000E 00 0.255000E-01	0.000000E 00 0.000000E 00 -0.185000E-01	0.000000E 00 0.000000E 00 0.000000E 00

SYSTEM EIGENVALUES

VARIABLE	REAL	IMAG
----------	------	------

1	-0.960340E-01	0.000000E 00
2	-0.505225E-01	0.000000E 00
3	-0.700745E-01	0.000000E 00
4	-0.247194E-01	0.000000E 00
5	-0.337243E-01	0.000000E 00
6	-0.823316E-02	0.000000E 00
7	-0.200682E-01	0.000000E 00
8	-0.489846E-03	0.000000E 00
9	-0.139995E-01	0.000000E 00
10	-0.325079E-02	0.000000E 00
11	-0.173768E-01	0.000000E 00

MATRIX B

ROW = 1	0.000000E 00	0.000000E 00
ROW = 2	-0.300000E-05	0.300000E-04
ROW = 3	-0.500000E-05	0.500000E-04
ROW = 4	-0.500000E-05	0.500000E-04
ROW = 5	-0.500000E-05	0.500000E-04
ROW = 6	-0.500000E-05	0.500000E-04
ROW = 7	-0.500000E-05	0.500000E-04
ROW = 8	-0.200000E-04	0.500000E-04
ROW = 9	-0.400000E-04	0.400000E-04
ROW = 10	-0.200000E-04	0.200000E-04
ROW = 11	0.460000E-03	0.460000E-03

VARIABLES OF INTEREST

1 3 8 11

FREQUENCIES FITTED

0.100000E-05	0.200000E-05	0.400000E-05	0.700000E-05
0.100000E-04	0.200000E-04	0.400000E-04	0.700000E-04
0.100000E-03	0.200000E-03	0.400000E-03	0.700000E-03
0.100000E-02	0.150000E-02	0.200000E-02	0.300000E-02
0.400000E-02	0.600000E-02	0.800000E-02	0.900000E-02
0.100000E-01	0.150000E-01	0.200000E-01	0.300000E-01
0.350000E-01	0.400000E-01	0.450000E-01	0.500000E-01

REDUCED MODEL DATA

MATRIX B

ROW = 1	0.606442E-07	0.197498E-05
ROW = 2	-0.485930E-05	0.381084E-04
ROW = 3	-0.242061E-04	0.443965E-04
ROW = 4	0.445093E-03	0.476773E-03

A MATRIX

ROW = 1	-0.114390E-01	0.177334E-02	-0.212368E-03	-0.105470E-04
ROW = 2	-0.220372E-01	0.257858E-02	0.173486E-03	-0.449704E-03
ROW = 3	-0.248335E-01	0.773227E-02	-0.400497E-02	-0.726149E-03
ROW = 4	0.458094E-01	-0.478805E-02	0.732789E-02	-0.184917E-01

SYSTEM EIGENVALUES

VARIABLE	REAL	IMAG
1	-0.184134E-01	0.000000E 00
2	-0.570491E-03	0.000000E 00
3	-0.810913E-02	0.000000E 00
4	-0.425506E-02	0.000000E 00

A3.8 Model 4 reduced by frequency response matching:
step response for states 1/1 to input 1

