NSG- 3048

Department of

ELECTRICAL ENGINEERING

UNIVERSITY OF NOTRE DAME, NOTRE DAME, INDIANA

TIME OPTIMAL CONTROL OF A JET ENGINE USING A QUASI-HERMITE INTERPOLATION MODEL*

John G. Comiskey

May 1979

Technical Report No. EE-791

*This research was supported in part by the National Aeronautics and Space Administration under Grant NSG-3048, and also has been submitted in partial fulfillment of the requirements for the M.S.E.E. degree at the University of Notre Dame.

ABSTRACT

The scope of this work will be to make preliminary efforts to generate non-linear numerical models of a two-spooled turbofan jet engine, and subject these models to a known method of generating global, non-linear, time optimal control laws. The models will be derived numerically, directly from empirical data, as a first step in developing an automatic modelling procedure.

ACKNOWLEDGMENT

I wish to acknowledge the technical assistance of Dr. R.J. Leake, Dr. M.K. Sain, and Dr. R.S. Eikenberry; the clerical assistance of Ms. Tammy Youngs; the financial support of NASA; the emotional support of my family, John B., Jeanne, Michael and Elyse Comiskey; and the riotous diversions provided by my friends at Notre Dame. Without them this work would have been impossible.

Time Optimal Control of a Jet Engine Using
a Quasi-Hermite Interpolation Model

Table of Contents

CHAPTER I

INTRODUCTION

The scope of this work will be to make preliminary efforts to generate non-linear numerical models of a two-spooled turbofan jet engine, and subject these models to a known method of generating global, non-linear, time optimal control laws. The models will be derived numerically, directly from empirical data, as a first step in developing an automatic modelling procedure.

A hierarchy of models, including analytical and numerical models, will be established. The numerical models will be described in detail, and their step responses compared to those of the hypothetical jet engine from which they were derived. A method of generating time optimal control laws will be explained, programmed, and applied to the numerical models. Finally, these control laws will be tested, both on the models from which they were generated, and on the hypothetical jet engine.

This is the third in a series of similar works, whose ultimate goal is the development of an automated modelling method. Even though DYNGEN, an elaborate mathematical model, already exists, new models were developed for two reasons. First, DYNGEN uses too much cpu time to be called repeatedly by an iterative method such as dynamic programming; a smaller, faster model is required. Second, DYNGEN assumes the role of a physical plant in this work, since access to a real jet engine is impossible.

In his paper, Basso [9] uses two methods of generating optimal control sequences. The first is the dynamic programming successive approximations technique. This actually generates a control law, from which a control sequence can be derived. The second is a modified Fletcher-Reeves conjugate gradient method. This method generates a control sequence that drives the system to the target in minimum time. The modification consists of the introduction of constraints into the original method.

His findings were that both methods yielded similar results, and that the number of computations necessary to solve a problem increases geometrically with system order for dynamic programming, but only arithmetically for the conjugate gradient method.

Longenbaker [1] applies the dynamic programming method to several models of the F-100 engine. His models include several linear systems, and one non-linear, analytical system of differential equations derived from physical and mathematical relationships among the state and control variables. Longenbaker concludes that the agreement between this analytical model and the DYNGEN simulator is not strong enough to justify great faith in the control law generated.

In this paper, the same dynamic programming method is applied with a modification introduced to reduce cpu time, to several numerical, non-linear models of the F-100 jet engine. The conclusions are that better numerical agreement with DYNGEN is achieved by this numerical models than by Longenbaker's analytical model, with a much smaller expenditure of man hours. However, complex interpolation techniques cause these models to use extravagant amounts of cpu time. Either larger data bases and less interpolation, or a more economical technique like Basso's conjugate gradient method should be explored in future work.

CHAPTER II

TWO SPOOL TURBOFAN JET ENGINE MODELS

2.1 Introduction

A form was chosen for the system model, which isolates static and dynamic portions of the system behavior, so that each of these can be modelled independently. Several methods of modelling these two portions were tried, resulting in a hierarchy of models. Each model was subjected to the same analysis for purposes of comparison.

In this chapter, the system model form is derived, and the modelling methods outlined. These methods will be treated in detail in later chapters.

2.2 Basic Approximation Approach [2]

Now consider a method for obtaining nonlinear models. Let

$$\dot{x} = f(x,u) \qquad (2.2\text{-}1)$$

with x an n vector and u an m vector denoting a dynamical system such as a jet engine, in which the state variables and parameters u remain positive throughout the system operation and there is a function $g(u)$ such that for each equilibrium point

$$f(x,u) = 0 \longleftrightarrow x = g(u) \qquad (2.2\text{-}2)$$

The steady state system analysis involves the study of the function $g(u)$.

We propose to approximate the system (2.2-1) by

$$\dot{x}=A(x,u)[x - g(u)] \qquad (2.2\text{-}3)$$

where $A(x)$ is a square matrix which varies as a function of x. Notice that if x_D is an equilibrium point of (2.2-1), $x_D = g(u_D)$, then a linearization about this equilibrium point results in the linear system

$$\delta\dot{x} = A_D\delta x + B_D\delta u \qquad (2.2\text{-}4)$$

and a linearization of the approximating system (2) at $x_D = g(u_D)$ results in

$$\delta\dot{x} = A(x_D)\delta x + [-A(x_D)\,\frac{\delta g}{\delta u}\,(u_D)]\delta u \qquad (2.2\text{-}5)$$

Hence, the linearization of (2) will match the linearization of (1) if and only if

$$A(x_D) = A_D \quad , \quad -A_D\,\frac{\delta g}{\delta u}\,(u_D) = B_D \qquad (2.2\text{-}6)$$

Also, if A_D is invertible, as is often the case for jet engine models, equation yields

$$\frac{\delta g}{\delta u}\,(u_D) = -A_D^{-1}B_D \qquad (2.2\text{-}7)$$

These static and dynamic data are available from known algorithms [7], leaving only the choice of interpolation methods for generating non-linear models.

2.3 Hierarchy of Models

This work has resulted in the formation of a hierarchy of models, each a step in the development of an automated modelling method. They are classified as follows:

Model 0: The actual F-100 type engine (hypothetical)

Model 1: The DYNGEN [6] simulation program, coded with data presumed to have been taken from experimental measurements on Model 0. This model solves 16 nonlinear differential equations and uses data maps and thermodynamic tables which cannot be expressed analytically.

Model 2: The Longenbaker [1] model, a 5th order, nonlinear, analytical model. It includes the 5 state differential equations which govern the dynamical behavior of the system, along with 20 algebraic equations which express the relationship between various engine variables. This model is discussed in detail in [1].

Model 3A: The linear affine power law model, which is a fit of steady state data to a selected form with linear, nonlinear and constant terms.

Model 3B: The straight linear affine model, generated in the same manner as 3A, without the non-linear terms, to serve as a comparison.

Model 4: The Quasi-Hermite interpolation model. Also a fit to steady state data, this model employs value and derivative matching over a two dimensional subset of the state space.

Models 3 and 4 will be outlined briefly here, and detailed in later chapters.

2.4 Linear Affine Power Law Model [2]

This model approximates the system by interpolating values of $A(x,u)$ from values of the matrix at two data points, and by generating values, for $g(u)$ by a fit of the form:

$$g_i(u) = c_{1i}u_1 + c_{2i}u_2 + c_{5i}u_1^{c_{3i}}u_2^{c_{4i}} + c_{6i} \quad , \quad i = 1,\ldots,5 \qquad (2.4\text{-}1)$$

to the same two data points.

2.5 Quasi-Hermite Interpolation Model

This model approximates the system by interpolating values of $A(x,u)$ from values of the matrix at five data points, and by interpolating values of $g(u)$ from 15 data points using a two dimensional adaptation of Hermite interpolation. This model represents new work for this thesis.

2.6 Summary

Having chosen to express the model in the form:

$$\dot{x} = A(x,u)[x - g(u)] \tag{2.2-3}$$

it remains to derive the function $g(u)$ and the matrix $A(x,u)$ to correspond to empirical data. The function $g(u)$ represents a mapping from the control space U into the state space X which yields steady state values for given controls. Empirical data available (i.e. DYGABCD output) includes both steady state values, and derivatives at those points with respect to the various inputs. It is desirable to choose functions which match as many of the available data as possible.

CHAPTER III

LINEAR AFFINE POWER LAW MODEL 3

3.1 Formation of $A(x,u)$ [2]

Values of $A(x,u)$ are interpolated from the values of $A_D = A(x_D,u_D)$ and $A_W = A(x_W,u_W)$ in the following manner:

$$A(x,u) = A_W \,\mathrm{diag}\left(\frac{x_{Dj} - x_j}{x_{Dj} - x_{Wj}}\right) + A_D \,\mathrm{diag}\left(\frac{x_j - x_{Wj}}{x_{D_j} - x_{Wj}}\right) \tag{3.1-1}$$

where diag(·) is a diagonal matrix which causes the j^{th} column of $A(x)$ to be interpolated linearly between the jth columns of A_W and A_D with x_j as the interpolation variable.

3.2 Approximation of $g(u)$ [2]

The parameter vector u is presumed to be made up of physical control variables, and parameters such as fuel flow and nozzle area. The equilibrium function is to be approximated in a manner such that both the equilibrium values and the linearizations of the approximating system (3) match those of system (1) at both x_D and x_W. This requires then that

$$g(u_D) = x_D \qquad g(u_W) = x_W \tag{3.2-1}$$

and also

$$\frac{\delta g}{\delta u}(u_D) = -A_D^{-1}B_D \quad , \quad \frac{\delta g}{\delta u}(u_W) = -A_W^{-1}B_W \ . \tag{3.2-2}$$

The method we propose here is to approximate each scalar component $g_i(u)$ of $g(u)$ by a <u>linear affine power law</u> form

$$g(u) = c_1u_1 + \ldots + c_mu_m + c_{2m+1}u_1^{c_{m+1}} u_2^{c_{m+2}} \ldots u_m^{c_{m+m}} + c_{2m+2} \tag{3.2-3}$$

for which the jth partial derivative is

$$\frac{\delta g_i}{\delta u_j} = c_j + c_{2m+1}c_{m+j}\,\frac{u_1^{c_{m+1}}\cdots u_m^{c_{m+m}}}{u_j}\,. \qquad (3.2\text{-}4)$$

Now, if the variables are normalized and scaled such that

$$u_D = (1,1,\ldots,1) = \underline{1} \qquad u_W = (a,a,\ldots,a) = \underline{a} \qquad (3.2\text{-}5)$$

then, the conditions of (11) and (12) can be put in the form

$$\begin{aligned}
k_j &= \frac{\delta g_i}{\delta u_j}(\underline{1}) = c_j + c_{2m+1}c_{m+j} \\
k_{m+j} &= \frac{\delta g_i}{\delta u_j}(\underline{a}) = c_j + c_{2m+1}c_{m+j}a^{\sum c_{m+j}-1} \\
k_{2m+1} &= g_i(\underline{1}) = \sum c_j + c_{2m+1} + c_{2m+2} \\
k_{2m+2} &= g_i(a) = a\sum c_j + c_{2m+1}a^{\sum c_{m+j}} + c_{2m+2}
\end{aligned} \qquad (3.2\text{-}6)$$

and summing the first two of these over j yields

$$\begin{aligned}
\sum k_j &= \sum c_j + c_{2m+1}\sum c_{m+j} \\
\sum k_{m+j} &= \sum c_j + c_{2m+1}a^{\sum c_{m+j}-1}\sum c_{m+j} \\
k_{2m+1} &= \sum c_j + c_{2m+1} + c_{2m+2} \\
k_{2m+2} &= a\sum c_j + c_{2m+1}a^{\sum c_{m+j}} + c_{2m+2}
\end{aligned} \qquad (3.2\text{-}7)$$

which is of the form

$$\begin{aligned}
s_1 &= r_1 + r_3r_2 \\
s_2 &= r_1 + r_3r_2a^{r_2-1} \\
s_3 &= r_1 + r_3 + r_4 \\
s_4 &= ar_1 + r_3a^{r_2} + r_4
\end{aligned} \qquad (3.2\text{-}8)$$

which, incidentally is the $m=1$ condition also. This set of transcendental equations is solved numerically for r_1, r_2, r_3, r_4 and (3.2-6) is then used to solve for each c_j. In the event that (3.2-8) has no solution, a best fit is made on the second equation by varying r_2 while the other conditions are satisfied exactly.

3.3 Computational Algorithm [2]

In this section, we present an algorithm which serves to automate the process of finding a nonlinear model for a system

$$\dot{x} = f(x,u) \tag{3.3-1}$$

to be approximated from $x_D, u_D, x_W, u_W, A_D, B_D, A_W, B_W$, by a normalized system. The algorithm will automatically perform the normalization and, hence, actually approximate the system

$$\dot{\hat{x}} = \hat{f}(\hat{x},\hat{u}) \tag{3.3-2}$$

where $\hat{x}_i = x_i/x_{D_i}$, $\hat{u}_j = u_j/u_{D_j}$. The approximating system is of the form

$$\dot{\hat{x}} = \hat{A}(\hat{x})[\hat{x}-\hat{g}(\hat{u})] \tag{3.3-3}$$

where

$$\hat{A}(\hat{x}) = \hat{A}_W \operatorname{diag} \frac{\hat{x}_{D_i} - \hat{x}_i}{\hat{x}_{D_i} - \hat{x}_{W_i}} + \hat{A}_D \operatorname{diag} \frac{\hat{x}_i - \hat{x}_{W_i}}{\hat{x}_{D_i} - \hat{x}_{W_i}} \tag{3.3-4}$$

and

$$\hat{g}_i = \sum_j c_j^i u_j^* + c_{2m+1}^i \prod_j u_j^{*\,c_{m+j}^i} + c_{2m+2}^i \tag{3.3-5}$$

where

$$u_j^* = \alpha_j \hat{u}_j + \beta_j . \tag{3.3-6}$$

Algorithm I.

1. Input: $x_D, u_D, A_D, B_D, m, n, a, \varepsilon, x_W, u_W, A_W, B_W$

   where m = number of controls

   n = number of states

2. Calculate:

$$\hat{A}_D = \mathrm{diag}(1/x_{D_i})\ A_D \mathrm{diag}(x_{D_i})$$
$$\hat{A}_W = \mathrm{diag}(1/x_{D_i})\ A_W \mathrm{diag}(x_{D_i})$$
$$\hat{B}_D = \mathrm{diag}(1/x_{D_i})\ B_D \mathrm{diag}(u_{D_i})$$
$$\hat{B}_W = \mathrm{diag}(1/x_{D_i})\ B_W \mathrm{diag}(u_{D_i})$$

3. Calculate:

$$\alpha_j = (1-a)u_{D_i}/(u_{D_j}-u_{W_j}) \qquad j = 1,\ldots,m$$
$$\beta_j = (au_{D_j}-u_{W_j})/(u_{D_j}-u_{W_j})$$

4. Calculate:

$$k^i_j = (-\hat{A}_D^{-1}\hat{B}_D)_{ij} \qquad k^i_{2m+1} = 1 = \hat{x}_{D_i} \qquad j = 1,\ldots,m$$
$$k^i_{m+j} = (-\hat{A}_W^{-1}\hat{B}_W)_{ij} \qquad k^i_{2m+2} = x_{W_i}/x_{D_i} = \hat{x}_{W_i} \qquad i = 1,\ldots,n$$

5. Calculate:

$$s^i_1 = \sum_{j=1}^{m} \qquad s^i_2 = \sum_{j=1}^{m} k^i_{m+j}$$
$$i = 1,\ldots,n$$
$$s^i_2 = k^i_{2m+1} \qquad s^i_4 = k^i_{2m+2}$$

6. Go to Algorithm II.

Send: $s^i_1,\ s^i_2,\ s^i_3,\ s^i_4,\ a,\ \varepsilon$ $\qquad i = 1,\ldots,n$

Receive: $r^i_1,\ r^i_2,\ r^i_3,\ r^i_4,\ \gamma$

7. Calculate:

$$c^i_{2m+1} = r^i_3 \qquad c^i_{2m+2} = r^i_4 \qquad j = 1,\ldots,m$$

$$c_{m+j}^{i} = \frac{k_{m+j}^{i} - k_{j}^{i} + \gamma}{r_{j}^{i}(a^{r_2^i - 1} - 1)} \qquad c_{j}^{i} = k_{j}^{i} - r_{3}^{i}\, c_{m+j}^{i} \qquad i = 1,\ldots,n$$

8. Output:

$c_1^i, \ldots, c_{2m+2}^i$

α_i, β_j $\qquad j = 1,\ldots,m$

$\hat{x}_{D_i}, \hat{x}_{W_i}$ $\qquad i = 1,\ldots,n$

$\hat{A}_D, \hat{A}_W$

Algorithm II.

1. Input: $s_1, s_2, s_3, s_4, \varepsilon, a$

2. Calculate:

$$p_1 = s_1 - \frac{s_3 - s_4}{1-a}$$

$$p_2 = s_2 - \frac{s_3 - s_4}{1-a}$$

3. Minimize by line search:

$$\left| p_2 - p_1 \frac{ax^{x-1} - \frac{a^x - 1}{a-1}}{x - \frac{a^x - 1}{a-1}} \right|$$

for $-10 \le x \le 10, \qquad x \ne 0, \qquad x \ne 1$

4. Calculate:

$$r_2 = x \qquad r_3 = \frac{p_1}{r_2 - \frac{a^{r_2} - 1}{a-1}}$$

$$r_1 = \frac{s_3 - s_4 + r_3(a^{r_2} - a)}{1-a} \qquad \gamma = \frac{1}{m}\left(s_1 - s_2 + r_2 r_3 (a^{r_2 - 1} - 1)\right)$$

$$r_2 = \frac{s_4 - as_3 - r_3(a^{r_2} - a)}{1-a}$$

5. Return to Algorithm I.6

3.4 Straight Linear Affine Model

As a check that the Power law term has significant effect on the function $g(u)$, a straight linear affine approximation to $g(u)$ was generated. This model is then subjected to the same analysis as models 3A and 4.

3.5 Numerical Results [2]

The algorithm of the previous section was applied to data obtained using DYNGEN with x_D and u_D specified as in Section 2. An off-design point was obtained using $u_W = (.72727, .72727)$, with the resulting normalized state $\hat{x}_W = (.9000, .7897, .7381, .9401, .9454)$. The normalized A and B matrices are

$$\hat{A}_D = \begin{bmatrix} -3.8 & -1.277 & 2.067 & -1.152 & 1.448 \\ 2.748 & -5.39 & 1.585 & -1.991 & 1.071 \\ 377.9 & 49.51 & -264.9 & 86.807 & 78.91 \\ 31.26 & 139.39 & -6.269 & -88.69 & 27.83 \\ -176.5 & 23.91 & -10.27 & -37.4 & -246.7 \end{bmatrix} \quad \hat{B}_W = \begin{bmatrix} -.00259 & .3553 \\ .2116 & -.31618 \\ 12.54 & -13.774 \\ -.6201 & -99.3 \\ 157.78 & 6.84 \end{bmatrix} \tag{3.5-1}$$

$$\hat{A}_W = \begin{bmatrix} -4.744 & -1.3888 & 3.2468 & -1.4591 & 1.1969 \\ .82.86 & -26.726 & 2.5585 & -1.8609 & .45548 \\ 475.73 & 137.55 & -328.91 & 27.791 & 91.495 \\ -50.103 & 110.91 & 63.188 & -116.69 & 8.2883 \\ -186.77 & -67.682 & -41.681 & 24.586 & -243.23 \end{bmatrix} \quad \hat{B}_W = \begin{bmatrix} -.04546 & .0013 \\ .0086 & -.0121 \\ 2.434 & -.613 \\ .67865 & -97.467 \\ 203.44 & .64755 \end{bmatrix} \tag{3.5-2}$$

Using the parameter value $a = .7$, the c_j^i coefficients which specify the equilibrium functions $\hat{g}(\hat{u})^i$ as in Section 3 are given by the matrix .

$$C = \begin{bmatrix} .24267 & -.00218 & 1.90082 & 8.09916 & .02864 & .73088 \\ 1.01593 & .85407 & .89872 & .66919 & -.81879 & -.05121 \\ .73445 & .10133 & 6.90586 & 3.09409 & .011495 & .15272 \\ .77234 & -.35905 & 2.45867 & 2.87415 & -075198 & .66191 \\ .39503 & -.27262 & -3.44682 & 13.4468 & .01838 & .85921 \end{bmatrix} \qquad (3.5\text{-}3)$$

This matrix together with the values $\alpha=1.1$ and $\beta=0.1$ and the matrices $\hat{A}_D$ and $\hat{A}_W$ completely specify Model 3A.

Another model which we will call Model 3B is easily obtained by using a <u>linear</u> <u>affine</u> approximation to $\hat{g}(\hat{u})$ such that $\hat{g}(\hat{u}_D) = \hat{x}_D$, $\hat{g}(\hat{u}_W) = \hat{x}_W$. Model 3B is specified by $a = e^{-1}$, $\alpha = 2.31778$, $\beta = -1.31778$ and the coefficient matrix

$$C = \begin{bmatrix} .1553 & .0028 & 1.0 & 1.0 & 0. & .8418 \\ .1619 & .1707 & 1.0 & 1.0 & 0. & .6674 \\ .5351 & -.1208 & 1.0 & 1.0 & 0. & .5857 \\ .5878 & -.49313 & 1.0 & 1.0 & 0. & .9053 \\ .2962 & -.2099 & 1.0 & 1.0 & 0. & .9137 \end{bmatrix} \qquad (3.5\text{-}4)$$

CHAPTER IV

QUASI-HERMITE INTERPOLATION MODEL 4

4.1 Introduction

A known method for matching a polynomial to the values and derivatives of a function at several points is Hermite interpolation. However, this method is formulated in general only for the one dimensional case. [3] Some works exist which apply this method to an n dimensional case, but only under certain narrow restrictions. [5] The single variable case, its restricted application in n dimensions, and a modified application in two dimensions, are discussed in this chapter.

4.2 Hermite Interpolation for a Single Variable

This presentation of the Hermite interpolation method is drawn from Hildebrand. [3] His notation is preserved, as closely as possible, here and in the resultant computer program.

If the values of $g(u)$ are known at m points, $u = u^1, u^2, \ldots, u^m$, define:

$$\pi(u) = (u - u^1)(u - u^2)\ldots(u - u^m) \tag{4.2.1}$$

and:

$$\ell^i(u) = \frac{\pi(u)}{(u-u^i)\ \pi'(u^i)} \tag{4.2.2}$$

with the properties:

$$\pi(u^j) = 0 \qquad j = 1,\ldots,m \tag{4.2.3}$$

and:

$$\ell^i(u^j) = \delta_{ij} \qquad i = 1,\ldots,m \qquad j = 1,\ldots,m \tag{4.2.4}$$

where δ_{ij} is the Kronecker delta ($\delta_{ii} = 1$, $\delta_{ij} = 0$ for all $i \neq j$).

With these defined, the polynomial of degree $m-1$ which takes on the values $g(u^1), g(u^2),\ldots,g(u^m)$ can be expressed as:

$$y(u) = \sum_{k=1}^{m} \ell^k(u)g(u^k) \tag{4.2.5}$$

Suppose both $g(u)$ and $g'(u)$ are known for $u = u^1, u^2,\ldots,u^m$, it is possible to determine a polynomial of degree $2m-1$ with these values and derivatives. We shall assume this polynomial is expressible in the form:

$$y(u) = \sum_{k=1}^{m} h^k(u)\ g(u^k) + \sum_{k=1}^{m} \bar{h}^k(u)\ g'(u^k) \tag{4.2.6}$$

where $h^i(u)$ and $\bar{h}^i(u)$, $i = 1,\ldots,m$ are polynomials of maximum degree $2m-1$. The requirement $y(u^j) = g(u^j)$ will be satisfied if:

$$h^i(u^j) = \delta_{ij} \quad \text{and} \quad \bar{h}^i(u^j) = 0 \tag{4.2.7}$$

and the requirement $y'(u^j) = g'(u^j)$ will be satisfied if:

$$h'^i(u^j) = 0 \quad \text{and} \quad \bar{h}'^i(u^j) = \delta_{ij} \tag{4.2.8}$$

Since $\ell^i(u)$ is a polynomial of degree $m-1$ which satisfied (4.2.4), then $[\ell^i(u)]^2$ is a polynomial of degree $2m-2$ which satisfies (4.2.4) and whose derivative is zero at u_i^j when $i \neq j$. So if $h^i(u)$ and $\bar{h}^i(u)$ are polynomials of degree $2m-1$, then:

$$h^i(u) = r^i(u)[\ell^i(u)]^2 \quad \text{and} \quad \bar{h}^i(u) = s^i(u)[\ell^i(u)]^2 \tag{4.2.9}$$

where $r^i(u)$ and $s^i(u)$ are linear functions of u, so that (4.2.7) and (4.2.8) will be satisfied when $i \neq j$. These four conditions, when $i = j$, then yield:

$$r^i(u^i) = 1 \quad r'^i(u^i) + 2\,\ell'^i(u^i) = 0 \tag{4.2.10}$$

$$s^i(u^i) = 0 \quad s'^i(u^i) = 1 \tag{4.2.11}$$

from which follows:

$$r^i(u) = 1-2\,\ell'^i(u^i)\,(u - u^i) \quad \text{and} \quad s^i(u) = (u - u^i) \tag{4.2.12}$$

So, by combining (4.2.6), (4.2.9), and (4.2.12) we obtain the desired polynomial in the form:

$$y(u) = \sum_{k=1}^{m} h^k(u)\, g(u^k) + \sum_{k=1}^{m} \bar{h}^k(u)\, g'(u^k) \tag{4.2.13}$$

where:

$$h^i(u) = r^i(u)[\ell^i(u)]^2 \quad \text{and} \quad \bar{h}^i(u) = s^i(u)[\ell^i(u)]^2 \tag{4.2.14}$$

and:

$$r^i(u) = 1-2\,\ell'^i(u^i)(u - u^i) \quad \text{and} \quad s^i(u) = (u - u^i) \,. \tag{4.2.15}$$

This result is known as Hermite's interpolation formula, or the formula for osculating interpolation.

4.3 Problems of Hermite Interpolation in n Dimensions

If Hermite Interpolation were to be applied in n dimensions, the task would be to determine m sets of $n+1$ polynomials with properties similar to h and $\bar{h}$. Specifically, if we assume that the desired polynomial can be expressed in the form:

$$y(u) = \sum_{k=1}^{m} h^k(u)\, g(u^k) + \sum_{j=1}^{n} \sum_{k=1}^{m} \bar{h}^{jk}(u)\, \frac{dg}{du^j}\,(u^k) \tag{4.3.1}$$

then these polynomials must have the properties:

$$h^i(u^j) = \delta_{ij} \quad \text{and} \quad \bar{h}^{ik}(u^j) = 0 \tag{4.3.2}$$

and:

$$\frac{\delta h^{i}}{\delta u_{j}}(u^{k}) = 0 \quad \text{and} \quad \frac{\delta \bar{h}^{jk}}{\delta u_{j}}(u^{k}) = \delta_{jk} \qquad (4.3.3)$$

corresponding to the conditions (4.2.7) and (4.2.8). However, the further condition:

$$\frac{\delta \bar{h}^{i\ell}}{\delta u_{j}}(u^{k}) = 0 \quad \text{for all} \quad i \neq k \qquad (4.3.4)$$

must also be satisfied. This final condition cannot be satisfied by the polynomials described in the previous section. In [5], Niijima treats a special case, in which the existence of certain orthogonal polynomials allows the application of Hermite interpolation to carefully chosen data in two dimensions. However, this method is not universally applicable to arbitrary data.

4.4 Quasi-Hermite Approach for Two Controls

Given that no general method of Hermite interpolation in two variables was found, the following adaptation of the one dimensional case was applied. The value of control u_2 (nozzle area) was held constant at the design point value, and Hermite interpolation was applied to a set of points generated by varying u_1 (fuel flow). Both values, and derivatives with respect to u_1 were matched at these data points. Then, for each value of u_1, a value Δ was chosen, and control u_2 was varied by this amount, both plus and minus. A function was then chosen to match values at these new points, without altering the function at the original points. The resulting polynomial is of the form:

$$y(u) = \sum_{k=1}^{m} h^{k}(u_1)\, g(u_1^{k},u_2^{1})\, \theta^{k}(u_2) + \sum_{k=1}^{m} \bar{h}^{k}(u_1)\, g'(u_1^{k},u_2^{1}) \qquad . \; (4.4.1)$$

where:

$$\theta^k(u_2) = 1 + [\frac{g(u_1^k,\ u_2^1 + \Delta^k) - g(u_1^k,\ u_2^1 - \Delta^k)}{2\ \Delta^k\ g(u_1^k,u_2^1)}]\ u_2$$

$$+ [\frac{g(u_1^k,u_2^1 + \Delta^k) + g(u_1^k,u_2^1 - \Delta^k) - 2g(u_1^k,u_2^1)}{2(\Delta^k)^2\ g(u_1^k,u_2^1)}]\ (u_2)^2 \qquad (4.4.2)$$

This function has the property that:

$$\theta^k(u_2) = 1 \quad \text{when} \quad u_2 = u_2^1 \qquad (4.4.3)$$

and:

$$\theta^k(u_2) = g(u_1^k,u_2^1 + \Delta^k)/g(u^k) \quad \text{when} \quad u_2 = u_2^1 + \Delta^k$$

$$\theta^k(u_2) = g(u_1^k,u_2^1 - \Delta^k)/g(u^k) \quad \text{when} \quad u_2 = u_2^1 - \Delta^k \qquad (4.4.4)$$

and since $h^i(u^j) = \delta_{ij}$, the resultant polynomial will match values and derivatives with respect to u_1 along the $u_2 = u_2^1$ line, and values at $u_2 = u_2^1 \pm \Delta^k$.

4.5 Formation of $A(x,u)$

Having chosen an approximation to $g(u)$, it remains to choose a method for interpolating values for $A(x,u)$ to complete the model. Lagrangian interpolation was used to match values only at three points along the $u_2 = u_2^1$ line, and at $u = (u_1^1,\ u_2^1 \pm \Delta^1)$. The results of this approach are embodied in the following equations. First define:

$$A1 = A(x,u) \quad \text{at} \quad u = u^1 = (u_1^1,u_2^1), \qquad x = g(u^1)$$

$$A3 = A(x,u) \quad \text{at} \quad u = u^3 = (u_1^3,u_2^3), \qquad x = g(u^3)$$

$$A5 = A(x,u) \quad \text{at} \quad u = u^5 = (u_1^5,u_2^5), \qquad x = g(u^5) \qquad (4.5.1)$$

$$AP = A(x,u) \quad \text{at} \quad u = u^P = (u_1^1,u_2^1 + \Delta^1), x = g(u^P)$$

$$AM = A(x,u) \quad \text{at} \quad u = u^M = (u_1^1,u_2^1 - \Delta^1), x = g(u^M)$$

and:

$$\begin{aligned}
FR1 &= [(x_1 - x_1^3)(x_1 - x_1^5)]/[(x_1^1 - x_1^3)(x_1^1 - x_1^5)] \\
FR3 &= [(x_1 - x_1^1)(x_1 - x_1^5)]/[(x_1^3 - x_1^1)(x_1^3 - x_1^5)] \\
FR5 &= [(x_1 - x_1^1)(x_1 - x_1^3)]/[(x_1^5 - x_1^1)(x_1^5 - x_1^3)] \\
FRP &= [(u_2 - u_2^1)(u_2 - (u_2^1 - \Delta^1))]/[2(\Delta^1)^2] \\
FR &= [(u_2 - (u_2^1 + \Delta^i))(u_2 - (u_2^1 - \Delta^1)]/[(-(\Delta^1)^2] \\
FRM &= [(u_2 - (u_2^1 + \Delta))(u_2 - u_2^1)]/[2(\Delta^1)^2]
\end{aligned} \tag{4.5.2}$$

then:

$$\begin{aligned}
A_{ij}(x,u) = {} & [FR1(A1_{ij}) + FR3(A3_{ij}) + FR5(A5_{ij})] \\
& x\ [FRP(AP_{ij}/A1_{ij}) + FR + FRM(AM_{ij}/A1_{ij})]
\end{aligned} \tag{4.5.3}$$

CHAPTER V

MODEL RESPONSE COMPARISONS

5.1 Introduction

Before subjecting the models to the Dynamic Programming Algorithm, some effort was made to examine their closeness of fit to DYNGEN data. Steady state values of models 1 and 4 are compared, and fuel flow step responses of models 3A, 3B, and 4 are plotted in comparison to DYNGEN responses. A description of the step response program is also included.

5.2 Steady State Comparison of Models 1 and 4

The function $g(u)$ represents a mapping from the control space into the state space, relating fixed controls to steady states. It is not only useful in the model form:

$$\dot{x} = A(x,u)[x - g(u)] \tag{5.2-1}$$

but should also approximate the operating line of the plant.

Such a comparison is made here between $g(u)$ for model 4 and the DYNGEN simulator. Nozzle area was held constant, as fuel flow was varied from 9.0 to 1.1 by 0.02. All values are normalized. Percentage error is also computed, and shows the model's excellent agreement in its range of accuracy, and rapid deterioration outside that range.

Table 5.2-1

X(1) = NC

fuel flow	DYNGEN	Model 4	% error
0.90	.97275	.97288	0.01
0.92	.97761	.97790	0.03
0.94	.98326	.98326	0.00
0.96	.98887	.98892	0.01
0.98	.99445	.99461	0.02
1.00	1.0000	1.0000	0.00
1.02	1.0051	1.0051	0.00
1.04	1.0102	1.0113	0.11
1.06	1.0152	1.0230	0.77
1.08	1.0201	1.0513	3.06
1.10	1.0244	1.1195	9.28

Table 5.2-2

X(2) = NF

fuel flow	DYNGEN	Model 4	% error
0.90	.97132	.97099	-0.03
0.92	.97883	.97817	-0.07
0.94	.98427	.98425	0.00
0.96	.98961	.98948	-0.01
1.00	1.0000	1.0000	0.00
1.02	1.0046	1.0011	-0.35
1.04	1.0091	.98046	-2.84
1.06	1.0136	.89094	-12.10
1.08	1.0180	.62787	-38.32
1.10	1.0221	-.013230	-101.29

Table 5.2-3

$X(3) = P'4$

fuel flow	DYNGEN	Model 4	% error
0.90	.92038	.92042	0.00
0.92	.93623	.93633	0.01
0.94	.95225	.95225	0.00
0.96	.96821	.96824	0.00
0.98	.98413	.98434	0.02
1.00	1.0000	1.0000	0.00
1.02	1.0148	1.0125	-0.23
1.04	1.0295	1.0136	-1.54
1.06	1.0441	.98374	-5.78
1.08	1.0587	.88135	-16.75
1.10	1.0727	.62822	-41.44

Table 5.2-4

X(4) = P7

fuel flow	DYNGEN	Model 4	% error
0.90	.94118	.94109	-0.01
0.92	.95358	.95340	-0.02
0.94	.96532	.96531	0.00
0.96	.97697	.97693	0.00
0.98	.98853	.98856	0.00
1.00	1.0000	1.0000	0.00
1.02	1.0107	1.0081	-0.26
1.04	1.0213	1.0013	-1.96
1.06	1.0319	.94911	-8.02
1.08	1.0425	.78384	-24.81
1.10	1.0527	.37203	-64.66

Table 5.2-5

X(5) = U4

fuel flow	DYNGEN	Model 4	% error
0.90	.96625	.96624	0.00
0.92	.97304	.97308	0.00
0.94	.97992	.97992	0.00
0.96	.98670	.98665	-0.01
0.98	.99339	.99320	-0.02
1.00	1.0000	1.0000	0.00
1.02	1.0074	1.0089	0.15
1.04	1.0147	1.0248	1.00
1.06	1.0219	1.0573	3.46
1.08	1.0290	1.1231	9.14
1.10	1.0365	1.2494	20.54

5.3 Program Layout

The method chosen for generating time response data was a Euler integration with a user varied time step. After specifying initial controls, the user provides a control sequence of time step, duration (in iterations), and controls. This structure allows the user to provide smaller time increments for the steeper portions of the response, and to alter the controls during the response. The step response program creates a file of time-state n-tuples, which are plotted against similar DYNGEN data by another program.

5.4 Fuel Step Response for Models 3A, 3B, and 4

Each of the three models was subjected to a fuel flow step from 0.8 to 1.0, and the response plotted against the same response by DYNGEN. These graphs show that all three models match DYNGEN closely, but that model 4 is a better fit than either 3A or 3B.

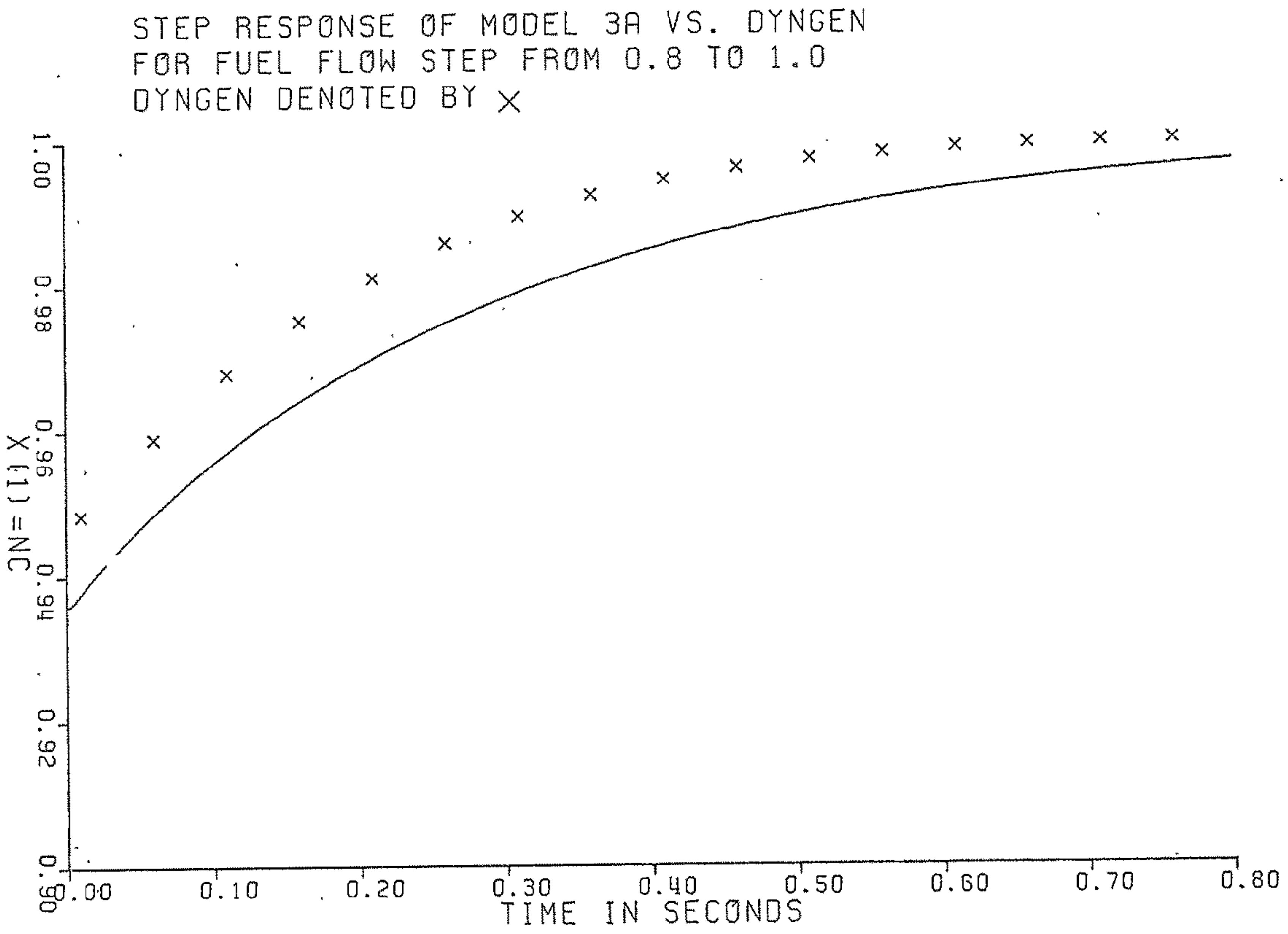

STEP RESPONSE OF MODEL 3A VS. DYNGEN
FOR FUEL FLOW STEP FROM 0.8 TO 1.0
DYNGEN DENOTED BY ×
X(1)=NC
0.90
0.92
0.94
0.96
0.98
1.00
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 5.4-1

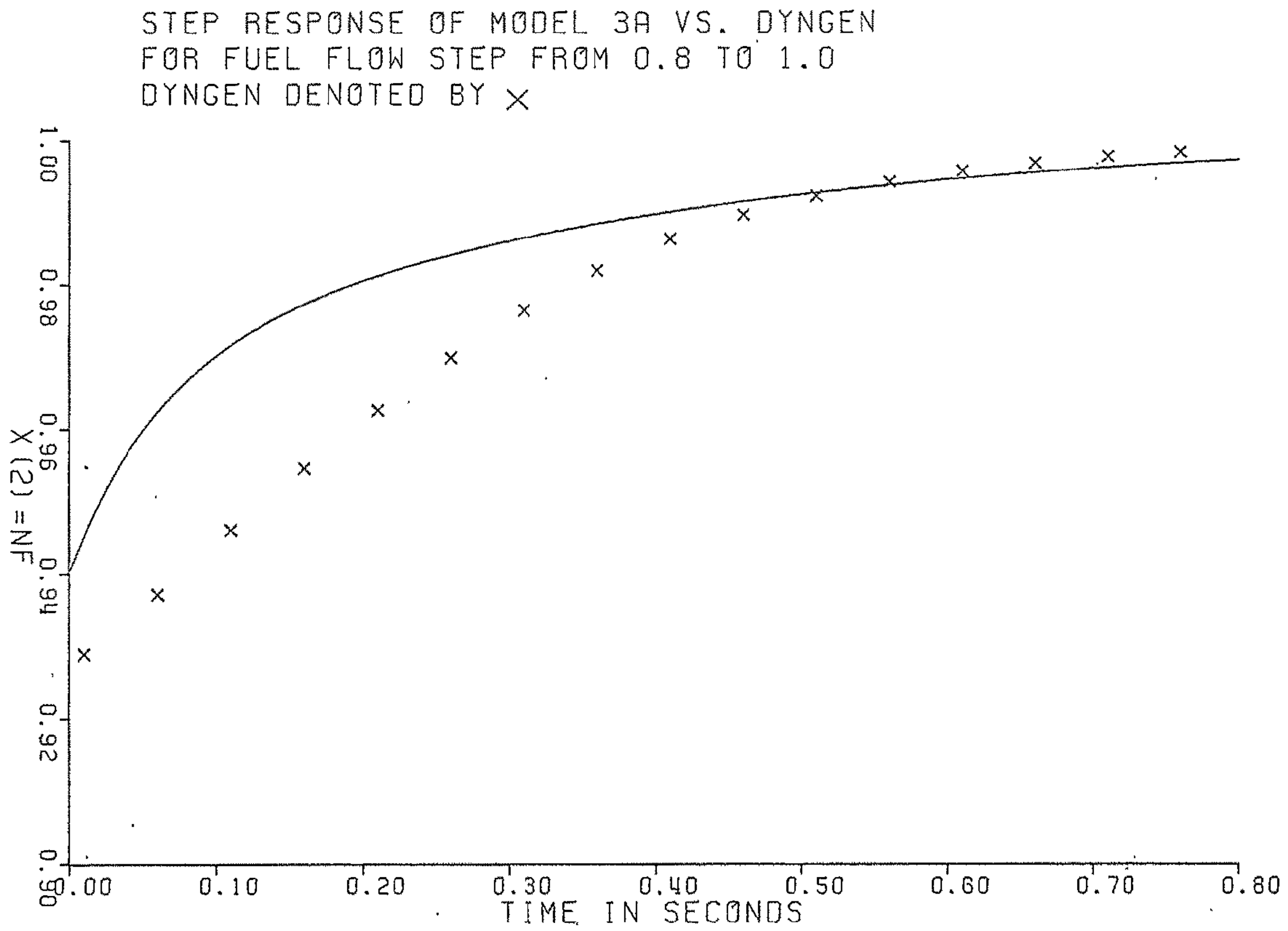

STEP RESPONSE OF MODEL 3A VS. DYNGEN
FOR FUEL FLOW STEP FROM 0.8 TO 1.0
DYNGEN DENOTED BY X
X(2)=NF
1.00
0.98
0.96
0.94
0.92
0.90
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 5.4-2

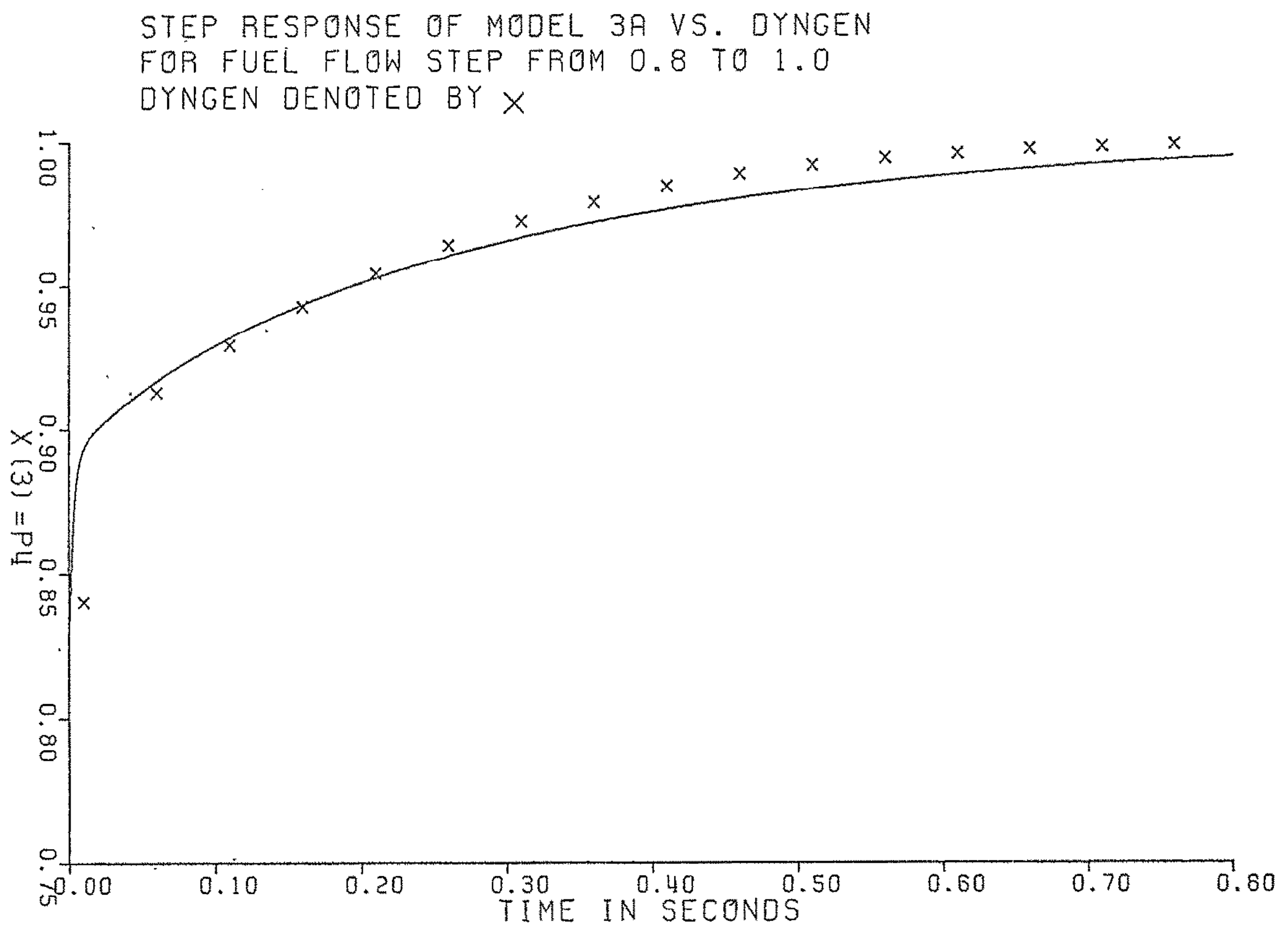

STEP RESPONSE OF MODEL 3A VS. DYNGEN
FOR FUEL FLOW STEP FROM 0.8 TO 1.0
DYNGEN DENOTED BY X
X(3)=P4
1.00
0.95
0.90
0.85
0.80
0.75
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 5.4-3

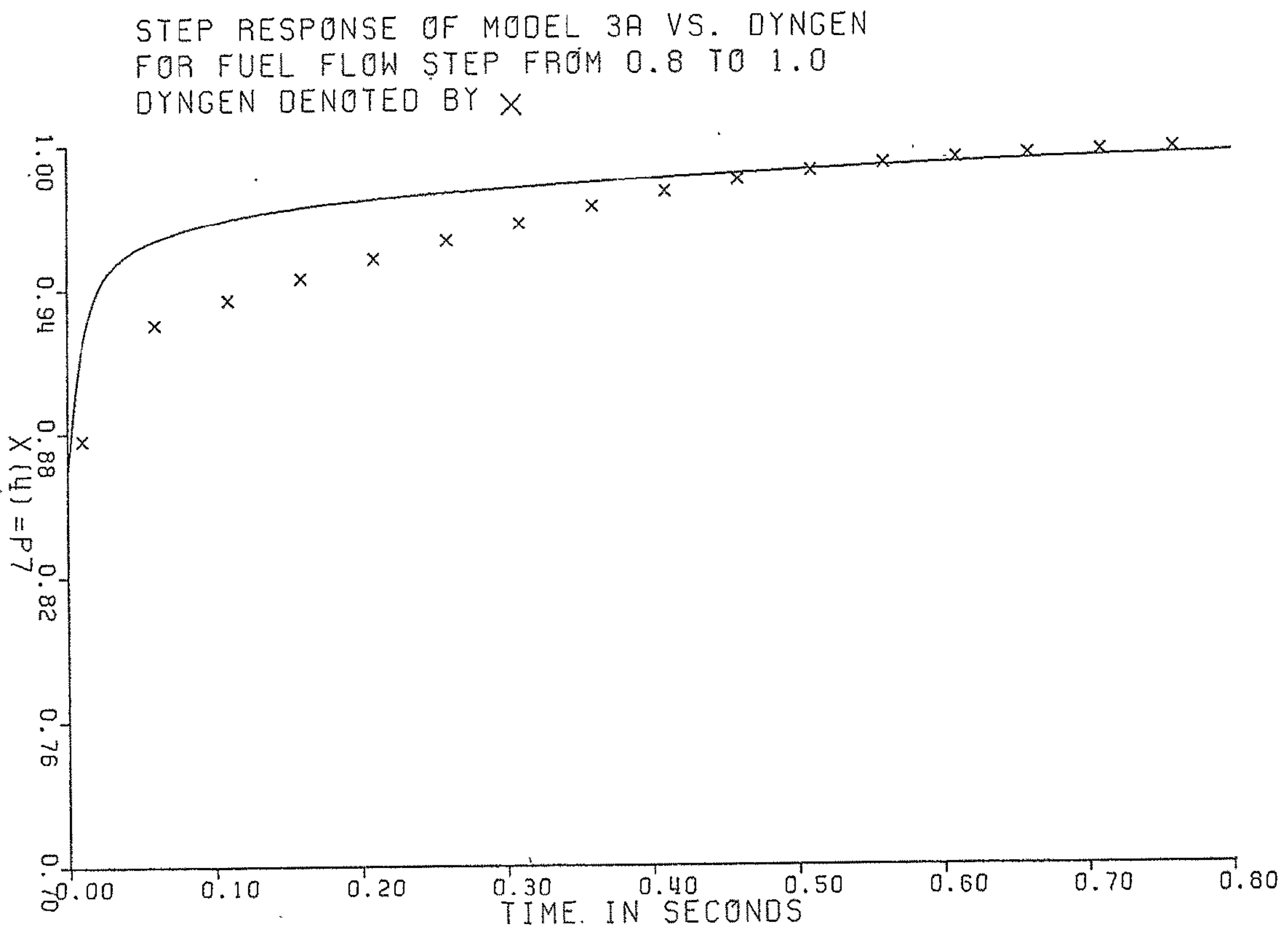
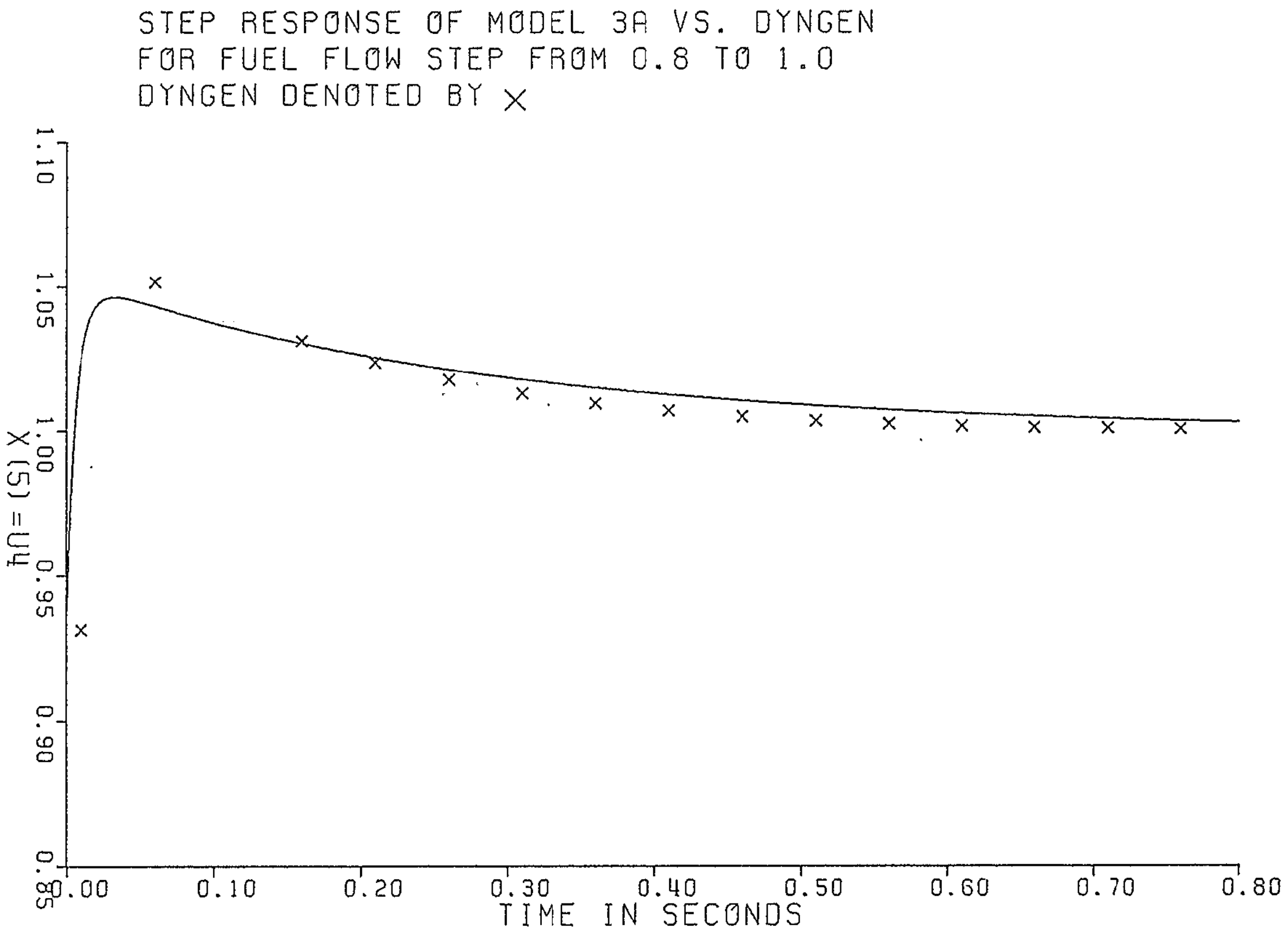
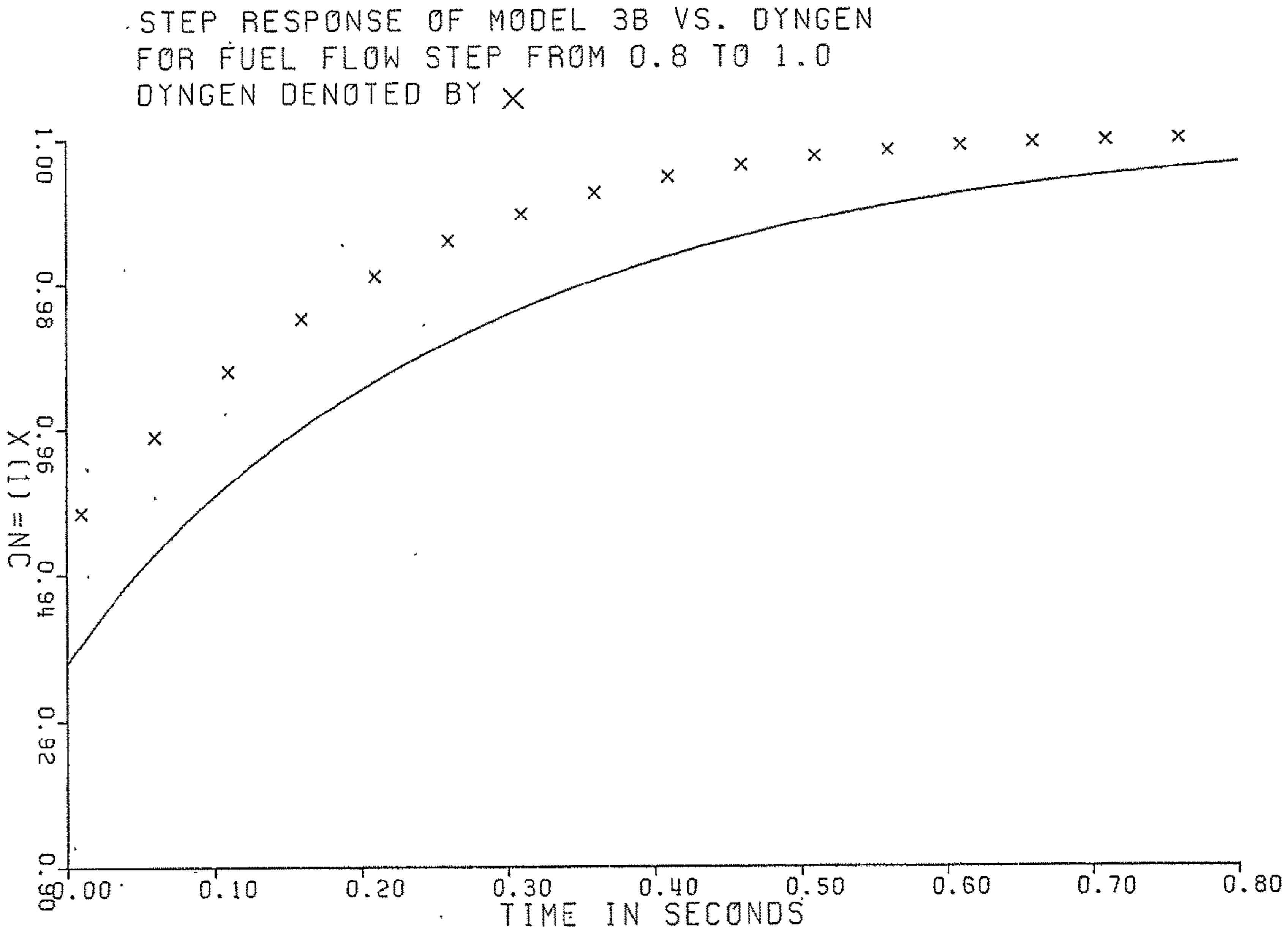

