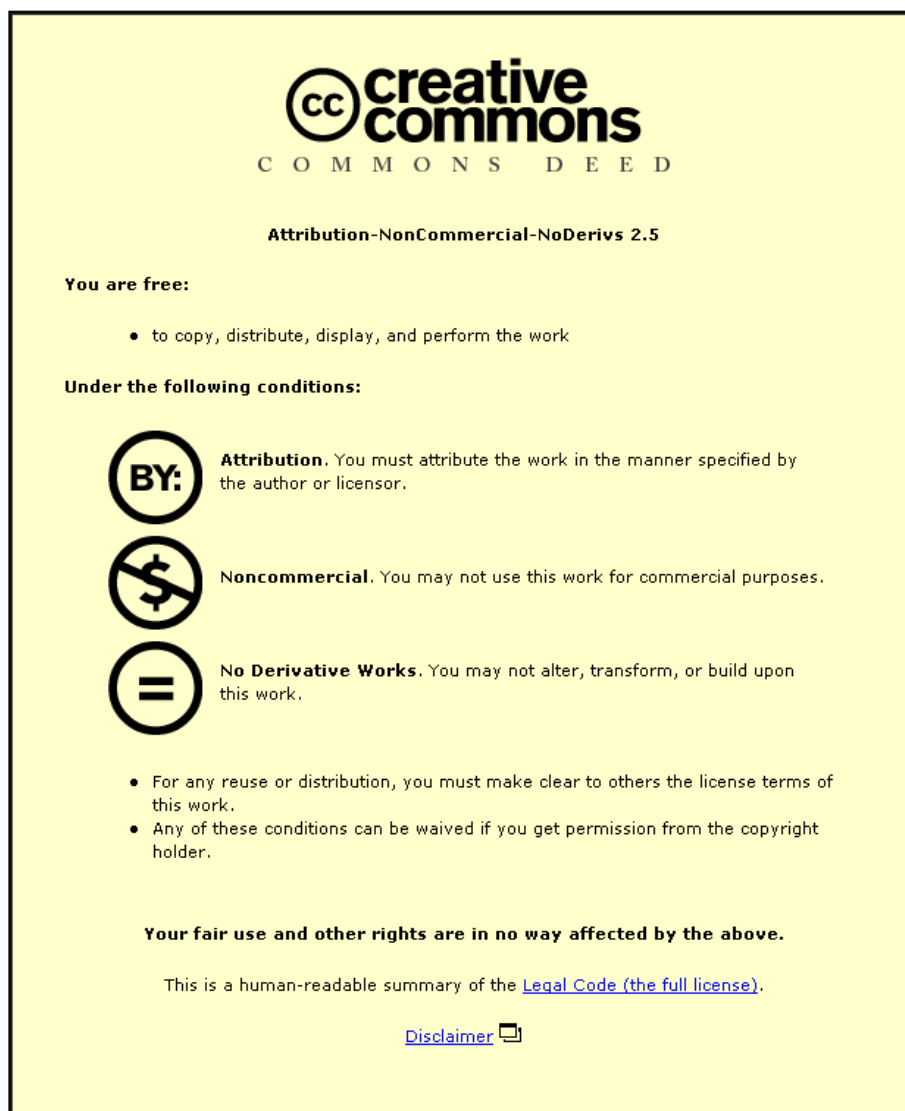


This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (<https://dspace.lboro.ac.uk/>) under the following Creative Commons Licence conditions.



For the full text of this licence, please go to:
<http://creativecommons.org/licenses/by-nc-nd/2.5/>

**LOUGHBOROUGH
UNIVERSITY OF TECHNOLOGY
LIBRARY**

AUTHOR/FILING TITLE

MOHAMMAD, S.A.

ACCESSION/COPY NO.

040129482

VOL. NO.

CLASS MARK

Loan copy

040129482X



BADMINTON PRESS
18 THE HALFCROFT
SYSTEM
LEICESTER LE7 4LD
ENGLAND
TEL 0116 260 2917
FAX 0116 269 6639

A Numerical Study of some Hybrid Conjugate Gradient Methods in Optimal Control

by

S A Mohammadi


A Doctoral Thesis

Submitted in Partial Fulfilment of the Requirements

For the Award of Doctor of Philosophy
of Loughborough University of Technology

November 1995.

Supervisors: Professor Colin Storey, B.Sc., D.Sc., C. Math., F.I.M.A.
Professor Roger Smith, B.Sc., Ph.D., C.Phys., C.Eng.
Department of Mathematical Sciences.

 Loughborough University Library	
Date	Aug 96
Class	
Acc No.	040129482

9/6446694

Contents

1	INTRODUCTION	1
1.1	Introductory	1
1.2	Introduction to Optimal Control	1
1.2.1	The General Optimal Control Problem	1
1.2.2	The First Variation of J	2
1.2.3	The second variation of J	3
1.2.4	Singular Control Problems	3
1.2.5	Necessary Conditions for Singular Optimal Control	4
1.2.6	Sufficient Conditions and Necessary and Sufficient Conditions for Optimality	5
1.2.7	Outline of Chapters	6
1.2.8	The Problems Considered	6
1.2.9	Some Notations Used Throughout the Thesis	7
2	CONJUGATE GRADIENT AND HYBRID CONJUGATE GRADIENT METHODS	8
2.1	Finite dimensional problems	8
2.1.1	Types of problems	9
2.2	Descent Methods and their Convergence Properties	10
2.2.1	Some theorems on convergence of descent methods	11
2.2.2	Steepest descent method	14
2.3	Conjugate Gradient Methods and their Convergence Properties	15
2.3.1	A study of quadratic functions and their properties	19
2.3.2	Conjugacy of quadratic functions	21
2.3.3	The convergence properties of conjugate gradient methods on quadratic functions	24
2.3.4	Extension of conjugate gradient methods to general functions	24
2.4	Fletcher Reeves Method and its Convergence Properties	25
2.4.1	Descent property and global convergence in the case of exact line search	26
2.4.2	Descent property and global convergence in the case of inexact line search	27
2.5	The Polak-Ribière Method and its Global Convergence	27
2.5.1	Global convergence	28
2.6	Hybrid Conjugate Gradient Methods	28
2.6.1	Hybrid 1 algorithm	29
2.6.2	Descent property and global convergence in the case of exact line search	29

2.6.3	Descent property and global convergence in the case of inexact line search	30
2.7	Angle Test Hybrid	30
2.8	Hybrid 3	31
2.8.1	Descent property and global convergence	32
2.9	Infinite Dimensional Problems	32
2.9.1	Theoretical methods	33
2.10	The Conjugate Gradient Method for Optimal Control Problems	33
2.10.1	The algorithm for conjugate gradient method	34
2.10.2	Convergence	36
3	NUMERICAL CONSIDERATION	38
3.1	Numerical Computations	38
3.2	Numerical Integration Methods	38
3.2.1	Runge Kutta Methods	38
3.2.2	Order and Convergence of the General Explicit One- step Method	39
3.2.3	Algorithms for Runge-Kutta Methods	41
3.2.4	Error Bounds for Runge-Kutta Methods.	44
3.2.5	Estimation of Error for Runge-Kutta Methods	48
3.2.6	Weak Stability for Runge-Kutta Methods	51
3.3	Simpson's Numerical Integration Rule	54
3.3.1	Accuracy	55
3.4	The Linear Search	56
3.4.1	Linear Search at Constant Step	60
3.4.2	Quadratic Interpolation Method	62
4	PROBLEM 1	66
4.1	A Simple Control Problem	66
4.2	Analytical Solution	66
4.3	Numerical Solution	68
4.3.1	The State and Adjoint Equations	68
4.4	Results and Discussion	69
4.4.1	Gradient Method	69
4.4.2	Steepest Descent	71
4.4.3	Fletcher-Reeves	73
4.4.4	Polak-Ribière	74
4.4.5	Hybrid 1	77
4.4.6	Angle Test Hybrid	78
4.4.7	Hybrid 3	79
4.5	Summary of the Results	81
4.6	Conclusion	81
5	PROBLEM 2	83
5.1	Minimum Drag (MD) Problem	83
5.2	Analytical Solution	84
5.3	Numerical Solutions	89
5.3.1	The state and adjoint equations	89
5.4	Results and Discussions	91
5.4.1	Gradient method	91

8.3.5	Hybrid 1	145
8.3.6	Angle test hybrid	146
8.3.7	Hybrid 3	148
8.4	Summary of the Results	149
8.5	Conclusion	149
9	PROBLEM 6	151
9.1	A Nonlinear Singular Control Problem	151
9.2	Numerical Solutions	153
9.2.1	The state and adjoint equations	153
9.3	Results and Discussion	157
9.3.1	Gradient method	157
9.3.2	Steepest descent	159
9.3.3	Fletcher-Reeves	160
9.3.4	Polak-Ribière	161
9.3.5	Hybrid 1	163
9.3.6	Angle test hybrid	164
9.3.7	Hybrid 3	165
9.4	Summary of the Results	166
9.5	Conclusion	167
10	CONCLUSION	168

Acknowledgements

I would like to express my very sincere thanks to Professor C Storey, for his most valuable supervision, guidance and comments, throughout this work.

I would also like to thank Dr A C Pugh, my Research Director and Professor G A Evans and Professor R Smith, for their valuable help and assistance during my study.

I express my deep gratitude to my Wife and my Parents, for their great moral support, patience and understanding during my research.

Finally, I wish to thank Louise, Nicci and Helen, for helping me with their excellent typing.

Key words:

Numerical Optimal Control, comparison of Conjugate Gradient methods,
Penalty functions.

Abstract

The main work of this thesis is concerned with the comparison of conjugate gradient with hybrid conjugate gradient methods when they are applied to optimal control problems. Descriptions of the conjugate gradient and hybrid conjugate gradient methods, for general optimisation, in finite and infinite dimensions are also given. The numerical methods for solving the differential equations and the line searches required in the optimisation are discussed next.

We have applied seven different numerical techniques to six problems.

1. Gradient in function space.
2. Steepest descent.
3. Fletcher Reeves.
4. Polak-Ribière.
5. Hybrid1.
6. Angle test Hybrid.
7. Hybrid3.

The problems considered are as follows:

1. A simple reactor control problem.
2. An unconstrained problem called minimum drag (MD).
3. A discontinuous optimal control problem consisting of a singular arc followed by a nonsingular arc.
4. An optimal control problem with fixed end points.
5. A nonlinear continuous stirred-tank reactor.
6. A nonlinear singular control problem.

The methods are compared from the point of view of accuracy of the cost function, convergence rate and finally the computing time taken.

In all the computational exercises attempted in this work the programming is done in the Fortran language.

The first problem is run on the Prime b operating system, and the other five problems are run on Hewlett Packard (HP-UX) system.

In the final chapter a critical comparison of the hybrid methods against the conventional methods is undertaken.

Chapter 1

INTRODUCTION

1.1 Introductory

Nowadays as computing facilities are improved greatly, we can compare different techniques of optimization more effectively, but nevertheless since there are several techniques for solving optimal control problems, it may not be easy to choose the best out of all and it seems that each single method has some advantages over the others. However certain types of problems may perhaps best be tackled by one specific technique compared with any other, and as computational experience increases it may well be possible to type cast problems in this way.

1.2 Introduction to Optimal Control

1.2.1 The General Optimal Control Problem

It is well known that the fundamental problem of optimal control theory can be formulated as a problem of Bolza, Mayer or Lagrange. These three formulations are quite equivalent to one another (Bliss, 1946) [1].

The problem of Bolza in optimal control theory is the following. Determine the control function $u(\cdot)$ which minimizes the cost functional

$$J = F[x(t_f), t_f] + \int_{t_0}^{t_f} L(x, u, t) dt \quad (1.2.1)$$

where the system equation is

$$\dot{x} = f(x, u, t) \quad (1.2.2)$$

subject to the constraints

$$x(t_0) = x_0 \quad (1.2.3)$$

$$\Psi[x(t_f), t_f] = 0 \quad (1.2.4)$$

$u(\cdot)$ is a member of the set U , t a member of the interval

$$[t_0, t_f] \quad (1.2.5)$$

Here x is an n -dimensional state vector and u is an m -dimensional control vector. The function L and F are scalar and the terminal constraint function Ψ is an s -dimensional column vector function of $x(t_f)$ at t_f .

The functions L , F and Ψ are assumed smooth. The set U is defined by

$$U \equiv \{u(\cdot); u_i(\cdot) \text{ is piecewise continuous in time,}$$

$$|u_i(t)| < \infty, \quad t_0 \leq t \leq t_f, \\ i = 1, 2, \dots, m\}. \quad (1.2.6)$$

The initial time t_0 is given explicitly but the final time t_f may be unspecified.

The n -dimensional vector function f of equation (1.2.2) and the scalar functions F and L are assumed to be at least twice continuously differentiable in each argument.

1.2.2 The First Variation of J

Introducing the Hamiltonian function for the problem of Bolza in optimal control theory as

$$H(x, u, \lambda, t) = L(x, u, t) + \lambda^T f(x, u, t) \quad (1.2.7)$$

The following necessary conditions (Pontryagin's principle) can be shown to hold along an optimal trajectory (refer to Bell & Jacobson (1975) [132]):

$$-\dot{\lambda} = H_x(\bar{x}, \bar{u}, \lambda, t) \quad (1.2.8)$$

$$\lambda(t_f) = F_x[\bar{x}(t_f), t_f] + \Psi_x^T \nu \quad (1.2.9)$$

$$H(t_f) = -F_t[\bar{x}(t_f), t_f] - \Psi_t^T \nu \quad (1.2.10)$$

where

$$\bar{u} = \arg \min_u H(\bar{x}, u, \lambda, t) \quad (1.2.11)$$

$u(\cdot)$ a member of U .

Here $\bar{x}(\cdot)$, $\bar{u}(\cdot)$ denote the candidate state and control functions respectively, $\lambda(\cdot)$ denotes an n -dimensional vector of Lagrange multiplier functions of time, and ν is an s -dimensional vector of Lagrange multipliers associated with Ψ .

1.2.3 The second variation of J

A further necessary condition for a minimizing arc is known in the classical literature as the Jacobi condition. The Jacobi condition may be stated as follows (Bryson and Ho, 1969) [2]: an optimal trajectory contains no conjugate point between its end points. This will be the case if the matrix S remains finite for $t_0 \leq t \leq t_f$ where S satisfies the matrix differential equation

$$-\dot{S} = H_{xx} + S f_x + f_x^T S - (H_{xu} + S f_u) H_{uu}^{-1} (H_{ux} + f_u^T S) \quad (1.2.12)$$

at end condition

$$S(t_f) = \Phi_{xx}[x(t_f), t_f]. \quad (1.2.13)$$

where $\Phi(x(t_f), t_f) = F + \nu^T \Psi$.

1.2.4 Singular Control Problems

The class of problems to which this analysis applies is the following: Determine the control $u^*(t)$, $t \in [t_0, t_f]$, which minimizes the functional

$$J(u) = F(x(t_1), t_f) + \int_{t_0}^{t_f} [L_0(t, x) + L_u(t, x)u] dt \quad (1.2.14)$$

where the system equation is

$$\dot{x} = f_0(t, x) + f_u(t, x)u \quad (1.2.15)$$

subject to the constraints

$$|u(t)| \leq k(t) \in [t_0, t_f], \quad (1.2.16)$$

$$\{t_0, x(t_0), t_f, x(t_f)\} \in S. \quad (1.2.17)$$

Here x is an n -vector and S is a closed subset of R^{2n+2} . The functions f_0 , f_u , L_0 , L_u are assumed to be analytic in both arguments in a suitable domain $k(t)$ is assumed to be analytic in a neighbourhood of each junction and $|u(t)| < k(t)$ almost everywhere on the singular subarcs. Of course, the usual case $|u| \leq k$ with $k = \text{const}$ is included as a special case. We restrict attention to a scalar control in order to simplify notation.

A similar analysis holds for each component of a vector control. Clearly, the Hamiltonian for this problem is linear in the control, i.e.

$$H(t, x, \lambda, u) = \lambda^T f_0(t, x) + L_0(t, x) + [\lambda^T f_u(t, x) + L_u(t, x)]u. \quad (1.2.18)$$

The multiplier equations are given by;

$$\dot{\lambda} = -H_x(t, x, \lambda, u) \quad (1.2.19)$$

where H_x is linear in u . The coefficient of u in (1.2.18) is called the switching function, which we shall designate as $\phi(t)$, i.e.

$$\phi(t) = H_u(t, x(t), \lambda(t)). \quad (1.2.20)$$

The minimum principle (i.e. Pontryagin's maximum principle in a minimum form) states that for almost every $t \in [t_0, t_f]$ and each u satisfying $|u(t)| \leq k(t)$, the optimal control $u^*(t)$ must satisfy;

$$H(t, x(t), \lambda(t), u^*(t)) \leq H(t, x(t), \lambda(t), u(t)). \quad (1.2.21)$$

Therefore, as is well known, on each open subinterval of $[t_0, t_f]$ there are two distinct possibilities for u^* .

Either

$$u^*(t) = -k(t) \operatorname{sgn} \phi(t) \quad (1.2.22)$$

or

$$\phi(t) = 0 \quad (1.2.23)$$

Equations (1.2.22) and (1.2.23) define, respectively the nonsingular and singular subarcs of the optimal control. The class of problems defined above will be called singular control problems, even though only a portion of the total trajectory may be singular.

1.2.5 Necessary Conditions for Singular Optimal Control

The following definition will clearly clarify the terminology used in this section.

Definition 1. Let u be an optimal singular control on the interval $[t_1, t_2]$, and let $(d^{2q}/dt^{2q})[H_u(\bar{x}, \lambda, t)]$ be the lowest order total derivative of H_u in which u appears explicitly with a coefficient which is not identically zero on $[t_1, t_2]$. Then the integer q is called the order of the singular arc.

Implicit in this definition is the property that u first appears explicitly in an even order derivative of H_u ; i.e. it is correct to refer to q as an integer. For a proof of this property see Robbins (1967) [3].

When the control is a scalar, Kelley (1964) [4] deduced a new necessary condition for a singular optimal control by studying the second variation under a special control variation. Kelley's method was generalized by Tait (1965) [5], Kopp and Moyer (1965) [6], Kelley et al. (1967) [7] to give what has since become known as the generalized Legendre-Clebsch condition (Kelley-Contensou test):

$$(-1)^q \frac{\partial}{\partial u} \frac{d^{2q}}{dt^{2q}} H_u(\bar{x}, \lambda, t) \geq 0 \quad (1.2.24)$$

where the integer q is the order of the singular problem.

The generalized Legendre-Clebsch condition for a vector control was obtained by Robbins (1967) [8] and Goh (1966) [9]. In this case the controls can appear in an odd time derivative of H_u but if this does occur then there necessarily exists a control u in U such that the second variation is negative (for a minimization problem). Hence, the generalized Legendre-Clebsch condition for vector control is

$$\frac{\partial}{\partial u} \frac{d^p}{dt^p} H_u = 0 \quad \text{for all } t \text{ in } [t_0, t_f] \quad p \text{ odd} \quad (1.2.25)$$

and

$$(-1)^q \frac{\partial}{\partial u} \frac{d^{2q}}{dt^{2q}} H_u \geq 0 \quad \text{for all } t \text{ in } [t_0, t_f]. \quad (1.2.26)$$

1.2.6 Sufficient Conditions and Necessary and Sufficient Conditions for Optimality

By establishing the generalized Legendre-Clebsch condition Research began into finding a generalized sufficient conditions for the nonsingular problem. Before such a generalization was found, a new necessary condition for non-negativity of the singular second variation, not equivalent to the generalized Legendre-Clebsch condition was derived (Jacobson, (1969, 1970) [10]. This set of new conditions known as Jacobson's condition are as follows:

$$H_{ux}f_u + f_u^T Q f_u \geq 0 \quad (1.2.27)$$

where

$$-\dot{Q} = H_{xx} + f_x^T Q + Q f_x \quad (1.2.28)$$

$$Q(t_f) = F_{xx}[\bar{x}(t_f), t_f] \quad (1.2.29)$$

assuming that the matrices H_{xx} , H_{ux} , f_x and f_u are sufficiently differentiable with respect to time, and the partial derivatives f_u , H_{ux} and f_x are all evaluated along the singular arc $\bar{x}(\cdot)$, $\bar{u}(\cdot)$. A strong version of this condition is given by Mayne (1973) [11].

It has been shown by Jacobson (1970) [12] that in general the generalized Legendre-Clebsch condition and the Jacobson condition together are not sufficient for optimality. A sufficient condition for a weak local minimum in a non-singular problem is that the second variation be strongly positive (Gelfand and Fomin, (1963) [13]. This condition gives rise to a well known matrix Riccati differential equation. In singular problems the second variation can not be strongly positive (Tait, 1965) [4], (Johnson, 1966) [14] but investigations have shown that Riccati-type conditions do exist for the singular case. Jacobson (1970) [10], using a direct approach, derived sufficient conditions for the second variation to be non-negative in both singular and nonsingular control problems. These conditions are that there exist a real symmetric bounded, matrix function of time $P(\cdot)$ such that

$$H_{ux} + f_u^T P = 0 \quad \text{for } t \in [t_0, t_f] \quad (1.2.30)$$

$$\dot{P} + H_{xx} + f_x^T P + P f_x \geq 0 \quad \text{for } t \in [t_0, t_f] \quad (1.2.31)$$

and

$$z^T [F_{xx} + \nu^T \Psi_{xx} - P]_{t_f} z \geq 0 \quad (1.2.32)$$

where z is the $n \times (n - s)$ matrix

$$z = \left[\frac{-D_1^{-1} D_2}{I} \right] \quad (1.2.33)$$

and the $s \times s$ matrix D_1 and the $s \times (n - s)$ matrix D_2 are such that

$$\Psi_x = (D_1 \quad D_2) \quad (1.2.34)$$

The above conditions in strengthened form are sufficient conditions for a weak relative minimum. It is demonstrated that the two necessary conditions for singular optimal control discussed in section 1.2.5 above, are implied by the new conditions for the case of totally singular control and unconstrained terminal states.

A general sufficiency theorem for non-negativity of a large class of second variations was presented by Jacobson (1971) [15] for the partially singular case. It is that there should exist for all $t \in [t_0, t_f]$ a continuously differentiable, symmetric, matrix function of time $P(\cdot)$ such that

$$\begin{bmatrix} \dot{P} + H_{xx} + Pf_x + f_x^T P & H_{xu} + Pf_u \\ H_{ux} + f_u^T P & H_{uu} \end{bmatrix} \geq 0 \quad (1.2.35)$$

for $t \in [t_0, t_f]$, together with (1.2.32 – 34).

Jacobson then applied this condition to both the totally singular case and to the nonsingular case. Sufficient conditions were thus obtained for the two special cases, demonstrating that both singular and nonsingular second variations can be treated in a common general framework. The results developed by Jacobson and Speyer [16] through the transformation approach and the limit approach lead to the conclusion that the sufficiency conditions are also necessary for optimality for a large class of problems. In the nonsingular case the well known Riccati differential equation emerges and since this is known to be a necessary condition for the nonsingular second variation, this implies that the sufficiency conditions of the theorem are also necessary. In the singular case the algebraic and differential inequalities (1.2.30 – 32) in strengthened form are obtained and these have been proved necessary and sufficient conditions for optimality in the 1971 papers of Jacobson and Speyer [16]. Thus, in the singular case, the sufficiency conditions are also necessary.

1.2.7 Outline of Chapters

The main work of this thesis is contained in Chapters 4 to 9. Chapter 2 describes the conjugate gradient and Hybrid conjugate gradient techniques in general optimization in finite dimensions and infinite dimensions.

Chapter 3 deals with the numerical work put into solving the differential equations and line searches involving the problems. In Chapters 4 to 9 we tackle some practical problems by using conjugate gradient and Hybrid conjugate gradient methods, also at the end of each chapter the comparison of the methods and the advantages and disadvantages of each individual method over the other are discussed in details.

Finally, Chapter 10 is the conclusion.

1.2.8 The Problems Considered

The six problems considered in this work are outlined in Chapters 4 to 9.

In problem 1 a simple control function problem taken from Abdul Wahab Jusoh (1979) [17] is considered. The problem is first solved Analytically and then numerically using conjugate gradient and Hybrid conjugate gradient techniques developed by D. Touati Ahmed and C. Storey (1987) [18].

In problem 2 an unconstrained problem called Minimum Drag (MD) taken from S C Carg (1977) [19] is tackled numerically using conjugate gradient and Hybrid conjugate gradient methods.

In problem 3 a discontinuous optimal control problem consisting of a singular arc followed by a nonsingular arc taken from E R Edge and W F Powers [20] is solved analytically and also by conjugate gradient and Hybrid conjugate gradient techniques.

In problem 4 an optimal control problem with fixed end points is considered taken from Barnett and Cameron (1985) [144], and is solved analytically, and also numerically using conjugate gradient and Hybrid conjugate gradient techniques.

In Problem 5 a non linear continuous stirred-tank reactor taken from Rein Luss and Marco Galli (1990) [21] is considered and tackled by conjugate and Hybrid conjugate gradient methods.

Finally, in Problem 6 a non-linear singular control problem taken from Rein Luss (1990) [22] is solved using conjugate and Hybrid conjugate gradient techniques.

In all the computational exercises attempted in this work, the programming is done in the Fortran language.

The first problem is run on the operating system Prime b, and the other five problems are run on Hewlett Packard (HP-UX) system.

1.2.9 Some Notations Used Throughout the Thesis

For convenience we use the following abbreviations in this thesis:

- $A \equiv$ Analytical Solution.
- $ACC \equiv \|g\| \equiv$ ACCURACY.
- $ATH \equiv$ Angle test hybrid.
- $C_{ptime} \equiv$ Computing time in seconds.
- $e_{abs} \equiv$ Absolute error.
- $FR \equiv$ Fletcher-Reeves.
- $GFS \equiv$ Gradient in function space.
- $GLC \equiv$ Legendre Clebsch condition.
- $h \equiv$ Integration step.
- $H1 \equiv$ Hybrid 1.
- $H3 \equiv$ Hybrid 3.
- $K \equiv$ Coefficient of the penalty function.
- $m \equiv$ Number of iterations.
- $N \equiv$ Integration step number.
- $PR \equiv$ Polak-Ribière.
- $SD \equiv$ Steepest descent.
- $u_0 \equiv$ Initial control.
- $\varepsilon \equiv$ Step length factor.

Chapter 2

CONJUGATE GRADIENT AND HYBRID CONJUGATE GRADIENT METHODS

2.1 Finite dimensional problems

The basic problem here is to minimize a scalar function L of parameters π subject to constraints:

$$\min L(\pi), \pi \in P \subseteq \underline{E}^p \quad (2.1.1)$$

$$\text{with } f(\pi) = 0, f(\pi) \in \underline{E}^n \quad (2.1.2)$$

$$h(\pi) \leq 0, h(\pi) \in \underline{E}^q \quad (2.1.3)$$

where L is a single valued mapping from P , a subset of the p -dimensional real Euclidean space \underline{E}^p , to the real line \underline{E}' . In most cases conditions on continuity and differentiability of $L(\pi)$ are required. The functions f and h are vector-valued mappings from P to \underline{E}^n and \underline{E}^q respectively, where $n < p$ and the region R formed by the intersection of the constraints is assumed to be nonempty in order that a solution exists.

The equality constraints f mean that there are $m = p - n$ free parameters to minimize L subject to (2.1.3). In most methods, constraints f and h are also required to possess continuous derivatives to the first or second order.

These requirements will not be detailed further and will be tacitly assumed to be met, except when convergence theorems are stated.

A classical treatment of the equality constrained problem is the theory of maxima and minima as discussed, for example by Hancock (1917) [23]. Recent developments in this field have occurred mostly in 1950's and onwards. Several textbooks and monographs are available on general [24-29] and more specialized [30 - 34] aspects. The unifying concepts of functional analysis have been used to carry over many of these results to the infinite-dimensional case, also the unconstrained optimal control methods are all extensions, both basically and historically, of their finite-dimensional analogs.

2.1.1 Types of problems

There can be a classification according to special characteristics of the functions L, h and f . We define some of these terms here. References are given for specialized topics.

Unconstrained optimization: Here f and h are absent, and the classical theory of maxima and minima applies [23, 27].

Penalty Functions: In these methods, L is augmented by suitable functionals of f and h to generate a sequence of unconstrained problems whose solutions converge to the solution of (2.1.1 - 2.1.3).

Mathematical Programming (MP): These problems are formulated so that f is absent. Conceptually, of course, $f = 0$ is equivalent to $(f \geq 0, -f \geq 0)$, but this device is troublesome in practice. Linear Programming (LP): An important special case of MP where L and h are linear in π . A formulation of LP is often used where inequalities are converted into equalities (i.e. h absent but f present) by introducing extra variables called 'slack' variables. Note that in the absence of inequality constraints, a linear function has no maximum or minimum [31]. Quadratic programming: A special case of MP where the constraints are linear but L is quadratic in π , i.e., $L = \pi^T Q \pi + b^T \pi + a$ [24, ch.8].

Nonlinear Programming: A general term used when some or all of L, f and h are nonlinear. Usually reserved for the case where h is nonlinear and L non-quadratic. Equality constraints may or may not be present [32, 33].

Geometric programming: A class of specialized but sometimes very effective methods for MP when L, h are polynomials in π ; based initially on Cauchy's inequality between arithmetic and geometric means [25].

Integer Programming: A special case of LP, frequently encountered in industry, where the elements of π are restricted to be integers.

Classical techniques do not apply here [35].

Dynamic Programming: A powerful technique for problems not usually of the form (2.1.1), but representable as a multistage decision process [23]. It is mentioned here only because some integer programming problems lend themselves naturally to it.

Separable Programming: A special case of LP where L and h are sums of functions of a single variable:

$$L(\pi) = \sum L_j(\pi_j), h_i(\pi) = \sum h_{ij}(\pi_j) \quad (2.1.4)$$

This special structure enables dynamic programming to be used. It also makes it possible to solve larger LP problems [24].

It can be seen from the above that these problems can be divided into two basic types:

Unconstrained problems and constrained, i.e. mathematical programming problems.

Linear programming is a special case of the latter which has found a very large number of applications in business, operational research and some aspects of engineering design, where a large number of variables and constraints are present. It is also unique in that the global minima can be found. The size of these problems can be up to several thousands of variables. An important practical origin of nonlinear unconstrained problems is in nonlinear

least-square curve fitting, while constrained nonlinear problems arise routinely in design, with many inequality constraints. When the functions are mildly nonlinear, linear programming methods can sometimes be applied successively. A problem to which not known solution exists in the general case is to determine the global minimum on a given set. In many cases it cannot even be proved that an algorithm converges to a local minimum, as distinct from a stationary point.

Most theoretical results are available only for convex (e.g. quadratics with positive definite Hessians) functions L and constraints, a condition that cannot be easily verified in most practical problems, except in linear programming where it exists by definition.

The current situation for strongly nonlinear problems with many constraints and variables is not entirely satisfactory.

It is essential to rely on experience and intuition, both of which fail progressively as the dimensionality of the problem increases.

Now we shall review some existing gradient and conjugate direction algorithms for finite-dimensional problems. It is not intended here to duplicate the detailed discussion available in several books, only to present a brief survey.

2.2 Descent Methods and their Convergence Properties

We wish to consider methods that in addition to using function values also make use of the gradient of the objective function. Descent methods form a class of method in which the solution of the general unconstrained minimization problem:

$$\min f(x), \quad x \in \mathbb{R}^n \quad (2.2.1)$$

is bound by solving a sequence of one dimensional problems. From the point of view of practical computations conjugate direction methods are one of the most important descent methods for solving the general unconstrained minimization problem (2.2.1).

It is assumed that in the neighbourhood of the minimum, the function can be closely approximated by a positive definite quadratic form, and this is the major assumption made in the development of descent methods in general.

When we apply descent methods, we attempt to move from the current point $x^{(k)}$ in such a way as to reduce the value of the objective function f .

Each descent method is characterised by the provision of a descent vector $s^{(k)}$ at each iteration k (Search Direction), and the next point, $x^{(k+1)}$, is found by solving the one dimensional minimum problem, in which the function to be minimized is:

$$\phi(\alpha) = f(x^{(k)} + \alpha s^{(k)}). \quad (2.2.2)$$

Geometrically, we proceed in the direction $s^{(k)}$ to lower and lower level surfaces until $x^{(k+1)}$ is reached.

As f can not be reduced in this direction, $s^{(k)}$ must lie in the tangent plane to the level surface through $x^{(k+1)}$. Therefore when exact line search is applied we have:

$$s^{(k)T} g^{(k+1)} = 0 \quad (2.2.3)$$

where $g^{(k+1)} = \nabla f(x^{(k+1)})$ denotes the gradient f at $x^{(k+1)}$. See Figure 2.2.1 below:

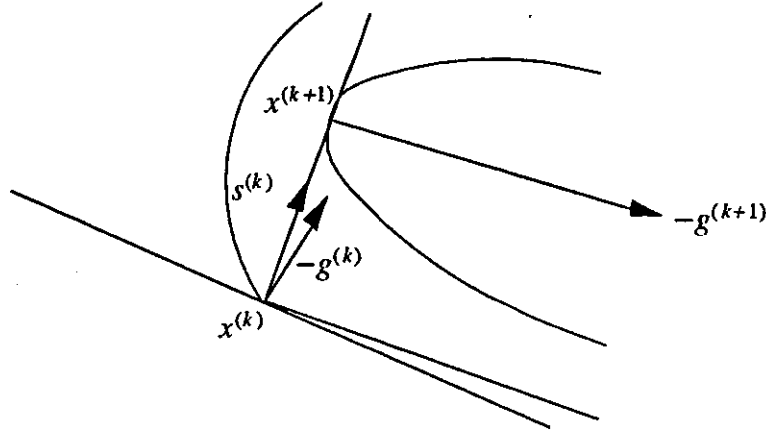


Figure 2.2.1: The descent vector $s^{(k)}$ lies in the tangent plane to the level surface through $x^{(k+1)}$.

The following property is satisfied at iteration k of a descent method:

$$s^{(k)T} g^{(k)} < 0 \quad (2.2.4)$$

The property (2.2.4) is called the descent property and $s^{(k)}$ is said to be a descent vector.

In descent methods, the descent vector $s^{(k)}$ is usually a certain transformation of the gradient of f at $x^{(k)}$: $g^{(k)}$. We could say that all descent methods use the basic iterative step:

$$x^{(k+1)} = x^{(k)} + \alpha B g^{(k)} \quad (2.2.5)$$

where B is a matrix which defines a transformation of the gradient and α is the step length in the direction $s^{(k)} = B.g^{(k)}$. What is important here is to choose a suitable direction $s^{(k)}$, in other words how to choose a suitable transformation matrix B of the gradient $g^{(k)}$. In contrast to the one-dimensional case, where the only moves are in the positive or negative directions, even in two dimensions there are a finite number of possible choices. Before considering a specific algorithm, we shall first consider some necessary and sufficient conditions for the convergence for descent methods.

2.2.1 Some theorems on convergence of descent methods

In order to establish the convergence of descent methods, it would be considerably easier by considering a model algorithm. All descent methods have the basic general form of the model algorithm below:

Step 1: Set $k = 1$

Step 2: Compute a descent search direction $s^{(k)} \neq 0$

Step 3: Compute a step length $\alpha^{(k)}$ such that

$$f(x^{(k)} + \alpha^{(k)} s^{(k)}) = \min_{\alpha} f(x^{(k)} + \alpha s^{(k)})$$

This is called a line search.

Step 4: Compute $x^{(k+1)} = x^{(k)} + \alpha s^{(k)}$

Step 5: Has the process converged? Yes: Then $x^{(k+1)}$ is the required minimum. No: Set $k = k + 1$ then go to step 2. Since the search direction $s^{(k)}$ computed in step 2 is a descent direction, it follows that there must exist $\alpha > 0$ such that $f(x^{(k)} + \alpha s^{(k)}) < f(x^{(k)})$. The requirement $f(x^{(k+1)}) < f(x^{(k)})$ by itself is not sufficient to ensure that the sequence $\{x^{(k)}\}_{k \in \mathbb{N}}$ converges to a minimum of f . Ideally $\alpha^{(k)}$ is chosen to solve the one-dimensional minimization problem

$$\min f(x^{(k)} + \alpha s^{(k)}), \quad \alpha \in \mathbb{R}^+ - \{0\}, \quad (2.2.6)$$

but in practice however, an exact solution to problem (2.2.6) is not usually possible and one accepts any value of $\alpha^{(k)}$ that satisfied certain standard conditions.

Let $x^{(1)}$ be an arbitrary starting point for a descent method of the form of the model algorithm above to solve the problem (2.2.1). The level set for this problem is defined as follows:

$$\{x \in \mathbb{R}^n : f(x) \leq f(x^{(1)})\}. \quad (2.2.7)$$

We also introduce:

$$\cos(\theta^{(k)}) = -\frac{g^{(k)T} s^{(k)}}{\|g^{(k)}\| \|s^{(k)}\|}, \quad (2.2.8)$$

where $\theta^{(k)}$ is the angle test between the negative gradient $-g^{(k)}$ and the search direction $s^{(k)}$.

$\cos(\theta^{(k)}) > 0$ holds for all descent methods.

The following convergence theorem has been established.

Theorem 2.2.1 (Zoutendijk (1976)) [36]

1. Suppose f is twice continuously differentiable in (2.2.7);
2. (2.2.7) is bounded;
3. The Hessian matrix $H(x)$ of second partial derivatives of f is bounded in (2.2.7);
4. $\alpha^{(k)} = \text{Arg} \min_{\alpha > 0} f(x^{(k)} + \alpha s^{(k)})$
5. $\sum_k \cos^2(\theta^{(k)}) = +\infty$.

Then $\nabla f(\bar{x}) = 0$ for at least one point of accumulation \bar{x} of the sequence $\{x^{(k)}\}$ generated by the model algorithm.

Theorem 2.2.2 (Zoutendijk (1976)) [36]

If in Theorem (2.2.1) condition (5) is replaced by the following condition:

$$(5'). \quad \exists k' \in \mathbb{N} \text{ and } \exists \theta > 0 : \forall k \geq k' \quad \cos(\theta^{(k)}) \geq \theta > 0,$$

then $\nabla f(\bar{x}) = 0$ for all points of accumulation \bar{x} of the sequence $\{x^{(k)}\}$ generated by the model algorithm.

The conditions for theorem 2.2.1 to hold are quite reasonable. The second condition prevents infinite solutions and in particular convergence of $f(x^{(k)})$ to a finite value while $x^{(k)} \rightarrow \pm\infty$.

This will exclude functions such as e^{-x} that are bounded below but strictly decreasing as $\|x\|$ becomes infinite. The third condition is eligible to hold if only the first condition holds, and it puts a restriction on the curvature of the objective function. It can be expected that functions with unbounded second partial derivatives will cause some problems. The fifth condition states that the angle between the negative gradient and the search direction should not go to $\frac{\pi}{2}$ too rapidly. This is strengthened in theorem 2.2.2. A limit should exist on the closeness of $s^{(k)}$ to orthogonality with the negative gradient. This prevents $s^{(k)}$ to be chosen so that to first order, f is almost constant along $s^{(k)}$ which will only occur if $s^{(k)}$ is almost parallel to the first order approximation to the contour line $f(x) = f(x^{(k)})$, so that the negative gradient and the search direction are almost orthogonal (i.e. $-\nabla f(x^{(k)})^T s^{(k)} / \|\nabla f(x^{(k)})\| \|s^{(k)}\|$ is close to zero). The stronger requirement (5') implies strong convergence ($\nabla f(\bar{x}) = 0$ for all points of accumulation); condition (5) only implies weak convergence (there exist a point of accumulation \bar{x} with $\nabla f(\bar{x}) = 0$). If there is only one point of accumulation, which is usually the case in practice, then it does not make any difference; if there is more than one point of accumulation. Then they obviously all have the same value. The type of convergence result, that has just been mentioned, is termed "Global Convergence", since there is no restriction on the closeness of $x^{(1)}$ to a stationary point. For more details on global convergence we can refer to Ostrowski (1966) [37], Wolfe (1969) [38], Sargent and Sebastian (1973) [39], Polak, Sargent and Sebastian (1974) [40], Zoutendijk (1970) [41] and (1976) [36], Fletcher (1980) [42] and Gill Murray and Wright (1981) [43].

Even if it can be proved theoretically that a sequence $\{x^{(k)}\}_{k \in \mathbb{N}}$ generated by the model algorithm above will converge in the limit to the required minimum, a method will be practicable only if convergence occurs with some rapidity. Global convergence does not give a measure for this rapidity and therefore gives no idea practically on the efficiency of a method. Now we try to discuss briefly some means of characterizing the rate of convergence at which such sequences converge.

Let $\{x^{(k)}\}_{k \in \mathbb{N}}$ be a sequence of points generated by the model algorithm above, converging to x^* . In order to simplify the discussion, we assume that the elements of this sequence are distinct, and for no value of k does $x^{(k)}$ equal x^* .

An effective technique for judging the rate of convergence is to compare the improvements at two consecutive steps, i.e., to measure the closeness of $x^{(k+1)}$ to x^* relative to that of $x^{(k)}$ to x^* .

A sequence $\{x^{(k)}\}_{k \in \mathbb{N}}$ is said to converge with order r , if r is the largest number such that

$$0 \leq \lim_{k \rightarrow \infty} \frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|^r} < \infty \quad (2.2.9)$$

r is usually known as the asymptotic rate of convergence.

If $r = 1$, the sequence is said to have linear convergence and if $r = 2$, it

is said to have quadratic convergence. If the sequence $\{x^{(k)}\}_{k \in \mathbb{N}}$ has order of convergence r , the asymptotic error constant is the value γ that satisfied:

$$\gamma = \lim_{k \rightarrow \infty} \frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|^r}. \quad (2.2.10)$$

when $r = 1$, γ must be strictly less than 1 in order for convergence to occur.

If a sequence has linear convergence, the step-wise decrease in $\|x^{(k)} - x^*\|$ varies substantially with the value of the asymptotic error constant. If the latter is zero when $r = 1$, the associated type of convergence is given the name superlinear convergence. Any value of r greater than 1 implies superlinear convergence. The importance of superlinear convergence is if it holds then $\lim_{k \rightarrow \infty} \frac{\|x^{(k+1)} - x^{(k)}\|}{\|x^{(k)} - x^*\|} \rightarrow 1$ so that that closeness of successive iterations is an indication of the closeness of $x^{(k)}$ to x^* . For more details about rates of convergence we can refer to Polak (1971) [44], Wolfe (1978) [45] and Gill, Murray and Wright (1981) [43].

2.2.2 Steepest descent method

Steepest descent method is one of the oldest and easiest methods for finding a solution to the general problem (2.2.1).

The method was first proposed by Cauchy (1847) [46] for the solution of systems of nonlinear equations.

The algorithm for the steepest descent method is as follows:

1. Set $k = 1$
2. Compute $\nabla f(x^{(k)}) \equiv g^{(k)}$.
3. Compute $\alpha^{(k)}$ such that $f(x^{(k)} - \alpha^{(k)}g^{(k)}) = \min_{\alpha} f(x^{(k)} - \alpha g^{(k)})$.
4. Compute $x^{(k+1)} = x^{(k)} - \alpha g^{(k)}$.
5. Has the process converged? If yes then $x^{(k+1)}$ is the required minimum if no set $k = k + 1$ Go to (2).

Consider the linear approximation to f based on the Taylor series expansion about $x^{(k)}$:

$$f(x^{(k)} + s) \approx f(x^{(k)}) + g^{(k)T} \cdot s \quad (2.2.11)$$

If we assume that a step of unity is taken along s , it will appear that a good way to achieve a large reduction in $f(x)$ is to choose s so that $g^{(k)T} \cdot s$ is large and negative. Naturally some normalization must be imposed on s , otherwise for any \bar{s} such that $g^{(k)T} \bar{s} < 0$, one would simply choose s as an arbitrary large positive multiple of \bar{s} . The aim, is to choose s so that amongst all suitably normalized vectors, $g^{(k)T} s$ is minimum.

Given some norm $\|\cdot\|$, $s^{(k)}$ is thus the solution of the minimization problem:

$$\min_{s \in \mathbb{R}^n} \frac{g^{(k)T} s}{\|s\|}. \quad (2.2.12)$$

The solution of the problem (2.2.12) depends on the choice of norm. When the norm is defined by a given symmetric positive definite matrix, C say, i.e., $\|s\|_C = (s^T C s)^{\frac{1}{2}}$, the solution of (2.2.12) is:

$$s^{(k)} = -C^{-1}g^{(k)}. \quad (2.2.13)$$

The matrix B defined in (2.2.5) used as a transformation of the gradient is in this case $B = -C^{-1}$.

If the two-norm is used, i.e. $s = (s^T s)^{\frac{1}{2}}$, the solution of (2.2.12) is just the negative gradient:

$$s^{(k)} = -g^{(k)}. \quad (2.2.14)$$

In this case we have $B = -I$, where I is the identity matrix.

Since (2.2.14) solves (2.2.12) with respect to the two norms, the negative gradient direction $s^{(k)}$ in (2.2.14) is termed the direction of steepest descent and the algorithm using this direction at every iteration is called the steepest descent algorithm.

Unless the gradient vanishes, the steepest descent direction is clearly a descent direction, since the vectors $s^{(k)}$ and $g^{(k)}$ are bounded away from orthogonality and the directional derivative is such that

$$g^{(k)T} s^{(k)} = -g^{(k)T} g^{(k)} < 0.$$

Consequently a suitable line search technique, may be combined with the steepest descent algorithm to yield a method with guaranteed global convergence.

Unfortunately, a proof of global convergence for an algorithm does not ensure that it is an efficient method. Although this method is useful for a large class of well conditioned problems, experience has shown it is too slow. Kowalik and Osborne (1968) [27] discuss some of the reasons for this inefficiency.

The weakness of the method is mainly due to the fact that the search directions generated by the algorithm are not linearly independent. However the fact that the method only uses successive linear approximations to the objective function is another factor in its efficiency. The steepest descent is shown to have at best a linear rate of convergence. When applied to quadratic the rate of convergence is

$$a.e.c \text{ (asymptotic error constant)} \leq \frac{(\lambda_M - \lambda_m)}{(\lambda_M + \lambda_m)}$$

for proofs of global and local convergence we can refer to Gill, Murray and Wright (1981) [43].

2.3 Conjugate Gradient Methods and their Convergence Properties

In order to overcome the weakness we faced in using the steepest descent method described in section 2.2.2 we introduce the conjugate gradient methods. These methods were originally proposed by Hestenes and Stiefel (1952)

[47] for the solution of systems of linear equations and are therefore based on the assumption that in the neighbourhood of a nondegenerate minimum point, a function behaves like a quadratic function. These methods were adapted later as methods for optimisation by Fletcher and Reeves (1964) [48].

For problem (2.2.1), a quadratic model at $x^{(k)}$ is

$$\min_{x \in \mathbb{R}^n} F^{(k)}(x), \quad (2.3.1)$$

where

$$F^{(k)}(x) = f^{(k)} + g^{(k)T}(x - x^{(k)}) + \frac{1}{2}(x - x^{(k)})^T B^{(k)}(x - x^{(k)}) \quad (2.3.2)$$

and $\beta^{(k)} \in \mathbb{R}^{n \times n}$ is an approximate to the Hessian at x , or is the Hessian itself. Let $\beta^{(k)} = G^{(k)}$ and write $F^{(k)}$ as

$$F^{(k)}(x) = \frac{1}{2}x^T G^{(k)}x + b^T x + c. \quad (2.3.3)$$

Two distinct nonzero directions $s^{(i)}$ and $s^{(j)}$ are said to be conjugate to each other with respect to $G^{(k)}$ if

$$s^{(i)T} G^{(k)} s^{(j)} = 0.$$

If we have a set of n mutually conjugate directions $s^{(1)}, \dots, s^{(n-1)}$, then by carrying out exact line searches along each direction (assume they are descent directions and $G^{(k)}$ is positive definite) we shall find the minimum of (2.3.3) within n steps.

The search directions generated by conjugate gradient algorithms are of the form:

$$s^{(k)} = \begin{cases} -g^{(k)} & \text{if } k = 1 \\ -g^{(k)} + \beta^{(k)} s^{(k-1)} & \text{if } k > 1 \end{cases} \quad (2.3.4)$$

where β can be calculated by one of the following formulae:

Hestenes and Stiefel (1952) [47]:

$$\beta_{HS}^{(k)} = \frac{(g^{(k+1)} - g^{(k)})^T g^{(k+1)}}{(g^{(k+1)} - g^{(k)T}) s^{(k)}}. \quad (2.3.5)$$

Fletcher and Reeves (1964) [48]:

$$\beta_{FR}^{(k)} = \frac{g^{(k+1)T} g^{(k+1)}}{g^{(k)T} g^{(k)}}. \quad (2.3.6)$$

Daniel (1967) [49]:

$$\beta_D^{(k)} = \frac{g^{(k+1)T} G^{(k+1)} s^{(k)}}{s^{(k)T} G^{(k+1)} s^{(k)}}. \quad (2.3.7)$$

Sorensen (1969) [50]:

$$\beta_s^{(k)} = \frac{g^{(k+1)T} (g^{(k+1)} - g^{(k)})}{s^{(k)T} (g^{(k+1)} - g^{(k)})}. \quad (2.3.8)$$

and

$$\beta_{GS} = \frac{-g^{(k+1)T}(g^{(k+1)} - g^{(k)})}{s^{(k)T}g^{(k)}}. \quad (2.3.9)$$

Polak and Ribière (1969) [51]:

$$\beta_{PR}^{(k)} = \frac{g^{(k+1)T}(g^{(k+1)} - g^{(k)})}{g^{(k)T}g^{(k)}}. \quad (2.3.10)$$

Dixon (1972) [52] attributes the following formula to Myers:

$$\beta_D^{(k)} = -\frac{g^{(k+1)T}g^{(k+1)}}{s^{(k)T}g^{(k)}}. \quad (2.3.11)$$

Lin and Storey (1991) [53]:

$$\beta_{LS}^{(k)} = \frac{g^{(k+1)T}G^{(k+1)}s^{(k)}g^{(k+1)T}g^{(k+1)} - g^{(k+1)T}G^{(k+1)}g^{(k+1)}s^{(k)T}g^{(k+1)}}{s^{(k)T}G^{(k+1)}s^{(k)}g^{(k+1)T}g^{(k+1)} - g^{(k+1)T}G^{(k+1)}s^{(k)}s^{(k)T}g^{(k+1)}},$$

where

$$s^{(k+1)} = \frac{(ug^{(k+1)T}s^{(k)} - tg^{(k+1)T}g^{(k)} + (ug^{(k+1)T}g^{(k+1)} - vg^{(k+1)T}s^{(k)})s^{(k)})}{w},$$

and

$$w = tv - u^2, t = s^{(k)T}G^{(k+1)}s^{(k)}, v = g^{(k+1)T}G^{(k+1)}g^{(k+1)}$$

and

$$u = g^{(k+1)T}G^{(k+1)}s^{(k)}. \quad (2.3.12)$$

Generalized Polak-Ribière (1992) [54]:

$$\beta_{GPR}^{(k)} = -g^{(k+1)T}(g^{(k+1)} - \bar{g}^{(k)})/\bar{g}^{(k)T}\bar{s}^{(k)}$$

where

$$\bar{s}^{(k)} = s^{(k)} - (g^{(k+1)T}s^{(k)}/g^{(k+1)T}g^{(k+1)})g^{(k+1)}$$

and

$$\begin{aligned} \delta &= \min(1, \sqrt{\eta}/\sqrt{\bar{s}^{(k)T}\bar{s}^{(k)}}) \\ \bar{g}^{(k)} &= g(x^{(k+1)} - \delta\bar{s}^{(k)}). \end{aligned} \quad (2.3.13)$$

When all these β are used on quadratics with exact arithmetic and start with the **steepest descent** direction on the first iteration they are equivalent. When they are used on general non-quadratic functions, we do not try to minimize the quadratic approximation (2.3.3) at $x^{(k)}$ by using several iterations of the conjugate gradient method on it, as this would lead us to the truncated Newton method. Instead in each iteration, after finding the direction (2.3.4), a line search is carried out on the function f to get a new point, then the quadratic approximation at the new point is taken as the quadratic approximation (2.3.3) and a new direction of the form (2.3.4) is calculated.

This is known as the conjugate gradient method of approximation for general functions. When the various methods with different β are applied we could not guarantee the same efficiency, as a matter of fact the difference could be finite significant. Among the two well known formulae (2.3.6) and

(2.3.10), without restarts, it was found in most cases the the PR performs better than the FR. One reason (Powell, 1977) [55] for this is that if at some iteration the search direction is nearly orthogonal to the gradient, then it is likely that little reduction in function value will be achieved and the two subsequent gradients will be close to each other. If the FR formulae is used in such a case the new direction will again be nearly orthogonal to the gradient, and this inefficient situation will be maintained for several iterations before recovering. But if the PR formula is used then it will give $\beta \approx 0$, which amounts to restarting the algorithm "automatically". In the study that (Hu and Storey, 1990) [56] carried out on the conjugate gradient algorithms, they found that the search direction given by the PR formula is generally much closer to the corresponding two dimensional Newton direction than is the direction given by the FR formula. They have the idea that lacking the "automatic" restart property does account for the inefficiency of the FR algorithm in some cases, but possessing it may not imply an efficient algorithm. They explained the differences in efficiency of different conjugate gradient algorithms by the differences in their closeness to the two dimensional newton algorithm.

The global convergence of the conjugate gradient method was investigated by Powell (1984) [57] and Al-Baali (1985) [58]. At this stage before going any further let us introduce some definitions and convergence.

Definition 2.3.1

A model iterative procedure Algorithm for solving (2.2.1) is said to be globally convergent if starting from any point $x^{(0)}$, the sequence $\{x^{(k)}\}_{k=0,1,\dots}$ satisfies

$$\liminf_{k \rightarrow \infty} \|g^{(k)}\| = 0.$$

Powell (1984) [57] and Al-Baali (1985) [58] found that the FR method is globally convergent if the line search conditions (Goldstein (1965) [59], Wolfe (1969) [38], Powell (1976) [60])

$$f(x^{(k)} + \alpha^{(k)} s^{(k)}) \leq f(x^{(k)}) + \rho \lambda^{(k)} s^{(k)T} g^{(k)} \quad (2.3.17)$$

and

$$s^{(k)T} g(x^{(k)} + \alpha^{(k)} s^{(k)}) \geq \sigma s^{(k)T} g^{(k)}, \quad (2.3.18)$$

where $0 < \rho \leq \sigma < 1$ are satisfied with $0 < \rho < \sigma < \frac{1}{2}$, but counter examples (Powell (1984) [57]) show that PR method may not be globally convergent even on some smooth functions with exact line searches. Efforts were made to combine the globally convergent property of the FR method with the better numerical performance of the PR method (Touati Ahmed and Storey, 1990) [61] and also some results by (Gilbert and Nocedal, 1990) [62] shows that any conjugate gradient method is globally convergent if β is restricted to be positive, the search direction satisfies a sufficient descent condition and the method has some kind of "automatic" restart property. The research done by Hu and Storey (1991) [63] refined some of the results of Touati-Ahmed and Storey (1990) [61].

The conjugate gradient methods required very little storage and very little operational cost. It has been established in numerical experiments that

they generally prefer accurate line searches, and restart with steepest descent directions should often be made. A restart criterion which was adapted by Powell (1977) [55] was $|g^{(k)T}g^{(k-1)}| > 0.2\|g^{(k)}\|^2$. Since the immediate effect of restarting with steepest descent directions may not be good, other restart procedures were studied by Beale (1972) [64], Powell (1977) [55] and Nazareth (1977) [65] and substantial improvement in numerical performance was observed.

Compared with the quasi-Newton methods the conjugate gradient methods are simple, but they generally require more iterations and more function (gradient) evaluations, therefore they are suitable for use when function and gradient evaluations are cheap, or when the sizes of the problems are very large and computer storage becomes a limiting factor.

2.3.1 A study of quadratic functions and their properties

A quadratic function can be described as:

$$F(x) = \frac{1}{2}x^T Gx + b^T x + C. \quad (2.3.19)$$

where n is the dimension of the vector x with n a fixed integer, G a real symmetric $n \times n$ matrix, b a fixed n -dimensional vector and C a scalar. The gradient of F at x is the vector:

$$g(x) = Gx + b. \quad (2.3.20)$$

A stationary point of f is a point x such that $g(x) = 0$. for the quadratic function F we could say x is a stationary point if and only if x is a solution of the linear system of equations:

$$Gx = -b. \quad (2.3.21)$$

The system (2.3.21) may or may not have a solution, but if G is non singular then there is only one solution, i.e.

$$x^* = -G^{-1}b. \quad (2.3.22)$$

where x^* is the unique stationary point of F . If G is singular and x^* is a solution of (2.3.21) then every solution of (2.3.21) can be expressed in the form $x = x^* + z$, where z is a null vector of G , i.e. a vector z such that $Gz = 0$. hence, if x^* is a stationary point of F , then every stationary point of F differs from x^* by a null vector z of G .

Since F is quadratic the following identity exists:

$$F(x + s) = F(x) + s^T(Gx + b) + \frac{1}{2}s^T Gs. \quad (2.3.23)$$

This is based on the fact that the quadratic approximation to F based on Taylor series expression:

$$F((x^{(k)} + s) \approx F(x^{(k)}) + g^{(k)T}s + \frac{1}{2}s^T H^{(k)}s \quad (2.3.24)$$

where $H^{(k)} = \nabla^2 F(x^{(k)})$ is the matrix of second partial derivatives at $x^{(k)}$ or Hessian matrix.

Therefore the quadratic function (2.3.23) can be rewritten as:

$$F(x + s) = F(x) + s^T g(x) + \frac{1}{2} s^T H(x) s. \quad (2.3.25)$$

The matrix $H(x) = G$ is the Hessian matrix of F . If x^* is a stationary point, then, by replacing x by x^* and s by $x - x^*$ in (2.3.25) we obtain the formula:

$$F(x) = F(x^*) + \frac{1}{2} (x - x^*)^T G (x - x^*) \quad (2.3.26)$$

for F relative to a stationary point of F .

Geometrically this function states that, when G is non singular, the stationary point $x^* = -G^{-1}b$ of F is the centre of the quadratic surface

$$F(x) = \gamma. \quad (2.3.27)$$

where γ is a constant.

A minimum point x^* of F is a stationary point of F and by (2.3.26) a stationary point x^* of F is a minimum point of F if and only if G is nonnegative, i.e. if and only if the inequality $s^T G s \geq 0$ holds for every vector s . If $s^T G s > 0$ whenever $s \neq 0$, i.e. if G is positive definite, then $x^* = -G^{-1}b$ is the unique minimum point of F . F is said to be a positive definite quadratic function if G is positive definite. The level surfaces (2.3.27) for a positive definite quadratic function F are ellipsoids having $x^* = -G^{-1}b$ as their common centre. Then in fact the problem of minimizing F is equivalent to the geometrical problem of finding the centre of an ellipsoid. This can lead us to a geometrical description of the methods of conjugate directions and conjugate gradients.

It should be noted that the level surfaces

$$F(x) = \gamma \quad \text{for} \quad \gamma > F(x^*)$$

are similar ellipsoids.

This means that if x_α and x_β are the points in which a ray emanating from the common centre x^* cuts the level surfaces $F(x) = \alpha$ and $F(x) = \beta$, respectively, then the ratio

$$\frac{|x_\beta - x^*|}{|x_\alpha - x^*|} = r$$

of the distances of x_α and x_β from x^* is independent of the choice of this ray. This situation is illustrated as follows in figure 2.3.1:

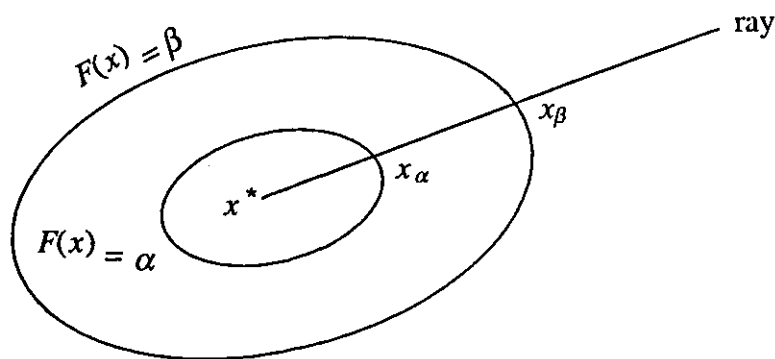


Figure 2.3.1

The matrix G is positive definite so that it ensures the existence of the minimum point $x^* = -G^{-1}b$.

2.3.2 Conjugacy of quadratic functions

Two distinct non zero vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$ are said to be conjugate with respect to an $n \times n$ symmetric positive definite matrix, G , if $x^T G y = 0$. We can see that if G is the identity matrix I , then the definition of conjugacy becomes that of orthogonality. Therefore conjugacy is a generalization of the concept of orthogonality by letting the scalar product be

$$[x, y] = x^T G y. \quad (2.3.28)$$

On the other hand if x and y are conjugate with respect to the matrix G then the vectors x and $z = G y$ are orthogonal.

A System of vectors $\{z^{(1)}, z^{(2)}, \dots, z^{(n)}\} \in \mathbb{R}^n$ is said to be a system of conjugate vectors with respect to matrix G if

$$z^{(i)T} G z^{(j)} = 0 \quad \forall i \neq j. \quad (2.3.29)$$

Since G is a symmetric matrix, the system $\{r^{(1)}, r^{(2)}, \dots, r^{(n)}\}$ of its eigenvectors is a system of conjugate vectors. So we have:

$$r^{(i)T} G r^{(j)} = r^{(i)T} (\lambda_j r^{(j)}) = \lambda_j r^{(i)T} r^{(j)} = 0 \quad \text{for } i \neq j \quad (2.3.30)$$

where λ_j is the j th eigenvalue of G .

A system of conjugate vectors with respect to the matrix G can be constructed from any linearly independent system of vectors $x^{(1)}, \dots, x^{(n)}$ by using the orthogonalization method of Gram-Schmidt using the scalar product (2.3.28). Refer to (Dahlquist and Björk (1974) [66].

The procedure is as follows:

Put $z^{(1)} = x^{(1)}$. Then for $i = 2, 3, \dots, n$ successively put

$$z^{(i)} = x^{(i)} + \sum_{j=1}^{i-1} \beta_{ij} z^{(j)}, \quad (2.3.31)$$

where the coefficients β_{ij} are chosen to make $z^{(i)T} G z^{(k)} = 0, \quad \forall k = 1, 2, \dots, i-1$.

This means that β_{ij} must satisfy the equations:

$$x^{(i)T} G z^{(k)} + \sum_{j=1}^{i-1} \beta_{ij} z^{(j)T} G z^{(k)} = 0 \quad (2.3.32)$$

In principle (2.3.32) defines $i-1$ simultaneous equations for the $i-1$ unknown β_{ij} . But if we use the fact that $z^{(1)}, \dots, z^{(i-1)}$ are mutually conjugate with respect to the matrix G , we find that they reduce to the $i-1$ simple equations:

$$\beta_{ij} = -x^{(i)T} G z^{(j)} / z^{(j)T} G z^{(j)} \quad j = 1, \dots, i-1. \quad (2.3.33)$$

Since $x^{(j)}$ are assumed linearly independent then $z^{(j)}$ cannot vanish (they are linear combinations of $x^{(j)}$) and hence the denominator in (2.3.33) is not zero if G is positive definite.

Now we can describe the properties of the quadratic function (2.3.19), which the algorithms of conjugate gradients are based on.

Theorem 2.3.1:

The minimum points of the quadratic function (2.3.19) on parallel lines, lie on an $(n-1)$ plane π_{n-1} through the minimum point x^* of f . The $(n-1)$ -plane π_{n-1} is defined by the equation

$$s^T (Gx + b) = 0. \quad (2.3.34)$$

where s is a direction vector for these parallel lines. The vector G_s is normal to π_{n-1} .

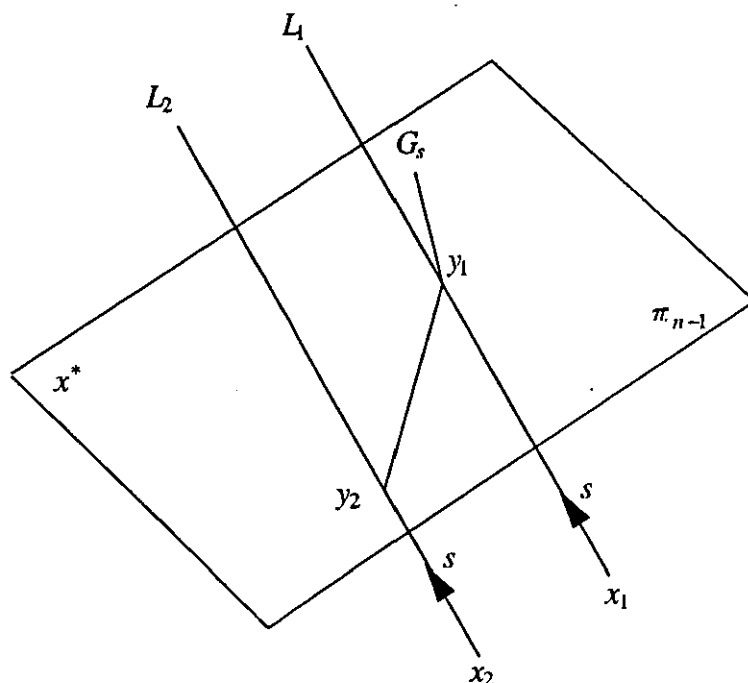


Figure 2.3.2

For the proof refer to Touati-Ahmed (1989) [67].

Theorem 2.3.2:

Let $x^{(1)} \in \mathbb{R}^n$ and let $\{z^{(1)}, z^{(2)}, \dots, z^{(n)}\}$ be a system of conjugate vectors with respect to the positive definite matrix G of the quadratic function (2.3.19). Let z be the set:

$$Z = \{x = x^{(1)} + z | z \in [z^{(1)}, \dots, z^{(n)}]\}$$

where $[z^{(1)}, \dots, z^{(n)}]$ is the subspace of the linear combination of $z^{(1)}, z^{(2)}, \dots, z^{(n)}$. Then:

The minimum point x^* of the quadratic function (2.3.19) over the set z can be calculated by minimizing f over the points $z^{(i)} \in Z$ such that

$$z^{(i)T}G(x^{(i)} - x^{(1)}) = 0$$

successively. In other words:

$$F(x^{(i)} + \alpha_i^* z^{(i)}) = \min_{\alpha_i} F(x^{(i)} + \alpha_i z^{(i)}) \implies F(x^*)$$

$$= \min_{x \in z} F(x) \text{ where } x^* = x^{(1)} + \alpha_1^* z^{(1)} + \alpha_2^* z^{(2)} + \dots + \alpha_n^* z^{(n)}.$$

For the proof refer to Touati-Ahmed (1989) [67].

Now computational methods can be developed to minimize the positive definite quadratic function F of (2.3.19). These methods consist of minimizing F successfully along lines. Where these lines are mutually conjugate, the method is called a conjugate direction method for finding the minimum point $x^* = -G^{-1}b$ of F .

Considering theorem 2.3.2, a conjugate direction method terminates in $m \leq n$ steps if there is no round off error. Also Theorem 2.3.1 confirms this fact, as it can be seen from the following geometrical description of a conjugate direction method.

- (1) Select a point $x^{(1)}$ and a line L_1 through $x^{(1)}$ in a direction $s^{(1)}$.
- (2) Find the minimum point $x^{(2)}$ of F on L_1 .
- (3) Construct the $(n-1)$ -plane π_{n-1} through $x^{(2)}$ which is conjugate to $s^{(1)}$.

By theorem 2.3.1 the minimum x^* of F is in π_{n-1} . Consequently, our next search can be limited to π_{n-1} so that the dimensionality of our space of search is reduced by one.

The process can now be repeated, restricting to the $(n-1)$ -plane π_{n-1} .

- (1) Select a line L_2 in π_{n-1} through $x^{(2)}$ in a direction $s^{(2)}$.
- (2) Find the minimum point $x^{(3)}$ of F on L_2 .
- (3) Construct the $(n-2)$ -plane π_{n-2} through $x^{(3)}$ which is conjugate to $s^{(2)}$.

Again using theorem 2.3.1, with π_{n-1} playing the role of the Euclidean space E_n , the minimum point x^* of F is in π_{n-2} so that our search can be limited to π_{n-2} . Again the dimension of our space of search has been reduced by one. Proceeding like this the dimensionality of our space of search is reduced by one at each step.

At the n th step our space of search in the line π_1 , through x^* so that the minimum point $x^{(n+1)}$ of F on π_1 coincides with the minimum point x^* . There are rare occasions that we have $x^{(m+1)} = x^*$ at an m th step ($m < n$), in which case we can terminate in $m < n$ steps. In a quadratic with $m < n$ distinct eigenvalues convergence is in m steps.

A conjugate gradient method is a conjugate direction method characterized by the construction in step (3) of each iteration k of the $(n-k)$ -plane π_{n-k} , through the minimum point $x^{(k+1)}$ of F on the line L_k , which is conjugate to the direction $s^{(k)}$ of the line L_k .

2.3.3 The convergence properties of conjugate gradient methods on quadratic functions

We now consider the convergence properties of conjugate gradient methods when the objective function is a positive definite quadratic function.

When the conjugate gradient method is applied on a positive definite quadratic function using the formulae of Fletcher-Reeves or Polak-Ribière to compute the update $\beta^{(k)}$ it will terminate in at most n steps from any starting point $x^{(1)}$, provided the starting direction is the steepest descent direction $s^{(1)} = -g^{(1)}$. This result follows from Theorem 2.3.2. It can also be shown that the number of steps required for termination to occur, is equal to the number of distinct eigenvalues of the matrix G . For a proof of this result see Hestenes (1980). [68].

Crowder and Wolfe (1971) [69] have shown that if the wrong starting direction is used, then the rate of convergence is at best linear. Also the extension of this result has been carried out by Powell (1976) [60], to show that if the objective function is a positive definite quadratic function and if the initial search direction is an arbitrary descent direction, then if termination does not occur within $n + 1$ steps, the rate of convergence is only linear. Therefore when F is quadratic superlinear convergence never occurs. Powell (1976) [60] also found that linear convergence can be obtained and every sequence of $(\ell + 1)$ consecutive search directions is mutually conjugate, where ℓ is a positive integer less than $(n - 1)$. He also found conditions to improve on $x^{(1)}$ and $s^{(1)}$ that are necessary and sufficient for termination to occur in a finite number of steps.

From this result of Powell's one can see that if $s^{(1)}$ is fixed and if the components of $x^{(1)}$ are chosen at random, for instance from the uniform distribution in $[-1, 1]$, then that probability of obtaining termination by chance, where $n \geq 3$ and there are no discrepancies, is zero. Thus a linear rate of convergence is usual when the conjugate gradient algorithm is applied on a general convex quadratic function and when both $s^{(1)}$ and $x^{(1)}$ are arbitrary.

2.3.4 Extension of conjugate gradient methods to general functions

We can combine the conjugate gradient algorithm for minimizing a quadratic function with Newton's method for minimizing a non-quadratic function f to obtain an effective method for finding the maximum x^* of f . Here we assume that the Hessian matrix of f is positive definite. The only significant modification done is that the step lengths $\alpha^{(k)}$ are no longer easily provided by an exact line search but they are usually computed by an inexact line search procedure.

The finite termination property of the method on quadratic functions suggest that the definition:

$$s^{(k+1)} = -g^{(k+1)} + \beta^{(k)}s^{(k)}$$

should be abandoned after a cycle of linear searches and that $s^{(k)}$ should

then be set to the steepest descent direction $-g^{(k)}$. This strategy is known as restarting; it is also known as "resetting" or "reinitialization".

The combined method is as follows:

1. Select a starting point $x^{(1)}$.
2. Set up the Newton approximation:

$$F(s) = f(x^{(1)} + s) = f(x^{(1)}) + g^{(1)T}s + \frac{1}{2}s^T H(x^{(1)})s.$$

3. Setting $s^{(1)} = 0$, as the initial point of F , use a conjugate gradient routine to obtain the minimum point $s^{(n+1)}$ of F .
4. Repeat computations (2) and (3) with $x^{(1)}$ replaced by $x^{(1)} + s^{(n+1)}$. Terminate if $|g^{(k)}|$ is so small that $x^{(k)} = x^{(1)} + s^{(k(n+1))}$ is an acceptable estimate of the minimum point x^* of f .

It should be pointed out that the approximation is not done explicitly, and due to difficulty in obtaining an exact step length without complications, instead inexact line search procedures are used.

2.4 Fletcher Reeves Method and its Convergence Properties

Conjugate gradient methods form a class of methods that generate directions of search without storing any matrix. The aim is to solve the unconstrained minimization problem:

$$\text{Minimize } f(x), \quad x \in \mathbb{R}^n \quad (2.4.1)$$

using a sequence of line searches:

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)}s^{(k)}, \quad (2.4.2)$$

Starting from $x^{(1)}$ an estimate of a minimizer of x^* . Where the line search is exact, the stepsize $\alpha^{(k)}$ in the direction of $s^{(k)}$ is as follows:

$$\alpha^{(k)} = \text{Arg min}_{\alpha} f(x^{(k)} + \alpha s^{(k)}). \quad (2.4.3)$$

But in practice, and exact line search is not usually possible, therefore certain conditions are forced on the value of $\alpha^{(k)}$. Fletcher (1980) [42] suggests that $\alpha^{(k)}$ is such that $x^{(k+1)}$ satisfies the condition:

$$|g^{(k+1)}s^{(k)}| \leq s - \sigma g^{(k)T}s^{(k)}, \quad (2.4.4)$$

together with the Goldstein (1965) [59] requirement:

$$f(x^{(k+1)}) \leq f(x^{(k)}) + \rho \alpha^{(k)} g^{(k)T}s^{(k)}, \quad (2.4.5)$$

where $g^{(k)} = \nabla f(x^{(k)})$ is the gradient vector at $x^{(k)}$, $\rho \in (0, \frac{1}{2})$, $\sigma \in (0, 1)$ and $\rho < \sigma$.

The search direction $s^{(k)}$ when it is descent can be defined as:

$$g^{(k)T} s^{(k)} < 0 \quad (2.4.6)$$

It holds for all k such that $g^{(k)} \neq 0$, which ensures that $f(x)$ can be decreased in the line search.

Condition (2.4.4) ensures that the modulus of the slope is reduced by a factor σ or less in the line search. Refer to Fletcher (1980) [42].

The search direction $s^{(k)}$, for conjugate gradient methods is defined as follows:

$$\left. \begin{aligned} s^{(1)} &= -g^{(1)} \\ s^{(k+1)} &= -g^{(k+1)} + \beta^{(k)} s^{(k)}, \quad k > 1 \end{aligned} \right\} \quad (2.4.7)$$

We consider $\beta^{(k)}$ as follows:

The Fletcher-Reeves update (1964) [48]:

$$\beta^{(k)} = \|g^{(k+1)}\|^2 / \|g^{(k)}\|^2. \quad (2.4.8)$$

Where $\|\cdot\|$ denotes the Euclidean norm.

We assume that the level set:

$$\{x : f(x) \leq f(x^{(1)})\} \quad (2.4.9)$$

is bounded. This will ensure that $\alpha^{(k)}$ is well defined for all k .

2.4.1 Descent property and global convergence in the case of exact line search

It is obvious that for any conjugate gradient algorithm when $s^{(1)T} g^{(1)} = -g^{(1)T} g^{(1)} < 0$, the descent property (2.4.6) holds on the first iteration, moreover when the line search is exact we have:

$$g^{(k+1)T} s^{(k)} = 0 \quad \text{for } k \geq 1. \quad (2.4.10)$$

Hence from (2.4.7) and (2.4.10) it follows:

$$\begin{aligned} g^{(k+1)T} s^{(k+1)} &= g^{(k+1)T} (-g^{(k+1)} + \beta^{(k)} s^{(k)}) \\ &= -\|g^{(k+1)}\|^2 < 0 \end{aligned} \quad (2.4.11)$$

This shows that a descent property holds on all iterations for any conjugate gradient formula and in particular for Fletcher-Reeves.

In the case of Global convergence Powell (1983) [70] shows that if the level set (2.4.9) is bounded, if $\alpha^{(k)}$ is defined so that (2.4.10) holds for all k , and if $f(x)$ is twice continuously differentiable, then the Fletcher-Reeves method achieves the limit:

$$\lim_{k \rightarrow \infty} \inf \|g^{(k)}\| = 0 \quad (2.4.12)$$

2.4.2 Descent property and global convergence in the case of inexact line search

Powell's result was extended by Al-Baali (1985) [58] to show that even for an inexact line search, the descent property (2.4.6) holds for all k and global convergence is achieved for the Fletcher-Reeves method.

Theorem 2.4.1: (Al-Baali (1985) [58])

If $\alpha^{(k)}$ is calculated which satisfies (2.4.4) with $\sigma \in (0, \frac{1}{2}]$ for all k such that $g^{(k)} \neq 0$, then the descent property for the Fletcher-Reeves method holds for all such k . He also proves a theorem for the global convergence of the Fletcher-Reeves method with an inexact line search satisfying (2.4.4) and (2.4.5).

Theorem 2.4.2: (Al-Baali (1985) [58])

If the level set (2.4.9) is bounded, if $f(x)$ is twice continuously differentiable, and if an inexact line search satisfying (2.4.4) and (2.4.5) with $\rho < \sigma < \frac{1}{2}$ is used, then the Fletcher-Reeves achieves the limit (2.4.12).

2.5 The Polak-Ribière Method and its Global Convergence

If equation (2.4.8) is replaced by the expression

$$\beta^{(k)} = g^{(k+1)T}(g^{(k+1)} - g^{(k)}) / \|g^{(k)}\|^2 \quad (2.5.1)$$

the conjugate gradient method of Polak and Ribière (1969) [51] results.

When numerical computations are carried out it is usually found that the method of Polak-Ribière is more successful than that of Fletcher-Reeves.

Powell (1977) [55] and Touati-Ahmed (1989) [67] have solved some minimization problems to confirm this.

To try to explain this remark, we ask whether either conjugate gradient method can be highly inefficient. Specifically, we suppose that the k -th iteration of a conjugate gradient method has made a change $\|x^{(k+1)} - x^{(k)}\|$ that is much smaller than the step that would have been taken by the steepest descent algorithm, and we ask whether the next iteration can be as bad. A relatively tiny value of $\|x^{(k+1)} - x^{(k)}\|$ occurs only if the angle θ_k say, between $s^{(k)}$ and $-g^{(k)}$ is close to $\frac{\pi}{2}$. Since the line search along $s^{(k-1)}$ and equation

$$s^{(k)} = -g^{(k)} + \beta^{(k)} + s^{(k-1)} \quad (2.5.2)$$

imply the value

$$\begin{aligned} \cos \theta^{(k)} &= -s^{(k)T}g^{(k)} / (\|s^{(k)}\| \|g^{(k)}\|) \\ &= \|g^{(k)}\| / \|s^{(k)}\|, \end{aligned} \quad (2.5.3)$$

we must have $\|s^{(k)}\| \gg \|g^{(k)}\|$, and we wish to avoid $\|s^{(k+1)}\| \gg \|g^{(k+1)}\|$. Now the relatively small value of $\|x^{(k+1)} - x^{(k)}\|$ gives $g^{(k+1)} \approx g^{(k)}$, so the

Fletcher-Reeves and Polak-Ribière formulae yield $\beta^{(k+1)} \approx 1$ and $|\beta^{(k+1)}| \ll 1$ respectively. Further using the line search along $s^{(k)}$ and the definition of $s^{(k+1)}$, we deduce the relation

$$\|s^{(k+1)}\| = \left[\|g^{(k+1)}\|^2 + \beta^{(k+1)^2} \|s^{(k)}\|^2 \right]^{\frac{1}{2}}. \quad (2.5.4)$$

It follows that the Fletcher-Reeves formula gives $\|s^{(k+1)}\| \approx \|s^{(k)}\|$, but the Polak-Ribière formula gives $\|s^{(k+1)}\| \ll \|s^{(k)}\|$ as required. Refer to Powell (1985) [71] for more details.

2.5.1 Global convergence

Although numerical computations indicate that the Polak-Ribière method is superior to the Fletcher-Reeves, it has not been possible to establish for the Polak-Ribière method, the global convergence results obtained for the Fletcher-Reeves by Powell (1983) [70] and Al-Baali (1985) [58]. But when some additional conditions are imposed such as f is a convex function or the step lengths $\|x^{(k+1)} - x^{(k)}\|$ tend to zero, the global convergence of the Polak-Ribière method with exact line search can be obtained. Powell (1977) [55] established the following theorem:

Theorem 2.5.1 (Powell 1977) [55]

If the level set (2.4.9) is bounded, if f is continuously differentiable and if the step lengths $\|x^{(k+1)} - x^{(k)}\|$ tend to zero, then the Polak-Ribière method without restart achieves the limit (2.4.12).

Until 1983, it was not known whether either of the Fletcher Reeves or the Polak-Ribière methods provide the limit (2.4.12) for a twice continuously differentiable function with bounded level set from an arbitrary starting point. But with the help of theorem 2.5.1 and some conditions developed by Powell (1977) [55], it is straight forward to show for the Polak-Ribière method that if the sequence $\{x^{(k)}, k = 1, 2, \dots\}$ converges to a limit x^* say, then $\nabla f(x^*) = 0$. Hence, one could think that establishing (2.4.12) would be easier for the Polak-Ribière method than for that of Fletcher-Reeves. Powell (1983) [70] came out with the surprising negative results. Not only has he shown by a standard method of proof that the limit (2.4.12) is always achieved by the Fletcher Reeves method a result that has been extended by Al-Baali (1985) [58] for the case of inexact line search, but he also found that if the Polak-Ribière method is used, then with exact arithmetic and an exact line search, there exists a twice continuously differentiable function with bounded level set for which the gradient $\{\|g\|, k = 1, 2, \dots\}$ are bounded away from zero.

2.6 Hybrid Conjugate Gradient Methods

Although the superiority of Polak-Ribière method in numerical computations over the Fletcher-Reeves in most cases has been observed it has not been possible to establish global convergence results for the Polak-Ribière method as was done for the Fletcher-Reeves. Furthermore as it was mentioned before powell (1983) [70] shows that if $\beta^{(k)}$ is chosen to satisfy (2.5.1), then even

with exact line search and exact arithmetic, there exist twice continuously differentiable functions with bounded level sets (2.4.9) for which the gradients norms $\{\|g^{(k)}\|, k = 1, 2, \dots\}$ are bounded away from zero. However, it was found that if the Polak-Ribière $\beta^{(k)}$ are restricted to remain non negative and at the same time less than or equal to the Fletcher-Reeves $\beta^{(k)}$, then the convergence proofs given by Powell (1983) [70] and by Al-Baali (1985) [58] both apply to the Polak-Ribière method also, but these restrictions are unfortunately not always satisfied.

All this has consequently led to thoughts by D. Touati Ahmed and C Storey (1986), [72], (1987) [18], on how to combine the desirable computational aspects of the Polak-Ribière method and the useful theoretical features of the Fletcher-Reeves method in an attempt to construct some efficient hybrid algorithms that are globally convergent.

2.6.1 Hybrid 1 algorithm

The new hybrid algorithms were developed by D Touati-Ahmed and C Storey (1990) [61]. Let $\beta^{(k)}$ (F.R) and $\beta^{(k)}$ (P.R) denote the betas satisfying (2.4.8) and (2.5.1) respectively. We then have:

$$\beta^{(k)}(P.R) = \beta^{(k)}(F.R) - g^{(k+1)T} g^{(k)} / \|g^{(k)}\|^2 \quad (2.6.1)$$

If $\beta^{(k)}$ (P.R) were always non-negative, we would have

$$\begin{aligned} \beta^{(k)}(P.R) \geq 0 &\implies \beta^{(k)}(F.R) \geq g^{(k+1)T} g^{(k)} / \|g^{(k)}\|^2 \\ &\implies \|g^{(k+1)}\|^2 \geq g^{(k+1)T} g^{(k)} \end{aligned}$$

and if $\beta^{(k)}$ (P.R) were always less than or equal to $\beta^{(k)}$ (F.R) we would have

$$\begin{aligned} \beta^{(k)}(P.R) \leq \beta^{(k)}(F.R) &\implies -g^{(k+1)T} g^{(k)} / \|g^{(k)}\|^2 \leq 0 \\ &\implies g^{(k+1)T} g^{(k)} \geq 0 \end{aligned} \quad (2.6.2)$$

Hence the restrictions that imposed on the Polak-Ribière beta will hold if

$$0 \leq g^{(k+1)T} g^{(k)} \leq \|g^{(k+1)}\|^2 \quad (2.6.3)$$

As consequence of this remark the first hybrid algorithm, Hybrid 1, was suggested, using formula (2.5.1) whenever condition (2.6.3) is satisfied and formula (2.4.8) otherwise. In other words the $\beta^{(k)}$ is defined as follows:

$$\beta^{(k)} = \left. \begin{aligned} &g^{(k+1)T} (g^{(k+1)} - g^{(k)}) / \|g^{(k)}\|^2 && \text{if (2.6.3) is true} \\ &\|g^{(k+1)}\|^2 / \|g^{(k)}\|^2 && \text{otherwise} \end{aligned} \right\} \quad (2.6.4)$$

2.6.2 Descent property and global convergence in the case of exact line search

It has already been seen (2.4.11) that if the line search is exact then the descent property holds on all iterations for both Fletcher Reeves and Polak-Ribière methods, so it is obvious that it holds also for Hybrid 1 method.

Now for the global convergence of Hybrid 1 the following theorem was established by D. Touati-Ahmed and C. Storey (1986) [72].

Theorem 2.6.1 (D Touati-Ahmed and C Storey (1986) [72])

If $f(x)$ is twice continuously differentiable, if the level set (2.4.9) is bounded and if an exact line search is performed at each iteration, then the limit (2.4.12) is achieved by Hybrid 1.

2.6.3 Descent property and global convergence in the case of inexact line search

Hybrid 1 can also be shown to have the descent property and to be globally convergent when an inexact line search is performed. The following theorems established by D Touati-Ahmed and C Storey confirms this

Theorem 2.6.2: (D Touati-Ahmed and C Storey (1986) [72]).

If an $\alpha^{(k)}$ is calculated which satisfies (2.4.4) with $\sigma \in (0, \frac{1}{2}]$ for all k such that $g^{(k)} \neq 0$, then the descent property (2.4.6) for Hybrid 1, holds for all such k .

A consequence of this descent property is the following global convergent result.

Theorem 2.6.3: (D Touati-Ahmed and C Storey (1986) [72]).

If the set (2.4.9) is bounded, if $f(x)$ is twice continuously differentiable, and if $\alpha^{(k)}$ is any value satisfying (2.4.4) and (2.4.5) with $\rho < \sigma < \frac{1}{2}$, then equation (2.4.12) holds for Hybrid 1.

2.7 Angle Test Hybrid

Shanno (1985) [73] gives an angle test to determine when conjugate gradient algorithms should be restarted with a steepest descent direction. Using the fact that the Fletcher-Reeves algorithm is globally convergent when exact line searches are used we have:

$$\sum_k \cos^2(\theta^{(k)}) = \sum_k \frac{1}{\|g^{(k)}\|^2 \sum_{\ell=1}^k \|g^{(\ell)}\|^{-2}}$$

is a divergent series. Obviously if $\sum_k \cos^2(\theta^{(k)})$ is divergent then for any $\tau > 0$, $\tau \sum_k \cos^2(\theta^{(k)})$ is divergent. Thus for any $\tau > 0$, $\gamma^{(k)}$ is defined by:

$$\gamma^{(k)^2} = \frac{\tau}{\|g^{(k)}\|^2 \sum_{\ell=1}^k \|g^{(\ell)}\|^{-2}} \quad (2.7.1)$$

Shanno's algorithm is then to use any conjugate gradient formula and restart the algorithm with a steepest descent direction whenever

$$\cos^2(\theta^{(k)}) \geq \gamma^{(k)^2} \quad (2.7.2)$$

is not satisfied, where

$$\cos(\theta^{(k)}) = -g^{(k)T} s^{(k)} / \|g^{(k)}\| \|s^{(k)}\| \quad (2.7.3)$$

A possible hybrid algorithm (the angle test hybrid) would be to use this test for every $\beta^{(k)}$ (P.R) for which condition (2.6.3) is not satisfied. D Touati-Ahmed and C Storey (1987) [18] suggested the use of the following algorithm to compute $\beta^{(k)}$ in (2.4.7).

$$\left. \begin{array}{l} \text{Step 1: If } \beta^{(k)}(P.R) < 0, \text{ then } \beta^{(k)} = \beta^{(k)}(F.R), \text{ return to} \\ \text{main program. Otherwise, go to Step 2.} \\ \text{Step 2: if (2.6.3) holds, then } \beta^{(k)} = \beta^{(k)}(P.R), \\ \text{return to main program. Otherwise, go to Step 3.} \\ \text{Step 3: If (2.7.2) holds, then } \beta^{(k)} = \beta^{(k)}(P.R), \text{ return} \\ \text{to main program. Otherwise set } \beta^{(k)} = \beta^{(k)}(F.R), \text{ return} \\ \text{to main program.} \end{array} \right\} \quad (2.7.4)$$

When exact line searches were performed throughout the algorithm, it can be shown that a descent property holds and that the method is globally convergent. But when inexact line searches are performed, condition (2.7.2) can no be used directly to ensure a descent property with which it would be possible to prove global convergence.

2.8 Hybrid 3

A further hybrid algorithm (Hybrid 3 algorithm) was proposed by D Touati-Ahmed and C Storey (1990) [61]. For the purpose of what follows consider

$$\beta^{(k)}(P.R) \leq \frac{1}{2\mu} \frac{\|g^{(k+1)}\|^2}{\|g^{(k)}\|^2}, \quad \mu > \sigma \quad (2.8.1)$$

where σ is the constant in the line search.

If at each iteration we ensure that:

$$\lambda \|g^{(k+1)}\|^2 \leq (2\mu)^{k+1}, \quad \frac{1}{2} > \mu > \sigma \quad (2.8.2)$$

for some $\lambda > 0$, then the method will be globally convergent. In other words if algorithm (2.7.4) with condition (2.7.2) replaced by (2.8.1), whenever (2.8.2) is satisfied and restart with a steepest descent direction otherwise, the descent property (2.4.6) holds and the limit (2.4.12) is achieved.

Consequently the use of the following algorithm to compute $\beta^{(k)}$ in (2.4.7): was suggested as Hybrid 3 algorithm:

$$\left. \begin{array}{l} \text{Step 1: If (2.8.2) holds go to Step 2. Otherwise } \beta^{(k)} = 0, \\ \text{return to main program.} \\ \text{Step 2: If } \beta^{(k)}(P.R) < 0, \text{ then } \beta^{(k)} = \beta^{(k)}(F.R), \text{ return to main} \\ \text{program. Otherwise go to step 3.} \\ \text{Step 3: If (2.8.1) holds, then } \beta^{(k)} = \beta^{(k)}(P.R), \text{ return to main} \\ \text{program. Otherwise } \beta^{(k)} = \beta^{(k)}(F.R), \text{ return to main program.} \end{array} \right\} \quad (2.8.3)$$

It should be noted that step 2 of algorithm (2.7.4) has been missed out because the set of numbers that satisfy (2.8.1) includes those that satisfy (2.6.3) and therefore it is not necessary to check whether (2.6.3) is satisfied.

This is so of course, because $\mu < \frac{1}{2}$. The case $\mu \geq \frac{1}{2}$ is covered in Hybrid 1.

2.8.1 Descent property and global convergence

It was shown by D Touati Ahmed and C Storey (1990) [61] that when Hybrid 3 algorithm is used to find the least value of a twice continuously differentiable function $f(x)$ with bounded level set (2.4.9), a descent property holds on all iterations and the limit (2.4.12) is achieved. The following theorems confirms this when an exact line search and when an inexact line search satisfying (2.4.4) and (2.4.5) are used respectively.

Theorem 2.8.1 (D Touati-Ahmed and C Storey (1990) [61]).

If $f(x)$ is twice continuously differentiable, if the level set (2.4.9) is bounded and if $\alpha^{(k)}$ satisfies (2.4.3) on all iterations then the limit (2.4.12) is achieved by Hybrid 3.

Theorem 2.8.2 (D Touati-Ahmed and C Storey (1990) [61]).

If an $\alpha^{(k)}$ is calculated which satisfied (2.4.4) with $\alpha \in (0, \frac{1}{2})$ for all k such that $g^{(k)} \neq 0$, then the descent property (2.4.6) for Hybrid 3, holds for all such k .

2.9 Infinite Dimensional Problems

Here we are concerned with functionals defined on an infinite-dimensional space. The classic example of this is the simplest problem of the calculus of variations, where in we minimize

$$F(x) = \int_a^b L(t, x, x') dt; \quad x(a), x(b) \text{ fixed}, \quad (2.9.1)$$

$$x(t) \in \underline{c}[a, b] = \underline{x}$$

and \underline{c} is the linear space of functions having piecewise continuous derivatives, defined on the closed interval $[a, b]$. Considering that x is determined given $x(0)$ and $x'(t)$, an equivalent formulation is:

$$\begin{aligned} \min F(x, v) &= \int_a^b L(x, v, t) dt \\ x' &= v, x(a), x(b) \text{ fixed} \\ \text{and } v &\in \underline{c}_0[a, b] = \underline{V} \end{aligned} \quad (2.9.2)$$

where \underline{c}_0 is the space of piecewise continuous functions on $[a, b]$. This second formulation can be interpreted as a functional defined on the product space $\underline{x} \times \underline{V}$, minimized subject to the equality constraint $x' = v$. Alternatively, since the constraint in fact determines x uniquely given V , we may regard F as a functional defined on \underline{V} alone. In either case, an additional constraint exists on $x(b)$. These are called terminal constraints and may in general be of the form $\psi(x, b) = 0$. The 'constraint' $x' = v$ may be regarded as a special case of $x' = f(t, x, v)$. Other conditions of the form $h(v(t)) \leq 0, h(x(t), V(t), t) < 0$ may also exist, where h is either a set of functional defined on $\underline{x} \times \underline{V}$, or a set of ordinary functions of x, V which, evaluated at any $t \in [a, b]$, satisfy the inequality. Combinations of these may also exist.

The class of problems considered here will not include explicit statement of inequality constraints $h \leq 0$. these may of course be included by penalty

functions. Moreover, we allow x, V to be functions that take values in ordinary Euclidean spaces $\underline{E}^n, \underline{E}^m$ respectively. Such problems are sometimes called finite dimensional control problems [74]. In this terminology, infinite-dimensional problems are those involving partial differential equations, i.e. distributed parameter systems.

2.9.1 Theoretical methods

Historically, such problems were first treated about the time the calculus was invented, when the brachistochrone problem was proposed by Jacobi Bernoulli and solved by Newton. An account of the early development of the calculus of variations is given by Bliss (1936) [75]. The first systematic necessary conditions were found by Euler and the subject approached relative completeness with the theory of the problem of Bolza as developed by Bliss and co-workers at the university of Chicago [76], at about 1950. This theory is founded on the theory of differential equations and was able to cope only partly with the demands placed on it by rapidly increasing applications in the computer age. A control theory viewpoint was taken by Pontryagin and co-workers, leading to the new widely used maximum principle [76]. While this principle is similar to the multiplier rules found by the Chicago school, in particular the work of McShane (1939) [77], it was able to solve problems with control constraints and discontinuous controls more rapidly. An even more general approach was taken by Bellman in his method of dynamic programming [78], formulated in terms of a multistage decision process and a 'principle of optimality'. This approach was of great value in treating discrete, stochastic and highly constrained problems not amenable to older techniques. Later connections between the variational calculus, Pontryagin's principle and dynamic programming are numerous and are explored, for example by Dreyfus [79] and Hestenes [80].

2.10 The Conjugate Gradient Method for Optimal Control Problems

In this section we consider the extension of the conjugate gradient minimization method of Fletcher-Reeves to optimal control problems. The technique is directly applicable only to those problems, where terminal conditions and inequality constraints are not present. If such constraints are present the problem must be converted to an unconstrained form, e.g. by penalty functions, [81] to [84]. Only the gradient trajectory, its norm, and one additional trajectory, the actual direction of search, need to be stored. These search directions are generated from past and present values of the objective and its gradient. Successive points are determined by linear minimization down these directions, which are always directions of descent.

Thus the method tends to converge, even from poor approximations to the minimum. Since, near it minimum, a general non linear problem can be approximated by one with a linear system and quadratic objective. The rate

of convergence is studied by considering this case. Here, the directions of search are conjugate and hence the objective is minimized over an expanding sequence of sets. Also, the distance from the current point to the minimum is reduced at each step. It is evident that the efficiency of these methods depends greatly on the technique used to solve the unconstrained optimal control problem.

Presently available techniques all have short comings. The convergence of steepest descent methods is often slow [81] whereas second-variational and Newton methods may not converge at all. Thus there is strong motivation for developing more efficient means for solving unconstrained optimal control problems.

Similar difficulties existed until the late 1960's in the field of finite dimensional optimization, i.e. mathematical programming. However, later several rapidly convergent finite dimensional unconstrained minimization techniques have been developed. Among these are the method of Fletcher and Powell (1963) [85] and the Fletcher Reeves (1964) [48] adaption of the conjugate gradient method of Hestenes and Stiefel (1952) [47].

The combination of these properties implies that the methods converge rapidly to the nearest local minimum for a general function of n variables. Experience has shown that both techniques converge much more rapidly, in general, than the method of steepest descent while requiring only function and gradient evaluation.

Function space analogs of the steepest descent and second-order Newton techniques have been developed and applied to problems of optimal control. In particular Kelly (1960) [86], McReynolds and Bryson (1965) [87] and Mitter (1966) [88], and others have developed steepest descent and second order methods. Since these methods are considered by some authors [85], [89] to be the most powerful presently available for finite dimensional minimization problems, it seems appropriate to consider their generalization to optimal control.

2.10.1 The algorithm for conjugate gradient method

Consider the following problem:

$$\text{minimize } J = \phi((x(t_f))) \quad (2.10.1)$$

$$\text{subject to } \dot{x} = f(x, u, t) \quad (2.10.2)$$

$$x(t_0) = c \quad (2.10.3)$$

where x is an n vector, u is an m vector, and t_0, t_f are fixed. It is assumed that given a control u , (2.10.2) and (2.10.3) can be solved for a unique $x = x(u)$, and thus $J = J(u)$ is a function of u alone. Furthermore, the existence of the gradient of $J(u)$, $\nabla J(u) = g(u)$ is assumed. The objective function (2.10.1) may include penalty function terms to account for constraints.

Here for the convenience only the case of a scalar control function $u(t)$ ($m = 1$) will be considered.

The extension to the multicontrol case is straightforward. In order to compute the conjugate gradient algorithm we need the gradient trajectory.

Let

$$H = \sum_{i=1}^n \lambda_i f_i \quad (2.10.4)$$

where

$$\dot{\lambda}_i = - \sum_{j=1}^n \lambda_j \frac{\partial f_j}{\partial x_i} \quad (2.10.5)$$

$$\lambda_i(t_f) = \frac{\partial \phi}{\partial x_i} \Big|_{t=t_f} \quad i = 1, \dots, n. \quad (2.10.6)$$

Then the gradient is

$$g(u) = \frac{\partial H}{\partial u}. \quad (2.10.7)$$

Let $u_i(t)$ be the i th approximation to the optimal control $u^0(t)$. The corresponding gradient $g(u_i)$ is computed by solving the state equations (2.10.2) and (2.10.3) forwards with $u = u_i$, solving the adjoint equations (2.10.5) and (2.10.6) backwards and then computing $g(u_i)$ from (2.10.7).

The algorithm proceeds as follows:

$$u_0 = \text{arbitrary} \quad (2.10.8)$$

$$g_0 = g(u_0) \quad (2.10.9)$$

$$s_0 = -g_0 \quad (2.10.10)$$

choose

$$\alpha = \alpha_i \text{ to minimize } J(u_i + \alpha s_i) \quad (2.10.11)$$

and then

$$u_{i+1} = u_i + \alpha s_i \quad (2.10.12)$$

$$g_{i+1} = g(u_{i+1}) \quad (2.10.13)$$

Now β_i can be computed as

$$\beta_i = (g_{i+1}, g_{i+1}) / (g_i, g_i) = \frac{\|g_{i+1}\|^2}{\|g_i\|^2} \quad (2.10.14)$$

which is Fletcher-Reeves or

$$\beta_i = \frac{(g_{i+1}, g_{i+1})}{(g_i, g_i)} - \frac{(g_{i+1}, g_i)}{(g_i, g_i)} \quad (2.10.15)$$

which is Polak-Ribière.

In the case of Hybrid methods β_i can be computed as (2.6.4) for Hybrid 1, (2.7.4) for angle test hybrid and (2.8.3) for hybrid 3.

$$s_{i+1} = -g_{i+1} + \beta_i s_i \quad (2.10.16)$$

where the norms $\|g_i\|^2$ and $\|g_{i+1}\|^2$ can be calculated as follows:

$$(g_i, g_j) = \int_{t_0}^{t_f} g_i(t) g_j(t) dt \quad (2.10.17)$$

we should note that the new direction of search s_{i+1} is not the negative direction $-g_{i+1}$ but is computed via (2.10.16). The step length in this direction is determined by one dimensional minimization in (2.10.11).

2.10.2 Convergence

Let that control u be an element of a Hilbert space H and $J(u)$ a Frechet differentiable mapping from H to the real numbers. The conjugate gradient method when applied to $J(u)$ generates directions s_i which are always directions of descent, i.e.

$$\frac{d}{d\alpha} J(u_i + \alpha s_i)|_{\alpha=0} < 0 \quad (2.10.18)$$

and this ensures that $J(u)$ is decreased at each step. The following theorems state this fact and the proofs can be found in L.S. Lasdon, S.K. Mitter, and A.D. Waves (1967) [90].

Theorem 2.10.1

If $g(u_i) = g_i \neq 0$ then

$$(s_i, g_{i+1}) = 0$$

and $\frac{d}{d\alpha} J(u_i + \alpha s_i)|_{\alpha=0} = (s_i, g_i) = -\|g_i\|^2$.

Theorem 2.10.2

If $g_i \neq 0$ then $j(u_{i+1}) < j(u_i)$.

The sequence of real numbers $\{J(u_i)\}$ is thus monotone decreasing and therefore has a limit J_∞ in the extended real numbers. Also of interest is the limiting behaviour of the sequences $\{u_k\}$ and $\{g_k\}$. Results similar to those that have been obtained for the method of steepest descent [91], [92] are given below.

Theorem 2.10.3

If the following assumptions are made

1. $J(u)$ is bounded below
2. $J(u)$ and $g(u)$ are continuous
3. $D^2 J(u, h, h)$ exists and

$$|D^2 J(u, h, h)| \leq m \|h\|^2, \quad h \in H, \text{ and } m > 0$$

4. $\{u_k\}$ has a cluster point u^*

Then the sequence $\{u_k\}$ formed with arbitrary u_0 by the conjugate gradient method has the following properties:

- (a) $\lim_{k \rightarrow \infty} J(u_k) = J(u^*)$
- (b) $\lim_{k \rightarrow \infty} g(u_k) = g(u^*) = 0$.

By assumption 4, $\{u_k\}$ contains a convergent subsequence $\{\hat{u}_k\}$, with limit point u^* . Then continuity of $g(u)$ implies that $g(u^*) = 0$. From Theorem 2.10.2, $J(u_{k+1}) < J(u_k)$ and hence by the convergence of u_k and continuity of $J(u)$, property 1 follows.

Computational experience has shown that methods which decrease the function J at each step will generally converge to the nearest local minimum. Since the function is generally convex in some neighbourhood of a local minimum. This statement is supported by the following result.

Theorem 2.10.4

If the assumptions of theorem (2.10.3) hold and if the following assumption is made:

$$D^2 J(u, h, h) \geq M \|h\|^2 \quad \underset{u, h \in H}{M} > 0,$$

then

$$J(u^*) = \min_{u \in H} J(u).$$

Chapter 3

NUMERICAL CONSIDERATION

3.1 Numerical Computations

In this chapter we consider the numerical work involved in solving the differential equations, calculating the norms and also the linear search for the problems in Chapters 4 to 9. The importance of numerical stability brings about the need of selecting accurate methods for numerical integrations. Therefore, in this chapter we briefly review some of the well-known methods that are useful.

3.2 Numerical Integration Methods

For the problems being considered in the following Chapters we need to solve the state and adjoint equations using a suitable integration routine, also an appropriate technique is needed to find the norm described in Chapter 2 Section 2.10.1, Equation (2.10.17). In the coming Sub Sections we briefly review the Runge Kutta method and also Simpson's Rule.

3.2.1 Runge Kutta Methods

Consider the initial value problem

$$y' = f(x, y), \quad y(a) = \eta. \quad (3.2.1)$$

The easiest of all computational methods for the numerical solution of this problem to implement is Euler's rule,

$$y_{n+1} - y_n = hf(x_n, y_n) \equiv hf_n. \quad (3.2.2)$$

It is explicit and, being a one step method, it requires no additional starting values and easily allows a change of step-length during the computation. Being low order, of course, makes it of limited practical value. Linear multistep

methods achieve higher order by sacrificing the one-step nature of the algorithm, whilst retaining linearity with respect to y_{n+j} , f_{n+j} , $j = 0, 1, \dots, k$.

Higher order methods can also be achieved by sacrificing linearity, but preserving the one step nature of the algorithm. This is the philosophy behind the methods first proposed by Runge (1895) [93] and subsequently developed by Kutta (1901) [94] and Heun (1900) [95]. Runge-Kutta methods thus retain the advantages of one-step methods but, due to the loss of linearity, error analysis is considerably more difficult than in the case of linear multi-step methods. Traditionally, Runge-Kutta methods are all explicit although since the early 1970's, implicit Runge-Kutta methods, which have improved weak stability characteristics, have been considered. Hence we shall use the expression 'Runge-Kutta method' to mean 'explicit Runge-Kutta method'.

Thus a Runge-Kutta method may be regarded as a particular case of the general explicit one-step method

$$y_{n+1} - y_n = h\phi(x_n, y_n, h). \quad (3.2.3)$$

3.2.2 Order and Convergence of the General Explicit One-step Method

The fact that the general method (3.2.3) makes no mention of the function $f(x, y)$, which defines the differential equation, makes it impossible to define the order of the method independently of the differential equation, as is the case with linear multi step methods.

Definition 3.2.1

The method (3.2.3) is said to have order p if p is the largest integer for which

$$y(x+h) - y(x) - h\phi(x, y(x), h) = O(h^{p+1}) \quad (3.2.4)$$

holds, where $y(x)$ is the theoretical solution of the initial value problem.

Definition 3.2.2

The method (3.2.3) is said to be consistent with the initial value problem if

$$\phi(x, y, 0) \equiv f(x, y). \quad (3.2.5)$$

If the method (3.2.3) is consistent with the initial value problem (for future reference we shall simply say 'consistent'), then

$$y(x+h) - y(x) - h\phi(x, y(x), h) = hy'(x) - h\phi(x, y(x), 0) + O(h^2) = O(h^2),$$

since $y'(x) = f(x, y(x)) = \phi(x, y(x), 0)$, by (3.2.5). Thus a consistent method has order at least one.

The only linear multistep method which falls within the class (3.2.3) is Euler's rule which we obtain by setting

$$\phi(x, y, h) = \phi_E(x, y, h) \equiv f(x, y).$$

(The subscript E denotes 'Euler':) The consistency condition (3.2.5) is then obviously satisfied and a simple calculation shows that the order, according to (3.2.4), is one.

The Taylor algorithm of order p also falls within the class (3.2.3) and is obtained by setting

$$\begin{aligned} \phi(x, y, h) = \phi_T(x, y, h) \equiv & f(x, y) + \frac{h}{2!} f^{(1)}(x, y) + \\ & \dots + \frac{h^{p-1}}{p!} f^{(p-1)}(x, y), \end{aligned} \quad (3.2.6)$$

where $f^{(q)}(x, y) \equiv \frac{d^q}{dx^q} f(x, y)$, $q = 1, 2, \dots, (p-1)$.

(The subscript T denotes 'Taylor'.)

The following Theorem, whose proof may be found in Henrici (1962) [96], states conditions on $f(x, y)$ which guarantee the existence of a unique solution of the initial value problem (3.2.1).

Theorem 3.2.1

Let $f(x, y)$ be defined and continuous for all points (x, y) in the region D defined by $a \leq x \leq b$, $-\infty < y < \infty$, a and b finite and let $f(x, y)$ satisfy Lipschitz condition, i.e., there exists a constant L such that, for every x, y, y^* such that (x, y) and (x, y^*) are both in D ,

$$|f(x, y) - f(x, y^*)| \leq L|y - y^*|. \quad (3.2.7)$$

Then, if η is any given number, there exists a unique solution $y(x)$ of the initial value problem (3.2.1), where $y(x)$ is continuous and differentiable for all (x, y) in D .

The following theorem, whose proof may also be found in Henrici (1962) [96], states necessary and sufficient conditions from the method (3.2.3) to be convergent.

Theorem 3.2.2

(i) Let the function $\phi(x, y, h)$ be continuous jointly as a function of its three arguments, in the region \mathcal{D} defined by $x \in [a, b]$, $y \in (-\infty, \infty)$, $h \in [0, h_0]$, $h_0 > 0$.

(ii) Let $\phi(x, y, h)$ satisfy a Lipschitz condition of the form

$$|\phi(x, y^*, h) - \phi(x, y, h)| \leq M|y^* - y|$$

for all points (x, y^*, h) , (x, y, h) in \mathcal{D} .

Then the method (3.2.3) is convergent of and only if it is consistent. For all the Runge-Kutta methods we shall consider, in the next sections condition (i) and (ii) are satisfied if $f(x, y)$ satisfies the conditions stated in theorem 3.2.1. For such methods consistency is necessary and sufficient for convergence.

We should note that there is no requirement corresponding to zero-stability, since no parasitic solutions can arise with a one-step method.

3.2.3 Algorithms for Runge-Kutta Methods

The general R -stage Runge-Kutta method is defined by

$$y_{n+1} - y_n = h\phi(x_n, y_n, h), \quad (3.2.8)$$

$$\left. \begin{aligned} \phi(x, y, h) &= \sum_{r=1}^R c_r k_r \\ k_1 &= f(x, y) \\ k_r &= f\left(x + ha_r, y + h \sum_{s=1}^{r-1} b_{rs} k_s\right), \quad r = 2, 3, \dots, R, \end{aligned} \right\} \quad (3.2.9)$$

$$a_r = \sum_{s=1}^{r-1} b_{rs}, \quad r = 2, 3, \dots, R. \quad (3.2.10)$$

Note that an R -stage Runge-Kutta method involves R function evaluation per step. Each of the functions $k_r(x, y, h)$, $r = 1, 2, \dots, R$, may be interpreted as an approximation to the derivative $y'(x)$, and the function $\phi(x, y, h)$ as a weighted mean of these approximations. We should also note that consistency demands that $\sum_{r=1}^R c_r = 1$. If we can choose values for the constants c_r, a_r, b_{rs} such that the expansion of the function $\phi(x, y, h)$ defined by (3.2.9) in powers of h differs from the expansion for $\phi_T(x, y, h)$ given by (3.2.6) only in the p th and higher powers of h , then the method clearly has order p . (Note that in (3.2.6) we are assuming that $y(x) \in C^p[a, b]$.)

Here we quote some of the well known Runge-Kutta methods. For the derivation and further information on Runge-Kutta methods of 3rd order and higher we could refer to Lambert, J. D. (1986) [97]. Butcher (1963-1965) [98] to [103], and Huta, (1957) [104].

Algorithm 3.2.1

The following formula is known as Henn's third order formula which is one of the well known third-order Runge-Kutta methods.

$$\left. \begin{aligned} y_{n+1} - y_n &= \frac{h}{4}(k_1 + 3k_3), \\ k_1 &= f(x_n, y_n), \\ k_2 &= f(x_n + \frac{1}{3}h, y_n + \frac{1}{3}hk_1), \\ k_3 &= f(x_n + \frac{2}{3}h, y_n + \frac{2}{3}hk_2). \end{aligned} \right\} \quad (3.2.11)$$

Hence k_r denotes the function $k_r(x_n, y_n, h)$ and h is integration step length.

Algorithm 3.2.2

The following method is known as Kutta's third order rule.

$$\left. \begin{aligned} y_{n+1} - y_n &= \frac{h}{6}(k_1 + 4k_2 + k_3), \\ k_1 &= f(x_n, y_n), \\ k_2 &= f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1), \\ k_3 &= f(x_n + h, y_n - hk_1 + 2hk_2). \end{aligned} \right\} \quad (3.2.12)$$

It is the most popular third-order Runge-Kutta method for desk computation, mainly because the coefficient $\frac{1}{2}$ is preferable to $\frac{1}{3}$, which appears frequently in (3.2.11).

Algorithm 3.2.3

The following formula is known as a Runge-Kutta fourth-order method.

$$\left. \begin{aligned} y_{n+1} - y_n &= \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \\ k_1 &= f(x_n, y_n), \\ k_2 &= f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1), \\ k_3 &= f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2), \\ k_4 &= f(x_n + h, y_n + hk_3). \end{aligned} \right\} \quad (3.2.13)$$

This method is undoubtedly the most popular of all Runge-Kutta methods. Indeed it is frequently referred to, loosely as even 'The Runge-Kutta Method'.

Algorithm 3.2.4

The following method is also known as a fourth-order Runge-Kutta method.

$$\left. \begin{aligned} y_{n+1} - y_n &= \frac{h}{8}(k_1 + 3k_2 + 3k_3 + k_4), \\ k_1 &= f(x_n, y_n), \\ k_2 &= f(x_n + \frac{1}{3}h, y_n + \frac{1}{3}hk_1), \\ k_3 &= f(x_n + \frac{2}{3}h, y_n - \frac{1}{3}hk_1 + hk_2), \\ k_4 &= f(x_n + h, y_n + hk_1 - hk_2 + hk_3). \end{aligned} \right\} \quad (3.2.14)$$

The derivation of fourth-order Runge-Kutta methods involves tedious manipulation, it transpires that with $R = 4$, fourth order and no higher, can be obtained. Full details may be found in Ralston (1965) [105].

Algorithm 3.2.5

The following formula is the fifth-order Kutta-Nyström method which is a six-stage method.

$$\left. \begin{aligned} y_{n+1} - y_n &= \frac{h}{192}(23k_1 + 125k_3 - 81k_5 + 125k_6), \\ k_1 &= f(x_n, y_n), \\ k_2 &= f(x_n + \frac{1}{3}h, y_n + \frac{1}{3}hk_1), \\ k_3 &= f(x_n + \frac{2}{5}h, y_n + \frac{1}{25}h(4k_1 + 6k_2)), \\ k_4 &= f(x_n + h, y_n + \frac{1}{4}h(k_1 - 12k_2 + 15k_3)), \\ k_5 &= f(x_n + \frac{2}{3}h, y_n + \frac{1}{81}h(6k_1 + 90k_2 - 50k_3 + 8k_4)), \\ k_6 &= f(x_n + \frac{4}{5}h, y_n + \frac{1}{75}h(6k_1 + 36k_2 + 10k_3 + 8k_4)). \end{aligned} \right\} \quad (3.2.15)$$

Algorithm 3.2.6

The following formula is the six-order eight-stage method.

$$\left. \begin{aligned} y_{n+1} - y_n &= \frac{h}{840}(41k_1 + 216k_3 + 27k_4 + 272k_5 + 27k_6 + 216k_7 + 41k_8), \\ k_1 &= f(x_n, y_n), \\ k_2 &= f(x_n + \frac{1}{9}h, y_n + \frac{1}{9}hk_1), \\ k_3 &= f(x_n + \frac{1}{6}h, y_n + \frac{1}{24}h(k_1 + 3k_2)), \\ k_4 &= f(x_n + \frac{1}{3}h, y_n + \frac{1}{6}h(k_1 - 3k_2 + 4k_3)), \\ k_5 &= f(x_n + \frac{1}{2}h, y_n + \frac{1}{8}h(-5k_1 + 27k_2 - 24k_3 + 6k_4)), \\ k_6 &= f(x_n + \frac{2}{3}h, y_n + \frac{1}{9}h(221k_1 - 981k_2 + 867k_3 - 102k_4 + k_5)), \\ k_7 &= f(x_n + \frac{5}{6}h, y_n + \frac{1}{48}h(-183k_1 + 678k_2 - 472k_3 - 66k_4 + 80k_5 + 3k_6)), \\ k_8 &= f(x_n + h, y_n + \frac{1}{82}h(716k_1 - 2079k_2 + 1002k_3 + 834k_4 - 454k_5 - 9k_6 + 72k_7)). \end{aligned} \right\} \quad (3.2.16)$$

The derivation of the sixth-order eight-stage method could be found in Huřa (1957) [104].

3.2.4 Error Bounds for Runge-Kutta Methods.

In the previous section we mentioned only the order of an explicit one-step method and no mention of its truncation error.

Definition 3.2.3

The local truncation error at x_{n+1} of the general explicit one step method (3.2.3) is defined to be T_{n+1} where

$$T_{n+1} = y(x_{n+1}) - y(x_n) - h\phi(x_n, y(x_n), h) \quad (3.2.17)$$

and $y(x)$ is the theoretical solution of the initial value problem. If we make the assumption about (3.2.3) that no previous errors have been made (namely that $y_n = y(x_n)$) then, from (3.2.17) and (3.2.3) it follows that

$$T_{n+1} = y(x_{n+1}) - y_{n+1}.$$

Thus the truncation error defined by (3.2.17) is local. We can define the global truncation error e_{n+1} as $e_{n+1} = y(x_{n+1}) - y_{n+1}$, where now it is no longer assumed that no previous truncation errors have been made.

If $y(x)$ is assumed to be sufficiently differentiable, the local truncation error for the non-linear method (3.2.3) of order p can be written in the form

$$T_{n+1} = \Psi(x_n, y(x_n))h^{p+1} + O(h^{p+2}), \quad (3.2.18)$$

where we shall call $\Psi(x, y)$ the principal error function, and $\Psi(x_n, y(x_n))h^{p+1}$ the principal local truncation error.

Consider, for example, the general two-evaluation Runge-Kutta method obtained by setting $R = 2$, in [(3.2.8) to (3.2.10)], the local truncation error can be obtained as

$$T_{n+1} = h \left[f + \frac{1}{2} h F + \frac{1}{6} h^2 (F f_y + G) \right]_{\substack{x=x_n \\ y=y(x_n)}} \\ - h [(c_1 + c_2) f + h c_2 a_2 F + \frac{1}{2} h^2 c_2 a_2^2 G]_{\substack{x=x_n \\ y=y(x_n)}} + O(h^4),$$

where

$$F = f_x + f f_y, \quad G = f_{xx} + 2f f_{xy} + f^2 f_{yy}.$$

If the order is two, then we obtain

$$T_{n+1} = h^3 \left[\frac{1}{6} F f_y + \left(\frac{1}{6} - \frac{1}{4} a_2 \right) G \right]_{\substack{x=x_n \\ y=y(x_n)}} + O(h^4).$$

For more detail we can refer to Lambert (1986) [97]. Thus the principal error function for the general second-order Runge-Kutta method is given by

$$\Psi(x, y) = \frac{1}{6} F f_y + \left(\frac{1}{6} - \frac{1}{4} a_2 \right) G. \quad (3.2.19)$$

Following an argument originally proposed by Lotkin (1951) [106] we can find a bound for $\Psi(x, y)$, if we assume that the following bounds for f and its partial derivatives hold for $x \in [a, b]$, $y \in (-\infty, \infty)$:

$$|f(x, y)| < Q, \quad \left| \frac{\partial^{i+j} f(x, y)}{\partial x^i \partial y^j} \right| < P^{i+j} / Q^{j-1}, \quad i + j \leq P. \quad (3.2.20)$$

Where P and Q are positive constants, and p is the order of the method (in this case 2). Then

$$|f_y| < P,$$

$$|F| = |f_x + f f_y| < PQ + QP = 2PQ,$$

$$|G| = |f_{xx} + 2f f_{xy} + f^2 f_{yy}| < P^2Q + 2QP^2 + a^2 P^2|Q| = 4P^2|Q| = 4P^2Q.$$

Hence from (3.2.19)

$$|\Psi(x, y)| < \left(\frac{1}{3} + \left| \frac{2}{3} - a_2 \right| \right) P^2Q,$$

and we obtain the following bound for the principal local truncation error:

$$|\Psi(x_n, (x_n)) h^3| < \left(\frac{1}{3} + \left| \frac{2}{3} - a_2 \right| \right) h^3 P^2Q. \quad (3.2.21)$$

It can be shown that the bound we have just found for the principal local truncation error is also a bound for the whole local truncation error T_{n+1} . (Refer to Henrici (1962) [96], where, however, the bounds assumed for the partial derivatives of f are not those we have assumed in (3.2.20).) This is a consequence of the fact that the Runge-Kutta method is a one-step explicit method. We must write in place of (3.2.21),

$$|T_{n+1}| < \left(\frac{1}{3} + \left| \frac{2}{3} - a_2 \right| \right) h^3 P^2Q. \quad (3.2.22)$$

The corresponding bound for the local truncation error of the general Third-order Runge-Kutta method obtained by setting $R = 3$ in [(3.2.8) to (3.2.10)] is shown by Ralston (1962) [107] to be as follows, for case $a_2 \neq 0$, $a_3 \neq 0$, $a_2 \neq a_3$:

$$\left. \begin{aligned} |T_{n+1}| &< \left\{ \frac{1}{12} + 8|A_1| + |A_2| + |2A_2 + A_3| + |A_2 + A_3| + 2|A_3| \right\} h^4 P^3 Q, \\ \text{where} \\ A_1 &= \frac{1}{24} - \frac{1}{36}(2a_2 + 2a_3 - 3a_2a_3), \\ A_2 &= \frac{1}{24} - \frac{1}{12}a_2, \\ A_3 &= \frac{1}{8} - \frac{1}{6}a_3. \end{aligned} \right\} \quad (3.2.23)$$

It is assumed that the following conditions are satisfied;

$$\left. \begin{aligned} c_1 + c_2 + c_3 &= 1, \\ c_2a_2 + c_3a_3 &= \frac{1}{2}, \\ c_2a_2^2 + c_3a_3^2 &= \frac{1}{3}, \\ c_3a_2b_{32} &= \frac{1}{6}. \end{aligned} \right\} \quad (3.2.24)$$

and also a_2 and a_3 are the two free parameters.

In the case of the general fourth order Runge-Kutta method, the bound for the local truncation error is much more complicated; it may be found in Ralston (1962) [107]. For the popular fourth-order method (3.2.13), Lotkin (1951) [106], using the above analysis shows that

$$|T_{n+1}| < \frac{73}{720} h^5 P^4 Q. \quad (3.2.25)$$

Alternative bounds for the local truncation error can be found by bounding the partial derivatives of f in a manner other than that of (3.2.20). Thus the well known bound of Bieberbach (1930) [108] for the local truncation error of (3.2.13) is given by

$$|T_{n+1}| < 6h^5 QN(1 + N + N^2 + N^3 + N^4), \quad (3.2.26)$$

where, in the neighbourhood $|x - x_0| < A$, $|y - y_0| < B$,

$$|f(x, y)| < Q, \quad \left| \frac{\partial^{i+j} f(x, y)}{\partial x^i \partial y^j} \right| < N/Q^{j-1}, \quad i + j \leq 4,$$

$$|x - x_0|N < 1 \quad \text{and} \quad AQ < B.$$

Numerical evidence suggests that (3.2.25) gives a sharper bound than (3.2.26) (refer to Lotkin (1951) [106]). It can be shown that for the general explicit one step method (3.2.3) the bound for the global truncation error is an order

of magnitude greater than the bound for the local truncation error. (Refer to Lambert (1963) [97]).

If the local truncation error T_{n+1} defined by (3.2.17) satisfies

$$|T_{n+1}| \leq Kh^{P+1}, \quad (3.2.27)$$

where K is a constant, then the global truncation error $e_n \equiv y(x_n) - y_n$ satisfies the inequality

$$|e_n| \leq \frac{h^P K}{L} [\exp(L(x_n - a)) - 1], \quad (3.2.28)$$

where $L(> 0)$ is the Lipschitz constant of $f(x, y)$ with respect to y . For the proof of this result, and an extension to take account of round-off error, we could refer to Henrici (1962) [96].

Carr (1958) [109] gives an alternative bound for the global truncation error of the fourth-order method (3.2.13); it applies, however, only to restricted class of initial value problems. Assume that the local truncation error, T_n , satisfies $|T_n| < E$. Then Carr's Theorem states that if $\frac{\partial f}{\partial y}$ is continuous, negative, and bounded from above and below in a region D of the $x - y$ plane by

$$-M_2 < \frac{\partial f}{\partial y} < -M_1 < 0, \quad (3.2.29i)$$

them for all points (x_n, y_n) in a region D^* of the $x - y$ plane, the global truncation error of the method (3.2.13) satisfies

$$|e_n| \leq 2E/hM_1, \quad (3.2.29ii)$$

provided that the step-length is chosen such that

$$h < \min(M_1/M_2^2, 4M_1^3/M_2^4). \quad (3.2.29iii)$$

The region D^* is such that if $(x_n, y_n) \in D^*$, then $(x_n, y(x_n)) \in D$. Note that (3.2.29), like (3.2.28), indicates that the bound for the global error is an order of magnitude, greater than that for the local error. Carr's result can also take account of round-off error in the sense that if E bounds not the local truncation error, but the total local error, including round-off, then (3.2.29ii) holds with e_n replaced by \tilde{e}_n (which denotes the total global error). An extension of this result, which applies to a more general class of Runge-Kutta methods, can be found in Galler and Resenberg (1960) [110]. The error bounds we have discussed in this section may be very hard to apply in practice. In particular it is quite impracticable to attempt to form a step control policy on the basis of (3.2.27) and (3.2.28). However, there are two good reasons for studying bounds for the local truncation errors of Runge-Kutta methods. Firstly, an important application of Runge-Kutta methods is to provide additional starting values for a multistep predictor-corrector algorithm. In such circumstances, the Runge-Kutta method is applied only for a small number of steps, and a bound on the local error rather than on the global error is adequate. Moreover, it is then necessary to find bounds for the partial derivatives of f only in the immediate vicinity of the initial point.

If, in our choice of steplength for the Runge-Kutta starting method (which need not be the same as the steplength of the predictor corrector algorithm) we are guided by an error bound which turns out to be too conservative, then at least the resulting computational inefficiency is restricted to the starting process; we have the consolation of knowing that in the remainder of the numerical solution starting errors will not dominate.

The second point concerns the choice for the free parameters in Runge-Kutta methods. In the classical methods, these were chosen in order to give simple coefficients. Ralston (1962) [107] has investigated the possibility of choosing these coefficients to minimize the bound for the local truncation error.

3.2.5 Estimation of Error for Runge-Kutta Methods

In the previous section we have seen that bounds for the local truncation error do not form a suitable basis for monitoring local truncation error with a view to constructing a step-control policy. What is needed, in place of a bound, is a readily computable estimate of the local truncation error. Several such estimates exist and we shall discuss a number of them briefly in this section. The most commonly used estimate arises from an application of the process of the differed approach to the limit, alternatively called Richardson extrapolation (Richardson (1927) [111]).

Under the usual localising assumption that no previous errors have been made, we have seen that we may write, using (3.2.18)

$$y(x_{n+1}) - y_{n+1} = T_{n+1} = \Psi(x_n, y(x_n))h^{p+1} + O(h^{p+2}). \quad (3.2.30)$$

where p is the order of the Runge-Kutta method. Now let us compute y_{n+1}^* , a second approximation to $y(x_{n+1})$, obtained by applying the same method at $x_n - 1$ with steplength $2h$.

Under the same localising assumption, it follows that

$$\left. \begin{aligned} y(x_{n+1}) - y_{n+1}^* &= \Psi(x_{n-1}, y(x_{n-1}))(2h)^{p+1} + O(h^{p+2}) \\ &= \Psi(x_n, y(x_n))(2h)^{p+1} + O(h^{p+2}), \end{aligned} \right\} \quad (3.2.31)$$

on expanding $\Psi(x_{n-1}, y(x_{n-1}))$ about $(x_n, y(x_n))$. On subtracting (3.2.30) from (3.2.31) we obtain

$$y_{n+1} - y_{n+1}^* = (2^{p+1} - 1)\Psi(x_n, y(x_n))h^{p+1} + O(h^{p+2}).$$

Thus the principal local truncation error, which is taken as an estimate for the local truncation error, may be written as

$$\Psi(x_n, y(x_n))h^{p+1} = (y_{n+1} - y_{n+1}^*)/(2^{p+1} - 1). \quad (3.2.32)$$

Thus to apply Richardson extrapolation we compute over two successive steps using steplength h , and then recompute over the double step using

steplength $2h$. The difference between the values for y so obtained, divided by 31 in the case of a fourth-order method, is then an estimate of the local truncation error. This estimate is usually quite adequate for the purpose of step-control, but it involves a considerable increase in computational effort; thus to obtain such an estimate at every second step will call for an increase of roughly 50% in computational effort. Estimates for the local truncation error which do not involve additional evaluations of the function $f(x, y)$ have been considered by Kuntzmann (1959) [112]. These estimates, however, involve data calculated at a number of previous steps and essentially estimate the average local truncation error over these steps. In a situation where the local truncation error is changing rapidly and this is the important case from the point of view of step-control policy, such estimates can be misleading. One such estimate, quoted by Scraton (1964) [113], which applies for the fourth-order Runge-Kutta method (3.2.13) is

$$30T_{n+3} = 10y_n + 9y_{n+1} - 18y_{n+3} - y_{n+3} + 3h[f_n + 6f_{n+1} + 3f_{n+2}], \quad (3.2.33)$$

where $f_{n+j} = f(x_{n+j}, y_{n+j})$, $j = 0, 1, 2, \dots$; note that these evaluations of f will already have been made in applying (3.2.13).

The idea of deriving a special Runge-Kutta method which admits an easily calculated error estimate which does not depend on quantities calculated at previous steps was first proposed by Merson (1957) [114]. Merson's method is

$$\left. \begin{aligned} y_{n+1} - y_n &= \frac{h}{6}(k_1 + 4k_4 + k_5), \\ k_1 &= f(x_n, y_n), \\ k_2 &= f(x_n + \frac{1}{3}h, y_n + \frac{1}{3}hk_1), \\ k_3 &= f(x_n + \frac{1}{3}h, y_n + \frac{1}{6}hk_1 + \frac{1}{6}hk_2), \\ k_4 &= f(x_n + \frac{1}{2}h, y_n + \frac{1}{8}hk_1 + \frac{3}{8}hk_3), \\ k_5 &= f(x_n + h, y_n + \frac{1}{2}hk_1 - \frac{3}{2}hk_3 + 2hk_4). \end{aligned} \right\} \quad (3.2.34)$$

The method has order four and an estimate of the local truncation error is given by

$$30T_{n+1} = h(-2k_1 + 9k_3 - 8k_4 + k_5) \quad (3.2.35)$$

This method has been widely used for non-linear problems, although, as pointed out by Scraton (1964) [113], the error estimate is valid only when the differential equation is linear in both x and y , that is of the form

$$y' = ax + by + c.$$

when (3.2.34) is applied to a non-linear differential equation, the error estimate (3.2.35) frequently grossly over estimates the local truncation error

and occasionally (England (1969) [115]) under estimates it. A fourth order method which admits an error estimate which is valid for a non-linear differential equation is derived by Scraton (1964) [113]. It is

$$\left. \begin{aligned} y_{n+1} - y_n &= h \left[\frac{17}{162} k_1 + \frac{81}{170} k_3 + \frac{32}{135} k_4 + \frac{250}{1377} k_5 \right], \\ k_1 &= f(x_n, y_n), \\ k_2 &= f(x_n + \frac{2}{9}h, y_n + \frac{2}{9}hk_1), \\ k_3 &= f(x_n + \frac{1}{3}h, y_n + \frac{1}{12}hk_1 + \frac{1}{4}hk_2), \\ k_4 &= f(x_n + \frac{3}{4}h, y_n + \frac{3h}{128}(23k_1 - 81k_2 + 90k_3)), \\ k_5 &= f(x_n + \frac{9}{10}h, y_n + \frac{9h}{10000}(-345k_1 + 2025k_2 - 1224k_3 + 544k_4)). \end{aligned} \right\} \quad (3.2.36)$$

The estimate for the local truncation error of (3.2.36) is given by

$$T_{n+1} = hqr/s,$$

where

$$\left. \begin{aligned} q &= -\frac{1}{18}k_1 + \frac{27}{170}k_3 + \frac{4}{15}k_4 + \frac{25}{153}k_5, \\ r &= \frac{19}{24}k_1 - \frac{27}{8}k_2 + \frac{57}{26}k_3 - \frac{4}{15}k_4, \\ s &= k_4 - k_1. \end{aligned} \right\} \quad (3.2.37)$$

We should note that the methods of both Merson and Scraton do not require additional function evaluations in order to compute the error estimate. However, when we observe that both methods have fourth order and require five function evaluations per step, whereas we know that there exist fourth-order Runge-Kutta methods which require only four evaluations per step, we see that additional function evaluations are in effect required if we demand an error estimate.

Scraton's estimate, although more realistic than Merson's when applied to a general non-linear differential equation, has the disadvantage that it is not linear in the k_r . As a result it is applicable only to a single differential equation and does not extend to a system of equations. In order to find a method which admits an error estimate which is linear in the k_r , and thus holds for a general non-linear differential equation or system of equations, it is necessary to make further sacrifices in the form of additional function evaluations. Thus England (1969) [115] gives the following fourth order six-stage method;

$$\left. \begin{aligned} y_{n+1} - y_n &= \frac{h}{6}(k_1 + 4k_3 + k_4), \\ k_1 &= f(x, y_n), \\ k_2 &= f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1), \\ k_3 &= f(x_n + \frac{1}{2}h, y_n + \frac{1}{4}hk_1 + \frac{1}{4}hk_2), \\ k_4 &= f(x_n + h, y_n - hk_2 + 2hk_3), \\ k_5 &= f(x_n + \frac{2}{3}h, y_n + \frac{h}{27}(7k_1 + 10k_2 + k_4)), \\ k_6 &= f(x_n + \frac{1}{5}h, y_n + \frac{h}{625}(28k_1 - 125k_2 + 546k_3 + 54k_4 - 378k_5)). \end{aligned} \right\} \quad (3.2.38)$$

The associated estimate for the local truncation error is

$$T_{n+1} = \frac{h}{336}(-42k_1 - 224k_3 - 21k_4 + 162k_5 + 125k_6). \quad (3.2.39)$$

We should note that if (3.2.38) is used without the estimate (3.2.39) it is essentially a four stage method. Further Runge-Kutta processes which admit error estimates have been derived by Shintani (1965-1966) [116] to [118]. We can conclude that all the estimates for the local truncation error of Runge-Kutta methods we have discussed either average the error over a number of steps or require, in one way or another, additional function evaluations.

3.2.6 Weak Stability for Runge-Kutta Methods

It is possible to develop a theory of weak stability for Runge-Kutta methods along exactly the same line as for linear multi step methods. It can be shown for the linear multi step methods (refer to Lambert, J.D. (1986) [97]) that the linearised equation satisfied by the total error $\tilde{e}_n = y(x_n) - \tilde{y}_n$ (which includes round-off as well as truncation error) generated by the linear multi step method

$$\sum_{j=0}^k \alpha_j y_{n+j} = h \sum_{j=0}^k \beta_j f_{n+j} \quad (3.2.40)$$

is of the form

$$\sum_{j=0}^k (\alpha_j - h\lambda\beta_j) \tilde{e}_{n+j} = \phi,$$

provided that we make the assumptions

$$\frac{\partial f}{\partial y} = \lambda, \quad \text{constant} \quad (3.2.41i)$$

and,

$$\text{local error} = \text{constant}. \quad (3.2.41ii)$$

Subsequently, we make no use of the function ϕ and, in effect, define intervals of absolute and relative stability, in terms of the behaviour of the solution of the equation

$$\sum_{j=0}^k (\alpha_j - h\lambda\beta_j) \tilde{e}_{n+j} = 0. \quad (3.2.42)$$

Let us, instead apply (3.2.40) directly to the test equation

$$y' = \lambda y,$$

for which the assumption (3.2.41i) is obviously valid.

We obtain

$$\sum_{j=0}^k (\alpha_j - h\lambda\beta_j) y_{n+j} = 0, \quad (3.2.43)$$

which is exactly the same equation as (3.2.42) with simply a change of argument. Thus, for example, we obtain exactly the same results as in linear

multistep methods (refer to Lambert (1986) [97]) if we define the interval of absolute stability to be that interval of the $h\lambda$ line for which all solutions of the difference equation, obtained by applying the method (3.2.40) to the test equation $y' = \lambda y$ tend to zero as n tends to infinity. To interpret the result in terms of the general equation $y' = f(x, y)$ subject to the unavoidable assumptions (3.2.41) we simply take λ to be an estimate of $\partial f / \partial y$. In this approach we are linearizing the original differential equation, as opposed to linearizing the error equation.

Let us now apply the Runge-Kutta method (3.2.8) to (3.2.10) with $R = 3$ to the test equation $y' = \lambda y$

$$\begin{aligned} k_1 &= f(x, y) = \lambda y, \\ k_2 &= f(x + ha_2, y + ha_2k_1) = \lambda(y + ha_2\lambda y) = \lambda y(1 + a_2h\lambda), \\ k_3 &= f(x + ha_3, y + h(a_3 - b_{32})k_1 + hb_{32}k_2) \\ &= \lambda[y + h\lambda y(a_3 - b_{32}) + h\lambda yb_{32}(1 + a_2h\lambda)] \\ &= \lambda y(1 + a_3h\lambda + a_2b_{32}h^2\lambda^2). \end{aligned}$$

Hence

$$\begin{aligned} \phi(x, y, h) &= c_1k_1 + c_2k_2 + c_3k_3 \\ &= \lambda y[(c_1 + c_2 + c_3) + (c_2a_2 + c_3a_3)h\lambda + c_3a_2b_{32}h^2\lambda^2], \end{aligned}$$

and from (3.2.8) we obtain the difference equation

$$y_{n+1} - y_n = h\lambda[c_1 + c_2 + c_3 + (c_2a_2 + c_3a_3)h\lambda + c_3a_2b_{32}h^2\lambda^2]y_n.$$

Introducing the notation $\bar{h} = h\lambda$, we obtain

$$y_{n+1}/y_n = 1 + (c_1 + c_2 + c_3)\bar{h} + (c_2a_2 + c_3a_3)\bar{h}^2 + c_3a_2b_{32}\bar{h}^3.$$

The general solution of this equation is

$$y_n = d_1 r_1^n,$$

where d_1 is an arbitrary constant and

$$r_1 = 1 + (c_1 + c_2 + c_3)\bar{h} + (c_2a_2 + c_3a_3)\bar{h}^2 + c_3a_2b_{32}\bar{h}^3. \quad (3.2.44)$$

We can then define the three stage Runge-Kutta method to be absolutely stable on the interval (α, β) if r_1 , given by (3.2.44) satisfies $|r_1| < 1$ whenever $\bar{h} \in (\alpha, \beta)$.

If the Runge-Kutta method under discussion is consistent, then $c_1 + c_2 + c_3 = 1$, and from (3.2.44) we may write

$$r_1 = 1 + \bar{h} + O(\bar{h}^2).$$

Hence for sufficiently small positive \bar{h} , $r_1 > 1$, we may conclude that, the interval of absolute stability has the form $(\alpha, 0)$. Moreover, if the three stage method has order three, then equation (3.2.24) holds, and (3.2.44) yields

$$r_1 = 1 + \bar{h} + \frac{1}{2}\bar{h}^2 + \frac{1}{6}\bar{h}^3. \quad (3.2.45)$$

A plot of this function against \bar{h} reveals that $|r_1| < 1$ whenever $\bar{h} \in (-2.51, 0)$. Recalling that (3.2.24) does not specify the coefficients of the method uniquely, we may conclude that all three-stage Runge-Kutta methods of order three have the same interval of absolute stability, namely $(-2.51, 0)$.

We can obtain a generalization of this result from the following alternative approach. In Section 3.2.3 we saw that if the general R -stage Runge-Kutta method (3.2.8) to (3.2.10) has order p , then $\phi(x, y, h)$, defined by (3.2.9) differs from $\phi_T(x, y, h)$, defined by (3.2.6), by terms of order h^p . Thus for a method of order p , by (3.2.8)

$$\begin{aligned} y_{n+1} - y_n &= h\phi_T(x_n, y_n, h) + O(h^{p+1}) \\ &= hf(x_n, y_n) + \frac{h^2}{2!}f^{(1)}(x_n, y_n) + \dots \\ &\quad + \frac{h^p}{p!}f^{(p-1)}(x_n, y_n) + O(h^{p+1}). \end{aligned}$$

For the test function $y' = \lambda y$, $f(x_n, y_n) = \lambda y_n$, $f^{(q)}(x_n, y_n) = \lambda^{q+1}y_n$, $q = 1, 2, \dots, p-1$. Hence

$$y_{n+1} - y_n = (h\lambda + \frac{1}{2!}h^2\lambda^2 + \dots + \frac{1}{p!}h^p\lambda^p)y_n + O(h^{p+1}),$$

or

$$y_{n+1}|y_n = r_1 = 1 + \bar{h} + \frac{1}{2!}\bar{h}^2 + \dots + \frac{1}{p!}\bar{h}^p + O(\bar{h}^{p+1}). \quad (3.2.46)$$

On the other hand, it is clear from a generalisation of the analysis leading to (3.2.44) that for a R -stage method, r_1 is a polynomial of degree R in \bar{h} .

We know that if an R -stage method has order p , then $R \geq p$, and R can equal p only for $p = 1, 2, 3, 4$. Hence for a p -stage method of order p , ($p \leq 4$), we have

$$r_1 = 1 + \bar{h} + \frac{1}{2}\bar{h}^2 + \dots + \frac{1}{p!}\bar{h}^p \quad (3.2.47)$$

irrespective of the values given to the parameter left free after satisfying the order requirements. It follows that, for a given p , $p = 1, 2, 3, 4$, all p -stage Runge-Kutta methods of order p have the same interval of absolute stability. These intervals are given in Table (3.2.1), where Rp denotes any p -stage Runge-Kutta method of order p , $p = 1, 2, 3, 4$.

If the R -stage method has order $p < R$ (and this will always be the case for $p > 4$) then r_1 takes the form

$$r_1 = 1 + \bar{h} + \frac{1}{2}\bar{h}^2 + \dots + \frac{1}{p!}\bar{h}^p + \sum_{q=p+1}^R \nu_q \bar{h}^q \quad (3.2.48)$$

where the coefficients ν_q are functions of the coefficients of the Runge-Kutta method, but are not determined by the order requirement. In such a case the interval will depend on the particular choice for the free parameters. If, for example for $R = 3$ we consider methods of order two, then only the first two of the equation (3.2.24) need be satisfied and we obtain from (3.2.44)

$$r_1 = 1 + \bar{h} + \frac{1}{2}\bar{h}^2 + \nu_3\bar{h}^3,$$

where $\nu_3 = c_3 a_2 b_{32}$. If $\nu_3 = 0$, the interval of absolute stability is clearly $(-2, 0)$, whereas for $\nu_3 = \frac{1}{6}$ it is, from table 3.2.1, $(-2.51, 0)$. (Note that $\nu_3 = \frac{1}{6}$ is not a sufficient condition for the order of the method to be three, since the third of the equations (3.2.24) is not necessarily satisfied). For $\nu_3 = \frac{1}{12}$, the interval of absolute stability becomes $(-4.52, 0)$.

(Table 3.2.1)

method	τ_1	Interval of absolute stability
R_1	$1 + h$	$(-2, 0)$
R_2	$1 + \bar{h} + \frac{1}{2}\bar{h}^2$	$(-2, 0)$
R_3	$1 + \bar{h} + \frac{1}{2}\bar{h}^2 + \frac{1}{6}\bar{h}^3$	$(-2.51, 0)$
R_4	$1 + \bar{h} + \frac{1}{2}\bar{h}^2 + \frac{1}{6}\bar{h}^3 + \frac{1}{24}\bar{h}^4$	$(-2.78, 0)$

Here we should mention that the algorithm we have used for the solution of state and adjoint differential equations for our problems in the following chapters is the Runge-Kutta fourth order method, i.e. Algorithm 3.2.3, Formula (3.2.13).

3.3 Simpson's Numerical Integration Rule

Calculating the definite integral of a given real function $f(x)$,

$$\int_a^b f(x) dx,$$

is a classic problem. For some simple integrals $f(x)$, the indefinite integral

$$\int_a^x f(x)dx = F(x), \quad F'(x) = f(x),$$

can be obtained in closed form and then

$$\int_a^b f(x)dx = F(b) - F(a).$$

See Gröbner and Hofreiter (1961) [119] for a comprehensive collection of formulas describing such indefinite integrals and many important definite integrals.

As a rule, however, definite integrals are computed using discretization methods which approximate the integral by finite sums corresponding to some portion of the interval of integration $[a, b]$ ('numerical quadrature'). A typical representative of this class of methods is Simpson's rule, which is still one of the best-known and most widely used integration method.

Simpson's rule corresponds to quadratic approximation; thus, for $x_j \leq x \leq x_j + 2h$,

$$\begin{aligned}
 \int_{x_j}^{x_j+2h} f(x)dx &= h \int_0^2 f(x_j + \theta h) d\theta \\
 &\approx h \int_0^2 [1 + \theta\Delta + \frac{1}{2}\theta(\theta-1)\Delta^2] f_j d\theta \\
 &= h[(\theta + \frac{1}{2}\theta^2\Delta + (\frac{1}{6}\theta^3 - \frac{1}{4}\theta^2)\Delta^2) f_j]_0^2 \\
 &= h[2f_j + 2(f_{j+1} - f_j) + \frac{1}{3}(f_{j+2} - 2f_{j+1} + f_j)] \\
 &= \frac{1}{3}h(f_j + 4f_{j+1} + f_{j+2}).
 \end{aligned}$$

where the forward difference operator Δ is defined by $\Delta = E - 1$, where E is called the shift operator and $E^k f_j = f_{j+k}$. Furthermore, $\Delta^k f_j = \Delta^{k-1} f_{j+1} - \Delta^{k-1} f_j$, where k is any integer.

A parabolic arc is fitted to the curve $y = f(x)$ at the three tabular points x_j , $x_j + h$ and $x_j + 2h$. Consequently, if $N = (b - a)/h$ is even, one obtains Simpson's rule:

$$\begin{aligned}
 \int_a^b f(x)dx &= \int_{x_0}^{x_2} f(x)dx + \int_{x_2}^{x_4} f(x)dx + \dots \\
 + \int_{x_{N-2}}^{x_N} f(x)dx &= \frac{1}{3}h[f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + \dots + 4f_{N-1} + f_N],
 \end{aligned} \tag{3.3.1}$$

where

$$f_j = f(x_j) = f(a + jh), \quad j = 0, 1, 2, \dots, N.$$

Integration by Simpson's rule involves computing a finite sum of values given by the integrand $f(x)$. Simpson's rule is also effective for automatic computation, and one direct application is desk calculation usually gives sufficient accuracy.

3.3.1 Accuracy

In automatic computation involving a known integrand $f(x)$, we emphasize that it is quite appropriate to program increased interval subdivision to provide the desired accuracy, but that for desk calculation a (truncation) error bound is again useful. Suppose that in $x_{j-h} \leq x \leq x_{j+h}$ the function $f(x)$ has the Taylor expansion

$$f(x) = f_j + (x - x_j)f'_j + \frac{1}{2!}(x - x_j)^2 f''_j + \dots,$$

then

$$\int_{x_{j-h}}^{x_{j+h}} f(x)dx = 2h \left[f_j + \frac{1}{3} \frac{h^2}{2!} f''_j + \frac{1}{5} \frac{h^4}{4!} f^{(4)}_j + \dots \right].$$

One may re-express the quadrature rule for $x_j - h \leq x \leq x_j + h$ by

$$f_{j+1} = f(x_j + h) \quad \text{and} \quad f_{j-1} = f(x_j - h)$$

as Taylor series; thus

$$\begin{aligned}
 \int_{x_j-h}^{x_j+h} f(x)dx &\approx \frac{1}{3}h(f_{j-1} + 4f_j + f_{j+1}) \\
 &= \frac{1}{3}h[(f_j - hf'_j + \frac{1}{2!}h^2f''_j - \dots) + 4f_j \\
 &\quad + (f_j + hf'_j + \frac{1}{2!}h^2f''_j + \dots)] \\
 &= 2h(f_j + \frac{1}{3}\frac{h^2}{2!}f''_j + \frac{1}{3}\frac{h^4}{4!}f^{(4)}_j + \dots).
 \end{aligned}$$

Comparison of these two forms shows that the correction is needed and it is

$$2h\left(\frac{1}{5} - \frac{1}{3}\right)\frac{h^4}{4!}f^{(4)}_j + \dots = -\frac{1}{90}h^5f^{(4)}_j + \dots$$

Ignoring higher order terms, we conclude that the approximate bound on the truncation error estimating

$$\int_a^b f(x)dx$$

by Simpson's rule (with $\frac{N}{2}$ subintervals of width $2h$) is

$$-\frac{N}{2}\frac{1}{90}h^5\max\{f^{(4)}(x)\} = -\frac{(b-a)h^4}{180}\max\{f^{(4)}(x)\}.$$

Here we should mention that the algorithm we have used for the solution of numerical integration, the norms for our problems in the following chapters is the Simpson's $\frac{1}{3}$ rule, i.e. (3.3.1).

3.4 The Linear Search

In this section we discuss some of the linear search techniques, i.e. methods for finding a maximum or minimum of $f(x)$ along a given line; this is essentially a one-dimensional problem. Some linear search techniques involve the evaluation of the derivatives of $f(x)$, others do not; both types are in common use. Linear searches are used in conjunction with many gradient methods. (Refer to Walsh (1975) [120]).

The linear search can occupy a large proportion of the total time for the solution of the problem and should, therefore, obviously be as efficient as possible. Some suitable balance should be struck between the order of the polynomial required to obtain a given accuracy and the amount of computation required.

Let us consider the minimization problem:

$$\min_{\alpha>0} \Phi(\alpha), \tag{3.4.1}$$

where Φ is defined as (2.2.2) in Chapter 2. Here we shall describe some of the techniques that only impose some weak acceptance criteria on $\alpha^{(k)}$, that

lead to methods that perhaps as well in theory and in practice. Let $\mu^{(k)}$ be the first value of α such that

$$\nabla f(x^{(k)} + \mu^{(k)} s^{(k)})^T s^{(k)} = \sigma \nabla f(x^{(k)})^T s^{(k)},$$

where $0 < \sigma < 1$. In this case we have:

$$\mu^{(k)} \geq -\frac{1-\sigma}{\Omega} \cos(\theta^{(k)}) \cdot \|\nabla f(x^{(k)})\|.$$

where Ω is an upper bound on $\|\nabla^2 f(x)\|$.

It indicates that convergence can still be maintained even if we do not require $\alpha^{(k)}$ to be the solution of the one-dimensional minimization problem (3.4.1) but only require that $\alpha^{(k)} \geq \mu^{(k)}$.

Therefore if during the line search an α is found such that

$$\nabla f(x^{(k)} + \alpha s^{(k)})^T s^{(k)} \geq \sigma \nabla f(x^{(k)})^T s^{(k)} \quad (3.4.2)$$

then the search for α can be stopped.

A frequently used method for determining the step lengths $\alpha^{(k)}$ is to estimate a local minimizer of $\Phi(\alpha)$. Then $\alpha^{(k)}$ satisfies at least the requirement:

$$\alpha^{(k)} = \text{Arg} \min_{\alpha > 0} f(x^{(k)} + \alpha s^{(k)}) \quad (3.4.3)$$

That means, $\alpha^{(k)}$ satisfies at least approximately the requirement:

$$\Phi'(\alpha) = 0, \quad (3.4.4)$$

where Φ' is the first derivative of Φ .

In general, since (3.4.4) is a non linear equation it can not be solved analytically. Hence a numerical method is selected to find the value of α which satisfies (3.4.3). We use the term 'Exact line search', when $\alpha^{(k)}$ is chosen such that it satisfies (3.4.4) exactly. When (3.4.4) is satisfied only approximately, the procedure is termed an 'Inexact line search'.

A sufficient decrease in f at each iteration, is an essential requirement for a step length algorithm, associated with a descent method. In order to assume the convergence, the step length must produce a "sufficient" or "satisfactory" decrease in $f(x)$. Although it seems to be common sense to require that

$$f(x^{(k+1)}) < f(x^{(k)}), \quad (3.4.5)$$

it comes as no great surprise that this simple condition does not guarantee that the sequence $\{x^{(k)}\}_{k \in \mathbb{N}}$ will converge to a minimizer of f . Counter examples to prove this statement can be found for example in Gill, Murray and Wright (1981) [43] and Dennis and Schnabel (1983) [121].

A great deal of effort has been expended upon the construction of efficient line search procedures. Most of these procedures are based on the principles of Goldstein (1962) [122], (1965) [42], [123] and (1967) [128] and Armijo (1966) [124] from which the well known Goldstein-Armijo principle is derived.

We can describe the Goldstein-Armijo principle as follows:

A sufficient decrease in $f(x)$ is achieved if $\alpha^{(k)}$ satisfies:

$$\begin{aligned}\alpha^{(k)} q_2 \nabla f(x^{(k)})^T s^{(k)} &\leq f(x^{(k+1)}) - f(x^{(k)}) \\ &\leq \alpha^{(k)} q_1 \nabla f(x^{(k)})^T s^{(k)},\end{aligned}\tag{3.4.6}$$

where q_1 and q_2 are scalars satisfying $0 < q_1 < q_2 < 1$ and $q_1 = 1 - q_2$. To ensure that $\alpha^{(k)}$ is neither too large nor too small we set suitable upper and lower bounds in (3.4.6).

It should be emphasized, that condition (3.4.6) alone can not guarantee a good value of $\alpha^{(k)}$. Although this strategy would be 'efficient' in that a suitable $\alpha^{(k)}$ would be found with only a single function evaluation per iteration, but it would be extremely inefficient if any descent method uses such a step length algorithm. It is true that minimizing the number of function evaluations for the sake of computational labour required is important, but it is also essential to consider the performance of a step length algorithm not merely in terms of that, but also in terms of the overall reduction in f achieved at each step. To balance this, some flexibility is desirable in specifying the conditions to be satisfied by $\alpha^{(k)}$.

Fletcher (1980) [42] suggests that $\alpha^{(k)}$ is such that $x^{(k+1)}$ satisfies the condition:

$$|g^{(k+1)T} s^{(k)}| \leq -\sigma g^{(k)T} s^{(k)},\tag{3.4.7}$$

together with the Goldstein (1965) [59] requirement that:

$$f(x^{(k+1)}) \leq f(x^{(k)}) + \rho \alpha^{(k)} g^{(k)T} s^{(k)}.\tag{3.4.8}$$

where $g^{(k)} = \nabla f(x^{(k)})$ is the gradient vector at $x^{(k)}$, and where $\rho \in (0, \frac{1}{2})$, $\sigma \in (0, 1)$ and $\rho < \sigma$.

Condition (3.4.7) ensures that $\alpha^{(k)}$ is not too small. σ determines the accuracy with which $\alpha^{(k)}$ approximates a stationary point of f along $s^{(k)}$, and also provides a means of controlling the balance of effort to be expended in computing $\alpha^{(k)}$. When $\sigma = 0$, we have the case of exact line search, condition (3.4.8) requires that $f(x^{(k)} + \alpha s^{(k)})$ lies on or below the line $\rho(\alpha) = f(x^{(k)}) + \rho \alpha g^{(k)T} s^{(k)}$, and ensures a sufficient decrease when used together with (3.4.7).

The advantage that condition (3.4.7) has as an acceptance criterion in that its interpretation in terms of a local minimizer suggests efficient methods for computing a good value of $\alpha^{(k)}$. In particular safeguarded polynomial fitting techniques for univariate minimization converge very rapidly on well

balanced functions. We can refer to Gill, Murray and Wright (1981) [43] for a full description of these techniques. Also Dennis and Schnabel (1983) [121] describe a similar procedure. They give an algorithm of back tracing line search procedure using quadratic and cubic interpolation. An investigation of line search procedures was carried out by Al-Baali and Fletcher (1986) [125]. They mainly aimed at developing a line search method that is applicable to the nonlinear least-square problem in which $f(x)$ is a sum of squares of nonlinear functions:

$$f(x) = \frac{1}{2} \sum_{i=1}^m (r_i(x))^2. \quad (3.4.9)$$

If $r \in \mathbb{R}^n$ denotes the column vector whose elements are the functions $r_i(x)$ and $A = \nabla r^T$ is the jacobian matrix of r , then it follows that:

$$g(x) = A(x) \cdot r(x). \quad (3.4.10)$$

Therefore in order to evaluate $f(x)$, it needs the evaluation of $r(x)$, and also to evaluate $g(x)$ it requires in addition an evaluation of the jacobian matrix.

In their work they studied various iterative schemes for the line search sub-problem that guarantee finding an acceptable step length α in a finite number of steps. This can be achieved by first bracketing an interval of acceptable α values, and then reducing this bracket uniformly by repeated sectioning of the bracket in a systematic way. These schemes for the line search sub-problem include a scheme for finding an acceptable value of α that satisfies (3.4.2) together with the Goldstein (1965) [59] requirement (3.4.8). This is then generalised in order to find an acceptable value of α that satisfies (3.4.7) and (3.4.8) as well as that of Fletcher (1980) [42].

Further investigations lead to modifications suggested with the aim of producing schemes in which the gradient vector is evaluated as frequently as possible, on the assumption that this is the major cost in using the methods. Their works show in particular that substantial gains in efficiency can be obtained in non-linear least square problems by making polynomial approximations to the individual functions r_i , rather than the overall function f of (3.4.9).

The problem of using proper line search conditions to ensure global convergence of unconstrained optimization algorithms was studied by many authors (Goldstein (1965)) [59], Wolf (1969) [38], Zoutendijk (1976) [36], Powell (1976) [50]).

A well known result is that, if the angle between $-g^{(k)}$ and the search direction $s^{(k)}$ is bounded away from $\frac{\pi}{2}$, and if the line search conditions (3.4.7) and (3.4.8) are satisfied, then global convergence is assumed. More precisely, let

$$\cos \theta^{(k)} = -\frac{s^{(k)T} g^{(k)}}{\|s^{(k)}\| \|g^{(k)}\|},$$

then we have

Theorem 3.4.1 (Lemaréchal, 1981) [126]).

Let f be bounded below on the level set

$$s = \{x | f(x) \leq f(x_0)\},$$

and ∇f be Lipschitzian on s , that is, there exists a constant $L > 0$ such that

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \forall x, y \in s.$$

Let the line searches in the algorithm satisfy the line search conditions (3.4.7) and (3.4.8) at each iteration, and let

$$\sum_{i=0}^{\infty} \cos^2 \theta^{(k)} = +\infty,$$

then the algorithm is globally convergent, that is

$$\lim_{k \rightarrow \infty} \inf \|g^{(k)}\| = 0.$$

Moreover, if

$$\cos \theta^{(k)} \geq c > 0, \quad k = 1, 2, \dots,$$

then

$$\lim_{k \rightarrow \infty} g^{(k)} = 0.$$

In practice, the following line search conditions which are slightly stronger than (3.4.7) and (3.4.8) are often used,

$$\Phi(\alpha^{(k)}) \leq \Phi(0) + \rho \alpha^{(k)} \Phi'(0), \quad (3.4.11i)$$

$$|\phi(\alpha^{(k)})| \leq -\sigma \Phi'(0), \quad (3.4.11ii)$$

where $0 < \rho \leq \sigma < 1$. Usually we require $\rho < \frac{1}{2}$ to preserve superlinear convergence (Fletcher 1987 [127]). This set of conditions gives line searches that are as accurate as we like when $\sigma \rightarrow 0$, and the two conditions are consistent (there exists an interval of acceptable points) as long as $\Phi(\alpha)$ is bounded below, $0 < e < \sigma$ and $\Phi'(0) < 0$.

In the next sections we describe the two linear search techniques we have used in this thesis to solve our problems.

3.4.1 Linear Search at Constant Step

In this technique the step is retained at a constant value until a reduction in the objective function is achieved. A quadratic is then fitted through the last three points. Obviously, in the regions remote from the final solution, the smaller the step the better the fit. Computational experience with this technique in conjunction with the conjugate gradient method requires the inclusion of a step-halving procedure. Thus, if at any stage the step is found to be too large, it is automatically halved and the new value is retained in subsequent searches unless further reduction is necessary. The procedures for this technique is as follows:

Algorithm 3.4.1

To locate the maximum of the function $X = X(\varepsilon)$ set $\varepsilon = 0$ calculate $X = X_A$

$$\varepsilon = \varepsilon + \delta \quad \text{calculate } X = X_B$$

Case (1) or Case (2) is now appropriate.

Case (1) $X_B \geq X_A$

Stage (i) $\varepsilon = \varepsilon + \delta$ calculate $X = X_C$

Stage (ii) if $X_C \geq X_B$
Set $X_A = X_B$, $X_B = X_C$
Repeat stage (i) and (ii)
until $X_C < X_B$

Stage (iii) When $X_C < X_B$
use formula (1) for the ε_{\max}
Calculate $X = X(\varepsilon_{\max})$
The larger of X_B and X is then selected.

Case (2) $X_B < X_A$

Stage (i) Set $X_C = X_B$
 $\varepsilon = \frac{1}{2}\varepsilon$ calculate $X = X_B$

Stage (ii) if $X_B \leq X_A$
Repeat stage (i) and (ii)
until $X_B > X_A$

Stage (iii) When $X_B > X_A$
use formula (2) for ε_{\max}
Calculate $X = X(\varepsilon_{\max})$.
The larger of X_B and X is then selected.

Formula (1)

$$\varepsilon_{\max} = \varepsilon - \frac{\delta (Z_1 - 3Z_2)}{2 (Z_1 - Z_2)}$$

Formula (2)

$$\varepsilon_{\max} = \frac{(3Z_1 - Z_2)}{2(Z_1 - Z_2)} \varepsilon$$

where

$$Z_1 = X_B - X_A$$

$$Z_2 = X_C - X_B$$

The derivations of formula (1) and (2) can be found in Walder (1969) [129].

3.4.2 Quadratic Interpolation Method

Consider the linear search problem of minimizing the function $f(x)$ along the line $x = x_k + \lambda s$, where x_k is the current point and s is a given direction. Powell (1964) [130] published a simple algorithm for determining the minimizing value of λ , using quadratic interpolation together with a few common-sense rules. This algorithm forms part of Powell's more general method for finding the minimum value of a function $f(x)$ without calculating derivatives (see Walsh (1975) [120]). However, it may also be used in conjunction with any gradient method or, more generally, with any optimization technique that requires a one-dimensional search. In each iteration, Powell's algorithm finds a quadratic function $h(\lambda)$ which takes the same values as $f(x_k + \lambda s)$ for three current values of λ . Having found the value of λ ($\lambda = \tilde{\lambda}^*$, say) which minimizes $h(\lambda)$, one of the three current values of λ is discarded and is replaced by $\tilde{\lambda}^*$. The iterations continue until the desired accuracy is attained.

Let

$$\begin{aligned} f_A &= f(x_k + As), \\ f_B &= f(x_k + Bs), \\ f_C &= f(x_k + Cs), \end{aligned}$$

be function values at three points, not necessarily consecutive, on the line $x = x_k + \lambda s$.