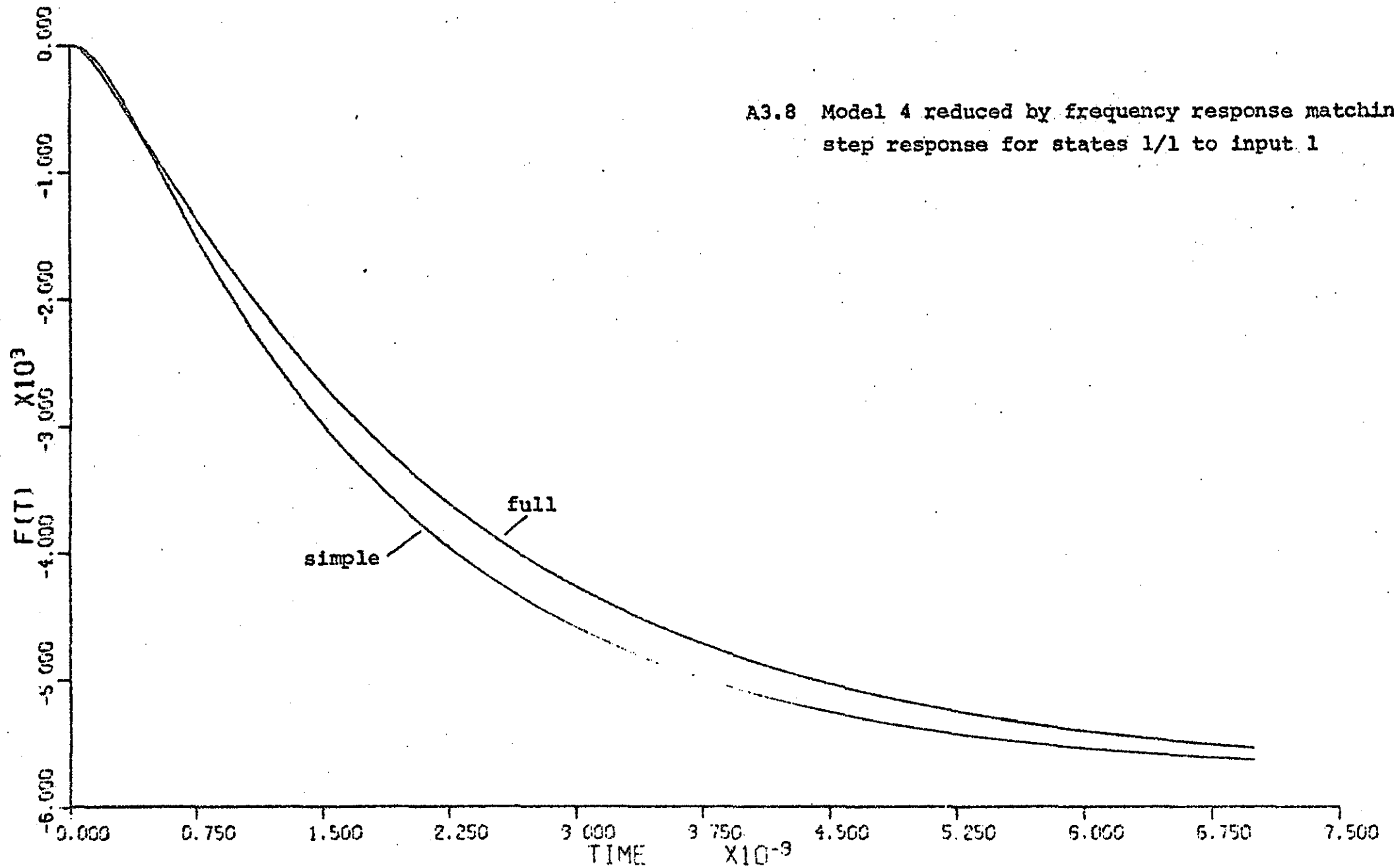