STEP RESPONSE OF MODEL 3A VS. DYNGEN
FOR FUEL FLOW STEP FROM 0.8 TO 1.0
DYNGEN DENOTED BY X
X(4)=P7
1.00
0.94
0.88
0.82
0.76
0.70
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 5.4-4



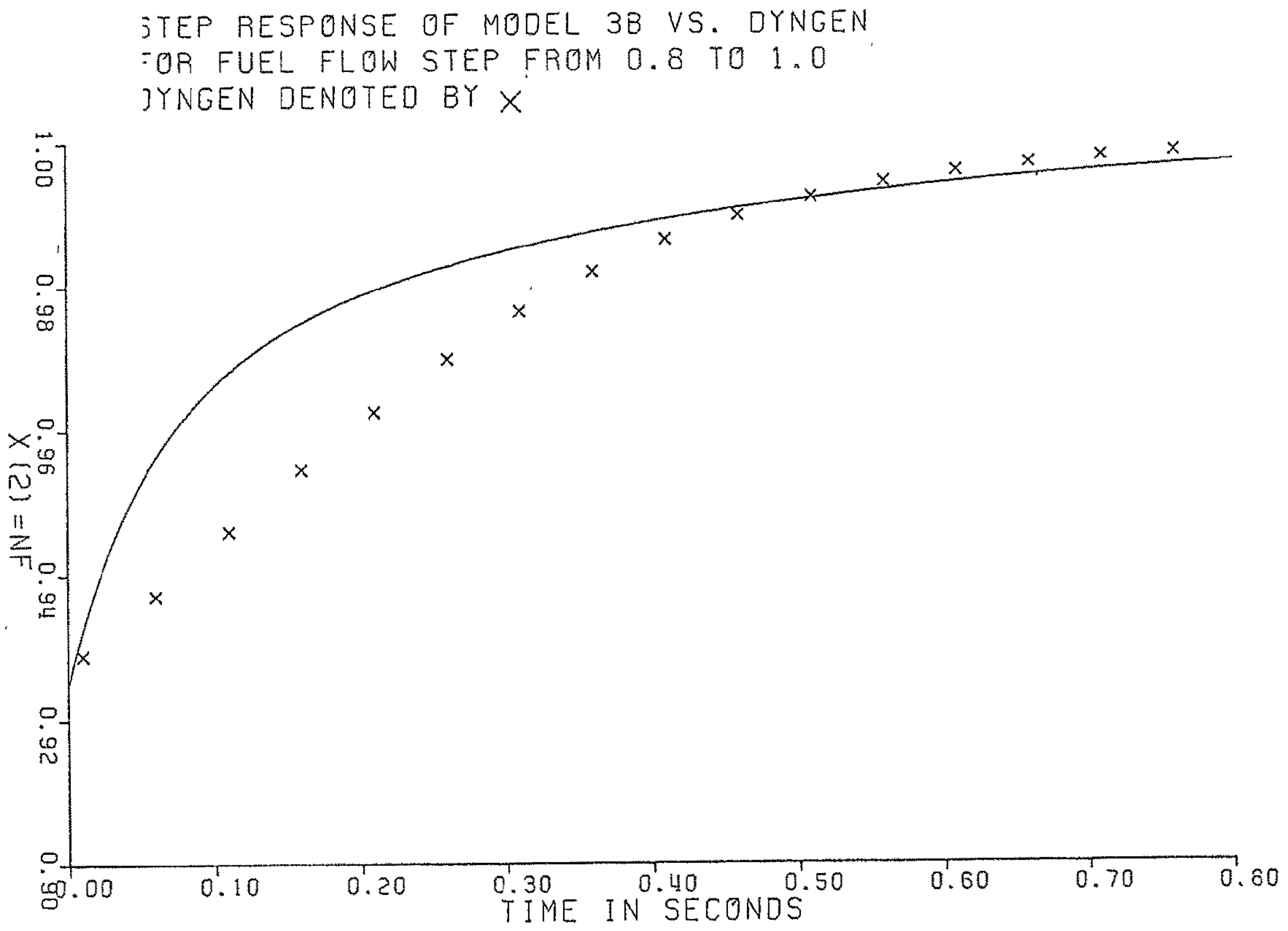
STEP RESPONSE OF MODEL 3A VS. DYNGEN
FOR FUEL FLOW STEP FROM 0.8 TO 1.0
DYNGEN DENOTED BY X
X(5)=U4
1.10
1.05
1.00
0.95
0.90
0.85
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 5.4-5



STEP RESPONSE OF MODEL 3B VS. DYNGEN
FOR FUEL FLOW STEP FROM 0.8 TO 1.0
DYNGEN DENOTED BY ×
1.00
0.98
0.96
0.94
0.92
0.90
X(1)=NC
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 5.4-6



STEP RESPONSE OF MODEL 3B VS. DYNGEN
FOR FUEL FLOW STEP FROM 0.8 TO 1.0
DYNGEN DENOTED BY ×
X (2) =NF
1.00
0.98
0.96
0.94
0.92
0.90
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 5.4-7

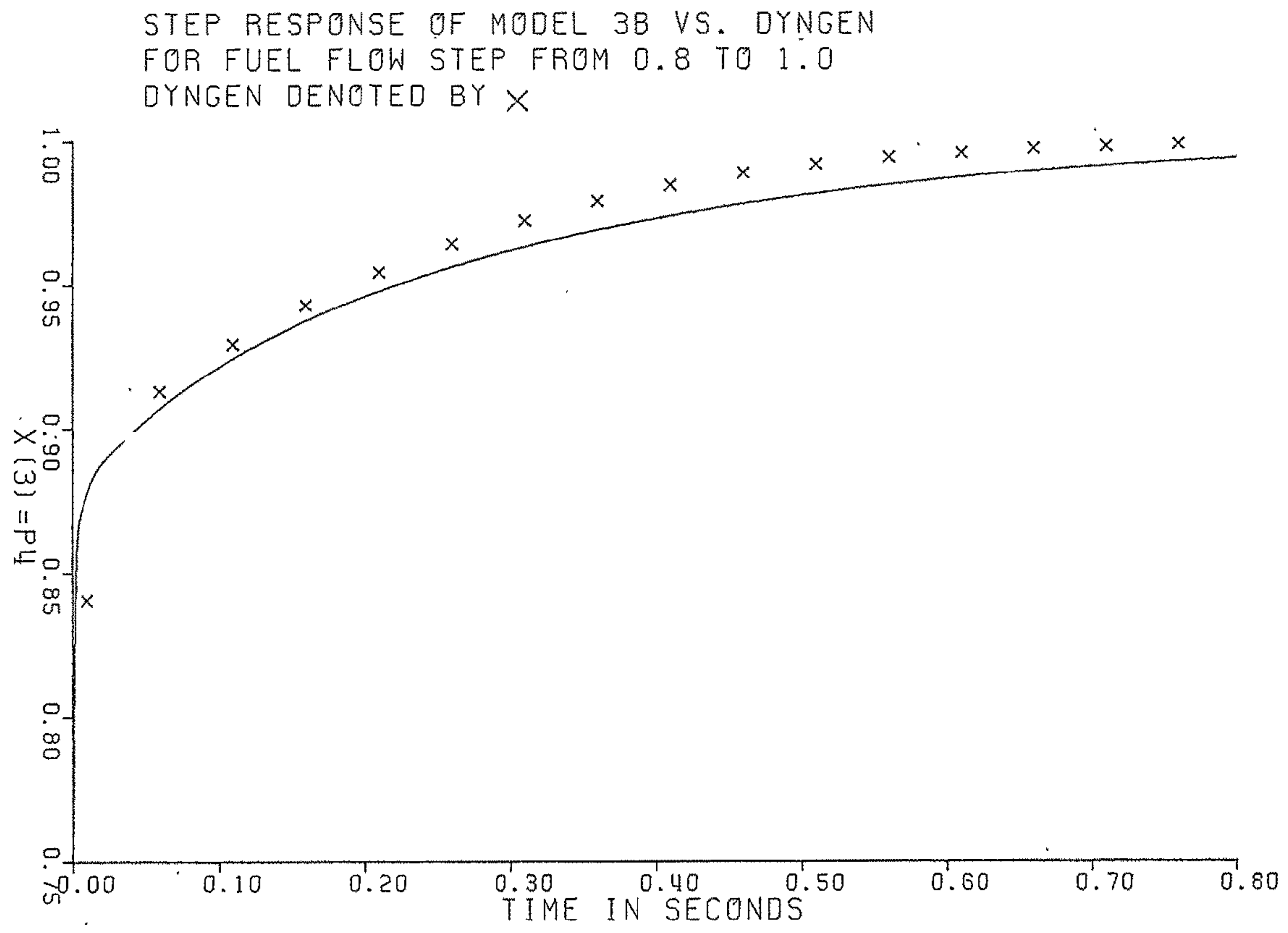

STEP RESPONSE ØF MØDEL 3B VS. DYNGEN
FØR FUEL FLØW STEP FRØM 0.8 TØ 1.0
DYNGEN DENØTED BY ×
X(3)=P4
TIME IN SECØNDS
0.75
0.80
0.85
0.90
0.95
1.00
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80

Figure 5.4-8

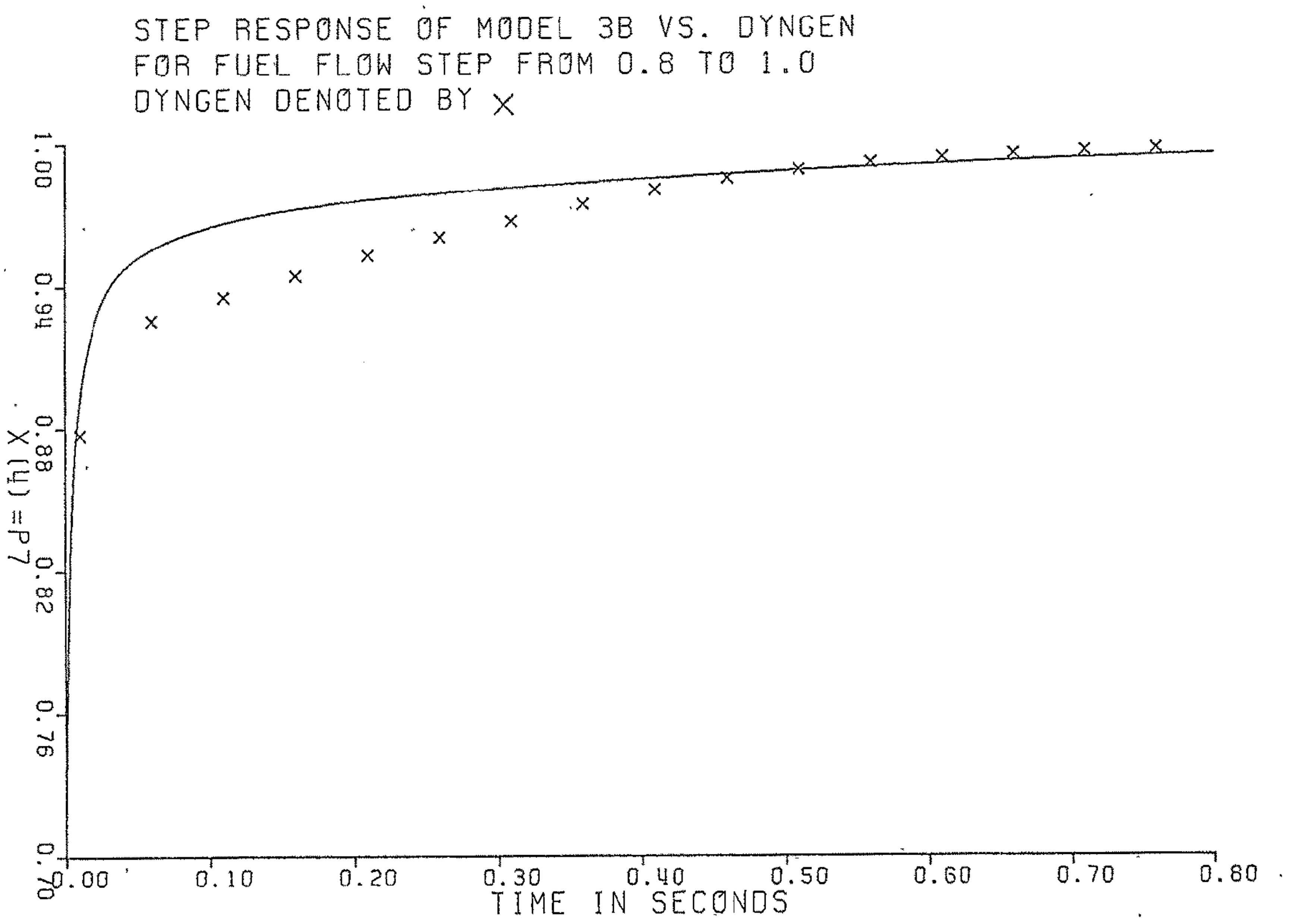

STEP RESPONSE OF MODEL 3B VS. DYNGEN
FOR FUEL FLOW STEP FROM 0.8 TO 1.0
DYNGEN DENOTED BY X
X(4)=P7
1.00
0.94
0.88
0.82
0.76
0.70
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 5.4-9

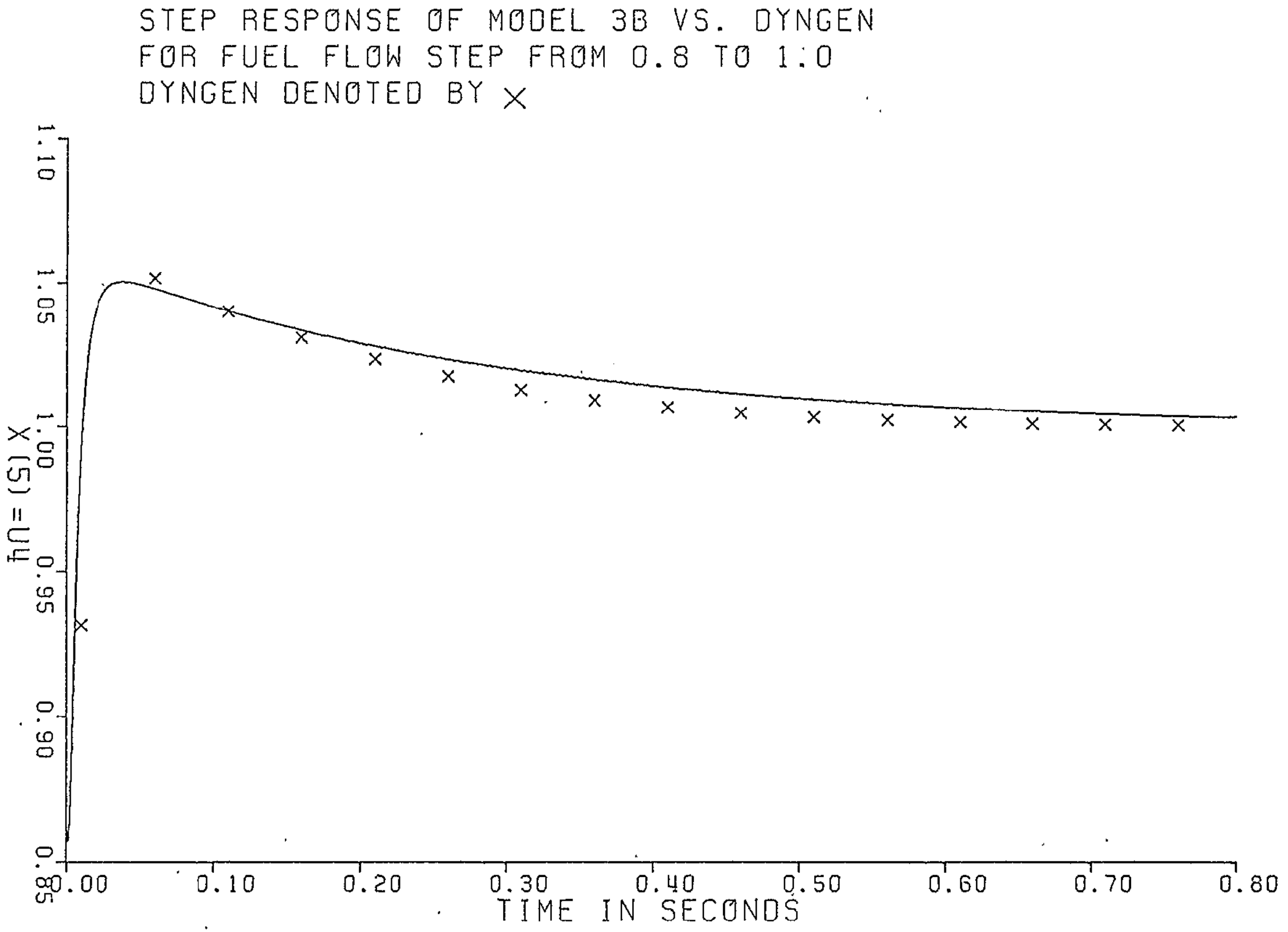

STEP RESPONSE OF MODEL 3B VS. DYNGEN
FOR FUEL FLOW STEP FROM 0.8 TO 1.0
DYNGEN DENOTED BY ×
X(5)=U4
1.10
1.05
1.00
0.95
0.90
0.85
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 5.4-10

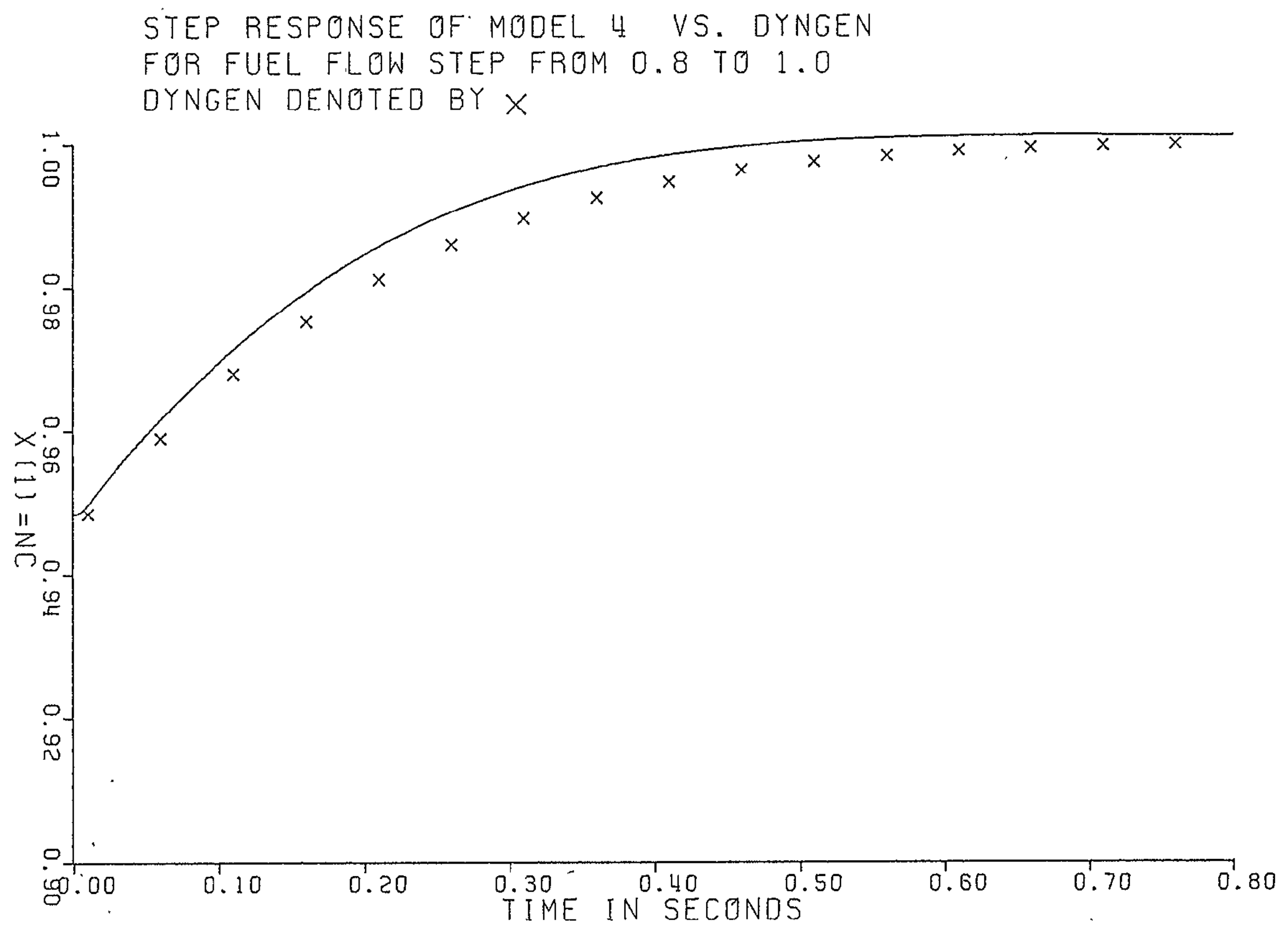

STEP RESPONSE OF MODEL 4 VS. DYNGEN
FOR FUEL FLOW STEP FROM 0.8 TO 1.0
DYNGEN DENOTED BY ×
X(1)=NC
0.90
0.92
0.94
0.96
0.98
1.00
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 5.4-11

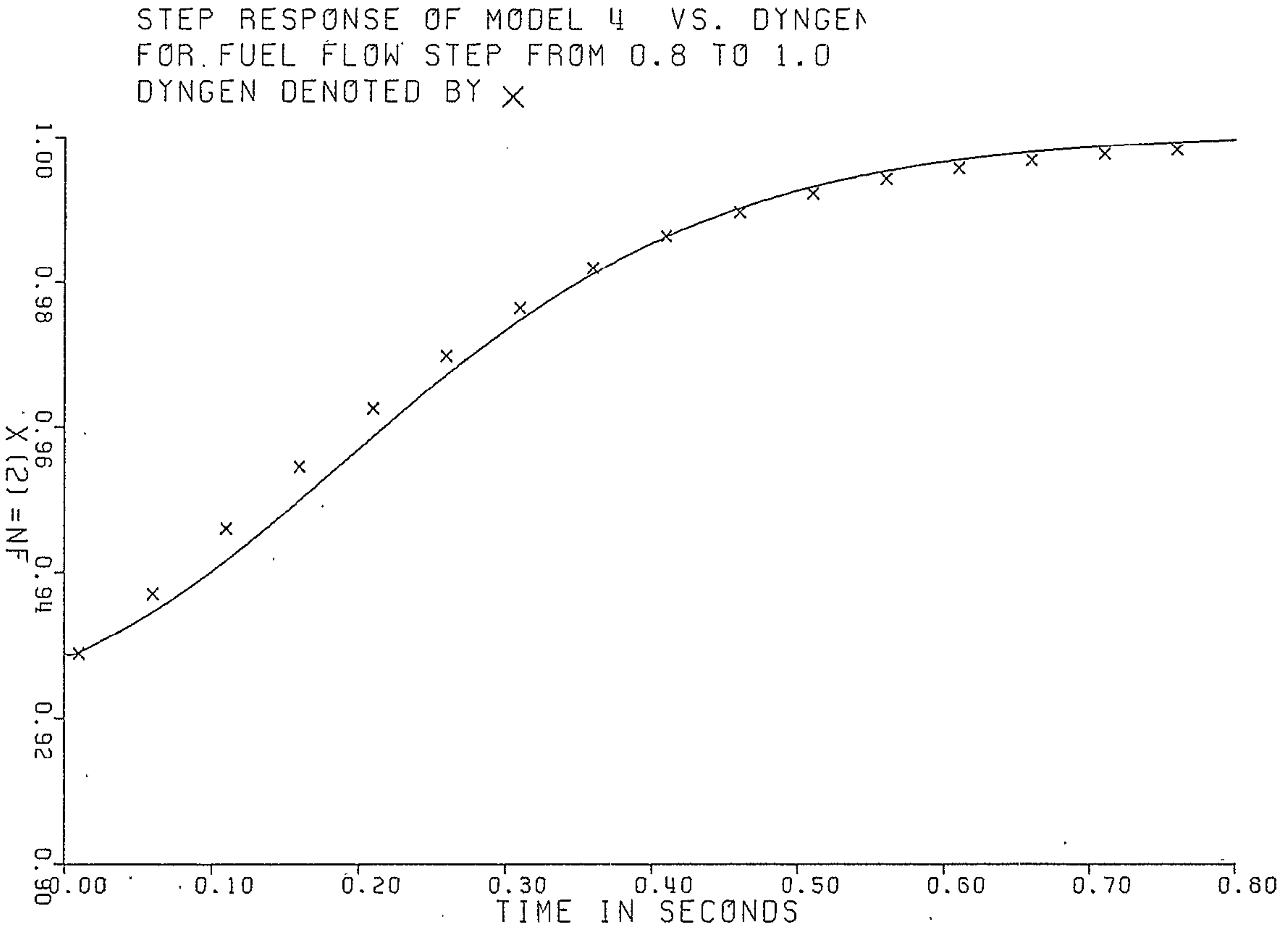

STEP RESPONSE OF MODEL 4 VS. DYNGEN
FOR FUEL FLOW STEP FROM 0.8 TO 1.0
DYNGEN DENOTED BY X
X(2)=NF
1.00
0.98
0.96
0.94
0.92
0.90
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 5.4-12

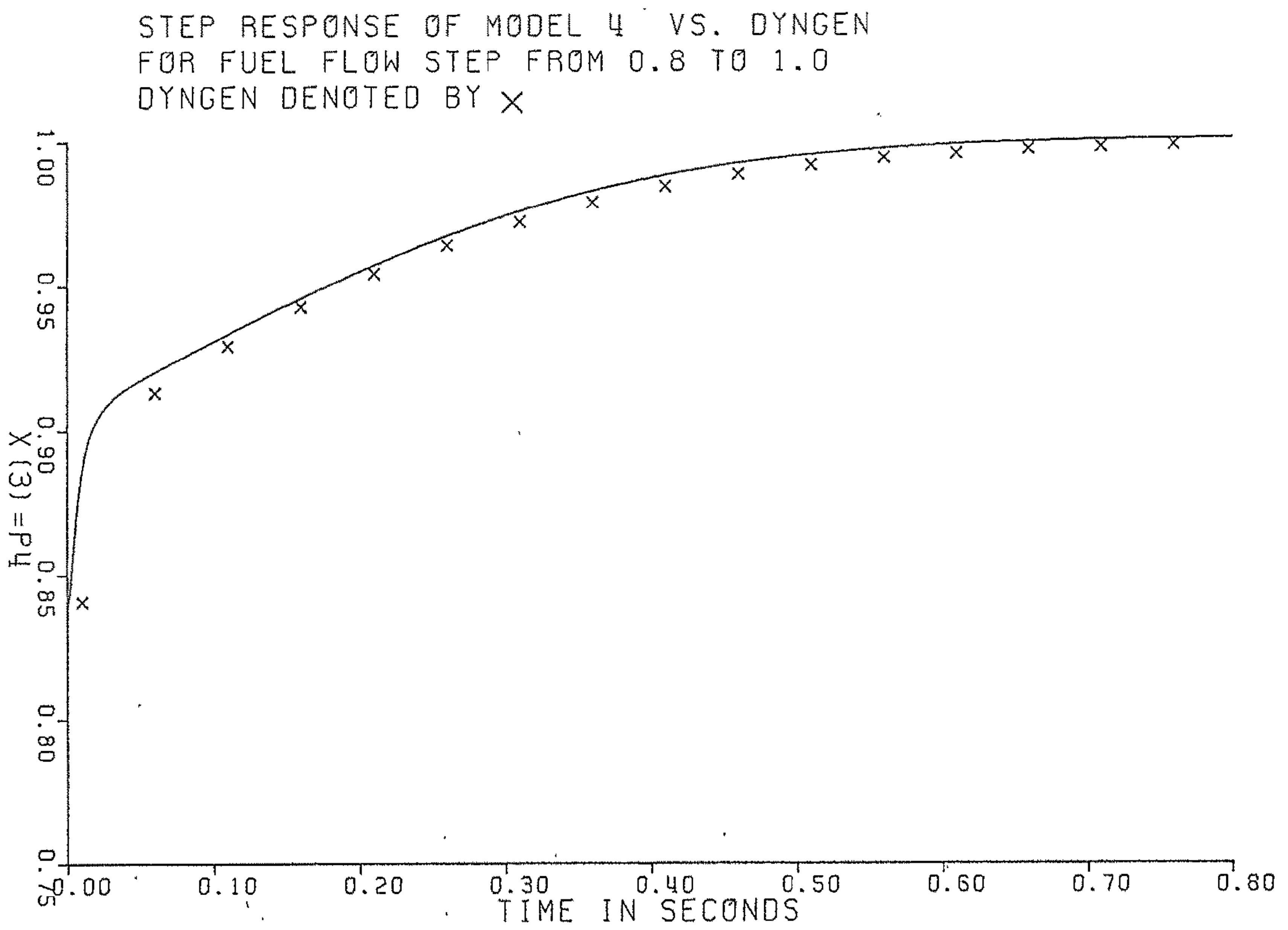

STEP RESPONSE OF MODEL 4 VS. DYNGEN
FOR FUEL FLOW STEP FROM 0.8 TO 1.0
DYNGEN DENOTED BY ×
X(3)=P4
1.00
0.95
0.90
0.85
0.80
0.75
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 5.4-13

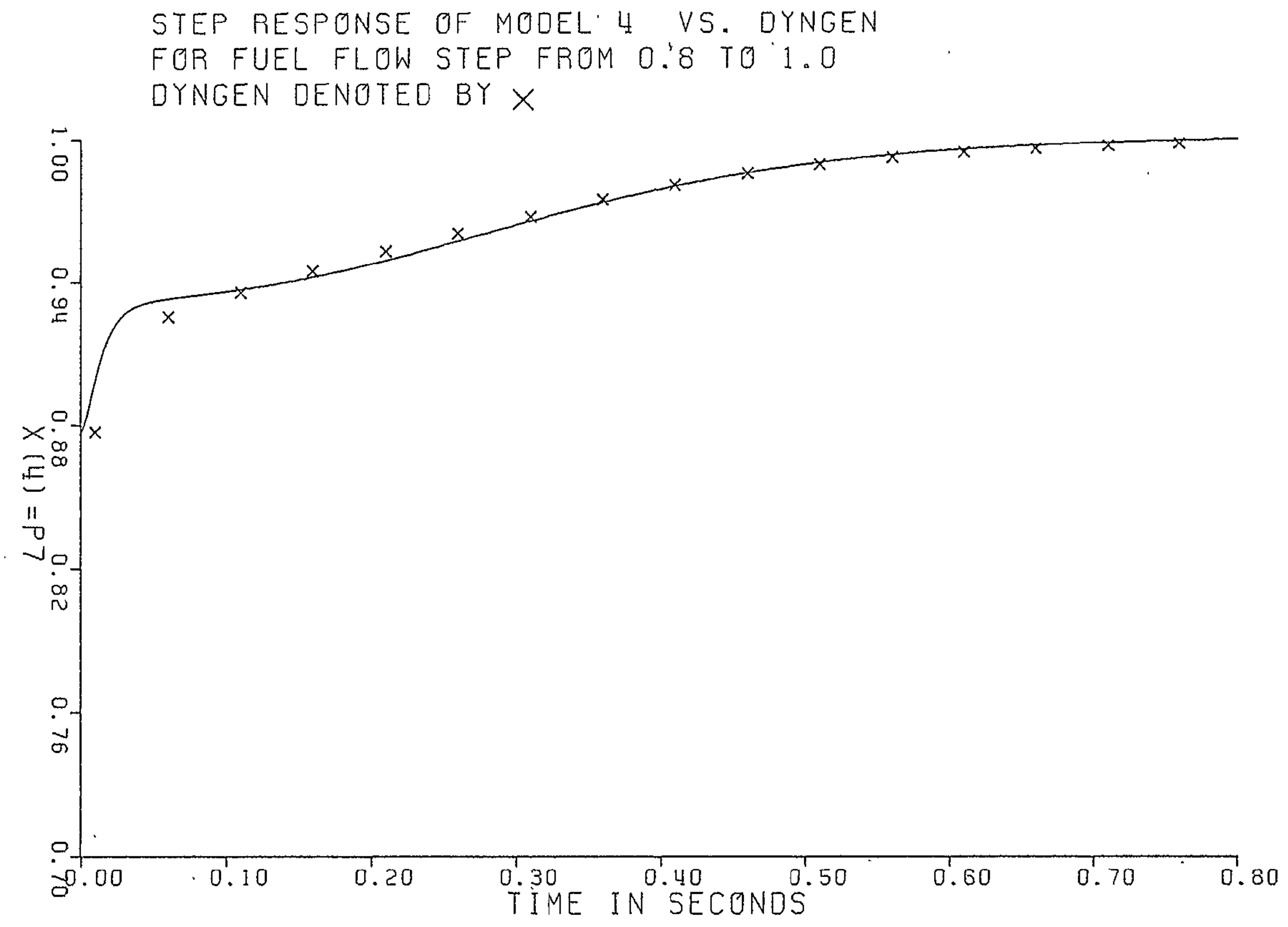

STEP RESPONSE OF MODEL 4 VS. DYNGEN
FOR FUEL FLOW STEP FROM 0.8 TO 1.0
DYNGEN DENOTED BY X
X(4) = P7
TIME IN SECONDS
0.70
0.76
0.82
0.88
0.94
1.00
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80

Figure 5.4-14

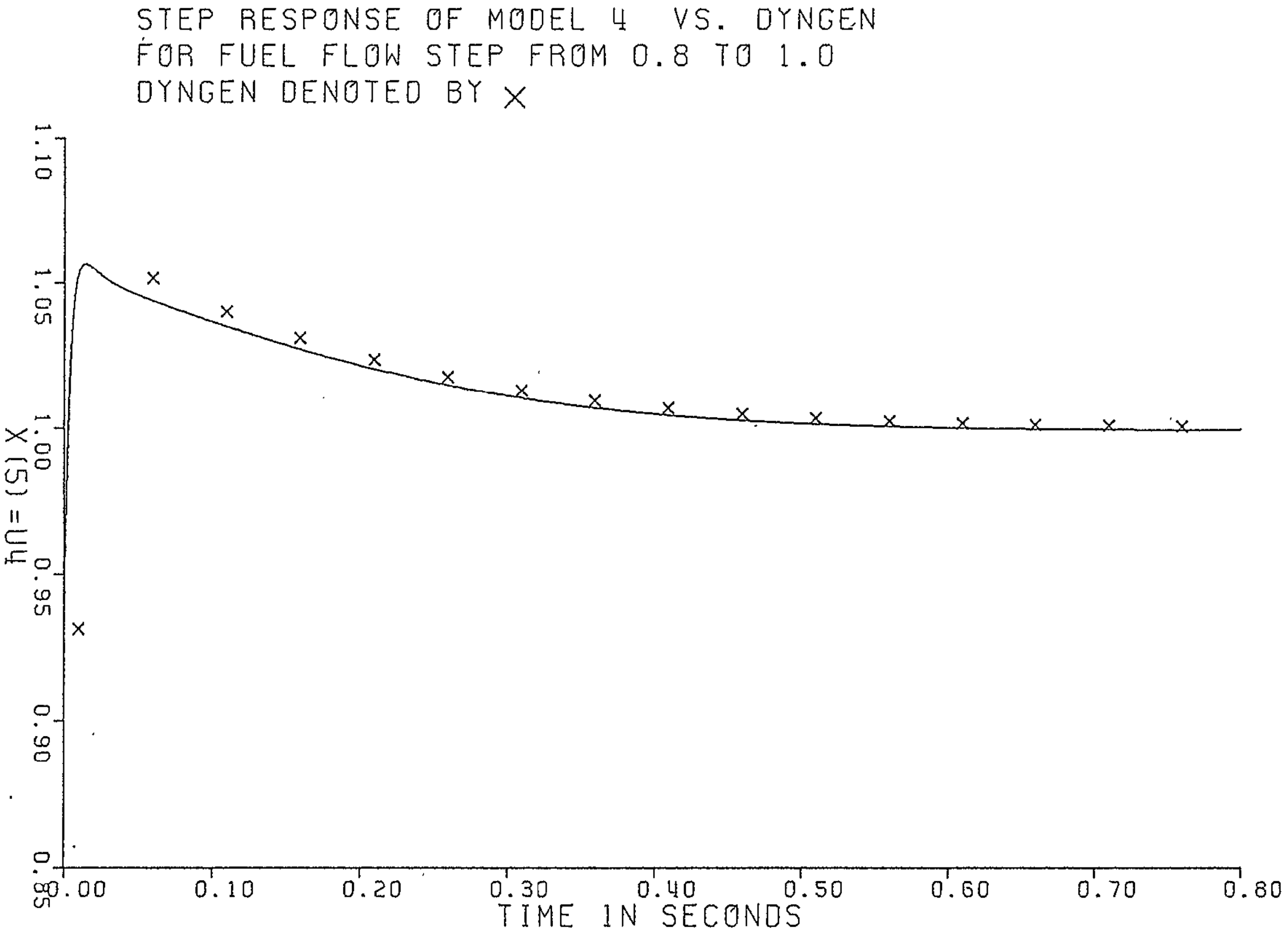

STEP RESPONSE OF MODEL 4 VS. DYNGEN
FOR FUEL FLOW STEP FROM 0.8 TO 1.0
DYNGEN DENOTED BY X
X(5)=U4
TIME IN SECONDS
0.85
0.90
0.95
1.00
1.05
1.10
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80

Figure 5.4-15

CHAPTER VI

SUCCESSIVE APPROXIMATION DYNAMIC PROGRAMMING

6.1 Introduction

The successive approximation dynamic programming method is described in detail by Longenbaker [1]. This chapter will state the problem, and describe refinements made to the previous software. In addition, the general structure of the software will be discussed, and repitition of an example from Longenbaker will serve to verify its accuracy.

6.2 Time Optimal Control Problem [1]

The necessary first step is to reformulate the models as discrete time systems. Let

$$x(t + \Delta t) = x(t) + \Delta t \cdot f(x(t),\ u(t)) \tag{6.2-1}$$

represent the system with starting time k and terminal time N. It is understood that

$$f(x(t),\ u(t)) = Ax(t) + B\ u(t) \tag{6.2-2}$$

for linear models. Let $x(k)$ be the starting state and let the terminal time N be defined as the first instant at which the system state reaches the designated target set S. All $x(t)$ are εX, the state set. The performance index

$$j(x,\underline{u}) = \sum_{t=k}^{N-\Delta t} \Delta t \tag{6.2-3}$$

$$t = k,\ k + \Delta t, \ldots, N-\Delta t$$

is to be minimized with $u(t)\ \varepsilon\ U$, the control set, and $\underline{u}$ defined as the control sequence:

$$\underline{u} = u(k),\ u(k + \Delta t), \ldots u(N-\Delta t) \qquad (6.2\text{-}4)$$

Furthermore, the minimization is subject to hard constraints of the form

$$y_i\ (x(t),\ u(t)) \leq c_i \qquad (6.2\text{-}5)$$

6.3 Refinement of the Dynamic Programming Method

The dynamic programming technique requires that the initial estimates of the cost function be greater than or equal to the final values. Furthermore, the method guarantees that the output of each iteration meets the same requirement. There are, however, two sources of error in the intermediate outputs: error due to a non-optimal choice of controls, and error due to overly large initial estimates. The former can only be eliminated by searches of the entire control space, but the latter can be eliminated by repeated iterations on a fixed control law.

Each control law, whether optimal or non-optimal, yields a set of cost values, which can be determined by the same successive approximations technique. By the definition of optimal, the cost values generated by any control law are guaranteed to be greater than or equal to the corresponding optimal values, and therefore qualify as input to an iteration of the general dynamic programming technique.

To effect this change, a subiteration loop repeats the successive approximations technique on a fixed control law until the values converge on the actual costs for that fixed set of controls. These values are then used as input to another iteration, including a search of the entire control space for a better control law. In this manner, a large number of control space searches are eliminated and a similarly large amount of c.p.u.

time is saved.

6.4 General Program Structure

Since the dynamic programming method was to be applied to several dissimilar models, an effort was made to isolate the two portions of the system, the dynamic programming method and the particular model, as much as possible. To that end, all the models were constructed as sets of subroutines, with matching subprogram names. The main program then refers to these routines to access model dependent quantities.

A brief description of the main program routine follows:

MAIN: The dynamic programming algorithm: reads data describing the state and control spaces, initial cost estimates, and iteration control values; performs iterations and subiterations until convergence occurs or iteration counts are exhausted; records results on disk and printer.

SPIRAL: This subroutine computes the indices of the next statespace point, spiralling out from the target, as described by Longenbaker [1].

V: This function subroutine interpolates new values of the cost function from current state and cost data.

NEXTX: This subroutine computes $x(t + \Delta t)$ and tests the new value against state space and output constraints.

The routines refered to by the dynamic programming algorithm, and therefore required in all models are:

INIT: This subroutine performs any initialization necessary for operation of the model.

XDOT: This subroutine computes derivative data as a function of state and control.

OUTPUT: This subroutine computes output data as a function of state and control, which may then be used in constraint testing.

COMPLT: This subroutine computes values for states not specified by the dynamic programming algorithm, should the order of the model be greater than two.

With this structure, all that need be done to change models is to concatenate the FORTRAN source for the dynamic programming algorithm and the desired model, and provide reference to any data sets the model may require. For more information concerning the operation of the algorithm, refer to the commented listing in appendix F.

6.5 Verification of Longenbaker 1L2 Unconstrained

Since new dynamic programming software was developed to increase generality and operating efficiency, it is necessary to verify proper performance. To this end, comparison was made between the results of the current software and that of Longenbaker for a linear example. In the unconstrained case, cost values differ by no more than 0.0001 seconds, which can be considered insignificant. The control laws differ in only a few cases, and then only by a single control space increment. Given the inherent inaccuracy of the control space quantization, this too is insignificant.

N_C

N_F	0.90	0.92	0.94	0.96	0.98	1.00	1.02	1.04	1.06	1.08	1.10	
1.10	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.10 0.70	0.70 0.70	0.50 0.70	0.50 0.70	0.50 0.70	0.50 0.70	1.10
1.08	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.10 0.70	0.50 0.70	0.50 0.70	0.50 0.70	0.50 0.70	0.50 0.70	1.08
1.06	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.10 0.70	0.50 0.70	0.50 0.70	0.50 0.70	0.50 0.70	0.50 0.70	1.06
1.04	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.05 0.70	0.50 0.70	0.50 0.70	0.50 0.70	0.50 0.70	0.50 0.70	1.04
1.02	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.05 0.70	0.50 0.70	0.50 0.70	0.50 0.70	0.50 0.70	0.50 0.70	1.02
1.00	1.40 1.20	1.40 1.20	1.40 1.10	1.40 1.00	1.40 0.90	1.00 1.00	0.50 1.20	0.50 1.20	0.50 1.20	0.50 1.15	0.50 1.20	1.00
0.98	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.35 1.20	1.30 1.20	0.50 1.20	0.50 1.20	0.50 1.20	0.98
0.96	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.30 1.20	1.30 1.20	1.35 1.20	1.30 1.20	0.50 1.20	0.50 1.20	0.96
0.94	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.30 1.20	1.30 1.20	1.35 1.20	1.35 1.20	1.30 1.20	0.50 1.20	0.94
0.92	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.35 1.20	1.35 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.25 1.20	0.92
0.90	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	0.90
	0.90	0.92	0.94	0.96	0.98	1.00	1.02	1.04	1.06	1.08	1.10	

FIGURE 6.5-1A. Model 1L2 (Unconstrained) - Optimal Control Law as per Longenbaker [1]

N_C

N_F	0.90	0.92	0.94	0.96	0.98	1.00	1.02	1.04	1.06	1.08	1.10	
1.10	0.2928	0.2776	0.2616	0.2450	0.2281	0.2102	0.1910	0.1718	0.1562	0.1486	0.1505	1.10
1.08	0.2786	0.2618	0.2459	0.2251	0.2053	0.1842	0.1620	0.1421	0.1311	0.1324	0.1439	1.08
1.06	0.2634	0.2447	0.2244	0.2026	0.1791	0.1535	0.1282	0.1117	0.1118	0.1259	0.1464	1.06
1.04	0.2469	0.2260	0.2027	0.1769	0.1483	0.1159	0.0901	0.0873	0.1058	0.1315	0.1579	1.04
1.02	0.2290	0.2054	0.1785	0.1475	0.1111	0.0675	0.0552	0.0835	0.1170	0.1481	0.1756	1.02
1.00	0.2096	0.1028	0.1514	0.1134	0.0653	0.0	0.0605	0.1052	0.1406	0.1702	0.1955	1.00
0.98	0.1908	0.1625	0.1308	0.0969	0.0674	0.0709	0.1135	0.1470	0.1741	0.1970	0.2174	0.98
0.96	0.1770	0.1509	0.1253	0.1054	0.1018	0.1222	0.1542	0.1804	0.2024	0.2213	0.2382	0.96
0.94	0.1706	0.1501	0.1344	0.1291	0.1391	0.1618	0.1869	0.2082	0.2265	0.2427	0.2572	0.94
0.92	0.1720	0.1508	0.1529	0.1576	0.1729	0.1937	0.2142	0.2319	0.2475	0.2613	0.2743	0.92
0.90	0.1801	0.1740	0.1757	0.1858	0.2021	0.2204	0.2374	0.2525	0.2660	0.2784	0.2898	0.90
	0.90	0.92	0.94	0.96	0.98	1.00	1.02	1.04	1.06	1.08	1.10	

FIGURE 6.5-1B. Model 1L2 (Unconstrained) - Cost as per Longenbaker [1]

N_F \ N_C	0.90	0.92	0.94	0.96	0.98	1.00	1.02	1.04	1.06	1.08	1.10
1.10	0.2928	0.2776	0.2616	0.2450	0.2280	0.2101	0.1910	0.1718	0.1562	0.1486	0.1505
1.08	0.2766	0.2618	0.2439	0.2250	0.2053	0.1842	0.1620	0.1421	0.1311	0.1324	0.1439
1.06	0.2634	0.2447	0.2244	0.2026	0.1791	0.1535	0.1282	0.1117	0.1118	0.1258	0.1464
1.04	0.2469	0.2260	0.2027	0.1769	0.1483	0.1159	0.0901	0.0873	0.1058	0.1315	0.1579
1.02	0.2290	0.2054	0.1785	0.1475	0.1111	0.0675	0.0551	0.0835	0.1170	0.1480	0.1756
1.00	0.2096	0.1828	0.1514	0.1134	0.0654	0.0	0.0605	0.1051	0.1406	0.1701	0.1954
0.98	0.1908	0.1625	0.1309	0.0969	0.0675	0.0709	0.1135	0.1470	0.1740	0.1969	0.2175
0.96	0.1770	0.1509	0.1253	0.1054	0.1018	0.1222	0.1542	0.1804	0.2024	0.2213	0.2381
0.94	0.1707	0.1501	0.1344	0.1291	0.1391	0.1617	0.1869	0.2081	0.2265	0.2426	0.2571
0.92	0.1721	0.1588	0.1529	0.1576	0.1728	0.1937	0.2141	0.2318	0.2474	0.2615	0.2745
0.90	0.1801	0.1740	0.1757	0.1858	0.2020	0.2203	0.2374	0.2524	0.2660	0.2783	0.2897

Figure 6.5-2-A Model 1L2 Unconstrained Cost

	N_C										
N_F	0.90	0.92	0.94	0.96	0.98	1.00	1.02	1.04	1.06	1.08	1.10
1.10	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.10 0.70	0.70 0.70	0.50 0.70	0.50 0.70	0.50 0.70	0.50 0.70
1.08	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.10 0.70	0.50 0.70	0.50 0.70	0.50 0.70	0.50 0.70	0.50 0.70
1.06	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.10 0.70	0.50 0.70	0.50 0.70	0.50 0.70	0.50 0.70	0.50 0.70
1.04	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.05 0.70	0.50 0.70	0.50 0.70	0.50 0.70	0.50 0.70	0.50 0.70
1.02	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.40 0.70	1.05 0.70	0.50 0.70	0.50 0.70	0.50 0.70	0.50 0.70	0.50 0.70
1.00	1.40 1.20	1.40 1.15	1.40 1.10	1.40 1.00	1.40 0.90	1.00 1.00	0.50 1.20	0.50 1.20	0.50 1.20	0.50 1.15	0.50 1.20
0.98	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.35 1.20	1.30 1.20	0.50 1.20	0.50 1.20	0.50 1.20
0.96	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.35 1.20	1.30 1.20	1.35 1.20	1.30 1.20	0.50 1.20	0.50 1.20
0.94	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.30 1.20	1.30 1.20	1.35 1.20	1.35 1.20	1.30 1.20	0.50 1.20
0.92	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.35 1.20	1.35 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.25 1.20
0.90	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20	1.40 1.20

Figure 6.5-2-B Model 1L2 Unconstrained Optimal Control Law

6.6 Verification of Longenbaker 1L2 Constrained

There is considerable disagreement between the results of the two programs, giving rise to the question: Which, if either, of the results are correct? Without extensive tracing of program operation, this question cannot be completely resolved, however, there is some evidence to motivate a choice of the new software.

Consider the case of $x(1) = 1.02$ and $x(2) = 1.00$. Both programs select identical optimal controls, $u(1) = 0.50$ and $u(2) = 1.20$. The same controls applied to a state adjacent (in terms of the state space quantization) to the target should yield the same cost value. This is not the case, with the Longenbaker software giving 0.0735 seconds and the new software giving 0.0606 seconds.

Note, however, that the unconstrained case yields precisely the same control $(u(1) = 0.50, u(2) = 1.20)$, and both the Longenbaker and new software give cost values of 0.0605 seconds. With or without constraints, the same control applied to the same state adjacent to the target should yield the same cost. With three of four calculations in agreement, it can only be concluded that the fourth is incorrect. The source of this disagreement, especially in light of the agreement with the unconstrained case, is functionally indeterminable.

N_C

N_F	0.90	0.92	0.94	0.96	0.98	1.00	1.02	1.04	1.06	1.08	1.10	
1.10	0.60 0.95	0.65 0.95	0.50 0.90	0.55 0.90	0.60 0.90	0.65 0.90	0.50 0.85	0.50 0.85	0.50 0.85	0.50 0.85	0.50 0.80	1.10
1.08	0.65 0.95	0.70 0.95	0.55 0.90	0.60 0.90	0.65 0.90	0.70 0.90	0.55 0.85	0.50 0.85	0.50 0.85	0.50 0.85	0.50 0.80	1.08
1.06	1.15 1.05	0.50 0.90	0.55 0.90	0.65 0.90	0.70 0.90	0.50 0.85	0.60 0.85	0.65 0.85	0.50 0.85	0.50 0.80	0.50 0.80	1.06
1.04	1.15 1.05	1.05 1.00	0.60 0.90	0.70 0.90	0.75 0.90	0.80 0.90	0.65 0.85	0.50 0.85	0.50 0.80	0.50 0.80	0.50 0.80	1.04
1.02	1.15 1.05	1.10 1.00	1.15 1.00	0.75 0.90	0.80 0.90	0.85 0.90	0.50 0.85	0.50 0.80	0.50 1.10	0.50 1.20	0.50 1.20	1.02
1.00	1.15 1.05	1.15 1.00	1.20 1.00	1.20 1.00	1.25 1.00	1.00 1.00	0.50 1.20	0.50 1.20	0.50 1.20	0.50 1.20	0.50 1.20	1.00
0.98	1.15 1.05	1.15 1.10	1.20 1.05	1.15 1.20	1.15 1.20	1.15 1.20	1.15 1.20	1.15 1.20	1.10 1.20	0.65 1.20	0.50 1.20	0.98
0.96	1.15 1.05	1.10 1.20	1.15 1.15	1.15 1.20	1.15 1.20	1.20 1.20	1.20 1.20	1.20 1.20	1.25 1.20	1.30 1.20	1.30 1.20	0.96
0.94	1.10 1.15	1.10 1.20	1.15 1.15	1.15 1.20	1.15 1.20	1.20 1.20	1.20 1.20	1.25 1.20	1.25 1.20	1.30 1.20	1.30 1.20	0.94
0.92	1.10 1.15	1.10 1.20	1.10 1.20	1.15 1.20	1.15 1.20	1.15 1.20	1.20 1.20	1.20 1.20	1.25 1.20	1.25 1.20	1.25 1.20	0.92
0.90	1.10 1.10	1.05 1.20	1.05 1.20	1.10 1.20	1.10 1.20	1.10 1.20	1.15 1.20	1.15 1.20	1.20 1.20	1.20 1.20	1.20 1.20	0.90
	0.90	0.92	0.94	0.96	0.98	1.00	1.02	1.04	1.06	1.08	1.10	

FIGURE 6.6.1A. Model 1L2 (Constrained) - Optimal Control Law as per Longenbaker [1].