Assume that the quadratic function

$$h(\lambda) = a + b\lambda + c\lambda^2, \quad (3.4.12)$$

takes the values of f_A , f_B , f_C at $\lambda = A$, B , C , respectively, i.e. assume,

$$\left. \begin{aligned} a + bA + cA^2 &= f_A, \\ a + bB + cB^2 &= f_B, \\ a + bC + cC^2 &= f_C. \end{aligned} \right\} \quad (3.4.13)$$

Equation (3.4.13) gives,

$$\left. \begin{aligned} a &= \left[\frac{f_A BC(C - B) + f_B CA(A - C) + f_C AB(B - A)}{(A - B)(B - C)(C - A)} \right], \\ b &= \left[\frac{f_A(B^2 - C^2) + f_B(C^2 - A^2) + f_C(A^2 - B^2)}{(A - B)(B - C)(C - A)} \right], \\ c &= - \left[\frac{f_A(B - C) + f_B(C - A) + f_C(A - B)}{(A - B)(B - C)(C - A)} \right]. \end{aligned} \right\} \quad (3.4.14)$$

The quadratic function $h(\lambda)$ of (3.4.12) has a turning point at $\lambda = -\frac{b}{2c}$ and has a minimum there if $c > 0$. Hence using equation (3.4.14), we can find that this turning point is at $\lambda = \tilde{\lambda}^*$, where

$$\tilde{\lambda}^* = \frac{f_A(B^2 - C^2) + f_B(C^2 - A^2) + f_C(A^2 - B^2)}{2[f_A(B - C) + f_B(C - A) + f_C(A - B)]}, \quad (3.4.15)$$

and $h(\lambda)$ has a minimum there if,

$$\frac{f_A(B - C) + f_B(C - A) + f_C(A - B)}{(A - B)(B - C)(C - A)} < 0. \quad (3.4.16)$$

Given an initial point x_k and a direction of search s , Powell's quadratic interpolation method for minimizing the general function $f(x)$ on the line $x = x_k + \lambda s$ is as follows:

1. Choose a step length $h|s|$; the vector s need not be a unit vector.
2. Evaluate $f(x_k)$ and $f(x_k + hs)$.
3. If $f(x_k) < f(x_k + hs)$, evaluate $f(x_k - hs)$, otherwise evaluate $f(x_k + 2hs)$.
4. Find the turning point $\lambda = \tilde{\lambda}^*$ of the quadratic function $h(\lambda)$ fitted through these three points, using (3.4.15) and test for a minimum, using (3.4.16). Go to rule 5, 6 or 7, as appropriate.
5. If the point $\lambda = \tilde{\lambda}^*$ corresponds to a maximum of $h(\lambda)$ or if it corresponds to a minimum which is at a greater distance than $H|s|$ (where H is prescribed), from the nearest of the three current points, proceed as follows. Discard the point which is furthest from the turning point and obtain a step $H|s|$ in the direction in which the function decreases; this step (Fig 3.4.1) is taken from the point furthest from (nearest to) the turning point when turning point corresponds to a maximum (minimum). Return to rule 4.

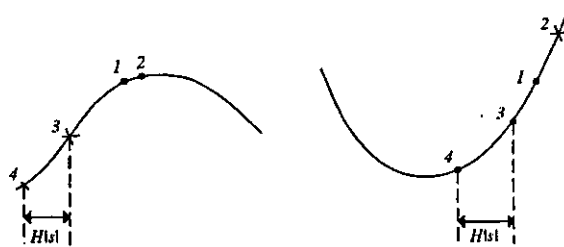


Figure (3.4.1): Rule 5 of Powell's quadratic interpolation method

6. If the point $\lambda = \tilde{\lambda}^*$ corresponds to a minimum of $h(\lambda)$ and it is within a small prescribed distance $\varepsilon|s|$ of the nearest point $\lambda = A$, say of the three current points, take

$$\min\{f(x_k + \tilde{\lambda}^*s), f(x_k + As)\}$$

as the required minimum value of $f(x)$.

7. If the point $\lambda = \tilde{\lambda}^*$ corresponds to a minimum of $h(\lambda)$ to which neither rule 5 nor rule 6 applies, i.e., if it is not further than $H|s|$ from the nearest of the three current points, but not within $\varepsilon|s|$ of it, discard the point with the highest function value and replace it by $\lambda = \tilde{\lambda}^*$. Return to rule 4.

Note: It is always desirable to locate the next turning point by interpolation rather than by extrapolation; an exception is made to rule 7 to allow for this.

In Figure (3.4.2), point 3 has the highest function value, though point 2 is discarded because the minimum lies between points 1 and 3.

As a very useful case, the three current points on the line $x = x_k + \lambda s$ may be equally spaced. It is then possible by suitable changes of origin and scale to take $A = -1$, $B = 0$, $C = 1$, and equation (3.4.15) reduces to

$$\tilde{\lambda}^* = \frac{f_A - f_C}{2(f_A - 2f_B + f_C)}. \quad (3.4.17)$$

from equations (3.4.12), (3.4.14) and (3.4.17), we find the turning value of $h(\lambda)$ in this case to be



Figure (3.4.2): Exception to rule 7 of Powell's quadratic interpolation method

$$y(\tilde{\lambda}^*) = f_B - \frac{(f_A - f_C)^2}{8(f_A - 2f_B + f_C)}, \quad (3.4.18)$$

and the condition (3.4.16) that is a minimum reduces to $f_A - 2f_B + f_C > 0$.

In applying Powell's quadratic interpolation method, the number of function evaluations may be reduced when more than one search has to be made in the same direction by noting that three function values are sufficient to predict the second derivative.

$$\frac{\partial^2}{\partial \lambda^2} [f(x_k + \lambda s)].$$

The prediction is,

$$D = 2C = \frac{2[C - B]f_A + (A - C)f_B + (B - A)f_C}{(A - B)(B - C)(C - A)}$$

Then if f_A , f_B and D are known, the minimum of $h(\lambda)$ is predicted to be at $\lambda = \tilde{\lambda}^*$, where,

$$\tilde{\lambda}^* = \frac{1}{2}(A + B) - \frac{(f_A - f_B)}{D(A - B)}. \quad (3.4.19)$$

Equation (3.4.19), which replaces equation (3.4.15) for the second and subsequent searches in the direction s , is derived by subtracting the second of equations (3.4.13) from the first and using the relations,

$$\tilde{\lambda}^* = -\frac{b}{2c} = \frac{-b}{D}.$$

For the flow chart of the quadratic interpolation method see Figure (3.4.3). Also for more details on this method refer to Walsh (1975) [120] and Rao (1979) [131].

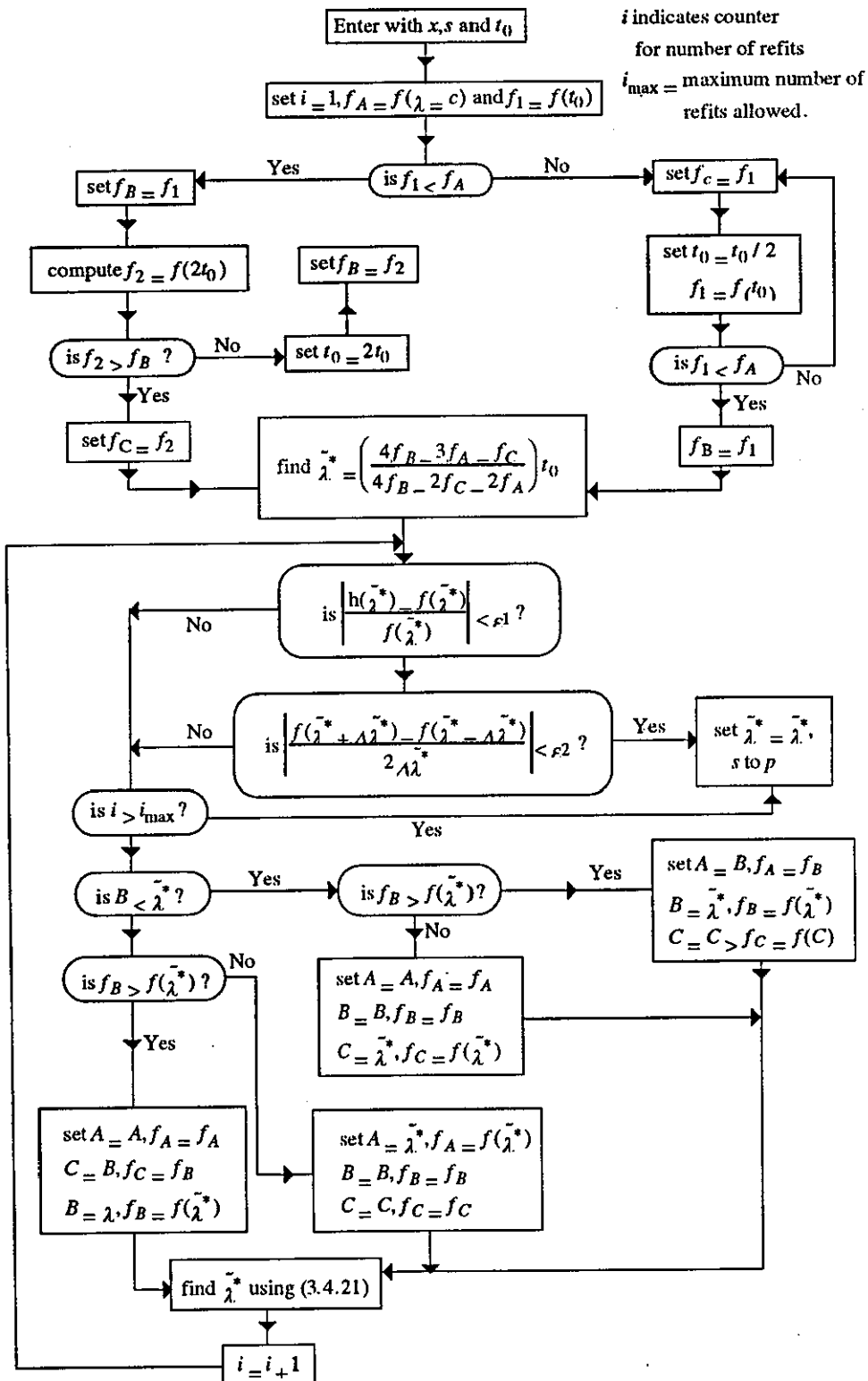


Figure (3.4.3)

Chapter 4

PROBLEM 1

4.1 A Simple Control Problem

A simple optimal control problem is considered (see for example, Abdul Wahab (1979) [17]).

Let the state equation of a simple process be

$$\frac{dx_1}{dt} = -ax_1 + u \quad (4.1.1)$$

with

$$x_1(0) = c_1. \quad (4.1.2)$$

The problem is to select a control $u(t)$, $t_0 \leq t \leq t_f$, to minimize the integral

$$J = \frac{1}{2} \int_{t_0}^{t_f} [x_1^2 + u^2] dt \quad (4.1.3)$$

The problem is solved first analytically and then numerically, using the seven methods described in Chapter 2, (sections 2.2.2, 2.4, 2.5, 2.6.1, 2.7, 2.8) and also this chapter, section 4.4.1.

4.2 Analytical Solution

Let us transform the problem (4.1.1), (4.1.2), (4.1.3) in the usual way.

By introducing an additional state variable,

$$x_2(t) = \frac{1}{2} \int_{t_0}^t [(x_1)^2 + (u)^2] dt \quad (4.2.1)$$

we get the alternative formulation:

$$\text{minimize } J = x_2(t_f) \quad (4.2.2)$$

subject to:

$$\left. \begin{aligned} \frac{dx_1}{dt} &= -ax_1 + u \equiv f_1 & , & & x_1(t_0) = c_1 \\ \frac{dx_2}{dt} &= \frac{1}{2}x_1^2 + \frac{1}{2}u^2 \equiv f_2 & , & & x_2(t_0) = 0 \end{aligned} \right\} \quad (4.2.3)$$

The values of $a = 1$, $t_f = 1$, $t_0 = 0$ and $c_1 = 12$ were chosen for computational purposes.

Now using the maximum principle, we solve the problem (4.2.2) to (4.2.3) analytically.

Here the Hamiltonian is given by,

$$H = \sum_{i=1}^2 \lambda_i f_i = \lambda_1 f_1 + \lambda_2 f_2 = \lambda_1(-ax_1 + u) + \lambda_2 \left(\frac{1}{2}x_1^2 + \frac{1}{2}u^2 \right)$$

and the adjoint equations are then as follows:

$$\dot{\lambda}_i = - \sum_{j=1}^2 \lambda_j \frac{\partial f_j}{\partial x_i} \quad , \quad i = 1, 2,$$

so that,

$$\left. \begin{aligned} \dot{\lambda}_1 &= a\lambda_1 - \lambda_2 x_1 , & \lambda_1(t_f) &= 0 , \\ \dot{\lambda}_2 &= 0 , & \lambda_2(t_f) &= 1. \end{aligned} \right\} \quad (4.2.4)$$

The boundary conditions in (4.2.4) arise from (4.2.2).

To find the optimal control u^* , set

$$\frac{\partial H}{\partial u} = 0,$$

i.e.

$$\lambda_1 + \lambda_2 u = 0,$$

so that

$$u^* = -\frac{\lambda_1}{\lambda_2} = -\lambda_1. \quad (4.2.5)$$

Now by substituting u^* and also the parameters a , t_f and c_1 with their values given in (4.2.3) we get, the state equations,

$$\left. \begin{aligned} \dot{x}_1 &= -x_1 - \lambda_1 , & x_1(0) &= 12 , \\ \dot{x}_2 &= \frac{1}{2}x_1^2 + \frac{1}{2}\lambda_1^2 , & x_2(0) &= 0 , \end{aligned} \right\} \quad (4.2.6)$$

and the adjoint equations become,

$$\left. \begin{aligned} \dot{\lambda}_1 &= \lambda_1 - \lambda_2 x_1 , & \lambda_1(1) &= 0 , \\ \dot{\lambda}_2 &= 0 , & \lambda_2(1) &= 1. \end{aligned} \right\} \quad (4.2.7)$$

From (4.2.6) and (4.2.7) we have

$$\left. \begin{aligned} \dot{x}_1 &= -x_1 - \lambda_1 , & x_1(0) &= 12 \\ \dot{\lambda}_1 &= \lambda_1 - x_1 , & \lambda_1(1) &= 0 , \end{aligned} \right\} \quad (4.2.8)$$

so that,

$$\left. \begin{aligned} \text{and, } x_1(t) &= Ae^{\sqrt{2}t} + Be^{-\sqrt{2}t}, \\ \lambda_1(t) &= -A(\sqrt{2}e^{\sqrt{2}t} + e^{\sqrt{2}t}) - B(e^{-\sqrt{2}t} - \sqrt{2}e^{-\sqrt{2}t}) \end{aligned} \right\} \quad (4.2.8A)$$

where,

$$B = \frac{12(\sqrt{2}e^{\sqrt{2}} + e^{\sqrt{2}})}{(\sqrt{2} + 1)e^{\sqrt{2}} + (\sqrt{2} - 1)e^{-\sqrt{2}}},$$

and

$$A = \frac{12(\sqrt{2} - 1)e^{-\sqrt{2}}}{(1 + \sqrt{2})e^{\sqrt{2}} + (\sqrt{2} - 1)e^{-\sqrt{2}}}$$

Then from (4.2.5) and (4.2.8A) we get the optimal control u^* as,

$$u^* = \frac{12(e^{-\sqrt{2}(1-t)} - e^{\sqrt{2}(1-t)})}{(\sqrt{2} + 1)e^{\sqrt{2}} + (\sqrt{2} - 1)e^{-\sqrt{2}}}. \quad (4.2.9)$$

By substituting λ_1 and x_1 in (4.2.6) we solve for \dot{x}_2 to get,

$$\begin{aligned} x_2(t) &= \frac{1}{2} \left(\frac{A^2}{2\sqrt{2}} e^{2\sqrt{2}t} - \frac{B^2}{2\sqrt{2}} e^{-2\sqrt{2}t} \right) \\ &+ \frac{1}{2} \left\{ \frac{A^2(3 + 2\sqrt{2})}{2\sqrt{2}} e^{2\sqrt{2}t} - \frac{B^2(3 - 2\sqrt{2})e^{-2\sqrt{2}t}}{2\sqrt{2}} \right\} + C. \end{aligned}$$

where $C = 29.21006166$.

When $t_f = 1$, the optimal value of the performance index J^* is given by

$$J^* = x_2(1) = 27.77893885 \text{ to 8 decimal places.}$$

4.3 Numerical Solution

4.3.1 The State and Adjoint Equations

The state differential equations (4.2.3) are solved using the fourth order Runge-Kutta method by setting

$$\left. \begin{aligned} x_{1,n+1} &= x_{1,n} + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \\ x_{2,n+1} &= x_{2,n} + \frac{1}{6}(v_1 + 2v_2 + 2v_3 + v_4), \end{aligned} \right\} \quad (4.3.1)$$

where,

$$\begin{aligned} k_1 &= h(-ax_{1,n} + u_n), \\ v_1 &= h\left(\frac{1}{2}x_{1,n}^2 + \frac{1}{2}u_n^2\right), \\ k_2 &= h\left[-a\left(x_{1,n} + \frac{1}{2}k_1\right) + u_n\right], \\ v_2 &= h\left[\frac{1}{2}\left(x_{1,n} + \frac{1}{2}k_1\right)^2 + \frac{1}{2}u_n^2\right], \\ k_3 &= h\left[-a\left(x_{1,n} + \frac{1}{2}k_2\right) + u_n\right], \\ v_3 &= h\left[\frac{1}{2}\left(x_{1,n} + \frac{1}{2}k_2\right)^2 + \frac{1}{2}u_n^2\right], \\ k_4 &= h\left[-a(x_{1,n} + k_3) + u_n\right], \\ v_4 &= h\left[\frac{1}{2}(x_{1,n} + k_3)^2 + \frac{1}{2}u_n^2\right], \end{aligned}$$

and $a = 1$, $c_1 = 12$.

The adjoint equations:

$$\left. \begin{aligned} \dot{\lambda}_1 &= a\lambda_1 - \lambda_2 x_1, & \lambda_1(t_f) &= 0 \\ \dot{\lambda}_2 &= 0, & \lambda_2(t_f) &= 1, \end{aligned} \right\} \quad (4.3.2)$$

are also solved using the fourth order Runge-Kutta method by setting,

$$\left. \begin{aligned} \lambda_{1,n} &= \lambda_{1,n+1} + \frac{1}{6} \{w_1 + 2w_2 + 2w_3 + w_4\}, \\ \lambda_{2,n} &= \lambda_{2,n+1} + \frac{1}{6} \{z_1 + z_2 + 2z_3 + z_4\}, \end{aligned} \right\} \quad (4.3.3)$$

where,

$$\begin{aligned} w_1 &= -h(a\lambda_{1,n+1} - \lambda_{2,n+1}x_{1,n+1}), \\ z_1 &= 0, \\ w_2 &= -h[a(\lambda_{1,n+1} + \frac{1}{2}w_1) - (\lambda_{2,n+1} + \frac{1}{2}z_1)x_{1,n+1}], \\ z_2 &= 0, \\ w_3 &= -h[a(\lambda_{1,n+1} + \frac{1}{2}w_2) - (\lambda_{2,n+1} + \frac{1}{2}z_2)x_{1,n+1}], \\ z_3 &= 0, \\ w_4 &= -h[a(\lambda_{1,n+1} + w_3) - (\lambda_{2,n+1} + z_3)x_{1,n+1}], \\ z_4 &= 0, \end{aligned}$$

and $a = 1$, $t_f = 1$.

4.4 Results and Discussion

4.4.1 Gradient Method

The algorithm for the gradient in function space method applied to problem 1 is as follows:

- i. Select an initial control as a possible estimate of u^* .
- ii. Compare $g_0 = g(u_0)$, where $g = \frac{\partial}{\partial u} = \lambda_1 + \lambda_2 u$. (4.4.1)
- iii. Choose ε so that u_{i+1} is a better solution to the optimal profile u_i .
- iv. Compare $u_i = u_i - \varepsilon g_i$.
- v. Has the process converged? If yes u_{i+1} is the optimal profile, if no go to (iv).

In order to analyse the efficiency of the gradient method in function space, applied to the problem, we compare the results obtained numerically when we vary the three critical parameters, step length factor (ε), integration step size (N) and initial control (u_0), with the analytical solution found earlier in section 4.2.

The results for this method and also other methods in this chapter show the minimum number of iterations to achieve the best absolute error (e_{abs}) so that any further iterations produced no improvement. Notice that the absolute error falls from approximately 1.5×10^{-5} when $N = 100$ to approximately .07 when $N = 5$.

Table (4.4.1), shows the effect of ε and N in achieving the minimum yield $x_2(1) = J^*$, from an initial constant control $u_0 = -4$ i.e. the best starting control. The best value obtained was 27.77908793 to 3 decimal places with $e_{abs} \simeq 1.5 \times 10^{-4}$. The corresponding parameter values were $\varepsilon = 0.4$, $N = 100$, or 400 and $m = 29$. Here the step length factor played an important role, i.e. as the step length increased to 0.4 the number of iterations and cpu time decreased but remained relatively similar after that.

The effect of N is also clear from Table (4.4.1) i.e. by choosing larger values for N we can obtain a more accurate minimum yield $x_2(1)$ for a given number of iterations.

Table (4.4.2), shows the effect of choice of initial control u_0 on the minimum yield $x_2(1)$, with sufficiently large enough $N = 100$ and $\varepsilon = 0.4$. For example, by selecting u_0 at -4 or -5 we can achieve $J^* = x_2(1)$ with $e_{abs} \simeq 1.5 \times 10^{-4}$ in fewer iterations and less computing time than when u_0 is taken as 0, -1 , -2 or -3 .

Fig (4.4.1) and Fig (4.4.2) show typical growth curves for the yield x_2 and the control u respectively along the time axis, when we compare the analytical solutions with the numerical ones, with $N_1 = m = 1$, $N_2 = m = 29$ and $A = \text{Analytical}$, using $N = 100$, $u_0 = -4$ and $\varepsilon = 0.4$. As it can be seen when $m = 29$ the behaviour of the curves both for x_2 and u are closer to the analytical one.

Fig (4.4.3) compares the analytical solution for x_2 with the numerical solutions with the number of integration steps $N_1 = N = 10$ and $N_2 = N = 100$ and the number of iterations 28 and 29 respectively, with $u_0 = -4$ and $\varepsilon = 0.4$, to give the best yield, and Fig (4.4.4) gives similar results for control u . It may not be much clear from x_2 to see the difference graphically for different N 's but for u we can see that when $N = 100$ the behaviour of the curve is close to the analytical solution.

Figures (4.4.5) and (4.4.6) demonstrate the effect on the yield x_2 and the control u respectively of using step length factors $N_1 = \varepsilon = 0.1$ and $N_2 = \varepsilon = 0.4$, when the starting control is -4 and $N = 100$. Here also for x_2 we may not see much difference graphically for different ε 's, but for u we can see that when $\varepsilon = 0.4$, the curve behaves closer to the analytical solution.

Fig (4.4.7) shows how $x_2(1)$ varies with m for $u_0 = -4$, $N = 100$ and $\varepsilon = 0.4$.

Finally Fig (4.4.8) shows the effect on $x_2(1)$ of changing u_0 for different number of iterations ($m = 0, 1$ and 29) with $N = 100$ and $\varepsilon = 0.4$. Here we can see that as m increases the curves for all starting controls get closer to each other and merge to the optimal $x_2(1)$.

Although for this simple problem the results obtained above may not show much significant improvement in minimizing the yield $x_2(1)$, when we vary the critical parameters u_0 , ε and N , but still some advantages exist by selecting a proper combination of these parameters.

The choice of initial control regardless of ε and N is a factor in achieving the best yield, i.e. by referring to Fig (4.4.8) even with 1 iteration we can see how the change in u_0 can effect the yield $x_2(1)$, but as m increases it seems the effect fades away and for the higher number of iterations the yield $x_2(1)$ does not vary much with the change in u_0 . Now we consider the interaction of the other two essential factors ε and N and their affect on minimizing $x_2(1)$.

As we established before clearly the best combination was found to be with the larger ε and larger N i.e. 0.4 and 100 respectively, but if we choose smaller values for N , we can compensate by selecting larger values for ε and vice versa. The problem of numerical instability can occur when we select ε too large or N too small e.g., when we take ε equal to 0.8 with $N = 100$ and $u_0 = -4$, we get $x_2(1)$ as 27.77908796 to 8 decimal places after 30 iterations, which is worse than minimum obtained by $\varepsilon = 0, 0.4$. Increasing ε even more, for example to 1.8 , gives a complete numerical break down. Also by taking N too small, we get numerical instability, e.g. when $N = 5$ and $m = 29$, with $u_0 = -4$ and $\varepsilon = 0.4$, we get $x_2(1)$ as 27.83208930 to 8 decimal places and even if we increase m to 98 still there is no improvement on the yield $x_2(1)$. In view of the above discussion when we are selecting the best combination of parameters, appropriate care is necessary and balance should be made between selecting N and ε to cut down, computer storage requirements on one hand and to prevent numerical instability on the other. Also a proper choice of initial control will help to speed up the process of convergence.

4.4.2 Steepest Descent

The algorithm for steepest descent applied to the problem 1 is as described in Chapter 2, Section 2.2.2. The line search technique used for this method is linear search at constant step, which was described in Chapter 3, Section 3.4.1. The gradient (g) is obtained in the same way as for gradient method in function space (see (4.4.1)).

Here we again examine the effect of selecting step length factor, integration step and initial control in the solution of problem 1. Table (4.4.3), shows the effect of ε and N in obtaining the minimum yield $J^* = x_2(1)$, with the best starting $u_0 = -4$. The best yield achieved was 27.77908783 to 8 decimal places with $e_{abs} \simeq 1.5 \times 10^{-4}$, $\varepsilon = 0.4$ and $N = 100$ or 400 after 21 iterations.

Selecting a larger step length factor speeds up the search for the minimum yield $x_2(1)$, in terms of reducing the number of iterations and consequently the computing time. Table (4.4.4) shows how the starting control affects the yield $x_2(1)$ when $N = 100$ and $\varepsilon = 0.4$, i.e. by selecting initial control as -4 or -5 we can achieve $x_2(1)$ with $e_{abs} \simeq 1.5 \times 10^{-4}$ in slightly fewer iterations than when u_0 is taken as $0, -1, -2$ or -3 .

Figure (4.4.9) and Fig (4.4.10) show the growth curves for the yield x_2 and the control u respectively along the time axis, when the numerical solutions are compared with the analytical one, with $N_1 = m = 1$ and $N_2 = m = 21$ with $N = 100$, $u_0 = -4$ and $\varepsilon = 0.4$. Hence for x_2 with both $m = 1$ and $m = 29$ we don't see much difference graphically, but for u we can see that when $m = 29$ the behaviour of the curve is closer to the analytical one than when m is selected as 1 .

Fig (4.4.11) and Fig (4.4.12) compare the analytical solutions for x_2 and u respectively with the numerical ones, with the number of integration steps $N_1 = N = 10$ and $N_2 = N = 100$, with $u_0 = -4$, $\varepsilon = 0.4$ and the number of iterations 15 and 21 to give the best yield. There is not much difference graphically for x_2 , between $N = 10$ and $N = 100$, but for u , it can be seen that when $N = 100$, the curvature is closer to the analytical curve than when $N = 10$. Figures (4.4.13) and (4.4.14) show the effect on the yield x_2 and the control u respectively of using step length factors $N_1 = \varepsilon = 0.1$ and $N_2 = \varepsilon = 0.4$ for 21 iterations with $u_0 = -4$ and $N = 100$. As can be seen from the graph of x_2 , there is not much difference between $\varepsilon = 0.1$ and $\varepsilon = 0.4$, graphically, but for u , a little difference can be seen in the behaviour of the curves between the two ε 's, i.e. when $\varepsilon = 0.4$, the curvature is closer to the analytical one.

Fig (4.4.15) shows how the yield $x_2(1)$ varies with m , when $u_0 = -4$, $N = 100$ and $\varepsilon = 0.4$.

Finally fig (4.4.16) demonstrates the effect on $x_2(1)$ of changing initial control for different number of iterations, ($m = 0, 1$ and 21) with $N = 100$ and $\varepsilon = 0.4$.

Here also we can see as m increases, no matter what starting control was selected, they all converge towards optimal $J^* = x_1(1)$.

The main difference between the steepest descent method and the gradient methods was that in the former the minimum was formed along each gradient, but in the latter a fixed step length is used throughout. The results achieved did not show much significant difference between the two in terms of selecting u_0 and the interaction between ε and N .

The instability pattern is also the same as GFS with ε too large or N too small, e.g., $\varepsilon = 0.8$, $u_0 = -4$, $N = 100$ and $m = 21$, we get $x_2(1) = 27.77908804$ to 8 decimal places, which is worse than with $\varepsilon = 0.4$, and if ε is increased more, say to 1.9 we get instabilities.

Also, when $N = 5$ with $u_0 = -4$, $\varepsilon = 0.4$ and $m = 21$, we get $x_2(1)$ as 27.8320893 to 8 decimal places and by increasing the number of iterations even to 98 still there is no change in the yield $x_2(1)$.

Thus the conclusions reached for the GFS on selecting the best combinations of u_0 , ε and N is also applicable for SD for this particular problem.

4.4.3 Fletcher-Reeves

The algorithm for Fletcher-Reeves applied to the problem 1 is as described in Chapter 2 Section 2.4. The line search technique used for this method is linear search at constant step (see Chapter 3 Section 3.4.1). A further numerical technique was necessary to estimate the norm of each gradient trajectory, where the norms can be calculated as was described in Chapter 2 Section 2.10.1. The gradient (g) is obtained in the same way as for gradient method (refer to equation (4.4.1)).

Table (4.4.5), shows the effect of ε and N in achieving the minimum yield $x_2(1)$, with the best starting $u_0 = -4$. The best $J^* = x_2(1)$ was 27.77908774 to 8 decimal places with $e_{abs} \simeq 1.5 \times 10^{-4}$. The corresponding parameter values were $\varepsilon = 0.4$, $N = 100$ or 400 and $m = 8$.

Again the selection of a larger N gives the minimum yield in fewer iterations for a required e_{abs} . Similarly the effect of ε can also be seen from Table (4.4.5).

Table (4.4.6) shows the effect of selecting different initial controls u_0 , when $N = 100$ and $\varepsilon = 0.4$, i.e. by choosing $u_0 = -3$ or -4 we can achieve $x_2(1)$ with $e_{abs} \simeq 1.5 \times 10^{-4}$, in fewer iterations and less computing time than when u_0 is taken as 0, -1 , -2 or -5 .

Fig (4.4.17) and Fig (4.4.18) indicate the growth curves for the yield x_2 and control u along the time axis, comparing the analytical growth with the numerical ones, when $N_1 = m = 1$, and $N_2 = m = 28$, with $u_0 = -4$, $\varepsilon = 0.4$ and $N = 100$. As can be seen from x_2 , there is not much difference graphically, between $m = 1$ and $m = 8$, but for u we can see that by taking $m = 8$, the behaviour of the curve is much closer to the analytical one than $m = 1$.

Figures (4.4.19) and (4.4.20), show the comparison of the analytical solutions with the numerical ones for x_2 and u respectively with $N_1 = N = 10$, and $N_2 = N = 1 = 100$, when $m = 10$ and $m = 8$, with $u_0 = -4$ and $\varepsilon = 0.4$ to give the best yield. Here for x_2 , there is not much difference graphically for different N 's but for u , with $N = 100$, behaviour of the curve is closer to the analytical solution.

Figures (4.4.21) and (4.4.22), demonstrate the growth curve of time against the yield x_2 and also the change in u respectively, when we compare the analytical solutions with the numerical ones, where the step length factors are $N_1 = \varepsilon = 0.1$ and $N_2 = \varepsilon = 0.4$, for $m = 8$, $u_0 = -4$ and $N = 100$. Here as it can be seen, for x_2 , there is not much difference graphically between $\varepsilon = 0.1$ and $\varepsilon = 0.4$. Also for u not much difference can be seen in the behaviour of the curves for $\varepsilon = 0.1$ and $\varepsilon = 0.4$.

Fig (4.4.23), shows how the yield $x_2(1)$ varies with m , when $u_0 = -4$, $N = 100$ and $\varepsilon = 0.4$.

Finally Fig (4.4.24) demonstrates the effect on $x_2(1)$ of changing initial control for different number of iterations ($m = 0, 1$ and 8), with $N = 100$ and $\varepsilon = 0.4$. Here also as m increases, the curvature of all starting controls merges towards the optimal $x_2(1)$.

The effect of u_0 and the interaction between ε and N are similar to GFS and SD, apart from the fact that, it takes fewer iterations to get to an optimal $x_2(1)$ for a required e_{abs} , for FR than the other two methods.

The numerical instability behaviour is also similar to GFS. For example with $\varepsilon = 0.8$, at $u_0 = -4$, $N = 100$ and $m = 8$, $x_2(1) = 27.77908801$ to 8 decimal places, which is getting distant from the optimal minimum value obtained with $\varepsilon = 0.4$ and as we extend ε even more to 1.9, then instability occurs. Also when we select N too small say 5 at $u_0 = -4$, $\varepsilon = 0.4$ and $m = 8$, we get $x_2(1) = 27.83209026$ to 8 decimal places, with relatively poor $e_{abs} \simeq 0.05$, and as we increase the iterations to 50, then we get a complete break down.

The recommendation on selecting u_0 , ε and N can be followed as suggested for the previous methods, for the problem 1.

4.4.4 Polak-Ribière

The algorithm for Polak-Ribière method is described in Chapter 2, Section 2.5. The line search technique and the calculation of the norms are as described for FR.

Table (4.4.7), shows the effect of ε and N on the optimal $x_2(1)$, with the best $u_0 = -4$. The best yield was 27.77908793 to 8 decimal places with $e_{abs} \simeq 1.5 \times 10^{-4}$ and corresponding parameter values, $\varepsilon = 0.4$, $N = 100$ or 400 and $m = 17$. Like previous methods with larger N and ε a better minimum yield in fewer iterations is achieved for a better e_{abs} .

Table (4.4.8), shows that selection of a proper initial control speeds up finding the optimal yield. Here also a sufficient large $N = 100$ was taken with $\varepsilon = 0.4$. By selecting $u_0 = -4$ or -5 compared with 0, -1 , -2 and

–3 the best minimum yield were obtained in fewer number of iterations and less computing time.

Figures (4.4.25) and (4.4.26), show the growth curve of time against the yield x_2 and u respectively, comparing the analytical solutions with the numerical ones, $N_1 = m = 1$ and $N_2 = m = 17$, $\varepsilon = 0.4$, $u_0 = -4$ and $N = 100$. Here for x_2 , we don't see much difference graphically, by changing $m = 1$ to $m = 17$. But for the graph of control we can see taking $m = 17$ confirms the behaviour of the curve closer to the analytical one, compared with $m = 1$.

Figures (4.4.27) and Fig (4.4.28), show similar growth for x_2 and u with $N_1 = N = 10$ and $N_2 = N = 100$, with corresponding number of iterations 25 and 17 respectively, where $u_0 = -4$ and $\varepsilon = 0.4$. As can be seen there is not much difference graphically for x_2 , between selecting N as 10 with N as 100. But for the graph of control we can see that taking $N = 100$ can lead to a curvature closer to the analytical solution, than $N = 10$.

Figures (4.4.29) and (4.4.30), show the growth curve of time against the yield x_2 and also change in u respectively, when the analytical solutions are compared with the numerical ones, where $N_1 = \varepsilon = 0.1$ and $N_2 = \varepsilon = 0.4$, for $m = 17$, $u_0 = -4$ and $N = 100$. Here also for x_2 there is not much difference graphically between $\varepsilon = 0.1$ and $\varepsilon = 0.4$ and they are both close to the analytical one. But some difference can be seen from the graph of u , as when $\varepsilon = 0.4$ is selected the behaviour of the curve is similar to the analytical one, where as for $\varepsilon = 0.1$, the curvature is not quite as close as the previous one.

Fig (4.4.31), shows how varying m affects the yield $x_2(1)$ when $u_0 = -4$, $N = 100$ and $\varepsilon = 0.4$.

Finally fig (4.4.32) demonstrates how the change in u_0 can affect $x_2(1)$ for different number of iterations, e.g., $m = 0, 1$ and 8 , with $N = 100$ and $\varepsilon = 0.4$. Here we can see that as m increases all the starting controls merge towards the optimal $x_2(1)$.

The examination of PR method was carried out the same as previous methods, and the findings were similar in terms of proper choice of u_0 and interaction between ε and N .

Also we found out that the numerical instability happens for ε too large or N too small.

When ε is taken as 0.8 with $N = 100$ and $u_0 = -4$, we get $x_2(1)$ as 27.77908804 to 8 decimal places after 17 iterations which is worse than optimal minimum achieved by 0.4 , and as we increased ε to 1.8 then complete break down happens.

Also by taking N too small, i.e. ($N = 5$) with $u_0 = -4$, and $\varepsilon = 0.4$ for $m = 17$ we find $x_2(1)$ as 27.83208930 to 8 decimal places and when we increase m to 50, instability occurs. Therefore we come to the same conclusion as for previous methods, where we have to select appropriate u_0 and find the best combinations of ε and N .

4.4.5 Hybrid 1

The algorithm for the Hybrid 1 method is described in Chapter 2, Section 2.6.1. The line search technique used and calculation of the norms are the same as for FR and PR.

Table (4.4.9), shows the effect of ε and N in achieving the minimum yield $x_2(1)$, with the best starting control $u_0 = -4$. The best yield was 27.77908774 to 8 decimal places with the corresponding $e_{abs} \simeq 1.5 \times 10^{-4}$, $\varepsilon = 0.4$, $N = 100$ or 400 and $m = 8$. Again the selection of larger N and ε gives the minimum yield in fewer iterations for a required accuracy.

Table (4.4.10), shows the effect of selecting different starting controls with $N = 100$ and $\varepsilon = 0.4$, i.e. by choosing $u_0 = -2$, -3 or -4 we can obtain better minimum $x_2(1)$ with $e_{abs} \simeq 1.5 \times 10^{-4}$ in fewer iterations and less computing time than when u_0 is taken as 0, -1 or -5 .

Fig (4.4.43) and Fig (4.4.34) show the growth curve of time against x_2 and u respectively and compare the analytical solution with the numerical ones with $N_1 = m = 1$ and $N_2 = m = 8$, where $\varepsilon = 0.4$, $N = 100$ and $u_0 = -4$. For the graph of x_2 , we can not see much difference in the shape of the curves for $m = 1$ and $m = 8$ and they both are similar to the analytical solution. But for u we can see that with $m = 8$, the behaviour of the curve is similar to the analytical one, where with $m = 1$, a difference can be seen from the analytical solution. Figures (4.4.35) and (4.4.36) demonstrate the effect of N on the yield x_2 and the control (u) respectively, when the analytical solutions are compared with the numerical ones, with $N_1 = N = 10$ and $N_2 = N = 100$ for $m = 8$, $u_0 = -4$ and $\varepsilon = 0.4$.

Here also for x_2 graphically, there is not much difference between $N = 10$ and $N = 100$, but for u , selecting $N = 100$ can lead to a closer behaviour of the curve to the analytical one, than $N = 10$.

Fig (4.4.37) and Fig (4.4.38), show similarly, the effect of ε on the yield $x_2(1)$ for x_2 and u respectively with $\varepsilon = 0.1$ and $\varepsilon = 0.4$ for $m = 8$, $u_0 = -4$ and $N = 100$.

Here, both for the graphs of x and u , we can not see much difference in the way the curves behave, between $\varepsilon = 0.1$ and $\varepsilon = 0.4$, and they both are close to the analytical one.

Fig (4.4.39), shows how the yield $x_2(1)$ varies with m , when $u_0 = -4$, $N = 100$ and $\varepsilon = 0.4$.

Finally Fig (4.4.40), shows how the change in u_0 affects the yield $x_2(1)$ with $m = 0$, 1 and 8 where $N = 100$ and $\varepsilon = 0.4$. Here also we can see that as m increases all the starting controls behave similarly and the curve gets closer to the optimal $x_2(1)$.

The Hybrid 1 method produced similar results to FR in terms of proper selection of u_0 and interaction between ε and N .

The instability behaviour is also similar, i.e. it occurs with ε too large or N too small.

With ε taken as 0.8, at $u_0 = -4$, $N = 100$ and $m = 8$, we obtain $x_2(1)$ as 27.77908801 to 8 decimal places, which is worse than $\varepsilon = 0.4$, and as we increase the value of ε to 1.9 then complete instability occurs.

When N is selected too small and say 5 for example with $u_0 = -4$, $\varepsilon = 0.4$ and $m = 8$ we get $x_2(1) = 27.83209252$ to 8 decimal places and as m is increased to 50, complete break down happens.

Thus, we arrive at the same conclusion as previous methods, when we come to select u_0 and the best combination of ε and N .

4.4.6 Angle Test Hybrid

The algorithm for Angle test hybrid method is described in Chapter 2, Section 2.7. The line search and calculation of the norms are the same as for H1. For this method we had to consider the effect of another parameter, namely $\tau > 0$.

Table (4.4.11), shows the effect of ε and N on finding the optimal $x_2(1)$, with the best starting control $u_0 = -4$, and $\tau = 0.00001$. The best yield $x_2(1)$ was 27.77808774 to 8 decimal places after 8 iterations with $N = 100$ or 400 and $e_{abs} \simeq 1.5 \times 10^{-4}$. The effect of ε and N is also clear from the table, i.e. by selecting larger ε and N , we get better minimum yield in fewer iterations.

Table (4.4.12) can also reveal the effect of selecting u_0 in achieving a better minimum yield $x_2(1)$, with $N = 100$, $\tau = 0.00001$ and $\varepsilon = 0.4$. By choosing $u_0 = 2$, -3 or -4 we can obtain a better yield with $e_{abs} \simeq 1.5 \times 10^{-4}$ in fewer iterations and less computing time compared with $u_0 = 0$, -1 or -5 .

The effect of τ on obtaining the minimum $x_2(1)$ was practically negligible.

Figures (4.4.41) and (4.4.42), show the growth curve of time against x_2 and u respectively comparing the analytical solutions with the numerical ones, with $N_2 = m = 1$ and $N_2 = m = 8$, where $\varepsilon = 0.4$, $N = 100$, $u_0 = -4$ and $\tau = 0.00001$.

Here for the graph of x we can see little difference in the behaviour of the curves with $m = 1$ and $m = 8$, since they both are similar to the analytical one. But for the graph of u , we can see, with $m = 8$ the behaviour of the curve is closer to the analytical solution, than $m = 1$.

Figures (4.4.43) and (4.4.44) similarly give the graphs of time against x_2 and u respectively, showing the effect of N taken as $N_1 = 10$ and $N_2 = 100$, with $\varepsilon = 0.4$, $u_0 = -4$ and $m = 8$. Here for x_2 both $N = 10$ and $N = 100$ had similar curvature close to the analytical one. But for u , the behaviour of the curve was closer to the analytical one when N was taken as 100.

Figures (4.4.45) and (4.4.46), in the same way demonstrate the effect of ε on x_2 and u respectively with $N_1 = \varepsilon = 0.1$ and $N_2 = \varepsilon = 0.4$, and corresponding parameter values, $N = 100$, $u_0 = -4$, $m = 8$ and $\tau = 0.00001$. Here also, for both the graphs of x and u , there is not much difference between $\varepsilon = 0.1$ and $\varepsilon = 0.4$, and they both are close to the analytical one.

Fig (4.4.47) demonstrates the effect of m on the minimum yield $x_2(1)$ with $u_0 = -4$, $\varepsilon = 0.4$, $N = 100$ and $\tau = 0.00001$.

Finally Fig (4.4.48), shows the change in u_0 against the minimum yield $x_2(1)$ with $m = 0, 1$ and 8 , where $\varepsilon = 0.4$, $N = 100$ and $\tau = 0.00001$. Here also as m increases, the behaviour of the curve for all the starting controls were similar and get closer to the optimal $x_2(1)$.

For the Angle test hybrid, the effect of u_0 and the best combinations of ε and N , were the same as for previous methods. Similarly numerical instability occurs for ε too large or N too small.

When ε is taken as 0.8, with $u_0 = -4$, $N = 100$, $m = 8$ and $\tau = 0.00001$, $x_2(1)$ is found as 27.77908801 to 8 decimal places, which is worse, than the optimal value obtained, with $\varepsilon = 0.4$, and as the value of ε is increased to 1.9 instability occurs.

Also when N is selected too small, e.g. $N = 5$ with $u_0 = -4$, $\varepsilon = 0.4$, $m = 8$ and $\tau = 0.00001$ we get $x_2(1) = 27.83208930$ to 8 decimal places and as m is increased further to 50 we get complete break down.

The same conclusions are reached for the previous methods on selecting proper u_0 and best combinations of ε and N are also true for ATH.

4.4.7 Hybrid 3

The algorithm for Hybrid 3 method can also be found in Chapter 2, Section 2.8. The calculation of the norms and line search are the same as for H1 and ATH. The effect of the new parameters, $\lambda > 0$ and $u < \frac{1}{2}$ had to be considered for this method.

Table (4.4.13), shows the effect of ε and N in achieving the minimum yield $x_2(1)$ with $u_0 = -4$ as the best starting control, $\lambda = 0.00001$ and $\mu = 0.45$. The best yield was found as 27.77908774 to 8 decimal places with $e_{abs} \simeq 1.5 \times 10^{-4}$, $\varepsilon = 0.4$ and $N = 100$ or 400 after 8 iterations. As in previous methods the selection of a larger N gives the minimum yield in

fewer iterations, and less computing time for a required accuracy of e_{abs} . Similarly the effect of ε can be seen from Table (4.4.13).

Table (4.4.14), shows the effect of selecting different initial controls u_0 , with $N = 100$, $\varepsilon = 0.4$, $\lambda = 0.00001$ and $\mu = 0.45$. By choosing $u_0 = -2, -3$ or -4 we can get $x_2(1)$ with $e_{abs} \simeq 1.5 \times 10^{-4}$ in fewer iterations and less computing time than when u_0 is taken as $0, -1$ or -5 . The effect of λ and μ on minimising the yield $x_2(1)$, had practically no significant importance.

Fig (4.4.49) and (4.4.50), compare the analytical solutions with the numerical ones for x_2 and u respectively, with $N_1 = m = 1$ and $N_2 = m = 8$, with corresponding parameter values, $\varepsilon = 0.4$, $N = 100$, $u_0 = -4$, $\lambda = 0.00001$ and $\mu = 0.45$. Here also for the graph of x there is little difference between $m = 1$ and $m = 8$, and the behaviour of the curves are similar to the analytical one. But for the graph of control we can see that when $m = 8$, the behaviour of the curve is similar to the analytical one and for $m = 1$ a difference can be seen with the analytical one. Figures (4.4.51) and (4.4.52), show the effect of N for $N_1 = N = 10$ and $N_2 = N = 100$ and corresponding values of $m = 10$ and 8 respectively, with $u_0 = -4$, $\varepsilon = 0.4$, $\lambda = 0.00001$ and $\mu = 0.45$. Here for x_2 we can see no difference in the curvature, between $N = 10$ and $N = 100$ and analytical solution. But for u , we can see that the behaviour of the curve is closer to the analytical one when N is selected as 100 .

Figures (4.4.53) and (4.4.54) show the effect of ε for $N_1 = \varepsilon = 0.1$ and $N_2 = \varepsilon = 0.4$, with $N = 100$, $u_0 = -4$, $m = 8$, $\lambda = 0.00001$ and $\mu = 0.45$. As it can be seen for both graphs of x and u , the curves of $\varepsilon = 0.4$ and 0.1 behave similarly and close to the analytical solution. Fig (4.4.55), shows how the yield $x_2(1)$ reacts with different number of iterations, when $u_0 = -4$, $N = 100$, $\varepsilon = 0.4$, $\lambda = 0.00001$ and $\mu = 0.45$.

Finally Fig (4.4.56) demonstrates how the change in u_0 can affect $x_2(1)$ for different number of iterations ($m = 0, 1$ and 8) with $N_2 = 100$, $\varepsilon = 0.4$, $\lambda = 0.00001$ and $\mu = 0.45$. Here also as m increases the behaviour of the curves for all the starting controls, were similar and get closer to the optimal $x_2(1)$.

For Hybrid 3 method, the effect of u_0 and interaction between ε and N , are found to be the same as for previous methods.

The numerical instability also happens for ε too large or N too small.

When ε is selected as 0.8 , at $u_0 = -4$, $N = 100$, $m = 8$ we get $x_2(1)$ as 27.77908801 to 8 decimal places which is worse than the optimal value obtained with $\varepsilon = 0.4$, and as we take larger values of ε , say 1.9 , then we get complete break down. Also, when we select N too small, e.g. $N = 5$ at $u_0 = -4$, $\varepsilon = 0.4$ and $m = 8$, we get $x_2(1) = 27.83208930$ to 8 decimal places and even by increasing m to 50 not only we get no improvement on the yield, but also we get a complete break down. The recommendation on selecting u_0 , ε and N can be followed as suggested for the previous methods.

4.5 Summary of the Results

Referring to the summary table (4.5.1) and Fig (4.5.1) when $\varepsilon = 0.4$, $N = 100$, $\mu = 0.45$, $\tau = 0.0001$, $\lambda = 0.00001$ and $e_{abs} \simeq 1.5 \times 10^{-4}$, the following results can be seen.

At $u_0 = 0$, FR performed the best in terms of converging to the minimum $x_2(1)$, then the four methods of PR, H1, ATH and H3 performed the same, then SD and finally GFS.

At $u_0 = -1$, PR, H1, ATH and H3 performed the same and the best from convergency point of view, then SD, FR and finally GFS.

At $u_0 = -2$, the three methods of H1, ATH and H3 performed the same and the best in terms of convergency, then FR, then SD and PR performed the same and finally GFS.

At $u_0 = -3$, the four methods of FR, H1, ATH and H3 performed the same and the best from convergency point of view, then PR and SD performed fairly the same and finally GFS.

At $u_0 = -4$, the four methods of FR, H1, ATH and H3 performed the same and the best in terms of convergency, then PR, SD and finally GFS.

At $u_0 = -5$, the four methods of FR, H1, ATH and H3 performed the same and the best. Also in terms of convergency, then PR, SD and finally GFS.

4.6 Conclusion

In this chapter for the problem concerned all seven methods produced consistent results.

They all produced better values for $x_2(1)$ when larger step length factors were selected. Also choosing smaller integration step helped in producing a better minimum yield $x_2(t_f)$, i.e. closer answer to the analytical solution.

Overall for this particular problem it seems that the Hybrid methods performed better in most cases from the convergency point of view than other methods, since they could converge to $\min x_2(t_f)$ in fewer iterations than other techniques for a required e_{abs} , and therefore less computing time was consumed. Here we should note that when starting control was selected as $u_0 = 0$, the performance of FR was better than Hybrid methods. If the number of function evaluations were to be considered, GFS method had obtained its optimal $x_2(t_f)$ in fewer NFE than other techniques. But for this problem we could say after the Hybrid conjugate gradient techniques and

FR, Polak-Ribière performed better than steepest descent and also in turn steepest descent performed better than Gradient method in function space.

TABLE (4.4.1): Results of GFS with varying ϵ and N.

N ϵ	5	10	50	100	400
0.1	J=27.85312659 m=93 eabs=0.0741877 Cputime<1 NFE=94	J=27.79289991 m=91 eabs= 1.0×10^{-2} Cputime<1 NFE=92	J=27.77953739 m=77 eabs= 6.0×10^{-4} Cputime=2 NFE=78	J=27.77909949 m=68 eabs= 1.6×10^{-4} Cputime=4 NFE=69	J=27.77909948 m=68 eabs= 1.6×10^{-4} Cputime=7 NFE=69
0.2	J=27.85237891 m=87 eabs=0.0734401 Cputime<1 NFE=88	J=27.79289556 m=80 eabs= 1.0×10^{-2} Cputime<1 NFE=81	J=27.77953398 m=39 eabs= 6.0×10^{-4} Cputime=1 NFE=40	J=27.77908799 m=69 eabs= 1.5×10^{-4} Cputime=4 NFE=70	J=27.77908798 m=69 eabs= 1.5×10^{-4} Cputime=7 NFE=70
0.3	J=27.84223178 m=71 eabs=0.0632929 Cputime<1 NFE=72	J=27.79289556 m=60 eabs= 1.0×10^{-2} Cputime<1 NFE=61	J=27.77953351 m=25 eabs= 6.0×10^{-4} Cputime=1 NFE=26	J=27.77908794 m=42 eabs= 1.5×10^{-4} Cputime=3 NFE=43	J=27.77908794 m=42 eabs= 1.5×10^{-4} Cputime=6 NFE=43
0.4	J=27.83208930 m=29 eabs=0.0531505 Cputime<1 NFE=30	J=27.79289466 m=28 eabs= 1.0×10^{-2} Cputime<1 NFE=29	J=27.77953068 m=21 eabs= 6.0×10^{-4} Cputime=1 NFE=22	J=27.77908794 m=29 eabs= 1.5×10^{-4} Cputime=2 NFE=30	J=27.77908794 m=29 eabs= 1.5×10^{-4} Cputime=4 NFE=30
0.5	J=27.83206541 m=29 eabs=0.0531266 Cputime<1 NFE=30	J=27.79289467 m=28 eabs= 1.0×10^{-2} Cputime<1 NFE=29	J=27.77953152 m=22 eabs= 6.0×10^{-4} Cputime=1 NFE=23	J=27.77908795 m=30 eabs= 1.5×10^{-4} Cputime=2 NFE=31	J=27.77908795 m=30 eabs= 1.5×10^{-4} Cputime=4 NFE=31
0.8	J=27.83307121 m=29 eabs=0.0541324 Cputime<1 NFE=30	J=27.79289467 m=28 eabs= 1.0×10^{-2} Cputime<1 NFE=29	J=27.77953152 m=22 eabs= 6.0×10^{-4} Cputime=1 NFE=23	J=27.77908796 m=30 eabs= 1.5×10^{-4} Cputime=2 NFE=31	J=27.77908796 m=30 eabs= 1.5×10^{-4} Cputime=4 NFE=31

TABLE (4.4.2): Results of GFS with change in u_0 .

u_0	m	J^*	ϵ_{abs}	Cputime	NFE
0	35	27.77908811	1.5×10^{-4}	3	36
-1	34	27.77908804	1.5×10^{-4}	3	35
-2	32	27.77908799	1.5×10^{-4}	2	33
-3	32	27.77908798	1.5×10^{-4}	2	33
-4	29	27.77908793	1.5×10^{-4}	2	30
-5	29	27.77908793	1.5×10^{-4}	2	30

TABLE (4.4.3): Results of SD with varying ϵ and N.

N ϵ	5	10	50	100	400
0.1	J=27.83211421 m=53 $e_{abs}=5.0 \times 10^{-2}$ Cputime<1 NFE=269	J=27.79292059 m=53 $e_{abs}=1.0 \times 10^{-2}$ Cputime<1 NFE=269	J=27.77953061 m=78 $e_{abs}=6.0 \times 10^{-4}$ Cputime=8 NFE=394	J=27.77908841 m=68 $e_{abs}=1.5 \times 10^{-4}$ Cputime=15 NFE=344	J=27.77908840 m=68 $e_{abs}=1.5 \times 10^{-4}$ Cputime=25 NFE=344
0.2	J=27.83210113 m=55 $e_{abs}=5.0 \times 10^{-2}$ Cputime<1 NFE=283	J=27.79289556 m=50 $e_{abs}=1.0 \times 10^{-2}$ Cputime<1 NFE=253	J=27.77952994 m=42 $e_{abs}=6.0 \times 10^{-4}$ Cputime=5 NFE=213	J=27.77908790 m=45 $e_{abs}=1.5 \times 10^{-4}$ Cputime=14 NFE=228	J=27.77908790 m=45 $e_{abs}=1.5 \times 10^{-4}$ Cputime=23 NFE=228
0.3	J=27.83210113 m=26 $e_{abs}=5.0 \times 10^{-2}$ Cputime<1 NFE=142	J=27.79289556 m=24 $e_{abs}=1.0 \times 10^{-2}$ Cputime<1 NFE=122	J=27.77952985 m=20 $e_{abs}=6.0 \times 10^{-4}$ Cputime=3 NFE=102	J=27.77908790 m=45 $e_{abs}=1.5 \times 10^{-4}$ Cputime=11 NFE=227	J=27.77908790 m=45 $e_{abs}=1.5 \times 10^{-4}$ Cputime=20 NFE=227
0.4	J=27.83208930 m=21 $e_{abs}=5.0 \times 10^{-2}$ Cputime<1 NFE=97	J=27.79289496 m=15 $e_{abs}=1.0 \times 10^{-2}$ Cputime<1 NFE=77	J=27.77953026 m=15 $e_{abs}=6.0 \times 10^{-4}$ Cputime=2 NFE=77	J=27.77908783 m=21 $e_{abs}=1.5 \times 10^{-4}$ Cputime=6 NFE=107	J=27.77908783 m=21 $e_{abs}=1.5 \times 10^{-4}$ Cputime=12 NFE=107
0.5	J=27.83209130 m=15 $e_{abs}=5.0 \times 10^{-2}$ Cputime<1 NFE=77	J=27.79289499 m=15 $e_{abs}=1.0 \times 10^{-2}$ Cputime<1 NFE=77	J=27.77953031 m=15 $e_{abs}=6.0 \times 10^{-4}$ Cputime=2 NFE=77	J=27.77908790 m=21 $e_{abs}=1.5 \times 10^{-4}$ Cputime=6 NFE=107	J=27.77908789 m=21 $e_{abs}=1.5 \times 10^{-4}$ Cputime=12 NFE=107
0.8	J=27.83209241 m=17 $e_{abs}=5.0 \times 10^{-2}$ Cputime<1 NFE=87	J=27.7936722 m=15 $e_{abs}=1.0 \times 10^{-2}$ Cputime<1 NFE=77	J=27.77954121 m=15 $e_{abs}=6.0 \times 10^{-4}$ Cputime=2 NFE=77	J=27.77908864 m=21 $e_{abs}=1.5 \times 10^{-4}$ Cputime=6 NFE=107	J=27.77908880 m=21 $e_{abs}=1.5 \times 10^{-4}$ Cputime=12 NFE=107

TABLE (4.4.4): Results of SD with change in u_0 .

u_0	m	J^*	e_{abs}	Cputime	NFE
0	24	27.77908798	1.5×10^{-4}	6	122
-1	23	27.77908799	1.5×10^{-4}	6	117
-2	23	27.77908806	1.5×10^{-4}	6	117
-3	23	27.77908804	1.5×10^{-4}	6	117
-4	21	27.77908783	1.5×10^{-4}	6	107
-5	21	27.77908788	1.5×10^{-4}	6	107

TABLE (4.4.5): Results of FR with varying ϵ and N.

N ϵ	5	10	50	100	400
0.1	J=27.83209137 m=30 $e_{abs}=5.0 \times 10^{-2}$ Cputime<1 NFE=154	J=27.79289562 m=30 $e_{abs}=1.0 \times 10^{-2}$ Cputime<1 NFE=154	J=27.77953059 m=22 $e_{abs}=6.0 \times 10^{-4}$ Cputime=3 NFE=114	J=27.77908788 m=30 $e_{abs}=1.5 \times 10^{-4}$ Cputime=7 NFE=154	J=27.77908787 m=30 $e_{abs}=1.5 \times 10^{-4}$ Cputime=12 NFE=154
0.2	J=27.83209026 m=29 $e_{abs}=5.0 \times 10^{-2}$ Cputime<1 NFE=151	J=27.79289556 m=25 $e_{abs}=1.0 \times 10^{-2}$ Cputime<1 NFE=128	J=27.77953006 m=17 $e_{abs}=6.0 \times 10^{-4}$ Cputime=2 NFE=88	J=27.77908787 m=15 $e_{abs}=1.5 \times 10^{-4}$ Cputime=4 NFE=78	J=27.77908787 m=14 $e_{abs}=1.5 \times 10^{-4}$ Cputime=7 NFE=74
0.3	J=27.83209026 m=26 $e_{abs}=5.0 \times 10^{-2}$ Cputime<1 NFE=129	J=27.79289556 m=20 $e_{abs}=1.0 \times 10^{-2}$ Cputime<1 NFE=102	J=27.77953026 m=15 $e_{abs}=6.0 \times 10^{-4}$ Cputime=2 NFE=77	J=27.77908789 m=12 $e_{abs}=1.5 \times 10^{-4}$ Cputime=3 NFE=62	J=27.77908789 m=12 $e_{abs}=1.5 \times 10^{-4}$ Cputime=5 NFE=62
0.4	J=27.83209026 m=8 $e_{abs}=5.0 \times 10^{-2}$ Cputime<1 NFE=42	J=27.79289556 m=10 $e_{abs}=1.0 \times 10^{-2}$ Cputime<1 NFE=52	J=27.77953026 m=15 $e_{abs}=6.0 \times 10^{-4}$ Cputime=2 NFE=77	J=27.77908774 m=8 $e_{abs}=1.5 \times 10^{-4}$ Cputime=3 NFE=42	J=27.77908774 m=8 $e_{abs}=1.5 \times 10^{-4}$ Cputime=5 NFE=42
0.5	J=27.83209256 m=11 $e_{abs}=5.0 \times 10^{-2}$ Cputime<1 NFE=54	J=27.79289556 m=11 $e_{abs}=1.0 \times 10^{-2}$ Cputime<1 NFE=54	J=27.77953028 m=16 $e_{abs}=6.0 \times 10^{-4}$ Cputime=2 NFE=80	J=27.77908781 m=9 $e_{abs}=1.5 \times 10^{-4}$ Cputime=3 NFE=44	J=27.77908780 m=9 $e_{abs}=1.5 \times 10^{-4}$ Cputime=5 NFE=44
0.8	J=27.83209256 m=12 $e_{abs}=5.0 \times 10^{-2}$ Cputime<1 NFE=64	J=27.79289563 m=12 $e_{abs}=1.0 \times 10^{-2}$ Cputime<1 NFE=64	J=27.77953031 m=17 $e_{abs}=6.0 \times 10^{-4}$ Cputime=2 NFE=90	J=27.77908801 m=8 $e_{abs}=1.5 \times 10^{-4}$ Cputime=3 NFE=43	J=27.77908512 m=8 $e_{abs}=1.5 \times 10^{-4}$ Cputime=5 NFE=43

TABLE (4.4.6): Results of FR with change in u_0 .

u_0	m	J^*	e_{abs}	Cputime	NFE
0	18	27.77908790	1.5×10^{-4}	5	92
-1	23	27.77908802	1.5×10^{-4}	6	117
-2	9	27.77908789	1.5×10^{-4}	2	47
-3	8	27.77908778	1.5×10^{-4}	2	42
-4	8	27.77908774	1.5×10^{-4}	2	42
-5	10	27.77908783	1.5×10^{-4}	3	52

TABLE (4.4.7): Results of PR with varying ϵ and N.

N ϵ	5	10	50	100	400
0.1	J=27.8321882 m=71 eabs= 5.0×10^{-2} Cputime<1 NFE=375	J=27.79290468 m=69 eabs= 1.0×10^{-2} Cputime<1 NFE=349	J=27.77952985 m=93 eabs= 6.0×10^{-4} Cputime=12 NFE=469	J=27.77909549 m=50 eabs= 1.6×10^{-4} Cputime=12 NFE=254	J=27.7790547 m=49 eabs= 1.5×10^{-4} Cputime=23 NFE=245
0.2	J=27.83208931 m=25 eabs= 5.0×10^{-2} Cputime<1 NFE=253	J=27.79289585 m=25 eabs= 1.0×10^{-2} Cputime<1 NFE=253	J=27.77953015 m=55 eabs= 6.0×10^{-4} Cputime=8 NFE=278	J=27.77908799 m=40 eabs= 1.5×10^{-4} Cputime=10 NFE=203	J=27.77908799 m=40 eabs= 1.5×10^{-4} Cputime=18 NFE=203
0.3	J=27.83208931 m=18 eabs= 5.0×10^{-2} Cputime<1 NFE=91	J=27.79289556 m=40 eabs= 1.0×10^{-2} Cputime<1 NFE=102	J=27.77952980 m=22 eabs= 6.0×10^{-4} Cputime=3 NFE=112	J=27.77908794 m=19 eabs= 1.5×10^{-4} Cputime=5 NFE=97	J=27.77908794 m=19 eabs= 1.5×10^{-4} Cputime=9 NFE=97
0.4	J=27.83208930 m=17 eabs= 5.0×10^{-2} Cputime<1 NFE=87	J=27.79289556 m=25 eabs= 1.0×10^{-2} Cputime<1 NFE=127	J=27.77952929 m=18 eabs= 6.0×10^{-4} Cputime=2 NFE=92	J=27.77908793 m=17 eabs= 1.5×10^{-4} Cputime=5 NFE=87	J=27.77908793 m=17 eabs= 1.5×10^{-4} Cputime=9 NFE=87
0.5	J=27.83208932 m=20 eabs= 5.0×10^{-2} Cputime<1 NFE=105	J=27.79289556 m=27 eabs= 1.0×10^{-2} Cputime<1 NFE=131	J=27.77952980 m=20 eabs= 6.0×10^{-4} Cputime=3 NFE=98	J=27.77908794 m=19 eabs= 1.5×10^{-4} Cputime=5 NFE=97	J=27.77908804 m=19 eabs= 1.5×10^{-4} Cputime=9 NFE=97
0.8	J=27.83208941 m=20 eabs= 5.0×10^{-2} Cputime<1 NFE=106	J=27.79289565 m=27 eabs= 1.0×10^{-2} Cputime<1 NFE=132	J=27.77952985 m=21 eabs= 6.0×10^{-4} Cputime=3 NFE=108	J=27.77908804 m=17 eabs= 1.5×10^{-4} Cputime=5 NFE=89	J=27.77908804 m=17 eabs= 1.5×10^{-4} Cputime=9 NFE=89

TABLE (4.4.8): Results of PR with change in u_0 .

u_0	m	J^*	e_{abs}	Cputime	NFE
0	18	27.77908807	1.5×10^{-4}	5	92
-1	23	27.77908798	1.5×10^{-4}	6	117
-2	23	27.779087806	1.5×10^{-4}	6	117
-3	23	27.77908805	1.5×10^{-4}	6	117
-4	17	27.77908793	1.5×10^{-4}	5	87
-5	17	27.77908818	1.5×10^{-4}	5	87

TABLE (4.4.9): Results of H1 with varying ϵ and N.

N ϵ	5	10	50	100	400
0.1	J=27.83209137 m=30 eabs= 5.0×10^{-2} Cputime<1 NFE=154	J=27.79289562 m=30 eabs= 1.0×10^{-2} Cputime<1 NFE=154	J=27.77953059 m=22 eabs= 6.0×10^{-4} Cputime=3 NFE=114	J=27.77908788 m=30 eabs= 1.5×10^{-4} Cputime=7 NFE=154	J=27.77908787 m=30 eabs= 1.5×10^{-4} Cputime=12 NFE=154
0.2	J=27.83209026 m=29 eabs= 5.0×10^{-2} Cputime<1 NFE=151	J=27.79289556 m=25 eabs= 1.0×10^{-2} Cputime<1 NFE=128	J=27.77953006 m=17 eabs= 6.0×10^{-4} Cputime=2 NFE=88	J=27.77908787 m=15 eabs= 1.5×10^{-4} Cputime=4 NFE=78	J=27.77908787 m=14 eabs= 1.5×10^{-4} Cputime=7 NFE=74
0.3	J=27.83209026 m=26 eabs= 5.0×10^{-2} Cputime<1 NFE=129	J=27.79289556 m=20 eabs= 1.0×10^{-2} Cputime<1 NFE=102	J=27.77953026 m=15 eabs= 6.0×10^{-4} Cputime=2 NFE=77	J=27.77908789 m=12 eabs= 1.5×10^{-4} Cputime=3 NFE=62	J=27.77908789 m=12 eabs= 1.5×10^{-4} Cputime=5 NFE=62
0.4	J=27.83209252 m=8 eabs= 5.0×10^{-2} Cputime<1 NFE=42	J=27.79289556 m=10 eabs= 1.0×10^{-2} Cputime<1 NFE=52	J=27.77953026 m=15 eabs= 6.0×10^{-4} Cputime=2 NFE=77	J=27.77908774 m=8 eabs= 1.5×10^{-4} Cputime=3 NFE=42	J=27.77908774 m=8 eabs= 1.5×10^{-4} Cputime=5 NFE=42
0.5	J=27.83209253 m=12 eabs= 5.0×10^{-2} Cputime<1 NFE=64	J=27.79289556 m=11 eabs= 1.0×10^{-2} Cputime<1 NFE=54	J=27.77953028 m=16 eabs= 6.0×10^{-4} Cputime=2 NFE=80	J=27.77908781 m=9 eabs= 1.5×10^{-4} Cputime=3 NFE=44	J=27.77908780 m=9 eabs= 1.5×10^{-4} Cputime=5 NFE=44
0.8	J=27.83209256 m=12 eabs= 5.0×10^{-2} Cputime<1 NFE=64	J=27.79289563 m=12 eabs= 1.0×10^{-2} Cputime<1 NFE=64	J=27.77953031 m=17 eabs= 6.0×10^{-4} Cputime=2 NFE=90	J=27.77908801 m=8 eabs= 1.5×10^{-4} Cputime=3 NFE=43	J=27.77908512 m=8 eabs= 1.5×10^{-4} Cputime=5 NFE=43

TABLE (4.4.10): Results of H1 with change in u_0 .

u_0	m	J^*	e_{abs}	Cputime	NFE
0	18	27.77908807	1.5×10^{-4}	5	92
-1	23	27.77908798	1.5×10^{-4}	6	117
-2	8	27.77908776	1.5×10^{-4}	2	42
-3	8	27.77908778	1.5×10^{-4}	2	42
-4	8	27.77908774	1.5×10^{-4}	2	42
-5	10	27.77908783	1.5×10^{-4}	3	52

TABLE (4.4.11): Results of ATH with varying ϵ and N.

N ϵ	5	10	50	100	400
0.1	J=27.83209137 m=30 eabs= 5.0×10^{-2} Cputime<1 NFE=154	J=27.79289562 m=30 eabs= 1.0×10^{-2} Cputime<1 NFE=154	J=27.77953059 m=22 eabs= 6.0×10^{-4} Cputime=3 NFE=114	J=27.77908788 m=30 eabs= 1.5×10^{-4} Cputime=7 NFE=154	J=27.77908787 m=30 eabs= 1.5×10^{-4} Cputime=12 NFE=154
0.2	J=27.83209026 m=29 eabs= 5.0×10^{-2} Cputime<1 NFE=151	J=27.79289556 m=25 eabs= 1.0×10^{-2} Cputime<1 NFE=128	J=27.77953006 m=17 eabs= 6.0×10^{-4} Cputime=2 NFE=88	J=27.77908787 m=15 eabs= 1.5×10^{-4} Cputime=4 NFE=78	J=27.77908787 m=14 eabs= 1.5×10^{-4} Cputime=7 NFE=74
0.3	J=27.83209026 m=26 eabs= 5.0×10^{-2} Cputime<1 NFE=129	J=27.79289556 m=20 eabs= 1.0×10^{-2} Cputime<1 NFE=102	J=27.77953026 m=15 eabs= 6.0×10^{-4} Cputime=2 NFE=77	J=27.77908789 m=12 eabs= 1.5×10^{-4} Cputime=3 NFE=62	J=27.77908789 m=12 eabs= 1.5×10^{-4} Cputime=5 NFE=62
0.4	J=27.83208930 m=8 eabs= 5.0×10^{-2} Cputime<1 NFE=42	J=27.79289556 m=10 eabs= 1.0×10^{-2} Cputime<1 NFE=52	J=27.77953026 m=15 eabs= 6.0×10^{-4} Cputime=2 NFE=77	J=27.77908774 m=8 eabs= 1.5×10^{-4} Cputime=3 NFE=42	J=27.77908774 m=8 eabs= 1.5×10^{-4} Cputime=5 NFE=42
0.5	J=27.83208931 m=12 eabs= 5.0×10^{-2} Cputime<1 NFE=64	J=27.79289556 m=11 eabs= 1.0×10^{-2} Cputime<1 NFE=54	J=27.77953028 m=16 eabs= 6.0×10^{-4} Cputime=2 NFE=80	J=27.77908781 m=9 eabs= 1.5×10^{-4} Cputime=3 NFE=44	J=27.77908780 m=9 eabs= 1.5×10^{-4} Cputime=5 NFE=44
0.8	J=27.83209256 m=12 eabs= 5.0×10^{-2} Cputime<1 NFE=64	J=27.79289563 m=12 eabs= 1.0×10^{-2} Cputime<1 NFE=64	J=27.77953031 m=17 eabs= 6.0×10^{-4} Cputime=2 NFE=90	J=27.77908801 m=8 eabs= 1.5×10^{-4} Cputime=3 NFE=43	J=27.77908512 m=8 eabs= 1.5×10^{-4} Cputime=5 NFE=43

TABLE (4.4.12): Results of ATH with change in u_0 .

u_0	m	J^*	e_{abs}	Cputime	NFE
0	18	27.77908807	1.5×10^{-4}	5	92
-1	23	27.77908798	1.5×10^{-4}	6	117
-2	8	27.77908776	1.5×10^{-4}	2	42
-3	8	27.77908778	1.5×10^{-4}	2	42
-4	8	27.77908774	1.5×10^{-4}	2	42
-5	10	27.77908783	1.5×10^{-4}	3	52

TABLE (4.4.13): Results of H3 with varying ϵ and N.

N ϵ	5	10	50	100	400
0.1	J=27.83209137 m=30 $e_{abs}=5.0 \times 10^{-2}$ Cputime<1 NFE=154	J=27.79289562 m=30 $e_{abs}=1.0 \times 10^{-2}$ Cputime<1 NFE=154	J=27.77953059 m=22 $e_{abs}=6.0 \times 10^{-4}$ Cputime=3 NFE=114	J=27.77908788 m=30 $e_{abs}=1.5 \times 10^{-4}$ Cputime=7 NFE=154	J=27.77908787 m=30 $e_{abs}=1.5 \times 10^{-4}$ Cputime=12 NFE=154
0.2	J=27.83209026 m=29 $e_{abs}=5.0 \times 10^{-2}$ Cputime<1 NFE=151	J=27.79289556 m=25 $e_{abs}=1.0 \times 10^{-2}$ Cputime<1 NFE=128	J=27.77953006 m=17 $e_{abs}=6.0 \times 10^{-4}$ Cputime=2 NFE=88	J=27.77908787 m=15 $e_{abs}=1.5 \times 10^{-4}$ Cputime=4 NFE=78	J=27.77908787 m=14 $e_{abs}=1.5 \times 10^{-4}$ Cputime=7 NFE=74
0.3	J=27.83209026 m=26 $e_{abs}=5.0 \times 10^{-2}$ Cputime<1 NFE=129	J=27.79289556 m=20 $e_{abs}=1.0 \times 10^{-2}$ Cputime<1 NFE=102	J=27.77953026 m=15 $e_{abs}=6.0 \times 10^{-4}$ Cputime=2 NFE=77	J=27.77908789 m=12 $e_{abs}=1.5 \times 10^{-4}$ Cputime=3 NFE=62	J=27.77908789 m=12 $e_{abs}=1.5 \times 10^{-4}$ Cputime=5 NFE=62
0.4	J=27.83208930 m=8 $e_{abs}=5.0 \times 10^{-2}$ Cputime<1 NFE=42	J=27.79289556 m=10 $e_{abs}=1.0 \times 10^{-2}$ Cputime<1 NFE=52	J=27.77953026 m=15 $e_{abs}=6.0 \times 10^{-4}$ Cputime=2 NFE=77	J=27.77908774 m=8 $e_{abs}=1.5 \times 10^{-4}$ Cputime=3 NFE=42	J=27.77908774 m=8 $e_{abs}=1.5 \times 10^{-4}$ Cputime=5 NFE=42
0.5	J=27.83208931 m=12 $e_{abs}=5.0 \times 10^{-2}$ Cputime<1 NFE=64	J=27.79289556 m=11 $e_{abs}=1.0 \times 10^{-2}$ Cputime<1 NFE=54	J=27.77953028 m=16 $e_{abs}=6.0 \times 10^{-4}$ Cputime=2 NFE=80	J=27.77908781 m=9 $e_{abs}=1.5 \times 10^{-4}$ Cputime=3 NFE=44	J=27.77908780 m=9 $e_{abs}=1.5 \times 10^{-4}$ Cputime=5 NFE=44
0.8	J=27.83209256 m=12 $e_{abs}=5.0 \times 10^{-2}$ Cputime<1 NFE=64	J=27.79289563 m=12 $e_{abs}=1.0 \times 10^{-2}$ Cputime<1 NFE=64	J=27.77953031 m=17 $e_{abs}=6.0 \times 10^{-4}$ Cputime=2 NFE=90	J=27.77908801 m=8 $e_{abs}=1.5 \times 10^{-4}$ Cputime=3 NFE=43	J=27.77908512 m=8 $e_{abs}=1.5 \times 10^{-4}$ Cputime=5 NFE=43

TABLE (4.4.14): Results of H3 with change in u_0 .

u_0	m	J^*	e_{abs}	Cputime	NFE
0	18	27.77908807	1.5×10^{-4}	5	92
-1	23	27.77908798	1.5×10^{-4}	6	117
-2	8	27.77908776	1.5×10^{-4}	2	42
-3	8	27.77908778	1.5×10^{-4}	2	42
-4	8	27.77908774	1.5×10^{-4}	2	42
-5	10	27.77908783	1.5×10^{-4}	3	52

Table (4.5.1): Summary table for the seven methods

method μ_0	GFS	SD	FR	PR	H1	ATH	H3
0	$x_2(t_f) = 27.77908811$ $m = 35$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 3 NFE = 36	$x_2(t_f) = 27.77908798$ $m = 24$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 6 NFE = 122	$x_2(t_f) = 27.77908790$ $m = 18$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 5 NFE = 92	$x_2(t_f) = 27.77908807$ $m = 18$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 5 NFE = 92	$x_2(t_f) = 27.77908807$ $m = 18$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 5 NFE = 92	$x_2(t_f) = 27.77908807$ $m = 18$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 5 NFE = 92 $\tau = 0.00001$	$x_2(t_f) = 27.77908807$ $m = 18$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 5 NFE = 92 $\lambda = 0.00001, \mu = 0.45$
-1	$x_2(t_f) = 27.77908804$ $m = 34$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 3 NFE = 35	$x_2(t_f) = 27.77908799$ $m = 23$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 6 NFE = 117	$x_2(t_f) = 27.77908802$ $m = 23$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 6 NFE = 117	$x_2(t_f) = 27.77908798$ $m = 23$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 6 NFE = 117	$x_2(t_f) = 27.77908798$ $m = 23$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 6 NFE = 117	$x_2(t_f) = 27.77908798$ $m = 23$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 6 NFE = 117 $\tau = 0.00001$	$x_2(t_f) = 27.77908798$ $m = 23$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 6 NFE = 117 $\lambda = 0.00001, \mu = 0.45$
-2	$x_2(t_f) = 27.77908799$ $m = 34$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 2 NFE = 33	$x_2(t_f) = 27.77908806$ $m = 23$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 6 NFE = 117	$x_2(t_f) = 27.77908789$ $m = 9$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 2 NFE = 47	$x_2(t_f) = 27.77908806$ $m = 23$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 6 NFE = 117	$x_2(t_f) = 27.77908776$ $m = 8$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 2 NFE = 42	$x_2(t_f) = 27.77908776$ $m = 8$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 2 NFE = 42 $\tau = 0.00001$	$x_2(t_f) = 27.77908776$ $m = 8$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 2 NFE = 42 $\lambda = 0.00001, \mu = 0.45$
-3	$x_2(t_f) = 27.77908798$ $m = 32$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 2 NFE = 30	$x_2(t_f) = 27.77908804$ $m = 23$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 6 NFE = 117	$x_2(t_f) = 27.77908778$ $m = 8$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 2 NFE = 42	$x_2(t_f) = 27.77908805$ $m = 23$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 6 NFE = 117	$x_2(t_f) = 27.77908778$ $m = 8$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 2 NFE = 42	$x_2(t_f) = 27.77908778$ $m = 8$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 2 NFE = 42 $\tau = 0.00001$	$x_2(t_f) = 27.77908778$ $m = 8$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 2 NFE = 42 $\lambda = 0.00001, \mu = 0.45$
-4	$x_2(t_f) = 27.77908793$ $m = 29$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 2 NFE = 30	$x_2(t_f) = 27.77908783$ $m = 21$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 6 NFE = 107	$x_2(t_f) = 27.77908774$ $m = 8$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 2 NFE = 42	$x_2(t_f) = 27.77908793$ $m = 17$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 5 NFE = 87	$x_2(t_f) = 27.77908774$ $m = 8$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 2 NFE = 42	$x_2(t_f) = 27.77908774$ $m = 8$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 2 NFE = 42 $\tau = 0.00001$	$x_2(t_f) = 27.77908774$ $m = 8$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 2 NFE = 42 $\lambda = 0.00001, \mu = 0.45$
-5	$x_2(t_f) = 27.77908793$ $m = 29$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 2 NFE = 30	$x_2(t_f) = 27.77908788$ $m = 21$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 6 NFE = 107	$x_2(t_f) = 27.77908783$ $m = 10$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 3 NFE = 52	$x_2(t_f) = 27.77908818$ $m = 17$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 5 NFE = 87	$x_2(t_f) = 27.77908783$ $m = 10$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 3 NFE = 52	$x_2(t_f) = 27.77908783$ $m = 10$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 3 NFE = 52 $\tau = 0.00001$	$x_2(t_f) = 27.77908783$ $m = 10$ $e_{abs} = 1.5 \times 10^{-4}$ Cputime = 3 NFE = 52 $\lambda = 0.00001, \mu = 0.45$

GFS

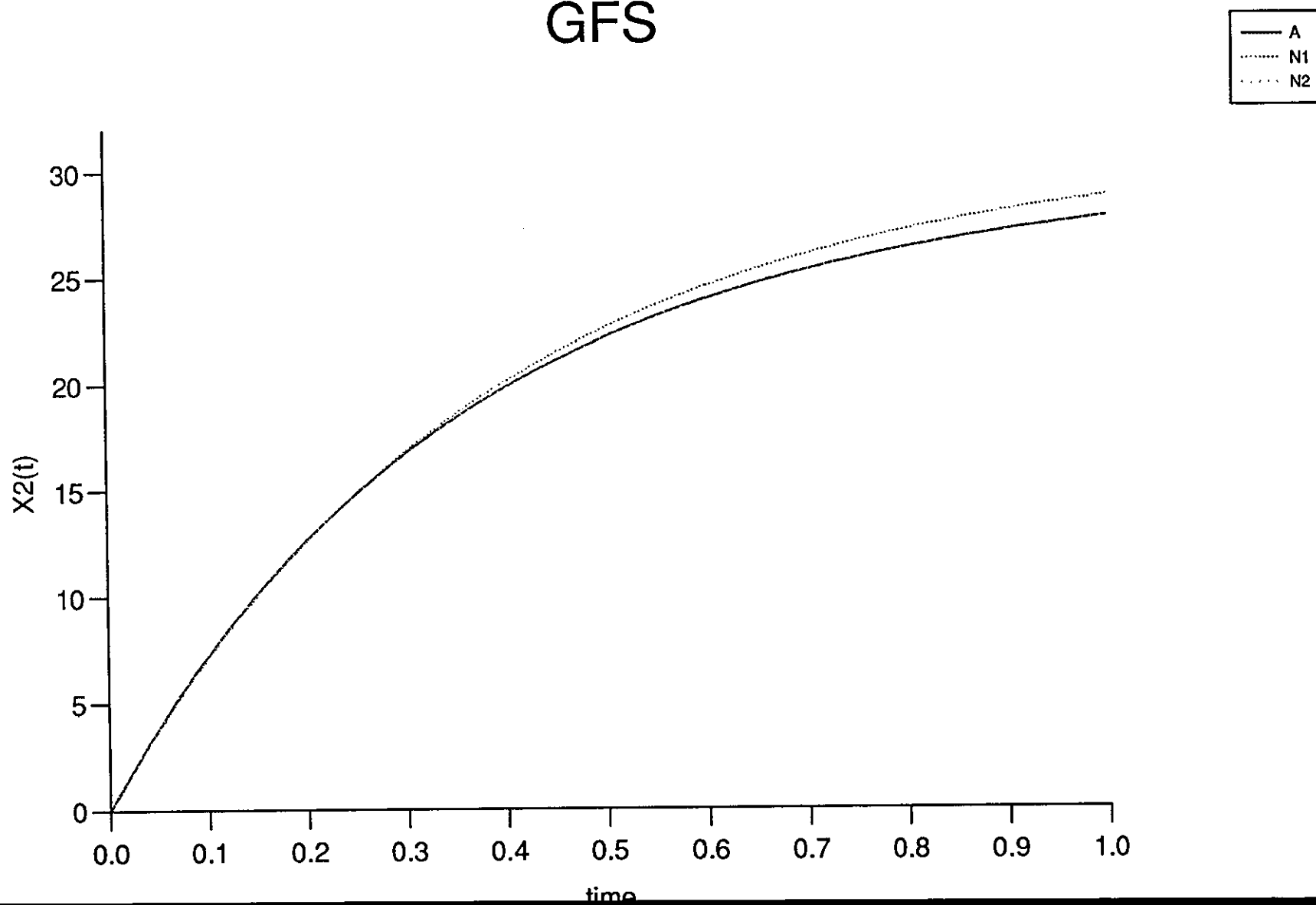


Figure (4.4.1)

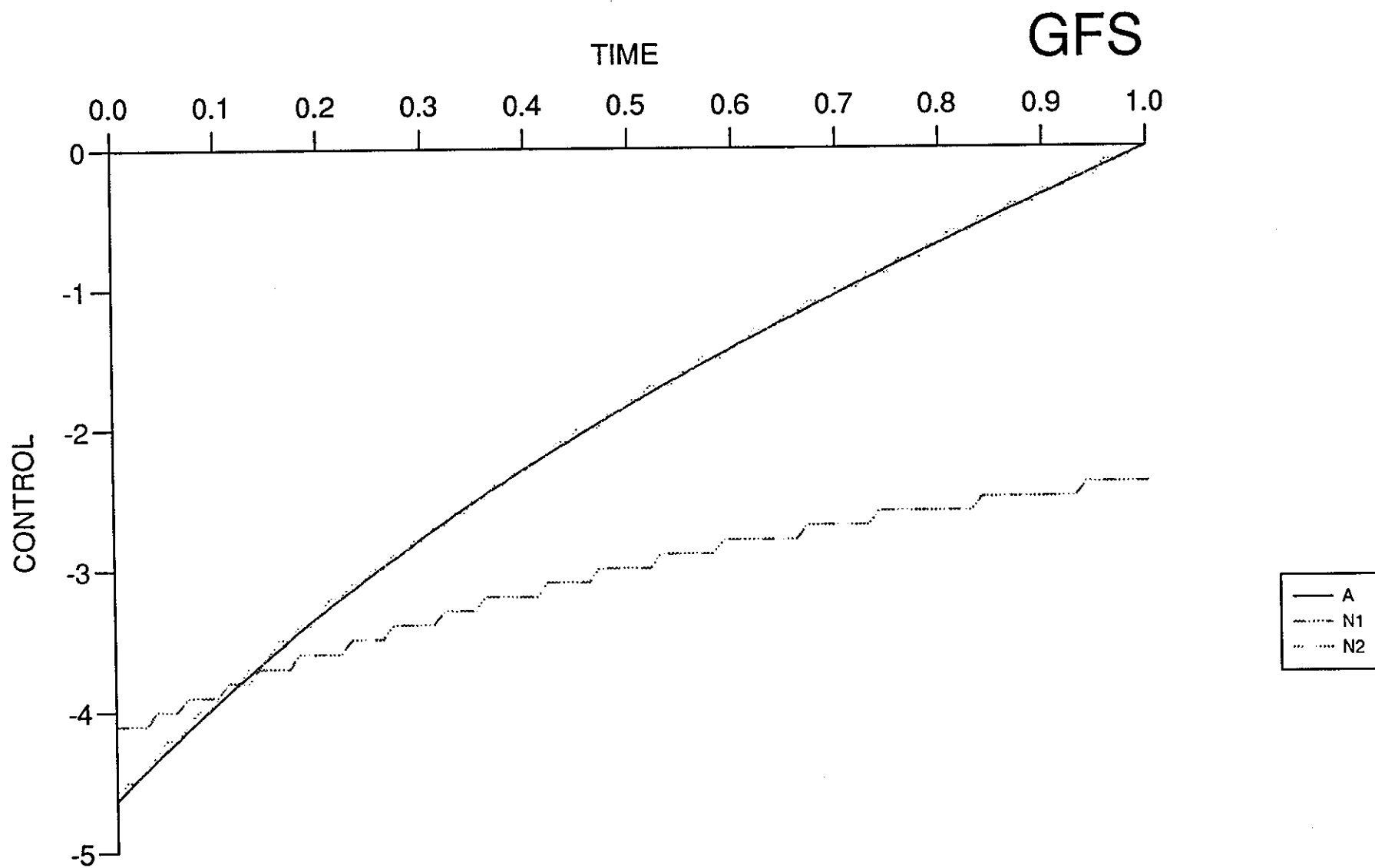


Figure (4.4.2)

GFS

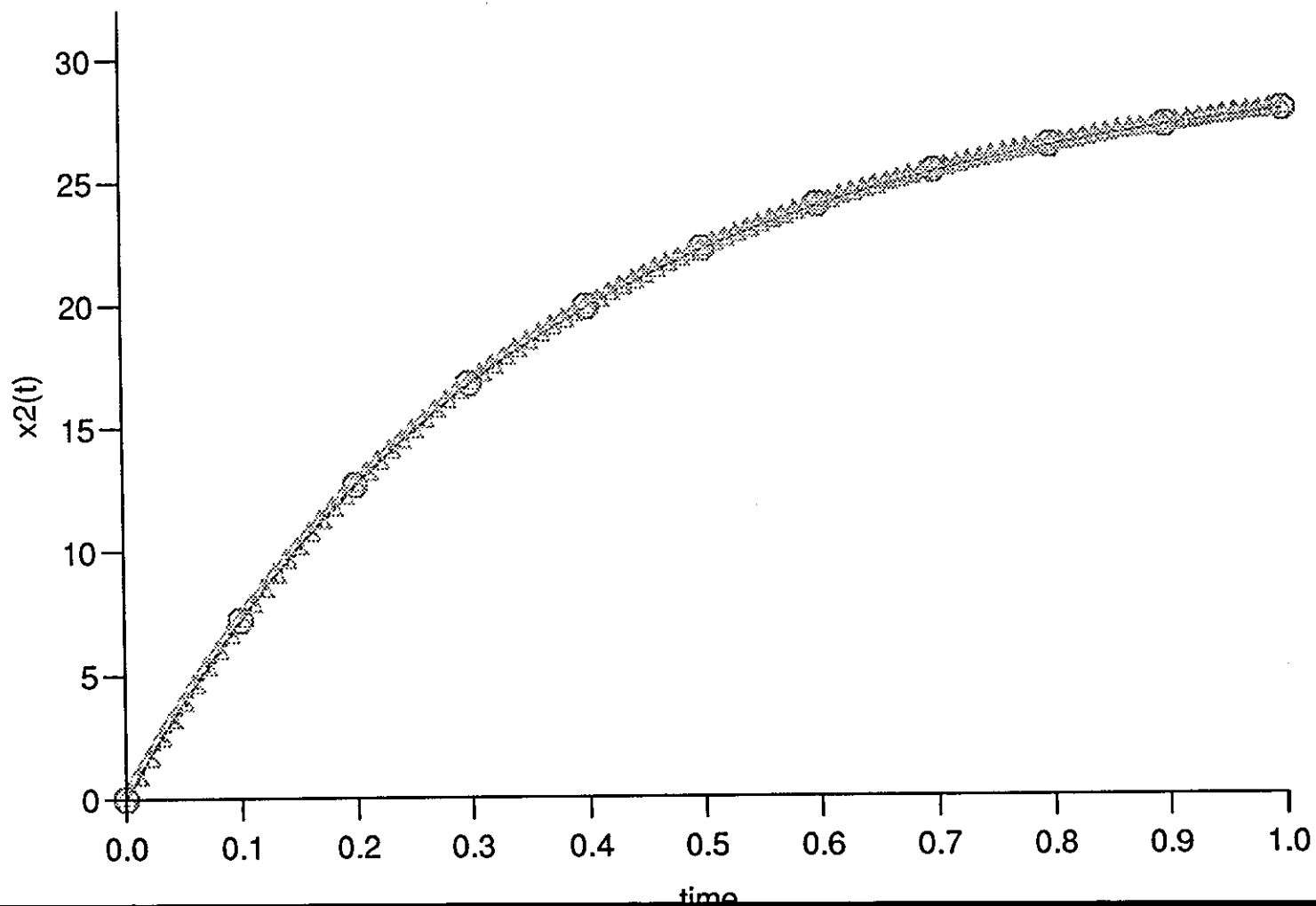
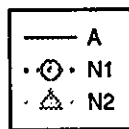


Figure (4.4.3)

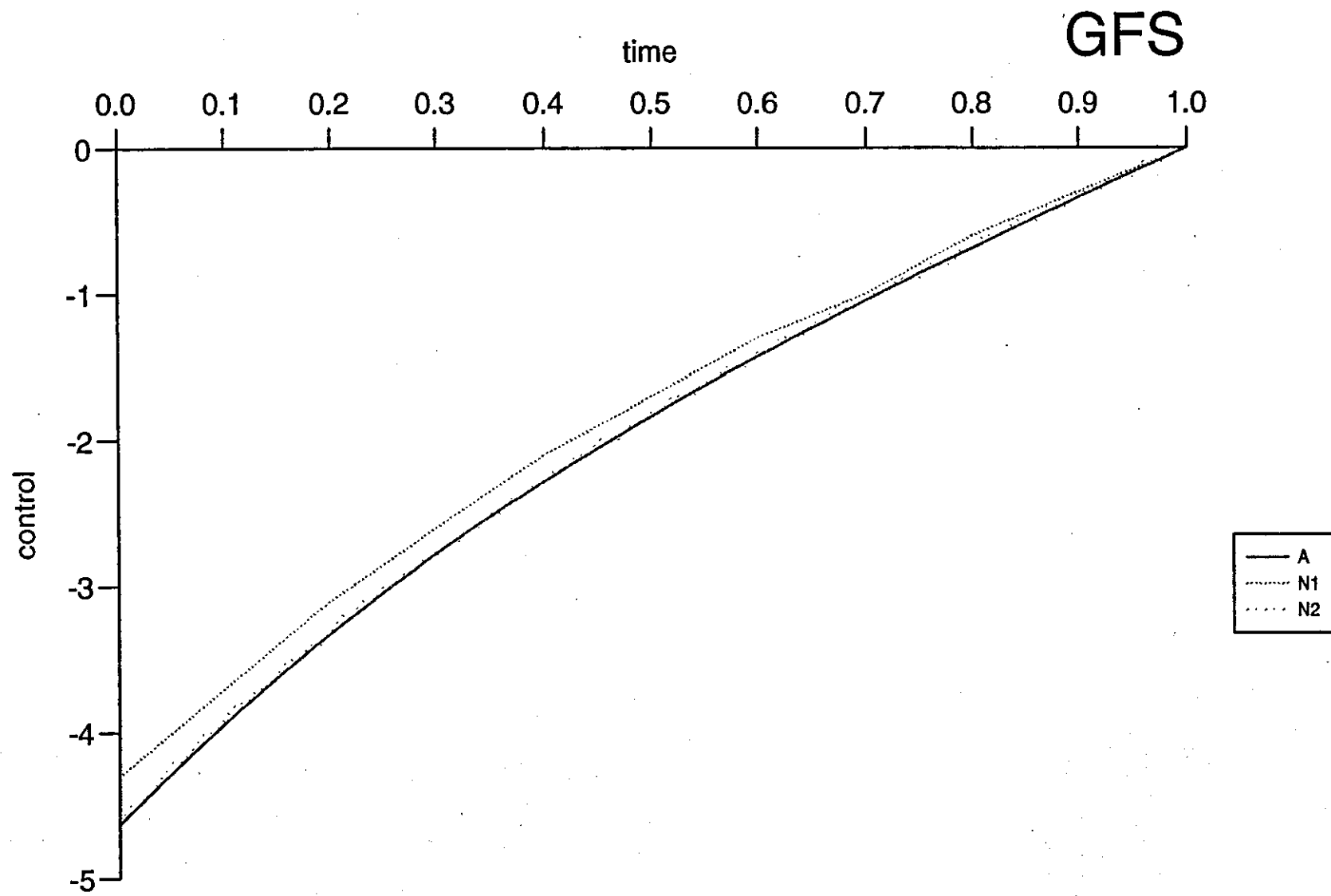


Figure (4.4.4)

GFS

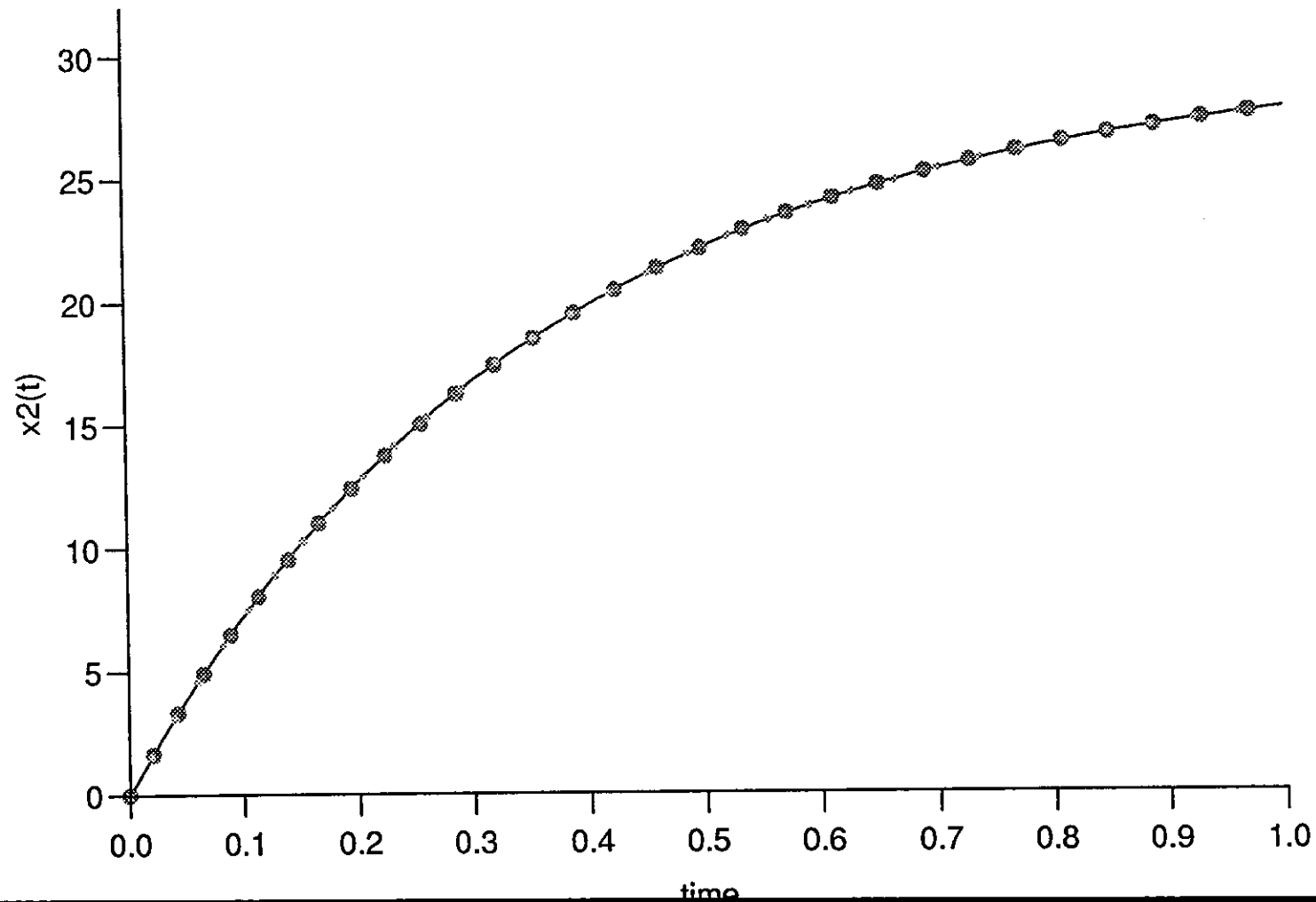


Figure (4.4.5)

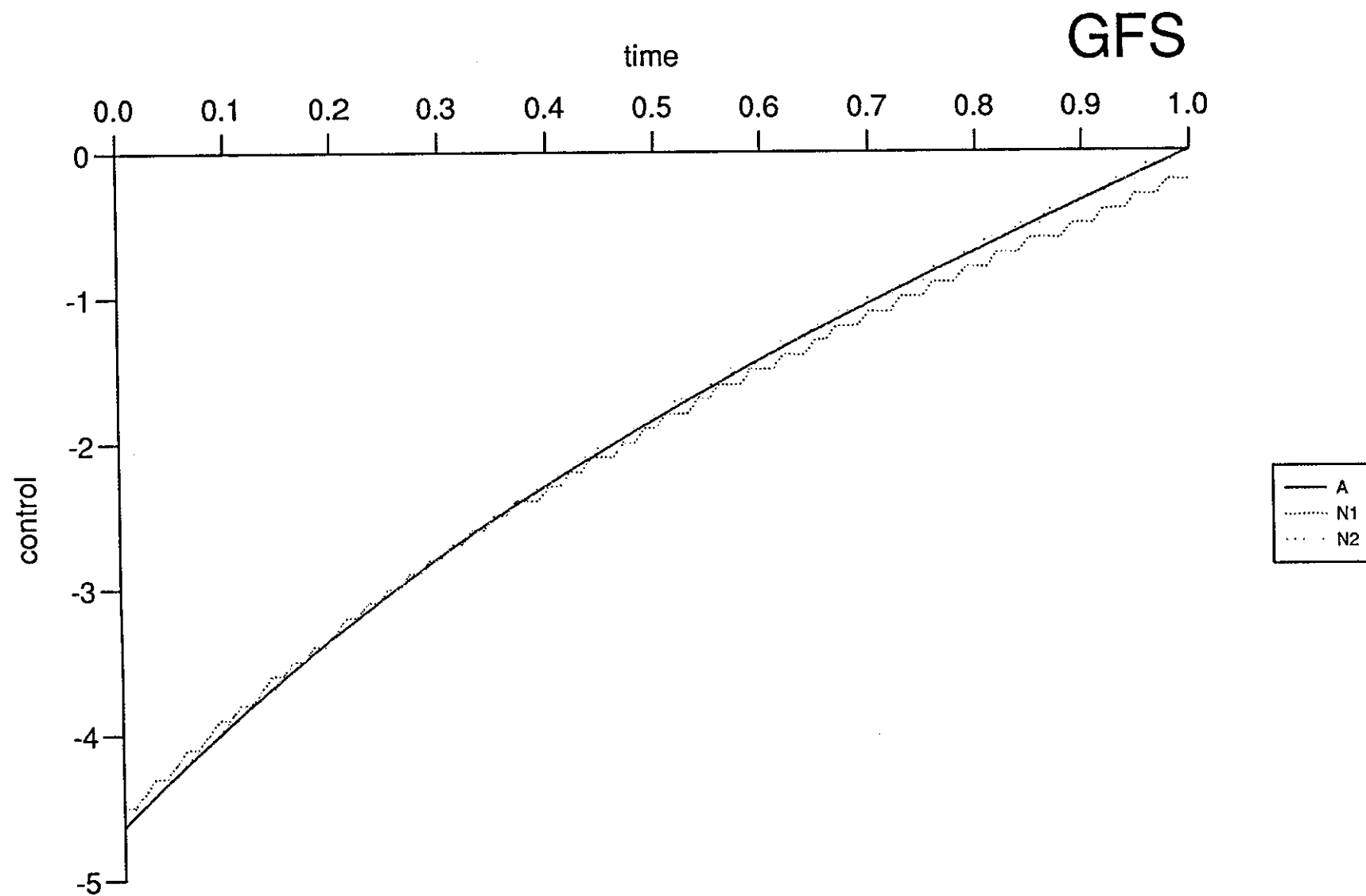


Figure (4.4.6)

GFS

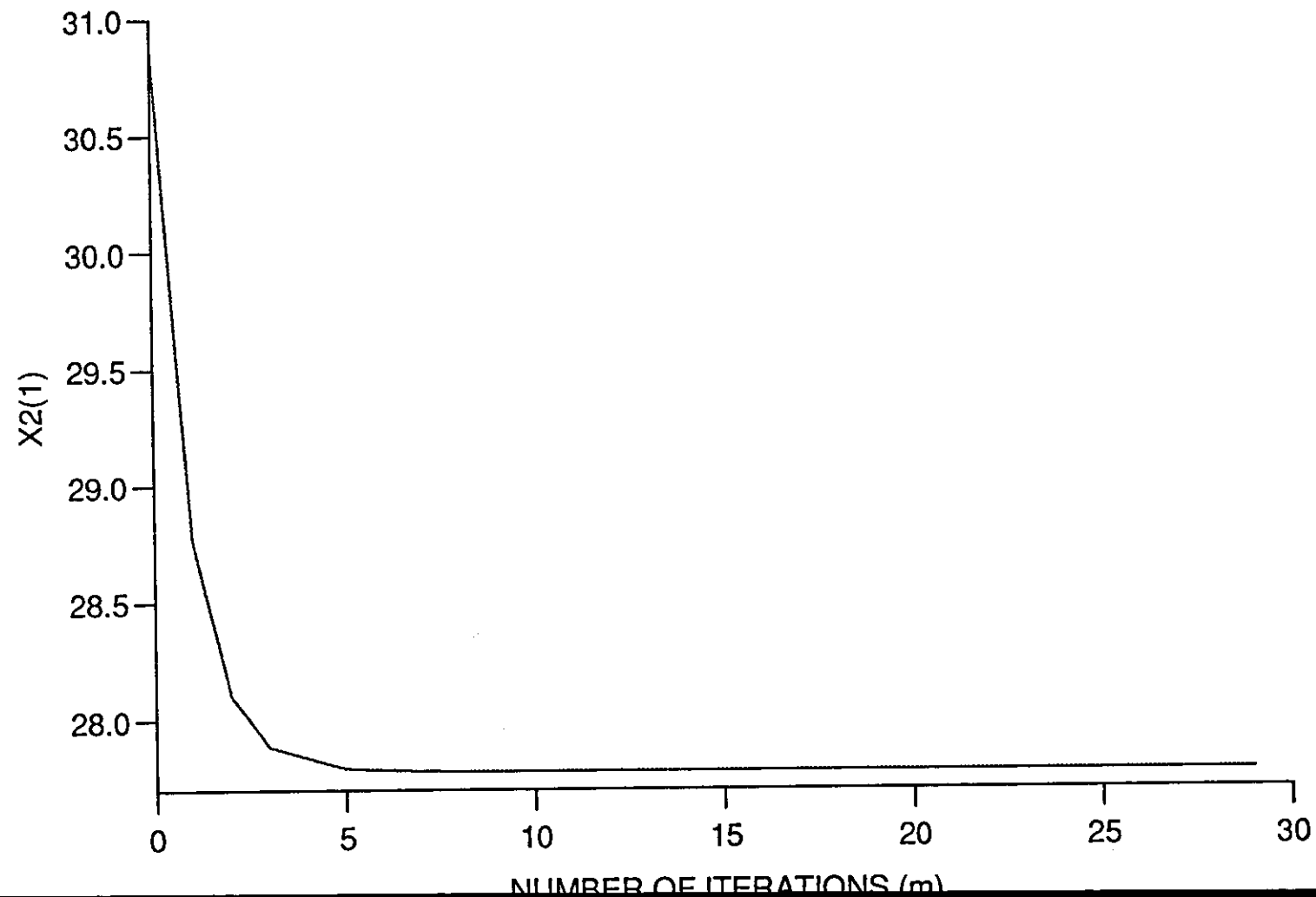


Figure (4.4.7)

GFS

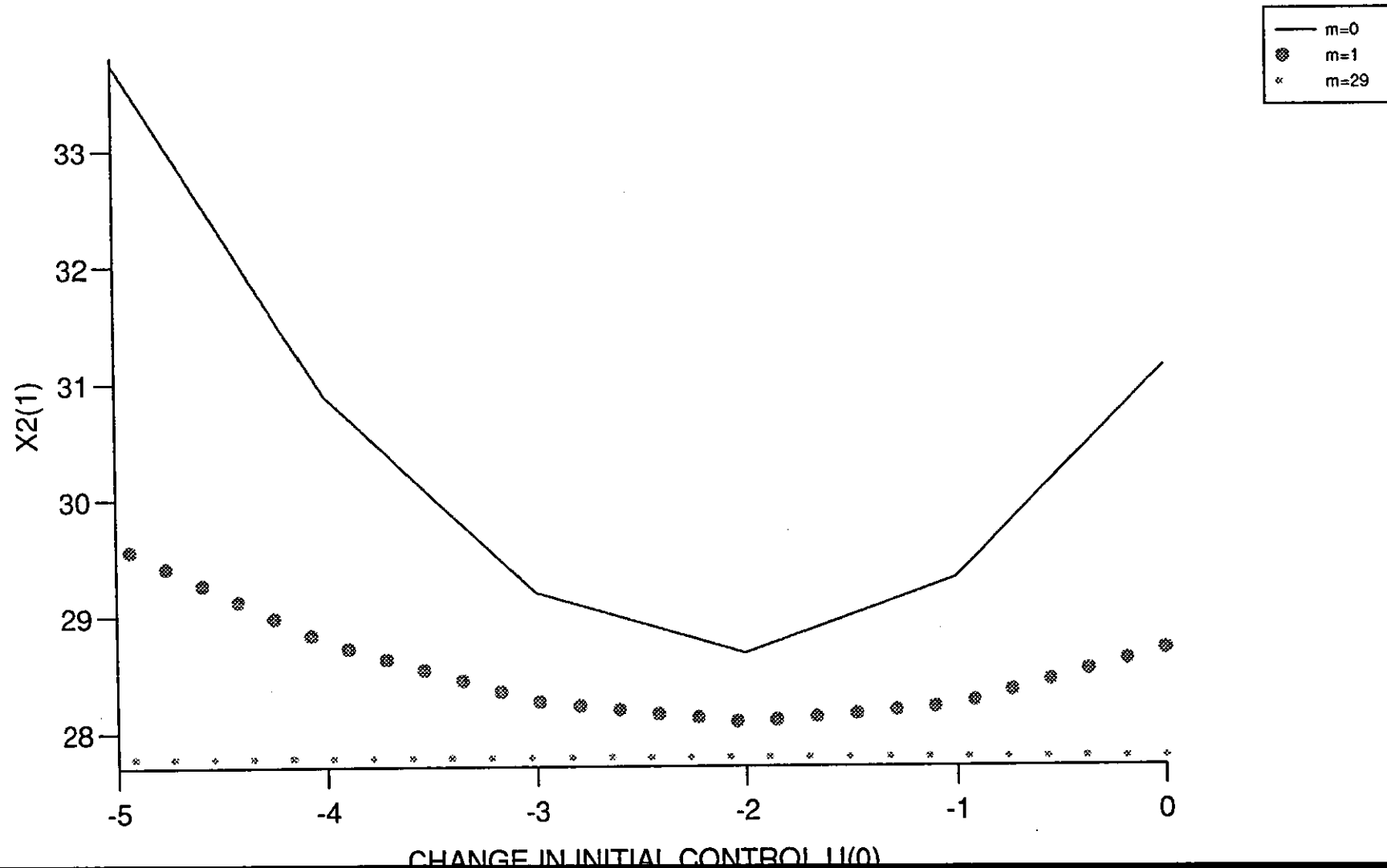


Figure (4.4.8)

SD

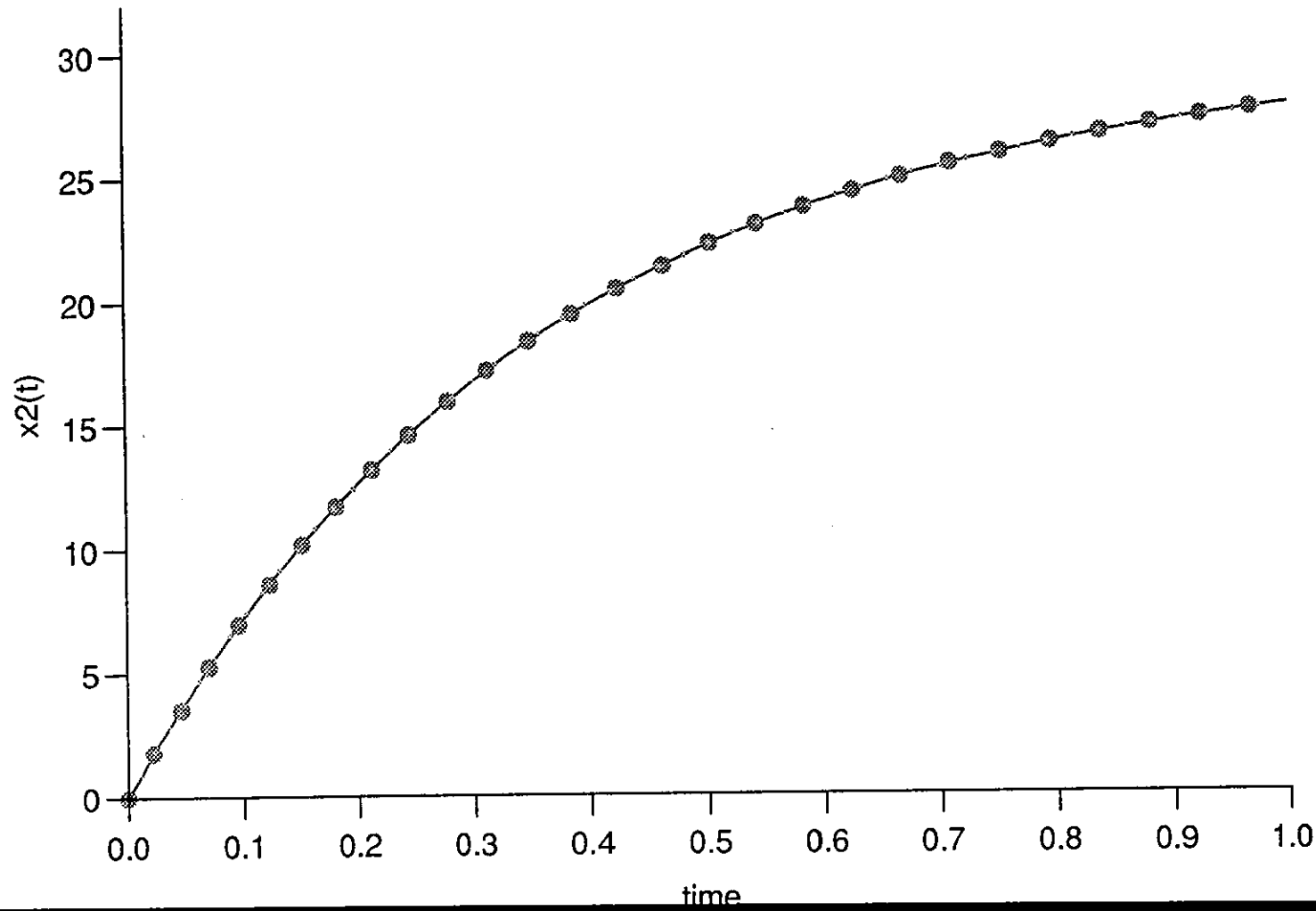
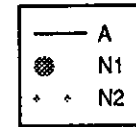


Figure (4.4.9)

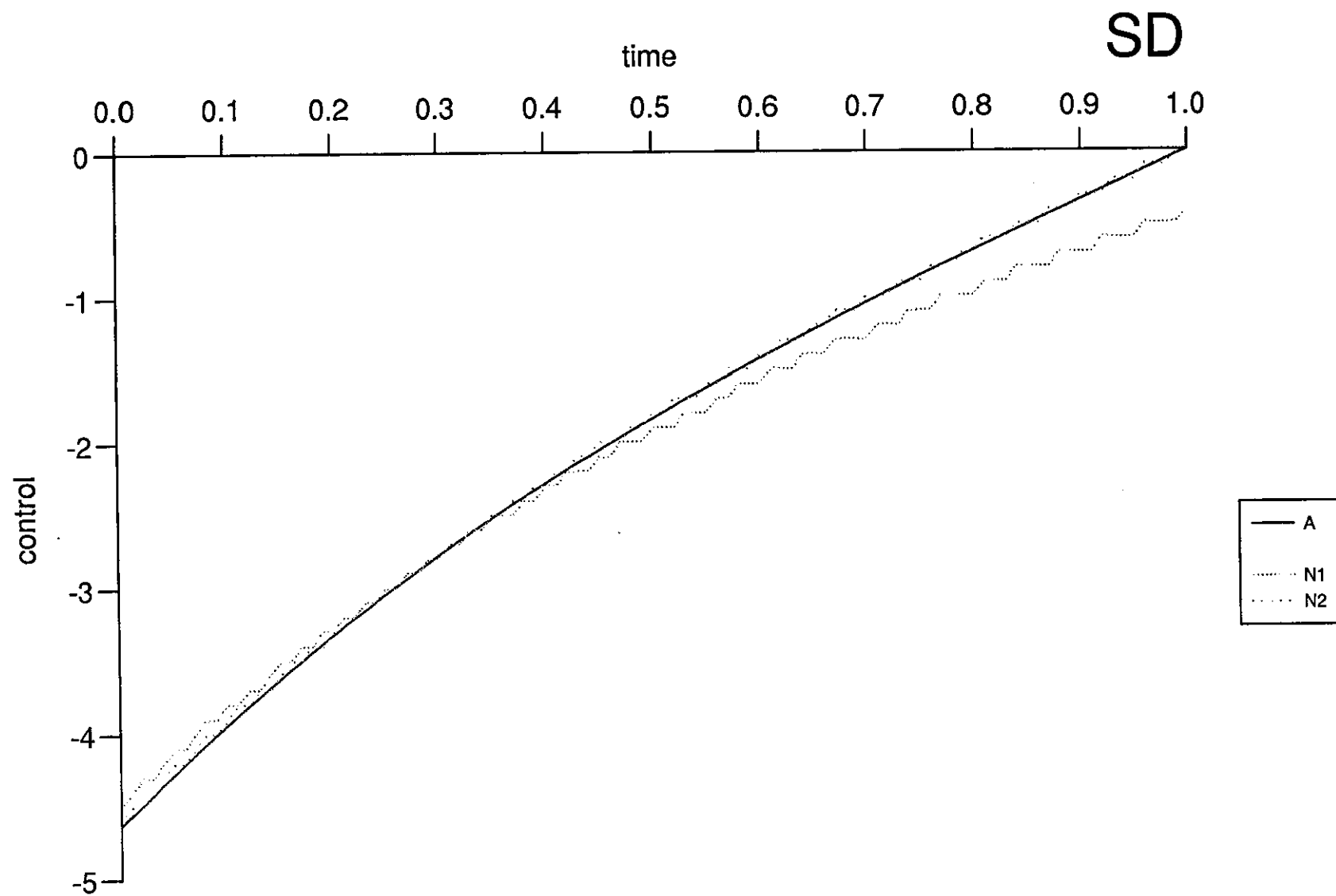


Figure (4.4.10)

SD

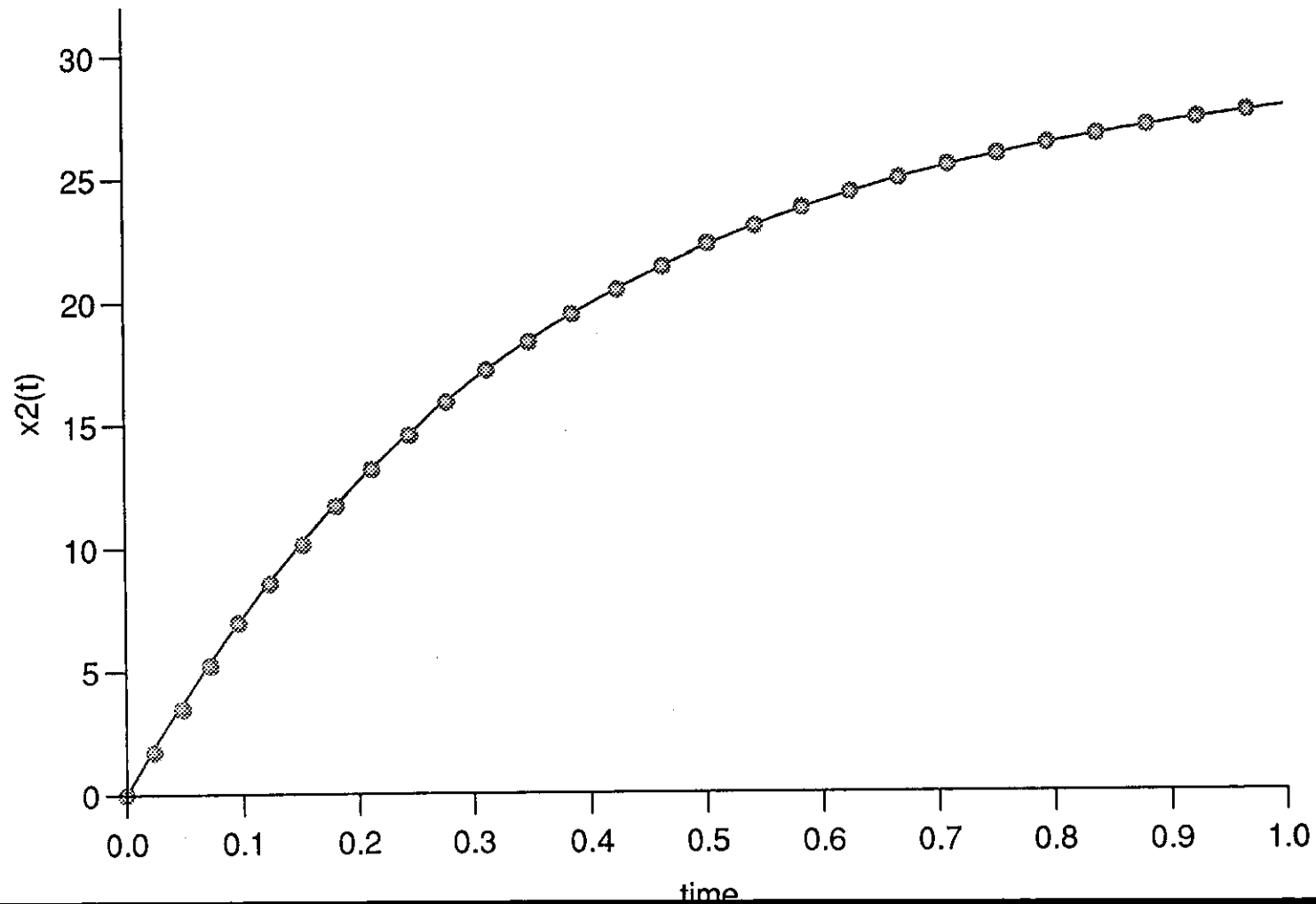
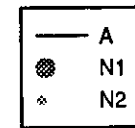


Figure (4.4.11)

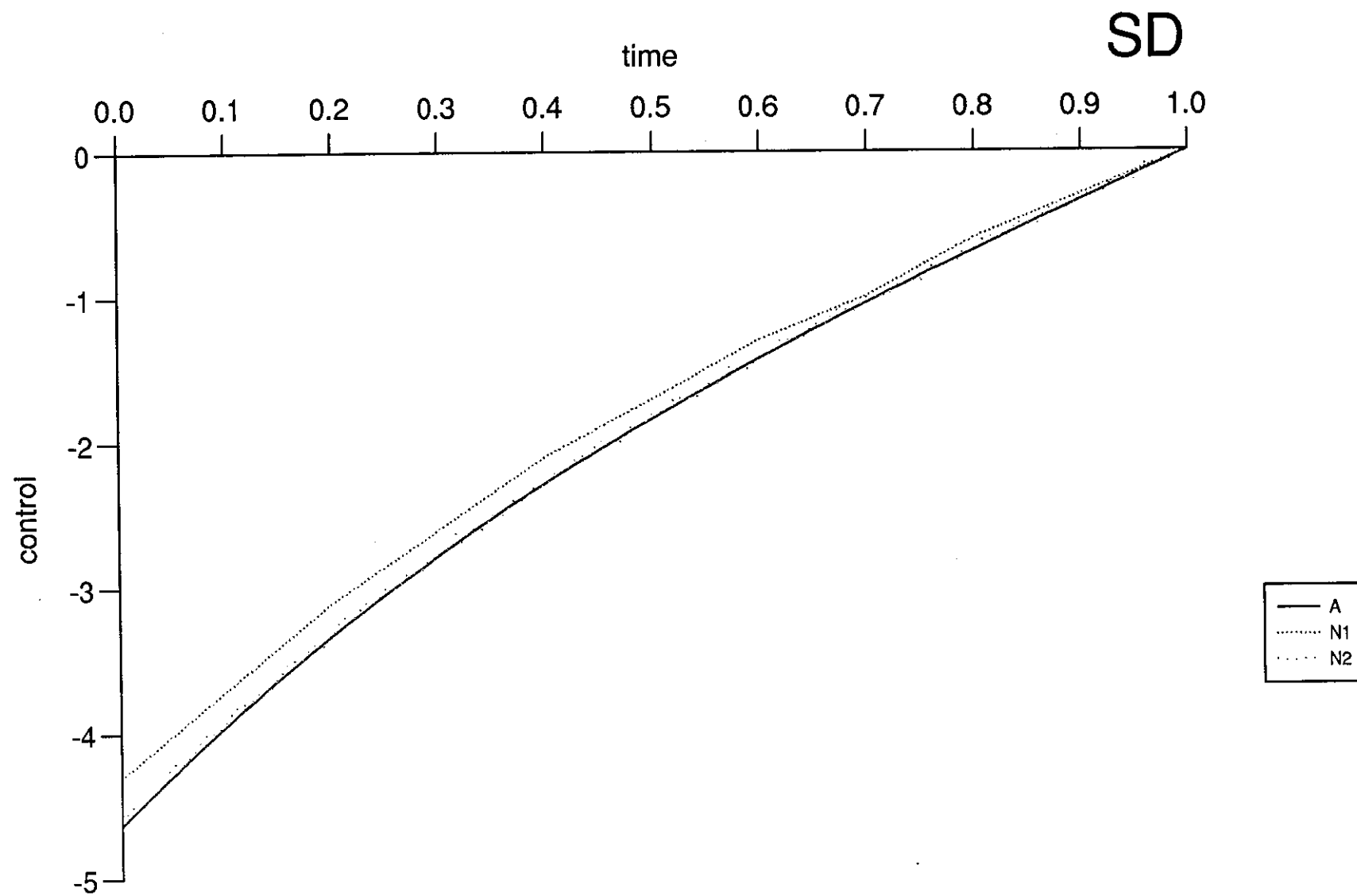


Figure (4.4.12)

SD

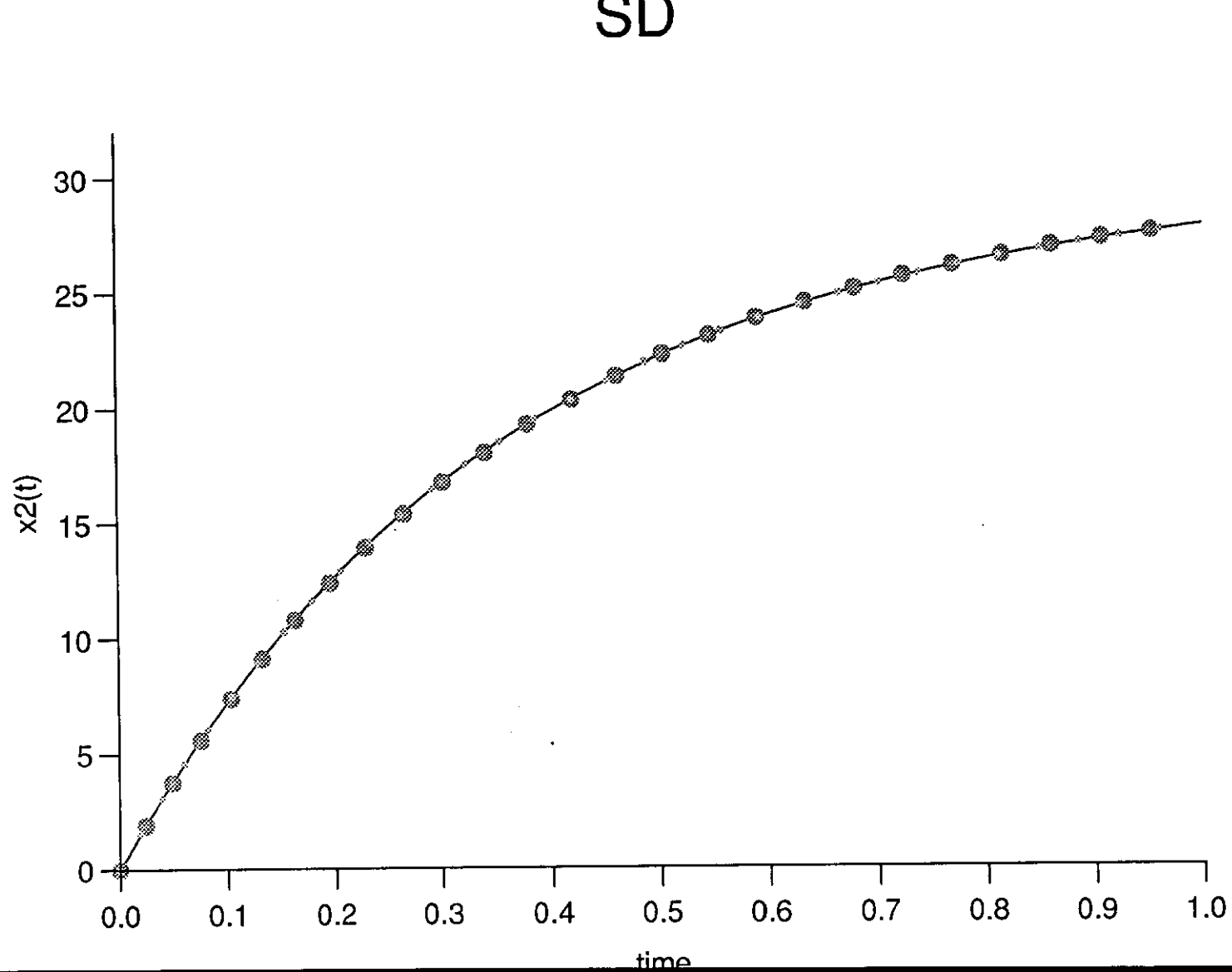


Figure (4.4.13)

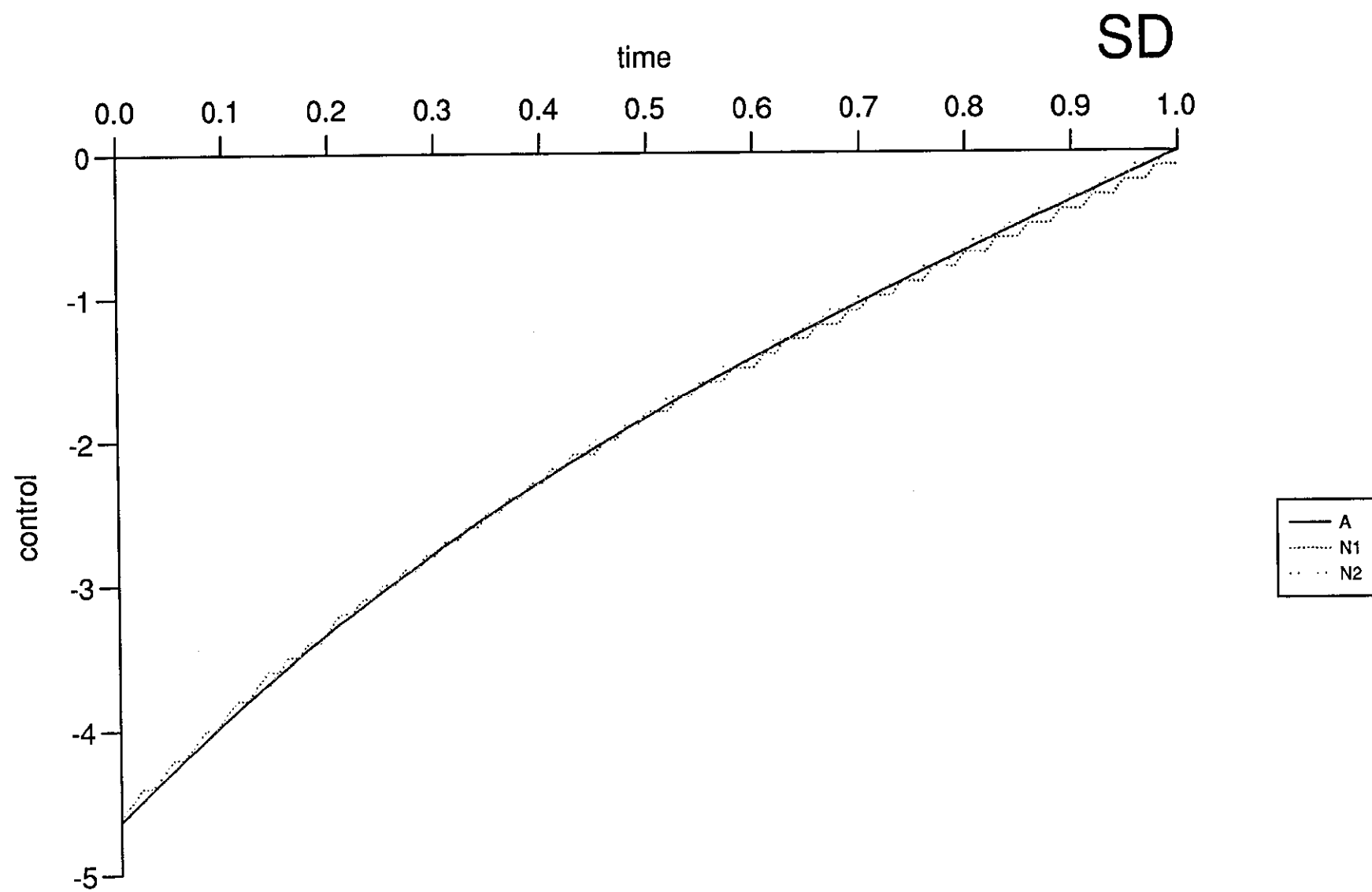


Figure (4.4.14)

SD

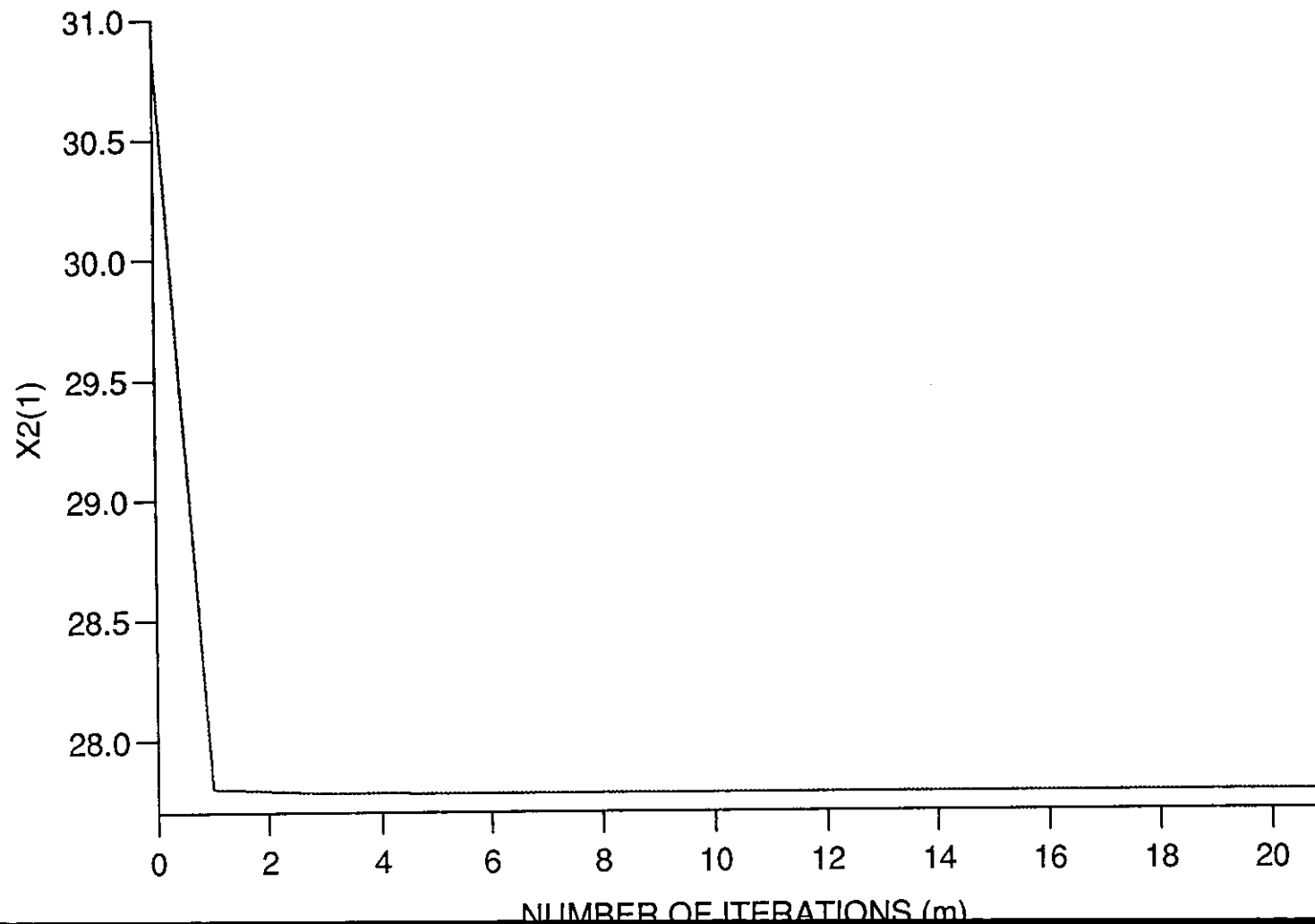


Figure (4.4.15)

SD

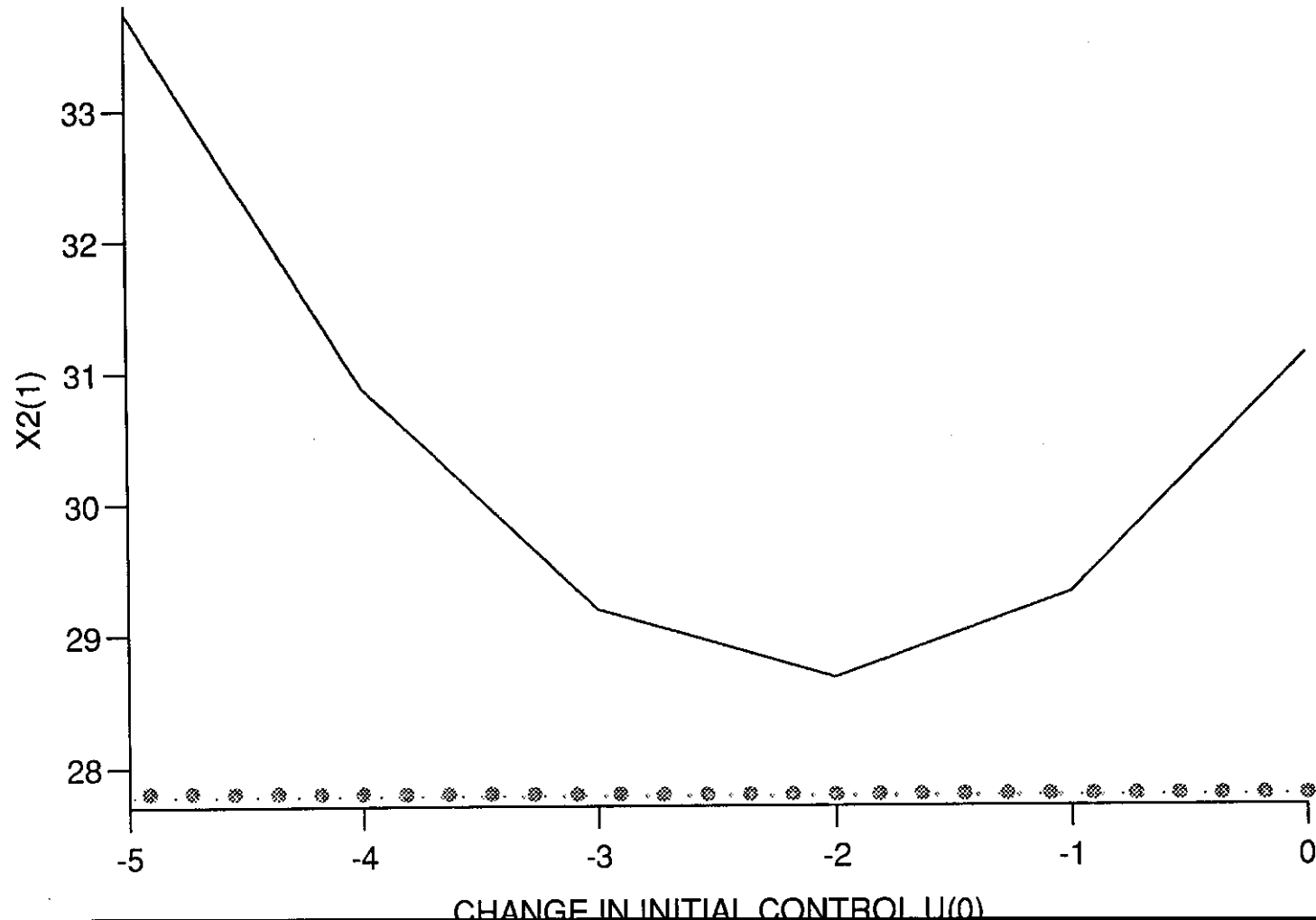
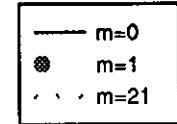


Figure (4.4.16)

FR

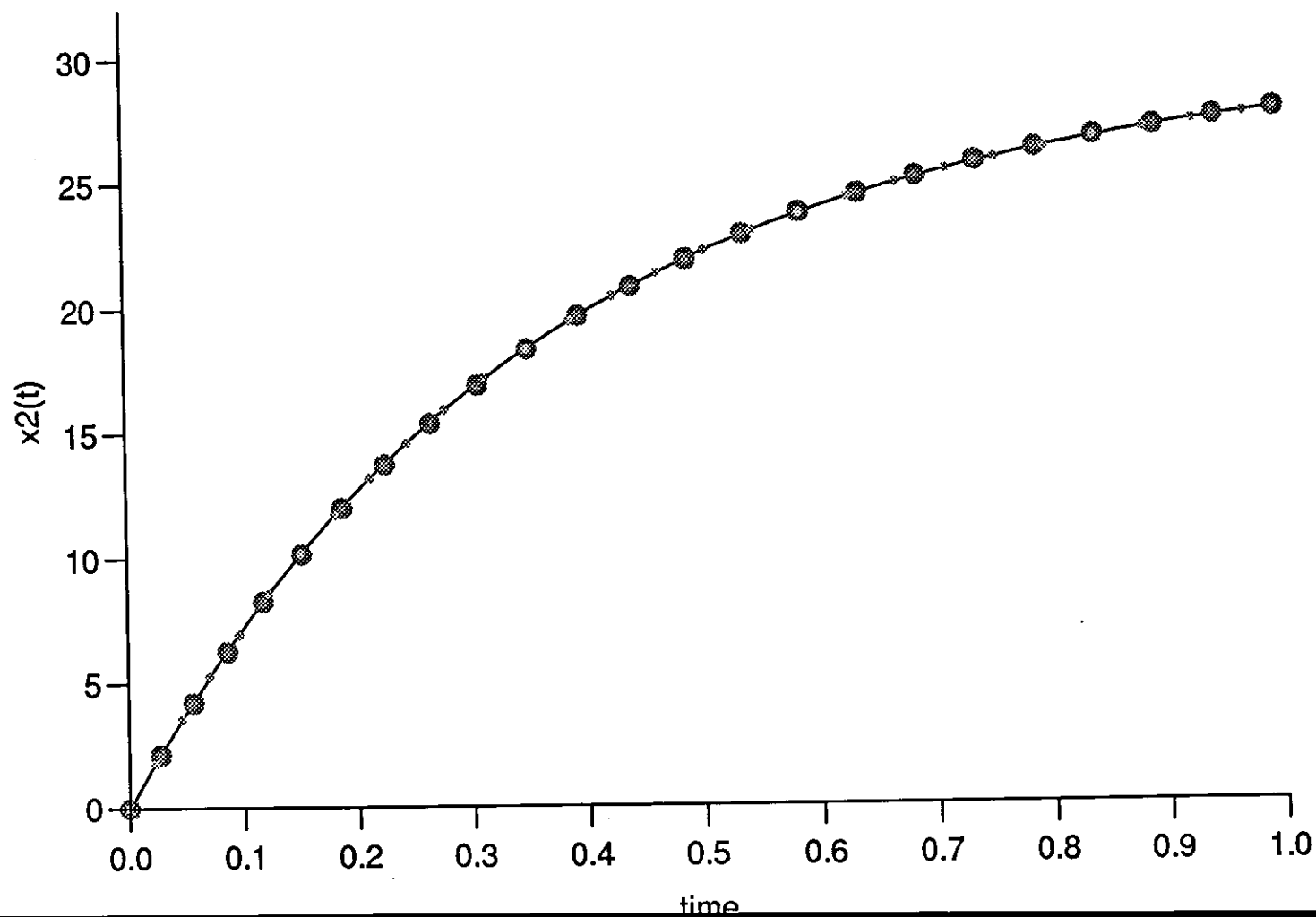
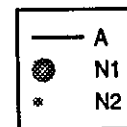


Figure (4.4.17)

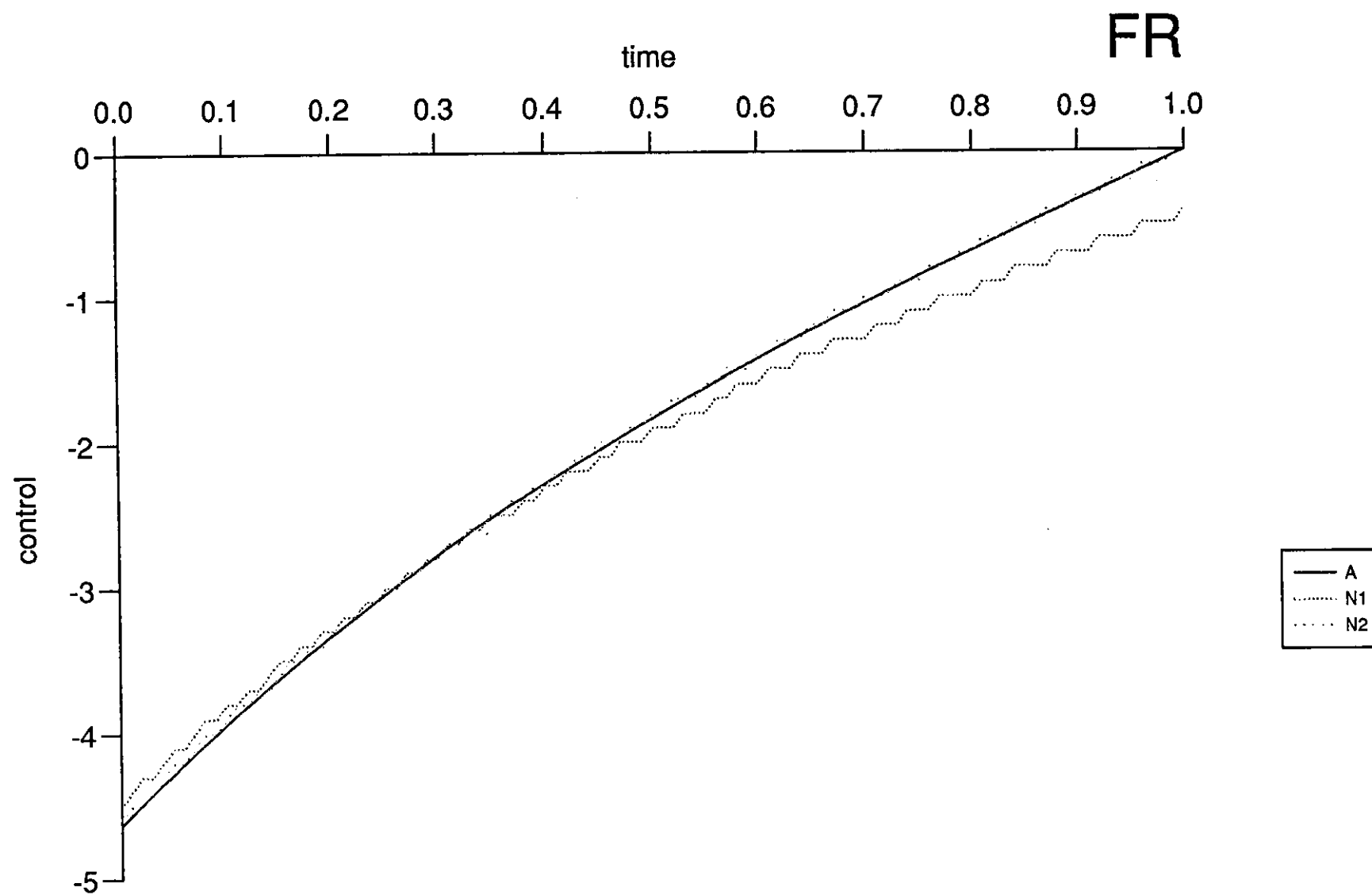


Figure (4.4.18)

FR

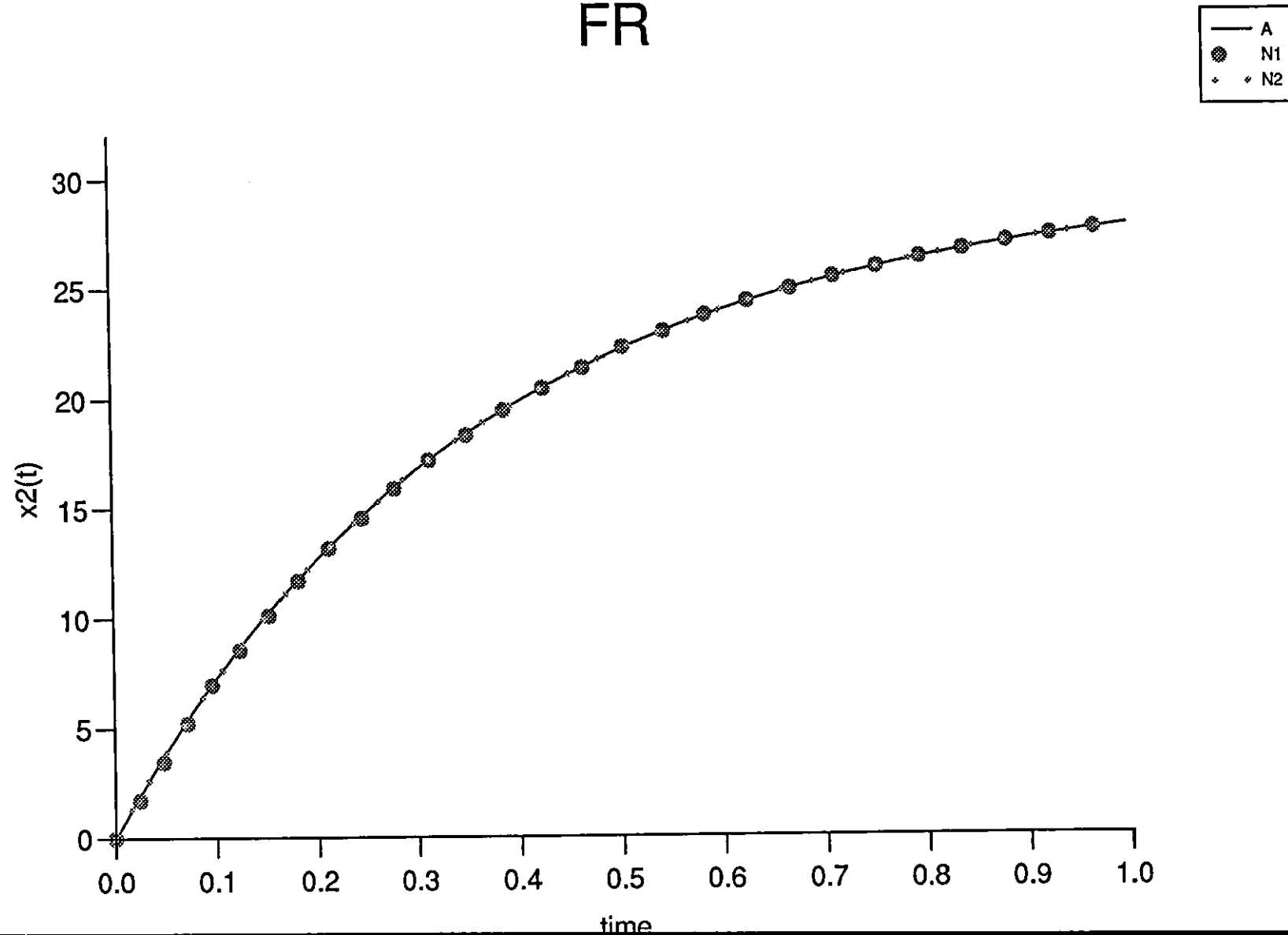


Figure (4.4.19)

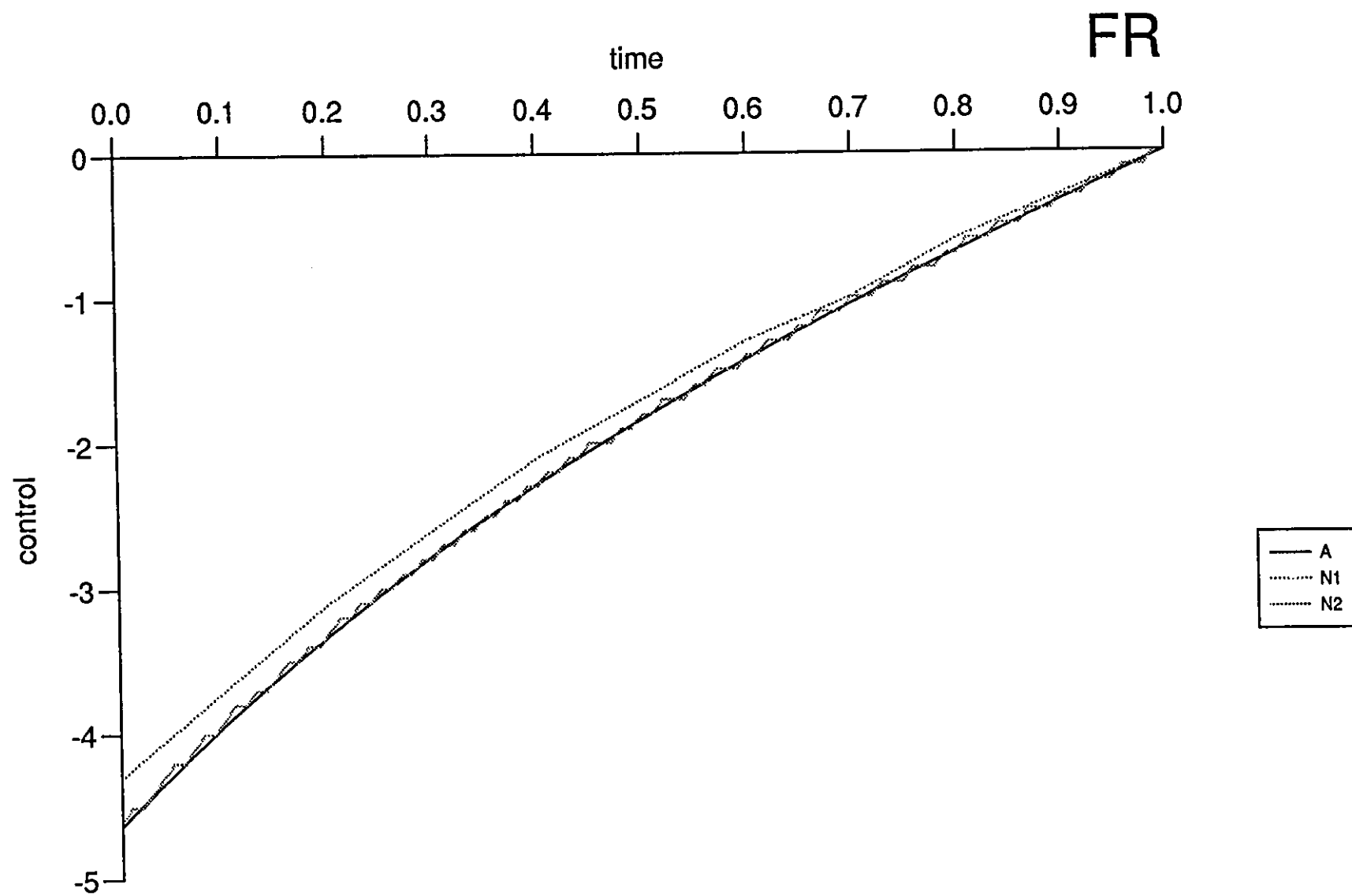


Figure (4.4.20)

FR

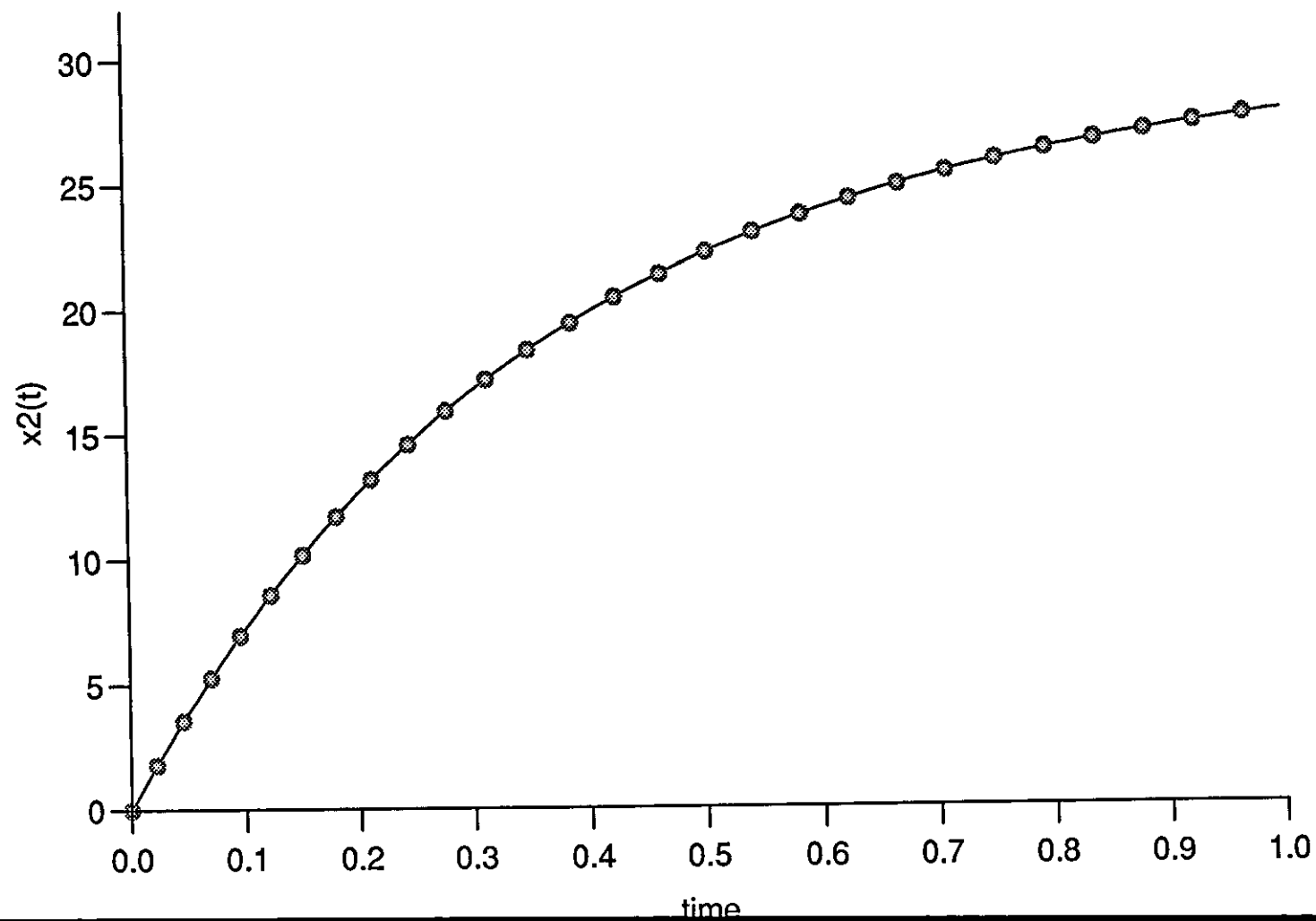
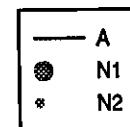


Figure (4.4.21)

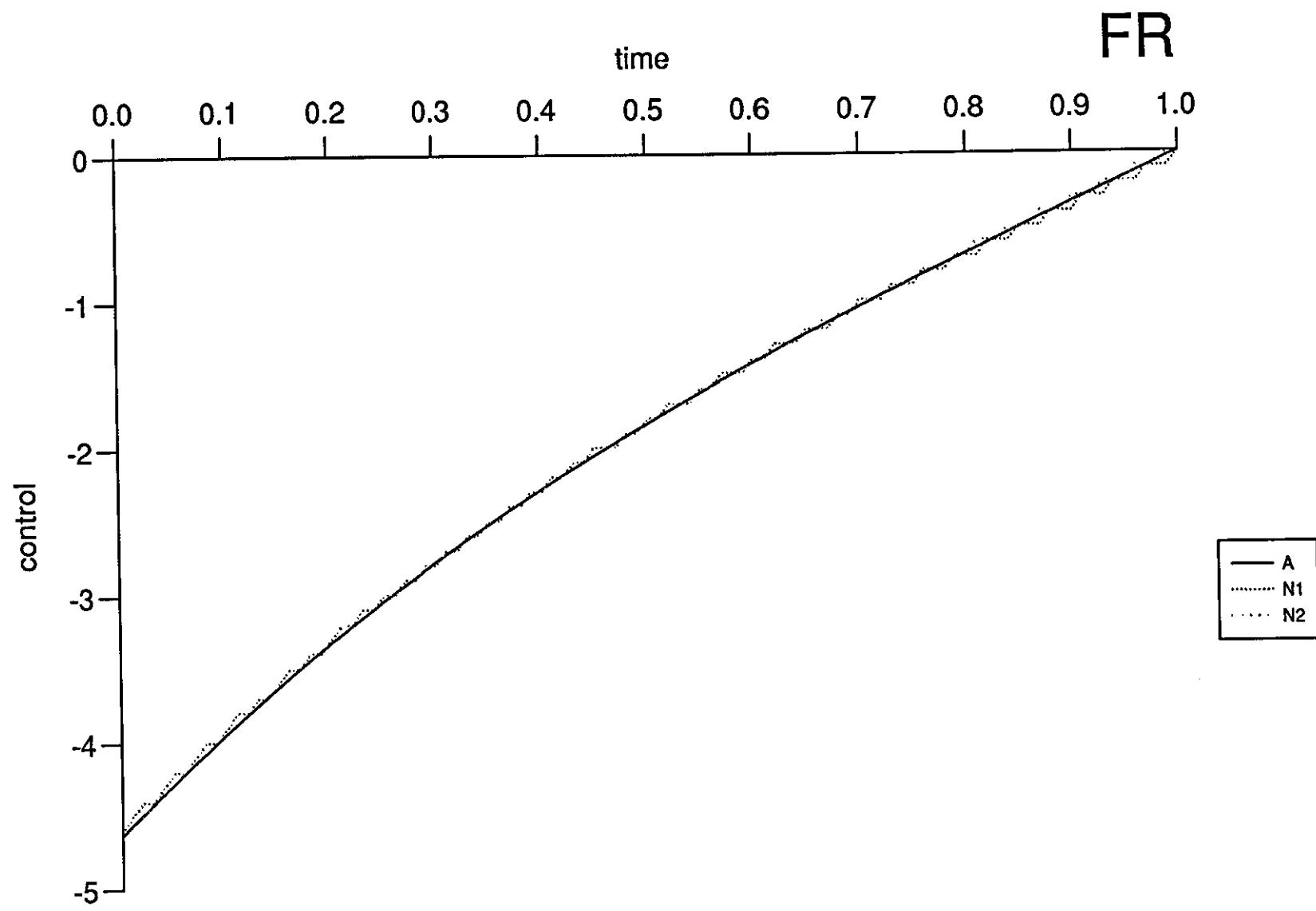


Figure (4.4.22)