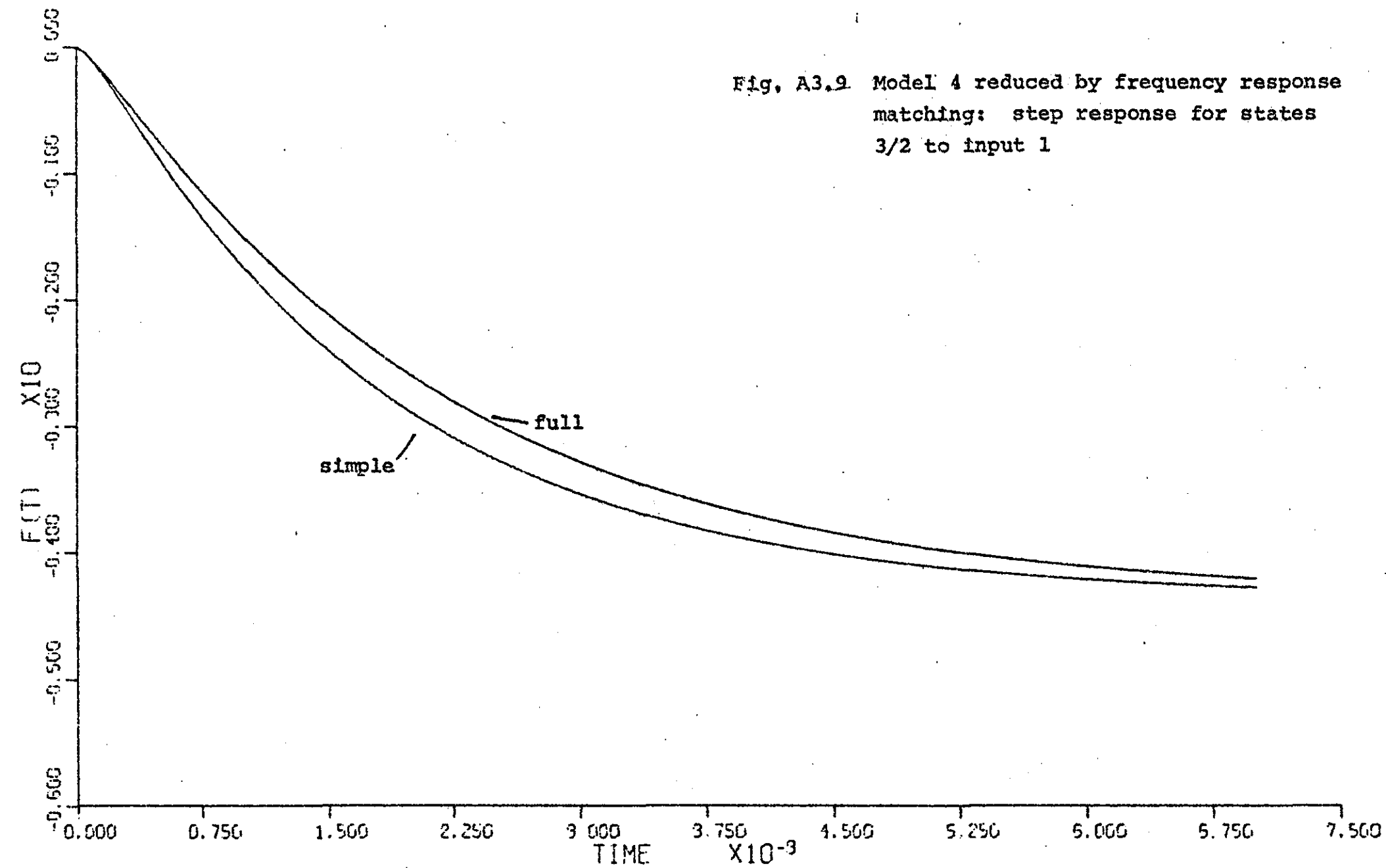
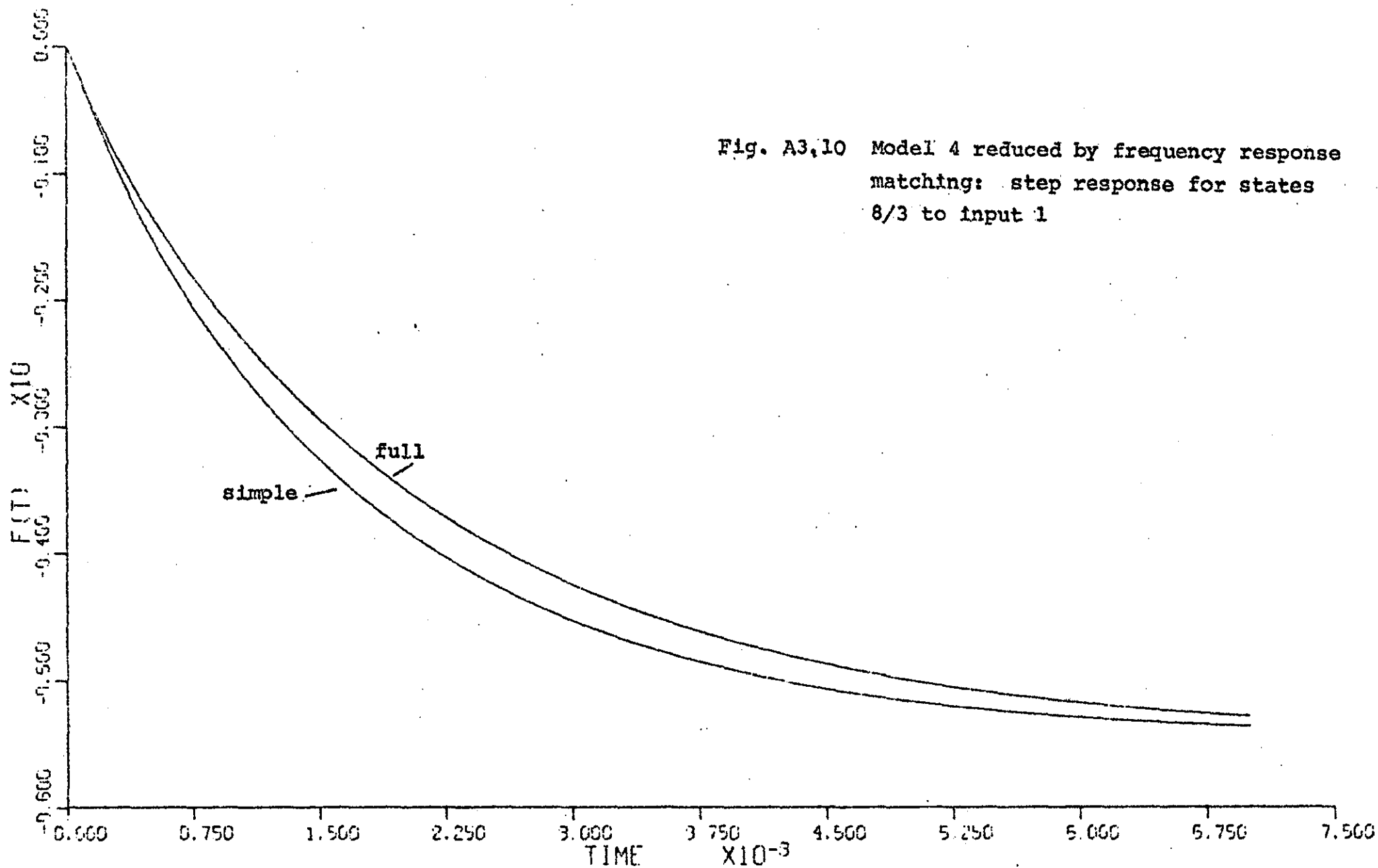