N_C

N_F		0.90	0.92	0.94	0.96	0.98	1.00	1.02	1.04	1.05	1.08	1.10		
		*	*	*	*	*	*	*	*	*	*	*		
1.10	*	0.4219	0.4076	0.3919	0.3750	0.3567	0.3360	0.3107	0.2812	0.2474	0.2095	0.1728	*	1.10
1.08	*	0.4045	0.3885	0.3706	0.3513	0.3296	0.3037	0.2719	0.2345	0.1909	0.1403	0.1380	*	1.08
1.06	*	0.3851	0.3667	0.3463	0.3235	0.2964	0.2628	0.2215	0.1712	0.1212	0.1199	0.1471	*	1.06
1.04	*	0.3633	0.3417	0.3178	0.2895	0.2543	0.2089	0.1508	0.0919	0.1022	0.1374	0.1703	*	1.04
1.02	*	0.3391	0.3126	0.2831	0.2465	0.1985	0.1316	0.0564	0.0860	0.1288	0.1641	0.1936	*	1.02
1.00	*	0.3136	0.2799	0.2402	0.1897	0.1180	0.0	0.0735	0.1232	0.1605	0.1904	0.2159	*	1.00
0.98	*	0.2882	0.2490	0.2017	0.1435	0.0843	0.0899	0.1351	0.1694	0.1969	0.2199	0.2398	*	0.98
0.96	*	0.2648	0.2225	0.1748	0.1309	0.1192	0.1481	0.1796	0.2053	0.2268	0.2454	0.2621	*	0.96
0.94	*	0.2453	0.2050	0.1681	0.1506	0.1641	0.1905	0.2142	0.2341	0.2516	0.2673	0.2816	*	0.94
0.92	*	0.2328	0.2004	0.1808	0.1833	0.2025	0.2237	0.2420	0.2582	0.2729	0.2864	0.2989	*	0.92
0.90	*	0.2315	0.2079	0.2054	0.2163	0.2338	0.2505	0.2654	0.2790	0.2916	0.3034	0.3144	*	0.90
		*	*	*	*	*	*	*	*	*	*	*		
		0.90	0.92	0.94	0.96	0.98	1.00	1.02	1.04	1.06	1.08	1.10		

FIGURE 6.6.1-B. Model 1L2 (Constrained) - Cost as per Longenbaker [1].

		N_C										
		0.90	0.92	0.94	0.96	0.98	1.00	1.02	1.04	1.06	1.08	1.10
N_F	1.10	0.3727	0.3576	0.3415	0.3236	0.3033	0.2800	0.2540	0.2283	0.2074	0.1949	0.1901
	1.08	0.3558	0.3388	0.3204	0.2997	0.2760	0.2479	0.2175	0.1908	0.1741	0.1691	0.1725
	1.06	0.3375	0.3181	0.2966	0.2721	0.2437	0.2090	0.1742	0.1507	0.1439	0.1494	0.1636
	1.04	0.3173	0.2947	0.2691	0.2396	0.2046	0.1598	0.1239	0.1137	0.1232	0.1430	0.1657
	1.02	0.2952	0.2681	0.2368	0.2004	0.1550	0.0940	0.0736	0.0932	0.1224	0.1514	0.1778
	1.00	0.2719	0.2384	0.1990	0.1517	0.0881	0.0	0.0606	0.1052	0.1409	0.1706	0.1961
	0.98	0.2558	0.2205	0.1794	0.1353	0.0908	0.0713	0.1137	0.1471	0.1742	0.1972	0.2177
	0.96	0.2472	0.2117	0.1755	0.1405	0.1183	0.1226	0.1544	0.1806	0.2026	0.2215	0.2384
	0.94	0.2427	0.2102	0.1807	0.1569	0.1497	0.1621	0.1872	0.2084	0.2267	0.2429	0.2574
	0.92	0.2420	0.2140	0.1918	0.1778	0.1794	0.1941	0.2144	0.2321	0.2477	0.2617	0.2746
	0.90	0.2471	0.2236	0.2077	0.2006	0.2065	0.2208	0.2378	0.2528	0.2663	0.2787	0.2900

FIGURE6.6.2-A MODEL 1L2 CONSTRAINED COST

						N_C					
N_F	0.90	0.92	0.94	0.96	0.98	1.00	1.02	1.04	1.06	1.08	1.10
1.10	1.10 1.05	1.15 1.05	1.20 1.05	0.80 0.95	0.85 0.95	0.65 0.90	0.50 0.85	0.50 0.85	0.50 0.85	0.50 0.85	0.50 0.80
1.08	1.15 1.05	1.20 1.05	1.20 1.05	0.85 0.95	0.65 0.90	0.70 0.90	0.55 0.85	0.50 0.85	0.50 0.85	0.50 0.85	0.50 0.80
1.06	1.15 1.05	1.15 1.05	1.20 1.05	1.15 1.00	0.70 0.90	0.75 0.90	0.50 0.85	0.50 0.85	0.50 0.85	0.50 0.80	0.50 0.80
1.04	1.15 1.05	1.15 1.05	1.20 1.05	1.20 1.00	0.75 0.90	0.55 0.85	0.50 0.85	0.50 0.85	0.50 0.80	0.50 0.80	0.50 0.80
1.02	1.15 1.05	1.15 1.05	1.15 1.00	1.25 1.00	0.55 0.85	0.60 0.85	0.50 0.85	0.50 0.80	0.50 0.80	0.50 0.80	0.50 0.80
1.00	1.10 1.20	1.15 1.10	1.20 1.05	1.20 1.00	1.25 1.00	1.00 1.00	0.50 1.20	0.50 1.20	0.50 1.20	0.50 1.15	0.50 1.10
0.98	1.10 1.20	1.10 1.20	1.15 1.20	1.15 1.20	1.15 1.20	1.20 1.20	1.20 1.20	1.25 1.20	0.50 1.20	0.50 1.20	0.50 1.20
0.96	1.10 1.15	1.10 1.20	1.15 1.15	1.15 1.20	1.15 1.20	1.20 1.20	1.20 1.20	1.25 1.20	1.25 1.20	0.50 1.20	0.50 1.20
0.94	1.10 1.15	1.10 1.20	1.10 1.20	1.15 1.20	1.15 1.20	1.20 1.20	1.20 1.20	1.25 1.20	1.25 1.20	1.30 1.20	0.50 1.20
0.92	1.10 1.15	1.10 1.20	1.10 1.20	1.15 1.20	1.15 1.20	1.15 1.20	1.20 1.20	1.20 1.20	1.25 1.20	1.25 1.20	1.25 1.20
0.90	1.00 1.20	1.05 1.20	1.05 1.20	1.10 1.20	1.10 1.20	1.10 1.20	1.15 1.20	1.15 1.20	1.20 1.20	1.20 1.20	1.20 1.20

FIGURE 6.6.2B MODEL 1L2 CONSTRAINED OPTIMAL CONTROL LAW

CHAPTER VII

OPTIMAL CONTROL LAWS

7.1 Introduction

Before subjecting each model to the dynamic programming algorithm, care must be taken in choosing state and control space parameters, to insure the accuracy of the results. The target point and state space quantization must be chosen so that the entire state space window lies within the model's range of accuracy. The steady state controls for the target must also be known, since they will be needed during simulation. The third of model 4's five data points ($u(1) = 0.8$, $u(2) = 1.0$; $x(1) = 0.948434$, $x(2) = 0.928751$) was chosen as the target and 0.01 as the quantization, to satisfy these requirements.

Control space limits must also be chosen to guarantee accuracy. Only controls within the range of those used to generate the data points can be used with certainty. For models 3A and 3B, $0.74 \leq u(1) \leq 1.0$ and $0.74 \leq u(2) \leq 1.0$ were used; for model 4 the control space limits are $0.74 \leq u(1) \leq 1.0$ and $0.87 \leq u(2) \leq 1.13$. These limits correspond closely to the controls used in data point generation.*

7.2 Model 3A Constrained

The extremal controls appear almost exclusively throughout three quarters of the state space window, indicating that system performance is restricted by the control space limits. These limits may not be relaxed, however, since they already represent the limits of the model's accuracy.

The initial state for simulations will be $X(1) = 1.0$, $X(2) = 1.0$, in

*For the constrained cases; $ZC \leq 1.15$, $ZF \leq 1.105$, and $T4 \leq 1.08$.

the less restricted quarter of the state space, so this control law will still be of interest.

7.3 Model 3B Constrained

As with model 3A, extremal controls appear throughout three quarters of the state space window. Again, X(1) = 1.0, X(2) = 1.0 lies in the less restricted quarter.

7.4 Model 4 Constrained

Because different control space limits were used, this control law predicts better performance throughout the state space window. The extremal controls do appear frequently, however, indicating that even better performance is possible with less restricted controls. Again, these control limits represent the range of accuracy of the model and may not be relaxed with confidence.

		N_C										
		0.8984	0.9084	0.9184	0.9284	0.9384	0.9484	0.9584	0.9684	0.9784	0.9884	0.9964
	0.9788	0.5771	0.5367	0.4927	0.4444	0.3903	0.3278	0.2600	0.2081	0.1836	0.1854	0.2026
	0.9688	0.5665	0.5241	0.4773	0.4246	0.3633	0.2883	0.2086	0.1586	0.1469	0.1613	0.1865
	0.9588	0.5561	0.5116	0.4615	0.4033	0.3323	0.2378	0.1517	0.1148	0.1235	0.1516	0.1853
	0.9488	0.5458	0.4990	0.4452	0.3805	0.2969	0.1749	0.0958	0.0870	0.1154	0.1511	0.1875
	0.9388	0.5357	0.4864	0.4284	0.3561	0.2568	0.0981	0.0487	0.0786	0.1164	0.1543	0.1917
N_F	0.9288	0.5256	0.4736	0.4111	0.3300	0.2107	0.0	0.0407	0.0809	0.1203	0.1589	0.1967
	0.9188	0.5156	0.4608	0.3932	0.3015	0.1577	0.0202	0.0517	0.0876	0.1256	0.1640	0.2018
	0.9088	0.5056	0.4477	0.3742	0.2700	0.1267	0.0345	0.0626	0.0950	0.1313	0.1692	0.2069
	0.8988	0.4956	0.4344	0.3544	0.2401	0.1075	0.0465	0.0721	0.1023	0.1371	0.1743	0.2119
	0.8888	0.4856	0.4208	0.3337	0.2133	0.0954	0.0565	0.0803	0.1090	0.1426	0.1793	0.2168
	0.8788	0.4756	0.4068	0.3129	0.1913	0.0891	0.0649	0.0875	0.1151	0.1479	0.1841	0.2215

FIGURE 7.2 -A MODEL 3A CONSTRAINED COST

N_F \ N_C	0.8984	0.9084	0.9184	0.9284	0.9384	0.9484	0.9584	0.9684	0.9784	0.9884	0.9984
0.9788	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	1.00 0.94	0.94 0.92	0.88 0.90	0.82 0.88	0.84 0.88	0.78 0.86
0.9688	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	0.94 0.92	0.88 0.90	0.80 0.88	0.74 0.86	0.76 0.66	0.76 0.86
0.9588	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	0.86 0.90	0.80 0.88	0.74 0.66	0.74 0.86	0.74 0.86	0.74 0.86
0.9488	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	0.80 0.88	0.78 0.68	0.74 0.86	0.74 0.86	0.74 0.88	0.74 0.92
0.9388	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	0.82 0.88	0.74 0.66	0.74 0.86	0.74 0.90	0.74 0.92	0.74 0.92
0.9288	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	1.00 0.92	0.80 1.00	0.74 0.98	0.74 0.96	0.74 0.96	0.74 0.94	0.74 0.94
0.9188	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	0.94 1.00	0.74 1.00	0.74 1.00	0.74 1.00	0.74 1.00	0.74 1.00
0.9088	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	0.74 1.00	0.74 1.00	0.74 1.00	0.74 1.00	0.74 1.00
0.8988	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	0.74 1.00	0.74 1.00	0.74 1.00	0.74 1.00	0.74 1.00
0.8888	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	0.74 1.00	0.74 1.00	0.74 1.00	0.74 1.00	0.74 1.00
0.8788	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	1.00 1.00	0.74 1.00	0.74 1.00	0.74 1.00	0.74 1.00	0.74 1.00

FIGURE 7.2-B MODEL 3A CONSTRAINED OPTIMAL CONTROL LAW

N_F \ N_C	0.8984	0.9084	0.9184	0.9284	0.9384	0.9484	0.9584	0.9684	0.9784	0.9884	0.9984
0.9788	0.4708	0.4301	0.3858	0.3370	0.2824	0.2193	0.1622	0.1344	0.1384	0.1597	0.1866
0.9688	0.4604	0.4180	0.3712	0.3187	0.2580	0.1848	0.1260	0.1099	0.1263	0.1546	0.1844
0.9588	0.4504	0.4061	0.3566	0.2999	0.2322	0.1464	0.0928	0.0936	0.1224	0.1543	0.1863
0.9488	0.4406	0.3944	0.3421	0.2807	0.2049	0.1036	0.0653	0.0876	0.1226	0.1577	0.1912
0.9388	0.4310	0.3828	0.3274	0.2610	0.1760	0.0557	0.0488	0.0688	0.1278	0.1642	0.1980
0.9288	0.4215	0.3713	0.3127	0.2407	0.1450	0.0	0.0530	0.0974	0.1367	0.1725	0.2058
0.9188	0.4122	0.3599	0.2978	0.2196	0.1112	0.0244	0.0698	0.1101	0.1471	0.1814	0.2135
0.9088	0.4030	0.3484	0.2826	0.1974	0.0938	0.0417	0.0825	0.1200	0.1552	0.1885	0.2201
0.8988	0.3939	0.3370	0.2671	0.1770	0.0847	0.0569	0.0932	0.1282	0.1621	0.1947	0.2260
0.8888	0.3848	0.3255	0.2514	0.1593	0.0805	0.0691	0.1021	0.1353	0.1682	0.2003	0.2312
0.8788	0.3758	0.3139	0.2361	0.1454	0.0803	0.0795	0.1098	0.1416	0.1737	0.2054	0.2361

FIGURE 7.3 -A MODEL 3B CONSTRAINED COST

		N_C										
		0.8984	0.9084	0.9184	0.9284	0.9384	0.9484	0.9584	0.9684	0.9784	0.9884	0.9984
N_F	0.9788	1.00	1.00	1.00	1.00	1.00	0.92	0.84	0.78	0.78	0.74	0.74
		1.00	1.00	1.00	1.00	1.00	0.92	0.90	0.88	0.88	0.86	1.00
	0.9688	1.00	1.00	1.00	1.00	1.00	0.94	0.78	0.76	0.74	0.74	0.74
		1.00	1.00	1.00	1.00	1.00	0.92	0.88	0.88	0.86	1.00	1.00
	0.9588	1.00	1.00	1.00	1.00	1.00	0.86	0.76	0.74	0.74	0.74	0.74
		1.00	1.00	1.00	1.00	1.00	0.90	0.88	0.86	1.00	1.00	1.00
	0.9488	1.00	1.00	1.00	1.00	1.00	0.80	0.74	0.74	0.74	0.74	0.74
		1.00	1.00	1.00	1.00	1.00	0.88	0.88	0.86	1.00	1.00	1.00
	0.9388	1.00	1.00	1.00	1.00	1.00	0.82	0.74	0.74	0.74	0.74	0.74
		1.00	1.00	1.00	1.00	1.00	0.88	0.86	1.00	1.00	1.00	1.00
	0.9288	1.00	1.00	1.00	1.00	1.00	0.80	0.74	0.74	0.74	0.74	0.74
		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	0.9188	1.00	1.00	1.00	1.00	1.00	0.96	0.74	0.74	0.74	0.74	0.74
		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	0.9088	1.00	1.00	1.00	1.00	1.00	1.00	0.74	0.74	0.74	0.74	0.74
		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	0.8988	1.00	1.00	1.00	1.00	1.00	1.00	0.74	0.74	0.74	0.74	0.74
		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	0.8888	1.00	1.00	1.00	1.00	1.00	1.00	0.74	0.74	0.74	0.74	0.74
		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	0.8788	1.00	1.00	1.00	1.00	1.00	1.00	0.74	0.74	0.74	0.74	0.74
		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

FIGURE 7.3-B MODEL 3B CONSTRAINED OPTIMAL CONTROL LAW

N_C

N_F	0.8984	0.9084	0.9184	0.9284	0.9384	0.9484	0.9584	0.9684	0.9784	0.9884	0.9984
0.9788	0.2711	0.2584	0.2484	0.2407	0.2349	0.2305	0.2268	0.2233	0.2200	0.2173	0.2166
0.9688	0.2475	0.2329	0.2208	0.2112	0.2039	0.1982	0.1934	0.1898	0.1879	0.1889	0.1940
0.9588	0.2214	0.2043	0.1897	0.1780	0.1686	0.1612	0.1555	0.1531	0.1555	0.1632	0.1766
0.9488	0.1922	0.1719	0.1545	0.1403	0.1286	0.1182	0.1128	0.1162	0.1286	0.1473	0.1703
0.9388	0.1593	0.1349	0.1144	0.0972	0.0818	0.0664	0.0698	0.0927	0.1211	0.1499	0.1776
0.9288	0.1210	0.0926	0.0689	0.0473	0.0254	0.0	0.0634	0.1087	0.1427	0.1678	0.1900
0.9188	0.1571	0.1490	0.1415	0.1341	0.1292	0.1332	0.1471	0.1594	0.1755	0.1906	0.2060
0.9088	0.2052	0.2075	0.2059	0.2021	0.1984	0.1968	0.1973	0.2002	0.2056	0.2124	0.2220
0.8988	0.2483	0.2532	0.2513	0.2462	0.2403	0.2349	0.2311	0.2292	0.2295	0.2322	0.2373
0.8888	0.2834	0.2866	0.2824	0.2756	0.2682	0.2612	0.2553	0.2511	0.2488	0.2488	0.2513
0.8788	0.3111	0.3108	0.3047	0.2970	0.2888	0.2810	0.2741	0.2685	0.2648	0.2630	0.2637

FIGURE 7.4-A MODEL 4 CONSTRAINED COST

N_C

N_F	0.8984	0.9084	0.9184	0.9284	0.9384	0.9484	0.9584	0.9684	0.9784	0.9884	0.9984
0.9788	0.76 1.13	0.74 0.97	0.74 0.97	0.74 0.95	0.74 0.95	0.74 0.93	0.74 0.93	0.74 0.93	0.74 0.93	0.74 0.91	0.74 0.91
0.9688	0.76 1.13	0.74 0.97	0.74 0.97	0.74 0.95	0.74 0.93	0.74 0.93	0.74 0.93	0.74 0.91	0.74 0.91	0.74 0.91	0.74 0.89
0.9588	0.76 1.13	0.74 0.97	0.74 0.95	0.74 0.95	0.74 0.93	0.74 0.93	0.74 0.91	0.74 0.89	0.74 0.89	0.74 0.87	0.74 0.87
0.9488	0.76 1.13	0.74 0.95	0.74 0.95	0.74 0.93	0.74 0.93	0.74 0.93	0.74 0.89	0.74 0.87	0.74 0.87	0.74 0.87	0.74 0.87
0.9388	0.76 1.13	1.00 1.13	0.74 0.93	0.74 0.93	0.74 0.93	0.74 0.91	0.74 0.87	0.74 0.87	0.74 0.87	0.74 0.87	0.74 0.87
0.9288	1.00 1.13	1.00 1.13	1.00 1.13	1.00 1.13	1.00 1.13	0.80 1.00	0.74 1.11	0.74 1.07	0.74 1.13	0.74 1.13	0.74 1.13
0.9188	1.00 1.13	1.00 1.13	1.00 1.13	1.00 1.11	1.00 1.11	0.98 1.09	0.92 1.09	0.80 1.13	0.80 1.13	0.80 1.13	0.74 1.13
0.9088	1.00 1.09	1.00 1.09	1.00 1.11	1.00 1.11	1.00 1.11	1.00 1.11	1.00 1.09	0.98 1.09	0.80 1.13	0.80 1.13	0.80 1.13
0.8988	1.00 1.07	1.00 1.09	1.00 1.11	1.00 1.13	1.00 1.13	1.00 1.13	1.00 1.11	1.00 1.11	1.00 1.09	0.90 1.11	0.80 1.13
0.8888	1.00 1.07	1.00 1.11	1.00 1.13	1.00 1.13	1.00 1.13	1.00 1.13	1.00 1.13	1.00 1.13	1.00 1.11	1.00 1.09	0.90 1.11
0.8788	1.00 1.07	1.00 1.13	1.00 1.13	1.00 1.13	1.00 1.13	1.00 1.13	1.00 1.13	1.00 1.13	1.00 1.11	1.00 1.11	1.00 1.09

FIGURE 7.4-B MODEL 4 CONSTRAINED OPTIMAL CONTROL LAW

CHAPTER VIII

CONTROLLER SIMULATIONS

8.1 Introduction

The performance of each control law generated was tested in two ways: by imposing the control law on the model from which it was generated, and by imposing it on the DYNGEN simulator. In each case, the initial state was the design point (u(1) = 1.0, u(2) = 1.0, x(1) = 1.0, x(2) = 1.0), and the final state was the control law target point (u(1) = 0.8, u(2) = 1.0, x(1) = 0.948434, x(2) = 0.928751). The simulation program used to impose the control laws on the models from which they were generated employs an Euler integration technique with a user controlled time step. This allows for the use of a smaller time increment in the steeper portions of a time response, without sacrificing the programming simplicity of the Euler technique as compared to higher order numerical integration methods.

The results of these simulations were not altogether satisfactory. Despite precautions taken to insure model accuracy, two of the model/control law systems are unstable. Experimentation with the integration time increment, and the initial state, seems to indicate that the instability is not generated by the integration method. Application of the control laws to the DYNGEN simulator did not result in instability either, so this response is not inherent in the control law.

This result does have a positive interpretation, however. Since it occured on the less complex models, 3A and 3B, one may conclude that the inclusion of more data points and derivative information, along with a more sophisticated interpolation technique, does yield a better model.

8.2 Model 3A with 3A-Controller

The first application of a control law to its model resulted in instability. After only 0.041 seconds, the states and outputs are beyond all physical limitations. Just prior to this overflow, u(1) oscillates sharply across a discontinuity in the control law, an effect Longenbaker [1] warns of in his work:

> "Interpolation will often lead to error when you are interpolating in a region of state space where the control laws change abruptly. Obviously, the optimal control which is desired is either one extreme or the other, and not something in-between."
>
> ([1] 6.1 p. 78)

This result would seem to indicate that model 3A, in its simplicity is incapable of modelling the response of a system to a continuously varying control.

The states move outside the model's range of accuracy, and the model breaks down almost immediately.

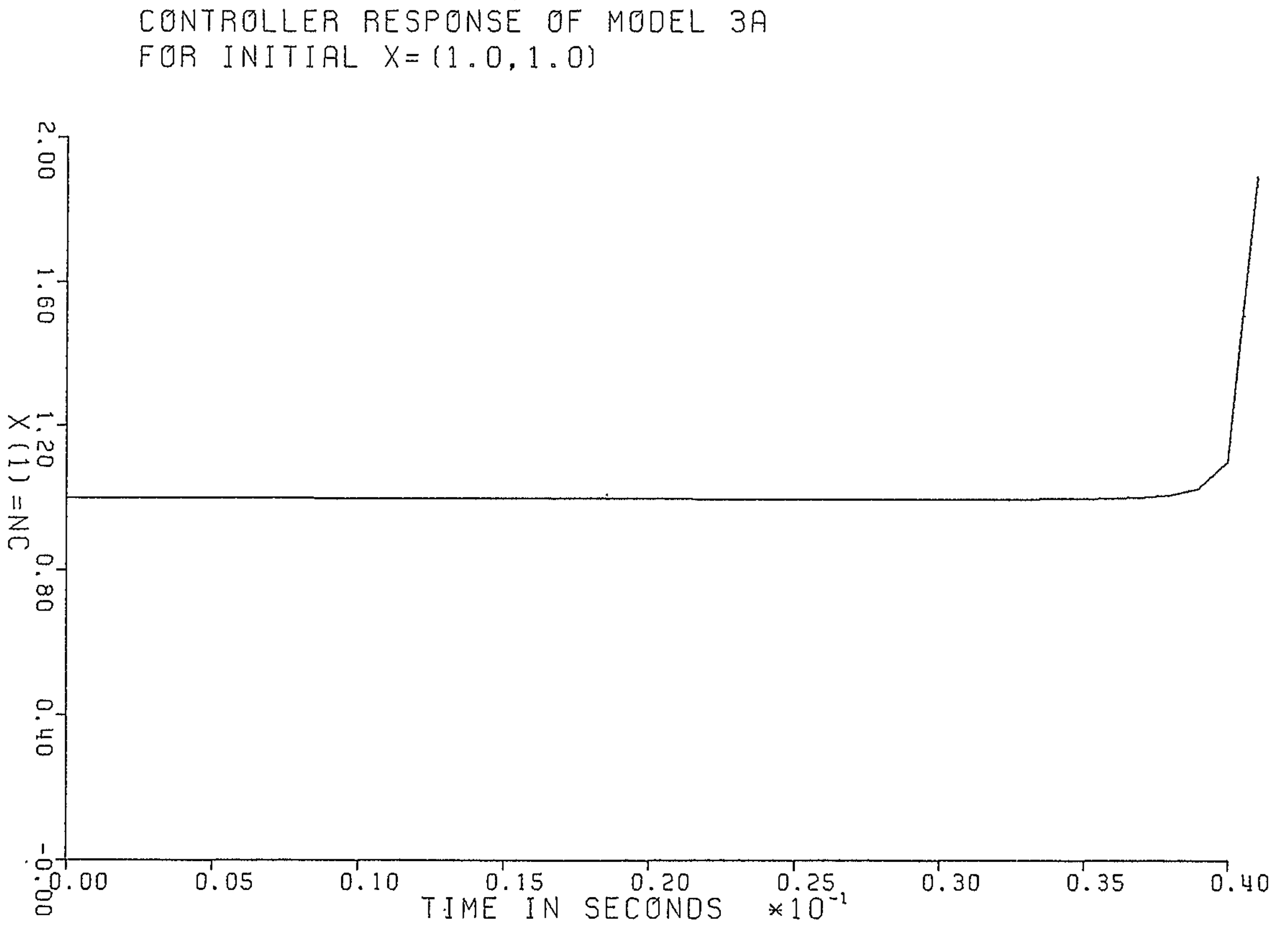

CONTROLLER RESPONSE OF MODEL 3A
FOR INITIAL X=(1.0,1.0)
2.00
1.60
1.20
0.80
0.40
-0.00
X(1)=NC
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS $\times 10^{-1}$

Figure 8.2-1

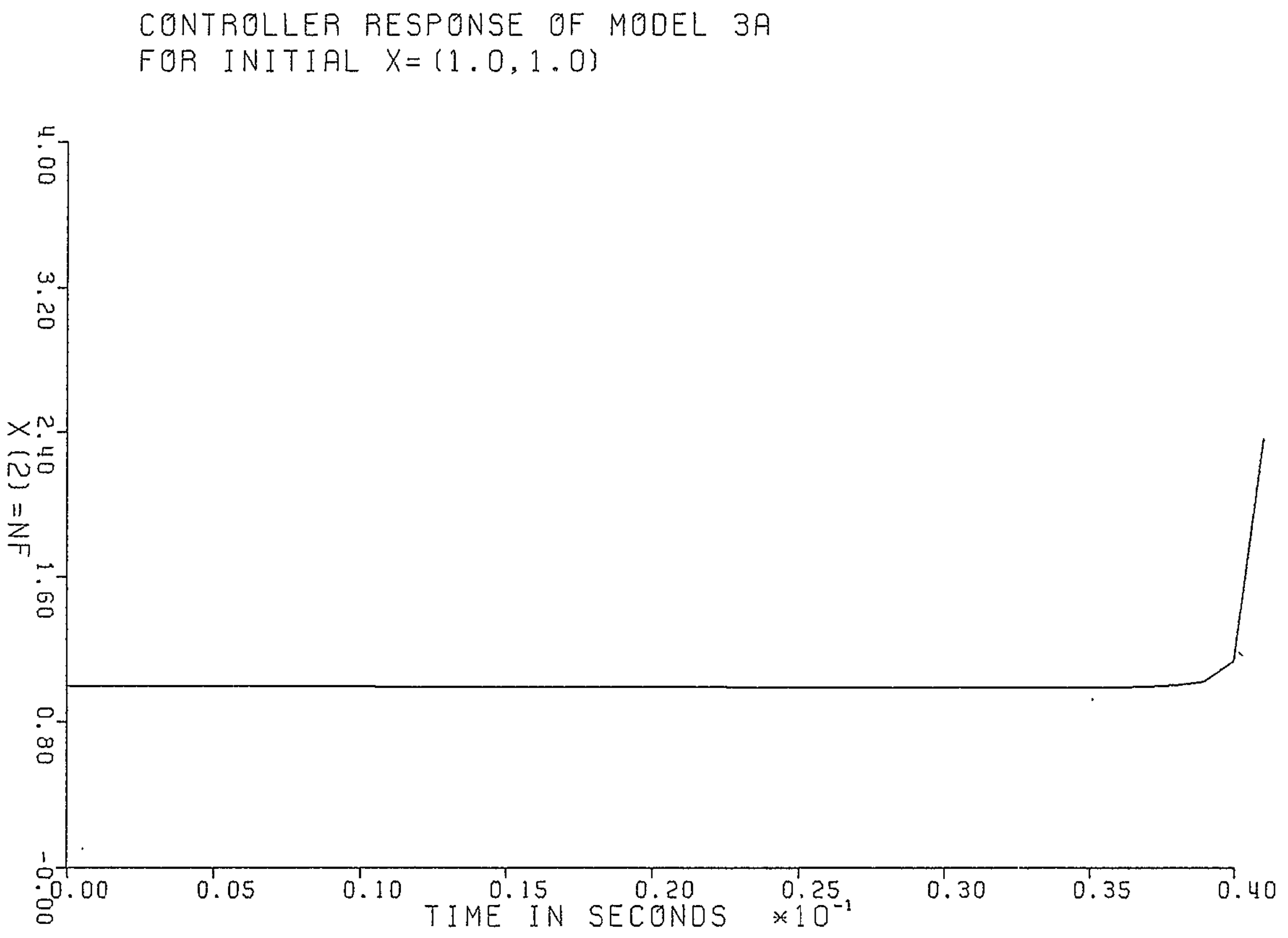

CONTROLLER RESPONSE OF MODEL 3A
FOR INITIAL X=(1.0,1.0)
X(2)=NF
4.00
3.20
2.40
1.60
0.80
-0.00
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS $*10^{-1}$

Figure 8.2-2

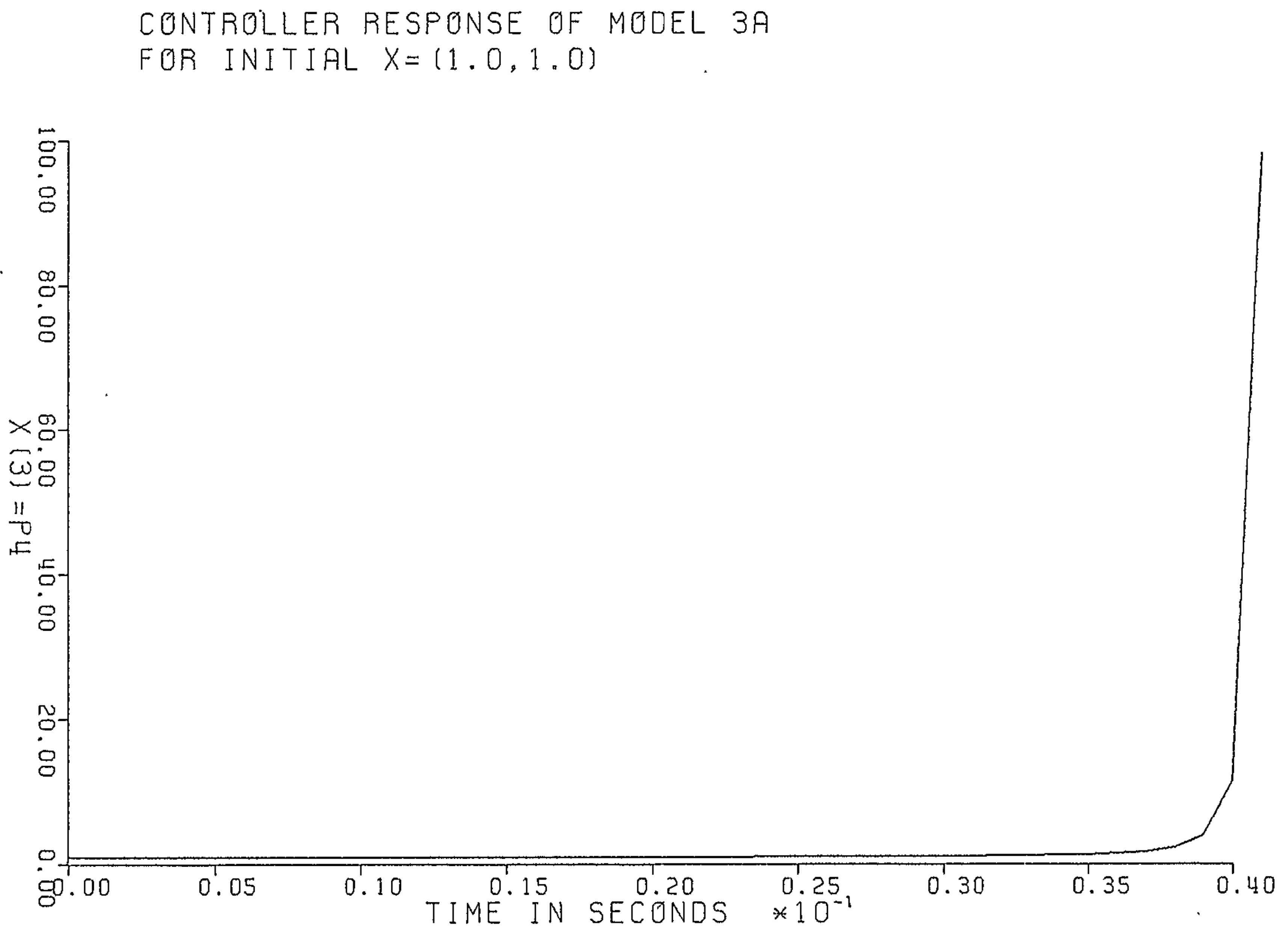

CONTROLLER RESPONSE OF MODEL 3A
FOR INITIAL X=(1.0,1.0)
X(3)=P4
0.00
20.00
40.00
60.00
80.00
100.00
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS *10^-1

Figure 8.2-3

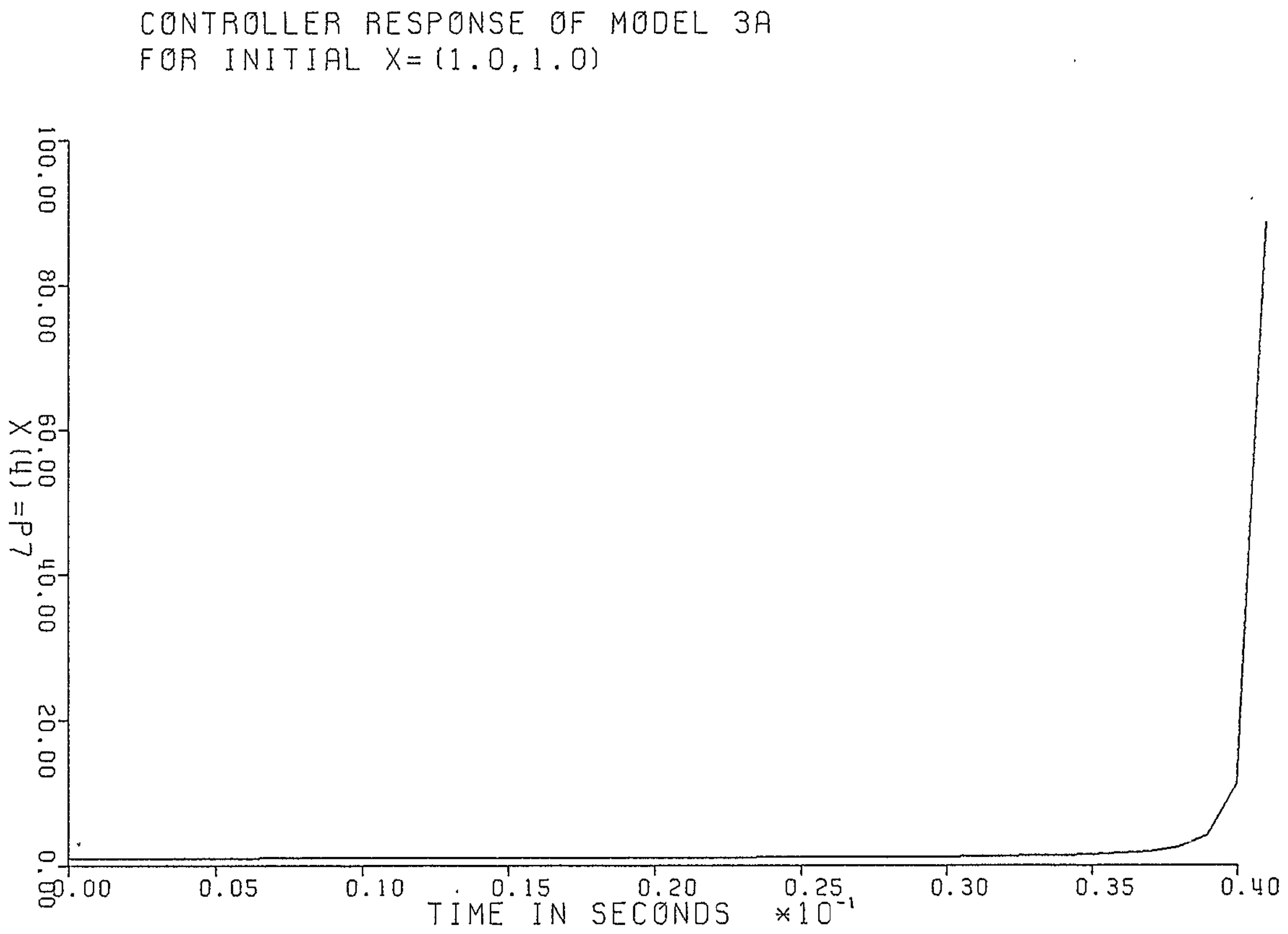

CONTROLLER RESPONSE OF MODEL 3A
FOR INITIAL X=(1.0,1.0)
X(4)=P7
0.00
20.00
40.00
60.00
80.00
100.00
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS $*10^{-1}$

Figure 8.2-4

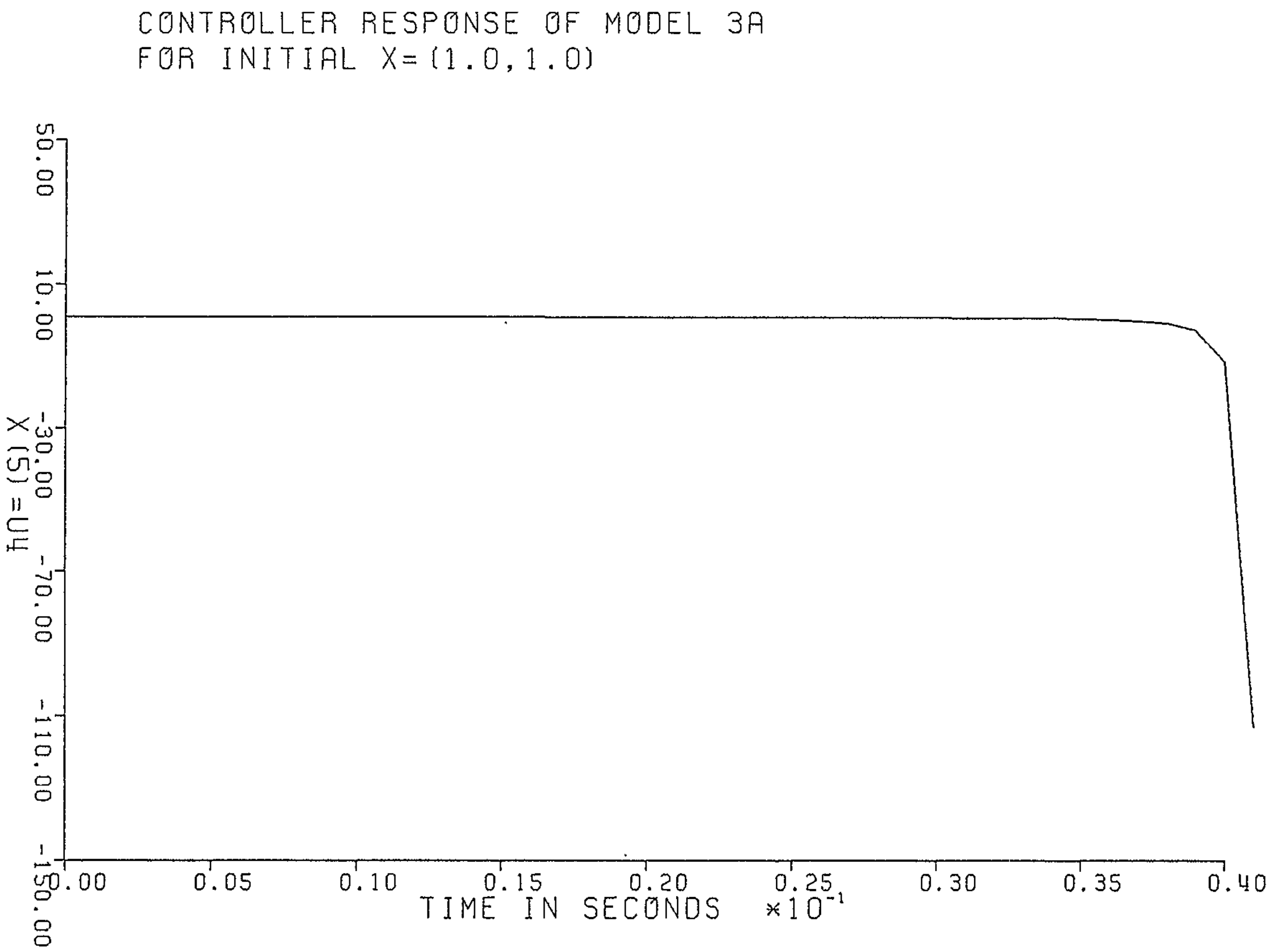

CONTROLLER RESPONSE OF MODEL 3A
FOR INITIAL X=(1.0,1.0)
50.00
10.00
-30.00
-70.00
-110.00
-150.00
X(5)=U4
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS ×10^-1

Figure 8.2-5

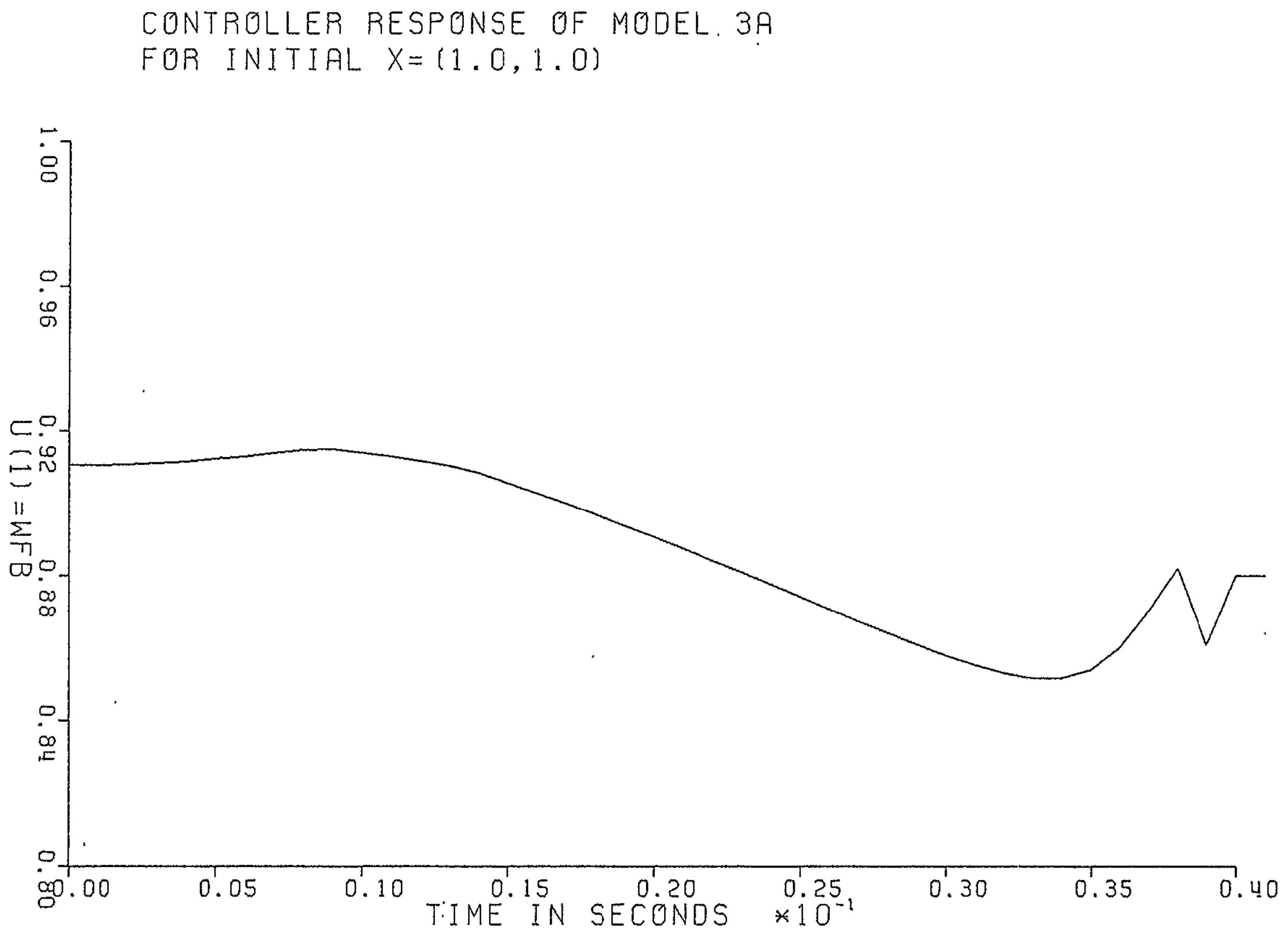

CONTROLLER RESPONSE OF MODEL 3A
FOR INITIAL X=(1.0,1.0)
U(1)=WFB
1.00
0.96
0.92
0.88
0.84
0.80
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS ×10⁻¹

Figure 8.2-6

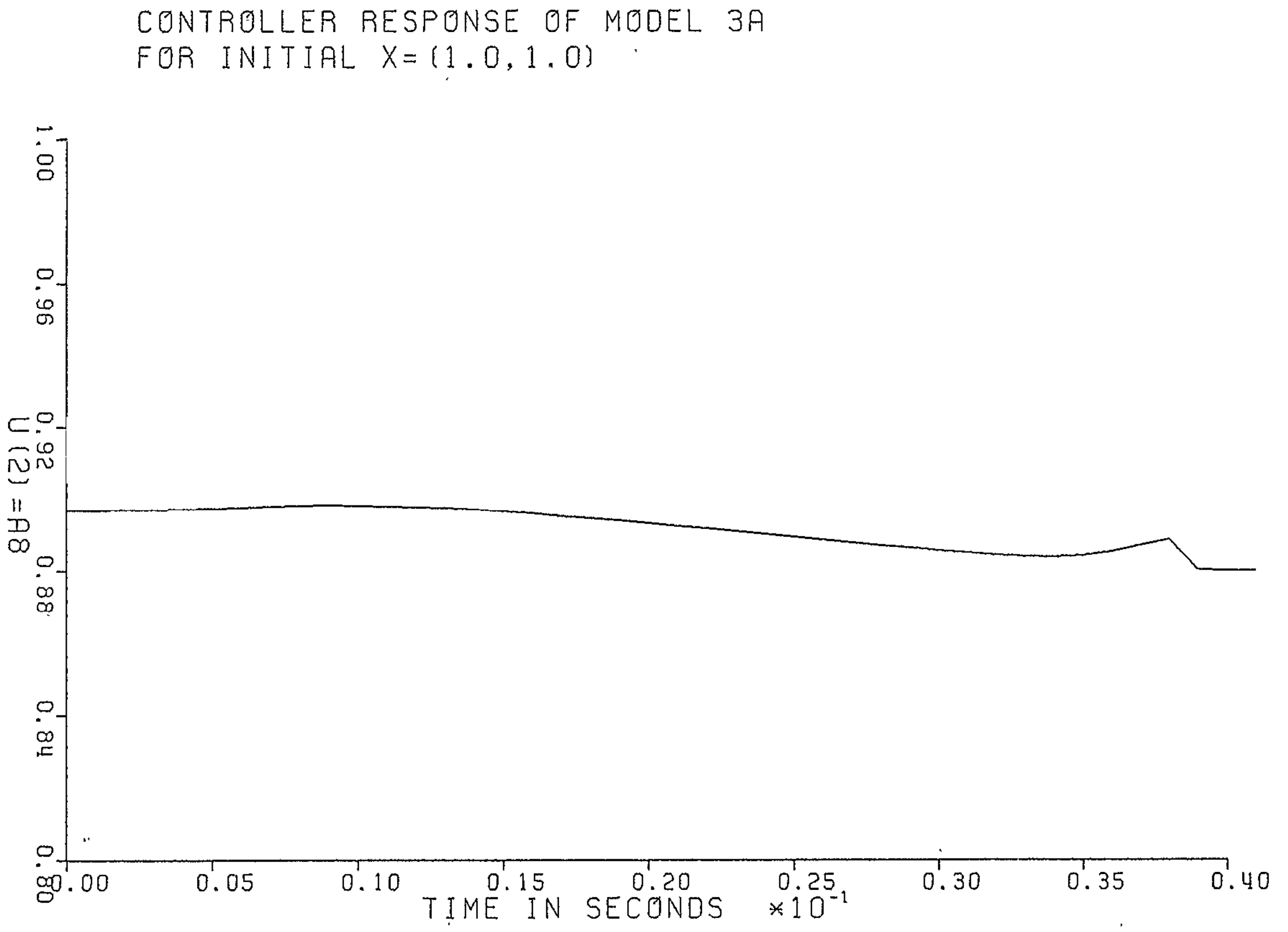

CONTROLLER RESPONSE OF MODEL 3A
FOR INITIAL X=(1.0,1.0)
U(2)=A8
1.00
0.96
0.92
0.88
0.84
0.80
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS $\times 10^{-1}$

Figure 8.2-7

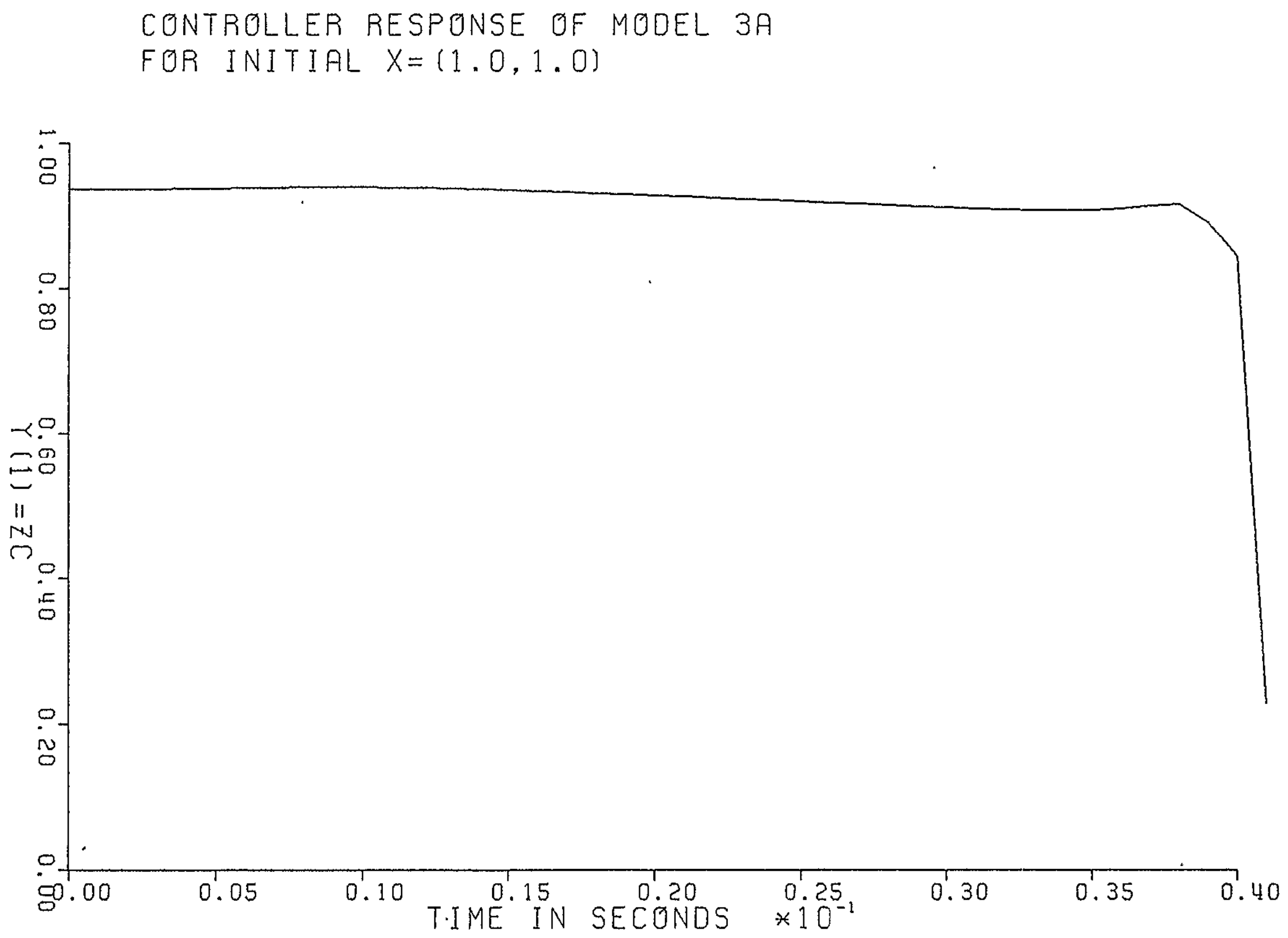

CONTROLLER RESPONSE OF MODEL 3A
FOR INITIAL X=(1.0,1.0)
Y(1)=ZC
1.00
0.80
0.60
0.40
0.20
0.00
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS *10⁻¹

Figure 8.2-8

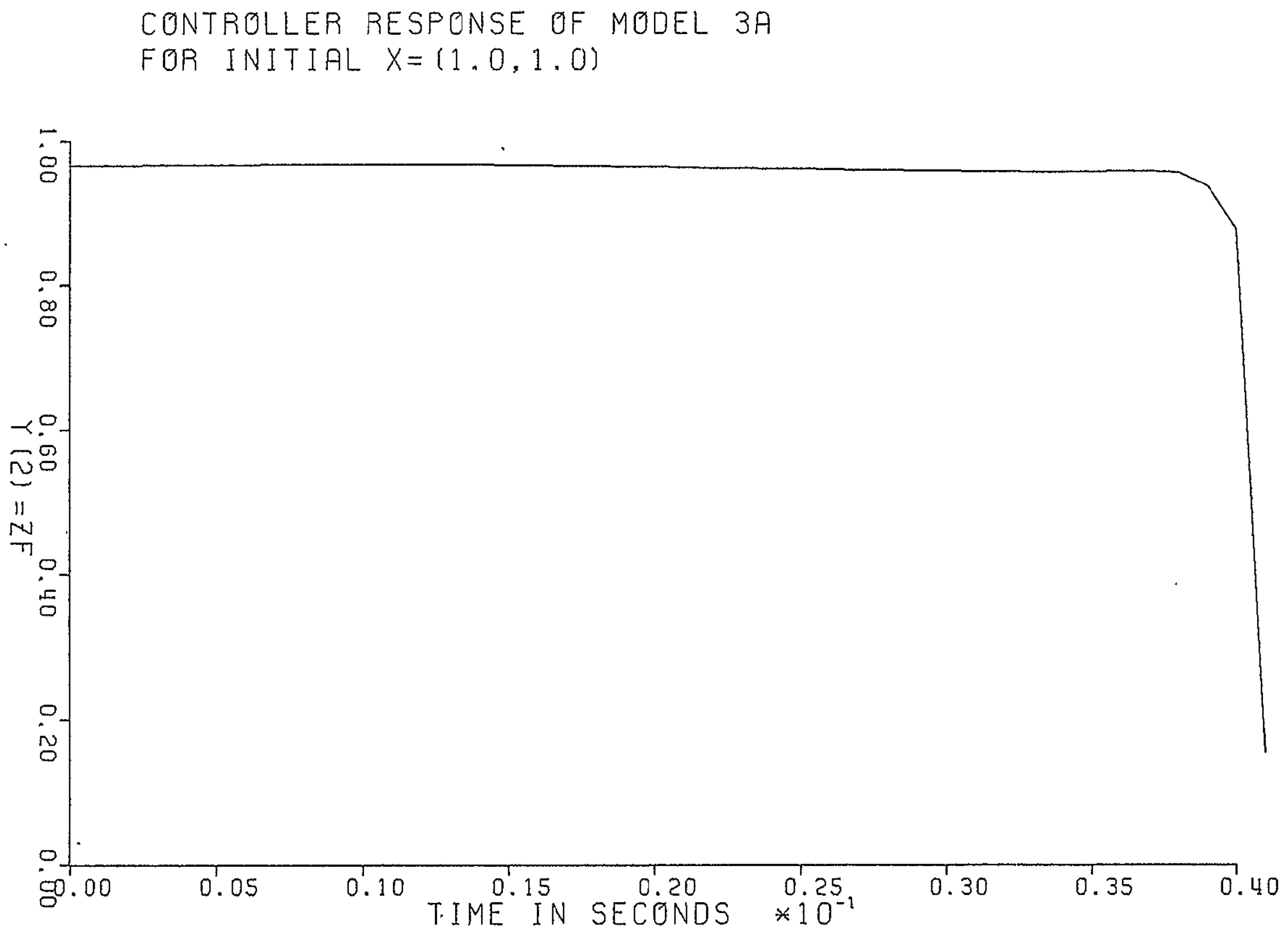

CONTROLLER RESPONSE OF MODEL 3A
FOR INITIAL X=(1.0,1.0)
Y(2)=ZF
0.00
0.20
0.40
0.60
0.80
1.00
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS *10^-1

Figure 8.2-9

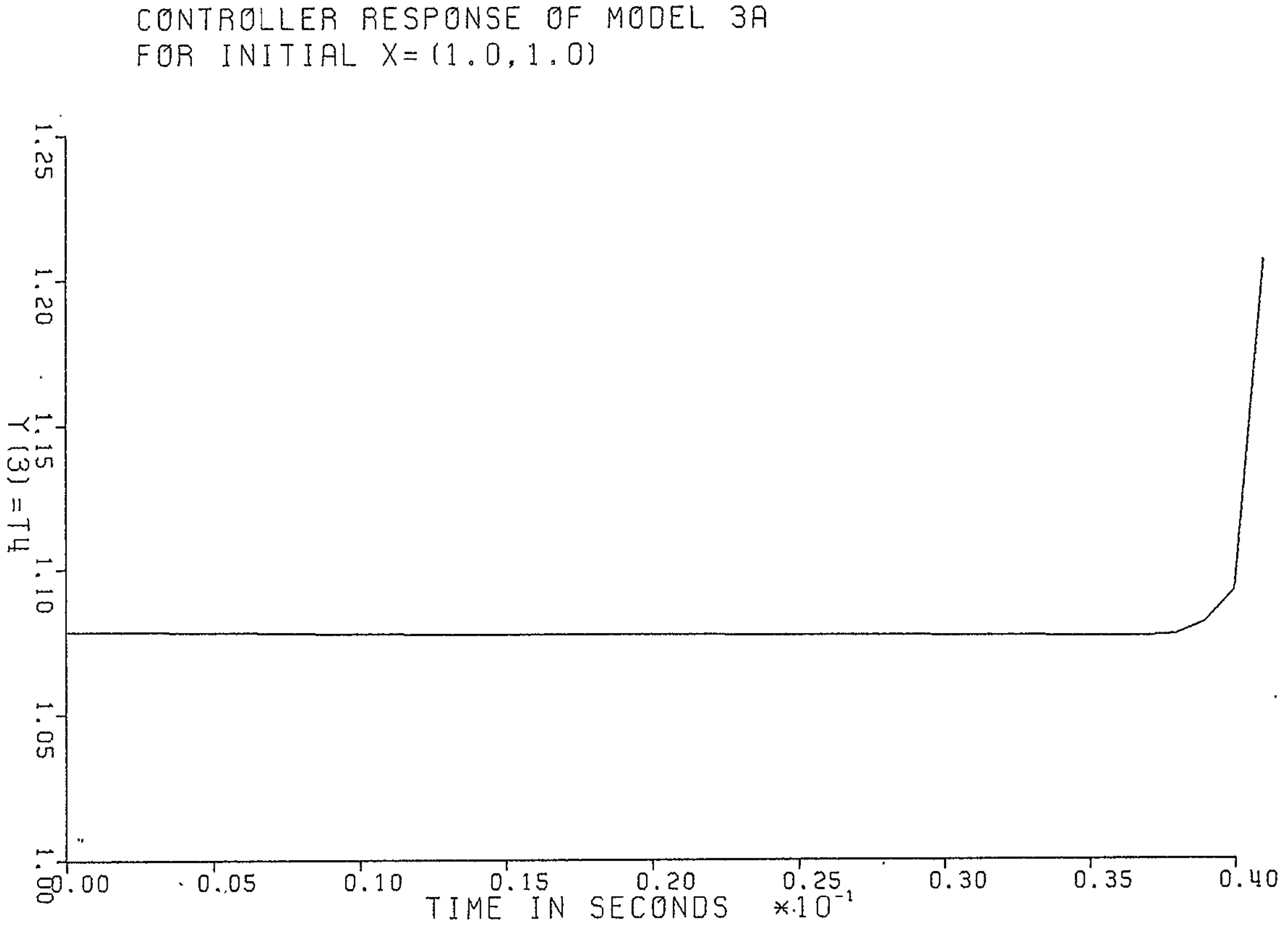

CONTROLLER RESPONSE OF MODEL 3A
FOR INITIAL X=(1.0,1.0)
Y(3)=T4
1.00
1.05
1.10
1.15
1.20
1.25
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS $\times 10^{-1}$

Figure 8.2-10

8.3 Model 3B with 3B-Controller

This model/control law system is also unstable. After 0.027 seconds the states and outputs have exceeded physical limitations. In this instance, u(2) oscillates just before overflow, throwing the system into instability. Again, the indication is that the system is incapable of modelling the response to a continuously varying control, and model breakdown occurs almost immediately.