FR

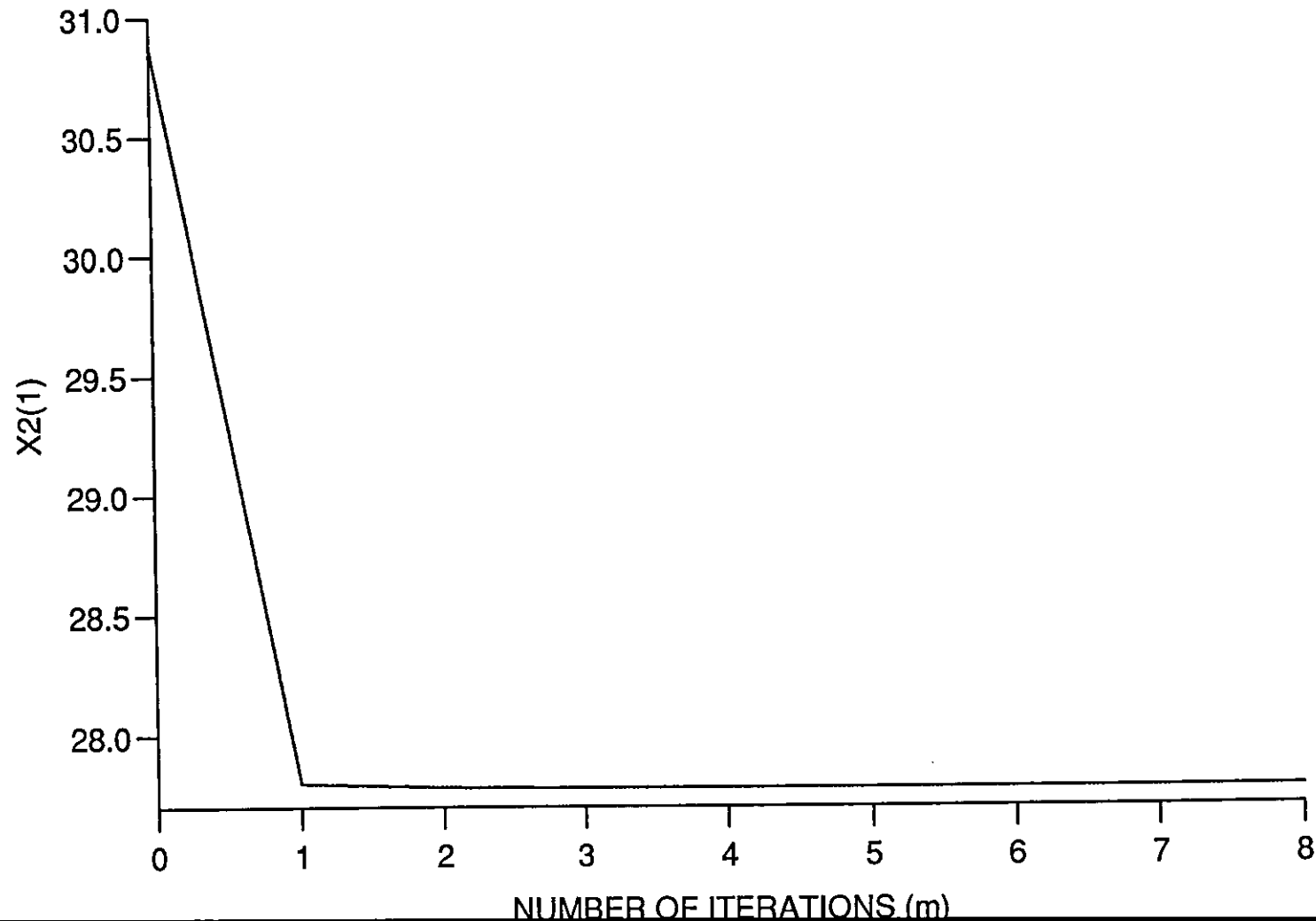


Figure (4.4.23)

FR

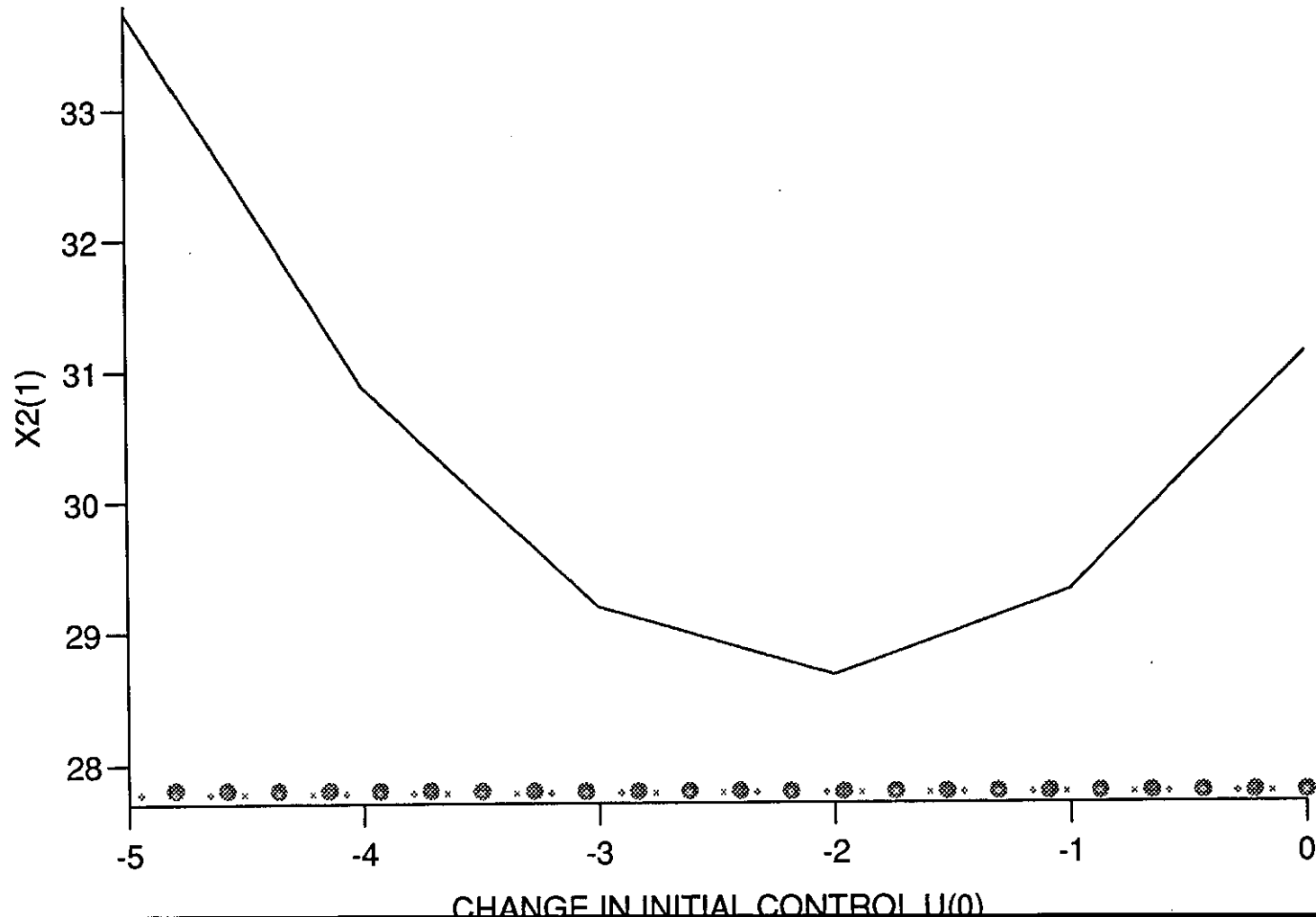
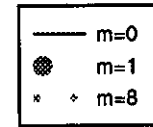


Figure (4.4.24)

PR

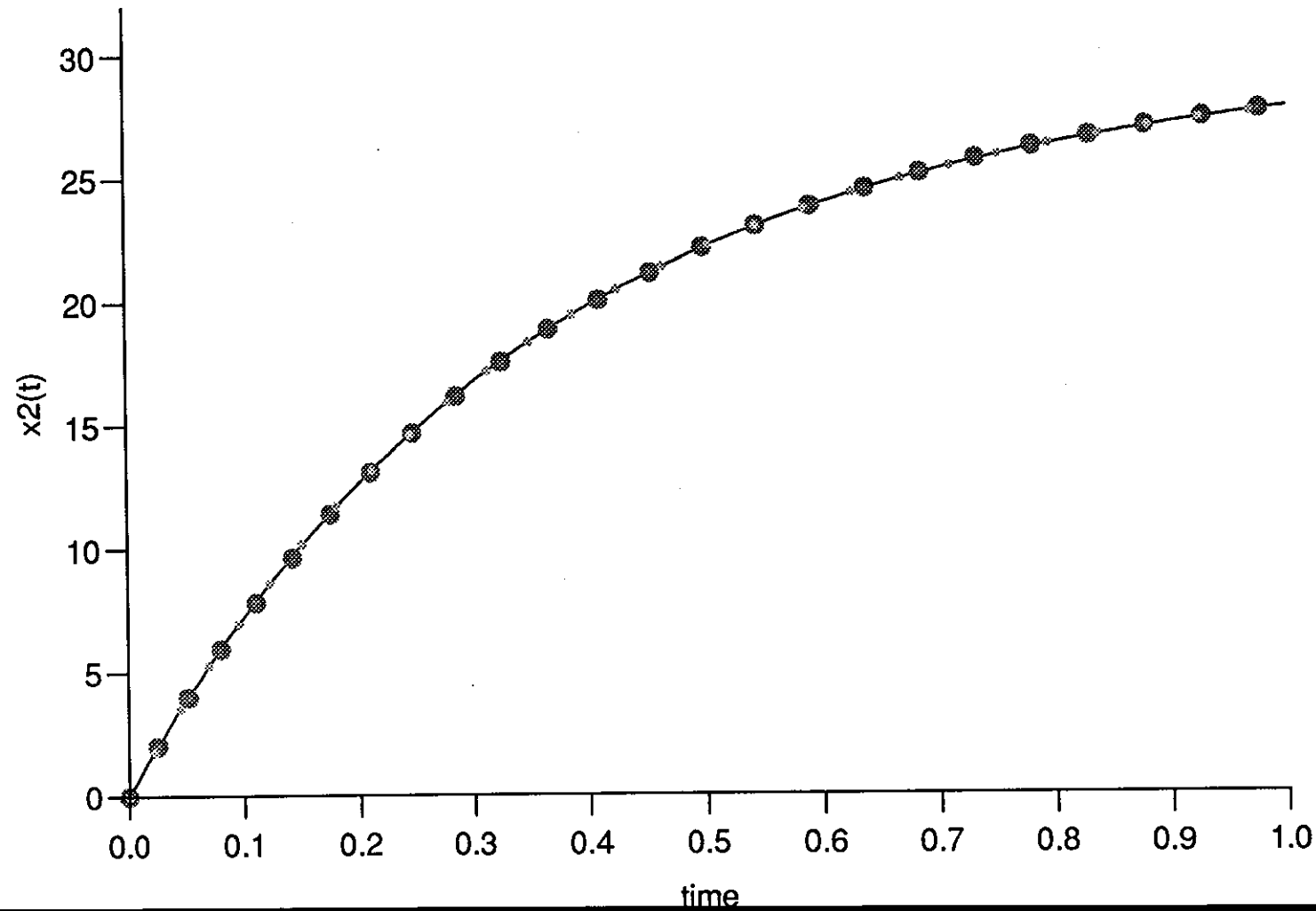
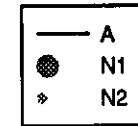


Figure (4.4.25)

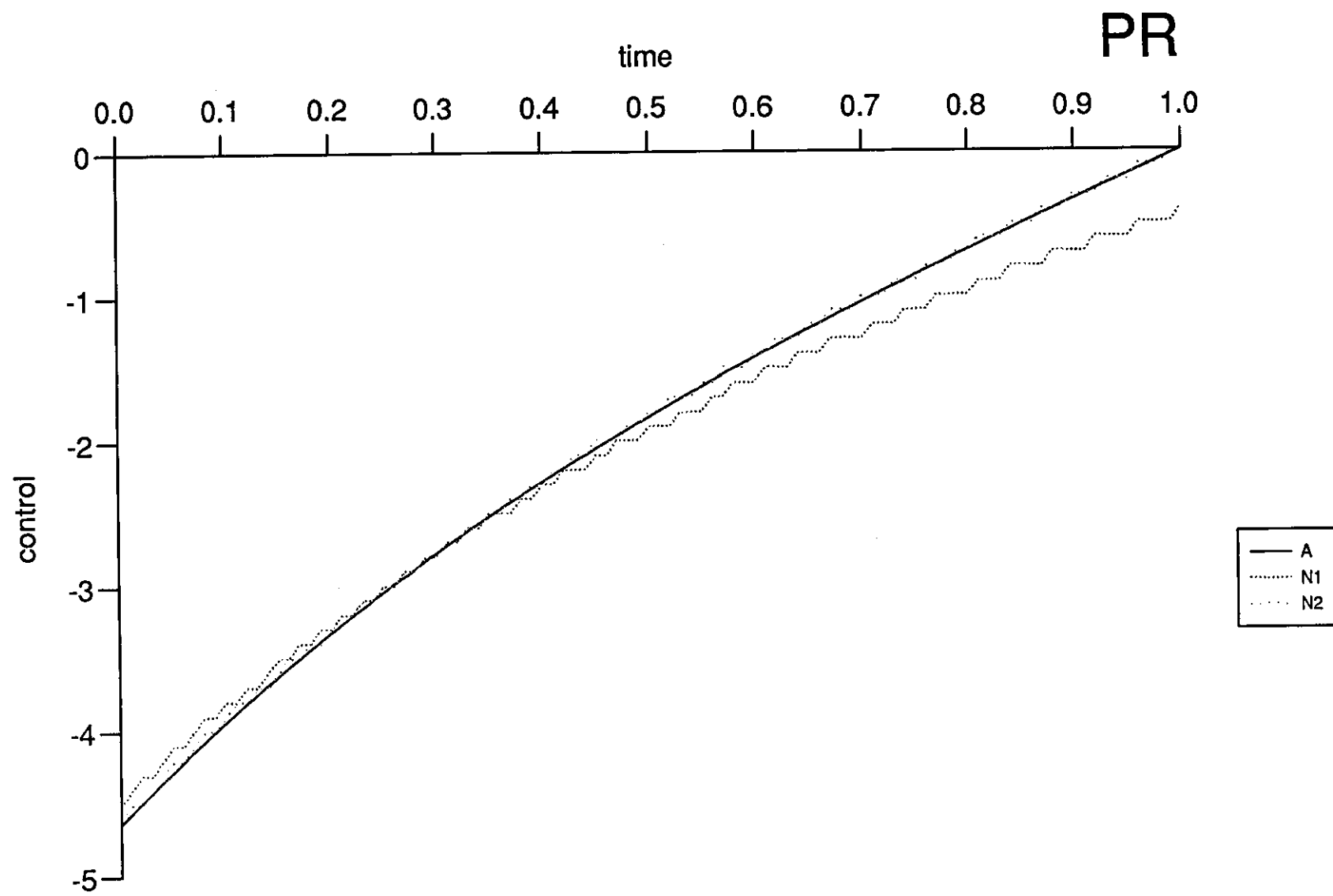


Figure (4.4.26)

PR

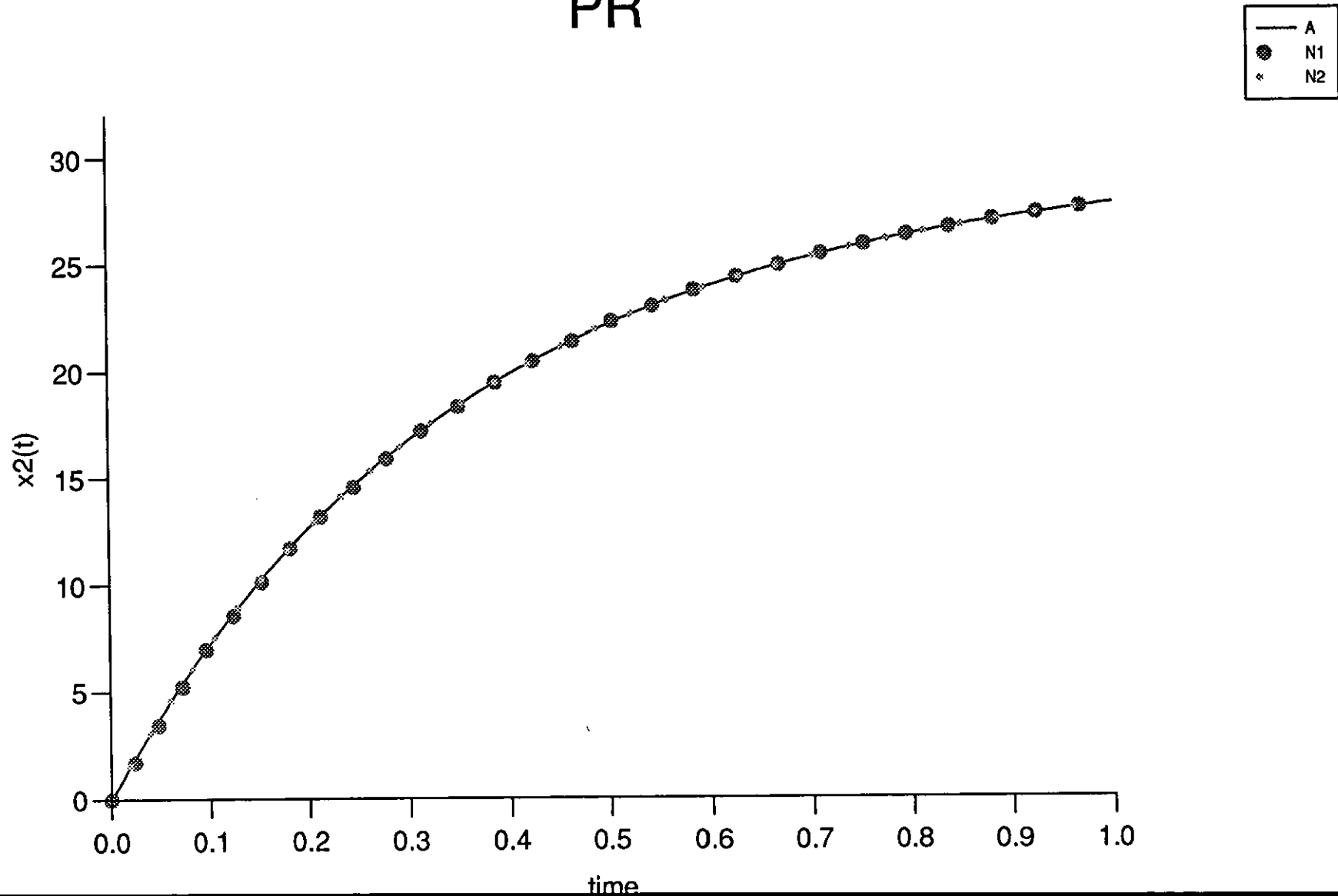


Figure (4.4.27)

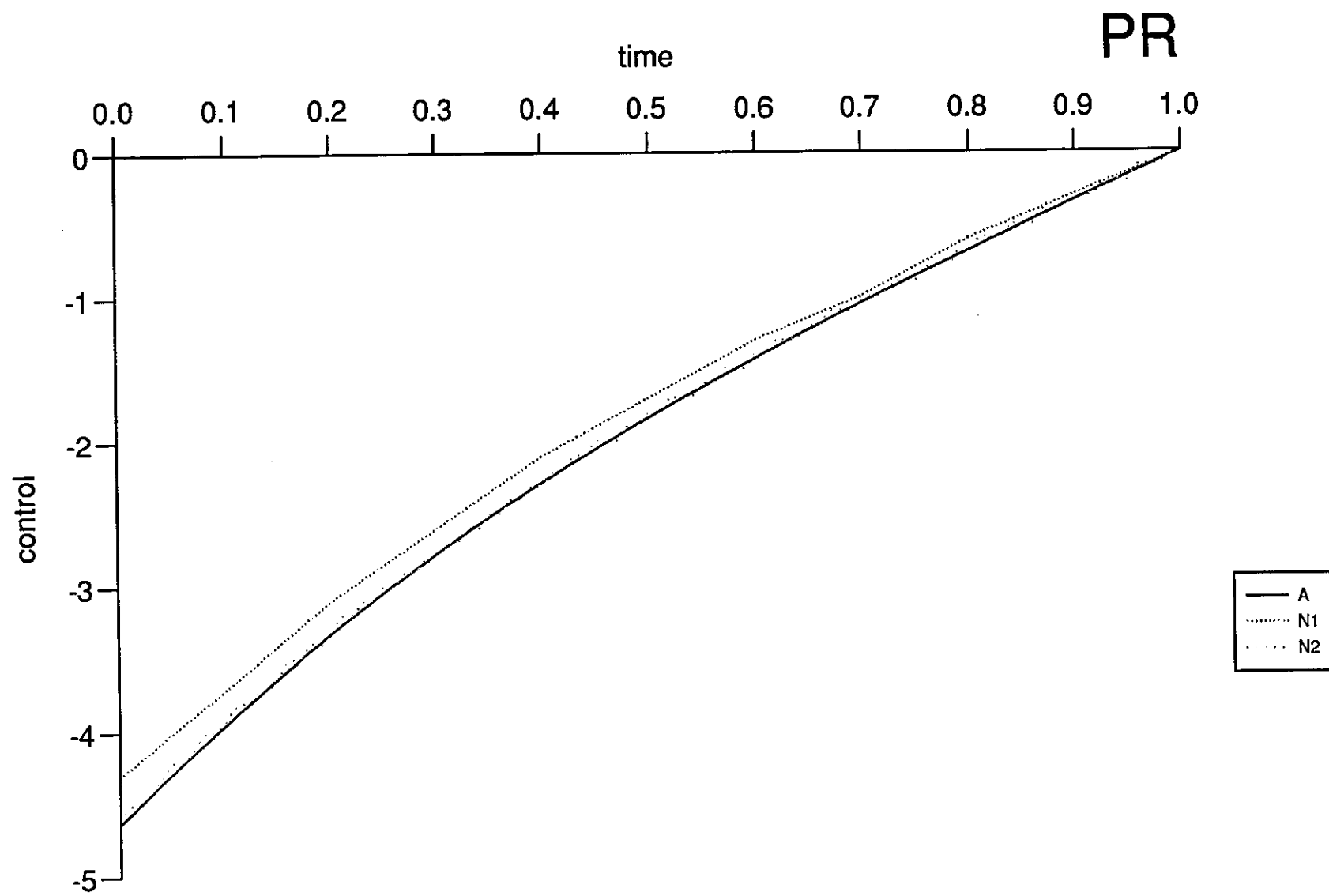


Figure (4.4.28)

PR

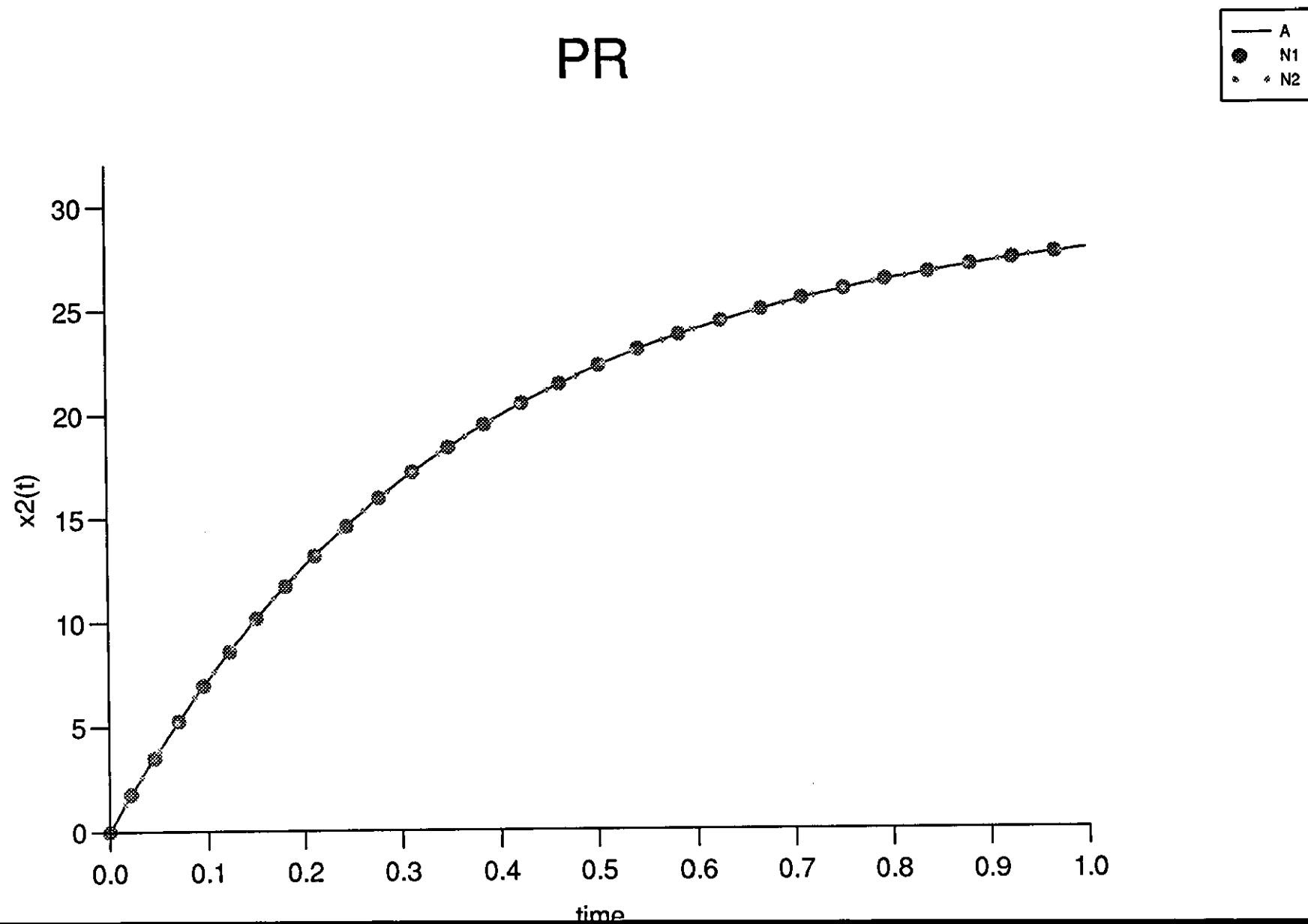


Figure (4.4.29)

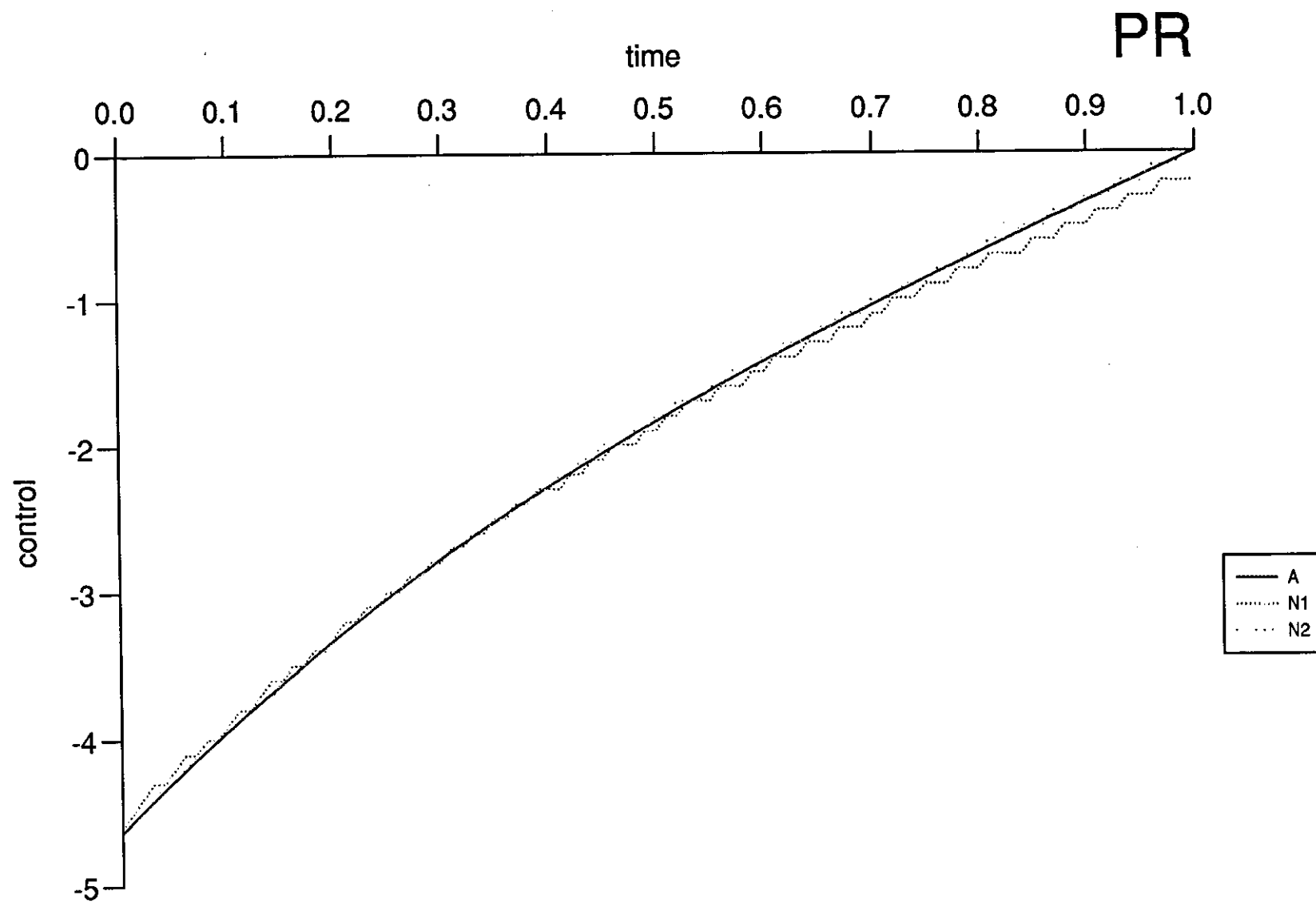


Figure (4.4.30)

PR

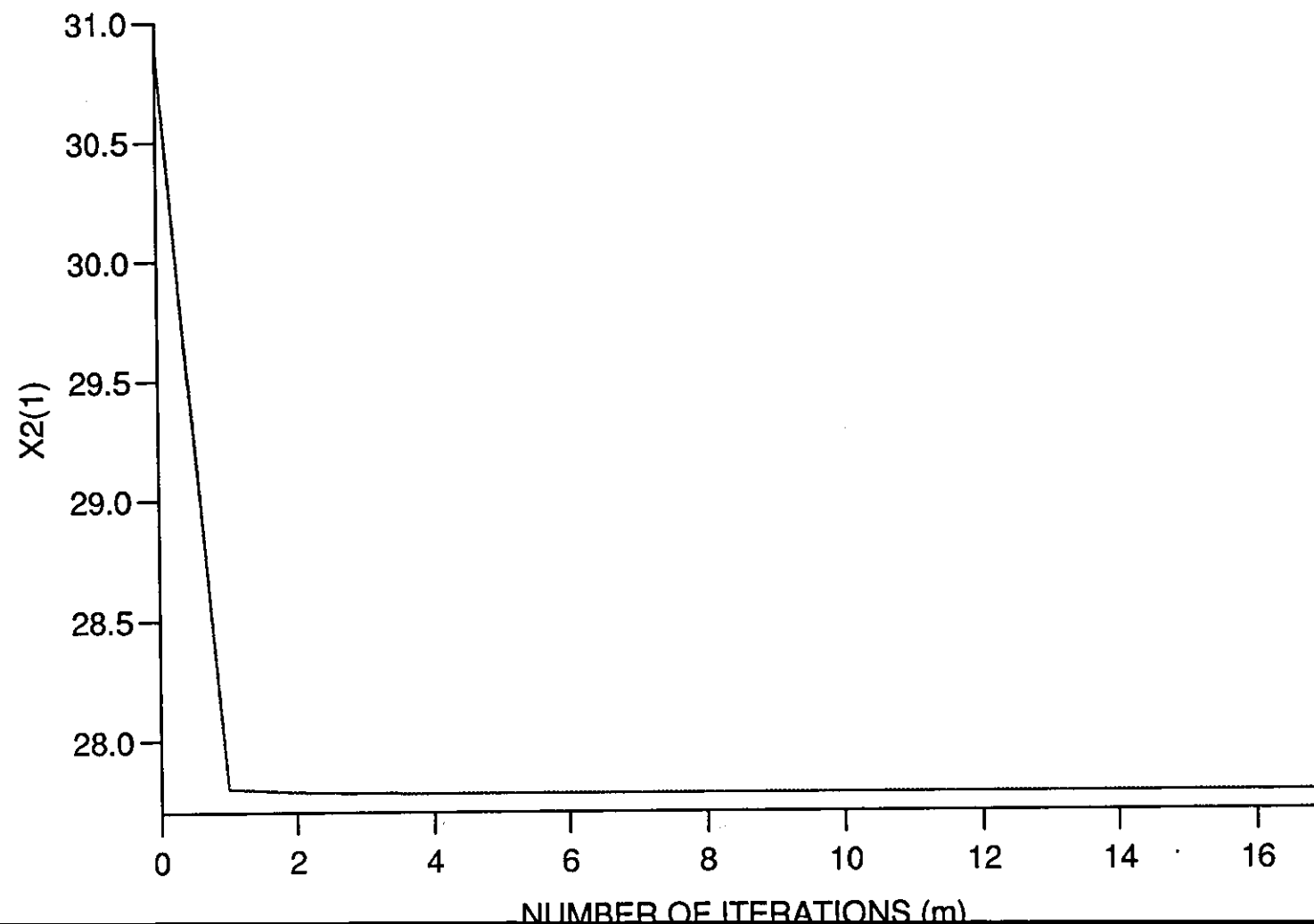


Figure (4.4.31)

PR

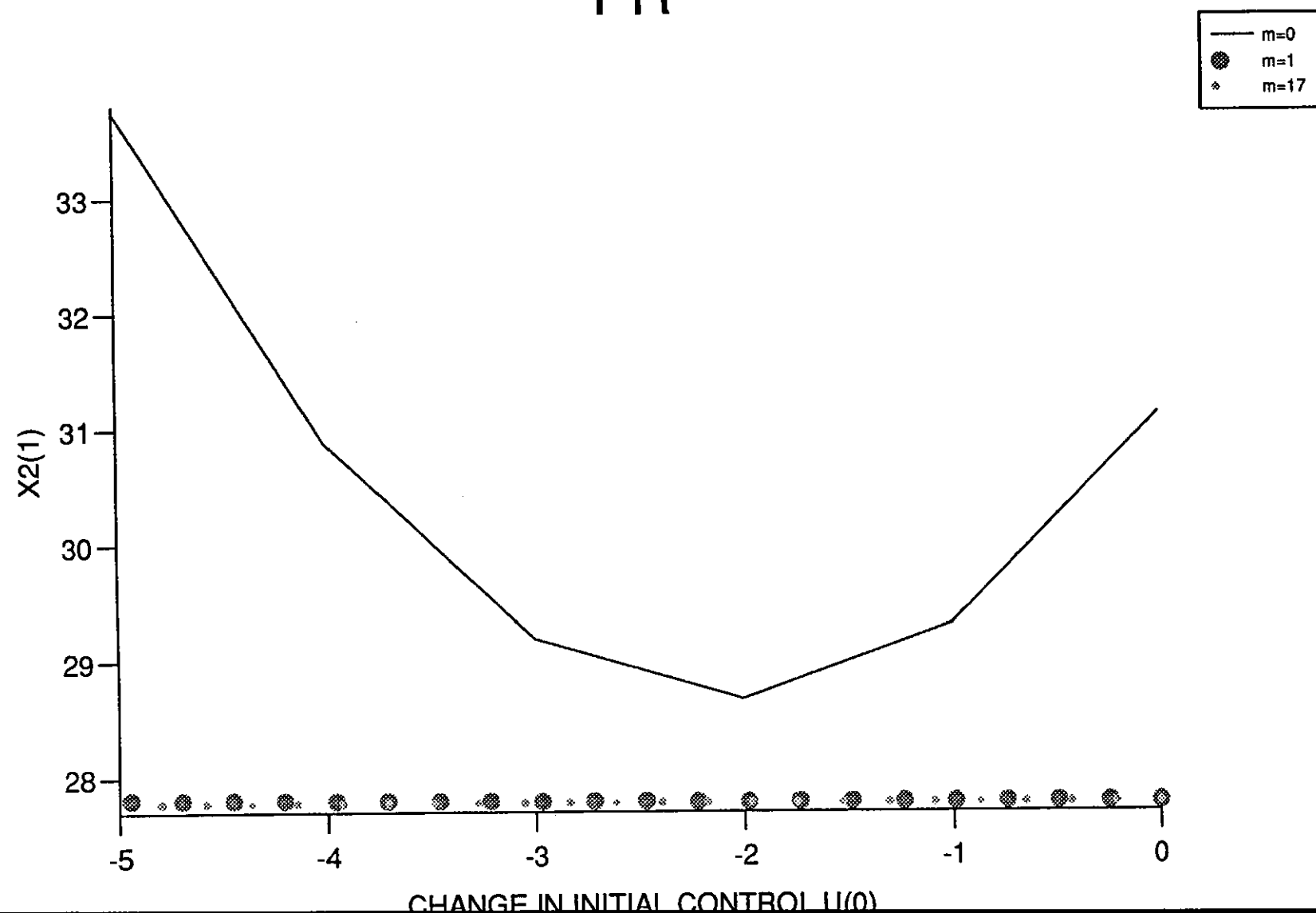


Figure (4.4.32)

H1

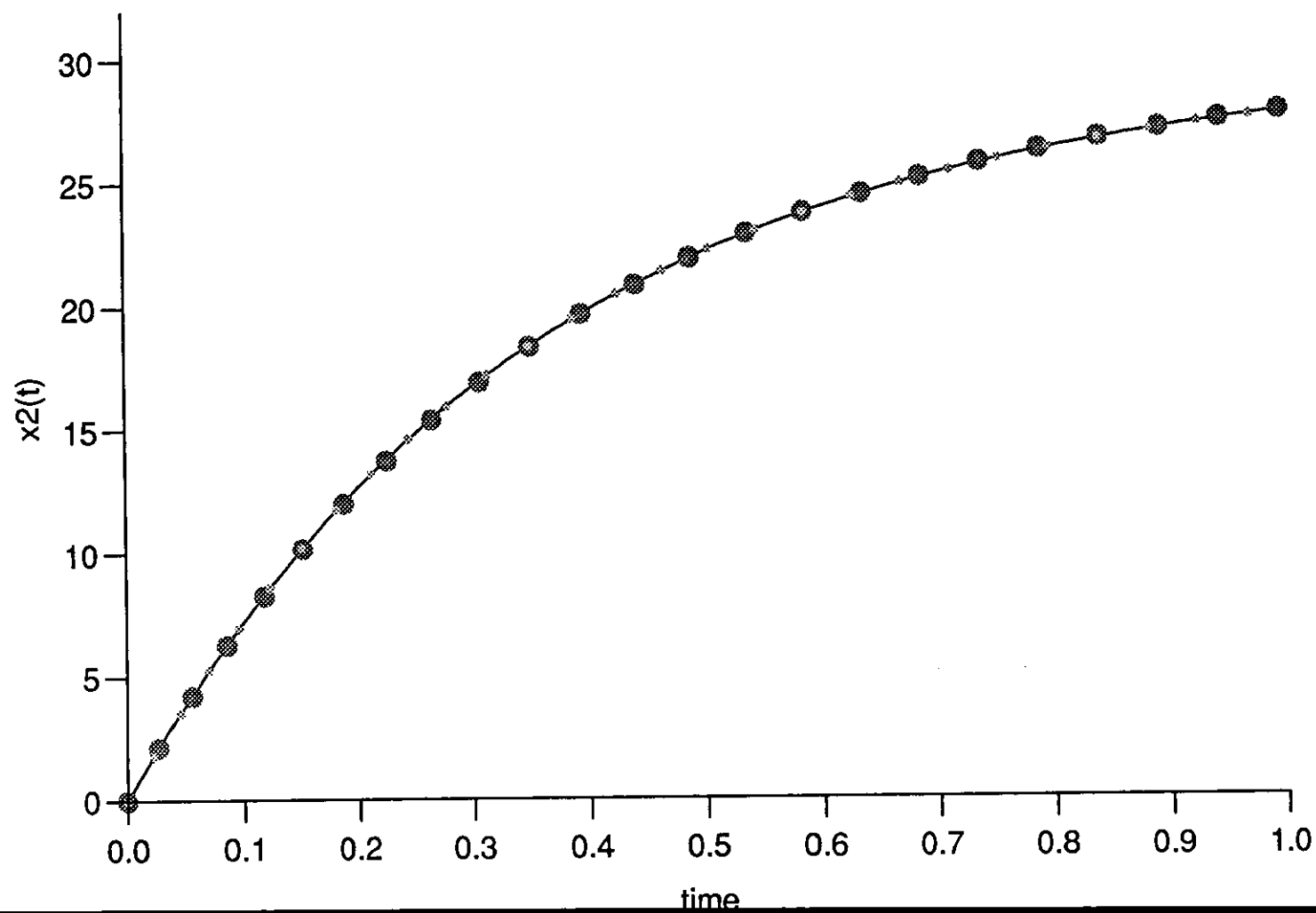
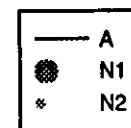


Figure (4.4.33)

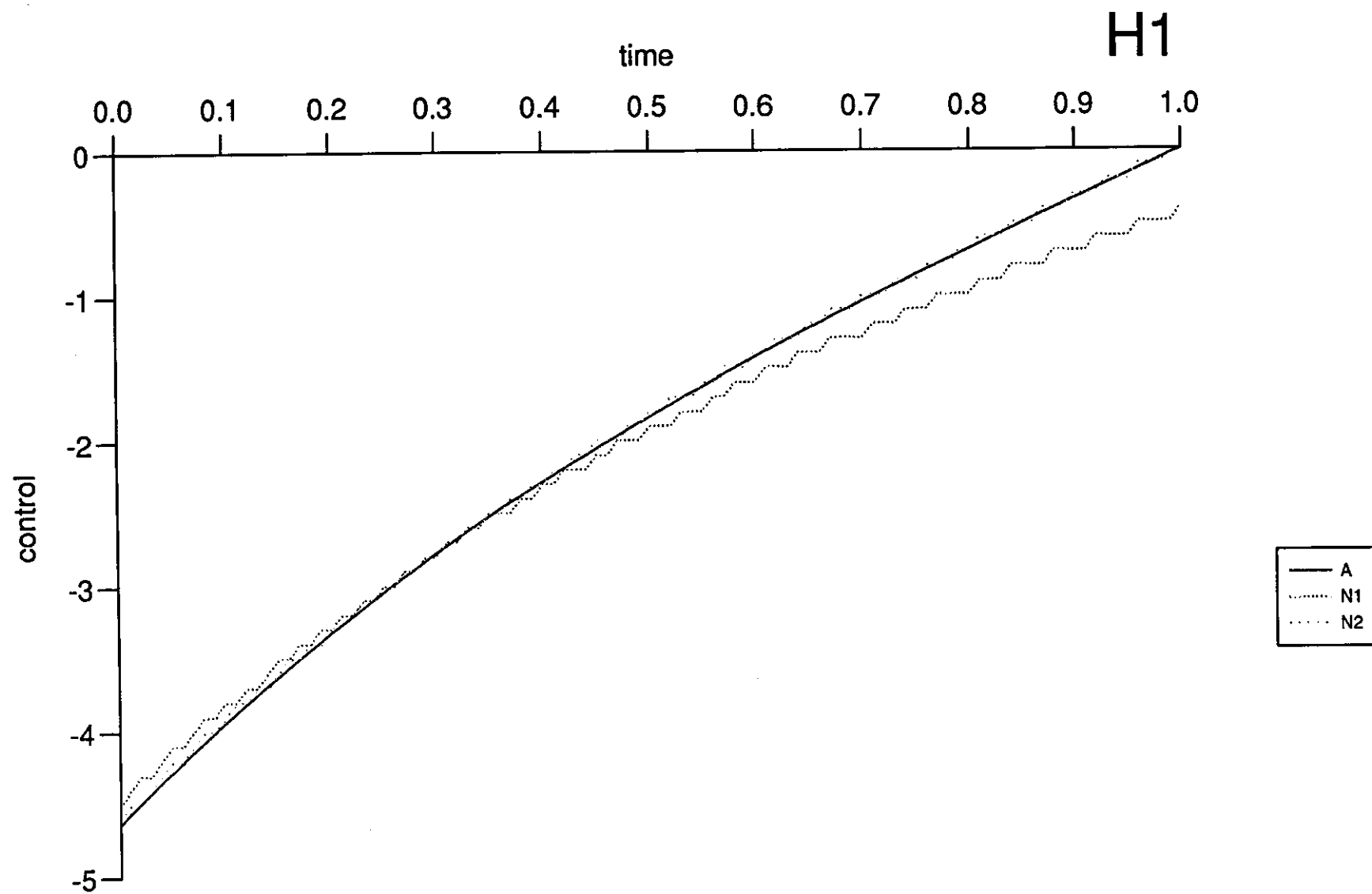


Figure (4.4.34)

H1

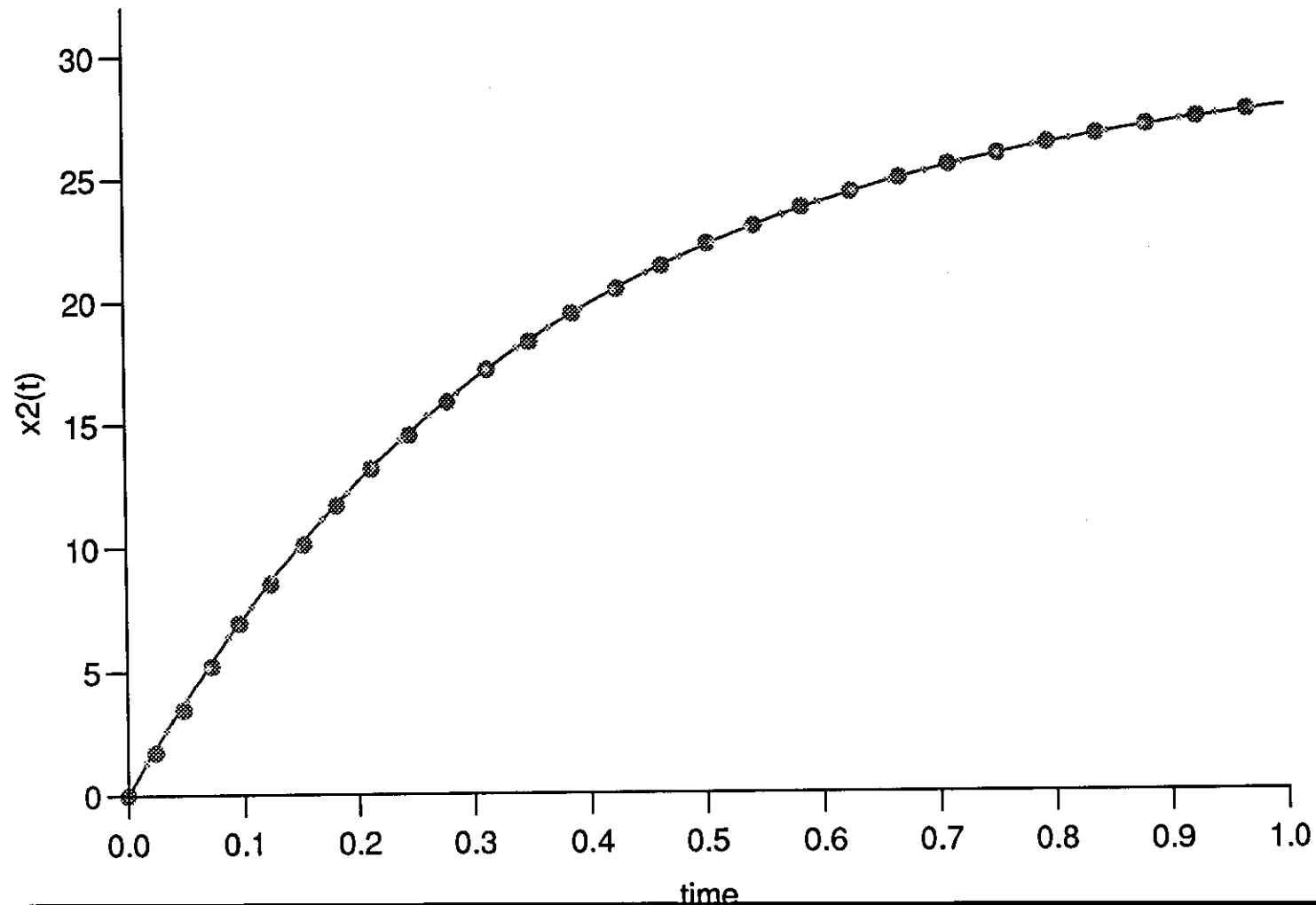
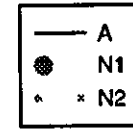


Figure (4.4.35)

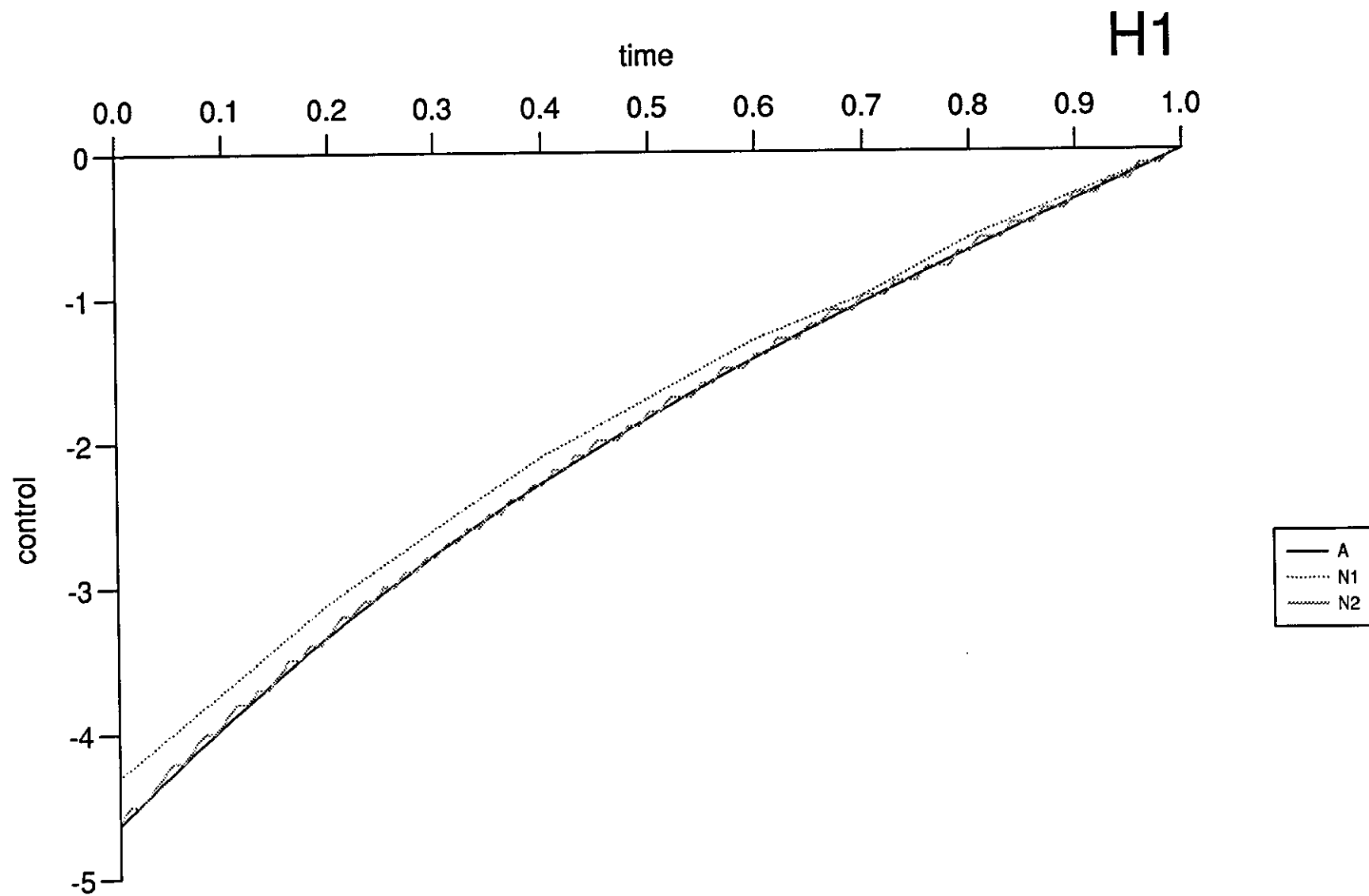


Figure (4.4.36)

H1

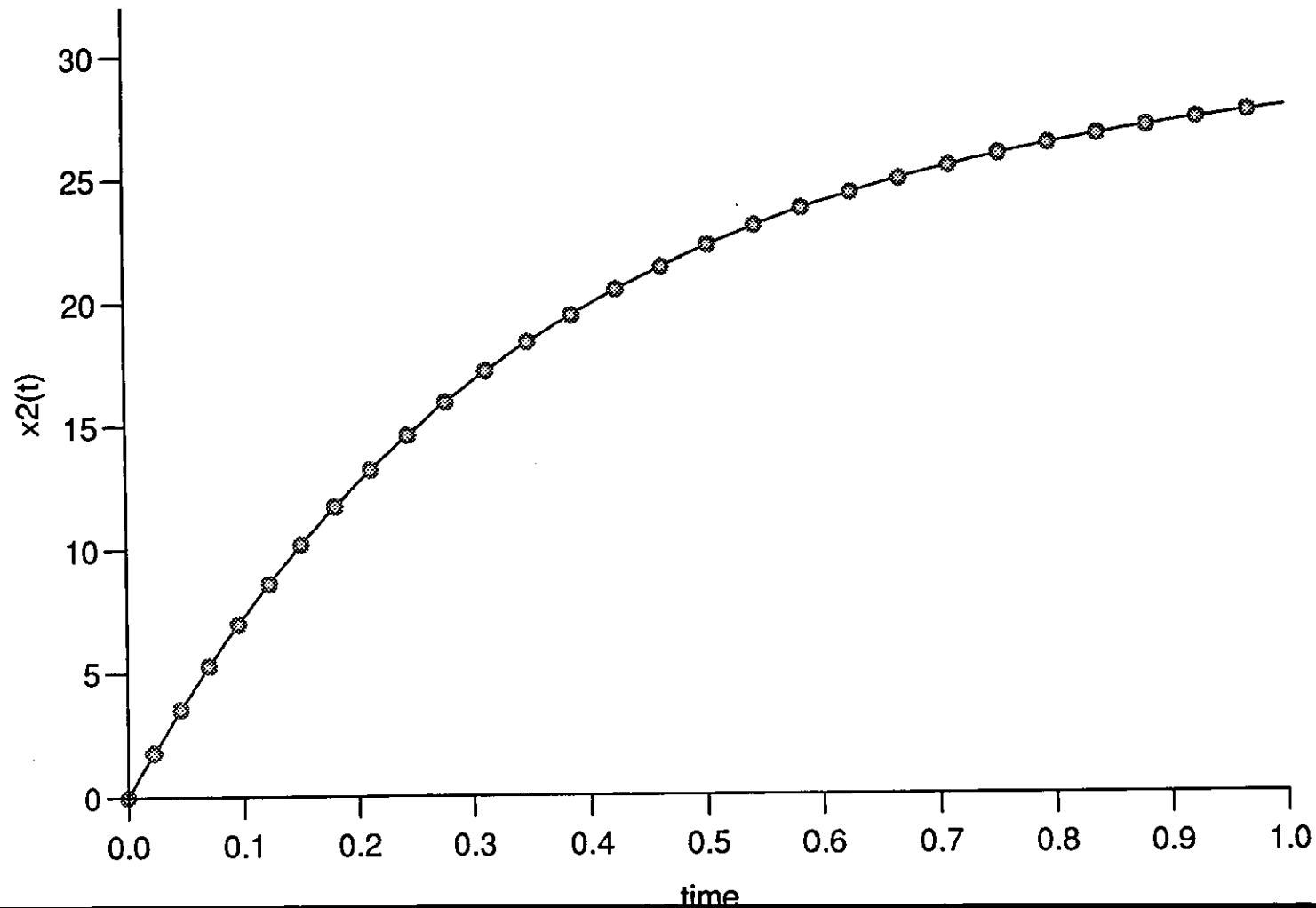
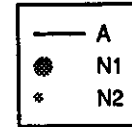


Figure (4.4.37)

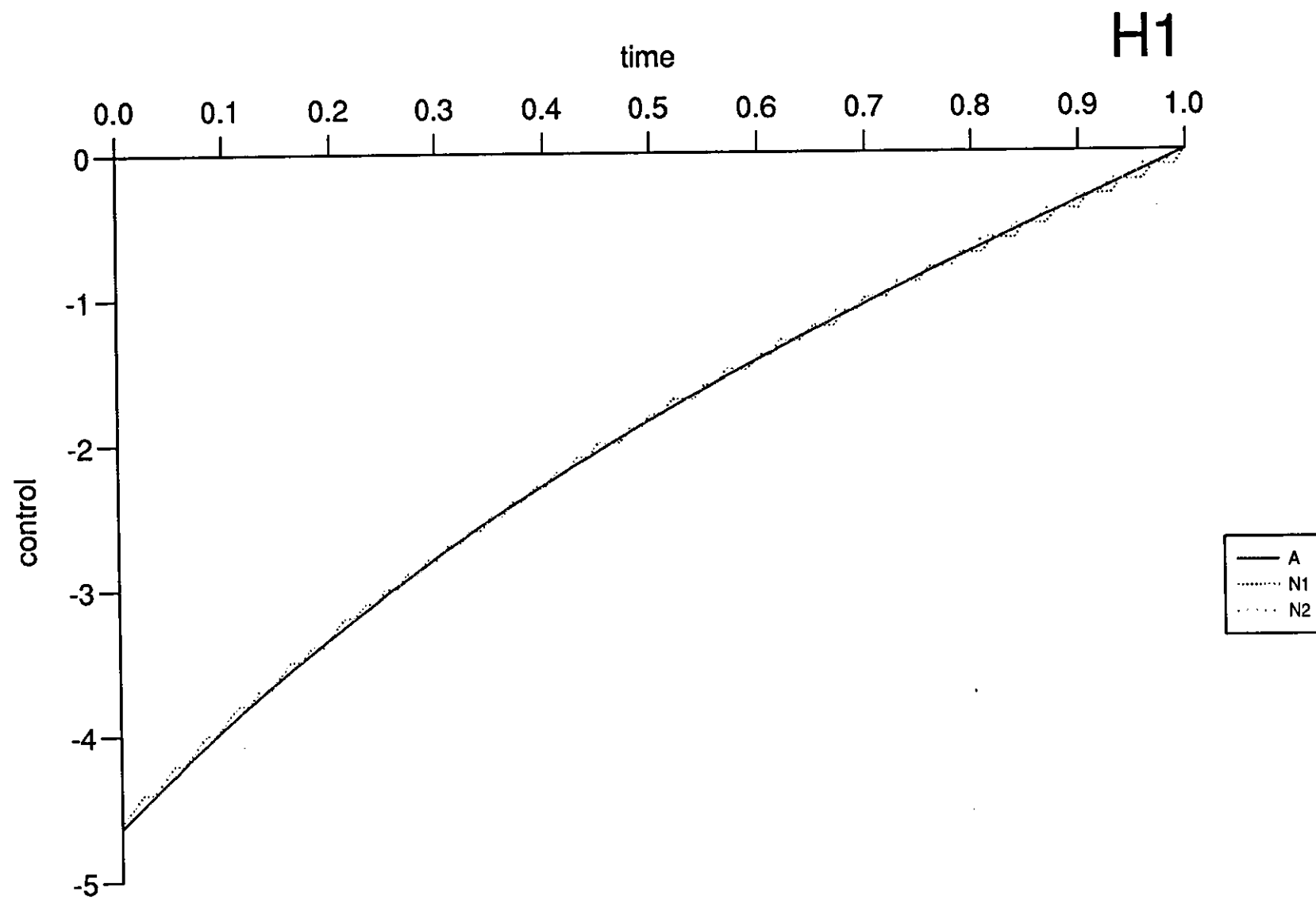


Figure (4.4.38)

H1

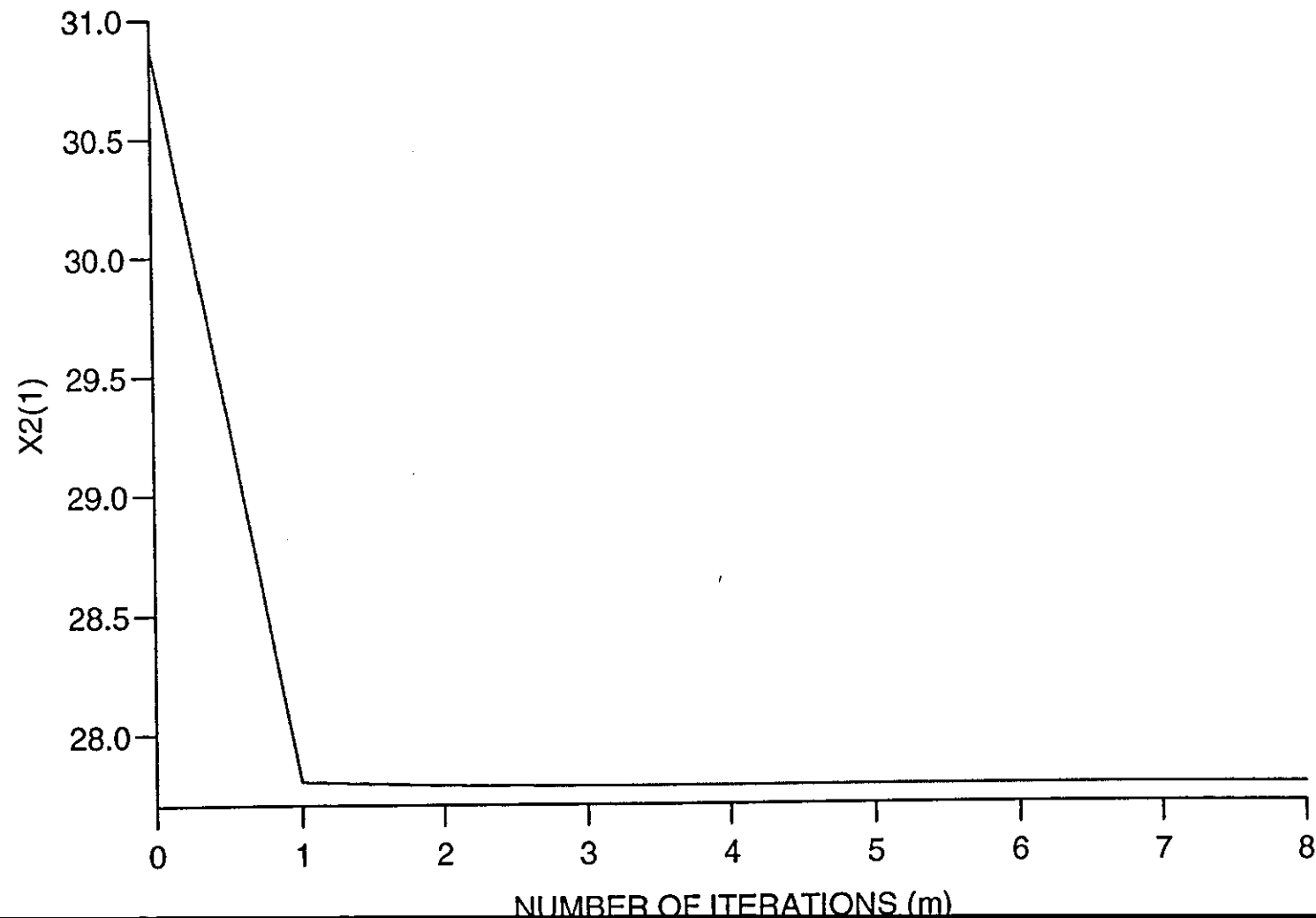


Figure (4.4.39)

H1

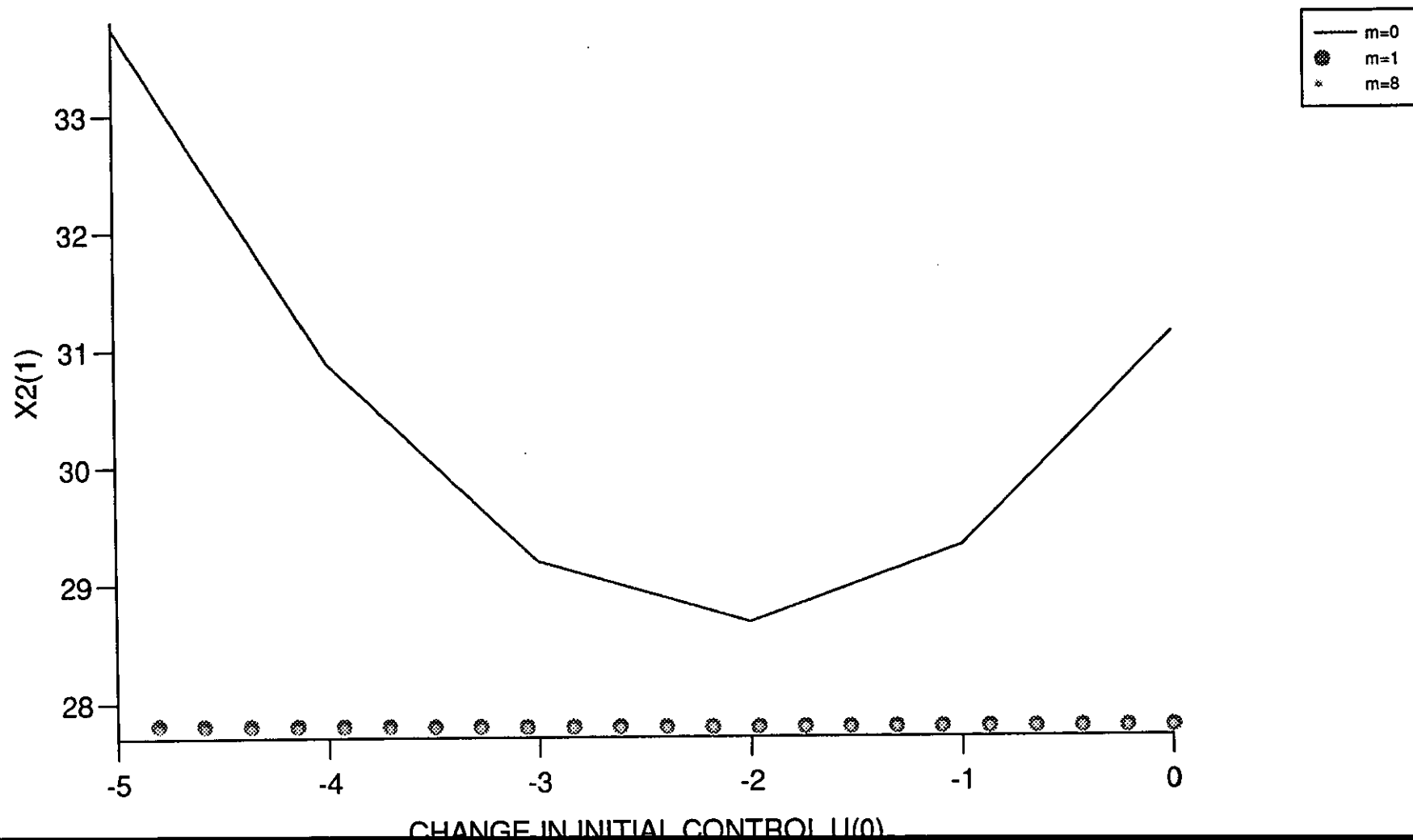


Figure (4.4.40)

ATH

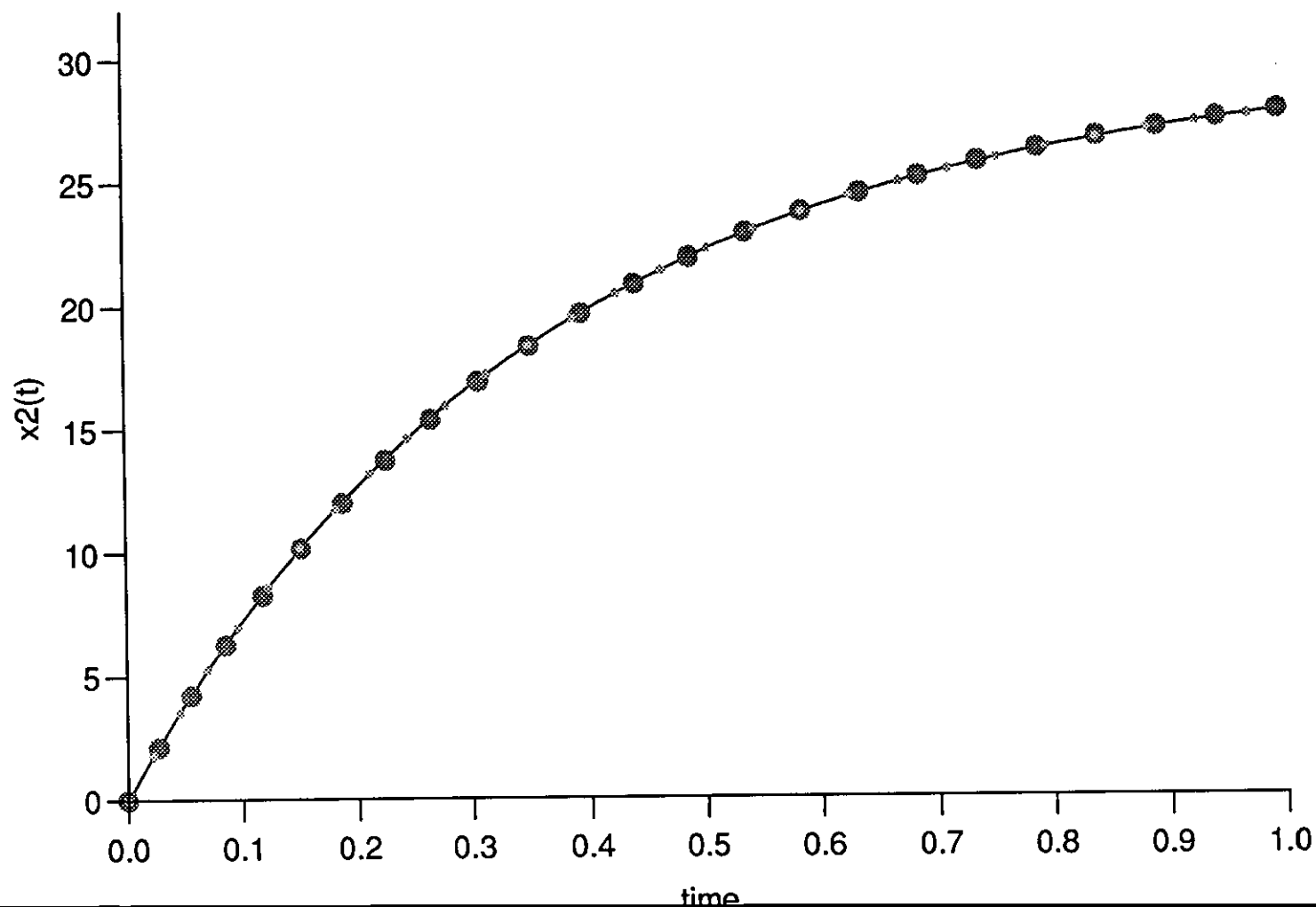
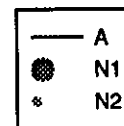


Figure (4.4.41)

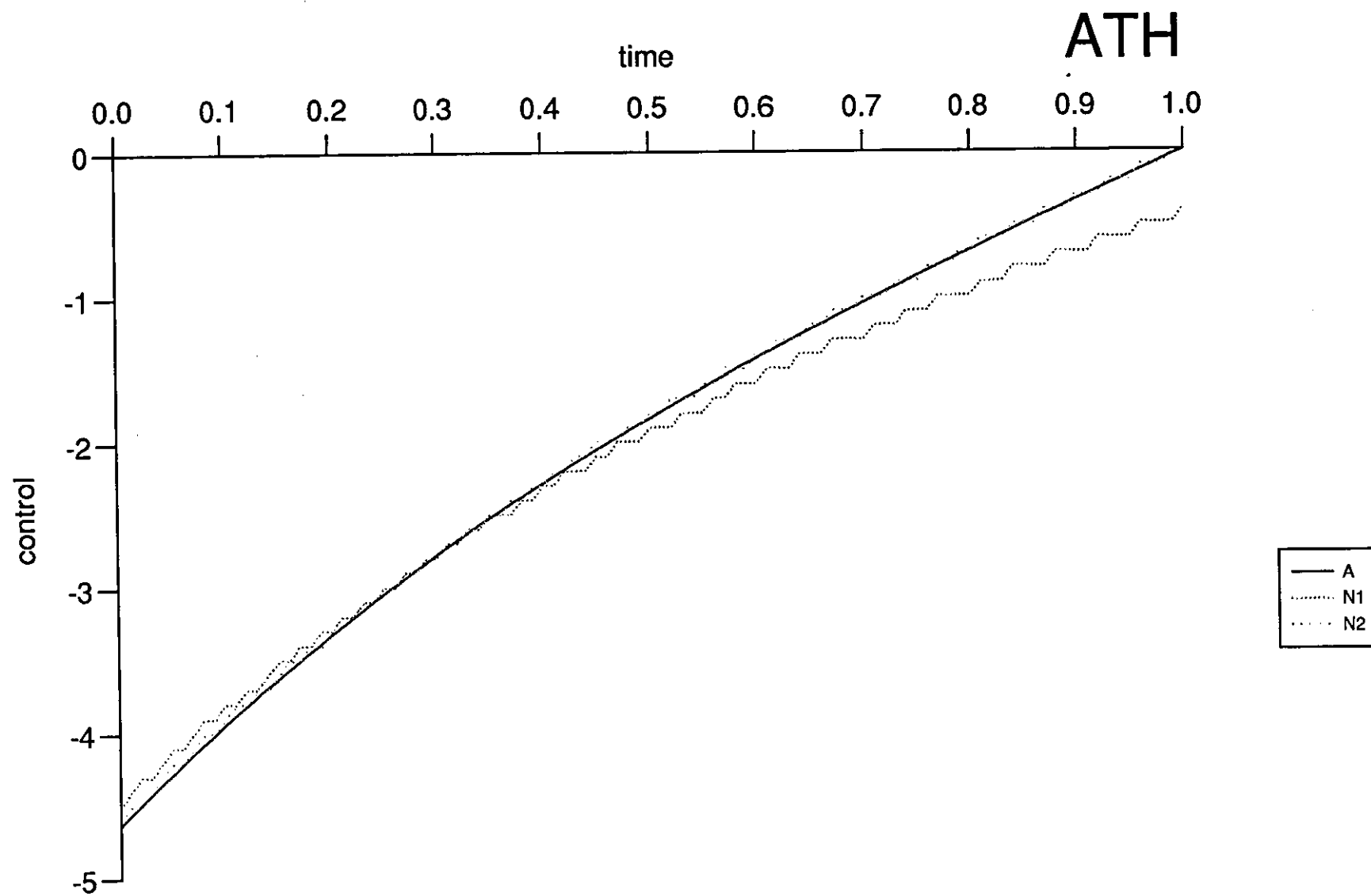


Figure (4.4.42)

ATH

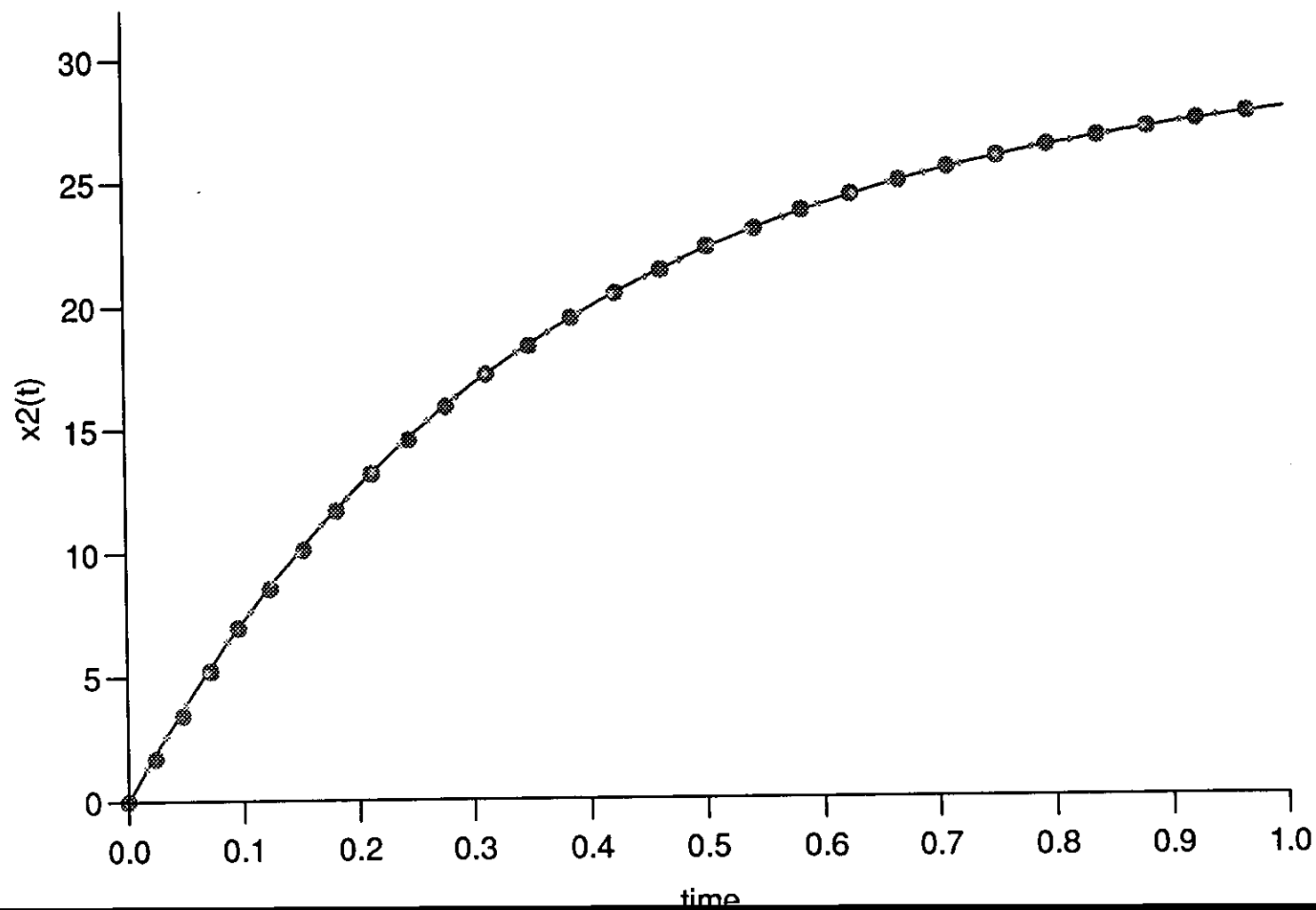
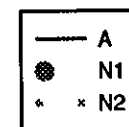


Figure (4.4.43)

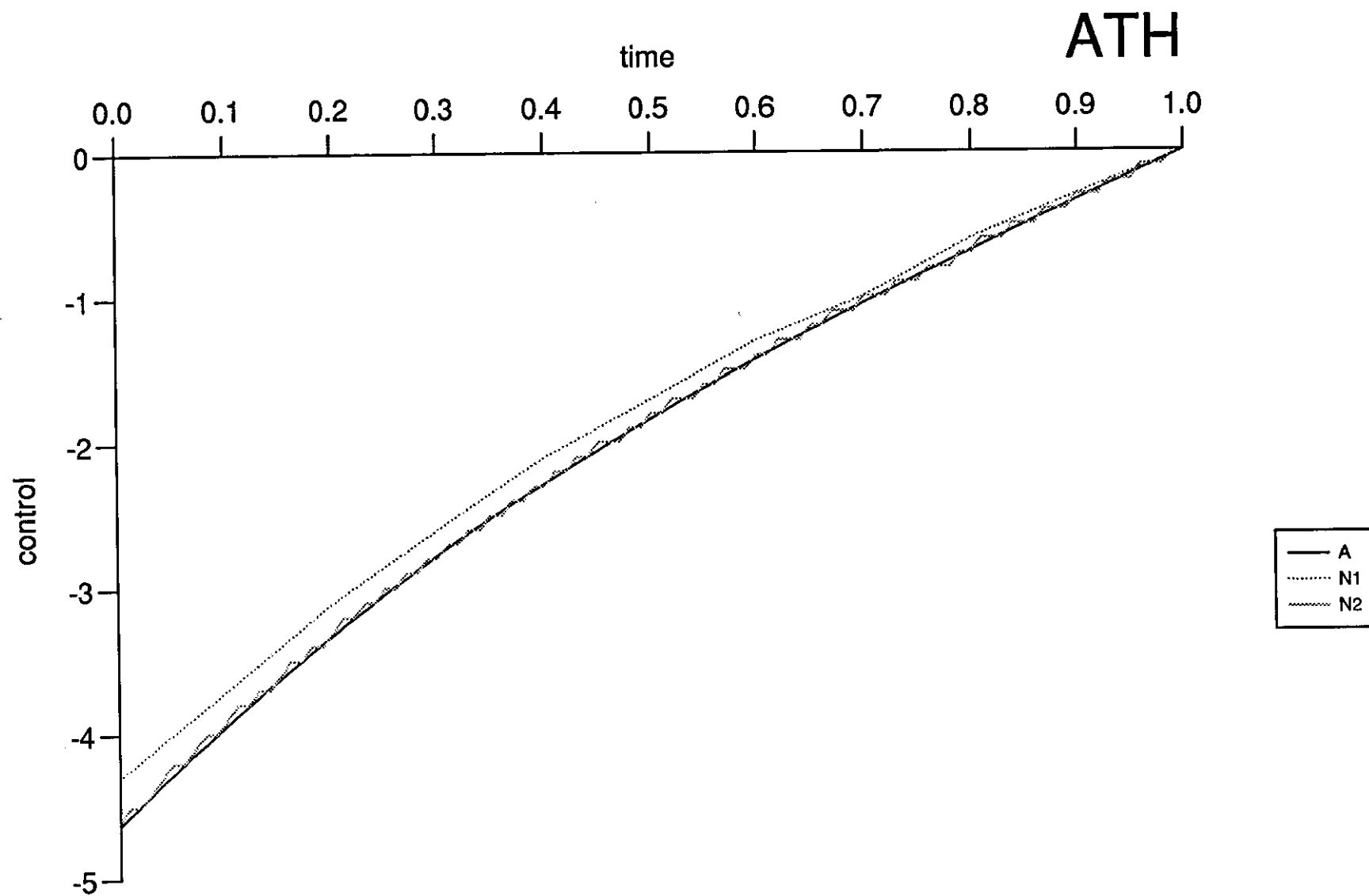


Figure (4.4.44)

ATH

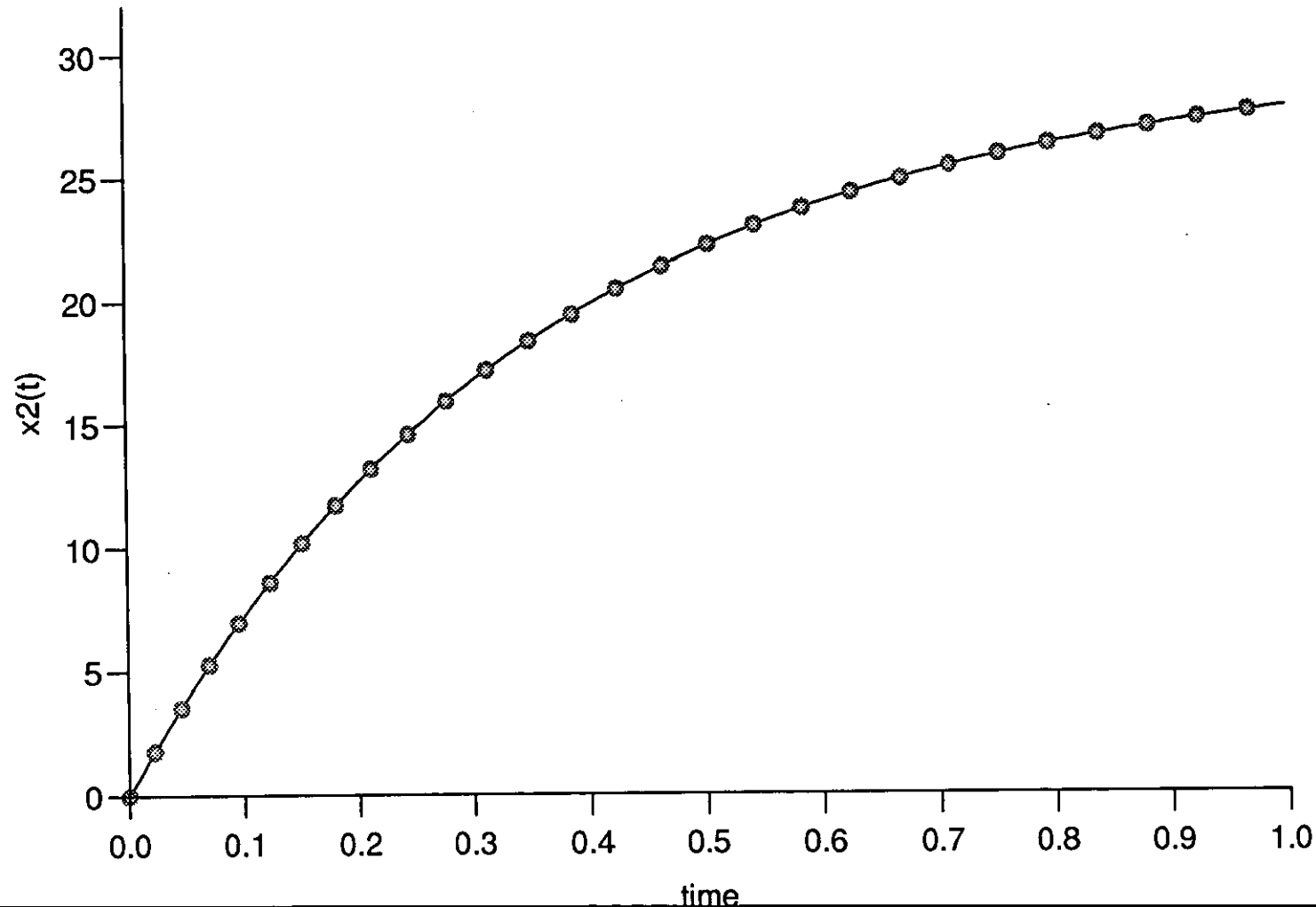
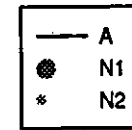


Figure (4.4.45)

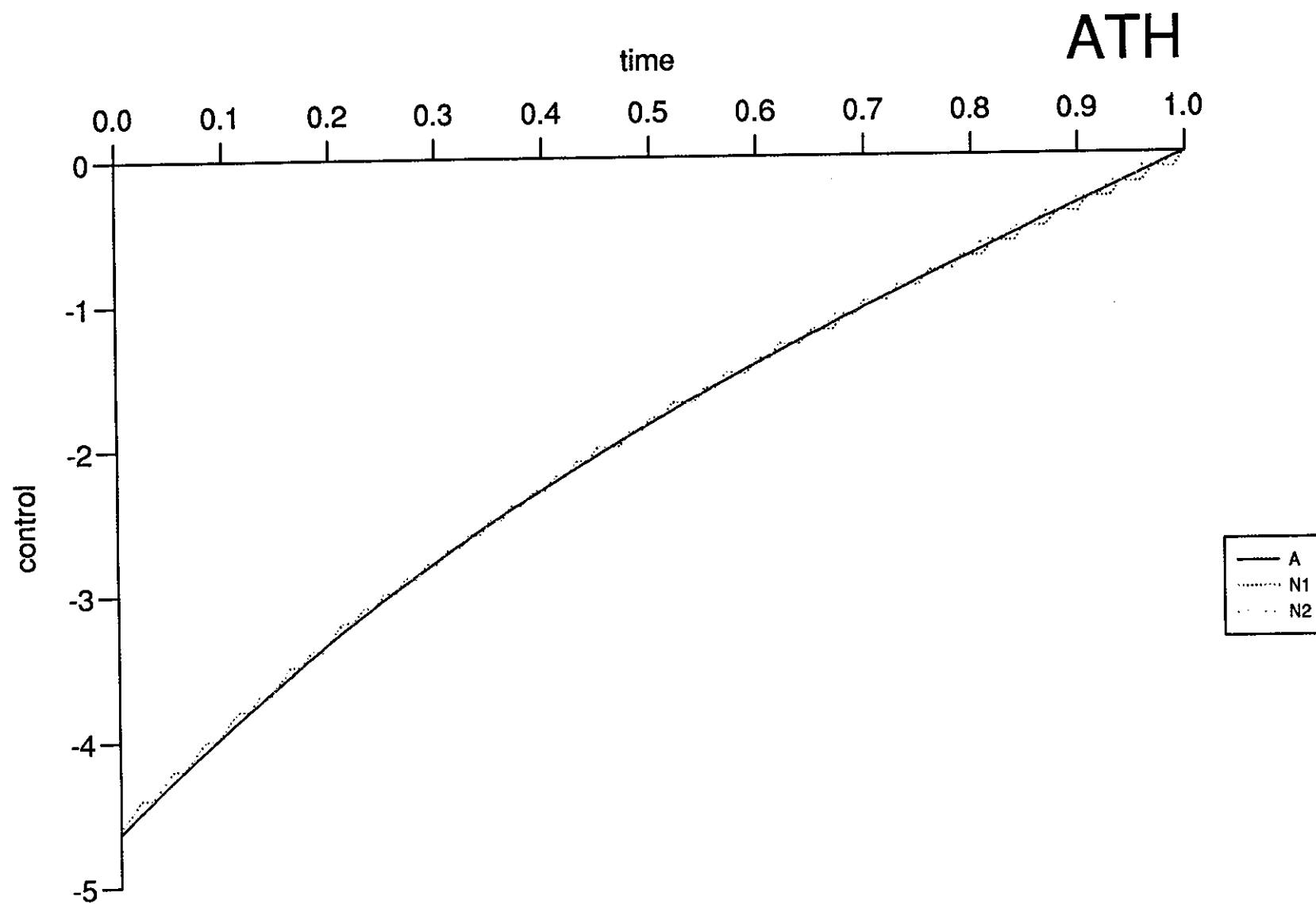


Figure (4.4.46)

ATH

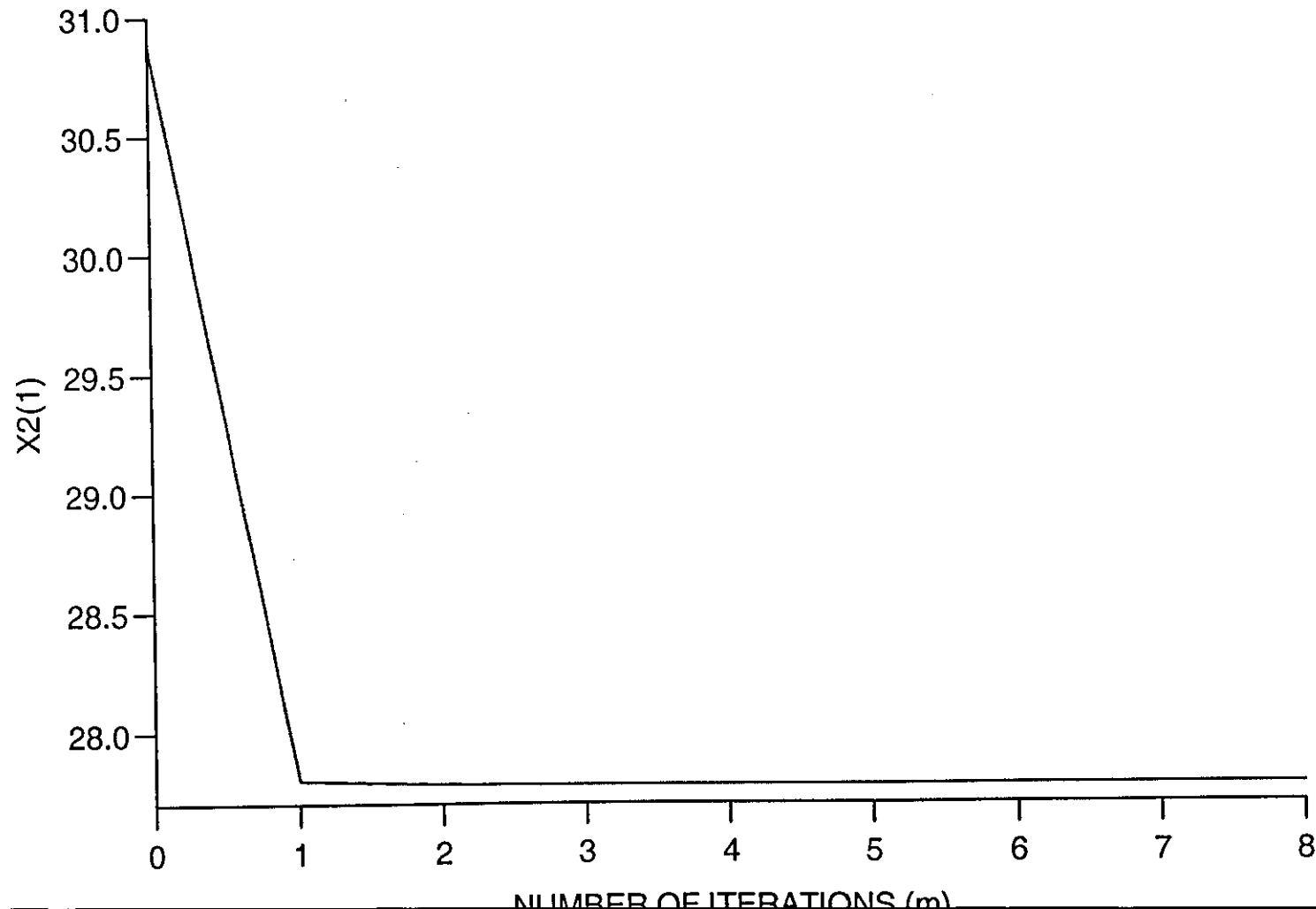


Figure (4.4.47)

ATH

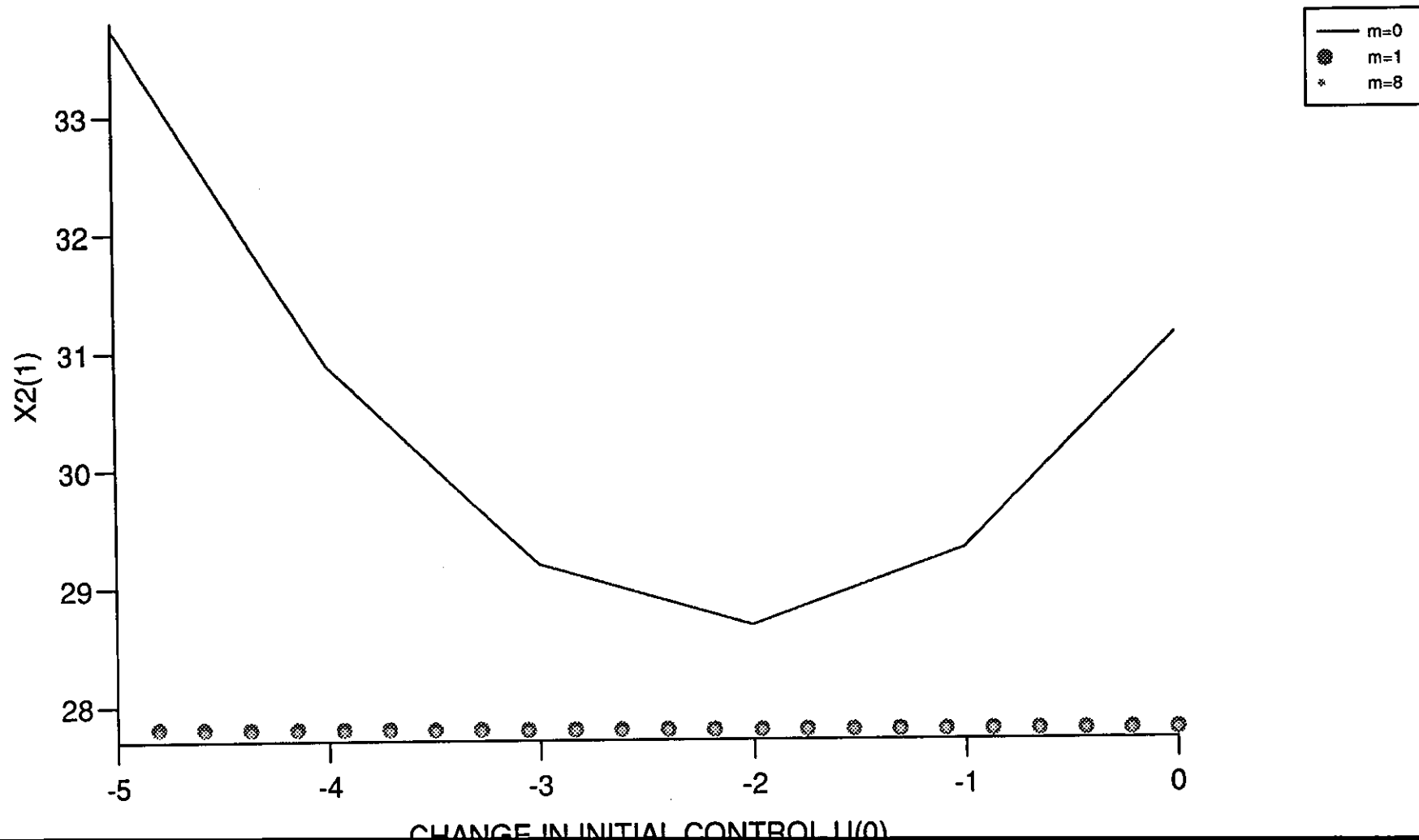


Figure (4.4.48)

H3

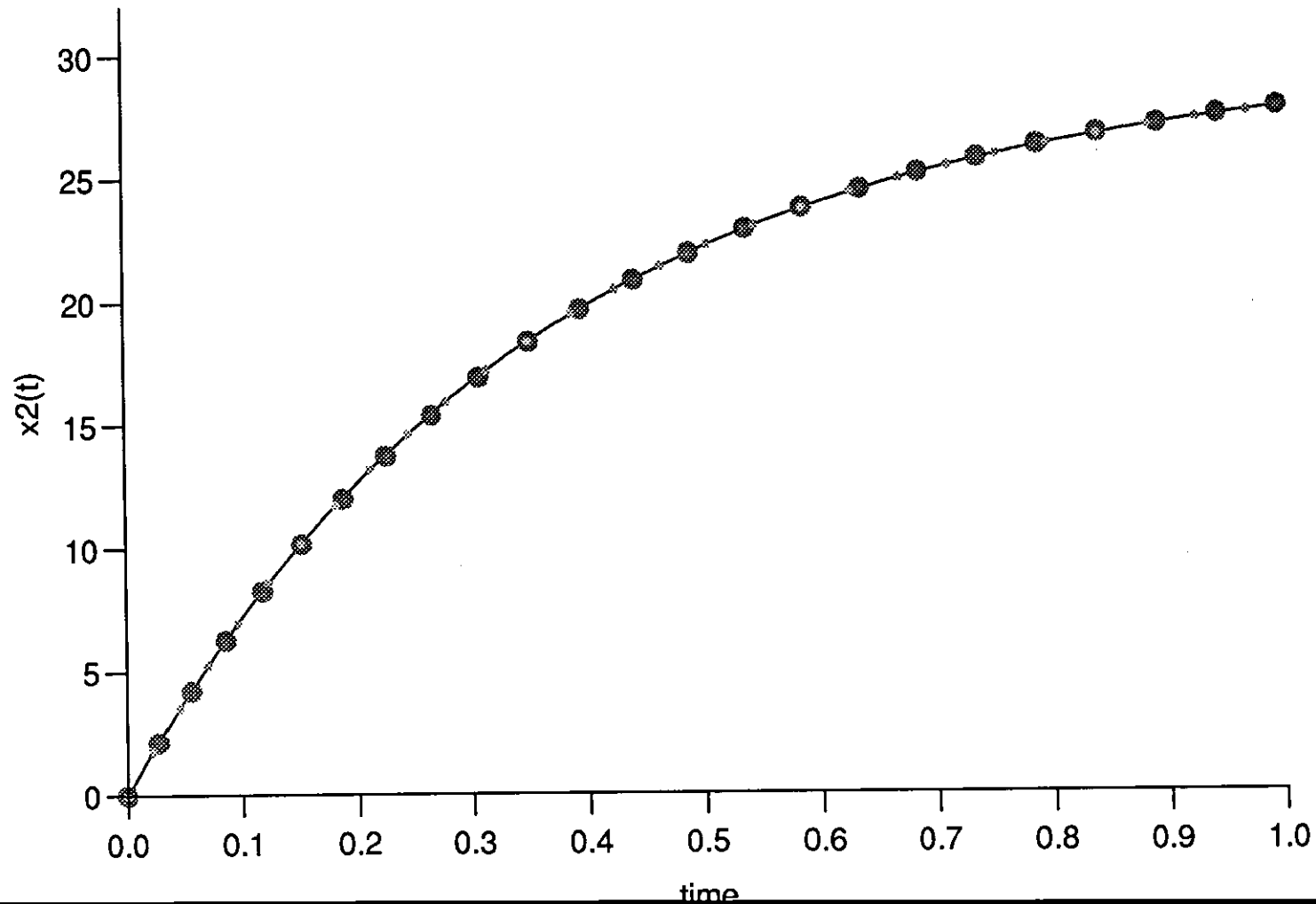
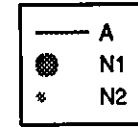


Figure (4.4.49)

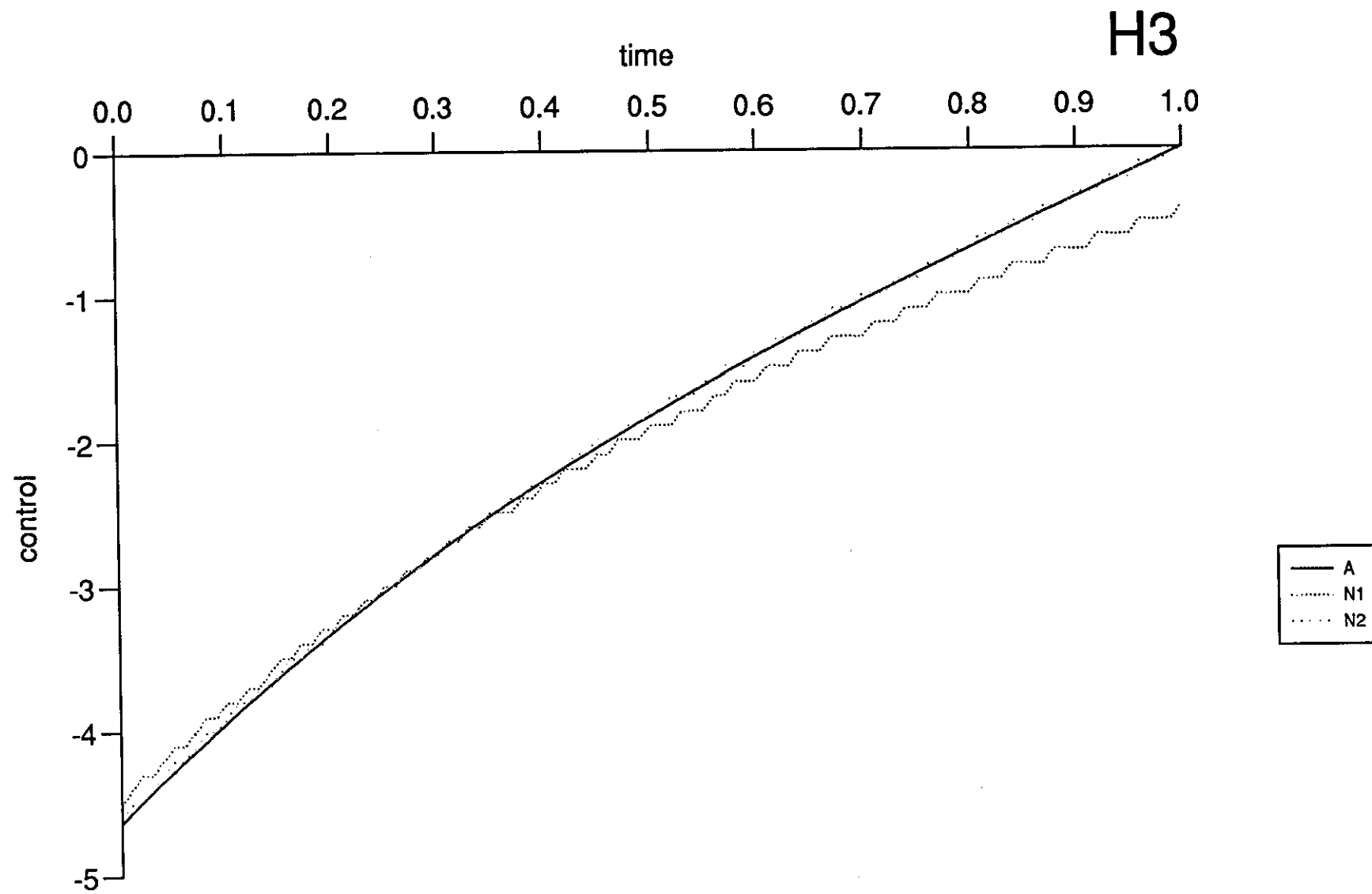


Figure (4.4.50)

H3

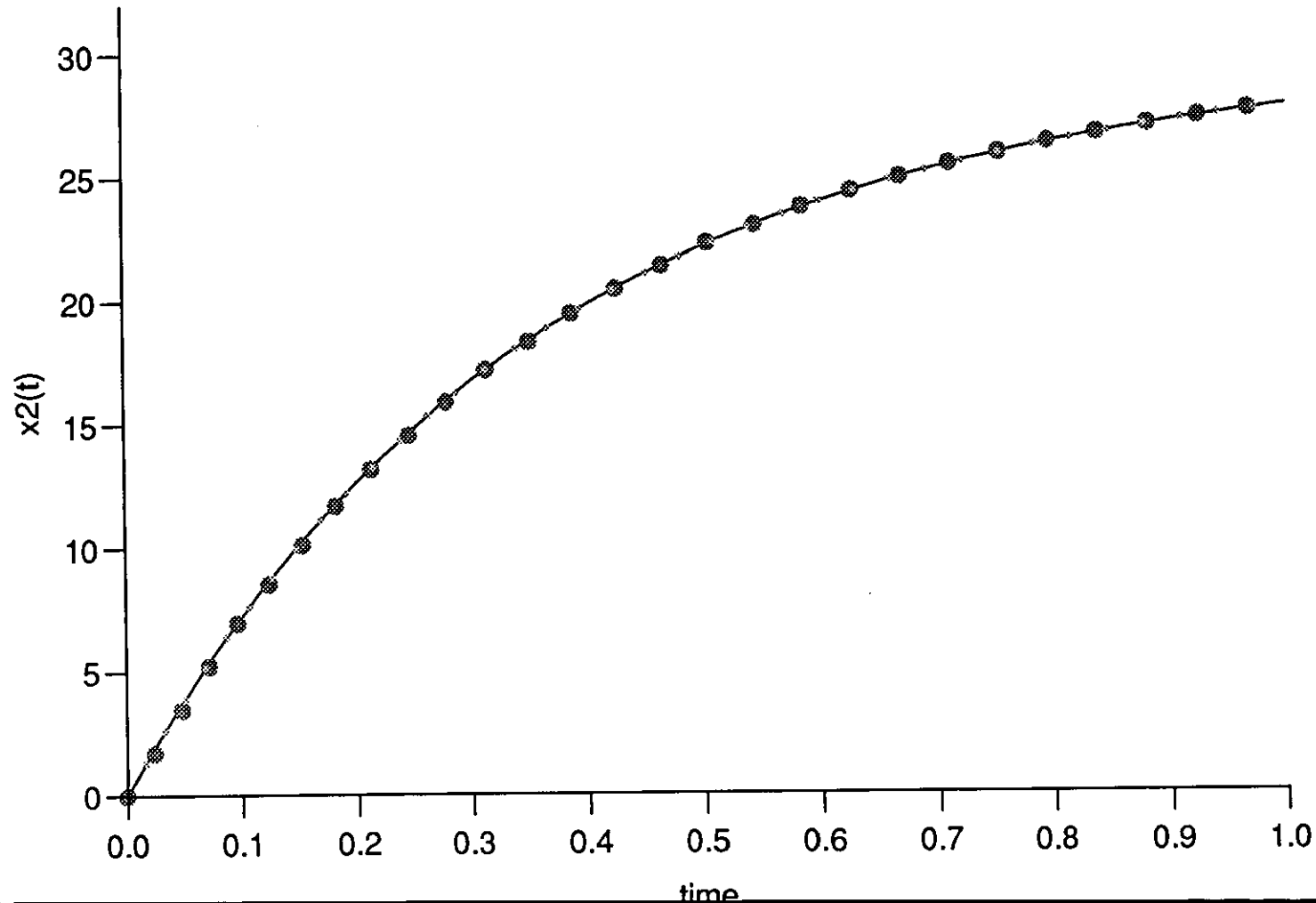
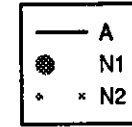


Figure (4.4.51)

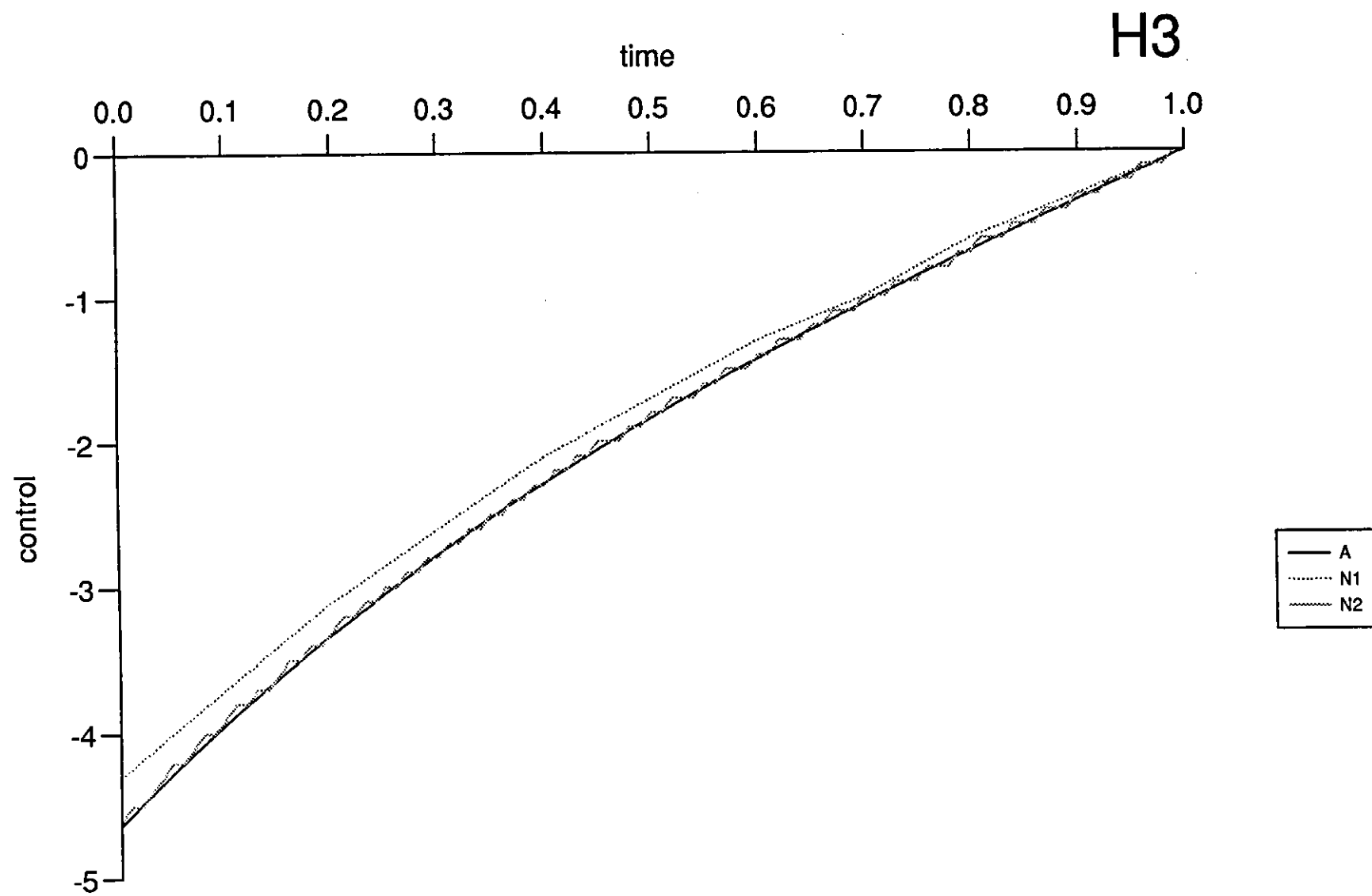


Figure (4.4.52)

H3

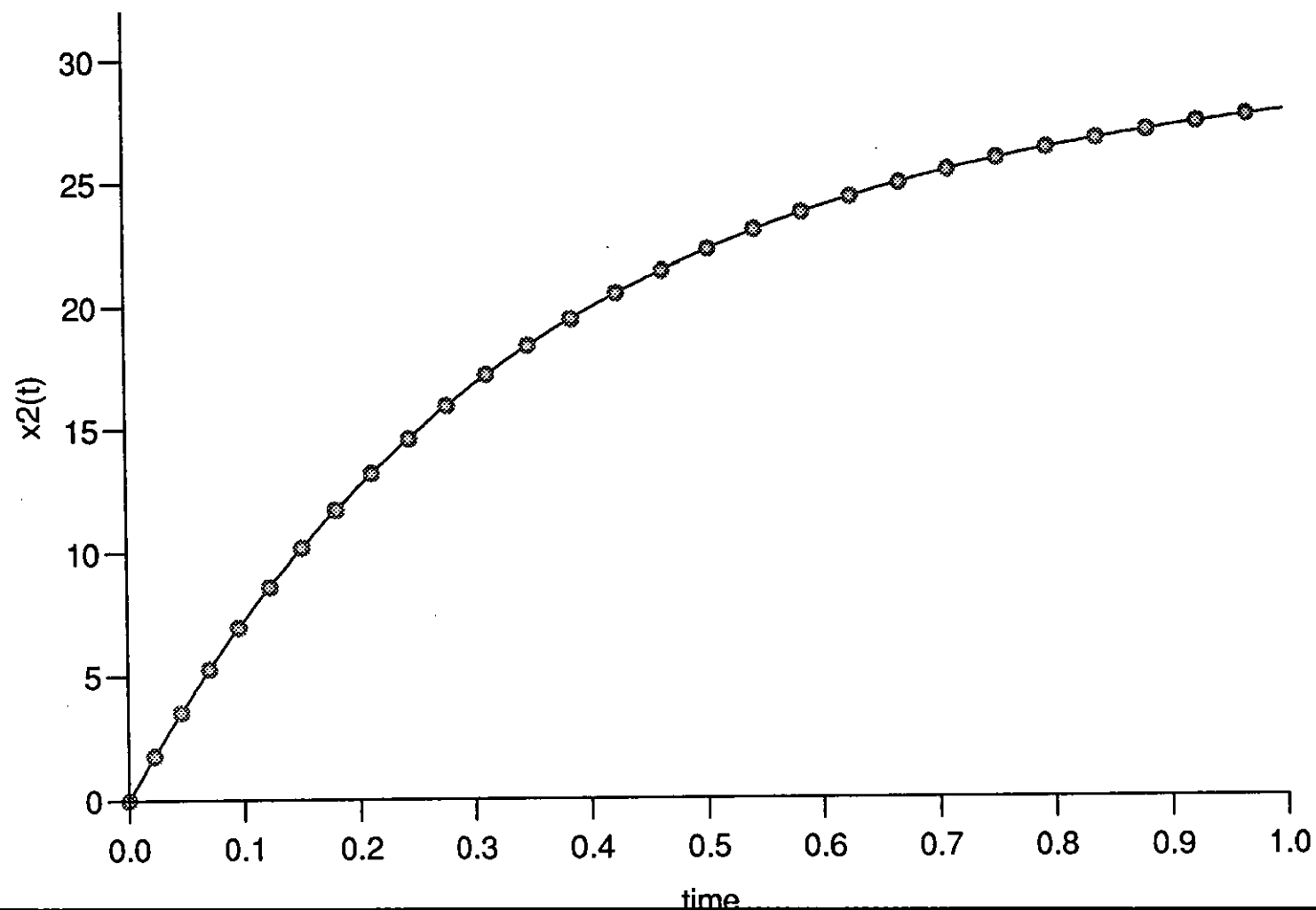
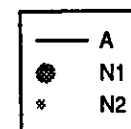


Figure (4.4.53)

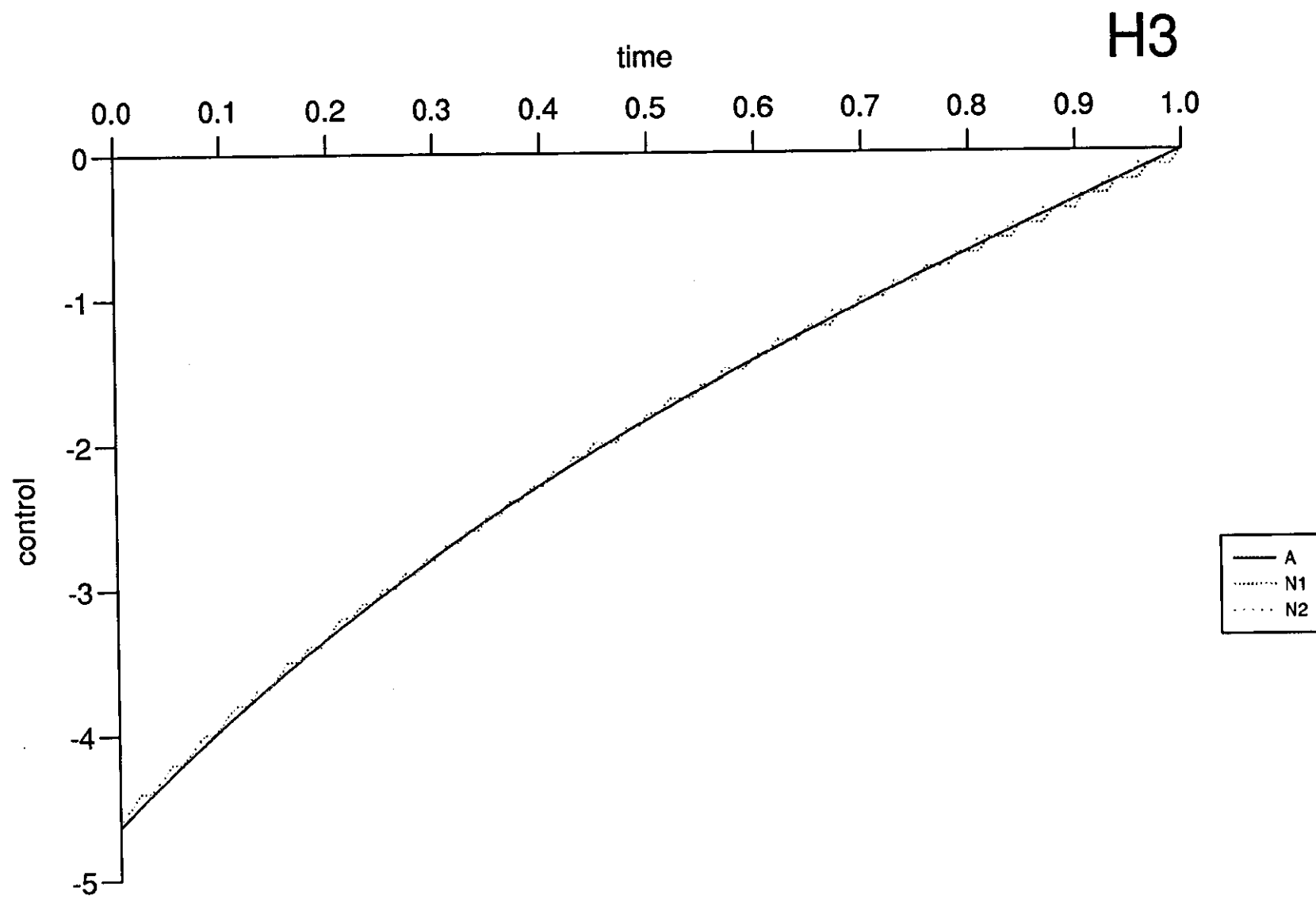


Figure (4.4.54)

H3

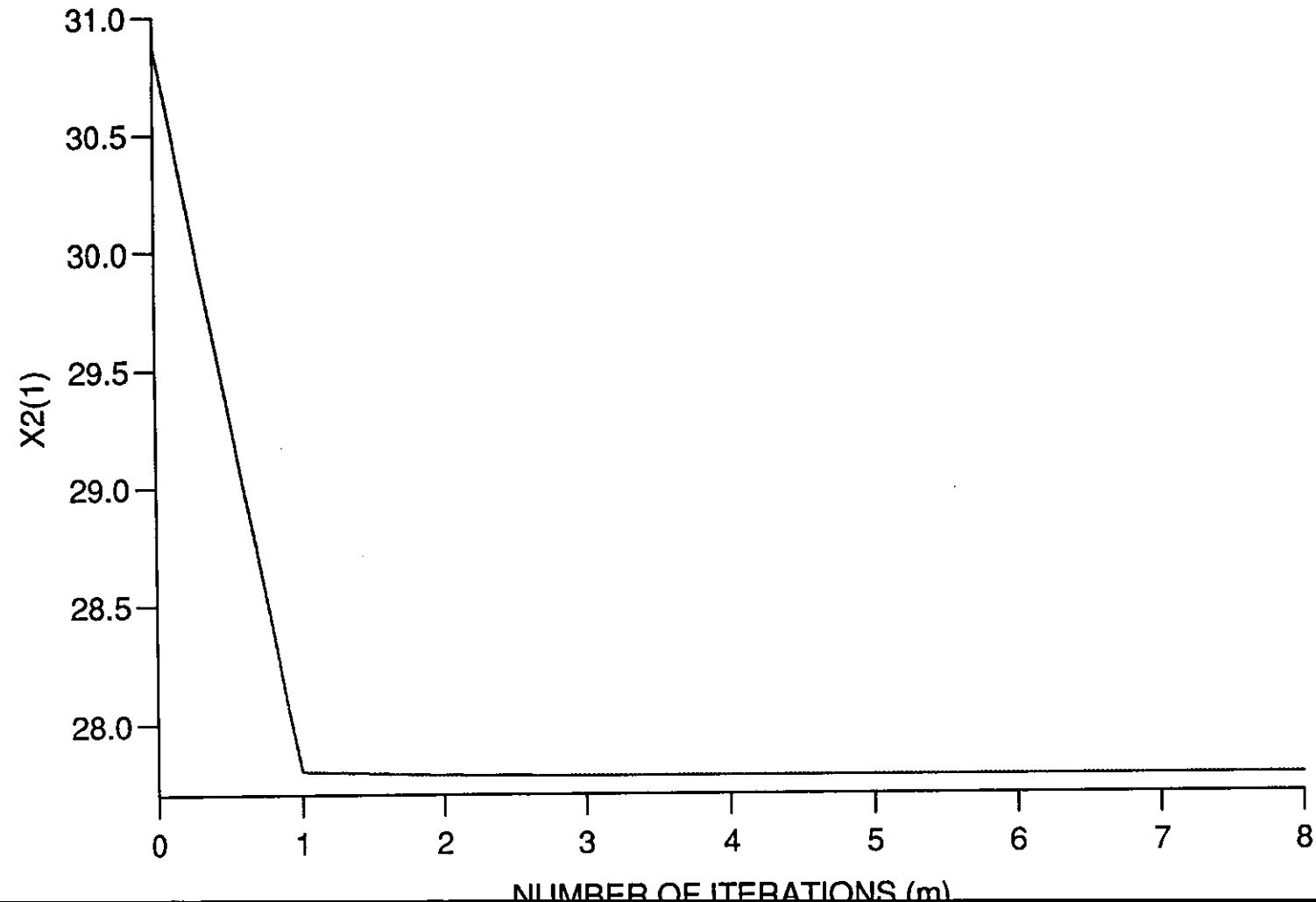


Figure (4.4.55)

H3

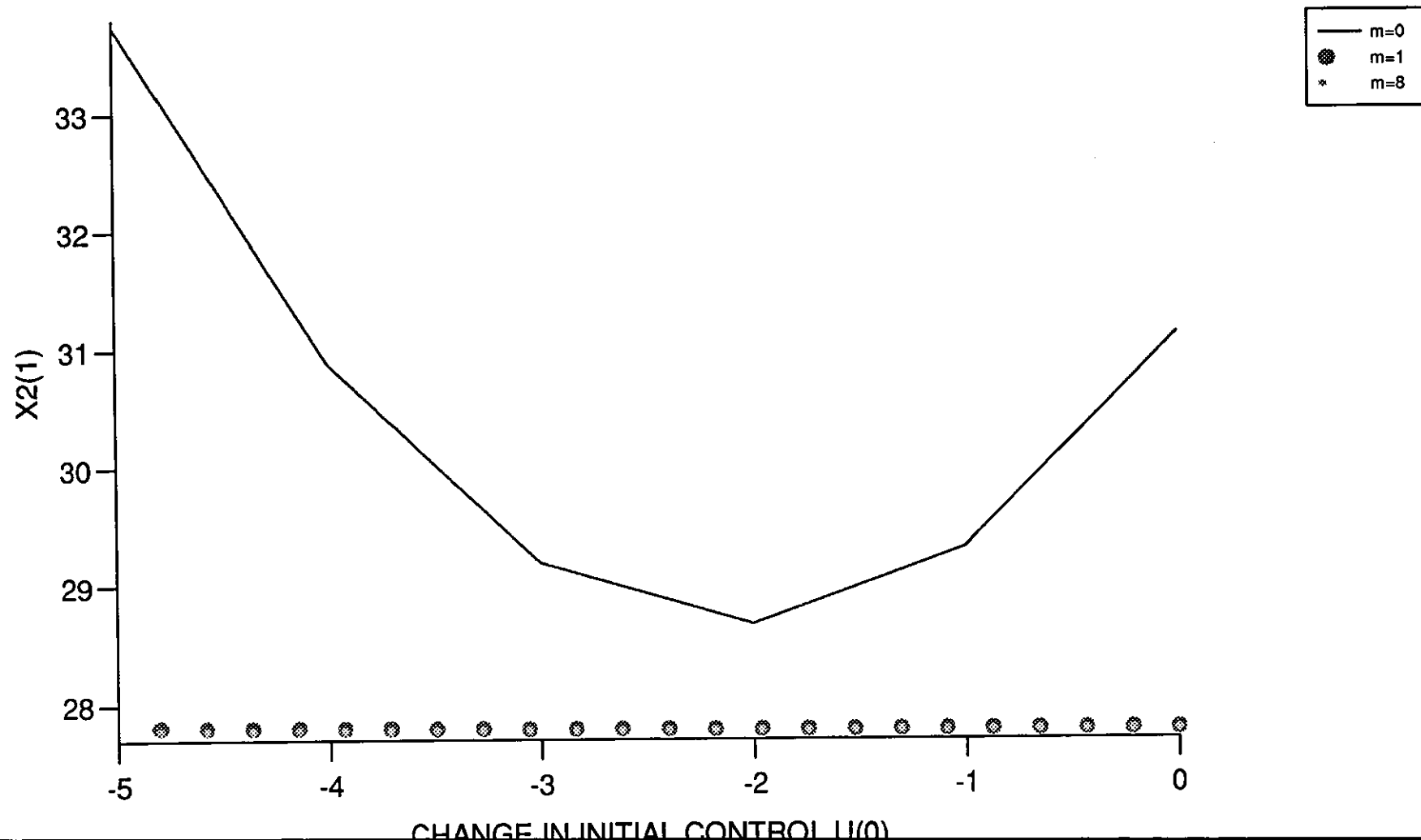


Figure (4.4.56)

comparison of the seven methods

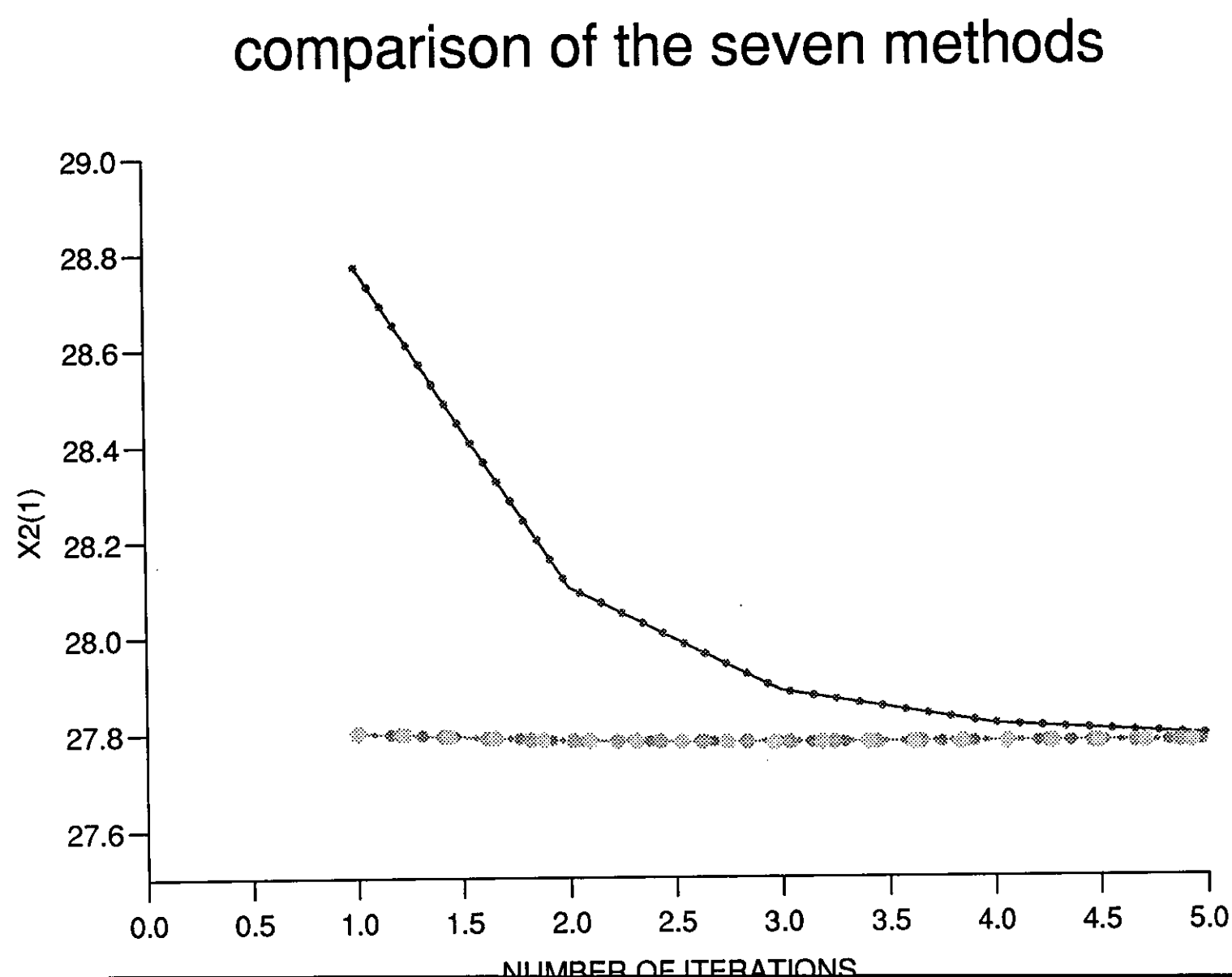


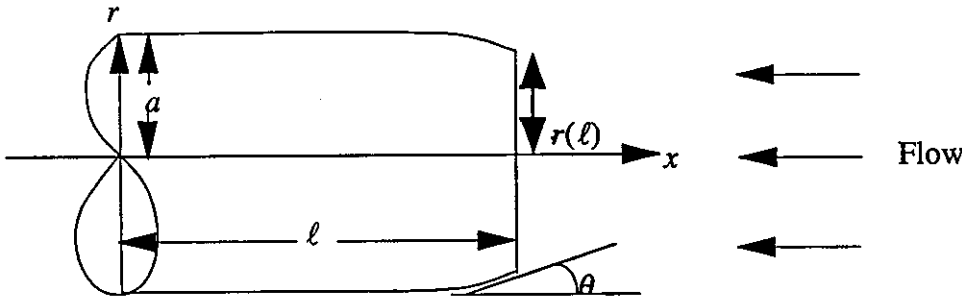
Figure (4.5.1)

Chapter 5

PROBLEM 2

5.1 Minimum Drag (MD) Problem

An unconstrained optimal control problem taken from a report by S.C. Carg (1979) [19] is considered in this chapter. This problem was also considered by Bryson and Ho (1969) [2]. The aim is to minimize the Drag Coefficient of an axisymmetric body in Newtonian hypersonic flow. The pressure drag of a body in hypersonic flow is given by the expression (see sketch)



$$D = -2\Pi q \left[\int_{x=0}^{\ell} C_p(\theta) r dr - r^2(\ell) \right] \quad (5.1.1)$$

where

$$C_p(\theta) = \begin{cases} 2 \sin^2 \theta; & \theta \geq 0 \\ 0; & \theta < 0 \end{cases} = \text{pressure coefficient.} \quad (5.1.2)$$

(Newtonian approximation),

q = dynamic pressure,

x = axial distance from point of maximum radius,

$r = r(x)$ = radius of body,

ℓ = length of body,

and $r(0) = a$ = maximum radius of body.

The slope is given by $dr/dx = -\tan \theta$, the problem can be cast into the form of a fixed terminal time, unconstrained optimal control problem as follows:

Select $\tan \theta = u$ as the control variable, then

$$C_p dr = 2 \sin^2 \theta \frac{dr}{dx} dx = \frac{-2u^3}{1+u^2} dx. \quad (5.1.3)$$

Allow for the possibility of a blunt tip by writing (5.1.1) in the form

$$\begin{aligned} \frac{D}{4\Pi q} &= \frac{-2\Pi q}{4\Pi q} \left[\int_0^\ell C_p(\theta) r dr - r^2(\ell) \right] \\ \frac{D}{4\Pi q} &= \frac{1}{2} r^2(\ell) - \frac{1}{2} \int_0^\ell \frac{-2ru^3}{1+u^2} dx, \end{aligned}$$

i.e.,

$$\frac{D}{4\Pi q} = \frac{1}{2} r^2(\ell) + \int_0^\ell \frac{ru^3}{1+u^2} dx. \quad (5.1.4)$$

By nondimensionalizing the variables, make $x_1 = r/\ell$, $\dot{x}_2 = 1/\ell \times$ the integrand in (5.1.4) and insert a non-negativity constraint on u to enforce (5.1.2). The independent variable $t = x/\ell$.

The problem now reduces to:

Minimize

$$\left. \begin{aligned} \phi &= \frac{1}{2} x_1^2(1) + x_2(1) \\ \text{with} \\ \dot{x}_1 &= -u, x_1(0) = \frac{a}{\ell}, u \geq 0 \\ \dot{x}_2 &= \frac{u^3 x_1}{1+u^2}, x_2(0) = 0 \\ \text{and} \\ C_D &= 4(\ell/a)^2 \phi. \end{aligned} \right\} \quad (5.1.5)$$

When C_D the drag coefficient is defined as $\frac{D}{\Pi q a^2}$. Refer to [2]. See also Appendix A1 for the complete derivation of (5.1.5). Once again the problem has been tackled analytically and then numerically using conjugate gradient and hybrid conjugate gradient method mentioned in chapter 4, section 4.1.

5.2 Analytical Solution

Here we use the maximum principle to solve problem (5.1.5). We set up the Hamiltonian as follows:

$$H = \sum_{i=1}^{i=2} \lambda_i f_i = -\lambda_1 u + \lambda_2 \frac{u^3 x_1}{1+u^2} \quad (5.2.1)$$

The adjoint equations can be written as,

$$\begin{aligned} \dot{\lambda}_i &= -\frac{\partial H}{\partial x_i}, \quad i = 1, 2, \\ \lambda_i(t_f) &= \frac{\partial \phi}{\partial x_i} \Big|_{t=t_f}, \quad i = 1, 2, \end{aligned}$$

i.e.,

$$\left. \begin{aligned} \dot{\lambda}_1 &= -\frac{\partial H}{\partial x_1} = -\lambda_2 \frac{u^3}{1+u^2}, \lambda_1(t_f) = x_1(1), \\ \dot{\lambda}_2 &= -\frac{\partial H}{\partial x_2} = 0, \lambda_2(t_f) = 1, \end{aligned} \right\} \quad (5.2.2)$$

so that $\lambda_2 = 1$.

To find the optimal u we set $H_u = 0$.

$$H_u = \frac{\partial H}{\partial u} = -\lambda_1 + \frac{u^2 \lambda_2 x_1 (3 + u^2)}{(1 + u^2)^2} = 0,$$

i.e.,

$$\lambda_1 = \frac{\lambda_2 x_1 u^2 (3 + u^2)}{(1 + u^2)^2}.$$

Since $\lambda_2 = 1$, λ_1 becomes,

$$\lambda_1 = \frac{x_1 u^2 (3 + u^2)}{(1 + u^2)^2}.$$

Now substituting for λ_1 in H gives,

$$H = -\lambda_1 u + \frac{\lambda_2 u^3 x_1}{(1 + u^2)},$$

i.e.,

$$\begin{aligned} H &= \frac{-x_1 u^2 (3 + u^2) u}{(1 + u^2)^2} + \frac{\lambda_1 u^3 x_1}{(1 + u^2)} \\ &= \frac{-x_1 u^2 (3 + u^2) u + \lambda_2 u^3 x_1 (1 + u^2)}{(1 + u^2)^2} \\ &= \frac{-3x_1 u^2 - x_1 u^5 + u^3 x_1 + u^5 x_1}{(1 + u^2)^2} \\ &= \frac{-2x_1 u^3}{(1 + u^2)^2}. \end{aligned}$$

Now $H = \text{const}$, since the problem does not involve t explicitly. Therefore,

$$H = \frac{-2x_1 u^3}{(1 + u^2)^2} = \text{const.} \quad (5.2.3)$$

At $t_f = 1$,

$$\lambda_1(1) = \frac{x_1(1) u^2(1) (3 + u^2(1))}{(1 + u(1)^2)^2}$$

and since from (5.2.2), $\lambda_1(t_f) = \lambda_1(1) = x_1(1)$, the above equation reduces to,

$$u^2(1) = 1 \implies u(1) \pm 1,$$

but because of the non-negativity condition we must have $u(1) = 1$.

From equation (5.2.3), at $t = t_f = 1$ we have,

$$\frac{-2x_1(1) u^3(1)}{(1 + u(1)^2)^2} = \text{const},$$

and since $u(1) = 1$, it follows that,

$$\frac{-2x_1(1)}{4} = \text{const.} \quad (5.2.4)$$

From (5.2.3) and (5.2.4) we get,

$$\begin{aligned}\frac{-2x_1u^3}{(1+u^2)^2} &= \frac{-2}{4}x_1(1), \text{ i.e.,} \\ \frac{x_1}{x_1(1)} &= \frac{(1+u^2)^2}{4u^3}.\end{aligned}\quad (5.2.5)$$

Thus,

$$x_1(t) = \frac{(1+u^2(t))^2}{4u^3(t)}x_1(1). \quad (5.2.6)$$

Equation (5.2.6) gives the body shape in terms of u and $x_1(1)$. To introduce t into the solution we proceed as follows:

$$\begin{aligned}\frac{dx_1}{du} &= \frac{-x_1(1)}{2u^2} + \frac{x_1(1)}{4} - \frac{3x_1(1)}{4u^4}, \text{ and,} \\ \frac{dx_1}{du} &= \frac{dx_1}{dt} \cdot \frac{dt}{du}.\end{aligned}\quad (5.2.7)$$

But from (5.1.5),

$$\frac{dx_1}{dt} = -u(t),$$

so from (5.2.7) we get,

$$\frac{dx_1}{du} = -u \frac{dt}{du},$$

i.e.,

$$\frac{1}{u} \cdot \frac{dx_1}{du} = -\frac{dt}{du}.$$

Now integrating both sides,

$$\begin{aligned}\int_1^u \frac{1}{u} \frac{dx_1}{du} du &= \int_{u=1}^{u=u} -dt, \\ \int_1^u \frac{1}{u} \frac{dx_1}{du} du &= \int_1^u \left(\frac{-x_1(1)}{2u^3} + \frac{x_1(1)}{4u} - \frac{3x_1(1)}{4u^5} \right) du \\ &= x_1(1) \int_1^u \left(\frac{-1}{2}u^{-3} + \frac{1}{4}u^{-1} - \frac{3}{4}u^{-5} \right) du \\ &= x_1(1) \frac{1}{4} \left[\frac{1}{u^2} - \ln \frac{1}{u} + \frac{3}{4u^4} - \frac{7}{4} \right].\end{aligned}\quad (5.2.8)$$

From (5.2.5) at $u = 1$,

$$\frac{x_1}{x_1(1)} = 1$$

i.e.,

$$x_1(t) = x_1(1), \text{ thus at } u = 1, t = 1 \text{ and at } u = u, t = t.$$

$$\text{Thus } \int_1^u \frac{1}{u} \frac{dx_1}{du} du = \int_{t=1}^{t=t} -dt = -t + 1, \text{ i.e.,}$$

$$1 - t = \int_1^u \frac{1}{u} \frac{dx_1}{du} du. \quad (5.2.9)$$

From (5.2.8) and (5.2.9) we get,

$$\begin{aligned}1 - t &= \frac{x_1}{4} \left[\frac{1}{u^2} - \ln \frac{1}{u} + \frac{3}{4u^4} - \frac{7}{4} \right] \\ \text{or } \frac{1 - t}{x_1(1)} &= \frac{1}{4} \left[\frac{3}{4u^4} + \frac{1}{u^2} - \frac{7}{4} - \ln \frac{1}{u} \right].\end{aligned}\quad (5.2.10)$$

From (5.2.5) at $t = 0$ we get,

$$\frac{x_1(0)}{x_1(1)} = \frac{(1 + u_0^2)^2}{4u_0^3},$$

and since $x_1(0) = a/\ell$ from (5.1.5), then by selecting the fineness ratio $\frac{a}{\ell} = 0.5$ for simplicity we get,

$$\frac{0.5}{x_1(1)} = \frac{(1 + u_0^2)^2}{4u_0^3}. \quad (5.2.11)$$

Now from (5.2.10) when $t = 0$ we have,

$$\frac{1}{x_1(1)} = \frac{1}{4} \left[\frac{3}{4u_0^4} + \frac{1}{u_0^2} - \frac{7}{4} - \ell n \frac{1}{u_0} \right]. \quad (5.2.12)$$

By multiplying both sides of (5.2.11) by 2 we get,

$$\frac{1}{x(1)} = \frac{(1 + u_0^2)^2}{2u_0^3}. \quad (5.2.13)$$

From (5.2.12) and (5.2.13) we have,

$$\frac{1}{4} \left[\frac{3}{4u_0^4} + \frac{1}{u_0^2} - \frac{7}{4} - \ell n \frac{1}{u_0} \right] = \frac{(1 + u_0^2)^2}{2u_0^3},$$

i.e.,

$$\frac{3}{8u_0} + \frac{u_0}{2} - \frac{7}{8}u_0^3 = \frac{u_0^3}{2}\ell n \frac{1}{u_0} - (1 + u_0^2)^2 = 0. \quad (5.2.14)$$

Solving equation (5.2.14) numerically using the Newton Raphson method gives, $u_0 = 0.3342450$.

Substituting u_0 in (5.2.12) we get, $x_1(1) = 0.0604275$. From (5.2.5) and (5.1.5) we can write,

$$\frac{u^3 x_1(1) \frac{(1 + u^2)^2}{4u^3}}{(1 + u^2)} = \frac{x_1(1)(1 + u^2)}{4(1 + u^2)},$$

so that,

$$\int_0^1 dx_2 = \int_0^1 \frac{x_1(1)(1 + u^2)^2}{4} dt,$$

and

$$x_2(1) = \int_0^1 \frac{x_1(1)}{4} (1 + u^2) dt, \quad (5.2.15)$$

since $x_2(0) = 0$. From (5.1.5),

$$\phi = \frac{1}{2}x_1^2(1) + x_2(1). \quad (5.2.16)$$

Now substitute (5.2.15) into (5.2.16) to get,

$$\phi = \frac{1}{2}x_1^2(1) + x_1(1) \int_0^1 \frac{(1 + u^2)}{4} dt. \quad (5.2.17)$$

From (5.2.10) we have,

$$1 - t = \frac{x_1(1)}{4} \left[\frac{3}{4u^4} + \frac{1}{u^2} - \frac{7}{4} - \ell n \frac{1}{u} \right]$$

or

$$1 - \frac{x_1(1)}{4} \left[\frac{3}{4u^4} + \frac{1}{u^2} - \frac{7}{4} - \ell n \frac{1}{u} \right] = t.$$

Hence

$$\begin{aligned} \frac{dt}{du} &= \frac{d}{du} \left\{ 1 - \frac{x_1(1)}{4} \left[\frac{3}{4}u^{-4} + u^{-2} - \frac{7}{4} - \ell n u \right] \right\} \\ &= \frac{-x_1(1)}{4} \left[\frac{-12}{4}u^{-5} - 2u^{-3} + \frac{1}{u} \right] \end{aligned}$$

i.e.,

$$\begin{aligned} \frac{dt}{du} &= \frac{-x_1(1)}{4} \left[-3u^{-5} - 2u^{-3} + \frac{1}{u} \right] \\ dt &= \frac{x_1(1)}{4} \left[3u^{-5} + 2u^{-3} - \frac{1}{u} \right] du. \end{aligned} \quad (5.2.18)$$

Substituting (5.2.18) into (5.2.17) we get,

$$\begin{aligned} \phi &= \frac{1}{2}x_1^2(1) + \frac{x_1^2(1)}{16} \int_{u_0}^1 (1+u^2)(3u^{-5} + 2u^{-3} - \frac{1}{u}) du \\ &= \frac{1}{2}x_1^2(1) \left[1 + \frac{1}{8} \int_{u_0}^1 (1+u^2) \left(\frac{3u^{-5} + 2u^{-3} - u^{-4}}{u^5} \right) du \right] \\ &= \frac{1}{2}x_1^2(1) \left[1 + \frac{1}{8} \int_{u_0}^1 (3u^{-5} + 2u^{-3} - u^{-1} + 3u^{-3} + 2u^{-1} - u) du \right] \\ &= \frac{1}{2}x_1^2(1) \left[1 + \frac{1}{8} \int_{u_0}^1 (3u^{-5} + 5u^{-3} + u^{-1} - u) du \right]. \end{aligned} \quad (5.2.19)$$

At $t = 0, u(0) = u_0, t = 1$ and $u(1) = 1$ we have,

$$\begin{aligned} &\int_{u_0}^1 (3u^{-5} + 5u^{-3} + u^{-1} - u) du \\ &= \left[\frac{-3}{4}u^{-4} - \frac{5}{2}u^{-2} + \ell n u - \frac{1}{2}u^2 \right]_{u_0}^1 \\ &= -\frac{15}{4} + \frac{3}{4}u_0^{-4} + \frac{5}{2}u_0^{-2} - \ell n u_0 + \frac{1}{2}u_0^2. \end{aligned} \quad (5.2.20)$$

Substituting (5.2.20) into (5.2.19) we obtain,

$$\phi = \frac{x_1^2(1)}{2} \left[1 + \frac{1}{8} \left(-\frac{15}{4} + \frac{3}{4}u_0^{-4} + \frac{5}{2}u_0^{-2} - \ell n u_0 + \frac{1}{2}u_0^2 \right) \right]. \quad (5.2.21)$$

Now from (5.1.5) we have,

$$C_D = 4 \left(\frac{\ell}{a} \right)^2 \phi,$$

and,

$$x_1(0) = \frac{a}{\ell}$$

also from (5.2.5) at $t = 0$

$$x_1(0) = x_1(1) \frac{(1 + u_0^2)^2}{4u_0^3},$$

i.e.,

$$\frac{a}{\ell} = x_1(1) \frac{(1 + u_0^2)^2}{4u_0^3}. \quad (5.2.22)$$

From (5.2.12) we can write,

$$x_1(1) = \frac{4}{\left[\frac{3}{4u_0^2} + \frac{1}{u_0^2} - \frac{7}{4} - \ell n \frac{1}{u_0} \right]}, \quad (5.2.23)$$

and from (5.2.22) and (5.2.23) we get,

$$\frac{a}{\ell} = \frac{4(1 + u_0^2)^2}{4u_0^3 \left[\frac{3}{4u_0^2} + \frac{1}{u_0^2} - \frac{7}{4} - \ell n \frac{1}{u_0} \right]}.$$

From (5.2.22),

$$\frac{\ell}{a} = \frac{4u_0^3}{x_1(1)(1 + u_0^2)^2}. \quad (5.2.24)$$

Since,

$$C_D = 4 \left(\frac{\ell}{a} \right)^2 \phi,$$

from (5.2.21) and (5.2.24) we get,

$$C_D = 4 \left[\frac{4u_0^3}{x_1(1)(1 + u_0^2)^2} \right]^2 \frac{x_1^2(1)}{2} \left[1 + \frac{1}{8} \left(-\frac{15}{4} + \frac{3}{4}u_0^{-4} + \frac{5}{2}u_0^{-2} - \ell n u_0 + \frac{1}{2}u_0^2 \right) \right],$$

i.e.,

$$C_D = \frac{32u_0^6}{(1 + u_0^2)^4} \left[1 + \frac{1}{8} \left(-\frac{15}{4} + \frac{3}{4}u_0^{-4} + \frac{5}{2}u_0^{-2} - \ell n u_0 + \frac{1}{2}u_0^2 \right) \right]. \quad (5.2.25)$$

At $u_0 = 0.3342450$, $C_D = 0.3208516$ to 7 decimal places.

5.3 Numerical Solutions

5.3.1 The state and adjoint equations

The state equations are:

$$\left. \begin{aligned} \dot{x}_1 = f_1 = \frac{dx_1}{dt} &= -u(t), & x_1(0) &= 0.5 \\ \dot{x}_2 = f_2 = \frac{dx_2}{dt} &= \frac{u^3 x_1}{(1 + u^2)}, & x_2(0) &= 0. \end{aligned} \right\} \quad (5.3.1)$$

Using the Runge-Kutte 4th order method for numerical solution of (5.3.1) we get,

$$\begin{aligned} x_{1,n+1} &= x_{1,n} + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ x_{2,n+1} &= x_{2,n} + \frac{1}{6}(v_1 + 2v_2 + 2v_3 + v_4), \end{aligned}$$

where,

$$\begin{aligned}
 k_1 &= hf_1(x_{1,n}, x_{2,n}) = -hu_n \\
 v_1 &= hf_2(x_{1,n}, x_{2,n}) = h \frac{u_n^3 x_{1,n}}{(1 + u_n^2)} \\
 k_2 &= hf_1(x_{1,n} + \frac{1}{2}k_1, x_{2,n} + \frac{1}{2}v_1) = -hu_n \\
 v_2 &= hf_2(x_{1,n} + \frac{1}{2}k_1, x_{2,n} + \frac{1}{2}v_1) = hu_n^3(x_{1,n} + \frac{1}{2}k_1)/(1 + u_n^2) \\
 k_3 &= hf_1(x_{1,n} + \frac{1}{2}k_2, x_{2,n} + \frac{1}{2}v_2) = -hu_n \\
 v_3 &= hf_2(x_{1,n} + \frac{1}{2}k_2, x_{2,n} + \frac{1}{2}v_2) = \frac{hu_n^3(x_{1,n} + \frac{1}{2}k_2)}{(1 + u_n^2)} \\
 k_4 &= hf_1(x_{1,n} + k_3, x_{2,n} + v_3) = -hu_n \\
 v_4 &= hf_2(x_{1,n} + k_3, x_{2,n} + v_3) = \frac{hu_n^3(x_{1,n} + k_3)}{(1 + u_n^2)}.
 \end{aligned}$$

The adjoint equations are:

$$\left. \begin{aligned} \dot{\lambda}_1 &= f_1 = \frac{-\lambda_2 u^3}{(1 + u^2)}, & \lambda_1(1) &= x_1(1) \\ \dot{\lambda}_2 &= f_2 = 0, & \lambda_2(1) &= 1 \end{aligned} \right\} \quad (5.3.2)$$

Using the Runge-Kutta 4th order method for numerical solution of (5.3.2) we get,

$$\begin{aligned}
 \lambda_{1,n} &= \lambda_{1,n+1} + \frac{1}{6}(w_1 + 2w_2 + 2w_3 + w_4) \\
 \lambda_{2,n} &= \lambda_{2,n+1} + \frac{1}{6}(z_1 + 2z_2 + 2z_3 + z_4),
 \end{aligned}$$

where,

$$\begin{aligned}
 w_1 &= -hf_1(\lambda_{1,n+1}, \lambda_{2,n+1}) = -h \left[\frac{-\lambda_{2,n+1} u_{n+1}^3}{(1 + u_{n+1}^2)} \right] \\
 z_1 &= 0 \\
 w_2 &= -hf_1(\lambda_{1,n+1} + \frac{1}{2}w_1, \lambda_{2,n+1} + \frac{1}{2}z_1) \\
 &= -h \left[\frac{-(\lambda_{2,n+1} + \frac{1}{2}z_1) u_{n+1}^3}{(1 + u_{n+1}^2)} \right] = -h \left[\frac{-\lambda_{2,n+1} u_{n+1}^3}{(1 + u_{n+1}^2)} \right] \\
 z_2 &= 0 \\
 w_3 &= -hf_1(\lambda_{1,n+1} + \frac{1}{2}w_2, \lambda_{2,n+1} + \frac{1}{2}z_2) \\
 &= -h \left[\frac{-(\lambda_{2,n+1} + \frac{1}{2}z_2) u_{n+1}^3}{(1 + u_{n+1}^2)} \right] = -h \left[\frac{-\lambda_{2,n+1} u_{n+1}^3}{(1 + u_{n+1}^2)} \right] \\
 z_3 &= 0 \\
 w_4 &= -hf_1(\lambda_{1,n+1} + w_3, \lambda_{2,n+1} + z_3) \\
 &= -h \left[\frac{-(\lambda_{2,n+1} + z_3) u_{n+1}^3}{(1 + u_{n+1}^2)} \right] = -h \left[\frac{-\lambda_{2,n+1} u_{n+1}^3}{(1 + u_{n+1}^2)} \right] \\
 z_4 &= 0.
 \end{aligned}$$

5.4 Results and Discussions

5.4.1 Gradient method

The algorithm for the gradient in function space method applied to problem 2 is the same as that described in Chapter 4, section 4.4.1, where for this problem

$$g = -\lambda_1 + \frac{u^2 \lambda_2 x_1 (3 + u^2)}{(1 + u^2)^2}. \quad (5.4.1)$$

For examining the efficiency of this method when it is applied to the problem in this chapter once again we compare the results obtained numerically, when the critical parameters ϵ , N and u_0 are varied, with the analytical solution. Here, throughout this chapter the results show the minimum number of iterations to achieve the best e_{abs} so that any further iterations produced no improvement. Table (5.4.1) shows, the effect of ϵ and N in achieving the minimum Drag Coefficient (CD), from the best starting control, i.e. $u_0 = 1.0$. The best CD obtained was 0.3208517 to 7 decimal places with $e_{abs} \simeq 1 \times 10^{-7}$ with the corresponding parameter values, $\epsilon = 1.8$, $N = 800$ or 1600 and $m = 30$. Here by selecting a larger steplength factor, up to a limit of 1.8 in this case, we obtain the minimum in fewer iterations, and fewer function evaluations for the best possible e_{abs} . The effect of N is also clear from Table (5.4.1), we can see that a better CD is achieved as N is increased but at the expense of more cpu time. Table (5.4.2), shows that effect of initial control u_0 on the minimum CD, when a sufficiently large enough N is taken as 800. As can be seen the best CD with the best e_{abs} was achieved at $u_0 = 1.0$, also taking u_0 in the range $0 \leq u_0 \leq 0.3$ could achieve $e_{abs} \leq 4 \times 10^{-7}$, which performed better in terms of convergence to the minimum and number of iterations taken, than selecting u_0 as 0.4, 0.5 or 1.1. Figure (5.4.1) and figure (5.4.2), show the curves from the radius of body $x_1(t)$ and local slope $u(t)$ respectively against distance along body axis, for the analytical solutions and the numerical ones, with $m = 1$ and $m = 30$, using $N = 800$, $u_0 = 1.0$ and $\epsilon = 1.8$. Hence from the graph of x_1 we can see that when $m = 30$, the curve behaves similar to the analytical solution, but for $m = 1$, the difference is quite obvious. Also from the graph of u , we can see that with $m = 30$, the behaviour of the curve is similar to the analytical one, where as when $m = 1$, the difference is quite eminent.

Figure (5.4.3) compares the analytical curve for the radius of body $x_1(t)$ with the numerical solutions, when the number of integration steps are 100 and 800, respectively after 2 iterations, with $u_0 = 1.0$ and $\epsilon = 1.8$. Figure (5.4.4) gives similar results from the local slope $u(t)$. Here from both the graphs of x_1 and u , we can not see much difference in the behaviour of the numerical curves, between $N = 100$ and $N = 800$.

Figures (5.4.5) and (5.4.6) demonstrate the effect on the radius of body $x_1(t)$ and the local slope $u(t)$ respectively when using step length factors $\epsilon = 1.0$ and 1.8 after 2 iterations, when $u_0 = 1.0$ and $N = 800$. As can be seen from the graph of x_1 , taking $\epsilon = 1.8$ can lead to the behaviour of the curve closer to the analytical one than taking $\epsilon = 1.0$.

But from the graph of u , we can not see much difference in the curvature behaviour of the numerical solutions, between $\epsilon = 1.8$ and $\epsilon = 1.0$. Figure

(5.3.7), shows how the Drag Coefficient varies with m for $u_0 = 1.0, N = 800$ and $\epsilon = 1.8$.

Figure (5.4.8), shows the effect on the Drag Coefficient (CD) of changing u_0 for different number of iterations ($m = 0.1$ and 30) with $N = 800$ and $\epsilon = 1.8$. Here we can see that as m increases, for all the starting controls, the value of CD converges to the optimal one. Figure (5.4.9), shows the graph of error, when the optimal numerical results of distance along body axis against radius of body $x_1(t)$ are compared with the analytical solution. Note that here $x_1N(t)$ and $x_1A(t)$ denote the numerical solutions and analytical solutions of $x_1(t)$, respectively. Similarly figure (5.4.10), shows the graph of error of distance along body axis against the local slope $u(t)$ and also here $UN(t)$ and $UA(t)$ denote that numerical solutions and analytical solutions of $u(t)$ respectively. The parameters for both figures are $m = 30, \epsilon = 1.8, N = 800$ and $u_0 = 1.0$.

The results above show that some improvement can be obtained in minimizing the Drag (CD), by varying the critical parameters u_0, ϵ and N .

The correct choice of initial control is an important factor in achieving the best CD in fewer iterations and therefore less computing time and fewer number of function evaluations. Considering the interaction of the other two critical factors ϵ and N , the best combination exists with larger ϵ and larger N , in this case $\epsilon = 1.8$ and $N = 800$, but if we choose smaller values for N we can compensate by selecting larger values for ϵ and vice versa. But we should note that if the value of ϵ is taken too large, i.e. in this case $\epsilon > 1.8$, while N is large enough for accuracy purposes, we get no more improvement on the minimum yield CD. In view of the above comments, by selecting proper initial control and accurate enough integration steps, along with an appropriate step length factor a speed up in the convergency to a better minimum for CD is obtained.

5.4.2 Steepest descent

The algorithm for steepest descent applied to the problem 2 and also the line search techniques used for this method are described in Chapter 2, section 2.2.2 and chapter 3, section 3.4.1 respectively. The gradient (g) is obtained in the same way as for gradient method in function space in this chapter (see section (5.4.1)).

Here we again investigate the effect of selecting step length factor, integration step and initial control in the solution of problem 2.

Table (5.4.3), shows the effect of ϵ and N in obtaining the minimum CD with the best starting control $u_0 = 1.0$. The best minimum achieved was 0.3208526 to 7 decimal places with $e_{abs} \simeq 1 \times 10^{-6}$, $\epsilon = 2.0$ and $N = 800$ or $N = 1600$ after 243 iterations. Hence generally for larger N , selecting a larger step length factor (ϵ) speeds up the search from the minimum CD, in terms of computing time, reduction in number of iterations and also function evaluations, for a required e_{abs} . Table (5.4.4), shows how the starting control affects the minimum CD, i.e. by selecting u_0 as 1, 1.1 and 0.4 we achieve better minimum CD respectively compared with 0, 0.3 and 0.5. Here N was taken as 800.

Figure (5.4.11) and Figure (5.4.12), show the graphs of distance along

body axis against the radius of body $x_1(t)$ and the local slope $u(t)$ respectively, when the numerical solutions are compared with the analytical one, when $m = 1$ and $m = 243$, $N = 800$, $u_0 = 1.0$ and $\epsilon = 2.0$. As can be seen from the graph of x_1 taking $m = 243$ can produce solutions closer to the analytical one than $m = 1$. The same comment is also true for the graph of control.

Figure (5.4.13) and Figure (5.4.14) compare the analytical solutions for radius of body axis $x_1(t)$ and local slope $u(t)$ respectively with the numerical ones, with $N = 100$ and 800 respectively, with $m = 5$, $u_0 = 1.0$ and $\epsilon = 2.0$. Here as can be seen from the graphs of x_1 and u , there is not much difference in the behaviour of the curves between the numerical solutions of $N = 100$ and $N = 800$. figures (5.4.15) and (5.4.16), show the effect on radius of body $x_1(t)$ and local slope $u(t)$ respectively of using step length factors $\epsilon = 1.0$ and $\epsilon = 2.0$ from 5 iterations, when $u_0 = 1.0$ and $N = 800$. Here from the graph of x_1 we can see that the curve of $\epsilon = 2.0$, performs slightly better than $\epsilon = 1.0$, in being closer to the curve of analytical solution. But not much difference could be seen graphically, between $\epsilon = 1.0$ and 2.0 for the control.

Figure (5.4.17), shows how the Drag Coefficient (CD) varies with m , when $u_0 = 1.0$, $N = 800$ and $\epsilon = 2.0$. Figure (5.4.18), demonstrates the effect on Drag Coefficient of changing initial control for different number of iterations ($m = 0, 5$ and 243), with $N = 800$ and $\epsilon = 2.0$. Here as can be seen from the graph as m increases, the value of CD gets close to the optimal one. Figure (5.4.19) and Figure (5.4.20), show the graph of errors when the best numerical results of distance along body axis against radius of body $x_1(t)$ and local slope $u(t)$ respectively are compared with the analytical ones, where the parameters from numerical solutions are $u_0 = 1.0$, $N = 800$, $\epsilon = 2.0$ and $m = 243$.

From the results obtained we can see that even when N is large enough, i.e. 800 selecting ϵ too large say 2.1 does not lead to a better minimum value for CD.

Once again the results show that a proper choice of initial control is a factor in achieving the best minimum CD in fewer iterations, and also number of function evaluations and less computing time.