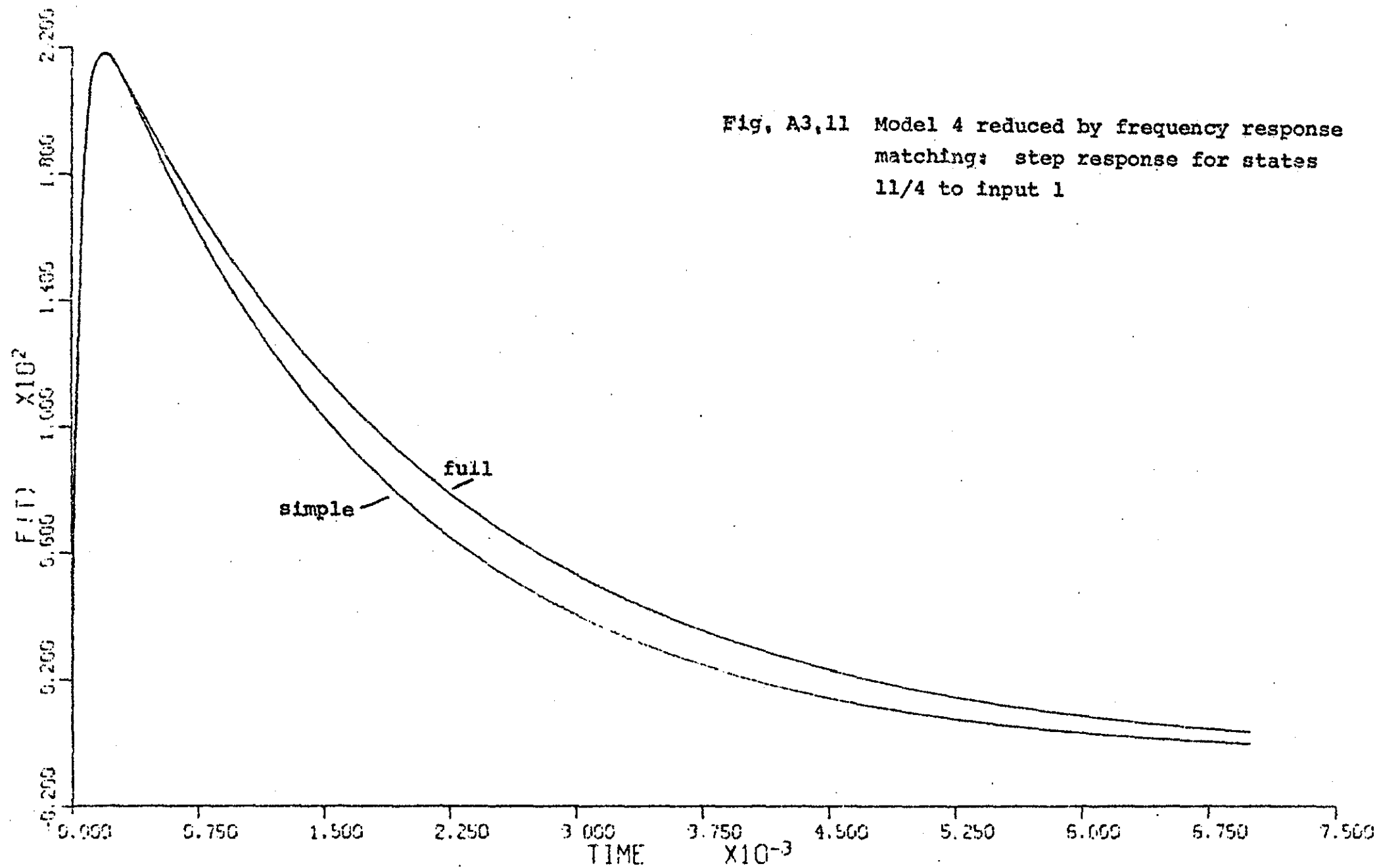