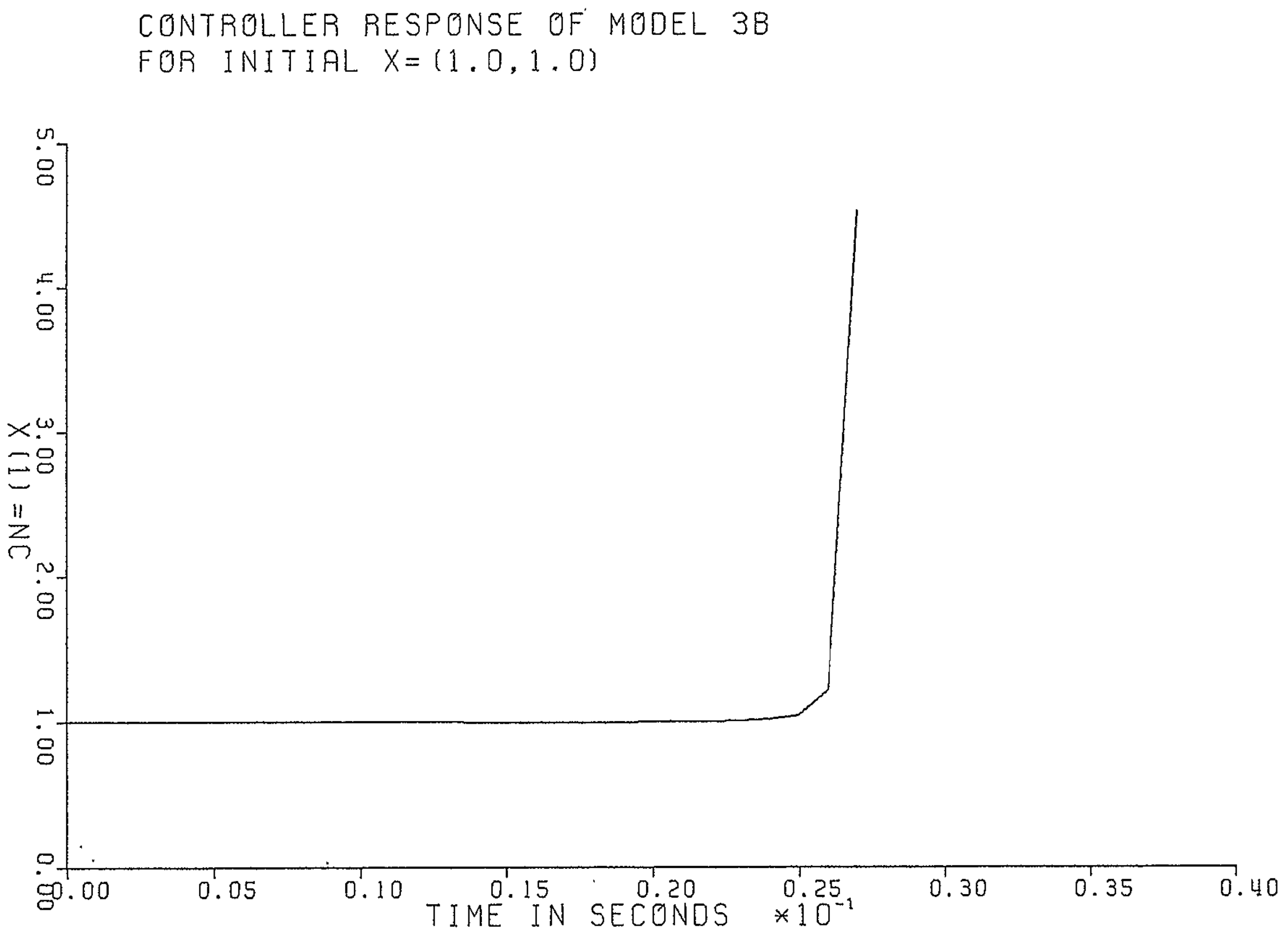

CONTROLLER RESPONSE OF MODEL 3B
FOR INITIAL X=(1.0,1.0)
X(1)=NC
0.00
1.00
2.00
3.00
4.00
5.00
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS $\times 10^{-1}$

Figure 8.3-1

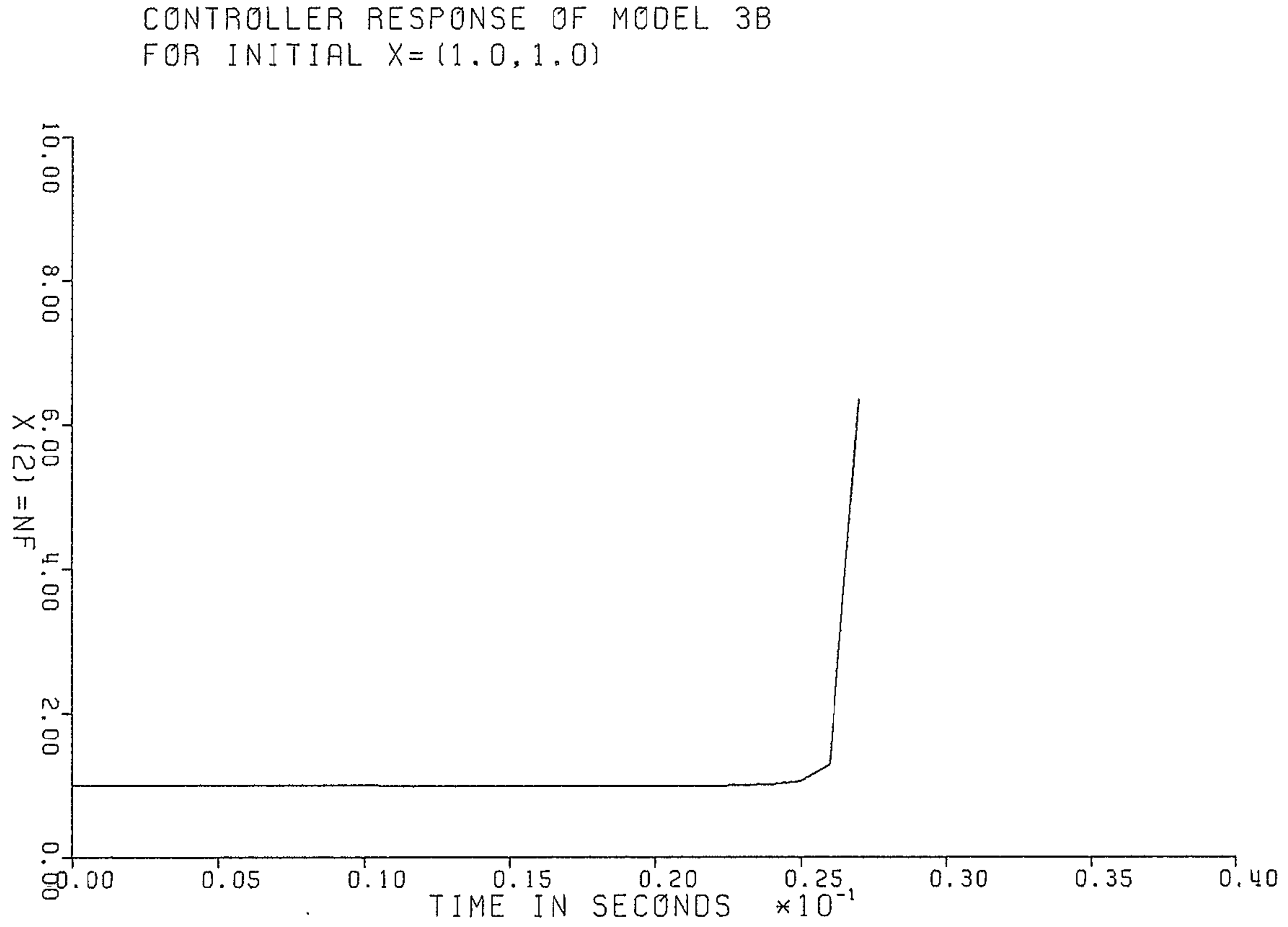

CONTROLLER RESPONSE OF MODEL 3B
FOR INITIAL X=(1.0,1.0)
X(2)=NF
10.00
8.00
6.00
4.00
2.00
0.00
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS $*10^{-1}$

Figure 8.3-2

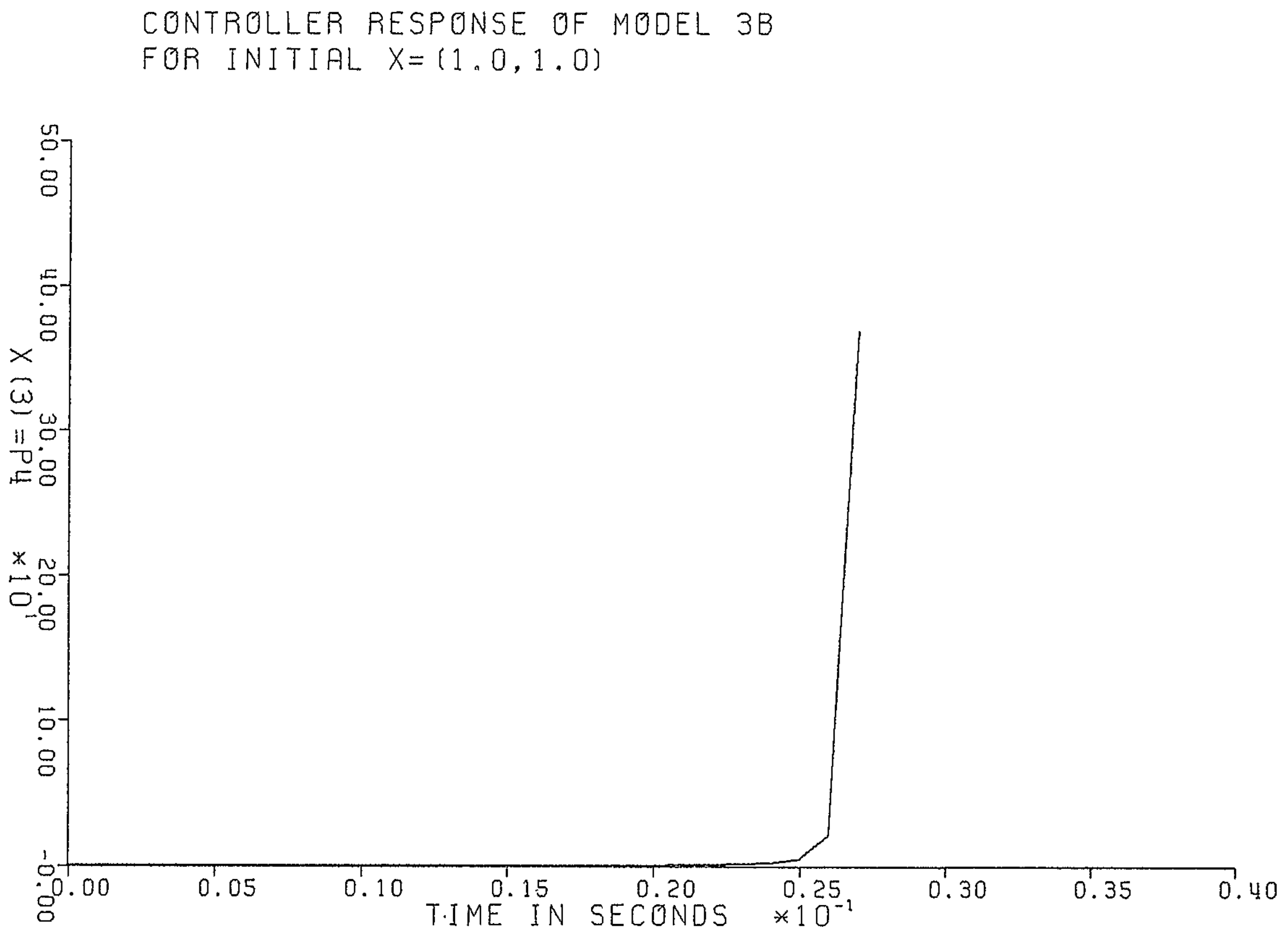

CONTROLLER RESPONSE OF MODEL 3B
FOR INITIAL X=(1.0,1.0)
X(3)=P4 *10^{1}
50.00
40.00
30.00
20.00
10.00
-0.00
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS *10^{-1}

Figure 8.3-3

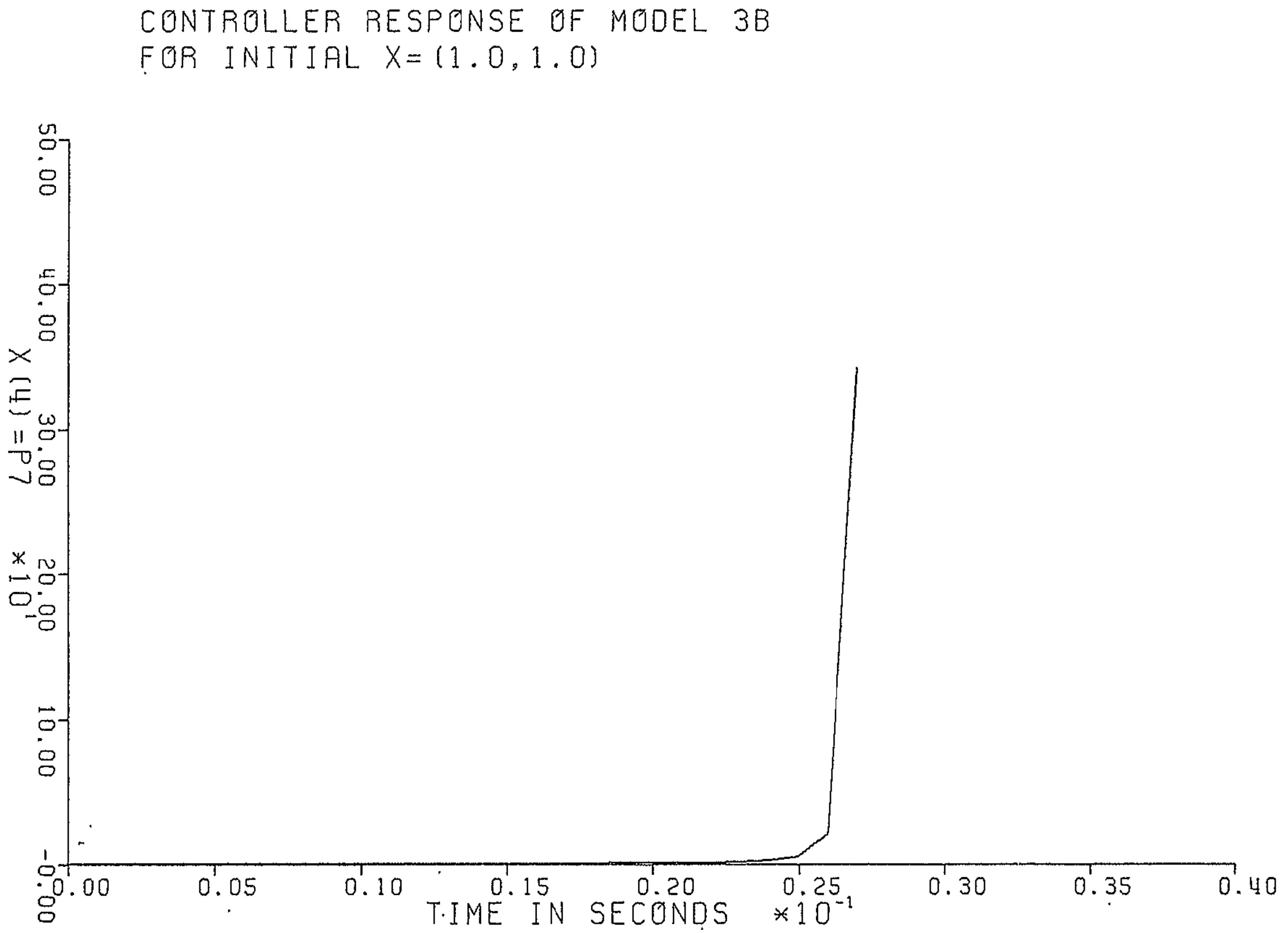
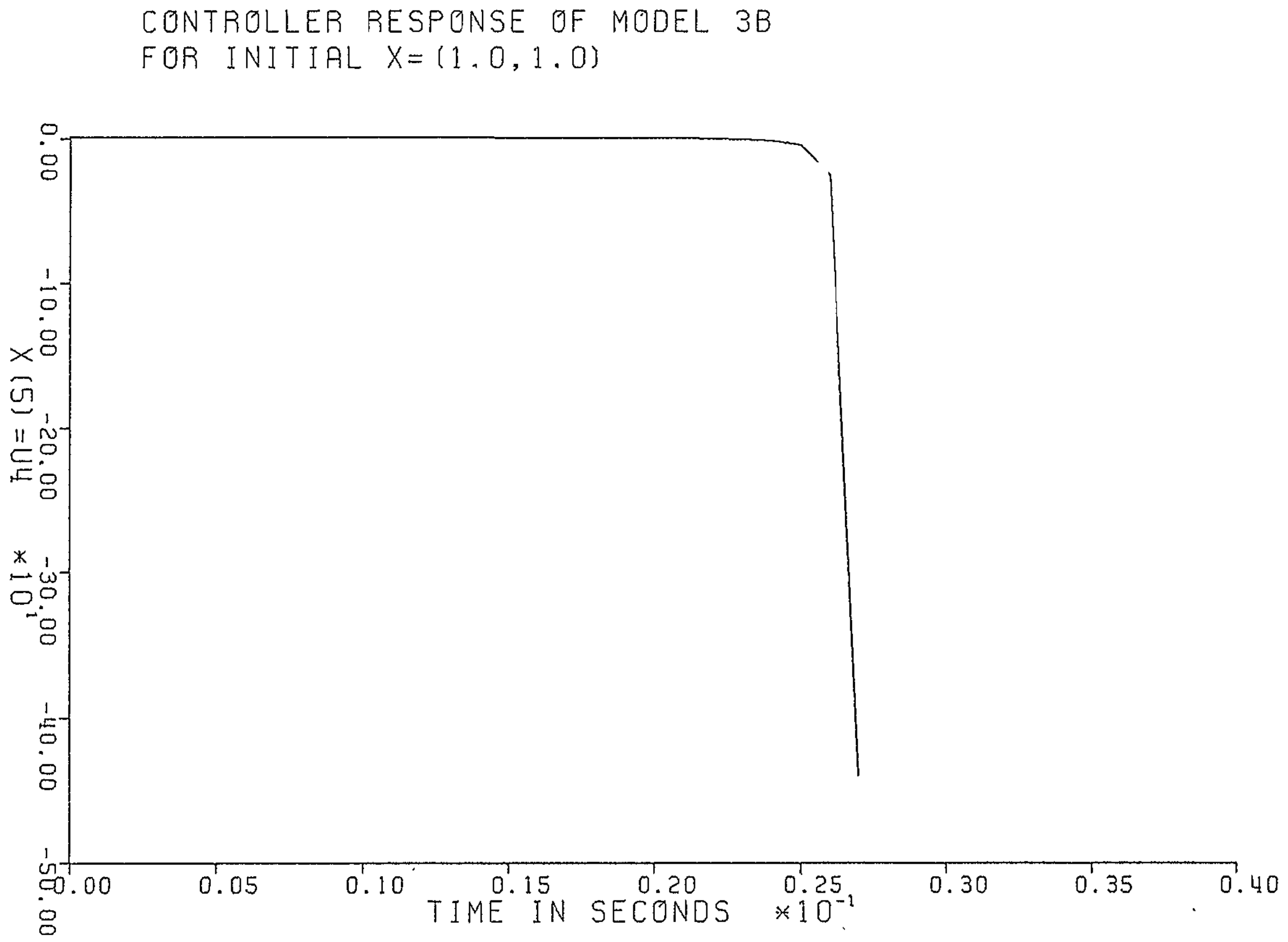

CONTROLLER RESPONSE OF MODEL 3B
FOR INITIAL X=(1.0,1.0)
X(4)=P7 ×10^1
-0.00
10.00
20.00
30.00
40.00
50.00
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS ×10^-1

Figure 8.3-4



CONTROLLER RESPONSE OF MODEL 3B
FOR INITIAL X=(1.0,1.0)
0.00
-10.00
-20.00
-30.00
-40.00
-50.00
X(5)=U4 *10^1
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS *10^-1

Figure 8.3-5

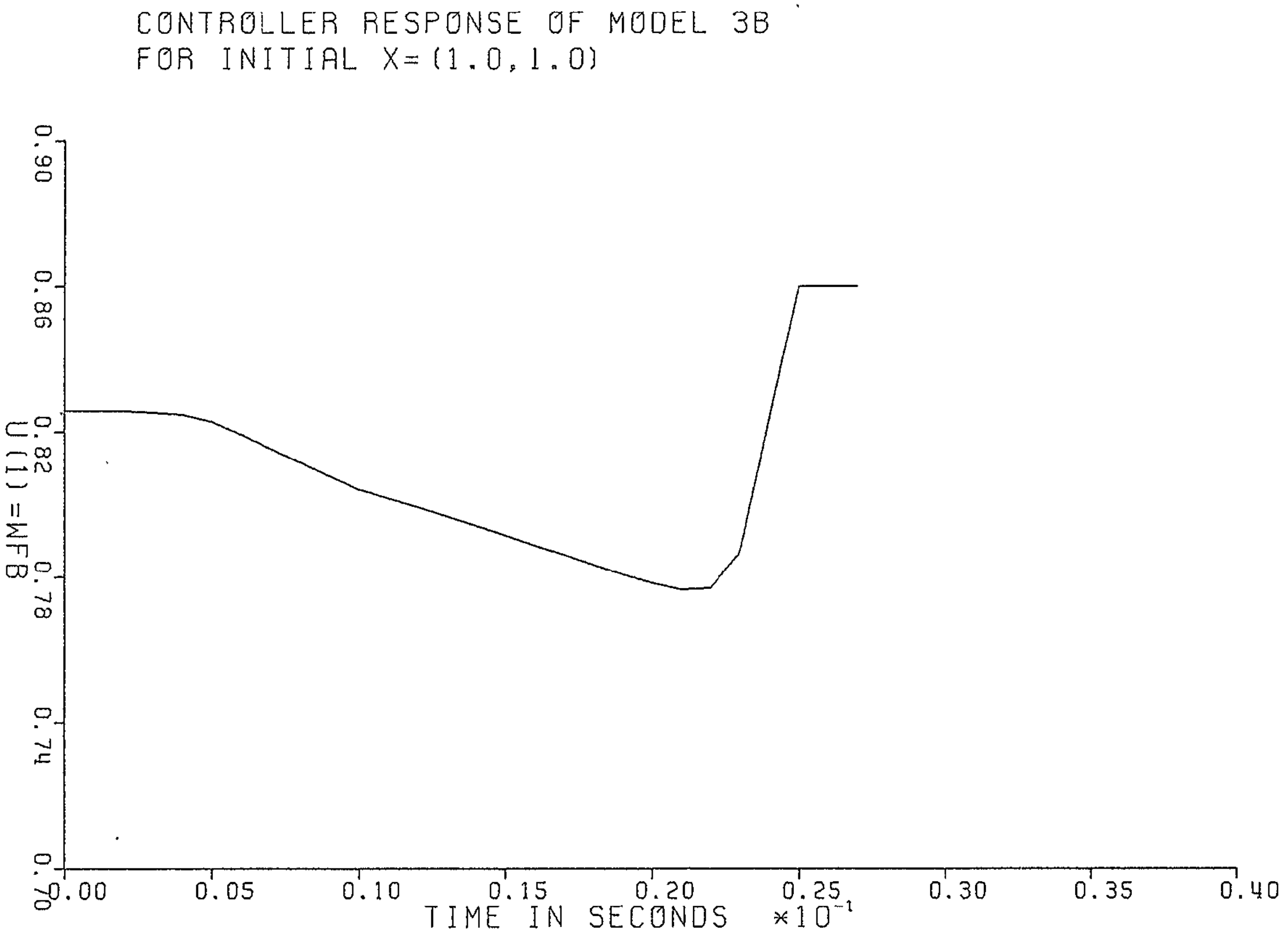

CONTROLLER RESPONSE OF MODEL 3B
FOR INITIAL X=(1.0,1.0)
U(1)=WFB
0.70
0.74
0.78
0.82
0.86
0.90
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS *10⁻¹

Figure 8.3-6

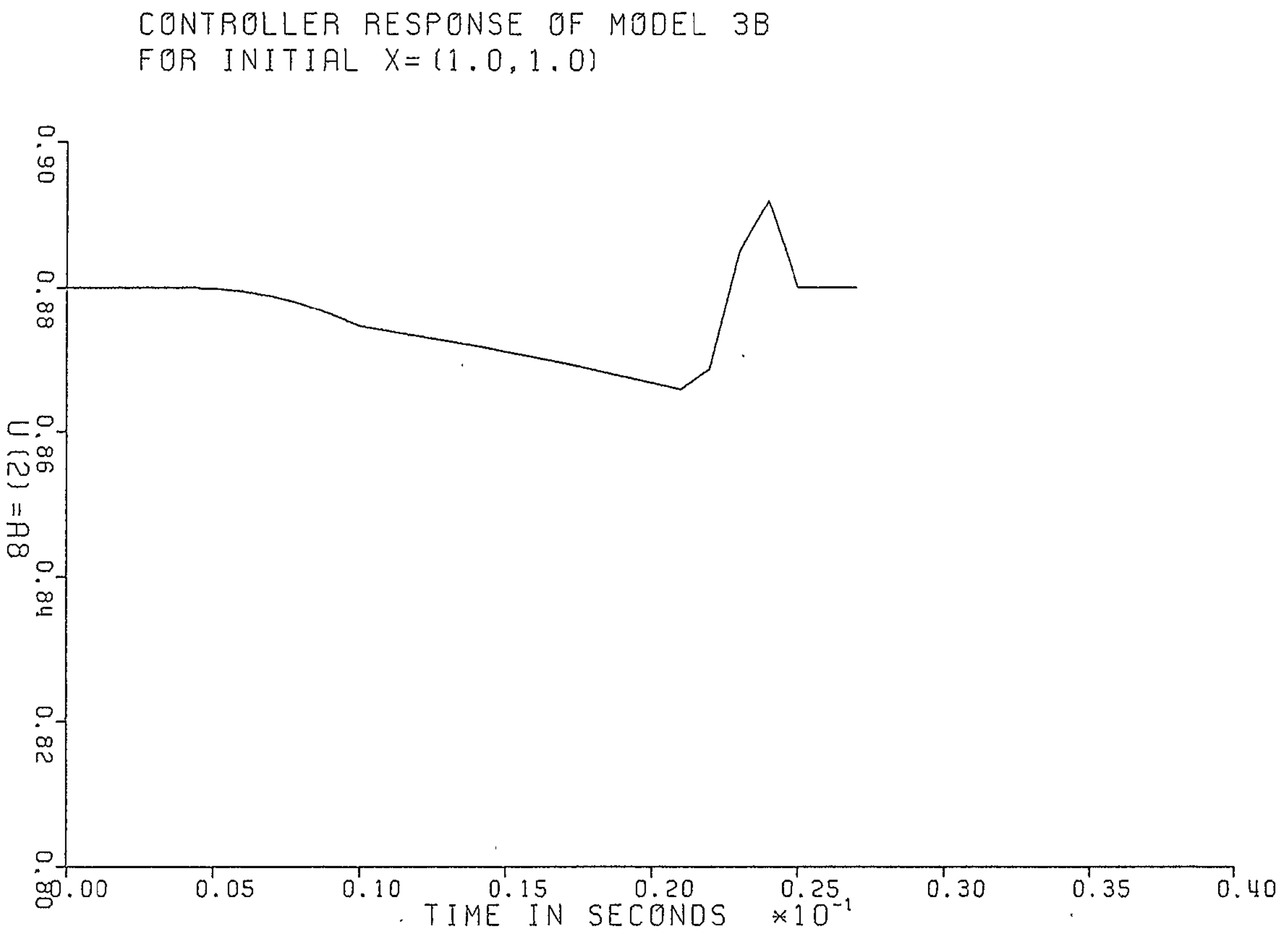

CONTROLLER RESPONSE OF MODEL 3B
FOR INITIAL X=(1.0,1.0)
U(2)=A8
0.80
0.82
0.84
0.86
0.88
0.90
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS ×10⁻¹

Figure 8.3-7

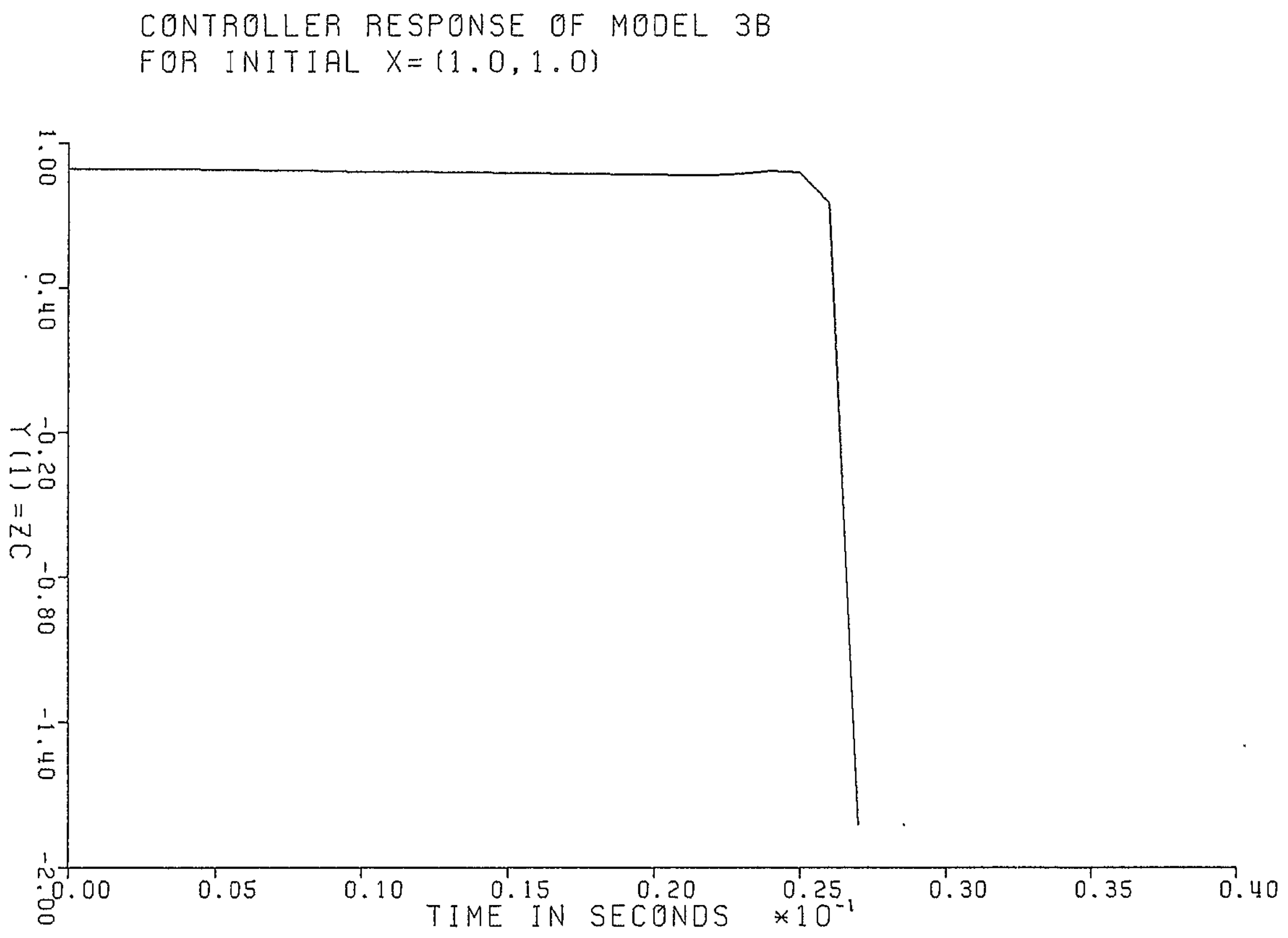

CONTROLLER RESPONSE OF MODEL 3B
FOR INITIAL X=(1.0,1.0)
1.00
0.40
-0.20
-0.80
-1.40
-2.00
Y(1)=ZC
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS ×10⁻¹

Figure 8.3-8

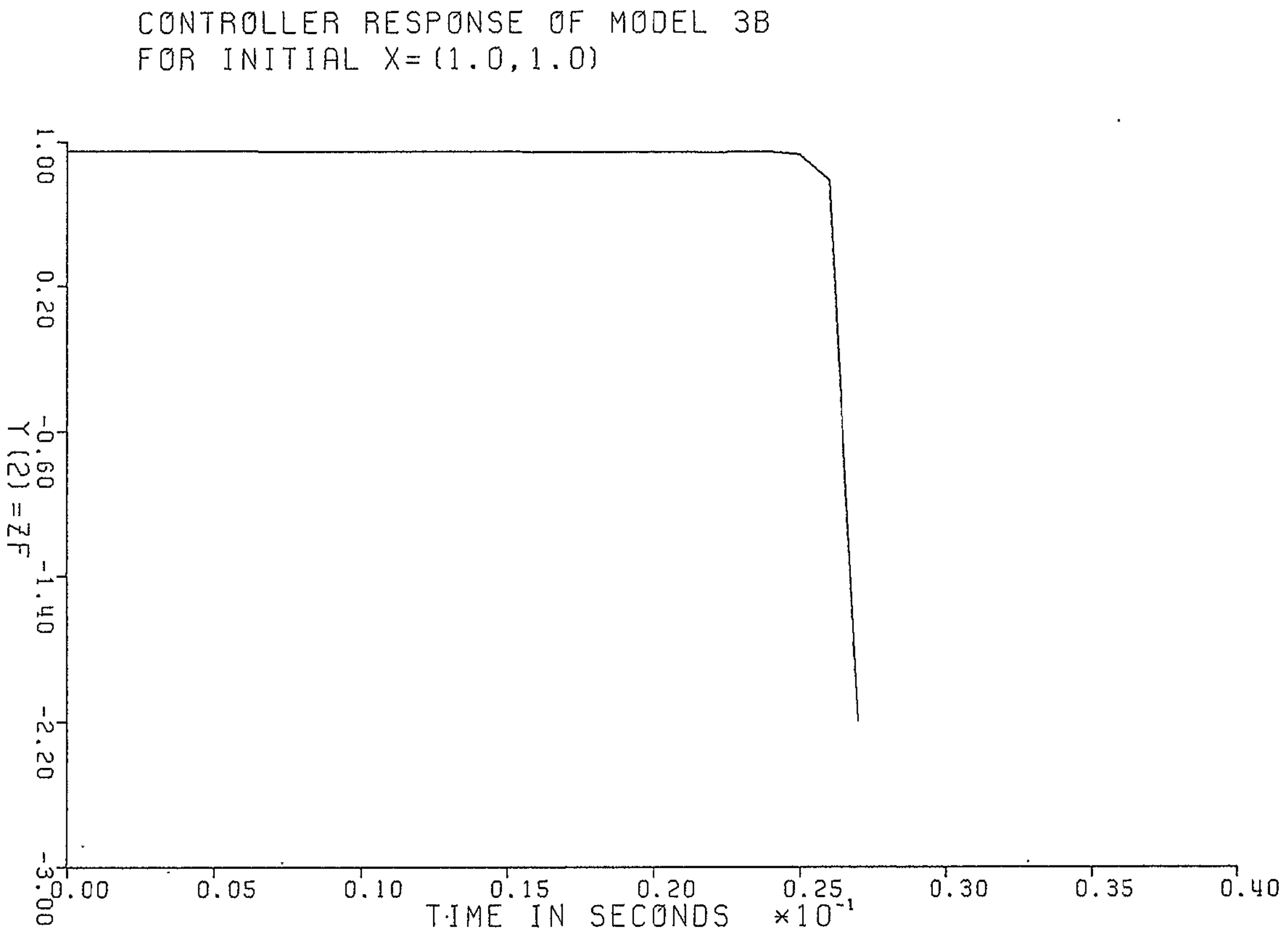

CONTROLLER RESPONSE OF MODEL 3B
FOR INITIAL X=(1.0,1.0)
1.00
0.20
-0.60
-1.40
-2.20
-3.00
Y(2)=ZF
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS *10⁻¹

Figure 8.3-9

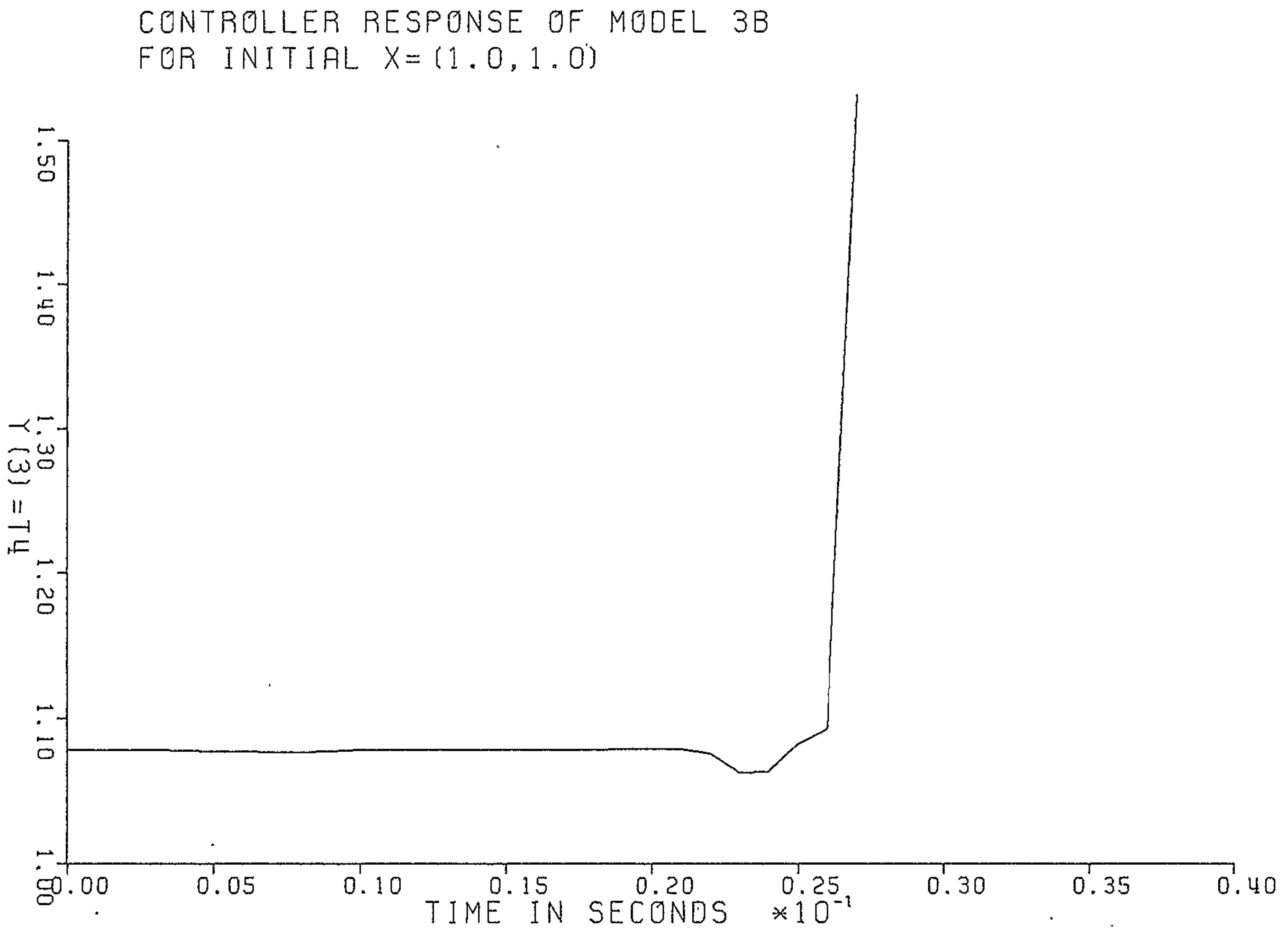

CONTROLLER RESPONSE OF MODEL 3B
FOR INITIAL X=(1.0,1.0)
Y(3)=T4
1.00
1.10
1.20
1.30
1.40
1.50
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TIME IN SECONDS $\times 10^{-1}$

Figure 8.3-10

8.4 Model 4 with 4-Controller

This simulation yields a response that is stable. The states x(1) and x(2) converge to the target values after a small overshoot, and the outputs remain safely below constraint values. The states x(3), x(4), and x(5) display spikes and larger overshoots, but remain within acceptable limits. However, the time estimate of 0.2642 seconds provided by the dynamic programming algorithm differs greatly from the actual target time of 0.5880 seconds. This can be attributed to the fact that the cost function is not linear, but its computation in the dynamic programming algorithm is by means of linear interpolation. Small errors in cost estimates near the target are both propagated and compounded outward through the state space.

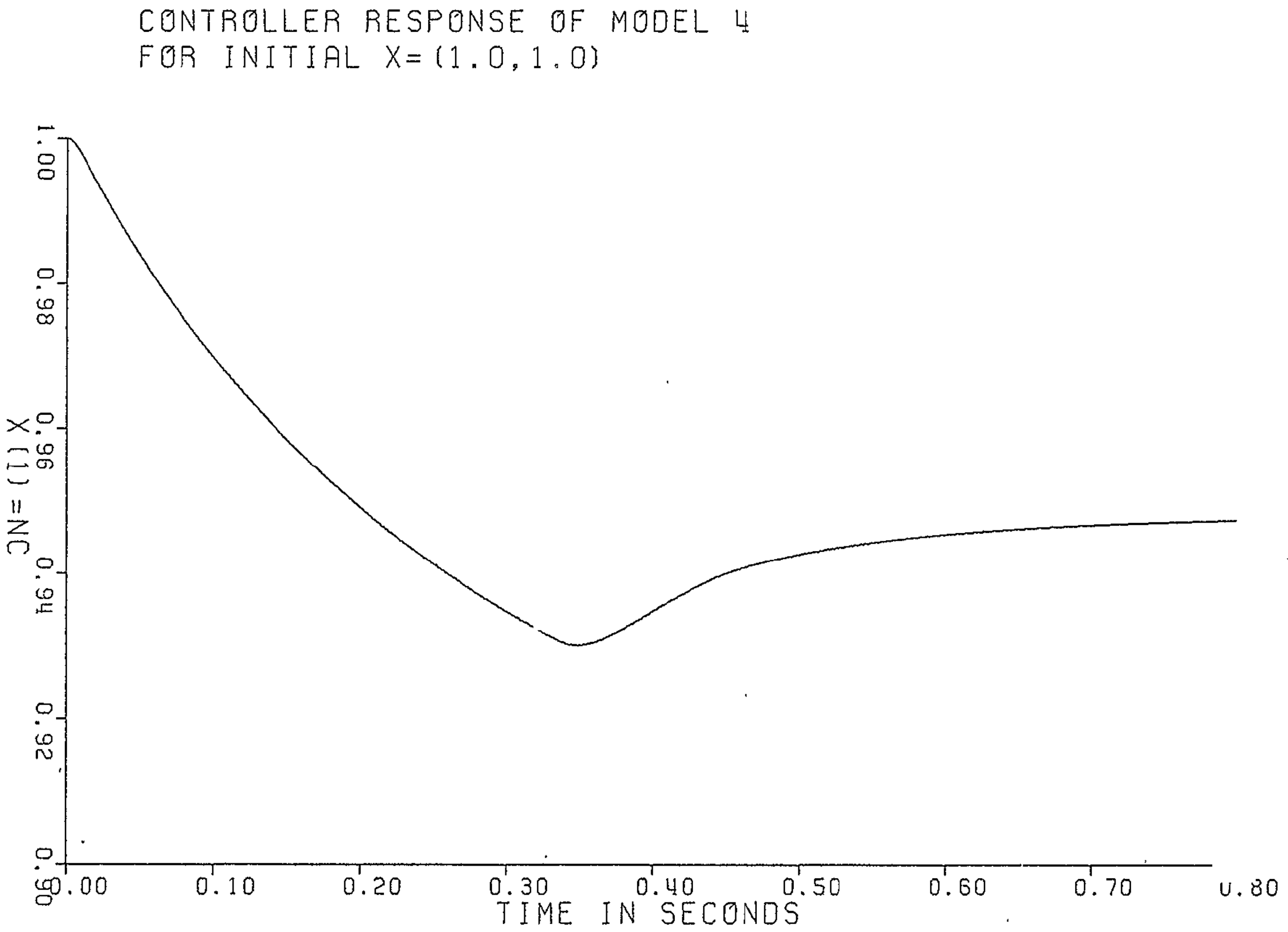
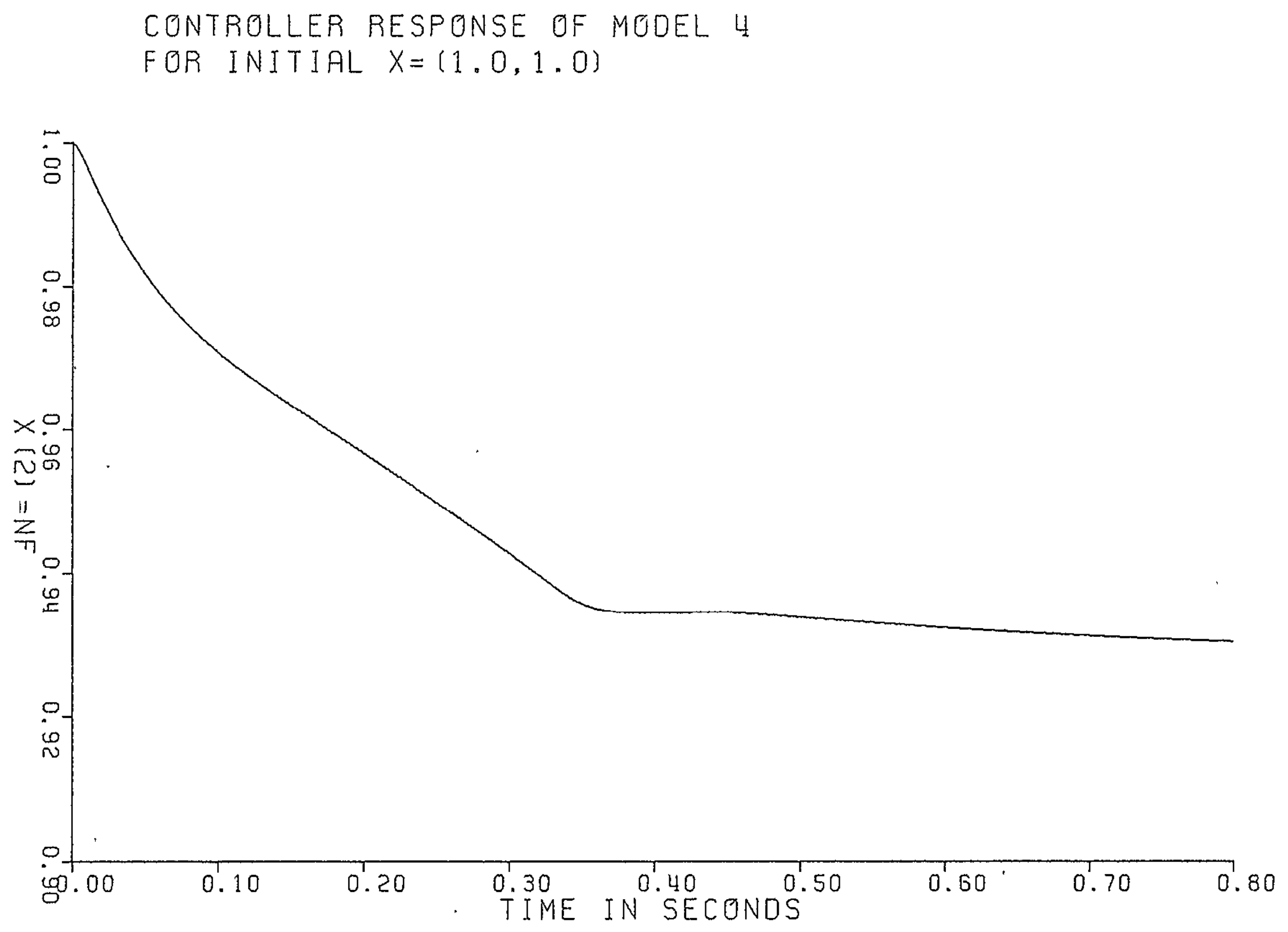

CONTROLLER RESPONSE OF MODEL 4
FOR INITIAL X=(1.0,1.0)
X(1)=NC
1.00
0.98
0.96
0.94
0.92
0.90
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.4-1



CONTROLLER RESPONSE OF MODEL 4
FOR INITIAL X=(1.0,1.0)
1.00
0.98
0.96
0.94
0.92
0.90
X(2)=NF
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.4-2

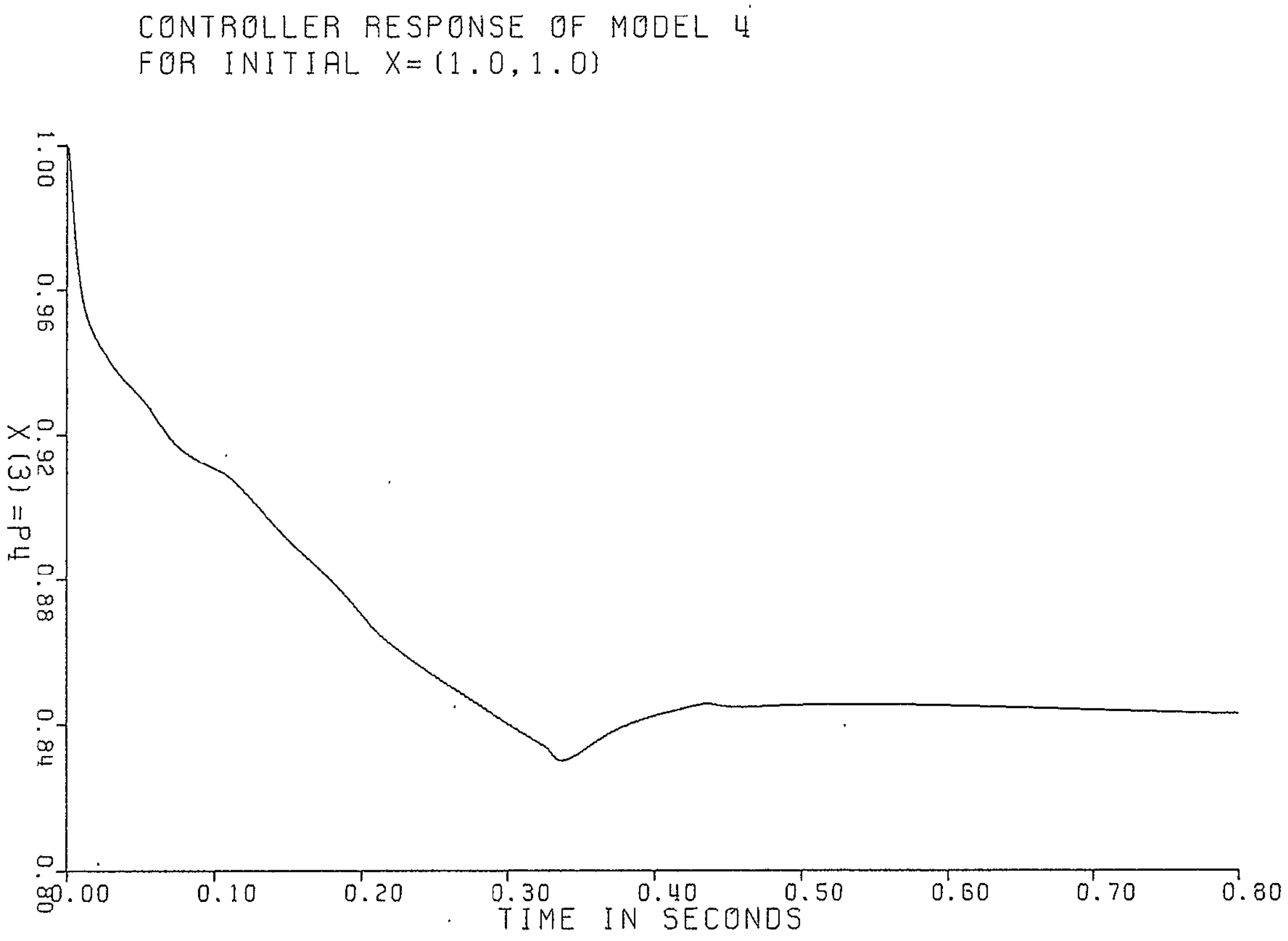

CONTROLLER RESPONSE OF MODEL 4
FOR INITIAL X=(1.0,1.0)
X(3)=P4
1.00
0.96
0.92
0.88
0.84
0.80
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.4-3