When the interaction between ϵ and N is examined, where N is not large enough, i.e. 100 we can not establish a suitable pattern between ϵ and N to justify the convergence from the minimum CD. The conclusions reached for the GFS on selecting the best combinations of u_0 , ϵ and N is also applicable for SD for this particular problem.

5.4.3 Fletcher-Reeves

The algorithm for Fletcher-Reeves method applied to problem 2 is as described in Chapter 2 Section 2.4. Also the line search technique used for this method is linear search at constant step (see Chapter 3 Section 3.4.1). The norm of each gradient trajectory are calculated as described in Chapter 2, Section 2.10.1. The gradient (g) is obtained in the same way as for GFS in this Chapter Section 5.4.1.

Table (5.4.5) shows the effect of ϵ and N in achieving the minimum Drag Coefficient (CD) with the best starting control $u_0 = 0.4$. The best CD

was 0.3208517 to 7 decimal places with $e_{abs} \simeq 1 \times 10^{-7}$. the corresponding parameter values were $\epsilon = 1.0$, $N = 800$ or $N = 1600$ and $m = 15$.

Again the selection of a larger N gives th minimum CD in fewer iterations and also number of function evaluations, and finally less computing time for a required e_{abs} . Also for larger N say e.g. 800 selecting larger ϵ up to a limit in this case less then 1.1 can guarantee a better convergence to the minimum value for CD.

Table (5.4.6) shows the effect of selecting different initial control u_0 , when N was taken as 800. Here taking $u_0 = 0.4$, leads to the best optimal value for CD, with the best $e_{abs} = 1.0 \times 10^{-7}$. But if less accuracy of e_{abs} is required, i.e. $\leq 6 \times 10^{-7}$, then selecting u_0 as 0 or 1.0 can get to a reasonable minimum value for CD, in slightly fewer iterations than others.

Figure (5.4.21) and Figure (5.4.22), show the graph of radius of body $x_1(t)$ and local slope $u(t)$ respectively, along the body axis, comparing the analytical curve with the numerical ones, when $m = 1$ and $m = 15$, with $\epsilon = 1.0$, $u = 0.4$ and $N = 800$. As can be seen from the graphs of x_1 and u , taking $m = 15$, lead to the behaviour of the curves closer to the analytical solutions than taking $m = 1$.

Figures (5.4.23) and (5.4.24), show the comparison of the analytical solution with the numerical ones for radius of body $x_1(t)$ and local slope $u(t)$ respectively with $N = 100$ and 800, with $u_0 = 0.4$, $m = 2$ and $\epsilon = 1.0$. Graphically we can not see much difference in the behaviour of the numerical curves, between $N = 100$ and $N = 800$, for both the graphs of x_1 and u . Figures (5.4.25) and (5.4.26) demonstrate the curves of distance along body axis against the radius of body $x_1(t)$, and also local slope $u(t)$, when we compare the analytical solutions with the numerical ones, where the step length factors are $\epsilon = 0.5$ and $\epsilon = 1.0$ for $m = 2$, $u_0 = 0.4$ and $N = 800$.

Again as can be seen from the graphs of x_1 and u , not much difference can be seen in the behaviour of the curves, between $\epsilon = 0.5$ and $\epsilon = 1.0$.

Figure (5.4.27), shows how the minimum Drag CD varies with m , when $u_0 = 0.4$, $N = 800$ and $\epsilon = 1.0$.

Figure (5.4.28), demonstrates the effect on CD of changing the initial control for different number of iterations ($m = 0, 1$ and 15), with $N = 800$ and $\epsilon = 1.0$. As can be seen from the graph by increasing m , the value of CD for all the starting controls gets closer to the optimal one.

Figures (5.4.29) and (5.4.30), show the curves of errors, (i.e., $(-\ln(x_1 N(t) - x_1 A(t)))$ and $-\ln(UN(t) - UA(t))$), when the best numerical solutions of distance along body axis against radius of body $x_1(t)$ and local slope $u(t)$ respectively are compared with the analytical solutions, where the parameters for numerical solutions are $u_0 = 0.4$, $N = 800$, $\epsilon = 1.0$ and $m = 15$.

In view of the above results, again for N large enough, i.e., 800, selecting ϵ too large say 1.1 does not give a better minimum for CD. Here also for smaller values of N say 100, we can see that, taking ϵ , relatively smaller say 0.5, can produce minimum CD with better e_{abs} , than larger ϵ 's. Also by selecting a proper initial control u_0 a better convergence for the minimum Drag Coefficient (DC) can be achieved, providing N and ϵ are sufficiently large enough.

5.4.4 Polak-Ribière

The algorithm for the Polak-Ribière method is described in chapter 2, section 2.5. The line search technique and the calculation of the norms are as described for FR.

Table (5.4.7), shows the effect of ϵ and N on the minimum Drag (CD), with $u_0 = 0.0$ which is the best starting control. The best yield was 0.3208522 to 7 decimal places, with $e_{abs} \simeq 6 \times 10^{-7}$ and corresponding parameter values of 0.9, $N = 800$ or $N = 1600$ and $m = 15$. By selecting larger N the minimum value for CD can be obtained in fewer iterations and also fewer number of function evaluations and consequently less computing time. For larger N , e.g., 800, selecting larger ϵ up to a limit in this case less than 1.0 can ensure a better convergence for minimum CD.

Table (5.4.8), shows that selection of a proper initial control would help in finding better convergence to the minimum CD. Here N is taken sufficiently large enough i.e., 800. Although for $u_0 = 0.5$ and 1.0, we can find a reasonable minimum for CD close to the optimal one, in fewer number of iterations than $u_0 = 0$ or 0.3, but as we increase the number of iterations for $u_0 = 0.5$ and 1.0, we get no more improvement in convergency, where as with $u_0 = 0$ or 0.3 for a few more iterations we obtain a better minimum CD with a better e_{abs} .

Figures (5.3.31) and (5.4.32) show the graphs of distance along body axis against the radius of body $x_1(t)$ and the local slope $u(t)$ respectively, comparing the analytical solutions with the numerical ones, with $m = 1$ and $m = 15$, $\epsilon = 0.9$, $u_0 = 0.0$ and $N = 800$. As can be seen from both the graph of x_1 and u , the behaviour of the curves, with $m = 15$ are closer to the analytical one than when $m = 1$.

Figures (5.4.33) and (5.4.34), show the curves of distance along body axis, against the radius of body $x_1(t)$ and also the local slope $u(t)$ respectively, when the analytical solutions are compared with the numerical ones, where $N = 100$ and $N = 800$ for $m = 2$, $\epsilon = 0.9$ and $u_0 = 0.0$. Here both for the graphs of x_1 and u we can not see much difference, between the behaviour of the curves, with $N = 100$ and $N = 800$. Figures (5.4.35) and (5.4.36), show similar graphs for radius of body $x_1(t)$ and local slope $u(t)$ with $\epsilon = 0.5$ and $\epsilon = 0.9$, with $m = 2$, $N = 800$ and $u_0 = 0.0$. As can be seen for both the graphs of x_1 and u the behaviour of the curves with $\epsilon = 0.5$ and 0.9 are similar and not much difference can be observed. Figure (5.4.37) shows how varying m affects the minimum CD, when $u_0 = 0.0$, $N = 800$ and $\epsilon = 0.9$.

Figure (5.4.38), demonstrates how the change in u_0 can affect the Drag Coefficient (CD) for different number of iterations ($m = 0, 1$ and 15), with $N = 800$ and $\epsilon = 0.9$. Figures (5.4.39) and (5.4.40) show the curves of error, when the best numerical solutions of distance along body axis, against radius of body $x_1(t)$ and local slope $u(t)$ respectively are compared with the analytical solutions, i.e. for the parameters, $u_0 = 0.0$, $m = 15$, $N = 800$ and $\epsilon = 0.9$.

From the results obtained here for sufficiently large N and ϵ large enough, we can find a better minimum for CD. But we should be aware, that selecting ϵ too large, does not improve the minimum. Also proper choice of initial control is an important factor in achieving the optimal CD for a more accurate e_{abs} .

5.4.5 Hybrid 1

The algorithm for the Hybrid 1 method is described in Chapter 2, section 2.6.1. The line search technique used and calculation of the norms are the same as FR and PR in this chapter.

Table (5.4.9), shows the effect of ϵ and N in achieving the minimum CD, with the best initial control $u_0 = 0.4$. The best yield was 0.3208520 to 7 decimal places, with $e_{abs} \simeq 4.0 \times 10^{-7}$ and corresponding parameter values, $\epsilon = 1.0, N = 800$ or $N = 1600$ and $m = 15$. Here also for larger n , e.g., 800 selecting larger ϵ up to a limit, in this case less than 1.0, can guarantee a better convergence for minimum Drag Coefficient (CD), with fewer number of iterations and number of function evaluations, and also less computing time.

Table (5.4.10), shows the effect of selecting different starting control on convergence of CD. Here, although for $u_0 = 0.5$ and $u_0 = 0$, we can find a reasonably good minimum for Drag Coefficient (CD) in fewer iterations than $u_0 = 0.4$, but as the number of iterations are increased for $u_0 = 0.5$ and 0, there is no improvement in the convergency, where as with $u_0 = 0.4$, for a few more iterations we obtain better minimum yield for Drag Coefficient (CD) with a better minimum yield for Drag Coefficient (CD) with a better e_{abs} .

Figure (5.4.41) and (5.4.42), show the graphs of distance along body axis against radius of body $x_1(t)$ and local slope $u(t)$ respectively and compare the analytical solutions with the numerical ones, with $m = 1$ and $m = 1.5$, where $\epsilon = 1.0, N = 800$ and $u_0 = 0.4$. Here we can see from both the graphs of x_1 and in that taking $m = 15$ produces results closer to the analytical ones than $m = 1$.

Figures (5.4.43) and (5.4.44), demonstrate the effect of N on the radius of body $x_1(t)$ and the local slope $u(t)$ respectively, when the analytical solutions are compared with the numerical ones, with $N = 100$ and $N = 800$ for $m = 2, \epsilon = 0.9$ and $u_0 = 0.4$. As can be seen from both the graphs of x_1 and u , there is not much difference between the behaviour of the curves of $N = 100$ and $N = 800$. Figure (5.4.45) and Figure (5.4.46), show similarly the effect of ϵ on the radius of body $x_1(t)$ and local slope $u(t)$ respectively, with $\epsilon = 0.5$ and $\epsilon = 1.0$ for $m = 2, N = 800$ and $u_0 = 0.4$.

Here also not much difference can be observed from both the curves of x_1 and u , between $\epsilon = 0.5$ and $\epsilon = 1.0$. Figure (5.4.47), shows how the minimum CD varies with m , when $u_0 = 0.4, N = 800$ and $\epsilon = 1.0$.

Figure (5.4.48), shows how the change in u_0 affects the minimum CD with $m = 0, 1$ and 15, where $N = 800$ and $\epsilon = 0.9$. Here also we can see that as m increases, the value of CD for all the starting controls get closer to the optimal one.

Figures (5.4.49) and (5.4.50), show the curves of error, when the best numerical solutions of distance along body axis, against radius of body $x_1(t)$ and local slope $u(t)$ respectively are compared with the analytical solution, i.e. for the parameters $u_0 = 0.4, N = 800, \epsilon = 1.0$ and $m = 15$.

In view of the above results, here also for sufficiently large enough N and ϵ large enough we find a better minimum for CD. Also like PR taking ϵ too large does not improve the convergency. Finally, a proper selection of initial control can help in finding a better minimum for CD.

5.4.6 Angle test hybrid

The algorithm for angle test hybrid method is described in Chapter 2, section 2.7. The line search and calculation of the norms are the same as for H1. For this method we had to consider the parameter $\tau > 0$ as well. The method was tested in the same way as the previous methods, plus the new parameter τ . Table (5.4.11), shows the effect of ϵ and N on finding the minimum Drag (CD), with the best starting control $u_0 = 1.0$ and $\tau = 0.000001$. The best minimum for CD was 0.3208527 to 7 decimal places after 12 iterations with $e_{abs} \simeq 1.1 \times 10^{-6}$ and corresponding parameter values, $\epsilon = 1.5, N = 800$ or $N = 1600$ and $m = 12$. Clearly for larger N and larger ϵ up to a limit of 1.5, the minimum Drag Coefficient can be obtained in fewer iterations and therefore fewer number of function evaluations and also less computing time. This relation between ϵ and N could also be held for smaller values of N . Table (5.4.12), can also reveal the effect of selecting u_0 in achieving a better minimum for Drag (CD), with $N = 800$ and $\tau = 0.000001$. By choosing $u_0 = 0.5$ and 1.0 , we can obtain a better minimum for CD with $e_{abs} = 1.2 \times 10^{-6}$ and $e_{abs} = 1.1 \times 10^{-6}$ respectively, in fewer iterations and number of function evaluations compared with $u_0 = 0, 0.3$ and 0.4 .

The value of τ was also tested with 0.01, 0.0001 and 0.000001, but the effect of it on obtaining the minimum CD was practically negligible.

Figures (5.4.51) and (5.4.52), show the graphs of distance along body axis against radius of body $x_1(t)$ and local slope $u(t)$ respectively, comparing the analytical solutions with the numerical ones, with $m = 1$ and $m = 12$, where $\epsilon = 1.5, N = 800, \tau = 0.000001$ and $u_0 = 1.0$. As can be seen from both the graphs of x_1 and u , with $m = 12$, the behaviour of the curves are closer to the analytical one compared with $m = 1$.

Figures (5.4.53) and (5.4.54) give the graphs of distance along body axis against radius of body $x_1(t)$ and local slope $u(t)$ respectively, showing the effect of N taken as 100 and 800, for $m = 2$, with $u_0 = 1.0, \tau = 0.000001$ and $\epsilon = 1.5$. Here we can not observe much difference both from the graphs of x_1 and u , when $N = 100$ is compared with $N = 800$ and graphically they show similar patterns.

Also Figures (5.4.55) and (5.4.56), in the same way demonstrate the effect of ϵ on radius of body axis $x_1(t)$ and local slope $u(t)$ respectively, with $\epsilon = 0.5$ and 1.5 and corresponding parameter values, $N = 800, \tau = 0.000001, u_0 = 1.0$ and $m = 2$. As can be seen from both the graphs of x_1 and u , there is not much difference in the behaviour of the curves between $\epsilon = 0.5$ and $\epsilon = 1.5$.

Figure (5.4.57), demonstrates the effect of m on the minimum CD with $u_0 = 1.0, \epsilon = 1.5, \tau = 0.000001$ and $N = 800$.

Figure (5.4.58), shows the change in u_0 against the minimum yield CD with $m = 0, 1$ and 12 , with $\epsilon = 1.5, N = 800$ and $\tau = 0.000001$. As can be seen, by increasing m , the values of CD for all the starting controls get closer to the optimal CD.

Figures (5.4.59) and (5.4.60), show the curves of errors when the best numerical solutions of distance along body axis against radius of body $x_1(t)$ and local slope $u(t)$ respectively are compared with the analytical solution, i.e. for the parameters, $u_0 = 1.0, N = 800, \epsilon = 1.5, m = 15$ and $\tau = 0.000001$.

From the above results we can see that taking ϵ in the range $1.0 \leq \epsilon \leq 1.5$, along with sufficiently large enough N can produce consistent results.

Also a proper choice of initial control will help to speed up the process of convergency.

5.4.7 Hybrid 3

The algorithm for Hybrid 3 method can be found in Chapter 2 Section 2.8. The calculations of the norms and line search are the same as for ATH. The effect of the new parameters, $\lambda > 0$ and $\mu < \frac{1}{2}$ had to be considered from this method.

Table (5.4.13), shows the effect of ϵ and N in achieving the minimum CD, with the best starting control $u_0 = 0.4, \mu = 0.45$ and $\lambda = 0.0001$. The best minimum for CD was 0.3208518 to 7 decimal places, with $e_{abs} \simeq 2.0 \times 10^{-7}$, $\epsilon = 1.0$ and $N = 800$ or $N = 1600$ after 15 iterations. As in previous method selection of a larger N gives the minimum yield in fewer iterations and also fewer number of function evaluations. For larger N , e.g., 800 selecting larger ϵ up to a limit in this case less than 1.1 can guarantee a better convergence for minimum Drag Coefficient (CD), with lower number of iterations and number of function evaluations.

Table (5.4.14), shows the effect of selecting different initial controls u_0 with $N = 800, \lambda = 0.0001$ and $\mu = 0.45$. Here although for $u_0 = 0, 0.3$ and 0.5 or 1.0 we can find a minimum for CD in fewer iterations than $u_0 = 0.4$, but as the number of iterations are increased for those u_0 's, there is no improvement in the convergence, where as with $u_0 = 0.4$ for a few more iterations we obtain better minimum for CD with more accurate e_{abs} .

The values of λ with 0.01, 0.001 and 0.0001, also μ with 0.1, 0.25 and 0.45 were tested on this problem and on minimizing the CD had practically no significant importance.

Figure (5.4.61) and Figure (5.4.62), compare the analytical solutions with the numerical ones for radius of body axis $x_1(t)$ and local slope $u(t)$ respectively with $m = 1$ and $m = 15$ and corresponding parameter values, $\lambda = 0.0001, \mu = 0.45, \epsilon = 1.0, N = 800$ and $u_0 = 0.4$. Here for both the graphs of x_1 and u , we can see, from the behaviour of the curves that when, $m = 15$ they get closer to the optimal curves than when $m = 1$.

Figures (5.4.63) and (5.4.64), show that effect of N for $N = 100$ and 800 , with $m = 2, u_0 = 0.4, \lambda = 0.0001$ and $\mu = 0.45$. As can be seen from both the graphs of x_1 and u , not much difference exist, between the curves of $N = 100$ and $N = 800$.

Figures (5.4.65) and Figure (5.4.66), similarly show the effect of ϵ 's for $\epsilon = 0.5$ and 1.0 , with $N = 800, u_0 = 0.4, m = 2, \lambda = 0.0001$ and $\mu = 0.45$.

Here also the curvature for both the graphs of x_1 and u are similar with $\epsilon = 0.5$ and $\epsilon = 1.0$.

Figure (5.4.67), shows how the minimum Drag (CD) reacts with different number of iterations, with $u_0 = 0.4, N = 800, \lambda = 0.0001, \mu = 0.45$ and $\epsilon = 1.0$.

Figure (5.4.68), demonstrates how the change in u_0 can affect CD for different number of iteration ($m = 0, 1$ and 8), with $N = 800, \epsilon = 1.0, \lambda = 0.0001$ and $\mu = 0.45$. It is clear graphically, as m increases the values of CD for all starting controls, get closer to the optimal one. Figure (5.4.69) and

Figure (5.4.70), show the curves of errors when the best numerical solutions of distance along body axis against radius of body $x_1(t)$ and local slope $u(t)$ respectively are compared with the analytical solutions, i.e. for the parameters, $u_0 = 0.4, \mu = 0.45, \lambda = 0.0001, \epsilon = 1.0, m = 15$ and $N = 800$.

Here from the above results, we can see that with N sufficiently large enough and ϵ in the range $0.9 \leq \epsilon \leq 1.0$ we can find better minimum CD in fewer number of iterations and also number of function evaluations, than selecting ϵ too small or too large or N too small. Here also a proper choice of initial control is an important factor in finding the optimal minimum CD.

5.5 Summary of the Results

Referring to the summary Table (5.5.1), with $N = 800, \mu = 0.45, \tau = 0.000001, \lambda = 0.0001$, also Figure (5.5.1) with the above parameters, plus the best u_0 for each method, the following results can be seen. The best performance in achieving the minimum CD, i.e. the most accurate e_{abs} , the methods performed as follows in terms of preferences;

At $u_0 = 0.0$, GFS, then H1 and PR performed the same, then FR, H3, ATH and finally SD.

At $u_0 = 0.3$, GFS, H1, then PR and FR performed the same, then H3, ATH and finally SD.

At $u_0 = 0.4$, FR, H3, H1, GFS, PR, ATH and finally SD.

At $u_0 = 0.5$, PR, ATH, then H1 and H3 performed the same, then FR and GFS performed the same and finally SD.

At $u_0 = 1.0$, GFS, FR, H1, SD and finally PR, ATH and H3 performed the same.

But taking into considerations the number of iterations taken, as well as a reasonable accuracy of e_{abs} say $\leq 2.0 \times 10^{-6}$, the methods performed as follows in terms of preferences;

At $u_0 = 0.0$, ATH, then H1, then H3 and FR performed the same, then, PR, GFS and finally SD.

At $u_0 = 0.3$, ATH and H3 performed the same, then, PR, then H1 and FR performed the same, the GFS and finally SD.

At $u_0 = 0.4$, ATH, then FR, H3 and H1 performed the same, then PR, GFS and finally SD.

At $u_0 = 0.5$, PR, ATH, H1 and H3 performed the same, then FR, GFS and finally SD.

At $u_0 = 1.0$, PR and ATH performed the same, then FR and H3 performed the same, the H1, GFS and finally SD.

5.6 Conclusion

Applying the seven methods of optimization to Problem 2, we can see that each technique produced its own best minimum value for CD, with different initial control, and step length factor. But one factor that was common among all of them was the choice of integration step number (N), i.e., by selecting larger N we could guarantee a better optimal result for CD in all

the methods. Although for larger N we could find an interaction between ϵ and N , i.e., by selecting larger ϵ we could achieve better minimum for CD, but we could not establish that for smaller integration step numbers.

Also the fact that overall for this problem the best minimum value for CD was obtained by GFS, but when we taken into consideration the number of iterations taken it did not seem to be the ideal method. Thus if the number of iterations as well as the convergency is going to be considered, when the best results are obtained for each method, with the best possible initial control, step length factor and integration step, the best performance may be selected in the following order;

FR, H3, H1, ATH, PR, GFS and finally SD.

TABLE (5.4.1):Results of GFS with varying ϵ and N.

N ϵ	100	400	800	1600
1.0	CD=.3209794 m=60 eabs= 1.3×10^{-4} Cputime<1 NFE=61	CD=.3208590 m=60 eabs= 7.4×10^{-6} Cputime<1 NFE=61	CD=.3208530 m=60 eabs= 1.4×10^{-6} Cputime=1 NFE=61	CD=.3208530 m=59 eabs= 1.4×10^{-6} Cputime=2 NFE=60
1.7	CD=.3209803 m=50 eabs= 1.3×10^{-4} Cputime<1 NFE=51	CD=.3208592 m=50 eabs= 7.6×10^{-6} Cputime<1 NFE=51	CD=.3208530 m=50 eabs= 1.4×10^{-6} Cputime=1 NFE=51	CD=.3208530 m=50 eabs= 1.4×10^{-6} Cputime=2 NFE=51
1.8	CD=.3209701 m=30 eabs= 1.2×10^{-4} Cputime<1 NFE=31	CD=.3208554 m=30 eabs= 3.8×10^{-6} Cputime<1 NFE=31	CD=.3208517 m=30 eabs= 1×10^{-7} Cputime=1 NFE=31	CD=.3208517 m=30 eabs= 1×10^{-7} Cputime=2 NFE=31
1.9	CD=.3237455 m=40 eabs= 2.9×10^{-3} Cputime<1 NFE=41	CD=.3219327 m=40 eabs= 1.1×10^{-3} Cputime<1 NFE=41	CD=.3217205 m=40 eabs= 8.7×10^{-4} Cputime=1 NFE=41	CD=.3217203 m=40 eabs= 8.7×10^{-4} Cputime=2 NFE=41
2.5	CD=.3258120 m=50 eabs= 5.0×10^{-3} Cputime<1 NFE=51	CD=.3221348 m=50 eabs= 1.3×10^{-3} Cputime<1 NFE=51	CD=.3219351 m=50 eabs= 1.1×10^{-3} Cputime=1 NFE=51	CD=.3219351 m=50 eabs= 1.1×10^{-3} Cputime=2 NFE=51

TABLE (5.4.2):Results of GFS with change in u_0 .

u_0	m	CD	e_{abs}	Cputime	eps	NFE
0	40	0.3208520	4×10^{-7}	1	1.8	41
0.3	50	0.3208519	3×10^{-7}	1	1.8	51
0.4	60	0.3208531	1.5×10^{-6}	1	1.8	61
0.5	70	0.3208531	1.5×10^{-6}	1	1.7	71
1.0	30	0.3208517	1×10^{-7}	1	1.8	31
1.1	60	0.3208531	1.5×10^{-6}	1	1.7	61

TABLE (5.4.3):Results of SD with varying ε and N.

N ε	100	400	800	1600
1.0	CD=.3209782 m=400 eabs= 1.27×10^{-4} Cputime=3 NFE=1614	CD=.32085683 m=400 eabs= 1.67×10^{-5} Cputime=12 NFE=1616	CD=.3208537 m=400 eabs= 2.1×10^{-6} Cputime=24 NFE=1626	CD=.3208535 m=400 eabs= 2.1×10^{-6} Cputime=43 NFE=1636
1.2	CD=.3209888 m=400 eabs= 1.37×10^{-4} Cputime=3 NFE=1611	CD=.3208679 m=400 eabs= 1.63×10^{-5} Cputime=12 NFE=1614	CD=.3208533 m=400 eabs= 1.7×10^{-6} Cputime=23 NFE=1619	CD=.3208533 m=400 eabs= 1.7×10^{-6} Cputime=43 NFE=1633
1.8	CD=.3209875 m=400 eabs= 1.36×10^{-4} Cputime=3 NFE=1608	CD=.3208623 m=300 eabs= 1.07×10^{-5} Cputime=12 NFE=1223	CD=.3208531 m=300 eabs= 1.5×10^{-6} Cputime=18 NFE=1232	CD=.3208530 m=300 eabs= 1.4×10^{-6} Cputime=36 NFE=1241
1.9	CD=.3209805 m=400 eabs= 1.29×10^{-4} Cputime=3 NFE=1604	CD=.3208612 m=300 eabs= 9.6×10^{-6} Cputime=8 NFE=1207	CD=.3208530 m=300 eabs= 1.4×10^{-6} Cputime=18 NFE=1213	CD=.3208530 m=300 eabs= 1.4×10^{-6} Cputime=36 NFE=1218
2.0	CD=.3209853 m=400 eabs= 1.3×10^{-4} Cputime=3 NFE=1604	CD=.3208593 m=270 eabs= 7.7×10^{-6} Cputime=7 NFE=1088	CD=.3208526 m=243 eabs= 1×10^{-6} Cputime=15 NFE=987	CD=.3208526 m=243 eabs= 1×10^{-6} Cputime=29 NFE=992
2.1	CD=.3209846 m=400 eabs= 1.33×10^{-4} Cputime=3 NFE=1604	CD=.3208715 m=300 eabs= 1.99×10^{-5} Cputime=8 NFE=1204	CD=.3208545 m=300 eabs= 2.9×10^{-6} Cputime=15 NFE=1206	CD=.3208545 m=300 eabs= 2.9×10^{-6} Cputime=29 NFE=1206

TABLE (5.4.4):Results of SD with change in u_0 .

u_0	m	CD	e_{abs}	Cputime	eps	NFE
0	400	0.3208534	1.8×10^{-6}	24	1.9	1624
0.3	400	0.3208534	1.8×10^{-6}	24	1.9	1624
0.4	300	0.3208535	1.9×10^{-6}	18	1.9	1222
0.5	400	0.3208535	1.9×10^{-6}	24	2.0	1622
1.0	243	0.3208526	1×10^{-6}	15	2.0	987
1.1	300	0.3208534	1.8×10^{-6}	16	2.0	1221

TABLE (5.4.5):Results of FR with varying ϵ and N.

N ϵ	100	400	800	1600
0.5	CD=.3210410 m=25 eabs= 1.89×10^{-4} Cputime<1 NFE=125	CD=.3208729 m=25 eabs= 2.13×10^{-5} Cputime<1 NFE=117	CD=.3208575 m=20 eabs= 5.9×10^{-6} Cputime=2 NFE=121	CD=.3208575 m=20 eabs= 5.7×10^{-6} Cputime=4 NFE=123
0.9	CD=.3213943 m=25 eabs= 5.43×10^{-4} Cputime<1 NFE=110	CD=.3208692 m=25 eabs= 1.76×10^{-5} Cputime=1 NFE=114	CD=.3208550 m=18 eabs= 3.4×10^{-6} Cputime=2 NFE=89	CD=.3208550 m=17 eabs= 3.4×10^{-6} Cputime=4 NFE=91
1.0	CD=.3211833 m=25 eabs= 3.32×10^{-4} Cputime<1 NFE=111	CD=.3209054 m=25 eabs= 5.38×10^{-6} Cputime=1 NFE=109	CD=.3208517 m=15 eabs= 1×10^{-7} Cputime=2 NFE=70	CD=.3208517 m=15 eabs= 1×10^{-7} Cputime=4 NFE=74
1.1	CD=.3218825 m=25 eabs= 1.03×10^{-3} Cputime<1 NFE=105	CD=.3208918 m=25 eabs= 4.02×10^{-5} Cputime=1 NFE=110	CD=.3208541 m=18 eabs= 2.5×10^{-6} Cputime=2 NFE=82	CD=.3208539 m=18 eabs= 2.3×10^{-6} Cputime=4 NFE=85
1.5	CD=.3219732 m=25 eabs= 1.12×10^{-3} Cputime<1 NFE=113	CD=.3209241 m=25 eabs= 7.25×10^{-5} Cputime=1 NFE=115	CD=.3208935 m=20 eabs= 4.19×10^{-5} Cputime=2 NFE=123	CD=.3208935 m=19 eabs= 4.19×10^{-5} Cputime=4 NFE=126

TABLE (5.4.6):Results of FR with change in u_0 .

u_0	m	CD	e_{abs}	Cputime	eps	NFE
0	14	0.3208525	9×10^{-7}	1	1.5	60
0.3	17	0.3208522	6×10^{-7}	2	1.0	81
0.4	15	0.3208517	1×10^{-7}	2	1.0	70
0.5	15	0.3208531	1.5×10^{-6}	2	1.2	71
1.0	14	0.3208522	6×10^{-7}	1	1.3	65

TABLE (5.4.7):Results of PR with varying ϵ and N.

N ϵ	100	400	800	1600
0.5	CD=.3209071 m=25 eabs= 1.20×10^{-4} Cputime<1 NFE=146	CD=.3208672 m=25 eabs= 1.56×10^{-5} Cputime=1 NFE=160	CD=.3208660 m=20 eabs= 1.44×10^{-5} Cputime=2 NFE=124	CD=.3208659 m=20 eabs= 1.43×10^{-5} Cputime=4 NFE=128
0.8	CD=.3209666 m=25 eabs= 1.15×10^{-4} Cputime<1 NFE=112	CD=.3208636 m=25 eabs= 1.2×10^{-5} Cputime=1 NFE=114	CD=.3208547 m=18 eabs= 3.1×10^{-6} Cputime=2 NFE=95	CD=.3208547 m=17 eabs= 3.1×10^{-6} Cputime=4 NFE=101
0.9	CD=.3209259 m=25 eabs= 7.43×10^{-5} Cputime<1 NFE=109	CD=.3208561 m=25 eabs= 4.5×10^{-6} Cputime=1 NFE=114	CD=.3208522 m=15 eabs= 6×10^{-7} Cputime=2 NFE=76	CD=.3208522 m=15 eabs= 6×10^{-7} Cputime=4 NFE=82
1.0	CD=.3209173 m=25 eabs= 6.57×10^{-5} Cputime<1 NFE=110	CD=.3208601 m=25 eabs= 8.5×10^{-6} Cputime=1 NFE=110	CD=.3208527 m=18 eabs= 1.1×10^{-6} Cputime=2 NFE=86	CD=.3208527 m=18 eabs= 1.1×10^{-6} Cputime=4 NFE=97
1.5	CD=.3209261 m=25 eabs= 7.45×10^{-5} Cputime<1 NFE=111	CD=.3208812 m=25 eabs= 2.96×10^{-5} Cputime=1 NFE=111	CD=.3208621 m=21 eabs= 1.05×10^{-5} Cputime=2 NFE=126	CD=.3208619 m=21 eabs= 1.03×10^{-5} Cputime=4 NFE=131

TABLE (5.4.8):Results of PR with change in u_0 .

u_0	m	CD	e_{abs}	Cputime	eps	NFE
-0.1	15	0.3208530	1.4×10^{-6}	2	0.9	77
0	15	0.3208522	6×10^{-7}	2	0.9	76
0.3	15	0.3208522	6×10^{-7}	2	0.9	76
0.4	16	0.3208533	1.7×10^{-6}	2	1.1	81
0.5	12	0.3208527	1.1×10^{-6}	1	1.0	62
1.0	12	0.3208527	1.1×10^{-6}	1	1.5	53

TABLE (5.4.9):Results of H1 with varying ϵ and N.

N ϵ	100	400	800	1600
0.5	CD=.32011382 m=25 $e_{abs}=2.87 \times 10^{-4}$ Cputime<1 NFE=124	CD=.3208667 m=25 $e_{abs}=1.51 \times 10^{-5}$ Cputime=1 NFE=151	CD=.3208613 m=20 $e_{abs}=9.7 \times 10^{-6}$ Cputime=2 NFE=125	CD=.3208613 m=19 $e_{abs}=9.7 \times 10^{-6}$ Cputime=4 NFE=126
0.9	CD=.3211284 m=25 $e_{abs}=2.77 \times 10^{-4}$ Cputime<1 NFE=109	CD=.3208770 m=25 $e_{abs}=2.54 \times 10^{-5}$ Cputime=1 NFE=114	CD=.3208544 m=18 $e_{abs}=2.8 \times 10^{-6}$ Cputime=2 NFE=89	CD=.3208544 m=18 $e_{abs}=2.8 \times 10^{-6}$ Cputime=4 NFE=91
1.0	CD=.3214749 m=25 $e_{abs}=6.23 \times 10^{-4}$ Cputime<1 NFE=110	CD=.3208654 m=25 $e_{abs}=2.54 \times 10^{-5}$ Cputime=1 NFE=111	CD=.3208520 m=15 $e_{abs}=4.0 \times 10^{-7}$ Cputime=2 NFE=73	CD=.3208520 m=15 $e_{abs}=4.0 \times 10^{-7}$ Cputime=4 NFE=77
1.1	CD=.3209794 m=25 $e_{abs}=1.28 \times 10^{-4}$ Cputime<1 NFE=110	CD=.3209494 m=25 $e_{abs}=9.78 \times 10^{-5}$ Cputime=1 NFE=109	CD=.3208544 m=18 $e_{abs}=2.8 \times 10^{-6}$ Cputime=2 NFE=52	CD=.3208543 m=18 $e_{abs}=2.7 \times 10^{-6}$ Cputime=4 NFE=57
1.5	CD=.3209821 m=25 $e_{abs}=1.31 \times 10^{-4}$ Cputime<1 NFE=121	CD=.3209616 m=25 $e_{abs}=1.1 \times 10^{-4}$ Cputime=1 NFE=113	CD=.3208591 m=20 $e_{abs}=7.5 \times 10^{-6}$ Cputime=2 NFE=85	CD=.3208591 m=19 $e_{abs}=7.5 \times 10^{-6}$ Cputime=4 NFE=87

TABLE (5.4.10):Results of H1 with change in u_0 .

u_0	m	CD	e_{abs}	Cputime	eps	NFE
0	14	0.3208522	6×10^{-7}	1	1.4	60
0.3	17	0.3208521	5×10^{-7}	2	1.0	82
0.4	15	0.3208520	4×10^{-7}	2	1.0	73
0.5	12	0.3208530	1.4×10^{-6}	1	1.4	53
1.0	16	0.3208523	7×10^{-7}	2	1.1	77

TABLE (5.4.11):Results of ATH with varying ϵ and N.

N ϵ	100	400	800	1600
1.0	CD=.3217221 m=20 $e_{abs}=8.7 \times 10^{-4}$ Cputime<1 NFE=90	CD=.3209072 m=20 $e_{abs}=5.56 \times 10^{-5}$ Cputime=1 NFE=91	CD=.3208558 m=18 $e_{abs}=4.2 \times 10^{-6}$ Cputime=2 NFE=81	CD=.3208558 m=17 $e_{abs}=4.2 \times 10^{-6}$ Cputime=4 NFE=83
1.4	CD=.3258842 m=20 $e_{abs}=5.03 \times 10^{-3}$ Cputime<1 NFE=83	CD=.3212676 m=20 $e_{abs}=4.16 \times 10^{-4}$ Cputime=1 NFE=86	CD=.3209447 m=16 $e_{abs}=9.31 \times 10^{-5}$ Cputime=2 NFE=81	CD=.3209446 m=16 $e_{abs}=9.30 \times 10^{-5}$ Cputime=4 NFE=82
1.5	CD=.3285636 m=20 $e_{abs}=7.71 \times 10^{-3}$ Cputime<1 NFE=83	CD=.3209389 m=20 $e_{abs}=8.73 \times 10^{-5}$ Cputime=1 NFE=84	CD=.3208527 m=12 $e_{abs}=1.1 \times 10^{-6}$ Cputime=1 NFE=53	CD=.3208527 m=12 $e_{abs}=1.1 \times 10^{-6}$ Cputime=2 NFE=55
1.6	CD=.3277094 m=20 $e_{abs}=6.86 \times 10^{-3}$ Cputime<1 NFE=82	CD=.3209494 m=20 $e_{abs}=2.02 \times 10^{-4}$ Cputime=1 NFE=82	CD=.3211262 m=16 $e_{abs}=2.75 \times 10^{-4}$ Cputime=2 NFE=66	CD=.3211262 m=16 $e_{abs}=2.75 \times 10^{-4}$ Cputime=4 NFE=68
2.0	CD=.3284531 m=20 $e_{abs}=7.60 \times 10^{-3}$ Cputime<1 NFE=83	CD=.3209578 m=20 $e_{abs}=1.06 \times 10^{-4}$ Cputime=1 NFE=84	CD=.3211271 m=16 $e_{abs}=2.76 \times 10^{-4}$ Cputime=2 NFE=66	CD=.3211269 m=16 $e_{abs}=2.75 \times 10^{-4}$ Cputime=4 NFE=67

TABLE (5.4.12):Results of ATH with change in u_0 .

u_0	m	CD	ϵ_{abs}	Cputime	eps	NFE
0	13	0.3208528	1.2×10^{-6}	1	1.0	66
0.3	13	0.3208533	1.7×10^{-6}	1	1.0	64
0.4	13	0.3208534	1.8×10^{-6}	1	1.0	65
0.5	12	0.3208528	1.2×10^{-6}	1	1.0	62
1.0	12	0.3208527	1.1×10^{-6}	1	1.5	53
1.1	20	0.3208534	1.8×10^{-6}	2	1.5	90

TABLE (5.4.13):Results of H3 with varying ϵ and N.

N ϵ	100	400	800	1600
0.5	CD=.3211382 m=25 eabs= 2.67×10^{-4} Cputime<1 NFE=124	CD=.3208648 m=25 eabs= 1.32×10^{-5} Cputime=1 NFE=150	CD=.3208613 m=20 eabs= 9.7×10^{-6} Cputime=2 NFE=125	CD=.3208611 m=20 eabs= 9.5×10^{-6} Cputime=4 NFE=127
0.9	CD=.3211462 m=25 eabs= 2.95×10^{-4} Cputime<1 NFE=110	CD=.3208776 m=25 eabs= 2.6×10^{-5} Cputime=1 NFE=114	CD=.3208543 m=18 eabs= 2.7×10^{-6} Cputime=2 NFE=89	CD=.3208543 m=17 eabs= 2.7×10^{-7} Cputime=4 NFE=90
1.0	CD=.3211598 m=25 eabs= 3.08×10^{-4} Cputime<1 NFE=110	CD=.3208650 m=25 eabs= 1.34×10^{-5} Cputime=1 NFE=112	CD=.3208518 m=15 eabs= 2.0×10^{-7} Cputime=2 NFE=73	CD=.3208518 m=15 eabs= 2.0×10^{-7} Cputime=4 NFE=76
1.1	CD=.3210917 m=25 eabs= 2.4×10^{-4} Cputime<1 NFE=109	CD=.3209494 m=25 eabs= 9.78×10^{-5} Cputime=1 NFE=109	CD=.3208576 m=18 eabs= 6.0×10^{-6} Cputime=2 NFE=81	CD=.3208575 m=18 eabs= 5.9×10^{-6} Cputime=4 NFE=84
1.5	CD=.3210971 m=25 eabs= 2.46×10^{-4} Cputime<1 NFE=115	CD=.3209641 m=25 eabs= 1.13×10^{-4} Cputime=1 NFE=113	CD=.3208587 m=18 eabs= 7.1×10^{-6} Cputime=2 NFE=83	CD=.3208587 m=18 eabs= 7.1×10^{-6} Cputime=4 NFE=85

TABLE (5.4.14):Results of H3 with change in u_0 .

u_0	m	CD	e_{abs}	Cputime	eps	NFE
0	14	0.3208526	1×10^{-6}	1	1.4	60
0.3	13	0.3208527	1.1×10^{-6}	1	1.7	53
0.4	15	0.3208518	2.0×10^{-7}	2	1.0	73
0.5	12	0.3208530	1.4×10^{-6}	1	1.4	53
1.0	14	0.3208527	1.1×10^{-6}	1	1.3	65

Table (5.5.1): Summary table for the seven methods

$\frac{\text{method}}{u_1}$	GFS	SD	FR	PR	H1	ATH	H3
0	CD = 0.3208520 $m = 40$ $e_{abs} = 4 \times 10^{-7}$ Cputime = 1 $\varepsilon = 1.8$ NFE = 41	CD = 0.3208534 $m = 400$ $e_{abs} = 1.8 \times 10^{-6}$ Cputime = 24 $\varepsilon = 1.9$ NFE = 1624	CD = 0.3208525 $m = 14$ $e_{abs} = 9 \times 10^{-7}$ Cputime = 1 $\varepsilon = 1.5$ NFE = 60	CD = 0.3208522 $m = 15$ $e_{abs} = 6 \times 10^{-7}$ Cputime = 2 $\varepsilon = 0.9$ NFE = 76	CD = 0.3208522 $m = 14$ $e_{abs} = 6 \times 10^{-7}$ Cputime = 2 $\varepsilon = 1.4$ NFE = 60	CD = 0.3208528 $m = 13$ $e_{abs} = 1.2 \times 10^{-6}$ Cputime = 1 $\varepsilon = 1.0$ NFE = 66	CD = 0.3208526 $m = 14$ $e_{abs} = 1 \times 10^{-6}$ Cputime = 1 $\varepsilon = 1.4$ NFE = 60
0.3	CD = 0.3208519 $m = 50$ $e_{abs} = 3 \times 10^{-7}$ Cputime = 1 $\varepsilon = 1.8$ NFE = 51	CD = 0.3208534 $m = 400$ $e_{abs} = 1.8 \times 10^{-6}$ Cputime = 24 $\varepsilon = 1.9$ NFE = 1624	CD = 0.3208522 $m = 17$ $e_{abs} = 6 \times 10^{-7}$ Cputime = 2 $\varepsilon = 1.0$ NFE = 81	CD = 0.3208522 $m = 15$ $e_{abs} = 6 \times 10^{-7}$ Cputime = 2 $\varepsilon = 0.9$ NFE = 76	CD = 0.3208521 $m = 17$ $e_{abs} = 5 \times 10^{-7}$ Cputime = 2 $\varepsilon = 1.0$ NFE = 82	CD = 0.3208533 $m = 13$ $e_{abs} = 1.7 \times 10^{-6}$ Cputime = 1 $\varepsilon = 1.0$ NFE = 64	CD = 0.3208527 $m = 13$ $e_{abs} = 1.1 \times 10^{-6}$ Cputime = 1 $\varepsilon = 1.7$ NFE = 53
0.4	CD = 0.3208531 $m = 60$ $e_{abs} = 1.5 \times 10^{-6}$ Cputime = 1 $\varepsilon = 1.8$ NFE = 61	CD = 0.3208535 $m = 300$ $e_{abs} = 1.9 \times 10^{-6}$ Cputime = 18 $\varepsilon = 1.9$ NFE = 1222	CD = 0.3208517 $m = 15$ $e_{abs} = 1 \times 10^{-7}$ Cputime = 2 $\varepsilon = 2$ NFE = 70	CD = 0.3208533 $m = 16$ $e_{abs} = 1.7 \times 10^{-6}$ Cputime = 2 $\varepsilon = 1.1$ NFE = 62	CD = 0.3208520 $m = 15$ $e_{abs} = 4 \times 10^{-7}$ Cputime = 1 $\varepsilon = 1.0$ NFE = 73	CD = 0.3208534 $m = 13$ $e_{abs} = 1.8 \times 10^{-6}$ Cputime = 1 $\varepsilon = 1.0$ NFE = 65	CD = 0.3208518 $m = 15$ $e_{abs} = 2.0 \times 10^{-7}$ Cputime = 1 $\varepsilon = 1.0$ NFE = 73
0.5	CD = 0.3208531 $m = 70$ $e_{abs} = 1.5 \times 10^{-6}$ Cputime = 1 $\varepsilon = 1.7$ NFE = 71	CD = 0.3208535 $m = 400$ $e_{abs} = 1.9 \times 10^{-6}$ Cputime = 24 $\varepsilon = 2.0$ NFE = 1622	CD = 0.3208531 $m = 15$ $e_{abs} = 1.5 \times 10^{-6}$ Cputime = 2 $\varepsilon = 1.2$ NFE = 62	CD = 0.3208527 $m = 12$ $e_{abs} = 1.1 \times 10^{-6}$ Cputime = 1 $\varepsilon = 1.0$ NFE = 62	CD = 0.3208530 $m = 12$ $e_{abs} = 1.4 \times 10^{-6}$ Cputime = 1 $\varepsilon = 1.4$ NFE = 53	CD = 0.3208528 $m = 12$ $e_{abs} = 1.2 \times 10^{-6}$ Cputime = 1 $\varepsilon = 1.0$ NFE = 62	CD = 0.3208530 $m = 12$ $e_{abs} = 1.4 \times 10^{-6}$ Cputime = 1 $\varepsilon = 1.4$ NFE = 53
1.0	CD = 0.3208517 $m = 30$ $e_{abs} = 1 \times 10^{-7}$ Cputime = 1 $\varepsilon = 1.7$ NFE = 31	CD = 0.3208526 $m = 243$ $e_{abs} = 1.0 \times 10^{-6}$ Cputime = 15 $\varepsilon = 2.0$ NFE = 987	CD = 0.3208522 $m = 14$ $e_{abs} = 6 \times 10^{-7}$ Cputime = 1 $\varepsilon = 1.3$ NFE = 65	CD = 0.3208527 $m = 12$ $e_{abs} = 1.1 \times 10^{-6}$ Cputime = 1 $\varepsilon = 1.5$ NFE = 53	CD = 0.3208523 $m = 16$ $e_{abs} = 7 \times 10^{-7}$ Cputime = 2 $\varepsilon = 1.1$ NFE = 77	CD = 0.3208527 $m = 12$ $e_{abs} = 1.1 \times 10^{-6}$ Cputime = 1 $\varepsilon = 1.5$ NFE = 53	CD = 0.3208527 $m = 14$ $e_{abs} = 1.1 \times 10^{-6}$ Cputime = 1 $\varepsilon = 1.3$ NFE = 65

GFS

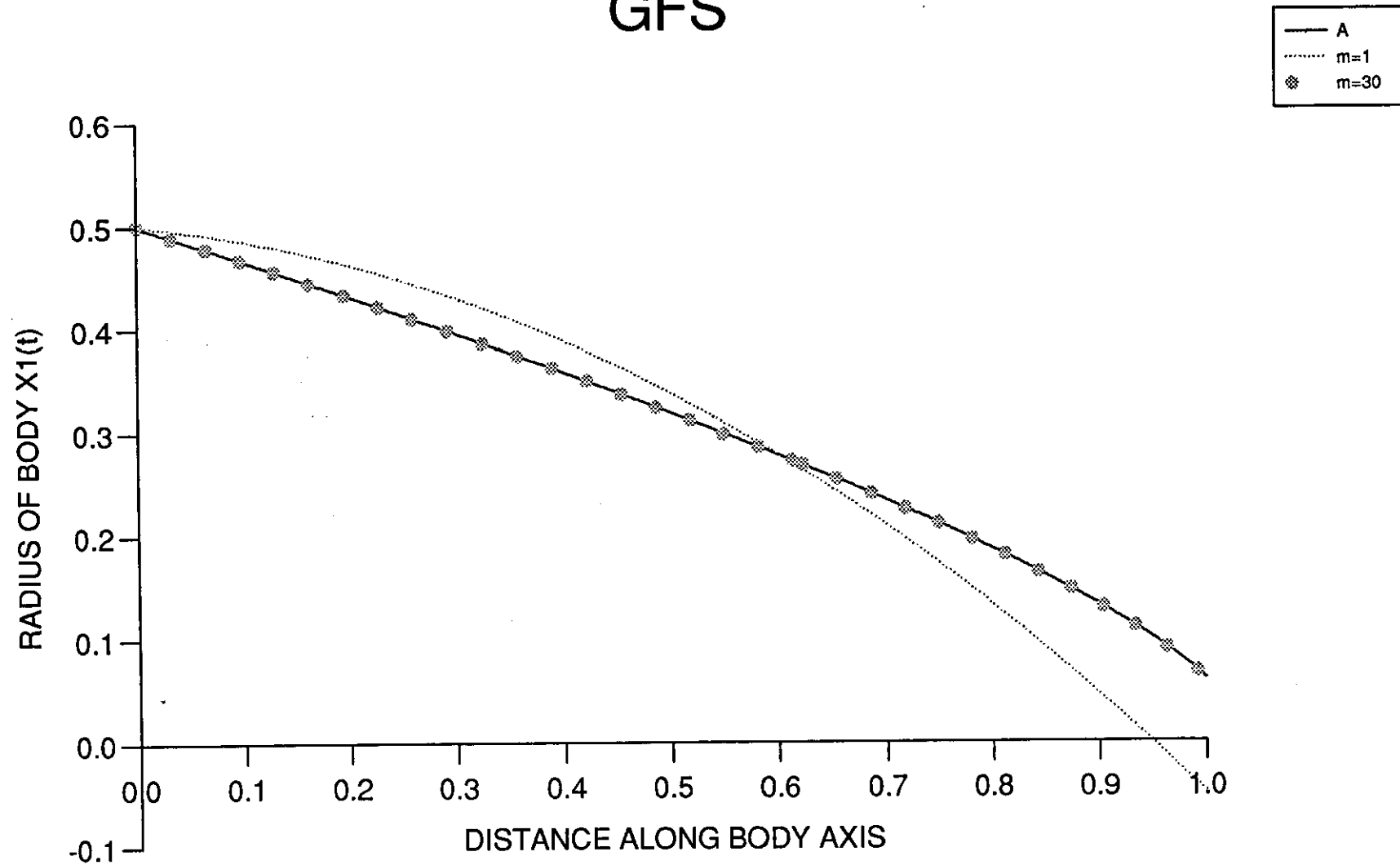


Figure (5.4.1)

GFS

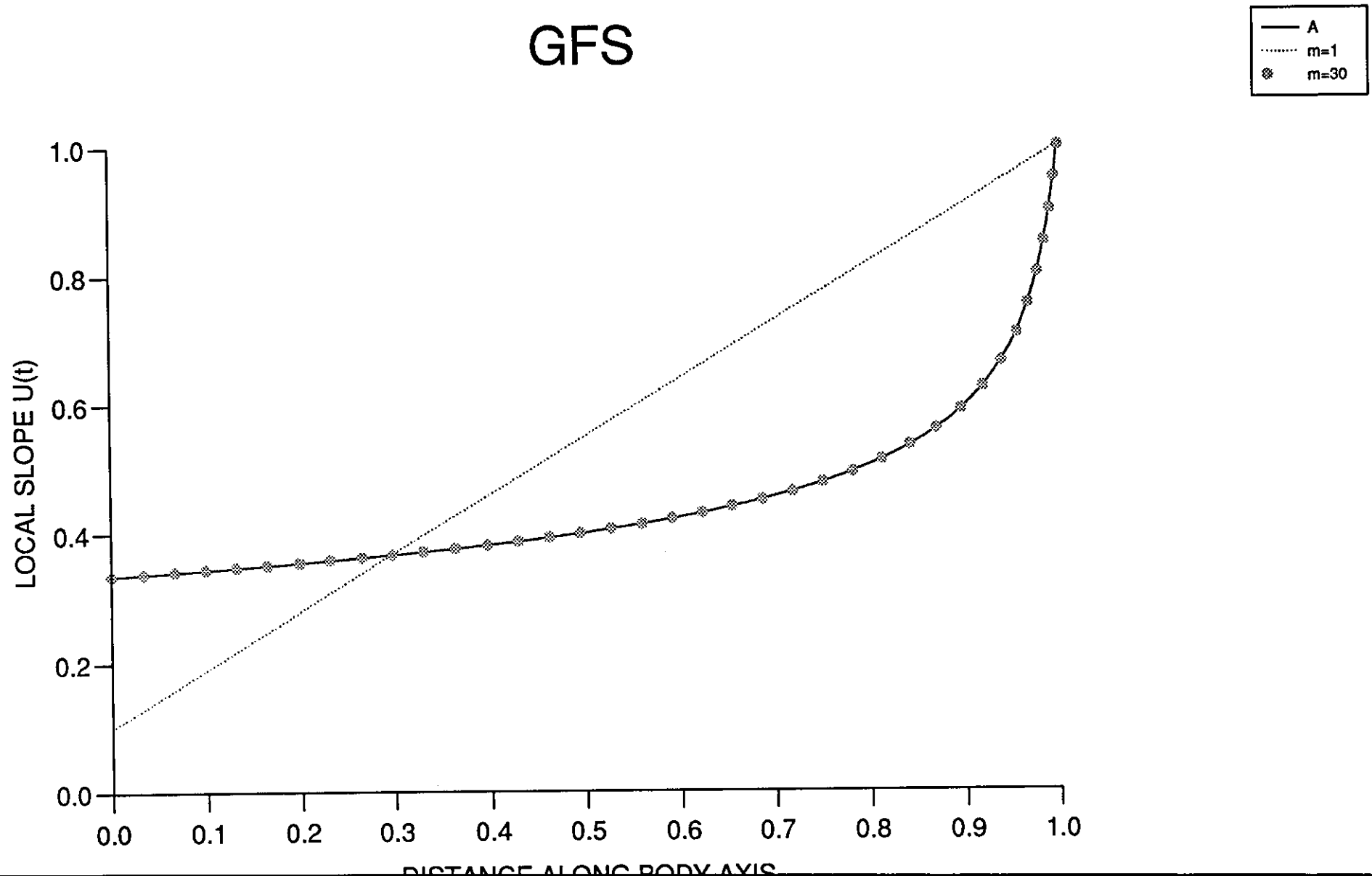


Figure (5.4.2)

GRADIENT IN FUNCTION SPACE

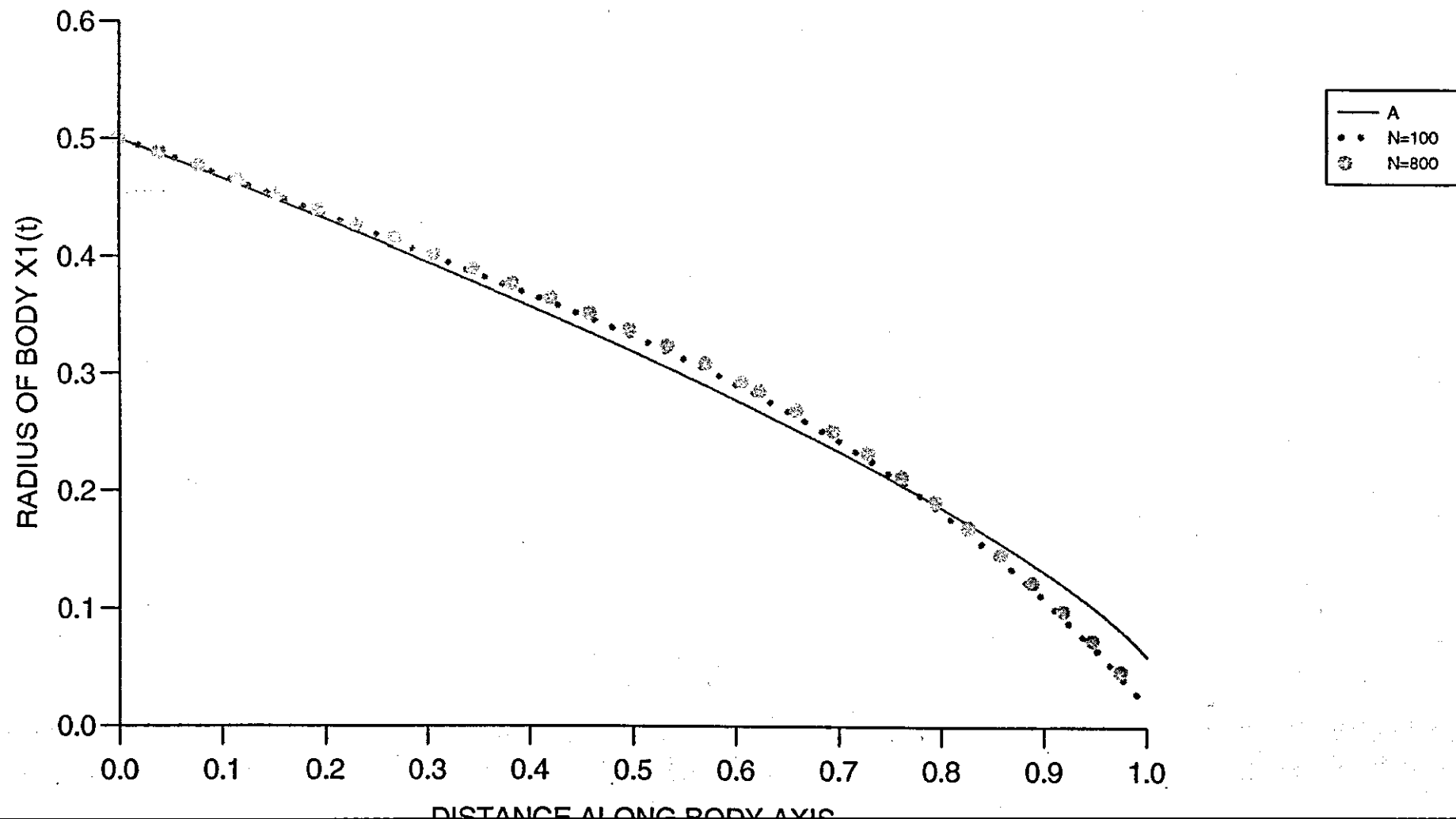


Figure (5.4.3)

GRADIENT IN FUNCTION SPACE

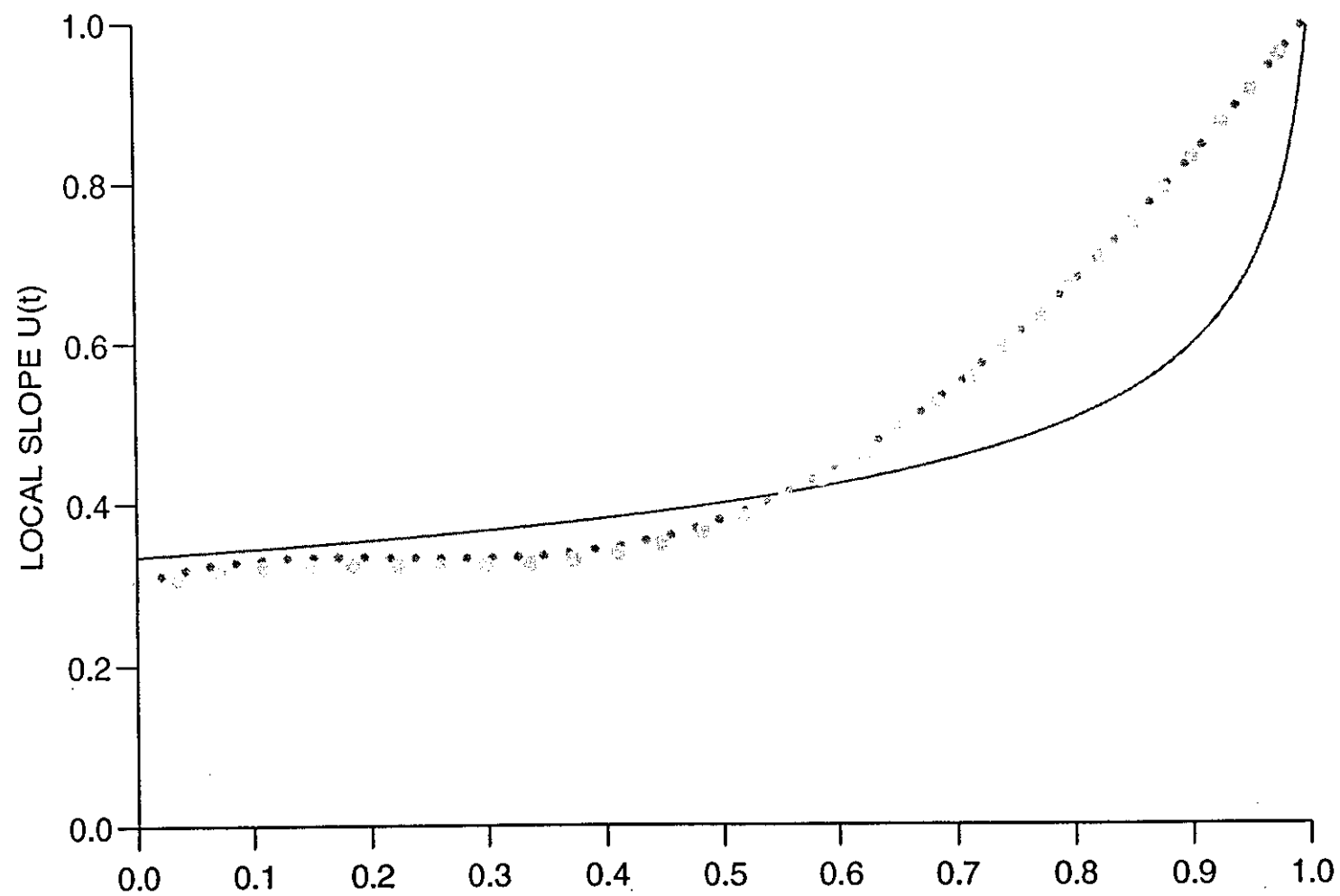


Figure (5.4.4)

GRADIENT IN FUNCTION SPACE

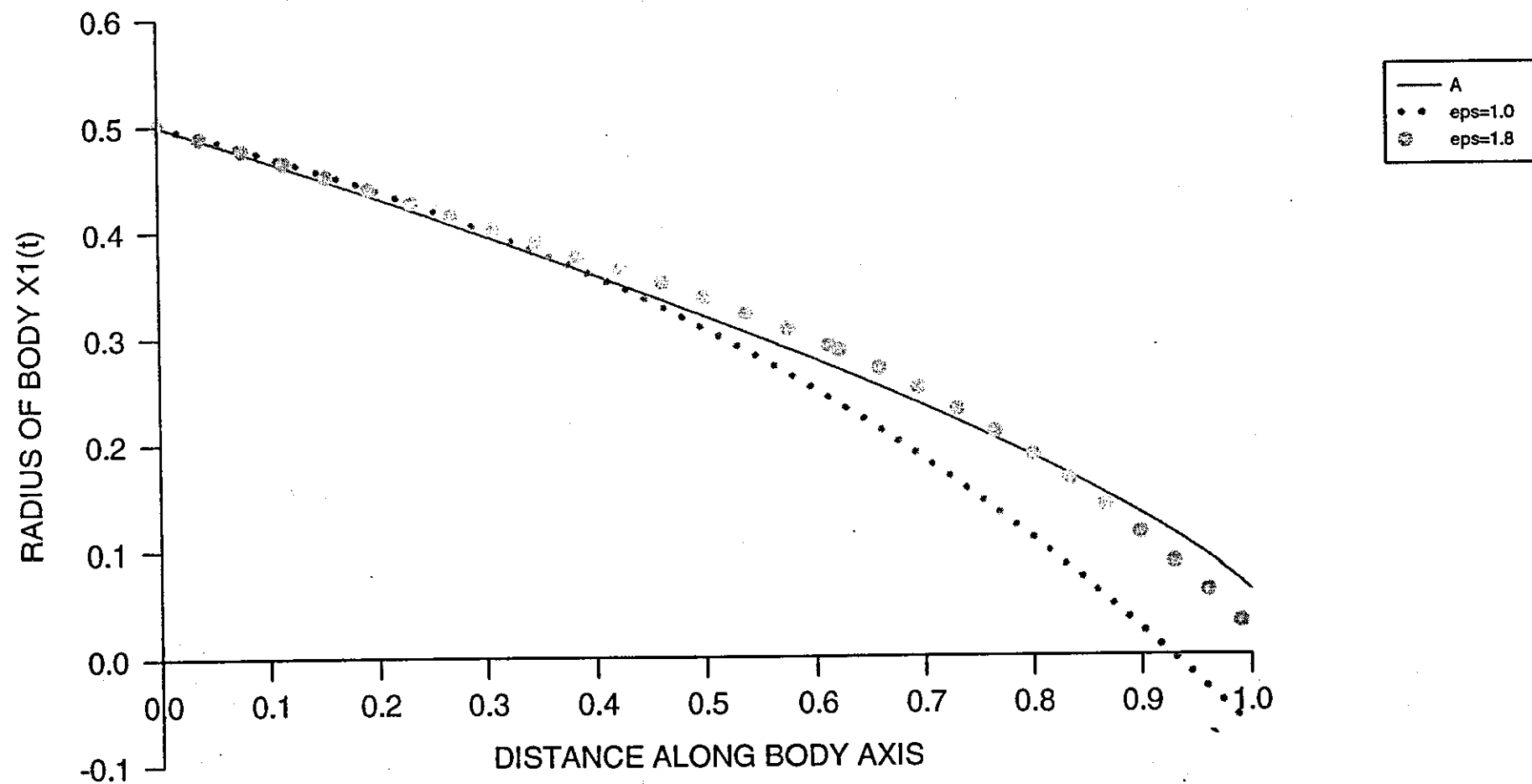


Figure (5.4.5)

GRADIENT IN FUNCTION SPACE

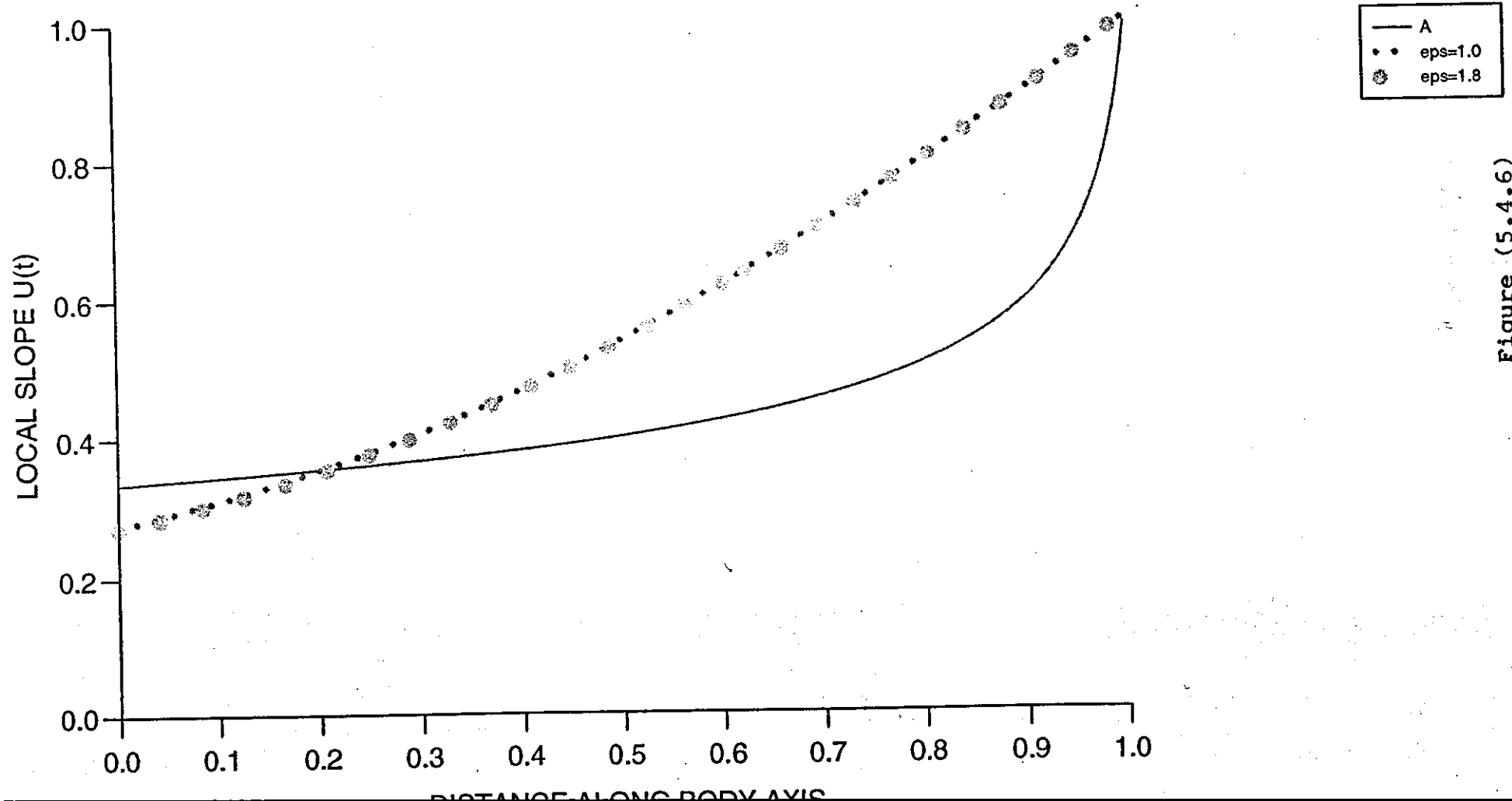


Figure (5.4.6)

GRADIENT IN FUNCTION SPACE

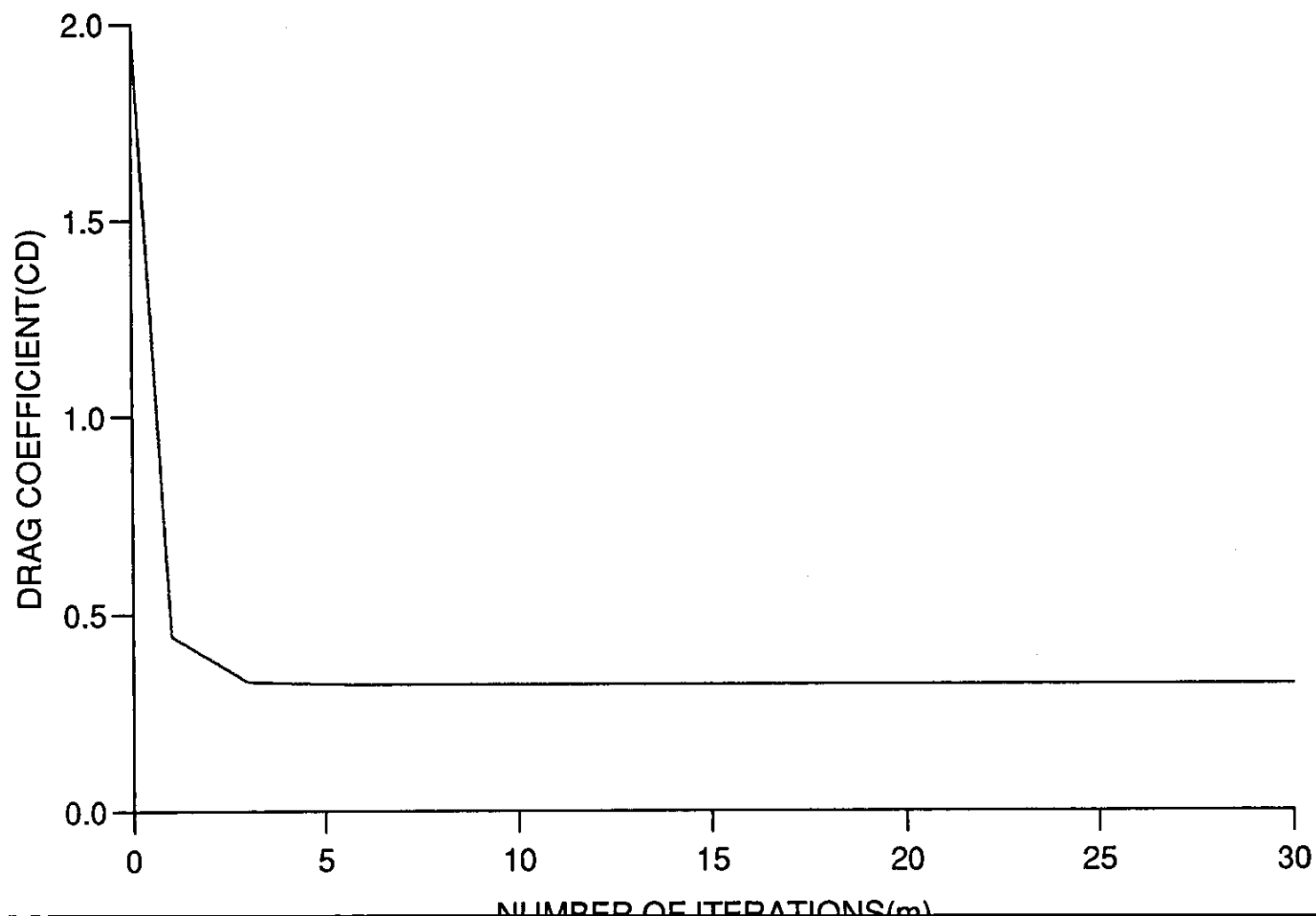


Figure (5.4.7)

GRADIENT IN FUNCTION SPACE

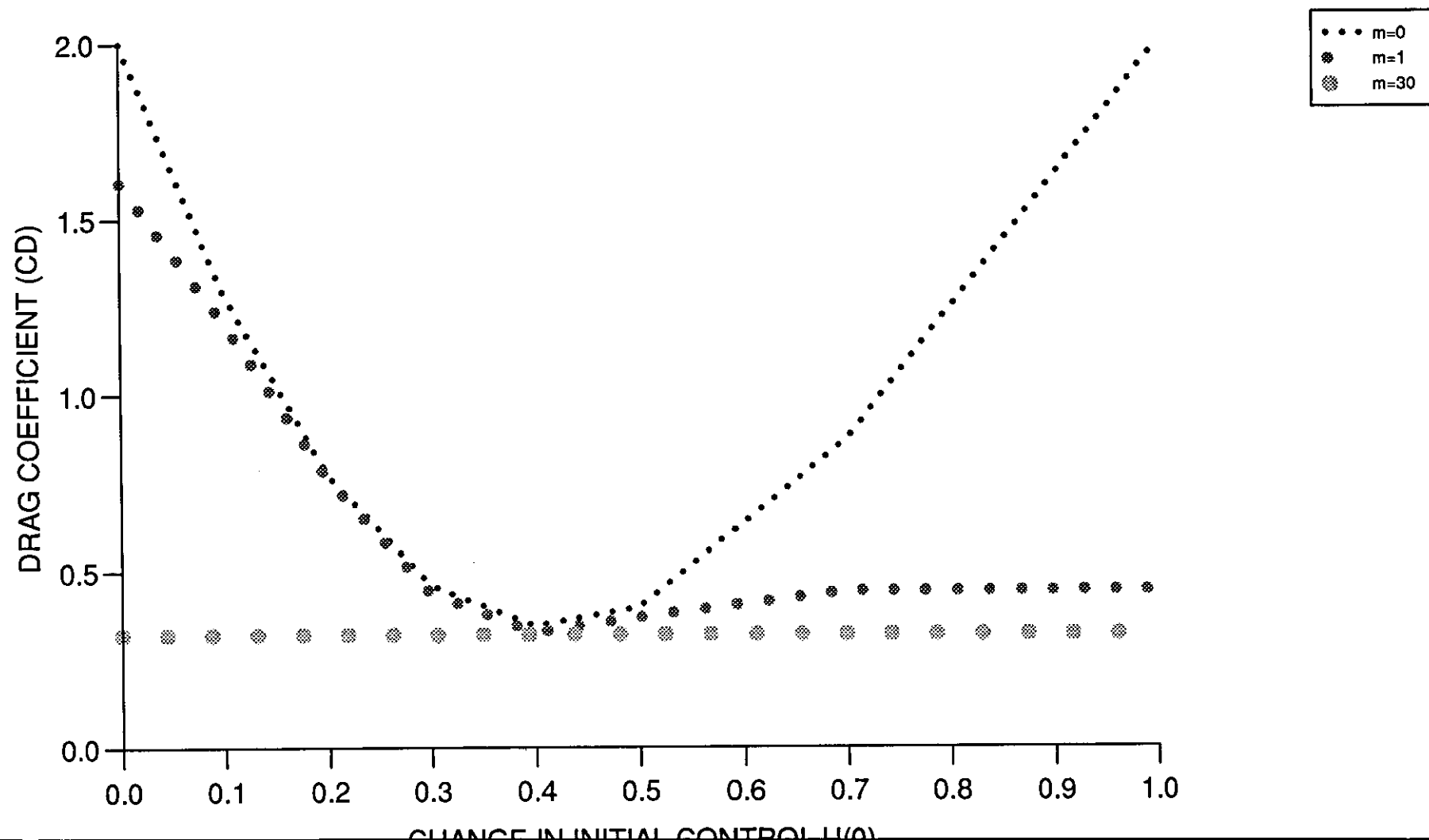


Figure (5.4.8)

GRADIENT IN FUNCTION SPACE

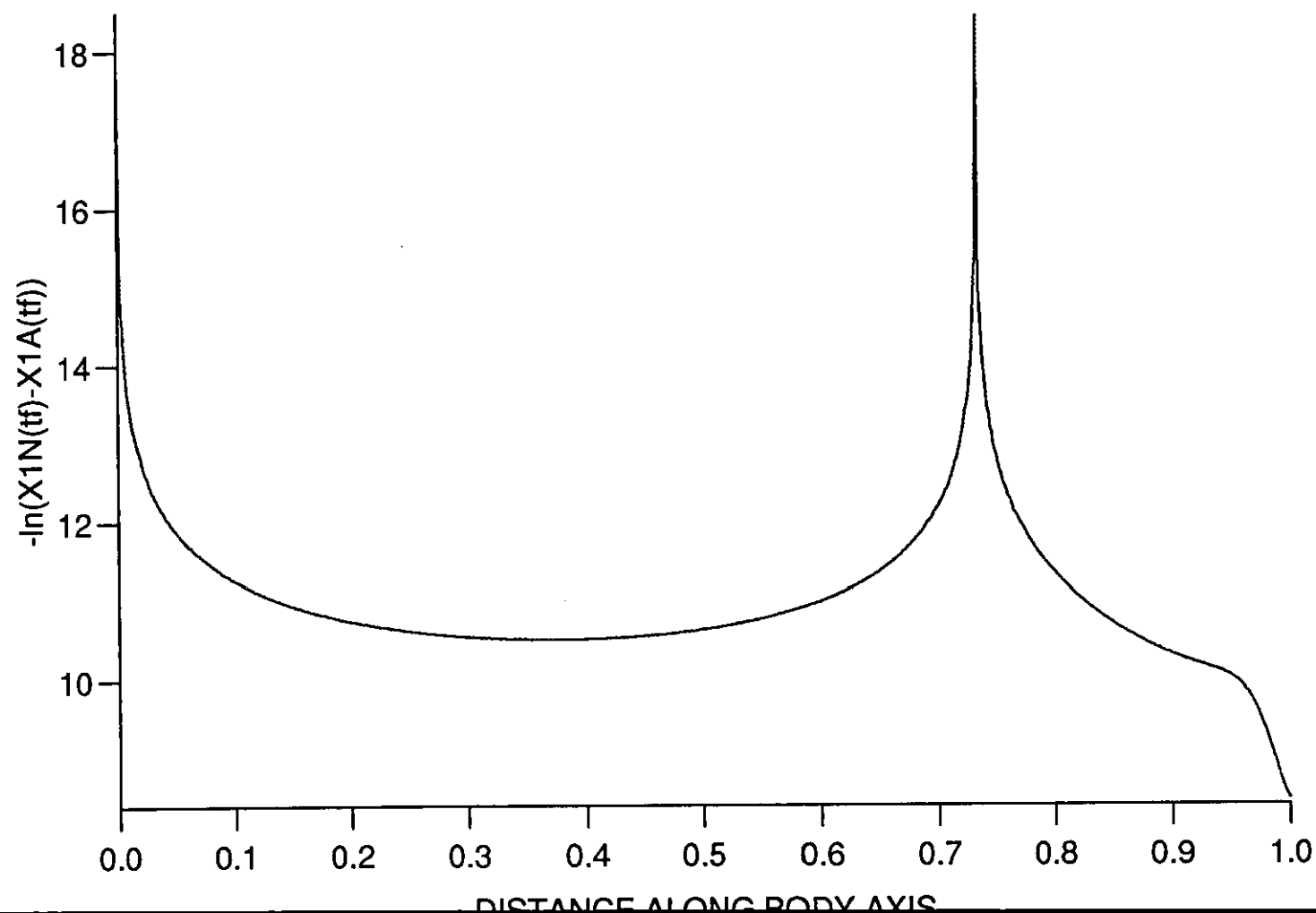


Figure (5.4.9)

GRADIENT IN FUNCTION SPACE

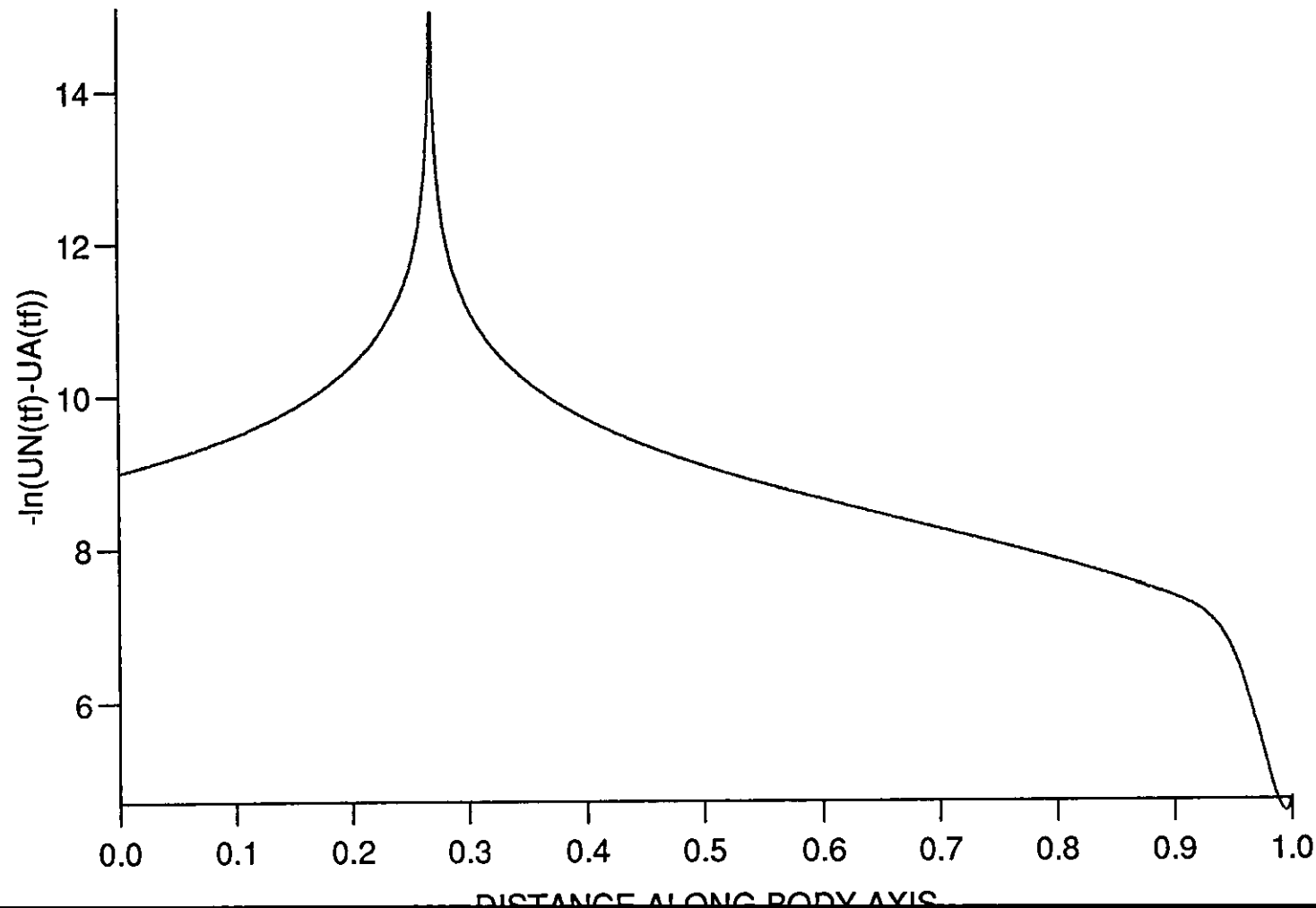


Figure (5.4.10)

STEEPEST DESCENT

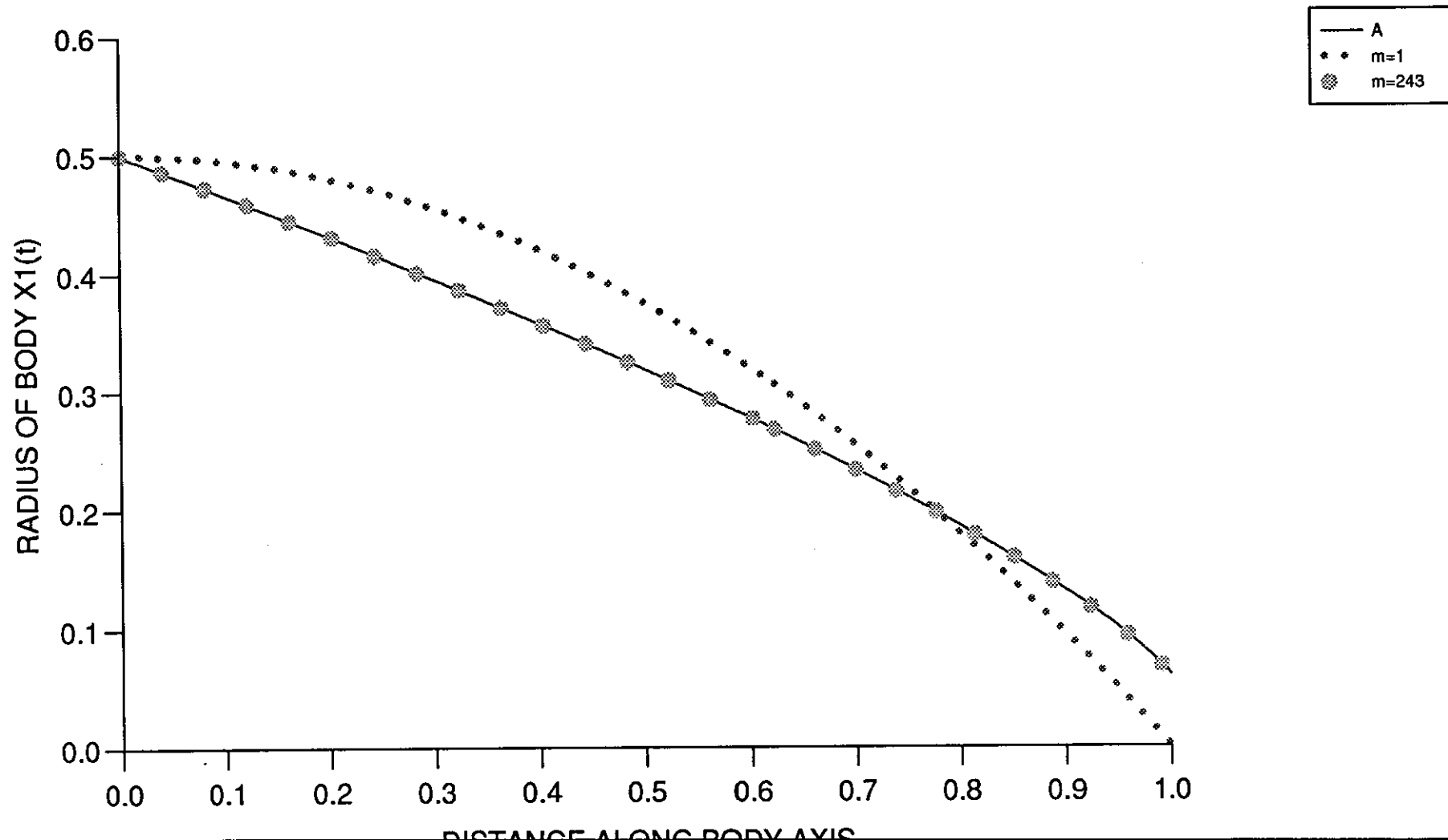


Figure (5.4.11)

STEEPEST DESCENT

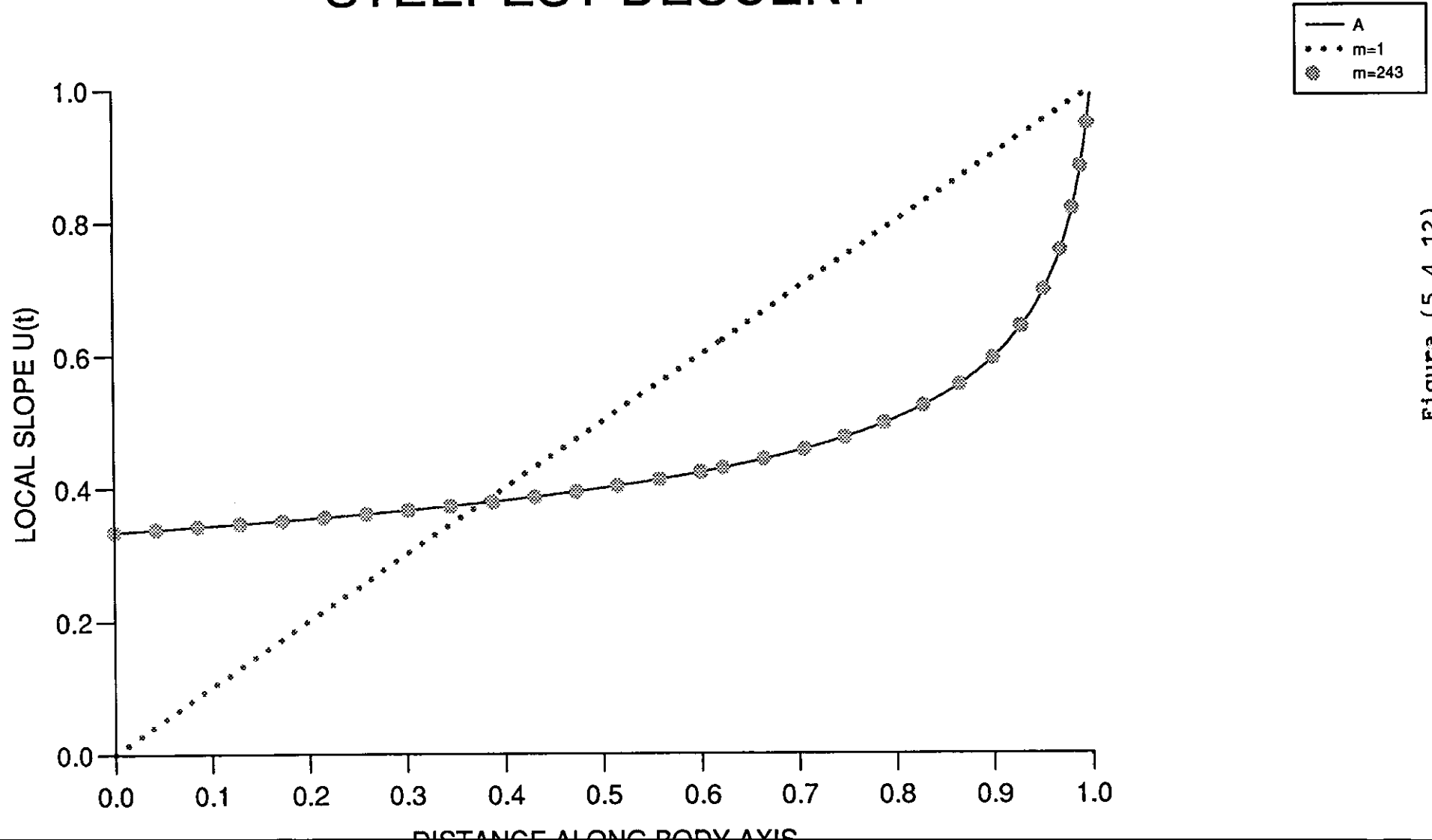


Figure (5.4.12)

STEEPEST DESCENT

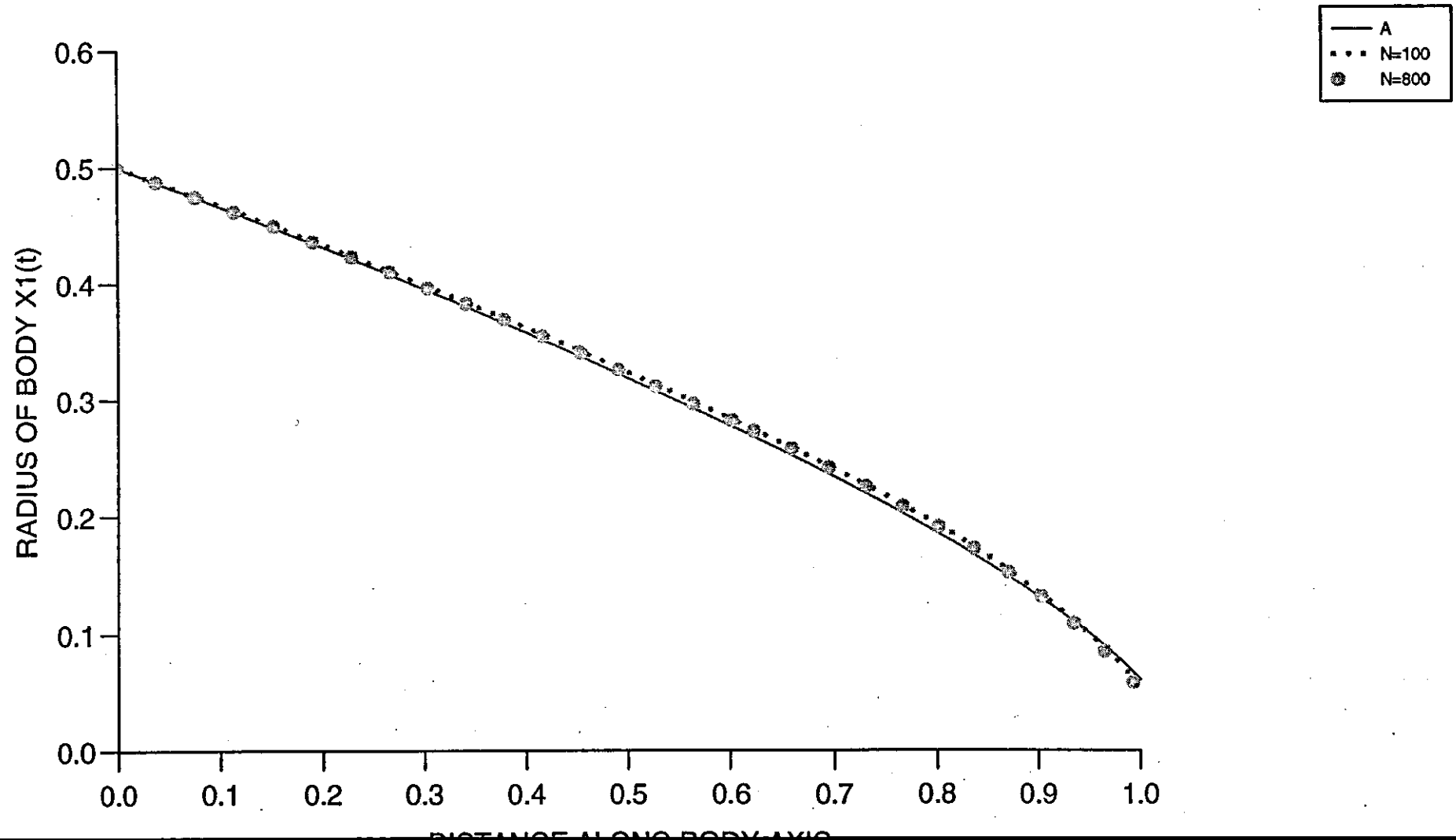


Figure (5.4.13)

STEEPEST DESCENT

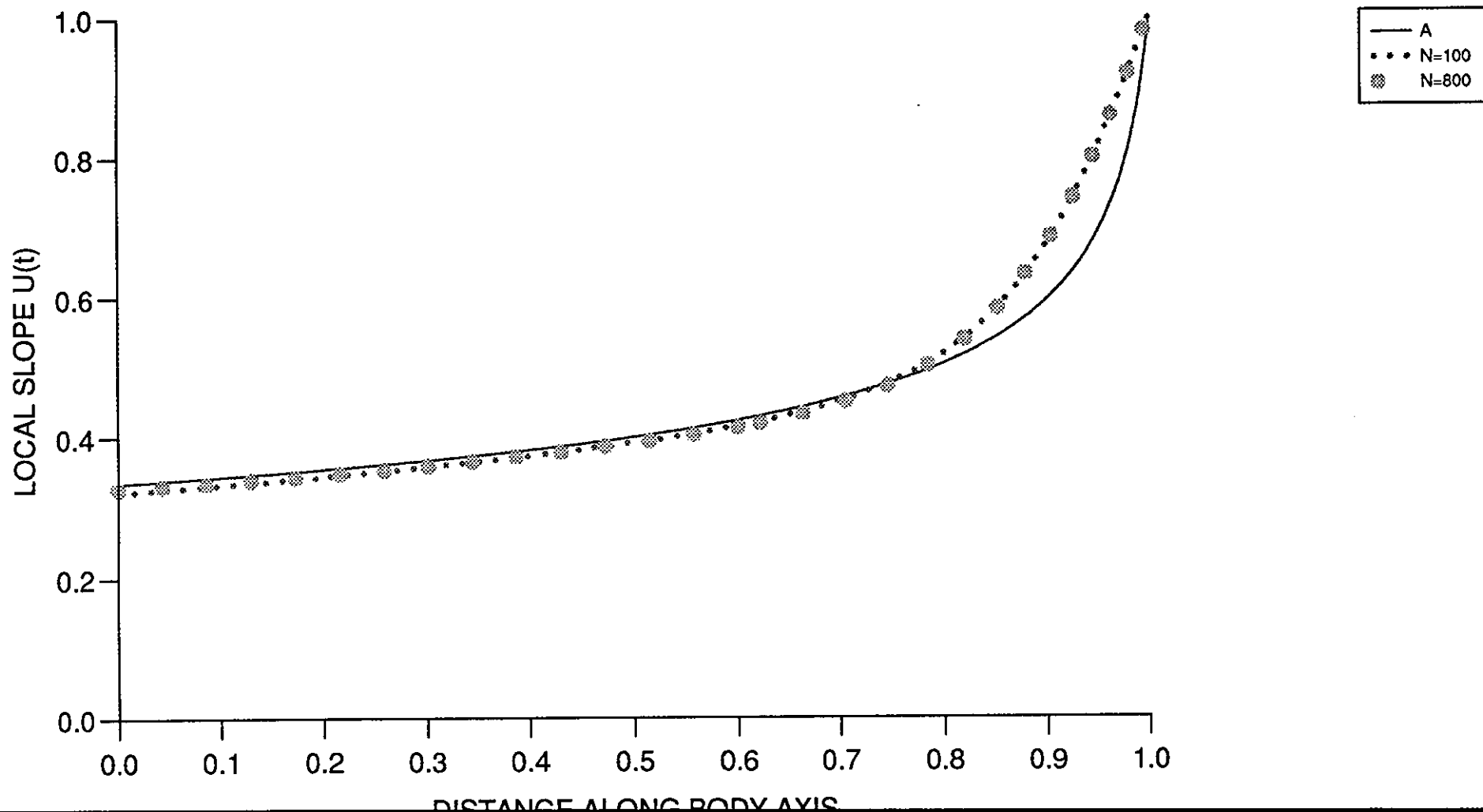


Figure (5.4.14)

STEEPEST DESCENT

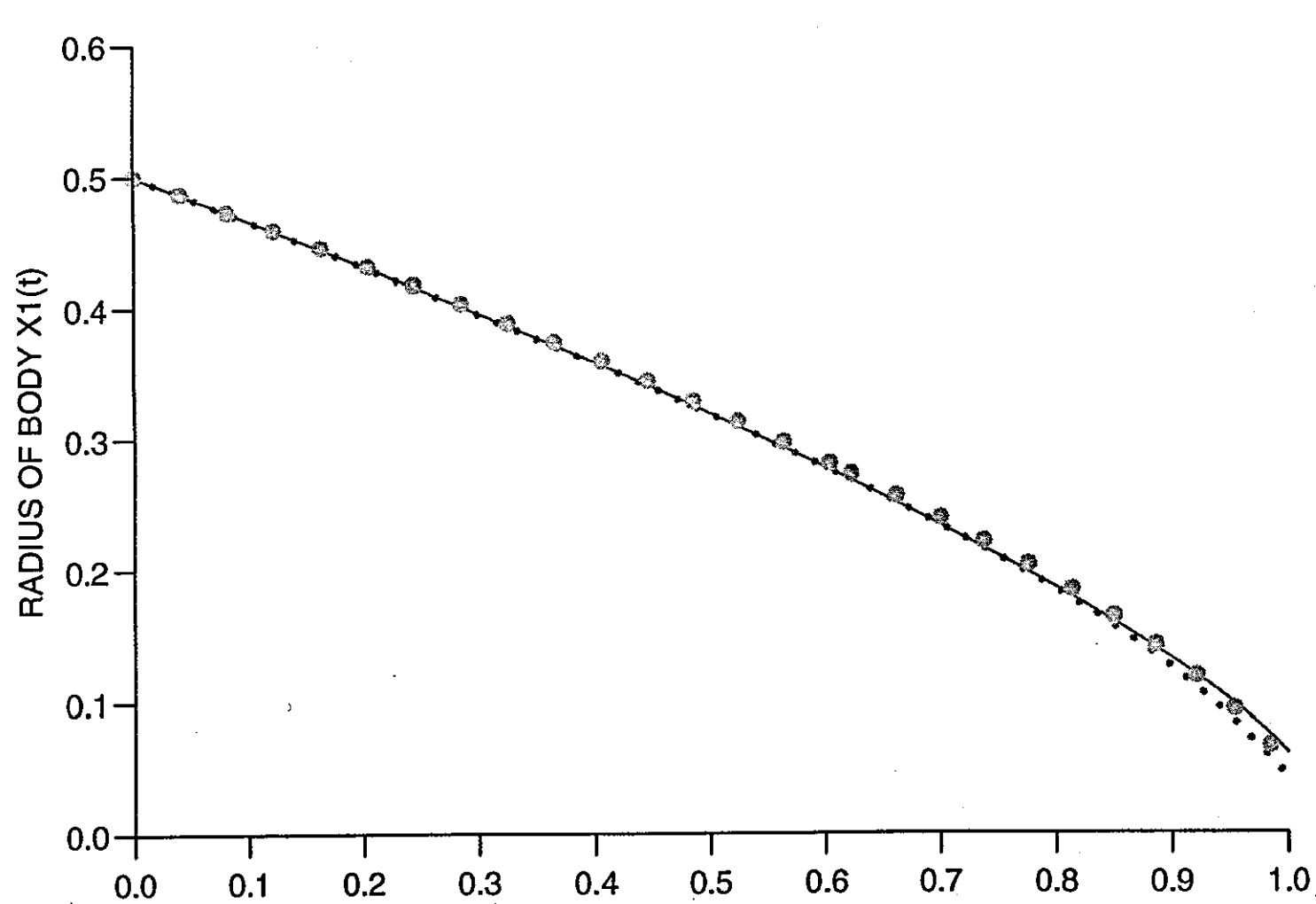


Figure (5.4.15)

STEEPEST DESCENT

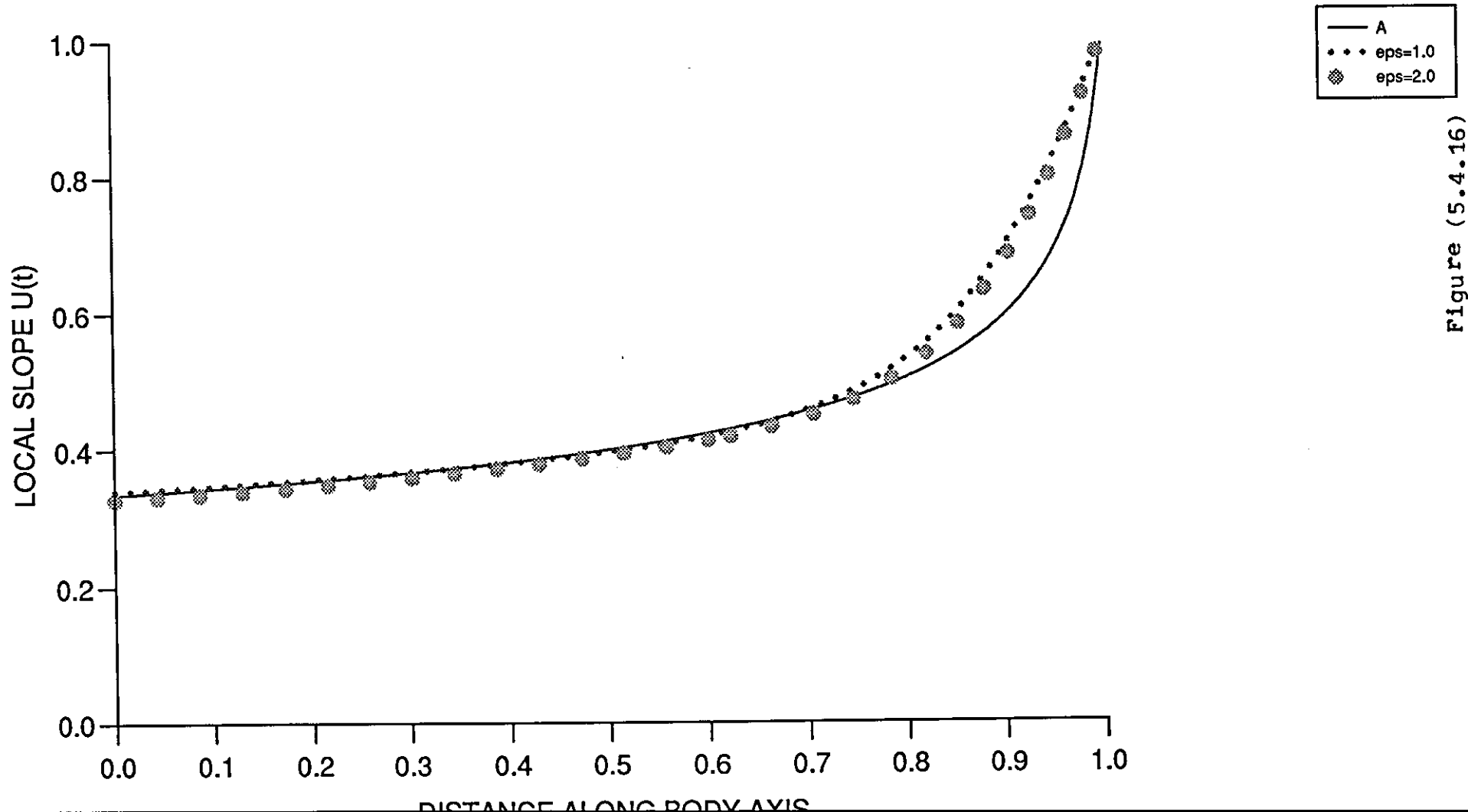


Figure (5.4.16)

STEEPEST DESCENT

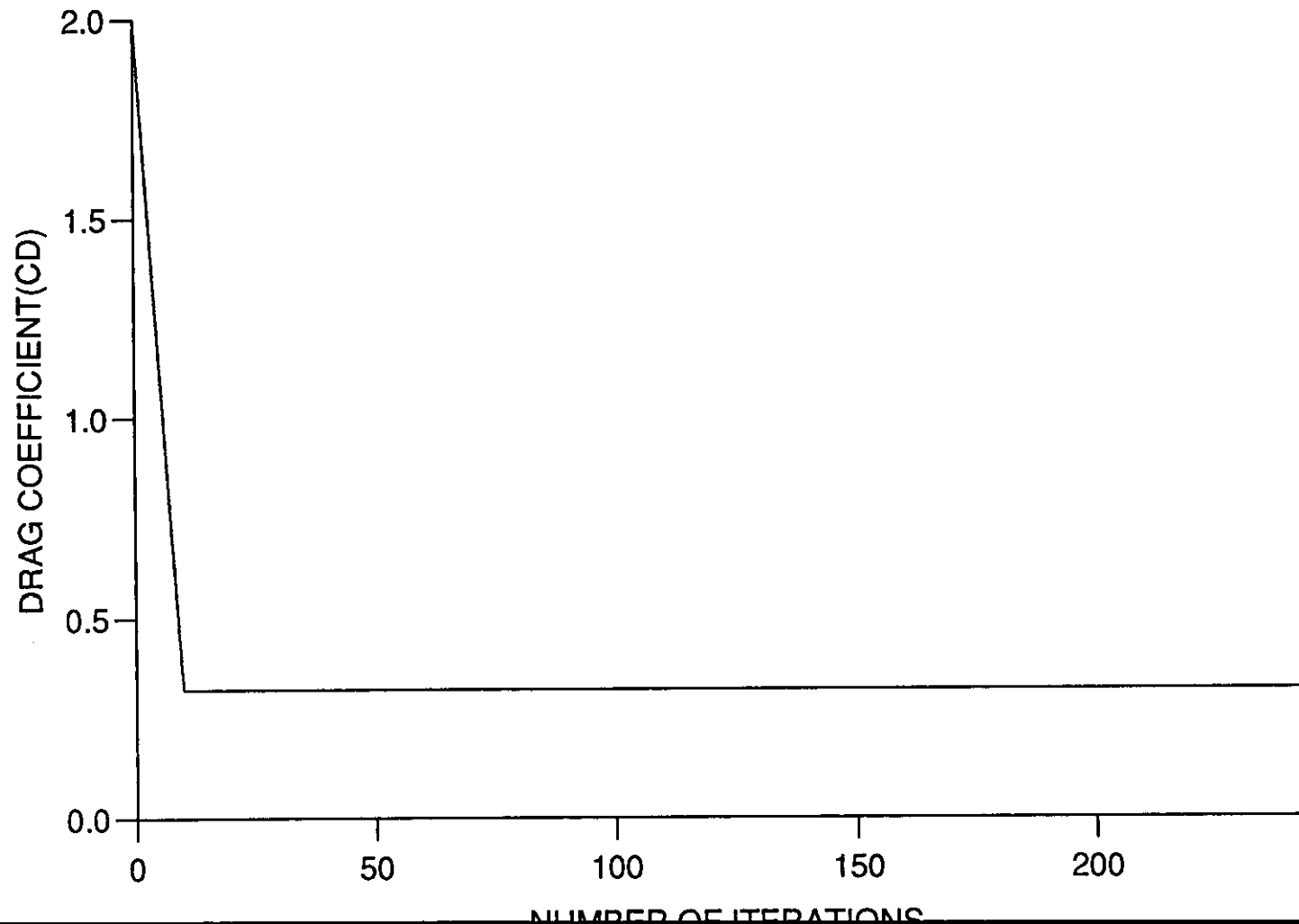


Figure (5.4.17)

STTEPEST DESCENT

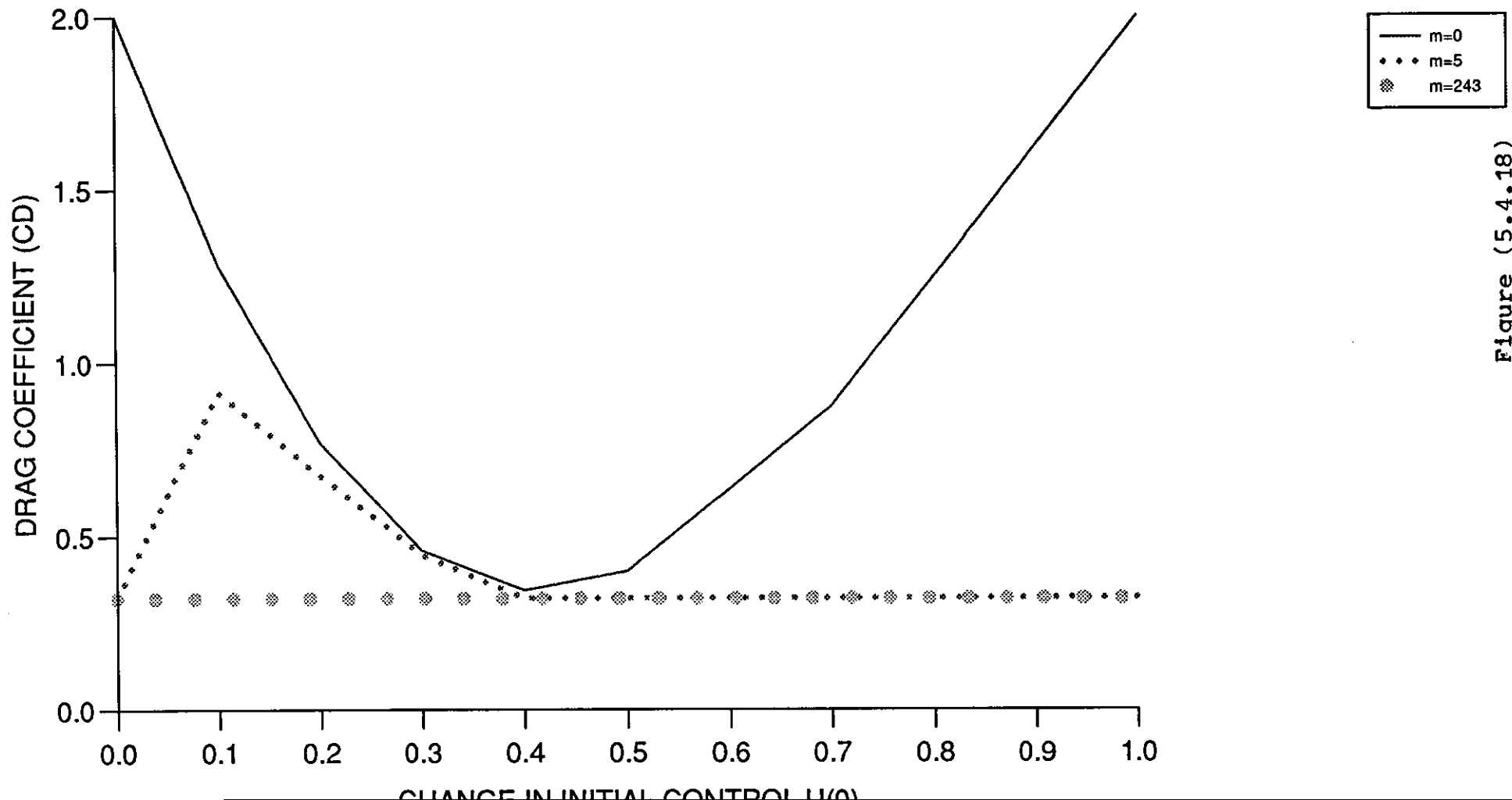


Figure (5.4.18)

STEEPEST DESCENT

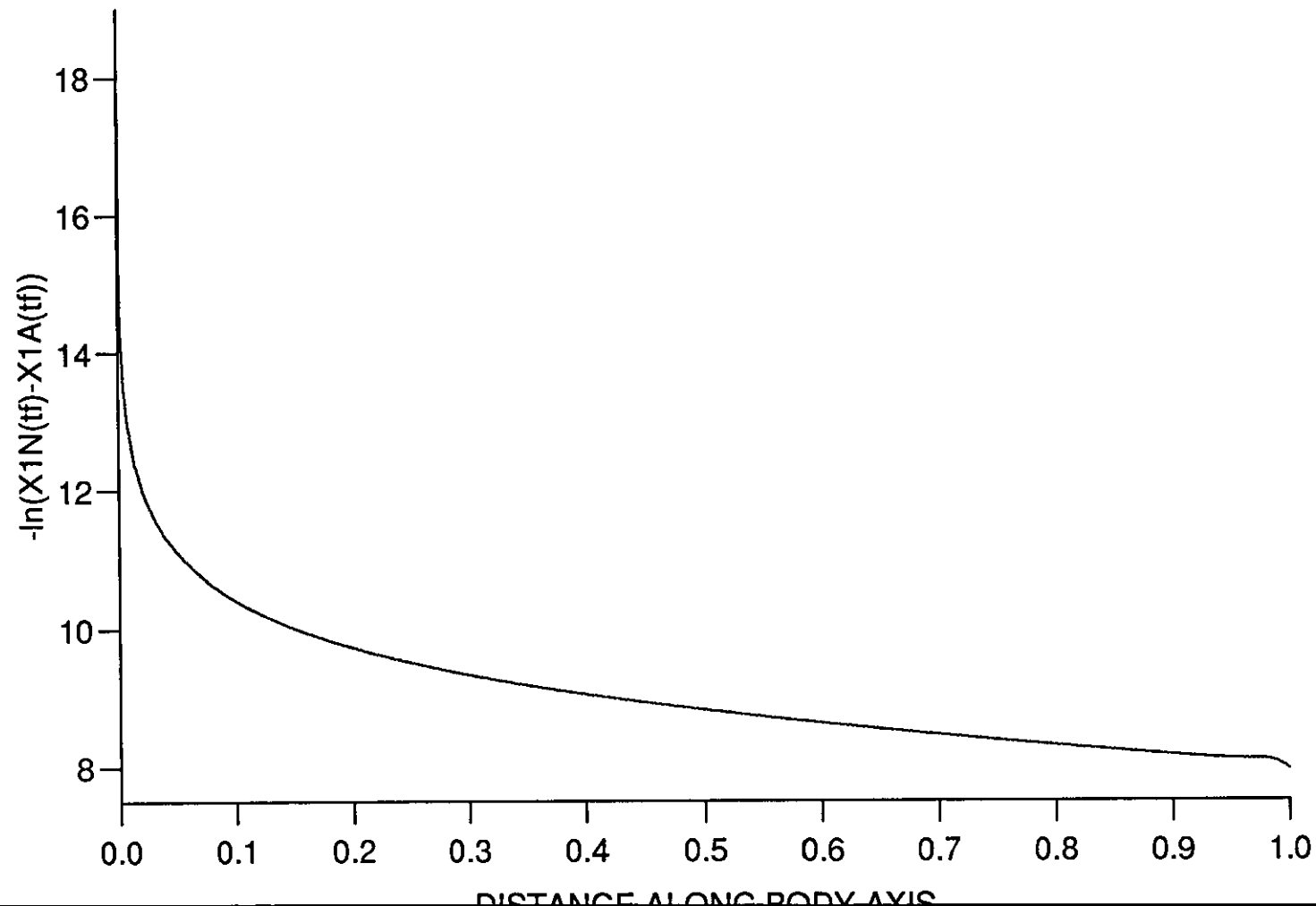


Figure (5.4.19)

STEEPEST DESCENT

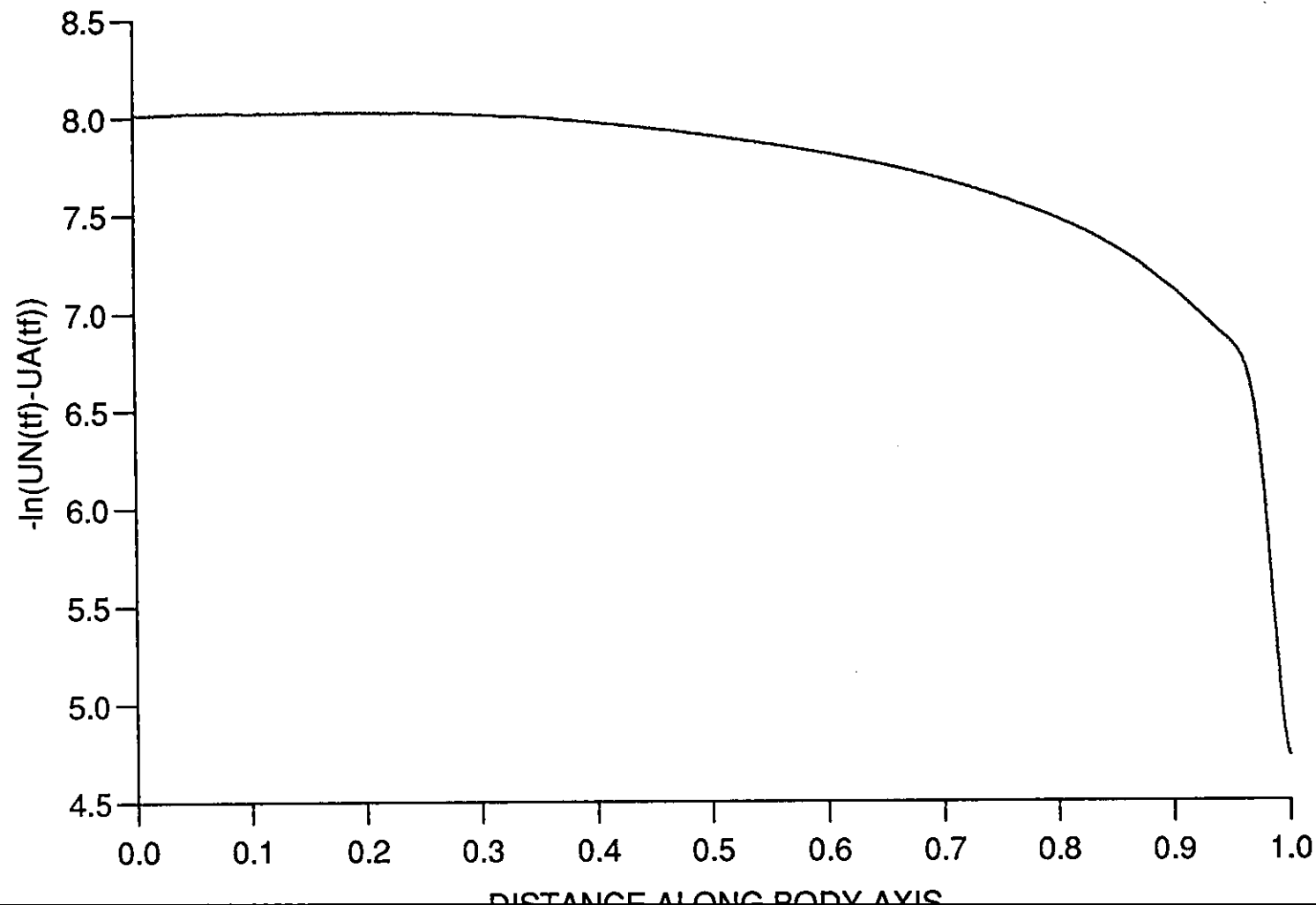


Figure (5.4.20)

FLETCHER REEVES

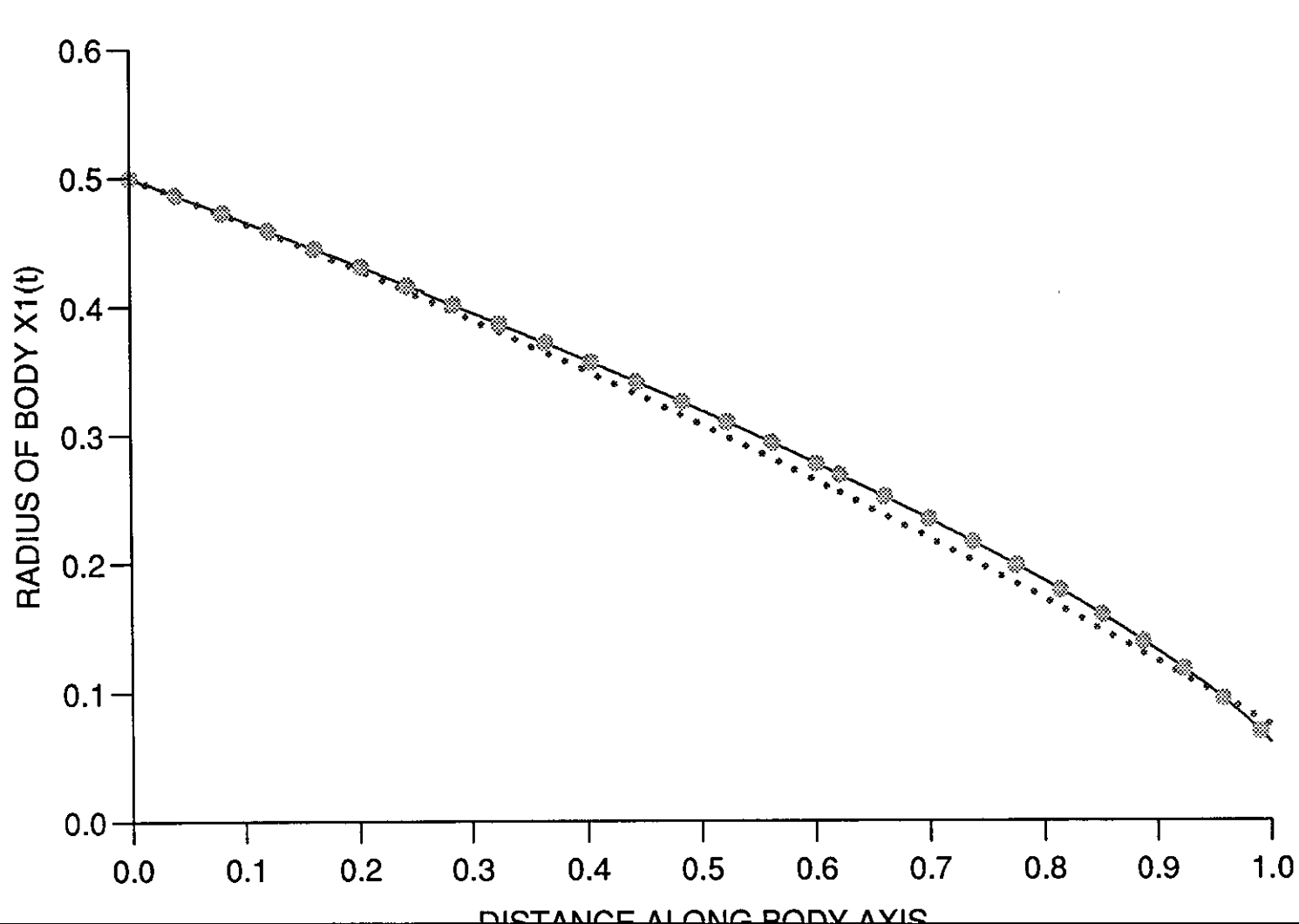


Figure (5.4.21)

FLETCHER REEVES

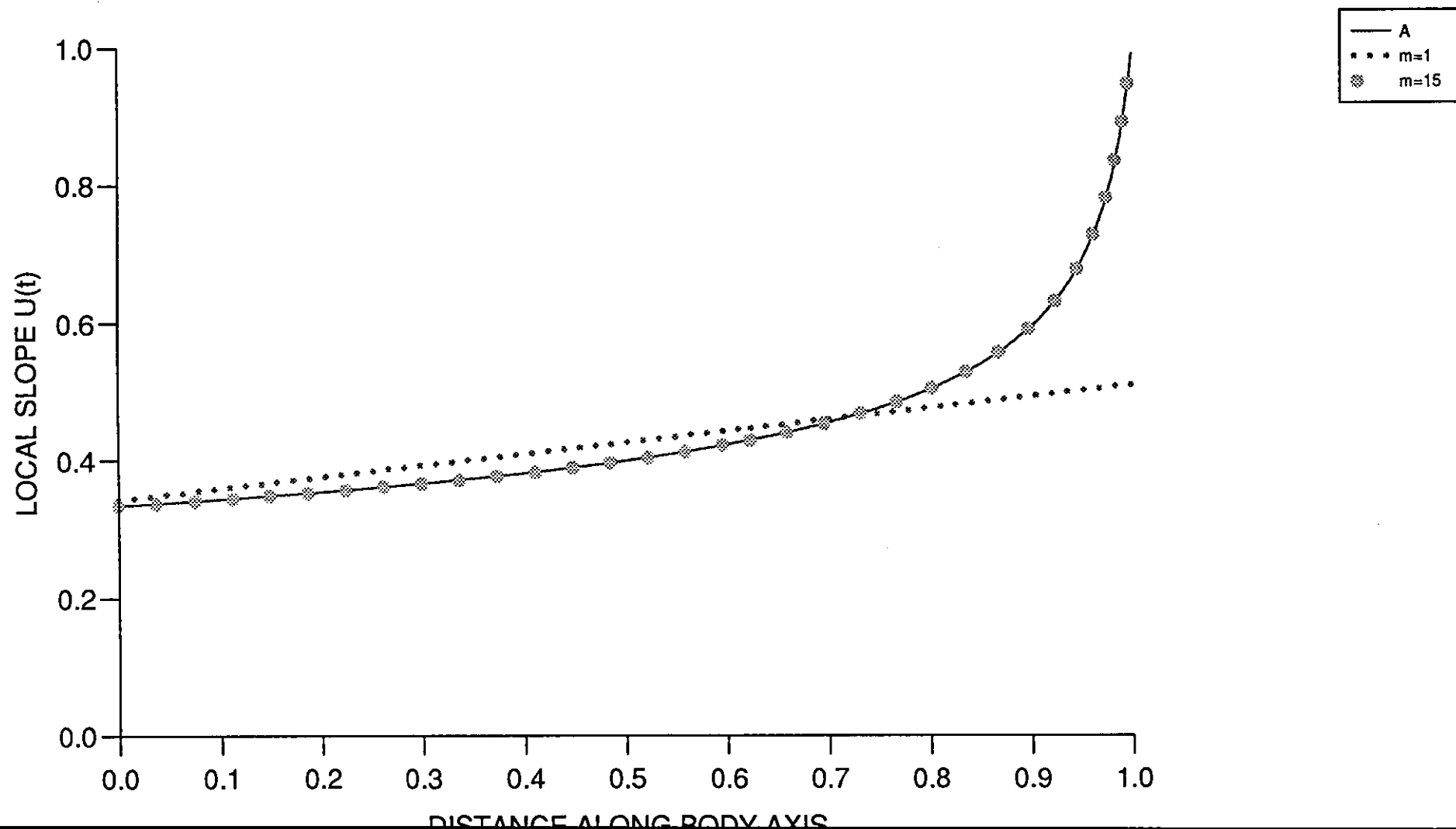


Figure (5.4.22)

FLETCHER REEVES

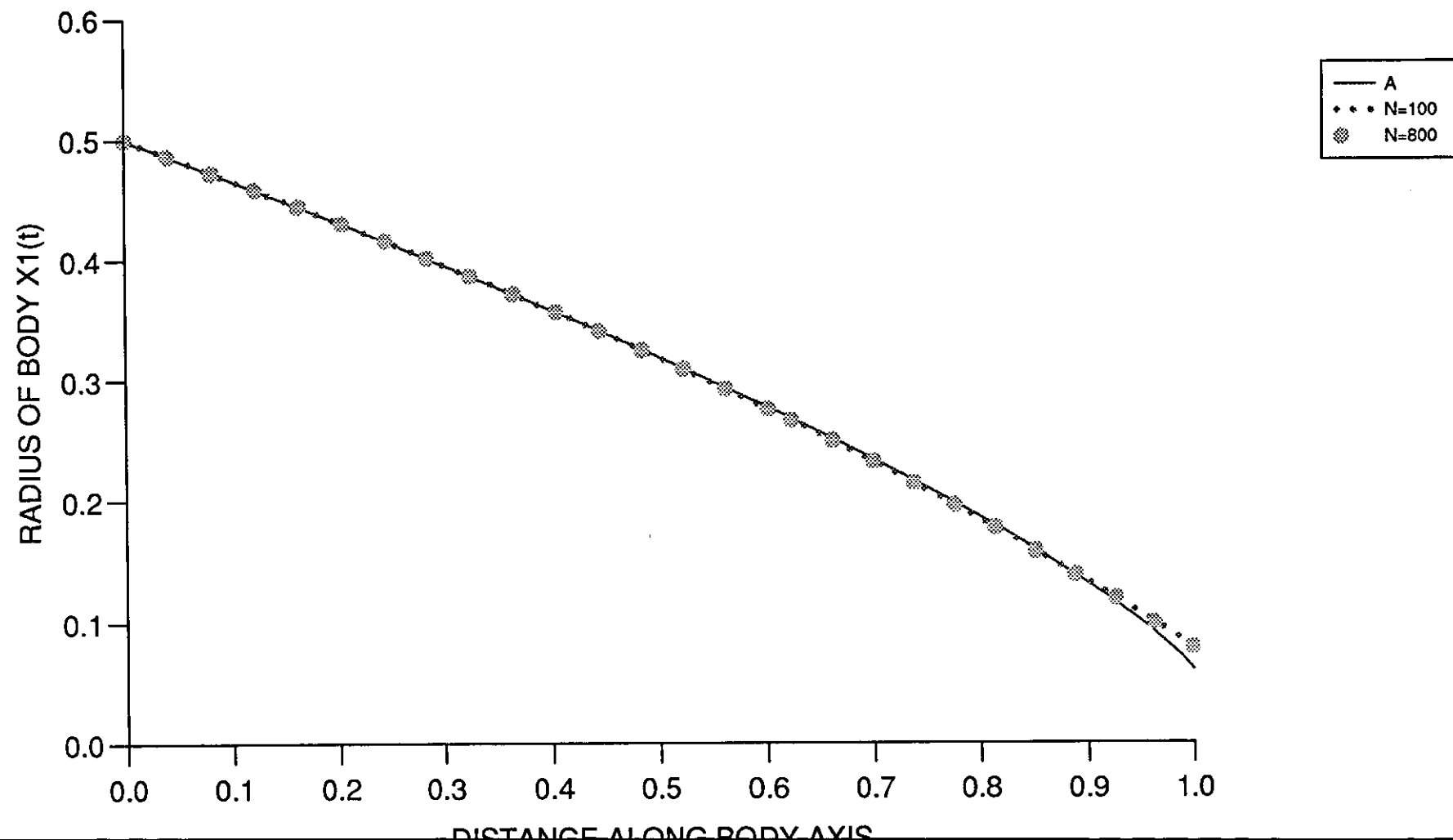


Figure (5.4.23)

FLETCHER REEVES

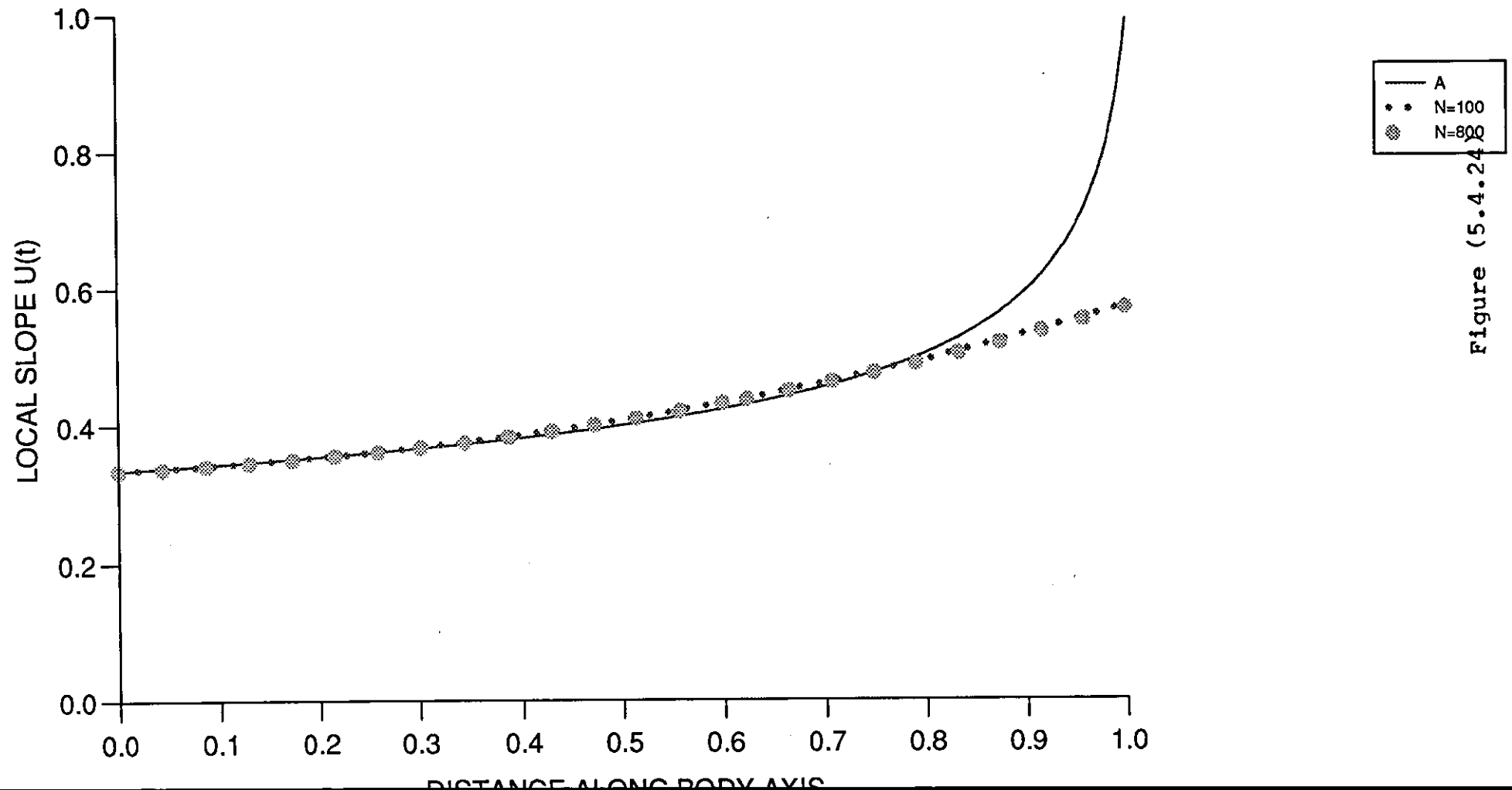


Figure (5.4.24)

FLETCHER REEVES

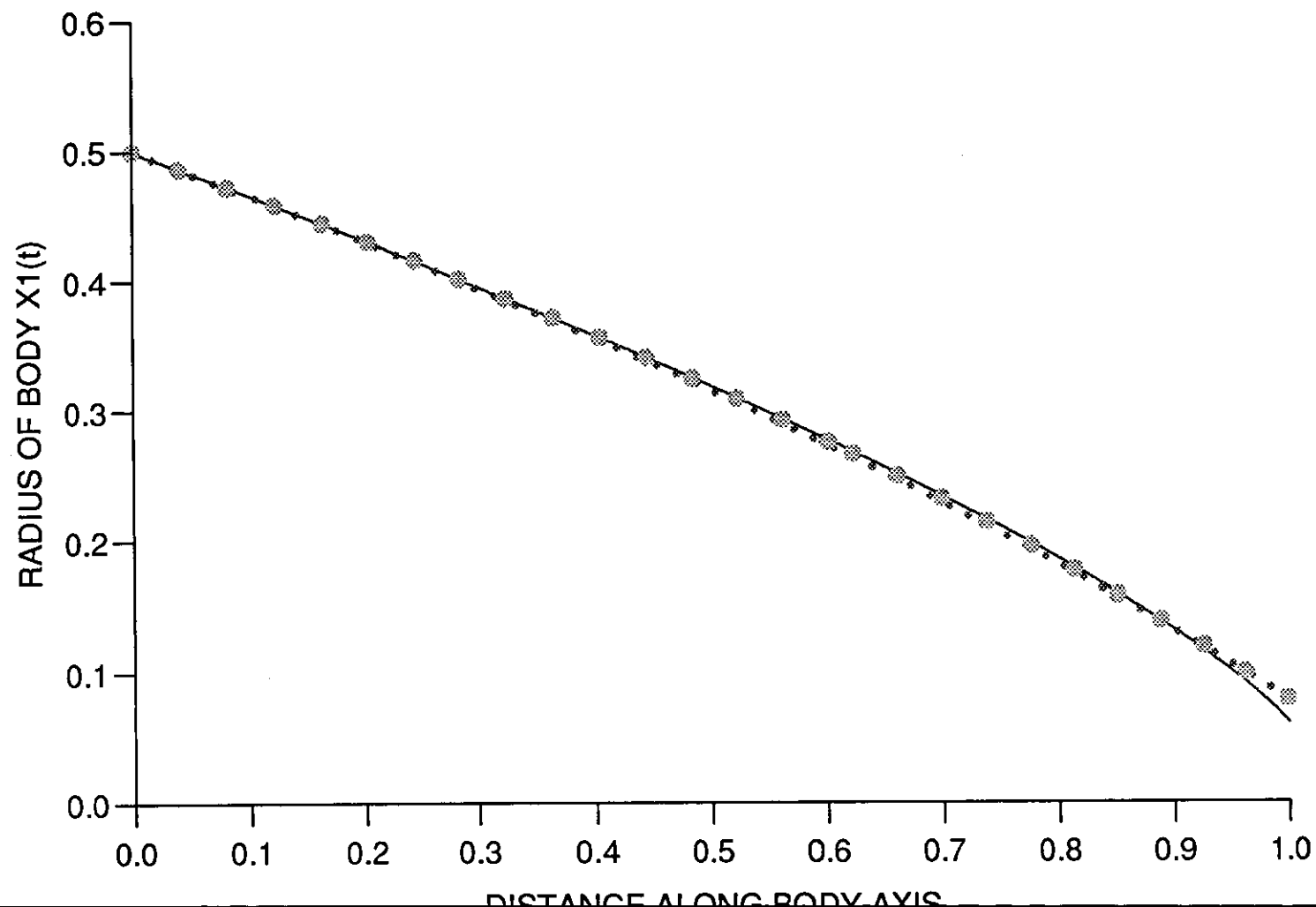


Figure (5.4.25)

FLETCHER REEVES

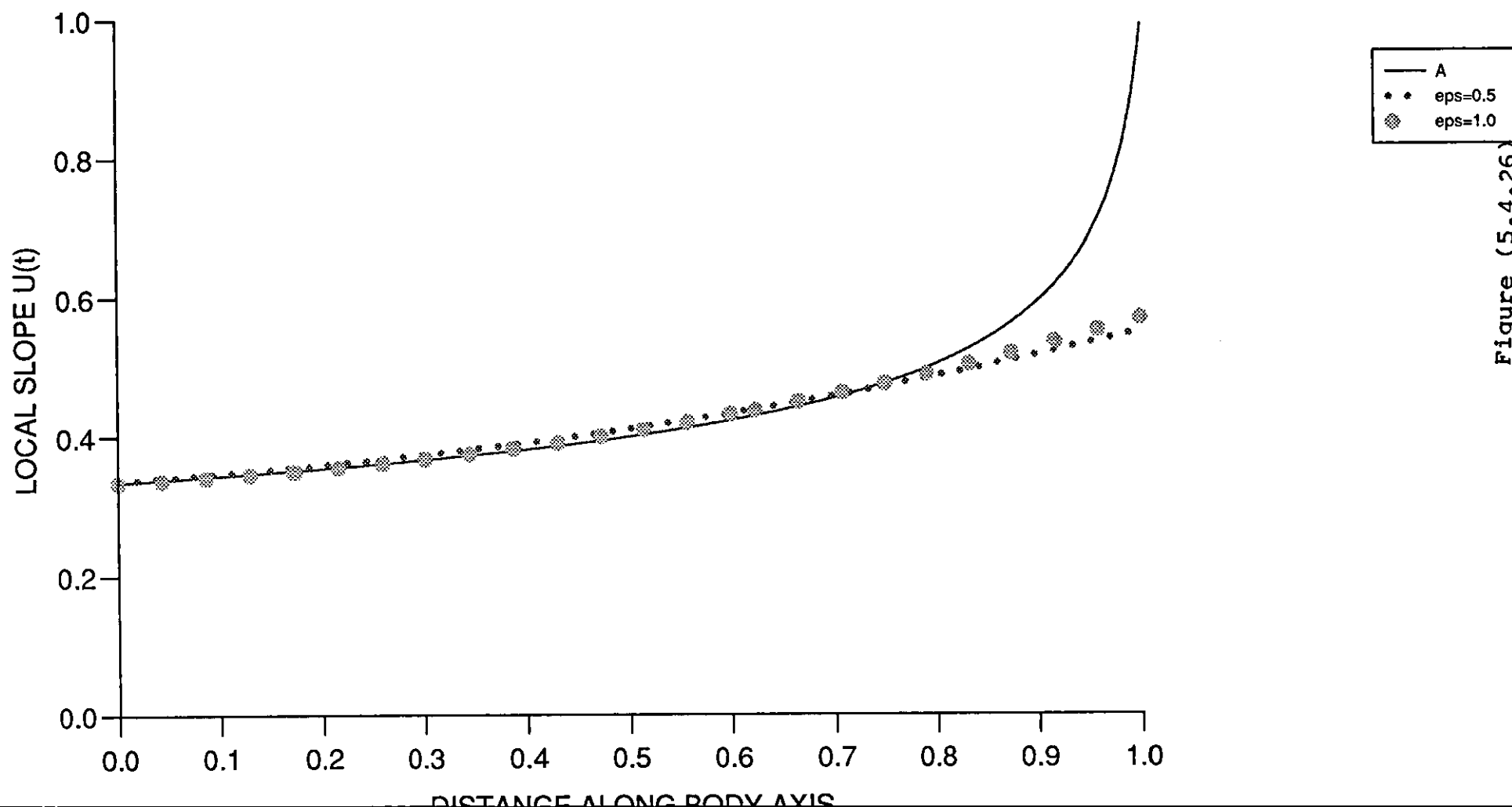


Figure (5.4.26)

FLETCHER REEVES

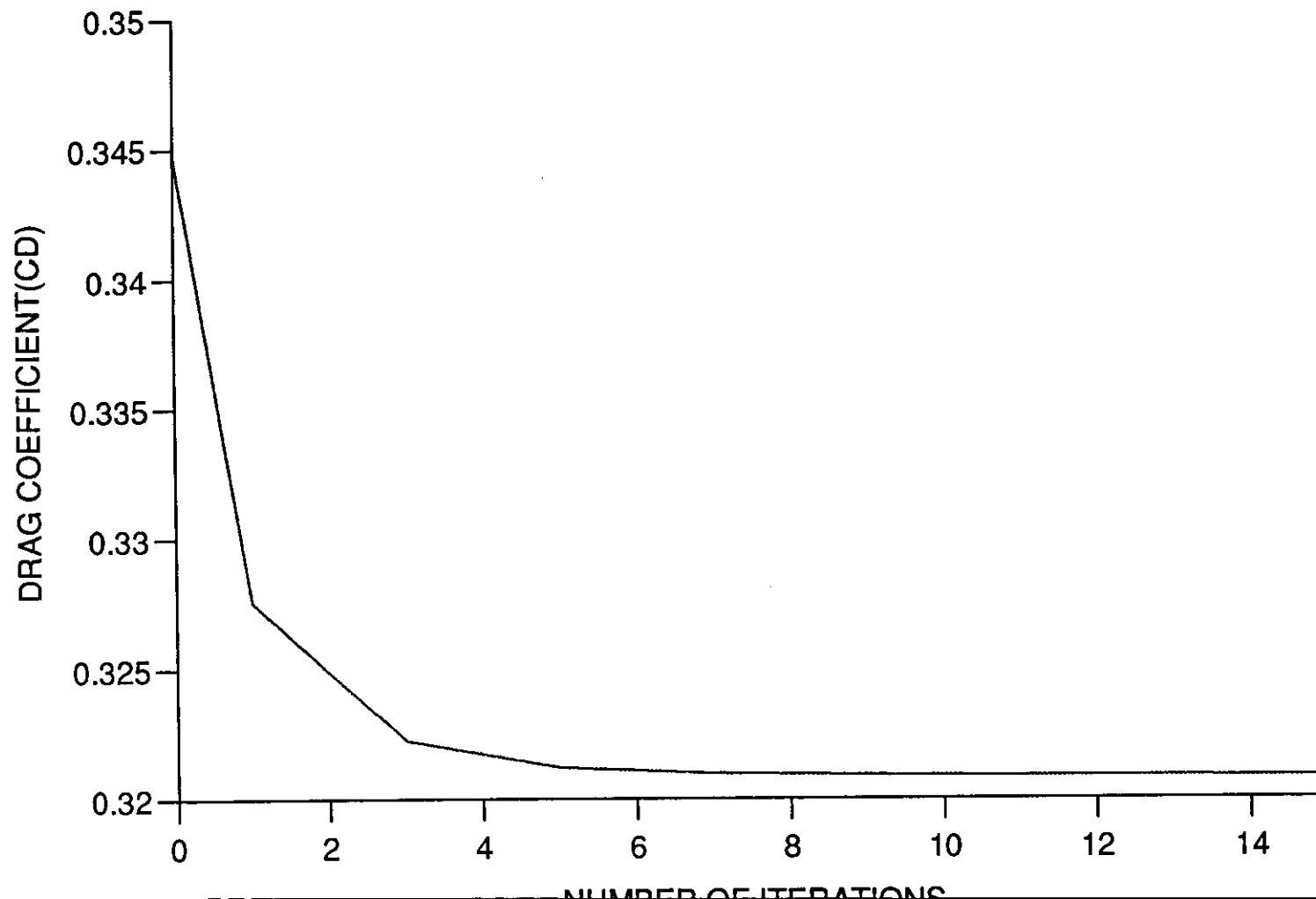


Figure (5.4.27)

FLETCHER REEVES

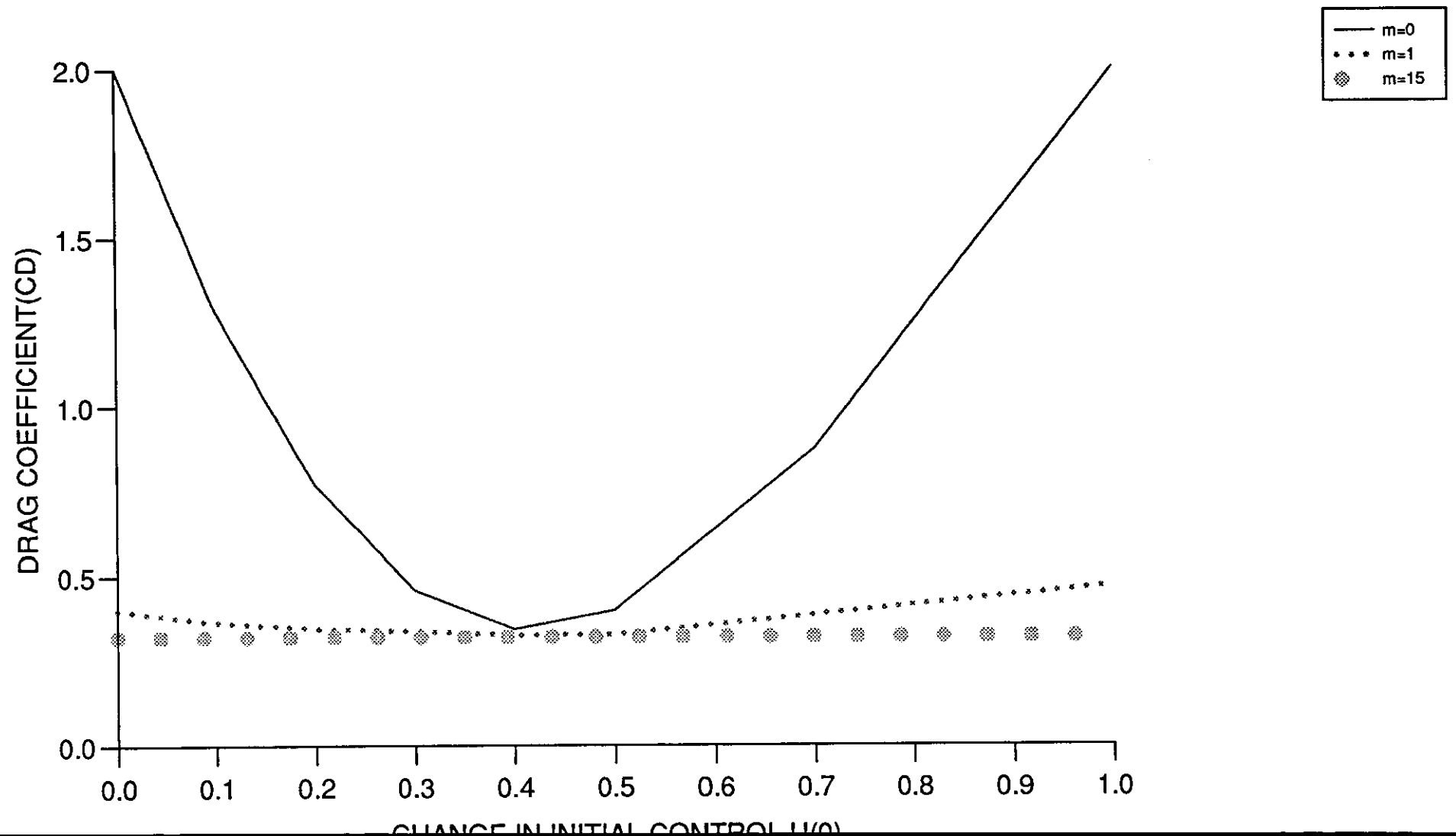


Figure (5.4.28)

FLETCHER REEVES

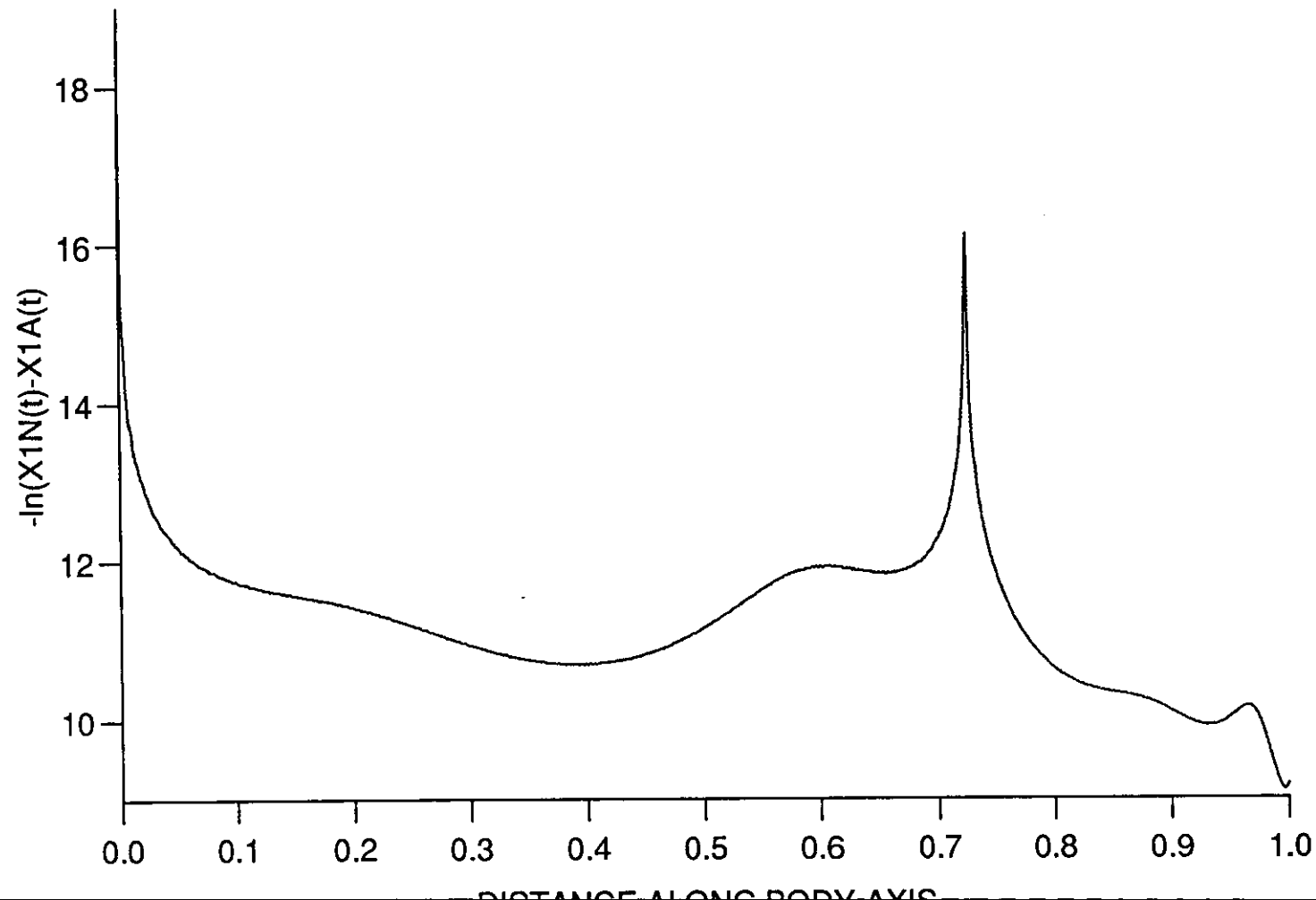


Figure (5.4.29)

FLETCHER REEVES

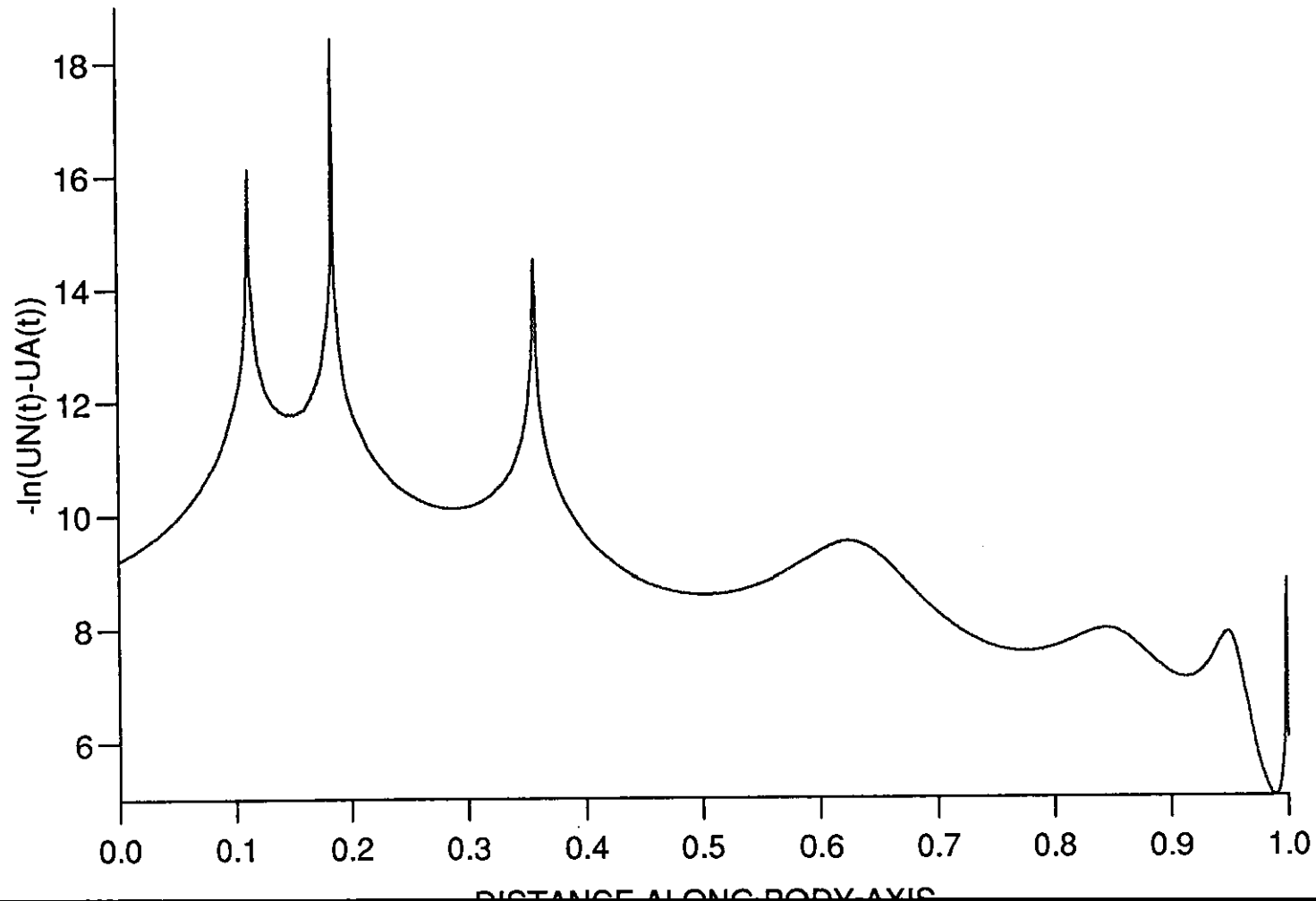


Figure (5.4.30)

POLAK RIBEIRE

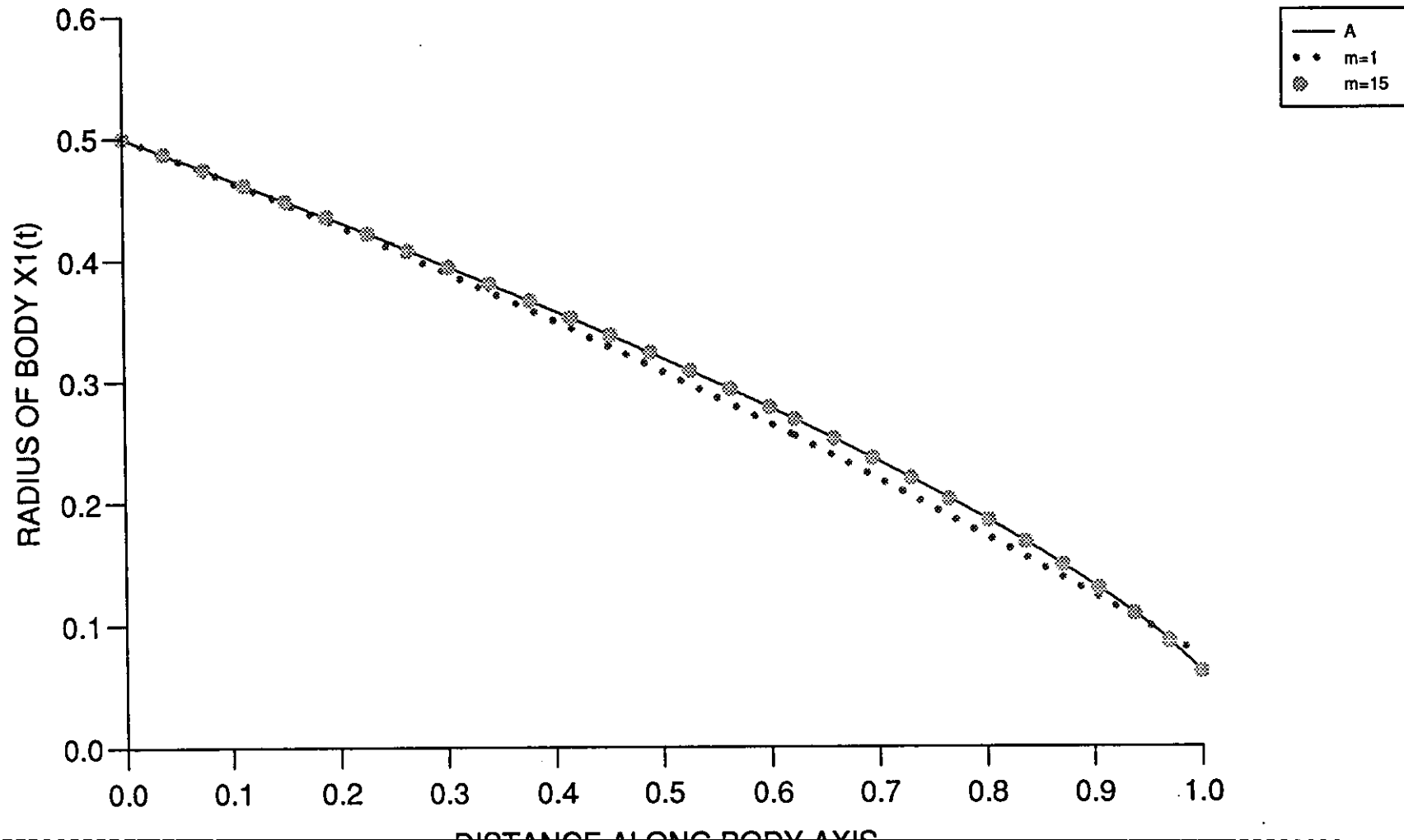


Figure (5.4.31)