Fig. A3.9 Model 4 reduced by frequency response
matching: step response for states
3/2 to input 1







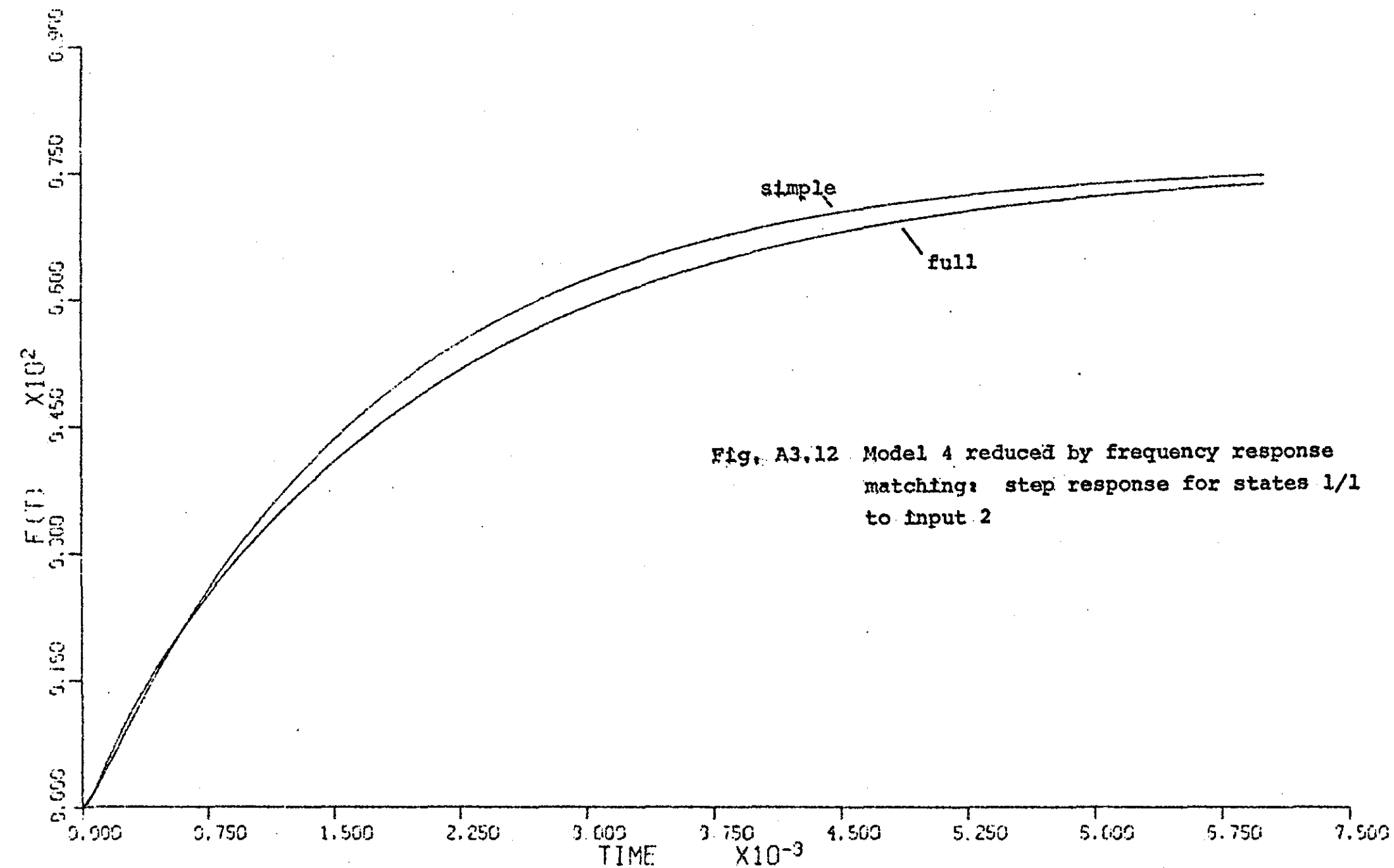


Fig. A3.12 Model 4 reduced by frequency response
matching: step response for states 1/1
to input 2