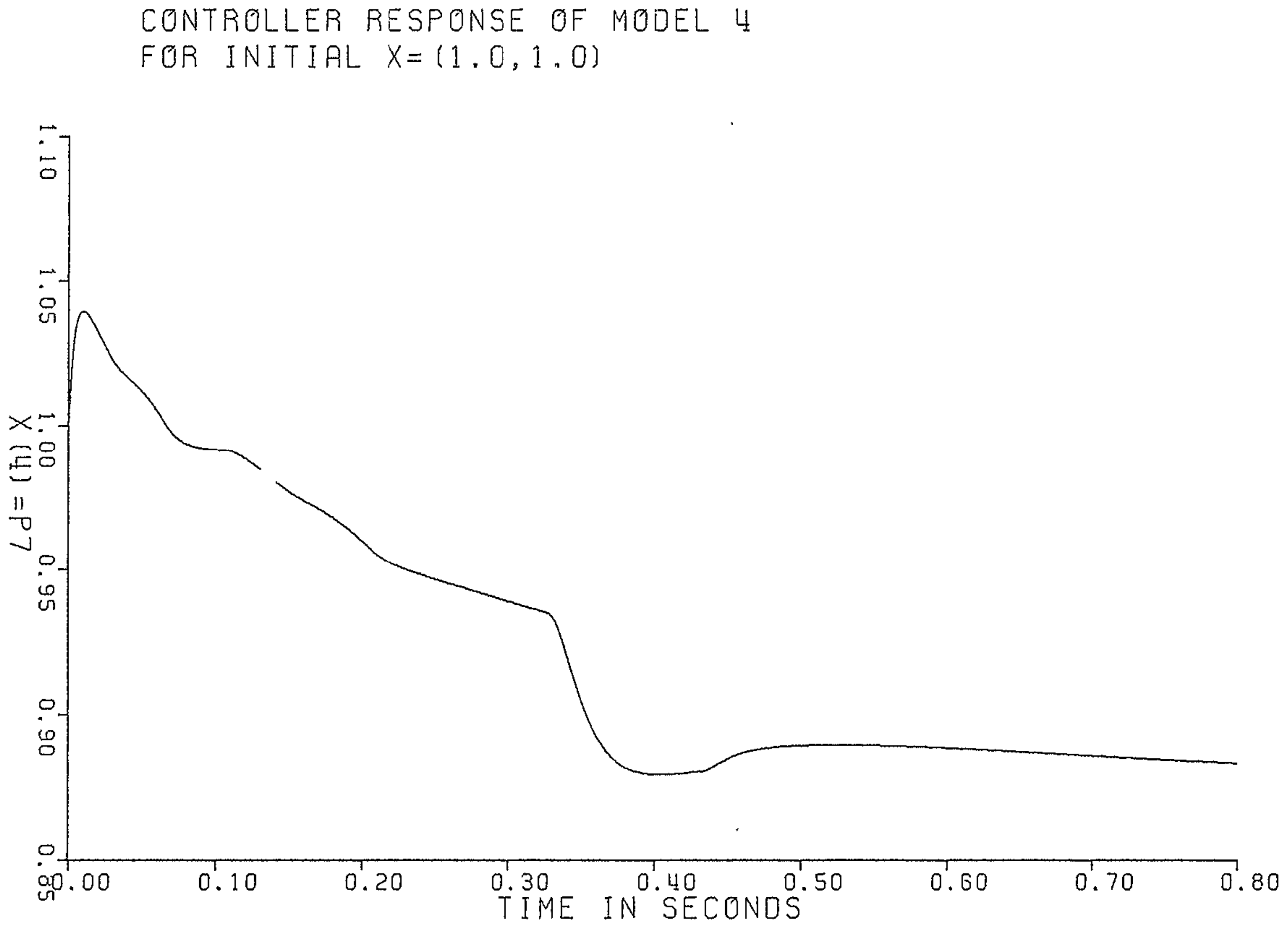

CONTROLLER RESPONSE OF MODEL 4
FOR INITIAL X=(1.0,1.0)
1.10
1.05
1.00
0.95
0.90
0.85
X(4)=P7
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.4-4

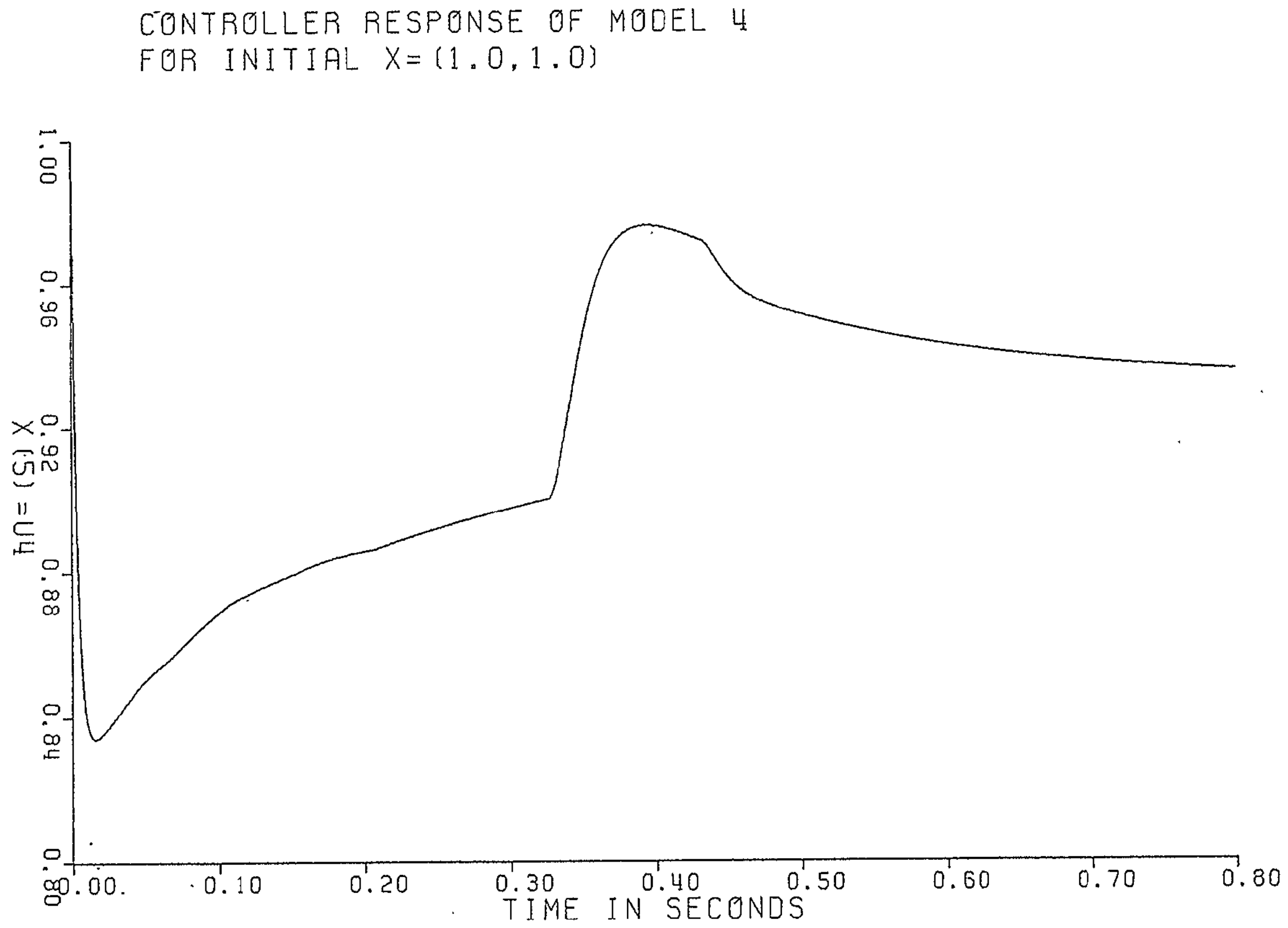
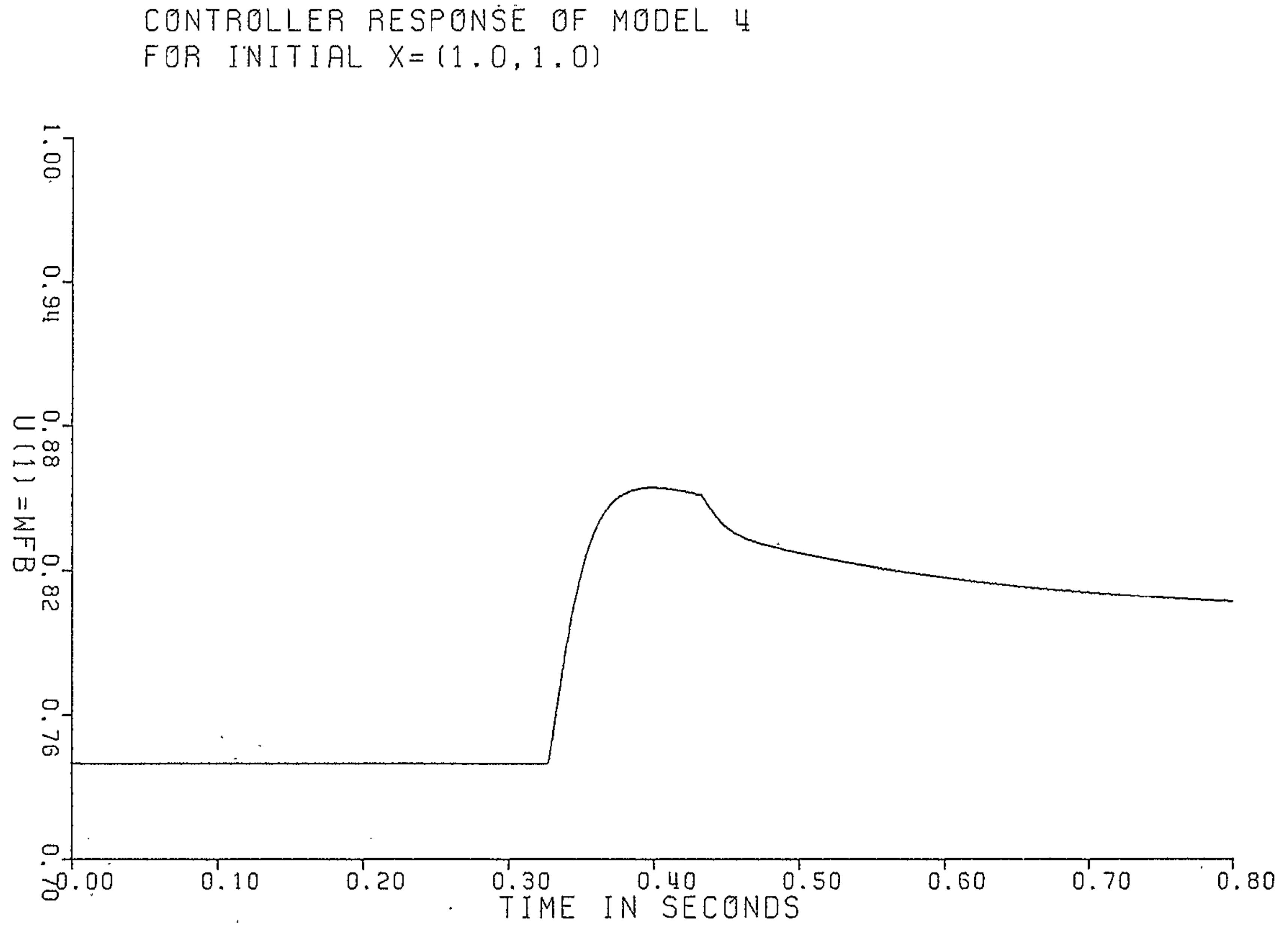

CONTROLLER RESPONSE OF MODEL 4
FOR INITIAL X=(1.0,1.0)
X(5)=U4
0.80
0.84
0.88
0.92
0.96
1.00
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.4-5



C-9

Figure 8.4-6

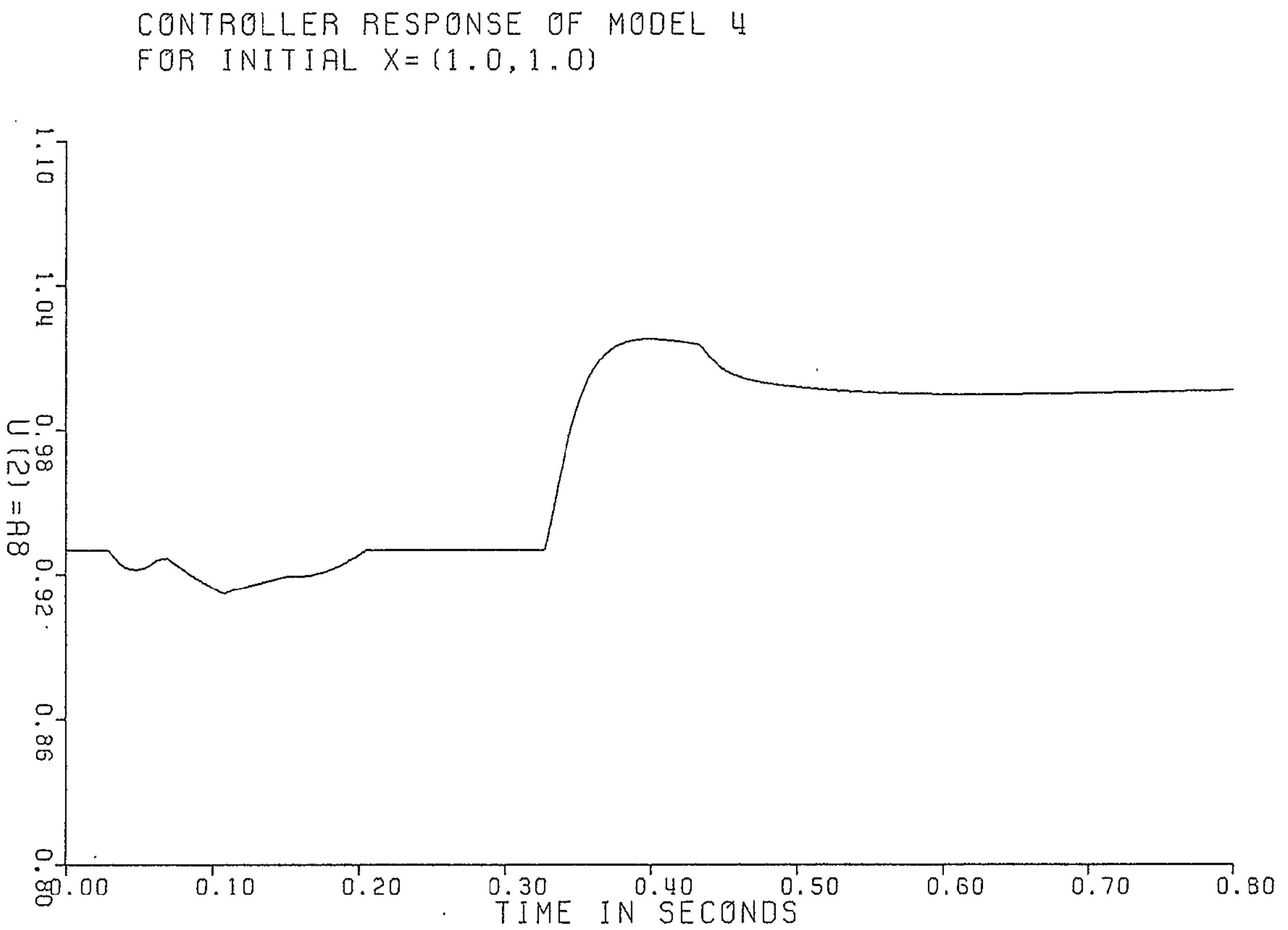
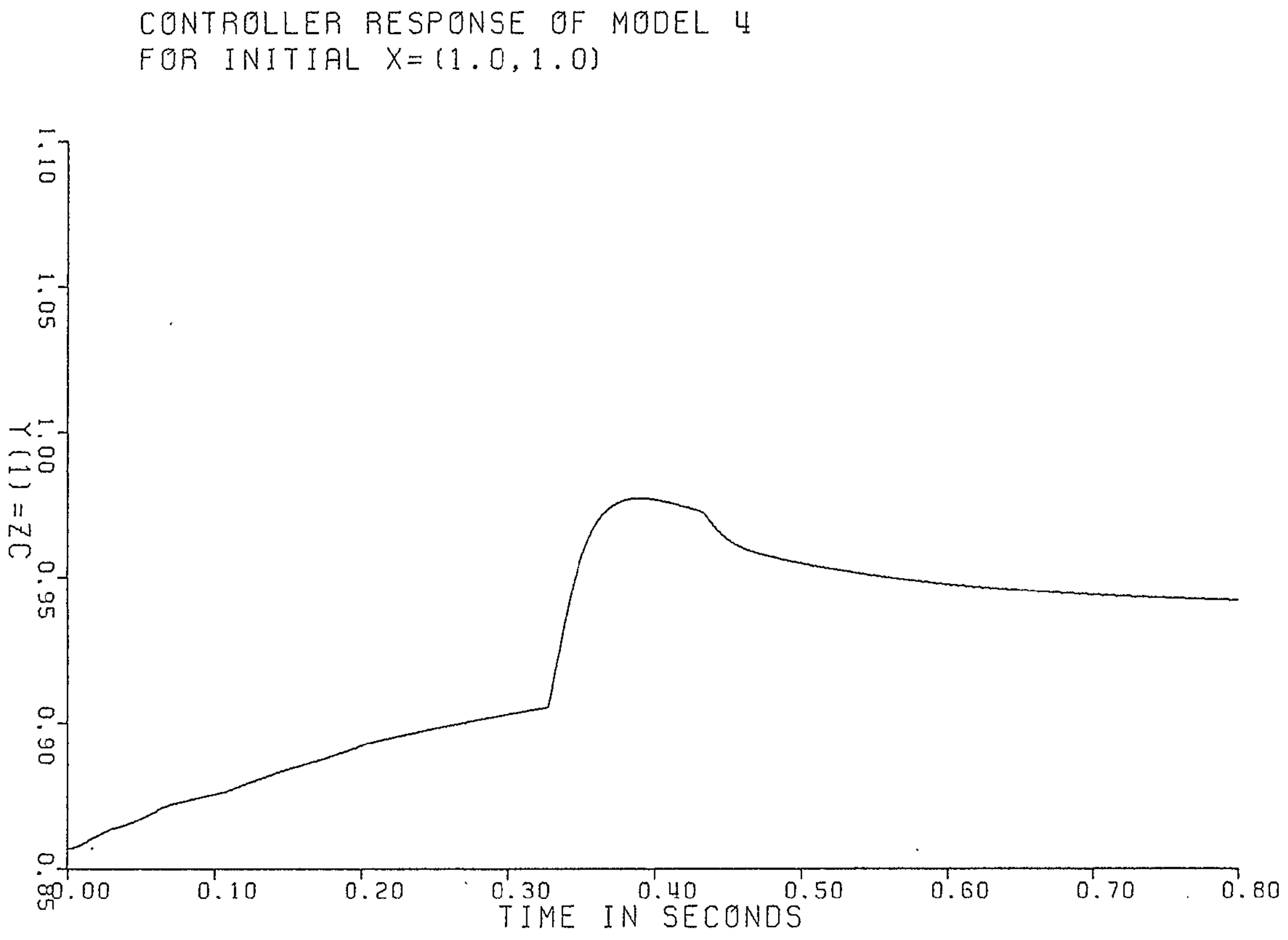

CONTROLLER RESPONSE OF MODEL 4
FOR INITIAL X=(1.0,1.0)
U(2)=A8
1.10
1.04
0.98
0.92
0.86
0.80
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.4-7



CONTROLLER RESPONSE OF MODEL 4
FOR INITIAL X=(1.0,1.0)
Y(1)=ZC
TIME IN SECONDS
0.85
0.90
0.95
1.00
1.05
1.10
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80

Figure 8.4-8

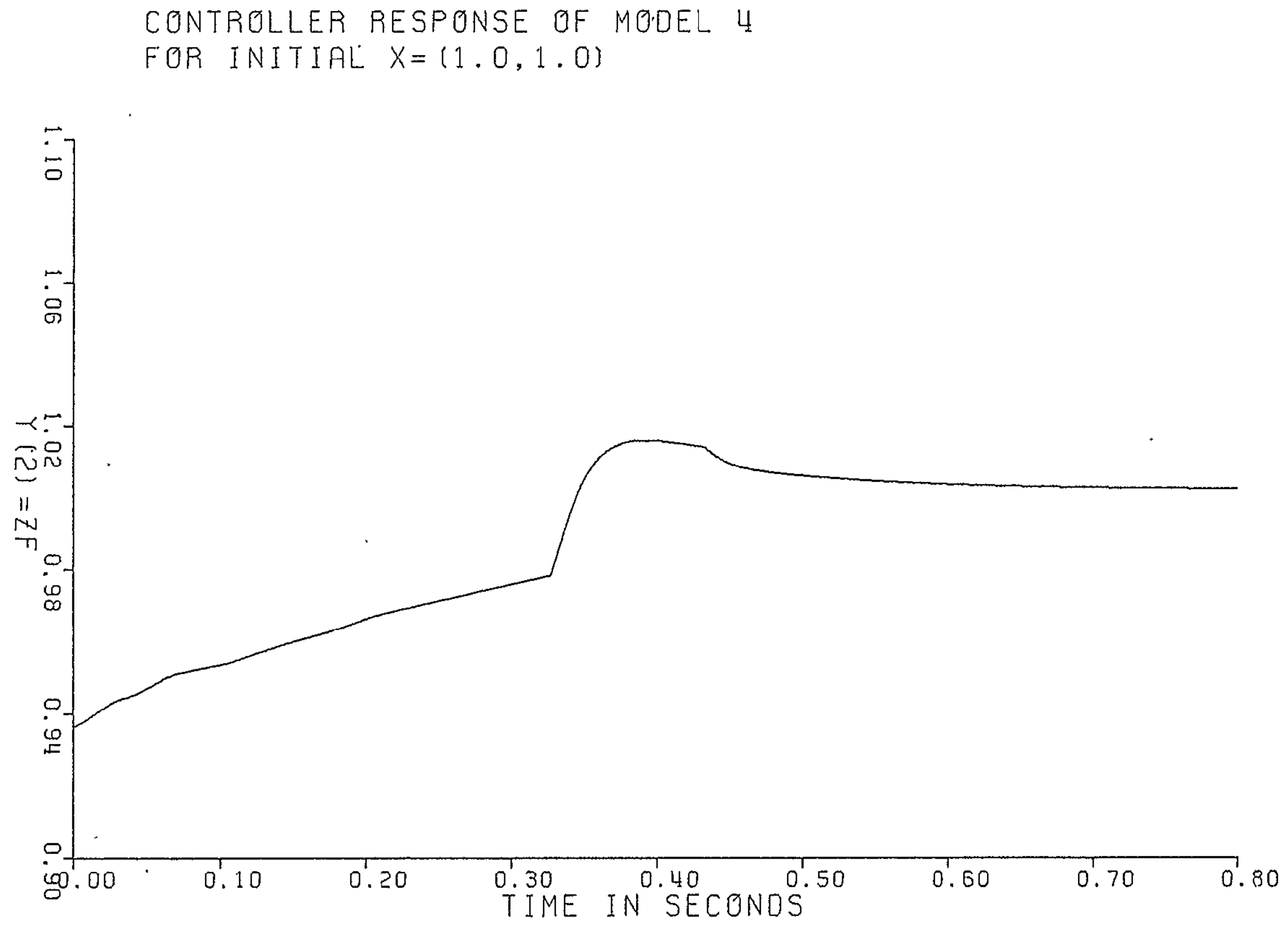

CONTROLLER RESPONSE OF MODEL 4
FOR INITIAL X=(1.0,1.0)
Y(2)=ZF
0.90
0.94
0.98
1.02
1.06
1.10
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.4-9

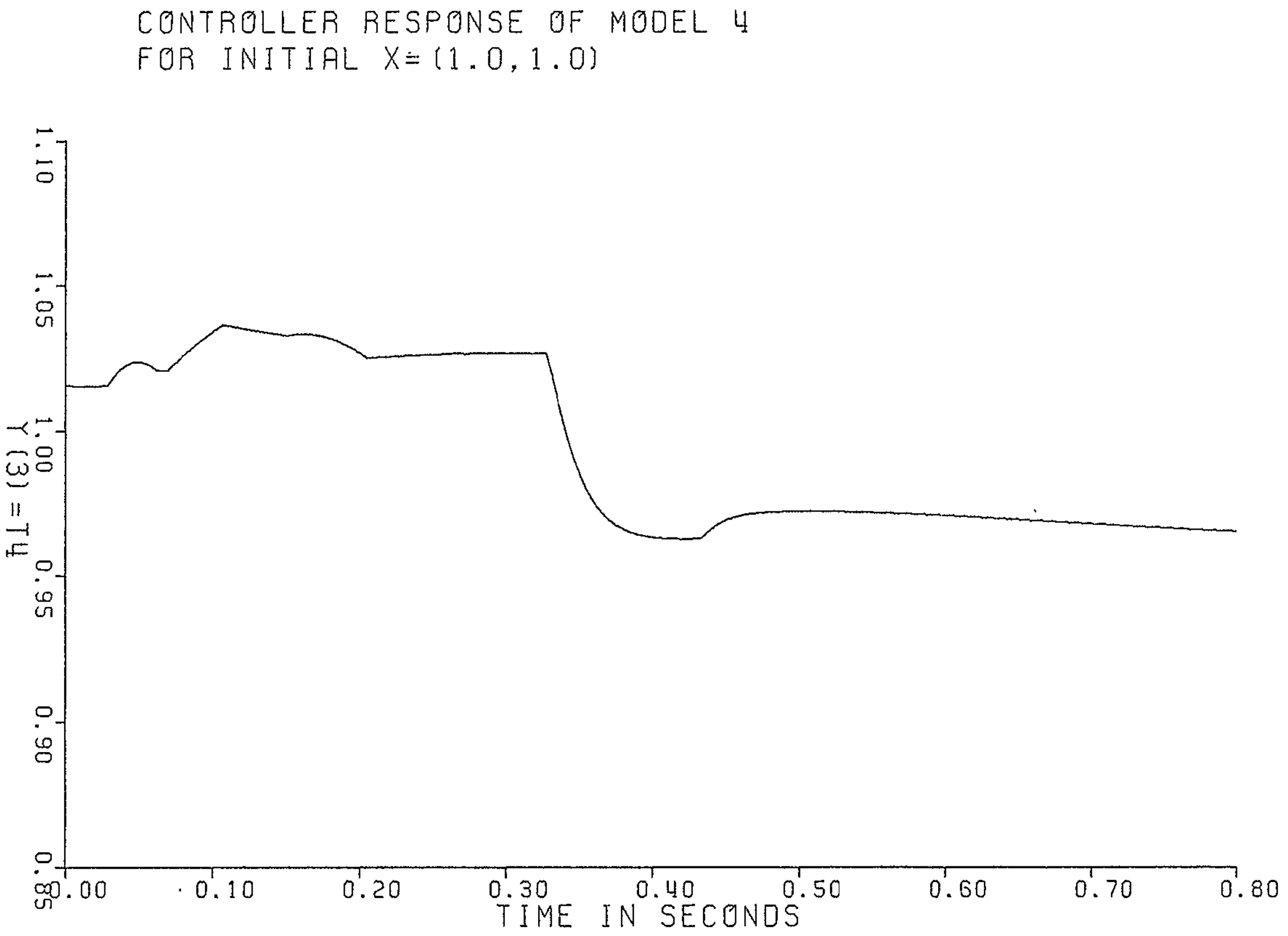

CONTROLLER RESPONSE OF MODEL 4
FOR INITIAL X≐(1.0,1.0)
Y(3)=T4
1.10
1.05
1.00
0.95
0.90
0.85
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.4-10

8.5 Model 1 with 3A-Controller

Despite the instability of model 3A with this controller, this simulation yields a smooth convergence to the target for states x(1) and x(2). The controls display the same oscillatory behavior through a portion of the simulation, but the system remains stable. Analytically derived models embody physical relationships lacking in numerical models. This, and the complex coupling of a 16 state simulator provide for greater stability in DYNGEN than in model 3A.

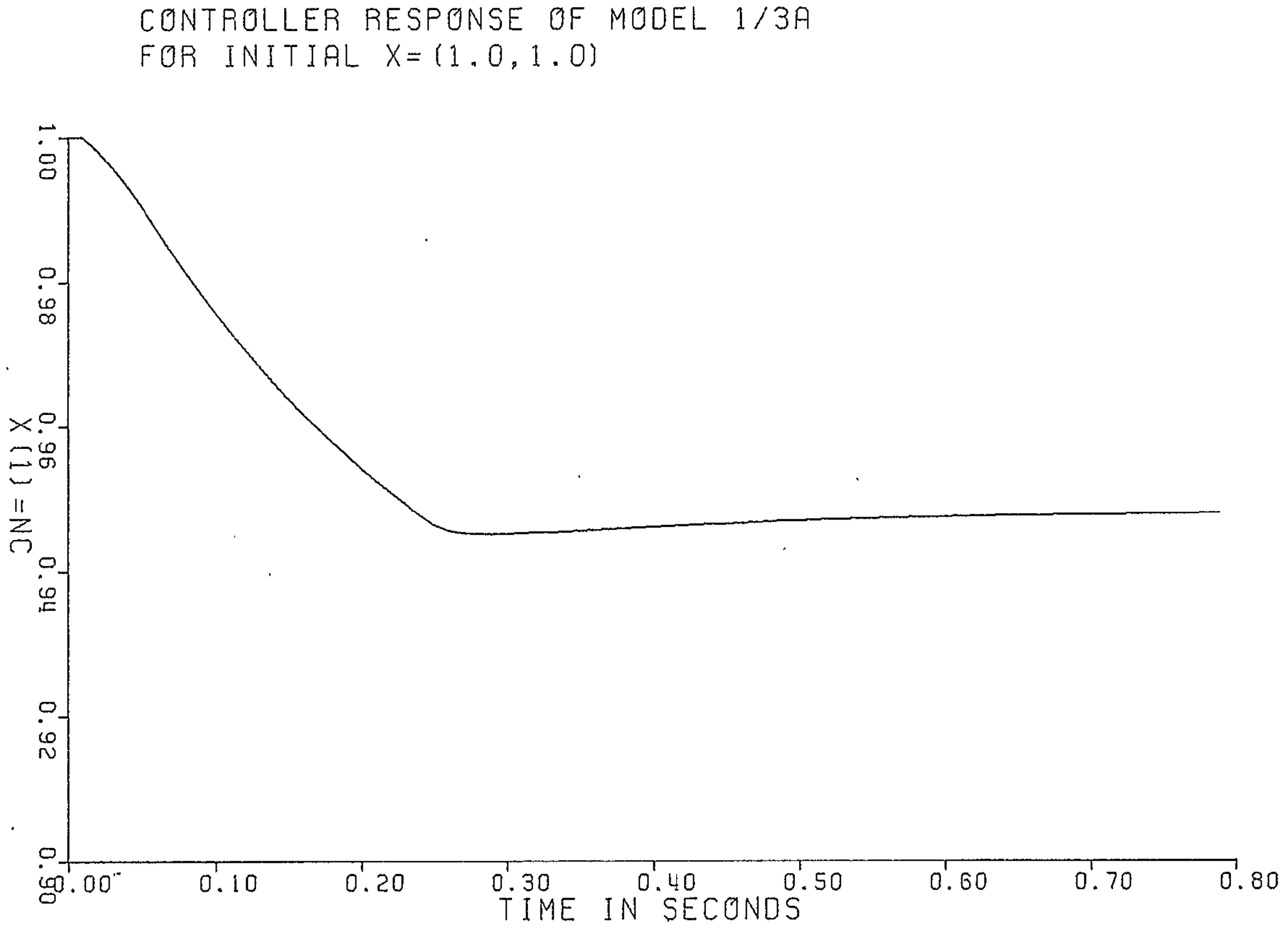

CONTROLLER RESPONSE OF MODEL 1/3A
FOR INITIAL X=(1.0,1.0)
X(1)=NC
1.00
0.98
0.96
0.94
0.92
0.90
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.5-1

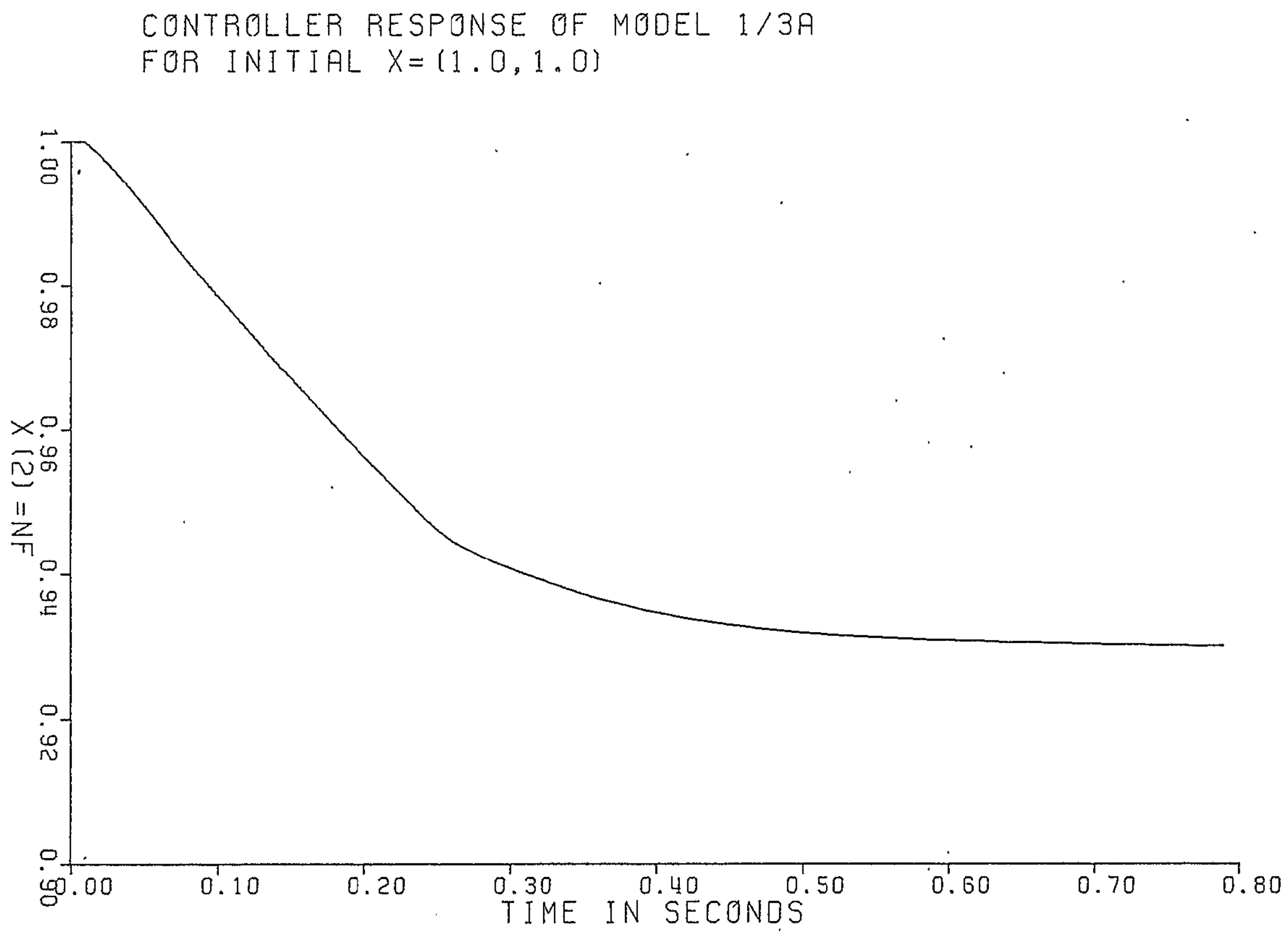

CONTROLLER RESPONSE OF MODEL 1/3A
FOR INITIAL X=(1.0,1.0)
X(2)=NF
1.00
0.98
0.96
0.94
0.92
0.90
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.5-2

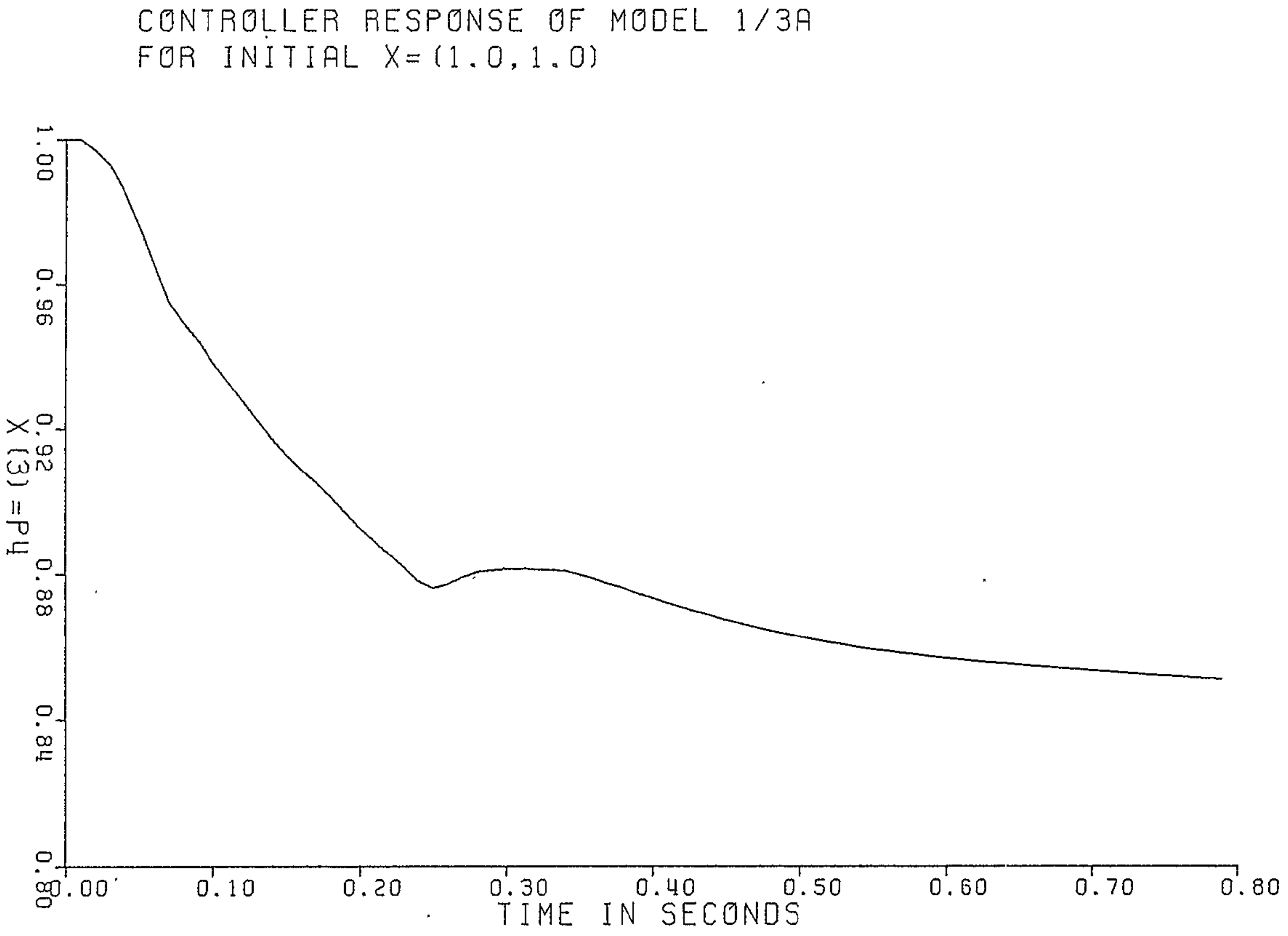

CONTROLLER RESPONSE OF MODEL 1/3A
FOR INITIAL X=(1.0,1.0)
X(3)=P4
1.00
0.96
0.92
0.88
0.84
0.80
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.5-3

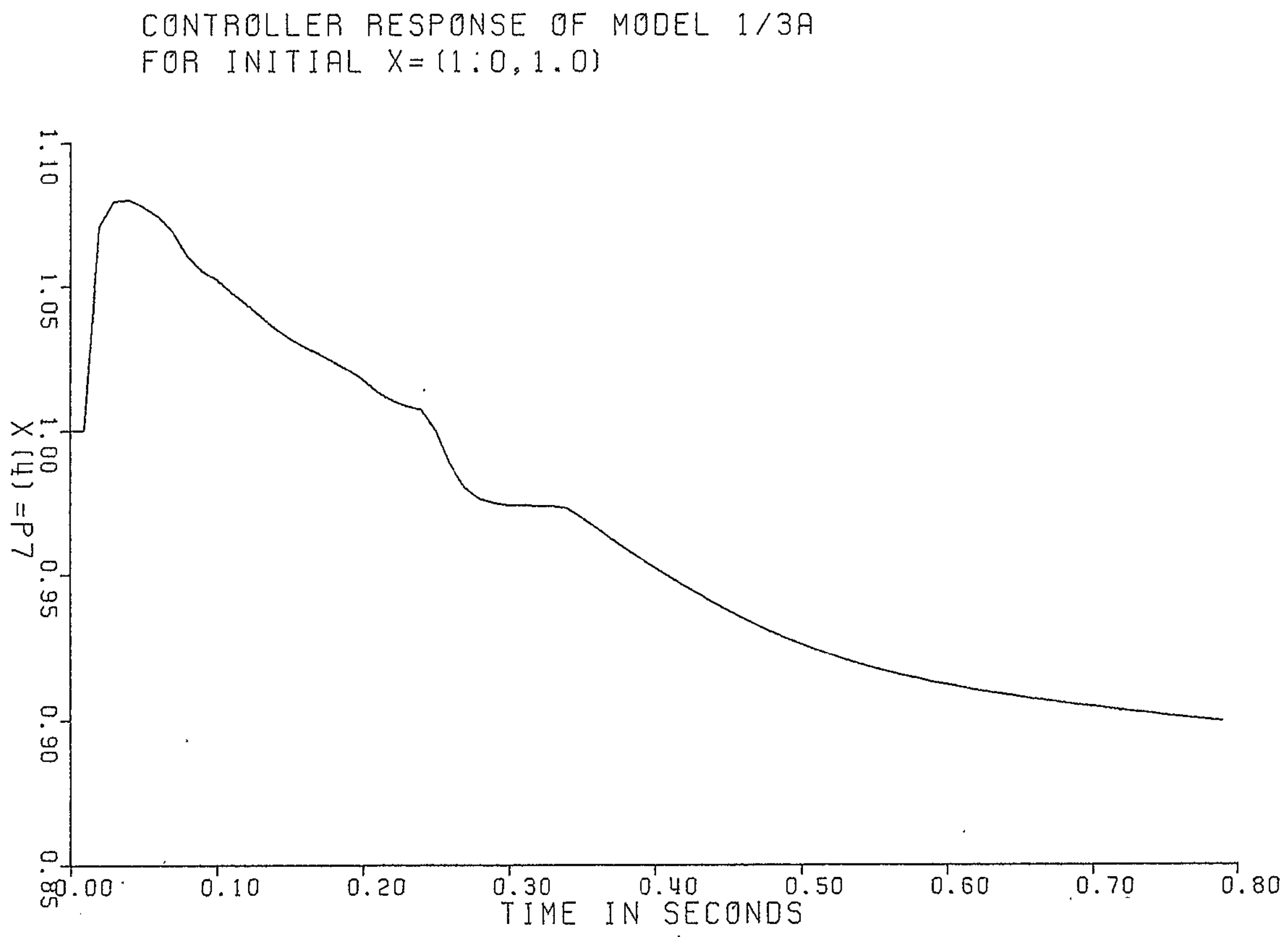

CONTROLLER RESPONSE OF MODEL 1/3A
FOR INITIAL X=(1.0,1.0)
1.10
1.05
1.00
0.95
0.90
0.85
X(4)=P7
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.5-4

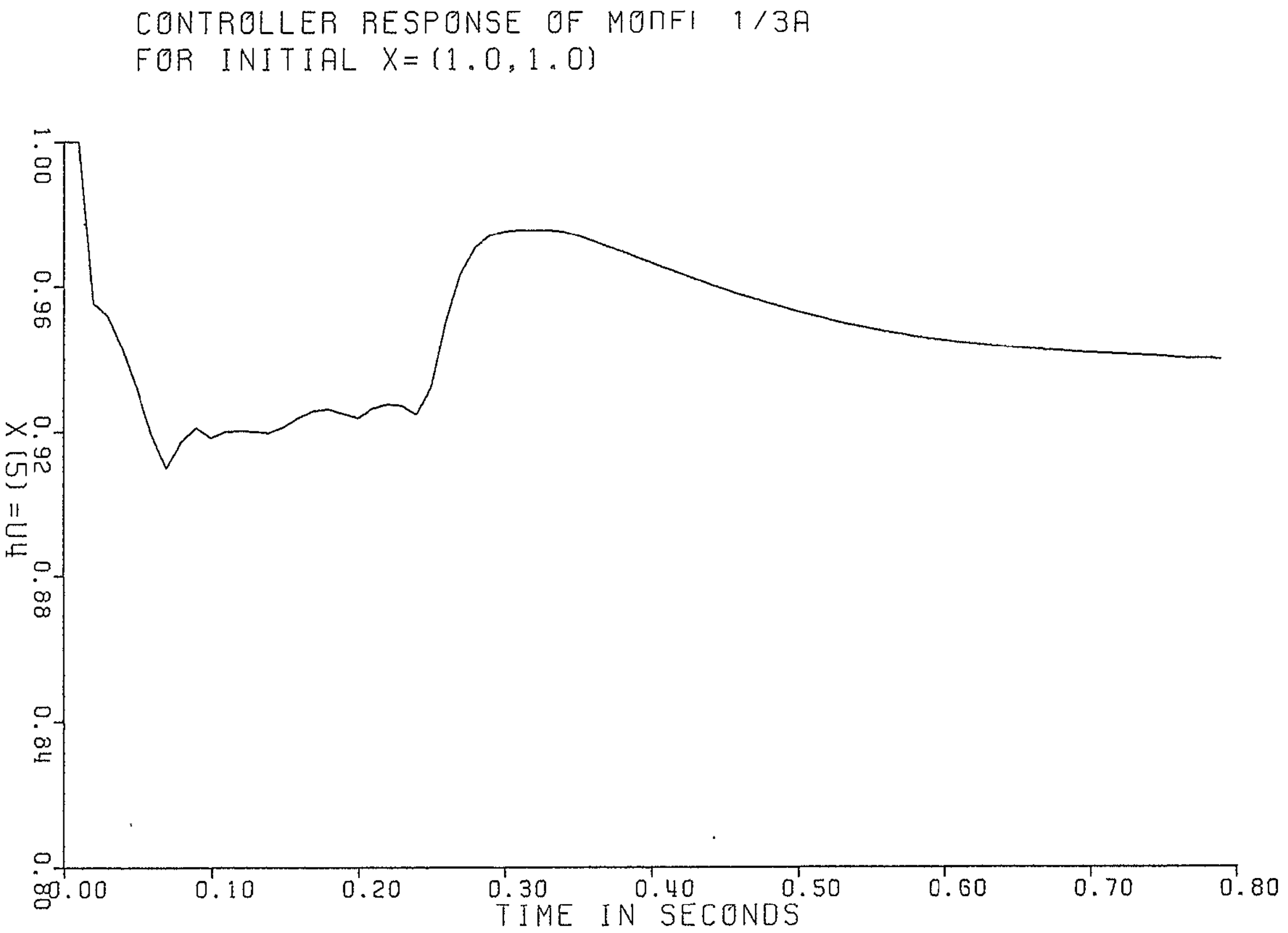

CONTROLLER RESPONSE OF MODEL 1/3A
FOR INITIAL X=(1.0,1.0)
1.00
0.96
0.92
0.88
0.84
0.80
X(5)=U4
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.5-5

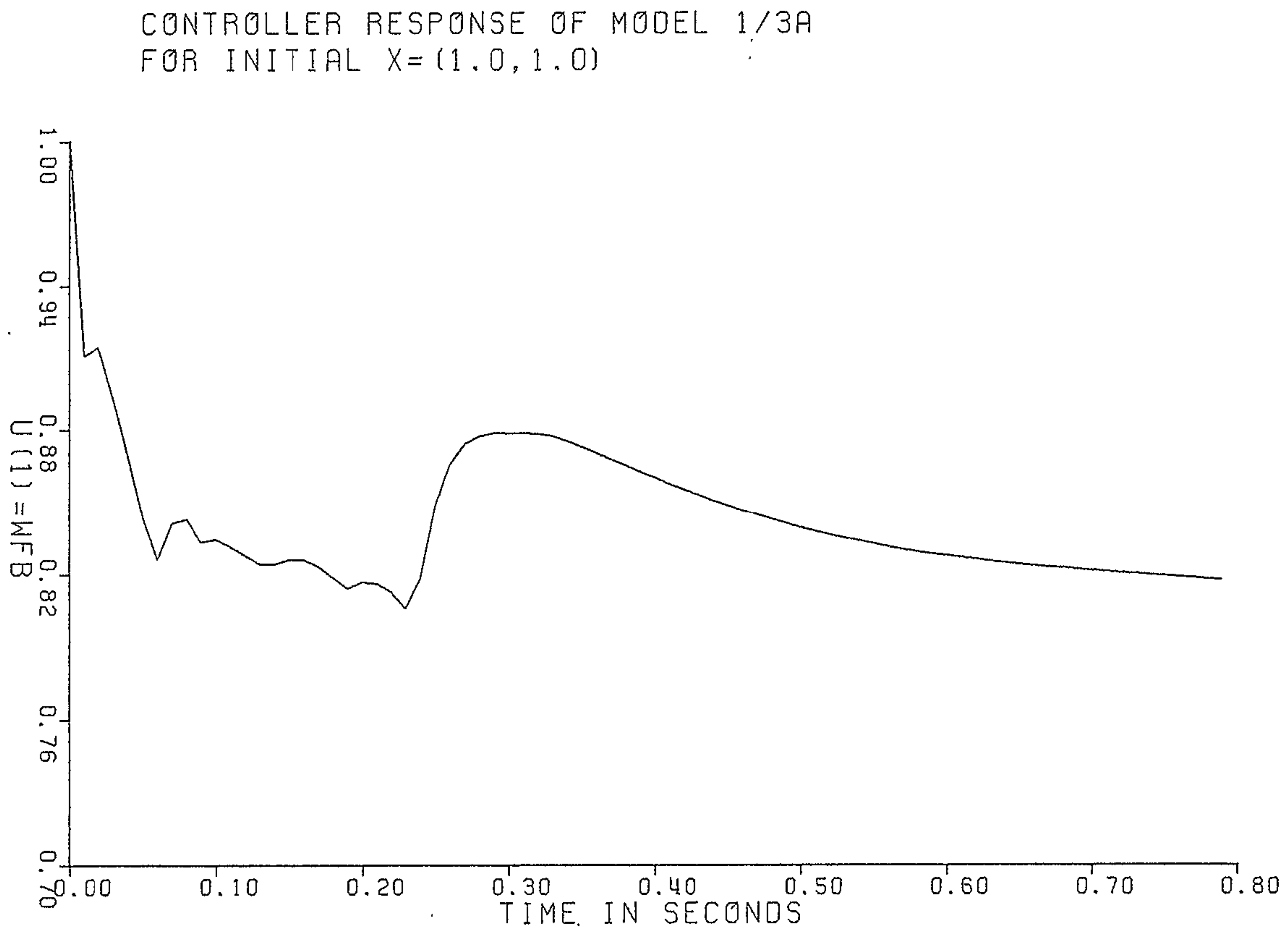

CONTROLLER RESPONSE OF MODEL 1/3A
FOR INITIAL X=(1.0,1.0)
1.00
0.94
0.88
0.82
0.76
0.70
U(1)=WFB
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.5-6

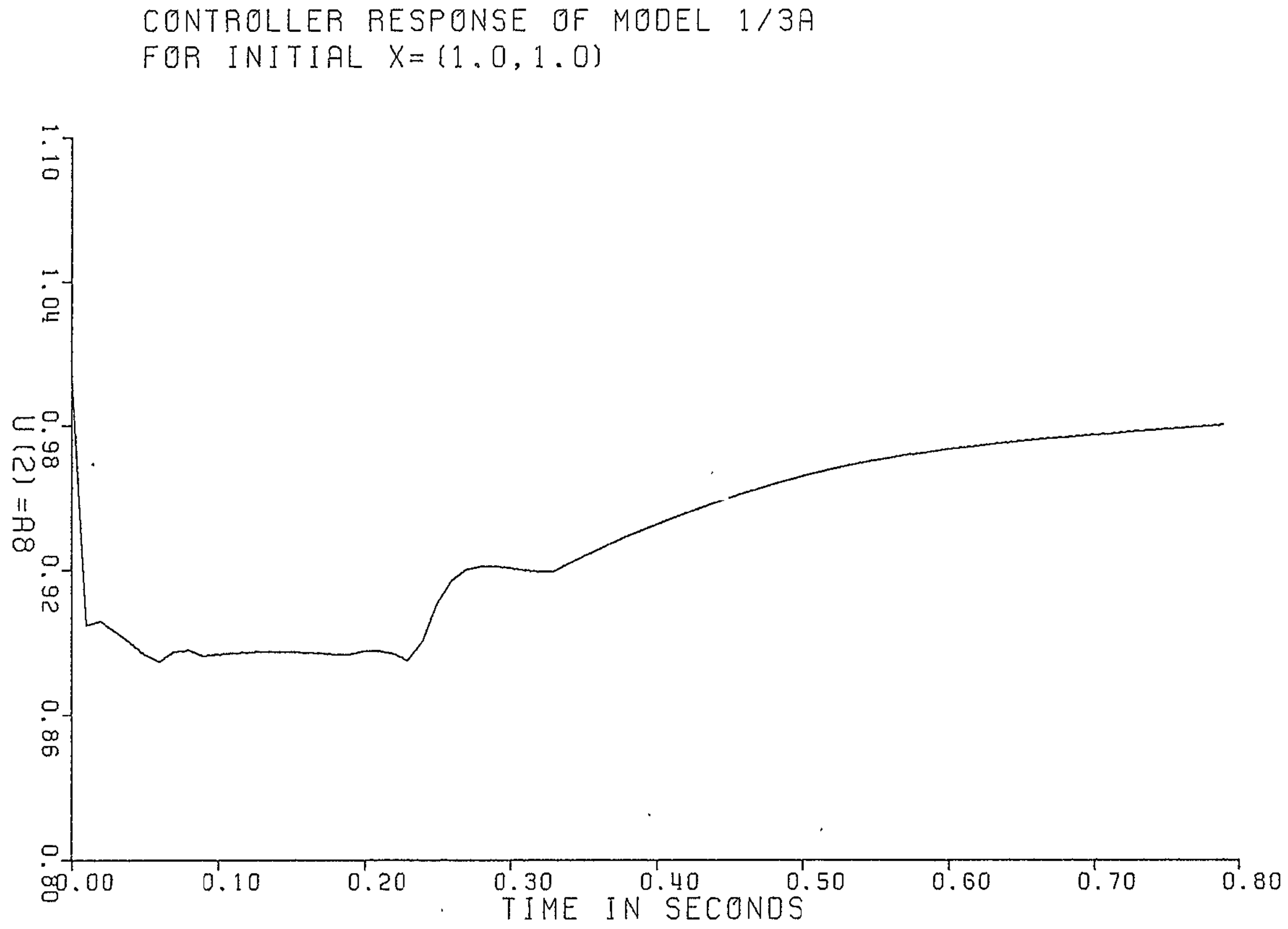

CONTROLLER RESPONSE OF MODEL 1/3A
FOR INITIAL X=(1.0,1.0)
U(2)=A8
1.10
1.04
0.98
0.92
0.86
0.80
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.5-7