POLAK RIBEIRE

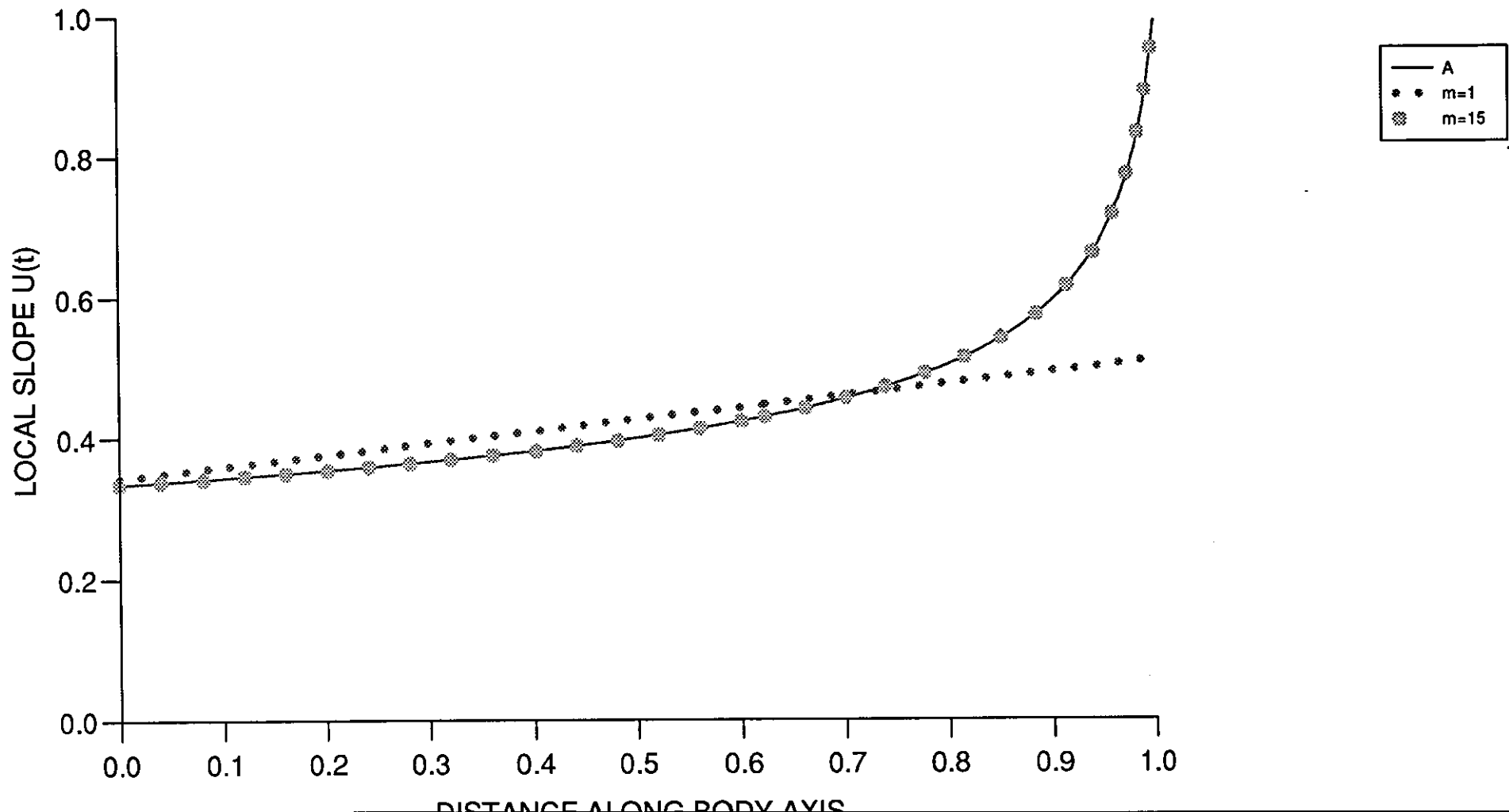


Figure (5.4.32)

POLAK RIBEIRE

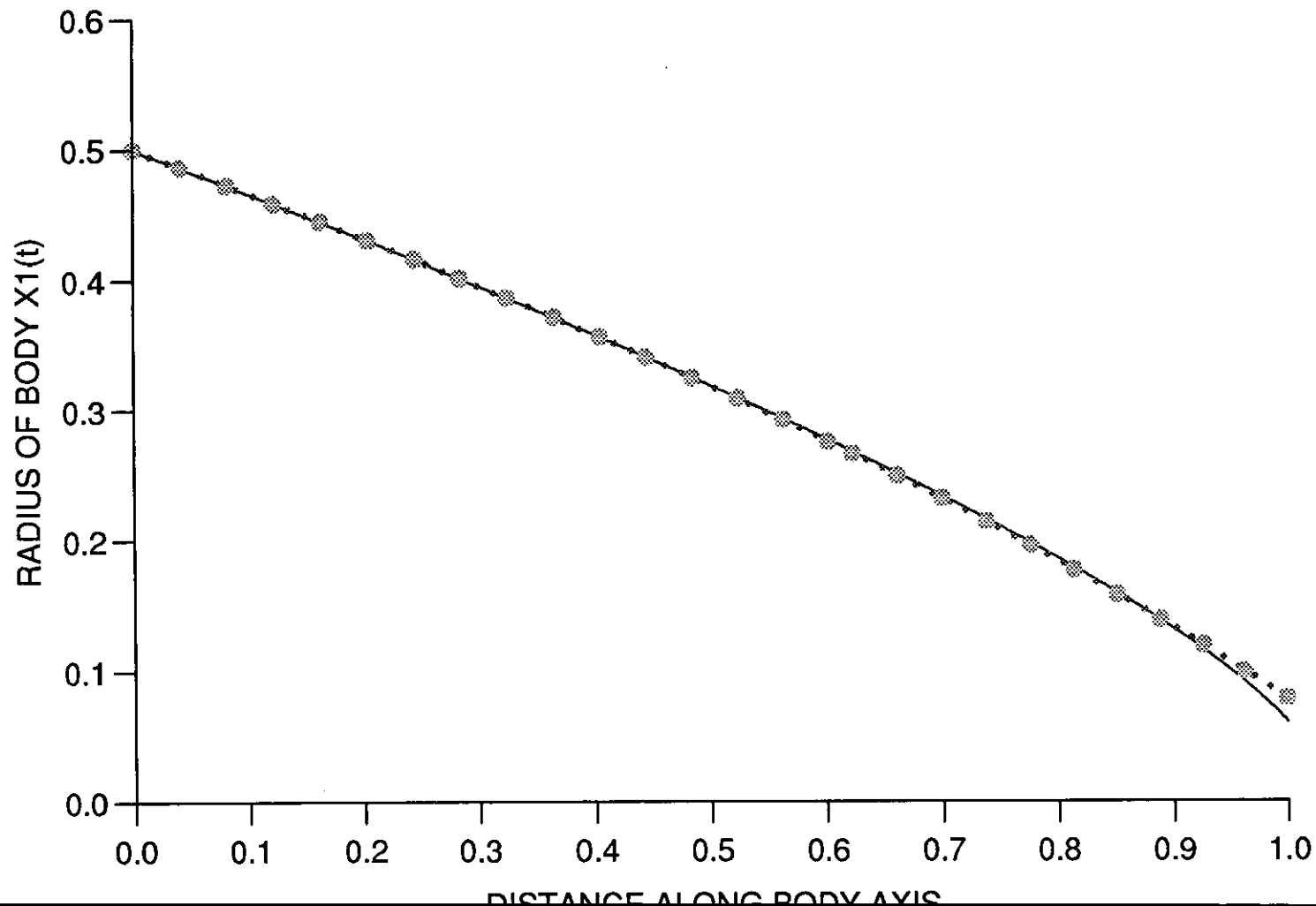


Figure (5.4.33)

POLAK RIBEIRE

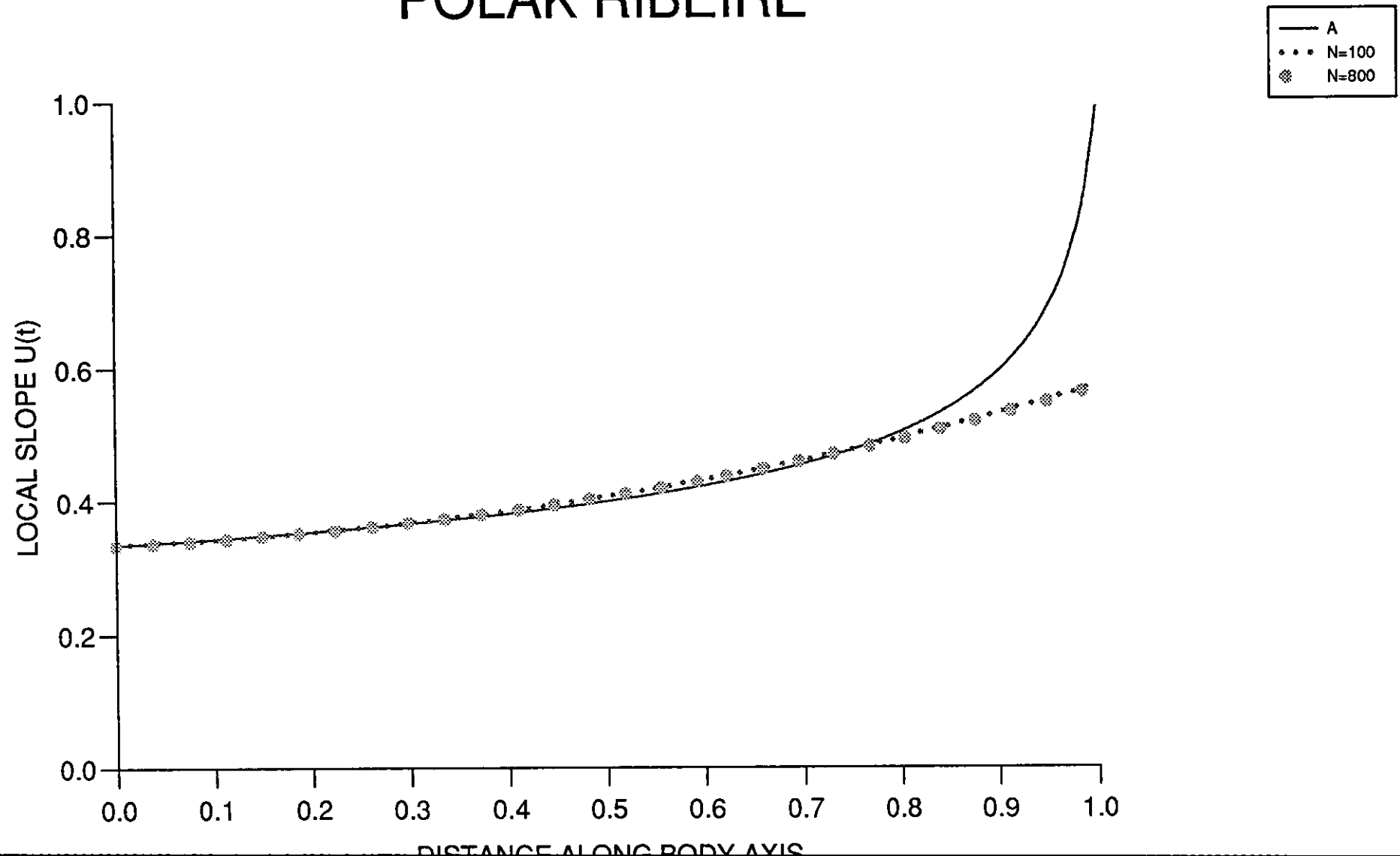


Figure (5.4.34)

POLAK RIBEIRE

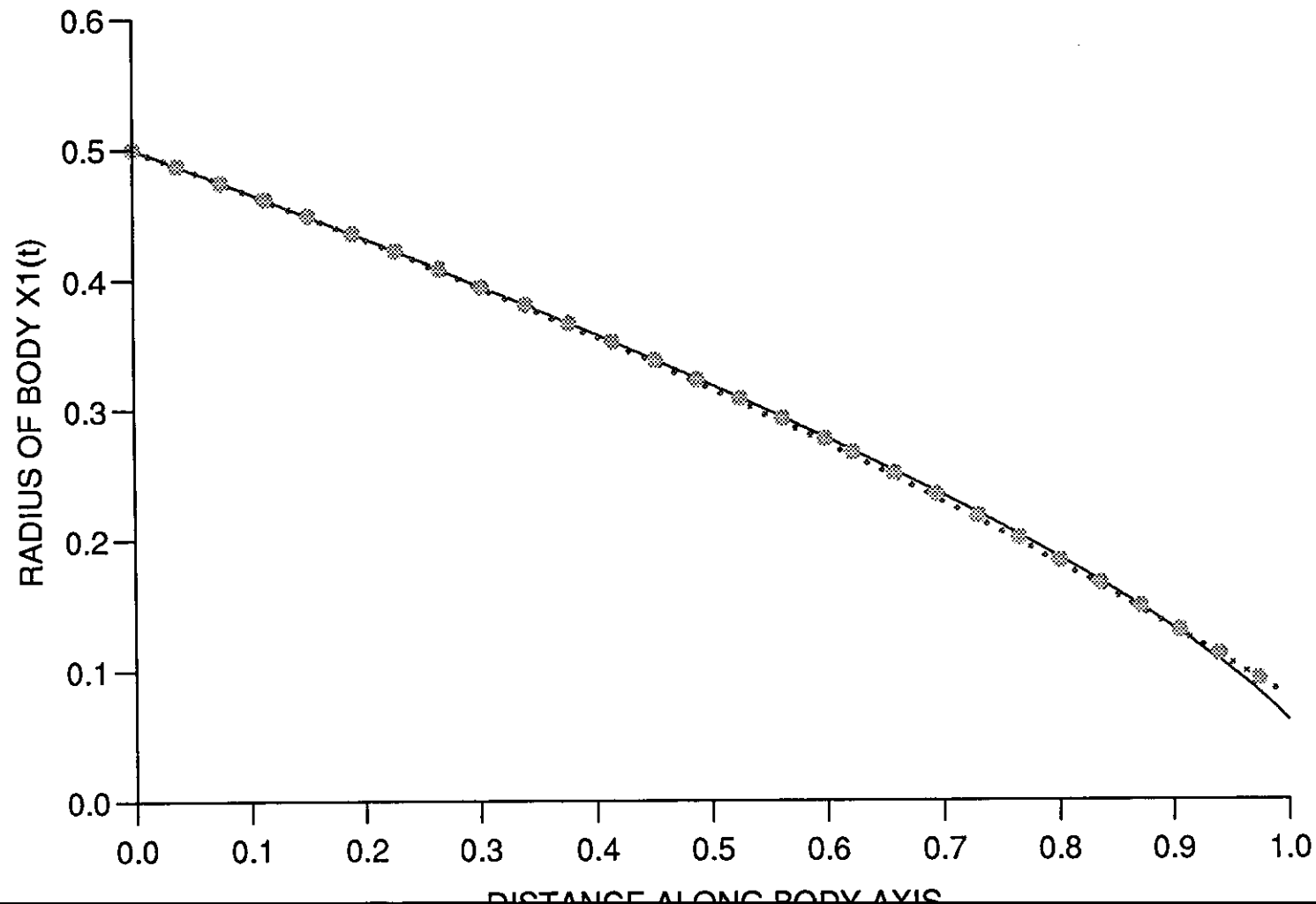


Figure (5.4.35)

POLAK RIBEIRE

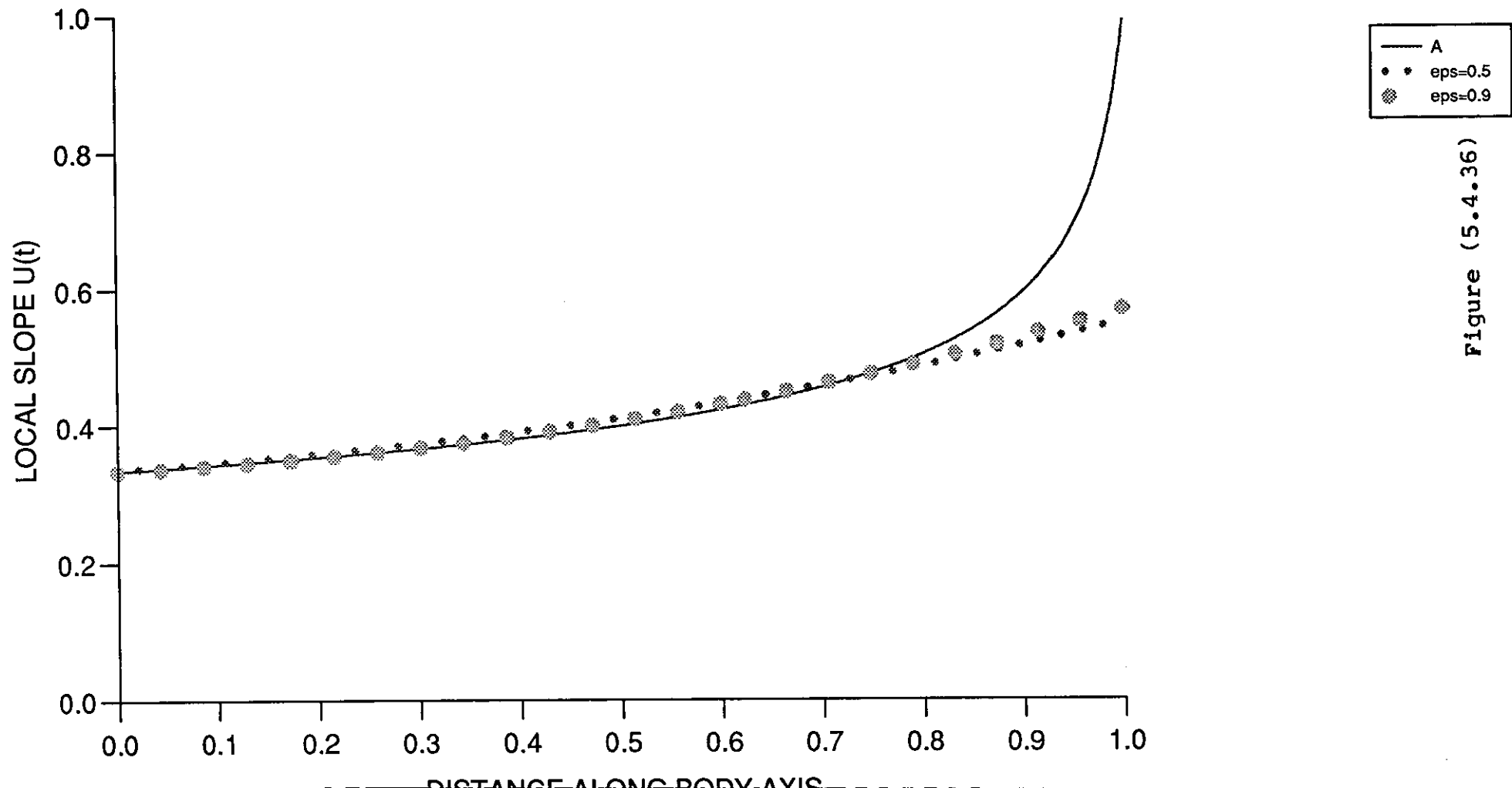


Figure (5.4.36)

POLAK RIBEIRE

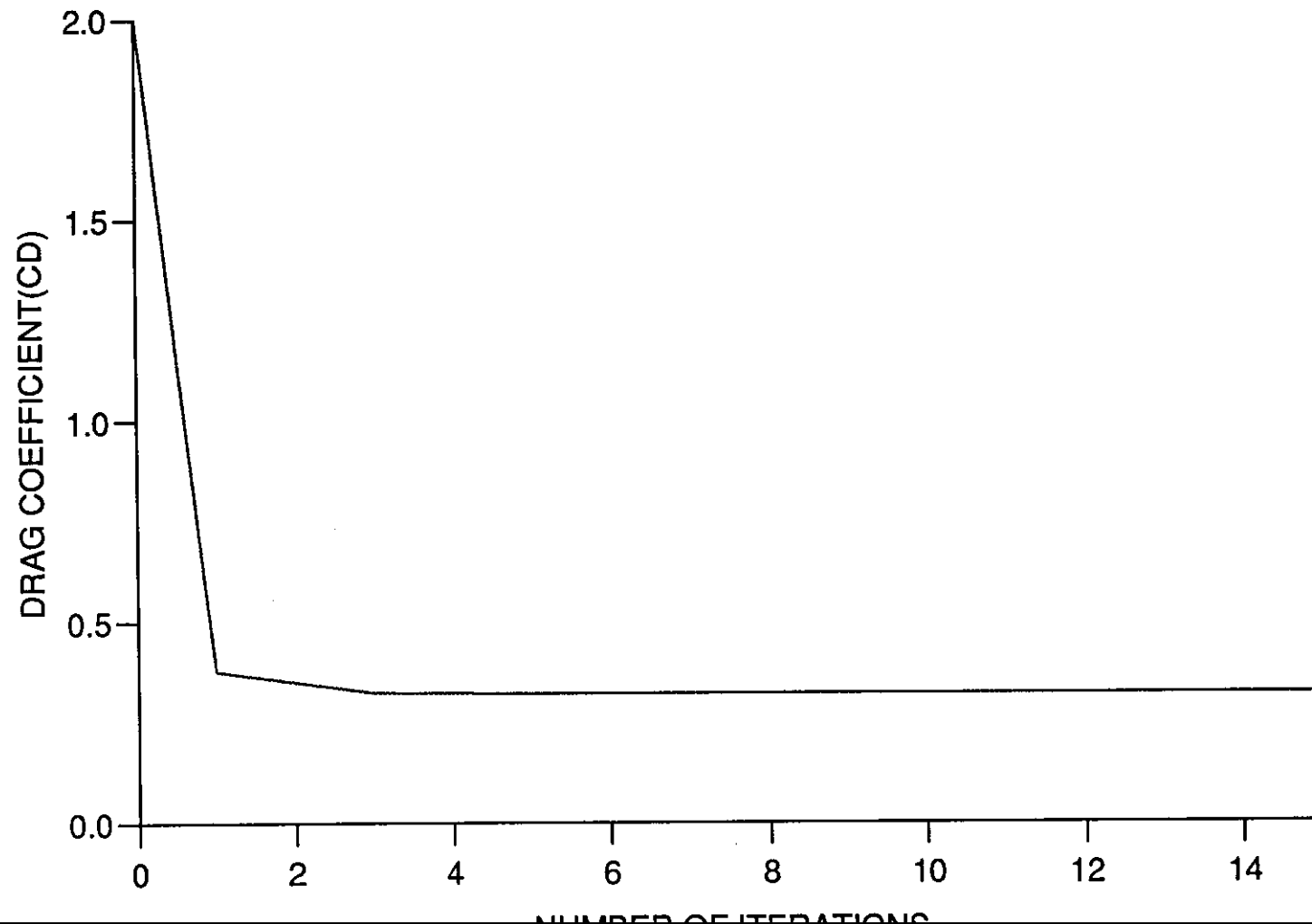


Figure (5.4.37)

POLAK RIBEIRE

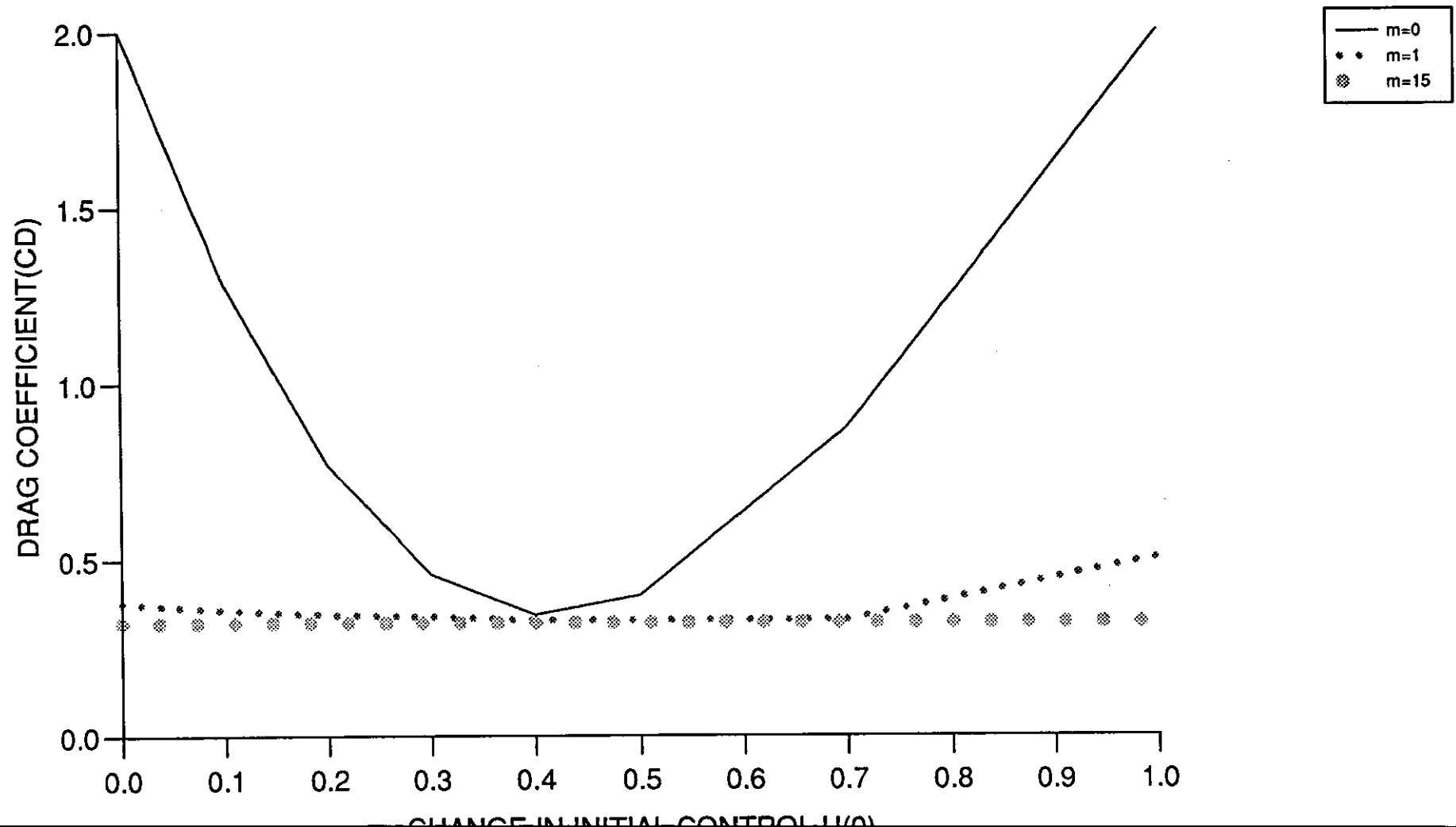


Figure (5.4.38)

POLAK RIBEIRE

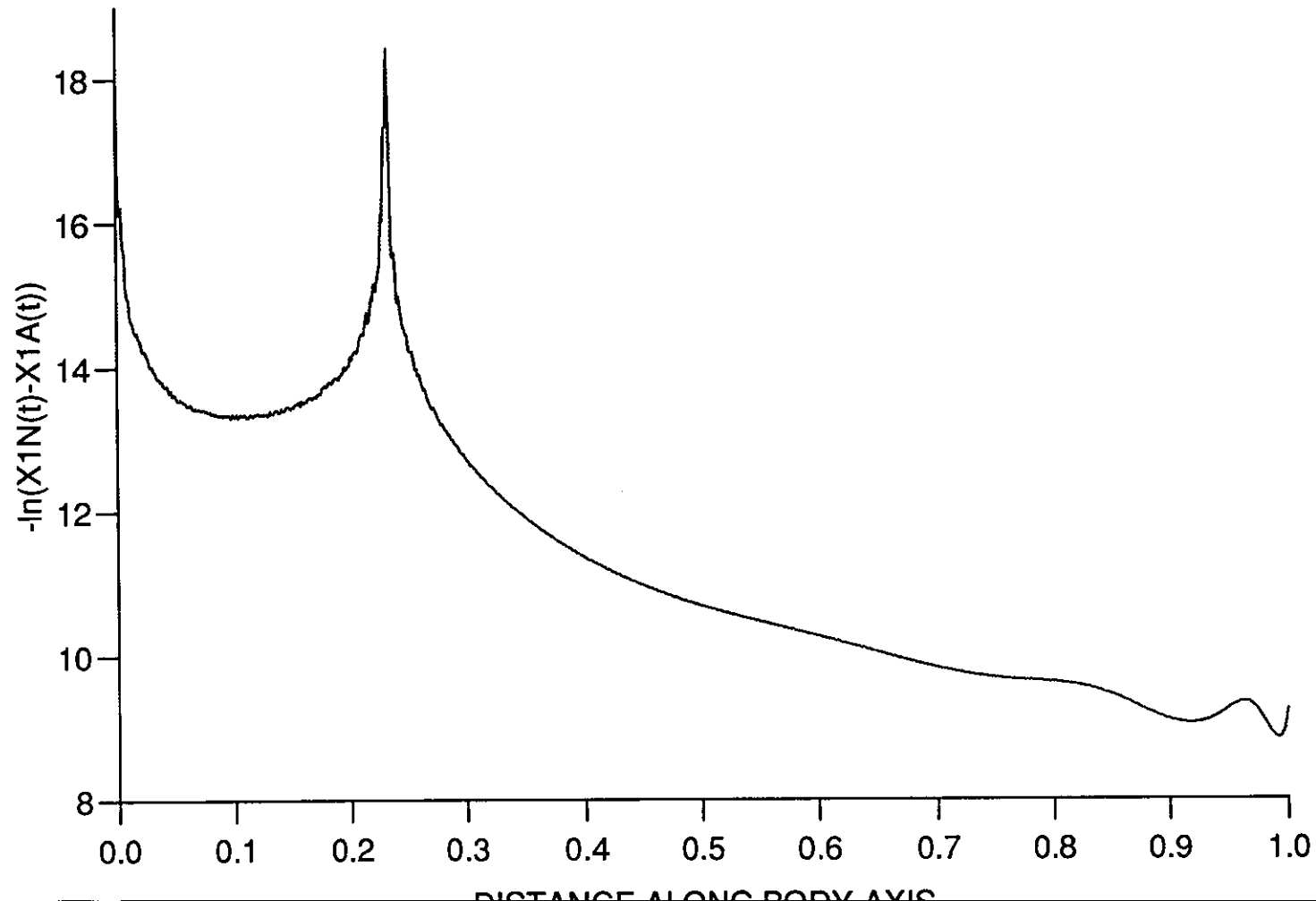


Figure (5.4.39)

POLAK RIBEIRE

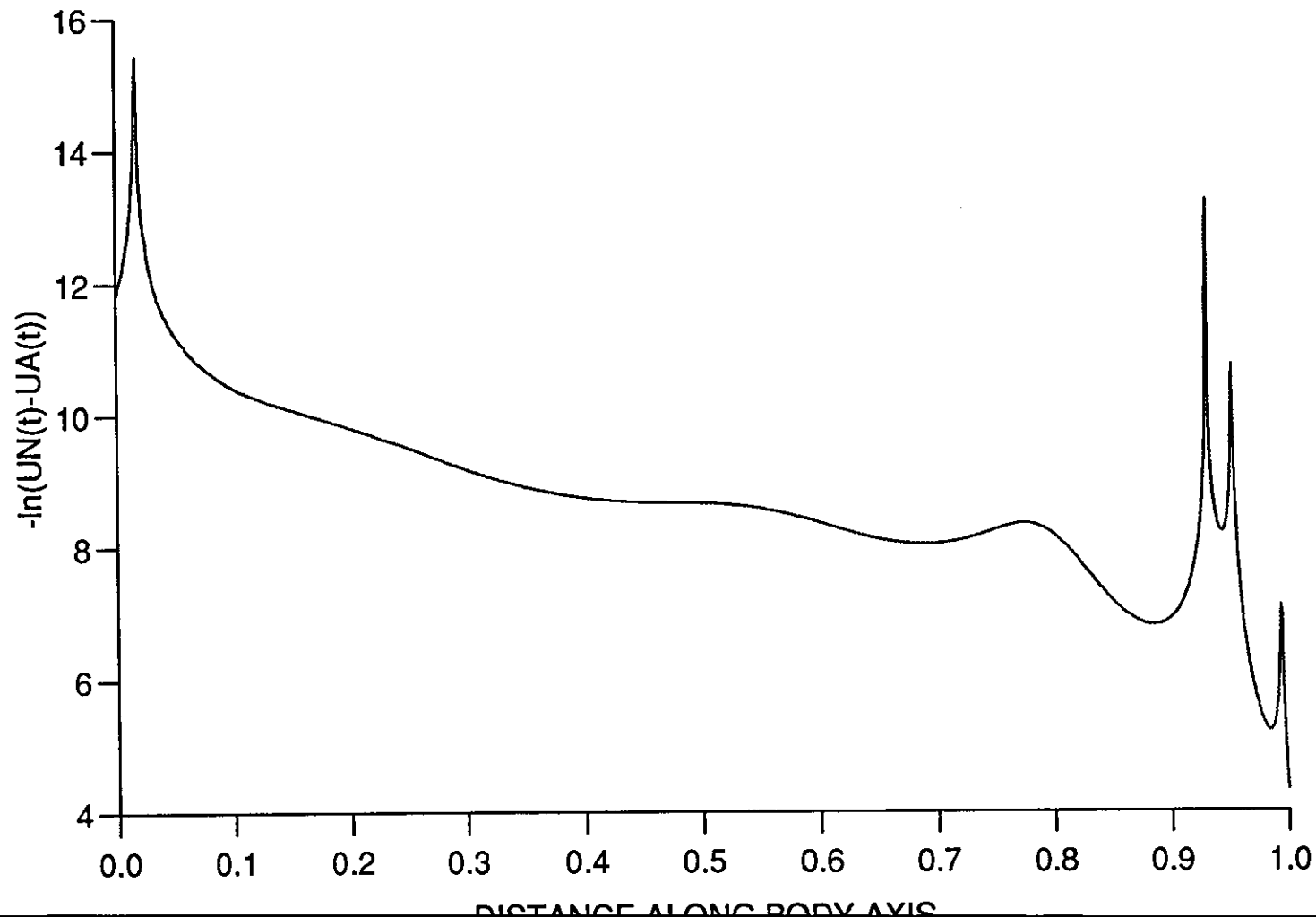


Figure (5.4.40)

HYBRID 1

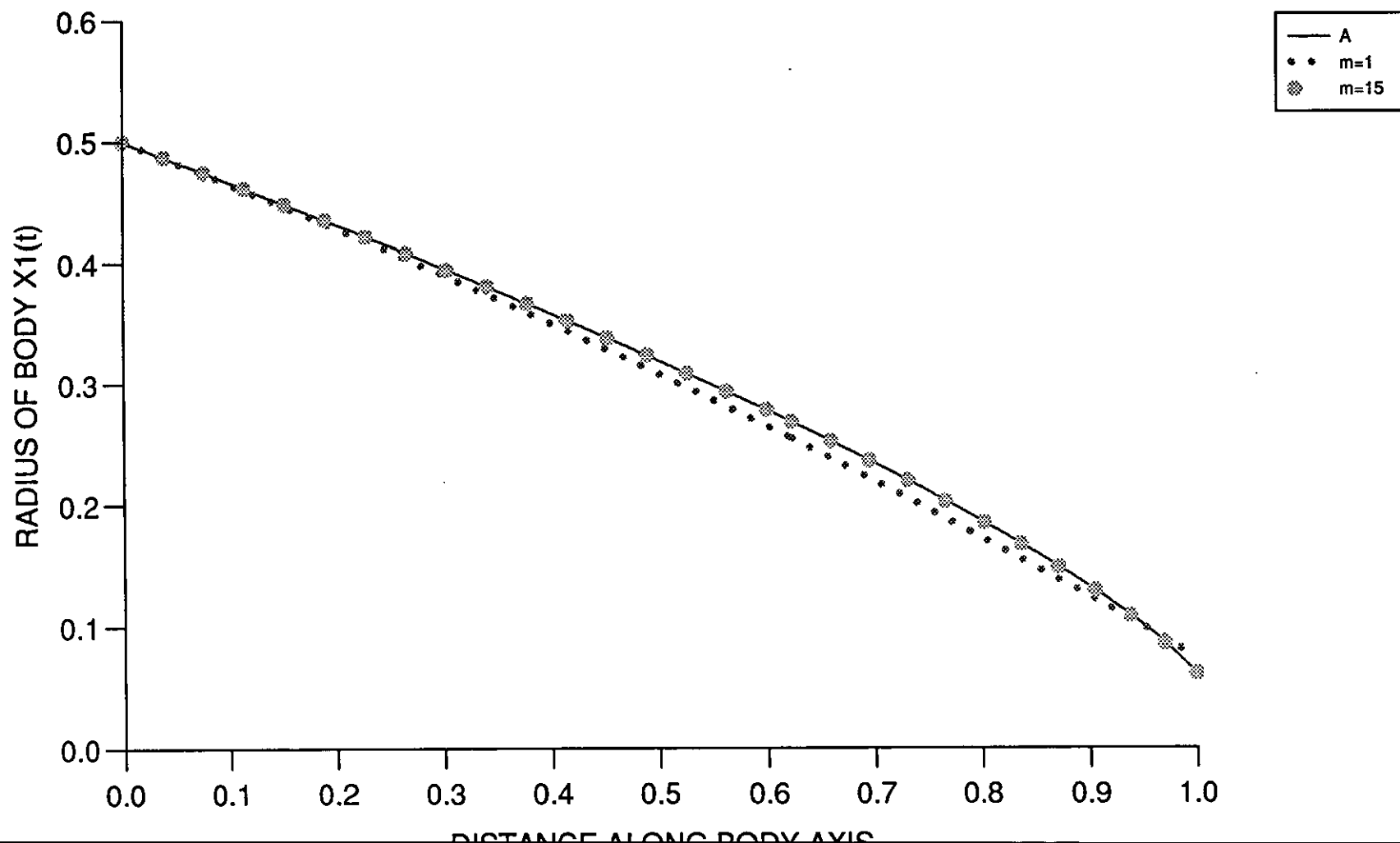


Figure (5.4.41)

HYBRID 1

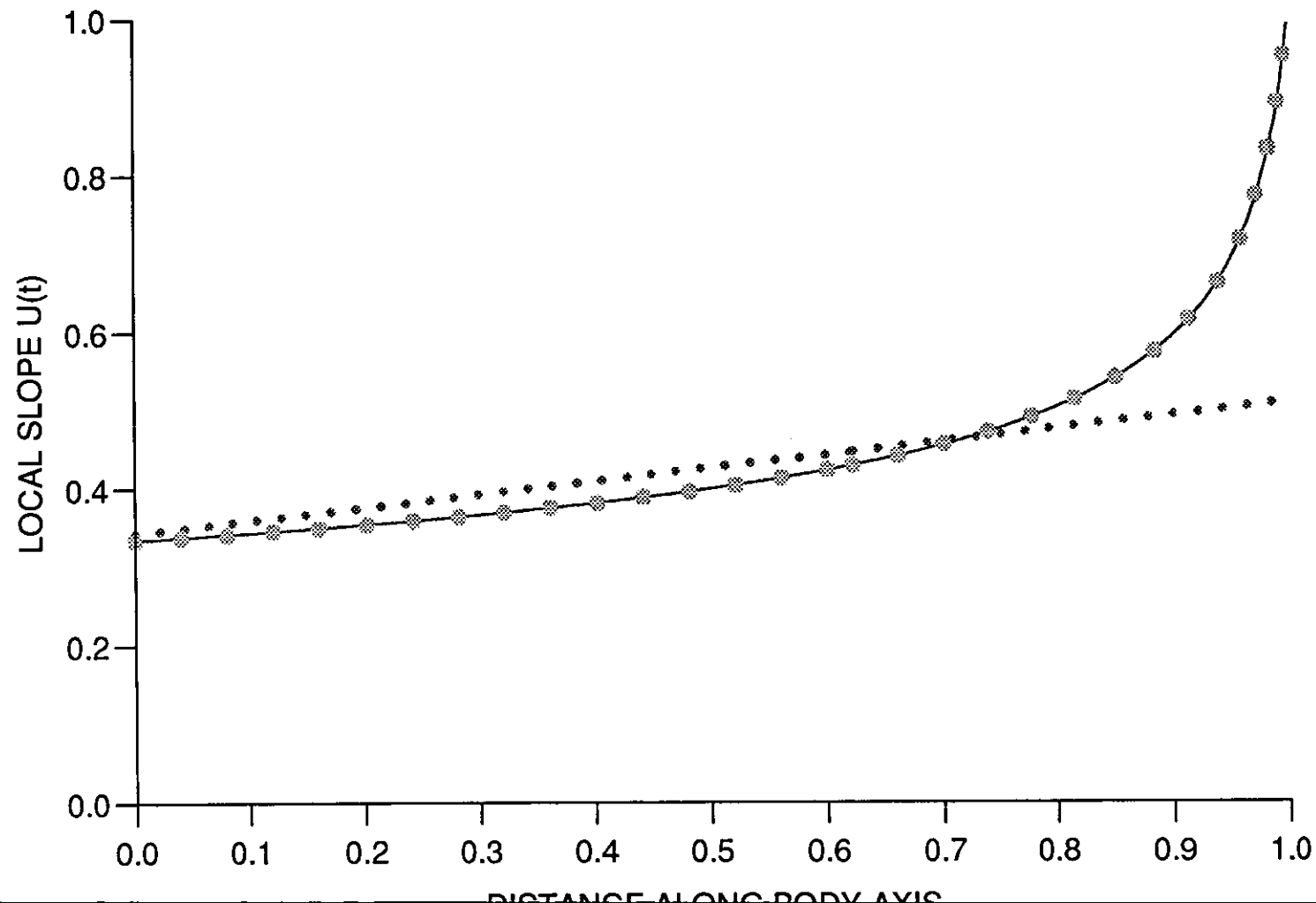


Figure (5.4.42)

HYBRID 1

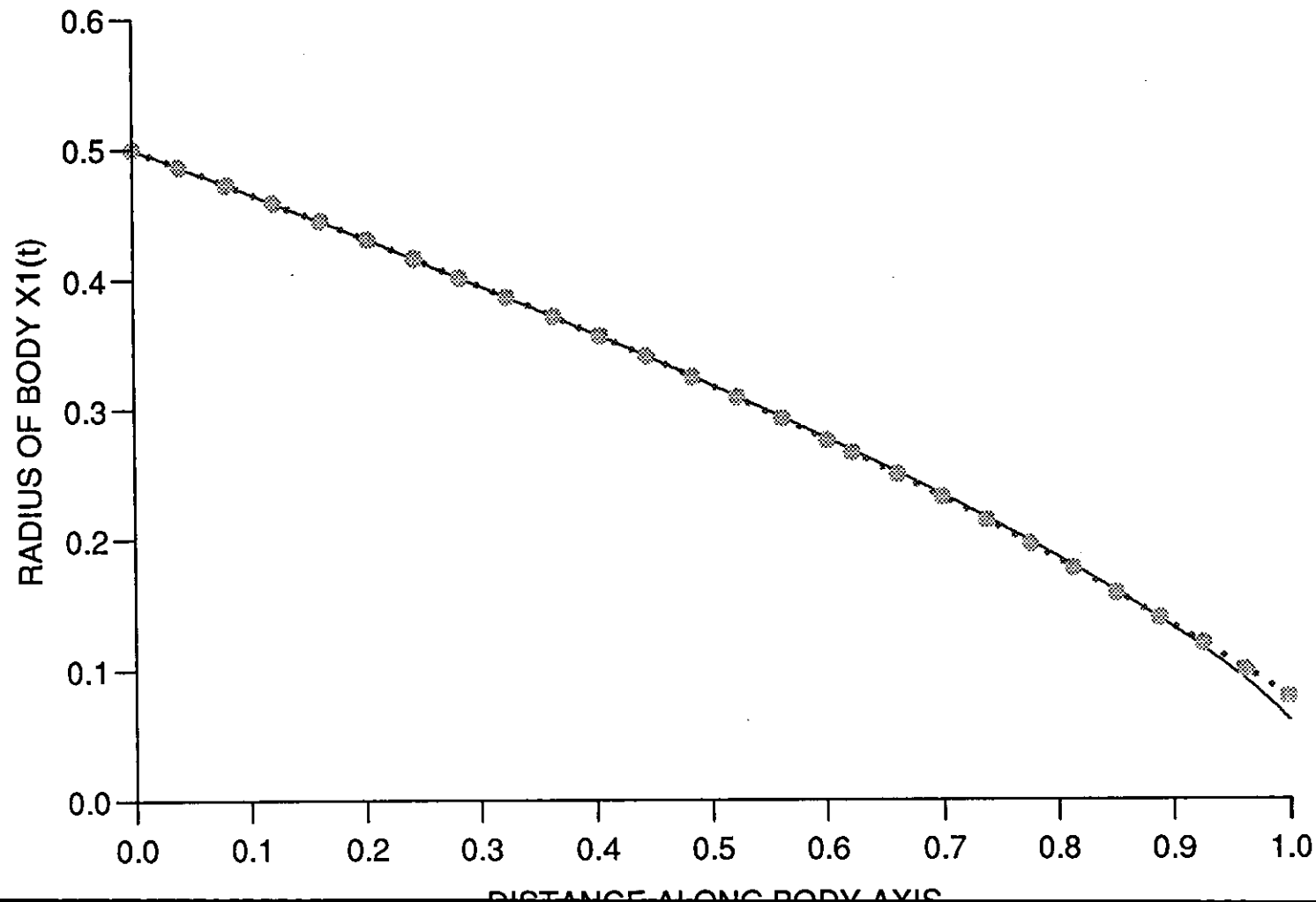


Figure (5.4.43)

HYBRID 1

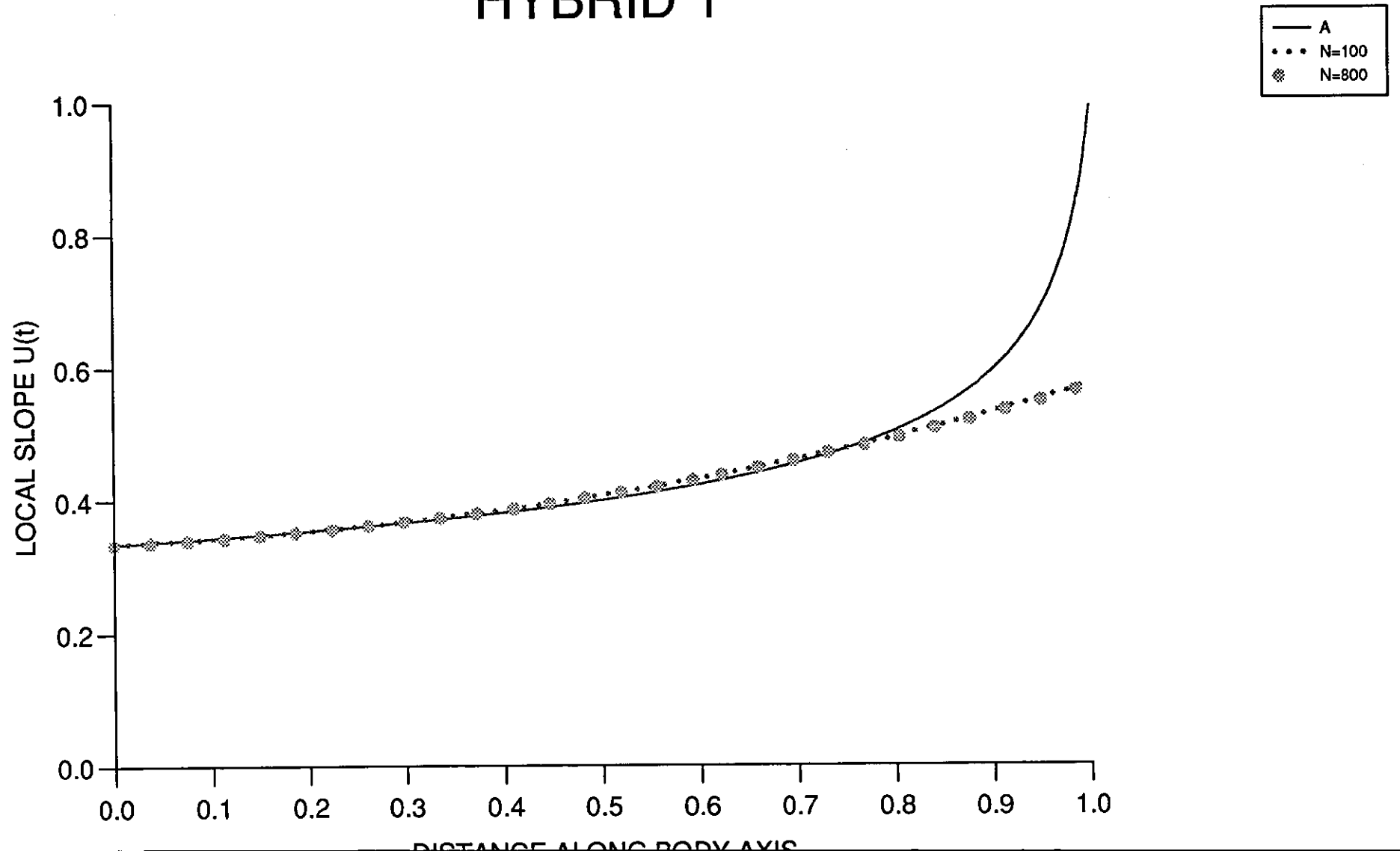


Figure (5.4.44)

HYBRID 1

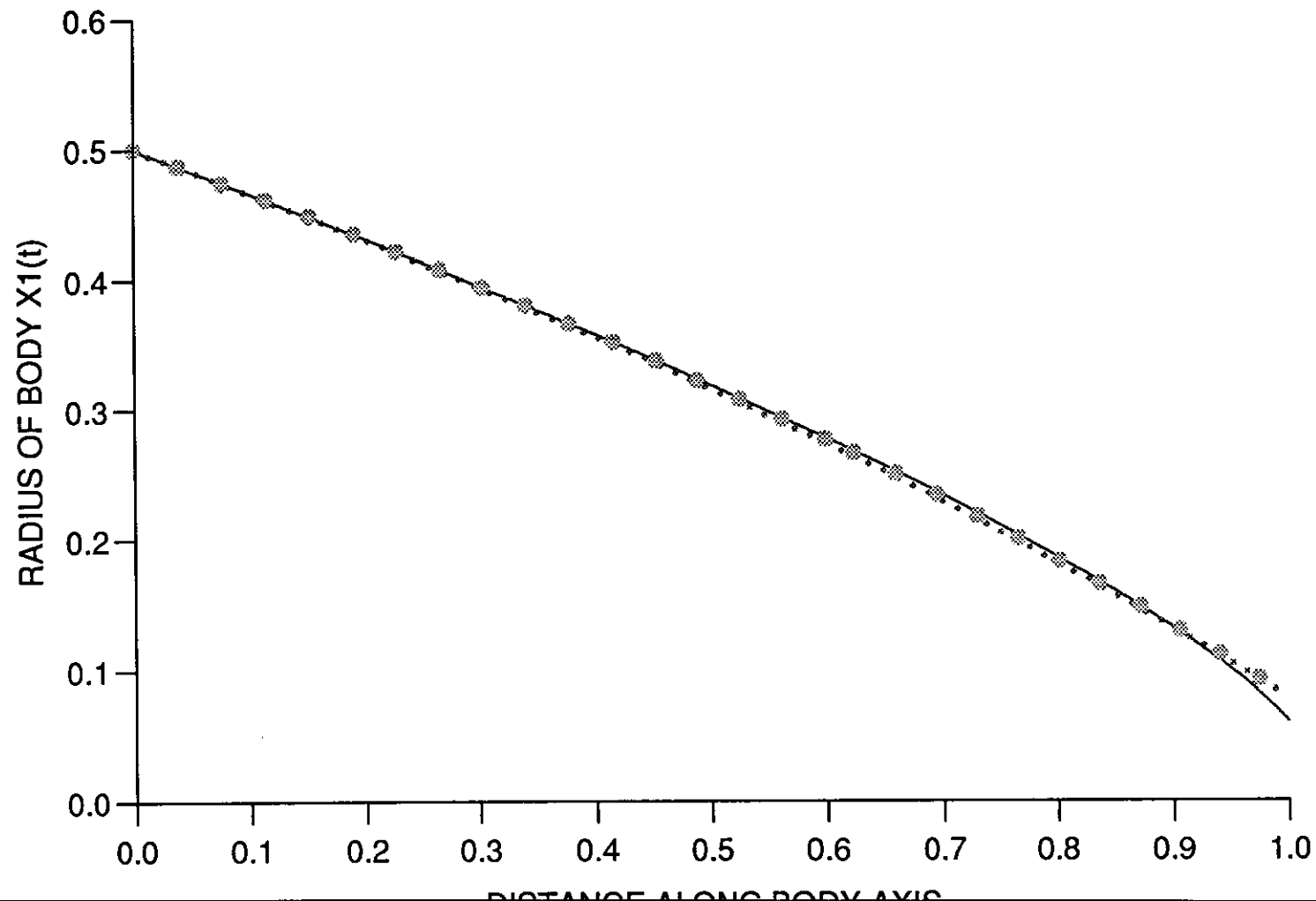


Figure (5.4.45)

HYBRID 1

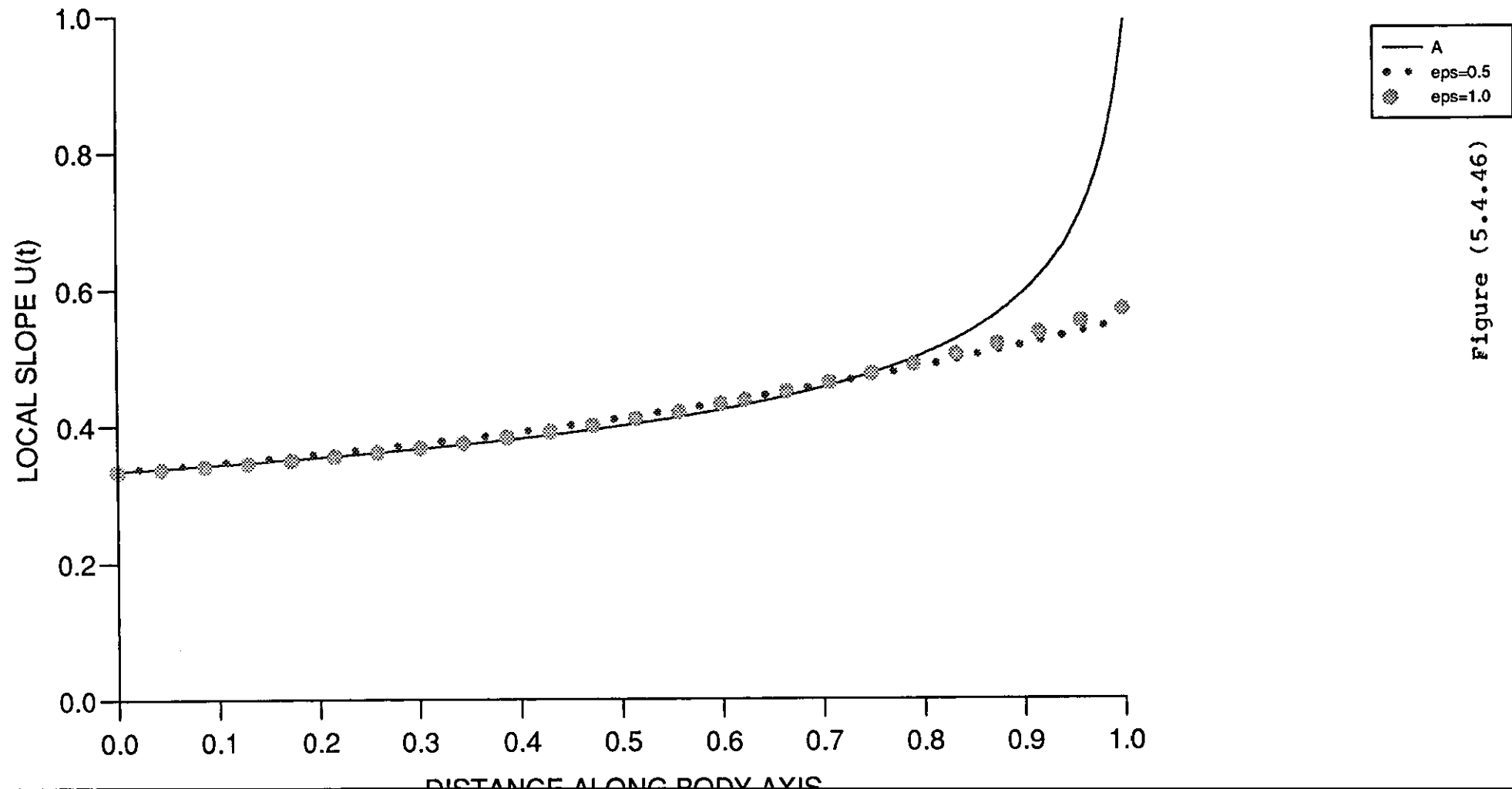


Figure (5.4.46)

HYBRID 1

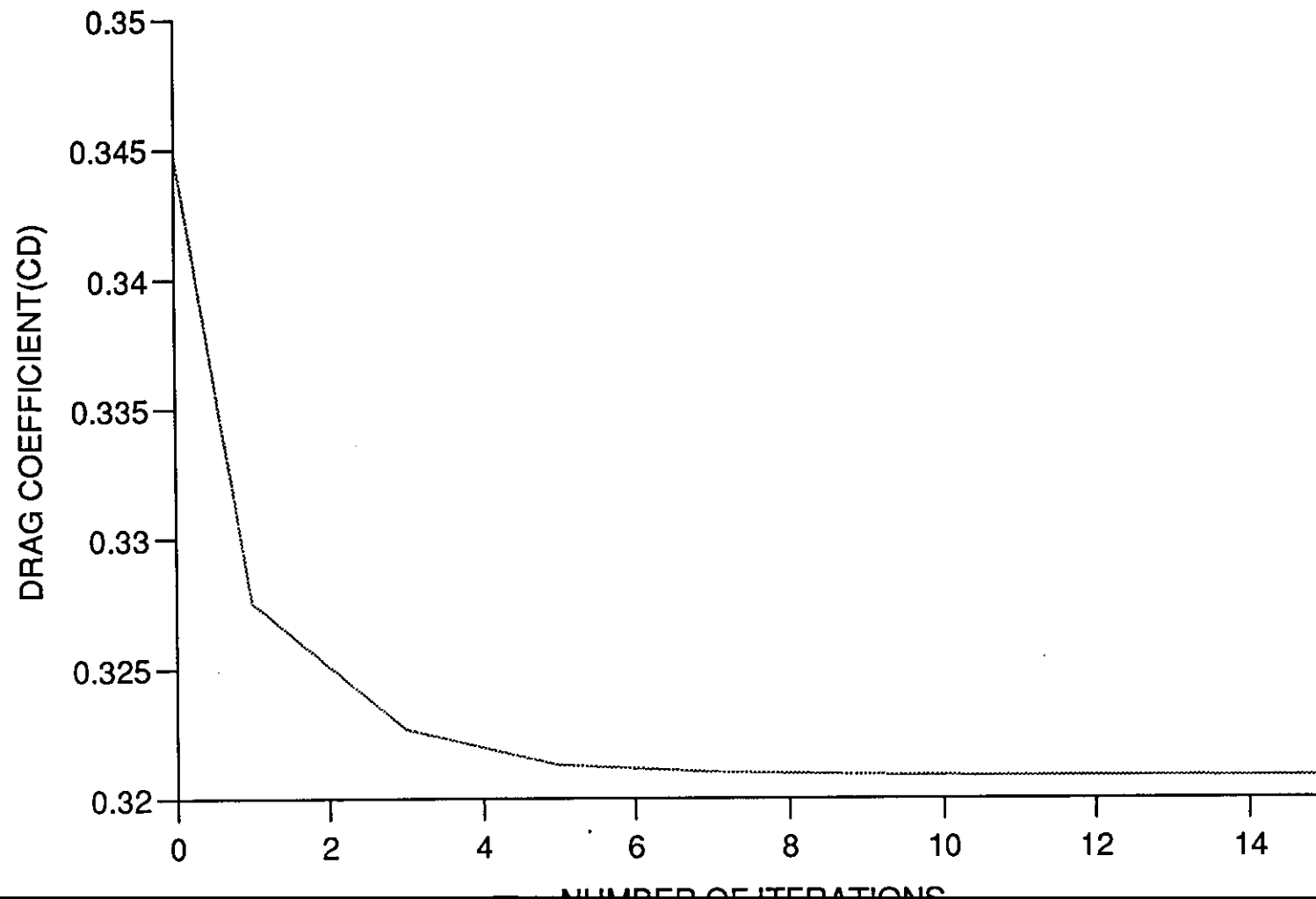


Figure (5.4.47)

HYBRID 1

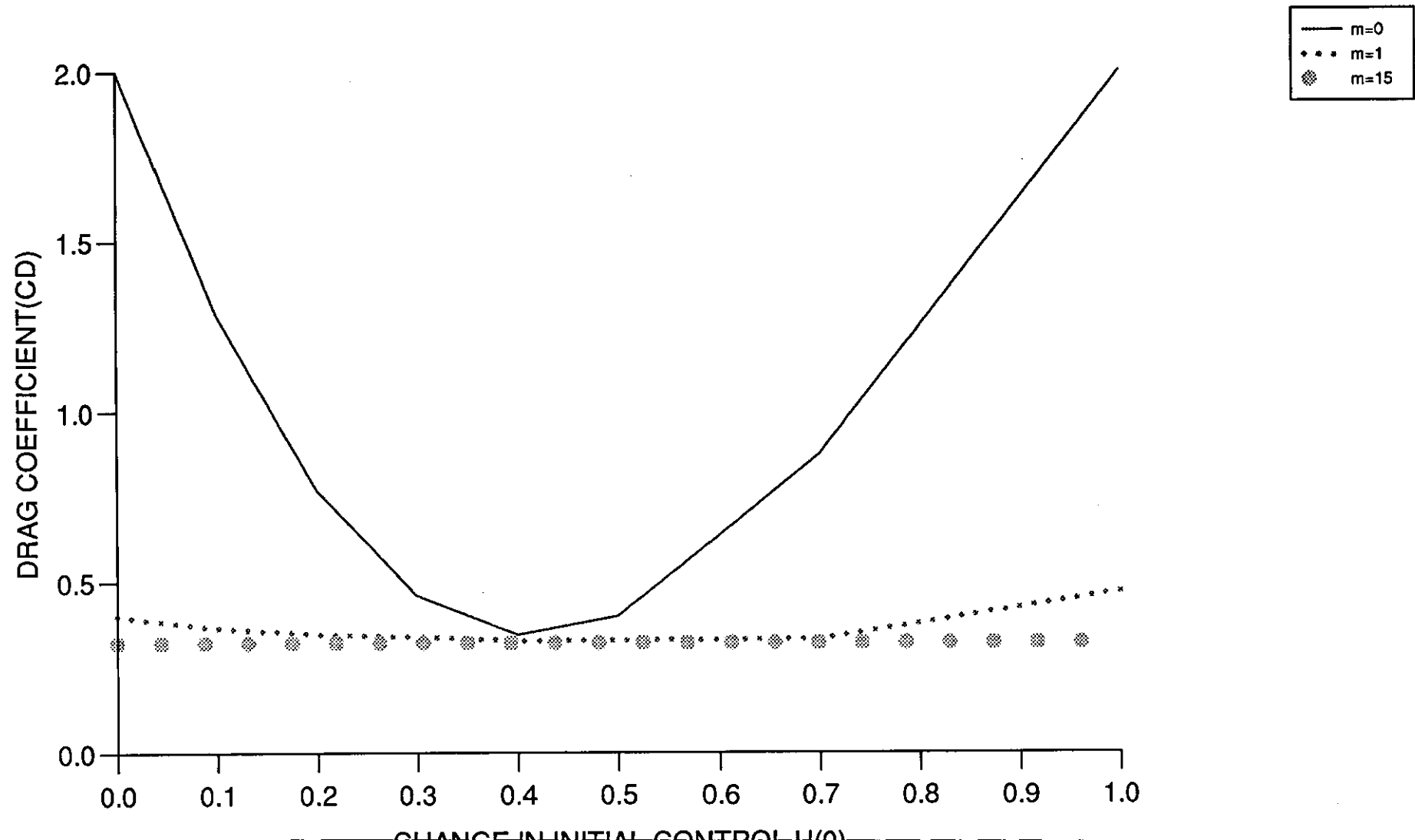


Figure (5.4.48)

HYBRID 1

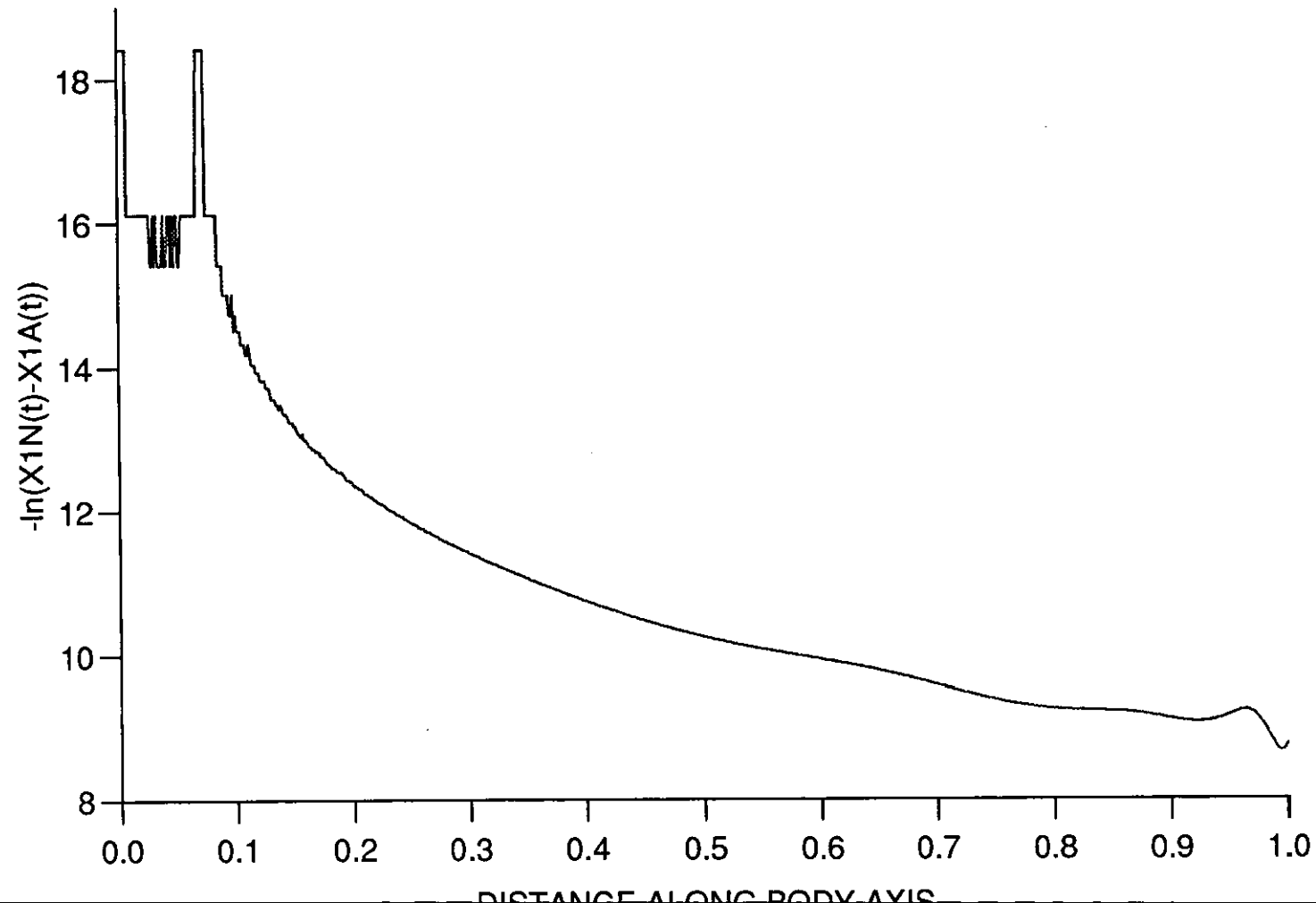
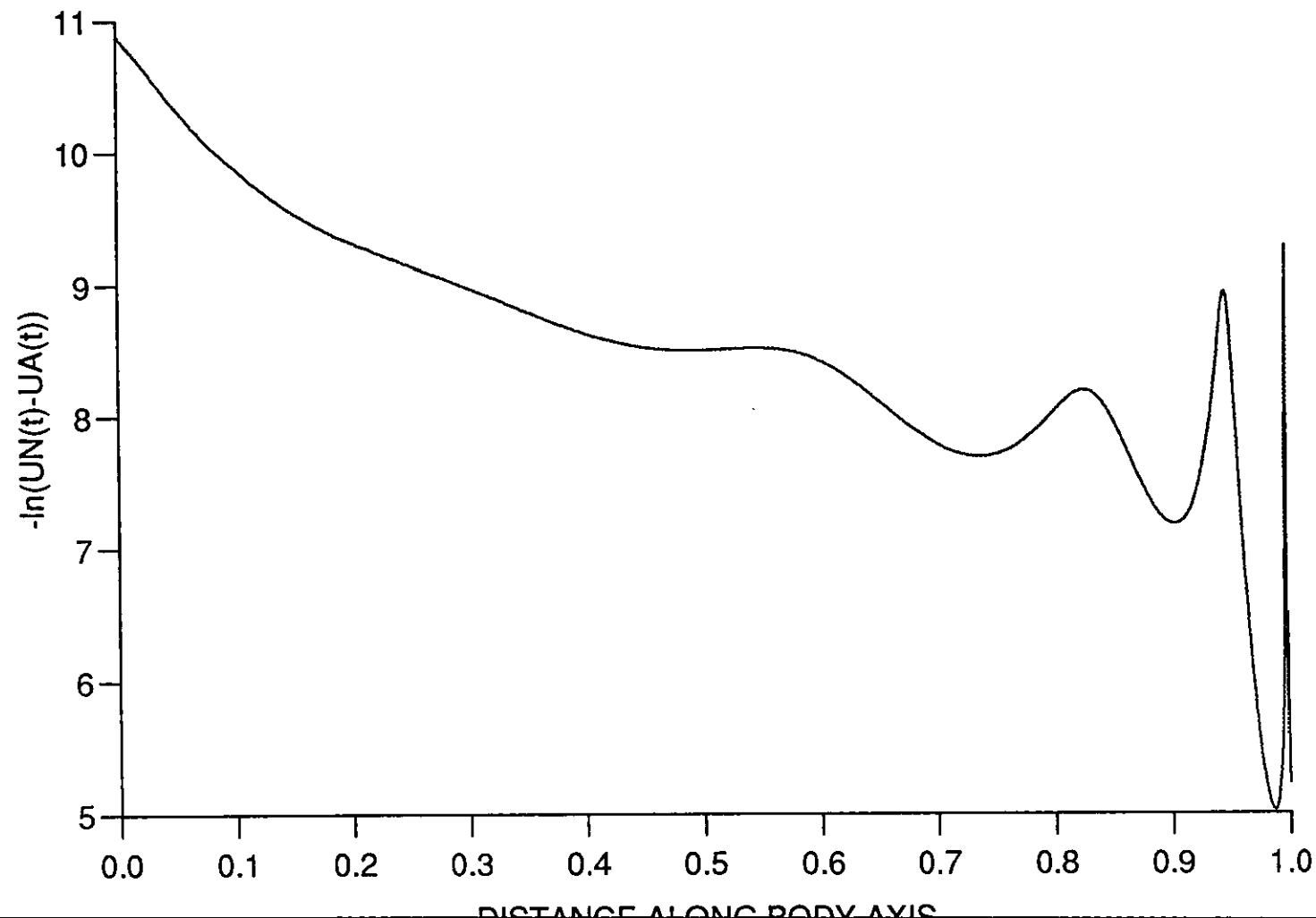


Figure (5.4.49)

HYBRID 1



Figure(5.4.50)

ANGLE TEST HYBRID

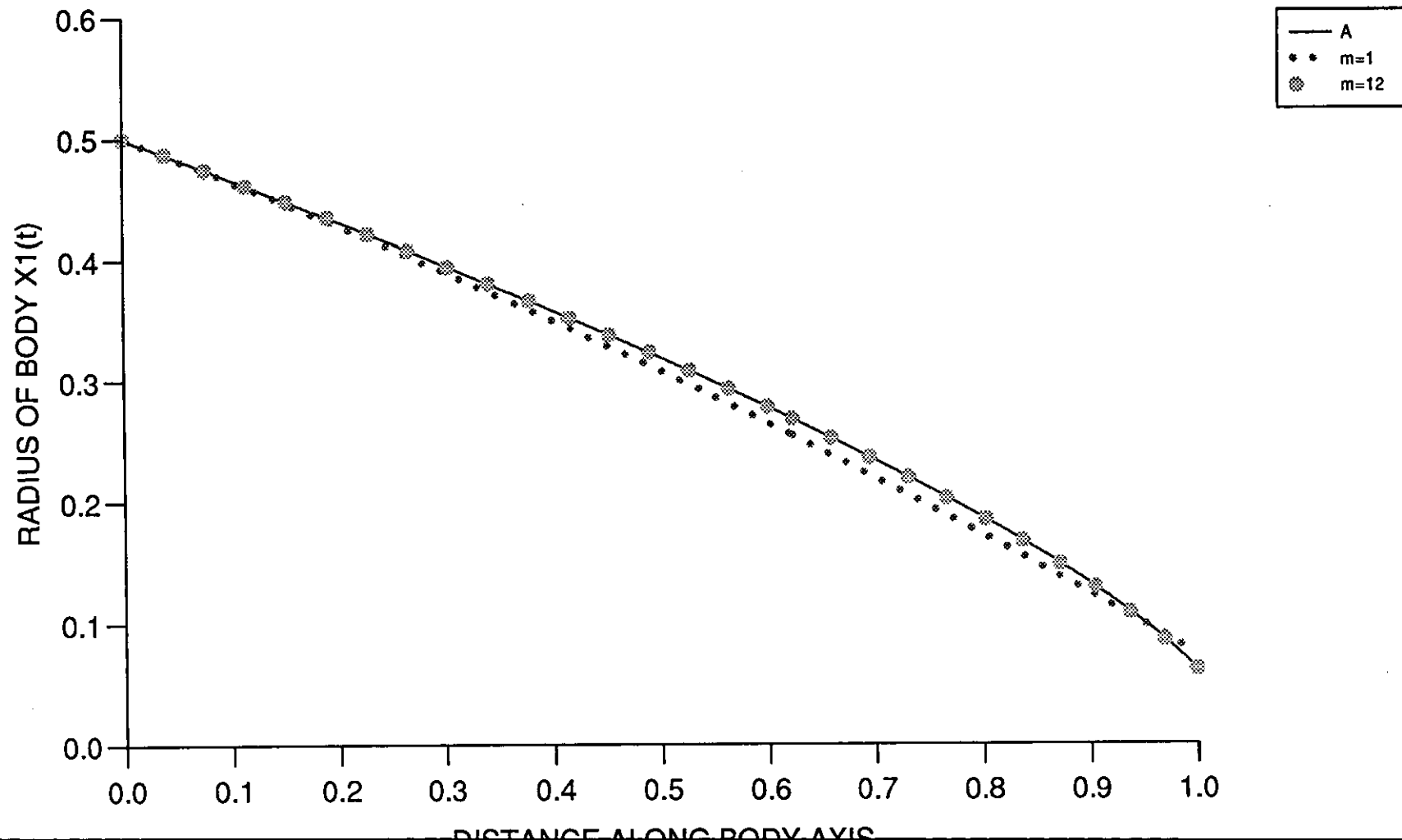


Figure (5.4.51)

ANGLE TEST HYBRID

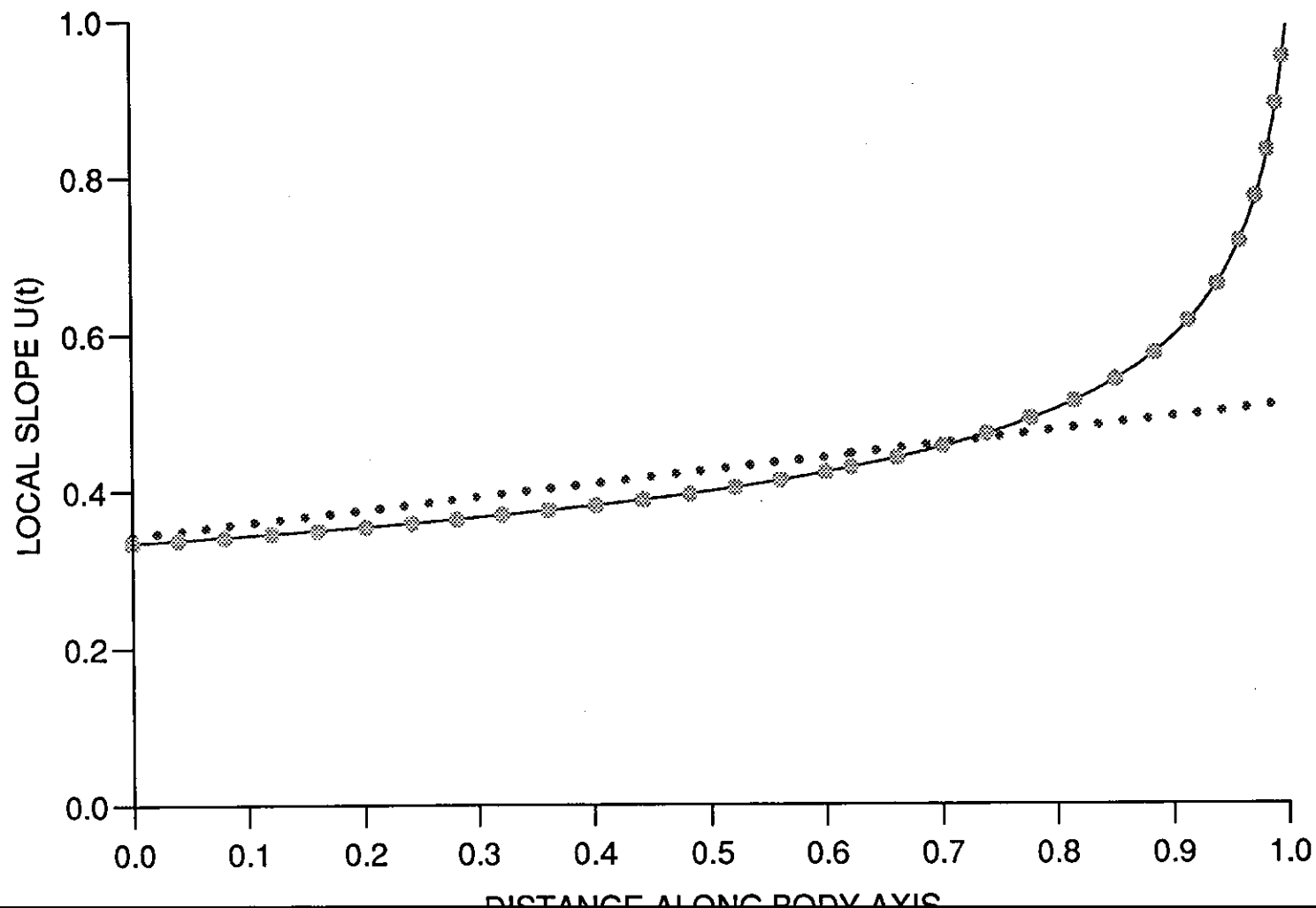


Figure (5.4.52)

ANGLE TEST HYBRID

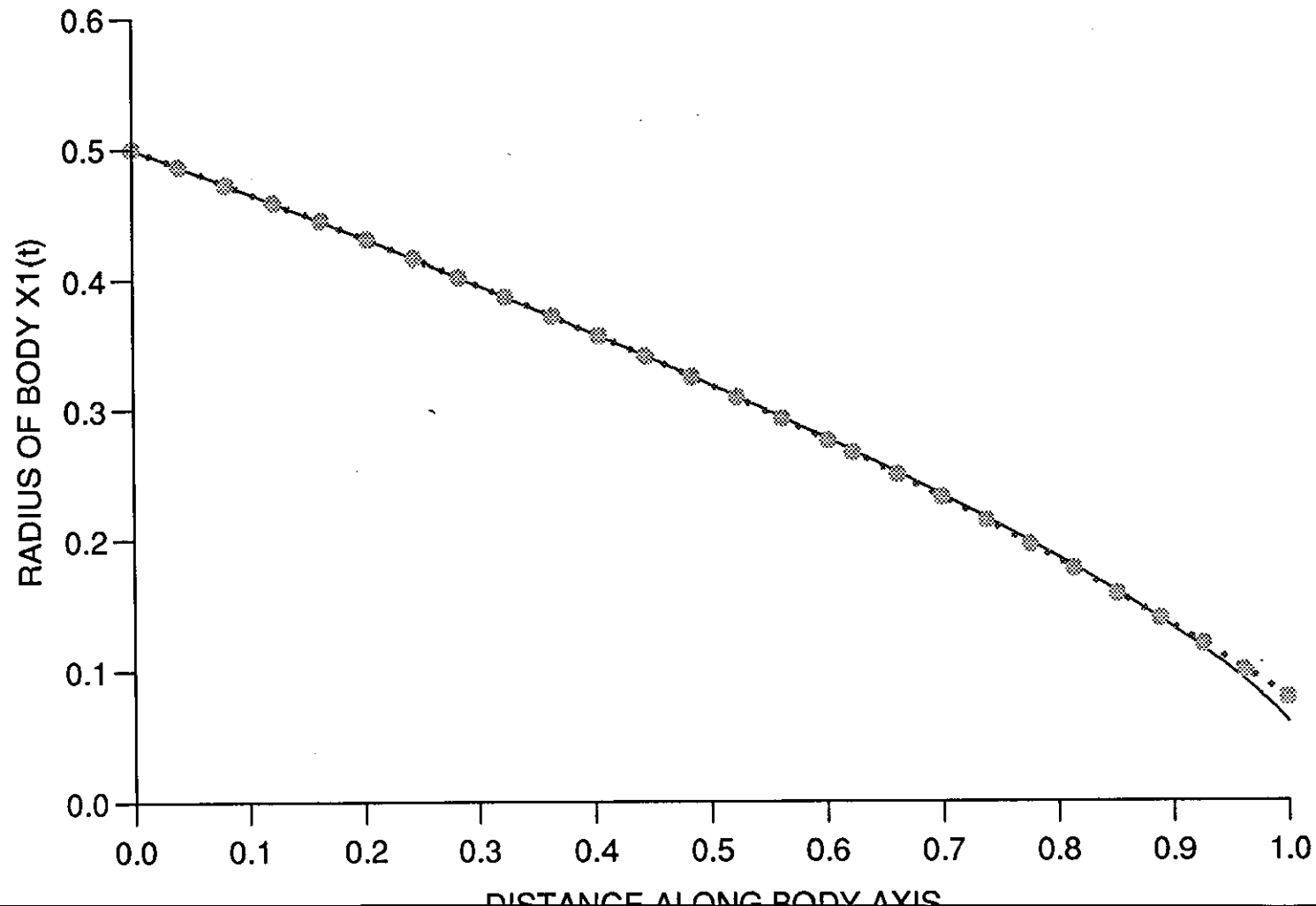


Figure (5.4.53)

ANGLE TEST HYBRID

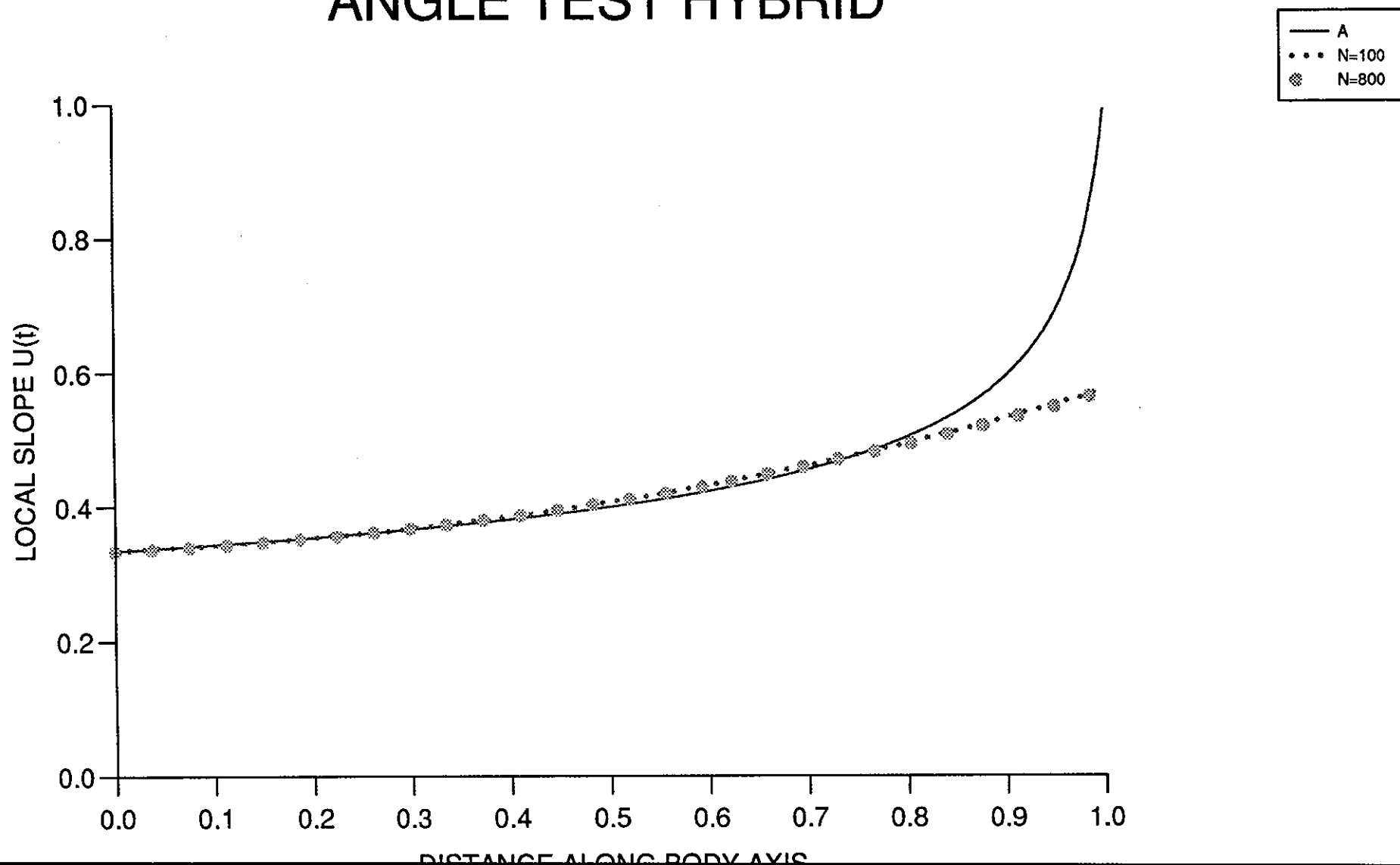


Figure (5.4.54)

ANGLE TEST HYBRID

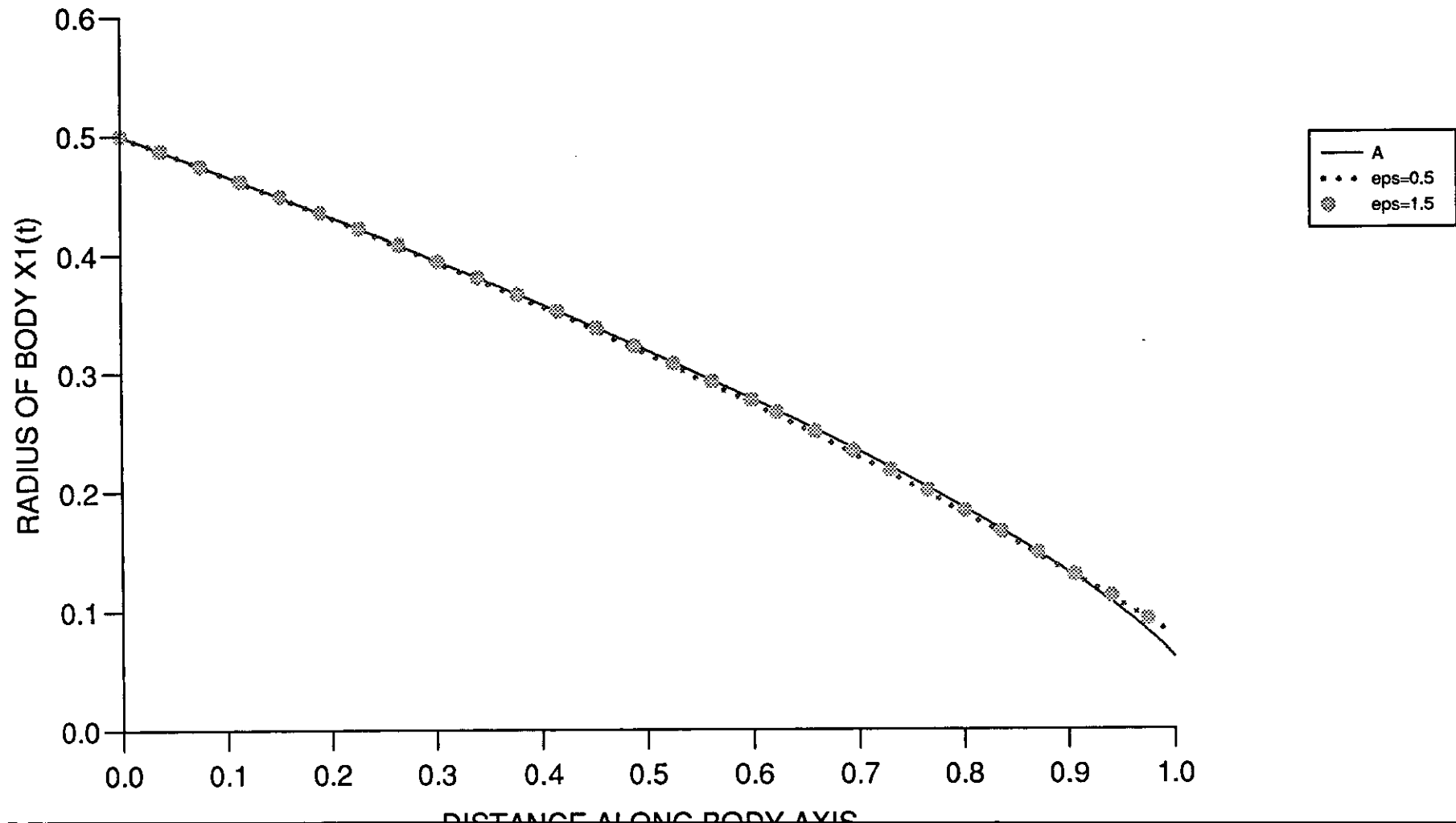


Figure (5.4.55)

ANGLE TEST HYBRID

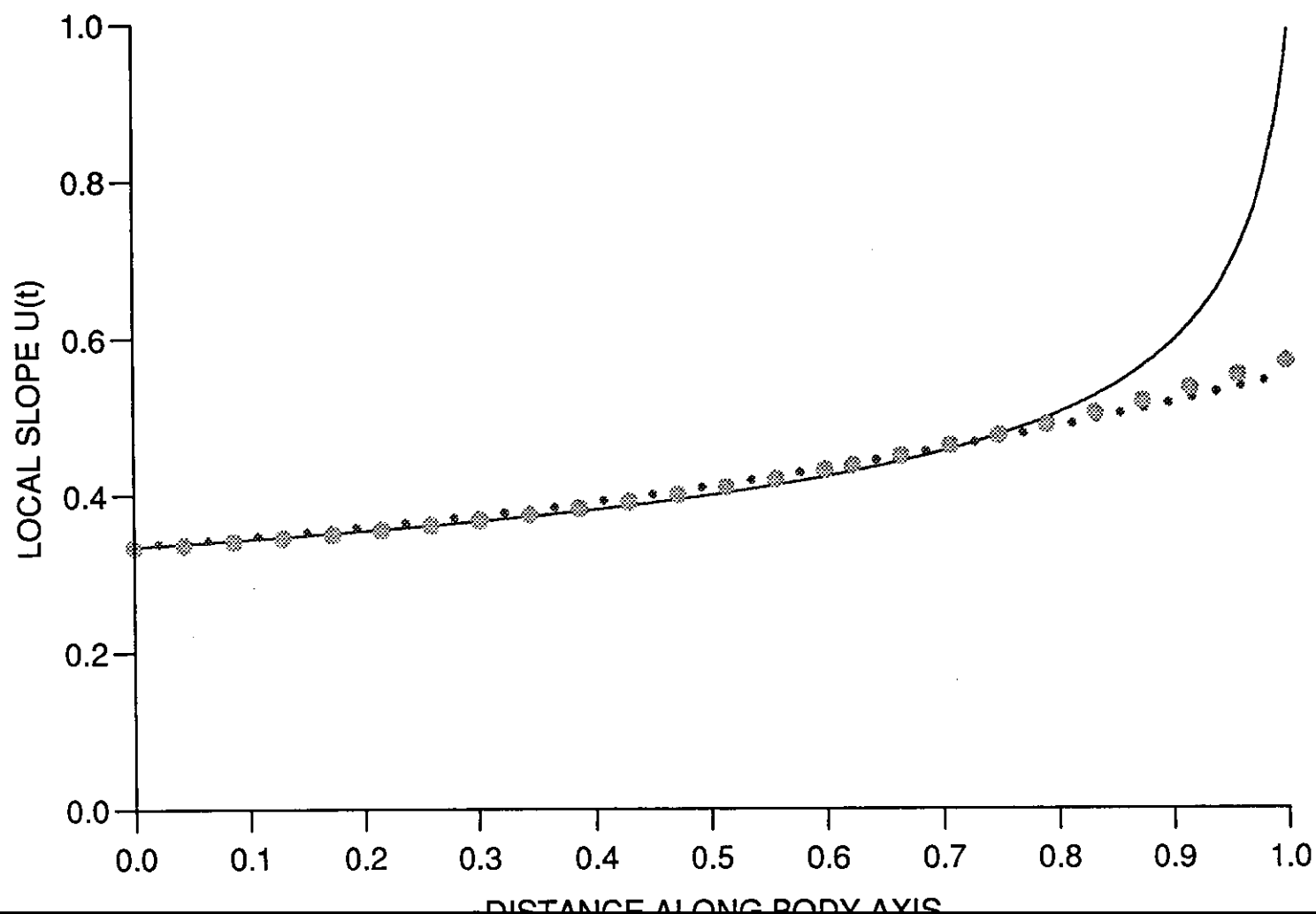


Figure (5.4.56)

ANGLE TEST HYBRID

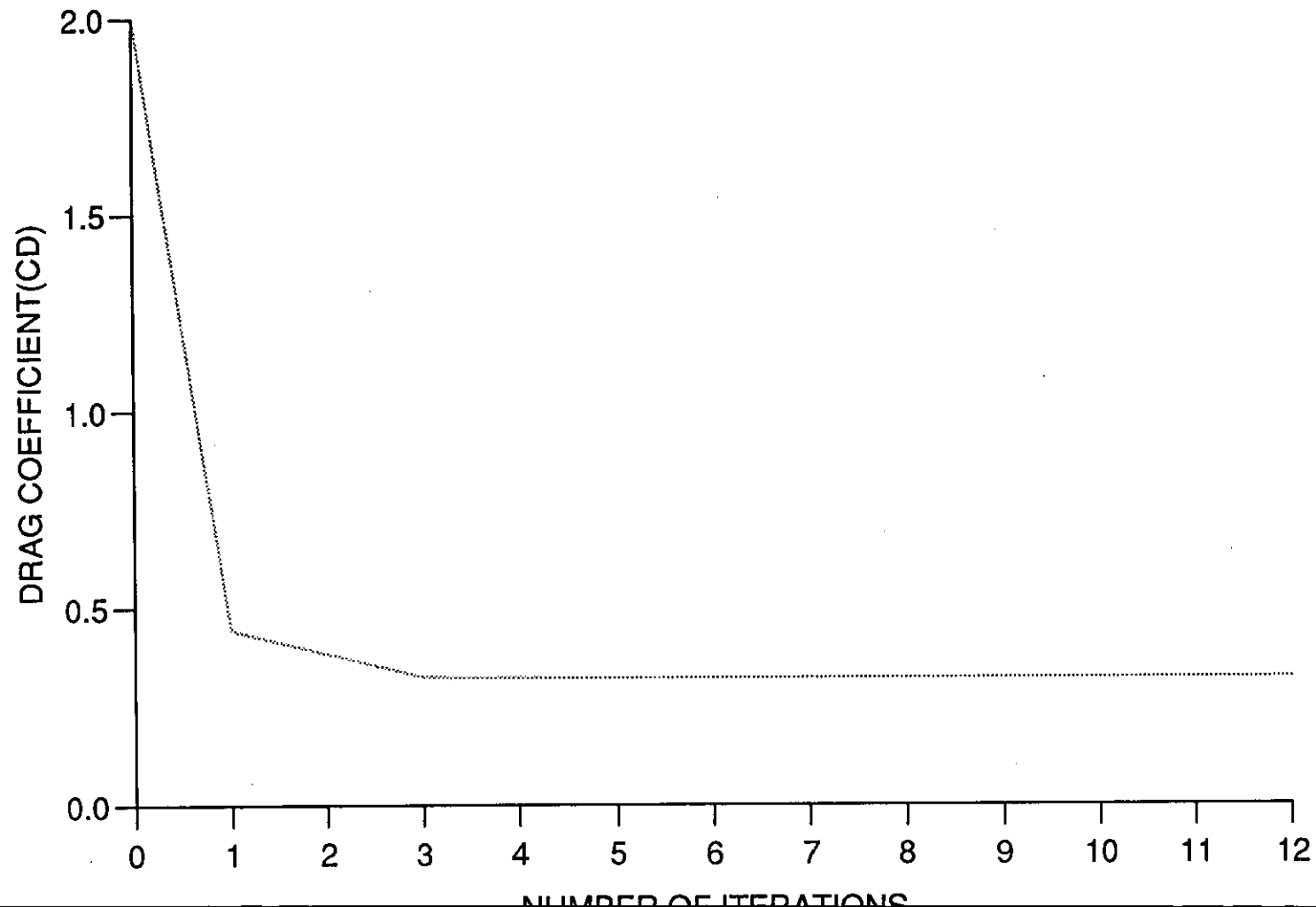


Figure (5.4.57)

ANGLE TEST HYBRID

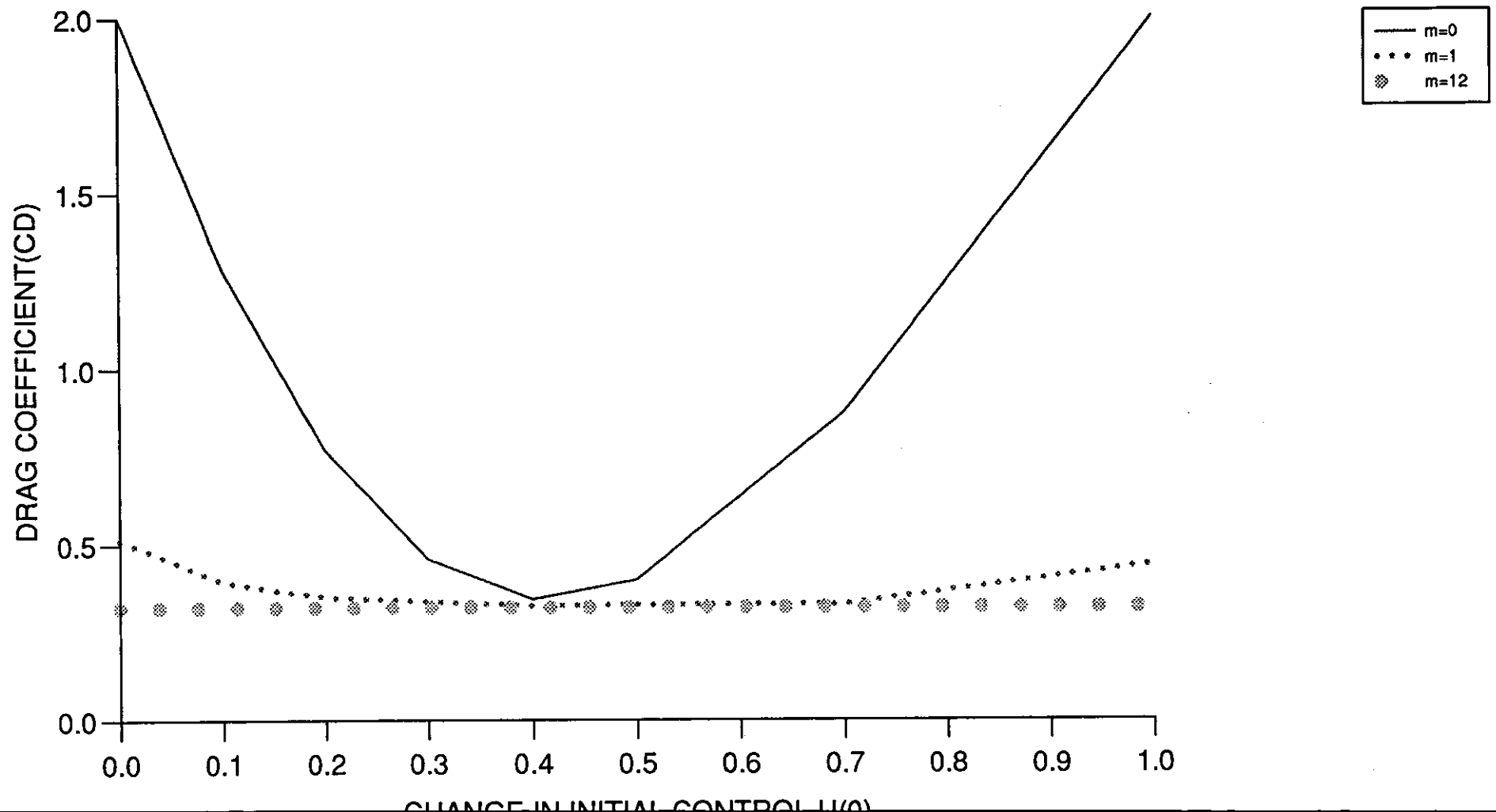


Figure (5.4.58)

ANGLE TEST HYBRID

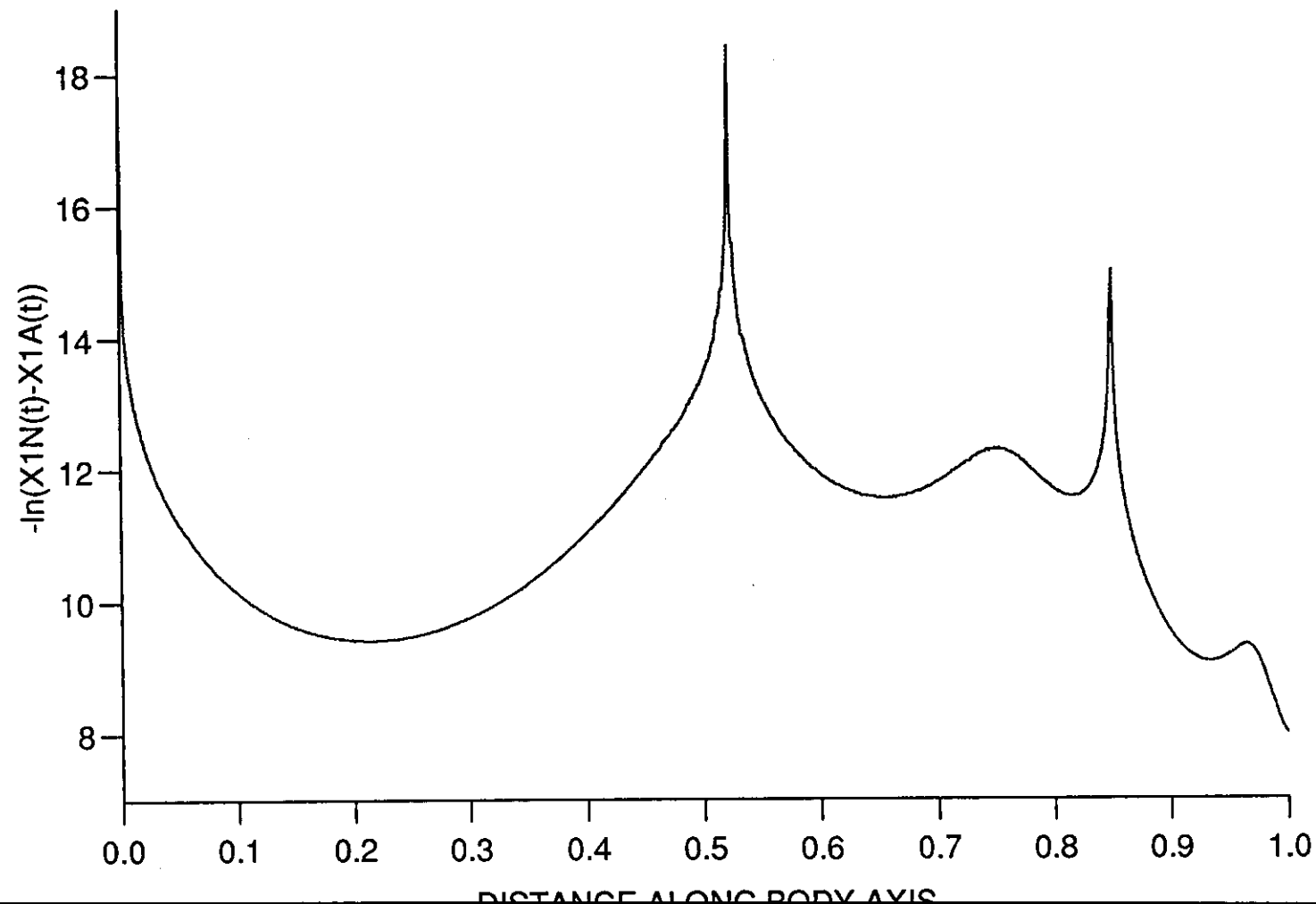


Figure (5.4.59)

ANGLE TEST HYBRID

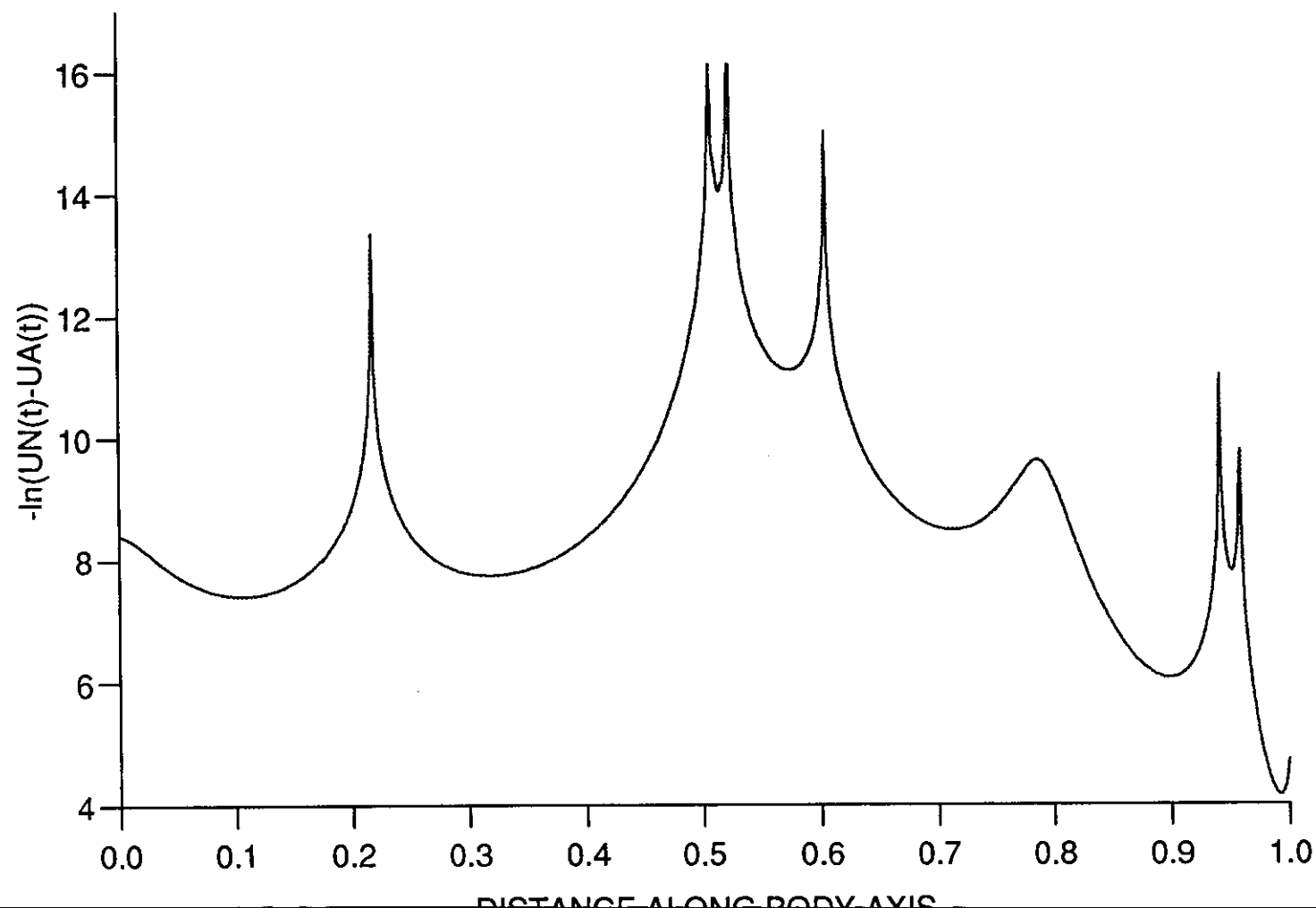


Figure (5.4.60)

HYBRID 3

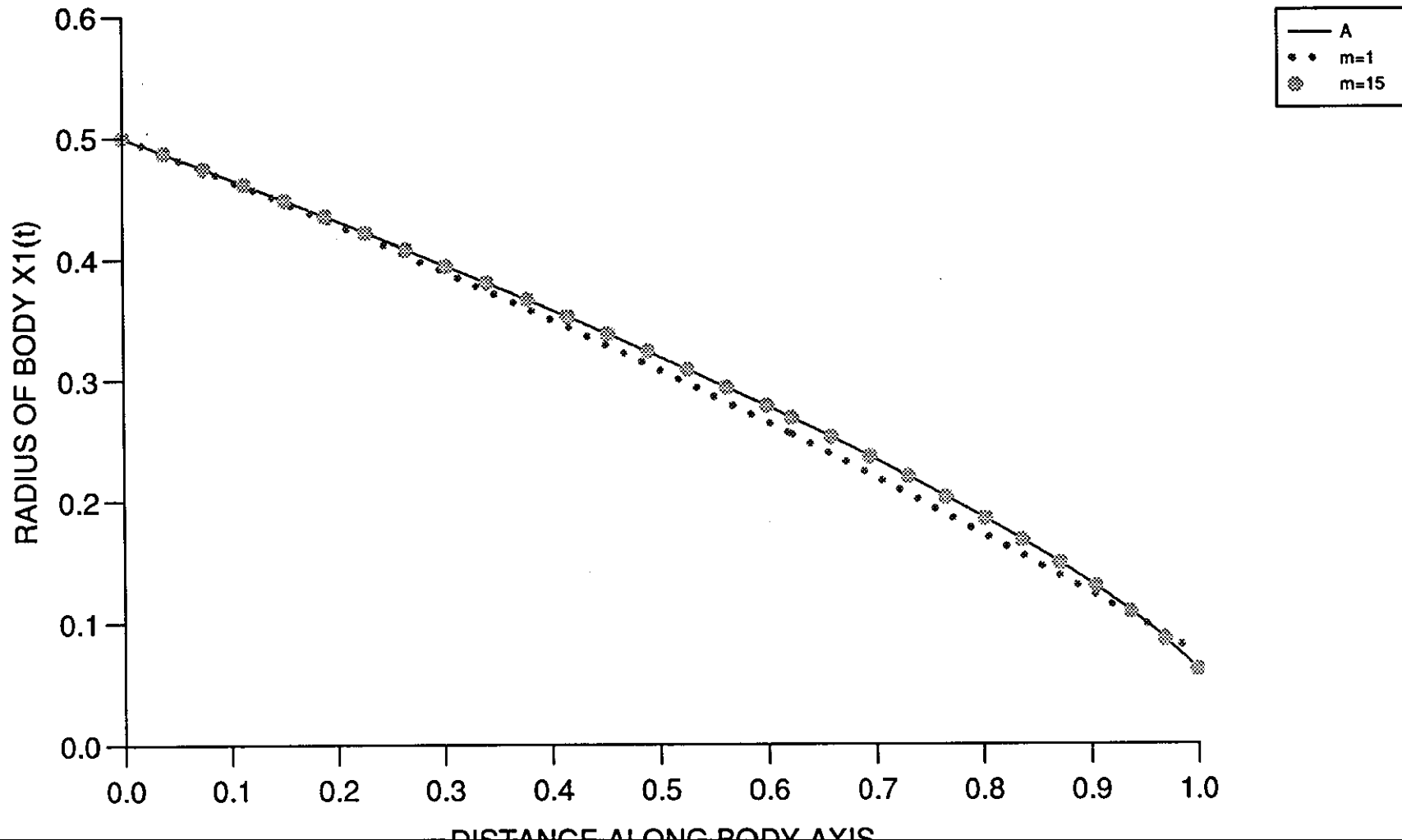


Figure (5.4.61)

HYBRID 3

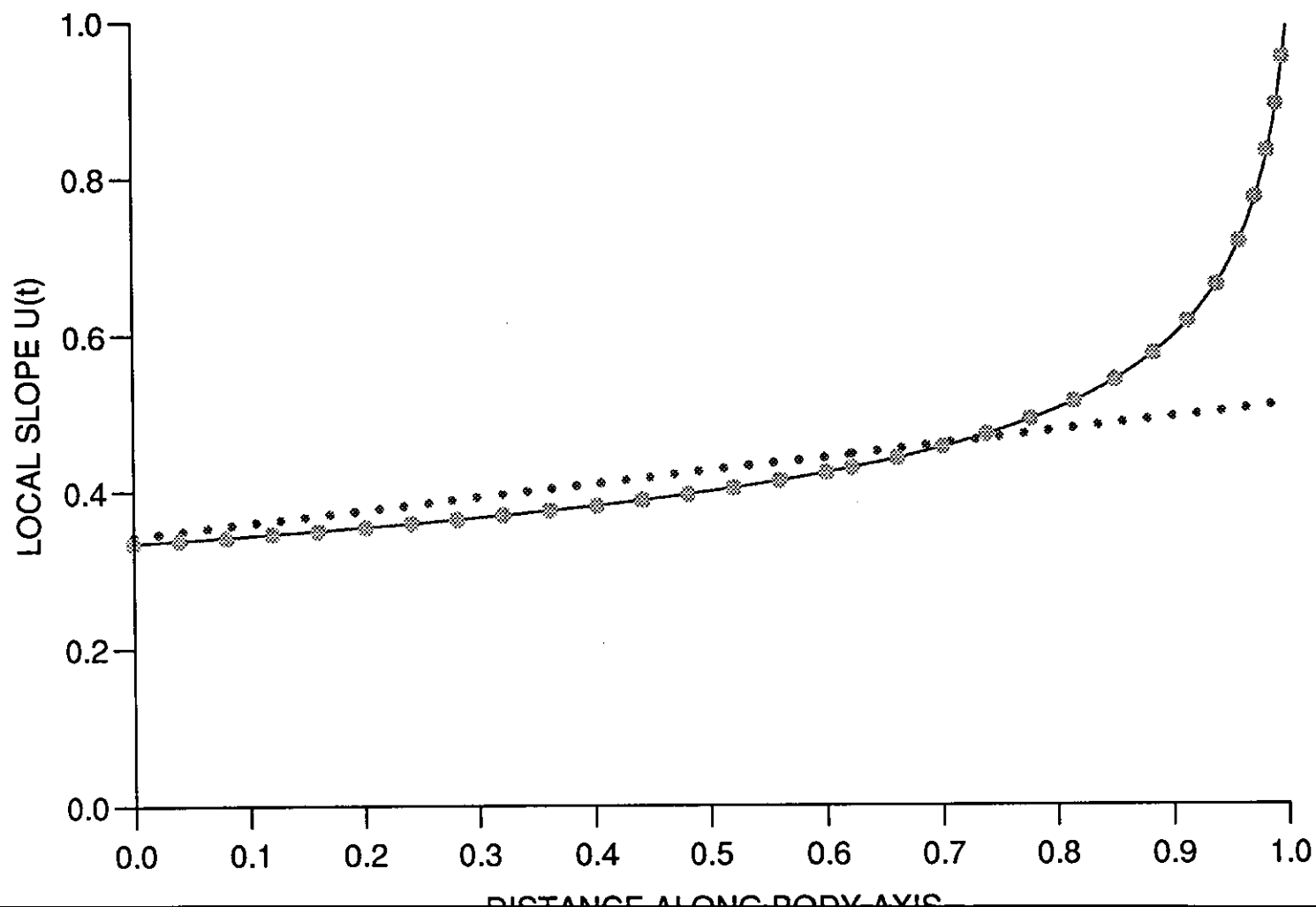


Figure (5.4.62)

HYBRID 3

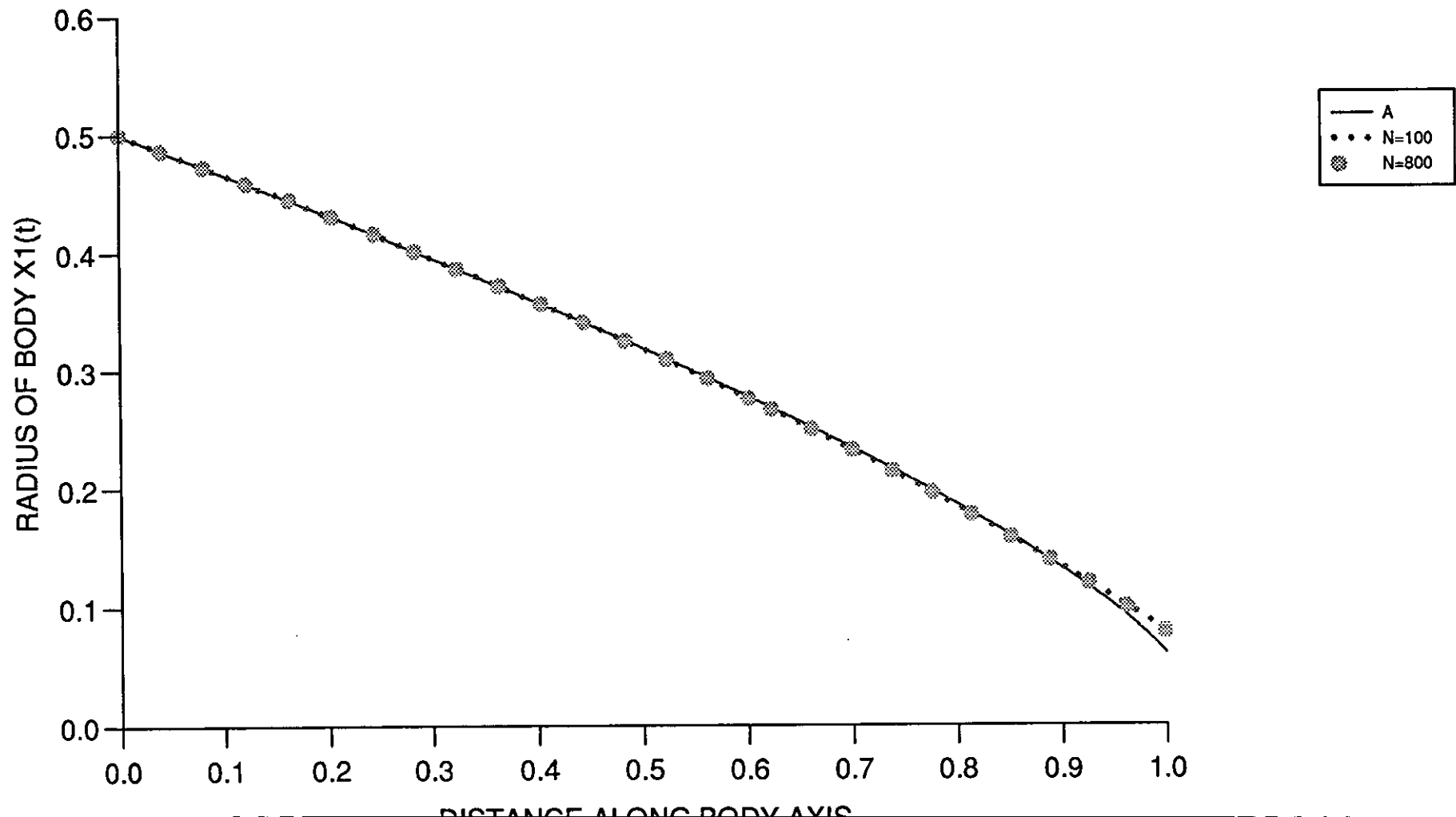


Figure (5.4.63)

HYBRID 3

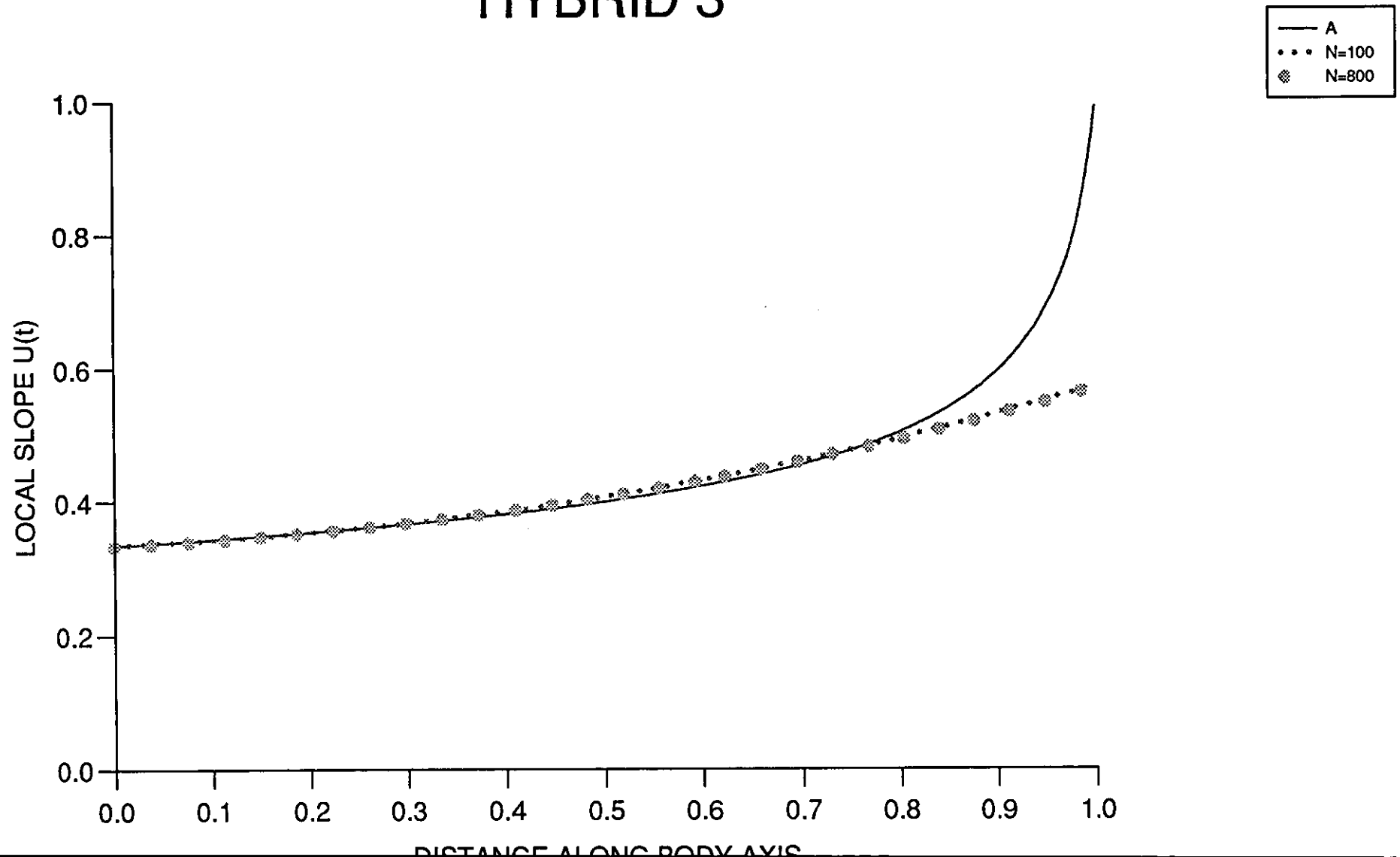


Figure (5.4.64)

HYBRID 3

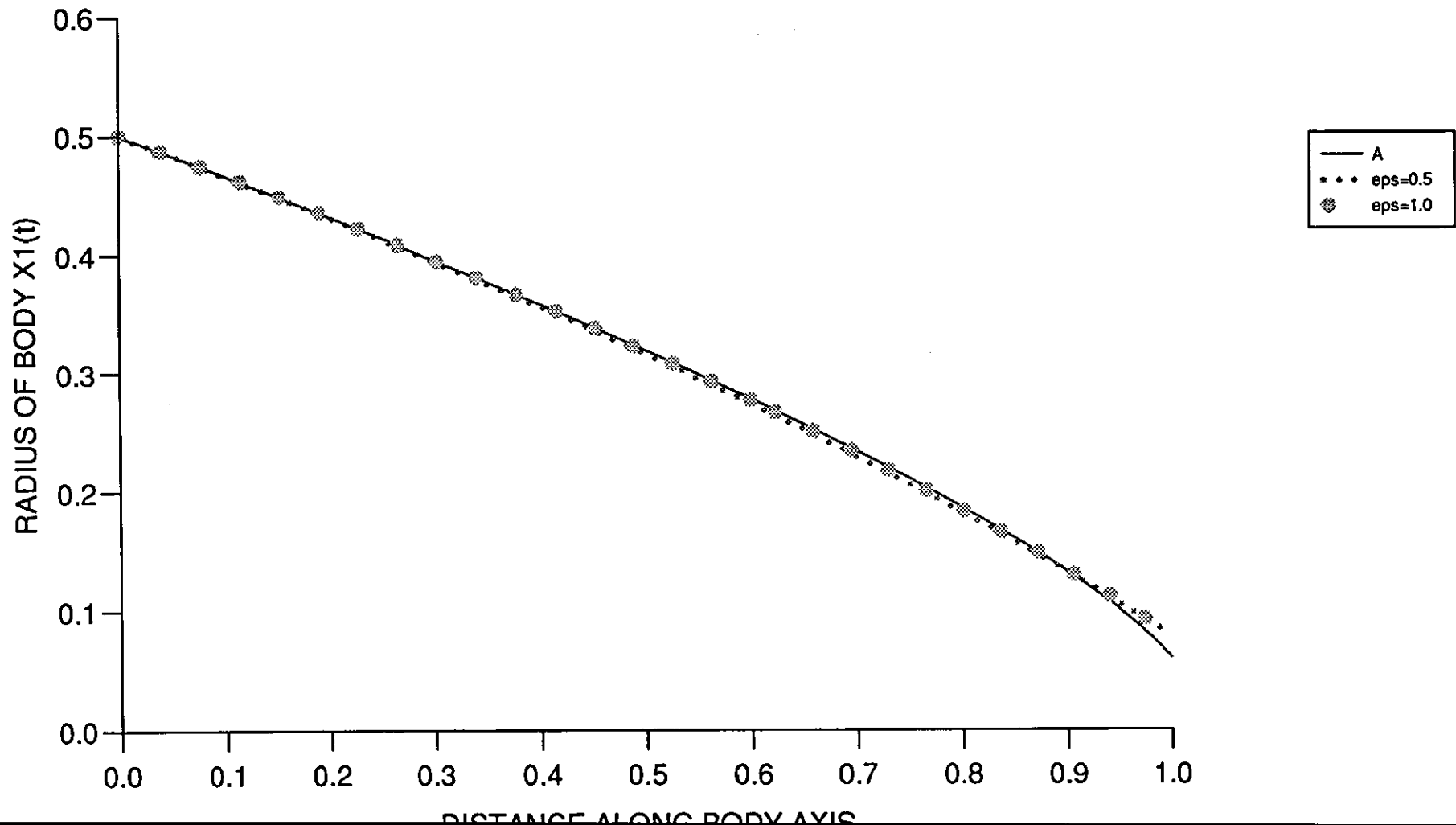


Figure (5.4.65)

HYBRID 3

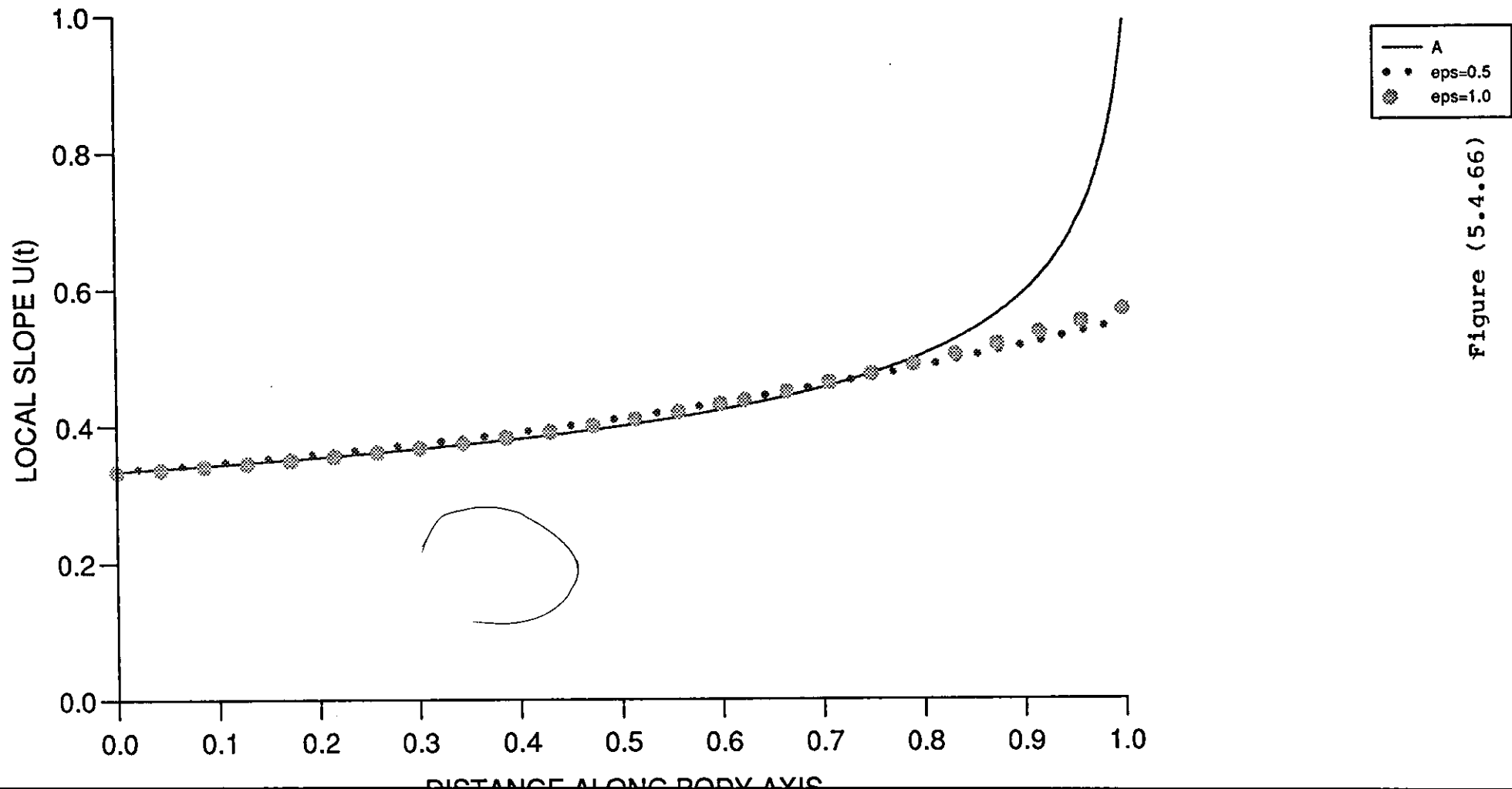


Figure (5.4.66)

HYBRID 3

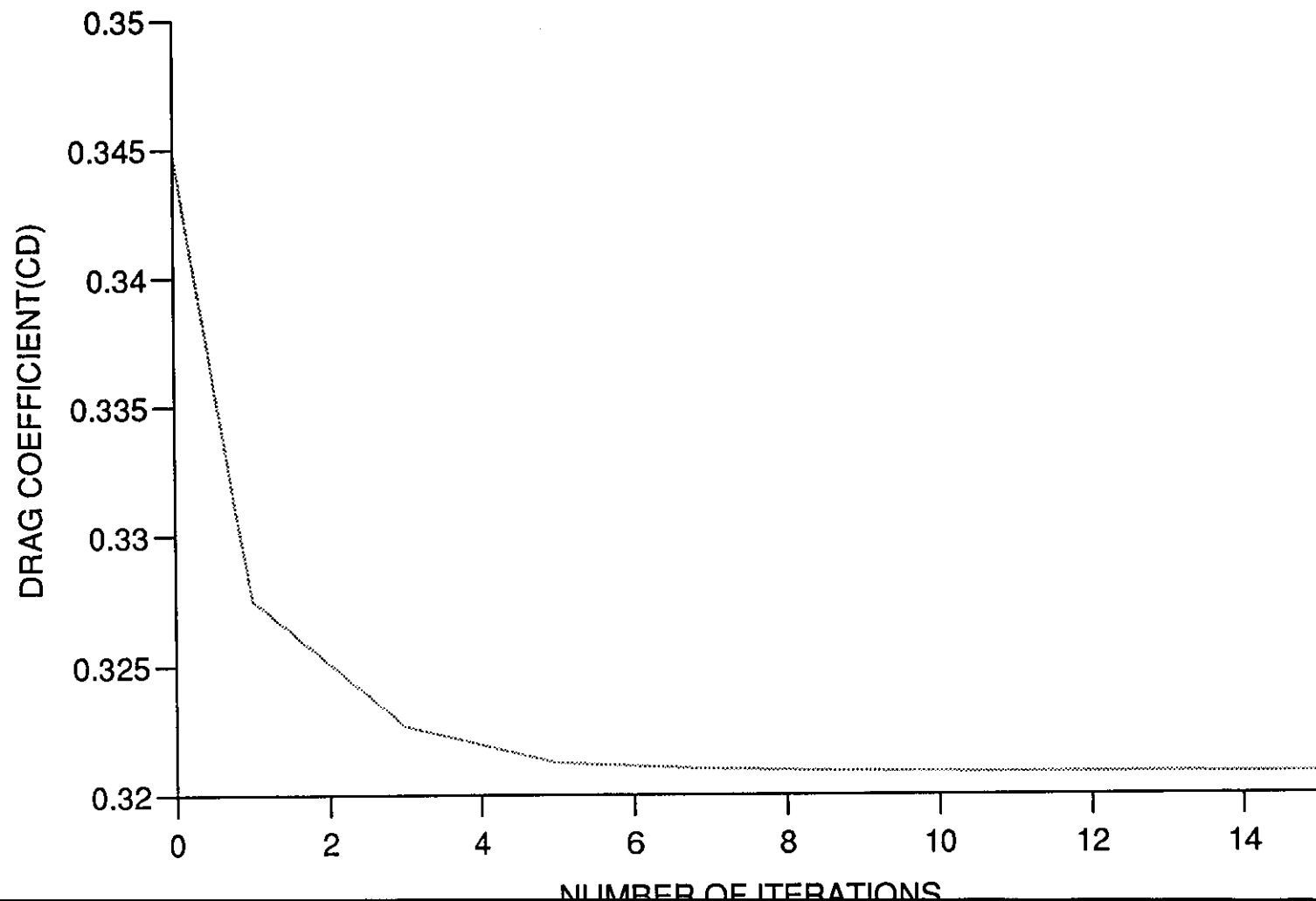


Figure (5.4.67)

HYBRID 3

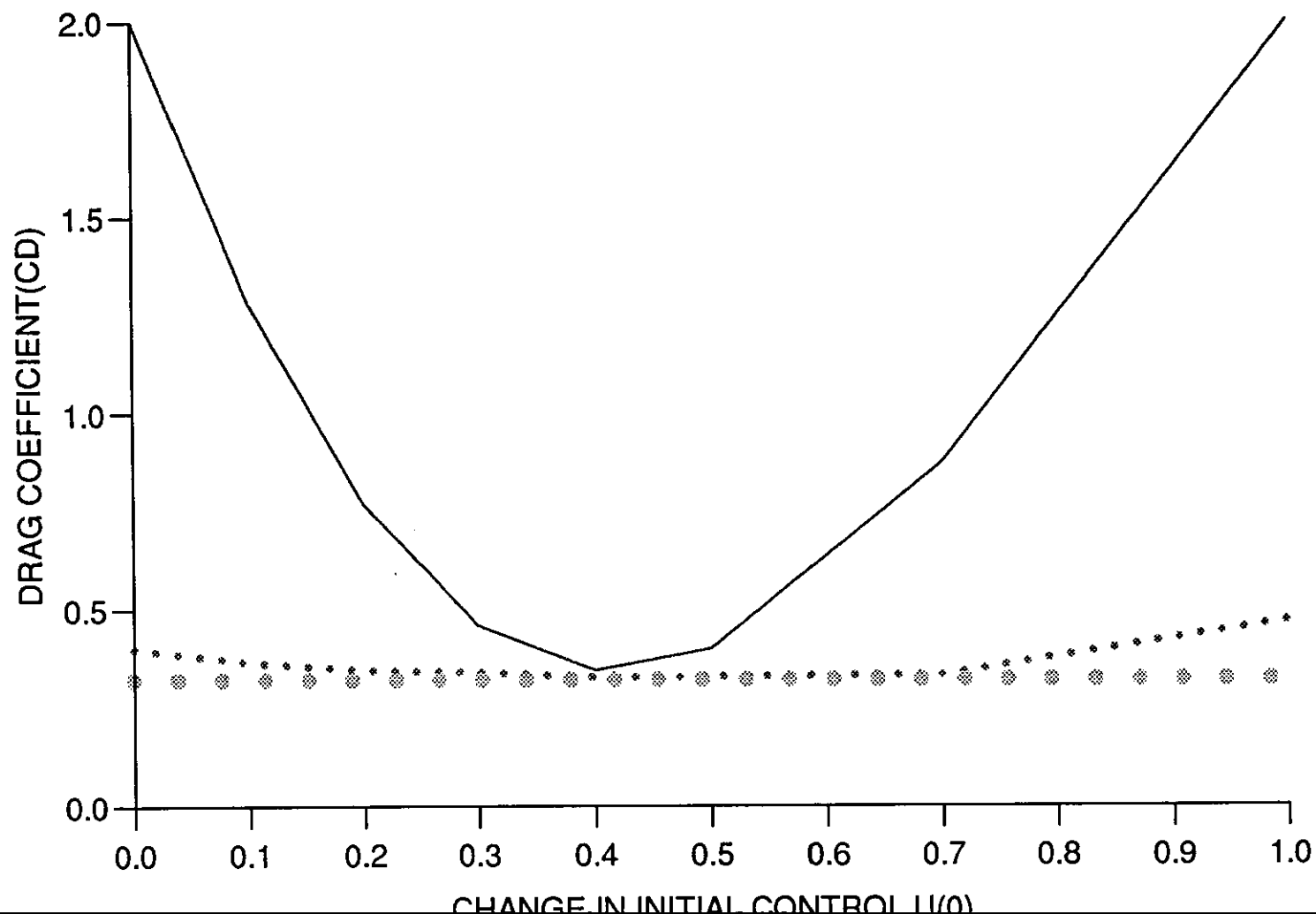


Figure (5.4.68)

HYBRID 3

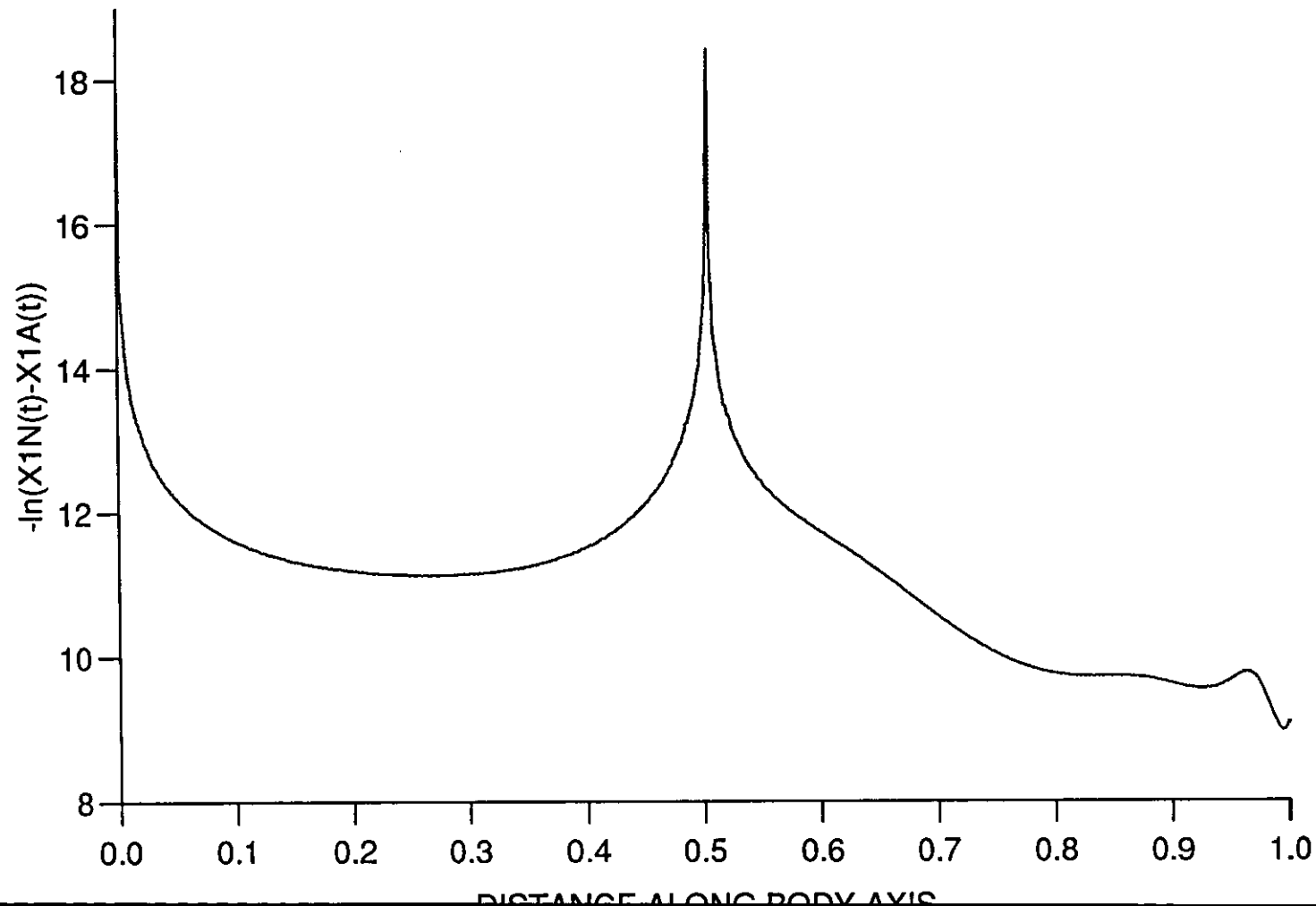


Figure (5.4.69)

HYBRID 3

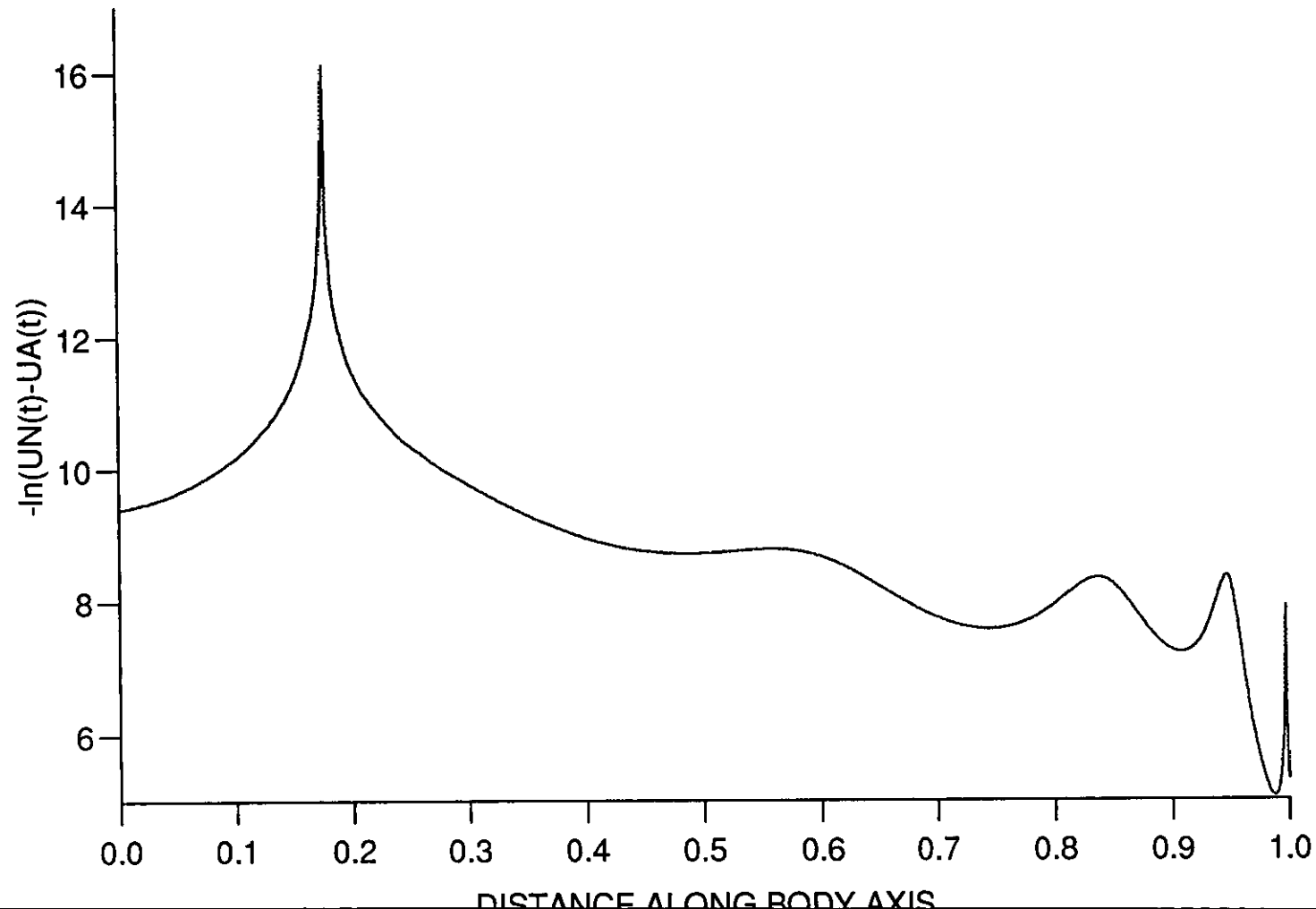


Figure (5.4.70)

COMPARISON OF THE SEVEN METHODS

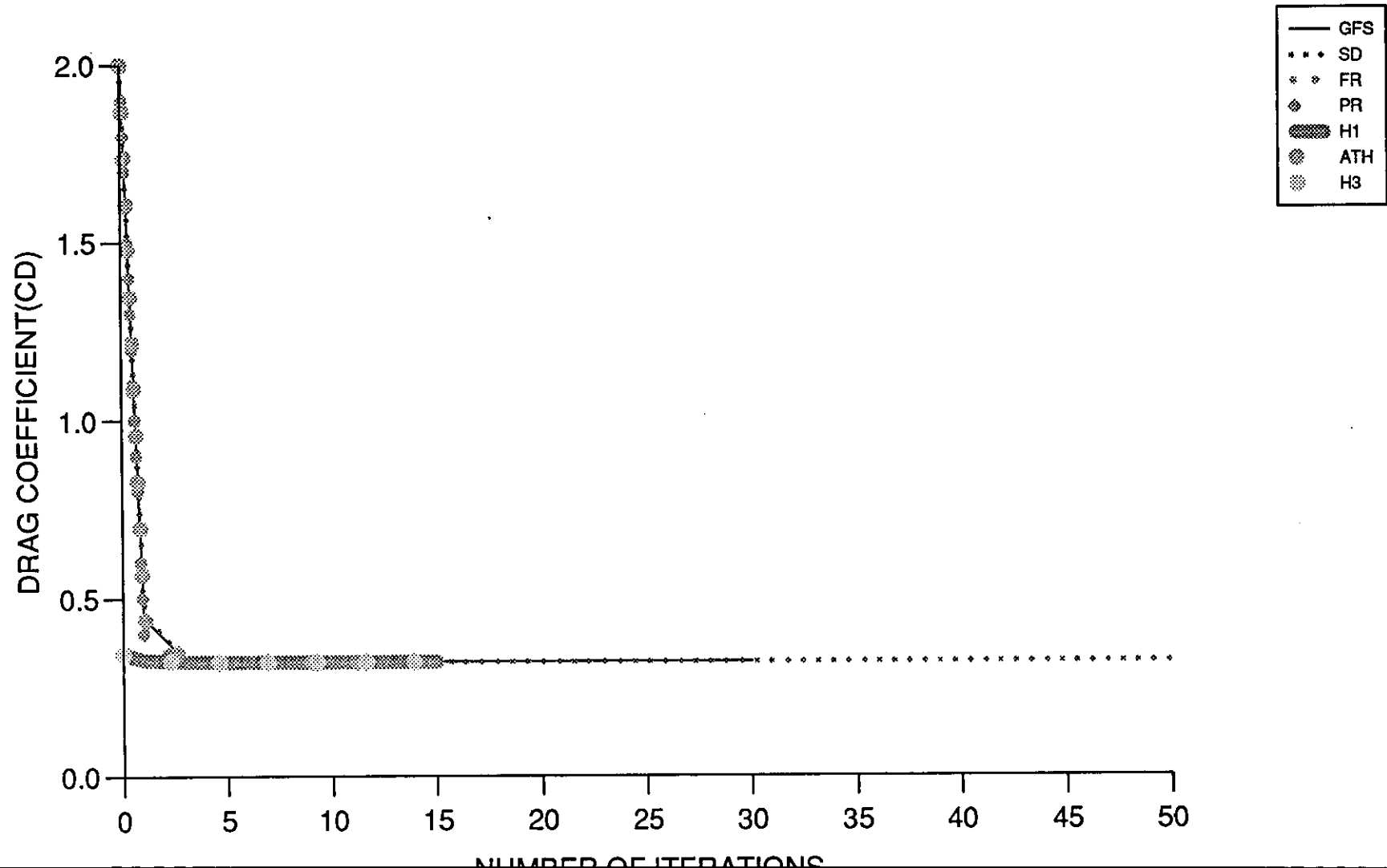


Figure (5.5.1)

Chapter 6

PROBLEM 3

6.1 An Optimal Control Problem Consisting of a Singular Arc Followed by a Nonsingular Arc

Here we consider an optimal control problem consisting of a singular arc followed by a non singular arc, taken from E.R. Edge and W.F. Powers (1976) [20].

$$\text{Minimize } J = \frac{1}{2} \int_0^{2.985} (x_2^2 - x_1^2) dt, \quad (6.1.1)$$

subject to,

$$\dot{x}_1 = x_2, x_1(0) = 0, \quad x_1(t_f) = 0.065, \quad (6.1.2)$$

$$\dot{x}_2 = u, \quad x_2(0) = 1, \quad x_2(t_f) = -1.336, \quad |u| \leq 1. \quad (6.1.3)$$

In the usual way we introduce an additional state variable;

$$x_3(t) = \frac{1}{2} \int_0^t (x_2^2 - x_1^2) dt,$$

then the state equation for x_3 will be

$$\frac{dx_3}{dt} = \frac{1}{2}(x_2^2 - x_1^2), \quad x_3(0) = 0.$$

Thus the problem is transformed into;

$$\text{minimize } J = x_3(t_f), \quad (6.1.4)$$

where, $t_f = 2.985$, subject to,

$$f_1 = \dot{x}_1 = x_2, \quad x_1(0) = 0, \quad x_1(t_f) = 0.065, \quad (6.1.5)$$

$$f_2 = \dot{x}_2 = u, \quad x_2(0) = 1, \quad x_2(t_f) = -1.336, \quad (6.1.6)$$

$$f_3 = \dot{x}_3 = \frac{1}{2}(x_2^2 - x_1^2), \quad x_3(0) = 0. \quad (6.1.7)$$

Since for this problem terminal constraints are involved, they can be handled using quadratic penalty terms, i.e., we introduce the augmented cost functional,

$$J' = 10(x_1(t_f) - 0.065)^2 + 10(x_2(t_f) + 1.335)^2 + J. \quad (6.1.8)$$

Here, the value of 10 in the augmented cost functional was suggested by Edge and Powers [20], as being a reasonable value to use.

The problem has been tackled both analytically and numerically using the conjugate gradient and Hybrid conjugate gradient methods described in chapter 4, section 4.1.

6.2 Analytical Solutions

Using the maximum principle, we solve the problem (6.1.1) to (6.1.3).

The Hamiltonian is given by

$$H_2 = \frac{1}{2}(x_2^2 - x_1^2) + \lambda_1 x_2 + \lambda_2 u \quad (6.2.1)$$

and the adjoint equations are,

$$\left. \begin{aligned} \dot{\lambda}_1 &= -\frac{\partial H}{\partial x_1} = x_1, \\ \dot{\lambda}_2 &= -\frac{\partial H}{\partial x_2} = -x_2 - \lambda_1. \end{aligned} \right\} \quad (6.2.2)$$

In order to determine the order of the singular arc we need to use the generalized Legendre-Clebsch condition (GLC) which was described in chapter 1 section 1.2.5, inequality (1.2.24). For convenience we rewrite it here,

$$(-1)^q \frac{\partial}{\partial u} \left[\frac{d^{2q}}{dt^{2q}} H_u \right] \geq 0. \quad (6.2.3)$$

The inequality (6.2.3) is called the strengthened GLC condition when strict inequality holds. In our problem the switching function and its second derivatives are,

$$H_u = \lambda_2 \quad (6.2.4)$$

$$\begin{aligned} \ddot{H}_u = \frac{d}{dt} \dot{H}_u &= \frac{d}{dt} \left[\frac{d\lambda_2}{dt} \right] = \frac{d}{dt} [-\lambda_1 - x_2] \\ &= -\dot{\lambda}_1 - \dot{x}_2 \end{aligned}$$

$$\text{i.e., } \ddot{H}_u = -x_1 - u. \quad (6.2.5)$$

Checking the GLC condition (6.2.3),

$$\frac{\partial}{\partial u} \ddot{H}_u = -1, q = 1, (-1)^q \frac{\partial}{\partial u} \left[\frac{d^{2q}}{dt^{2q}} H_u \right] = (-1)^1 (-1) = 1 \geq 0.$$

Thus the singular arc is first order. From (6.2.5) we see that the strengthened GLC condition is satisfied. Setting the right hand side of (6.2.5) to zero we get,

$$\ddot{H}_u = 0 \Rightarrow x_1 = -u,$$

substituting in (6.1.2) we get,

$$\left. \begin{aligned} -\frac{du}{dt} - x_2 &= 0, & u(0) &= 0, \\ \frac{dx_2}{dt} - u &= 0, & x_2(0) &= 1. \end{aligned} \right\} \quad (6.2.6)$$

The solution is as follows:

$$u(t) = -\sin t, x_2(t) = \cos t \text{ and } x_1(t) = \sin t. \quad (6.2.7)$$

Now before proceeding any further we introduce the junction theorems.

Theorem 6.2.1:

Let t_c be a point at which singular and nonsingular subarcs of an optimal control u are joined and let q be the order of the singular subarc, suppose the strengthened GLC condition is satisfied at t_c , i.e., $(-1)^q \left(\frac{\partial}{\partial u} \right) H_u^{(2q)} > 0$, and assume that the control is piecewise analytic in a neighborhood of t_c . Let $u^{(r)} (r \geq 0)$ be the lowest order derivative of u which is discontinuous at t_c . Then $q + r$ is an odd integer.

Proof. Refer to McDanell and Powers (1971) [137].

Theorem 6.2.2:

Let t_c be a point at which singular and nonsingular subarcs of an optimal control u are joined, and let q be the order of the singular arc. Assumed that the control is piecewise analytic in a neighbourhood of t_c . Let $u^{(r)} (r \geq 0)$ be the lowest order derivative of u which is discontinuous at t_c , and let $\beta^{(m)} (m \geq 0)$ be the lowest order derivative of the GLC expression $\left(\frac{\partial}{\partial u} \right) H_u^{(2q)} \equiv \beta$ which is nonzero at t_c .

Then,

(i) if $m \leq r, q+r+m$ is an odd integer; (ii) if $m > r, -\text{sgn}[\beta^{(m)}(t_c^+) \beta^{(m)}(t_c^-)] = (-1)^{q+r+m}$.

Proof. A proof of the above theorem was given by McDanell and Powers (1971) [137], but later on this proof was shown to be incorrect in general but true for when $q = m = 1$ (Bell (1979) [138]), and for $m = 1, q > 1$ (Bortins (1983) [140]).

Also part 2 of the conjecture has also been shown to be false, (Bell, (1987) [141] and Bell has also recently found a counter example to part 1 (to be published later). Now for the problem (6.1.1) to (6.1.3), when the terminal state (0.065, -1.336) is chosen to lie on the trajectory (6.2.7), the solution is singular with $u = -\sin t, t \in [0, \frac{3\pi}{4})$ and nonsingular with $u = -1, t \in (\frac{3\pi}{4}, 2.985]$, as can be shown by the sufficient conditions in [142] and [143].

For this case the optimal control is

$$u^* = \begin{cases} -\sin t, & t \in [0, \frac{3\pi}{4}), \\ -1, & t \in (\frac{3\pi}{4}, 2.985] \end{cases} \quad (6.2.8)$$

The above can be justified as follows:

Whenever $H_u = \lambda_2(t) \neq 0$, $u^*(t) = -\text{sgn } \lambda_2(t)$. Now there are 3 possibilities for $\lambda_2(t)$ that should be examined,

$$(i) \lambda_2 > 0, \quad (ii) \lambda_2 < 0 \quad \text{and} \quad (iii) \lambda_2 = 0.$$

When (i) is considered, then $u = -1$ and equations (6.1.2) to (6.1.3) become

$$\left. \begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -1, \end{aligned} \right\} u = -1 \quad (6.2.9)$$

and the adjoint equations are as (6.2.2).

Solving (6.2.9) we get

$$\begin{aligned} x_2 &= c_2 - t, \\ x_1 &= c_2 t - \frac{t^2}{2} + c_1. \end{aligned}$$

Since $x_1(t_f) = 0.065$ and $x_2(t_f) = -1.336$,

$$\begin{aligned} -1.336 &= c_2 - 2.985, \\ \text{i.e., } c_2 &= 1.649 \quad \text{and} \\ 0.065 &= c_2(2.985) - \frac{(2.985)^2}{2} + c_1 \\ \text{i.e., } c_1 &= -0.4021525. \end{aligned}$$

Thus

$$x_1 = -\frac{t^2}{2} + 1.649t - 0.4021525, \quad (6.2.10)$$

$$x_2 = 1.649 - t. \quad (6.2.11)$$

Considering the adjoint equations (6.2.2) related to (6.2.9) we have,

$$\begin{aligned} \dot{\lambda}_1 &= c_2 t - \frac{t^2}{2} + c_1 \\ \dot{\lambda}_2 &= -c_2 - \lambda_1(0) + (1 - c_1)t - \frac{c_2}{2}t^2 + \frac{t^3}{6}, \end{aligned}$$

and the solutions are,

$$\begin{aligned} \lambda_1 &= \lambda_1(0) + \frac{c_2}{2}t^2 - \frac{t^3}{6} + c_1 t \\ \lambda_2 &= \lambda_2(0) - (c_2 + \lambda_1(0))t + (1 - c_1)\frac{t^2}{2} - \frac{c_2}{6}t^3 - \frac{t^4}{24}. \end{aligned}$$

When (ii) is considered, then $u = 1$ and the equations (6.1.2) to (6.1.3) become

$$\left. \begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= 1 \end{aligned} \right\} u = 1 \quad (6.2.10)$$

Solving (6.2.10) we get,

$$\begin{aligned} x_2 &= B_2 + t, \\ x_1 &= B_2 t + \frac{t^2}{2} + B_1. \end{aligned}$$

Since $x_1(t_f) = 0.065$ and $x_2(t_f) = -1.336$, therefore,

$$\begin{aligned} -1.336 &= B_2 + 2.985, \\ \text{i.e., } B_2 &= -4.321 \text{ and} \\ 0.065 &= -4321(2.985) + \frac{(2.985)^2}{2} + B_2. \\ \text{i.e., } B_1 &= 8.5080725. \end{aligned}$$

Thus,

$$x_1 = \frac{t^2}{2} - 4.321t + 8.5080725, \quad (6.2.11)$$

$$x_2 = t - 4.321. \quad (6.2.12)$$

Considering the adjoint equations related to (6.2.10) we have,

$$\begin{aligned} \dot{\lambda}_1 &= B_2t + \frac{t^2}{2} + B_2 \\ \dot{\lambda}_2 &= -(\lambda_1(0) + B_2) - (1 + B_1)t - \frac{B_2}{2}t^2 \\ &\quad - \frac{t^3}{6}, \end{aligned}$$

and the solutions are

$$\begin{aligned} \lambda_1 &= \frac{t^3}{6} + \frac{B_2}{2}t^2 + B_1t + \lambda_1(0) \\ \lambda_2 &= -\frac{t^4}{24} - \frac{B_2}{6}t^3 - \frac{(1 + B_1)}{2}t^2 - (\lambda_1(0) + B_2)t + \lambda_2(0). \end{aligned}$$

When (iii) is considered, then $u = -\sin t$ and equations (6.1.2) to (6.1.3) become

$$\left. \begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\sin t \end{aligned} \right\} u = -\sin t \quad (6.2.13)$$

Solving (6.2.13) we get,

$$\begin{aligned} x_2 &= \cos t + c'_2 \\ x_1 &= \sin t + c'_2t + c'_1. \end{aligned}$$

Considering the adjoint equations related to (6.2.13) we have,

$$\begin{aligned} \dot{\lambda}_1 &= \sin t + c'_2t + c'_1, \\ \dot{\lambda}_2 &= -\frac{c'_2}{2}t^2 - c'_1t - (c'_2 + \lambda'_1(0)). \end{aligned}$$

The solutions are,

$$\begin{aligned} \lambda_1 &= -\cos t + \frac{c'_2}{2}t^2 + c'_1t + \lambda'_1(0) \\ \lambda_2 &= -\frac{c'_2}{6}t^3 - \frac{c'_1}{2}t^2 - (c'_2 + \lambda'_1(0))t + \lambda'_2(0). \end{aligned}$$

Here since $\lambda_2(t) \equiv 0$, therefore,

$$\lambda'_1(0) = \lambda'_2(0) = c'_2 = \lambda'_2(0) = 0.$$

Also

$$x_2 = \cos t, \quad (6.2.14)$$

$$x_1 = \sin t. \quad (6.2.15)$$

Now we need to find the switching time t_c . Suppose the control is singular up to t_c and $+1$ up to 2.985 then from (6.2.11), (6.2.12) and (6.2.14), (6.2.15) we have,

$$\left. \begin{aligned} -4.321 + t_c &= \cos t_c \\ -4.321t_c + \frac{t_c^2}{2} + 8.5080725 &= \sin t_c. \end{aligned} \right\} \quad (6.2.16)$$

Here, since $t_c \in [0, 2.985]$ and $-1 \leq \cos t_c \leq 1$, therefore we can find no solution to satisfy (6.2.16), i.e. the curve will not switch to $u = +1$. Now suppose control is singular up to t_c and -1 up to 2.985, then from (6.2.10), (6.2.11) and (6.2.14), (6.2.15) we have,

$$\left. \begin{aligned} 1.649 - t_c &= \cos t_c, \\ -\frac{t_c^2}{2} + 1.649t_c &= 0.4021525 = \sin t_c. \end{aligned} \right\} \quad (6.2.17)$$

which is satisfied by $t_c = \frac{3\pi}{4}$. Therefore the optimal control u^* can be as (6.2.8).