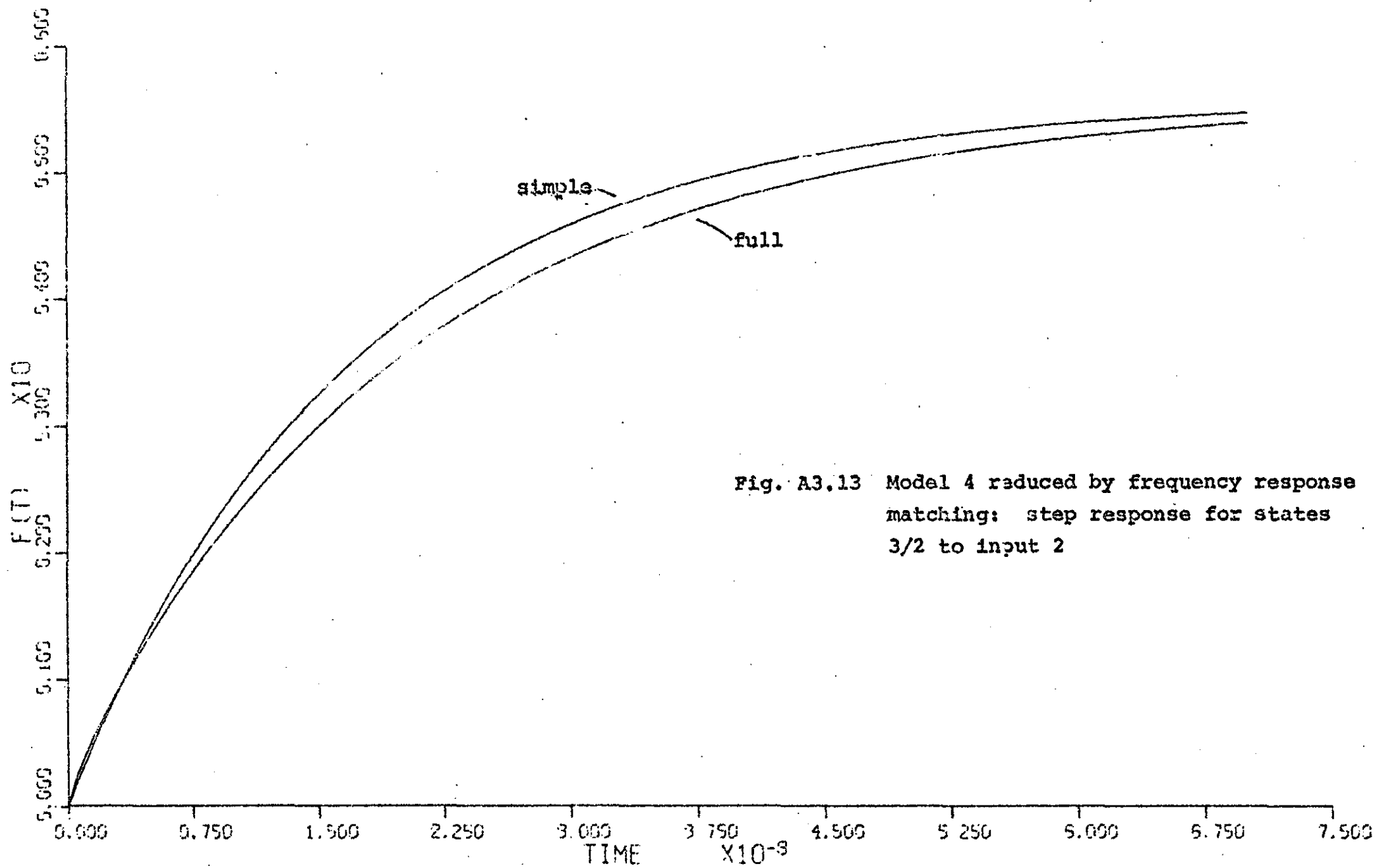


Fig. A3.13 Model 4 reduced by frequency response matching: step response for states 3/2 to input 2