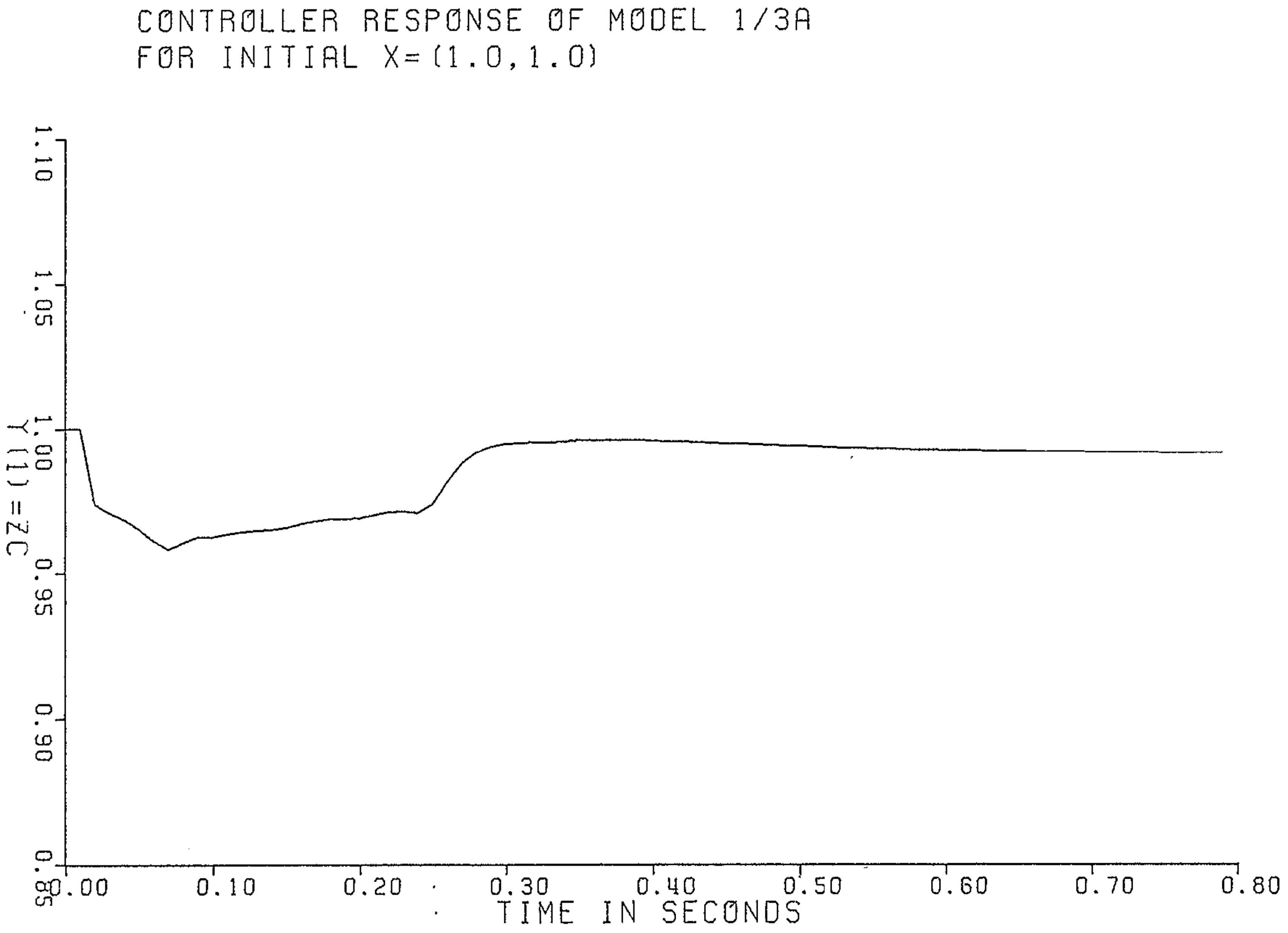

CONTROLLER RESPONSE OF MODEL 1/3A
FOR INITIAL X=(1.0,1.0)
1.10
1.05
1.00
0.95
0.90
0.85
Y(1)=ZC
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.5-8

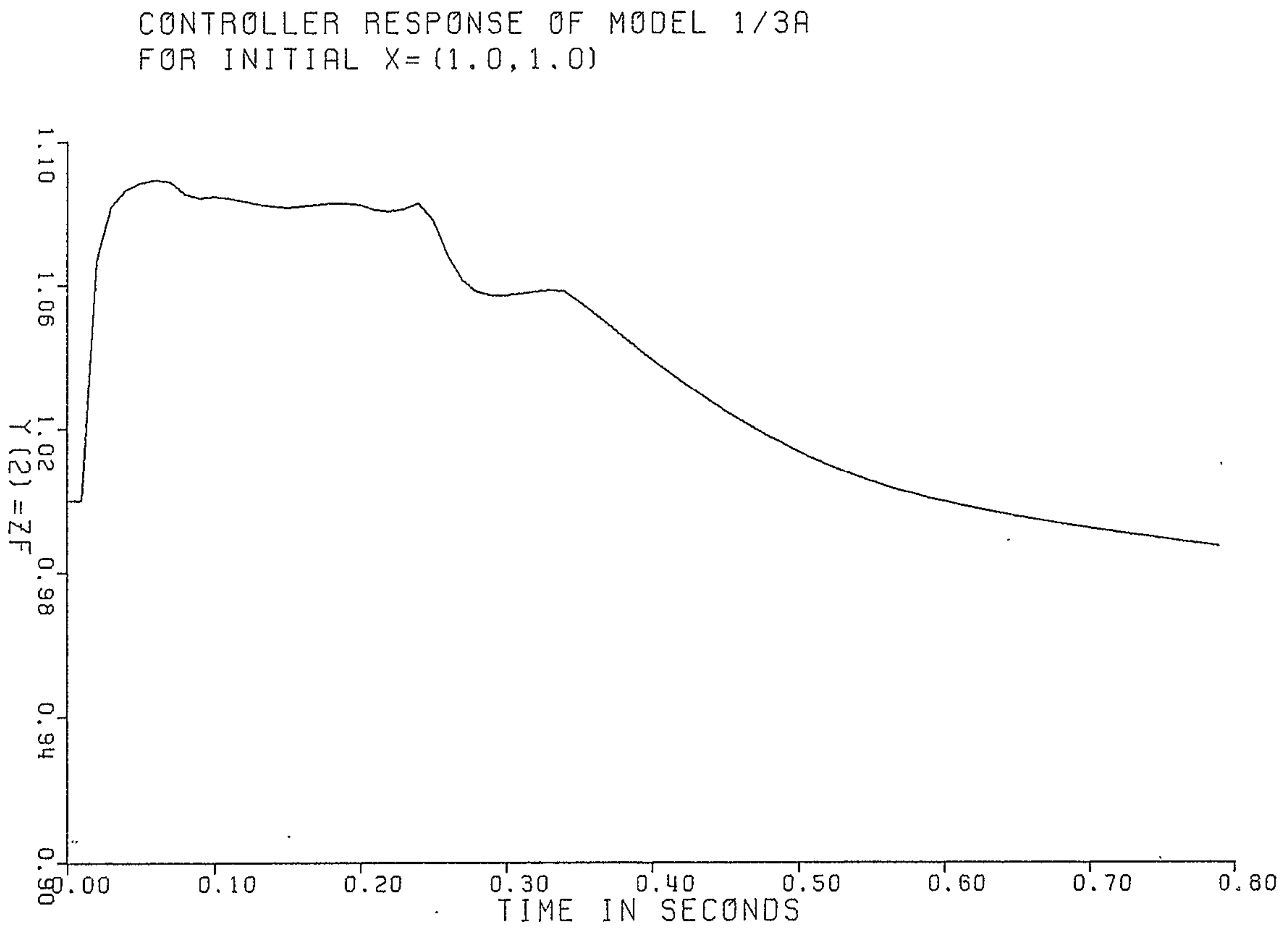

CONTROLLER RESPONSE OF MODEL 1/3A
FOR INITIAL X=(1.0,1.0)
Y(2)=ZF
1.10
1.06
1.02
0.98
0.94
0.90
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.5-9

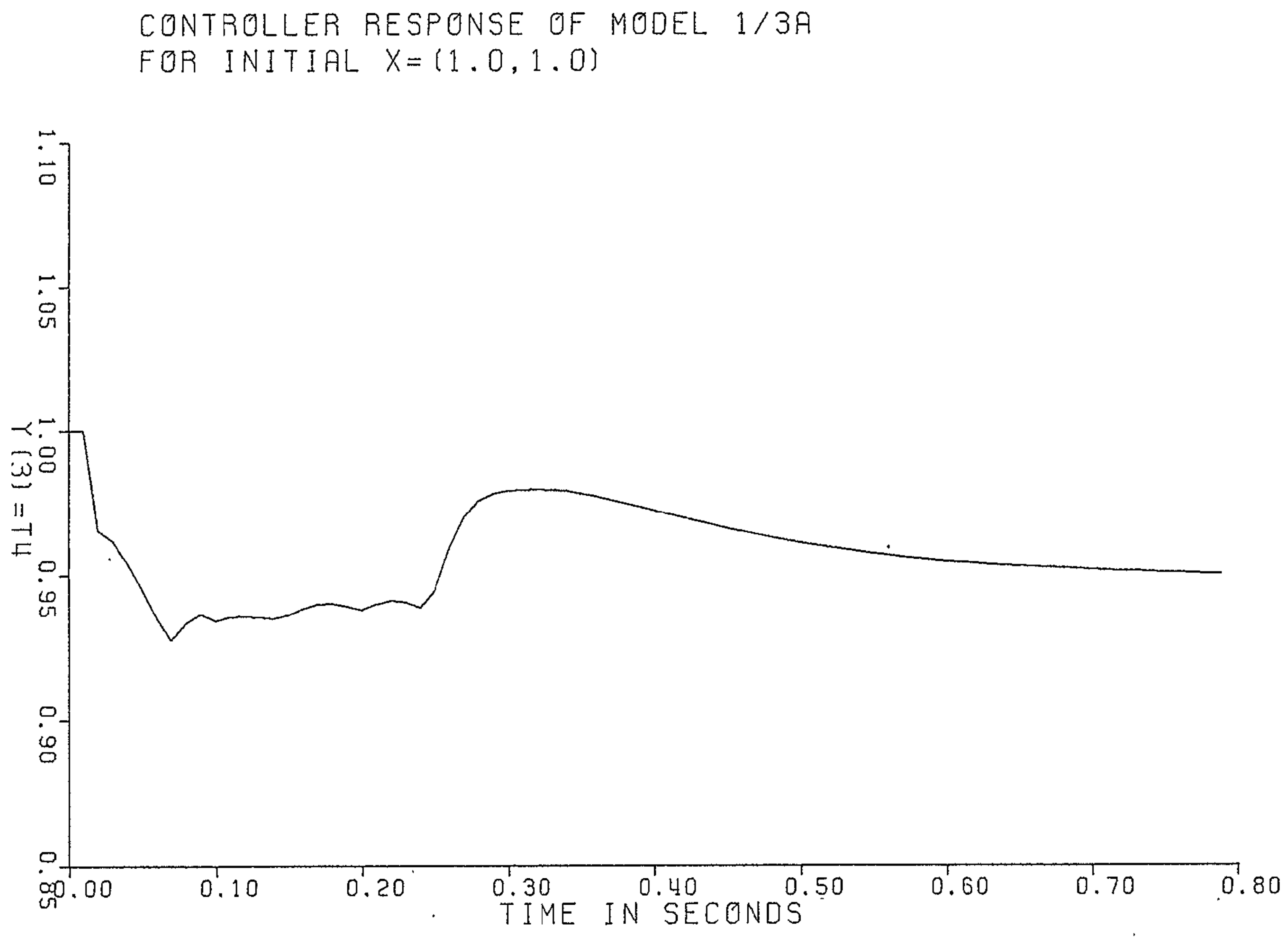

CONTROLLER RESPONSE OF MODEL 1/3A
FOR INITIAL X=(1.0,1.0)
Y(3)=T4
1.10
1.05
1.00
0.95
0.90
0.85
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.5-10

8.6 Model 1 with 3B-Controller

Again, DYNGEN is stable where the simpler model, 3B, is not. States x(1) and x(2) converge smoothly to the target, and output constraints are not violated. There seems to be little difference between the results from 3A and 3B, indicating that the power law term present in 3A is not particularly significant.

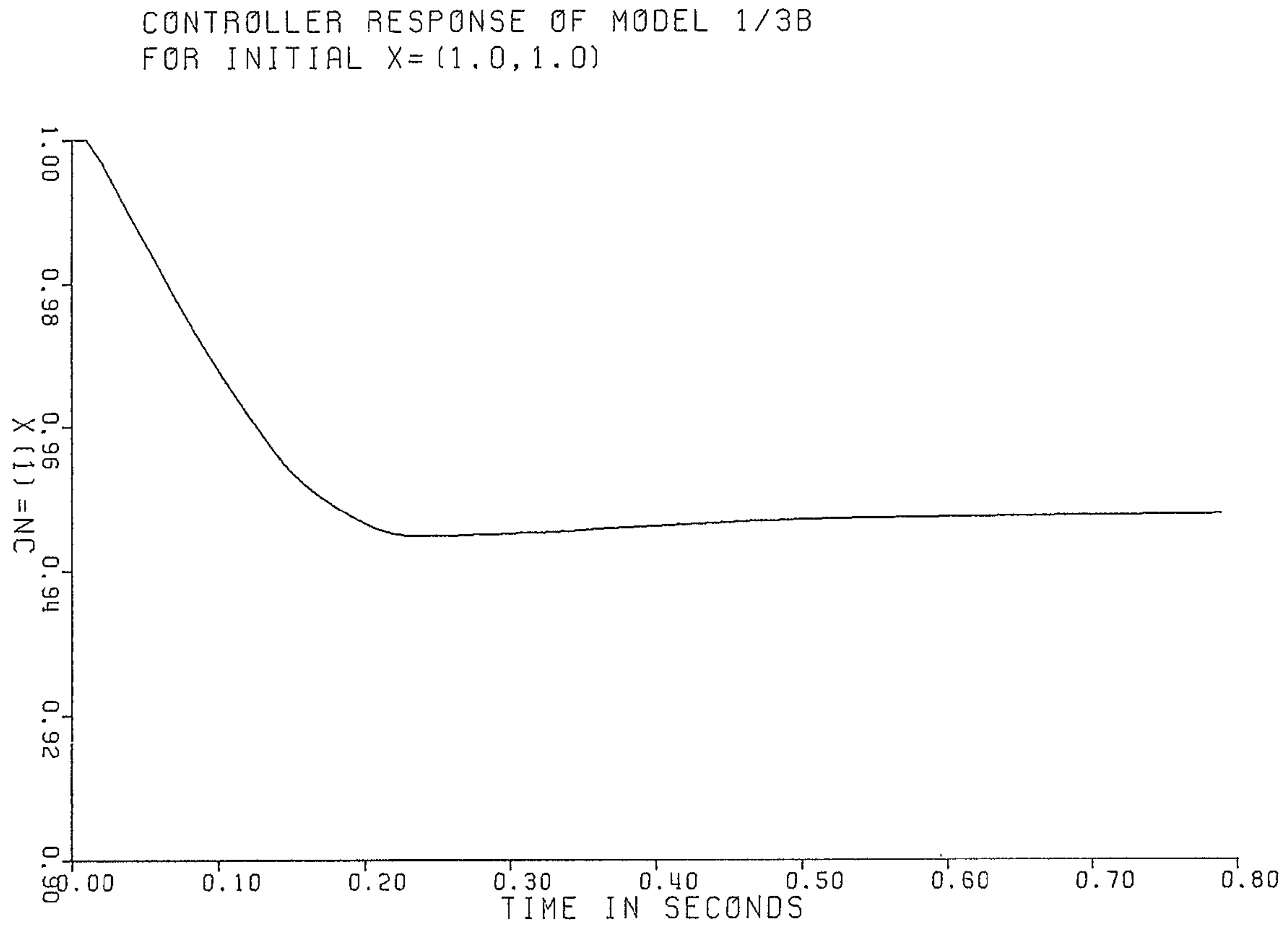

CONTROLLER RESPONSE OF MODEL 1/3B
FOR INITIAL X=(1.0,1.0)
1.00
0.98
0.96
0.94
0.92
0.90
X(1)=NC
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.6-1

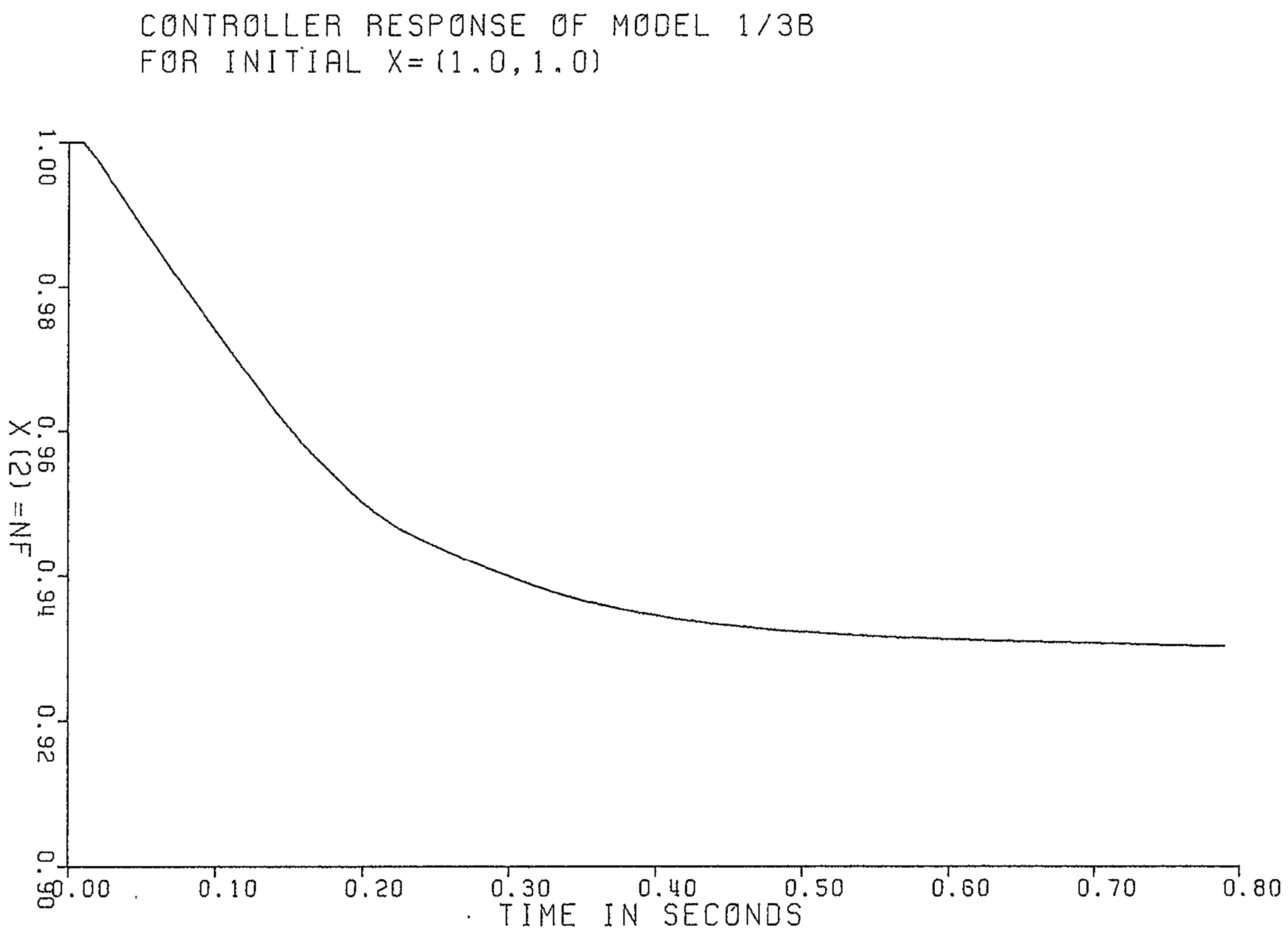

CONTROLLER RESPONSE OF MODEL 1/3B
FOR INITIAL X=(1.0,1.0)
X(2)=NF
1.00
0.98
0.96
0.94
0.92
0.90
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.6-2

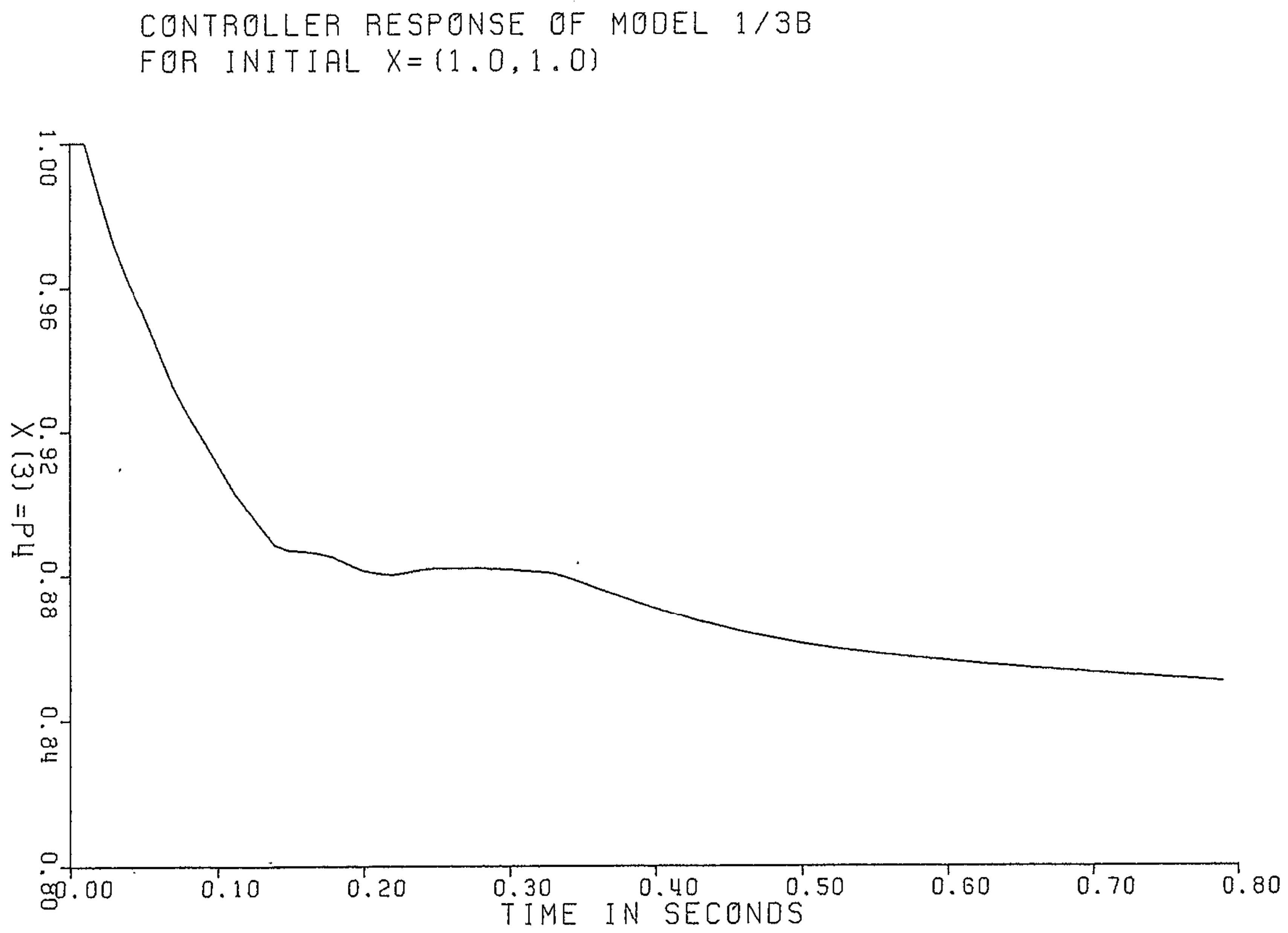

CONTROLLER RESPONSE OF MODEL 1/3B
FOR INITIAL X=(1.0,1.0)
X(3)=P4
1.00
0.96
0.92
0.88
0.84
0.80
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.6-3

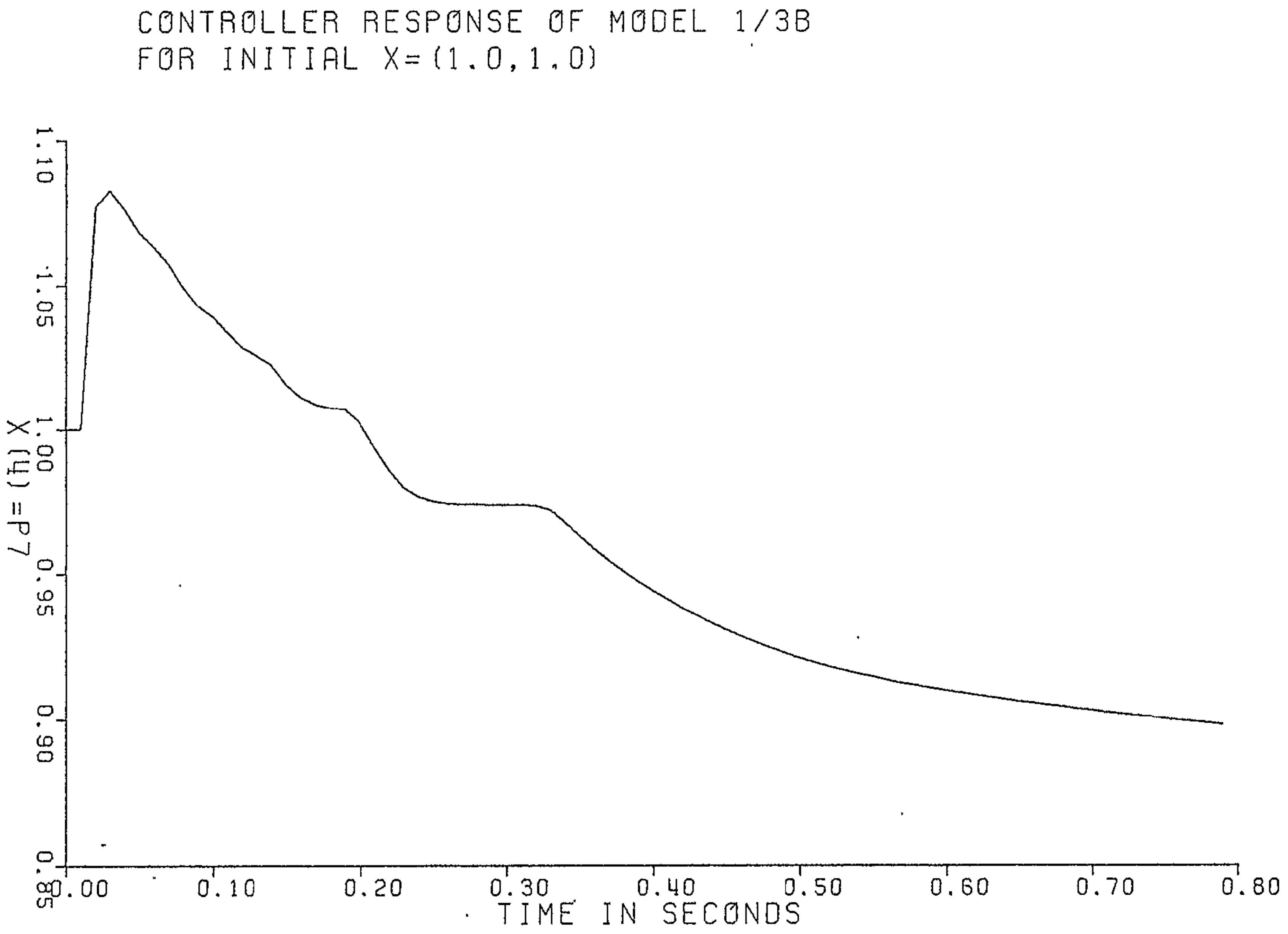

CONTROLLER RESPONSE OF MODEL 1/3B
FOR INITIAL X=(1.0,1.0)
X(4)=P7
1.10
1.05
1.00
0.95
0.90
0.85
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.6-4

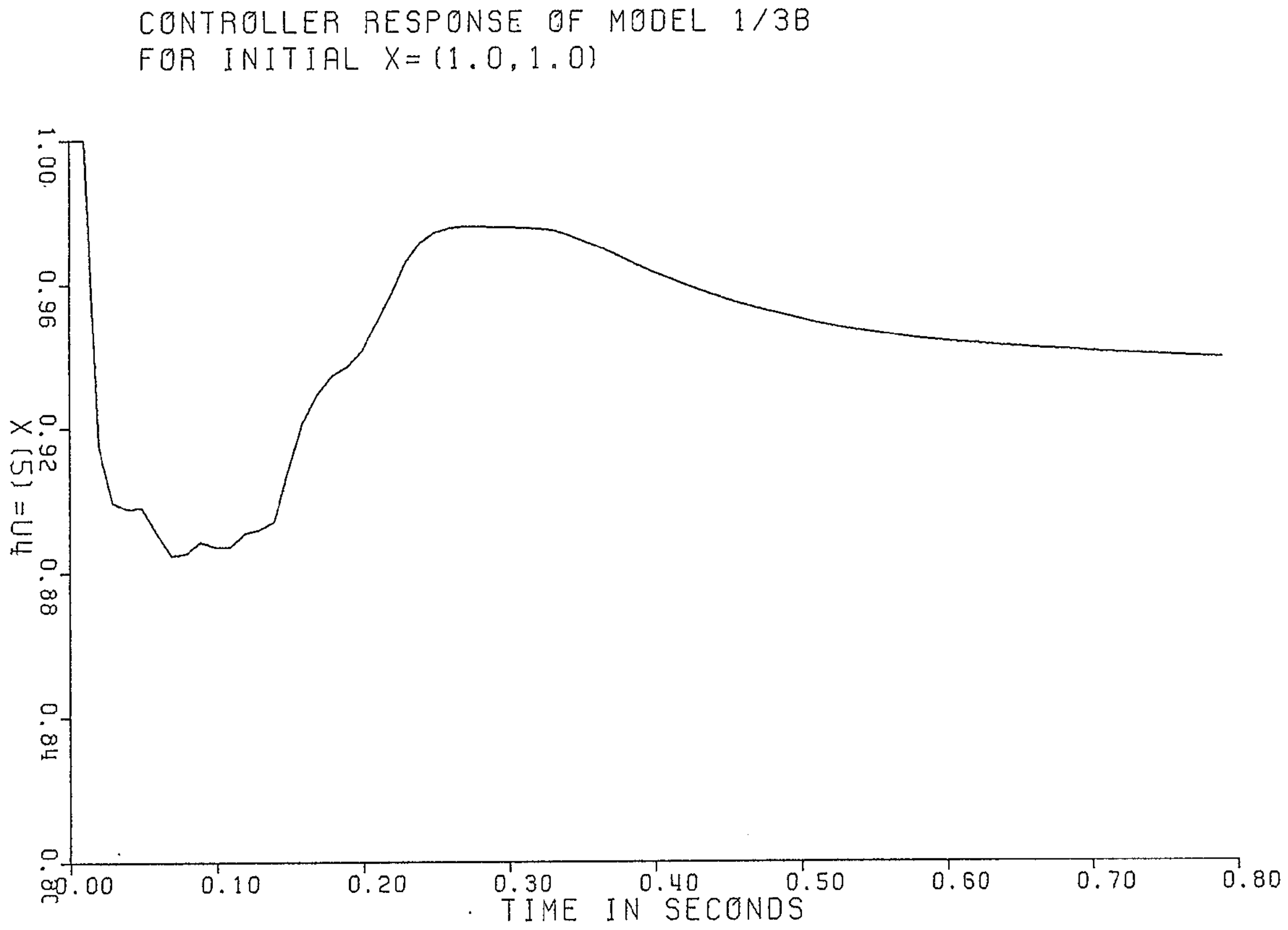

CONTROLLER RESPONSE OF MODEL 1/3B
FOR INITIAL X=(1.0,1.0)
X(5)=U4
1.00
0.96
0.92
0.88
0.84
0.80
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.6-5

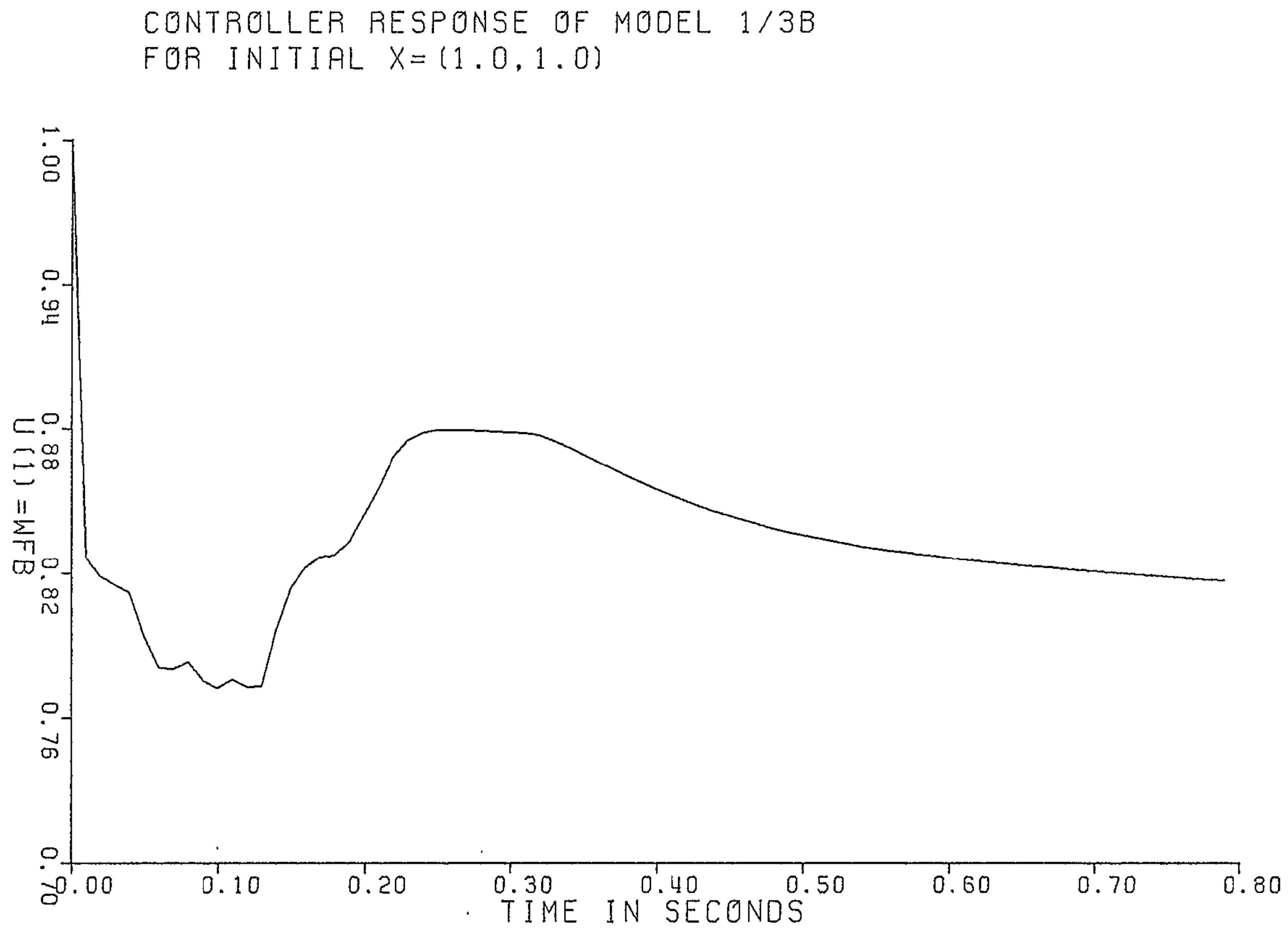

CONTROLLER RESPONSE OF MODEL 1/3B
FOR INITIAL X=(1.0,1.0)
U(1)=WFB
1.00
0.94
0.88
0.82
0.76
0.70
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.6.6

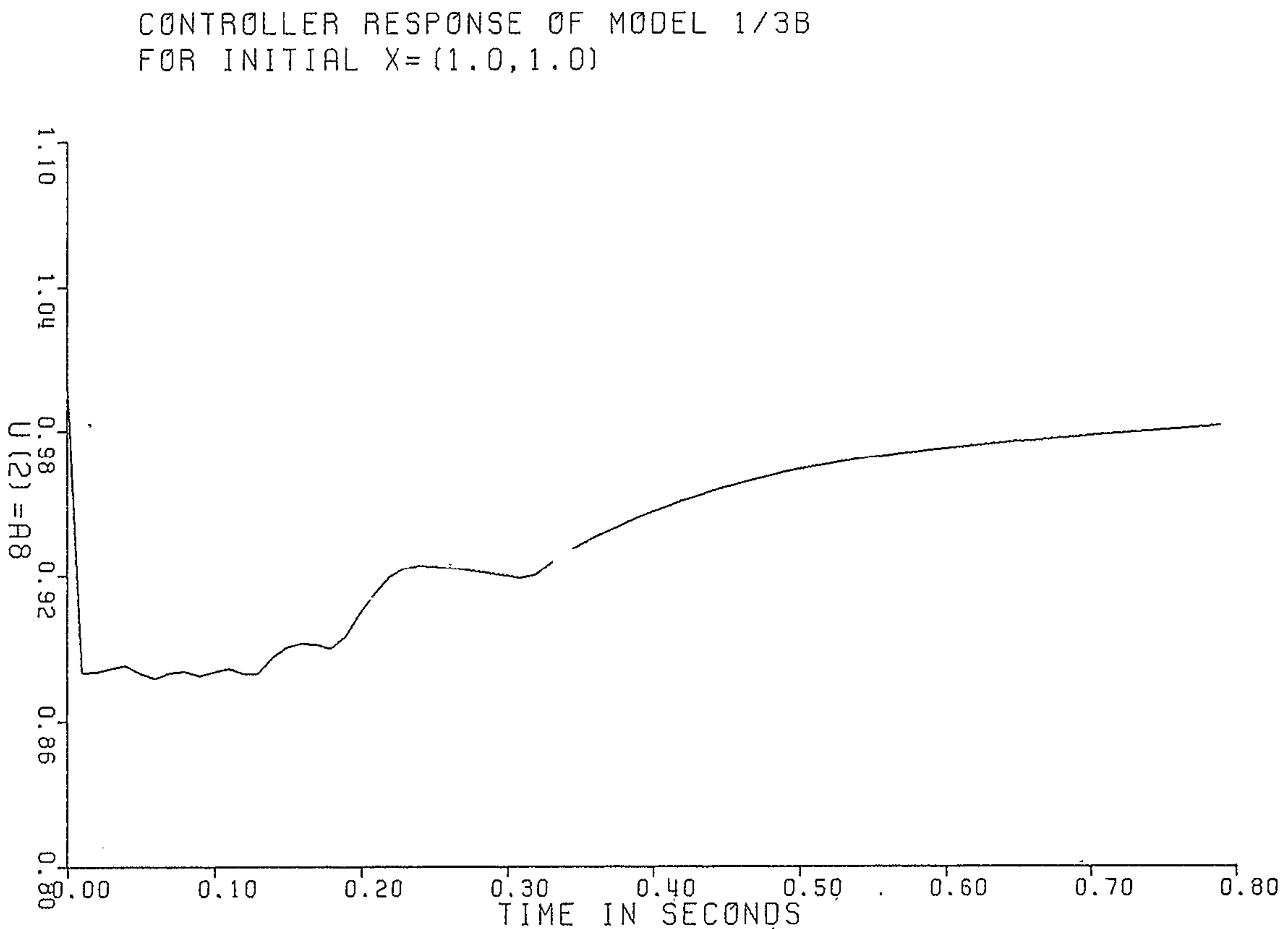

CONTROLLER RESPONSE OF MODEL 1/3B
FOR INITIAL X=(1.0,1.0)
1.10
1.04
0.98
0.92
0.86
0.80
U(2)=A8
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.6-7

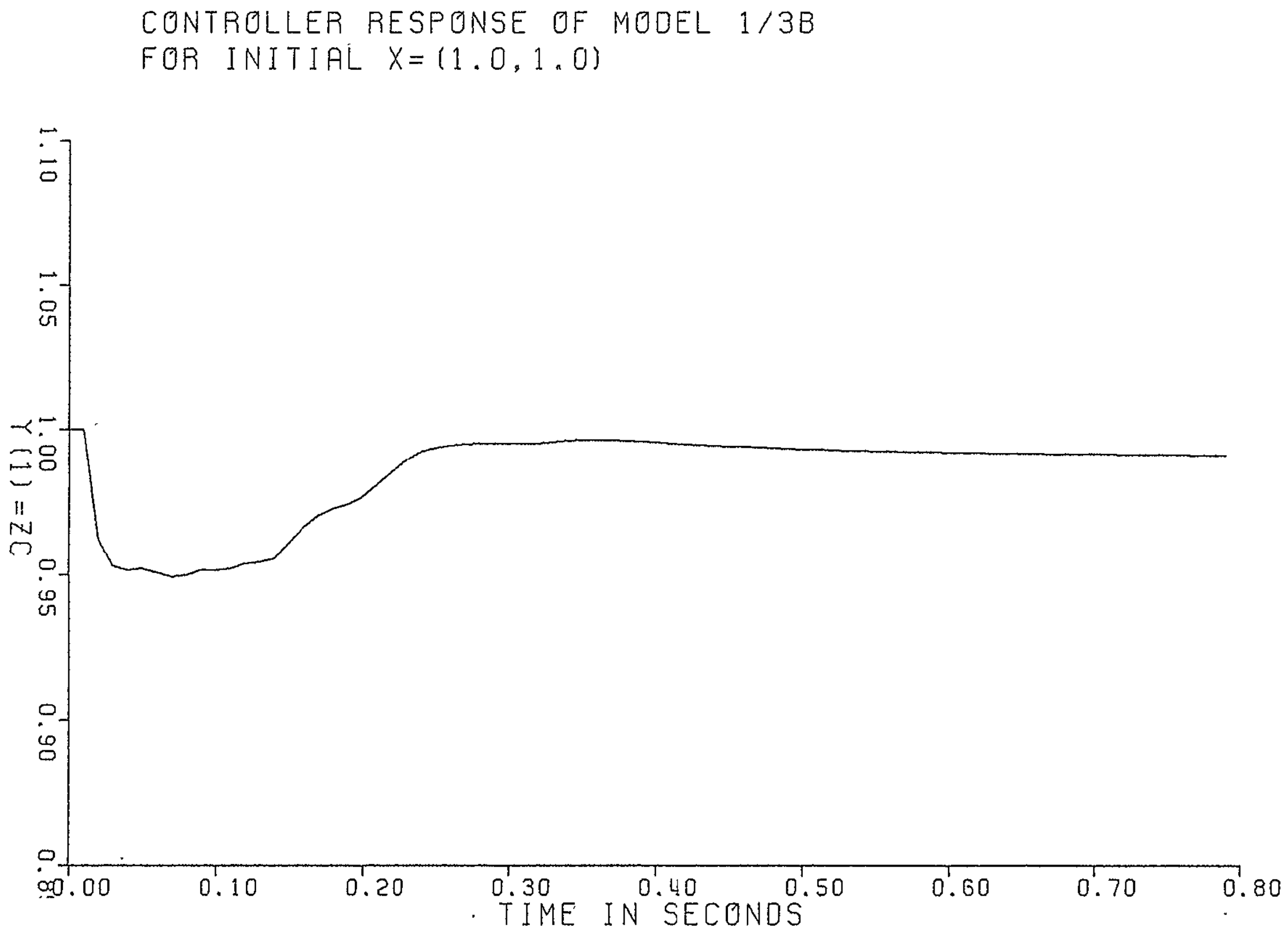

CONTROLLER RESPONSE OF MODEL 1/3B
FOR INITIAL X=(1.0,1.0)
Y(1)=ZC
1.10
1.05
1.00
0.95
0.90
0.85
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.6-8

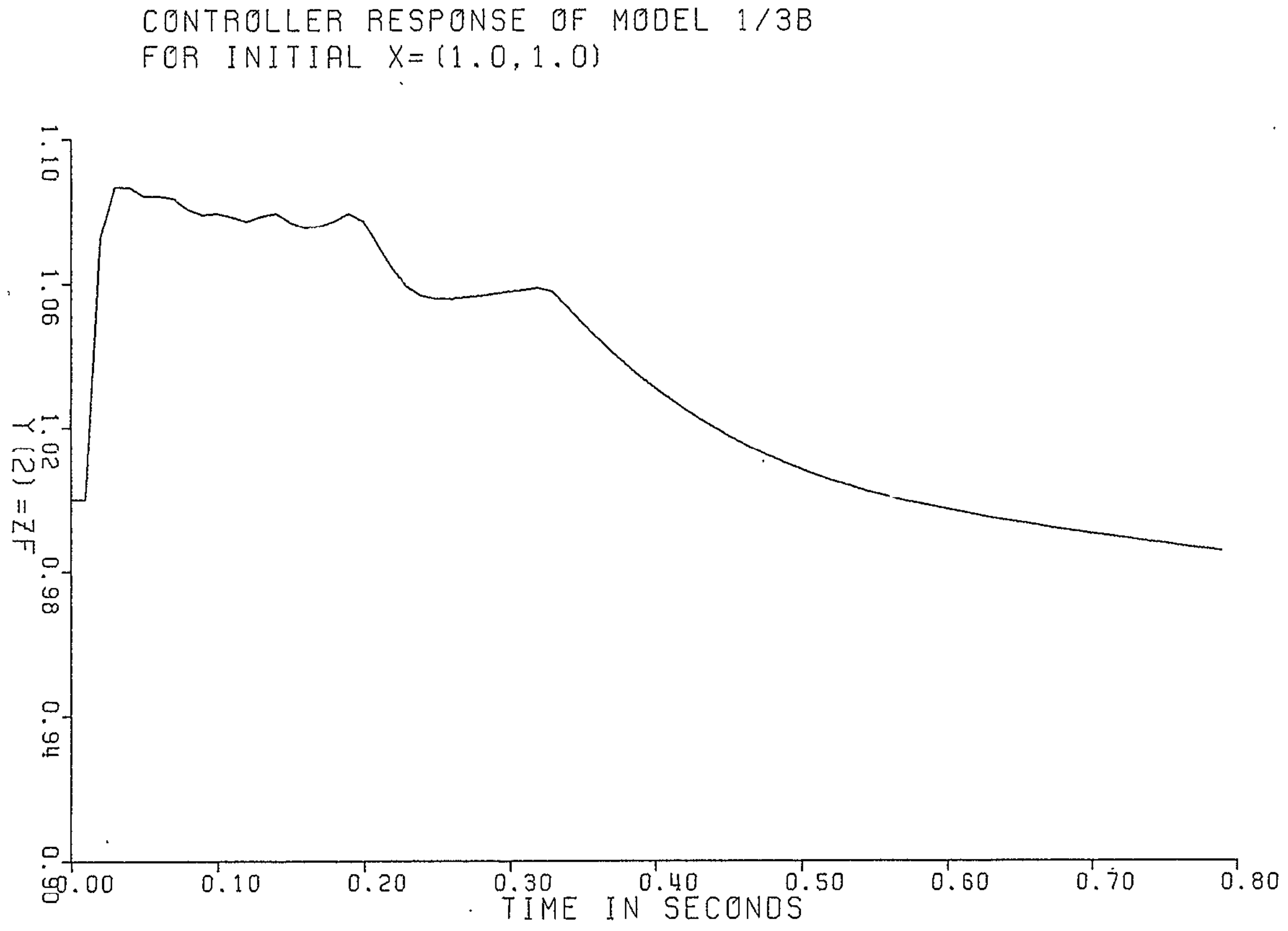

CONTROLLER RESPONSE OF MODEL 1/3B
FOR INITIAL X=(1.0,1.0)
1.10
1.06
1.02
0.98
0.94
0.90
Y(2)=ZF
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.6-9

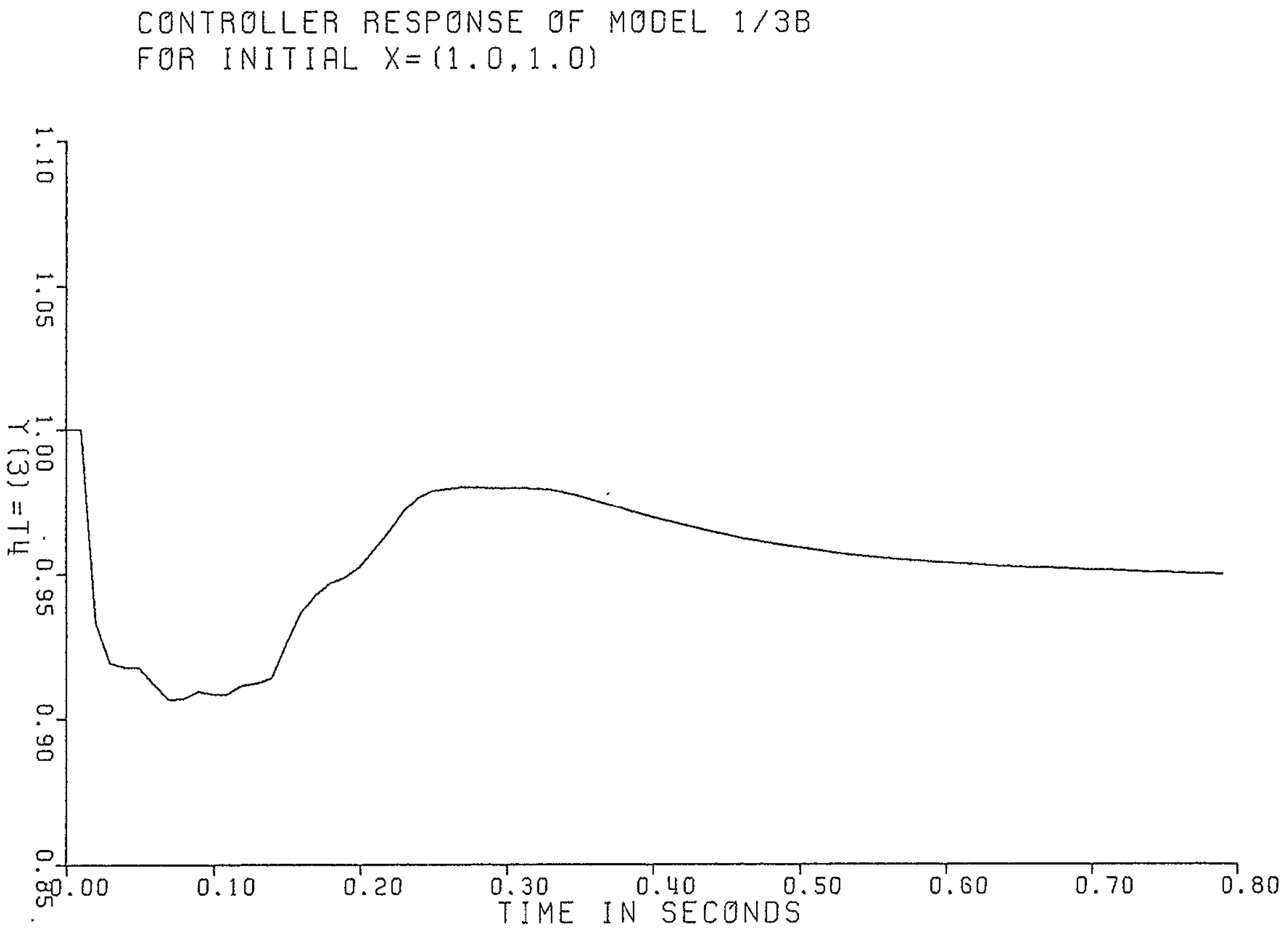

CONTROLLER RESPONSE OF MODEL 1/3B
FOR INITIAL X=(1.0,1.0)
Y(3)=T4
1.10
1.05
1.00
0.95
0.90
0.85
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.6-10

8.7 Model 1 with 4-Controller

The response of the states and controls for this system are nearly identical to those of model 4, the difference being that DYNGEN is about 0.05 seconds faster. The outputs, however, vary greatly from model 4, indicating that the linear affine approximation of the outputs may be inadequate. Despite this disagreement, the outputs do remain below the constraint values. As compared to models 3A and 3B, model 4 is certainly a step toward an automatically generated numerical model of a physical system.