Now in order to find the optimal cost J^* we have to find

$$J^* = J_1 + J_2. \quad (6.2.18)$$

where

$$J_1 = \frac{1}{2} \int_0^{\frac{3\pi}{4}} (x_2^2 - x_1^2) dt, \quad (6.2.19)$$

and

$$J_2 = \frac{1}{2} \int_{\frac{3\pi}{4}}^{2.985} (x_2^2 - x_1^2) dt. \quad (6.2.20)$$

To find J_1 the solution of equation (6.2.13) should be substituted for x_2 and x_1 in J_1 , therefore by substituting (6.2.14) and (6.2.15) into x_2 and x_1 in (6.2.19) respectively we get,

$$\begin{aligned} J_1 &= \frac{1}{2} \int_0^{\frac{3\pi}{4}} (\cos^2 t - \sin^2 t) dt, \\ &= \int_0^{\frac{3\pi}{4}} \left\{ \frac{1}{2}(1 + \cos 2t) - \frac{1}{2}(1 - \cos 2t) \right\} dt \\ &= \frac{1}{2} \int_0^{\frac{3\pi}{4}} \cos 2t dt, \\ &= \frac{1}{4} [\sin 2t]_0^{\frac{3\pi}{4}}, \\ &= -0.25. \end{aligned} \quad (6.2.21)$$

To find J_2 the solutions of equation (6.2.9) should be substituted for x_2 and x_1 in J_2 , therefore by substituting (6.2.10) and (6.2.11) into x_1 and x_2 in

(6.2.20) respectively we get,

$$\begin{aligned}
 J_2 &= \frac{1}{2} \int_{\frac{3\pi}{4}}^{2.985} \left\{ (-t + 1.649)^2 - \left(-\frac{1}{2}t^2 + 1.649t + 0.4021525 \right)^2 \right\} dt \\
 &= \frac{1}{2} \int_{\frac{3\pi}{4}}^{2.985} \left\{ -\frac{1}{4}t^4 + 1.649t^3 - 2.1213535t^2 - 1.9717011t + 2.5574744 \right\} dt \\
 &= \left[-\frac{1}{20}t^5 + 0.412250t^4 - 0.7071178t^3 - 0.9858506t^2 + 2.5574744t \right]_{\frac{3\pi}{4}}^{2.985} \\
 &= 0.2723751.
 \end{aligned} \tag{6.2.22}$$

Now substituting (6.2.21) and (6.2.22) into (6.2.18) we get the optimal cost $J^* = 0.0223751$ to 7 decimal places.

6.3 Numerical Solutions

6.3.1 The state and adjoint equations

The state equations are:

$$\left. \begin{aligned}
 \dot{x}_1 &= f_1 = x_2, & x_1(0) &= 0 \\
 \dot{x}_2 &= f_2 = u, & x_2(0) &= 1 \\
 \dot{x}_3 &= f_3 = \frac{1}{2}(x_2^2 - x_1^2), & x_3(0) &= 0
 \end{aligned} \right\} \tag{6.3.1}$$

Using the Runge-Kutta 4th order method for numerical solution of (6.3.1) we get,

$$\begin{aligned}
 x_{1,n+1} &= x_{1,n} + \frac{1}{6}(u_1 + 2u_2 + 2u_3 + u_4), \\
 x_{2,n+1} &= x_{2,n} + \frac{1}{6}(v_1 + 2v_2 + 2v_3 + v_4), \\
 x_{3,n+1} &= x_{3,n} + \frac{1}{6}(w_1 + 2w_2 + 2w_3 + w_4),
 \end{aligned}$$

where

$$\begin{aligned}
 u_1 &= hf_1(x_{1,n}, x_{2,n}, x_{3,n}) = hx_{2,n}, \\
 v_1 &= hf_2(x_{1,n}, x_{2,n}, x_{3,n}) = hu_n, \\
 w_1 &= hf_3(x_{1,n}, x_{2,n}, x_{3,n}) = \frac{h}{2}(x_{2,n}^2 - x_{1,n}^2), \\
 u_2 &= hf_1(x_{1,n} + \frac{1}{2}u_1, x_{2,n} + \frac{1}{2}v_1, x_{3,n} + \frac{1}{2}w_1) = h(x_{2,n} + \frac{1}{2}v_1), \\
 v_2 &= hf_2(x_{1,n} + \frac{1}{2}u_1, x_{2,n} + \frac{1}{2}v_1, x_{3,n} + \frac{1}{2}w_1) = hu_n, \\
 w_2 &= hf_3(x_{1,n} + \frac{1}{2}u_1, x_{2,n} + \frac{1}{2}v_1, x_{3,n} + \frac{1}{2}w_1) \\
 &= \frac{h}{2} \left[(x_{2,n} + \frac{1}{2}v_1)^2 - (x_{1,n} + \frac{1}{2}u_1)^2 \right], \\
 u_3 &= hf_1(x_{1,n} + \frac{1}{2}u_2, x_{2,n} + \frac{1}{2}v_2, x_{3,n} + \frac{1}{2}w_2) = h \left[x_{2,n} + \frac{1}{2}v_2 \right],
 \end{aligned}$$

$$\begin{aligned}
v_3 &= hf_2(x_{1,n} + \frac{1}{2}u_2, x_{2,n} + \frac{1}{2}v_2, x_{3,n} + \frac{1}{2}w_2) = hu_n, \\
w_3 &= hf_3(x_{1,n} + \frac{1}{2}u_2, x_{2,n} + \frac{1}{2}v_2, x_{3,n} + \frac{1}{2}w_2) \\
&= \frac{h}{2} \left[(x_{2,n} + \frac{1}{2}v_2)^2 - (x_{1,n} + \frac{1}{2}u_2)^2 \right], \\
u_4 &= hf_1(x_{1,n} + u_3, x_{2,n} + v_3, x_{3,n} + w_3) = h(x_{2,n} + v_3), \\
v_4 &= hf_2(x_{1,n} + u_3, x_{2,n} + v_3, x_{3,n} + w_3) = hu_n, \\
w_4 &= hf_3(x_{1,n} + u_3, x_{2,n} + v_3, x_{3,n} + w_3) \\
&= \frac{h}{2} \left[(x_{2,n} + v_3)^2 - (x_{1,n} + u_3)^2 \right].
\end{aligned}$$

The adjoint equations are:

$$\left. \begin{aligned}
\dot{\lambda}_1 &= f_1 = \lambda_3 x_1, & \lambda_1(t_f) &= 20(x_1(t_f) - 0.065), \\
\dot{\lambda}_2 &= f_2 = -\lambda_1 - \lambda_3 x_2, & \lambda_2(t_f) &= 20(x_2(t_f) + 1.33), \\
\dot{\lambda}_3 &= f_3 = 0, & \lambda_3(t_f) &= 1.
\end{aligned} \right\} \quad (6.3.2)$$

Using the Runge-Kutta 4th order method for numerical solutions of (6.3.1) we get,

$$\begin{aligned}
\lambda_{1,n} &= \lambda_{1,n+1} + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \\
\lambda_{2,n} &= \lambda_{2,n+1} + \frac{1}{6}(z_1 + 2z_2 + 2z_3 + z_4), \\
\lambda_{3,n} &= \lambda_{3,n+1} + \frac{1}{6}(y_1 + 2y_2 + 2y_3 + y_4),
\end{aligned}$$

where

$$\begin{aligned}
k_1 &= -hf_1(\lambda_{1,n+1}, \lambda_{2,n+1}, \lambda_{3,n+1}) = -h\lambda_{3,n+1}x_{1,n+1}, \\
z_1 &= -hf_2(\lambda_{1,n+1}, \lambda_{2,n+1}, \lambda_{3,n+1}) = -h(-\lambda_{1,n+1} - \lambda_{3,n+1}, x_{2,n+1}), \\
y_1 &= -hf_3(\lambda_{1,n+1}, \lambda_{2,n+1}, \lambda_{3,n+1}) = -h(0) = 0, \\
k_2 &= -hf_1(\lambda_{1,n+1} + \frac{1}{2}k_1, \lambda_{2,n+1} + \frac{1}{2}z_1, \lambda_{3,n+1} + \frac{1}{2}y_1), \\
&= -h \left[(\lambda_{3,n+1} + \frac{1}{2}y_1)x_{1,n+1} \right], \\
z_2 &= -hf_2(\lambda_{1,n+1} + \frac{1}{2}k_1, \lambda_{2,n+1} + \frac{1}{2}z_1, \lambda_{3,n+1} + \frac{1}{2}y_1), \\
&= -h \left[-(\lambda_{1,n+1} + \frac{1}{2}k_1) - (\lambda_{3,n+1} + \frac{1}{2}y_1)x_{2,n+1} \right], \\
y_2 &= -hf_3(\lambda_{1,n+1} + \frac{1}{2}k_1, \lambda_{2,n+1} + \frac{1}{2}z_1, \lambda_{3,n+1} + \frac{1}{2}y_1) = 0, \\
k_3 &= -hf_1(\lambda_{1,n+1} + \frac{1}{2}k_2, \lambda_{2,n+1} + \frac{1}{2}z_2, \lambda_{3,n+1} + \frac{1}{2}y_2) \\
&= -h \left[(\lambda_{3,n+1} + \frac{1}{2}y_2)x_{1,n+1} \right], \\
z_3 &= -hf_2(\lambda_{1,n+1} + \frac{1}{2}k_2, \lambda_{2,n+1} + \frac{1}{2}z_2, \lambda_{3,n+1} + \frac{1}{2}y_2), \\
&= -h \left[-(\lambda_{1,n+1} + \frac{1}{2}k_2) - (\lambda_{3,n+1} + \frac{1}{2}y_2)x_{2,n+1} \right], \\
y_3 &= -hf_3(\lambda_{1,n+1} + \frac{1}{2}k_2, \lambda_{2,n+1} + \frac{1}{2}z_2, \lambda_{3,n+1} + \frac{1}{2}y_2) = 0,
\end{aligned}$$

$$\begin{aligned}
k_4 &= -hf_1(\lambda_{1,n+1} + k_3, \lambda_{2,n+1} + z_3, \lambda_{3,n+1} + y_3), \\
&= -h[(\lambda_{3,n+1} + y_3)x_{1,n+1}], \\
z_4 &= -hf_2(\lambda_{1,n+1} + k_3, \lambda_{2,n+1} + z_3, \lambda_{3,n+1} + y_3), \\
&= -h[(-\lambda_{1,n+1} + k_3) - (\lambda_{3,n+1} + y_3)x_{2,n+1}], \\
y_4 &= -hf_3(\lambda_{1,n+1} + k_3, \lambda_{2,n+1} + z_3, \lambda_{3,n+1} + y_3) = 0.
\end{aligned}$$

6.4 Results and Discussion

Here, before proceeding to analyse the results it is worth pointing that, the value of k (the coefficient of penalty function) is an important factor in minimizing the optimal J , but since we were not able to find the value of k analytically, the tests were carried out using $k = 10$, as suggested by Edge and Powers [20.]

6.4.1 Gradient Method

The algorithm for the gradient in function space method applied to problem 3 is the same as that described in Chapter 4, Section 4.4.1, but for this problem $g = \lambda_2$.

In order to examine the efficiency of this method, when it is applied to the mentioned problem as usual a comparison is made, when the critical parameters, ε , h and u_0 are varied, with the analytical solution. Some criteria for accuracy of norms (Acc) have been set throughout this chapter. $\text{Acc} \leq 0.1$ and $\text{Acc} \leq 0.05$ were common in all methods, and also the best Acc that each method could achieve is included in those cases where the accuracy set could not be achieved. Here for GFS the best Acc was set to $\leq 2.0 \times 10^{-2}$.

Table (6.4.1), shows the effect of ε and N in achieving the optimal J with the best starting control $u_0 = 3.0$. The best J^* obtained was -0.00294302 to 8 decimal places, with $\|g\| = 1.891 \times 10^{-2}$ and the corresponding parameter values $\varepsilon = 0.008$, $N = 100$ or 200 with $m = 1000$. Choosing ε fairly small, between 0.007 and 0.00850 , can produce consistent results while N is sufficiently large, i.e. 50 or 100 or 200 , and as ε increases to 0.009 , then numerical instability occurs, resulting in large $\|g\|$ and J . Also for N too small, e.g. 10 , consistent results are obtained for ε between 0.007 and 0.008 , but instability occurs for $\varepsilon \geq 0.00850$. Here by selecting N as 100 we can achieve the optimal J^* in less computing time than $N = 200$.

Table (6.4.2), shows the effect of choice of u_0 on J , when Acc is ≤ 0.1 , for $N = 100$. It can be seen that it takes fewer iterations to get to the required accuracy, when u_0 is selected in the range $0.0 \leq u_0 \leq 1.0$.

Table (6.4.3), shows that the same effect occurs, when $\text{Acc} \leq 0.05$ and $N = 100$.

Table (6.4.4), shows the same effect, but for $\text{Acc} \leq 2.0 \times 10^{-2}$ and $N = 100$. Here the Acc can only be achieved, when u_0 is in the range $3.0 \leq u_0 \leq 4.0$.

Here for this method and other methods in this chapter, we have compared graphically the numerical solutions with the analytical ones, but it should be noted that the analytical graphs represented here are the ones, that were solved without the penalty function, whereas the numerical graphs are the ones solved with the penalty function with $k = 10$, and since by changing k in fact we are solving a different problem, therefore we expect a reasonable difference, even when the best optimal numerical solution is compared with the analytical one.

Fig (6.4.1), shows plots of control against time, for the analytical and numerical solutions, with $m = 10, 100$ and 1000 , respectively, using $N = 100$, $u_0 = 3.0$ and $\varepsilon = 0.008$. As can be seen from the figure, there are differences between the behaviour of the curves for different number of iterations. Although none of them could behave quite similarly to the analytical one, but as m increases to 1000 , the behaviour of the curve was closer to the analytical one than with the lower number of iterations.

Fig (6.4.2) compares the analytical with the numerical solutions for control, when the number of integration steps are $10, 50$ and 100 respectively, with their best corresponding step length factors, $0.007, 0.007$ and 0.008 , $m = 1000$ and $u_0 = 3.0$. Here also none of the curves behaved quite similarly to the analytical curve but the one that was closer to it was with $N = 100$.

Fig (6.4.3), demonstrates the effect on the control of using step length factors, $\varepsilon = 0.007, 0.008$ and 0.00850 , with 1000 iterations, when $u_0 = 3.0$ and $N = 100$. Here as can be seen from the figure, the curves of $\varepsilon = 0.008$ and $\varepsilon = 0.00850$, behaved fairly similarly, but by taking ε as 0.007 , although the pattern of behaviour up to time approximately 2.5 , was quite similar to the other two, but from there, as can be seen, it behaved differently. Here again, although none of the numerical curves behaved quite similarly to the analytical one, but the ones that were closer to it were, with $\varepsilon = 0.008$ and 0.00850 .

Fig (6.4.4), illustrates the effect of u_0 on optimal J for different values of m , when $N = 100$ and $\varepsilon = 0.008$. Here we can see from the graph that as m increases, the values of optimal J for all 3 starting controls, $u_0 = -1.0, u_0 = 1.0$ and $u_0 = 3.0$ converges to approximately near enough to 0.0 .

In view of the above results, we can see that the correct choice of initial control, can give a better value for optimal J , for higher degrees of accuracy (Acc) in fewer iterations and number of function evaluations and ultimately, computing time.

The interaction between the other two critical factors, ε and N , shows that the best combination exists with smaller ε and larger N , in this case

$\varepsilon = 0.008$ and $N = 100$, but if ε is taken too small say 0.007, then the convergence is worse. If the value of ε is taken too large, say, 0.009 the numerical instability will result.

In view of these comments, by selecting proper initial control and accurate enough integration steps, along with an appropriate step length factor, a speed up in the convergency to a better J^* is obtained. Also higher degrees of Acc are not obtainable from remote starting controls, but lower accuracies may be achieved even with remote initial control.

6.4.2 Steepest Descent

The algorithm for steepest descent applied to problem 3 is as described in Chapter 2, Section 2.2.2. The line search technique used for this method is the quadratic interpolation method of Powell which was described in Chapter 3 Section 3.4.2. The gradient, g , is the same as the one for gradient method in function space ((see (6.4.1)).

Here again we investigate the effect of step length factor, integration step and initial control on the solution.

Table (6.4.5), shows the effect of ε and N in obtaining J^* with the best starting control $u_0 = 0.0$. The best optimal J^* obtained was -0.00273340 to 8 decimal places with $\|g\| = 2.223 \times 10^{-2}$, $N = 100$ or 200 and $\varepsilon = 0.008$. By taking small step length factors in the range $0.007 \leq \varepsilon \leq 0.00850$, for larger N gave a better J^* . Here also by taking N as 100 the optimal J^* can be achieved in less computing time than $N = 200$.

Tables (6.4.6) to (6.4.8), show the effect of choice of u_0 on J^* for $\text{Acc} \leq 0.1$, $\text{Acc} \leq 0.05$ and finally the best Acc that could be achieved, i.e., $\text{Acc} \leq 0.023$ respectively, with $N = 100$.

From Table (6.4.6), it can be seen that for $\text{Acc} \leq 0.1$, taking u_0 as 0.0 or 0.5 results in getting J^* in fewer iterations than for other starting controls.

Table (6.4.7), show the same effects for $\text{Acc} \leq 0.05$. But as can be seen from Table (6.4.8), for higher degrees of accuracy, i.e., $\text{Acc} \leq 0.023$, some starting controls fail to converge. Here taking $u_0 = 0.0$ achieves this Acc in the smallest number of iterations.

Fig (6.4.5), shows the graph of time against control, when the numerical solutions are compared with the analytical one, for $m = 10, 500$ and 2000, with $N = 100$, $\varepsilon = 0.008$ and $u_0 = 0.0$. Here as can be seen from the plots of numerical solutions, as m increases to 2000, the behaviour of the curve is as close as it can get to the analytical solution.

Fig (6.4.6), compares the analytical solution for control, with the numerical ones, with the number of integration steps, 10, 50 and 100 respectively with their best corresponding ε 's of 0.008, 0.009 and 0.008, when $m = 2000$ and $u_0 = 0.0$. Here also we can observe that for the numerical solutions, with larger N 's, say 50 and 100, the behaviour of the curves are quite similar and closer to the analytical one, than with $N = 10$.

Fig (6.4.7), shows the effect on control of using step length factors, $\varepsilon = 0.007, 0.008$ and 0.00850 , with $N = 100$, $u_0 = 0.0$ and $m = 2000$. As can be seen from the graphs of numerical solutions, taking ε 's as 0.008 and 0.00850 , the curves behave, quite similar, with minor differences from $\varepsilon = 0.007$ and fairly close to the analytical curve.

Fig (6.4.8), demonstrates the effect of u_0 on optimal J for different values of m , when $N = 100$ and $\varepsilon = 0.008$. As can be seen as m increases we can get a better value of minimum J for all starting controls, but the best value for optimal J obtained with $u_0 = 0.0$ and the worst with $u_0 = -1.0$. The above results show that a proper choice of initial control is an important factor in achieving the best J^* in fewer iterations. The interaction between ε and N , shows that for small enough ε , and large enough N a better minimum J would be achieved. Also here for higher degrees of Acc, e.g., $\text{Acc} \leq 0.023$ remote starting controls would fail convergence, whereas for lower degrees of Acc say, $\text{Acc} \leq 0.1$ or ≤ 0.05 most starting controls would converge. In view of the results and discussions, the best minimum optimal value of J^* could be achieved with a proper starting control, along with sufficiently large N and small enough ε .

6.4.3 Fletcher-Reeves

The algorithm for the Fletcher-Reeves method applied to Problem 3 is as described in Chapter 2, Section 2.4.1. The calculation of the norms of the gradients are as given in Chapter 2, Section 2.10.1. The line search technique is the same as that for steepest descent in this Chapter, Section 6.4.2. The gradient, g is obtained in the way described in Section 6.4.1.

Table (6.4.9), shows the effect of ε and N in achieving the minimum J^* with the best starting control $u_0 = -1.0$. Best J^* was obtained as -0.00010353 to 8 decimal places with $\varepsilon = 0.008$, $\|g\| = 3.088 \times 10^{-2}$, $N = 100$ or 200 and $m = 47$. Here by increasing m up to a limit, not only J^* will not improve, but also it results in numerical instability. With large N , say 100, taking ε in the range $0.007 \leq \varepsilon \leq 0.008$ leads to a better value for J^* , but for $N = 10$, ε can be taken in the range $0.006 \leq \varepsilon \leq 0.009$ to produce consistent and fairly similar results. Selecting N as 100 as opposed to 200 could achieve the optimal J^* in less computing time.

Tables (6.4.10) to (6.4.12), show how varying u_0 would affect J^* for Acc's, ≤ 0.1 , ≤ 0.05 and, the best Acc possible in this case ≤ 0.031 , respectively, with $N = 100$. It can be seen that these accuracies can be obtained with u_0 ,

in the range $-1.0 \leq u_0 \leq -0.9$ in fewer number of iterations than for other starting controls. Also taking u_0 , in the range $0.0 \leq u_0 \leq 1.0$, none of the Acc's of ≤ 0.1 , ≤ 0.05 and ≤ 0.031 , could be achieved.

Fig (6.4.9), shows the comparison of the analytical solution, with the numerical ones, for control with $m = 5, 20$ and 47 , where $\varepsilon = 0.008$, $u_0 = -1.0$ and $N = 100$. As can be seen from the plots, none of the numerical curves matches the analytical one, but as m increases the numerical curves start to decline to control, -1 at time approximately 1.7 , and stay there up to the find time, $t_f = 2.985$.

Fig (6.4.10), compares the analytical solution, for control, with the numerical ones for different N 's, i.e., $N = 10$, with its best correspondings $\varepsilon = 0.007$ and $m = 48$, also $N = 50$, with its best correspondings $\varepsilon = 0.008$ and $m = 47$, all with starting controls $u_0 = -1.0$. Here also none of the numerical curves behaved similar to the analytical one, but for larger N , say 100 , the slope of the control is downward till it gets to time approximately 1.7 and $u = -1.0$, where it stays there up to $t_f = 2.985$.

Fig (6.4.11), demonstrates the curve of time against control, when we compare the analytical solution with the numerical ones, where the step length factors are $\varepsilon = 0.006$, with $m = 60$, $\varepsilon = 0.008$, with $m = 10$ and $\varepsilon = 0.009$ with $m = 10$, for $N = 100$ and $u_0 = -1.0$. Here, it can be seen, that the numerical curves with $\varepsilon = 0.006$ and $\varepsilon = 0.009$, behaved quite similarly, and they both had the same pattern of downward slope for control up to time approximately 2.5 with $u = -1.0$, and at this point the curves stay the same up to $t_f = 2.985$. But for $\varepsilon = 0.008$, the downward slope continues up to time approximately 2.3 with control -1.0 , and from there it stays the same till time $t_f = 2.985$.

Fig (6.4.12), shows the effect of u_0 on optimal J , for different values of m , when $N = 100$ and $\varepsilon = 0.008$. As can be seen from the graphs, the value of optimal J converges near enough to 0.0 , as m increases for all the starting controls, $u_0 = -1.1$, -1.0 and -0.5 .

In view of the results and comments above, the recommendations for obtaining the best possible J^* is to select u_0 in the range $[-1.0, -0.9]$, with N sufficiently large and ε relatively small in the range $[0.007, 0.008]$. Here we should note that for distant initial controls, i.e., u_0 's in the range $[0.0, 1.0]$, even for lower Acc's we could not achieve convergence.

6.4.4 Polak-Ribière

The algorithm for the Polak-Ribière method is as given in Chapter 2, Section 2.5. The line search technique, the calculation of the norms and g are as given for FR in this Chapter, Section 6.4.3.

Table (6.4.13), shows the effect of ε and N on the minimum J^* . Here when the Acc and m are considered, the best minimum J^* could be found as -0.00268271 to 8 decimal places, with the best starting control, $u_0 = 0.0$, $\|g\| = 2.30 \times 10^{-2}$ and corresponding parameter values, $\varepsilon = 0.007$, $N = 100$ or 200 and $m = 2000$. Here as can be seen from the Table for all N 's, not selecting a proper ε results in numerical instability as the number of iterations increases. Also selecting N as 100 can achieve optimal J^* in less computing time than $N = 200$.

Tables (6.4.14) to (6.4.16), show the effect of selecting the initial control on the minimum J^* for different Acc's, ≤ 0.01 , ≤ 0.05 and the best possible that could be obtained $\text{Acc} \leq 0.023$ with $N = 100$. Here as can be seen from Tables (6.4.14) and (6.4.15), for the lower Acc's of ≤ 0.1 or ≤ 0.05 , selecting $u_0 = 0.5$ could achieve convergence in fewest iterations, also fewer function evaluations, and less computing time, than other u_0 's, whereas for higher Acc of ≤ 0.023 from Table (6.4.16), we can see that selecting $u_0 = 0.0$, could achieve that, better than other u_0 's.

Fig (6.4.13), shows the graph of time against control, comparing the analytical solutions with the numerical ones, with $m = 100$, 500 and 2000 , where $\varepsilon = 0.007$, $N = 100$ and $u_0 = 0.0$. As can be seen from the plots, although none of the numerical curves are too close to the analytical one, as m increases to 2000 we can see that, compared with fewer iterations, the behaviour gets closer to the analytical one.

Fig (6.4.14), shows similar graphs for $N = 10$, 50 and 100 respectively, with their best corresponding step length factors of 0.00750 , 0.008 and 0.007 , where $m = 2000$ and $u_0 = 0.0$. As clearly can be seen from the plots the closest behaviour of the numerical curves to the analytical one was with $N = 100$.

Fig (6.4.15), shows the curve of time against the control, when the analytical solution is compared with the numerical ones, where $\varepsilon = 0.006$, 0.007 and 0.008 , respectively, with corresponding number of iterations 13 , 2000 and 34 , where $N = 100$ and $u_0 = 0.0$. Here as can be seen the closest numerical curve to the analytical one is with $\varepsilon = 0.007$.

Fig (6.4.16), demonstrates the effect of u_0 on optimal J , for different number of iterations, when $N = 100$ and $\varepsilon = 0.007$. As can be seen from the plots, selecting u_0 as 0.5 , results in a better optimal J up to approximately 300 iterations, but for m in the approximate range of $300 < m \leq 1300$, selecting $u_0 = 0.0$, leads to a better convergence for optimal J , and for higher number of iterations > 1300 , both $u_0 = 0.0$ and $u_0 = 0.5$, converge to similar values for optimal J .

In view of results obtained above in order to find the best minimum J^* , a proper u_0 should be selected in this case in the range $[0.0, 0.5]$. Also a suitable large N , with relatively small ε , where ε should be chosen with a great care, since it plays a sensitive role in avoiding numerical instabilities.

We should also note that for higher accuracy, i.e., $\text{Acc} \leq 0.023$, selecting distant u_0 's would not meet the requirements of convergency.

6.4.5 Hybrid 1

The algorithm for Hybrid 1 method is described in Chapter 2, Section 2.6.1. The line search techniques, the calculation of the norms and g are as given for FR in this Chapter, Section 6.4.3.

Table (6.4.17), shows the effect of ε and N in achieving the best minimum J^* . Considering the number of iterations and Acc the best J^* was obtained as -0.00327211 to 8 decimal places, with $\|g\| = 2.0 \times 10^{-2}$, $m = 1147$, $N = 100$ or 200 and $\varepsilon = 0.006$. Here for large enough N , say 100 taking value of ε in the range $[0.006, 0.007]$ can produce minimum J^* in fewer iterations, than taking ε too small, say 0.005 or larger say 0.008. When N is too small e.g. 10, numerical instability occurs, when ε is in the range $[0.006, 0.008]$, and only for ε small enough, say 0.005 can it be avoided. Here also selecting $N = 100$ also can achieve the optimal J^* in less computing time than $N = 200$.

Tables (6.4.18) to (6.4.20), show the effect of selecting u_0 on the minimum J^* for different Acc 's ≤ 0.1 , ≤ 0.05 and $\leq 2.0 \times 10^{-2}$ with $N = 100$. To achieve the $\text{Acc} \leq 0.1$ or ≤ 0.05 , taking the initial control in the range $[-1.0, -0.5]$, can produce minimum J^* in fewer iterations than other starting controls, although it should be pointed out that, the minimum obtained for J^* is not as good as when the starting control is selected in the range $[1.0, 2.0]$.

To obtain higher Acc say $\leq 2.0 \times 10^{-2}$, selecting u_0 in the range $[1.0, 2.0]$ can produce better minimum J^* in fewer iterations, than selecting u_0 in the range $[-0.5, 0.0]$, and selecting u_0 as -1.0 or 0.5 does not satisfy the accuracy required.

Fig (6.4.17), shows the graph of time against control, comparing the analytical solution with the numerical ones, with $m = 10, 500$ and 1100 , with $\varepsilon = 0.006$, $u_0 = 1.0$ and $N = 100$. As can be seen none of the numerical curves behaves similar to the analytical one, but as m increases the shapes of the curve could get closer to the analytical one.

Fig (6.4.18), demonstrates the effect of N on control, when the analytical solution is compared with the numerical ones, with $N = 10, 50$ and 100 , respectively, with corresponding (ε, m) of $(0.005, 930)$, $(0.007, 1300)$ and $(0.006, 1100)$ are with $u_0 = 1.0$. As can be seen again, none of the numerical curves are similar to the analytical one, but as N gets larger, the numerical curves become closer to each other.

Fig (6.4.19), shows similarly the effect of ε on the control, with $\varepsilon = 0.005, 0.006$ and 0.007 , with $m = 1100$, $N = 100$ and $u_0 = 1.0$. Here as can be seen none of the numerical solutions behave similarly to the analytical

one, and they are all different from each other, but those with behaviour closer to the analytical curve are with $\varepsilon = 0.006$ and $\varepsilon = 0.007$.

Fig (6.4.20), shows how the change in u_0 affects the optimal J for different numbers of iterations, when $N = 100$ and $\varepsilon = 0.006$. As can be seen with starting controls taken as 1.0 and 1.5, both converge to the same value for optimal J , and with $u_0 = 0.0$, for increased number of iterations, convergence is fairly close to the same value of optimal J as the other two.

In view of the results obtained above, the best possible minimum J^* could be achieved by selecting a proper critical control in the range $[1.0, 2.0]$, with a sufficiently large N and relatively small ε , in the range $[0.006, 0.007]$. It should also be noted that not all starting controls can achieve convergence for higher accuracy of norms.

6.4.6 Angle Test Hybrid

The algorithm for the Angle test hybrid is described in Chapter 2, Section 2.7. The line search technique, and calculation of the norms and gradient, g , are as given for FR in this Chapter, Section 6.4.3. For this method we had to consider the effect of another parameter $\tau > 0$, as well as the usual parameters.

Table (6.4.21), shows the effect of ε and N on finding the minimum J^* . Here for N too small say 10, no matter what value of ε is selected in the range $[0.005, 0.008]$, numerical instability occurs after a few iterations. But for sufficiently larger N , say 100 with ε small enough in the range $[0.005, 0.006]$, we can find a minimum for J^* . The best minimum J^* was -0.00324996 to 8 decimal places with $\tau = -0.00001$, $m = 1135$, $\varepsilon = 0.006$, $N = 100$ and the best starting control $u_0 = 0.5$. Taking $N = 100$ achieves optimal J^* in less computing time than $N = 200$.

Tables (6.4.22) to (6.4.24), show the effect of the selection of the initial control on minimum J^* for $\text{Acc} \leq 0.1$, ≤ 0.05 and the best possible that could be obtained $\leq 2.0 \times 10^{-2}$, respectively, with $N = 100$, $\tau = 0.000001$. To obtain accuracies of ≤ 0.1 and ≤ 0.05 , selecting u_0 in the range $[-1.0, -0.5]$, would achieve these in fewer iterations, than other starting controls, but the value of J^* is not as good as those with the starting controls in the range $[0.0, 1.0]$. However, for higher degrees of accuracy $\leq 2.0 \times 10^{-2}$, selecting u_0 in the range $[0.0, 0.5]$ would ensure a better minimum J^* with fewer iterations, fewer function evaluations and less computing time than other starting controls. Some starting controls, i.e., -1.0 , 1.0 and 2.0 , never achieved accuracy $\leq 2.0 \times 10^{-2}$.

Fig (6.4.21), shows the graph of time against control, comparing the analytical solution with the numerical ones, with $m = 10$, 500 and 1100 , when $\varepsilon = 0.006$, $u_0 = 0.5$, $\tau = 0.000001$ and $N = 100$. As can be seen from the plots, none of the numerical solutions behaved similarly to the analytical

one, but we could see some resemblance with higher number of iterations, i.e., $m = 1100$, between the numerical and the analytical curves.

Fig (6.4.22), gives the graph of control showing the effect of N taken as 10, 50 and 100 respectively, with corresponding (ε, m) of $(0.005, 150)$, $(0.005, 1500)$ and $(0.006, 1100)$, when $u_0 = 0.5$ and $\tau = 0.000001$. Here also can be seen that although none of the numerical curves behaved similarly to the analytical one, the one closest to it was with $N = 100$.

Fig (6.4.23), in the same way demonstrates the effect of ε on control, with $\varepsilon = 0.005, 0.006$ and 0.007 , when $u_0 = 0.5$, $N = 100$, $\tau = 0.000001$ and $m = 1100$. Here the curves, that behaved closer to the analytical solution were with $\varepsilon = 0.005$ and 0.006 .

Fig (6.4.24), shows the effect of change in u_0 , on optimal J for different numbers of iterations, when $\varepsilon = 0.006$, $\tau = 0.000001$ and $N = 100$. As can be seen from the plots, the convergence of different u_0 's for various m 's can be assessed as follows:

For m in the range $[0, 200]$, $u_0 = -0.5$ obtained better values for optimal J than $u_0 = 0.5$, $u_0 = 1.0$ and in turn, $u_0 = 0.5$, performed better than $u_0 = 1.0$.

For m in the range $[200, 600]$, $u_0 = 0.5$ obtained better values for optimal J than $u_0 = -0.5$ and $u_0 = 1.0$, and in turn, $u_0 = -0.5$, performed better than $u_0 = 1.0$.

For m in the range $[600, 800]$, $u_0 = 0.5$ obtained better values for optimal J than $u_0 = -0.5$ and $u_0 = 1.0$, and $u_0 = -0.5$ and $u_0 = 1.0$ performed similarly. Finally, for m in the range $[800, 1100]$, $u_0 = 0.5$ and $u_0 = 1.0$, performed similarly and obtained better values for optimal J than $u_0 = -0.5$.

In view of all the above results, the best possible minimum J^* could be achieved by selecting a proper initial control, where here for higher degrees of accuracy say, $\text{Acc} \leq 2.0 \times 10^{-2}$, the control can be selected in the range $[0.0, 0.5]$, with a sufficiently large N , i.e., 100 and relatively small ε in the range $[0.005, 0.006]$. We also tested different values of $\tau = 0.01, 0.0001$ and 0.000001 , and the effect on obtaining the minimum J^* was practically negligible. We should also note that on obtaining greater Acc, care should be taken in selecting u_0 , since some would fail to convergence.

6.4.7 Hybrid 3

The algorithm for the Hybrid 3 method can also be found in Chapter 2, Section 2.8. The calculation of the norms, the line search and also g are as for FR in this Chapter, Section 6.4.3. The effect of new parameters, $\lambda > 0$ and $\mu < \frac{1}{2}$ had to be considered for this method.

Table (6.4.25), shows the effect of ε and N in achieving the minimum J^* . Here for N small say 10, taking ε small say 0.005 can produce consistent results, but when ε is taken larger say in the range $[0.006, 0.008]$, numerical instability occurs. For N larger say 100, the best minimum for J^* can be obtained when ε is taken in the range $[0.006, 0.007]$ and if ε is too small say 0.005, it takes more iterations to get a minimum. Also for ε relatively large say 0.008, numerical instability results. Here the best minimum J^* obtained was -0.00327211 , with $\|g\| = 2.0 \times 10^{-2}$ and corresponding parameter values, $\varepsilon = 0.006$, $N = 100$, or $N = 200$ $m = 1100$, $\lambda = 0.000001$, $\mu = 0.4999$, and the best starting control $u_0 = 1.0$. But selecting $N = 100$ can achieve optimal J^* in less computing time than $N = 200$.

Tables (6.4.26) to (6.4.28), show how the selection of u_0 can affect the minimum J^* for $\text{Acc} \leq 0.1$, ≤ 0.05 and the best that can be obtained $\leq 2.0 \times 10^{-2}$ with $N = 100$ $\lambda = 0.000001$ and $\mu = 0.4999$.

Selecting u_0 in the range $[-1.0, -0.5]$ would achieve $\text{Acc} \leq 0.1$ and ≤ 0.05 , in fewer iterations than other starting controls, but for a higher degree of accuracy, i.e., $\text{Acc} \leq 2.0 \times 10^{-2}$, selecting u_0 in the range $[1.0, 2.0]$ will achieve that. Here taking u_0 as 0.5 could not achieve the lower accuracy of ≤ 0.1 and for higher $\text{Acc} \leq 2.0 \times 10^{-2}$, taking u_0 as -1.0 and 0.5 could not meet this requirement.

Fig (6.4.25), shows the graph of time against control, comparing the analytical solution with the numerical ones, with $m = 10, 500$ and 1100 , where $u_0 = 1.0$, $N = 100$, $\varepsilon = 0.006$, $\lambda = 0.000001$ and $\mu = 0.4999$. As can be seen none of the numerical curves behaves quite the same as the analytical one, but as m increases we can see some improvements with the numerical curves getting closer to the analytical one.

Fig (6.4.26), demonstrates the effect of N on control, when the analytical solution is compared with the numerical ones, with $N = 10, 50$ and 100 respectively, with their best corresponding ε of 0.005, 0.007 and 0.006, where $m = 1100$, $u_0 = 1.0$, $\lambda = 0.000001$ and $\mu = 0.4999$. Here we can see from the plots that as N increases the numerical curves are closer to the analytical one.

Fig (6.4.27) illustrates the effect of ε on control when the analytical solution is compared with the numerical ones, with $\varepsilon = 0.005, 0.006$ and 0.007 , when $N = 100$, $u_0 = 1.0$, $m = 1100$, $\lambda = 0.000001$ and $\mu = 0.4999$. Here also, the numerical solutions closest to the analytical curve, are with $\varepsilon = 0.006$ and 0.007 , where for t small, taking $\varepsilon = 0.007$, the curve behaves closer to the analytical one than $\varepsilon = 0.006$, but when it gets to time approximately 2.0 at control -1.0 , then from there on the behaviour of the curve with $\varepsilon = 0.006$ is closer to the analytical one than $\varepsilon = 0.007$.

Fig (6.4.28), shows the effect of u_0 on optimal J for different numbers of iterations, when $N = 100$, $\varepsilon = 0.006$, $\lambda = 0.000001$ and $\mu = 0.4999$. As can be seen from the graph as m increases the optimal J for all starting controls,

converges towards the same value, but taking u_0 as 1.0 or 1.5 can achieve this in fewer iterations than $u_0 = 0.0$.

In view of the results and comments given above, the best minimum J^* could be achieved by selecting a proper initial control, in this case 1.0, with a sufficient large N , i.e., 100 or 200 and relatively small $\varepsilon = 0.006$. Here λ and μ with different values of $\lambda = 0.01, 0.001$ and 0.000001 and $\mu = 0.15, 0.35$ and 0.4999 were tested, but their effect on minimizing J^* was practically negligible. Also in order to obtain a required Acc, special care should be taken in selecting u_0 , since if a distant control is selected it fails to convergence.

6.5 Summary of the Results

Summary of the results could be found in Table (6.5.1) and also the comparisons could be seen in Fig (6.5.1), when N was taken as 100, for ATH, $\tau = 0.00001$, and for H3, $\lambda = 0.000001$ and $\mu = 0.4999$. Considering all the effects of convergency for minimum J^* , i.e., the number of iterations, Acc and numerical stability, the methods performed as follows:

At $u_0 = -1.0$, FR performed the best, then in order of performance H1, ATH and H3 performed the same, then SD, PR and finally GFS.

At $u_0 = 0.0$, H1 and H3 performed the same and the best, then ATH, PR, SD and GFS. Numerical instability occurred for FR.

At $u_0 = 0.5$, in order of preference ATH performed the best, then PR, SD and finally GFS. The FR, H1 and H3 produced unstable results.

At $u_0 = 1.0$, H1 and H3 performed the same and the best, then in order of preference SD, GFS, ATH and finally PR. FR produced numerically unstable results.

At $u_0 = 3.0$, H1 and H3 performed the same and the best. Then in order of preference of performance, GFS, SD, PR and finally ATH. Once again FR produced numerically unstable results.

6.6 Conclusion

In this chapter we have tested the problem numerically using $k = 10$ and since by varying this value, i.e., the coefficient of the penalty function we practically solve a different problem, therefore the solution for minimizing J^* may not be the same as those obtained when the problem was solved analytically, without the penalty function. But however still a comparison is valid since k has been fixed the same for all the seven methods of optimization.

We could see that taking the initial control $u_0 = 3.0$ produced a better minimum J^* for GFS, and also in the same way, taking $u_0 = 0.0$ for SD and PR, $u_0 = -1.0$ for FR, $u_0 = 1.0$ for H1 and H3 and finally $u_0 = 0.5$ for ATH, taking into consideration the Acc, m , NFE and Cputime.

The common factors for obtaining stable numerical results in all these methods were choice of suitable large integration step (N) and small enough step length factor (ε).

Thus if the number of iterations, as well as the convergency, is going to be considered when the best results are obtained for each method, with the best possible initial control, step length factor and integration step, the best performances may be selected in the following order:

H3 and H1 performed the same as each other and the best, then in order of preference of performances, ATH, GFS, SD, PR and finally FR.

TABLE (6.4.1):Results of GFS with varying ϵ and N.

N ϵ	10	50	100	200
0.005	J*=-0.00284170 m=3000 $\ g\ =4.612 \times 10^{-2}$ Cputime=1 NFE=3001	J*=-0.00282311 m=1200 $\ g\ =2.301 \times 10^{-2}$ Cputime<1 NFE=1201	J*=-0.00251543 m=1000 $\ g\ =3.136 \times 10^{-2}$ Cputime=3 NFE=1001	J*=-0.00251764 m=1000 $\ g\ =3.024 \times 10^{-2}$ Cputime=5 NFE=1001
0.007	J*=-0.00285790 m=3000 $\ g\ =4.59 \times 10^{-2}$ Cputime=1 NFE=3001	J*=-0.00282821 m=1200 $\ g\ =2.132 \times 10^{-2}$ Cputime<1 NFE=1201	J*=-0.00270756 m=1000 $\ g\ =2.023 \times 10^{-2}$ Cputime=3 NFE=1001	J*=-0.00270757 m=1000 $\ g\ =2.012 \times 10^{-2}$ Cputime=5 NFE=1001
0.008	J*=-0.00283535 m=3000 $\ g\ =4.607 \times 10^{-2}$ Cputime=1 NFE=3001	J*=-0.00282291 m=1200 $\ g\ =2.306 \times 10^{-2}$ Cputime=1 NFE=1201	J*=-0.00294302 m=1000 $\ g\ =1.891 \times 10^{-2}$ Cputime=3 NFE=1001	J*=-0.00294302 m=1000 $\ g\ =1.891 \times 10^{-2}$ Cputime=5 NFE=1001
0.00850	J*=2.48713207 m=3000 $\ g\ =33.01036$ Cputime=1 NFE=3001	J*=-0.00280150 m=1200 $\ g\ =2.33 \times 10^{-2}$ Cputime=1 NFE=1201	J*=-.00294046 m=1000 $\ g\ =1.999 \times 10^{-2}$ Cputime=3 NFE=1001	J*=-.00295156 m=1000 $\ g\ =1.895 \times 10^{-2}$ Cputime=5 NFE=1001
0.009	J*=36.6976158 m=3000 $\ g\ =112.618$ Cputime=1 NFE=3001	J*=23.5867701 m=1200 $\ g\ =99.0522$ Cputime=1 NFE=1201	J*=9.24522591 m=1000 $\ g\ =62.9244$ Cputime=3 NFE=1001	J*=9.24511231 m=1000 $\ g\ =62.8551$ Cputime=5 NFE=1001

TABLE (6.4.2):Results of GFS with change in u_0 for $ACC \leq 0.1$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.0	132	0.00056562	9.897×10^{-2}	133	0.008	<1
-0.5	141	0.00056321	9.876×10^{-2}	142	0.008	<1
0.0	51	0.00019564	9.646×10^{-2}	52	0.008	<1
0.5	62	0.00029654	9.517×10^{-2}	63	0.008	<1
1.0	72	0.00020847	9.597×10^{-2}	73	0.008	<1
2.0	87	0.00012814	9.898×10^{-2}	88	0.008	<1
3.0	99	0.00001040	9.639×10^{-2}	100	0.008	<1
4.0	95	-0.00067814	9.741×10^{-2}	96	0.008	<1

TABLE (6.4.3):Results of GFS with change in u_0 for $ACC \leq 0.05$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.0	176	0.00007013	4.983×10^{-2}	177	0.008	<1
-0.5	185	0.00007133	4.970×10^{-2}	186	0.008	<1
0.0	72	-0.00010746	4.925×10^{-2}	73	0.008	<1
0.5	74	-0.00006646	4.902×10^{-2}	75	0.008	<1
1.0	85	-0.00016333	4.931×10^{-2}	86	0.008	<1
2.0	103	-0.00034711	4.802×10^{-2}	104	0.008	<1
3.0	114	0.00102974	4.946×10^{-2}	115	0.008	<1
4.0	109	-0.00067814	4.861×10^{-2}	110	0.008	<1

TABLE (6.4.4):Results of GFS with change in u_0 for $ACC \leq 2.0 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.0	—	—	—	—	0.008	—
-0.5	—	—	—	—	0.008	—
0.0	—	—	—	—	0.008	—
0.5	—	—	—	—	0.008	—
1.0	—	—	—	—	0.008	—
2.0	—	—	—	—	0.008	—
3.0	652	-0.00227928	1.99993×10^{-2}	653	0.008	1
4.0	986	-0.0027503	1.9998×10^{-2}	987	0.008	4

TABLE (6.4.5):Results of SD with varying ϵ and N.

N ϵ	10	50	100	200
0.005	J*=-0.00208572 m=1975 $\ g\ =4.731 \times 10^{-2}$ Cputime=1 NFE=7856	J*=-0.00232015 m=1971 $\ g\ =2.461 \times 10^{-2}$ Cputime=6 NFE=7849	J*=-0.00236213 m=1971 $\ g\ =2.237 \times 10^{-2}$ Cputime=14 NFE=7849	J*=-0.00244313 m=1970 $\ g\ =2.236 \times 10^{-2}$ Cputime=25 NFE=7838
0.007	J*=-0.00213163 m=1964 $\ g\ =4.708 \times 10^{-2}$ Cputime=1 NFE=7912	J*=-0.00240197 m=1959 $\ g\ =2.444 \times 10^{-2}$ Cputime=5 NFE=7821	J*=-0.00244705 m=1960 $\ g\ =2.291 \times 10^{-2}$ Cputime=14 NFE=7825	J*=-0.00244711 m=1960 $\ g\ =2.282 \times 10^{-2}$ Cputime=25 NFE=7825
0.008	J*=-0.00230381 m=1951 $\ g\ =4.646 \times 10^{-2}$ Cputime=1 NFE=7796	J*=-0.00251521 m=1856 $\ g\ =2.43 \times 10^{-2}$ Cputime=5 NFE=7811	J*=-0.00273340 m=2000 $\ g\ =2.223 \times 10^{-2}$ Cputime=14 NFE=8004	J*=-0.00273340 m=2000 $\ g\ =2.223 \times 10^{-2}$ Cputime=25 NFE=8004
0.00850	J*=-0.0022732 m=1985 $\ g\ =4.656 \times 10^{-2}$ Cputime=1 NFE=7942	J*=-0.00260150 m=1862 $\ g\ =2.40 \times 10^{-2}$ Cputime=6 NFE=7838	J*=-0.00271658 m=2000 $\ g\ =2.233 \times 10^{-2}$ Cputime=14 NFE=8004	J*=-0.00271754 m=2000 $\ g\ =2.233 \times 10^{-2}$ Cputime=25 NFE=8004
0.009	J*=1.66710067 m=12 $\ g\ =27.3374$ Cputime<1 NFE=26	J*=-0.00271616 m=1875 $\ g\ =2.376 \times 10^{-2}$ Cputime=6 NFE=7856	J*=5.67360449 m=15 $\ g\ =50.23397$ Cputime<1 NFE=35	J*=5.67351559 m=15 $\ g\ =50.23396$ Cputime=1 NFE=35

TABLE (6.4.6):Results of SD with change in u_0 for $ACC \leq 0.1$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.0	180	0.00057073	9.906×10^{-2}	724	0.008	1
-0.5	275	0.00058569	9.984×10^{-2}	1104	0.008	1
0.0	139	0.00057829	9.953×10^{-2}	560	0.008	2
0.5	143	0.00057110	9.838×10^{-2}	576	0.008	1
1.0	152	0.00058642	9.911×10^{-2}	612	0.008	1
2.0	164	0.00060927	9.994×10^{-2}	660	0.008	1
3.0	168	0.00061235	9.924×10^{-2}	676	0.008	1
4.0	174	0.00061537	9.858×10^{-2}	700	0.008	1

TABLE (6.4.7):Results of SD with change in u_0 for $ACC \leq 0.05$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.0	240	0.00006898	4.959×10^{-2}	964	0.008	1
-0.5	362	0.00007615	4.999×10^{-2}	1452	0.008	3
0.0	182	0.00007302	4.940×10^{-2}	732	0.008	1
0.5	185	0.00007970	4.952×10^{-2}	744	0.008	1
1.0	196	0.00008715	4.970×10^{-2}	788	0.008	1
2.0	210	0.00010089	4.995×10^{-2}	844	0.008	2
3.0	213	0.00010977	4.981×10^{-2}	856	0.008	2
4.0	220	0.00011542	4.943×10^{-2}	884	0.008	2

TABLE (6.4.8):Results of SD with change in u_0 for $ACC \leq 0.023$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.0	—	—	—	—	0.008	—
-0.5	—	—	—	—	0.008	—
0.0	1609	-0.00242712	2.30×10^{-2}	6440	0.008	11
0.5	—	—	—	—	0.008	—
1.0	1791	-0.00243135	2.30×10^{-2}	7168	0.008	13
2.0	1848	-0.00243241	2.30×10^{-2}	7396	0.008	13
3.0	1838	-0.00243313	2.30×10^{-2}	7356	0.008	13
4.0	1872	-0.00243298	2.292×10^{-2}	7492	0.008	14

TABLE (6.4.9):Results of FR with varying ε and N.

N ε	10	50	100	200
0.005	$J^*=0.00007521$ $m=60$ $\ g\ =5.322 \times 10^{-2}$ Cputime<1 NFE=244	$J^*=-0.00005802$ $m=60$ $\ g\ =3.361 \times 10^{-2}$ Cputime<1 NFE=244	$J^*=-0.00004723$ $m=60$ $\ g\ =3.241 \times 10^{-2}$ Cputime<1 NFE=244	$J^*=-0.00004723$ $m=60$ $\ g\ =3.241 \times 10^{-2}$ Cputime=1 NFE=244
0.006	$J^*=0.00007483$ $m=60$ $\ g\ =5.312 \times 10^{-2}$ Cputime<1 NFE=244	$J^*=-0.00005811$ $m=48$ $\ g\ =3.279 \times 10^{-2}$ Cputime<1 NFE=196	$J^*=-0.00004752$ $m=60$ $\ g\ =3.237 \times 10^{-2}$ Cputime<1 NFE=244	$J^*=-0.00004753$ $m=60$ $\ g\ =3.237 \times 10^{-2}$ Cputime=1 NFE=244
0.007	$J^*=-0.00003722$ $m=48$ $\ g\ =4.759 \times 10^{-2}$ Cputime<1 NFE=196	$J^*=-0.00003566$ $m=60$ $\ g\ =3.345 \times 10^{-2}$ Cputime<1 NFE=244	$J^*=-0.00009514$ $m=48$ $\ g\ =3.088 \times 10^{-2}$ Cputime<1 NFE=196	$J^*=-0.00009514$ $m=47$ $\ g\ =3.088 \times 10^{-2}$ Cputime=1 NFE=192
0.008	$J^*=0.00007483$ $m=60$ $\ g\ =5.312 \times 10^{-2}$ Cputime<1 NFE=244	$J^*=-0.00005942$ $m=45$ $\ g\ =3.265 \times 10^{-2}$ Cputime<1 NFE=184	$J^*=-0.00010353$ $m=47$ $\ g\ =3.088 \times 10^{-2}$ Cputime<1 NFE=192	$J^*=-0.00010353$ $m=47$ $\ g\ =3.088 \times 10^{-2}$ Cputime=1 NFE=192
0.00850	$J^*=0.00007245$ $m=60$ $\ g\ =5.288 \times 10^{-2}$ Cputime<1 NFE=244	$J^*=-0.00003566$ $m=60$ $\ g\ =3.345 \times 10^{-2}$ Cputime<1 NFE=244	$J^*=-0.00004752$ $m=60$ $\ g\ =3.237 \times 10^{-2}$ Cputime<1 NFE=244	$J^*=-0.00004754$ $m=60$ $\ g\ =3.236 \times 10^{-2}$ Cputime=1 NFE=244
0.009	$J^*=0.00007564$ $m=60$ $\ g\ =5.320 \times 10^{-2}$ Cputime<1 NFE=244	$J^*=-0.00003556$ $m=60$ $\ g\ =3.346 \times 10^{-2}$ Cputime<1 NFE=244	$J^*=-0.00004818$ $m=60$ $\ g\ =3.234 \times 10^{-2}$ Cputime<1 NFE=244	$J^*=-0.00004821$ $m=60$ $\ g\ =3.232 \times 10^{-2}$ Cputime=1 NFE=244

TABLE (6.4.10):Results of FR with change in u_0 for $ACC \leq 0.1$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.1	34	0.00049759	9.540×10^{-2}	140	0.008	<1
-1.0	31	0.00053036	9.734×10^{-2}	128	0.008	<1
-0.9	32	0.00051206	9.525×10^{-2}	132	0.008	<1
-0.6	35	0.00049006	9.282×10^{-2}	144	0.008	<1
-0.5	41	0.00050590	9.446×10^{-2}	168	0.008	<1
0.0	—	—	—	—	—	—
0.5	—	—	—	—	—	—
1.0	—	—	—	—	—	—

TABLE (6.4.11):Results of FR with change in u_0 for $ACC \leq 0.05$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.1	42	0.00008683	4.647×10^{-2}	172	0.008	<1
-1.0	39	0.00007492	4.547×10^{-2}	160	0.008	<1
-0.9	39	0.00010189	4.923×10^{-2}	160	0.008	<1
-0.6	42	0.00010246	4.870×10^{-2}	172	0.008	<1
-0.5	49	0.00010913	4.846×10^{-2}	200	0.008	<1
0.0	—	—	—	—	0.008	—
0.5	—	—	—	—	0.008	—
1.0	—	—	—	—	0.008	—

TABLE (6.4.12):Results of FR with change in u_0 for $ACC \leq 0.031$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.1	51	-0.00010108	3.064×10^{-2}	208	0.008	<1
-1.0	47	-0.00010353	3.088×10^{-2}	192	0.008	<1
-0.9	48	-0.00010692	3.091×10^{-2}	196	0.008	<1
-0.6	51	-0.00010042	3.098×10^{-2}	208	0.008	<1
-0.5	59	-0.00009258	3.081×10^{-2}	240	0.008	<1
0.0	—	—	—	—	—	—
0.5	—	—	—	—	—	—
1.0	—	—	—	—	—	—

TABLE (6.4.13):Results of PR with varying ϵ and N.

N ϵ	10	50	100	200
0.005	$J^*=0.02256417$ $m=181$ $\ g\ =0.9127626$ Cputime<1 NFE=810	$J^*=0.14578623$ $m=15$ $\ g\ =0.6723512$ Cputime<1 NFE=64	$J^*=11.73154631$ $m=15$ $\ g\ =0.7213542$ Cputime<1 NFE=64	$J^*=11.73154430$ $m=15$ $\ g\ =0.7213542$ Cputime=1 NFE=64
0.006	$J^*=0.02255347$ $m=174$ $\ g\ =0.9127525$ Cputime<1 NFE=700	$J^*=0.09389538$ $m=13$ $\ g\ =0.4693886$ Cputime<1 NFE=56	$J^*=9.64800358$ $m=13$ $\ g\ =0.4370945$ Cputime<1 NFE=56	$J^*=9.64800358$ $m=13$ $\ g\ =0.4370945$ Cputime=1 NFE=56
0.007	$J^*=407.9664307$ $m=139$ $\ g\ =1.62764$ Cputime<1 NFE=560	$J^*=1.68449748$ $m=89$ $\ g\ =15.97277$ Cputime<1 NFE=360	$J^*=-0.00268271$ $m=2000$ $\ g\ =2.231 \times 10^{-2}$ Cputime=14 NFE=8004	$J^*=-0.00268271$ $m=2000$ $\ g\ =2.231 \times 10^{-2}$ Cputime=25 NFE=8004
0.00750	$J^*=-0.00029134$ $m=2000$ $\ g\ =3.935 \times 10^{-2}$ Cputime=1 NFE=8004	$J^*=0.01673043$ $m=26$ $\ g\ =2.863 \times 10^{-2}$ Cputime<1 NFE=108	$J^*=4.62655735$ $m=52$ $\ g\ =0.8601911$ Cputime<1 NFE=212	$J^*=4.62655734$ $m=51$ $\ g\ =0.8601911$ Cputime=1 NFE=208
0.008	$J^*=0.04754090$ $m=2000$ $\ g\ =1.32203$ Cputime=1 NFE=8004	$J^*=-0.00029958$ $m=2000$ $\ g\ =2.596 \times 10^{-2}$ Cputime=8 NFE=8004	$J^*=16.40071869$ $m=34$ $\ g\ =1.88379$ Cputime<1 NFE=140	$J^*=16.40071869$ $m=34$ $\ g\ =1.88379$ Cputime=1 NFE=140
0.009	$J^*=-0.00021008$ $m=268$ $\ g\ =4.975 \times 10^{-2}$ Cputime<1 NFE=1076	$J^*=217.2554779$ $m=22$ $\ g\ =1.03771$ Cputime<1 NFE=92	$J^*=0.25288621$ $m=22$ $\ g\ =0.6614708$ Cputime<1 NFE=92	$J^*=0.25288531$ $m=21$ $\ g\ =0.6614708$ Cputime=1 NFE=87

TABLE (6.4.14):Results of PR with change in u_0 for $ACC \leq 0.1$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.0	196	0.00057927	9.971×10^{-2}	788	0.007	1
-0.5	291	0.00058020	9.931×10^{-2}	1168	0.007	2
0.0	329	-0.00028614	9.996×10^{-2}	1320	0.007	3
0.5	155	0.00058349	9.928×10^{-2}	624	0.007	1
1.0	227	0.00059323	9.916×10^{-2}	912	0.007	2
1.5	190	0.00059648	9.909×10^{-2}	764	0.007	1
2.0	191	0.00059942	9.881×10^{-2}	768	0.007	1

TABLE (6.4.15):Results of PR with change in u_0 for $ACC \leq 0.05$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.0	261	0.00007051	4.974×10^{-2}	1048	0.007	1
-0.5	418	0.00007712	4.992×10^{-2}	1676	0.007	4
0.0	419	-0.00056451	4.970×10^{-2}	1680	0.007	4
0.5	201	0.00008146	4.962×10^{-2}	808	0.007	2
1.0	292	0.00009179	4.998×10^{-2}	1172	0.007	2
1.5	243	0.00009563	4.983×10^{-2}	976	0.007	2
2.0	243	0.00010044	4.974×10^{-2}	976	0.007	2

TABLE (6.4.16):Results of PR with change in u_0 for $ACC \leq 0.023$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.0	–	–	–	–	0.007	–
-0.5	–	–	–	–	0.007	–
0.0	1649	-0.00242977	2.230×10^{-2}	6600	0.007	12
0.5	1749	-0.00242999	2.230×10^{-2}	7000	0.007	12
1.0	–	–	–	–	0.007	–
1.5	–	–	–	–	0.007	–
2.0	–	–	–	–	0.007	–

TABLE (6.4.17):Results of H1 with varying ϵ and N.

N ϵ	10	50	100	200
0.003	J*=-0.00291241 m=950 $\ g\ =4.415 \times 10^{-2}$ Cputime<1 NFE=3975	J*=0.07832151 m=401 $\ g\ =2.05642$ Cputime=1 NFE=1482	J*=-0.00107566 m=1501 $\ g\ =3.121 \times 10^{-2}$ Cputime=10 NFE=6183	J*=-0.00110431 m=1499 $\ g\ =3.12 \times 10^{-2}$ Cputime=19 NFE=6201
0.005	J*=-0.00292066 m=930 $\ g\ =4.398 \times 10^{-2}$ Cputime<1 NFE=3724	J*=0.07488351 m=347 $\ g\ =1.98921$ Cputime=1 NFE=1392	J*=-0.00108499 m=1492 $\ g\ =2.997 \times 10^{-2}$ Cputime=9 NFE=5972	J*=-0.00110566 m=1489 $\ g\ =2.912 \times 10^{-2}$ Cputime=16 NFE=6023
0.006	J*=3.21129179 m=153 $\ g\ =6.46442$ Cputime<1 NFE=616	J*=0.09107057 m=650 $\ g\ =0.8701492$ Cputime=2 NFE=2604	J*=-0.00327211 m=1147 $\ g\ =2.0 \times 10^{-2}$ Cputime=8 NFE=4592	J*=-0.00327211 m=1147 $\ g\ =2.0 \times 10^{-2}$ Cputime=15 NFE=4612
0.007	J*=11.23644543 m=32 $\ g\ =33.01036$ Cputime<1 NFE=132	J*=-0.00297981 m=1312 $\ g\ =2.150 \times 10^{-2}$ Cputime=5 NFE=5252	J*=-0.00236861 m=1200 $\ g\ =2.152 \times 10^{-2}$ Cputime=8 NFE=4004	J*=-0.00236971 m=1199 $\ g\ =2.151 \times 10^{-2}$ Cputime=15 NFE=4011
0.008	J*=0.07349106 m=266 $\ g\ =1.59355$ Cputime<1 NFE=1068	J*=0.00126187 m=1608 $\ g\ =4.762 \times 10^{-2}$ Cputime=6 NFE=6436	J*=-0.00009449 m=460 $\ g\ =4.374 \times 10^{-2}$ Cputime=3 NFE=1844	J*=-0.00010102 m=460 $\ g\ =4.372 \times 10^{-2}$ Cputime=5 NFE=1851

TABLE (6.4.18):Results of H1 with change in u_0 for $ACC \leq 0.1$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.0	42	0.00056063	9.943×10^{-2}	172	0.006	<1
-0.5	43	0.00054596	9.716×10^{-2}	176	0.006	<1
0.0	830	0.00154769	9.274×10^{-2}	3324	0.006	6
0.5	—	—	—	—	—	—
1.0	327	-0.00005675	9.644×10^{-2}	1312	0.006	3
1.5	255	0.00005321	9.598×10^{-2}	1024	0.006	2
2.0	1027	0.00016448	9.772×10^{-2}	4112	0.006	7

TABLE (6.4.19):Results of H1 with change in u_0 for $ACC \leq 0.05$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.0	52	0.00010941	4.869×10^{-2}	212	0.006	<1
-0.5	52	0.00011313	4.852×10^{-2}	212	0.006	<1
0.0	843	0.00112306	4.877×10^{-2}	3376	0.006	7
0.5	—	—	—	—	—	—
1.0	349	-0.00026370	4.918×10^{-2}	1400	0.006	3
1.5	309	-0.00015775	4.970×10^{-2}	1240	0.006	3
2.0	1034	-0.00021023	4.796×10^{-2}	4140	0.006	7

TABLE (6.4.20):Results of H1 with change in u_0 for $ACC \leq 2.0 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.0	—	—	—	—	0.006	—
-0.5	1832	-0.00325432	2.0×10^{-2}	7332	0.006	14
0.0	1322	-0.00329290	1.999×10^{-2}	5292	0.006	9
0.5	—	—	—	—	—	—
1.0	1147	-0.00327211	2.0×10^{-2}	4592	0.006	9
1.5	1289	-0.00325565	1.999×10^{-2}	5160	0.006	10
2.0	1469	-0.00328785	1.998×10^{-2}	5880	0.006	11

TABLE (6.4.21):Results of ATH with varying ϵ and N.

N ϵ	10	50	100	200
0.003	J*=0.13011223 m=150 $\ g\ =4.24021$ Cputime<1 NFE=604	J*=-0.00133513 m=1500 $\ g\ =2.914 \times 10^{-2}$ Cputime=5 NFE=6004	J*=-0.0035611 m=1200 $\ g\ =2.046 \times 10^{-2}$ Cputime=9 NFE=4804	J*=-0.00315422 m=1200 $\ g\ =2.046 \times 10^{-2}$ Cputime=15 NFE=4815
0.005	J*=0.12992722 m=150 $\ g\ =4.23756$ Cputime<1 NFE=604	J*=-0.00143401 m=1500 $\ g\ =2.903 \times 10^{-2}$ Cputime=5 NFE=6004	J*=-0.0031682 m=1200 $\ g\ =2.045 \times 10^{-2}$ Cputime=9 NFE=4804	J*=-0.0031691 m=1200 $\ g\ =2.044 \times 10^{-2}$ Cputime=16 NFE=4815
0.006	J*=28.81435013 m=150 $\ g\ =112.618$ Cputime<1 NFE=604	J*=28.81449509 m=1500 $\ g\ =112.7295$ Cputime=5 NFE=6004	J*=-0.00324996 m=1135 $\ g\ =2.0 \times 10^{-2}$ Cputime=8 NFE=4544	J*=-0.00324996 m=1134 $\ g\ =2.0 \times 10^{-2}$ Cputime=15 NFE=4549
0.007	J*=28.81435013 m=150 $\ g\ =112.618$ Cputime<1 NFE=604	J*=28.81449509 m=1500 $\ g\ =112.7295$ Cputime=5 NFE=6004	J*=28.81449509 m=1200 $\ g\ =112.7434$ Cputime=9 NFE=4804	J*=28.81450122 m=1199 $\ g\ =112.7434$ Cputime=15 NFE=4813
0.008	J*=28.81435013 m=150 $\ g\ =112.618$ Cputime<1 NFE=604	J*=817.1523438 m=1500 $\ g\ =598.4329$ Cputime=6 NFE=6004	J*=185.5828247 m=1200 $\ g\ =286.5673$ Cputime=9 NFE=4804	J*=185.5828247 m=1200 $\ g\ =286.5673$ Cputime=15 NFE=4815

TABLE (6.4.22):Results of ATH with change in u_0 for $ACC \leq 0.1$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.0	42	0.00056063	9.943×10^{-2}	172	0.006	<1
-0.5	49	0.00051258	9.434×10^{-2}	200	0.006	<1
0.0	212	0.00001982	9.361×10^{-2}	852	0.006	2
0.5	166	-0.00016274	9.0401×10^{-2}	668	0.006	1
1.0	545	0.00035368	9.050×10^{-2}	2184	0.006	5
1.5	938	0.00190120	8.419×10^{-2}	3756	0.006	7
2.0	—	—	—	—	—	—

TABLE (6.4.23):Results of ATH with change in u_0 for $ACC \leq 0.05$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.0	52	0.00010941	4.869×10^{-2}	212	0.006	<1
-0.5	59	0.00010454	4.723×10^{-2}	240	0.006	1
0.0	225	-0.00008712	4.947×10^{-2}	904	0.006	2
0.5	175	-0.00027688	4.771×10^{-2}	704	0.006	1
1.0	550	0.00001793	3.563×10^{-2}	21204	0.006	5
1.5	950	0.00162452	4.957×10^{-2}	3804	0.006	7
2.0	—	—	—	—	—	—

TABLE (6.4.24):Results of ATH with change in u_0 for $ACC \leq 2.0 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.0	—	—	—	—	0.006	—
-0.5	1893	-0.00321805	2.0×10^{-2}	7576	0.006	14
0.0	1348	-0.00326430	2.0×10^{-2}	5396	0.006	11
0.5	1135	-0.00324996	2.0×10^{-2}	4544	0.006	9
1.0	—	—	—	—	0.006	—
1.5	1377	-0.00328307	1.997×10^{-2}	5512	0.006	10
2.0	—	—	—	—	—	—

TABLE (6.4.25):Results of H3 with varying ϵ and N.

N ϵ	10	50	100	200
0.003	J*=-0.00301711 m=2000 $\ g\ =4.441 \times 10^{-2}$ Cputime=1 NFE=8004	J*=0.07461013 m=420 $\ g\ =1.51631$ Cputime=1 NFE=1710	J*=-0.00315421 m=1761 $\ g\ =2.215 \times 10^{-2}$ Cputime=12 NFE=7223	J*=-0.00315721 m=1751 $\ g\ =2.213 \times 10^{-2}$ Cputime=22 NFE=7228
0.005	J*=-0.00301902 m=2000 $\ g\ =4.438 \times 10^{-2}$ Cputime=1 NFE=8004	J*=0.07455767 m=410 $\ g\ =1.51627$ Cputime=1 NFE=1644	J*=-0.0031682 m=1742 $\ g\ =2.195 \times 10^{-2}$ Cputime=12 NFE=6972	J*=-0.0031684 m=1742 $\ g\ =2.195 \times 10^{-2}$ Cputime=22 NFE=6988
0.006	J*=817.1522217 m=1200 $\ g\ =597.6071$ Cputime<1 NFE=4804	J*=0.09107057 m=650 $\ g\ =0.871492$ Cputime=2 NFE=2604	J*=-0.00327211 m=1147 $\ g\ =2.0 \times 10^{-2}$ Cputime=8 NFE=4592	J*=-0.00327211 m=1146 $\ g\ =2.0 \times 10^{-2}$ Cputime=15 NFE=4601
0.007	J*=817.1522217 m=1200 $\ g\ =597.6071$ Cputime<1 NFE=4804	J*=-0.00315818 m=1300 $\ g\ =2.178 \times 10^{-2}$ Cputime=4 NFE=5204	J*=-0.00317778 m=1260 $\ g\ =2.087 \times 10^{-2}$ Cputime=9 NFE=5044	J*=-0.00317781 m=1259 $\ g\ =2.087 \times 10^{-2}$ Cputime=15 NFE=5052
0.008	J*=0.01000185 m=1475 $\ g\ =0.8199844$ Cputime=1 NFE=5904	J*=0.06731721 m=1300 $\ g\ =1.46247$ Cputime=4 NFE=5204	J*=817.1523475 m=1300 $\ g\ =598.5401$ Cputime=10 NFE=5204	J*=817.1523475 m=1300 $\ g\ =598.5401$ Cputime=18 NFE=5215

TABLE (6.4.26):Results of H3 with change in u_0 for $ACC \leq 0.1$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.0	42	0.00056063	9.943×10^{-2}	172	0.006	<1
-0.5	43	0.00054596	9.716×10^{-2}	176	0.006	<1
0.0	830	0.00154769	9.274×10^{-2}	3324	0.006	6
0.5	-	-	-	-	-	-
1.0	327	-0.00005675	9.644×10^{-2}	1312	0.006	3
1.5	255	0.00005321	9.598×10^{-2}	1024	0.006	2
2.0	1027	0.00016448	9.772×10^{-2}	4112	0.006	7

TABLE (6.4.27):Results of H3 with change in u_0 for $ACC \leq 0.05$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.0	52	0.00010941	4.869×10^{-2}	212	0.006	<1
-0.5	52	0.00011313	4.852×10^{-2}	212	0.006	<1
0.0	843	0.00112306	4.877×10^{-2}	3376	0.006	7
0.5	-	-	-	-	-	-
1.0	349	-0.00026370	4.918×10^{-2}	1400	0.006	3
1.5	309	-0.00015775	4.970×10^{-2}	1240	0.006	3
2.0	1034	-0.00021023	4.796×10^{-2}	4140	0.006	7

TABLE (6.4.28):Results of H3 with change in u_0 for $ACC \leq 2.0 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
-1.0	--	--	--	--	0.006	--
-0.5	1832	-0.00325432	2.0×10^{-2}	7332	0.006	14
0.0	1322	-0.00329290	1.999×10^{-2}	5292	0.006	9
0.5	--	--	--	--	--	--
1.0	1147	-0.00327211	2.0×10^{-2}	4592	0.006	9
1.5	1289	-0.00325565	1.999×10^{-2}	5160	0.006	10
2.0	1469	-0.00328785	1.998×10^{-2}	5880	0.006	11

Table (6.5.1): Summary table for the seven methods

method u_0	GFS	SD	FR	PR	H1	ATH	H3
-1.0	$m = 176$ $J^* = 0.00007013$ $\ g\ = 4.983 \times 10^{-2}$ $\varepsilon = 0.008$ NFE = 177 Cputime < 1	$m = 240$ $J^* = 0.00006898$ $\ g\ = 8.959 \times 10^{-2}$ $\varepsilon = 0.008$ NFE = 964 Cputime = 1	$m = 47$ $J^* = -0.00010353$ $\ g\ = 3.088 \times 10^{-2}$ $\varepsilon = 0.008$ NFE = 192 Cputime < 1	$m = 261$ $J^* = 0.00007051$ $\ g\ = 4.974 \times 10^{-2}$ $\varepsilon = 0.007$ NFE = 1048 Cputime = 1	$m = 52$ $J^* = 0.00010941$ $\ g\ = 4.869 \times 10^{-2}$ $\varepsilon = 0.006$ NFE = 212 Cputime < 1	$m = 52$ $J^* = 0.00010941$ $\ g\ = 4.869 \times 10^{-2}$ $\varepsilon = 0.006$ NFE = 212 Cputime < 1	$m = 52$ $J^* = 0.00010941$ $\ g\ = 4.869 \times 10^{-2}$ $\varepsilon = 0.006$ NFE = 212 Cputime < 1
0.0	$m = 72$ $J^* = -0.00010746$ $\ g\ = 4.925 \times 10^{-2}$ $\varepsilon = 0.008$ NFE = 73 Cputime < 1	$m = 1609$ $J^* = -0.00242712$ $\ g\ = 2.30 \times 10^{-2}$ $\varepsilon = 0.008$ NFE = 6440 Cputime = 11	—	$m = 2000$ $J^* = -0.00268271$ $\ g\ = 2.230 \times 10^{-2}$ $\varepsilon = 0.007$ NFE = 8004 Cputime = 14	$m = 1322$ $J^* = -0.00329290$ $\ g\ = 1.999 \times 10^{-2}$ $\varepsilon = 0.006$ NFE = 5292 Cputime = 9	$m = 1348$ $J^* = -0.00326430$ $\ g\ = 2.0 \times 10^{-2}$ $\varepsilon = 0.006$ NFE = 5396 Cputime = 11	$m = 1322$ $J^* = -0.00329290$ $\ g\ = 1.999 \times 10^{-2}$ $\varepsilon = 0.006$ NFE = 5292 Cputime = 9
0.5	$m = 78$ $J^* = -0.0006646$ $\ g\ = 4.902 \times 10^{-2}$ $\varepsilon = 0.008$ NFE = 75 Cputime < 1	$m = 185$ $J^* = 0.00007970$ $\ g\ = 4.952 \times 10^{-2}$ $\varepsilon = 0.008$ NFE = 744 Cputime = 1	—	$m = 1749$ $J^* = -0.00242999$ $\ g\ = 2.330 \times 10^{-2}$ $\varepsilon = 0.007$ NFE = 7000 Cputime = 12	—	$m = 1135$ $J^* = -0.00324996$ $\ g\ = 2.0 \times 10^{-2}$ $\varepsilon = 0.006$ NFE = 4544 Cputime = 9	—
1.0	$m = 85$ $J^* = -0.00016333$ $\ g\ = 4.931 \times 10^{-2}$ $\varepsilon = 0.008$ NFE = 86 Cputime < 1	$m = 1791$ $J^* = -0.00243135$ $\ g\ = 2.30 \times 10^{-2}$ $\varepsilon = 0.008$ NFE = 7168 Cputime = 13	—	$m = 292$ $J^* = 0.0009179$ $\ g\ = 4.998 \times 10^{-2}$ $\varepsilon = 0.007$ NFE = 1172 Cputime = 2	$m = 1147$ $J^* = -0.00327211$ $\ g\ = 2.0 \times 10^{-2}$ $\varepsilon = 0.006$ NFE = 4592 Cputime = 9	$m = 550$ $J^* = 0.00001793$ $\ g\ = 3.563 \times 10^{-2}$ $\varepsilon = 0.006$ NFE = 2204 Cputime = 5	$m = 1147$ $J^* = -0.00327211$ $\ g\ = 2.0 \times 10^{-2}$ $\varepsilon = 0.006$ NFE = 4592 Cputime = 10
3.0	$m = 1000$ $J^* = -0.00294302$ $\ g\ = 1.891 \times 10^{-2}$ $\varepsilon = 0.008$ NFE = 1001 Cputime = 3	$m = 2000$ $J^* = -0.00273340$ $\ g\ = 2.223 \times 10^{-2}$ $\varepsilon = 0.008$ NFE = 8004 Cputime = 14	—	—	$m = 1200$ $J^* = -0.00322215$ $\ g\ = 2.020 \times 10^{-2}$ $\varepsilon = 0.006$ NFE = 4804 Cputime = 8	$m = 1200$ $J^* = -0.00199214$ $\ g\ = 2.714 \times 10^{-2}$ $\varepsilon = 0.006$ NFE = 4804 Cputime = 8	$m = 1200$ $J^* = -0.00372215$ $\ g\ = 2.020 \times 10^{-2}$ $\varepsilon = 0.006$ NFE = 4804 Cputime = 8

GFS

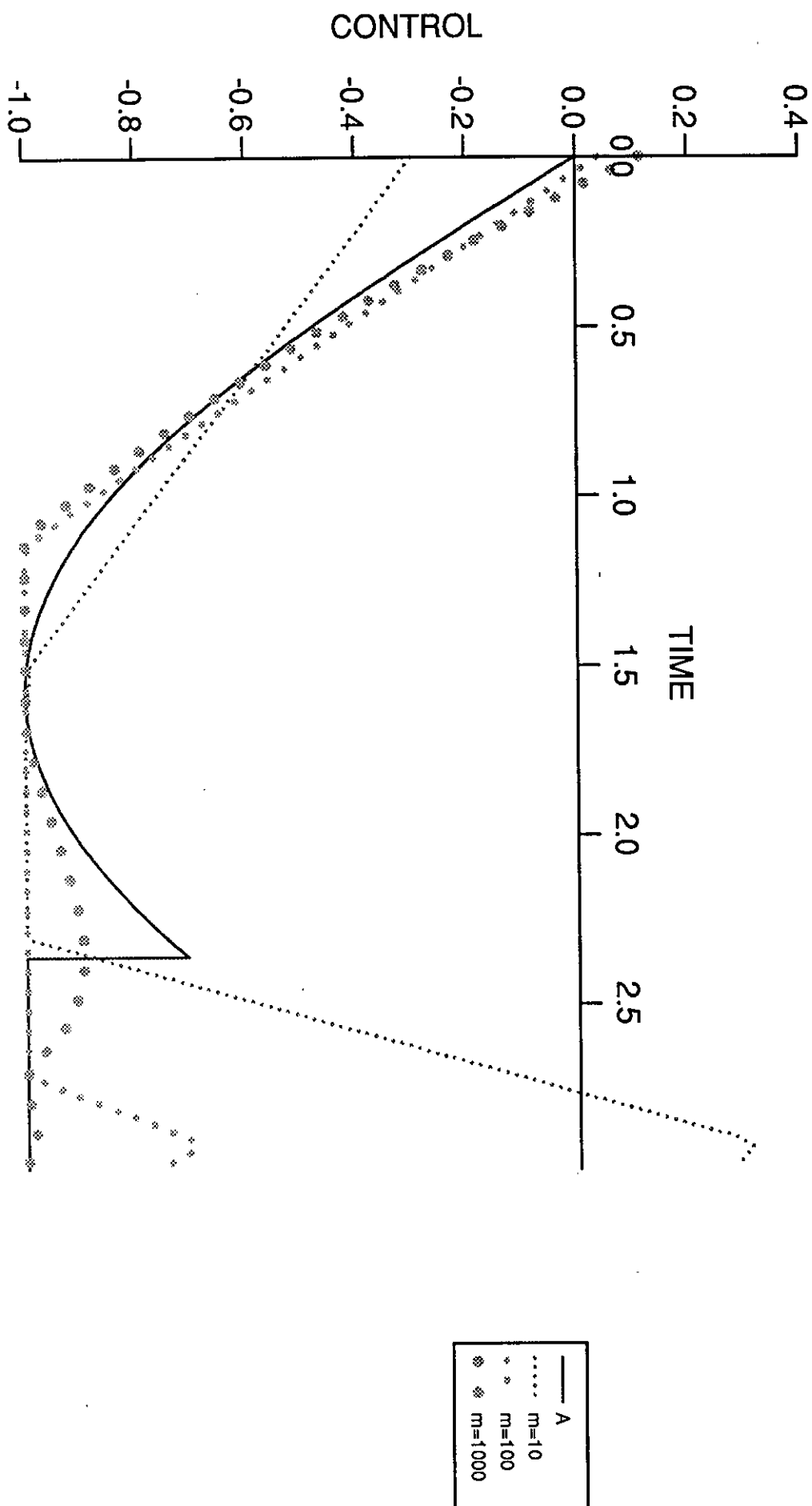


Figure (6.4.1)

GFS

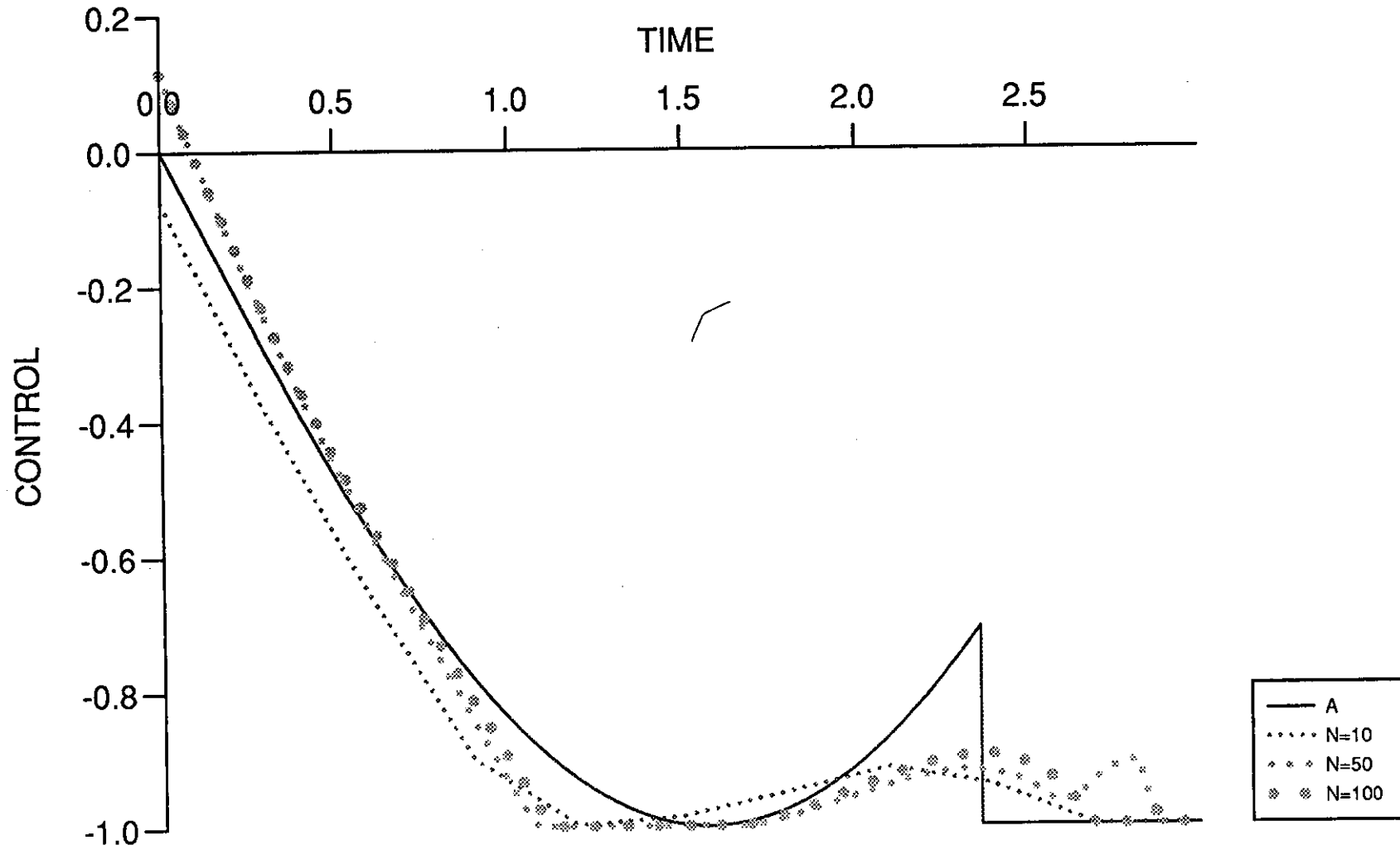


Figure (6.4.2)

GFS

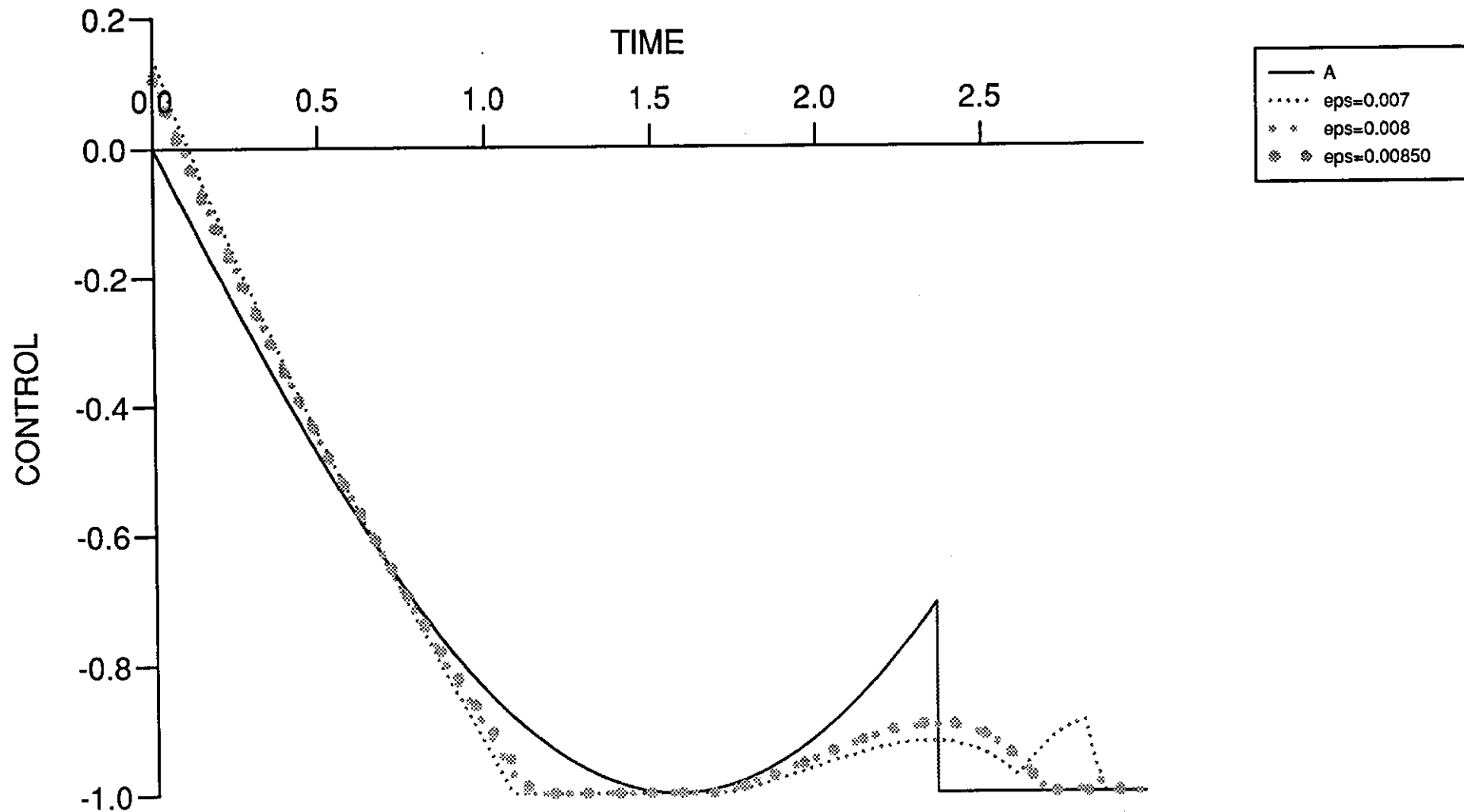


Figure (6.4.3)

GFS

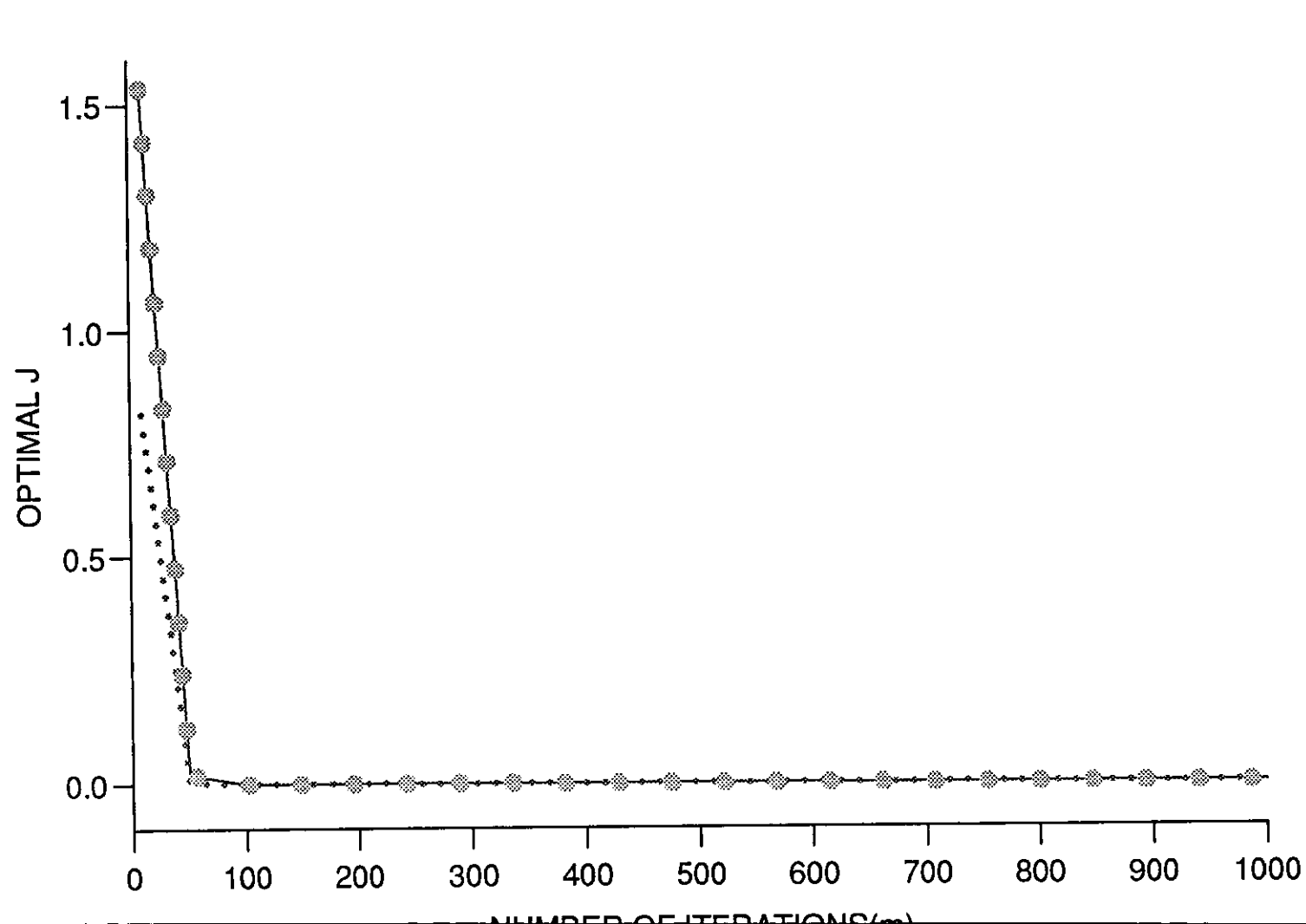


Figure (6.4.4)

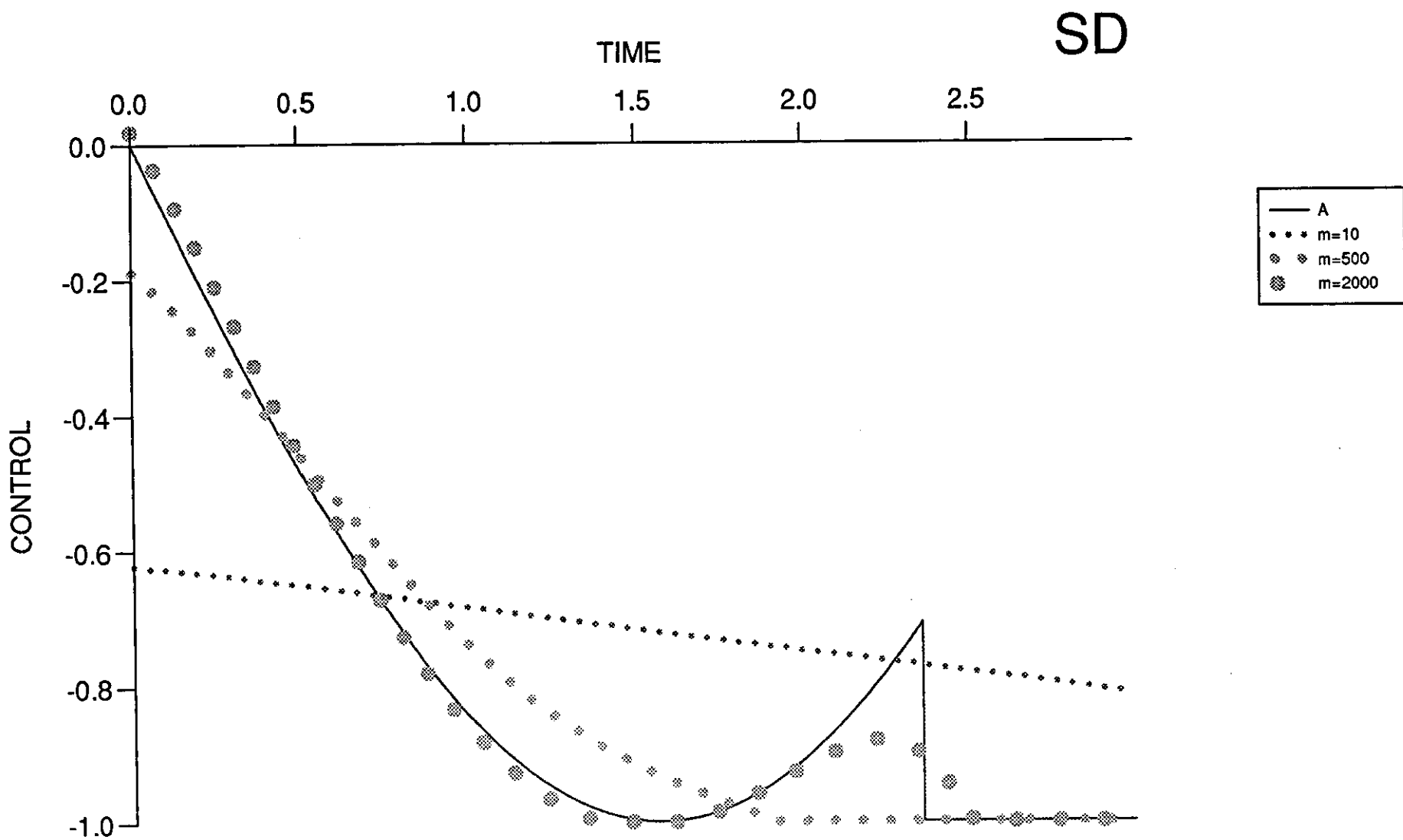


Figure (6.4.5)

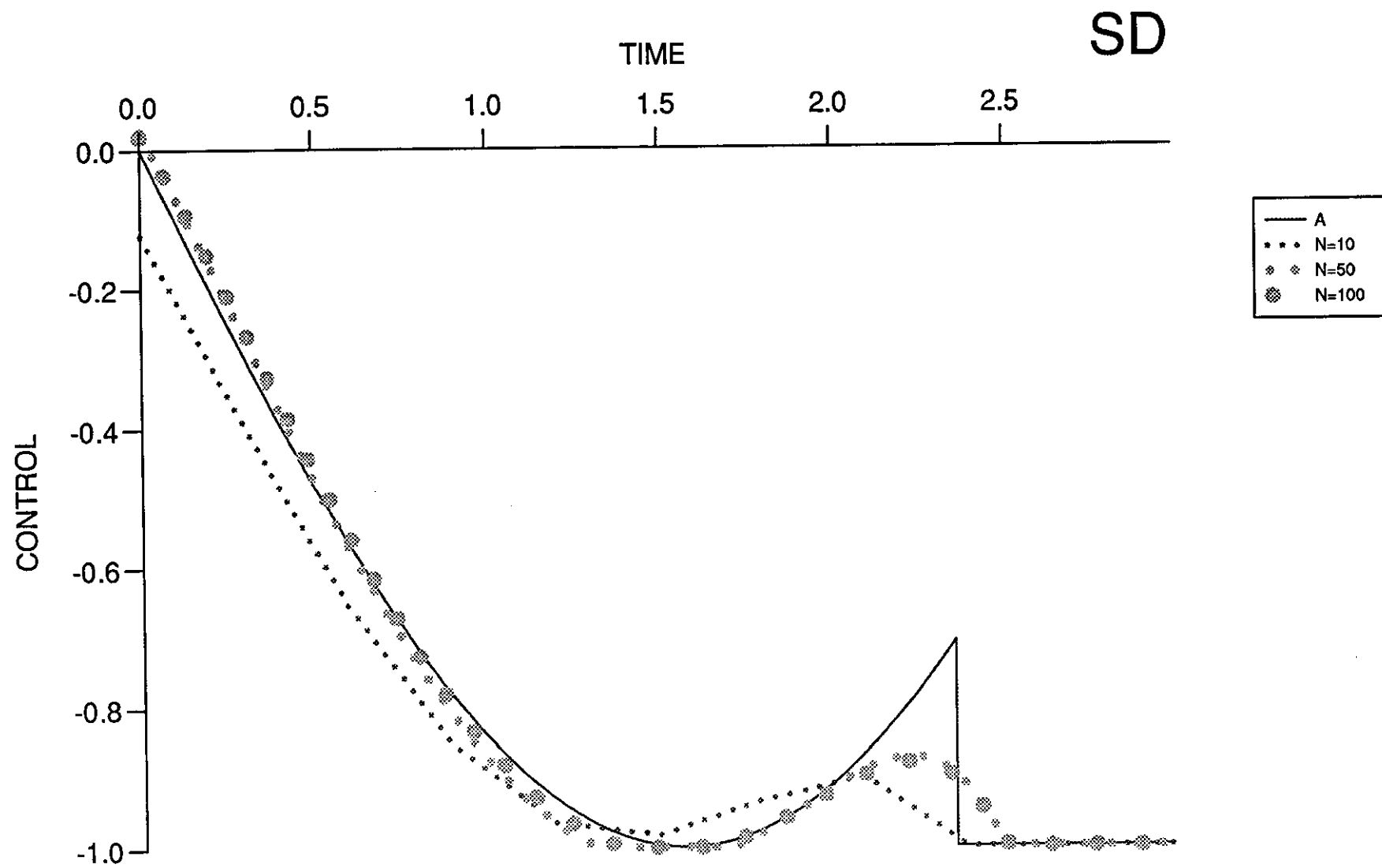
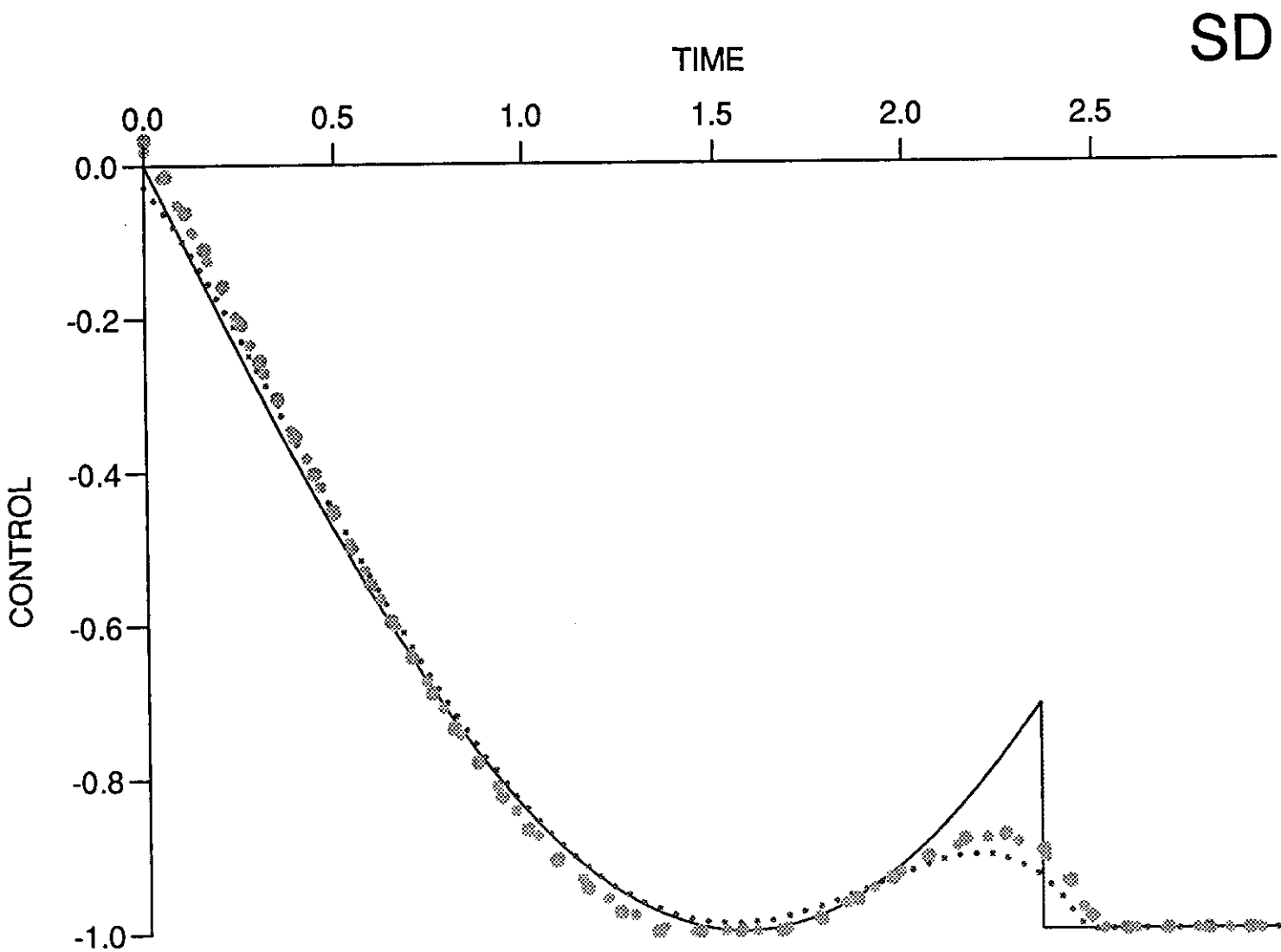


Figure (6.4.6)



— A
• • • eps=0.007
♦ ♦ ♦ eps=0.008
⊗ ⊗ ⊗ eps=0.00850

Figure (6.4.7)

SD

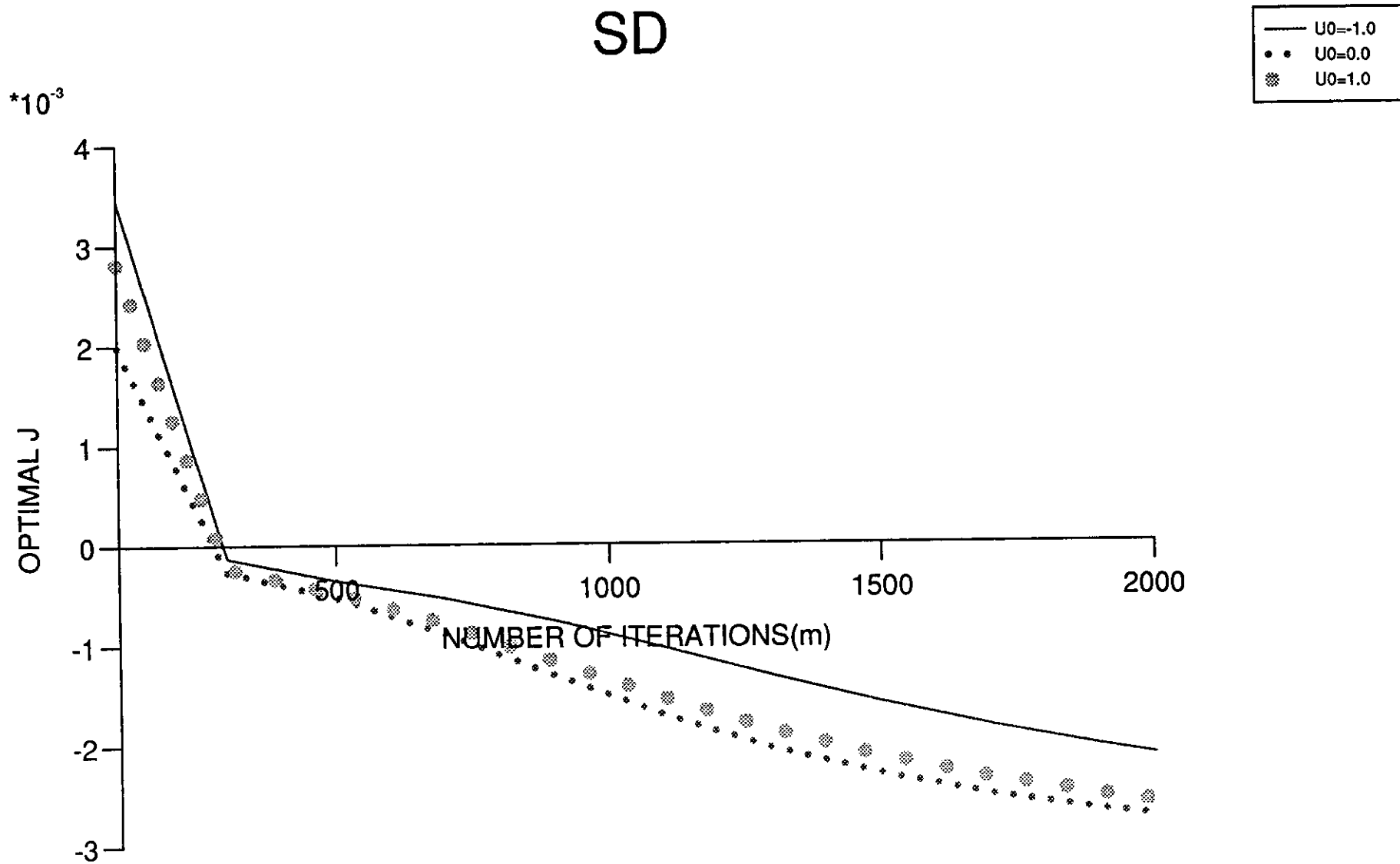


Figure (6.4.8)

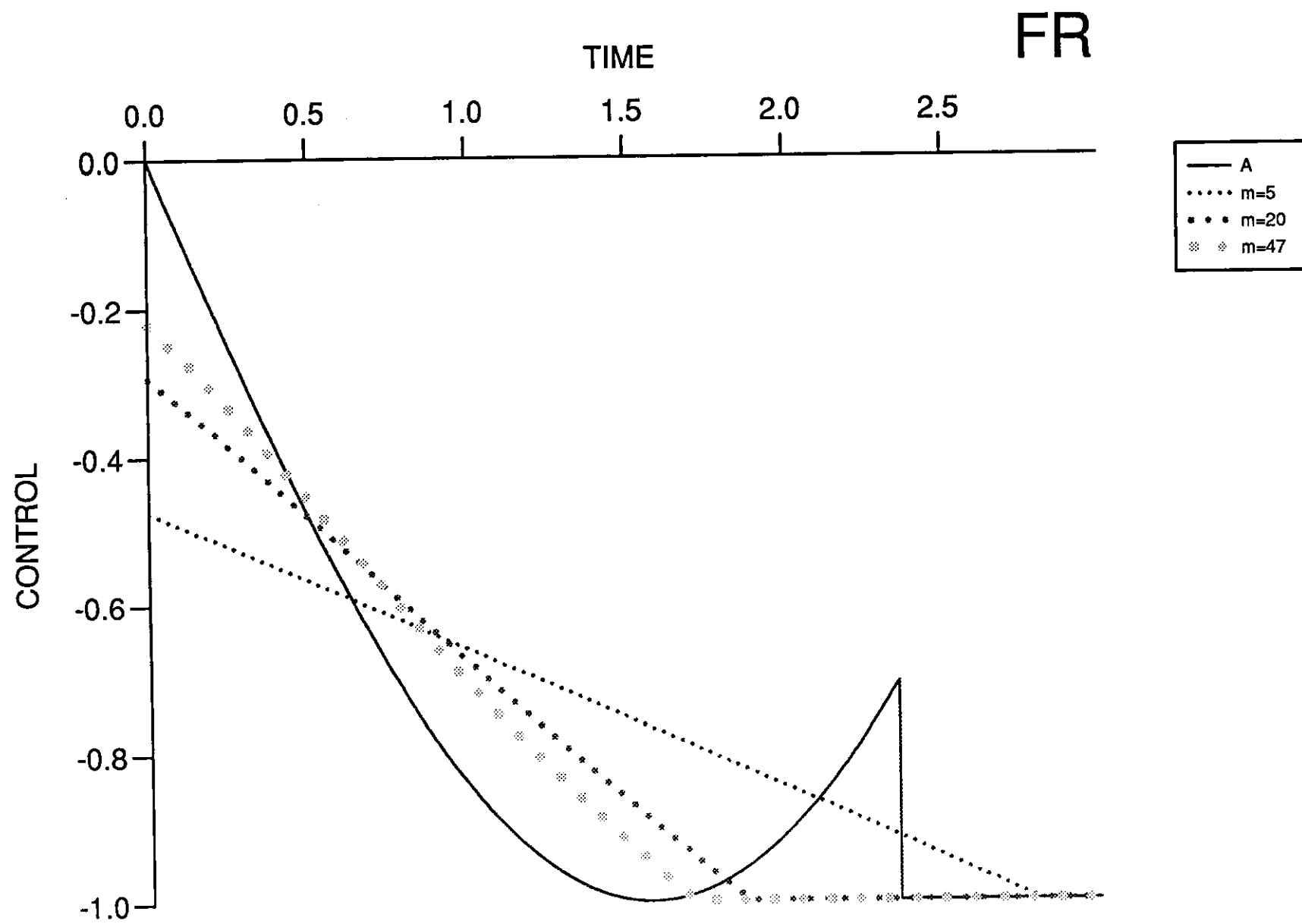


Figure (6.4.9)

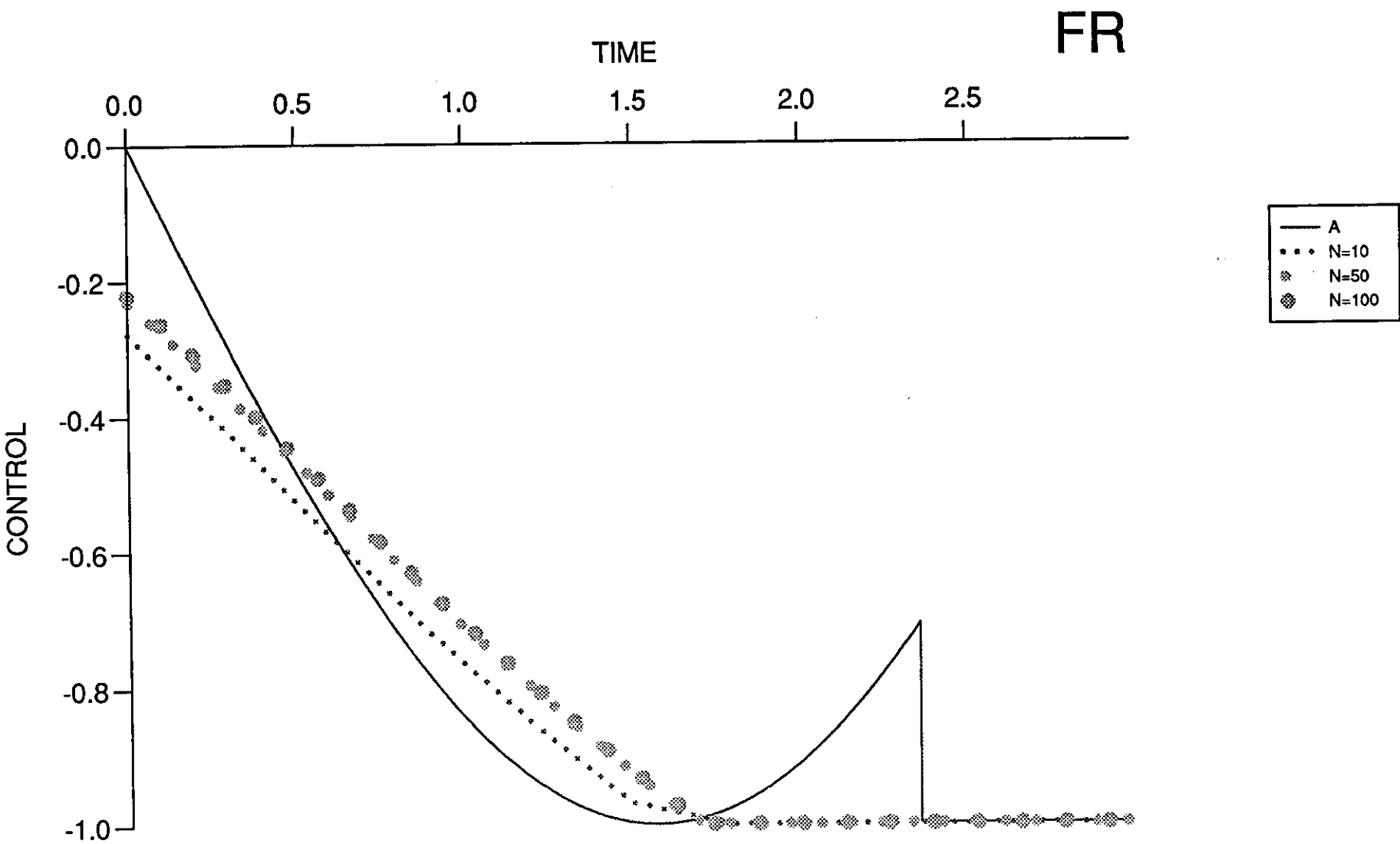


Figure (6.4.10)

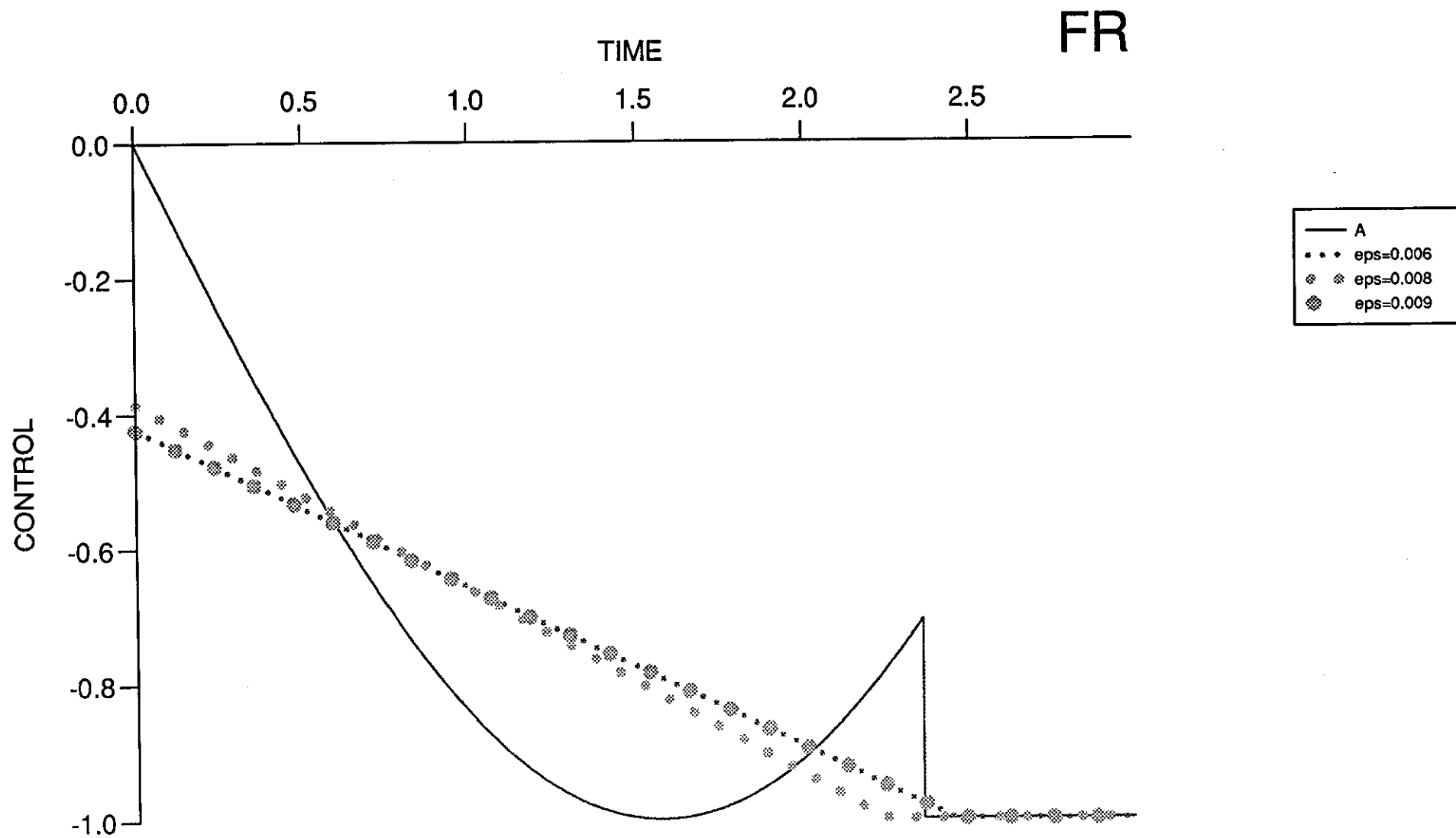


Figure (6.4.11)

FR

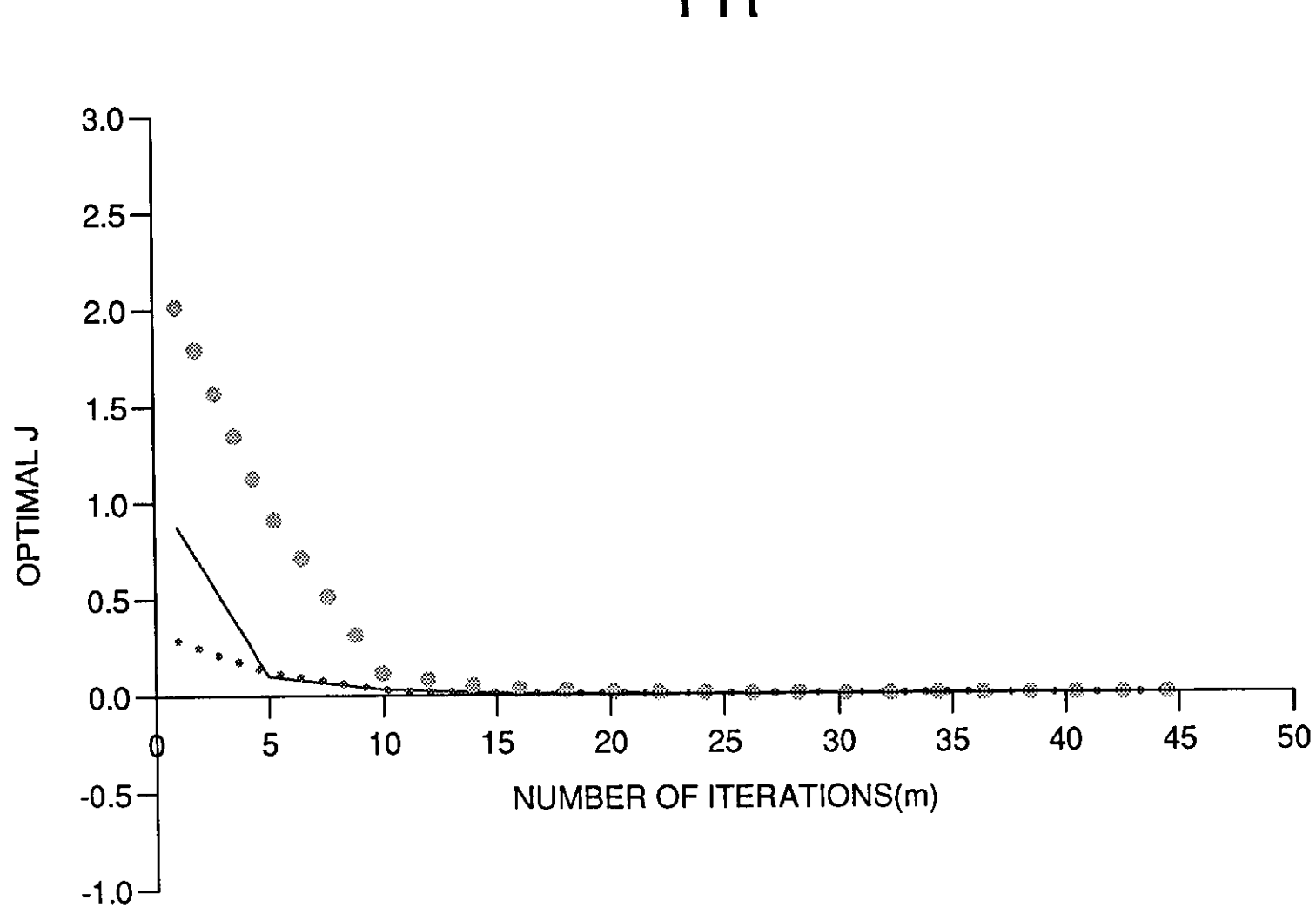


Figure (6.4.12)

PR

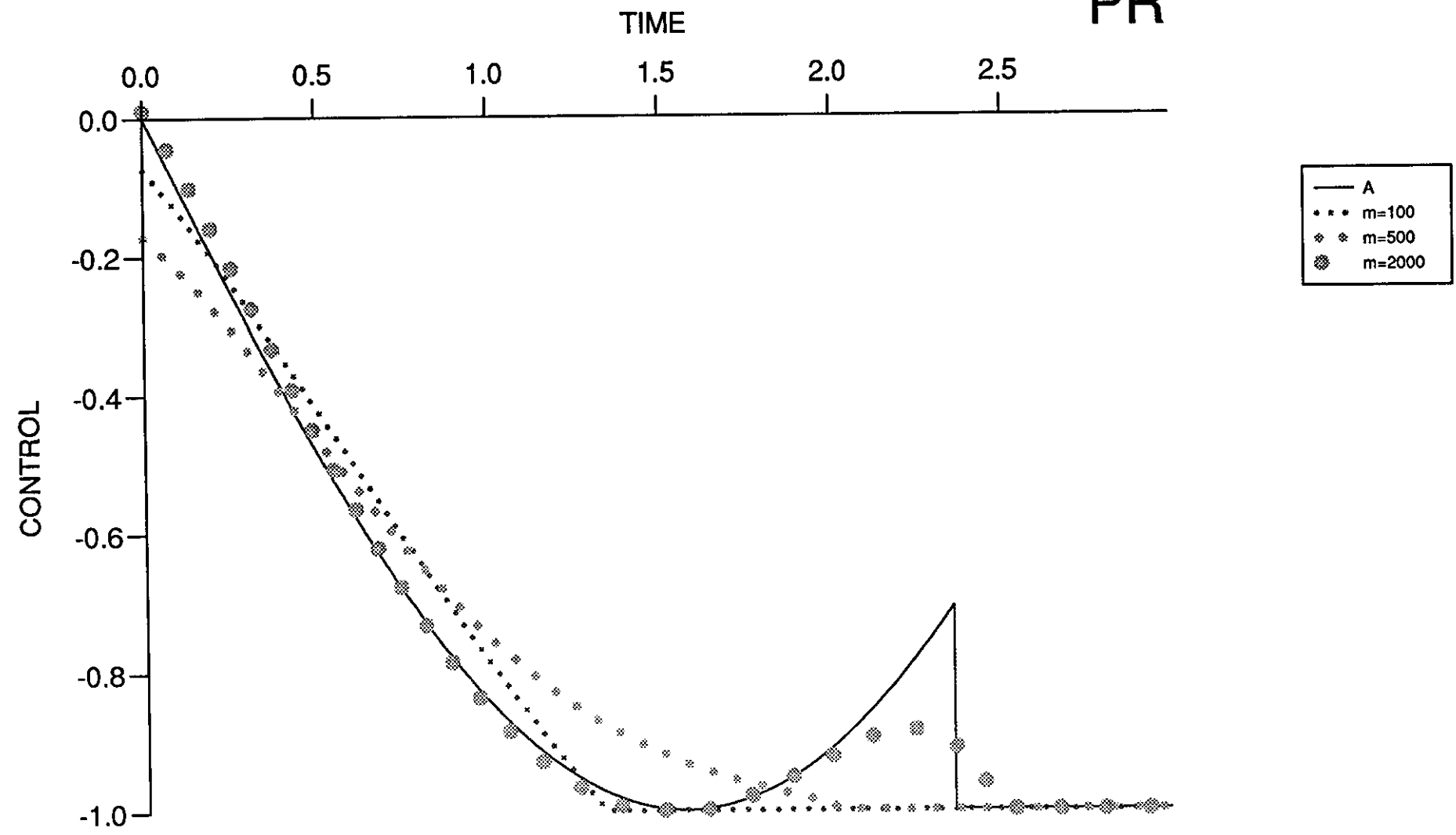


Figure (6.4.13)

PR

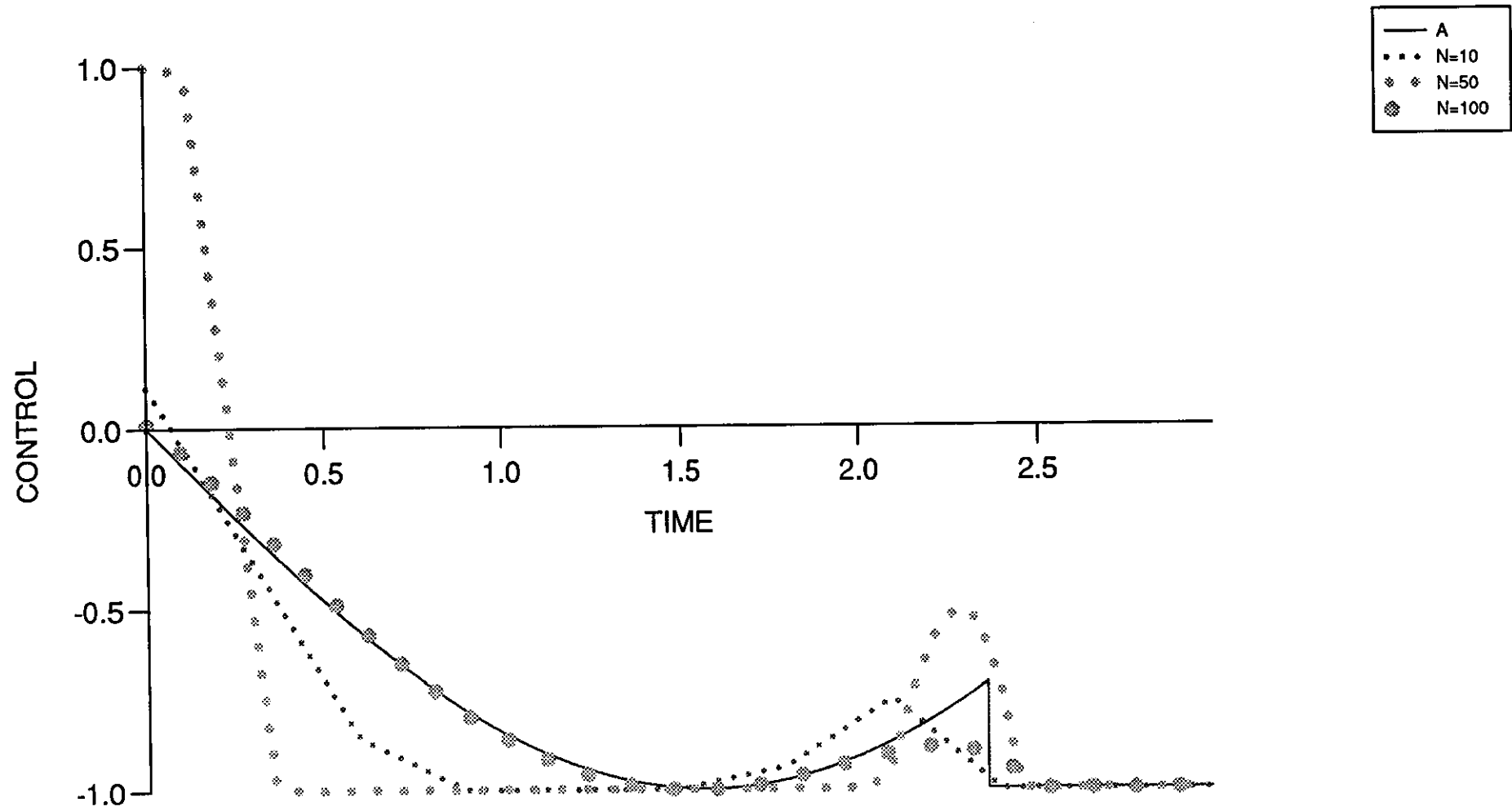


Figure (6.4.14)

PR

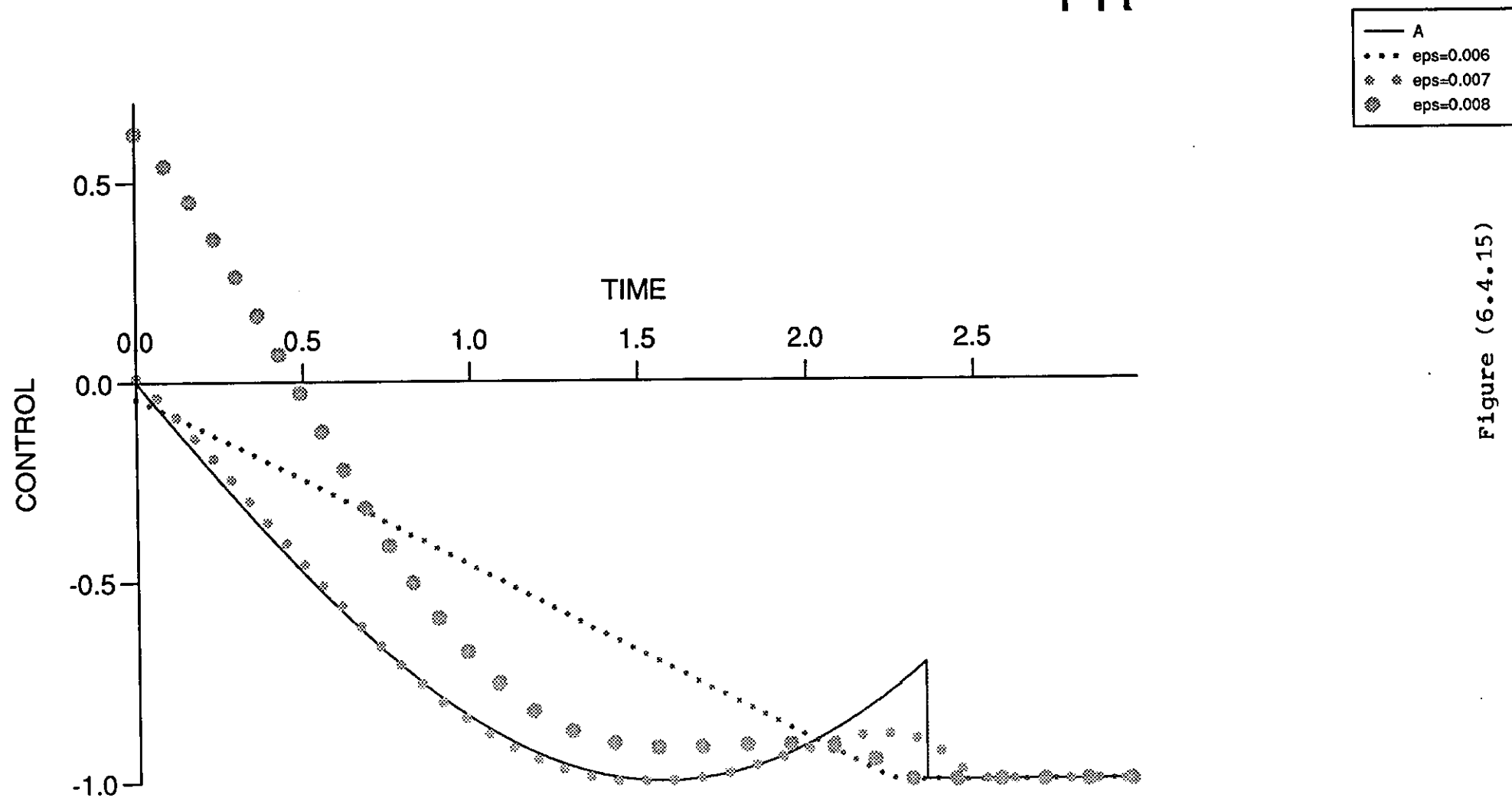


Figure (6.4.15)

PR

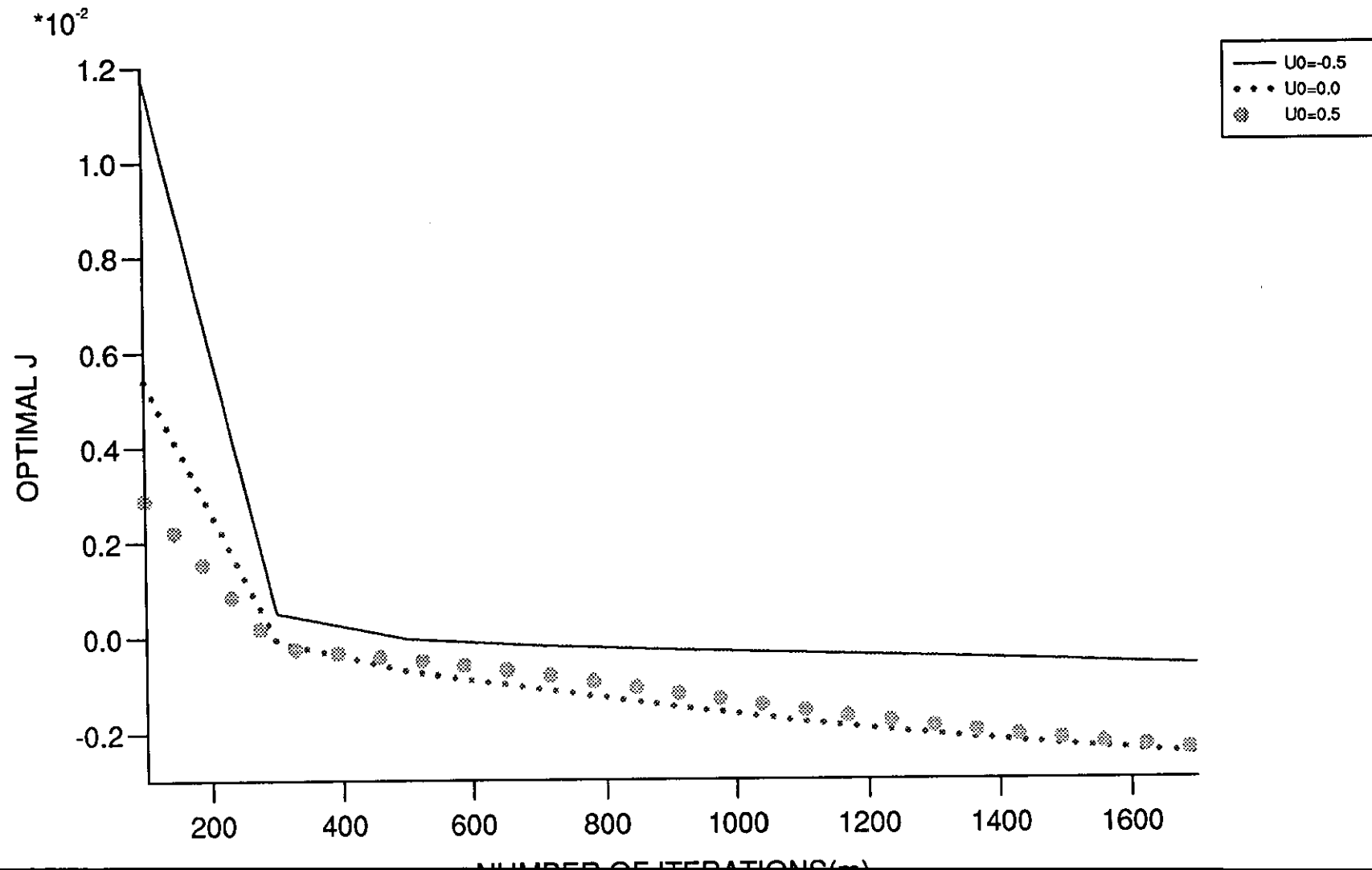


Figure (6.4.16)

H1

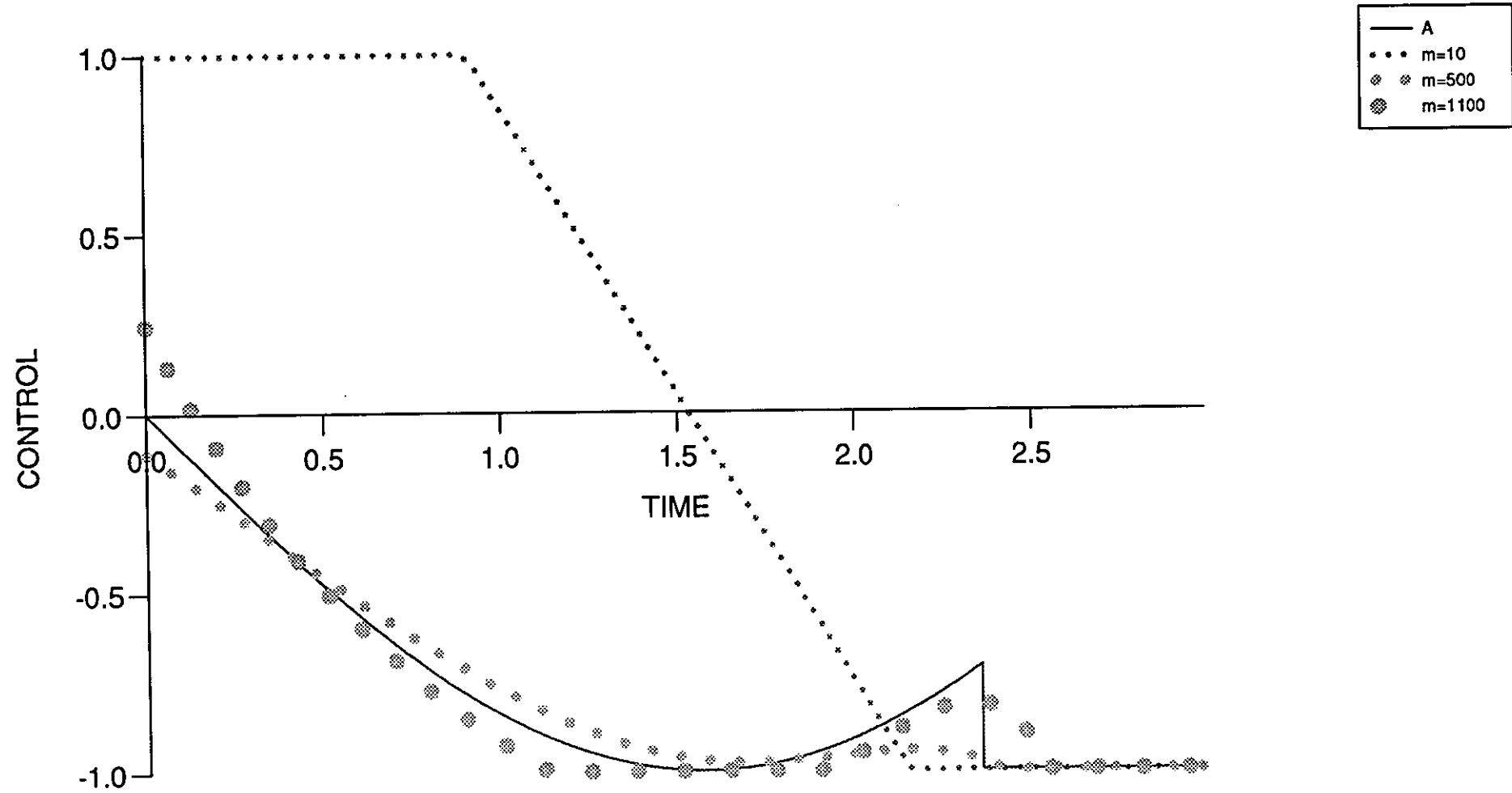


Figure (6.4.17)

H1

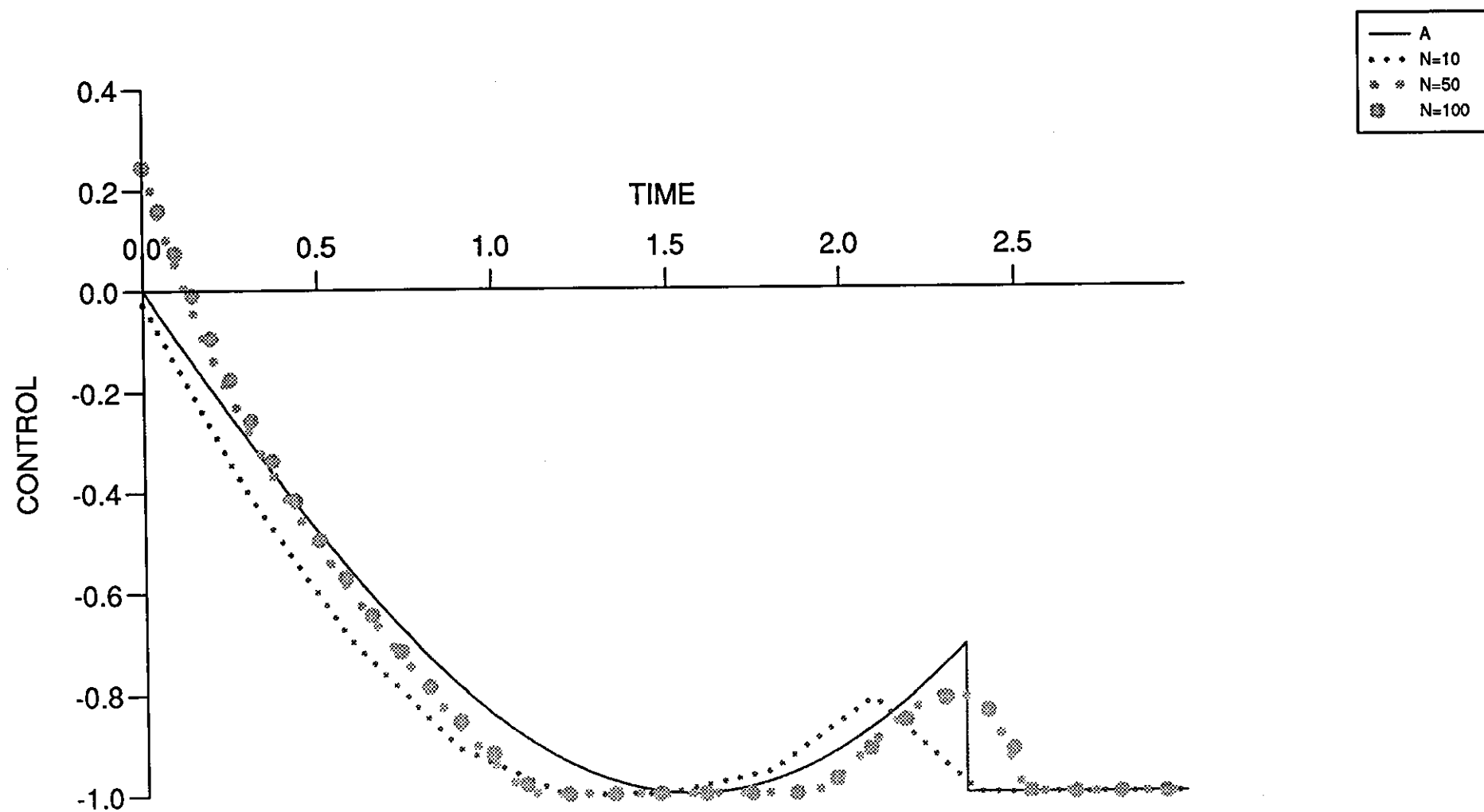


Figure (6.4.18)

H1

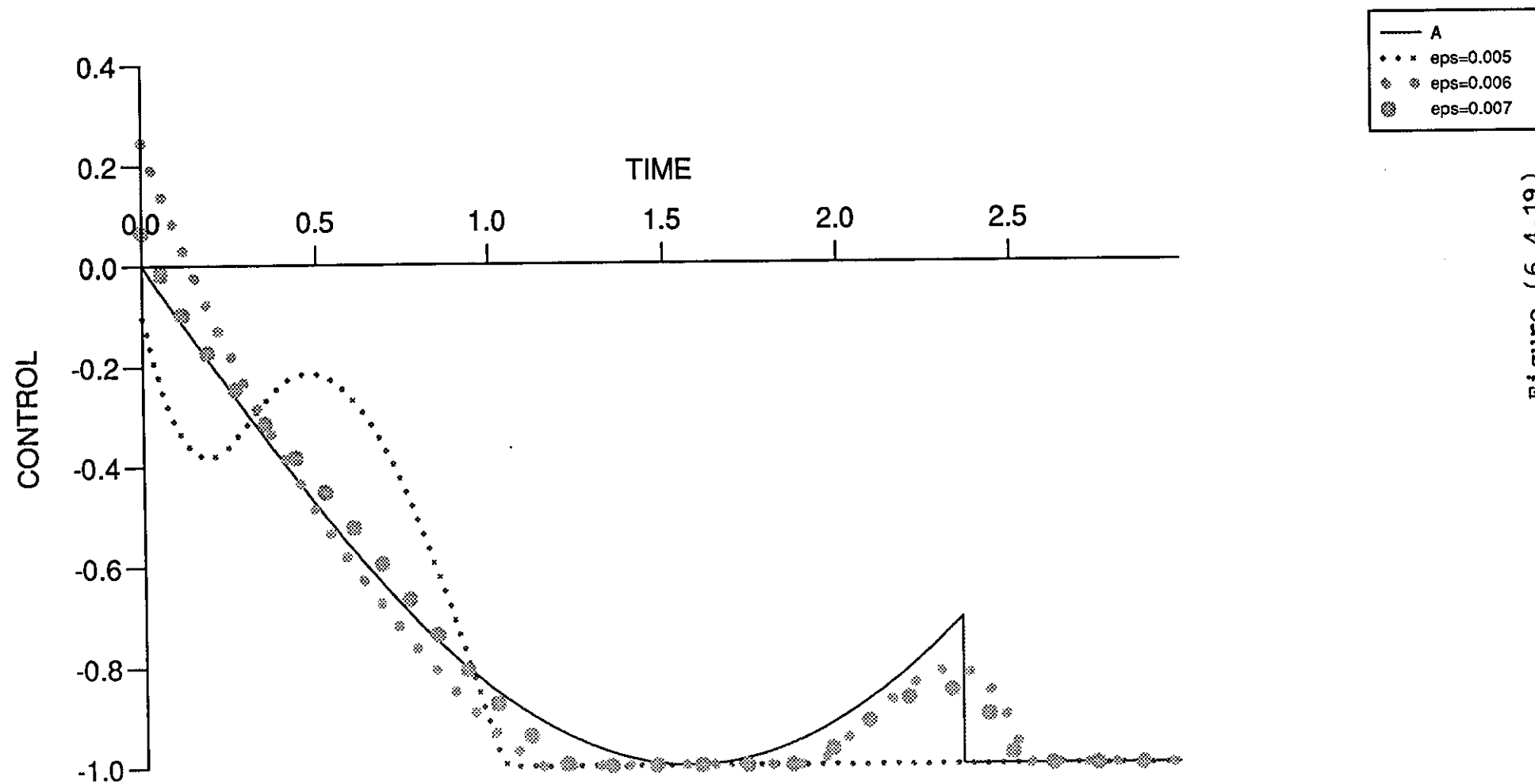


Figure (6.4.19)

H1

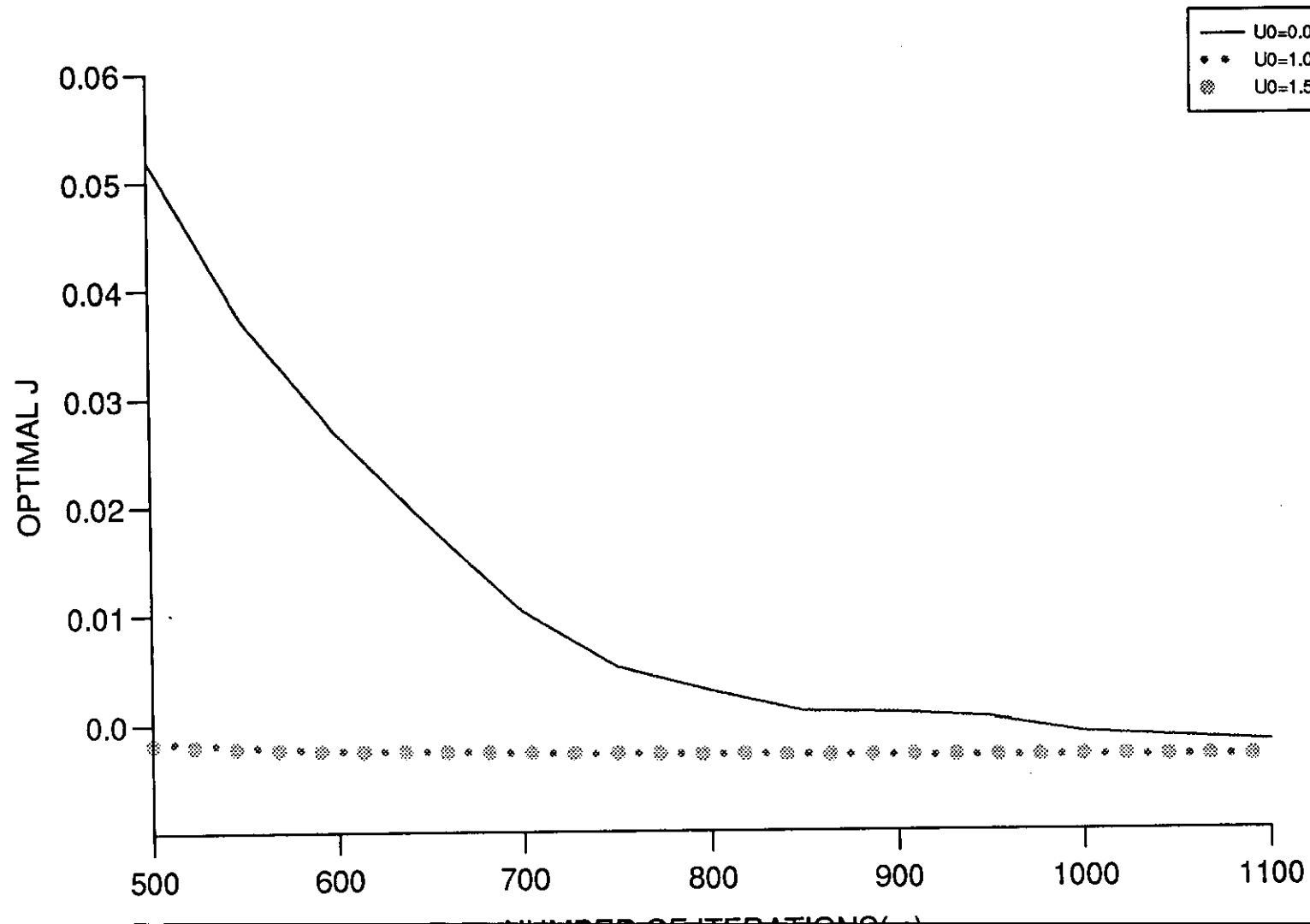


Figure (6.4.20)

ATH

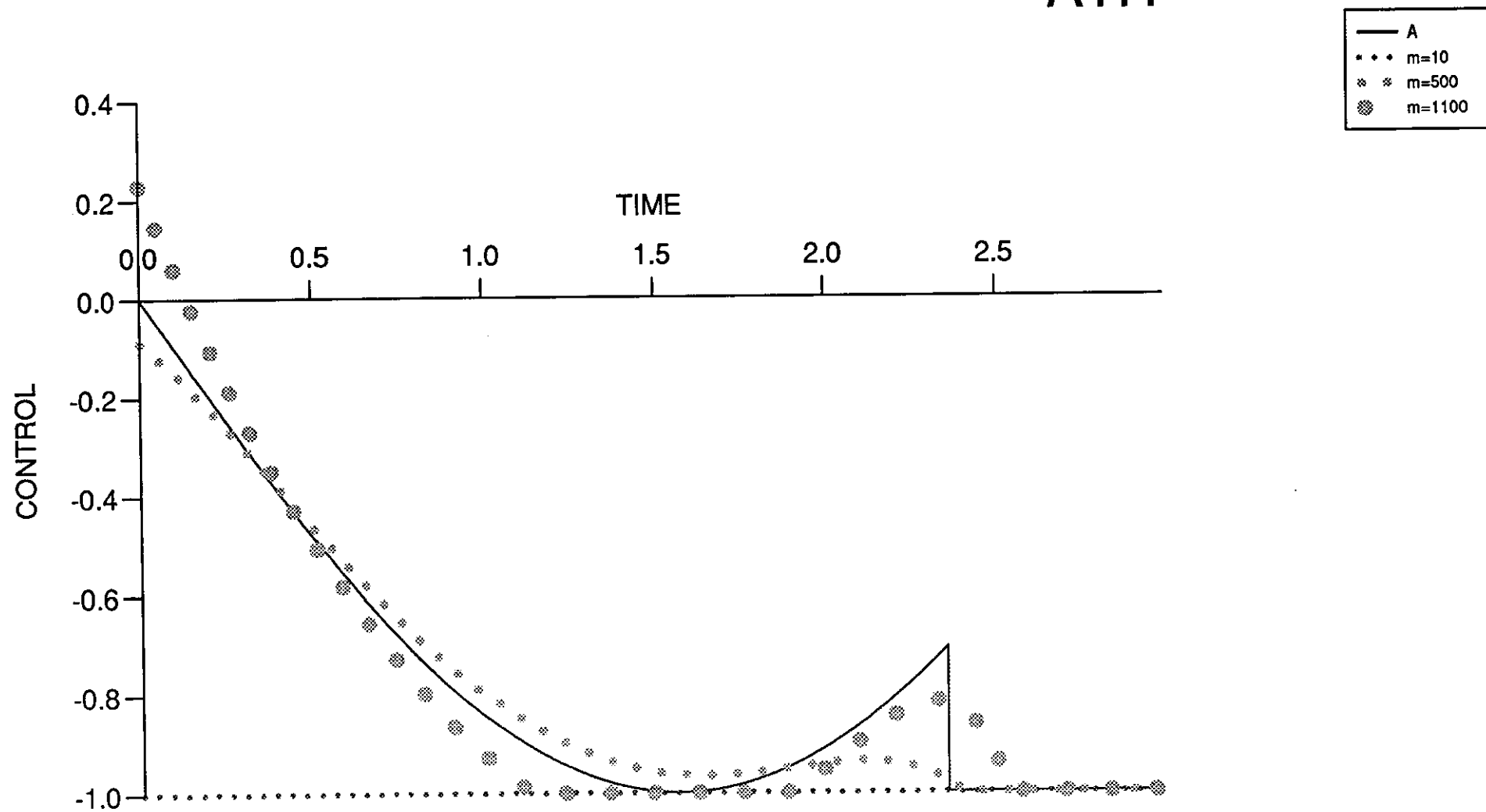


Figure (6.4.21)

ATH

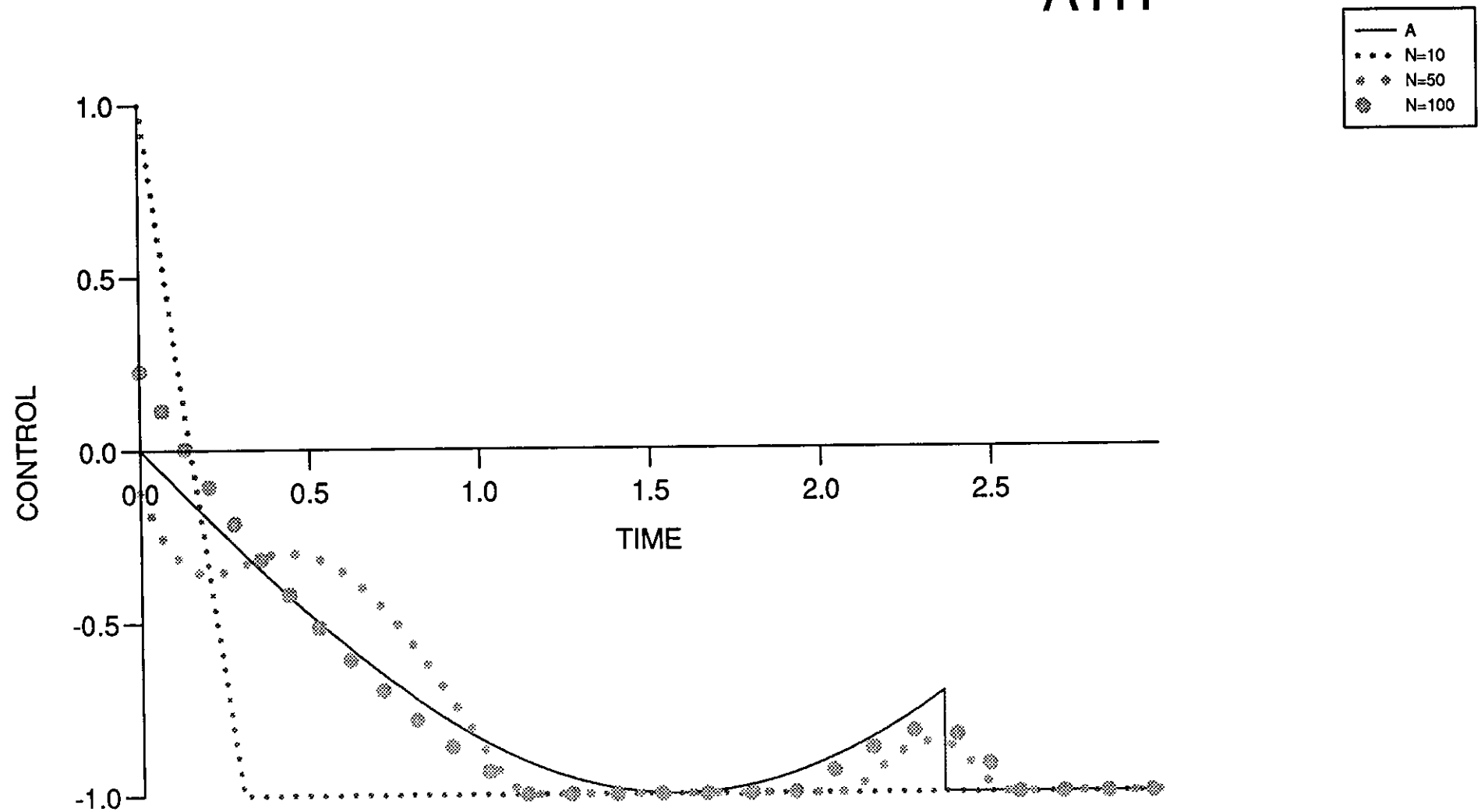


Figure (6.4.22)

ATH

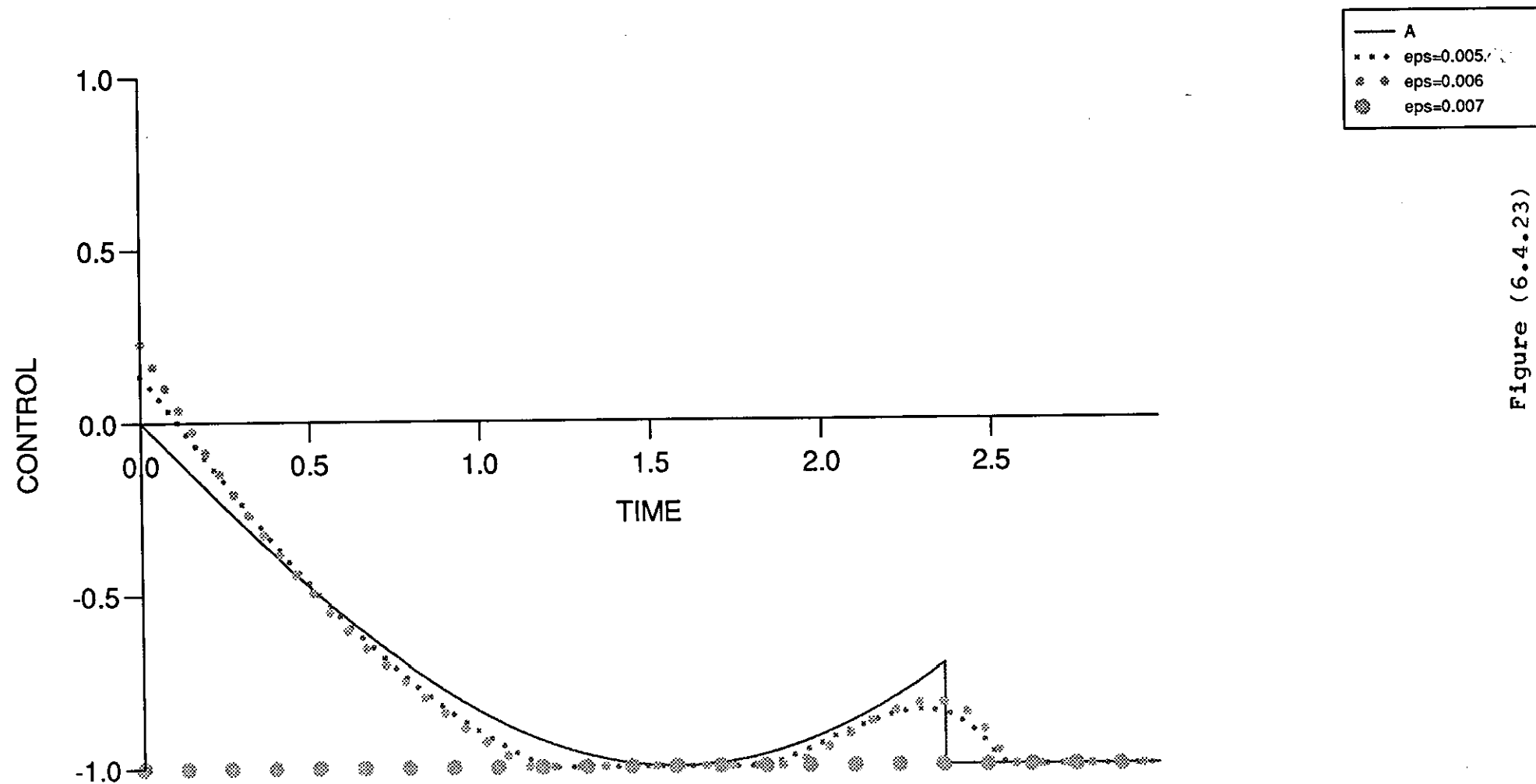


Figure (6.4.23)

ATH

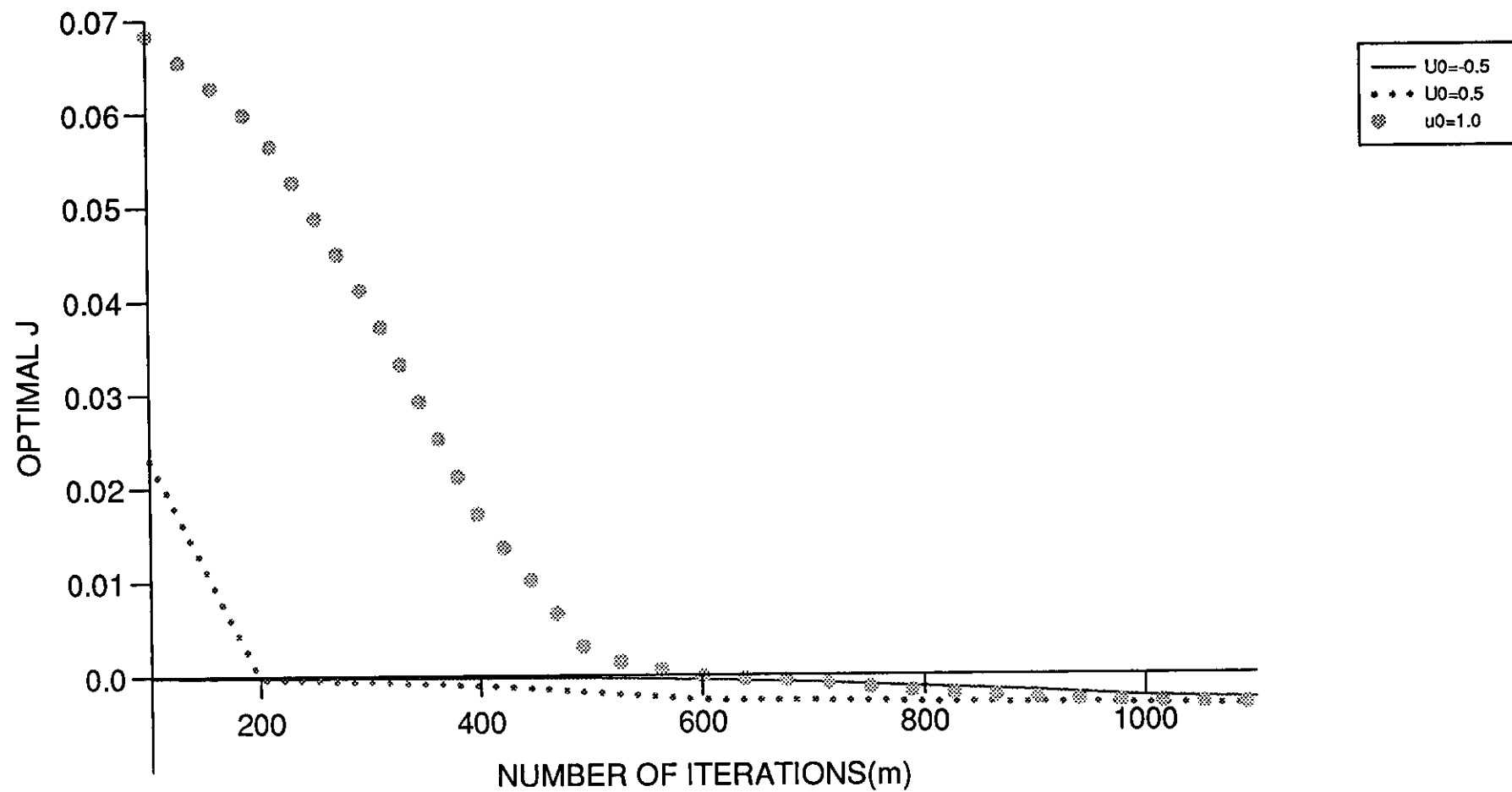


Figure (6.4.24)

H3

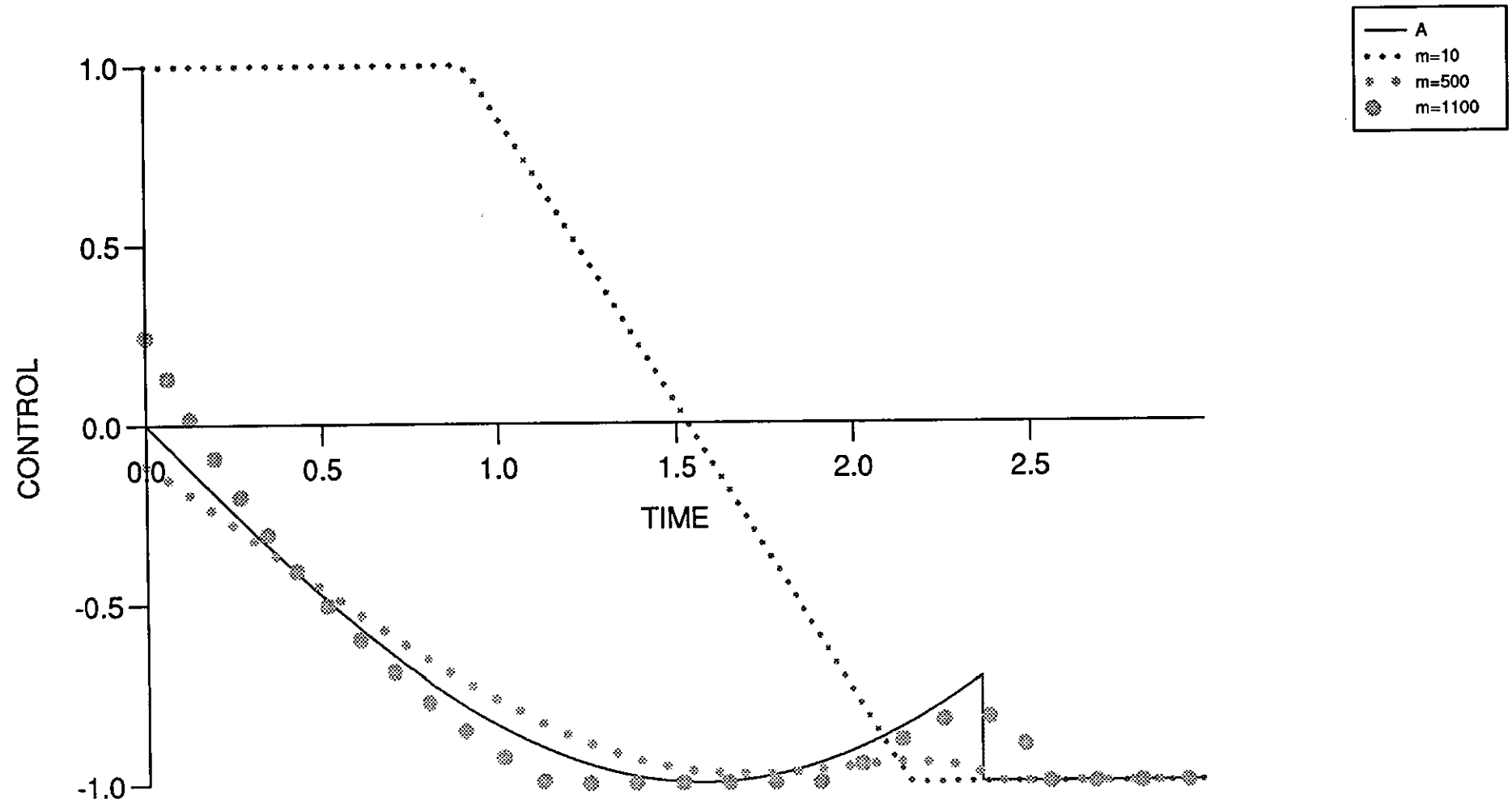


Figure (6. 4.25)

H3

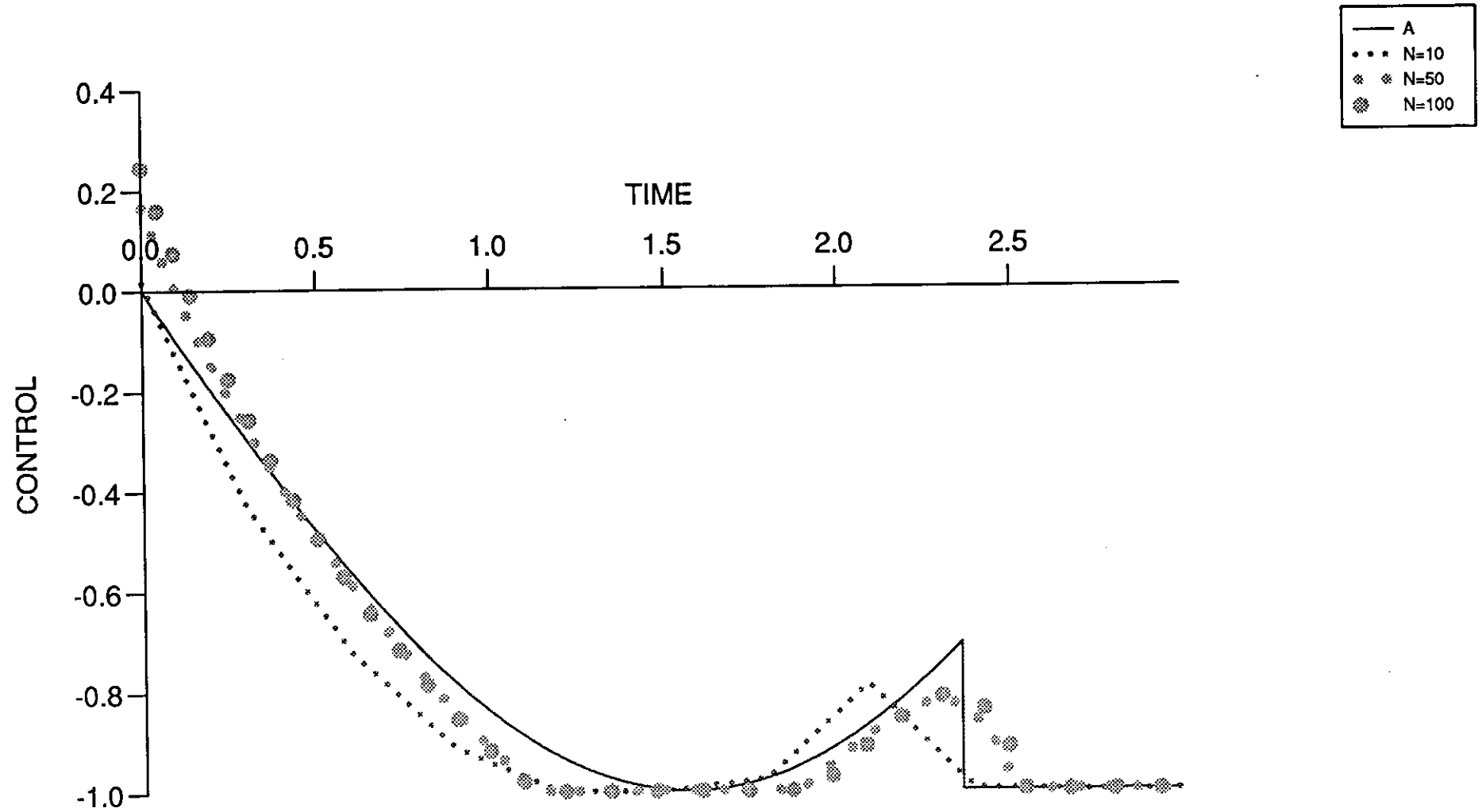


Figure (6.4.26)

H3

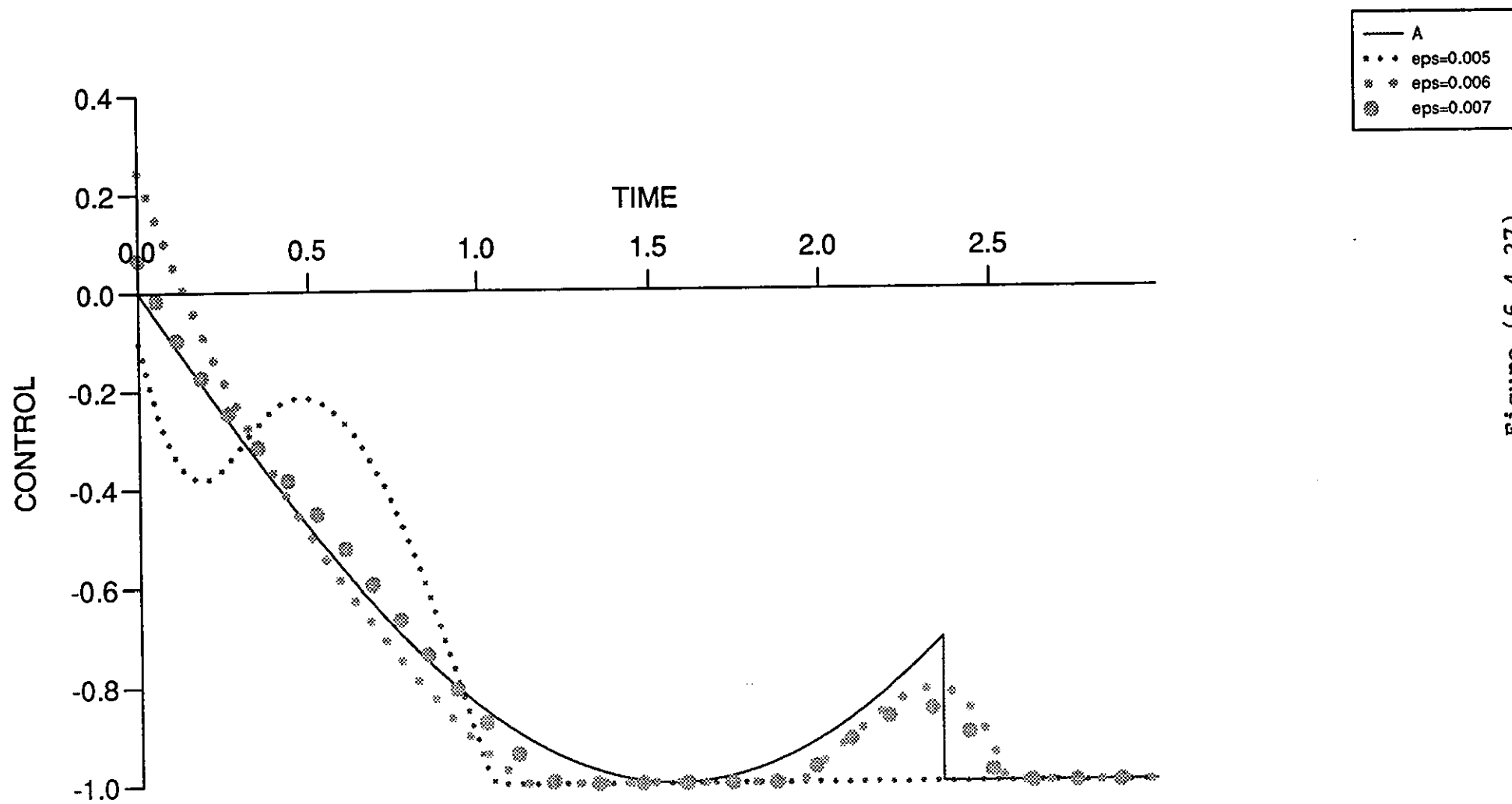


Figure (6.4.27)

H3

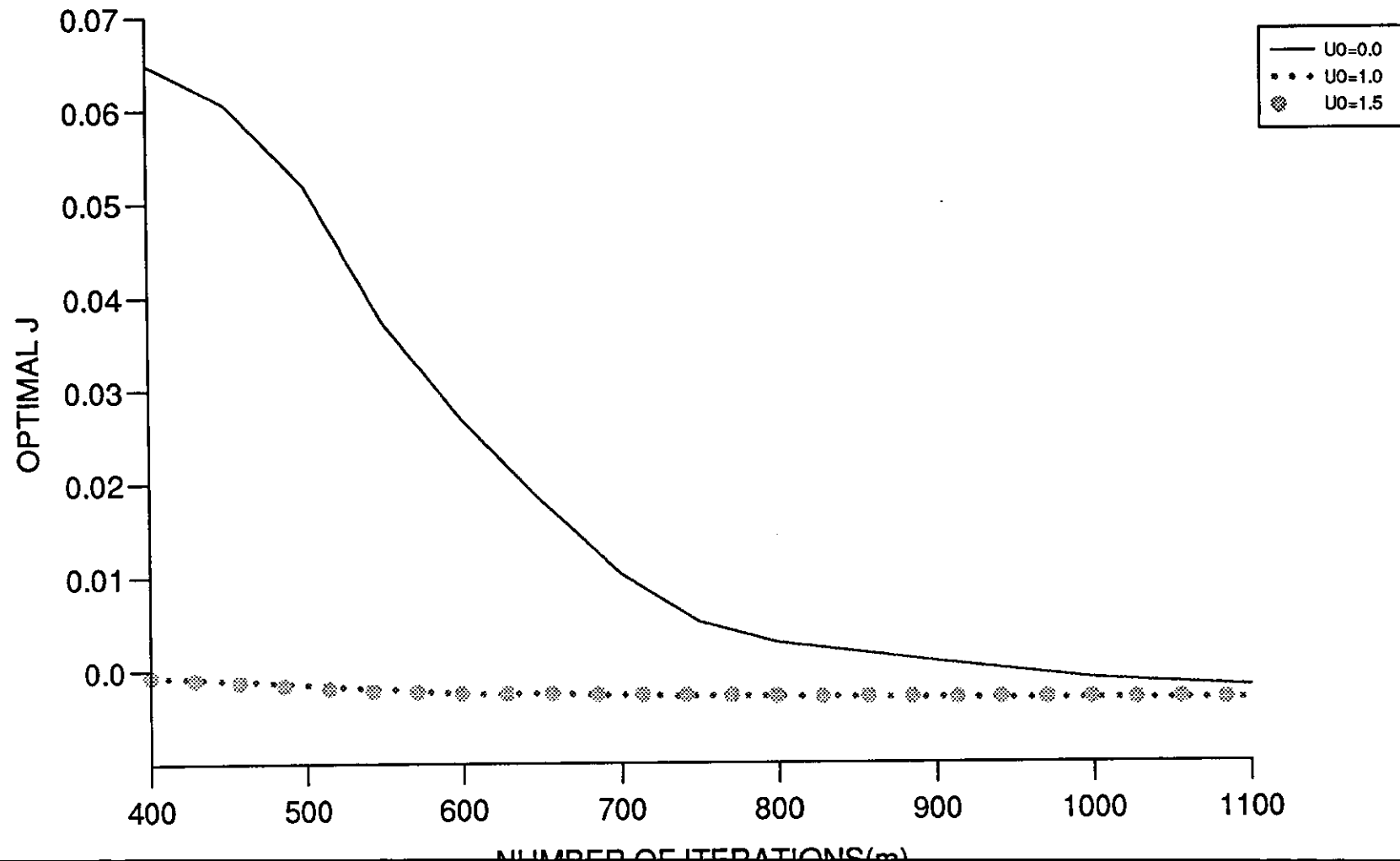


Figure (6.4.28)

COMPARISON OF THE SEVEN METHODS

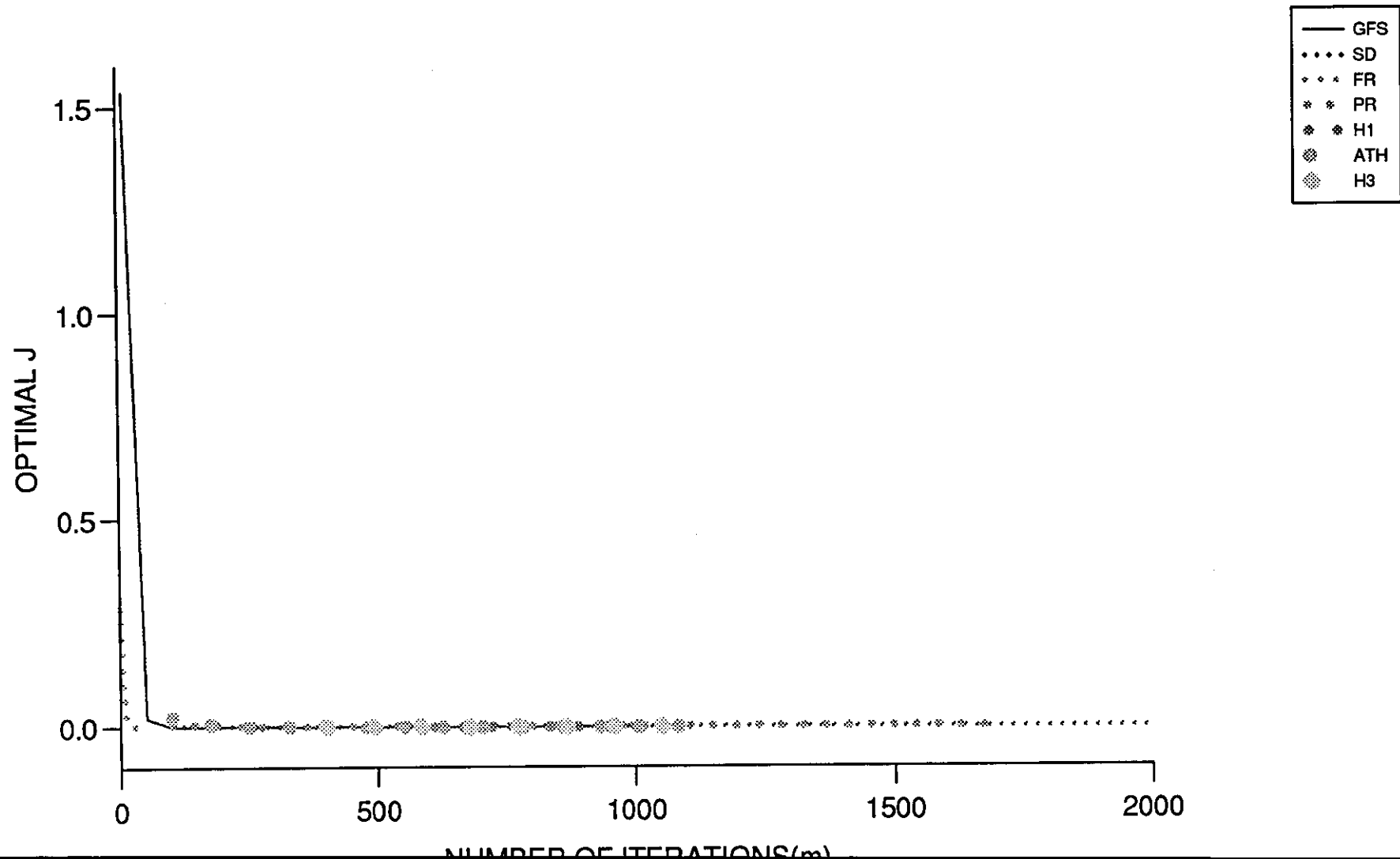


Figure (6.5.1)

Chapter 7

PROBLEM 4

7.1 An Optimal Control Problem with Fixed End Points

A system is described by,

$$\dot{x}_1 = -2x_1 + u, \quad x_1(0) = 1, \quad x_1(1) = 0, \quad (7.1.1)$$

$$0 \leq t \leq 1,$$

and the control $u(t)$ is to be chosen so as to minimize,

$$J = \int_0^1 u^2 dt. \quad (7.1.2)$$

This problem is taken from Barnett and Cameron (1985) [144].