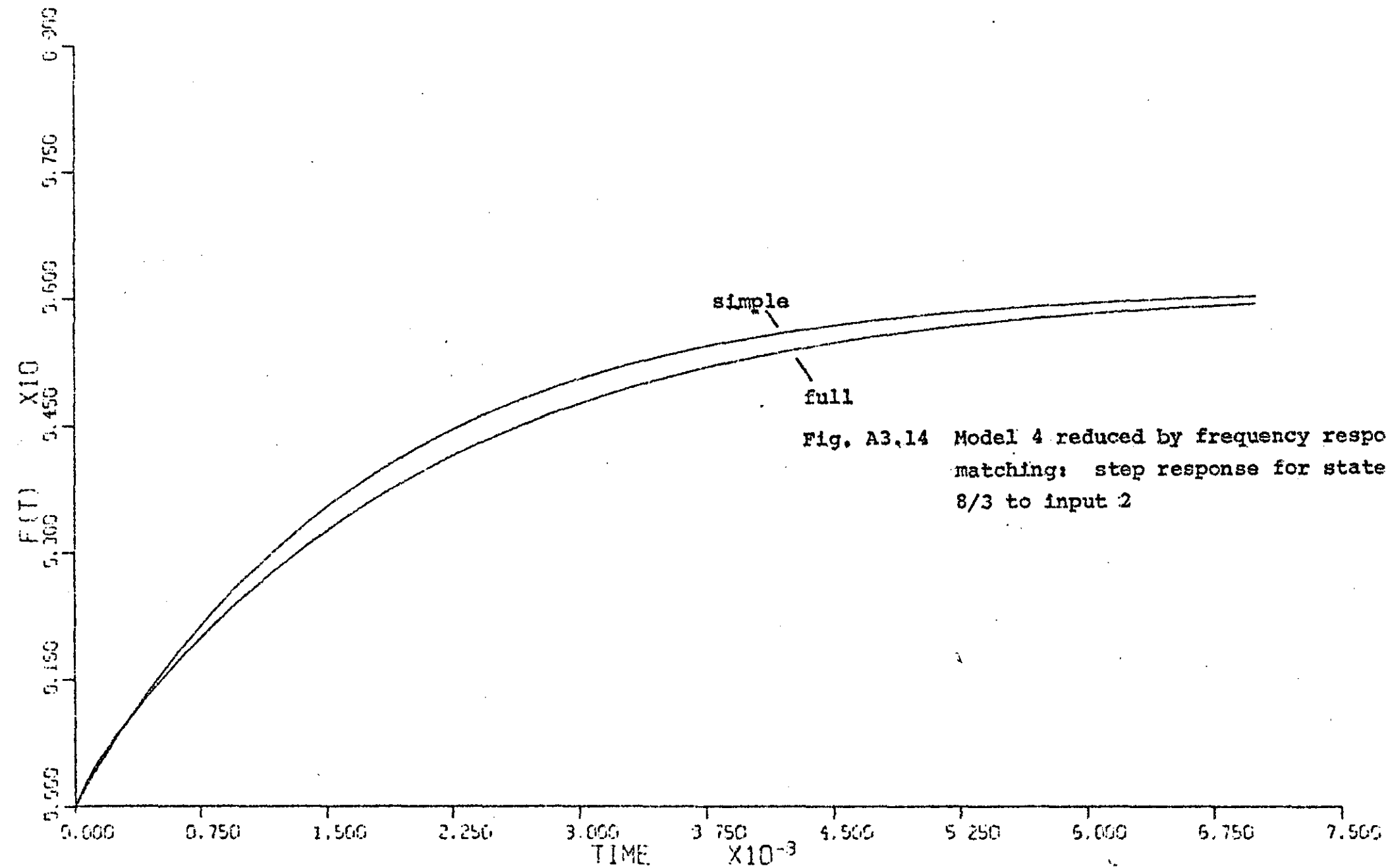


Fig. A3.14 Model 4 reduced by frequency response matching: step response for states 8/3 to input 2