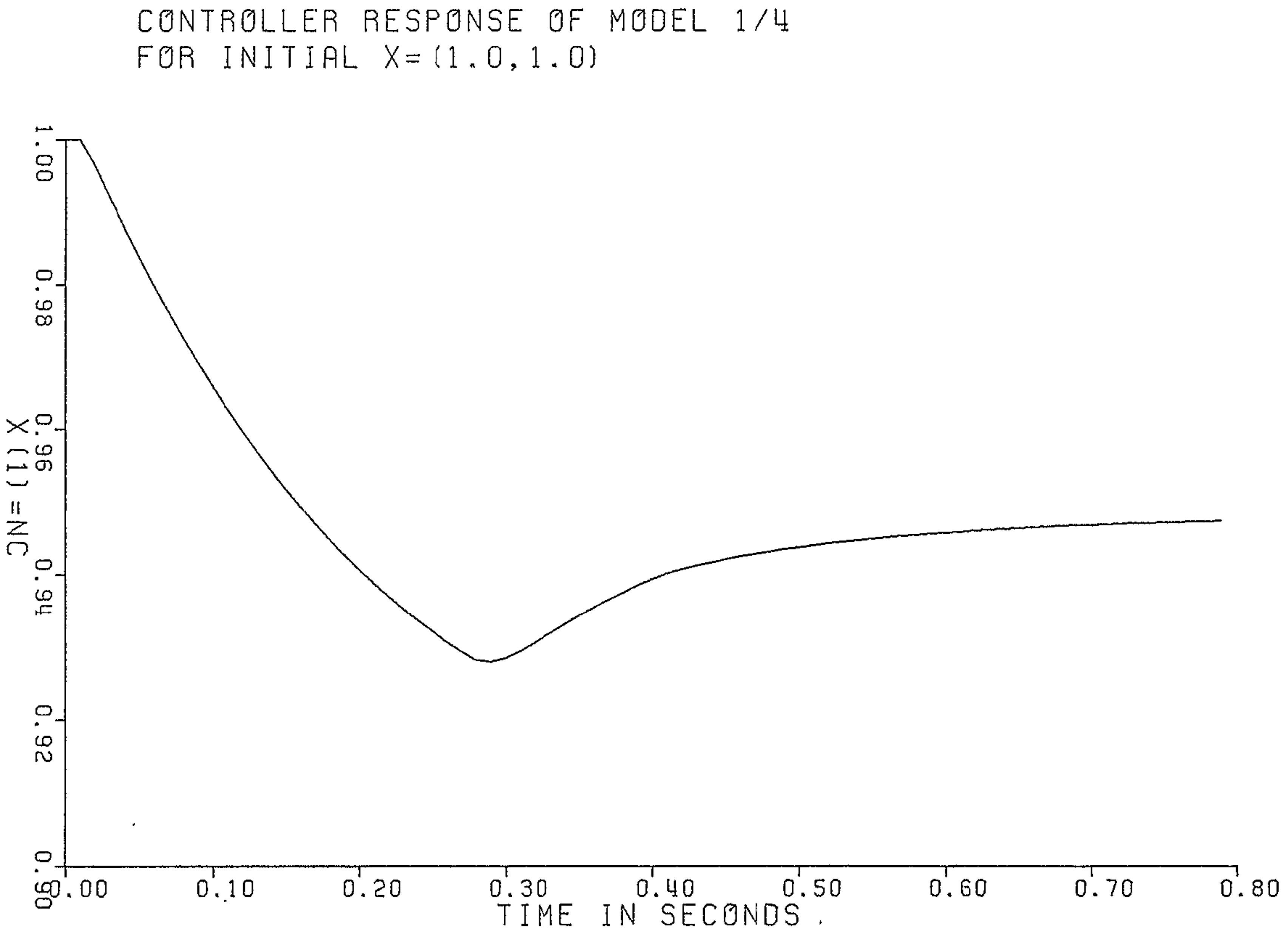

CONTROLLER RESPONSE OF MODEL 1/4
FOR INITIAL X=(1.0,1.0)
1.00
0.98
0.96
0.94
0.92
0.90
X(1)=NC
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.7-1

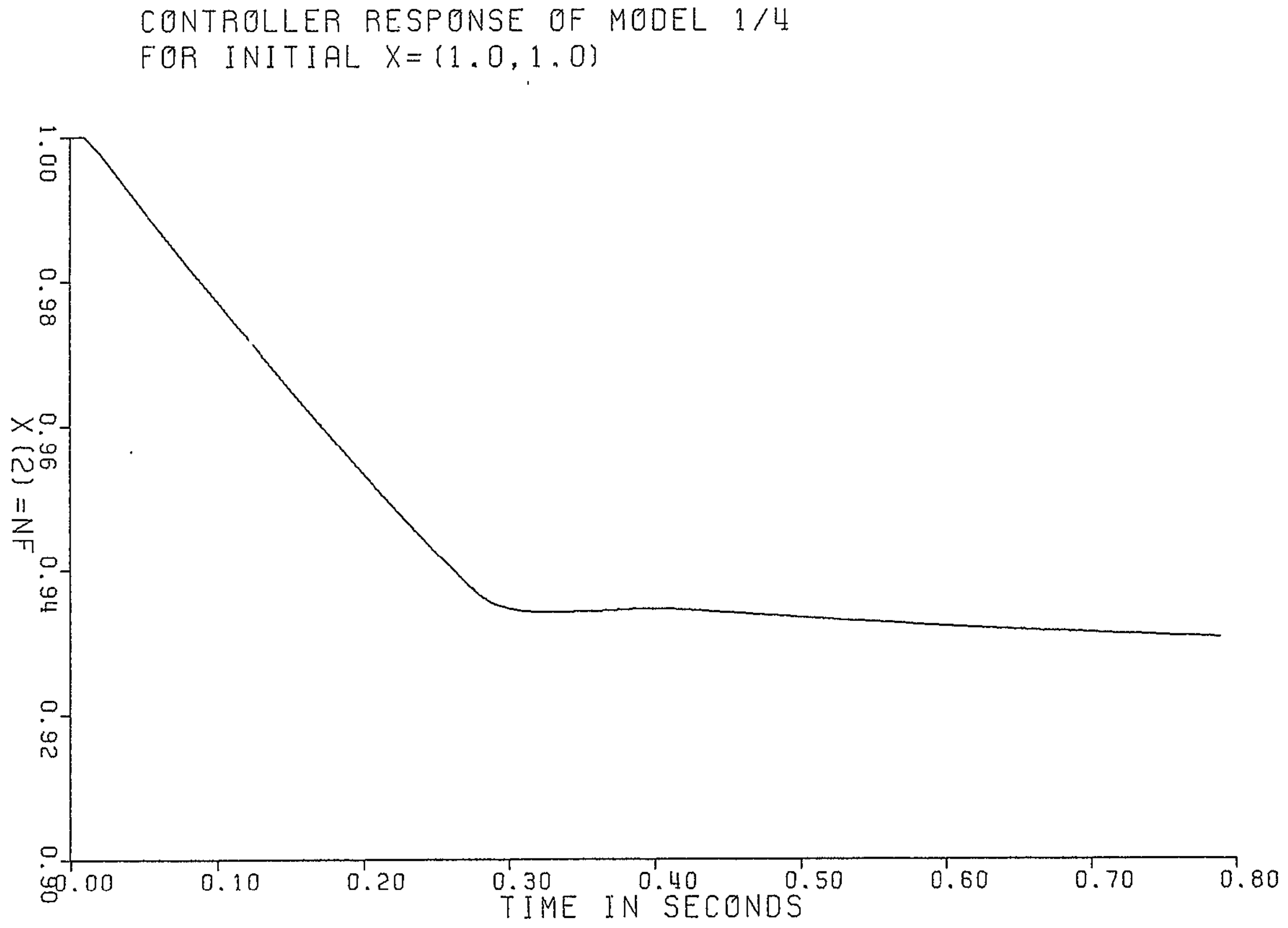

CONTROLLER RESPONSE OF MODEL 1/4
FOR INITIAL X=(1.0,1.0)
1.00
0.98
0.96
0.94
0.92
0.90
X(2)=NF
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.7-2

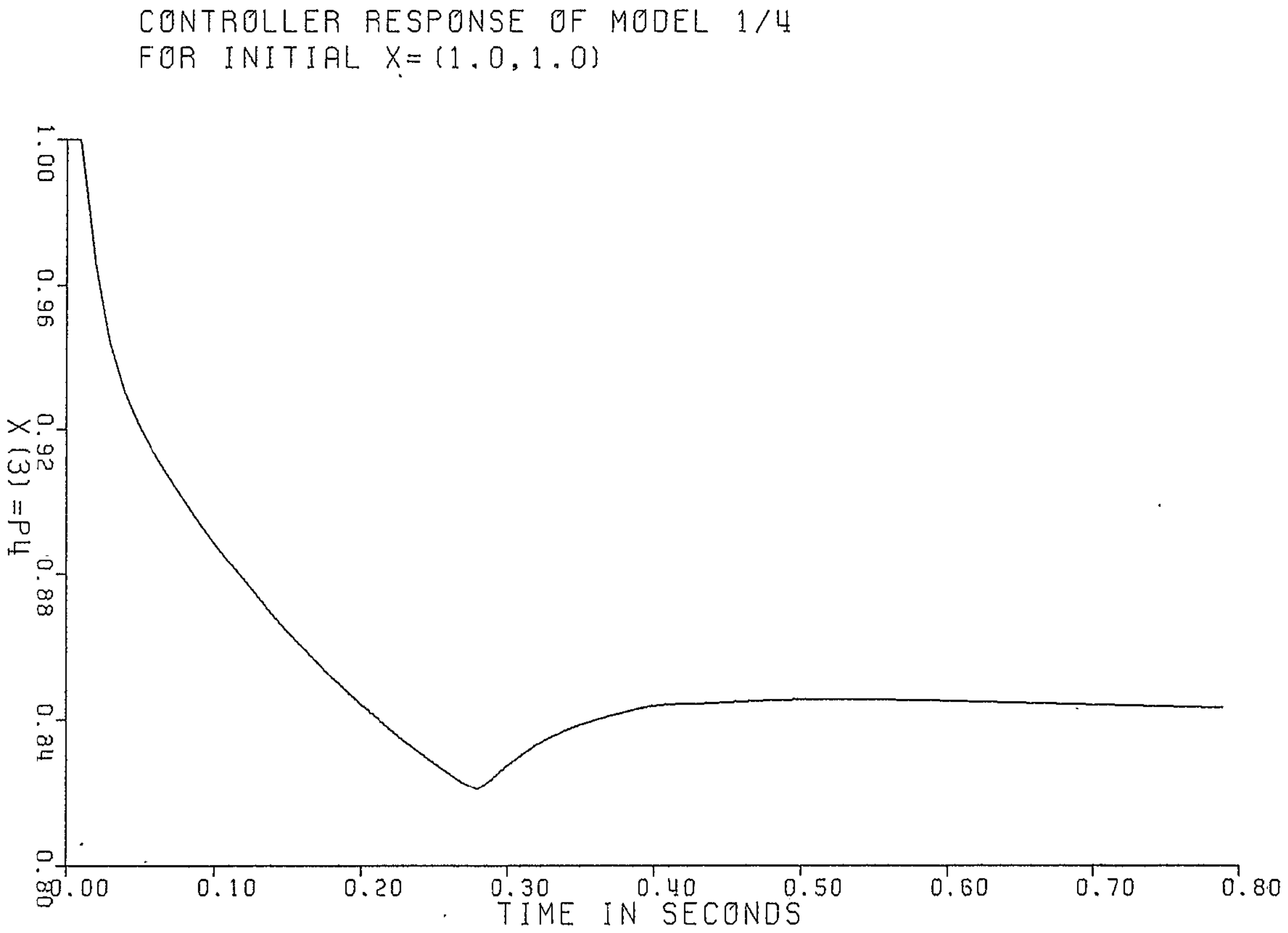

CONTROLLER RESPONSE OF MODEL 1/4
FOR INITIAL X=(1.0,1.0)
1.00
0.96
0.92
0.88
0.84
0.80
X(3)=P4
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.7-3

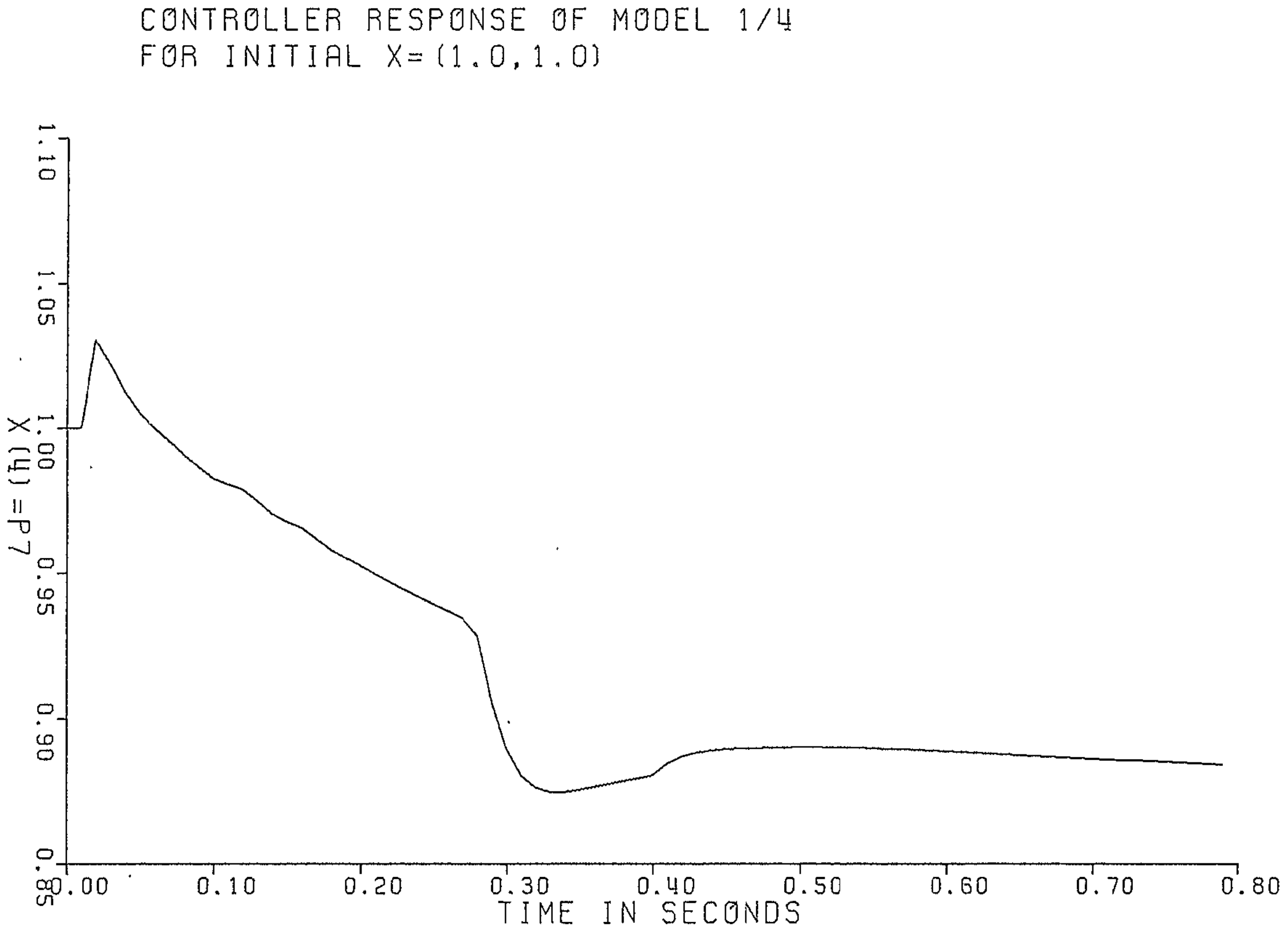
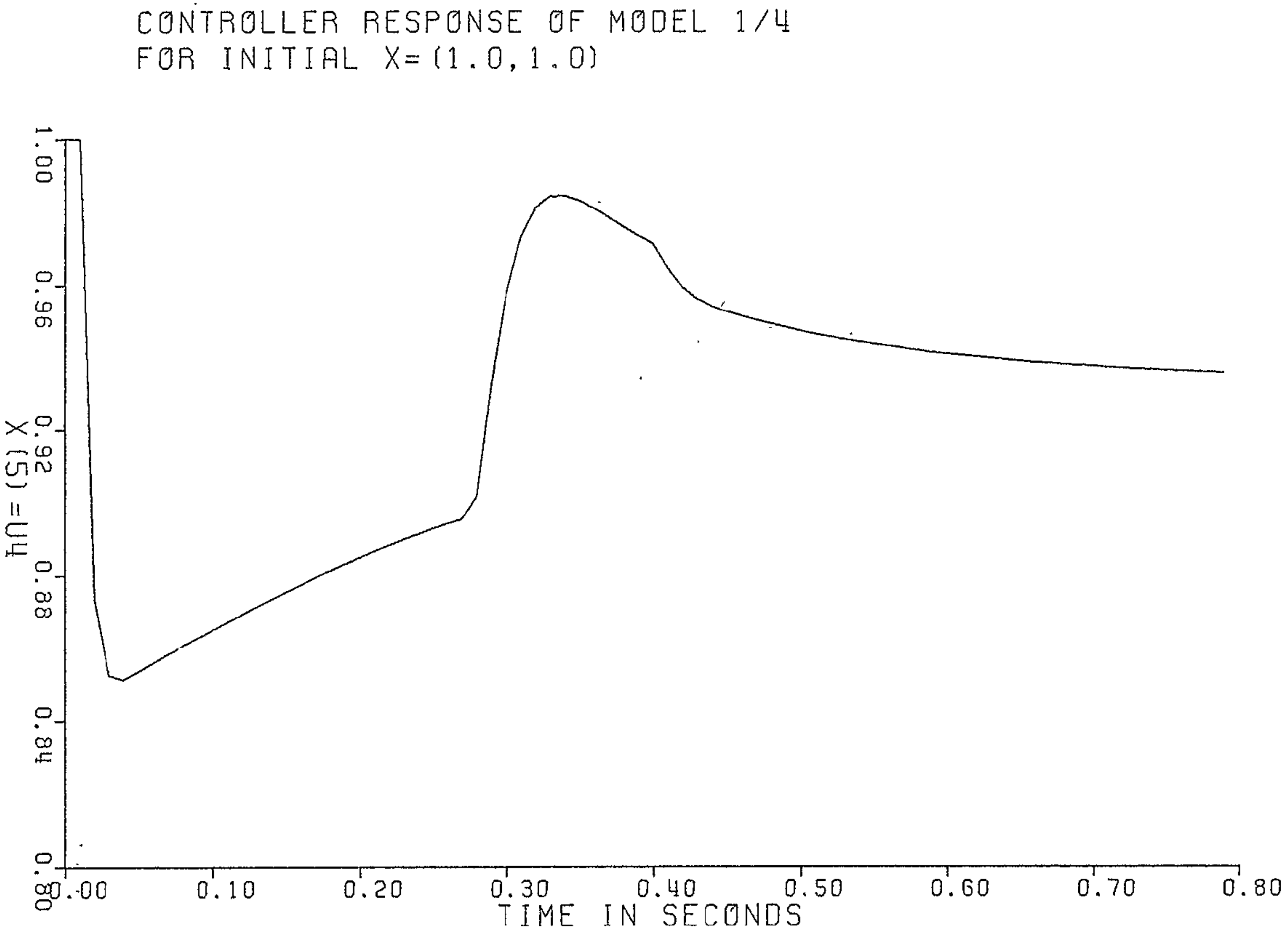
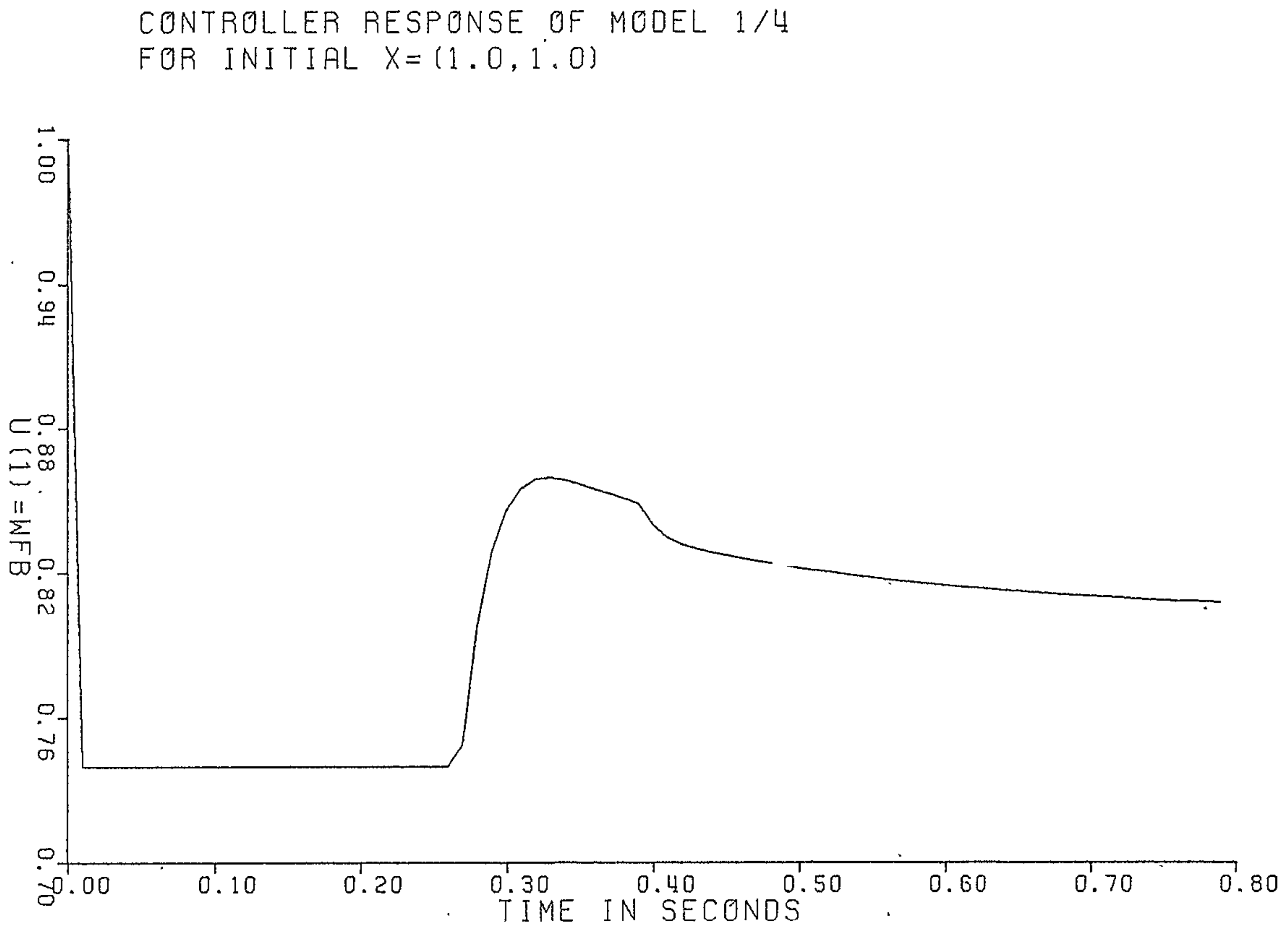

CONTROLLER RESPONSE OF MODEL 1/4
FOR INITIAL X=(1.0,1.0)
X(4)=P7
1.10
1.05
1.00
0.95
0.90
0.85
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.7-4



CONTROLLER RESPONSE OF MODEL 1/4
FOR INITIAL X=(1.0,1.0)
1.00
0.96
0.92
0.88
0.84
0.80
X(5)=U4
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.7-5



CONTROLLER RESPONSE OF MODEL 1/4
FOR INITIAL X=(1.0,1.0)
U(1)=WFB
1.00
0.94
0.88
0.82
0.76
0.70
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.7-6

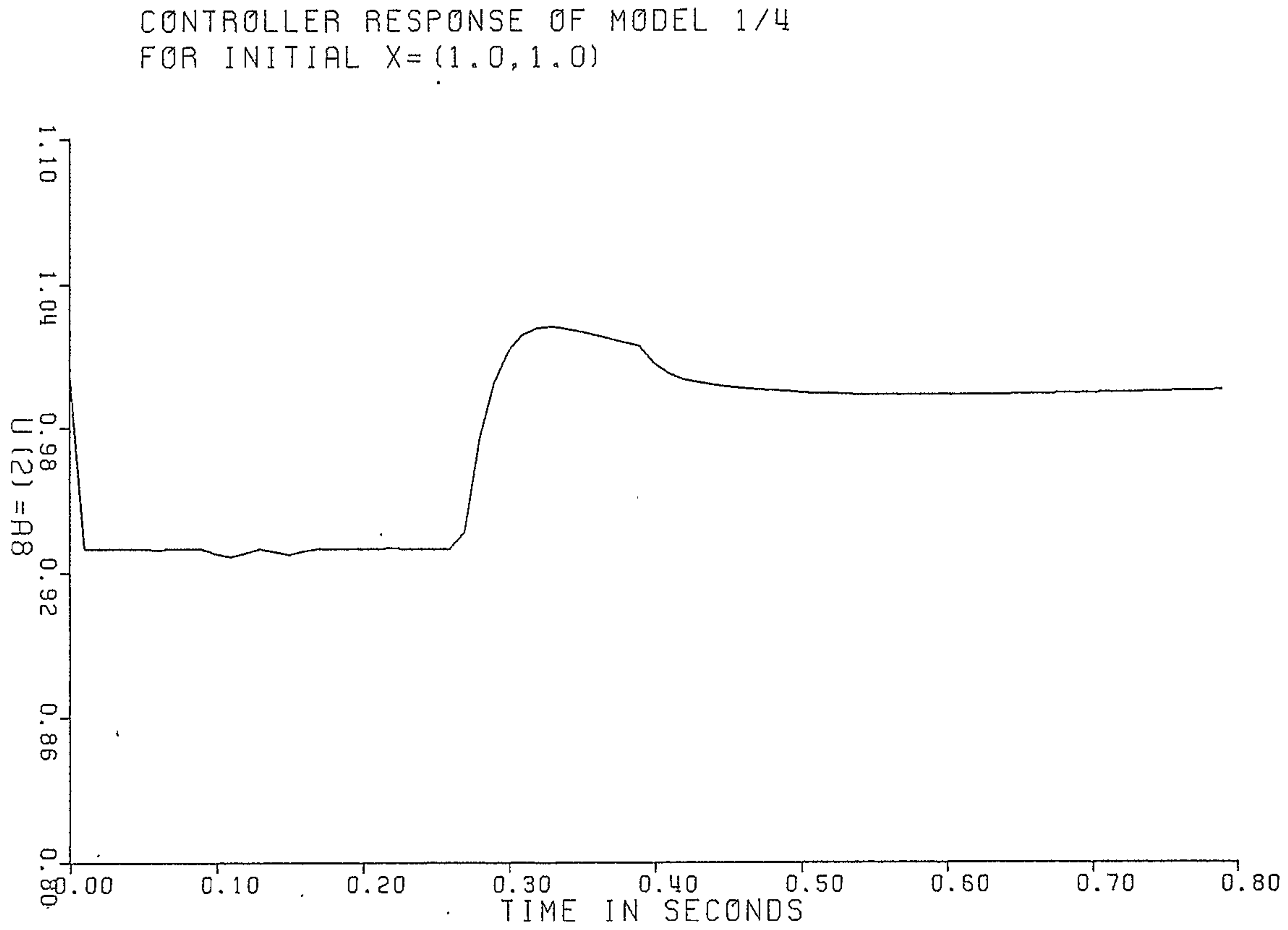

CONTROLLER RESPONSE OF MODEL 1/4
FOR INITIAL X=(1.0,1.0)
U(2)=A8
1.10
1.04
0.98
0.92
0.86
0.80
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.7-7

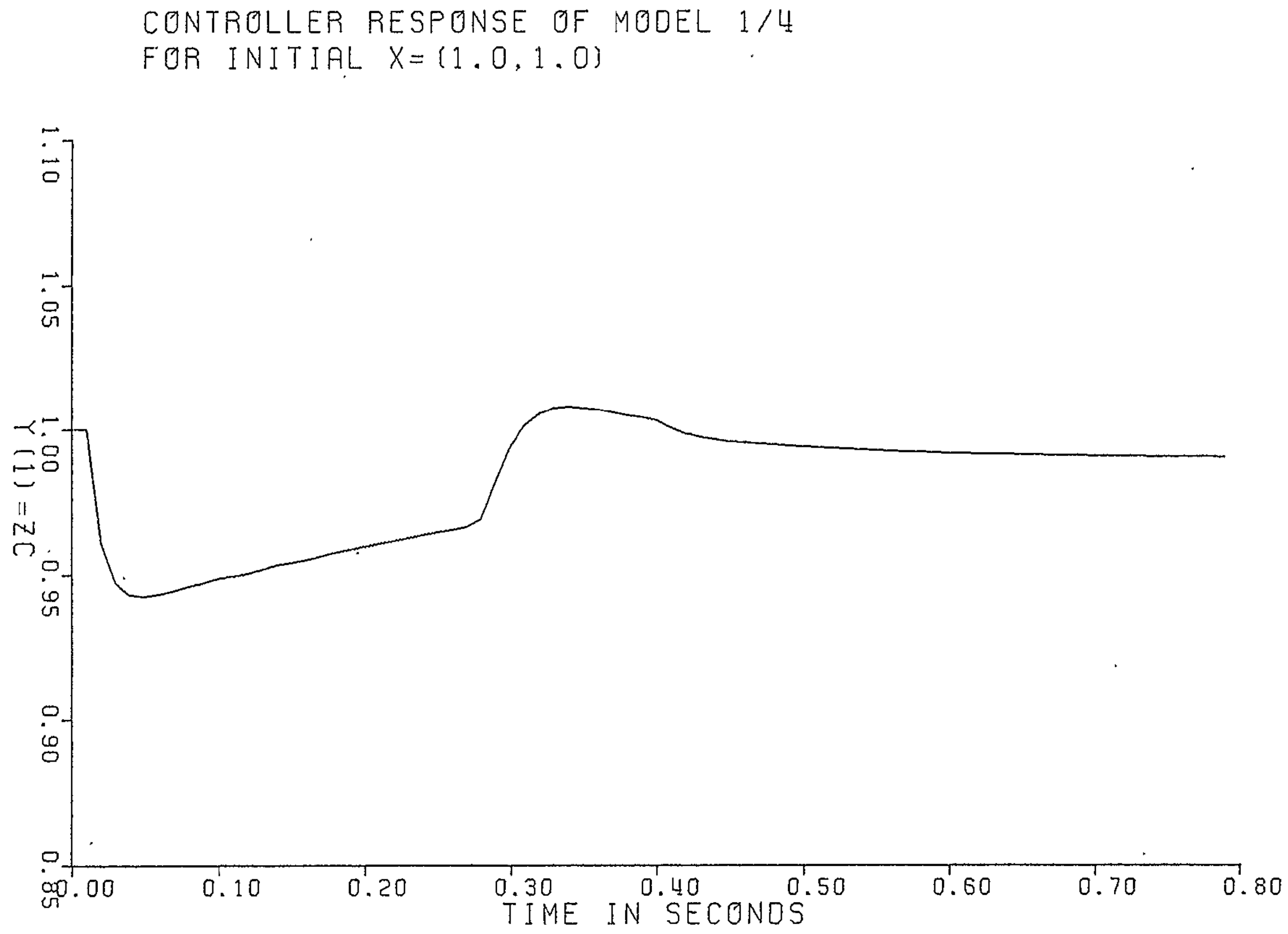

CONTROLLER RESPONSE OF MODEL 1/4
FOR INITIAL X=(1.0,1.0)
1.10
1.05
1.00
0.95
0.90
0.85
Y(1)=ZC
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.7-8

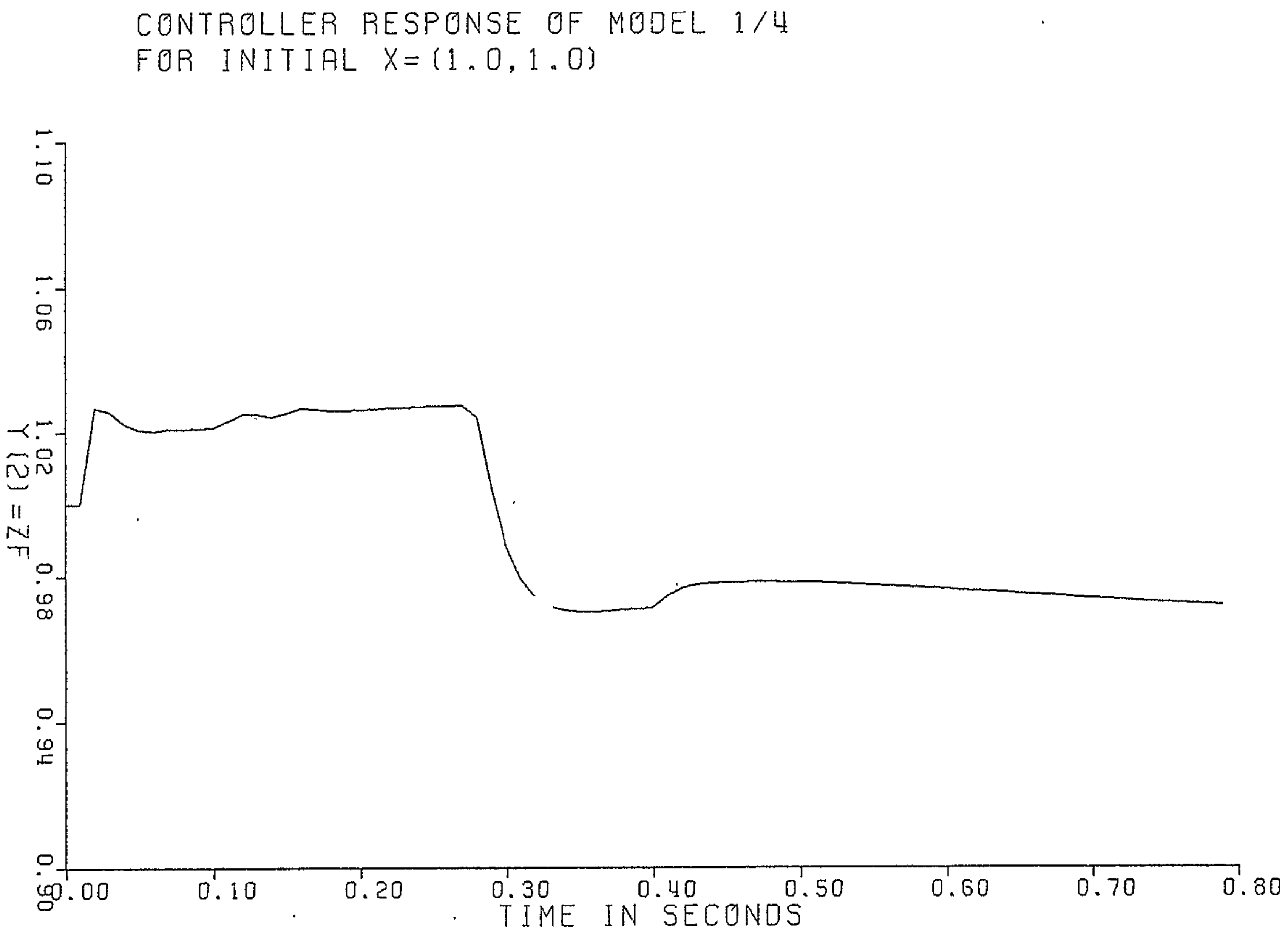

CONTROLLER RESPONSE OF MODEL 1/4
FOR INITIAL X=(1.0,1.0)
1.10
1.06
1.02
0.98
0.94
0.90
Y(2)=ZF
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.7-9

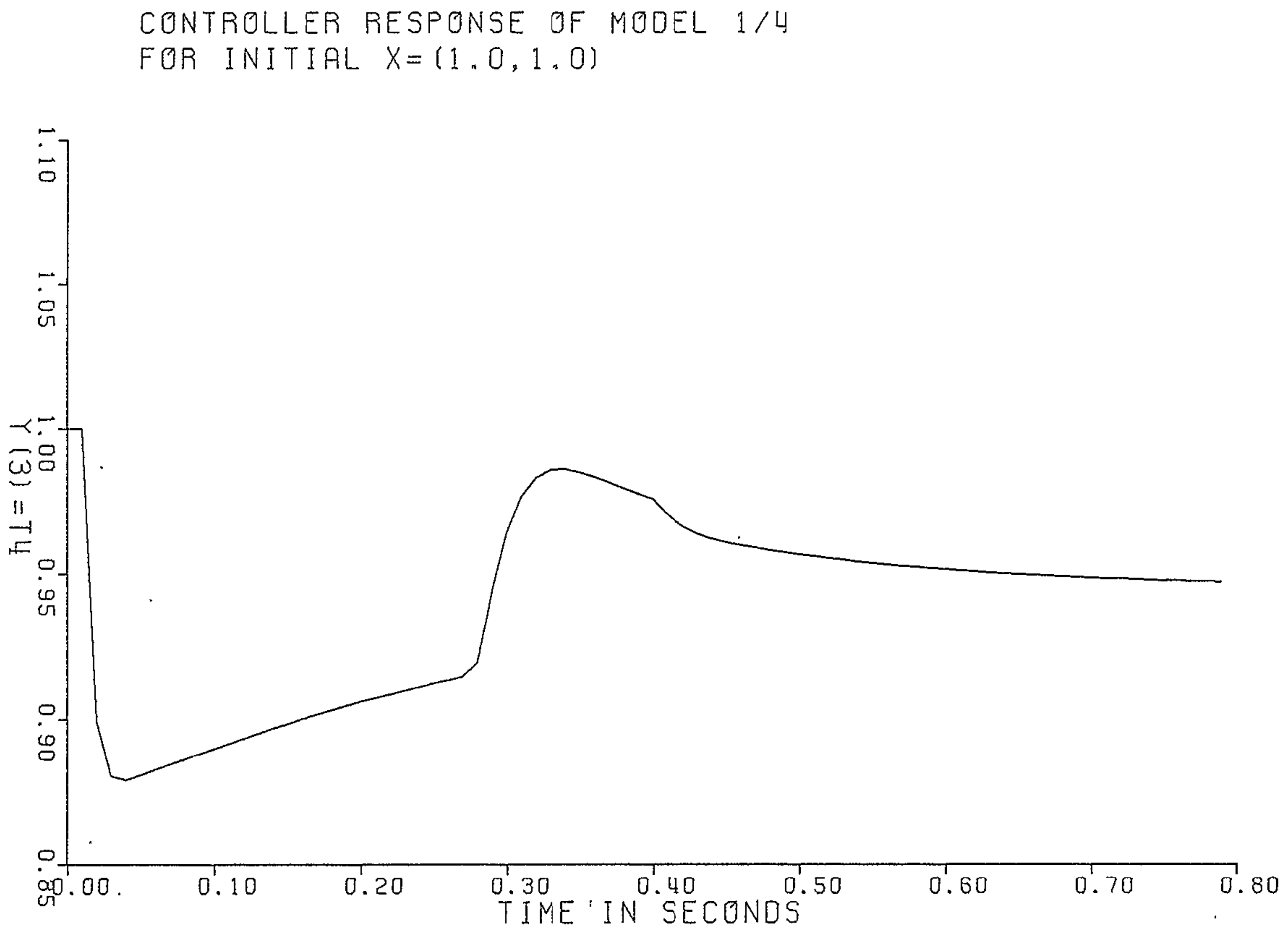

CONTROLLER RESPONSE OF MODEL 1/4
FOR INITIAL X=(1.0,1.0)
Y(3)=T4
1.10
1.05
1.00
0.95
0.90
0.85
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
TIME IN SECONDS

Figure 8.7-10

CHAPTER IX

SUMMARY AND CONCLUSIONS

A dissatisfactory discovery of this work is that the accuracy of numerical models deteriorates so rapidly outside the range of their data points, that the state and control space limits must be confined to that range. The numerical model designer must be sure to include steady state and dynamic data from the entire area of state and control space that he wishes to explore.

This rapid deterioration in accuracy is probably the cause of instability in models 3A and 3B. Having been derived from data at only two steady state points, these models are reliable only in the neighborhood of the line between these points. The transient generated by applying the control law carries the state outside that neighborhood, and accuracy deteriorates to the point of total model breakdown.

The accuracy and stability of the numerical models in this work were enhanced by the inclusion of more data, and a more sophisticated interpolation scheme. However, this more complex scheme resulted in extravagant expenditure of cpu time. The cost of cpu time on Notre Dame's IBM 370/168 is currently \$275.00 per hour, while core allocation time costs \$0.13 per K per hour. A typical run of 10 iterations of the dynamic programming algorithm with model 4 takes about 5 minutes of cpu time and 30 minutes of real time. A reduction of cpu time of 20% to 4 minutes would result in a savings of \$4.50. Assuming a similar reduction in real time to 24 minutes, this saving would justify an increase in core usage to 88K. A complete specification of 5 x 5 A matrices for 15 x 15 state space window, and 5 element g(u) function values for a 25 x 25 control space window would occupy only 35K. This would result in a model in which there was no need whatsoever for interpolation, and probably a cpu time savings of greater than 20%. Values for eliminated states could easily be included, eliminating another major source of model inaccuracy. Pursuit of larger data bases, rather than more complex interpolation techniques seems more promising, in light of the availability and price of these two computer resources.

The application of control laws to models involves interpolation between adjacent control values. In the continuous portions of a control law, the difference interpolated is seldom more than a single control space increment. At a discontinuity in the control law, as stated by Longenbaker [1] and demonstrated in this work, interpolation is actually undesirable. Since control interpolation is either insignificant or counterproductive, it should be abandoned in future work in favor of a closest point scheme.

A dynamic programming successive approximation algorithm was thoroughly developed, programmed, and tested as part of this work. Designed to interface easily with any model, it should be of value in future research.

In conclusion, the complex interpolation scheme developed for this work did perform well, but the cost in programming and cpu time indicates that a shift to more data and less interpolation would be more profitable.

APPENDIX A

INPUTS FOR DYNGEN SIMULATOR

This appendix includes the JCL used to run controller simulations on DYNGEN, the DISTRB subroutine that imposes the control law on DYNGEN, and the DYNGEN input data set, which includes the design point specification, three steady state requests, and a transient request. It is during the run of the transient that the control law is implemented. This JCL contains three separate jobs, one each for control law 3A, 3B, and 4.

```
//DYNGEN3A  JOB (CF,F081),711153015,TIME=10,REGION=320K,
// NOTIFY=F9G7LB
/*JOBPARM LINES=20
/*ROUTE  PRINT REMOTE2
/*ROUTE  PUNCH REMOTE2
/*SETUP    M0151,RAT
/*SETUP PLOTS,NOCODE
//STEP1 EXEC PURTINIT
//STEP2 EXEC PURTIN,TAPE=M0151
//MOVE.SYSPRINT DD DUMMY
//STEP3 EXEC PURTSOUR
//SOURCE.SYSPRINT DD DUMMY
//SOURCE.SYSIN DD DSN=F9G7LB.DYNGEN.FORT,DISP=SHR
//STEP4 EXEC PURTFORX,NAME=DISTRB
//FORT.SYSPRINT DD DUMMY
//LKED.SYSPRINT DD DUMMY
//STEP4A EXEC PGM=IEBGENER
//SYSIN DD DUMMY
//SYSPRINT DD DUMMY
//SYSUT2 DD DSN=&&CRDS,DISP=(NEW,PASS),UNIT=DISK,SPACE=(80,(10,10)),
//     DCB=(RECFM=FB,LRECL=80,BLKSIZE=80)
//SYSUT1 DD DSN=F9G7LB.DYNGEN.DATA,DISP=SHR
//STEP5 EXEC PGM=IEBGENER
//SYSIN DD DUMMY
//SYSPRINT DD DUMMY
//SYSUT1 DD DSN=&&CRDS,DISP=(OLD,PASS),UNIT=DISK
//SYSUT2 DD SYSOUT=A,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//STEP6 EXEC PURTFGO,NAME=DYNGEN1
//LKED.SYSPRINT DD DUMMY
//GO.FT07F001 DD DSN=&&PCH,UNIT=DISK,SPACE=(TRK,(2,1)),DISP=(,PASS)
//GO.FT08F001 DD DISP=(NEW,PASS),SPACE=(TRK,(10,10)),UNIT=DISK
//GO.FT09F001 DD *
1/3A
/*
//GO.FT10F001 DD DSN=F9G7LB.DYN3A.DATA,DISP=SHR
//GO.FT11F001 DD DSN=&&TRES,DISP=(,PASS),
// UNIT=DISK,SPACE=(TRK,(10,2))
//GO.SYSIN DD DSN=&&CRDS,DISP=(OLD,PASS),UNIT=DISK
//TPLOT EXEC FORTHP
//FORT.SYSPRINT DD DUMMY
//FORT.SYSIN DD DSN=F9G7LB.TPLOT.FORT,DISP=SHR
//LKED.SYSPRINT DD DUMMY
//GO.SYSIN DD *
0.0,0.8
0.9,1.0
0.9,1.0
0.8,1.0
0.85,1.1
0.8,1.0
0.7,1.0
0.8,1.1
0.85,1.1
0.9,1.1
0.85,1.1
/*
//GO.FT11F001 DD DSN=&&TRES,DISP=(OLD,DELETE)
//
//DYNGEN3B  JOB (CF,F081),711153015,TIME=10,REGION=320K,
// NOTIFY=F9G7LB
```

```
/*JOBPARM LINES=20
/*ROUTE  PRINT REMOTE2
/*ROUTE  PUNCH REMOTE2
/*SETUP     M0151,RAT
/*SETUP PLOTS,NOCODE
//STEP1 EXEC PURTINIT
//STEP2 EXEC PURTIN,TAPE=M0151
//MOVE.SYSPRINT DD DUMMY
//STEP3 EXEC PURTSOUR
//SOURCE.SYSPRINT DD DUMMY
//SOURCE.SYSIN DD DSN=F9G7LB.DYNGEN.FORT,DISP=SHR
//STEP4 EXEC PURTFORX,NAME=DISTRB
//FORT.SYSPRINT DD DUMMY
//LKED.SYSPRINT DD DUMMY
//STEP4A EXEC PGM=IEBGENER
//SYSIN DD DUMMY
//SYSPRINT DD DUMMY
//SYSUT2 DD DSN=&&CRDS,DISP=(NEW,PASS),UNIT=DISK,SPACE=(80,(10,10)),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=80)
//SYSUT1 DD DSN=F9G7LB.DYNGEN.DATA,DISP=SHR
//STEP5 EXEC PGM=IEBGENER
//SYSIN DD DUMMY
//SYSPRINT DD DUMMY
//SYSUT1 DD DSN=&&CRDS,DISP=(OLD,PASS),UNIT=DISK
//SYSUT2 DD SYSOUT=A,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//STEP6 EXEC PURTFGO,NAME=DYNGEN1
//LKED.SYSPRINT DD DUMMY
//GO.FT07F001 DD DSN=&&PCH,UNIT=DISK,SPACE=(TRK,(2,1)),DISP=(,PASS)
//GO.FT08F001 DD DISP=(NEW,PASS),SPACE=(TRK,(10,10)),UNIT=DISK
//GO.FT09F001 DD *
1/3B
/*
//GO.FT10F001 DD DSN=F9G7LB.DYN3B.DATA,DISP=SHR
//GO.FT11F001 DD DSN=&&TRES,DISP=(,PASS),
// UNIT=DISK,SPACE=(TRK,(10,2))
//GO.SYSIN DD DSN=&&CRDS,DISP=(OLD,PASS),UNIT=DISK
//TPLOT EXEC FORTHP
//FORT.SYSPRINT DD DUMMY
//FORT.SYSIN DD DSN=F9G7LB.TPLOT.FORT,DISP=SHR
//LKED.SYSPRINT DD DUMMY
//GO.SYSIN DD *
0.0,0.8
0.9,1.0
0.9,1.0
0.8,1.0
0.85,1.1
0.8,1.0
0.7,1.0
0.8,1.1
0.85,1.1
0.9,1.1
0.85,1.1
/*
//GO.FT11F001 DD DSN=&&TRES,DISP=(OLD,DELETE)
//
//DYNGEN4    JOB (CF,F081),711153015,TIME=10,REGION=320K,
// NOTIFY=F9G7LB
/*JOBPARM LINES=20
/*ROUTE  PRINT REMOTE2
```

```
/*ROUTE  PUNCH REMOTE2
/*SETUP     M0151,RAT
/*SETUP PLOTS,NOCODE
//STEP1 EXEC PURTINIT
//STEP2 EXEC PURTIN,TAPE=M0151
//MOVE.SYSPRINT DD DUMMY
//STEP3 EXEC PURTSOUR
//SOURCE.SYSPRINT DD DUMMY
//SOURCE.SYSIN DD DSN=F9G7LB.DYNGEN.FORT,DISP=SHR
//STEP4 EXEC PURTFORX,NAME=DISTRB
//FORT.SYSPRINT DD DUMMY
//LKED.SYSPRINT DD DUMMY
//STEP4A EXEC PGM=IEBGENER
//SYSIN DD DUMMY
//SYSPRINT DD DUMMY
//SYSUT2 DD DSN=&&CRDS,DISP=(NEW,PASS),UNIT=DISK,SPACE=(80,(10,10)),
//     DCB=(RECFM=FB,LRECL=80,BLKSIZE=80)
//SYSUT1 DD DSN=F9G7LB.DYNGEN.DATA,DISP=SHR
//STEP5 EXEC PGM=IEBGENER
//SYSIN DD DUMMY
//SYSPRINT DD DUMMY
//SYSUT1 DD DSN=&&CRDS,DISP=(OLD,PASS),UNIT=DISK
//SYSUT2 DD SYSOUT=A,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//STEP6 EXEC PURTFGO,NAME=DYNGEN1
//LKED.SYSPRINT DD DUMMY
//GO.FT07F001 DD DSN=&&PCH,UNIT=DISK,SPACE=(TRK,(2,1)),DISP=(,PASS)
//GO.FT08F001 DD DISP=(NEW,PASS),SPACE=(TRK,(10,10)),UNIT=DISK
//GO.FT09F001 DD *
1/4
/*
//GO.FT10F001 DD DSN=F9G7LB.DYN4.DATA,DISP=SHR
//GO.FT11F001 DD DSN=&&TRES,DISP=(,PASS),
// UNIT=DISK,SPACE=(TRK,(10,2))
//GO.SYSIN DD DSN=&&CRDS,DISP=(OLD,PASS),UNIT=DISI
//TPLOT EXEC FORTHP
//FORT.SYSPRINT DD DUMMY
//FORT.SYSIN DD DSN=F9G7LB.TPLOT.FORT,DISP=SHR
//LKED.SYSPRINT DD DUMMY
//GO.SYSIN DD *
0.0,0.8
0.9,1.0
0.9,1.0
0.8,1.0
0.85,1.1
0.8,1.0
0.7,1.0
0.8,1.1
0.85,1.1
0.9,1.1
0.85,1.1
/*
//GO.FT11F001 DD DSN=&&TRES,DISP=(OLD,DELETE)
```

```
./ REPL NAME=DISTRB
./ NUMBER NEW1=10,INCR=10
      SUBROUTINE DISTRB
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 X(5),U(2),Y(3)
      REAL*4 XMIN,XMAX,XINC,XTAR
      REAL*4 VOPT,UOPT
      LOGICAL INIT
      COMMON /COMALL/ COM(1062)
      COMMON XMIN(2),XMAX(2),XINC(2),XTAR(2)
      COMMON VOPT(15,15),UOPT(15,15,2)
      EQUIVALENCE (TIME,COM(993)),(DT,COM(994)),(TF,COM(995))
      EQUIVALENCE (WFB,COM(192)),(A8,COM(346))
      EQUIVALENCE (XNHP,COM(372)),(XNLP,COM(374))
      EQUIVALENCE (P4,COM(381)),(P7,COM(389))
      EQUIVALENCE (U4,COM(382))
      EQUIVALENCE (ZC,COM(300)),(ZF,COM(136)),(T4,COM(156))
      DATA T0/0.0/,X/5*1.0/,U/2*1.0/,Y/3*1.0/
      DATA INIT/.FALSE./
      DATA XNHPN/0.118991D+05/
      DATA XNLPN/0.987395D+04/
      DATA P4N/0.239299D+02/
      DATA P7N/0.255142D+01/
      DATA U4N/0.586468D+03/
      DATA WFBN/2.75/
      DATA A8N/2.9482558/
      DATA ZCN/0.814303D+00/
      DATA ZFN/0.833312D+00/
      DATA T4N/0.289208D+04/
      IF(INIT) GO TO 10
      READ(9,202) MDL
      WRITE(11,202) MDL
      WRITE(11,200) T0,(X(I),I=1,5)
      WRITE(11,201) (U(I),I=1,2),(Y(I),I=1,3)
      READ(10) IT,VOPT,UOPT
      READ(10) XMIN,XMAX,XINC,XTAR
      INIT=.TRUE.
   10 CONTINUE
      X(1)=XNHP/XNHPN
      X(2)=XNLP/XNLPN
      X(3)=P4/P4N
      X(4)=P7/P7N
      X(5)=U4/U4N
      CALL CNTRL(X,U,V)
      WFB=U(1)*WFBN
      A8=U(2)*A8N
      Y(1)=ZC/ZCN
      Y(2)=ZF/ZFN
      Y(3)=T4/T4N
      WRITE(11,200) TIME,(X(I),I=1,5)
      WRITE(11,201) (U(I),I=1,2),(Y(I),I=1,3)
      RETURN
  200 FORMAT(F6.3,5E14.6)
  201 FORMAT(6X,5E14.6)
  202 FORMAT(A4)
      END
      SUBROUTINE CNTRL(X,U,V)
C
C     THIS ROUTINE INTERPOLATES CONTROLS FROM THE
```

```
ENTRIES OF THE OPTIMAL CONTROL LAW.

REAL X(5),U(2)
COMMON XMIN(2),XMAX(2),XINC(2),XTAR(2)
COMMON VOPT(15,15),UOPT(15,15,2)
NX=15
FR1H=(X(1)-XMIN(1))/XINC(1)+1.0
IX1L=IFIX(FR1H)
FR1H=FR1H-FLOAT(IX1L)
IF(IX1L.GT.NX) IX1L=NX
IX1H=IX1L+1
IF(IX1H.GT.NX) IX1H=NX
FR1L=1.0-FR1H
FR2H=(X(2)-XMIN(2))/XINC(2)+1.0
IX2L=IFIX(FR2H)
FR2H=FR2H-FLOAT(IX2L)
IF(IX2L.GT.NX) IX2L=NX
IX2H=IX2L+1
IF(IX2H.GT.NX) IX2H=NX
FR2L=1.0-FR2H
V=FR1L*FR2L*VOPT(IX1L,IX2L)
U(1)=FR1L*FR2L*UOPT(IX1L,IX2L,1)
U(2)=FR1L*FR2L*UOPT(IX1L,IX2L,2)
V=V+FR1H*FR2L*VOPT(IX1H,IX2L)
U(1)=U(1)+FR1H*FR2L*UOPT(IX1H,IX2L,1)
U(2)=U(2)+FR1H*FR2L*UOPT(IX1H,IX2L,2)
V=V+FR1L*FR2H*VOPT(IX1L,IX2H)
U(1)=U(1)+FR1L*FR2H*UOPT(IX1L,IX2H,1)
U(2)=U(2)+FR1L*FR2H*UOPT(IX1L,IX2H,2)
V=V+FR1H*FR2H*VOPT(IX1H,IX2H)
U(1)=U(1)+FR1H*FR2H*UOPT(IX1H,IX2H,1)
U(2)=U(2)+FR1H*FR2H*UOPT(IX1H,IX2H,2)
RETURN
END
```

```
WFB
A8
XNHP
XNLP
P4
P7
U4
ZC
ZF
T4
FG
THEEND
 $DATAIN ISPOOL=2,FAN=.TRUE.,SI=.FALSE.,IDES=1,MODE=0,IDUMP=1,IAMTP=0
 IGASMX=2,ITRYS=200,FXFN2M=.FALSE.,FXM2CP=.FALSE.,AFTFAN=.FALSE.,
 DUMSPL=.TRUE.,TOLALL=1.E-6,DELFG=1.0,DELFN=1.0,DELSFC=1.0,
 PCNFDS=102.31,
 PRFDS=2.996,ETAFDS=.8499,PCNCDS=98.73,PRCDS=8.462,ETACDS=.8136,
 T4DS=2892.04,
 ETABDS=1.00,DPCODS=.0561,ETHPDS=.8713,ETLPDS=.9021,DPDUDS=.0584,
 T7DS=3583.6,
 ETAADS=.8430,DPAFDS=.0599,AM55=.283,AM6=.243,CVMNOZ=.9494,
 WAFCDS=221.573,
 WACCDS=54.988,HPEXT=0.0,AM=0.0,ALTP=0.0,PCBLF=0.0,PCBLC=.16,
 PCBLDU=.208,
 PCBLOB=0.0,PCBLHP=.726,PCBLLP=.066,AM23=.170,ZFDS=.8533,ZCDS=.8143,
 TFHPDS=50.0,CNHPDS=2.0,TFLPDS=130.0,CNLPDS=2.3,XNHPDS=10070.,
 XNLPDS=9651.,
 PMIHP=3.80,PMILP=4.50,VFAN=2.31,VCOMP=1.65,VCOMB=1.65,VHPTRB=.505,
 VLPTRB=.618,VAFTBN=49.77,VFDUCT=10.08, $END

 $DATAIN IDES=0,INIT=1,ITRAN=0,MODE=2,
 A8=2.9482558,WFB=2.75 $END

 $DATAIN IDES=0,INIT=1,ITRAN=0,MODE=2,
 A8=2.1442,WFB=2.0 $END

 $DATAIN IDES=0,INIT=1,ITRAN=0,MODE=2,
 A8=2.9187732,WFB=2.3375 $END

 $DATAIN MODE=2,ITRAN=1,INIT=1,
 DT=0.01,DTPRNT=0.1,TF=0.8,
 PMIHP=3.80,PMILP=4.50,
 VAFTBN=8.0,VCOMB=1.0,VCOMP=1.0,VFAN=1.0,
 VFDUCT=2.0,VHPTRB=1.0,VLPTRB=1.0,
 XNHPDS=10070.0,XNLPDS=9651.0,
 WFB=2.75,A8=2.9482558 $END

/*
```