It is solved first analytically as it stands and then with a penalty function introduced to satisfy $x_1(1) = 0$, thereby allowing known conditions for the adjoint equation at $t = 1$.

7.2 Analytical Solutions without the Penalty Function

Using the maximum principle, we solve the problem (7.1.1) and (7.1.2).

The Hamiltonian is given by,

$$H = u^2 + \lambda(-2x_1 + u), \quad (7.2.1)$$

and the adjoint equation is,

$$\dot{\lambda} = -\frac{\partial H}{\partial x_1} = 2\lambda. \quad (7.2.2)$$

To find the optimal control u^* , set

$$\frac{\partial H}{\partial u} = 0,$$

i.e.,

$$\lambda + 2u^* = 0,$$

so that,

$$u^* = -\frac{\lambda}{2}. \quad (7.2.3)$$

Now by solving the adjoint equation (7.2.2) we get,

$$\lambda = Ae^{2t}, \quad (7.2.4)$$

and substituting λ in (7.2.3) it follows that

$$u^* = -\frac{1}{2}Ae^{2t}. \quad (7.2.5)$$

Substituting u^* in the state equation (7.1.1) we get,

$$\dot{x}_1 + 2x_1 = -\frac{1}{2}Ae^{2t}. \quad (7.2.5)$$

so that,

$$x_1 = -\frac{A}{8}e^{2t} + Ce^{-2t}. \quad (7.2.6)$$

Since, $x_1(0) = 1$, (7.2.6) gives

$$C = 1 + \frac{A}{8}. \quad (7.2.7)$$

Also since, $x_1(1) = 0$, from (7.2.6) and (7.2.7) we get,

$$0 = -\frac{A}{8}e^2 + \left(1 + \frac{A}{8}\right)e^{-2}$$

so that

$$A = \frac{8e^{-2}}{e^2 - e^{-2}}. \quad (7.2.8)$$

By substituting A from (7.2.8) into (7.2.4) we get,

$$\lambda = \frac{8e^{-2+2t}}{e^2 - e^{-2}}. \quad (7.2.9)$$

Then from (7.2.9) and (7.2.3)

$$u^* = -\frac{1}{2} \left(\frac{8e^{-2+2t}}{e^2 - e^{-2}} \right),$$

i.e.,

$$u^* = \frac{-4e^{2t}}{e^4 - 1}. \quad (7.2.10)$$

x_1 can also be found by substituting C and A from (7.2.7) and (7.2.8) respectively into (7.2.6)

i.e.,

$$\begin{aligned} x_1 &= \frac{-e^{-2}}{(e^2 - e^{-2})} e^{2t} + \left(1 + \frac{e^{-2}}{(e^2 - e^{-2})}\right) e^{-2t}, \\ &= \frac{-e^{2t}}{(e^4 - 1)} + \frac{e^4}{(e^4 - 1)} e^{-2t}, \\ x_1 &= \frac{e^{4-2t} - e^{2t}}{e^4 - 1}. \end{aligned} \quad (7.2.11)$$

To find the optimal J , substitute u^* from (7.2.10) into (7.1.2) to give,

$$\begin{aligned} J^* &= \int_0^1 \left(\frac{-4e^{2t}}{e^4 - 1} \right)^2 dt, \\ &= \frac{16}{(e^4 - 1)^2} \int_0^1 e^{4t} dt, \\ &= \frac{4}{(e^4 - 1)^2} [e^{4t}]_0^1, \\ J^* &= \frac{4}{e^4 - 1}. \\ J^* &= 0.074629441 \text{ to 9 decimal places.} \end{aligned} \quad (7.2.12)$$

7.3 Analytical solutions with the Penalty Function

By introducing a penalty function the problem (7.1.1) to (7.1.2) can be transformed into the following;

$$\text{minimize } J = kx_1^2(1) + \int_0^1 u^2 dt, \quad (7.3.1)$$

subject to;

$$\dot{x}_1 = -2x_1 + u, \quad x_1(0) = 1, \quad 0 \leq t \leq 1. \quad (7.3.2)$$

Once again using the maximum principle, gives us the Hamiltonian and the adjoint equation, as in (7.2.1) and (7.2.2) respectively. However we now have,

$$\phi(x_1(1)) = kx_1^2(1),$$

and

$$\lambda(1) = \frac{\partial \phi}{\partial x}(x_1(1)) = 2kx_1(1).$$

The optimal control u^* can be obtained by setting

$$\frac{\partial H}{\partial u} = 0, \quad \text{i.e.,} \quad \lambda + 2u^* = 0,$$

so that, u^* can be obtained as (7.2.3).

Also by solving the adjoining equation (7.2.2) we get, λ as (7.2.4), and by substituting λ in (7.2.3), u^* follows as (7.2.5).

Substituting u^* in the state equation (7.3.2) we get (7.2.5), and x_1 follows as (7.2.6).

Here since $\lambda(1) = Ae^2 = 2kx_1(1)$, thus

$$x_1(t_f) = \frac{Ae^2}{2k}. \quad (7.3.3)$$

Since $x_1(t) = -\frac{A}{8}Ce^{2t} + Ce^{-2t}$ and $x_1(0) = 1$, C can be found as, $C = 1 + \frac{A}{8}$, thus,

$$\begin{aligned} x_1(t) &= -\frac{A}{8}e^{2t} + \left(1 + \frac{A}{8}\right)e^{-2t}, \\ x_1(1) &= -\frac{A}{8}e^2 + \left(1 + \frac{A}{8}\right)e^{-2}. \end{aligned} \quad (7.3.4)$$

From (7.3.3) and (7.3.4) we get,

$$\frac{Ae^2}{2k} = -\frac{A}{8}e^2 + \left(1 + \frac{A}{8}\right)e^{-2},$$

i.e.,

$$A = \frac{e^2}{\left[\frac{1}{2k}e^2 + \frac{1}{8}e^2 - \frac{1}{8}e^{-2}\right]},$$

or

$$A = \frac{1}{e^2 \left(-\frac{1}{8e^2} + \frac{e^2}{8} + \frac{e^2}{2k}\right)}.$$

And,

$$\begin{aligned} x_1(1) &= \frac{1}{8e^2 \left(-\frac{1}{8e^2} + \frac{e^2}{8} + \frac{e^2}{2k}\right)} (e^{-2} - e^2) + e^{-2}, \\ &= \frac{e^{-2} - e^2}{8e^2 \left(-\frac{1}{8e^2} + \frac{e^2}{8} + \frac{e^2}{2k}\right)} + e^{-2}, \\ &= \frac{e^{-2} - e^2 + 8e^2 \left(-\frac{1}{8e^2} + \frac{e^2}{8} + \frac{e^2}{2k}\right)e^{-2}}{8e^2 \left(-\frac{1}{8e^2} + \frac{e^2}{8} + \frac{e^2}{2k}\right)}, \\ &= \frac{\frac{4e^2}{k}}{-1 + e^4 + \frac{4e^4}{k}}. \end{aligned}$$

Multiply top and bottom by k we get,

$$x_1(1) = \frac{4e^2}{4e^4 + ke^4 - k}. \quad (7.3.5)$$

We also find λ and u^* as,

$$\lambda = \frac{e^{-2+2t}}{\left[\frac{1}{2k}e^2 + \frac{1}{8}e^2 - \frac{1}{8}e^{-2}\right]}, \quad (7.3.6)$$

and,

$$u^* = \frac{-e^{2t}}{\left[\frac{1}{k}e^4 + \frac{1}{4}e^4 - \frac{1}{4}\right]}, \quad (7.3.7)$$

$$\begin{aligned} J^* &= kx_1^2(1) + \int_0^1 \left[\frac{-e^{2t}}{\left[\frac{1}{k}e^4 + \frac{1}{4}e^4 - \frac{1}{4}\right]} \right]^2 dt, \\ &= kx_1^2(1) + \frac{1}{\left[\frac{1}{k}e^4 + \frac{1}{4}e^4 - \frac{1}{4}\right]^2} \int_0^1 e^{4t} dt, \\ &= kx_1^2(1) + \frac{1}{\left[\frac{1}{k}e^4 + \frac{1}{4}e^4 - \frac{1}{4}\right]^2} \left[\frac{1}{4}e^{4t} \right]_0^1. \end{aligned}$$

Hence in order to substitute for $x_1(1)$, let us multiply top and bottom of (7.3.5) by $\frac{1}{4k}$. Then we get

$$x_1(1) = \frac{e^2}{k \left[\frac{1}{k}e^4 + \frac{1}{4e^4} - \frac{1}{4} \right]},$$

and then substitute it back in J^* we get,

$$\begin{aligned} J^* &= \frac{e^4}{k \left[\frac{1}{k}e^4 + \frac{1}{4}e^4 - \frac{1}{4} \right]^2} + \frac{e^4 - 1}{4 \left[\frac{1}{k}e^4 + \frac{1}{4}e^4 - \frac{1}{4} \right]^2}, \\ &= \frac{4e^4 + ke^4 - k}{4k \left[\frac{1}{k}e^4 + \frac{1}{4}e^4 - \frac{1}{4} \right]^2}, \end{aligned}$$

multiply top and bottom by $\frac{1}{4k}$ we get,

$$J^* = \frac{1}{\frac{1}{k}e^4 + \frac{1}{4}e^4 - \frac{1}{4}}.$$

Multiply top and bottom by $4k$ we get,

$$J^* = \frac{4k}{4e^4 + ke^4 - k}. \quad (7.3.8)$$

In order to find the value of k , so that u^* with the penalty function is equivalent to u^* without it, we set up the following procedures:

From (7.2.10) and (7.3.7) we get,

$$\begin{aligned} \frac{-4e^{2t}}{e^4 - 1} &= \frac{e^{-2t}}{\left[\frac{1}{k}e^4 + \frac{1}{4}e^4 - \frac{1}{4} \right]}, \\ -4e^{2t} \left[\frac{1}{k}e^4 + \frac{1}{4}e^4 - \frac{1}{4} \right] &= -(e^4 - 1)e^{2t}, \\ \frac{1}{k}e^4 + \frac{e^4 - 1}{4} &= \frac{e^4 - 1}{4}, \\ \frac{1}{k}e^4 &= 0, \quad \frac{1}{k} = 0, \quad \text{when } k \rightarrow \infty. \end{aligned}$$

The same value of k can also be confirmed for J^* in the following manner;

From (7.2.20),

$$\lim_{k \rightarrow \infty} \frac{4k}{4e^4 + ke^4 - k} = \frac{4}{\frac{4}{k}e^4 + e^4 - 1} = \frac{4}{e^4 - 1}$$

which is equivalent to (7.2.12).

Thus the two problems are only equivalent when $k \rightarrow \infty$ and for any other k the solutions are different, i.e., the problem solved is that with $x_1(1)$ given by (7.1.1). Refer to Fig (7.3.1) to see how the optimal j varies with different k 's. Notice that J increases monotonically with k so that great care must be taken in numerical work to ensure that the correct optimal J has been obtained. In order to get J^* correct to 6 decimal places would require taking $k > 6.7 \times 10^8$.

7.4 Results and Discussion

7.4.1 Gradient Method

The algorithm for the gradient in function space method applied to problem 4 is the same as that described in Chapter 4, Section 4.4.1, where for this problem,

$$g = \lambda_1 + 2u\lambda_2. \quad (7.4.1)$$

The critical parameters involved in this problem are k , ε , h and u_0 , and we have examined the effect of these parameters on the optimal J .

Remark 7.4.1: Here we should note that throughout this chapter, the absolute error (e_{abs}) is the difference between the value of the optimal J obtained analytically, without the penalty function and the value of J^* obtained numerically, except in Tables (7.4.1), (7.4.4), (7.4.7), (7.4.10), (7.4.13), (7.4.16) and finally (7.4.19), where e_{abs} is the difference between the value of J^* obtained analytically with the penalty function, with $k = 500$, and the value of J^* obtained numerically. Also in this chapter the value of coefficient of penalty function k , has been taken as 500, which is sufficiently large enough to be selected as a reasonable value for testing.

The analytical value of J^* , when $k = 500$ is obtained from (7.3.8) and is 0.07402618 to 8 decimal places, and the absolute error between this and the exact analytical value of J^* (which obviously can not be better unless a larger k is used or because of numerical inaccuracy) is 6.03261×10^{-4} .

Table (7.4.1), shows the effect of ε and N in achieving the minimum value of J^* , from the best initial control $u_0 = 0.0$. Taking $\text{Acc} \leq 10^{-6}$, the best J^* obtained was 0.07402869 to 8 decimal places with $\|g\| = 7.05 \times 10^{-7}$, $e_{abs} = 2.51 \times 10^{-6}$ (refers to remark 7.4.1), and the corresponding parameter values, $\varepsilon = 0.004$, $k = 500$, $N = 100$ or $N = 200$ and $m = 7$. Selecting ε fairly small in the range $[0.003, 0.006]$, produced consistent results, for all N 's. When the best choice ε is selected with larger N , it takes fewer iterations to satisfy the $\text{Acc} \leq 10^{-6}$, for the optimal J^* . Here selecting N as 200 as opposed to 100 may result in slightly better minimum J^* for some ε 's but at the cost of more computing time.

Table (7.4.2), shows, the effect of selecting k , for different values of Acc i.e., $\leq 10^{-2}$, $\leq 10^{-4}$ and $\leq 10^{-6}$, with $u_0 = 0.0$ and $N = 100$. As can be seen from the Table the closest solution to the optimal one was obtained, when k was selected as 500, and when k was taken too small, say $k = 10$, the e_{abs} i.e., the difference between the analytical solution without the penalty function and numerical one was fairly great.

Table (7.4.3), shows the effect the choice of initial control u_0 on the minimum J^* , with $N = 100$, $k = 500$ and the best Acc possible, i.e. $\text{Acc} \leq 10^{-6}$. it is clear from the Table that there is a big difference between selecting u_0 as 0.0 as opposed to u_0 's in the range $[-0.6, -0.1]$, in terms of number of iterations taken to minimize J^* .

We have also compared the effect of the critical parameters, graphically, in Figures (7.4.1) to (7.4.5), which show the numerical solutions together with the analytical ones.

Figure (7.4.1), shows the plot of time against u for different values of k , i.e., $k = 10$, with $\varepsilon = 0.05$, $m = 35$, $k = 100$, with $\varepsilon = 0.005$, $m = 57$ and $k = 500$, with $\varepsilon = 0.004$, $m = 7$, when $N = 100$, $u_0 = 0.0$ and $\text{Acc} \leq 10^{-6}$.

As can be seen from the graphs, as the value of k increases, the behaviour of the curves get closer to the analytical one, without the penalty function.

Fig (7.4.2), shows the effect of m on u , with $N = 100$, $k = 500$, $u_0 = 0.0$ and $\varepsilon = 0.004$. Here as m increases, the curve of numerical solutions get closer to the analytical one. But here since, it does not take many iterations for GFS, to obtain the optimal J^* , therefore even by taking $m = 1$, it produces close solution to the analytical one.

Figure (7.4.3), compares similarly the effect of N on control, with $N = 10$, 50 and 100 respectively, with their corresponding number of iterations 9, 8 and 7, associated with their best step length factors of 0.005, 0.004 and 0.004, when $k = 500$, $m = 4$ and $u_0 = 0.0$. Here taking N large, i.e., 50 or 100 produced numerical curves closer to the analytical ones than, taking N small.

Fig (7.4.4), illustrates the effect on control of using, $\varepsilon = 0.0005$, 0.008 and 0.004, after 4 iterations, when the starting control is 0.0, $k = 500$ and $N = 100$. As can be seen from the plots, the best ε that its numerical curve behaved similar to the analytical one was $\varepsilon = 0.004$. Taking ε too small i.e., 0.0005 or large in this case 0.008 did not produce similar results as the analytical one.

Finally Fig (7.4.5), shows the effect of u_0 on optimal J for different values of m , with $u_0 = -0.6$, -0.3 and 0.0 , respectively with their best corresponding ε 's of 0.008, 0.008 and 0.004, where $N = 100$ and $k = 500$. Here by taking starting controls as $u_0 = -0.3$, and 0.0 , we can see that their behaviours are the same graphically and they both converge towards the optimal value of J , after the 1st iteration. But taking a distant initial control, i.e., $u_0 = -0.6$, we can see that the values of optimal J gradually converge, in the direction of minimum optimal J , but in a much slower rate of speed than the other two initial controls. The above results, show that improvement can be obtained in minimising the optimal J by varying the critical parameters, k , u_0 , ε and N . The correct choice of initial control is very crucial, in speeding up the search for an optimal J , by reducing the number of iterations and function evaluations and consequently computing time. The interaction between ε and N , show that for smaller N , it may require a relatively larger ε , but up to the limit of < 0.006 , in order to find minimum J^* in fewer iterations, but for larger N , sufficiently small ε can produce the minimum J^* in fewer iterations. In view of above comments, by taking k large enough, also selecting a proper

u_0 and large N , along with an appropriate value for ε , the convergency, to the minimum optimal J^* can be speeded up.

7.4.2 Steepest Descent

The algorithm for steepest descent applied to the problem 4 is as described in Chapter 2, Section 2.2.2. The line search technique used for this method is the quadratic interpolation method of Powell which was described in Chapter 3, Section 3.4.2. The gradient g is as (7.4.1). Here also as before we have examined the effect of critical parameters, k , ε , h and u_0 on the optimal J^* . Table (7.4.4), shows the effect of ε and N , in achieving the minimum optimal J^* , with the best starting control $u_0 = 0.0$, and $\text{Acc} \leq 10^{-6}$. The best J^* obtained was 0.07402869 to 8 decimal places, with $\|g\| = 5.23 \times 10^{-7}$, $e_{abs} = 2.5 \times 10^{-6}$, $\varepsilon = 0.007$ or 0.008 , $k = 500$, $N = 100$, or $N = 200$ and $m = 3$. Here also selecting ε fairly small in the range $[0.006, 0.009]$ can produce consistent results. Taking larger N , with appropriate ε , results in finding the minimum J^* , in fewer iterations and function evaluations. For N small, say 10, selecting ε in the range $[0.008, 0.009]$, results in finding J^* for the required Acc in slightly fewer iterations, than other ε 's.

Table (7.4.5), shows the effect of k on the values of Acc , ($\leq 10^{-2}$, $\leq 10^{-4}$ and $\leq 10^{-6}$), with $u_0 = 0.0$ and $N = 100$. Here also, the numerical experiment shows by taking larger k , the numerical solution gets closer to the analytical one without the penalty function.

Table (7.4.6), shows the effect of selecting u_0 on the optimal J^* with $N = 100$ and $k = 500$, in terms of the best Acc , possible for each u_0 . Here also as can be seen from the Table, selecting u_0 as 0.0 can make a lot of difference in achieving the best minimum J^* in fewer iterations, with the best Acc , compared to the other starting controls.

Fig (7.4.6), shows the plot of time against u , for different values of k , i.e., $k = 10$, 100 and 500 respectively, with their corresponding values of (ε, m) as (0.3, 4), (0.02, 4) and (0.07, 3) for the $\text{Acc} \leq 10^{-6}$, when $N = 100$ and $u_0 = 0.0$. From the plots, it can be seen, as the value of k increases the behaviour of the numerical curves get closer to the analytical one without the penalty function.

Fig (7.4.7), shows the effect of m on u , with $N = 100$, $k = 500$, and $\varepsilon = 0.007$. Here since, it takes only a few iterations to achieve J^* , even with one iteration, the numerical curve behave similar to the analytical one, and not much difference for higher number of iterations can be observed.

Fig (7.4.8), compares the effect of N on control, with $N = 10$, 50 and 100 respectively with their best corresponding step length factors of 0.009, 0.01 and 0.007, with $m = 1$, $u_0 = 0.0$ and $\text{Acc} \leq 10^{-6}$. Here as N gets larger the behaviour of the numerical curve is closer to the analytical one. Fig (7.4.9), illustrates similarly the effect of ε on control, with $\varepsilon = 0.0005$, 0.007 and

0.01 with $N = 100$, $u_0 = 0.0$ and $m = 1$. As can be seen from the graphs, taking ε too small, e.g., 0.0005, the numerical curve is far away from the analytical one, but with sufficient small ε 's, i.e., 0.007 or 0.01, the behaviour of the numerical curves are similar to the analytical one.

Fig (7.4.10), shows the effect of u_0 on optimal J , for different values of m , with $u_0 = -0.6$, -0.3 and 0.0 , respectively, with their best corresponding ε 's of 0.004, 0.004 and 0.007, where $N = 100$ and $k = 500$.

Here we can see that taking u_0 as 0.0, can speed up the process of convergence to optimal J , much faster than the other two starting controls, and the most distant starting control amongst them is $u_0 = -0.6$.

The numerical results, show that as with GFS, the right choice of k , u_0 , ε and N is crucial in achieving the best minimum for J^* . Here the correct choice of initial control is very important in obtaining the optimal J in fewer iterations and therefore less computing time.

7.4.3 Fletcher-Reeves

The algorithm for Fletcher-Reeves applied to the Problem 4, is as described in Chapter 2, Section 2.4. The norm of each gradient trajectory can be calculated as was described in Chapter 2, Section 2.10.1. The line search technique and the gradient g , are the same as the one for steepest descent in this Chapter 7.4.2. The effect of critical parameters, k , ε , N and u_0 on the optimal J , were examined and the results were similar to those obtained for SD. Refer to Tables (7.4.7) to (7.4.9) and Figures (7.4.11) to (7.4.15), with the same parameters as were taken for SD. Therefore the recommendations on selecting k , u_0 , ε and N can be followed as suggested for the steepest descent.

7.4.4 Polak-Ribière

The algorithm for the Polak-Ribière method is described in Chapter 2, Section 2.5. The line search techniques, the calculation of the norms and g are the same as those for FR in this chapter 3.4.3.

Hence also similar results obtained as SD and FR, when the critical parameters, k , u_0 , ε and N were tested. Refer to Tables (7.4.10) to (7.4.12) and Figures (7.4.16) to (7.4.20), with the same parameters as were taken for SD. Thus for the search of the best minimum optimal J , we should select k , u_0 , ε and N as we did for SD.

7.4.5 Hybrid 1

The algorithm for the Hybrid 1 method is described in Chapter 2, Section 2.6.1. The line search technique and the calculation of the norms, also g are the same as FR in this chapter, section 7.4.3.

The numerical experiments on critical parameters, k , u_0 , ε and N , are found to be similar to the previous methods, SD, FR and PR.

Refer to Tables (7.4.13) to (7.4.15) and Figures (7.4.21) to (7.4.25), with the same parameters as for SD.

Here also the recommendations on finding the best minimum optimal J^* can be followed as for SD.

7.4.6 Angle Test Hybrid

The algorithm for the Angle test hybrid method is described in Chapter 2, Section 2.7. The line search, calculation of the norms and g are the same as for FR in this chapter section 7.4.3.

The method was tested in the same way as the previous methods plus the new parameter $\tau > 0$, i.e., 0.01, 0.0001 and 0.000001.

The results obtained were similar to SD. Refer to Tables (7.4.16) to (7.4.18) and Figures (7.4.25) to (7.4.30), with the same parameters as for SD, plus $\tau = 0.000001$.

Here the effect of τ on obtaining the minimum J^* was practically negligible, therefore the same suggestions for achieving the optimal J^* for SD could be applied to ATH.

7.4.7 Hybrid 3

The algorithm for Hybrid 3 method can also be found in Chapter 2, Section 2.8. The calculation of the norms, and the line search, also g are the same as FR in this chapter, section 7.4.3.

The effect of new parameters $\lambda > 0$ and $\mu < \frac{1}{2}$ had to be considered for this method. So μ was taken as 0.2, 0.3, 0.45 and 0.4999 and λ as 0.01, 0.0001 and 0.000001.

The method was tested as before with the new parameters μ and λ and the results obtained were again similar to SD.

Refer to Tables (7.4.19) to (7.4.21) and Figures (7.4.31) to (7.4.35) with the same parameters as SD, plus $\mu = 0.45$ and $\lambda = 0.000001$.

Here the effect of λ and μ on the optimal J were negligible. Thus, the same recommendations on obtaining minimum J^* as SD on other critical parameters can also be applied to H3.

7.5 Summary of the Results

The results are summarized in Table (7.5.1) with $N = 100$, $k = 500$, for ATH, $\tau = 0.000001$ and for H3, $\lambda = 0.000001$, $\mu = 0.4999$, and also in Fig (7.5.1), that shows the plots of J against number of iterations for all seven optimization techniques, with their best ε , $N = 100$, $u_0 = 0.0$ and $k = 500$.

At u_0 in the range $[-0.6, -0.1]$, we can see that although for GFS, it takes more number of iterations to achieve minimum J^* than other 6 methods, but the $\text{Acc} \leq 10^{-6}$ obtained for this method is more accurate than the other methods, since they can not achieve this, even by increasing m .

At $u_0 = 0.0$ the six methods of SD, FR, PR, H1, ATH and H3 performed similarly, and they all found the minimum value of J^* in fewer iterations and also better Acc than GFS.

7.6 Conclusion

The results for this simple problem, show that the best value for the initial control was 0.0, for all the methods, to achieve their best minimum J^* , in terms of Acc, number of iterations, number of function evaluations and finally computing time.

For those methods that use line search the choice of initial control was very crucial, since with a bad guess the $\text{Acc} \leq 10^{-6}$, could never be achieved.

The other common factors for obtaining the optimal J^* for all the methods were choice of suitable large integration step number (N) and small step factor (ε).

All the methods that use line search performed about the same, and when u_0 was selected as 0.0, they all performed better than GFS, in terms of Acc and number of iterations to find the optimal J^* .

Obviously this problem is very easy to solve numerically and so does not bring out the expected improvement for the Hybrid methods, in terms of number of iterations. The importance of selecting k sufficiently large to get a reasonable approximation for the optimal J^* with the correct end condition on the state is also demonstrated.

TABLE (7.4.1):Results of GFS with vatying ϵ and N.

N ϵ	10	50	100	200
0.001	$J^*=0.07428361$ $m=20$ $\ g\ =9.15 \times 10^{-7}$ $e_{abs}=2.57 \times 10^{-4}$ Cputime<1 NFE=21	$J^*=0.07403651$ $m=17$ $\ g\ =9.72 \times 10^{-7}$ $e_{abs}=1.03 \times 10^{-5}$ Cputime<1 NFE=18	$J^*=0.07402891$ $m=17$ $\ g\ =9.83 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime<1 NFE=18	$J^*=0.07402889$ $m=17$ $\ g\ =9.85 \times 10^{-7}$ $e_{abs}=2.71 \times 10^{-6}$ Cputime=1 NFE=18
0.003	$J^*=0.07428352$ $m=18$ $\ g\ =8.08 \times 10^{-7}$ $e_{abs}=2.57 \times 10^{-4}$ Cputime<1 NFE=19	$J^*=0.07403623$ $m=15$ $\ g\ =1.54 \times 10^{-7}$ $e_{abs}=1.0 \times 10^{-5}$ Cputime<1 NFE=16	$J^*=0.07402869$ $m=16$ $\ g\ =1.61 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime<1 NFE=17	$J^*=0.07402869$ $m=16$ $\ g\ =1.61 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime=1 NFE=17
0.004	$J^*=0.07428351$ $m=12$ $\ g\ =8.29 \times 10^{-7}$ $e_{abs}=2.57 \times 10^{-4}$ Cputime<1 NFE=13	$J^*=0.07403623$ $m=8$ $\ g\ =1.39 \times 10^{-7}$ $e_{abs}=1.0 \times 10^{-5}$ Cputime<1 NFE=9	$J^*=0.07402869$ $m=7$ $\ g\ =7.05 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime<1 NFE=8	$J^*=0.07402869$ $m=7$ $\ g\ =7.05 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime=1 NFE=8
0.005	$J^*=0.07428351$ $m=9$ $\ g\ =8.30 \times 10^{-7}$ $e_{abs}=2.57 \times 10^{-4}$ Cputime<1 NFE=10	$J^*=0.07403623$ $m=13$ $\ g\ =3.60 \times 10^{-7}$ $e_{abs}=1.0 \times 10^{-5}$ Cputime<1 NFE=9	$J^*=0.07402869$ $m=14$ $\ g\ =3.23 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime<1 NFE=15	$J^*=0.07402869$ $m=14$ $\ g\ =3.23 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime=1 NFE=15
0.006	$J^*=0.07428351$ $m=17$ $\ g\ =8.39 \times 10^{-7}$ $e_{abs}=2.57 \times 10^{-4}$ Cputime<1 NFE=18	$J^*=0.07403623$ $m=26$ $\ g\ =8.12 \times 10^{-7}$ $e_{abs}=1.0 \times 10^{-5}$ Cputime<1 NFE=27	$J^*=0.07402869$ $m=24$ $\ g\ =9.82 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime<1 NFE=25	$J^*=0.07402869$ $m=24$ $\ g\ =9.82 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime=1 NFE=25

TABLE (7.4.2):Results of GFS with varying ACC and K.

ACC K	$\leq 10^{-2}$	$\leq 10^{-4}$	$\leq 10^{-6}$
10	$J^*=0.05303199$ $m=13$ $\ g\ =8.68 \times 10^{-3}$ $\epsilon=0.05$ $e_{abs}=0.0215974$ $Cputime<1$ $NFE=14$	$J^*=0.05302648$ $m=24$ $\ g\ =1.55 \times 10^{-5}$ $\epsilon=0.05$ $e_{abs}=0.0216030$ $Cputime<1$ $NFE=25$	$J^*=0.05302644$ $m=35$ $\ g\ =8.73 \times 10^{-7}$ $\epsilon=0.05$ $e_{abs}=0.021603$ $Cputime<1$ $NFE=36$
100	$J^*=0.07171135$ $m=26$ $\ g\ =9.14 \times 10^{-3}$ $\epsilon=0.005$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime<1$ $NFE=27$	$J^*=0.07171020$ $m=42$ $\ g\ =8.62 \times 10^{-5}$ $\epsilon=0.005$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime<1$ $NFE=43$	$J^*=0.07171019$ $m=57$ $\ g\ =9.87 \times 10^{-7}$ $\epsilon=0.005$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime<1$ $NFE=58$
500	$J^*=0.07402869$ $m=4$ $\ g\ =5.47 \times 10^{-3}$ $\epsilon=0.004$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime<1$ $NFE=5$	$J^*=0.07402869$ $m=5$ $\ g\ =1.07 \times 10^{-5}$ $\epsilon=0.004$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime<1$ $NFE=6$	$J^*=0.07402869$ $m=7$ $\ g\ =7.05 \times 10^{-7}$ $\epsilon=0.004$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime<1$ $NFE=8$

TABLE (7.4.3):Results of GFS with change in u_0 .

u_0	m	J^*	$\ g\ $	e_{abs}	NFE	eps	Cputime
-0.6	891	0.07402869	8.00283×10^{-7}	6.01×10^{-4}	892	0.008	1
-0.5	912	0.07402869	9.91989×10^{-7}	6.01×10^{-4}	918	0.008	1
-0.4	810	0.07402869	6.27786×10^{-7}	6.01×10^{-4}	811	0.008	1
-0.3	845	0.07402869	3.64625×10^{-7}	6.01×10^{-4}	846	0.008	1
-0.2	817	0.07402869	3.64625×10^{-7}	6.01×10^{-4}	818	0.008	1
-0.1	815	0.07402869	8.39578×10^{-7}	6.01×10^{-4}	816	0.008	1
0.0	7	0.07402869	7.0457×10^{-7}	6.01×10^{-4}	8	0.004	<1

TABLE (7.4.4):Results of SD with varying ϵ and N.

N ϵ	10	50	100	200
0.003	J*=0.07428351 m=8 $\ g\ =8.42 \times 10^{-7}$ eabs= 2.57×10^{-4} Cputime<1 NFE=38	J*=0.07403623 m=5 $\ g\ =1.43 \times 10^{-7}$ eabs= 1.0×10^{-5} Cputime<1 NFE=24	J*=0.07402869 m=5 $\ g\ =1.71 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime<1 NFE=24	J*=0.07402869 m=4 $\ g\ =1.61 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime=1 NFE=20
0.006	J*=0.07428351 m=6 $\ g\ =8.15 \times 10^{-7}$ eabs= 2.57×10^{-4} Cputime<1 NFE=28	J*=0.07403623 m=4 $\ g\ =1.43 \times 10^{-7}$ eabs= 1.0×10^{-5} Cputime<1 NFE=20	J*=0.07402869 m=4 $\ g\ =1.61 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime<1 NFE=20	J*=0.07402869 m=3 $\ g\ =5.23 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime=1 NFE=20
0.007	J*=0.07428351 m=7 $\ g\ =8.35 \times 10^{-7}$ eabs= 2.57×10^{-4} Cputime<1 NFE=32	J*=0.07403623 m=4 $\ g\ =1.43 \times 10^{-7}$ eabs= 1.0×10^{-5} Cputime<1 NFE=20	J*=0.07402869 m=3 $\ g\ =5.23 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime<1 NFE=16	J*=0.07402869 m=3 $\ g\ =5.23 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime=1 NFE=16
0.008	J*=0.07428351 m=5 $\ g\ =8.40 \times 10^{-7}$ eabs= 2.57×10^{-4} Cputime<1 NFE=24	J*=0.07403623 m=4 $\ g\ =1.43 \times 10^{-7}$ eabs= 1.0×10^{-5} Cputime<1 NFE=20	J*=0.07402869 m=3 $\ g\ =5.23 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime<1 NFE=16	J*=0.07402869 m=3 $\ g\ =5.23 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime=1 NFE=16
0.009	J*=0.07428352 m=4 $\ g\ =8.19 \times 10^{-7}$ eabs= 2.57×10^{-4} Cputime<1 NFE=20	J*=0.07403623 m=4 $\ g\ =1.41 \times 10^{-7}$ eabs= 1.0×10^{-5} Cputime<1 NFE=20	J*=0.07402869 m=4 $\ g\ =1.62 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime<1 NFE=20	J*=0.07402869 m=4 $\ g\ =1.62 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime=1 NFE=20
0.01	J*=0.07428351 m=6 $\ g\ =8.15 \times 10^{-7}$ eabs= 2.57×10^{-4} Cputime<1 NFE=28	J*=0.07403623 m=4 $\ g\ =1.40 \times 10^{-7}$ eabs= 1.0×10^{-5} Cputime<1 NFE=20	J*=0.07402869 m=4 $\ g\ =1.62 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime<1 NFE=20	J*=0.07402869 m=4 $\ g\ =1.62 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime=1 NFE=20

TABLE (7.4.5) Results of SD with varying ACC and K.

ACC K	$\leq 10^{-2}$	$\leq 10^{-4}$	$\leq 10^{-6}$
10	$J^*=0.05302645$ $m=2$ $\ g\ =3.91 \times 10^{-3}$ $\epsilon=0.3$ $e_{abs}=0.0216030$ $Cputime < 1$ $NFE=12$	$J^*=0.05302645$ $m=3$ $\ g\ =1.14 \times 10^{-5}$ $\epsilon=0.3$ $e_{abs}=0.0216030$ $Cputime < 1$ $NFE=16$	$J^*=0.05302644$ $m=4$ $\ g\ =1.55 \times 10^{-7}$ $\epsilon=0.3$ $e_{abs}=0.021603$ $Cputime < 1$ $NFE=20$
100	$J^*=0.07171019$ $m=2$ $\ g\ =5.29 \times 10^{-3}$ $\epsilon=0.02$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime < 1$ $NFE=12$	$J^*=0.07171019$ $m=3$ $\ g\ =1.96 \times 10^{-6}$ $\epsilon=0.02$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime < 1$ $NFE=16$	$J^*=0.07171019$ $m=4$ $\ g\ =8.47 \times 10^{-8}$ $\epsilon=0.02$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime < 1$ $NFE=20$
500	$J^*=0.07402869$ $m=2$ $\ g\ =5.45 \times 10^{-3}$ $\epsilon=0.007$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime < 1$ $NFE=12$	$J^*=0.07402869$ $m=7$ $\ g\ =2.90 \times 10^{-5}$ $\epsilon=0.002$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime < 1$ $NFE=33$	$J^*=0.07402869$ $m=3$ $\ g\ =5.23 \times 10^{-7}$ $\epsilon=0.007$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime < 1$ $NFE=16$

TABLE (7.4.6):Results of SD with change in u_0 .

u_0	m	J^*	$\ g\ $	e_{abs}	NFE	eps	Cputime
-0.6	700	0.07403014	2.42598×10^{-3}	5.99×10^{-4}	2804	0.004	3
-0.5	550	0.07403580	5.38615×10^{-3}	5.94×10^{-4}	2204	0.004	2
- 0.4	300	0.07407081	1.31377×10^{-2}	5.59×10^{-4}	1204	0.004	2
-0.3	550	0.07403020	2.59041×10^{-3}	5.99×10^{-4}	2204	0.004	2
-0.2	300	0.07403763	6.04926×10^{-3}	5.92×10^{-4}	1204	0.008	2
-0.1	400	0.07403089	2.99524×10^{-3}	5.98×10^{-4}	1604	0.003	2
0.0	3	0.07402869	5.22767×10^{-7}	6.01×10^{-4}	16	0.007	<1

TABLE (7.4.8):Results of FR with varying ACC and K.

ACC K	$\leq 10^{-2}$	$\leq 10^{-4}$	$\leq 10^{-6}$
10	$J^*=0.05302645$ $m=2$ $\ g\ =3.91 \times 10^{-3}$ $\epsilon=0.3$ $e_{abs}=0.0216030$ $Cputime<1$ $NFE=12$	$J^*=0.05302645$ $m=3$ $\ g\ =1.14 \times 10^{-5}$ $\epsilon=0.3$ $e_{abs}=0.0216030$ $Cputime<1$ $NFE=16$	$J^*=0.05302644$ $m=4$ $\ g\ =1.55 \times 10^{-7}$ $\epsilon=0.3$ $e_{abs}=0.021603$ $Cputime<1$ $NFE=20$
100	$J^*=0.07171019$ $m=2$ $\ g\ =5.29 \times 10^{-3}$ $\epsilon=0.02$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime<1$ $NFE=12$	$J^*=0.07171019$ $m=3$ $\ g\ =1.96 \times 10^{-6}$ $\epsilon=0.02$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime<1$ $NFE=16$	$J^*=0.07171019$ $m=4$ $\ g\ =8.47 \times 10^{-8}$ $\epsilon=0.02$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime<1$ $NFE=20$
500	$J^*=0.07402869$ $m=2$ $\ g\ =5.45 \times 10^{-3}$ $\epsilon=0.007$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime<1$ $NFE=12$	$J^*=0.07402869$ $m=7$ $\ g\ =2.90 \times 10^{-5}$ $\epsilon=0.002$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime<1$ $NFE=33$	$J^*=0.07402869$ $m=3$ $\ g\ =5.23 \times 10^{-7}$ $\epsilon=0.007$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime<1$ $NFE=16$

TABLE (7.4.9):Results of FR with change in u_0

u_0	m	J^*	$\ g\ $	e_{abs}	NFE	eps	Cputime
-0.6	700	0.07403014	2.42598×10^{-3}	5.99×10^{-4}	2804	0.004	3
-0.5	550	0.07403580	5.38615×10^{-3}	5.94×10^{-4}	2204	0.004	2
-0.4	300	0.07407081	1.31377×10^{-2}	5.59×10^{-4}	1204	0.004	2
-0.3	550	0.07403020	2.59041×10^{-3}	5.99×10^{-4}	2204	0.004	2
-0.2	300	0.07403763	6.04926×10^{-3}	5.92×10^{-4}	1204	0.008	2
-0.1	400	0.07403089	2.99524×10^{-3}	5.98×10^{-4}	1604	0.003	2
0.0	3	0.07402869	5.22767×10^{-7}	6.01×10^{-4}	16	0.007	<1

TABLE (7.4.10):Results of PR with varying ϵ and N.

N ϵ	10	50	100	200
0.003	J*=0.07428351 m=8 $\ g\ =8.42 \times 10^{-7}$ eabs= 2.57×10^{-4} Cputime<1 NFE=38	J*=0.07403623 m=5 $\ g\ =1.43 \times 10^{-7}$ eabs= 1.0×10^{-5} Cputime<1 NFE=24	J*=0.07402869 m=5 $\ g\ =1.71 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime<1 NFE=24	J*=0.07402869 m=4 $\ g\ =1.61 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime=1 NFE=20
0.006	J*=0.07428351 m=6 $\ g\ =8.15 \times 10^{-7}$ eabs= 2.57×10^{-4} Cputime<1 NFE=28	J*=0.07403623 m=4 $\ g\ =1.43 \times 10^{-7}$ eabs= 1.0×10^{-5} Cputime<1 NFE=20	J*=0.07402869 m=4 $\ g\ =1.61 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime<1 NFE=20	J*=0.07402869 m=3 $\ g\ =5.23 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime=1 NFE=20
0.007	J*=0.07428351 m=7 $\ g\ =8.35 \times 10^{-7}$ eabs= 2.57×10^{-4} Cputime<1 NFE=32	J*=0.07403623 m=4 $\ g\ =1.43 \times 10^{-7}$ eabs= 1.0×10^{-5} Cputime<1 NFE=20	J*=0.07402869 m=3 $\ g\ =5.23 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime<1 NFE=16	J*=0.07402869 m=3 $\ g\ =5.23 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime=1 NFE=16
0.008	J*=0.07428351 m=5 $\ g\ =8.40 \times 10^{-7}$ eabs= 2.57×10^{-4} Cputime<1 NFE=24	J*=0.07403623 m=4 $\ g\ =1.43 \times 10^{-7}$ eabs= 1.0×10^{-5} Cputime<1 NFE=20	J*=0.07402869 m=3 $\ g\ =5.23 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime<1 NFE=16	J*=0.07402869 m=3 $\ g\ =5.23 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime=1 NFE=16
0.009	J*=0.07428352 m=4 $\ g\ =8.19 \times 10^{-7}$ eabs= 2.57×10^{-4} Cputime<1 NFE=20	J*=0.07403623 m=4 $\ g\ =1.41 \times 10^{-7}$ eabs= 1.0×10^{-5} Cputime<1 NFE=20	J*=0.07402869 m=4 $\ g\ =1.62 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime<1 NFE=20	J*=0.07402869 m=4 $\ g\ =1.62 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime=1 NFE=20
0.01	J*=0.07428351 m=6 $\ g\ =8.15 \times 10^{-7}$ eabs= 2.57×10^{-4} Cputime<1 NFE=28	J*=0.07403623 m=4 $\ g\ =1.40 \times 10^{-7}$ eabs= 1.0×10^{-5} Cputime<1 NFE=20	J*=0.07402869 m=4 $\ g\ =1.62 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime<1 NFE=20	J*=0.07402869 m=4 $\ g\ =1.62 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime=1 NFE=20

TABLE (7.4.11):Results of PR with varying ACC and K.

ACC K	$\leq 10^{-2}$	$\leq 10^{-4}$	$\leq 10^{-6}$
10	$J^*=0.05302645$ $m=2$ $\ g\ =3.91 \times 10^{-3}$ $\epsilon=0.3$ $e_{abs}=0.0216030$ $Cputime<1$ $NFE=12$	$J^*=0.05302645$ $m=3$ $\ g\ =1.14 \times 10^{-5}$ $\epsilon=0.3$ $e_{abs}=0.0216030$ $Cputime<1$ $NFE=16$	$J^*=0.05302644$ $m=4$ $\ g\ =1.55 \times 10^{-7}$ $\epsilon=0.3$ $e_{abs}=0.021603$ $Cputime<1$ $NFE=20$
100	$J^*=0.07171019$ $m=2$ $\ g\ =5.29 \times 10^{-3}$ $\epsilon=0.02$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime<1$ $NFE=12$	$J^*=0.07171019$ $m=3$ $\ g\ =1.96 \times 10^{-6}$ $\epsilon=0.02$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime<1$ $NFE=16$	$J^*=0.07171019$ $m=4$ $\ g\ =8.47 \times 10^{-8}$ $\epsilon=0.02$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime<1$ $NFE=20$
500	$J^*=0.07402869$ $m=2$ $\ g\ =5.45 \times 10^{-3}$ $\epsilon=0.007$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime<1$ $NFE=12$	$J^*=0.07402869$ $m=7$ $\ g\ =2.90 \times 10^{-5}$ $\epsilon=0.002$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime<1$ $NFE=33$	$J^*=0.07402869$ $m=3$ $\ g\ =5.23 \times 10^{-7}$ $\epsilon=0.007$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime<1$ $NFE=16$

TABLE (7.4.12):Results of PR with change in u_0 .

u_0	m	J^*	$\ g\ $	e_{abs}	NFE	eps	Cputime
-0.6	700	0.07403014	2.42598×10^{-3}	5.99×10^{-4}	2804	0.004	3
-0.5	550	0.07403580	5.38615×10^{-3}	5.94×10^{-4}	2204	0.004	2
- 0.4	300	0.07407081	1.31377×10^{-2}	5.59×10^{-4}	1204	0.004	2
-0.3	550	0.07403020	2.59041×10^{-3}	5.99×10^{-4}	2204	0.004	2
-0.2	300	0.07403763	6.04926×10^{-3}	5.92×10^{-4}	1204	0.008	2
-0.1	400	0.07403089	2.99524×10^{-3}	5.98×10^{-4}	1604	0.003	2
0.0	3	0.07402869	5.22767×10^{-7}	6.01×10^{-4}	16	0.007	<1

TABLE (7.4.13):Results of H1 with varying ϵ and N.

N ϵ	10	50	100	200
0.003	$J^*=0.07428351$ $m=8$ $\ g\ =8.42 \times 10^{-7}$ $e_{abs}=2.57 \times 10^{-4}$ Cputime<1 NFE=38	$J^*=0.07403623$ $m=5$ $\ g\ =1.43 \times 10^{-7}$ $e_{abs}=1.0 \times 10^{-5}$ Cputime<1 NFE=24	$J^*=0.07402869$ $m=5$ $\ g\ =1.71 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime<1 NFE=24	$J^*=0.07402869$ $m=4$ $\ g\ =1.61 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime=1 NFE=20
0.006	$J^*=0.07428351$ $m=6$ $\ g\ =8.15 \times 10^{-7}$ $e_{abs}=2.57 \times 10^{-4}$ Cputime<1 NFE=28	$J^*=0.07403623$ $m=4$ $\ g\ =1.43 \times 10^{-7}$ $e_{abs}=1.0 \times 10^{-5}$ Cputime<1 NFE=20	$J^*=0.07402869$ $m=4$ $\ g\ =1.61 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime<1 NFE=20	$J^*=0.07402869$ $m=3$ $\ g\ =5.23 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime=1 NFE=20
0.007	$J^*=0.07428351$ $m=7$ $\ g\ =8.35 \times 10^{-7}$ $e_{abs}=2.57 \times 10^{-4}$ Cputime<1 NFE=32	$J^*=0.07403623$ $m=4$ $\ g\ =1.43 \times 10^{-7}$ $e_{abs}=1.0 \times 10^{-5}$ Cputime<1 NFE=20	$J^*=0.07402869$ $m=3$ $\ g\ =5.23 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime<1 NFE=16	$J^*=0.07402869$ $m=3$ $\ g\ =5.23 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime=1 NFE=16
0.008	$J^*=0.07428351$ $m=5$ $\ g\ =8.40 \times 10^{-7}$ $e_{abs}=2.57 \times 10^{-4}$ Cputime<1 NFE=24	$J^*=0.07403623$ $m=4$ $\ g\ =1.43 \times 10^{-7}$ $e_{abs}=1.0 \times 10^{-5}$ Cputime<1 NFE=20	$J^*=0.07402869$ $m=3$ $\ g\ =5.23 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime<1 NFE=16	$J^*=0.07402869$ $m=3$ $\ g\ =5.23 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime=1 NFE=16
0.009	$J^*=0.07428352$ $m=4$ $\ g\ =8.19 \times 10^{-7}$ $e_{abs}=2.57 \times 10^{-4}$ Cputime<1 NFE=20	$J^*=0.07403623$ $m=4$ $\ g\ =1.41 \times 10^{-7}$ $e_{abs}=1.0 \times 10^{-5}$ Cputime<1 NFE=20	$J^*=0.07402869$ $m=4$ $\ g\ =1.62 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime<1 NFE=20	$J^*=0.07402869$ $m=4$ $\ g\ =1.62 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime=1 NFE=20
0.01	$J^*=0.07428351$ $m=6$ $\ g\ =8.15 \times 10^{-7}$ $e_{abs}=2.57 \times 10^{-4}$ Cputime<1 NFE=28	$J^*=0.07403623$ $m=4$ $\ g\ =1.40 \times 10^{-7}$ $e_{abs}=1.0 \times 10^{-5}$ Cputime<1 NFE=20	$J^*=0.07402869$ $m=4$ $\ g\ =1.62 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime<1 NFE=20	$J^*=0.07402869$ $m=4$ $\ g\ =1.62 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime=1 NFE=20

TABLE (7.4.14):Results of H1 with varying ACC and K.

ACC K	$\leq 10^{-2}$	$\leq 10^{-4}$	$\leq 10^{-6}$
10	$J^*=0.05302645$ $m=2$ $\ g\ =3.91 \times 10^{-3}$ $\epsilon=0.3$ $e_{abs}=0.0216030$ $Cputime<1$ $NFE=12$	$J^*=0.05302645$ $m=3$ $\ g\ =1.14 \times 10^{-5}$ $\epsilon=0.3$ $e_{abs}=0.0216030$ $Cputime<1$ $NFE=16$	$J^*=0.05302644$ $m=4$ $\ g\ =1.55 \times 10^{-7}$ $\epsilon=0.3$ $e_{abs}=0.021603$ $Cputime<1$ $NFE=20$
100	$J^*=0.07171019$ $m=2$ $\ g\ =5.29 \times 10^{-3}$ $\epsilon=0.02$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime<1$ $NFE=12$	$J^*=0.07171019$ $m=3$ $\ g\ =1.96 \times 10^{-6}$ $\epsilon=0.02$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime<1$ $NFE=16$	$J^*=0.07171019$ $m=4$ $\ g\ =8.47 \times 10^{-8}$ $\epsilon=0.02$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime<1$ $NFE=20$
500	$J^*=0.07402869$ $m=2$ $\ g\ =5.45 \times 10^{-3}$ $\epsilon=0.007$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime<1$ $NFE=12$	$J^*=0.07402869$ $m=7$ $\ g\ =2.90 \times 10^{-5}$ $\epsilon=0.002$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime<1$ $NFE=33$	$J^*=0.07402869$ $m=3$ $\ g\ =5.23 \times 10^{-7}$ $\epsilon=0.007$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime<1$ $NFE=16$

TABLE (7.4.15):Results of H1 with change in u_0 .

u_0	m	J^*	$\ g\ $	e_{abs}	NFE	eps	Cputime
-0.6	700	0.07403014	2.42598×10^{-3}	5.99×10^{-4}	2804	0.004	3
-0.5	550	0.07403580	5.38615×10^{-3}	5.94×10^{-4}	2204	0.004	2
-0.4	300	0.07407081	1.31377×10^{-2}	5.59×10^{-4}	1204	0.004	2
-0.3	550	0.07403020	2.59041×10^{-3}	5.99×10^{-4}	2204	0.004	2
-0.2	300	0.07403763	6.04926×10^{-3}	5.92×10^{-4}	1204	0.008	2
-0.1	400	0.07403089	2.99524×10^{-3}	5.98×10^{-4}	1604	0.003	2
0.0	3	0.07402869	5.22767×10^{-7}	6.01×10^{-4}	16	0.007	<1

TABLE (7.4.16):Results of ATH with varying ϵ and N.

N ϵ	10	50	100	200
0.003	J*=0.07428351 m=8 $\ g\ =8.42 \times 10^{-7}$ eabs= 2.57×10^{-4} Cputime<1 NFE=38	J*=0.07403623 m=5 $\ g\ =1.43 \times 10^{-7}$ eabs= 1.0×10^{-5} Cputime<1 NFE=24	J*=0.07402869 m=5 $\ g\ =1.71 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime<1 NFE=24	J*=0.07402869 m=4 $\ g\ =1.61 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime=1 NFE=20
0.006	J*=0.07428351 m=6 $\ g\ =8.15 \times 10^{-7}$ eabs= 2.57×10^{-4} Cputime<1 NFE=28	J*=0.07403623 m=4 $\ g\ =1.43 \times 10^{-7}$ eabs= 1.0×10^{-5} Cputime<1 NFE=20	J*=0.07402869 m=4 $\ g\ =1.61 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime<1 NFE=20	J*=0.07402869 m=3 $\ g\ =5.23 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime=1 NFE=20
0.007	J*=0.07428351 m=7 $\ g\ =8.35 \times 10^{-7}$ eabs= 2.57×10^{-4} Cputime<1 NFE=32	J*=0.07403623 m=4 $\ g\ =1.43 \times 10^{-7}$ eabs= 1.0×10^{-5} Cputime<1 NFE=20	J*=0.07402869 m=3 $\ g\ =5.23 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime<1 NFE=16	J*=0.07402869 m=3 $\ g\ =5.23 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime=1 NFE=16
0.008	J*=0.07428351 m=5 $\ g\ =8.40 \times 10^{-7}$ eabs= 2.57×10^{-4} Cputime<1 NFE=24	J*=0.07403623 m=4 $\ g\ =1.43 \times 10^{-7}$ eabs= 1.0×10^{-5} Cputime<1 NFE=20	J*=0.07402869 m=3 $\ g\ =5.23 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime<1 NFE=16	J*=0.07402869 m=3 $\ g\ =5.23 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime=1 NFE=16
0.009	J*=0.07428352 m=4 $\ g\ =8.19 \times 10^{-7}$ eabs= 2.57×10^{-4} Cputime<1 NFE=20	J*=0.07403623 m=4 $\ g\ =1.41 \times 10^{-7}$ eabs= 1.0×10^{-5} Cputime<1 NFE=20	J*=0.07402869 m=4 $\ g\ =1.62 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime<1 NFE=20	J*=0.07402869 m=4 $\ g\ =1.62 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime=1 NFE=20
0.01	J*=0.07428351 m=6 $\ g\ =8.15 \times 10^{-7}$ eabs= 2.57×10^{-4} Cputime<1 NFE=28	J*=0.07403623 m=4 $\ g\ =1.40 \times 10^{-7}$ eabs= 1.0×10^{-5} Cputime<1 NFE=20	J*=0.07402869 m=4 $\ g\ =1.62 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime<1 NFE=20	J*=0.07402869 m=4 $\ g\ =1.62 \times 10^{-7}$ eabs= 2.51×10^{-6} Cputime=1 NFE=20

TABLE (7.4.17):Results of ATH with varying ACC and K.

ACC K	$\leq 10^{-2}$	$\leq 10^{-4}$	$\leq 10^{-6}$
10	$J^*=0.05302645$ $m=2$ $\ g\ =3.91 \times 10^{-3}$ $\epsilon=0.3$ $e_{abs}=0.0216030$ $Cputime<1$ $NFE=12$	$J^*=0.05302645$ $m=3$ $\ g\ =1.14 \times 10^{-5}$ $\epsilon=0.3$ $e_{abs}=0.0216030$ $Cputime<1$ $NFE=16$	$J^*=0.05302644$ $m=4$ $\ g\ =1.55 \times 10^{-7}$ $\epsilon=0.3$ $e_{abs}=0.021603$ $Cputime<1$ $NFE=20$
100	$J^*=0.07171019$ $m=2$ $\ g\ =5.29 \times 10^{-3}$ $\epsilon=0.02$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime<1$ $NFE=12$	$J^*=0.07171019$ $m=3$ $\ g\ =1.96 \times 10^{-6}$ $\epsilon=0.02$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime<1$ $NFE=16$	$J^*=0.07171019$ $m=4$ $\ g\ =8.47 \times 10^{-8}$ $\epsilon=0.02$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime<1$ $NFE=20$
500	$J^*=0.07402869$ $m=2$ $\ g\ =5.45 \times 10^{-3}$ $\epsilon=0.007$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime<1$ $NFE=12$	$J^*=0.07402869$ $m=7$ $\ g\ =2.90 \times 10^{-5}$ $\epsilon=0.002$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime<1$ $NFE=33$	$J^*=0.07402869$ $m=3$ $\ g\ =5.23 \times 10^{-7}$ $\epsilon=0.007$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime<1$ $NFE=16$

TABLE (7.4.18):Results of ATH with change in u_0 .

u_0	m	j^*	$\ g\ $	e_{abs}	NFE	eps	Cputime
-0.6	700	0.07403014	2.42598×10^{-3}	5.99×10^{-4}	2804	0.004	3
-0.5	550	0.07403580	5.38615×10^{-3}	5.94×10^{-4}	2204	0.004	2
- 0.4	300	0.07407081	1.31377×10^{-2}	5.59×10^{-4}	1204	0.004	2
-0.3	550	0.07403020	2.59041×10^{-3}	5.99×10^{-4}	2204	0.004	2
-0.2	300	0.07403763	6.04926×10^{-3}	5.92×10^{-4}	1204	0.008	2
-0.1	400	0.07403089	2.99524×10^{-3}	5.98×10^{-4}	1604	0.003	2
0.0	3	0.07402869	5.22767×10^{-7}	6.01×10^{-4}	16	0.007	<1

TABLE (7.4.19):Results of H3 with varying ϵ and N.

N ϵ	10	50	100	200
0.003	$J^*=0.07428351$ $m=8$ $\ g\ =8.42 \times 10^{-7}$ $e_{abs}=2.57 \times 10^{-4}$ Cputime<1 NFE=38	$J^*=0.07403623$ $m=5$ $\ g\ =1.43 \times 10^{-7}$ $e_{abs}=1.0 \times 10^{-5}$ Cputime<1 NFE=24	$J^*=0.07402869$ $m=5$ $\ g\ =1.71 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime<1 NFE=24	$J^*=0.07402869$ $m=4$ $\ g\ =1.61 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime=1 NFE=20
0.006	$J^*=0.07428351$ $m=6$ $\ g\ =8.15 \times 10^{-7}$ $e_{abs}=2.57 \times 10^{-4}$ Cputime<1 NFE=28	$J^*=0.07403623$ $m=4$ $\ g\ =1.43 \times 10^{-7}$ $e_{abs}=1.0 \times 10^{-5}$ Cputime<1 NFE=20	$J^*=0.07402869$ $m=4$ $\ g\ =1.61 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime<1 NFE=20	$J^*=0.07402869$ $m=3$ $\ g\ =5.23 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime=1 NFE=20
0.007	$J^*=0.07428351$ $m=7$ $\ g\ =8.35 \times 10^{-7}$ $e_{abs}=2.57 \times 10^{-4}$ Cputime<1 NFE=32	$J^*=0.07403623$ $m=4$ $\ g\ =1.43 \times 10^{-7}$ $e_{abs}=1.0 \times 10^{-5}$ Cputime<1 NFE=20	$J^*=0.07402869$ $m=3$ $\ g\ =5.23 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime<1 NFE=16	$J^*=0.07402869$ $m=3$ $\ g\ =5.23 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime=1 NFE=16
0.008	$J^*=0.07428351$ $m=5$ $\ g\ =8.40 \times 10^{-7}$ $e_{abs}=2.57 \times 10^{-4}$ Cputime<1 NFE=24	$J^*=0.07403623$ $m=4$ $\ g\ =1.43 \times 10^{-7}$ $e_{abs}=1.0 \times 10^{-5}$ Cputime<1 NFE=20	$J^*=0.07402869$ $m=3$ $\ g\ =5.23 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime<1 NFE=16	$J^*=0.07402869$ $m=3$ $\ g\ =5.23 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime=1 NFE=16
0.009	$J^*=0.07428352$ $m=4$ $\ g\ =8.19 \times 10^{-7}$ $e_{abs}=2.57 \times 10^{-4}$ Cputime<1 NFE=20	$J^*=0.07403623$ $m=4$ $\ g\ =1.41 \times 10^{-7}$ $e_{abs}=1.0 \times 10^{-5}$ Cputime<1 NFE=20	$J^*=0.07402869$ $m=4$ $\ g\ =1.62 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime<1 NFE=20	$J^*=0.07402869$ $m=4$ $\ g\ =1.62 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime=1 NFE=20
0.01	$J^*=0.07428351$ $m=6$ $\ g\ =8.15 \times 10^{-7}$ $e_{abs}=2.57 \times 10^{-4}$ Cputime<1 NFE=28	$J^*=0.07403623$ $m=4$ $\ g\ =1.40 \times 10^{-7}$ $e_{abs}=1.0 \times 10^{-5}$ Cputime<1 NFE=20	$J^*=0.07402869$ $m=4$ $\ g\ =1.62 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime<1 NFE=20	$J^*=0.07402869$ $m=4$ $\ g\ =1.62 \times 10^{-7}$ $e_{abs}=2.51 \times 10^{-6}$ Cputime=1 NFE=20

TABLE (7.4.20):Results of H3 with varying ACC and K.

ACC K	$\leq 10^{-2}$	$\leq 10^{-4}$	$\leq 10^{-6}$
10	$J^*=0.05302645$ $m=2$ $\ g\ =3.91 \times 10^{-3}$ $\epsilon=0.3$ $e_{abs}=0.0216030$ $Cputime<1$ $NFE=12$	$J^*=0.05302645$ $m=3$ $\ g\ =1.14 \times 10^{-5}$ $\epsilon=0.3$ $e_{abs}=0.0216030$ $Cputime<1$ $NFE=16$	$J^*=0.05302644$ $m=4$ $\ g\ =1.55 \times 10^{-7}$ $\epsilon=0.3$ $e_{abs}=0.021603$ $Cputime<1$ $NFE=20$
100	$J^*=0.07171019$ $m=2$ $\ g\ =5.29 \times 10^{-3}$ $\epsilon=0.02$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime<1$ $NFE=12$	$J^*=0.07171019$ $m=3$ $\ g\ =1.96 \times 10^{-6}$ $\epsilon=0.02$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime<1$ $NFE=16$	$J^*=0.07171019$ $m=4$ $\ g\ =8.47 \times 10^{-8}$ $\epsilon=0.02$ $e_{abs}=2.92 \times 10^{-3}$ $Cputime<1$ $NFE=20$
500	$J^*=0.07402869$ $m=2$ $\ g\ =5.45 \times 10^{-3}$ $\epsilon=0.007$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime<1$ $NFE=12$	$J^*=0.07402869$ $m=7$ $\ g\ =2.90 \times 10^{-5}$ $\epsilon=0.002$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime<1$ $NFE=33$	$J^*=0.07402869$ $m=3$ $\ g\ =5.23 \times 10^{-7}$ $\epsilon=0.007$ $e_{abs}=6.01 \times 10^{-4}$ $Cputime<1$ $NFE=16$

TABLE (7.4.21):Results of H3 with change in u_0 .

u_0	m	J^*	$\ g\ $	eabs	NFE	eps	Cputime
-0.6	700	0.07403014	2.42598×10^{-3}	5.99×10^{-4}	2804	0.004	3
-0.5	550	0.07403580	5.38615×10^{-3}	5.94×10^{-4}	2204	0.004	2
- 0.4	300	0.07407081	1.31377×10^{-2}	5.59×10^{-4}	1204	0.004	2
-0.3	550	0.07403020	2.59041×10^{-3}	5.99×10^{-4}	2204	0.004	2
-0.2	300	0.07403763	6.04926×10^{-3}	5.92×10^{-4}	1204	0.008	2
-0.1	400	0.07403089	2.99524×10^{-3}	5.98×10^{-4}	1604	0.003	2
0.0	3	0.07402869	5.22767×10^{-7}	6.01×10^{-4}	16	0.007	<1

$\frac{\text{method}}{u_0}$	GFS	SD	FR	PR	H1	ATH	H3
-0.6	$J^* = 0.07402869$ $m = 891$ $ g = 8.00283 \times 10^{-7}$ $e_{abs} = 6.01 \times 10^{-4}$ $\varepsilon = 0.008$ NFE = 892 Cputime = 1	$J^* = 0.07403014$ $m = 700$ $ g = 2.42598 \times 10^{-3}$ $e_{abs} = 5.99 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 2804 Cputime = 3	$J^* = 0.07403014$ $m = 700$ $ g = 2.42598 \times 10^{-3}$ $e_{abs} = 5.99 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 2804 Cputime = 3	$J^* = 0.07403014$ $m = 700$ $ g = 2.42598 \times 10^{-3}$ $e_{abs} = 5.99 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 2804 Cputime = 3	$J^* = 0.07403014$ $m = 700$ $ g = 2.42598 \times 10^{-3}$ $e_{abs} = 5.99 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 2804 Cputime = 3	$J^* = 0.07403014$ $m = 700$ $ g = 2.42598 \times 10^{-3}$ $e_{abs} = 5.99 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 2804 Cputime = 3	$J^* = 0.07403014$ $m = 700$ $ g = 2.42598 \times 10^{-3}$ $e_{abs} = 5.99 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 2804 Cputime = 3
-0.5	$J^* = 0.07402869$ $m = 912$ $ g = 91989 \times 10^{-7}$ $e_{abs} = 6.01 \times 10^{-4}$ $\varepsilon = 0.008$ NFE = 918 Cputime = 1	$J^* = 0.07403580$ $m = 550$ $ g = 5.38615 \times 10^{-3}$ $e_{abs} = 5.94 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 2204 Cputime = 2	$J^* = 0.07403580$ $m = 550$ $ g = 5.38615 \times 10^{-3}$ $e_{abs} = 5.94 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 2204 Cputime = 2	$J^* = 0.07403580$ $m = 550$ $ g = 5.38615 \times 10^{-3}$ $e_{abs} = 5.94 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 2204 Cputime = 2	$J^* = 0.07403580$ $m = 550$ $ g = 5.38615 \times 10^{-3}$ $e_{abs} = 5.94 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 2204 Cputime = 2	$J^* = 0.07403580$ $m = 550$ $ g = 5.38615 \times 10^{-3}$ $e_{abs} = 5.94 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 2204 Cputime = 2	$J^* = 0.07403580$ $m = 550$ $ g = 5.386 \times 10^{-3}$ $e_{abs} = 5.94 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 2204 Cputime = 2
-0.4	$J^* = 0.07402869$ $m = 810$ $ g = 6.27786 \times 10^{-7}$ $e_{abs} = 6.01 \times 10^{-4}$ $\varepsilon = 0.008$ NFE = 811 Cputime = 1	$J^* = 0.07407081$ $m = 300$ $ g = 1.31377 \times 10^{-2}$ $e_{abs} = 5.59 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 1204 Cputime = 2	$J^* = 0.07407081$ $m = 300$ $ g = 1.31377 \times 10^{-2}$ $e_{abs} = 5.59 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 1204 Cputime = 2	$J^* = 0.07407081$ $m = 300$ $ g = 1.31377 \times 10^{-2}$ $e_{abs} = 5.59 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 1208 Cputime = 2	$J^* = 0.07407081$ $m = 300$ $ g = 1.31377 \times 10^{-2}$ $e_{abs} = 5.59 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 1204 Cputime = 2	$J^* = 0.07407081$ $m = 300$ $ g = 1.31377 \times 10^{-2}$ $e_{abs} = 5.59 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 1204 Cputime = 2	$J^* = 0.074037081$ $m = 300$ $ g = 1.31377 \times 10^{-2}$ $e_{abs} = 5.59 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 1204 Cputime = 2
-0.3	$J^* = 0.07402869$ $m = 845$ $ g = 3.64625 \times 10^{-7}$ $e_{abs} = 6.01 \times 10^{-4}$ $\varepsilon = 0.008$ NFE = 846 Cputime = 1	$J^* = 0.07403020$ $m = 550$ $ g = 2.59041 \times 10^{-3}$ $e_{abs} = 5.99 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 2204 Cputime = 2	$J^* = 0.07403020$ $m = 550$ $ g = 2.59041 \times 10^{-3}$ $e_{abs} = 5.99 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 2204 Cputime = 2	$J^* = 0.07403020$ $m = 550$ $ g = 2.59041 \times 10^{-3}$ $e_{abs} = 5.99 \times 10^{-4}$ $\varepsilon = 0.006$ NFE = 2204 Cputime = 2	$J^* = 0.07403020$ $m = 550$ $ g = 2.59041 \times 10^{-3}$ $e_{abs} = 5.99 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 2204 Cputime = 2	$J^* = 0.07403020$ $m = 550$ $ g = 2.59041 \times 10^{-3}$ $e_{abs} = 5.99 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 2204 Cputime = 2	$J^* = 0.07403020$ $m = 550$ $ g = 2.59041 \times 10^{-3}$ $e_{abs} = 5.99 \times 10^{-4}$ $\varepsilon = 0.004$ NFE = 2204 Cputime = 2
-0.2	$J^* = 0.07402869$ $m = 817$ $ g = 3.64625 \times 10^{-7}$ $e_{abs} = 6.01 \times 10^{-4}$ $\varepsilon = 0.008$ NFE = 818 Cputime = 1	$J^* = 0.07403763$ $m = 300$ $ g = 6.04926 \times 10^{-3}$ $e_{abs} = 5.92 \times 10^{-4}$ $\varepsilon = 0.008$ NFE = 1204 Cputime =					

OPTIMAL J WITH DIFFERENT K'S

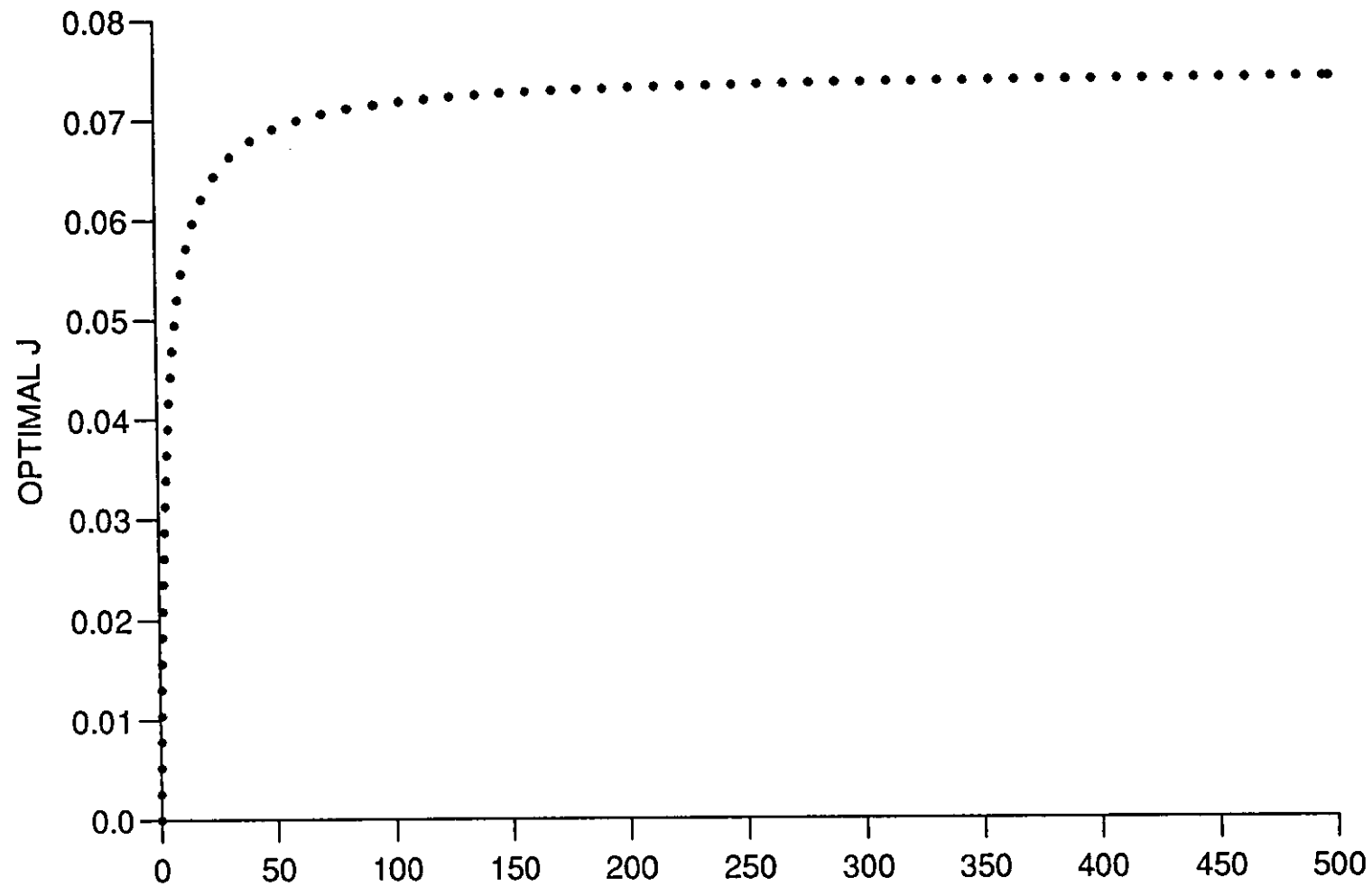


Figure (7.2.1)

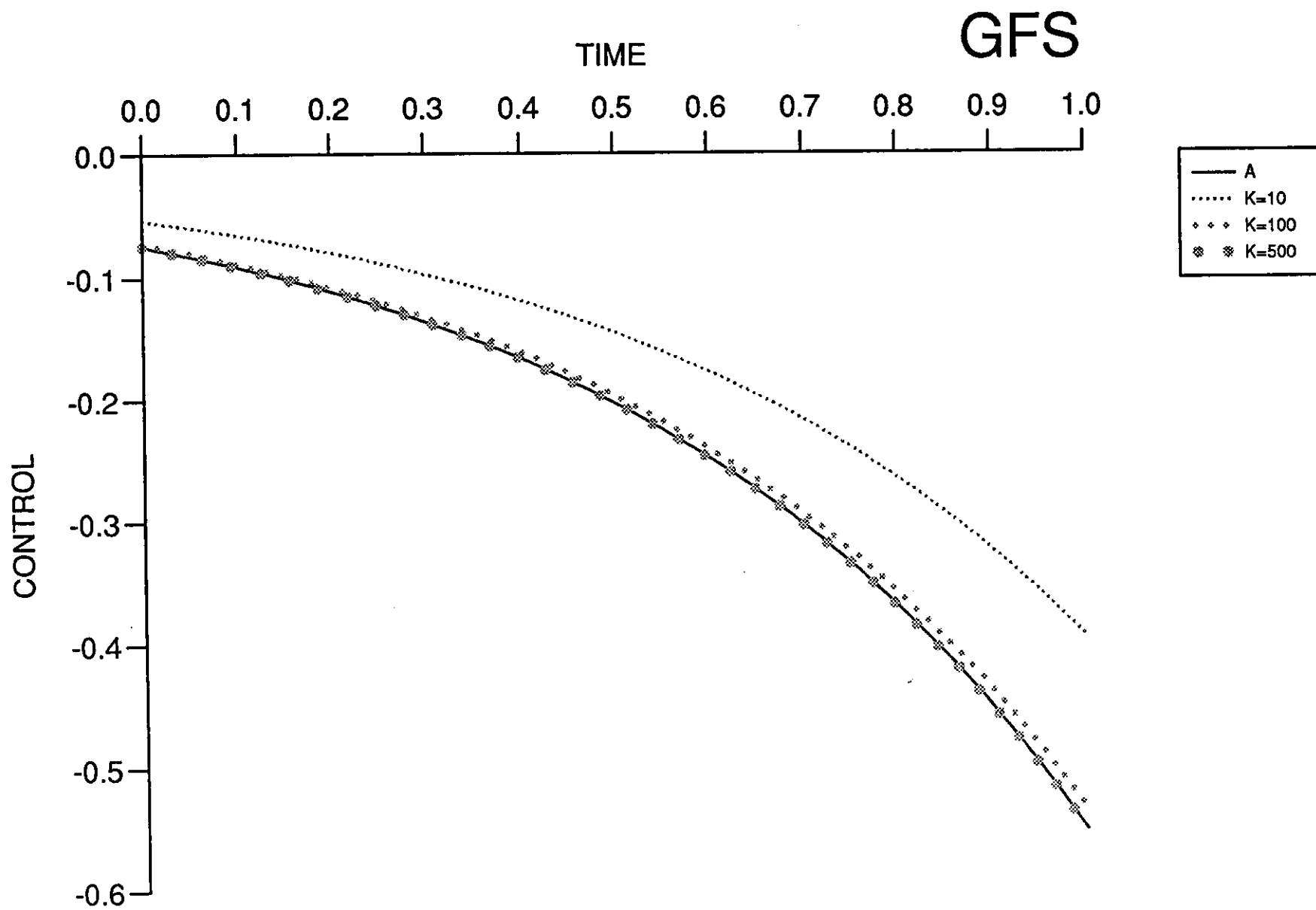


Figure (7.4.1)

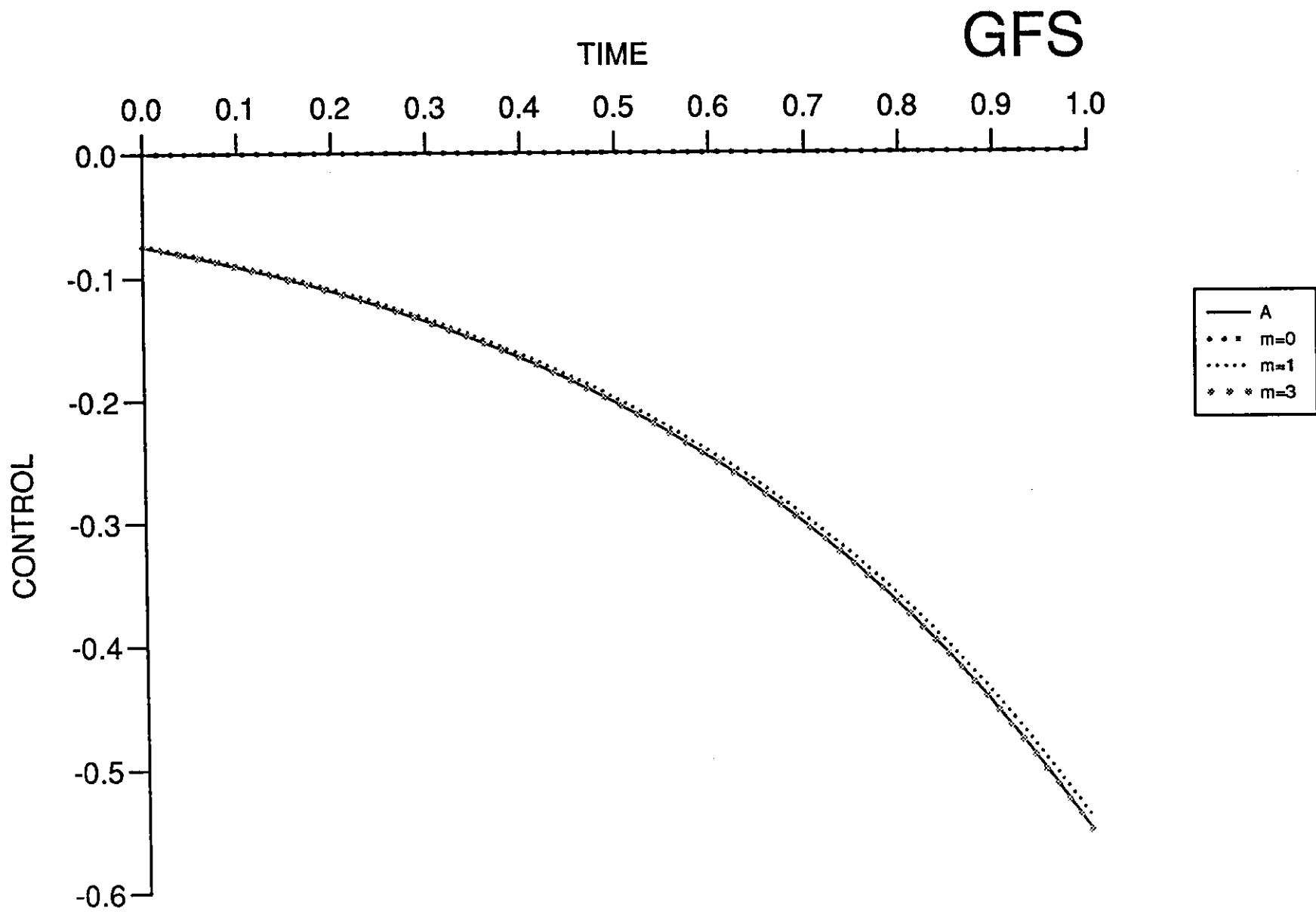


Figure (7.4.2)

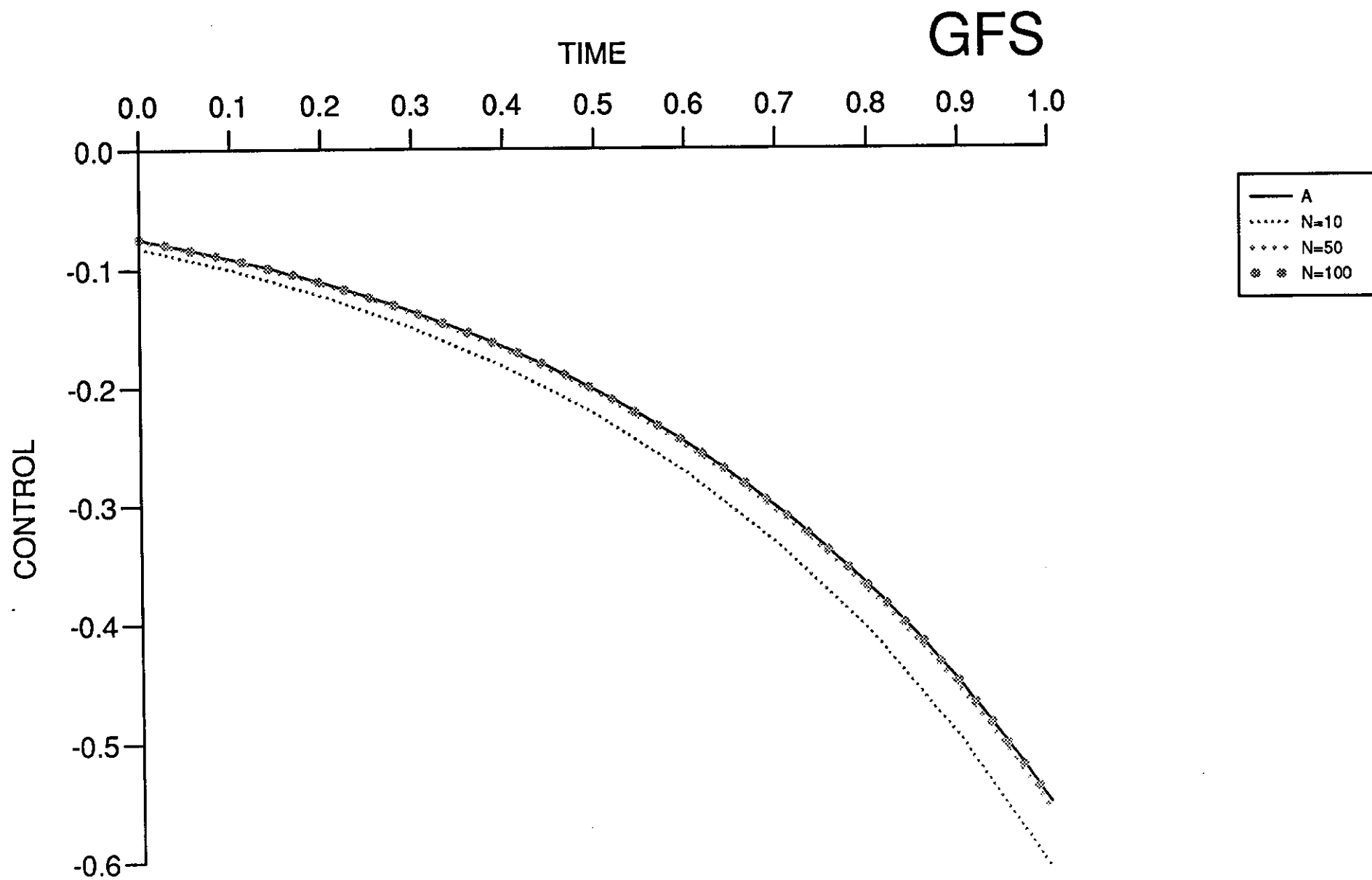


Figure (7.4.3)

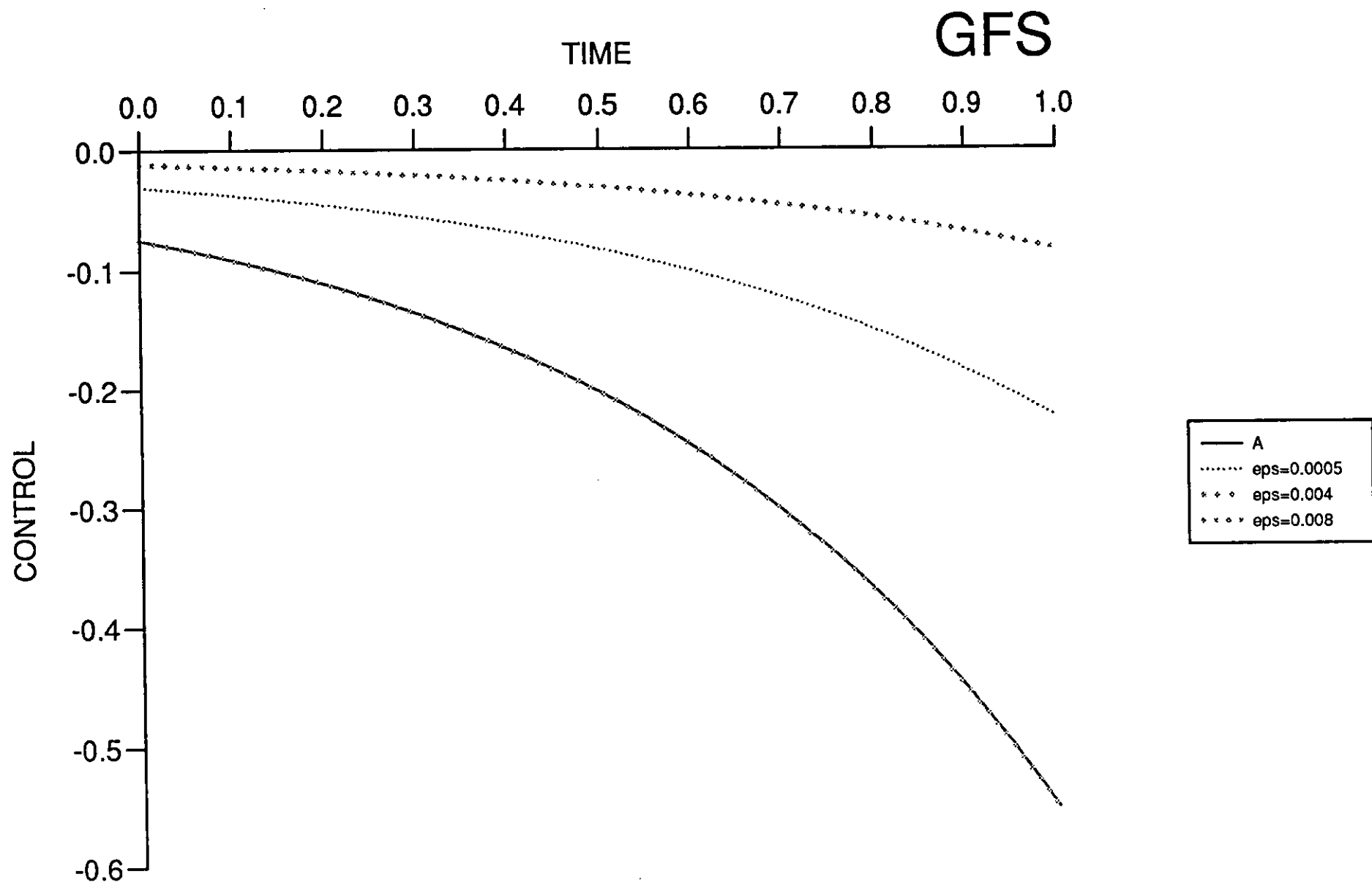


Figure (7.4.4)

GFS

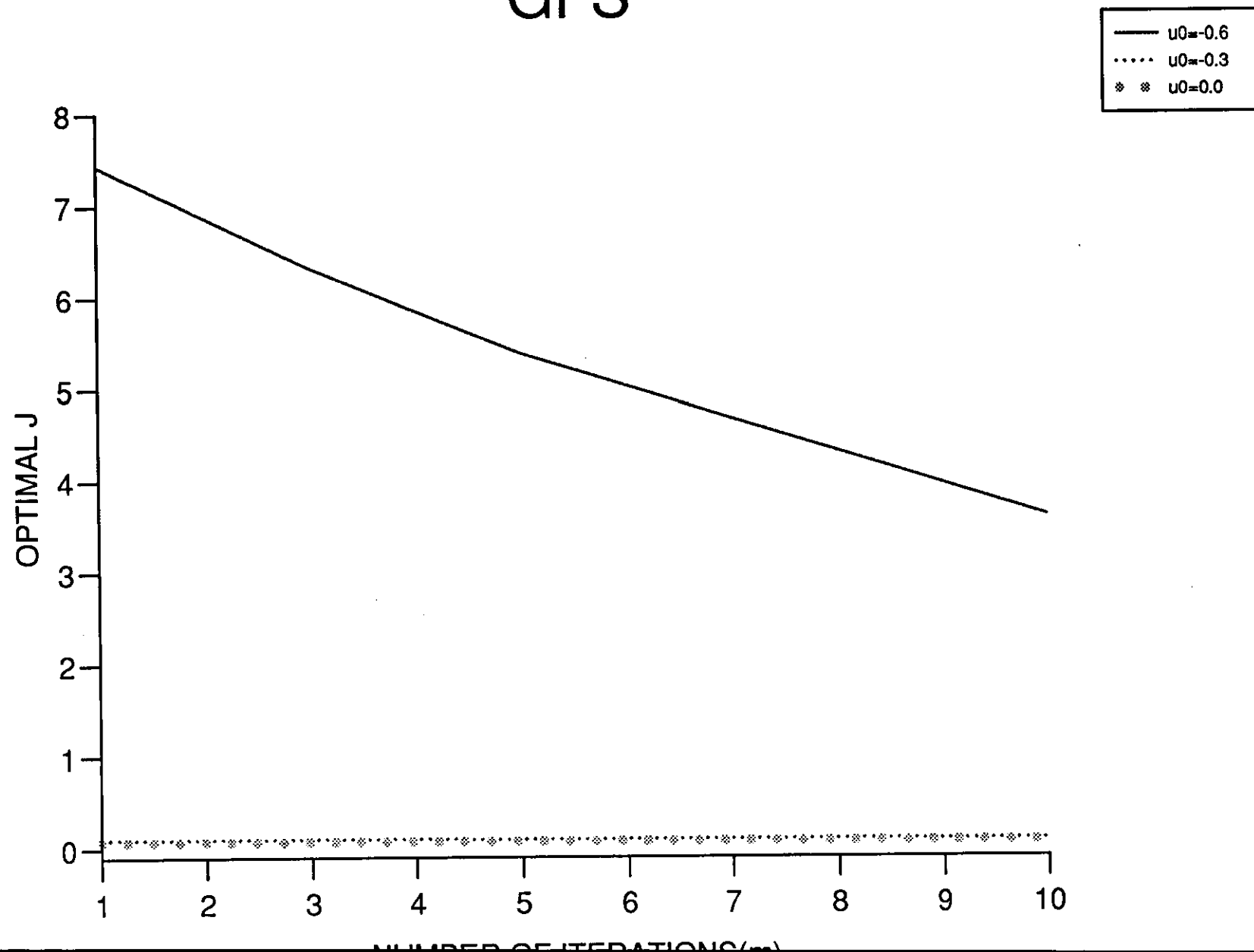


Figure (7.4.5)

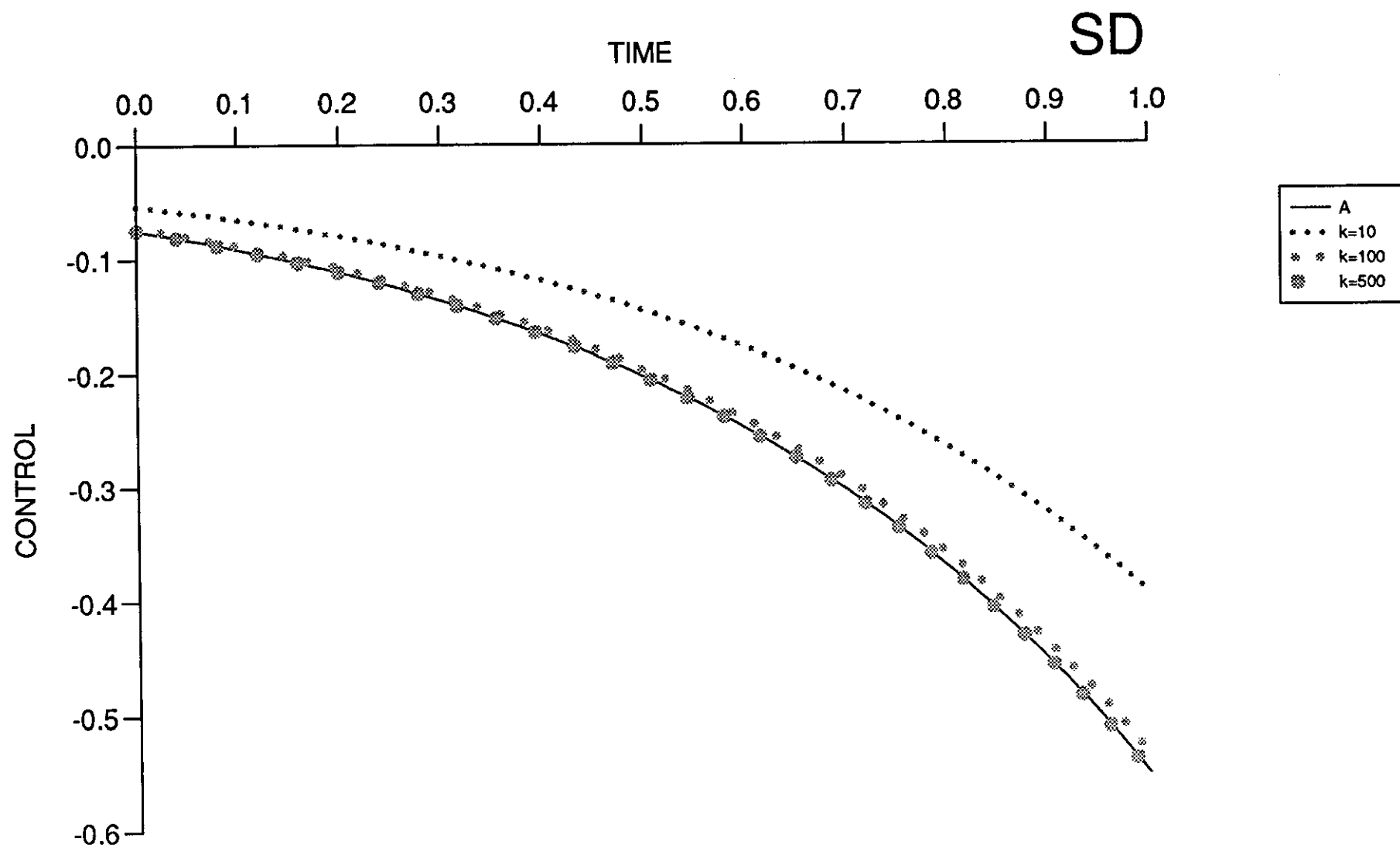


Figure (7.4.6)

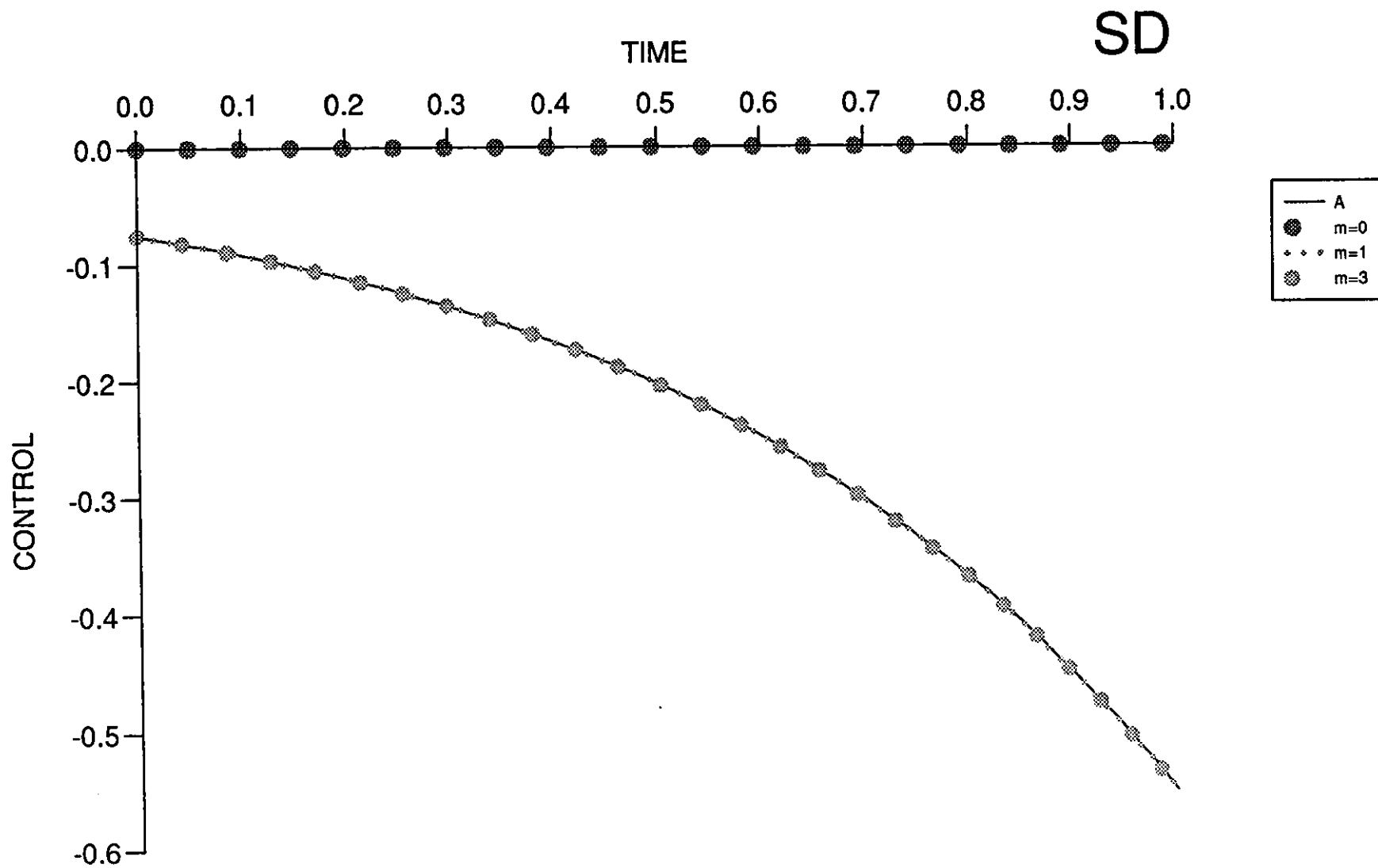


Figure (7.4.7).

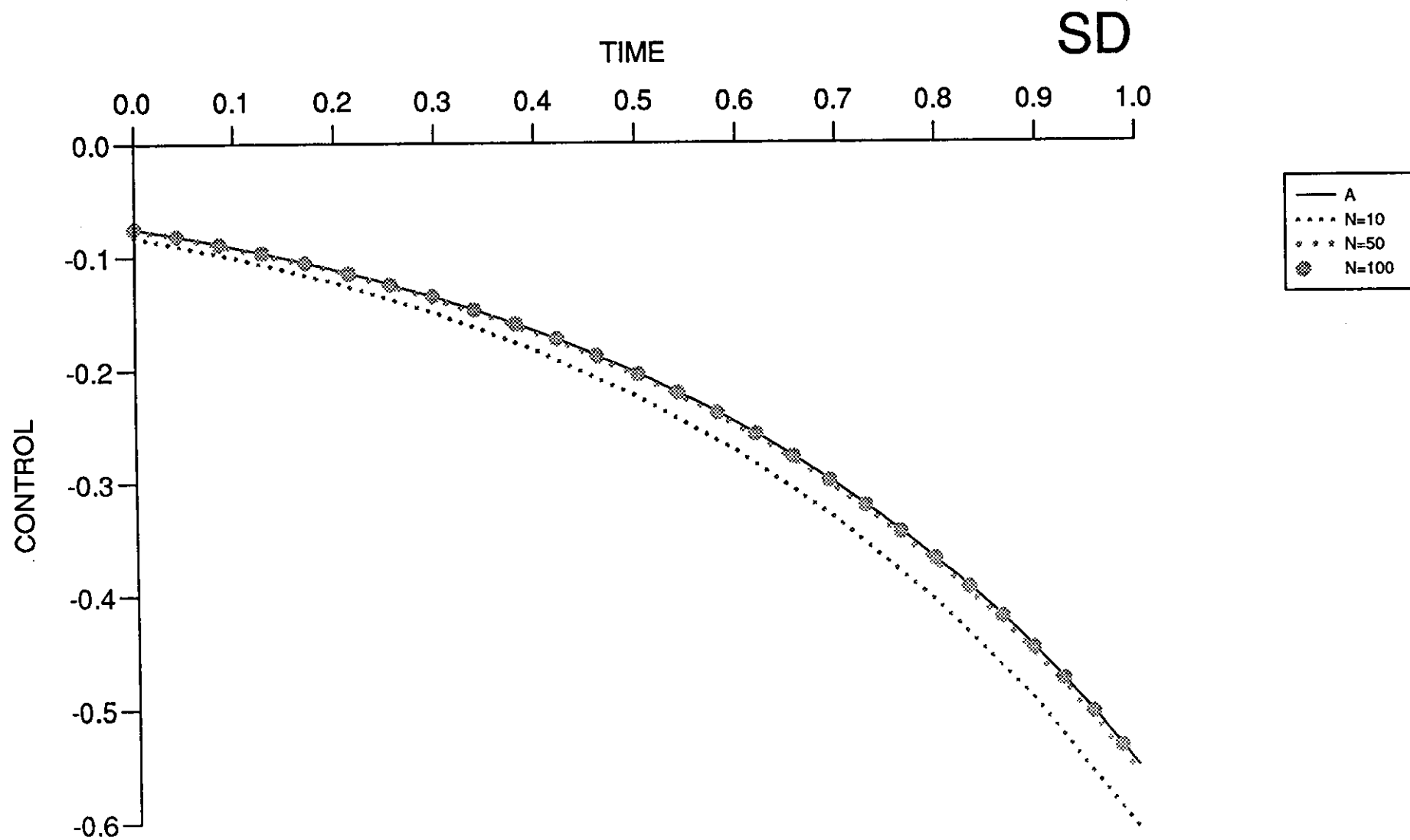


Figure (7.4.8)

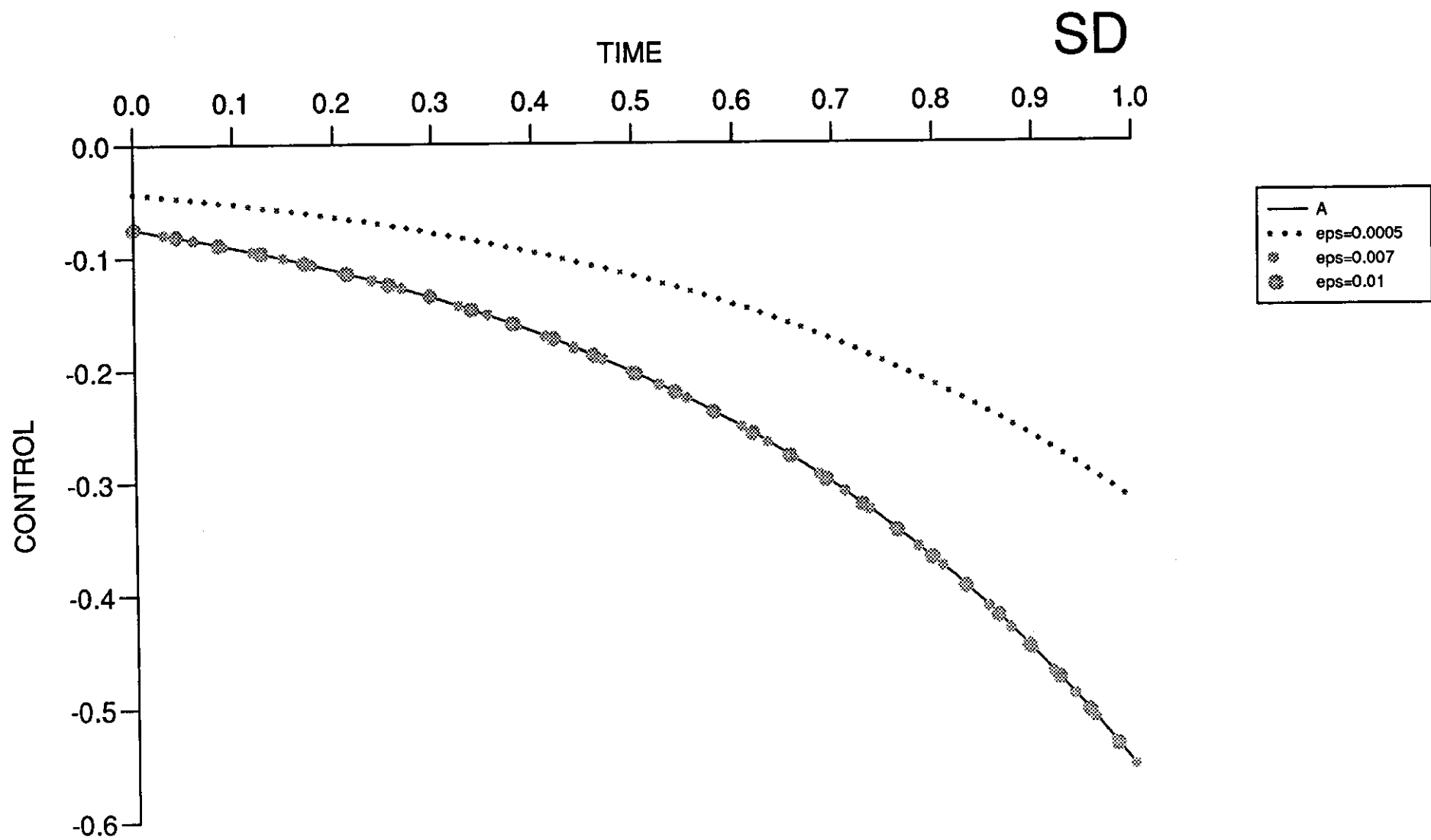


Figure (7.4.9)

SD

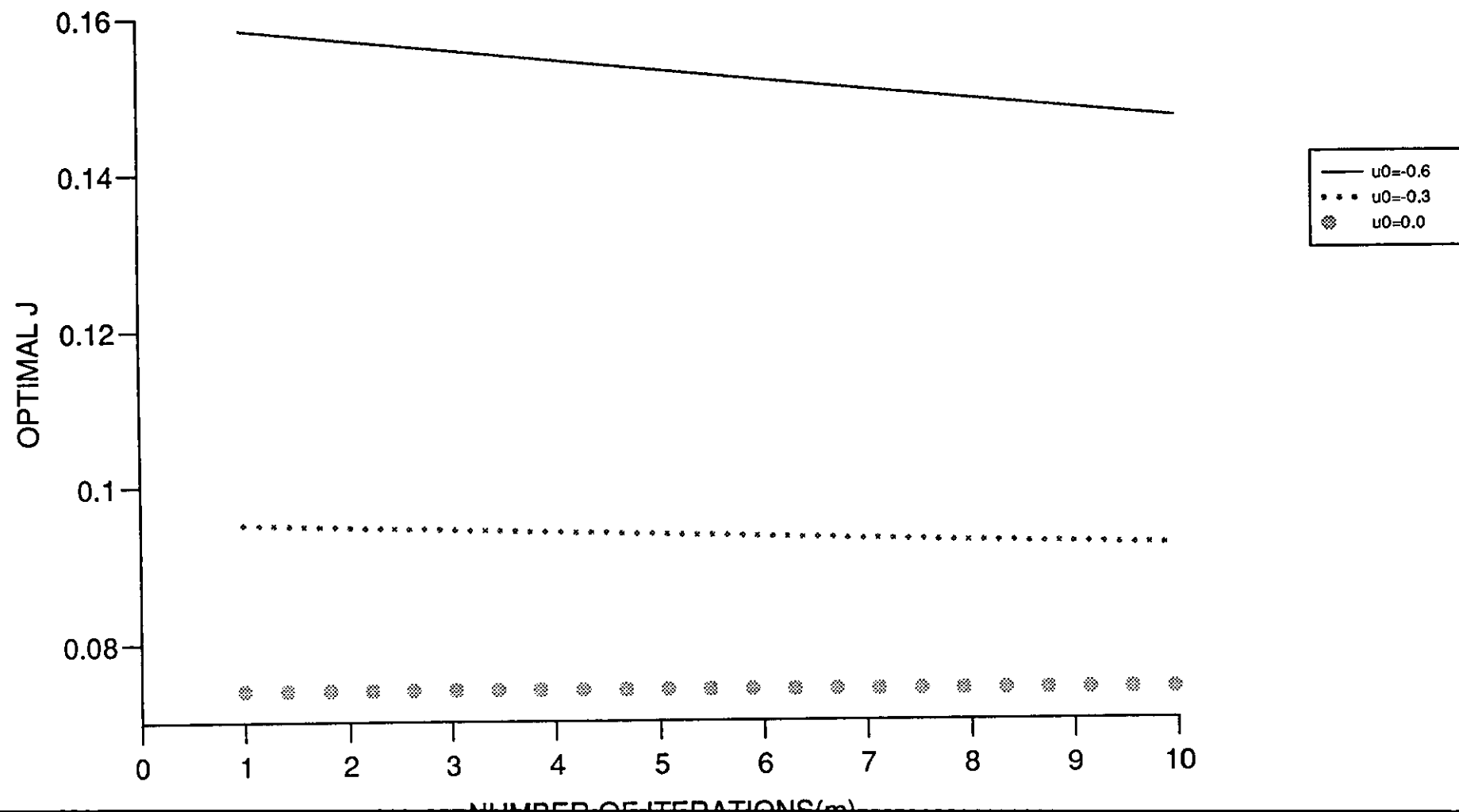


Figure (7.4.10)

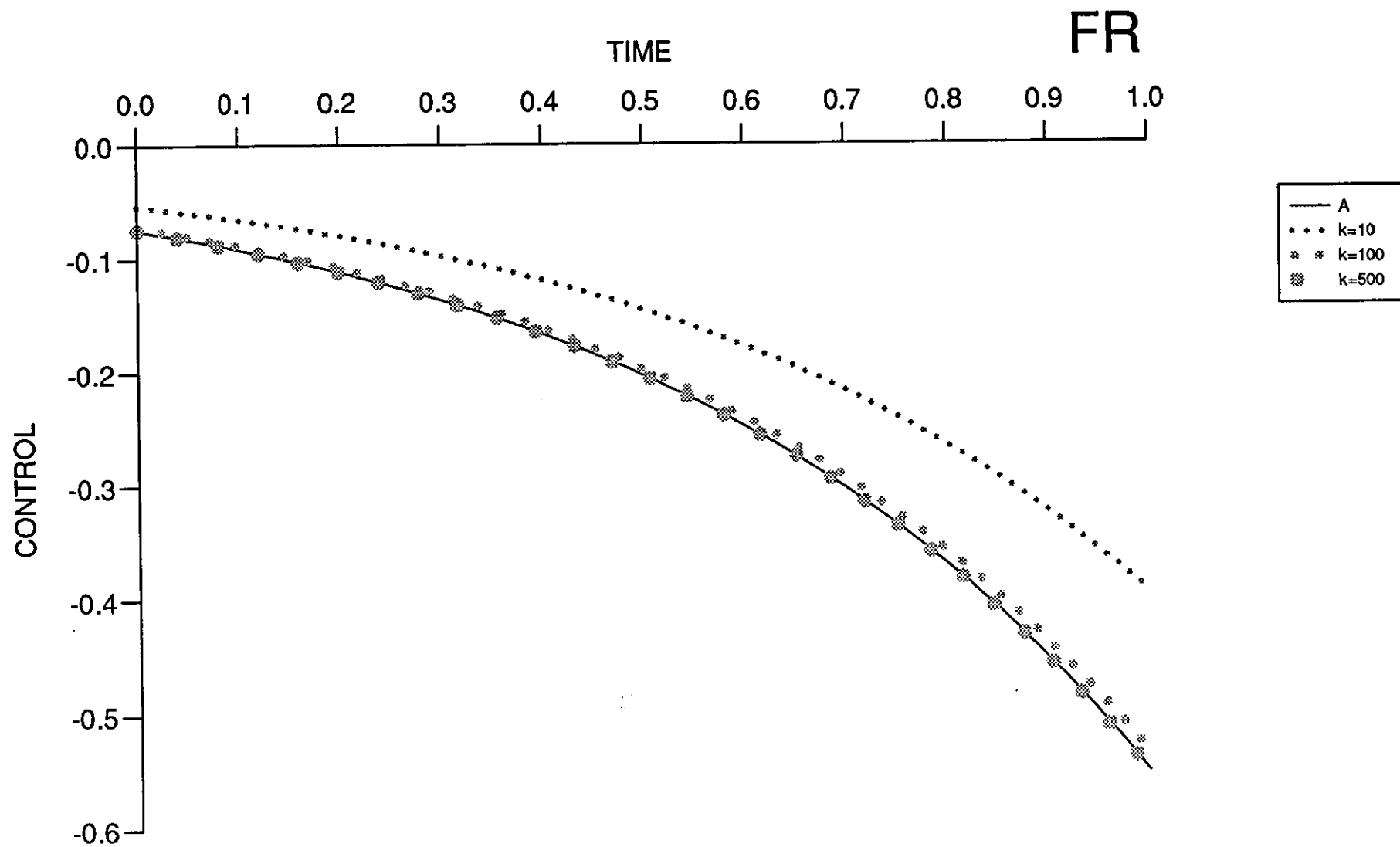


Figure (7.4.11)

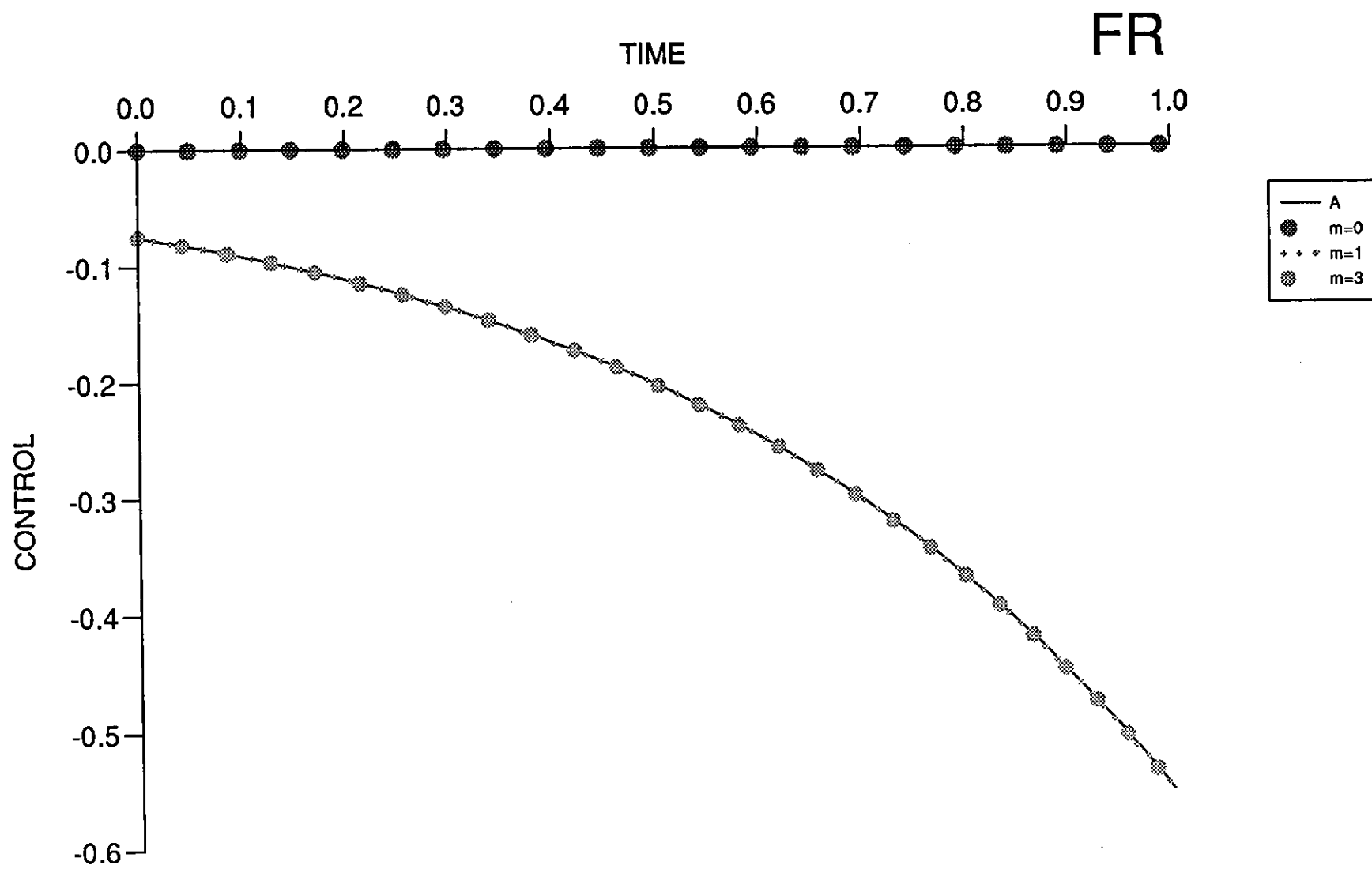


Figure (7.4.12)

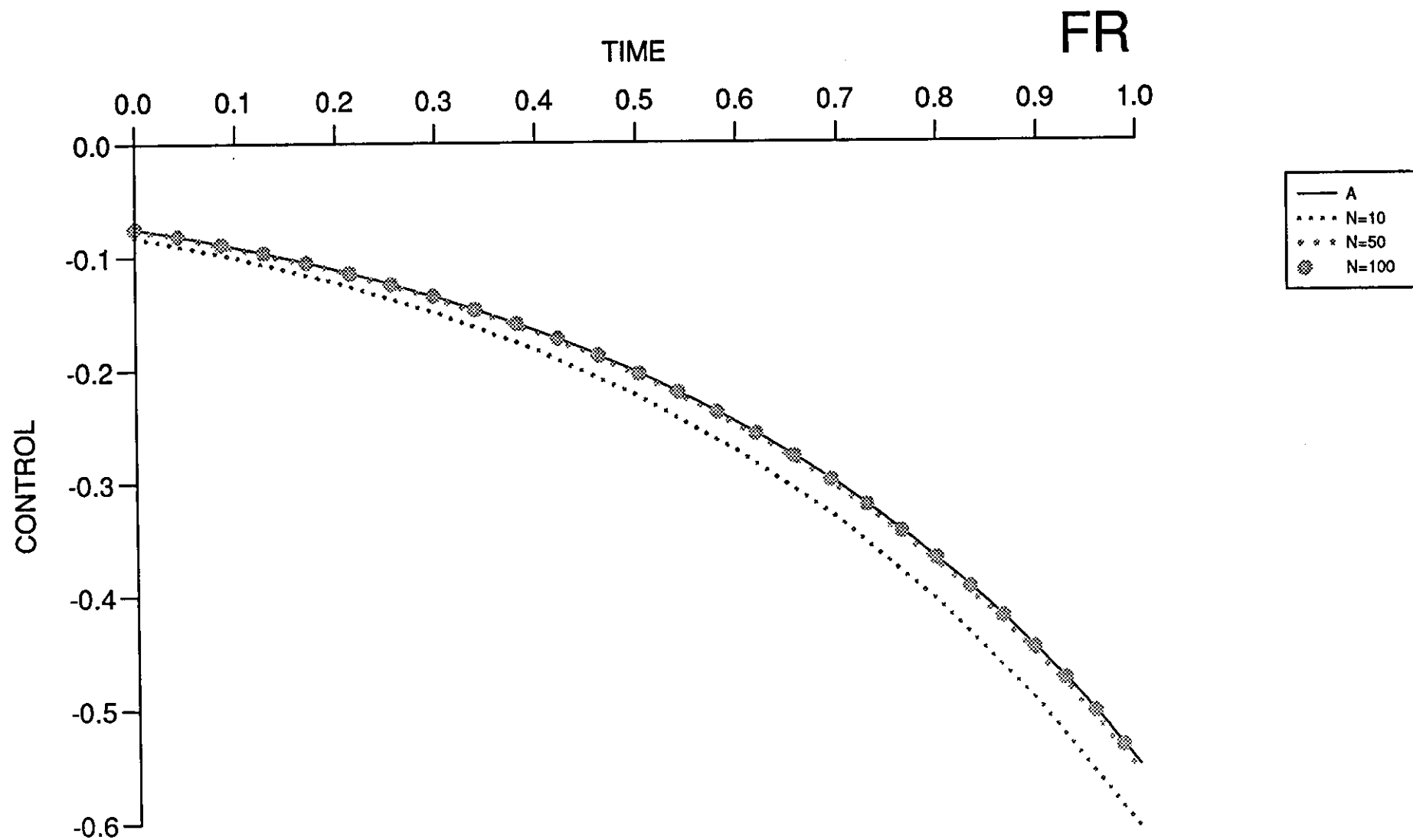


Figure (7.4.13)

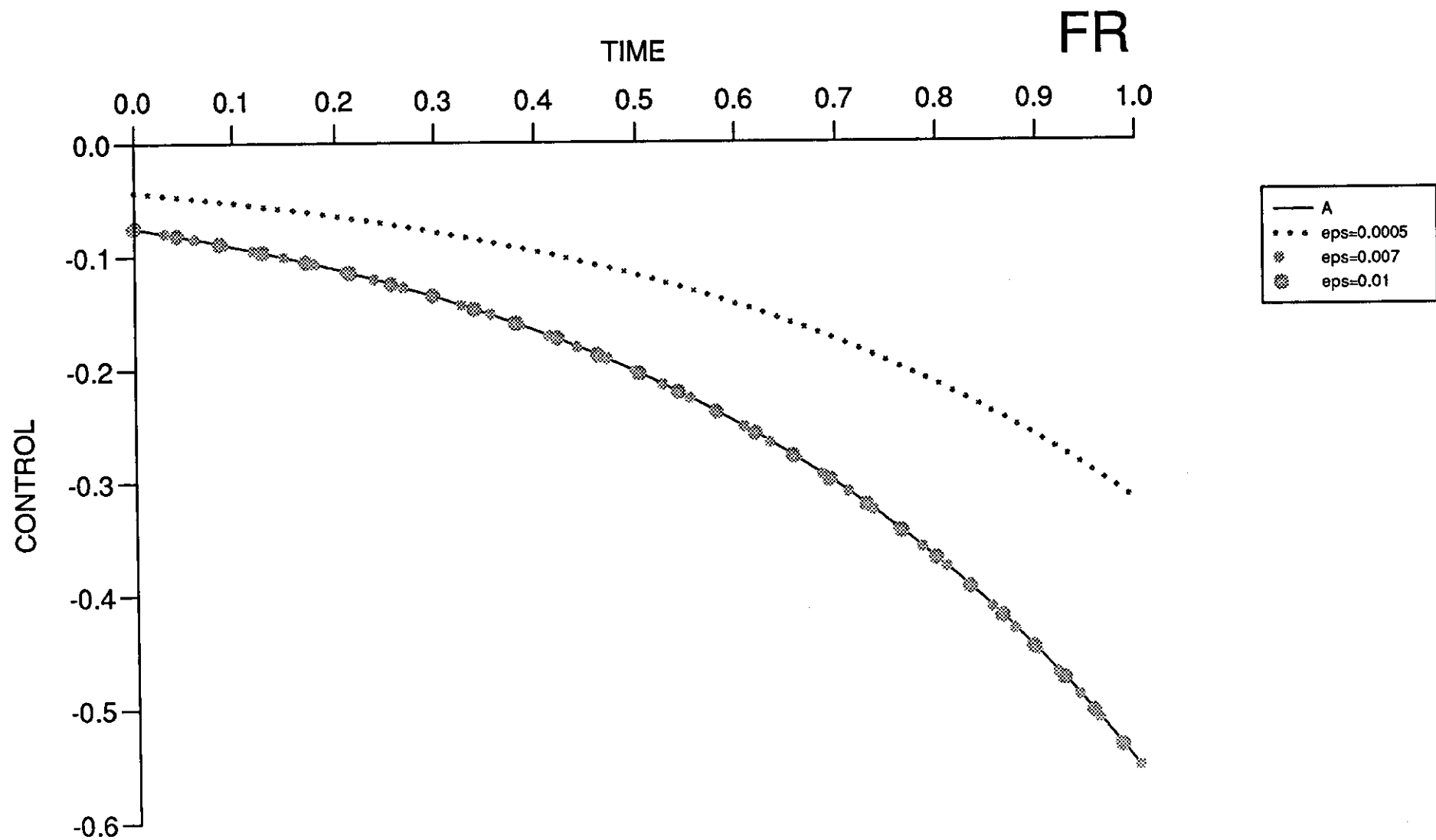


Figure (7.4.14)

FR

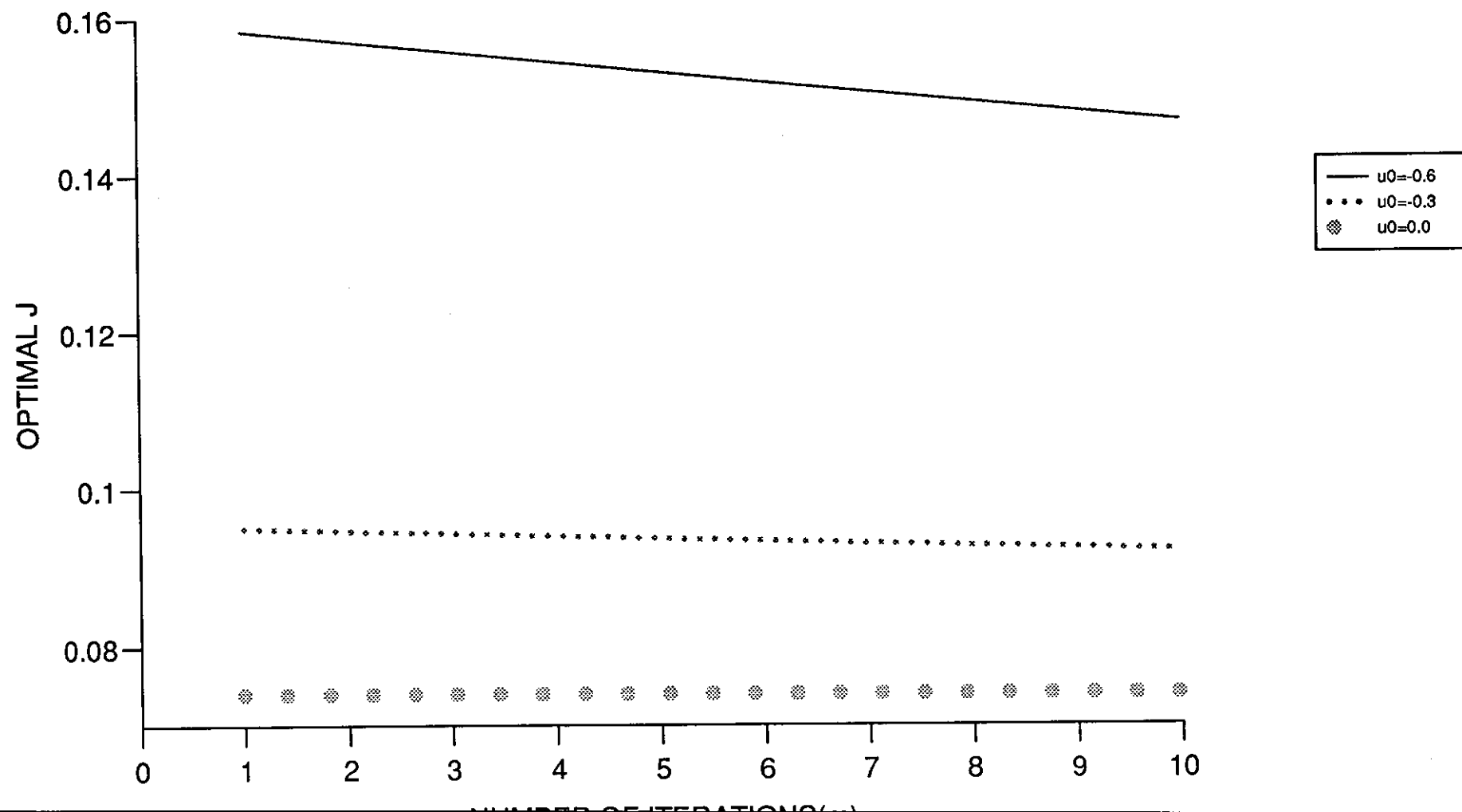


Figure (7.4.15)

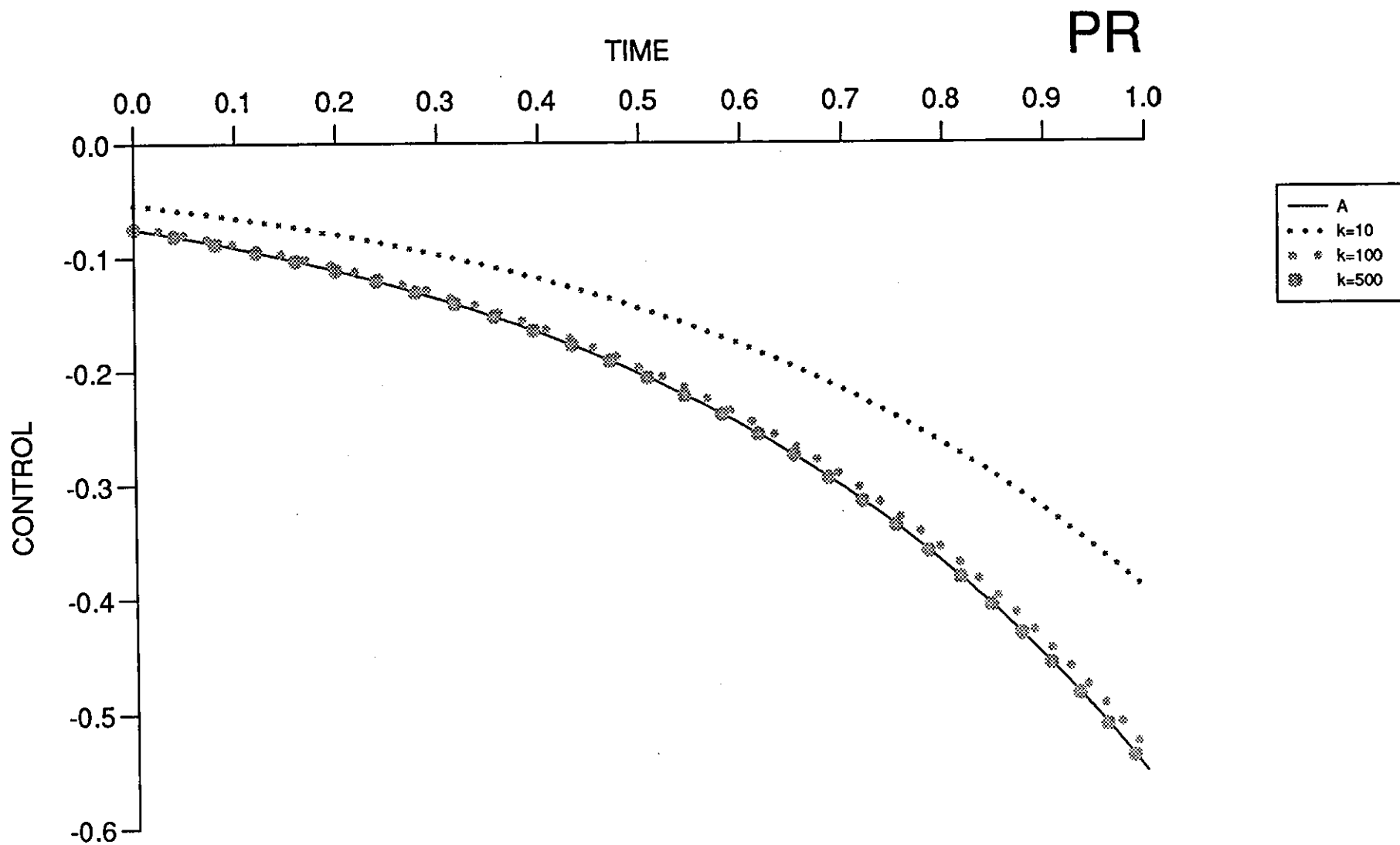


Figure (7.4.16)

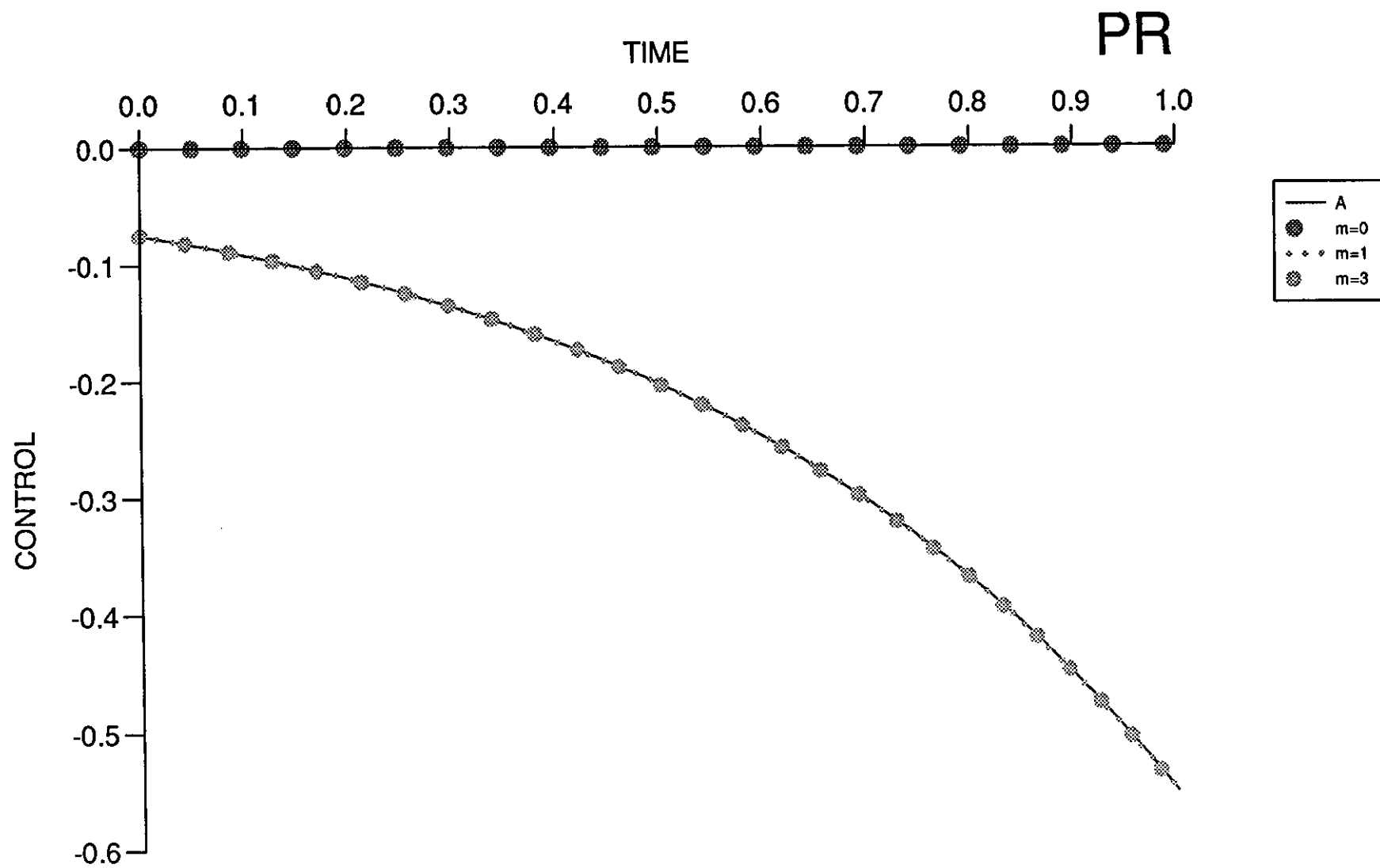


Figure (7.4.17)

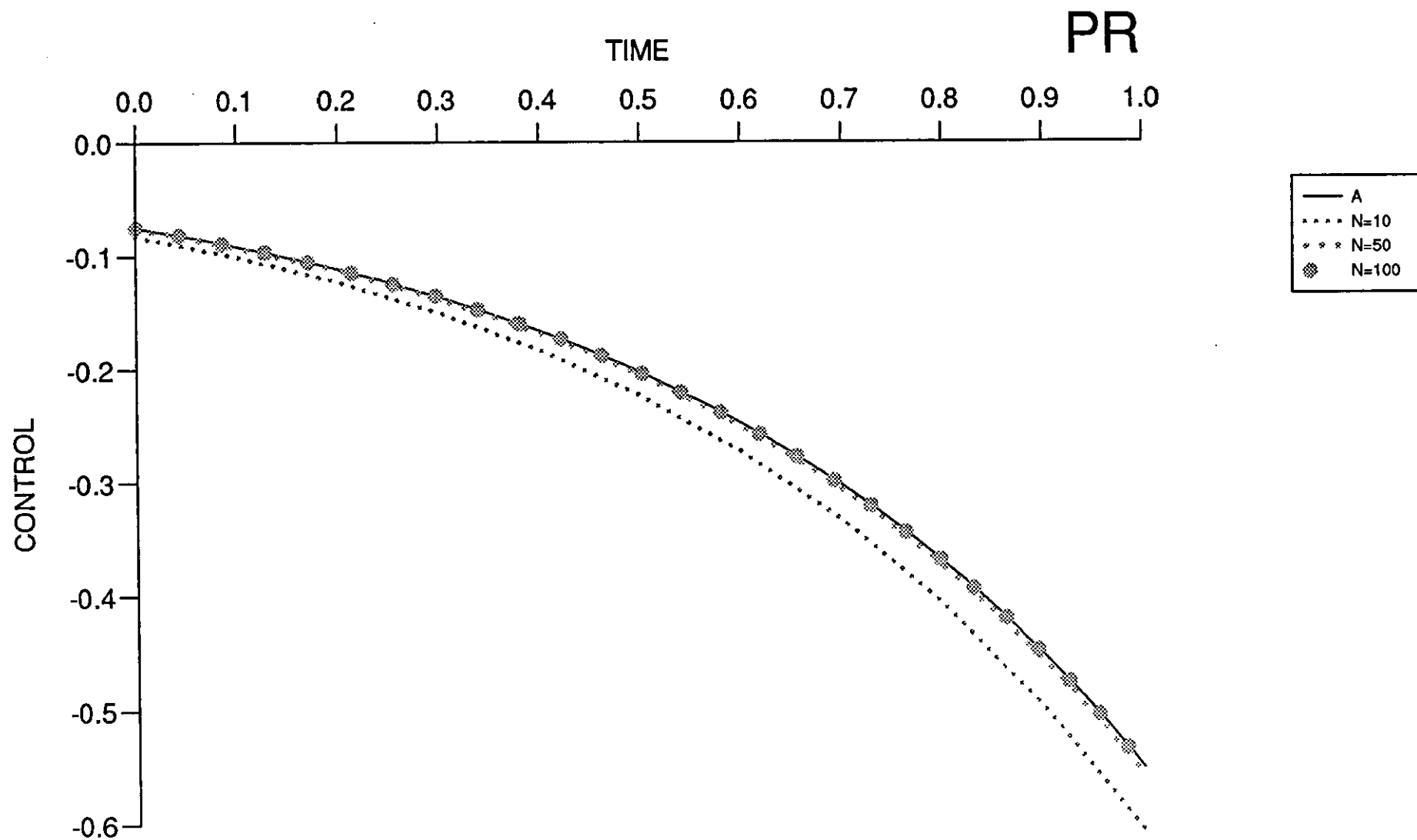


Figure (7.4.18)