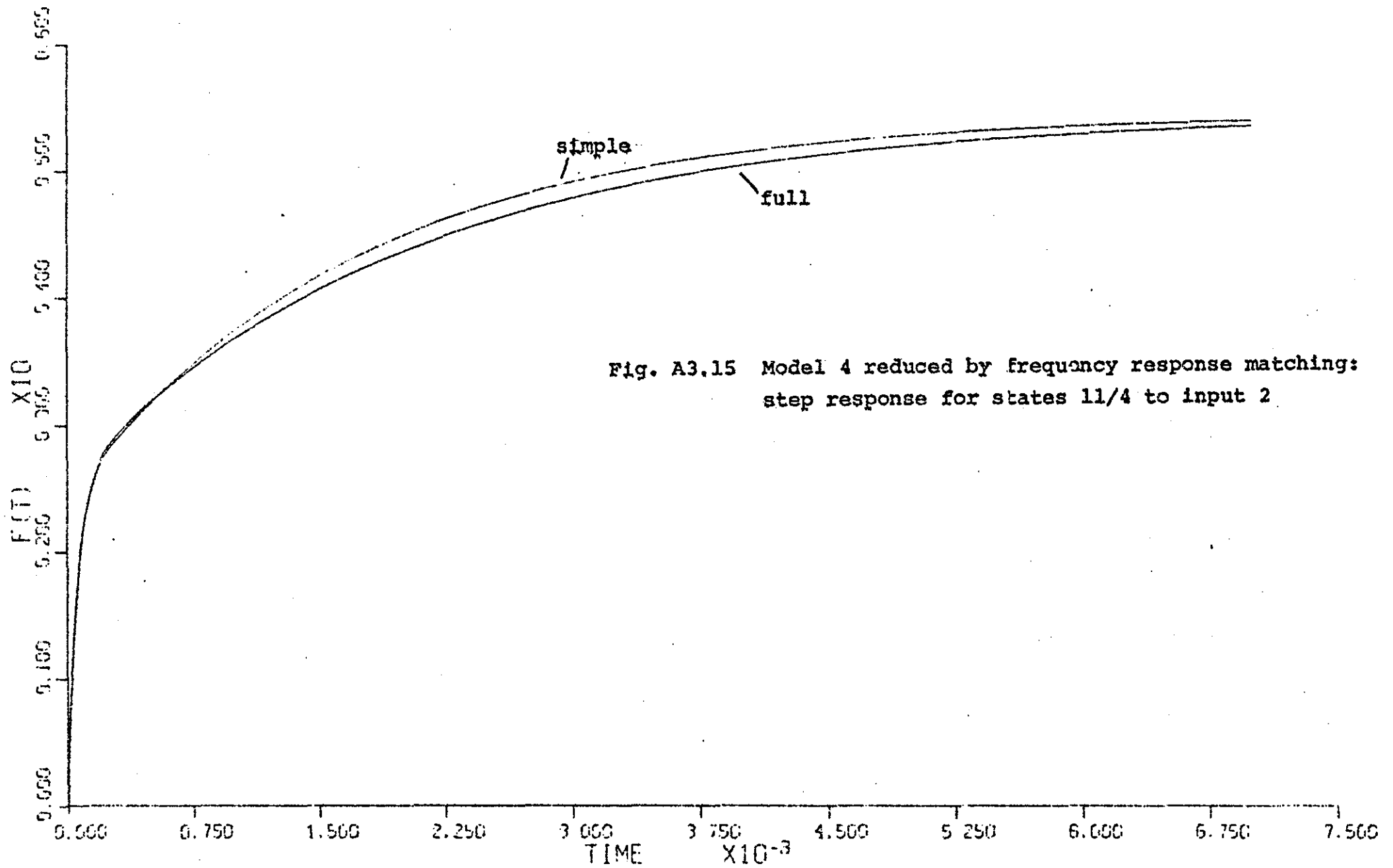


Fig. A3.15 Model 4 reduced by frequency response matching:
step response for states 11/4 to input 2