APPENDIX B

MODEL 3 ΔX GENERATOR PROGRAM

This appendix includes the Model 3 program, and the two input data sets used to generate derivative values for Models 3A and 3B. This program is of a form that allows it to be compiled with either the dynamic programming algorithm, or the general system simulator.

```
      SUBROUTINE INIT(MDL)

      INITIALIZE ALL VALUES NEEDED
      BY THIS MODEL.

      REAL ALPHA,BETA,XD(5),XW(5),AD(5,5),AW(5,5),C(5,6)
      COMMON/MDL/ ALPHA,BETA,XD,XW,AD,AW,C
      READ(1,100) MDL
      WRITE(6,200) MDL
      READ(1,*) ALPHA,BETA
      WRITE(6,201) ALPHA,BETA
      READ(1,*) XD
      WRITE(6,202) XD
      READ(1,*) XW
      WRITE(6,203) XW
      READ(1,*) ((AD(I,J),J=1,5),I=1,5)
      WRITE(6,204) ((AD(I,J),J=1,5),I=1,5)
      READ(1,*) ((AW(I,J),J=1,5),I=1,5)
      WRITE(6,205) ((AW(I,J),J=1,5),I=1,5)
      READ(1,*) ((C(I,J),J=1,6),I=1,5)
      WRITE(6,206) ((C(I,J),J=1,6),I=1,5)
      RETURN
100   FORMAT(1A4)
200   FORMAT(1X,'DATA FOR MODEL ',A4)
201   FORMAT(1X,'ALPHA=',E16.6,'    BETA=',E16.6)
202   FORMAT(1X,'XD=',T20,5E16.6)
203   FORMAT(1X,'XW=',T20,5E16.6)
204   FORMAT(/,1X,'AD=',5(T20,5E16.6,/))
205   FORMAT(/,1X,'AW=',5(T20,5E16.6,/))
206   FORMAT(/,1X,'C=',5(T20,6E16.6,/),'1')
      END
```

```
      SUBROUTINE STDST(U,X)

      THIS ROUTINE EVALUATES G(U),
      THE STEADY STATE FUNCTION.

      REAL U(2),X(5),US(2)
      REAL ALPHA,BETA,XD(5),XW(5),AD(5,5),AW(5,5),C(5,6)
      COMMON/MDL/ ALPHA,BETA,XD,XW,AD,AW,C
      DO 10 I=1,2
      US(I)=ALPHA*U(I)+BETA
   10 CONTINUE
      DO 20 I=1,5
      X(I)=C(I,1)*US(1)+C(I,2)*US(2)
     #+C(I,5)*US(1)**C(I,3)*US(2)**C(I,4)
     #+C(I,6)
   20 CONTINUE
      RETURN
      END
      SUBROUTINE XDOT(X,U,DX)
C
C     THIS ROUTINE EVALUATES
C     THE DERIVATIVE FOR ANY
C     STATE AND CONTROLS.
C
      REAL X(5),U(2),DX(5),G(5),WK(5),USV(2),A(5,5)
      REAL ALPHA,BETA,XD(5),XW(5),AD(5,5),AW(5,5),C(5,6)
      COMMON/MDL/ ALPHA,BETA,XD,XW,AD,AW,C
      CALL STDST(U,G)
      DO 10 J=1,5
      WK(J)=X(J)-G(J)
   10 CONTINUE
      CALL AMAT(X,U,A)
      CALL VMULFF(A,WK,5,5,1,5,5,DX,5,IER)
      RETURN
      END
```

```
      SUBROUTINE AMAT(X,U,A)
C
C     THIS ROUTINE GENERATES THE
C     A MATRIX FOR ANY STATE
C     AND CONTROLS.
C
      REAL X(5),U(2),A(5,5)
      REAL ALPHA,BETA,XD(5),XW(5),AD(5,5),AW(5,5),C(5,6)
      COMMON/MDL/ ALPHA,BETA,XD,XW,AD,AW,C
      DO 10 J=1,5
      FRW=(XD(J)-X(J))/(XD(J)-XW(J))
      FRD=(X(J)-XW(J))/(XD(J)-XW(J))
      DO 10 I=1,5
      A(I,J)=AW(I,J)*FRW+AD(I,J)*FRD
   10 CONTINUE
      RETURN
      END
```

```
3A
1.1,-0.1
1.0,1.0,1.0,1.0,1.0
0.9,0.7897,0.7381,0.9401,0.9454
-3.8,-1.277,2.067,-1.152,1.448
2.748,-5.39,1.585,-1.991,1.071
377.9,49.51,-264.9,86.807,78.91
31.26,139.39,-6.269,-88.69,27.83
-176.5,23.91,-10.27,-37.4,-246.7
-4.744,-1.3888,3.2468,-1.4591,1.1969
0.82186,-26.726,2.5585,-1.8609,0.45548
475.73,137.55,-328.91,27.791,91.495
-50.103,110.91,63.188,-116.69,8.2883
-186.77,-67.682,-41.681,24.586,-243.23
0.24267,-0.00218,1.90082,8.09916,0.02864,0.73088
1.01593,0.85407,0.89872,0.66919,-0.61879,-0.05121
0.73445,0.10133,6.90586,3.09409,0.011495,0.15272
0.77234,-0.35905,2.49867,2.87415,-0.075198,0.66191
0.39503,-0.27262,-3.44682,13.4468,0.01838,0.85921
```

```
3B
2.31778,-1.31778
1.0,1.0,1.0,1.0,1.0
0.9,0.7897,0.7381,0.9401,0.9454
-3.8,-1.277,2.067,-1.152,1.448
2.748,-5.39,1.585,-1.991,1.071
377.9,49.51,-264.9,86.807,78.91
31.26,139.39,-6.269,-88.69,27.83
-176.5,23.91,-10.27,-37.4,-246.7
-4.744,-1.3888,3.2468,-1.4591,1.1969
0.82186,-26.726,2.5585,-1.8609,0.45548
475.73,137.55,-328.91,27.791,91.495
-50.103,110.91,63.188,-116.69,8.2883
-186.77,-67.682,-41.681,24.586,-243.23
0.1553,0.0028,1.0,1.0,0.0,0.8418
0.1619,0.1707,1.0,1.0,0.0,0.6674
0.5351,-0.1208,1.0,1.0,0.0,0.5857
0.5878,-0.49313,1.0,1.0,0.0,0.9053
0.2962,-0.2099,1.0,1.0,0.0,0.9137
```

APPENDIX C

MODEL 4 $\Delta\dot{X}$ GENERATOR PROGRAM

This appendix includes the Model 4 program, and the input data set used to generate derivative values for Model 4. This program is of a form that allows it to be compiled with either the dynamic programming algorithm, or the general system simulator.

```
      SUBROUTINE INIT(MDL)
C
C     INITIALIZE ALL VALUES NEEDED
C     BY THIS MODEL.
C
      INTEGER ORD,N,IA(5)
      REAL U2,U1(5),XI(5,5),DXI(5,5),XIP(5,5),XIM(5,5),DEL(5)
      REAL X(5),U(2),A1(5,5),A3(5,5),A5(5,5),A(5,5)
      REAL AP(5,5),AM(5,5)
      COMMON/MDL/ ORD,N,IA,U2,U1,XI,DXI,XIP,XIM,DEL
      COMMON/MDL/ A1,A3,A5
      COMMON/MDL/ AP,AM
      DATA MDL4/'4   '/
      MDL=MDL4
      READ(1,*) ORD,N
      READ(1,*) U2
      DO 10 I=1,N
      READ(1,*) U1(I)
      READ(1,*) (XI(I,J),J=1,ORD)
      READ(1,*) (DXI(I,J),J=1,ORD)
      READ(1,*) IA(I),DEL(I)
      IF(IA(I).NE.1.AND.IA(I).NE.2) STOP 16
      READ(1,*) (XIP(I,J),J=1,ORD)
      IF(IA(I).EQ.1) GO TO 10
      READ(1,*) (XIM(I,J),J=1,ORD)
   10 CONTINUE
      READ(1,100) ((A1(I,J),J=1,ORD),I=1,ORD)
      READ(1,100) ((A3(I,J),J=1,ORD),I=1,ORD)
      READ(1,100) ((A5(I,J),J=1,ORD),I=1,ORD)
      READ(1,100) ((AP(I,J),J=1,ORD),I=1,ORD)
      READ(1,100) ((AM(I,J),J=1,ORD),I=1,ORD)
      RETURN
  100 FORMAT(5E13.6)
      END
```

```
C
C     THE NEXT EIGHT ROUTINES ARE THE
C     HERMITE INTERPOLATION METHOD OF
C     HILDEBRAND, MODIFIED AS PER THE TEXT.
C
      REAL FUNCTION L(I,U)
      INTEGER ORD,N,IA(5)
      REAL U2,U1(5),XI(5,5),DXI(5,5),XIP(5,5),XIM(5,5),DEL(5)
      REAL X(5),U(2),A1(5,5),A3(5,5),A5(5,5),A(5,5)
      REAL AP(5,5),AM(5,5)
      COMMON/MDL/ ORD,N,IA,U2,U1,XI,DXI,XIP,XIM,DEL
      COMMON/MDL/ A1,A3,A5
      COMMON/MDL/ AP,AM
      L=1.0
      DO 10 K=1,N
      IF(K.EQ.I) GO TO 10
      L=L*(U(1)-U1(K))/(U1(I)-U1(K))
   10 CONTINUE
      RETURN
      END
      REAL FUNCTION DL(I,UU)
      INTEGER ORD,N,IA(5)
      REAL U2,U1(5),XI(5,5),DXI(5,5),XIP(5,5),XIM(5,5),DEL(5)
      REAL X(5),U(2),A1(5,5),A3(5,5),A5(5,5),A(5,5)
      REAL AP(5,5),AM(5,5)
      COMMON/MDL/ ORD,N,IA,U2,U1,XI,DXI,XIP,XIM,DEL
      COMMON/MDL/ A1,A3,A5
      COMMON/MDL/ AP,AM
      DL=0.0
      DO 10 K=1,N
      IF(K.EQ.I) GO TO 10
      PRD=1.0
      DO 20 KK=1,N
      IF(KK.EQ.I) GO TO 20
      TERM=UU-U1(KK)
      IF(KK.EQ.K) TERM=1.0
      PRD=PRD*TERM/(U1(I)-U1(KK))
   20 CONTINUE
      DL=DL+PRD
   10 CONTINUE
      RETURN
      END



      REAL FUNCTION R(I,U)
      INTEGER ORD,N,IA(5)
      REAL U2,U1(5),XI(5,5),DXI(5,5),XIP(5,5),XIM(5,5),DEL(5)
      REAL X(5),U(2),A1(5,5),A3(5,5),A5(5,5),A(5,5)
      REAL AP(5,5),AM(5,5)
      COMMON/MDL/ ORD,N,IA,U2,U1,XI,DXI,XIP,XIM,DEL
      COMMON/MDL/ A1,A3,A5
      COMMON/MDL/ AP,AM
      R=1.0-2.0*DL(I,U1(I))*(U(1)-U1(I))
      RETURN
      END



      REAL FUNCTION S(I,U)
      INTEGER ORD,N,IA(5)
      REAL U2,U1(5),XI(5,5),DXI(5,5),XIP(5,5),XIM(5,5),DEL(5)
      REAL X(5),U(2),A1(5,5),A3(5,5),A5(5,5),A(5,5)
      REAL AP(5,5),AM(5,5)
      COMMON/MDL/ ORD,N,IA,U2,U1,XI,DXI,XIP,XIM,DEL
      COMMON/MDL/ A1,A3,A5
      COMMON/MDL/ AP,AM
      S=(U(1)-U1(I))
      RETURN
      END



      REAL FUNCTION H(I,U)
      INTEGER ORD,N,IA(5)
      REAL U2,U1(5),XI(5,5),DXI(5,5),XIP(5,5),XIM(5,5),DEL(5)
      REAL X(5),U(2),A1(5,5),A3(5,5),A5(5,5),A(5,5)
      REAL AP(5,5),AM(5,5)
      REAL L
      COMMON/MDL/ ORD,N,IA,U2,U1,XI,DXI,XIP,XIM,DEL
      COMMON/MDL/ A1,A3,A5
      COMMON/MDL/ AP,AM
      H=R(I,U)*(L(I,U))**2
      RETURN
      END
      REAL FUNCTION HB(I,U)
      INTEGER ORD,N,IA(5)
      REAL U2,U1(5),XI(5,5),DXI(5,5),XIP(5,5),XIM(5,5),DEL(5)
      REAL X(5),U(2),A1(5,5),A3(5,5),A5(5,5),A(5,5)
      REAL AP(5,5),AM(5,5)
      REAL L
      COMMON/MDL/ ORD,N,IA,U2,U1,XI,DXI,XIP,XIM,DEL
      COMMON/MDL/ A1,A3,A5
      COMMON/MDL/ AP,AM
      HB=S(I,U)*(L(I,U))**2
      RETURN
      END



      REAL FUNCTION BOX(I,J,U)
      INTEGER ORD,N,IA(5)
      REAL U2,U1(5),XI(5,5),DXI(5,5),XIP(5,5),XIM(5,5),DEL(5)
      REAL X(5),U(2),A1(5,5),A3(5,5),A5(5,5),A(5,5)
      REAL AP(5,5),AM(5,5)
      COMMON/MDL/ ORD,N,IA,U2,U1,XI,DXI,XIP,XIM,DEL
      COMMON/MDL/ A1,A3,A5
      COMMON/MDL/ AP,AM
      IF(IA(I).EQ.2) GO TO 10
      BOX=1.0+(U(2)-U2)*(XIP(I,J)-XI(I,J))/(DEL(I)*XI(I,J))
      RETURN
   10 CONTINUE
      BOX=1.0+(U(2)-U2)*(XIP(I,J)-XIM(I,J))/(2.0*DEL(I)*XI(I,J))+
     &(U(2)-U2)**2*(XIP(I,J)+XIM(I,J)-2.0*XI(I,J))/
     &(2.0*DEL(I)**2*XI(I,J))
      RETURN
      END



      REAL FUNCTION Y(J,U)
      INTEGER ORD,N,IA(5)
      REAL U2,U1(5),XI(5,5),DXI(5,5),XIP(5,5),XIM(5,5),DEL(5)
      REAL X(5),U(2),A1(5,5),A3(5,5),A5(5,5),A(5,5)
      REAL AP(5,5),AM(5,5)
      COMMON/MDL/ ORD,N,IA,U2,U1,XI,DXI,XIP,XIM,DEL
      COMMON/MDL/ A1,A3,A5
      COMMON/MDL/ AP,AM
      Y=0.0
      DO 10 K=1,N
      Y=Y+H(K,U)*XI(K,J)*BOX(K,J,U)+HB(K,U)*DXI(K,J)
   10 CONTINUE
      RETURN
      END



      SUBROUTINE STDST(U,X)

      THIS ROUTINE EVALUATES G(U),
      THE STEADY STATE FUNCTION.

      INTEGER ORD,N,IA(5)
      REAL U2,U1(5),XI(5,5),DXI(5,5),XIP(5,5),XIM(5,5),DEL(5)
      REAL X(5),U(2),A1(5,5),A3(5,5),A5(5,5),A(5,5)
      REAL AP(5,5),AM(5,5)
      COMMON/MDL/ ORD,N,IA,U2,U1,XI,DXI,XIP,XIM,DEL
      COMMON/MDL/ A1,A3,A5
      COMMON/MDL/ AP,AM
      DO 10 J=1,5
      X(J)=Y(J,U)
   10 CONTINUE
      RETURN
      END
      SUBROUTINE XDOT(X,U,DX)

      THIS ROUTINE EVALUATES THE
      DERIVATIVE FOR ANY
      STATE AND CONTROLS.

      INTEGER ORD,N,IA(5)
      REAL U2,U1(5),XI(5,5),DXI(5,5),XIP(5,5),XIM(5,5),DEL(5)
      REAL X(5),U(2),A1(5,5),A3(5,5),A5(5,5),A(5,5)
      REAL AP(5,5),AM(5,5)
      REAL DX(5),G(5),WK(5),USV(2)
      COMMON/MDL/ ORD,N,IA,U2,U1,XI,DXI,XIP,XIM,DEL
      COMMON/MDL/ A1,A3,A5
      COMMON/MDL/ AP,AM
      DATA USV/2*1.0/,G/5*1.0/
      IF(U(1).EQ.USV(1).AND.U(2).EQ.USV(2)) GO TO 5
      USV(1)=U(1)
      USV(2)=U(2)
      CALL STDST(U,G)
    5 CONTINUE
      DO 10 J=1,5
      WK(J)=X(J)-G(J)
   10 CONTINUE
      CALL AMAT(X,U,A)
      CALL VMULFF(A,WK,5,5,1,5,5,DX,5,IER)
      RETURN
      END
```

```
      SUBROUTINE AMAT(X,U,A)

      THIS ROUTINE GENERATES THE
      A MATRIX FOR ANY STATE
      AND CONTROLS.

      INTEGER ORD,N,IA(5)
      REAL U2,U1(5),XI(5,5),DXI(5,5),XIP(5,5),XIM(5,5),DEL(5)
      REAL X(5),U(2),A1(5,5),A3(5,5),A5(5,5),A(5,5)
      REAL AP(5,5),AM(5,5)
      COMMON/MDL/ ORD,N,IA,U2,U1,XI,DXI,XIP,XIM,DEL
      COMMON/MDL/ A1,A3,A5
      COMMON/MDL/ AP,AM
      FR1=(X(1)-XI(3,1))*(X(1)-XI(5,1))
     &/((XI(1,1)-XI(3,1))*(XI(1,1)-XI(5,1)))
      FR3=(X(1)-XI(1,1))*(X(1)-XI(5,1))
     &/((XI(3,1)-XI(1,1))*(XI(3,1)-XI(5,1)))
      FR5=(X(1)-XI(1,1))*(X(1)-XI(3,1))
     &/((XI(5,1)-XI(1,1))*(XI(5,1)-XI(3,1)))
      U1P=U2+DEL(1)
      U1M=U2-DEL(1)
      FRP=(U(2)-U2)*(U(2)-U1M)/((U1P-U2)*(U1P-U1M))
      FR=(U(2)-U1P)*(U(2)-U1M)/((U2-U1P)*(U2-U1M))
      FRM=(U(2)-U1P)*(U(2)-U2)/((U1M-U1P)*(U1M-U2))
      DO 10 I=1,ORD
      DO 10 J=1,ORD
      A(I,J)=A1(I,J)*FR1+A3(I,J)*FR3+A5(I,J)*FR5
      A(I,J)=A(I,J)*(FRP*AP(I,J)/A1(I,J)+FR+FRM*AM(I,J)/A1(I,J))
   10 CONTINUE
      RETURN
      END
```

```
5,5
1.0
1.0
 0.100000E+01 0.100000E+01 0.100000E+01 0.100000E+01 0.100000E+01
 0.258696E+00 0.232006E+00 0.741347E+00 0.536700E+00 0.368200E+00
2,0.13567
 0.102201E+01 0.103699E+01 0.100909E+01 0.922052E+00 0.100115E+01
 0.983596E+00 0.944368E+00 0.972825E+00 0.106272E+01 0.102339E+01
0.945455
 0.984789E+00 0.985735E+00 0.956604E+00 0.968504E+00 0.981774E+00
 0.280586E+00 0.266837E+00 0.798173E+00 0.582468E+00 0.338984E+00
2,0.13567
 0.101009E+01 0.102221E+01 0.969378E+00 0.894319E+00 0.980202E+00
 0.968989E+00 0.927607E+00 0.931285E+00 0.102997E+01 0.100344E+01
0.872727
 0.966384E+00 0.959990E+00 0.898753E+00 0.923883E+00 0.956888E+00
 0.234901E+00 0.417786E+00 0.794424E+00 0.635940E+00 0.345763E+00
2,0.13567
 0.990008E+00 0.100028E+01 0.913027E+00 0.855471E+00 0.953457E+00
 0.948022E+00 0.903912E+00 0.874094E+00 0.984805E+00 0.976756E+00
0.8
 0.948434E+00 0.928751E+00 0.840108E+00 0.876949E+00 0.931145E+00
 0.265795E+00 0.442441E+00 0.827564E+00 0.656344E+00 0.353039E+00
2,0.13567
 0.967771E+00 0.975912E+00 0.853931E+00 0.814828E+00 0.926353E+00
 0.926608E+00 0.878112E+00 0.816210E+00 0.938098E+00 0.948750E+00
0.727273
 0.928298E+00 0.895714E+00 0.779278E+00 0.828553E+00 0.904594E+00
 0.286381E+00 0.464184E+00 0.843807E+00 0.673048E+00 0.375500E+00
2,0.13567
 0.947921E+00 0.940920E+00 0.794242E+00 0.770606E+00 0.898666E+00
 0.907843E+00 0.845919E+00 0.758233E+00 0.887010E+00 0.919949E+00
-0.439400E+01-0.988129E+00 0.252252E+01-0.157586E+01 0.123898E+01
 0.940742E+00-0.609300E+01 0.266571E+01-0.184044E+01 0.707764E+00
 0.506886E+03 0.519848E+02-0.350590E+03 0.146869E+03 0.116621E+03
-0.585898E+02 0.128300E+03 0.607624E+02-0.121120E+03 0.113162E+02
-0.159711E+03 0.266566E+02-0.189252E+02-0.446012E+02-0.242670E+03
-0.471740E+01-0.124636E+01 0.298608E+01-0.187860E+01 0.143928E+01
 0.552601E+00-0.357390E+01 0.278204E+01-0.172109E+01 0.723029E+00
 0.450922E+03 0.813586E+02-0.332560E+03 0.130044E+03 0.113546E+03
-0.517156E+02 0.888692E+02 0.585464E+02-0.119240E+03 0.102770E+02
-0.130919E+03-0.451477E+01-0.341646E+02-0.372193E+02-0.247470E+03
-0.478520E+01-0.121176E+01 0.341043E+01-0.209109E+01 0.135367E+01
 0.499094E+00-0.347050E+01 0.258839E+01-0.140777E+01 0.625560E+00
 0.398468E+03 0.803642E+02-0.318460E+03 0.113763E+03 0.102113E+03
-0.434077E+02 0.844226E+02 0.561960E+02-0.116470E+03 0.944216E+01
-0.113790E+03-0.536230E+01-0.455125E+02-0.323419E+02-0.242430E+03
-0.731320E+01-0.635114E+00 0.284068E+01-0.166920E+01 0.150321E+01
 0.130344E+01-0.549860E+01 0.281015E+01-0.227371E+01 0.753975E+00
 0.526328E+03 0.521252E+02-0.329620E+03 0.121962E+03 0.127924E+03
-0.612061E+02 0.109032E+03 0.571358E+02-0.122370E+03 0.991498E+01
-0.120121E+03 0.167193E+02-0.340961E+02-0.332664E+02-0.247680E+03
-0.324940E+01-0.914027E+00 0.194351E+01-0.139379E+01 0.147837E+01
 0.100929E+01-0.360220E+01 0.259057E+01-0.227073E+01 0.665059E+00
 0.566953E+03 0.152253E+03-0.340280E+03 0.315925E+02 0.116803E+03
-0.595459E+02 0.119995E+03 0.635521E+02-0.123660E+03 0.102124E+02
-0.229269E+03-0.619973E+02-0.938650E+01 0.198160E+02-0.253130E+03
```

APPENDIX D

GENERAL SYSTEM SIMULATOR

This appendix includes the JCL used to run controller simulations on the model from which the control law was generated, and the general system simulator program, TRES. This program uses a Euler integration technique with a user specified time increment. A data set of time response values is generated, with each record consisting of time, state, control, and output values. This JCL contains three jobs, one each for models 3A, 3B and 4.

```
//TPLOT3A JOB (CF,F081),711153015,NOTIFY=F9G7LB,
// TIME=1
/*ROUTE  PRINT REMOTE2
/*SETUP PLOTS,NOCODE
//TRES EXEC FORTH,LIB4='IMSL.SINGLE'
//FORT.SYSPRINT DD DUMMY
//FORT.SYSIN DD DSN=F9G7LB.TRES.FORT,DISP=SHR
//             DD DSN=F9G7LB.MODEL3.FORT,DISP=SHR
//LKED.SYSPRINT DD DUMMY
//GO.FT01F001 DD DSN=F9G7LB.MODEL3A.DATA,DISP=SHR
//GO.SYSIN DD *
1.0,1.0,0.005
41,0.001
/*
//GO.FT10F001 DD DSN=F9G7LB.DYN3A.DATA,DISP=SHR
//GO.FT11F001 DD DSN=&&TRES,DISP=(,PASS),
//           UNIT=DISK,SPACE=(TRK,(10,2))
//LIST EXEC TSOLIST,NAME='&&TRES',COND=EVEN
//TPLOT EXEC FORTHP
//FORT.SYSPRINT DD DUMMY
//FORT.SYSIN DD DSN=F9G7LB.TPLOT.FORT,DISP=SHR
//LKED.SYSPRINT DD DUMMY
//GO.SYSIN DD *
0.0,0.04
0.0,2.0
0.0,4.0
0.0,100.0
0.0,100.0
-150.0,50.0
0.8,1.0
0.8,1.0
0.0,1.0
0.0,1.0
1.0,1.25
/*
//GO.FT11F001 DD DSN=&&TRES,DISP=(OLD,DELETE)
//
//TPLOT3B JOB (CF,F081),711153015,NOTIFY=F9G7LB,
// TIME=1
/*ROUTE  PRINT REMOTE2
/*SETUP PLOTS,NOCODE
//TRES EXEC FORTH,LIB4='IMSL.SINGLE'
//FORT.SYSPRINT DD DUMMY
//FORT.SYSIN DD DSN=F9G7LB.TRES.FORT,DISP=SHR
//             DD DSN=F9G7LB.MODEL3.FORT,DISP=SHR
//LKED.SYSPRINT DD DUMMY
//GO.FT01F001 DD DSN=F9G7LB.MODEL3B.DATA,DISP=SHR
//GO.SYSIN DD *
1.0,1.0,0.005
27,0.001
/*
//GO.FT10F001 DD DSN=F9G7LB.DYN3B.DATA,DISP=SHR
//GO.FT11F001 DD DSN=&&TRES,DISP=(,PASS),
//           UNIT=DISK,SPACE=(TRK,(10,2))
//LIST EXEC TSOLIST,NAME='&&TRES',COND=EVEN
//TPLOT EXEC FORTHP
//FORT.SYSPRINT DD DUMMY
//FORT.SYSIN DD DSN=F9G7LB.TPLOT.FORT,DISP=SHR
//LKED.SYSPRINT DD DUMMY
```

```
//GO.SYSIN DD *
0.0,0.04
0.0,5.0
0.0,10.0
0.0,500.0
0.0,500.0
-500.0,0.0
0.7,0.9
0.8,0.9
-2.0,1.0
-3.0,1.0
1.0,1.5
/*
//GO.FT11F001 DD DSN=&&TRES,DISP=(OLD,DELETE)
//
//TPLOT4  JOB (CF,F081),711153015,NOTIFY=F9G7LB,
// TIME=1
/*ROUTE  PRINT REMOTE2
/*SETUP PLOTS,NOCODE
//TRES EXEC FORTH,LIB4='IMSL.SINGLE'
//FORT.SYSPRINT DD DUMMY
//FORT.SYSIN DD DSN=F9G7LB.TRES.FORT,DISP=SHR
//             DD DSN=F9G7LB.MODEL4.FORT,DISP=SHR
//LKED.SYSPRINT DD DUMMY
//GO.FT01F001 DD DSN=F9G7LB.MODEL4.DATA,DISP=SHR
//GO.SYSIN DD *
1.0,1.0,0.005
800,0.001
/*
//GO.FT10F001 DD DSN=F9G7LB.DYN4.DATA,DISP=SHR
//GO.FT11F001 DD DSN=&&TRES,DISP=(,PASS),
//           UNIT=DISK,SPACE=(TRK,(10,2))
//LIST EXEC TSOLIST,NAME='&&TRES',COND=EVEN
//TPLOT EXEC FORTHP
//FORT.SYSPRINT DD DUMMY
//FORT.SYSIN DD DSN=F9G7LB.TPLOT.FORT,DISP=SHR
//LKED.SYSPRINT DD DUMMY
//GO.SYSIN DD *
0.0,0.8
0.9,1.0
0.9,1.0
0.8,1.0
0.85,1.1
0.8,1.0
0.7,1.0
0.8,1.1
0.85,1.1
0.9,1.1
0.85,1.1
/*
//GO.FT11F001 DD DSN=&&TRES,DISP=(OLD,DELETE)
```

```
      THIS PROGRAM COMPUTES THE TIME RESPONSE OF A MODEL
      UNDER THE CONTROL OF AN OPTIMAL CONTROL LAW,
      AS GENERATED BY THE DYNANIC PROGRAMMING METHOD.

      REAL X(5),DX(5),U(2),Y(3)
      COMMON XMIN(2),XMAX(2),XINC(2),XTAR(2)
      COMMON VOPT(15,15),UOPT(15,15,2)
      T=0.0
      CALL INIT(MDL)
      READ(10) IT,VOPT,UOPT
      READ(10) XMIN,XMAX,XINC,XTAR
      WRITE(11,200) MDL
      READ(5,*) U,EPS
      CALL STDST(U,X)
      CALL CNTRL(X,U,VI)
      WRITE(6,300) VI
      WRITE(11,201) T,X
      CALL OUTPUT(X,U,Y)
      WRITE(11,202) U,Y
    5 READ(5,*,END=999) N,DT
      DO 10 I=1,N
      CALL CNTRL(X,U,V)
      CALL XDOT(X,U,DX)
      DO 20 J=1,5
      X(J)=X(J)+DX(J)*DT
   20 CONTINUE
      CALL OUTPUT(X,U,Y)
      T=T+DT
      WRITE(11,201) T,X
      WRITE(11,202) U,Y
      DIST=SQRT((X(1)-XTAR(1))**2+(X(2)-XTAR(2))**2)
      REL=(T-VI)/T
      IF(DIST.LT.EPS) WRITE(6,301) T,REL
      IF(DIST.LT.EPS) EPS=0.0
   10 CONTINUE
      GO TO 5
  999 CONTINUE
      STOP
  200 FORMAT(A4)
  201 FORMAT(E10.3,5E14.6)
  202 FORMAT(10X,5E14.6)
  300 FORMAT(1X,'PREDICTED TIME =',F8.4)
  301 FORMAT(1X,'ACTUAL TIME =',F8.4/1X,'RELATIVE ERROR =',E11.4)
      END
      SUBROUTINE CNTRL(X,U,V)

      THIS ROUTINE INTERPOLATES CONTROLS FROM THE
      ENTRIES OF THE OPTIMAL CONTROL LAW.

      REAL X(5),U(2)
      COMMON XMIN(2),XMAX(2),XINC(2),XTAR(2)
      COMMON VOPT(15,15),UOPT(15,15,2)
      NX=15
      IX1L=NX
      IX1H=NX
      FR1L=1.0
      FR1H=0.0
      IF(X(1).GE.XMAX(1)) GO TO 10
      FR1H=(X(1)-XMIN(1))/XINC(1)+1.0
      IX1L=IFIX(FR1H)
      FR1H=FR1H-FLOAT(IX1L)
      IX1H=IX1L+1
      FR1L=1.0-FR1H
   10 CONTINUE
      IX2L=NX
      IX2H=NX
      FR2L=1.0
      FR2H=0.0
      IF(X(2).GE.XMAX(2)) GO TO 20
      FR2H=(X(2)-XMIN(2))/XINC(2)+1.0
      IX2L=IFIX(FR2H)
      FR2H=FR2H-FLOAT(IX2L)
      IX2H=IX2L+1
      FR2L=1.0-FR2H
   20 CONTINUE
      V=FR1L*FR2L*VOPT(IX1L,IX2L)
      U(1)=FR1L*FR2L*UOPT(IX1L,IX2L,1)
      U(2)=FR1L*FR2L*UOPT(IX1L,IX2L,2)
      V=V+FR1H*FR2L*VOPT(IX1H,IX2L)
      U(1)=U(1)+FR1H*FR2L*UOPT(IX1H,IX2L,1)
      U(2)=U(2)+FR1H*FR2L*UOPT(IX1H,IX2L,2)
      V=V+FR1L*FR2H*VOPT(IX1L,IX2H)
      U(1)=U(1)+FR1L*FR2H*UOPT(IX1L,IX2H,1)
      U(2)=U(2)+FR1L*FR2H*UOPT(IX1L,IX2H,2)
      V=V+FR1H*FR2H*VOPT(IX1H,IX2H)
      U(1)=U(1)+FR1H*FR2H*UOPT(IX1H,IX2H,1)
      U(2)=U(2)+FR1H*FR2H*UOPT(IX1H,IX2H,2)
      RETURN
      END


      SUBROUTINE OUTPUT(X,U,Y)

      THIS ROUTINE, COMMON TO ALL MODELS,
      EVALUATES OUTPUT VARIABLES.

      REAL X(2),U(2),Y(3)
      REAL C(3,2),D(3,2),E(3),WK(3)
      DATA C/-0.61059,-0.20154,-0.58229,-0.10759,-0.45813,0.46872/
      DATA D/0.50292,0.20423,0.18877,0.17689,0.14724,-0.92545/
      DATA E/1.03837,1.30796,1.85021/
      CALL VMULFF(C,X,3,2,1,3,2,Y,3,IER)
      CALL VMULFF(D,U,3,2,1,3,2,WK,3,IER)
      DO 10 I=1,3
      Y(I)=Y(I)+WK(I)+E(I)
   10 CONTINUE
      RETURN
      END
```

APPENDIX E

TIME RESPONSE PLOT PROGRAM

This appendix includes the time response plot program, designed to plot ten graphs: states, controls, and outputs versus time. The JCL in appendix D links this program to the general system simulator, and the JCL in appendix A links it to the DYNGEN simulator. In either case, this program's input consists of a data set of time response values, and user specified axis limits.

```
      THIS PROGRAM PLOTS THE TIME RESPONSE
      DATA GENERATED BY TRES, INCLUDING
      STATES, CONTROLS, AND OUTPUTS.

      REAL TS(1003),XS(1003,5),US(1003,2),YS(1003,3)
      INTEGER ITXT(2,10)
      DATA ITXT/'X(1)','=NC ','X(2)','=NF ',
     &'X(3)','=P4 ','X(4)','=P7 ','X(5)','=U4 ',
     &'U(1)','=WFB','U(2)','=A8 ',
     &'Y(1)','=ZC ','Y(2)','=ZF ','Y(3)','=T4 '/
      TLEN=8.0
      XLEN=5.0
      READ(11,200) MDL
      DO 20 NS=1,1001
      READ(11,*,END=21) TS(NS),(XS(NS,J),J=1,5)
      READ(11,*,END=21) (US(NS,J),J=1,2),(YS(NS,J),J=1,3)
   20 CONTINUE
   21 CONTINUE
      NS=NS-1
      CALL PLOTS(711153015)
      CALL PLOT(0.0,0.5,3)
      CALL PLOT(0.0,10.5,2)
      CALL PLOT(1.75,1.5,-3)
      READ(5,*) TMIN,TMAX
      TS(NS+1)=TMIN
      TS(NS+2)=(TMAX-TMIN)/TLEN
      DO 30 J=1,5
      CALL SYMBOL(-0.75,0.5,0.14,29HCONTROLLER RESPONSE OF MODEL ,
     &90.0,29)
      CALL SYMBOL(999.0,999.0,0.14,MDL,90.0,4)
      CALL SYMBOL(-0.5,0.5,0.14,23HFOR INITIAL X=(1.0,1.0),
     &90.0,23)
      READ(5,*) XMIN,XMAX
      XS(NS+1,J)=XMAX
      XS(NS+2,J)=(XMIN-XMAX)/XLEN
      CALL AXIS(0.0,0.0,ITXT(1,J),-8,XLEN,0.0,
     #XS(NS+1,J),XS(NS+2,J))
      CALL AXIS(XLEN,0.0,15HTIME IN SECONDS,-15,TLEN,90.0,
     #TS(NS+1),TS(NS+2))
      CALL LINE(XS(1,J),TS,NS,1,0,0)
      CALL PLOT(6.75,-1.0,3)
      CALL PLOT(6.75,9.0,2)
      CALL PLOT(8.5,0.0,-3)
   30 CONTINUE
      DO 40 J=1,2
      CALL SYMBOL(-0.75,0.5,0.14,29HCONTROLLER RESPONSE OF MODEL ,
     &90.0,29)
      CALL SYMBOL(999.0,999.0,0.14,MDL,90.0,4)
      CALL SYMBOL(-0.5,0.5,0.14,23HFOR INITIAL X=(1.0,1.0),
     &90.0,23)
      READ(5,*) UMIN,UMAX
      US(NS+1,J)=UMAX
      US(NS+2,J)=(UMIN-UMAX)/XLEN
      CALL AXIS(0.0,0.0,ITXT(1,J+5),-8,XLEN,0.0,
     #US(NS+1,J),US(NS+2,J))
      CALL AXIS(XLEN,0.0,15HTIME IN SECONDS,-15,TLEN,90.0,
     #TS(NS+1),TS(NS+2))
      CALL LINE(US(1,J),TS,NS,1,0,0)
      CALL PLOT(6.75,-1.0,3)
      CALL PLOT(6.75,9.0,2)
      CALL PLOT(8.5,0.0,-3)
   40 CONTINUE
      DO 50 J=1,3
      CALL SYMBOL(-0.75,0.5,0.14,29HCONTROLLER RESPONSE OF MODEL
     &90.0,29)
      CALL SYMBOL(999.0,999.0,0.14,MDL,90.0,4)
      CALL SYMBOL(-0.5,0.5,0.14,23HFOR INITIAL X=(1.0,1,0),
     &90.0,23)
      READ(5,*) YMIN,YMAX
      YS(NS+1,J)=YMAX
      YS(NS+2,J)=(YMIN-YMAX)/XLEN
      CALL AXIS(0.0,0.0,ITXT(1,J+7),-8,XLEN,0.0,
     #YS(NS+1,J),YS(NS+2,J))
      CALL AXIS(XLEN,0.0,15HTIME IN SECONDS,-15,TLEN,90.0,
     #TS(NS+1),TS(NS+2))
      CALL LINE(YS(1,J),TS,NS,1,0,0)
      CALL PLOT(6.75,-1.0,3)
      CALL PLOT(6.75,9.0,2)
      CALL PLOT(8.5,0.0,-3)
   50 CONTINUE
      CALL PLOT(0.0,0.0,999)
      RETURN
  200 FORMAT(A4)
      END
```

APPENDIX F

DYNAMIC SUCCESSIVE APPROXIMATION PROGRAM

This appendix includes the JCL used to generate control laws, and the dynamic successive approximation program. The program is designed to be compiled with a model program consisting of subroutines named INIT, XDOT, OUTPUT, and COMPLT. The program's input includes state and control space limits and quantization, target point, time increment, and output constraint values. A data set containing the control law (initialized to the target's steady state controls) and associated cost estimates is read by the program. A user specified number of iterations is performed, and the new control law is rewritten to the input data set. Subsequent runs then perform more iterations on this updated data set.

```
//DYNPRG3A JOB (CF,F081),711153015,NOTIFY=F9G7LB,
// TIME=4
/*ROUTE  PRINT REMOTE2
//DYNPRG EXEC FORTH,LIB4='IMSL.SINGLE'
//FORT.SYSPRINT DD DUMMY
//FORT.SYSIN DD DSN=F9G7LB.DYNPRG.FORT,DISP=SHR
// DD DSN=F9G7LB.MODEL3.FORT,DISP=SHR
//LKED.SYSPRINT DD DUMMY
//GO.FT01F001 DD DSN=F9G7LB.MODEL3A.DATA,DISP=SHR
//GO.SYSIN DD *
0.948434,0.928751
0.01,0.01
0.74,0.74
1.0,1.0
0.02,0.02
1.15,1.105,1.08
5,250,0.01,0.01
T,F,F
/*
//GO.FT10F001 DD DSN=F9G7LB.DYN3A.DATA,DISP=OLD
//
//DYNPRG3B JOB (CF,F081),711153015,NOTIFY=F9G7LB,
// TIME=4
/*ROUTE  PRINT REMOTE2
//DYNPRG EXEC FORTH,LIB4='IMSL.SINGLE'
//FORT.SYSPRINT DD DUMMY
//FORT.SYSIN DD DSN=F9G7LB.DYNPRG.FORT,DISP=SHR
// DD DSN=F9G7LB.MODEL3.FORT,DISP=SHR
//LKED.SYSPRINT DD DUMMY
//GO.FT01F001 DD DSN=F9G7LB.MODEL3B.DATA,DISP=SHR
//GO.SYSIN DD *
0.948434,0.928751
0.01,0.01
0.74,0.74
1.0,1.0
0.02,0.02
1.15,1.105,1.08
5,250,0,01,0.01
T,F,F
/*
//GO.FT10F001 DD DSN=F9G7LB.DYN3B.DATA,DISP=OLD
//
//DYNPRG4 JOB (CF,F081),711153015,NOTIFY=F9G7LB,
// TIME=9
/*ROUTE  PRINT REMOTE2
//LIST EXEC TSOLIST,NAME='F9G7LB.MODEL4.DATA'
//DYNPRG EXEC FORTH,LIB4='IMSL.SINGLE'
//FORT.SYSPRINT DD DUMMY
//FORT.SYSIN DD DSN=F9G7LB.DYNPRG.FORT,DISP=SHR
// DD DSN=F9G7LB.MODEL4.FORT,DISP=SHR
//LKED.SYSPRINT DD DUMMY
//GO.FT01F001 DD DSN=F9G7LB.MODEL4.DATA,DISP=SHR
//GO.SYSIN DD *
0.948434,0.928751
0.01,0.01
0.74,0.87
1.0,1.13
0.02,0.02
1.15,1.105,1.08


1,250,0.01,0.01
T,F,F
/*
//GO.FT10F001 DD DSN=F9G7LB.DYN4.DATA,DISP=OLD
//
```

```
THE FOLLOWING VARIABLES ARE USED IN THIS PROGRAM:

MDL           MODEL NUMBER (ALPHANUMERIC)
NX            STATE SPACE DIMENSION
IXTAR         TARGET POINT INDEX
IWS,IWE       STATE SPACE WINDOW START AND END INDICES
NNX           NUMBER OF STATE SPACE POINTS EXCEPTING THE TARGET
XTAR          TARGET POINT VALUES
XINC          STATE SPACE INCREMENT
IX            STATE SPACE INDEX
XS            STATE SPACE VALUES
XMIN          STATE SPACE MINIMA
XMAX          STATE SPACE MAXIMA
UMIN          CONTROL SPACE MINIMA
UMAX          CONTROL SPACE MAXIMA
UINC          CONTROL SPACE INCREMENT
NU1,NU2       CONTROL SPACE DIMENSIONS
IU1,IU2       CONTROL SPACE INDICES
YMAX          OUTPUT CONSTRAINT MAXIMA
NT            MAXIMUM ITERATIONS
NTS           MAXIMIUM SUBITERATIONS
DT            TIME INCREMENT
DTI           INTEGRATION TIME INCREMENT
IT            ITERATION NUMBER
VOPT          COST ESTIMATES
UOPT          OPTIMAL CONTROLS
PRIT          ITERATION PRINT CONTROL (LOGICAL)
PRITS         SUBITERATION PRINT CONTROL (LOGICAL)
RUN           RUN CONTROL (LOGICAL)
IX1,IX2       STATE SPACE INDICES
NCHNG         UPDATE COUNTER
XIN           ALLOWABLE VALUE INDICATOR (LOGICAL)
XINS          ARRAY OF SAVED XIN'S
XOPT          ARRAY OF SAVED RESPONSES
ITS           SUBITERATION NUMBER
NCHNGS        UPDATE COUNTER

REAL UMIN(2),UMAX(2),UINC(2),US(21,2)
REAL X(2),XN(5),U(2)
LOGICAL XIN,PRIT,PRITS,RUN,XINS(15,15)
COMMON DT,DTI,IX1,IX2,NX,XIN,IXTAR
COMMON XMIN(2),XMAX(2),XINC(2),XTAR(2),XS(15,2),YMAX(3)
COMMON VOPT(15,15),UOPT(15,15,2),XOPT(15,15,2)
CALL INIT(MDL)
NX=15
IXTAR=(NX+1)/2
IWS=IXTAR-5
IWE=IXTAR+5
NNX=NX**2-1
```

```
   DEFINE THE ELEMENTS AND DIMENSIONS
   OF THE STATE SPACE

   READ(5,*) XTAR
   WRITE(6,220) XTAR
   READ(5,*) XINC
   WRITE(6,221) XINC
   DO 10 IX=1,NX
   XS(IX,1)=XTAR(1)+XINC(1)*FLOAT(IX-IXTAR)
   XS(IX,2)=XTAR(2)+XINC(2)*FLOAT(IX-IXTAR)
10 CONTINUE
   XMIN(1)=XS(1,1)
   XMIN(2)=XS(1,2)
   XMAX(1)=XS(NX,1)
   XMAX(2)=XS(NX,2)

   DEFINE THE ELEMENTS AND DIMENSIONS
   OF THE CONTROL SPACE

   READ(5,*) UMIN
   WRITE(6,222) UMIN
   READ(5,*) UMAX
   WRITE(6,223) UMAX
   READ(5,*) UINC
   WRITE(6,224) UINC
   NU1=IFIX((UMAX(1)-UMIN(1))/UINC(1)+0.5)+1
   DO 20 IU1=1,NU1
   US(IU1,1)=UMIN(1)+UINC(1)*FLOAT(IU1-1)
20 CONTINUE
   NU2=IFIX((UMAX(2)-UMIN(2))/UINC(2)+0.5)+1
   DO 25 IU2=1,NU2
   US(IU2,2)=UMIN(2)+UINC(2)*FLOAT(IU2-1)
25 CONTINUE

   INPUT OTHER CONTROL PARAMETERSAND
   INITIAL COST ESTIMATES

   READ(5,*) YMAX
   READ(5,*) NT,NTS,DT,DTI
   READ(10) IT,VOPT,UOPT
   WRITE(6,225) NT,NTS,DT,DTI,IT
   REWIND 10
   READ(5,*) PRIT,PRITS,RUN
   VOPT(IXTAR,IXTAR)=0.0
   IF(.NOT.RUN) GO TO 90

   SCAN THE STATE AND CONTROL SPACES,
   EVALUATING THE COST FOR EACH SET OF
   CONTROLS.  REPLACE THE CONTROLS IF
   THE COST IS LESS THAN THE
   INITIAL ESTIMATE.

   DO 70 JT=1,NT
   IT=IT+1
   IX1=IXTAR
   IX2=IXTAR
   NCHNG=0
   DO 60 IX=1,NNX
```

```
      CALL SPIRAL
      X(1)=XS(IX1,1)
      X(2)=XS(IX2,2)
      XINS(IX1,IX2)=.FALSE.
      DO 50 IU1=1,NU1
      DO 50 IU2=1,NU2
      U(1)=US(IU1,1)
      U(2)=US(IU2,2)
      CALL NEXTX(X,U,XN)
      IF(.NOT.XIN)GO TO 50
      VTST=V(XN)
      IF(VTST.GE.VOPT(IX1,IX2)) GO TO 50
      NCHNG=NCHNG+1
      VOPT(IX1,IX2)=VTST
      UOPT(IX1,IX2,1)=U(1)
      UOPT(IX1,IX2,2)=U(2)
      XOPT(IX1,IX2,1)=XN(1)
      XOPT(IX1,IX2,2)=XN(2)
      XINS(IX1,IX2)=XIN
   50 CONTINUE
   60 CONTINUE
      IF(PRIT) WRITE(6,210) IT,NCHNG
      IF(NCHNG.EQ.0) IT=IT-1
      IF(NCHNG.EQ.0) GO TO 80

      SCAN THE STATE SPACE ONLY,
      REEVALUATING THE COST FOR
      A FIXED CONTROL LAW,
      ITERATE UNTIL THE TRUE COST
      FOR THIS LAW IS FOUND.

      DO 40 ITS=1,NTS
      NCHNGS=0
      IX1=IXTAR
      IX2=IXTAR
      DO 30 IX=1,NNX
      CALL SPIRAL
      IF(.NOT.XINS(IX1,IX2)) GO TO 30
      XN(1)=XOPT(IX1,IX2,1)
      XN(2)=XOPT(IX1,IX2,2)
      VTST=V(XN)
      IF(VTST.GE.VOPT(IX1,IX2)) GO TO 30
      NCHNGS=NCHNGS+1
      VOPT(IX1,IX2)=VTST
   30 CONTINUE
      IF(PRITS) WRITE(6,211) ITS,NCHNGS
      IF(NCHNGS.EQ.0) GO TO 45
   40 CONTINUE
   45 CONTINUE
   70 CONTINUE
   80 CONTINUE

      RECORD AND REPORT THE RESULTS

      WRITE(10) IT,VOPT,UOPT
      WRITE(10) XMIN,XMAX,XINC,XTAR
   90 CONTINUE
      WRITE(6,200)
      WRITE(6,201) (XS(IX1,1),IX1=IWS,IWE)
      DO 100 IX=IWS,IWE
      IX2=NX-IX+1
      WRITE(6,202) XS(IX2,2),(VOPT(IX1,IX2),IX1=IWS,IWE)
  100 CONTINUE
      WRITE(6,205) MDL
      IF(YMAX(1)*YMAX(2)*YMAX(3).EQ.0.0) WRITE(6,207)
      IF(YMAX(1)*YMAX(2)*YMAX(3).NE.0.0) WRITE(6,208)
      WRITE(6,200)
      WRITE(6,201) (XS(IX1,1),IX1=IWS,IWE)
      DO 110 IX=IWS,IWE
      IX2=NX-IX+1
      WRITE(6,203) XS(IX2,2),(UOPT(IX1,IX2,1),IX1=IWS,IWE)
      WRITE(6,204) (UOPT(IX1,IX2,2),IX1=IWS,IWE)
  110 CONTINUE
      WRITE(6,206) MDL
      IF(YMAX(1)*YMAX(2)*YMAX(3).EQ.0.0) WRITE(6,207)
      IF(YMAX(1)*YMAX(2)*YMAX(3).NE.0.0) WRITE(6,208)
      WRITE(6,200)
      WRITE(6,201) (XS(IX1,1),IX1=IWS,IWE)
      DO 120 IX=IWS,IWE
      IX2=NX-IX+1
      WRITE(6,212) XS(IX2,2),(XOPT(IX1,IX2,1),IX1=IWS,IWE)
      WRITE(6,213) (XOPT(IX1,IX2,2),IX1=IWS,IWE)
  120 CONTINUE
      STOP
  200 FORMAT('1'///////////////)
  201 FORMAT(1X,T21,11F8.4///)
  202 FORMAT(/8X,F8.4,T21,11F8.4/)
  203 FORMAT(/8X,F8.4,T20,11F8.2)
  204 FORMAT(1X,T20,11F8.2)
  205 FORMAT(///1X,T20,'FIGURE      -A  MODEL ',A4,T65,'COST')
  206 FORMAT(///1X,T20,'FIGURE      -B  MODEL ',A4,T65,
     &'OPTIMAL CONTROL LAW')
  207 FORMAT('+',T50,'UNCONSTRAINED')
  208 FORMAT('+',T50,'CONSTRAINED')
  210 FORMAT(1X,T50,'ITERATION =',I4,', NUMBER OF UPDATES =',I5)
  211 FORMAT(1X,'PREITERATION =',I4,', NUMBER OF UPDATES =',I5)
  212 FORMAT(/8X,F8.4,T21,11F8.4)
  213 FORMAT(1X,T21,11F8.4)
  220 FORMAT(1X,'XTAR=(',F8.4,',',F8.4,')')
  221 FORMAT(1X,'XINC=(',F8.4,',',F8.4,')')
  222 FORMAT(1X,'UMIN=(',F8.4,',',F8.4,')')
  223 FORMAT(1X,'UMAX=(',F8.4,',',F8.4,')')
  224 FORMAT(1X,'UINC=(',F8.4,',',F8.4,')')
  225 FORMAT(1X,'NT=',I3/1X,'NTS=',I4/1X,'DT=',E11.4/
     &1X,'DTI=',E11.4/1X,'IT=',I3/)
      END


      SUBROUTINE SPIRAL

      THIS ROUTINE UPDATES THE
      STATE SPACE INDICES,SPIRALLING
      OUTWARD FORM THE TARGET.

      COMMON DT,DTI,IX1,IX2,NX,XIN,IXTAR
      COMMON XMIN(2),XMAX(2),XINC(2),XTAR(2),XS(15,2),YMAX(3)
      COMMON VOPT(15,15),UOPT(15,15,2),XOPT(15,15,2)
      IF(IX1+IX2.GE.2*IXTAR.AND.IX1.LT.IX2) GO TO 10
      IF(IX1+IX2.GE.2*IXTAR.AND.IX1.GE.IX2) GO TO 20
      IF(IX1+IX2.LT.2*IXTAR.AND.IX1.GT.IX2) GO TO 30
      IF(IX1+IX2.LT.2*IXTAR.AND.IX1.LE.IX2) GO TO 40
      STOP 16
   10 IX1=IX1+1
      RETURN
   20 IX2=IX2-1
      RETURN
   30 IX1=IX1-1
      RETURN
   40 IX2=IX2+1
      RETURN
      END
      REAL FUNCTION V(XN)

      THIS ROUTINE EVALUATES THE COST
      FUNCTION AT ANYPOINT IN THE
      STATE SPACE, BY INTERPOLATING
      FROM NEIGHBORING GRID VALUES.

      REAL XN(5)
      LOGICAL XIN
      COMMON DT,DTI,IX1,IX2,NX,XIN,IXIAR
      COMMON XMIN(2),XMAX(2),XINC(2),XTAR(2),XS(15,2),YMAX(3)
      COMMON VOPT(15,15),UOPT(15,15,2),XOPT(15,15,2)
      FR1H=(XN(1)-XMIN(1))/XINC(1)+1.0
      IX1L=IFIX(FR1H)
      FR1H=FR1H-FLOAT(IX1L)
      IX1H=IX1L+1
      FR1L=1.0-FR1H
      FR2H=(XN(2)-XMIN(2))/XINC(2)+1.0
      IX2L=IFIX(FR2H)
      FR2H=FR2H-FLOAT(IX2L)
      IX2H=IX2L+1
      FR2L=1.0-FR2H
      V=DT+FR1L*FR2L*VOPT(IX1L,IX2L)
      IF(IX1H.GT.NX) GO TO 20
      V=V+FR1H*FR2L*VOPT(IX1H,IX2L)
   20 CONTINUE
      IF(IX2H.GT.NX) GO TO 30
      V=V+FR1L*FR2H*VOPT(IX1L,IX2H)
   30 CONTINUE
      IF(IX1H.GT.NX.OR.IX2H.GT.NX) GO TO 40
      V=V+FR1H*FR2H*VOPT(IX1H,IX2H)
   40 CONTINUE
      RETURN
      END




      SUBROUTINE NEXTX(X,U,XN)

      THIS ROUTINE PERFORMS A VARIABLE
      STEP EULER INTEGRATION, AND
      TESTS THE NEW STATE AGAINST ALL CONSTRAINTS.

      REAL X(2),U(2),XN(5),DX(5),Y(3)
      LOGICAL XIN
      COMMON DT,DTI,IX1,IX2,NX,XIN,IXIAR
      COMMON XMIN(2),XMAX(2),XINC(2),XTAR(2),XS(15,2),YMAX(3)
      COMMON VOPT(15,15),UOPT(15,15,2),XOPT(15,15,2)
      DO 10 J=1,2
      XN(J)=X(J)
   10 CONTINUE
      CALL COMPLT(XN,U)
      NI=IFIX(DT/DTI+0.5)
      DO 20 I=1,NI
      CALL XDOT(XN,U,DX)
      DO 20 J=1,5
      XN(J)=XN(J)+DTI*DX(J)
   20 CONTINUE
      XIN=.TRUE.
      IF(XN(1).LT.XMIN(1)) XIN=.FALSE.
      IF(XN(2).LT.XMIN(2)) XIN=.FALSE.
      IF(XN(1).GT.XMAX(1)) XIN=.FALSE.
      IF(XN(2).GT.XMAX(2)) XIN=.FALSE.
      CALL OUTPUT(X,U,Y)
      IF(Y(1).GT.YMAX(1).AND.YMAX(1).GT.0.0) XIN=.FALSE.
      IF(Y(2).GT.YMAX(2).AND.YMAX(2).GT.0.0) XIN=.FALSE.
      IF(Y(3).GT.YMAX(3).AND.YMAX(3).GT.0.0) XIN=.FALSE.
      RETURN
      END
      SUBROUTINE OUTPUT(X,U,Y)
C
C     THIS ROUTINE, COMMON TO ALL MODELS,
C     EVALUATES OUTPUTS FOR CONSTRAINT COMPARISON.
C
      REAL X(2),U(2),Y(3)
      REAL C(3,2),D(3,2),E(3),WK(3)
      DATA C/-0.61059,-0.20154,-0.58229,-0.10759,-0.45813,0.46872/
      DATA D/0.50292,0.20423,0.18877,0.17689,0.14724,-0.92545/
      DATA E/1.03837,1.30796,1.85021/
      CALL VMULFF(C,X,3,2,1,3,2,Y,3,IER)
      CALL VMULFF(D,U,3,2,1,3,2,WK,3,IER)
      DO 10 I=1,3
      Y(I)=Y(I)+WK(I)+E(I)
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE COMPLT(XN,U)
C
C     THIS ROUTINE, COMMON TO ALL MODELS,
C     EVALUATES MISSING STATES AS A
C     FUNCTION OF STATE AND CONTROLS.
C
      REAL XN(5),U(2)
      XN(3)=1.0+1.4663*(XN(1)-1.0)+0.53032*(XN(2)-1.0)
     &+0.40998*(U(1)-1.0)-0.18155*(U(2)-1.0)
      XN(4)=1.0-0.15936*(XN(1)-1.0)+0.78107*(XN(2)-1.0)
     &+0.43393*(U(1)-1.0)-0.88875*(U(2)-1.0)
      XN(5)=1.0-0.63063*(XN(1)-1.0)-0.99071*(XN(2)-1.0)
     &+1.0656*(U(1)-1.0)+0.17300*(U(2)-1.0)
      RETURN
      END
```