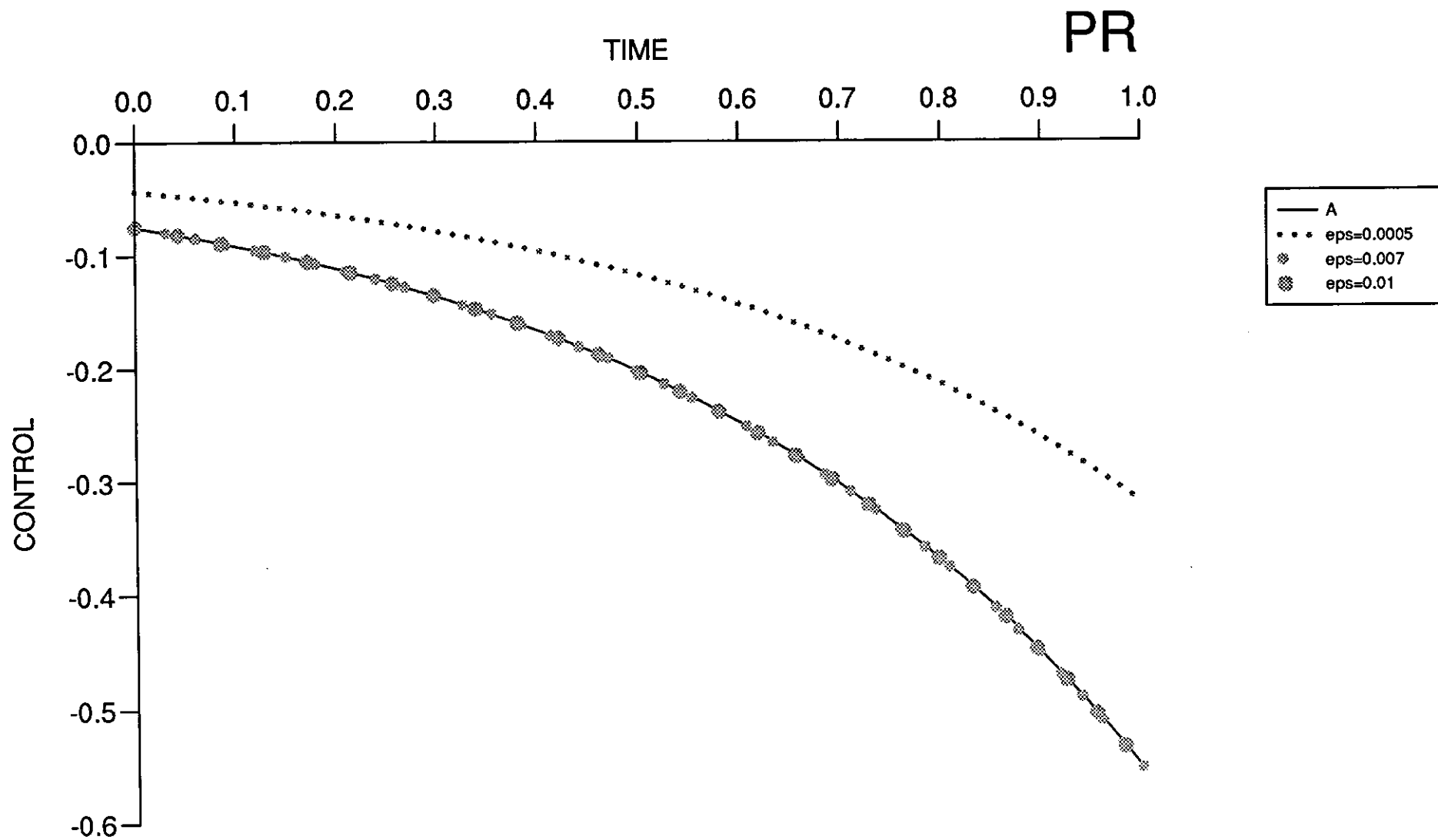


Figure (7.4.19)

PR

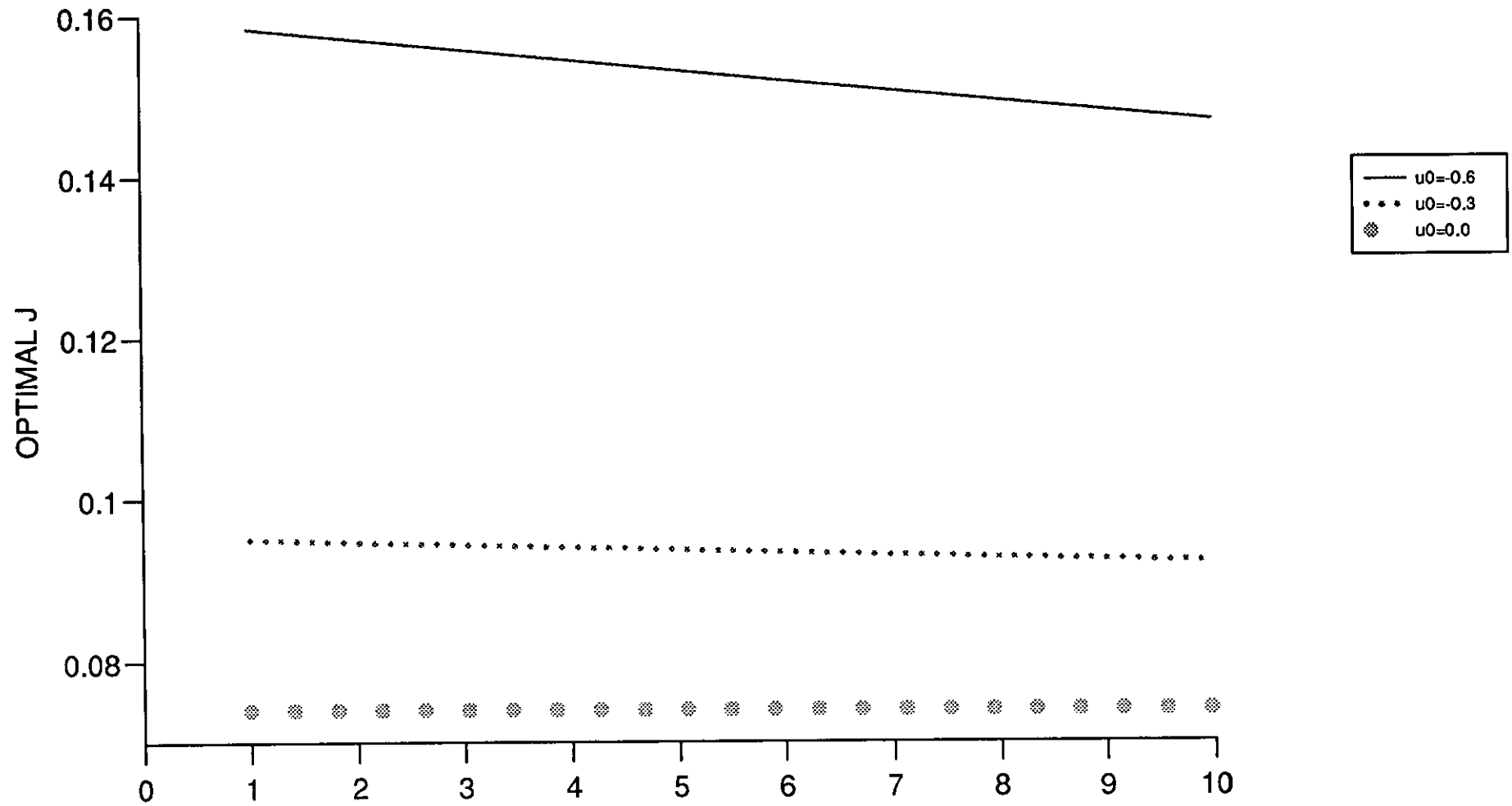


Figure (7.4.20)

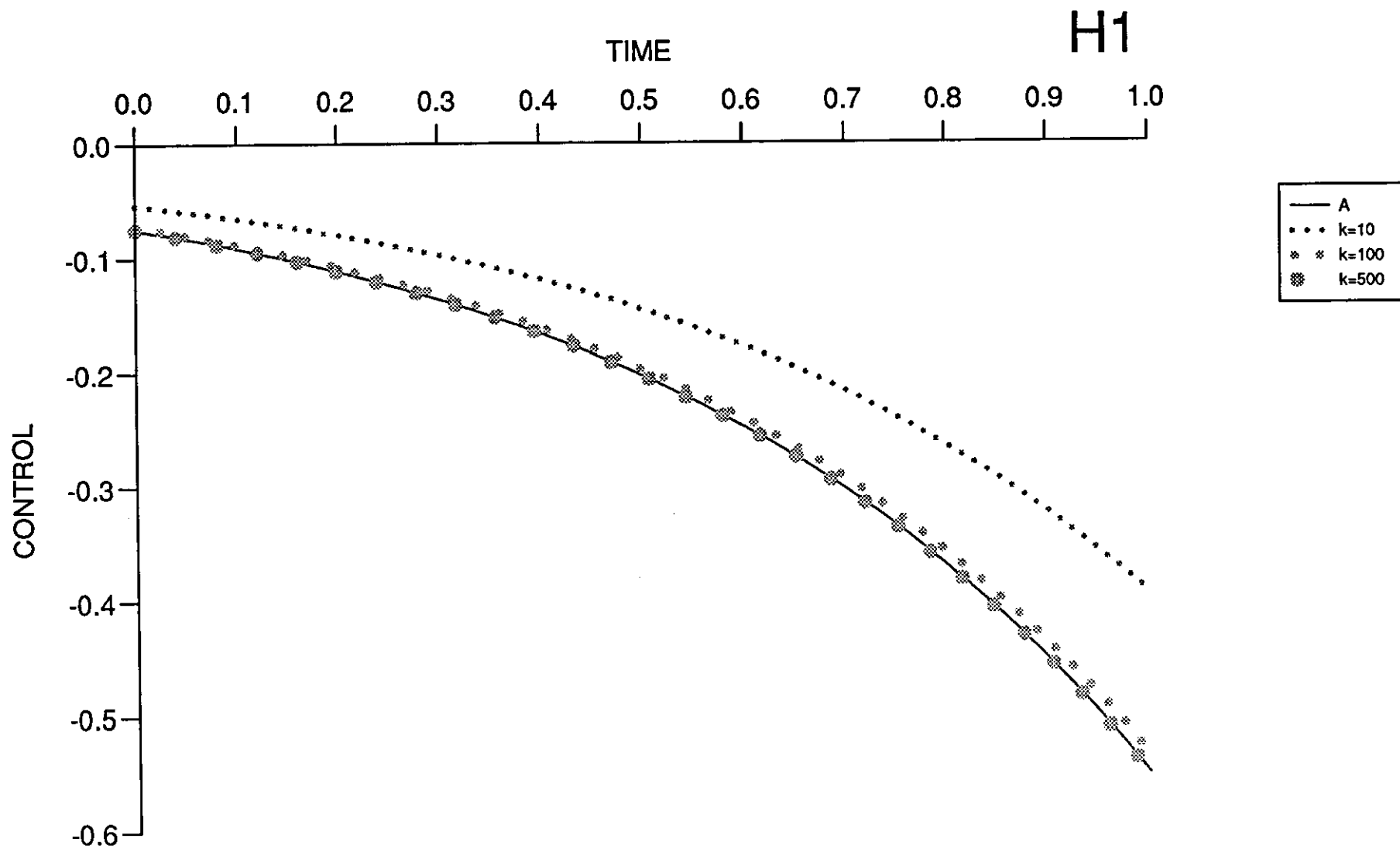


Figure (7.4.21)

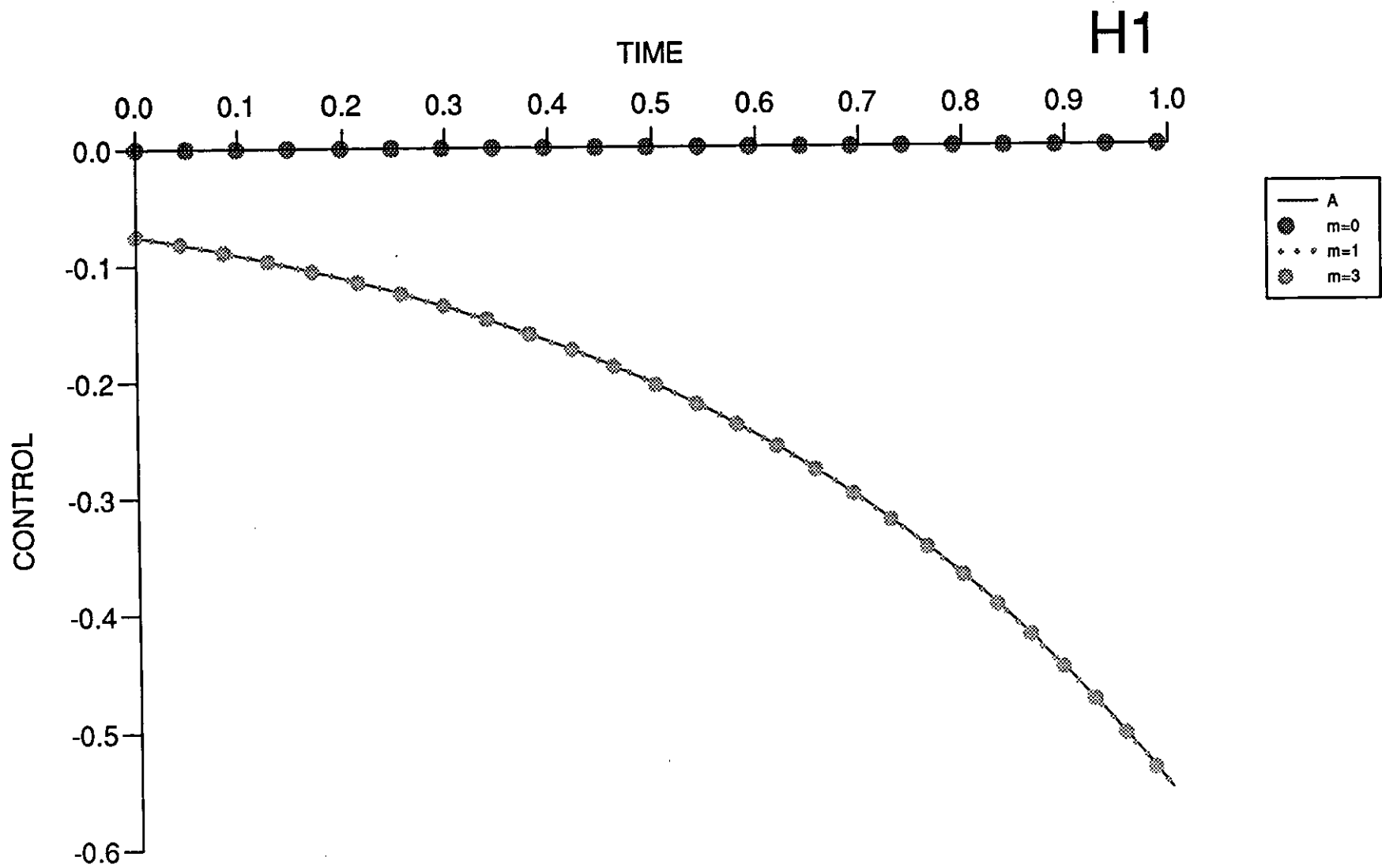


Figure (7.4.22)

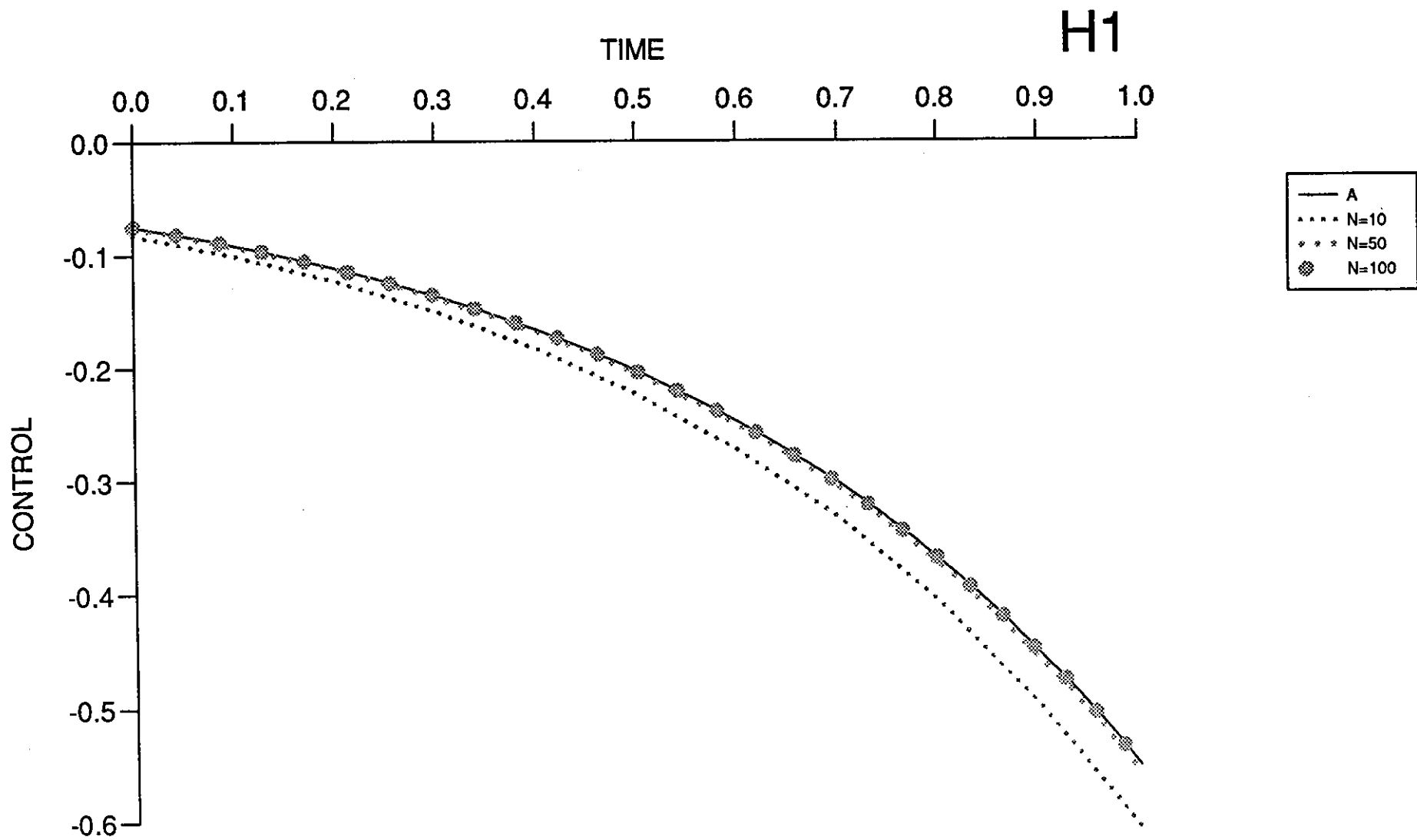


Figure (7.4.23)

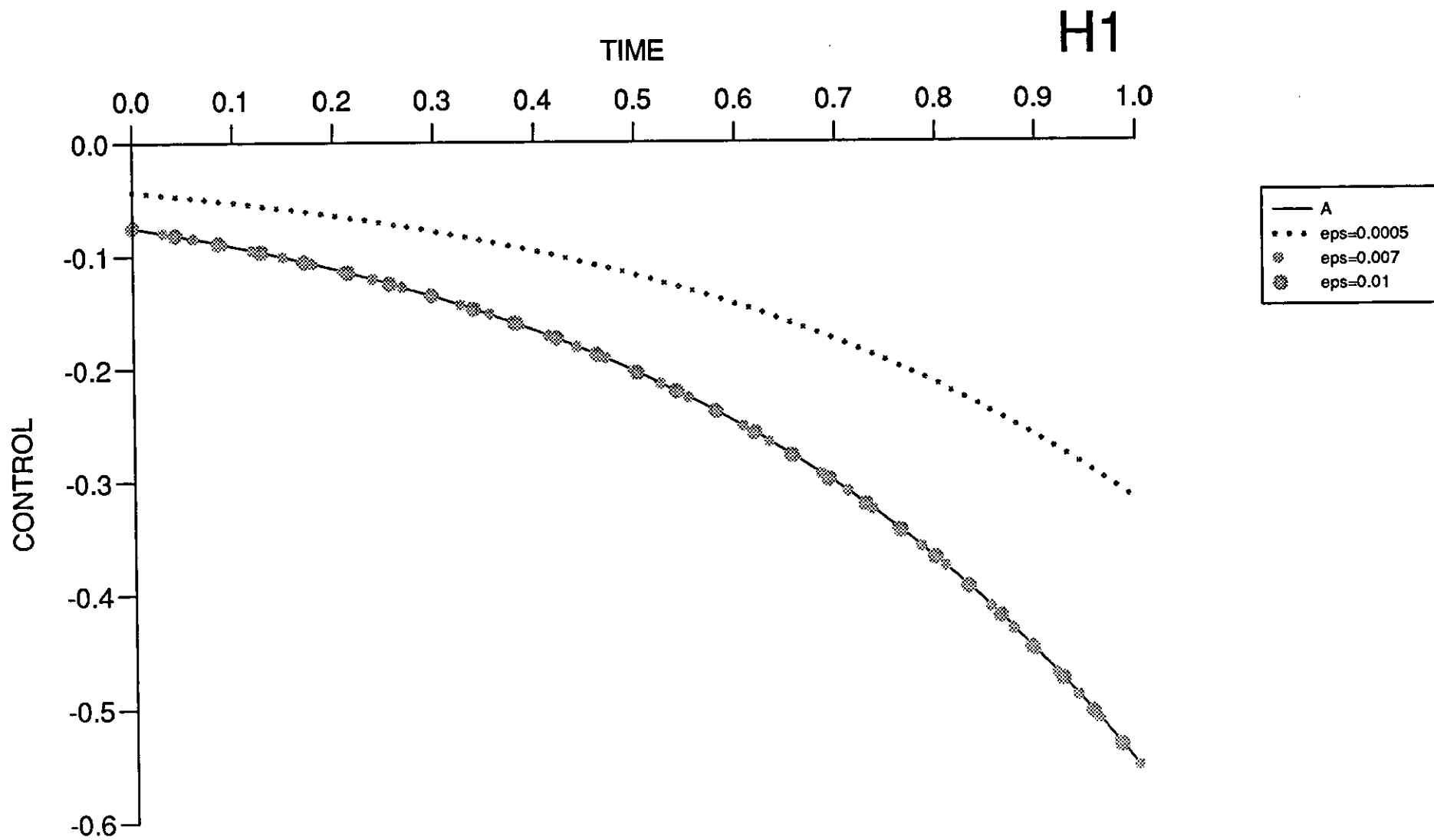


Figure (7.4.24)

H1

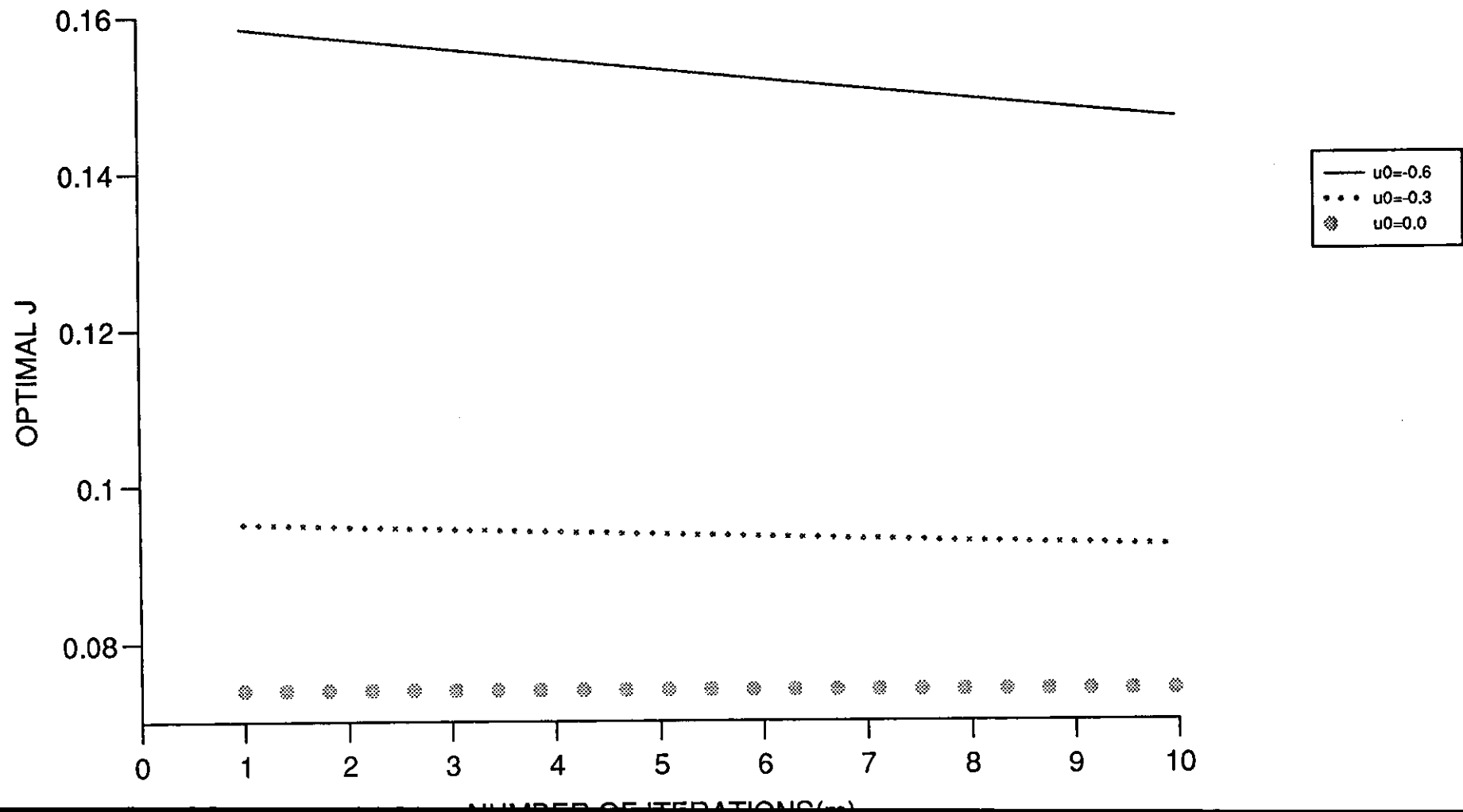


Figure (7.4.25)

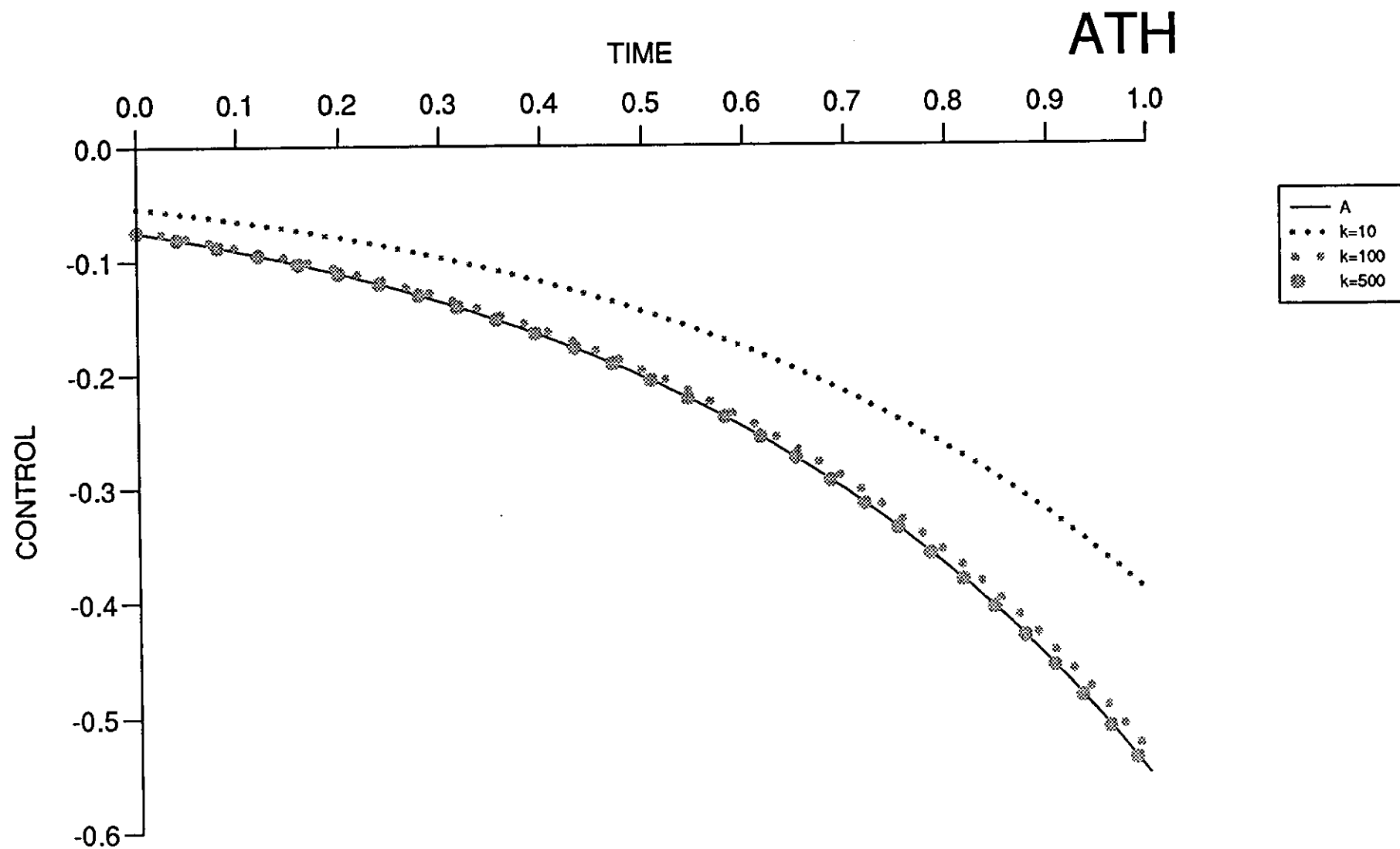


Figure (7.4.26)

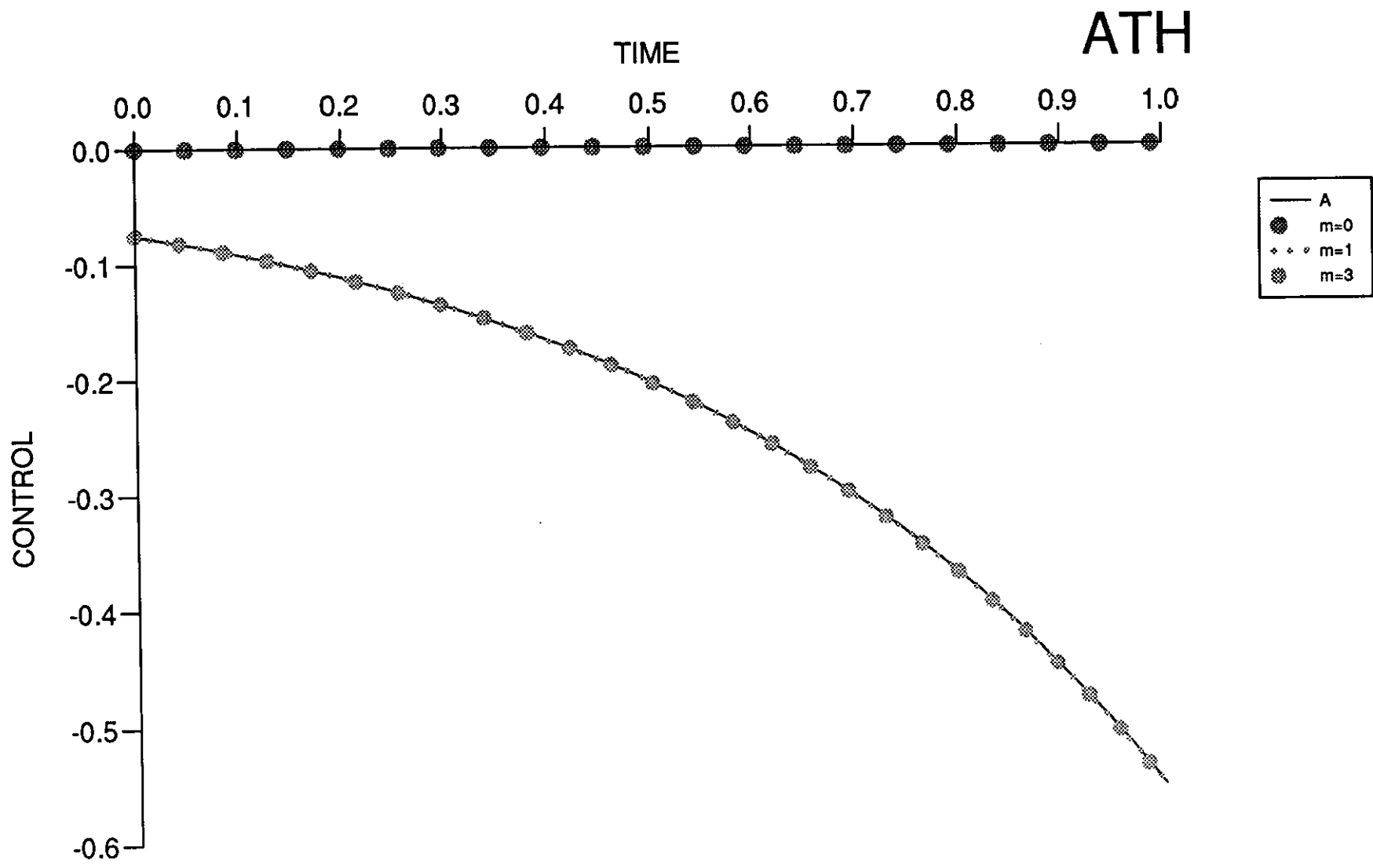


Figure (7.4.27)

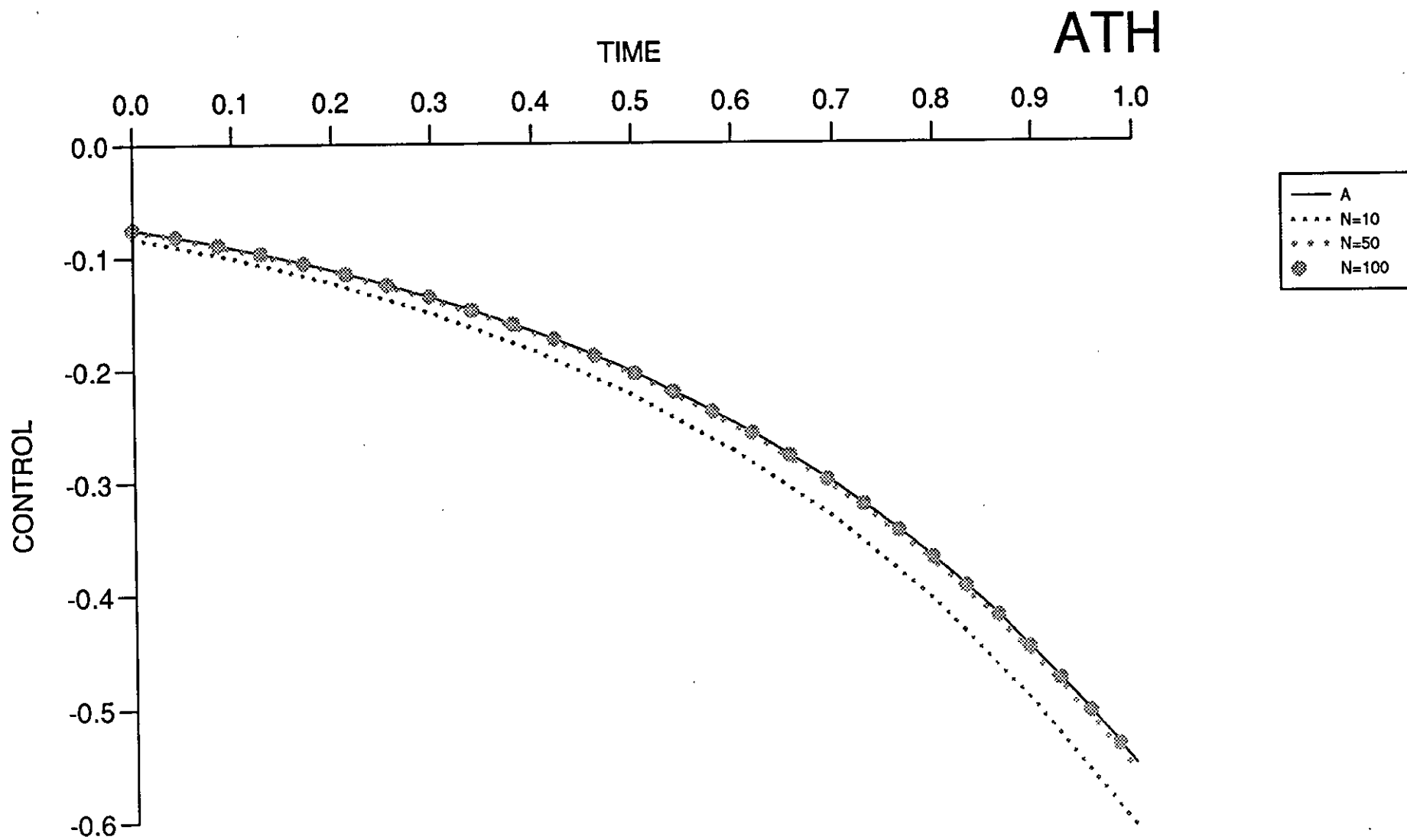


Figure (7.4.28)

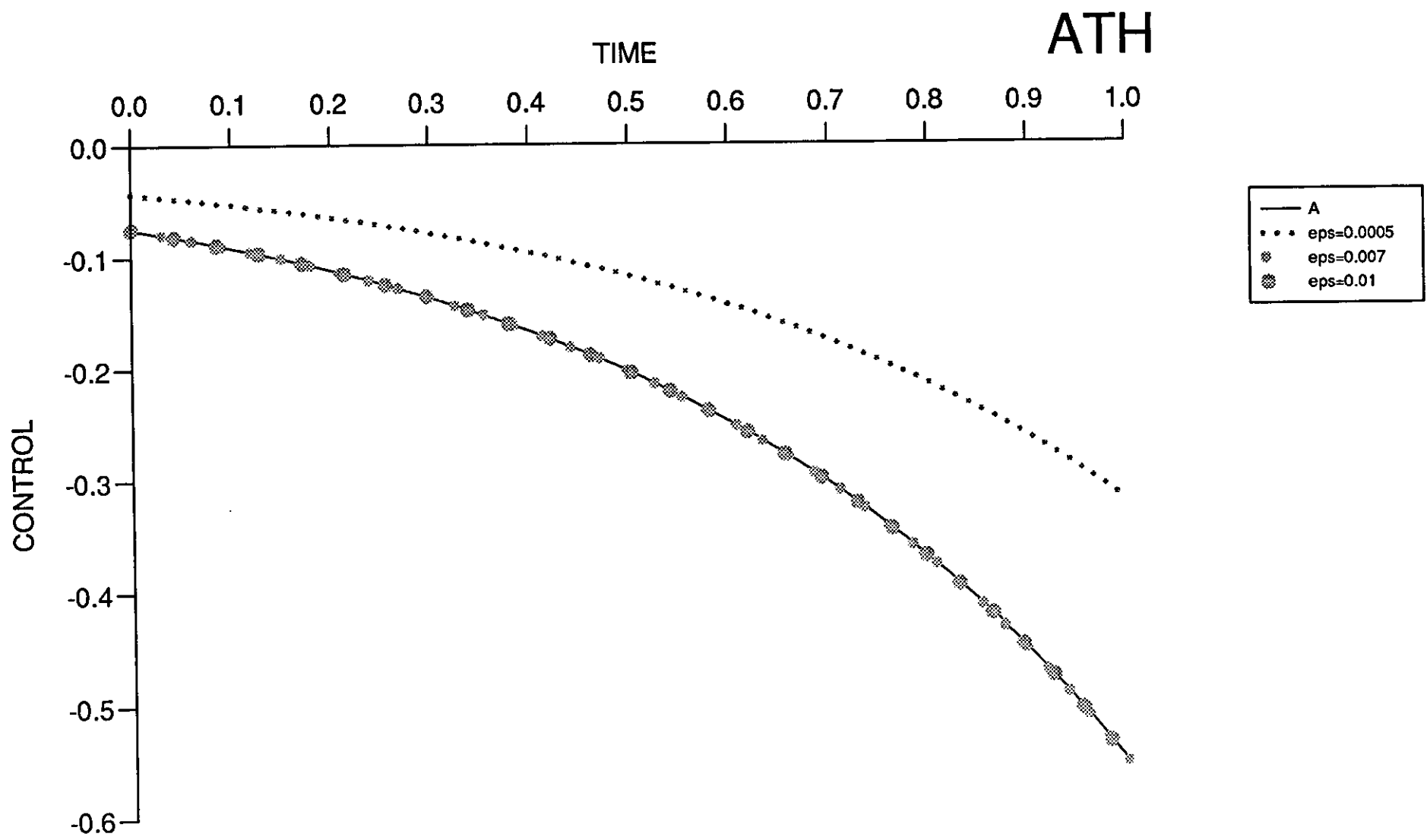


Figure (7.4.29)

ATH

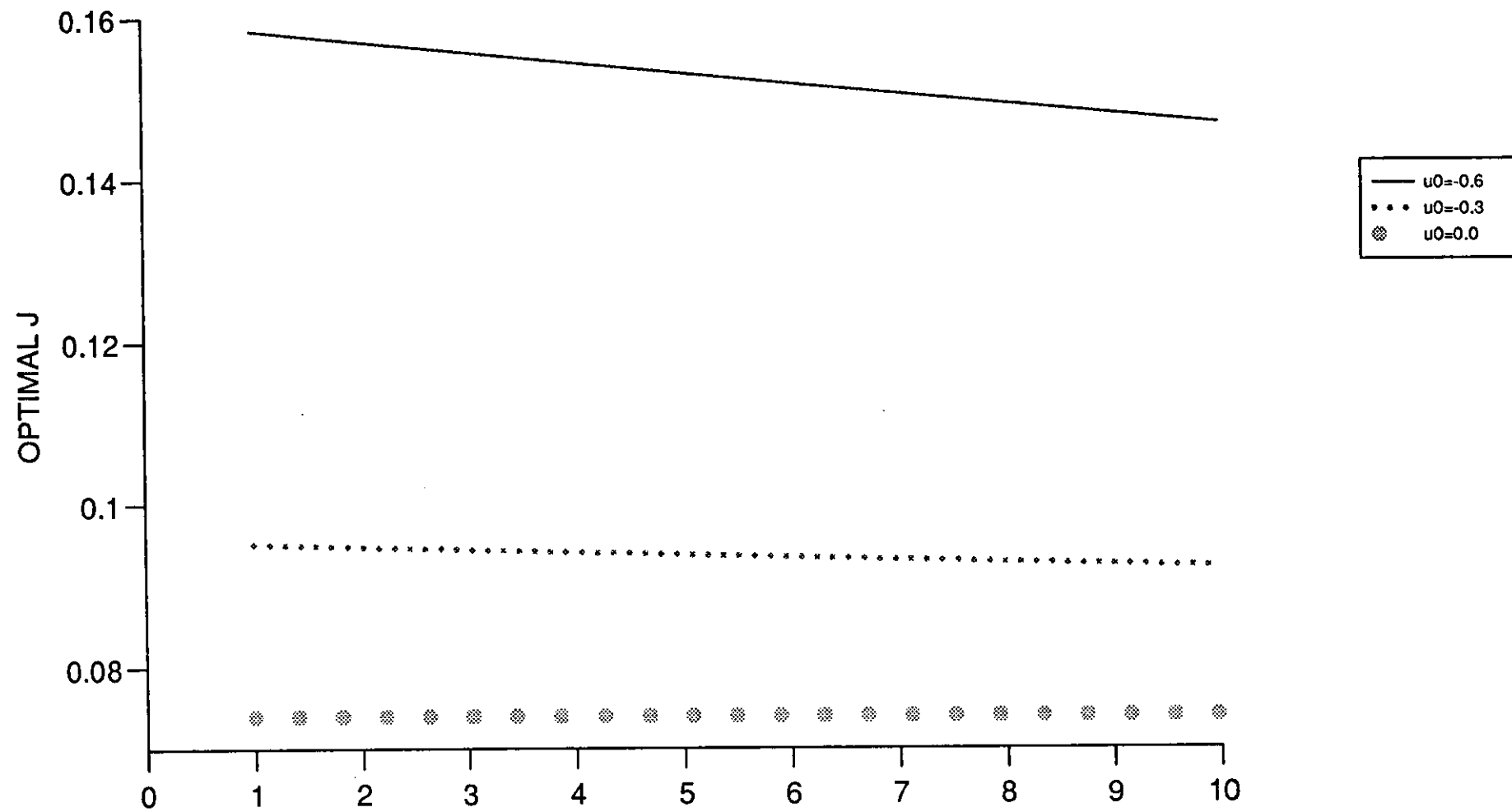


Figure (7.4.30)

H3

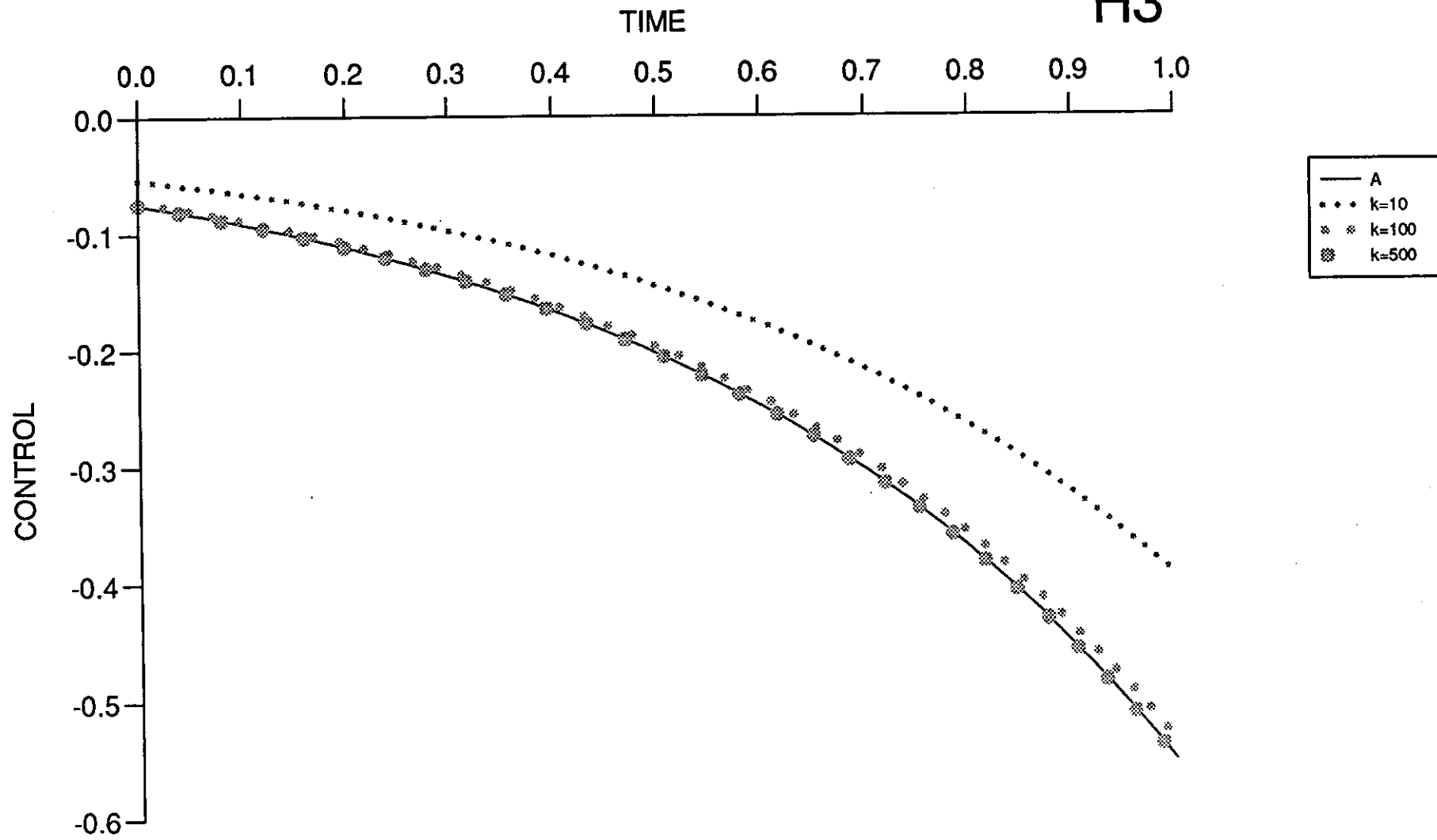


Figure (7.4.31)

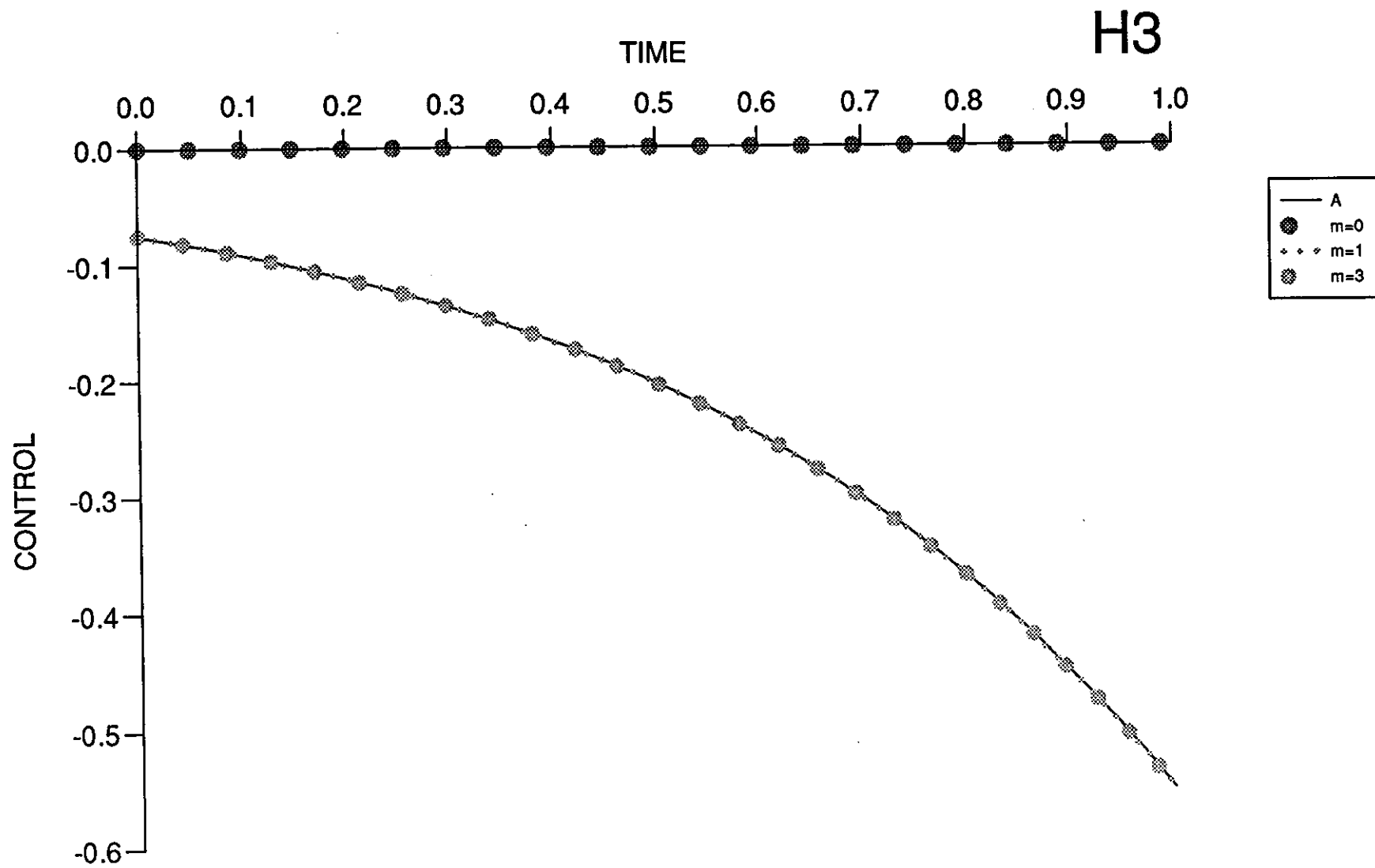


Figure (7.4.32)

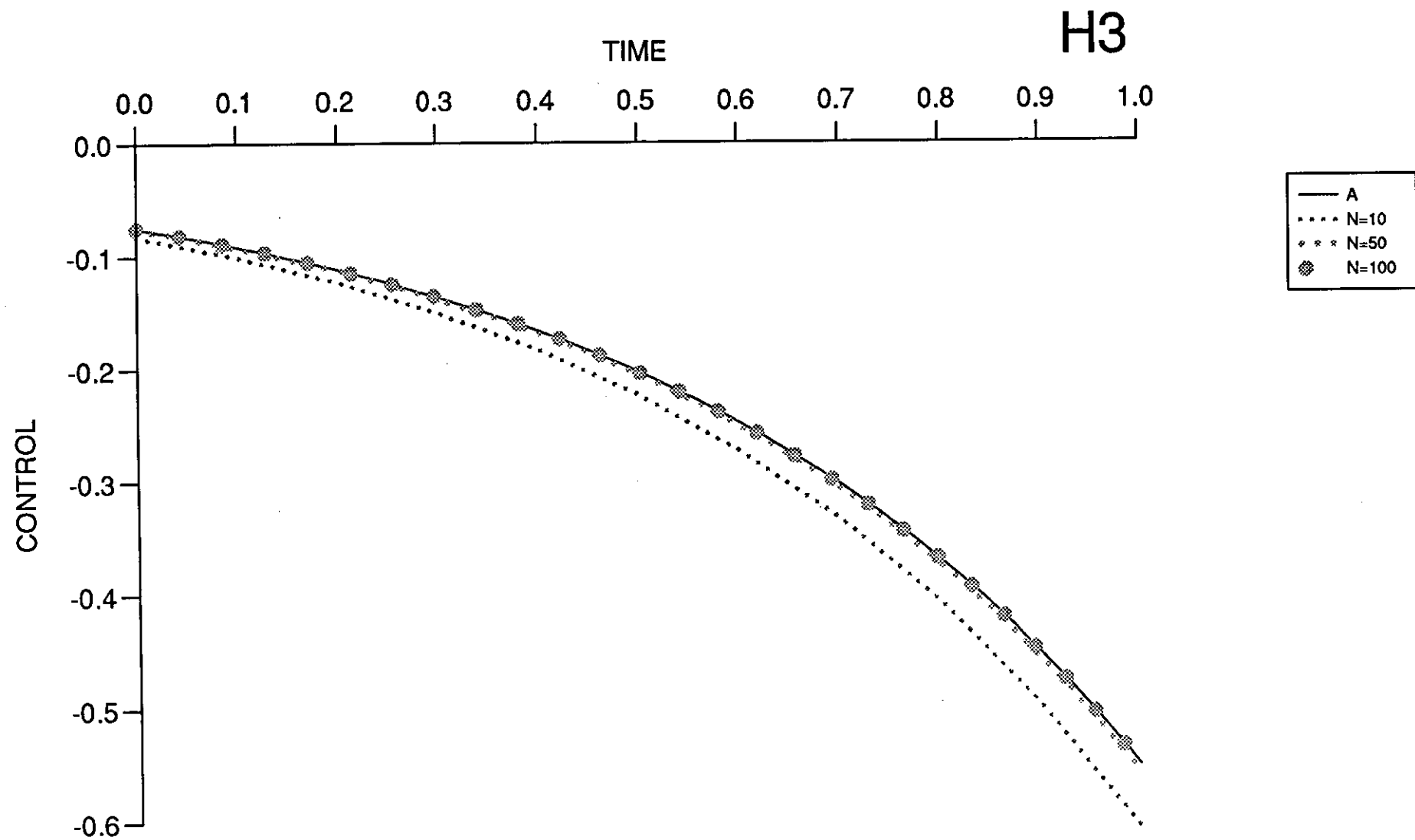


Figure (7.4.33)

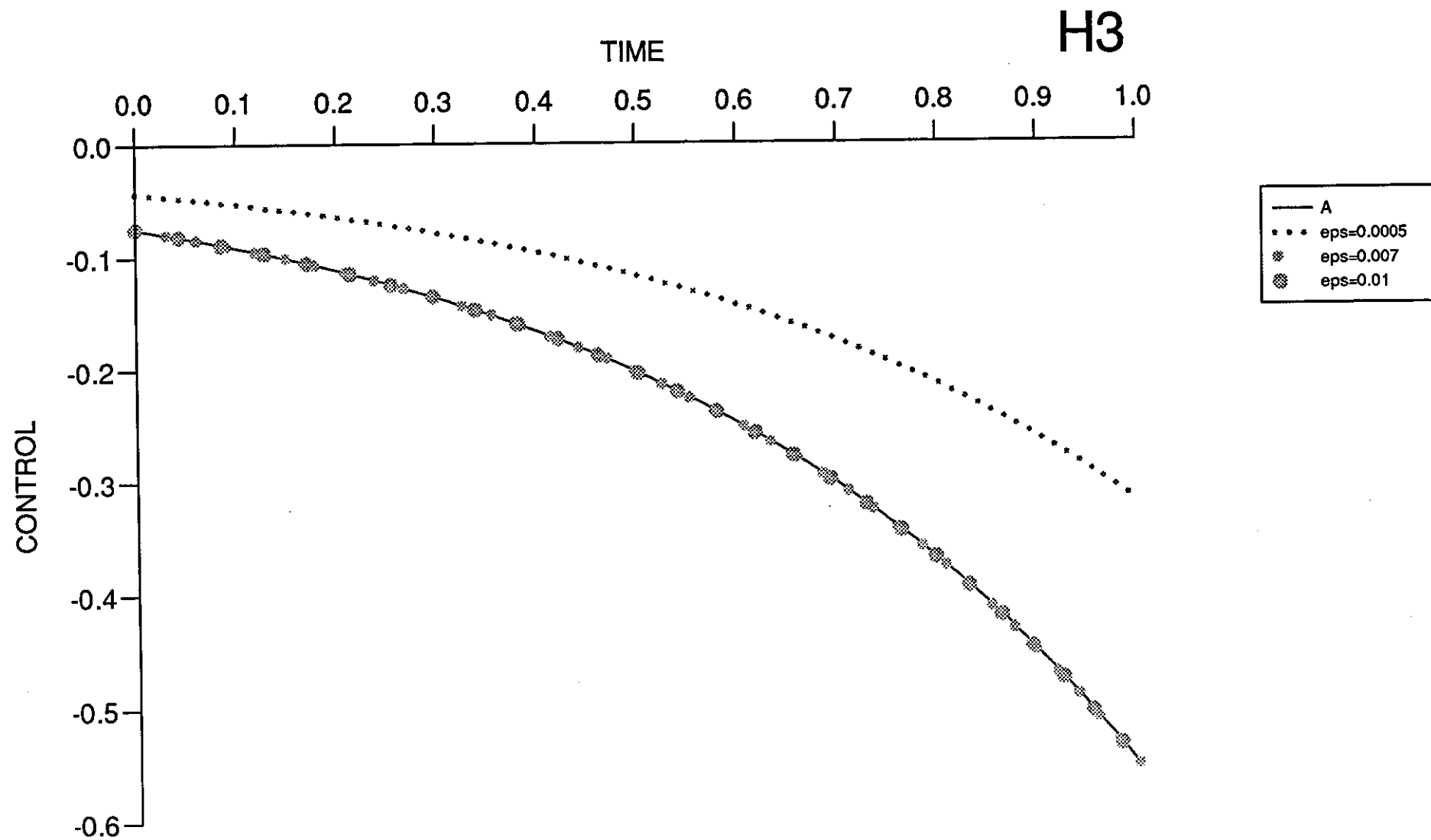


Figure (7.4.34)

H3

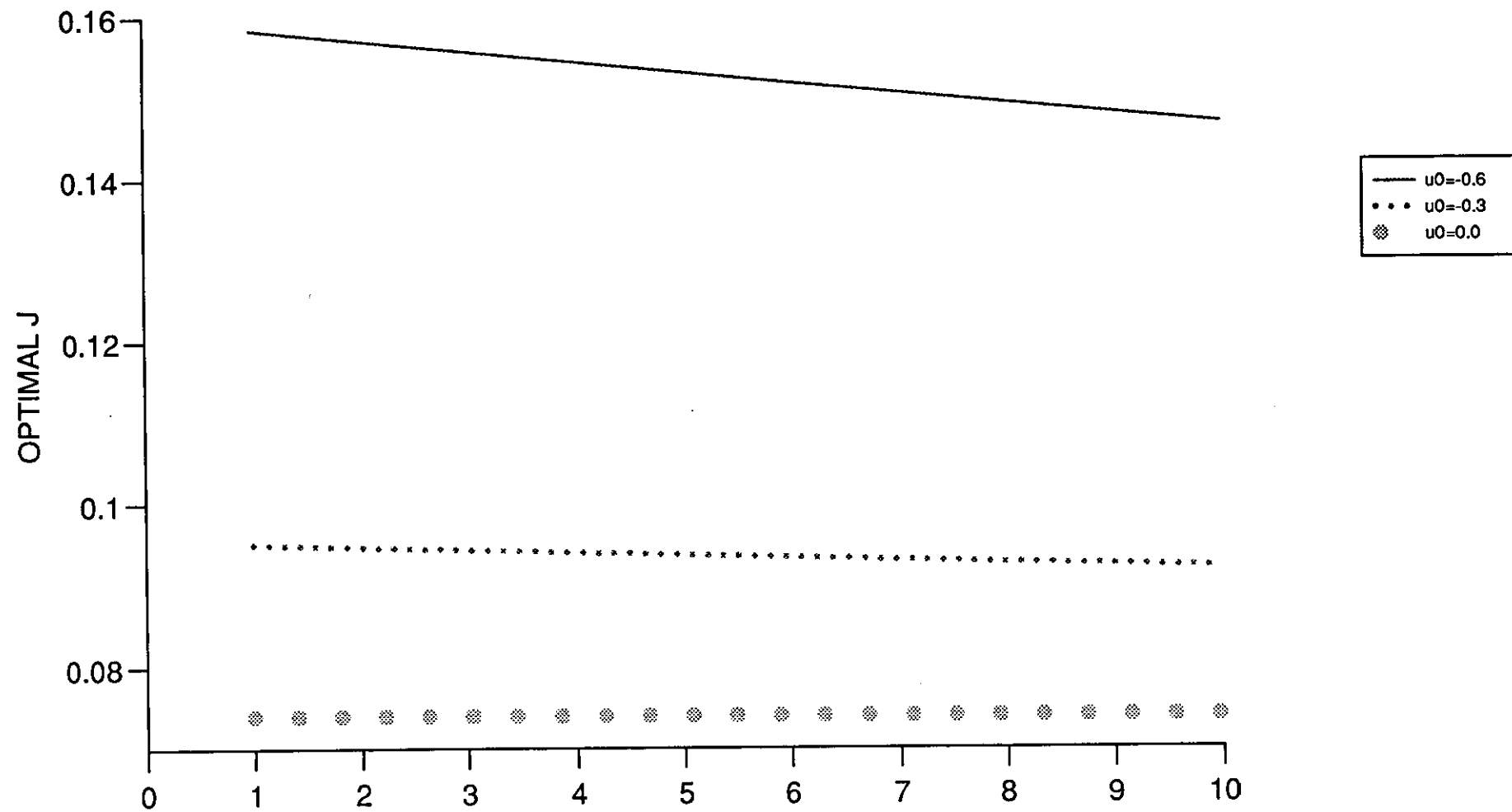


Figure (7.4.35)

COMPARISON OF THE SEVEN METHODS

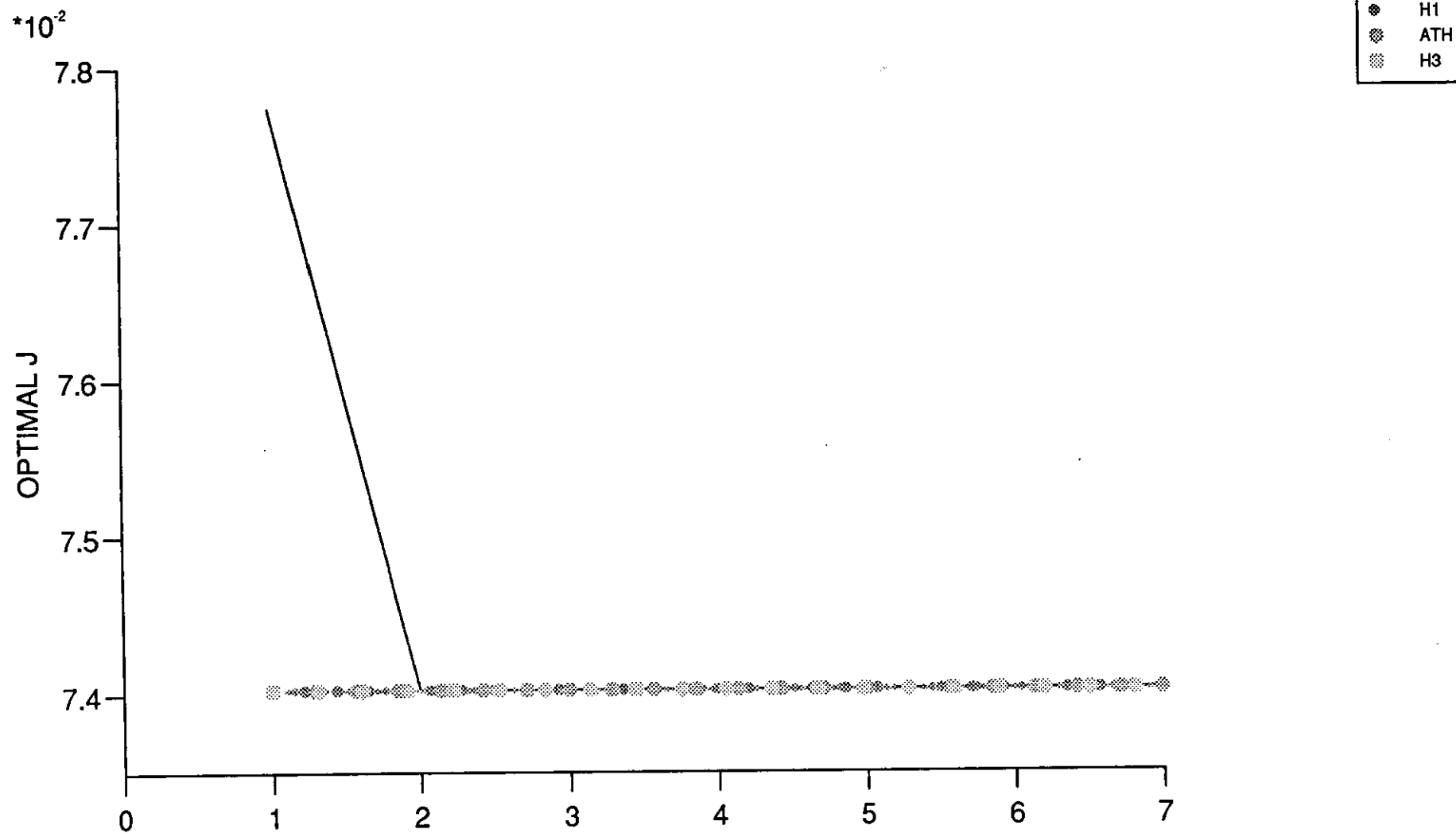


Figure (7.5.1)

Chapter 8

PROBLEM 5

8.1 A nonlinear continuous stirred tank reactor

Here we consider the numerical solution of a nonlinear continuous stirred tank reactor.

This physical system was at first modelled by Aris and Amundson (1958) [145] to [147] and then used for optimal control studies by Lapidus and Luss (1967) [148], and by Luss and Cornack (1972) [149]. Luss and Cornack suggested that, since the example that is chosen is a model of a physical system, one must be aware of the possibility of more than one solution to be necessary conditions for optimality, even when optimizing real engineering systems. To ensure that the global optimum has been reached, they recommended that the optimal control problem be solved not only by choosing several vastly different initial control policies, but by also cross-checking the final results by a different optimization procedure altogether.

The equations describing the chemical reactor are given by;

$$f_1 = \dot{x}_1 = -(2 + u)(x_1 + 0.25) + (x_2 + 0.5) \exp \left[\frac{25x_1}{x_2 + 2} \right], \quad x_1(0) = 0.09, \quad (8.1.1)$$

$$f_2 = \dot{x}_2 = 0.5 - x_2 - (x_2 + 0.5) \exp \left[\frac{25x_1}{x_1 + 2} \right], \quad x_2(0) = 0.09, \quad (8.1.2)$$

$$f_3 = \dot{x}_3 = x_1^2 + x_2^2 + 0.1u^2, \quad x_3(0) = 0.0, \quad (8.1.3)$$

where,

x_1 = deviation from dimensionless steady state temperature,

x_2 = deviation from dimensionless steady state concentration,

t_f = final time,

u = Control Variable.

The optimal control process is to choose the control u in the time interval, $0 \leq t \leq t_f$ to minimize,

$$J = x_3(t_f), \quad (8.1.4)$$

where the final time $t_f = 0.78$.

The Hamiltonian for this problem is,

$$\begin{aligned}
 H &= \sum_{i=1}^3 \lambda_i f_i, \\
 &= \lambda_1 f_1 + \lambda_2 f_2 + \lambda_3 f_3, \\
 &= \lambda_1 \left[-(2+u)(x_1 + 0.25) + (x_2 + 0.5) \exp \left(\frac{25x_1}{x_1 + 2} \right) \right] \\
 &= +\lambda_2 \left[0.5 - x_2 - (x_2 + 0.5) \exp \left(\frac{25x_1}{x_1 + 2} \right) \right] \\
 &= +\lambda_3 [x_1^2 + x_2^2 + 0.1u^2].
 \end{aligned} \tag{8.1.5}$$

The adjoint equations are,

$$\dot{\lambda}_i = \sum_{j=1}^3 \lambda_j \frac{\partial f_j}{\partial x_i}$$

with,

$$\lambda_i(t_f) = \frac{\partial \phi}{\partial x_i} \Big|_{t=t_f}, \quad i = 1, 2, 3,$$

i.e.,

$$\begin{aligned}
 \dot{\lambda}_1 &= - \left[\lambda_1 \frac{\partial f_1}{\partial x_1} + \lambda_2 \frac{\partial f_2}{\partial x_1} + \lambda_3 \frac{\partial f_3}{\partial x_1} \right] \\
 &= -\lambda_1 \left[-2 - u + (x_2 + 0.5) \frac{50}{(x_1 + 2)^2} e^{\frac{25x_1}{x_1 + 2}} \right] \\
 &\quad -\lambda_2 \left[-(x_2 + 0.5) \frac{50}{(x_1 + 2)^2} e^{\frac{25x_1}{x_1 + 2}} \right] - \lambda_3 [2x_1],
 \end{aligned} \tag{8.1.6}$$

$$\begin{aligned}
 \dot{\lambda}_2 &= - \left[\lambda_1 \frac{\partial f_1}{\partial x_2} + \lambda_2 \frac{\partial f_2}{\partial x_2} + \lambda_3 \frac{\partial f_3}{\partial x_2} \right] \\
 &= -\lambda_1 \left[e^{\frac{25x_1}{x_1 + 2}} \right] - \lambda_2 \left[-1 - (e^{\frac{25x_1}{x_1 + 2}}) \right] \\
 &\quad -\lambda_3 [2x_2],
 \end{aligned} \tag{8.1.7}$$

$$\dot{\lambda}_3 = - \left[\lambda_1 \frac{\partial f_1}{\partial x_3} + \lambda_2 \frac{\partial f_2}{\partial x_3} + \lambda_3 \frac{\partial f_3}{\partial x_3} \right] = 0, \tag{8.1.8}$$

with,

$$\lambda_1(t_f) = \frac{\partial}{\partial x_1} (x_3(t_f)) \Big|_{t=t_f} = 0, \tag{8.1.9}$$

$$\lambda_2(t_f) = \frac{\partial}{\partial x_2} (x_3(t_f)) \Big|_{t=t_f} = 0, \tag{8.1.10}$$

$$\lambda_3(t_f) = \frac{\partial}{\partial x_3} (x_3(t_f)) \Big|_{t=t_f} = 1. \tag{8.1.11}$$

8.2 Numerical Solutions

8.2.1 The state and adjoint equations

The state equations are:

$$\left. \begin{aligned} \dot{x}_1 &= f_1 = -(2+u)(x_1+0.25) + (x_2+0.5) \exp \left[\frac{25x_1}{x_1+2} \right], & x_1(0) &= 0.09 \\ \dot{x}_2 &= f_2 = 0.5 - x_2 - (x_2+0.5) \exp \left[\frac{25x_1}{x_1+2} \right], & x_2(0) &= 0.09 \\ \dot{x}_3 &= f_3 = x_1^2 + x_2^2 + 0.1u^2, & x_3(0) &= 0.0. \end{aligned} \right\} \quad (8.2.1)$$

Using the Runge-Kutta 4th order method for numerical solution of (8.2.1) we get,

$$\begin{aligned} x_{1,n+1} &= x_{1,n} + \frac{1}{6}(u_1 + 2u_2 + 2u_3 + u_4), \\ x_{2,n+1} &= x_{2,n} + \frac{1}{6}(v_1 + 2v_2 + 2v_3 + v_4), \\ x_{3,n+1} &= x_{3,n} + \frac{1}{6}(w_1 + 2w_2 + 2w_3 + w_4), \end{aligned}$$

where,

$$\begin{aligned} u_1 &= hf_1(x_{1,n}, x_{2,n}, x_{3,n}) = h \left\{ -(2+u_n)(x_{1,n}+0.25) \right. \\ &\quad \left. + (x_{2,n}+0.5) \exp \left[\frac{25x_{1,n}}{x_{1,n}+2} \right] \right\}, \\ v_1 &= hf_2(x_{1,n}, x_{2,n}, x_{3,n}) = h \left\{ 0.5 - x_{2,n} - (x_{2,n}+0.5) \exp \left[\frac{25x_{1,n}}{x_{1,n}+2} \right] \right\}, \\ w_1 &= hf_3(x_{1,n}, x_{2,n}, x_{3,n}) = h \left[x_{1,n}^2 + x_{2,n}^2 + 0.1u_n^2 \right], \\ u_2 &= hf_1\left(x_{1,n} + \frac{1}{2}u_1, x_{2,n} + \frac{1}{2}v_1, x_{3,n} + \frac{1}{2}w_1\right), \\ &= h \left[-(2+u_n)\left(x_{1,n} + \frac{1}{2}u_1 + 0.25\right) + \left(x_{2,n} + \frac{1}{2}v_1 + 0.5\right) \exp \right. \\ &\quad \left. \left(\frac{25\left(x_{1,n} + \frac{1}{2}u_1\right)}{\left(x_{1,n} + \frac{1}{2}u_1\right) + 2} \right) \right], \\ v_2 &= hf_2\left(x_{1,n} + \frac{1}{2}u_1, x_{2,n} + \frac{1}{2}v_1, x_{3,n} + \frac{1}{2}w_1\right), \\ &= h \left\{ 0.5 - \left(x_{2,n} + \frac{1}{2}v_1\right)\left(x_{2,n} + \frac{1}{2}v_1 + 0.5\right) \exp \right. \\ &\quad \left. \left[\frac{25\left(x_{1,n} + \frac{1}{2}u_1\right)}{\left(x_{1,n} + \frac{1}{2}u_1\right) + 2} \right] \right\}, \end{aligned}$$

$$\begin{aligned}
w_2 &= hf_3(x_{1,n} + \frac{1}{2}u_1, x_{2,n} + \frac{1}{2}v_1, x_{2,n} + \frac{1}{2}w_1), \\
&= h \left[(x_{1,n} + \frac{1}{2}u_1)^2 + (x_{2,n} + \frac{1}{2}v_1)^2 + 0.1u_n^2 \right],
\end{aligned}$$

$$\begin{aligned}
u_3 &= hf_1 \left(x_{1,n} + \frac{1}{2}u_2, x_{2,n} + \frac{1}{2}v_2, x_{3,n} + \frac{1}{2}w_2 \right), \\
&= h \left[-(2 + u_n)(x_{1,n} + \frac{1}{2}u_2 + 0.25) + (x_{2,n} + \frac{1}{2}v_2 + 0.5) \right. \\
&\quad \left. \exp \left(\frac{25(x_{1,n} + \frac{1}{2}u_2)}{(x_{1,n} + \frac{1}{2}u_2) + 2} \right) \right],
\end{aligned}$$

$$\begin{aligned}
v_3 &= hf_2(x_{1,n} + \frac{1}{2}u_2, x_{2,n} + \frac{1}{2}v_2, x_{3,n} + \frac{1}{2}w_2), \\
&= h \left\{ 0.5 - (x_{2,n} + \frac{1}{2}v_2) - (x_{2,n} + \frac{1}{2}v_2 + 0.5) \exp \right. \\
&\quad \left. \left[\frac{25(x_{1,n} + \frac{1}{2}u_2)}{(x_{1,n} + \frac{1}{2}u_2) + 2} \right] \right\},
\end{aligned}$$

$$\begin{aligned}
w_3 &= hf_3 \left(x_{1,n} + \frac{1}{2}u_2, x_{2,n} + \frac{1}{2}v_2, x_{3,n} + \frac{1}{2}w_2 \right), \\
&= h \left[(x_{1,n} + \frac{1}{2}u_2)^2 + (x_{2,n} + \frac{1}{2}v_2)^2 + 0.1u_n^2 \right],
\end{aligned}$$

$$\begin{aligned}
u_4 &= hf_1(x_{1,n} + u_3, x_{2,n} + v_3, x_{3,n} + w_3), \\
&= h \left\{ -(2 + u_n)(x_{1,n} + u_3 + 0.25) + (x_{2,n} + v_3 + 0.5) \right. \\
&\quad \left. \exp \left[\frac{25(x_{1,n} + u_3)}{(x_{1,n} + u_3) + 2} \right] \right\},
\end{aligned}$$

$$\begin{aligned}
v_4 &= hf_2(x_{1,n} + u_3, x_{2,n} + v_3, x_{3,n} + w_3), \\
&= h \left\{ 0.5 - [x_{2,n} + v_3] - [x_{2,n} + v_3 + 0.5] \exp \left[\frac{25(x_{1,n} + u_3)}{(x_{1,n} + u_3) + 2} \right] \right\},
\end{aligned}$$

$$\begin{aligned}
w_4 &= hf_3(x_{1,n} + u_3, x_{2,n} + v_3, x_{3,n} + w_3), \\
&= h \left\{ (x_{1,n} + u_3)^2 + (x_{2,n} + v_3)^2 + 0.1u_n^2 \right\}.
\end{aligned}$$

The adjoint equations are:

$$\left. \begin{aligned}
\dot{\lambda}_1 &= f_1 = -\lambda_1 \left[-(2 + u) + (x_2 + 0.5) \frac{50}{(x_1 + 2)^2} e^{\frac{25x_1}{x_1+2}} \right] \\
&\quad + \lambda_2 \left[(x_2 + 0.5) \frac{50}{(x_1 + 2)^2} e^{\frac{25x_1}{x_1+2}} \right] - 2\lambda_3 x_1, & \lambda_1(t_f) &= 0, \\
\dot{\lambda}_2 &= f_2 = -\lambda_1 e^{\frac{25x_1}{x_1+2}} - \lambda_2 \left[-1 - e^{\frac{25x_1}{x_1+2}} \right] - 2\lambda_3 x_2, & \lambda_2(t_f) &= 0, \\
\dot{\lambda}_3 &= f_3 = 0, & \lambda_3(t_f) &= 1.
\end{aligned} \right\} \quad (8.2.2)$$

Using the Runge-Kutta 4th order method for numerical solutions of (8.2.2) we get,

$$\lambda_{1,n} = \lambda_{1,n+1} + \frac{1}{6} \{k_1 + 2k_2 + 2k_3 + k_4\},$$

$$\lambda_{2,n} = \lambda_{2,n+1} + \frac{1}{6} \{z_1 + 2z_2 + 2z_3 + z_4\},$$

$$\lambda_{3,n} = \lambda_{3,n+1} + \frac{1}{6} \{y_1 + 2y_2 + 2y_3 + y_4\},$$

where,

$$\begin{aligned} k_1 &= -hf(\lambda_{1,n+1}, \lambda_{2,n+1}, \lambda_{3,n+1}), \\ &= -h \left\{ -\lambda_{1,n+1} \left(-(2 + u_{n+1}) + (x_{2,n+1} + 0.5) \frac{50}{(x_{1,n+1} + 2)^2} e^{\frac{25x_{1,n+1}}{x_{1,n+1}+2}} \right) \right. \\ &\quad \left. + \lambda_{2,n+1} \left[(x_{2,n+1} + 0.5) \frac{50}{(x_{1,n+1} + 2)^2} e^{\frac{25x_{1,n+1}}{x_{1,n+1}+2}} \right] \right. \\ &\quad \left. - 2\lambda_{3,n+1}x_{1,n+1} \right\}, \end{aligned}$$

$$\begin{aligned} z_1 &= -hf_2(\lambda_{1,n+1}, \lambda_{2,n+1}, \lambda_{3,n+1}), \\ &= -h \left\{ -\lambda_{1,n+1} e^{\frac{25x_{1,n+1}}{x_{1,n+1}+2}} - \lambda_{2,n+1} \left(-1 - e^{\frac{25x_{1,n+1}}{x_{1,n+1}+2}} \right) \right. \\ &\quad \left. - 2\lambda_{3,n+1}x_{2,n+2} \right\}, \end{aligned}$$

$$\begin{aligned} y_1 &= -hf_3(\lambda_{1,n+1}, \lambda_{2,n+1}, \lambda_{3,n+1}) = 0, \\ k_2 &= -hf_1(\lambda_{1,n+1} + \frac{1}{2}k_1, \lambda_{2,n+1} + \frac{1}{2}z_1, \lambda_{3,n+1} + \frac{1}{2}y_1), \\ &= -h \left\{ -(\lambda_{1,n+1} + \frac{1}{2}k_1) \left(-(2 + u_{n+1}) + (x_{2,n+2} + 0.5) \right. \right. \\ &\quad \left. \frac{50}{(x_{1,n+1} + 2)^2} e^{\frac{25x_{1,n+1}}{x_{1,n+1}+2}} \right) + (\lambda_{2,n+1} + \frac{1}{2}z_1) \\ &\quad \left((x_{2,n+1} + 0.5) \frac{50}{(x_{1,n+1} + 2)^2} e^{\frac{25x_{1,n+1}}{x_{1,n+1}+2}} \right) \\ &\quad \left. - 2 \left[\lambda_{3,n+1} + \frac{1}{2}y_1 \right] x_{1,n+1} \right\}, \end{aligned}$$

$$\begin{aligned} z_2 &= -hf_2(\lambda_{1,n+1} + \frac{1}{2}k_1, \lambda_{2,n+1} + \frac{1}{2}z_1, \lambda_{3,n+1} + \frac{1}{2}y_1), \\ y_2 &= -hf_3(\lambda_{1,n+1} + \frac{1}{2}k_1, \lambda_{2,n+1} + \frac{1}{2}z_1, \lambda_{3,n+1} + \frac{1}{2}y_1) = 0, \\ k_3 &= -hf_1(\lambda_{1,n+1} + \frac{1}{2}k_2, \lambda_{2,n+1} + \frac{1}{2}z_2, \lambda_{3,n+1} + \frac{1}{2}y_2), \\ &= -h \left\{ -(\lambda_{1,n+1} + \frac{1}{2}k_2) \left(-(2 + u_{n+1}) + (x_{2,n+1} + 0.5) \right. \right. \\ &\quad \left. \frac{50}{(x_{1,n+1} + 2)^2} e^{\frac{25x_{1,n+1}}{x_{1,n+1}+2}} \right) + (\lambda_{2,n+1} + \frac{1}{2}z_2) ((x_{2,n+1} + 0.5) \\ &\quad \left. \frac{50}{(x_{1,n+1} + 2)^2} e^{\frac{25x_{1,n+1}}{x_{1,n+1}+2}} \right) - 2(\lambda_{3,n+1} + \frac{1}{2}y_2)x_{1,n+1} \right\}, \end{aligned}$$

$$\begin{aligned}
z_3 &= -hf_2(\lambda_{1,n+1} + \frac{1}{2}k_2, \lambda_{2,n+1} + \frac{1}{2}z_2, \lambda_{3,n+1} + \frac{1}{2}y_2), \\
&= -h \left\{ -(\lambda_{1,n+1} + \frac{1}{2}k_2)e^{\frac{25x_{1,n+1}}{x_{1,n+1}+2}} - (\lambda_{2,n+1} + \frac{1}{2}z_2) \right. \\
&\quad \left. (-1 - e^{\frac{25x_{1,n+1}}{x_{1,n+1}+2}}) - 2(\lambda_{3,n+1} + \frac{1}{2}y_2)x_{2,n+1} \right\}, \\
y_3 &= -hf_3(\lambda_{1,n+1} + \frac{1}{2}k_2, \lambda_{2,n+1} + \frac{1}{2}z_2, \lambda_{3,n+1}) = 0,
\end{aligned}$$

$$\begin{aligned}
k_4 &= -hf_1(\lambda_{1,n+1} + k_3, \lambda_{2,n+1} + z_3, \lambda_{3,n+1} + y_3), \\
&= -h \left\{ -(\lambda_{1,n+1} + k_3) \left(-(2 + u_{n+1}) + (x_{2,n+1} + 0.5) \right. \right. \\
&\quad \left. \left. \frac{50}{(x_{1,n+1} + 2)^2} e^{\frac{25x_{1,n+1}}{x_{1,n+1}+2}} \right) + (\lambda_{2,n+1} + z_3) ((x_{2,n+1} + 0.5) \right. \\
&\quad \left. \left. \frac{50}{(x_{1,n+1} + 2)^2} e^{\frac{25x_{1,n+1}}{x_{1,n+1}+2}} \right) - 2(\lambda_{3,n+1} + y_3)x_{1,n+1} \right\},
\end{aligned}$$

$$\begin{aligned}
z_4 &= -hf_2(\lambda_{1,n+1} + k_3, \lambda_{2,n+1} + z_3, \lambda_{3,n+1} + y_3), \\
&= -h \left\{ -(-\lambda_{1,n+1} + k_3)e^{\frac{25x_{1,n+1}}{x_{1,n+1}+2}} - (\lambda_{2,n+1} + z_3) \right. \\
&\quad \left. \left(-1 - e^{\frac{25x_{1,n+1}}{x_{1,n+1}+2}} \right) - 2(\lambda_{3,n+1} + y_3)x_{2,n+1} \right\}, \\
y_4 &= -hf_3(\lambda_{1,n+1} + k_3, \lambda_{2,n+1} + z_3, \lambda_{3,n+1} + y_3).
\end{aligned}$$

8.3 Results and Discussion

Here before presenting the results, it is worth mentioning that this problem was found to have two distinct local minima and this was confirmed when all the seven different conjugate gradients and hybrid conjugate gradient methods were tested.

In most cases when u_0 was selected in $1.0 \leq u_0 \leq 1.8$, the optimal J^* converged to one of the minima and when u_0 was selected in $2.0 \leq u_0 \leq 3.0$ it converged to the other one.

8.3.1 Gradient method

The algorithm for the gradient in function space applied to problem 5 is the same as that described in chapter 4 section 4.4.1. Here,

$$g = -\lambda_1(x_1 + 0.25) + 0.2\lambda_3u.$$

The efficiency of the method is examined by varying the critical parameters, ϵ , N and u_0 . Here the stopping conditions are taken as $\text{Acc} (\leq 10^{-2}, \leq 10^{-4}$ and $\leq 10^{-6})$, also the best Acc possible for each method, throughout this chapter.

Table (8.3.1), shows the effect of ϵ and N in obtaining the best optimal J with $u_0 = 1.4$, which is the best starting control. The best J^* obtained was

0.24446096 to 8 decimal places, with $\|g\| = 7.840 \times 10^{-7}$ and the corresponding parameter values, $\epsilon = 4.0$, $N = 78$, or $N = 156$ and $m = 18$. Choosing ϵ in the range $1.0 \leq \epsilon \leq 4.0$, produced consistent results for N , sufficiently large, i.e., 78. For $N = 40$, the stopping condition of $\leq 10^{-6}$ was satisfied for $\epsilon = 1, 2$, and 3, but not for $\epsilon = 4$ and 5, provided enough iterations were carried out. and finally for N too small, say 10, numerical instability occurs.

Similarly Table (8.3.2), shows the effect of ϵ and N in obtaining the second local minimum. Here by taking the best initial control $u_0 = 2.0$, the best J^* obtained was 0.13327228 to 8 decimal places, with $\|g\| = 7.168 \times 10^{-3}$ and the corresponding parameter values $\epsilon = 0.17$, $N = 78$, or $N = 156$ and $m = 95$. Choosing ϵ in the range $0.1 \leq \epsilon \leq 0.17$, along with sufficient large N , say, 78, can produce consistent results with the best possible Acc ($\leq 7.2 \times 10^{-3}$), that can be obtained. By taking ϵ too large say, 0.18 or 0.2, even when the number of iterations are increased after a certain amount no improvement on J^* is obtained. For smaller N 's say 40 and 10, their best Acc can be achieved with $\epsilon = 0.16$. Here selecting N as 156 as opposed to 78 may result in slightly better minimum J^* for some ϵ 's but at the cost of more computing time.

Tables (8.3.3), (8.3.4) (8.3.5) and (8.3.6), show the effect of u_0 on J^* , when the best N is selected, i.e., $N = 78$, for Acc $\leq 10^{-2}$, ≤ 0.072 and $\leq 10^{-6}$. Here the best Acc that we could achieve for some u_0 's in the range $2.0 \leq u_0 \leq 3.0$ was Acc ≤ 0.0072 .

From Tables (8.3.3), (8.3.5) and (8.3.6), it can be seen that taking u_0 in the range $1.0 \leq u_0 \leq 1.8$, produces fairly similar results in achieving the optimal J , for the first local minimum with the required Acc's. It can be seen from the tables that selecting $u_0 = 1.4$, in most cases, can lead to a better minimum J^* with a better Acc than other starting controls.

From Tables (8.3.3) and (8.3.4), it can be seen that when u_0 is in the range $2.0 \leq u_0 \leq 3.0$, selecting $u_0 = 2.0$, can lead to a better minimum J^* , with a better Acc in fewer iterations, for the second local minimum. Various aspects of the effect of critical parameters are shown graphically in Figures (8.3.1) to (8.3.8). Figure (8.3.1), effect of m on u , with $m = 0, 2$ and 18, for $N = 78$, $u_0 = 1.4$ and $\epsilon = 4.0$. As can be seen there are minor differences between the curves of $m = 2$ and $m = 18$. Figure (8.3.2), effect of N on u , with $u_0 = 1.4$, $N = 40$ and 78 respectively, with their best corresponding (ϵ, m) as (2.0, 36) and (4.0, 18). As can be seen there is a little difference in the curvature of the plots, between $N = 40$ and $N = 80$.

For $N = 78$, at approximate time of 0.15, the slope of the curve is downward for the rest of the time, where as for $N = 40$, at the approximate time of 0.18, the slope of the curve goes up, until approximate time of 0.2, and then, it comes down steady, similar to the curve of $N = 78$.

Figure (8.3.3), effect of ϵ on u , with $\epsilon = 4.0$ and 5.0, with $N = 78$, $u_0 = 1.8$ and $m = 5.0$.

Here, we can see some differences, between the curvature, behaviour of $\epsilon = 4.0$ and $\epsilon = 5.0$, but the pattern of behaviour are fairly similar.

Figure (8.3.4), effect of u_0 on optimal J with $u_0 = 1.0, 1.4$ and 1.8, where, $N = 78$ and $\epsilon = 4.0$. As can be seen from the plots, as m , increases, the value of optimal J for all starting controls converges to the same value.

Figure (8.3.5), effect of m on u , with $m = 1, 20$ and 95, with $u_0 = 2.0$, $\epsilon =$

0.17 and $N = 78$. As m increases, the curve of control tends steady towards zero, along time axis.

Figure (8.3.6), effect of N on control, with $u_0 = 2.0$, $N = 10$ and 78 , respectively, with their best corresponding (ϵ, m) as $(0.16, 103)$ and $(0.17, 95)$. As can be seen from the graphs, the behaviour of the curves are fairly similar, apart from the fact that with $N = 78$, the curve of u , started at approximately, $u = 4.4$, where as with $N = 10$, it started at $u = 3.4$.

Figure (8.3.7), effect of ϵ on u , of using $\epsilon = 0.17$ and 0.2 , with $N = 78$, $u_0 = 2.0$ and $m = 5$. The behaviour of the curves, for both $\epsilon = 0.17$ and $\epsilon = 0.2$, are fairly similar, with the exception that, from approximate time of 0.1 , onwards the slope of curve of $\epsilon = 0.2$ is nearer to zero, then the slope of $\epsilon = 0.17$. Figure (8.3.8), effect of u_0 on optimal J , with $u_0 = 2.0, 2.4$ and $3, 0$, where $N = 78$ and $\epsilon = 0.17$. Here also as m increases the value of optimal J for all starting controls converges towards the same value.

In view of the above results, for both the local minimas, the correct choice of initial control can give a better J^* , with a more accurate Acc. The interaction between ϵ and N , shows that for the first local minima, the best combination exists, with fairly large ϵ , and sufficiently large N , i.e., $\epsilon = 4.0$ and $N = 78$. Also here for the first local minima, taking u_0 in the range $1.0 \leq u_0 \leq 1.8$ could produce consistent results even for higher accuracy of $\text{Acc} \leq 10^{-6}$.

For the second local minima, the best combination for achieving optimal J^* is with small ϵ and sufficiently large N , i.e., $\epsilon = 0.17$ and $N = 78$. But for this minima, although for lower accuracy of, $\text{Acc} \leq 10^{-2}$, taking u_0 in the range $2.0 \leq u_0 \leq 3.0$, could produce consistent results, but for higher $\text{Acc} \leq 0.0072$, not all the u_0 's in that range could achieve it, so it shows that special care should be taken, in selecting the initial control for higher Acc.

8.3.2 Steepest descent

The algorithm for steepest descent applied to the problem 5 is the same as that given in chapter 2, section, 2.2.2. The line search techniques used for this method is linear search at constant step, which was described in chapter 3, section 3.4.1.

Here again we investigate the effect of step length factor, integration step and initial control in the solution.

Table (8.3.7), shows the effect of ϵ and N in achieving the optimal J , with $u_0 = 1.0$, which is the best starting control to obtain this. The best J^* was found to be 0.24446096 to 8 decimal places, with $\|g\| = 3.053 \times 10^{-7}$, and the corresponding parameter values, $\epsilon = 6.0$, $N = 78$ or 156 and $m = 18$. Choosing ϵ in the range $3.0 \leq \epsilon \leq 6.0$, can produce consistent results, providing N is sufficiently large, i.e., 78 . With N small, i.e., 40 taking ϵ , relatively smaller, say 3.0 , would produce consistent results, but when ϵ is taken larger, there would be no more improvement in J , after a few iterations. Also, when N is too small, say, 10 , numerical instability occurs. Similarly Table (8.3.8), shows the effect of ϵ and N , in obtaining the second local minimum. By taking the best initial control $u_0 = 2.4$, the best minimum J^* obtained, was 0.13317500 to 8 decimal places, with $\|g\| = 3.483 \times 10^{-7}$, and the corresponding parameter values, $\epsilon = 0.5$, $N = 78$ or 156 and $m = 76$.

Here for N large, say 78, taking ϵ as 0.5, produces the best optimal J^* , in fewer iterations with a better Acc. For relatively smaller N , i.e., 40, taking ϵ in the range $0.1 \leq \epsilon \leq 0.6$, can produce consistent results. But when N is too small say, 10, it results in numerical instability. Here also selecting N as 156 as opposed to 78 may result in slightly better minimum J^* for some ϵ 's but at the cost of more computing time.

Tables (8.3.9), (8.3.10) and (8.3.11), show the effect of u_0 on J^* , when the best N is selected, i.e., $N = 78$, for $\text{Acc} \leq 10^{-2}$, $\text{Acc} \leq 10^{-4}$ and $\text{Acc} \leq 10^{-6}$. From Table (8.3.9), it can be seen that when u_0 is in the range $0.9 \leq u_0 \leq 1.8$, to obtain $\text{Acc} \leq 10^{-2}$, selecting $U_0 = 1.4$, results in a better minimum J^* .

Also when u_0 is selected in the range $2.0 \leq u_0 \leq 3.0$, we get two different sets of results, for $\text{Acc} \leq 10^{-2}$, i.e., taking $2.0 \leq u_0 \leq 2.4$ results in finding the second local minimum and taking $2.6 \leq u_0 \leq 3.0$, results in finding the first local minimum.

Table (8.3.11), shows that for better accuracy, i.e., $\text{Acc} \leq 10^{-6}$, the best starting control in order to find the best first local minimum J^* in fewer iterations can be selected as 1.0, also in order to find the second optimal local minimum J^* , the best starting control can be selected as $u_0 = 2.4$. Here we can also see that some starting controls would fail to achieve this Acc.

Various aspects of the effect of m, N, ϵ and u_0 are shown graphically in Figures (8.3.9) to (8.3.16). Figure (8.3.9), effect of m on u , with $m = 0, 2$ and 18, where $N = 78, u_0 = 1.0$ and $\epsilon = 6.0$. Here as can be seen from the graphs, there are some minor differences, between the curves of $m = 2$ and $m = 18$, but, the pattern of the curvature for both are fairly similar, with the difference that, for $m = 18$, the finishing touch is $u = 0$ at time 0.78, where as for $m = 2$, it, gets near to it, but does not quite touch it.

Figure (8.3.10), effect of N on u , with $N = 40$ and 78, respectively, with their best corresponding parameters, $\epsilon = 3.0, m = 43$ and $\epsilon = 6.0, m = 18$ where $u_0 = 1.0$. Here also, the pattern of the curves get more similar as time increases, and there are some differences in the early stages of time, as can be seen from the curves. Figure (8.3.11), effect of ϵ on control, of using step length factors, $\epsilon = 6.0$ and 7.0, with, $N = 78, u_0 = 1.0$ and $m = 2$. Here also, some differences can be observed from the curves, between $\epsilon = 6.0$ and $\epsilon = 7.0$, but generally the behaviour of the curves are similar, with some minor differences, e.g., at time 0.78, the curve of control for $\epsilon = 6.0$ is nearer to zero than, that of $\epsilon = 7.0$.

Figure (8.3.12), effect of u_0 on optimal J with $u_0 = 1.0, 1.4$ and 1.8 respectively, with their best corresponding ϵ 's of 6.0, 6.0 and 5.0 and $N = 78$. As can be seen from the graph, by increasing m , the value of optimal J , for all starting controls, converges to the same one.

Figure (8.3.13), effect of m on u , with $m = 0, 10$ and 76, where $m = 78, u_0 = 2.4$ and $\epsilon = 0.5$. As can be seen as m increases, the slope of the curve of control is downward, towards zero.

Figure (8.3.14), effect of N on u , with $N = 78, \epsilon = 0.5, m = 26$ and $N = 40, \epsilon = 0.3, m = 76$ where $u_0 = 2.4$. Here there is not much difference, between the two curves of $N = 40$ and 78, apart from the fact that, with $N = 78$, the curve touches zero, at time 0.78, where as with $N = 78$, it gets

near zero, but does not touch it.

Figure (8.3.15), effect of ϵ on u , with $\epsilon = 0.5$ and 0.1 , where $N = 78$, $u_0 = 2.4$ and $m = 2$. Here the slope of the curve, $\epsilon = 0.5$ is downward, steady, and at time 0.6, onwards, it is quite near zero, where as, although the slope of $\epsilon = 0.1$, is also downward, but for a short period of time, say up to 0.2, then it converges towards, 2.0 for the control, not zero, for the rest of the time.

Figure (8.3.16), effect of u_0 on optimal J for different values of m , when $N = 78$ and $\epsilon = 0.5$. Here, the value of optimal J , for $u_0 = 2.4$ and 3.0 , as m increases, converge similarly, towards approximately, 0.13, whereas, with $u_0 = 2.0$, the value converges towards, 0.25.

The above results, show that a proper choice of initial control is an important factor in achieving the best J^* in fewer iterations, since for higher Acc purposes, taking distant u_0 's may never achieve this.

The interaction, between ϵ and N , shows that for the first local optimal minimum, the best combination is achieved, with N , large enough and ϵ , in the range $3.0 \leq \epsilon \leq 6.0$. Also for the second local minimum, the best combination is achieved, with N sufficiently large and ϵ in the range $0.1 \leq \epsilon \leq 0.5$.

8.3.3 Fletcher-Reeves

The algorithm for the Fletcher-Reeves, method applied to problem 5, is as described in chapter 2, section 2.4. The line search technique is the same as that for steepest descent in this chapter section 8.3.2. The calculation of the norms are as described in chapter 2, section 2.10.1. The gradient g is in the way obtained in section 8.3.1.

Table (8.3.12), shows the effect of ϵ and N in achieving the minimum J^* , with the best initial control $u_0 = 1.0$. The best J^* was found to be 0.24446096 to 8 decimal places, with $\|g\| = 8.109 \times 10^{-7}$, and the corresponding parameter values of $\epsilon = 6.0$, $N = 78$ or 156 and $m = 13$. Choosing ϵ in the range $3.0 \leq \epsilon \leq 6.0$, can produce consistent results, providing N is sufficiently large enough, i.e., 78. Taking N , smaller, i.e., 40, with $\epsilon = 3.0$, produces better results in terms of Acc and J , than when ϵ is taken in the range $4.0 \leq \epsilon \leq 7.0$. Here also, when N is too small say 10, numerical, instability occurs. Similarly, Table (8.3.13), shows the effect of ϵ and N , in obtaining the second local minimum, by taking the best initial control, $u_0 = 2.4$, the best minimum J^* , obtained was 0.13317414 to 8 decimal places, with $\|g\| = 8.535 \times 10^{-4}$, with the corresponding parameter values, $\epsilon = 0.12$, $N = 78$ or 156 and $m = 28$. Here for N large enough, say 78, taking ϵ , in the range $0.1 \leq \epsilon \leq 0.13$, can produce consistent results. When N is too small, say 10, taking ϵ , in the range $0.1 \leq \epsilon \leq 0.12$, results in numerical instability, but taking $\epsilon = 0.13$, produces consistent results. Here also selecting N as 200 as opposed to 100 may result in slightly better minimum J^* for some ϵ 's but at the cost of more computing time.

Tables (8.3.14), (8.3.15), (8.3.16) and (8.3.17), show the effect of u_0 on J^* , when the best N is selected, i.e., $N = 78$, for $\text{Acc} \leq 10^{-2}$, $\text{Acc} \leq 10^{-3}$, $\text{Acc} \leq 10^{-4}$ and $\text{Acc} \leq 10^{-6}$. Here the $\text{Acc} \leq 10^{-3}$, was the best that could be achieved for some of the u_0 's in the range $2.0 \leq u_0 \leq 3.0$. From Table (8.3.14), it can be seen that, when u_0 is in the range $0.9 \leq u_0 \leq 1.8$,

to obtain $\text{Acc} \leq 10^{-2}$, selecting $u_0 = 1.4$, would result in a better value for J^* , than other u_0 's for the first local minima. Also when u_0 is selected in the range $2.0 \leq u_0 \leq 3.0$, to obtain $\text{Acc} \leq 10^{-2}$, selecting $u_0 = 2.4$, would result in a better J^* for the second local minima.

Table (8.3.15), show that only a few u_0 's can achieve the $\text{Acc} \leq 10^{-3}$, i.e., $u_0 = 2.4$ or 2.8 , which the best performance in this case was with $u_0 = 2.4$, in obtaining the best J^* in fewer iterations and also function evaluations.

Tables (8.3.16), shows in order to obtain $\text{Acc} \leq 10^{-4}$, starting controls in the range $0.9 \leq u_0 \leq 1.8$, could all achieve that for the first local minima, but none of the starting controls in the range $2.0 \leq u_0 \leq 3.0$ could achieve the required Acc for the second local minima.

Table (8.3.17), shows similarly for $\text{Acc} \leq 10^{-6}$, that taking u_0 in the range $0.9 \leq u_0 \leq 1.8$ could all achieve the first local minima, and the best result with fewer number of iterations and function evaluations was obtained with $u_0 = 1.0$, but for the second local minima none of the u_0 's in the range $2.0 \leq u_0 \leq 3.0$, could achieve the required accuracy.

Various aspects of effect of m, N, ϵ and u_0 are shown graphically figures (8.3.17) to (8.3.24). Figure (8.3.17), effect of m on u , with $m = 0, 2$ and 18 , where $u_0 = 1.0, N = 78$ and $\epsilon = 6.0$. Here as can be seen there are some difference, between various m 's, but as m increases, the pattern of behaviour gets closer to each other.

Figure (8.3.18), effect of N on u , with $N = 40, \epsilon = 3.0, m = 6$ and $N = 78, \epsilon = 6.0, m = 6$, where starting control is 1.0 . Here also as can be seen, there are some differences in the behaviour of the curves of control between $N = 40$ and $N = 78$, in the early stages, i.e., from time 0.0 to approximately 0.35 , and from there to the end, the curves of control behave similarly.

Figure (8.3.19), effect of ϵ on u , with $u_0 = 1.0, N = 78$ and $m = 2$. Here also, although the pattern of behaviour of the curves of control are similar, but some differences, exist from approximate time of 0.15 onwards, and it can be seen that at final time of 0.78 , the curve of control with $\epsilon = 6.0$, is nearer to zero, than $\epsilon = 7.0$. Figure (8.3.20), effect of u_0 on optimal J , for different values of m , with $u_0 = 1.0, \epsilon = 6.0, u_0 = 1.4, \epsilon = 6.0$ and $u_0 = 1.8, \epsilon = 2.0$, where $N = 78$. Here as m increases, the value of optimal J for all starting controls, converges to the same one.

Figure (8.3.21), effect of m on u , of $u_0 = 2.4, N = 78$ and $\epsilon = 0.12$. As can be seen, as m increases, the slope of curve of control tends towards zero along time axis.

Figure (8.3.22), effect of N on u , with $N = 10, \epsilon = 0.13$ and $N = 78, \epsilon = 0.12$, with $u_0 = 2.4$ and $m = 4$. As can be seen, there are some differences, between the curves of $N = 10$ and $N = 78$. At time 0.0 , the curve of control for $N = 10$, starts at approximately $u = 3.9$, whereas for $N = 78$, it starts at approximately $u = 3.6$ and at final time, the curve of control for $N = 10$, is nearer to zero than $N = 78$. Figure (8.3.23), effect of ϵ on u , with $N = 78, u_0 = 2.4$ and $m = 4$. Here also there are some difference, between the curves of $\epsilon = 0.12$ and $\epsilon = 0.15$. At time 0.0 , the curve of control for $\epsilon = 0.12$, starts at approximately $u = 3.6$, where as, for $\epsilon = 0.15$, it starts at approximately 3.0 , and at final time the curve of control for $\epsilon = 0.15$ is nearer to zero than

$\epsilon = 0.12$.

Figure (8.3.24), effect of u_0 on optimal J for different values of m , when $N = 78$, $u_0 = 2.0$ with $\epsilon = 0.18$, $u_0 = 2.4$, with $\epsilon = 0.12$ and finally $u_0 = 3.0$ with $\epsilon = 0.10$. As can be seen from the plots, by increasing m the value of optimal J , for all starting controls tends to the same one. The above results, show that a proper choice of initial control is an important factor in achieving the best J^* , in fewer iterations and function evaluations, and also for higher Acc requirements, e.g., $\text{Acc} \leq 10^{-6}$, that not all starting controls could satisfy them.

The interaction, between ϵ and N , shows that for the first local optimal, the best combination is achieved with N , large enough and ϵ , in the range $3.0 \leq \epsilon \leq 6.0$. Also for the second local minimum, the best J^* is obtained with N , sufficiently large and ϵ in the range $0.1 \leq \epsilon \leq 0.13$.

8.3.4 Polak-Ribière

The algorithm for Polak-Ribière method can be found in chapter 2, section 2.5. The line search technique and the calculation of the norms are as FR in this chapter section 8.3.4. The gradient g , also is, as obtained in section 8.3.1. Table (8.3.18), shows the effect of ϵ and N in achieving the minimum J^* , for the first local minima, with $u_0 = 1.0$, which is the best initial control to be selected. The best J^* was found to be 0.24446096 to 8 decimal places, with $\|g\| = 9.308 \times 10^{-7}$, and the corresponding parameter values $\epsilon = 6.0$, $N = 78$, or $N = 156$ and $m = 20$. Choosing ϵ in the range $3.0 \leq \epsilon \leq 6.0$, can produce, consistent results, providing N , is sufficiently large enough, e.g., $N = 78$. When ϵ is too large say 7.0, there will be no more improvement after a certain number of iterations for J , in this case after 5 iterations. When N is smaller, i.e., 40, taking $\epsilon = 3.0$, will produce better results, in terms of Acc and J , than when ϵ is taken in the range $4.0 \leq \epsilon \leq 7.0$. When N is too small, say 10, numerical, instability occurs.

Similarly Table (8.3.19), show, the effect of ϵ and N in obtaining the second local minimum. Here by taking, the best initial control, $u_0 = 2.0$, the best minimum J^* , obtained was 0.13318594, to 8 decimal places, with, $\|g\| = 1.985 \times 10^{-3}$, and the corresponding parameter, values, $\epsilon = 0.18$, $N = 78$ or 156 and $m = 27$. when N is sufficiently large, say, 78 taking ϵ in the range $0.1 \leq \epsilon \leq 0.18$, can produce consistent results, with better Acc. For smaller N , say 40, taking $\epsilon = 0.1$, or $\epsilon = 0.19$, could find J^* , with a better Acc, than other ϵ 's, but it took, a lot more iterations for $\epsilon = 0.1$, to achieve that than, $\epsilon = 0.19$. When N is too small, say, 10, taking ϵ too small, say 0.1, produces consistent results. Tables (8.3.20), (8.3.21), (8.3.22) and (8.3.23), show the effect of u_0 on J , when the best N is selected, i.e., $N = 78$ for $\text{Acc} \leq 10^{-2}$. $\text{Acc} \leq 2 \times 10^{-3}$, $\text{Acc} \leq 10^{-4}$ and $\text{Acc} \leq 10^{-6}$. Here we should note that $\text{Acc} \leq 2.0 \times 10^{-3}$, is the best Acc, that some of the u_0 's could achieve in the range $2.0 \leq u_0 \leq 3.0$. From Table (8.3.20), selecting $u_0 = 1.4$, can produce better J^* , than other starting values, from the required $\text{Acc} \leq 10^{-2}$, for the first local minimum. Also selecting $u_0 = 3.0$, produces better J^* , for the $\text{Acc} \leq 10^{-3}$, with fewer iterations, than other starting values.

Table (8.3.21), show that, the best $\text{Acc} \leq 2.0 \times 10^{-3}$, could be achieved for some initial controls, and not all starting controls could achieve that.

Table (8.3.22), shows that for the first local minima, all the u_0 's in the range $0.9 \leq u_0 \leq 1.8$ could achieve the required $\text{Acc} \leq 10^{-4}$, and the best u_0 , in terms of number of iterations and function evaluations, to achieve that was $u_0 = 1.0$. But none of the starting controls in the range $2.0 \leq u_0 \leq 3.0$, could achieve the required Acc for the second local minima. Also from Table (8.3.23), we can see that, all the starting controls, in the range $0.9 \leq u_0 \leq 1.8$, could achieve the required $\text{Acc} \leq 10^{-6}$, and the best of them in terms of number of iterations and function evaluations, was with $u_0 = 1.0$, but yet again, non of the u_0 's in the range $2.0 \leq u_0 \leq 3.0$, could achieve the required Acc for the second local minima.

Different, aspects of critical parameters, are also shown graphically, in Figures (8.3.25) to (8.3.32). Figure (8.3.25), effects of m on u , when $u_0 = 1.0$, $m = 0, 2$ and 20 , with $N = 78$ and $\epsilon = 6.0$. As can be seen for $m = 2$ and $m = 20$, the pattern of curves for control are fairly similar, and the difference lies towards the end of time, that, the curve with $m = 20$ touches zero, whereas, $m = 2$ does not.

Figure (8.3.26), effect of N on u , with $N = 40$, and 78 respectively, with their best corresponding ϵ 's of 3.0 and 6.0 , where $u_0 = 1.0$ and $m = 5$. Here as can be seen, there are some differences, in the early stages of time, i.e., 0.0 to approximately, 0.2 , between the graphs of controls of $N = 40$ and $N = 78$. Then from there to the final time, they behave similarly.

Figure (8.3.27), effect of ϵ on u , of using step length factors, $\epsilon = 6.0$ and 7.0 , with $N = 78$, $u_0 = 1.0$ and $m = 2$. Here, although the pattern of the curvature of the control is similar, for both ϵ 's, but there are some differences, that could be observed, i.e., at approximate time of 0.15 , the peak of the curve for $\epsilon = 6.0$, is higher than $\epsilon = 7.0$, then from, there, the slope of the cure for both of them are downward, but as it gets near the final time the slope of the curve for $\epsilon = 6.0$, is nearer to zero than $\epsilon = 7$.

Figure (8.3.28), effect of u_0 on optimal J for different values of m , when $N = 78$, $u_0 = 1.0$ with $\epsilon = 6.0$, $u_0 = 1.4$ with $\epsilon = 6.0$ and finally $u_0 = 1.8$ with $\epsilon = 4.0$.

Here as can be seen from the plots, although for the first few iterations, there may be some difference, between, different starting controls, but as m increases, they all converge, to the same value of optimal J .

Figure (8.3.29), effect of m on u , for $u_0 = 2.0$, with $m = 0, m = 5$ and $m = 27$, where $N = 78$ and $\epsilon = 0.18$. As can be seen by increasing m to 27 , the slope of the curve of control is downward along time axis, and as it gets to the final time, it touches zero. Figure (8.3.30), effect of N on u , when $u_0 = 2.0$, with $N = 10$ and 78 , respectively, with their best corresponding parameters, $\epsilon = 0.1, m = 61$ and $\epsilon = 0.18, m = 27$. The pattern of the curvature of the control, for both N 's are similar, except at the beginning that the curve of $N = 10$, starts at approximately $u = 3.4$, whereas the curve of $N = 78$, starts at approximately $u = 4.4$. Figure (8.3.31), effect of ϵ on u , of using the step length factors, $\epsilon = 0.18$ and 0.19 , with $N = 78$, $u_0 = 2.0$ and $m = 2$. Here also, the pattern of the behaviour for both the curves of $\epsilon = 0.18$ and $\epsilon = 0.19$, are similar, except at the start and finish points where for $\epsilon = 0.18$, the curve started at approximately $u = 3.9$ and finished at $u = 0.0$, whereas, for $\epsilon = 0.19$, the curve started at approximately $u = 2.8$ and finished near zero.

Figure (8.3.32), effect of u_0 on optimal J , for different values of m , when $N = 78$, $u_0 = 2.0$ with $\epsilon = 0.18$, $u_0 = 2.4$, with $\epsilon = 0.18$ and finally $u_0 = 3.0$, with $\epsilon = 0.11$. As can be seen, although, for the first few iterations, the value of optimal J , may be different for some starting controls, but as m increases, they all converge, to the same value for optimal J .

In view of the above results, a proper choice of initial control is an important factor in achieving the best J^* , in fewer iterations and function evaluations. also, for higher Acc purposes, selecting distant u_0 's may never achieve that.

The interaction between ϵ and N , shows that for the first local, optimal, the best combination is achieved with N large and ϵ in the range $3.0 \leq \epsilon \leq 6.0$.

Also, for the second local minimum, the best J^* is obtained, with N sufficiently large and ϵ in the range $0.1 \leq \epsilon \leq 0.18$.

8.3.5 Hybrid 1

The algorithm for the hybrid 1 method is described in chapter 2, section 2.6.1. The line search technique used and calculation of the norms are the same as FR, in this chapter section 8.3.3. The gradient g is also as obtained in section 8.3.1. Table (8.3.24), shows the effect of ϵ and N , in achieving the minimum J^* for the first local minima, with the best initial control $u_0 = 1.0$. Here the results are similar to those, obtained for FR, therefore, the best minimum J^* , also the analysis of the results, that were used for FR, could also be applied to H1.

Table (8.3.25), shows, the effect of ϵ and N , in obtaining the second local minimum. Here by taking the best initial control $u_0 = 2.4$, the best minimum J^* obtained was, 0.13318403, to 8 decimal places, with $m = 29$, $\|g\| = 8.832 \times 10^{-4}$, $N = 78$ or 156 and $\epsilon = 0.13$. Here for N large enough say, 78, taking ϵ , in the range $0.1 \leq \epsilon \leq 0.14$, can produce consistent results. when N is smaller, say, 40 taking ϵ in the range, $0.1 \leq \epsilon \leq 0.12$, produces consistent results. When N is too small, say, 10, taking $\epsilon = 0.13$, produces stable results but, for other values of ϵ numerical instability occurs.

Tables (8.3.26), (8.3.27), (8.3.28) and (8.3.29), show that effect of u_0 on J^* , when the best N is selected, i.e., 78, for $\text{Acc} \leq 10^{-2}$, $\text{Acc} \leq 10^{-3}$, $\text{Acc} \leq 10^{-4}$ and $\text{Acc} \leq 10^{-6}$. We should note that $\text{Acc} \leq 10^{-3}$, is the best Acc, that some of the u_0 's could achieve in the range $2.0 \leq u_0 \leq 3.0$. From Tables (8.3.26), (8.3.28) and (8.3.29), we can see that, for the u_0 's in the range, $0.9 \leq u_0 \leq 1.8$, similar results are obtained as FR, therefore the same comments, on selecting the best u_0 's are also applicable for H1.

From Table (8.3.26), when u_0 is selected in the range $2.0 \leq u_0 \leq 3.0$ to obtain $\text{Acc} \leq 10^{-2}$, selecting $u_0 = 2.4$, would result in a better J^* , for the second local minimum. The interesting result here is that, when $u_0 = 2.6$, is selected, the value of J converges to the first local minimum. From Table (8.3.27), it can be seen that in order to obtain $\text{Acc} \leq 10^{-3}$, selecting u_0 as 2.0 or 2.8, will fail this Acc, and selecting $u_0 = 2.6$, will result in finding the first local minimum and $u_0 = 2.2, 2.4, 2.6$ and 3.0 in finding the second local minimum.

From Tables (8.3.28) and (8.3.29), we can see that the $\text{Acc} \leq 10^{-4}$ and $\leq 10^{-6}$, are not achieved for the u_0 's in the range $2.0 \leq u_0 \leq 3.0$.

Various aspects of critical parameters, m, N, ϵ and u_0 are also shown graphically, in Figures (8.3.33) to (8.3.40). The effects of m, N, ϵ for the first local minima from Figures (8.3.33) to (8.3.36), are similar to figures (8.3.17) to (8.3.20) of FR and therefore similar comments on the behaviour of the curves can also be applied to H1.

Figure (8.3.37), effect of m on u , of $u_0 = 2.4$, with $m = 0, 5$ and 29 , where $N = 78$ and $\epsilon = 0.13$. As can be seen from the plots as m increases, the slope of the curve of control along time axis, is downward, and for $m = 29$, it touches zero at time 0.78 .

Figure (8.3.38), effect of N on u , where $N = 10$ and 78 , respectively, with their best ϵ of $\epsilon = 0.13, m = 4$ and $u_0 = 2.4$. Here the pattern of the behaviour of the curve of control for both $N = 10$ and 78 , are similar except the fact that the curve of control for $N = 10$, started at approximately $u = 3.9$ and finished at approximately, $u = 1.2$, whereas the curve of control for $N = 78$ started at approximately $u = 3.6$ and finished at approximately $u = 1.3$.

Figure (8.3.39), effect of ϵ on u , of using the step length factors, $\epsilon = 0.13$ and 0.15 , with $N = 78, u_0 = 2.4$ and $m = 4$. Here, the curve of control for $\epsilon = 0.13$, started at approximately, $u = 3.6$ and finished at approximately, 1.25 , where as, the curve of control, for $\epsilon = 0.15$, started at $u = 3.0$ and finished at approximately $u = 1.6$.

Figure (8.3.40), effect of u_0 on optimal J , for different values of m , when $N = 78, u_0 = 2.0$ with $\epsilon = 0.18, u_0 = 2.4$ with $\epsilon = 0.13$ and finally, $u_0 = 3.0$, with $\epsilon = 0.13$. Here as can be seen from the graphs, although, starting values of optimal J are different for various u_0 's but as the number of iterations, increases, they all converge to the same value of J .

The above results, show that a proper choice of initial control is again an important factor in finding the optimal J , in fewer iterations and better Acc. Also for higher Acc, e.g. $\text{Acc} \leq 10^{-3}$, for the u_0 's in the range $2.0 \leq u_0 \leq 3.0$, selecting distant u_0 , may neve achieve that. when u_0 is selected in the range $0.9 \leq u_0 \leq 1.8$, the same recommendations as FR can be applied to the interaction, between ϵ and N . For the second local minimum, the best J^* is obtained, with N , sufficiently large, and ϵ , in the range $0.1 \leq \epsilon \leq 0.12$.

8.3.6 Angle test hybrid

The algorithm for the angle test hybrid method is as described in chapter 2, section 2.7. The line search technique and the calculation of the norms are the same as FR, in this chapter, section 8.3.3. Also g can be obtained as GFS, in section 8.3.1. For this method, we had to consider, the parameter $\tau > 0$, as well. The method was tested, in the same way as the previous ones, plus the new parameter τ , with the values, $0.01, 0.0001$ and 0.0000001 . Table (8.3.30), shows the effect of ϵ and N , in achieving the minimum J^* with $u_0 = 1.0$, which is the best initial control to be selected, also the value of τ was taken as 0.0000001 . The results obtained are similar to those obtained by FR, therefore the best minimum J^* , for the first local minimum and its

corresponding parameters are also the same. Thus the analysis of the results that were applied to FR are also applicable to ATH.

Table (8.3.31), shows the effect of ϵ and N , in obtaining, the second local minimum. Here by taking the best initial control, $u_0 = 2.4, \tau = 0.0000001$, the best minimum J^* obtained was 0.13317505, to 8 decimal places, with $\|g\| = 9.674 \times 10^{-4}$ and corresponding parameter values, $\epsilon = 0.12$ and $N = 78$ or 156. Here for N , large enough, say, 78, taking ϵ in the range $0.12 \leq \epsilon \leq 0.13$, produces consistent results, when N , is smaller say, 40, taking ϵ in the range $0.1 \leq \epsilon \leq 0.11$, produces consistent results. When N is too small, say 10 only for $\epsilon = 0.13$, we get stable result and for other values of ϵ numerical instability occurs. Here again selecting N as 200 as opposed to 100 may result in slightly better minimum J^* for some ϵ 's but at the cost of more computing time.

Tables (8.3.32), (8.3.33), (8.3.34) and (8.3.35), show the effect of u_0 on J^* , when sufficient large enough N , i.e. 78 is selected and $\tau = 0.0000001$, for $\text{Acc} \leq 10^{-2}$, $\text{Acc} \leq 10^{-3}$, $\text{Acc} \leq 10^{-4}$ and $\text{Acc} \leq 10^{-6}$. Here $\text{Acc} \leq 10^{-3}$ is the best Acc , that could be achieved for the u_0 's in the range $2.0 \leq u_0 \leq 3.0$. When u_0 is in the range $0.9 \leq u_0 \leq 1.8$, the same results are obtained as FR, therefore the same comments and recommendations, could also be applied to ATH.

From Table (8.3.32) for the $\text{Acc} \leq 10^{-2}$, selecting $u_0 = 2.6$, finds reasonably, a better value for minimum J^* for the second local minima compared with others, considering the number of iterations and function evaluations, it has taken. From Table (8.3.33), for $\text{Acc} \leq 10^{-3}$, selecting $u_0 = 2.4$, takes fewer iterations than other starting values, to find minimum J^* for the second local minima. Here, we can see from, Tables (8.3.34) and (8.3.35), that for higher Acc 's, i.e. $\text{Acc} \leq 10^{-4}$ and $\text{Acc} \leq 10^{-6}$, selecting u_0 's in the range, $2.0 \leq u_0 \leq 3.0$, could not achieve those.

Various aspects of critical parameters, are also shown graphically in Figures (8.3.41) to (8.3.48), with $\tau = 0.0000001$.

The effects of m, N, ϵ and u_0 for the first local minima, from figures (8.3.41) to (8.3.44), are similar to figures (8.3.17) to (8.3.20) of FR and therefore similar comments on the behaviour of the curves can also be applied to H1.

Figure (8.3.45), effect of m on u , with $u_0 = 2.4, N = 78, \epsilon = 0.12$ and $m = 0, 5$ and 34. Here as can be seen from the graphs, as m increases, the slope of the curve of control, along time axis, is downward, and it touches zero for $m = 34$.

Figure (8.3.46), effects of N on u , of $u_0 = 2.4$, with $N = 10$ and 78 respectively, with their best corresponding ϵ 's of 0.13 and 0.12, where $m = 4$. As can be seen from the plots, the slope of the curves of control for both $N = 10$ and 78 are similar, with some minor differences at the start and finish points.

Figure (8.3.47), effect of ϵ on u of $u_0 = 2.4$, with $\epsilon = 0.12$ and $\epsilon = 0.15$, when $n = 78$ and $m = 4$. Here also, the slope of the curve of control is downward for both ϵ 's, but there are some differences at the starting and finishing points, i.e., the curve of control for $\epsilon = 0.12$ starts at approximately, $u = 3.6$ and finishes at approximately $u = 1.25$, where as the curve of control for $\epsilon = 0.15$, starts at approximately, $u = 3.0$ and finishes at approximately

$u = 3.0$ and finishes at approximately $u = 1.6$.

Figure (8.3.48), effect of u_0 on optimal J for different, values of m , where $N = 78$, $u_0 = 2.0$ with $\epsilon = 0.13$, $u_0 = 2.4$ with $\epsilon = 0.12$ and finally $u_0 = 3.0$, with $\epsilon = 0.14$. As can be seen from the plots, as the number of iterations increases, the value of optimal J , for all starting controls, tends to the same value.

in view of the above results and comments, we can see that an appropriate choice of u_0 is critical in finding the best minimum J^* in fewer iterations and a better Acc. Here also, selecting distant u_0 's may never achieve higher accuracies at all.

When u_0 is selected in the range $0.9 \leq u_0 \leq 1.8$ the same recommendations for FR can also be suggested for the interaction between ϵ and N .

For the second local minima, the best J^* is obtained, with N large enough and ϵ in the range $0.12 \leq \epsilon \leq 0.13$. Here the effect of τ on obtaining the minimum J^* was practically negligible.

8.3.7 Hybrid 3

The algorithm for Hybrid 3 method can also be found in chapter 2, section 2.8. The calculation of the norms, and line search are the same as for FR in this chapter, section 8.3.3. The gradient, g can also be calculated the same way as for GFS, in this chapter, section 8.3.1. The effect of new parameters, $\lambda, 0$, with the values of 0.01, 0.0001 and 0.0000001, and $\mu < \frac{1}{2}$, with the values of 0.15, 0.35 and 0.4999, were also considered and tested for this method.

Tables (8.3.36), shows the effect of ϵ and N , in achieving the minimum J^* , with $u_0 = 1.0$, which is the the best initial control to be selected, along with the value of $\lambda = 0.0000001$ and $\mu = 0.4999$. The results obtained for the best minimum J^* of the first local minima are the same as those obtained for FR or H1, therefore the same analysis of the results can be true for H3.

Table (8.3.37), shows, the effect of ϵ and N , in obtaining the second local minimum. Here by taking the best initial control $u_0 = 2.4$, $\lambda = 0.0000001$ and $\mu = 0.4999$, the results obtained are the same as those obtained from H1 and therefore the same comments and suggestions for selecting ϵ and N could be applied to H3.

Tables (8.3.38) to (8.3.41) show the effect of u_0 on J^* when $N = 78$, $\lambda = 0.0000001$ and $\mu = 0.4999$, for $\text{Acc} \leq 10^{-2}$, $\text{Acc} \leq 10^{-3}$, $\text{Acc} \leq 10^{-4}$ and $\text{Acc} \leq 10^{-6}$. Here the $\text{Acc} \leq 10^{-3}$, was the best, that could be achieved for u_0 's in the range $2.0 \leq u_0 \leq 3.0$. As can be seen these tables are similar to Tables (8.3.26) to (8.3.29) of H1, and therefore the same comments and recommendations could be applied to H3.

Figure (8.3.49) to (8.3.56), are similar to Figures (8.3.33) to (8.3.40) of H1 with $\lambda = 0.0000001$ and $\mu = 0.4999$, and therefore show similar effects for m, N, ϵ and u_0 . In view of all the above the same recommendations as H1 on selecting u_0 , as well as N and ϵ to find the best minimum J^* , for both local minimas can also be applied to H3. Here, the effect of λ and μ on minimizing J^* was practically negligible.

8.4 Summary of the Results

Summary of the results, could be found in Table (8.4.1), also comparison of the seven optimization techniques, graphically can be seen in figure (8.4.1), for the first local minima, and Figure (8.4.2), for the second local minima, where N is taken as 78, $\tau = 0.0000001$ for ATH, $\lambda = 0.0000001$, $\mu = 0.4999$ for H3. Here in Figure (8.4.1), the comparison of the methods are shown, with their best initial controls, i.e., for GFS, $u_0 = 1.4$ and the other methods with $u_0 = 1.0$. This has also been applied to Figure (8.4.1), with $u_0 = 2.0$ for GFS and PR, and $u_0 = 2.4$ for the other methods.

Considering all the aspects of convergency for minimizing J^* , i.e. the number of iterations, the best Acc, number of function evaluations, computing time and finally numerical stability, the methods performed as follows. At $u_0 = 1.0$, all the methods achieved the high Acc of $\leq 10^{-6}$, and also similar value of J^* up to 7 decimal places, for the first local minima. Taking into consideration the number of iterations, FR, H1, ATH and H3, performed similarly and the best, but it took more number of function evaluations than GFS to achieve that.

At $u_0 = 1.4$, all the methods could achieve the Acc $\leq 10^{-6}$, and produced similar J^* for the first local minima, up to 7 decimal places. Here also taking into consideration the number of iterations FR, H1, ATH and H3, performed similarly and the best, but it took more number of function evaluations than GFS to achieve that.

At $u_0 = 2.0$, considering the number of iterations taken to achieve, the best J^* for the second minima in fewer iterations, to 3 decimal places, FR performed the best, then H1 and H3 performed similarly, then in order of performance PR, ATH, SD and finally GFS. But regardless of number of iterations, the best J^* up to 8 decimal places were achieved in order of preference by ATH, then SD, then PR, then H1 and H3, that performed similarly, then GFS and finally FR.

At $u_0 = 2.4$, considering the number of iterations, the best minimum J^* for the second local minima, up to 3, decimal places, in order of performance were achieved as follows;

1-PR, 2-FR, 3-H1 and H3, 4-ATH, 5-SD and 6-GFS. But, if we consider the best J^* obtained up to 8 decimal places, regardless of m , the order of performance will be as follows;

1-FR, 2-SD, 3-ATH, 4-H1 and H3, 5-PR and 6-GFS.

8.5 Conclusion

In this chapter, we have tested, the problem numerically, using seven gradient and conjugate gradient techniques. The test, has shown that, there are two different local optimal minimum for this problem. Usually by taking u_0 in the range $[1.0, 1.8]$, lead to finding the first local minimum and taking u_0 in the range $[2.0, 3.0]$, confirmed the other. But in some odd cases taking some u_0 's in the range $[2.0, 3.0]$, produced the first local minimum. It happened when we used SD, H1 and H3.

The comparison of the methods were carried out by varying the critical

parameters, such as, ϵ , N and u_0 . In all cases selecting a proper initial control facilitates the procedure of finding the optima J in fewer iterations with better Acc and less computing time. Also always selecting large enough integration step number (N), with appropriate step length factor (ϵ), could help to overcome numerical instability in search for optimal J .

Although for this problem, one may not be able to recommend only one or two techniques, in absolute certainty as the best, but overall in view of convergency, number of iterations, computing time and numerical stability, we could see that the best J^* for the first local minimum could be achieved by FR, H1, ATH and H3, where they all performed similarly, and the best minimum J^* for the second local minimum could be achieved by FR.

TABLE (8.3.1):Results of GFS with varying ϵ and N.

N ϵ	10	40	78	156
1.0	-	J*=0.24473309 m=75 $\ g\ =9.440 \times 10^{-7}$ Cputime<1 NFE=76	J*=0.24446097 m=77 $\ g\ =9.670 \times 10^{-7}$ Cputime=1 NFE=78	J*=0.24446097 m=76 $\ g\ =9.668 \times 10^{-7}$ Cputime=2 NFE=77
2.0	-	J*=0.24473309 m=36 $\ g\ =9.020 \times 10^{-7}$ Cputime<1 NFE=37	J*=0.24446097 m=37 $\ g\ =8.457 \times 10^{-7}$ Cputime<1 NFE=38	J*=0.24446097 m=36 $\ g\ =8.448 \times 10^{-7}$ Cputime=1 NFE=37
3.0	-	J*=0.24473312 m=76 $\ g\ =9.48 \times 10^{-7}$ Cputime<1 NFE=77	J*=0.24446097 m=24 $\ g\ =6.853 \times 10^{-7}$ Cputime<1 NFE=25	J*=0.24446097 m=24 $\ g\ =6.853 \times 10^{-7}$ Cputime=1 NFE=25
4.0	-	J*=0.24473312 m=200 $\ g\ =3.835 \times 10^{-3}$ Cputime<1 NFE=201	J*=0.24446096 m=18 $\ g\ =7.840 \times 10^{-7}$ Cputime<1 NFE=19	J*=0.24446096 m=18 $\ g\ =7.840 \times 10^{-7}$ Cputime=1 NFE=19
5.0	-	J*=0.24494711 m=200 $\ g\ =1.551 \times 10^{-2}$ Cputime=1 NFE=201	J*=0.2446097 m=124 $\ g\ =9.671 \times 10^{-7}$ Cputime=1 NFE=125	J*=0.2446096 m=124 $\ g\ =9.671 \times 10^{-7}$ Cputime=2 NFE=125
6.0	-	J*=0.24495821 m=210 $\ g\ =1.662 \times 10^{-2}$ Cputime=1 NFE=211	J*=0.2446097 m=128 $\ g\ =9.834 \times 10^{-7}$ Cputime=1 NFE=129	J*=0.2446097 m=128 $\ g\ =9.834 \times 10^{-7}$ Cputime=2 NFE=129

TABLE (8.3.2):Results of GFS with varying ϵ and N.

N ϵ	10	40	78	156
0.1	J*=0.13766994 m=163 $\ g\ =7.131 \times 10^{-3}$ Cputime<1 NFE=76	J*=0.13349582 m=161 $\ g\ =7.191 \times 10^{-2}$ Cputime=1 NFE=162	J*=0.13328145 m=161 $\ g\ =7.184 \times 10^{-3}$ Cputime=2 NFE=162	J*=0.13328143 m=161 $\ g\ =7.173 \times 10^{-3}$ Cputime=3 NFE=162
0.16	J*=0.13766484 m=103 $\ g\ =7.025 \times 10^{-3}$ Cputime<1 NFE=103	J*=0.13348867 m=101 $\ g\ =7.017 \times 10^{-3}$ Cputime=1 NFE=102	J*=0.13327397 m=101 $\ g\ =7.172 \times 10^{-3}$ Cputime=1 NFE=102	J*=0.13327397 m=100 $\ g\ =7.171 \times 10^{-3}$ Cputime=2 NFE=101
0.17	J*=0.13789450 m=80 $\ g\ =1.363 \times 10^{-2}$ Cputime<1 NFE=81	J*=0.1347623 m=97 $\ g\ =1.440 \times 10^{-2}$ Cputime<1 NFE=102	J*=0.13327228 m=95 $\ g\ =7.168 \times 10^{-3}$ Cputime=1 NFE=96	J*=0.13327228 m=95 $\ g\ =7.168 \times 10^{-3}$ Cputime=2 NFE=96
0.18	J*=0.13788830 m=75 $\ g\ =1.342 \times 10^{-2}$ Cputime<1 NFE=76	J*=0.13373600 m=74 $\ g\ =1.353 \times 10^{-2}$ Cputime<1 NFE=75	J*=0.13338626 m=80 $\ g\ =1.379 \times 10^{-2}$ Cputime=1 NFE=81	J*=0.13338625 m=80 $\ g\ =1.371 \times 10^{-2}$ Cputime=2 NFE=81
0.2	J*=0.13766229 m=104 $\ g\ =8.108 \times 10^{-3}$ Cputime<1 NFE=105	J*=0.13541269 m=48 $\ g\ =3.436 \times 10^{-2}$ Cputime<1 NFE=49	J*=0.1348734 m=50 $\ g\ =2.991 \times 10^{-2}$ Cputime<1 NFE=51	J*=0.1348734 m=50 $\ g\ =2.991 \times 10^{-2}$ Cputime=1 NFE=51

TABLE (8.3.3):Results of GFS with change in u_0 for $ACC \leq 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	3	0.24444990	8.78792×10^{-3}	4	4.0	<1
1.2	4	0.24448322	5.35426×10^{-3}	5	4.0	<1
1.4	4	0.24448171	5.51124×10^{-3}	5	4.0	<1
1.6	4	0.24448836	6.33551×10^{-3}	5	4.0	<1
1.8	5	0.24449331	6.72935×10^{-3}	6	4.0	<1
2.0	86	0.13337086	9.92293×10^{-3}	87	0.17	1
2.2	89	0.13336717	9.80322×10^{-3}	90	0.17	1
2.4	91	0.13337527	9.99689×10^{-3}	92	0.17	1
2.6	94	0.13336457	9.7239×10^{-3}	95	0.17	1
2.8	96	0.13336681	9.77310×10^{-3}	97	0.17	1
3.0	98	0.13336672	0.76900×10^{-3}	99	0.17	1

TABLE (8.3.4):Results of GFS with change in u_0 for $ACC \leq 0.00722$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
2.0	95	0.13327228	7.16762×10^{-3}	96	0.17	1
2.2	97	0.13327950	7.18269×10^{-3}	98	0.17	1
2.4	—	—	—	—	—	—
2.6	—	—	—	—	—	—
2.8	—	—	—	—	—	—
3.0	—	—	—	—	—	—

TABLE (8.3.5):Results of GFS with change in u_0 for $ACC \leq 10^{-4}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	10	0.24446172	8.79201×10^{-5}	11	4.0	<1
1.2	11	0.24446172	6.24158×10^{-5}	12	4.0	<1
1.4	11	0.24446172	6.12983×10^{-5}	12	4.0	<1
1.6	11	0.24446173	7.26314×10^{-5}	12	4.0	<1
1.8	12	0.24446185	8.60888×10^{-5}	13	4.0	<1
2.0	—	—	—	—	—	—
2.2	—	—	—	—	—	—
2.4	—	—	—	—	—	—
2.6	—	—	—	—	—	—
2.8	—	—	—	—	—	—
3.0	—	—	—	—	—	—

TABLE (8.3.6):Results of GFS with change in u_0 for $ACC \leq 10^{-6}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	18	0.24446097	6.03057×10^{-7}	19	4.0	<1
1.2	18	0.24446097	7.93442×10^{-7}	19	4.0	<1
1.4	18	0.24446096	7.84019×10^{-7}	19	4.0	<1
1.6	18	0.24446097	8.71432×10^{-7}	19	4.0	<1
1.8	20	0.24446096	5.36816×10^{-7}	21	4.0	<1
2.0	—	—	—	—	—	—
2.2	—	—	—	—	—	—
2.4	—	—	—	—	—	—
2.6	—	—	—	—	—	—
2.8	—	—	—	—	—	—
3.0	—	—	—	—	—	—

TABLE (8.3.7):Results of SD with varying ϵ and N.

N ϵ	10	40	78	156
3.0	-	J*=0.24473308 m=43 $\ g\ =9.086 \times 10^{-7}$ Cputime=1 NFE=190	J*=0.24446096 m=31 $\ g\ =9.927 \times 10^{-7}$ Cputime=1 NFE=127	J*=0.24446096 m=30 $\ g\ =9.915 \times 10^{-7}$ Cputime=2 NFE=123
4.0	-	J*=0.24945948 m=2 $\ g\ =1.482 \times 10^{-2}$ Cputime<1 NFE=10	J*=0.24446096 m=25 $\ g\ =2.521 \times 10^{-7}$ Cputime<1 NFE=103	J*=0.24446096 m=25 $\ g\ =2.521 \times 10^{-7}$ Cputime<1 NFE=103
5.0	-	J*=0.24531672 m=1 $\ g\ =0.6623269$ Cputime<1 NFE=6	J*=0.24446096 m=20 $\ g\ =6.337 \times 10^{-7}$ Cputime=1 NFE=81	J*=0.24446096 m=19 $\ g\ =6.242 \times 10^{-7}$ Cputime=2 NFE=78
6.0	-	J*=0.26430095 m=1 $\ g\ =1.50813$ Cputime<1 NFE=6	J*=0.24446096 m=18 $\ g\ =3.053 \times 10^{-7}$ Cputime<1 NFE=73	J*=0.24446096 m=18 $\ g\ =3.053 \times 10^{-7}$ Cputime=1 NFE=73
7.0	-	J*=0.24655780 m=2 $\ g\ =2.408 \times 10^{-2}$ Cputime<1 NFE=9	J*=0.24446096 m=40 $\ g\ =5.029 \times 10^{-7}$ Cputime=1 NFE=178	J*=0.24446096 m=40 $\ g\ =5.029 \times 10^{-7}$ Cputime=2 NFE=178
8.0	-	J*=0.24725561 m=2 $\ g\ =0.9135220$ Cputime<1 NFE=9	J*=0.24446096 m=45 $\ g\ =5.345 \times 10^{-7}$ Cputime=1 NFE=201	J*=0.24446096 m=45 $\ g\ =5.345 \times 10^{-7}$ Cputime=2 NFE=201

TABLE (8.3.8):Results of SD with varying ϵ and N.

N ϵ	10	40	78	156
0.1	-	J*=0.13339408 m=200 $\ g\ =6.052 \times 10^{-6}$ Cputime=4 NFE=990	J*=0.13319629 m=100 $\ g\ =3.048 \times 10^{-3}$ Cputime=4 NFE=540	J*=0.13319619 m=100 $\ g\ =3.013 \times 10^{-3}$ Cputime=8 NFE=551
0.3	-	J*=0.13339239 m=200 $\ g\ =2.841 \times 10^{-6}$ Cputime=4 NFE=1153	J*=0.13319844 m=100 $\ g\ =3.392 \times 10^{-3}$ Cputime=3 NFE=419	J*=0.13319844 m=99 $\ g\ =3.365 \times 10^{-3}$ Cputime=7 NFE=421
0.4	-	J*=0.13339224 m=200 $\ g\ =1.760 \times 10^{-5}$ Cputime=4 NFE=1399	J*=0.13317670 m=100 $\ g\ =3.445 \times 10^{-4}$ Cputime=4 NFE=570	J*=0.13317670 m=100 $\ g\ =3.445 \times 10^{-4}$ Cputime=4 NFE=570
0.5	-	J*=0.13339251 m=200 $\ g\ =2.01 \times 10^{-5}$ Cputime=4 NFE=1214	J*=0.13317500 m=76 $\ g\ =3.483 \times 10^{-7}$ Cputime=3 NFE=450	J*=0.13317500 m=76 $\ g\ =3.483 \times 10^{-7}$ Cputime=7 NFE=450
0.6	-	J*=0.13337754 m=200 $\ g\ =1.948 \times 10^{-7}$ Cputime=3 NFE=817	J*=0.13317687 m=100 $\ g\ =1.061 \times 10^{-3}$ Cputime=3 NFE=410	J*=0.13317687 m=100 $\ g\ =1.061 \times 10^{-3}$ Cputime=7 NFE=410
0.7	-	J*=0.13338211 m=210 $\ g\ =2.03 \times 10^{-6}$ Cputime=4 NFE=953	J*=0.13317723 m=115 $\ g\ =3.512 \times 10^{-3}$ Cputime=4 NFE=614	J*=0.13317721 m=115 $\ g\ =3.51 \times 10^{-3}$ Cputime=8 NFE=602

TABLE (8.3.9):Results of SD with change in u_0 for $ACC \leq 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
0.9	2	0.24449588	5.26189×10^{-3}	9	6.0	<1
1.0	2	0.24455006	7.23059×10^{-3}	9	6.0	<1
1.2	2	0.24465877	9.5650×10^{-3}	9	6.0	<1
1.4	3	0.24446458	3.05202×10^{-3}	13	6.0	<1
1.6	3	0.24451868	4.31404×10^{-3}	13	6.0	<1
1.8	3	0.24480176	9.79006×10^{-3}	13	5.0	<1
2.0	35	0.13342755	7.20393×10^{-3}	153	0.5	1
2.2	34	0.13351020	8.58810×10^{-3}	147	0.5	1
2.4	33	0.13365698	9.31940×10^{-3}	145	0.5	1
2.6	7	0.24468170	9.37026×10^{-3}	50	0.5	<1
2.8	8	0.24456707	7.77207×10^{-3}	57	0.5	<1
3.0	8	0.24456794	8.72479×10^{-3}	57	0.5	<1

TABLE (8.3.10):Results of SD with change in u_0 for $ACC \leq 10^{-4}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
0.9	7	0.24446133	9.36289×10^{-5}	29	6.0	<1
1.0	8	0.24446157	5.86540×10^{-5}	23	6.0	<1
1.2	9	0.24446164	8.95227×10^{-5}	37	6.0	<1
1.4	10	0.24446172	7.95043×10^{-5}	41	6.0	<1
1.6	10	0.24446173	8.19126×10^{-5}	41	6.0	<1
1.8	12	0.24446172	8.79033×10^{-5}	49	5.0	2
2.0	55	0.13317762	5.41984×10^{-5}	259	0.5	2
2.2	56	0.13317508	6.10893×10^{-5}	260	0.5	2
2.4	56	0.13317770	5.49609×10^{-5}	264	0.5	2
2.6	93	0.24446164	9.74202×10^{-5}	417	0.5	3
2.8	93	0.24446163	9.92667×10^{-5}	417	0.5	3
3.0	92	0.24446163	9.93451×10^{-5}	416	0.5	3

TABLE (8.3.11):Results of SD with change in u_0 for $ACC \leq 10^{-6}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
0.9	22	0.24446096	6.35642×10^{-7}	185	6.0	1
1.0	18	0.24446096	3.05272×10^{-7}	73	6.0	<1
1.2	20	0.24446097	9.40090×10^{-7}	81	6.0	1
1.4	20	0.24446096	1.51156×10^{-7}	82	6.0	1
1.6	21	0.24446097	8.34666×10^{-7}	85	6.0	1
1.8	26	0.24446096	7.90829×10^{-7}	105	5.0	1
2.0	—	—	—	—	—	—
2.2	—	—	—	—	—	—
2.4	76	0.13317500	3.48297×10^{-7}	450	0.5	3
2.6	—	—	—	—	—	—
2.8	—	—	—	—	—	—
3.0	—	—	—	—	—	—

TABLE (8.3.12):Results of FR with varying ε and N.

N ε	10	40	78	156
3.0	-	$J^*=0.24585320$ $m=6$ $\ g\ =5.214 \times 10^{-3}$ Cputime<1 NFE=31	$J^*=0.24446097$ $m=19$ $\ g\ =9.001 \times 10^{-7}$ Cputime<1 NFE=77	$J^*=0.24446097$ $m=18$ $\ g\ =8.981 \times 10^{-7}$ Cputime=1 NFE=77
4.0	-	$J^*=0.24945948$ $m=2$ $\ g\ =1.482 \times 10^{-2}$ Cputime<1 NFE=10	$J^*=0.24446096$ $m=17$ $\ g\ =8.314 \times 10^{-7}$ Cputime<1 NFE=70	$J^*=0.24446096$ $m=17$ $\ g\ =8.314 \times 10^{-7}$ Cputime=1 NFE=70
5.0	-	$J^*=0.25531672$ $m=1$ $\ g\ =0.6623269$ Cputime<1 NFE=6	$J^*=0.24446097$ $m=14$ $\ g\ =1.038 \times 10^{-6}$ Cputime<1 NFE=57	$J^*=0.24446097$ $m=13$ $\ g\ =1.032 \times 10^{-6}$ Cputime=1 NFE=55
6.0	-	$J^*=0.26430095$ $m=1$ $\ g\ =1.50813$ Cputime<1 NFE=6	$J^*=0.24446096$ $m=13$ $\ g\ =8.109 \times 10^{-7}$ Cputime<1 NFE=53	$J^*=0.24446096$ $m=13$ $\ g\ =8.109 \times 10^{-7}$ Cputime=1 NFE=53
7.0	-	$J^*=0.24655780$ $m=2$ $\ g\ =2.408 \times 10^{-2}$ Cputime<1 NFE=9	$J^*=0.24455418$ $m=5$ $\ g\ =2.027 \times 10^{-3}$ Cputime<1 NFE=21	$J^*=0.24455418$ $m=5$ $\ g\ =2.027 \times 10^{-3}$ Cputime<1 NFE=21
8.0	-	$J^*=0.24655783$ $m=2$ $\ g\ =2.612 \times 10^{-2}$ Cputime<1 NFE=10	$J^*=0.24455223$ $m=5$ $\ g\ =2.821 \times 10^{-3}$ Cputime<1 NFE=23	$J^*=0.24455223$ $m=5$ $\ g\ =2.821 \times 10^{-3}$ Cputime<1 NFE=23

TABLE (8.3.13):Results of FR with varying ϵ and N.

N ϵ	10	40	78	156
0.1	-	J*=0.13544655 m=14 $\ g\ =4.241 \times 10^{-2}$ Cputime<1 NFE=79	J*=0.13335130 m=19 $\ g\ =8.425 \times 10^{-3}$ Cputime<1 NFE=102	J*=0.13335129 m=19 $\ g\ =8.391 \times 10^{-3}$ Cputime=1 NFE=99
0.11	-	J*=0.13342054 m=30 $\ g\ =4.547 \times 10^{-3}$ Cputime<1 NFE=141	J*=0.13451461 m=16 $\ g\ =3.092 \times 10^{-2}$ Cputime<1 NFE=83	J*=0.13451461 m=16 $\ g\ =3.092 \times 10^{-2}$ Cputime=1 NFE=83
0.12	-	J*=0.13366719 m=15 $\ g\ =4.920 \times 10^{-3}$ Cputime<1 NFE=78	J*=0.13317414 m=28 $\ g\ =8.535 \times 10^{-4}$ Cputime=1 NFE=133	J*=0.13317414 m=28 $\ g\ =8.535 \times 10^{-4}$ Cputime=2 NFE=133
0.13	J*=0.13751878 m=15 $\ g\ =3.398 \times 10^{-2}$ Cputime<1 NFE=78	J*=0.13367046 m=15 $\ g\ =1.034 \times 10^{-2}$ Cputime<1 NFE=75	J*=0.13326260 m=14 $\ g\ =3.053 \times 10^{-2}$ Cputime<1 NFE=76	J*=0.13326258 m=14 $\ g\ =3.033 \times 10^{-2}$ Cputime=1 NFE=76
0.15	J*=0.34263748 m=2 $\ g\ =0.3605352$ Cputime<1 NFE=14	J*=0.25859973 m=4 $\ g\ =0.5172537$ Cputime<1 NFE=25	J*=0.25873187 m=4 $\ g\ =0.5222807$ Cputime<1 NFE=25	J*=0.25872431 m=4 $\ g\ =0.49876431$ Cputime<1 NFE=25

TABLE (8.3.14):Results of FR with change in u_0 for $ACC \leq 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
0.9	2	0.24449082	5.70378×10^{-3}	9	5.0	<1
1.0	2	0.24455006	7.23059×10^{-3}	9	6.0	<1
1.2	2	0.24465877	9.56650×10^{-3}	9	6.0	<1
1.4	3	0.24446587	2.87109×10^{-3}	13	6.0	<1
1.6	3	0.24458319	6.63058×10^{-3}	16	2.0	<1
1.8	4	0.24459255	6.08296×10^{-3}	21	2.0	<1
2.0	12	0.13328645	6.60359×10^{-3}	63	0.18	<1
2.2	14	0.13327442	7.73252×10^{-3}	69	0.12	<1
2.4	18	0.13317278	8.64012×10^{-3}	92	0.12	<1
2.6	20	0.13319968	8.95821×10^{-3}	106	0.12	<1
2.8	13	0.133401854	3.43664×10^{-3}	75	0.11	<1
3.0	13	0.13319774	8.30700×10^{-3}	77	0.10	<1

TABLE (8.3.15):Results of FR with change in u_0 for $ACC \leq 10^{-3}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
2.0	—	—	—	—	—	—
2.2	—	—	—	—	—	—
2.4	28	0.13317414	8.53505×10^{-4}	133	0.12	1
2.6	—	—	—	—	—	—
2.8	30	0.13317522	9.74618×10^{-4}	151	0.11	1
3.0	—	—	—	—	—	—

TABLE (8.3.16):Results of FR with change in u_0 for $ACC \leq 10^{-4}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
0.9	7	0.24446170	9.32802×10^{-5}	29	5.0	<1
1.0	7	0.24446170	4.72530×10^{-5}	29	6.0	<1
1.2	8	0.24446179	5.66357×10^{-5}	33	6.0	<1
1.4	8	0.24446172	9.73333×10^{-5}	33	6.0	<1
1.6	13	0.24446175	7.55836×10^{-5}	59	2.0	<1
1.8	13	0.24446170	9.73085×10^{-5}	60	2.0	<1
2.0	—	—	—	—	—	—
2.2	—	—	—	—	—	—
2.4	—	—	—	—	—	—
2.6	—	—	—	—	—	—
2.8	—	—	—	—	—	—
3.0	—	—	—	—	—	—

TABLE (8.3.17):Results of FR with change in u_0 for $ACC \leq 10^{-6}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
0.9	14	0.24446096	3.47414×10^{-7}	57	5.0	<1
1.0	13	0.24446096	8.10990×10^{-7}	53	6.0	<1
1.2	14	0.24446097	9.52095×10^{-7}	57	6.0	<1
1.4	15	0.24446097	8.91567×10^{-7}	62	6.0	<1
1.6	24	0.24446097	4.80018×10^{-7}	104	2.0	<1
1.8	27	0.24446097	1.36489×10^{-7}	117	2.0	<1
2.0	—	—	—	—	—	—
2.2	—	—	—	—	—	—
2.4	—	—	—	—	—	—
2.6	—	—	—	—	—	—
2.8	—	—	—	—	—	—
3.0	—	—	—	—	—	—

TABLE (8.3.18):Results of PR with varying ϵ and N.

N ϵ	10	40	78	156
3.0	-	J*=0.24452166 m=5 $\ g\ =1.916 \times 10^{-3}$ Cputime<1 NFE=22	J*=0.24446096 m=37 $\ g\ =8.933 \times 10^{-7}$ Cputime=1 NFE=151	J*=0.24446096 m=36 $\ g\ =8.933 \times 10^{-7}$ Cputime=2 NFE=149
4.0	-	J*=0.24945948 m=2 $\ g\ =1.484 \times 10^{-2}$ Cputime<1 NFE=10	J*=0.24446096 m=27 $\ g\ =8.747 \times 10^{-7}$ Cputime=1 NFE=110	J*=0.24446096 m=27 $\ g\ =8.747 \times 10^{-7}$ Cputime=2 NFE=110
5.0	-	J*=0.24531672 m=1 $\ g\ =0.6623269$ Cputime<1 NFE=6	J*=0.24446096 m=25 $\ g\ =7.711 \times 10^{-7}$ Cputime=1 NFE=101	J*=0.24446096 m=25 $\ g\ =7.711 \times 10^{-7}$ Cputime=2 NFE=101
6.0	-	-	J*=0.24446096 m=20 $\ g\ =9.308 \times 10^{-7}$ Cputime<1 NFE=81	J*=0.24446096 m=20 $\ g\ =9.308 \times 10^{-7}$ Cputime<1 NFE=81
7.0	-	J*=0.24655780 m=2 $\ g\ =2.408 \times 10^{-2}$ Cputime<1 NFE=9	J*=0.24446918 m=5 $\ g\ =1.302 \times 10^{-5}$ Cputime<1 NFE=21	J*=0.24446918 m=5 $\ g\ =1.302 \times 10^{-7}$ Cputime<1 NFE=21
8.0	-	J*=0.24656351 m=2 $\ g\ =2.611 \times 10^{-2}$ Cputime<1 NFE=9	J*=0.24446921 m=5 $\ g\ =1.561 \times 10^{-5}$ Cputime<1 NFE=21	J*=0.24446919 m=5 $\ g\ =1.47 \times 10^{-5}$ Cputime<1 NFE=21

TABLE (8.3.19):Results of PR with varying ϵ and N.

N ϵ	10	40	78	156
0.1	J*=0.13758790 m=61 $\ g\ =1.894 \times 10^{-3}$ Cputime<1 NFE=297	J*=0.13340677 m=71 $\ g\ =1.961 \times 10^{-3}$ Cputime=1 NFE=349	J*=0.13318709 m=30 $\ g\ =3.212 \times 10^{-3}$ Cputime=1 NFE=161	J*=0.13318709 m=29 $\ g\ =3.131 \times 10^{-3}$ Cputime=2 NFE=159
0.17	J*=0.13785391 m=20 $\ g\ =1.194 \times 10^{-2}$ Cputime<1 NFE=95	J*=0.1345692 m=14 $\ g\ =5.743 \times 10^{-2}$ Cputime<1 NFE=72	J*=0.13318905 m=27 $\ g\ =8.468 \times 10^{-3}$ Cputime=1 NFE=134	J*=0.13318905 m=27 $\ g\ =8.468 \times 10^{-3}$ Cputime=2 NFE=134
0.18	J*=0.16595612 m=7 $\ g\ =0.2167893$ Cputime<1 NFE=39	J*=0.13377790 m=15 $\ g\ =1.555 \times 10^{-2}$ Cputime<1 NFE=76	J*=0.13318594 m=27 $\ g\ =1.985 \times 10^{-3}$ Cputime=1 NFE=123	J*=0.13318594 m=27 $\ g\ =1.985 \times 10^{-3}$ Cputime=2 NFE=123
0.19	J*=0.13879003 m=10 $\ g\ =2.659 \times 10^{-2}$ Cputime<1 NFE=50	J*=0.13343252 m=25 $\ g\ =3.805 \times 10^{-3}$ Cputime<1 NFE=125	J*=0.13353486 m=13 $\ g\ =4.903 \times 10^{-2}$ Cputime=1 NFE=68	J*=0.13353483 m=13 $\ g\ =4.899 \times 10^{-2}$ Cputime=2 NFE=68
0.2	J*=0.13765053 m=25 $\ g\ =5.091 \times 10^{-3}$ Cputime<1 NFE=120	J*=0.13375513 m=15 $\ g\ =1.072 \times 10^{-2}$ Cputime<1 NFE=78	J*=0.13319346 m=30 $\ g\ =2.921 \times 10^{-2}$ Cputime=1 NFE=136	J*=0.13319346 m=29 $\ g\ =2.921 \times 10^{-2}$ Cputime=2 NFE=134

TABLE (8.3.20):Results of PR with change in u_0 for $ACC \leq 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
0.9	2	0.24449082	5.70378×10^{-3}	9	5.0	<1
1.0	2	0.24455006	7.23059×10^{-3}	9	6.0	<1
1.2	2	0.24465877	9.56650×10^{-3}	9	6.0	<1
1.4	3	0.24446583	3.00946×10^{-3}	13	6.0	<1
1.6	3	0.24451984	4.31510×10^{-3}	13	6.0	<1
1.8	4	0.24454834	4.9951×10^{-3}	17	4.0	<1
2.0	15	0.13329974	9.66388×10^{-3}	74	0.18	<1
2.2	20	0.13323140	9.02354×10^{-3}	92	0.19	<1
2.4	17	0.13333426	8.16410×10^{-3}	87	0.18	<1
2.6	26	0.13317345	8.70550×10^{-3}	136	0.12	1
2.8	23	0.13320670	9.42516×10^{-3}	146	0.10	1
3.0	15	0.13318340	5.81214×10^{-3}	97	0.11	<1

TABLE (8.3.21):Results of PR with change in u_0 for $ACC \leq 10^{-3}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
2.0	27	0.13318554	1.98509×10^{-3}	123	0.18	1
2.2	—	—	—	—	—	—
2.4	—	—	—	—	—	—
2.6	43	0.13317397	1.94184×10^{-3}	207	0.12	1
2.8	42	0.13317129	1.93341×10^{-3}	228	0.10	1
3.0	40	0.13317174	1.98038×10^{-3}	204	0.11	1

TABLE (8.3.22):Results of PR with change in u_0 for $ACC \leq 10^{-4}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
0.9	10	0.24446134	8.90800×10^{-5}	41	5.0	<1
1.0	8	0.24446148	7.29565×10^{-5}	33	6.0	<1
1.2	10	0.24446169	7.69852×10^{-5}	41	6.0	<1
1.4	11	0.24446169	9.44105×10^{-5}	45	6.0	<1
1.6	11	0.24446172	9.35948×10^{-5}	45	6.0	<1
1.8	17	0.24446169	9.86105×10^{-5}	70	4.0	<1
2.0	—	—	—	—	—	—
2.2	—	—	—	—	—	—
2.4	—	—	—	—	—	—
2.6	—	—	—	—	—	—
2.8	—	—	—	—	—	—
3.0	—	—	—	—	—	—

TABLE (8.3.23):Results of PR with change in u_0 for $ACC \leq 10^{-6}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
0.9	21	0.24446096	7.49008×10^{-7}	85	5.0	<1
1.0	20	0.24446096	9.3078×10^{-7}	81	6.0	<1
1.2	23	0.24446096	9.40211×10^{-7}	93	6.0	1
1.4	24	0.24446097	2.41687×10^{-7}	98	6.0	1
1.6	24	0.24446097	2.3042110^{-7}	98	6.0	1
1.8	38	0.24446097	9.56529×10^{-7}	154	4.0	1
2.0	—	—	—	—	—	—
2.2	—	—	—	—	—	—
2.4	—	—	—	—	—	—
2.6	—	—	—	—	—	—
2.8	—	—	—	—	—	—
3.0	—	—	—	—	—	—

TABLE (8.3.24):Results of H1 with varying ϵ and N.

N ϵ	10	40	78	156
3.0	-	$J^*=0.24585320$ $m=6$ $\ g\ =5.214 \times 10^{-3}$ Cputime<1 NFE=31	$J^*=0.24446097$ $m=19$ $\ g\ =9.001 \times 10^{-7}$ Cputime<1 NFE=77	$J^*=0.24446097$ $m=18$ $\ g\ =8.981 \times 10^{-7}$ Cputime=1 NFE=77
4.0	-	$J^*=0.24945948$ $m=2$ $\ g\ =1.482 \times 10^{-2}$ Cputime<1 NFE=10	$J^*=0.24446096$ $m=17$ $\ g\ =8.314 \times 10^{-7}$ Cputime<1 NFE=70	$J^*=0.24446096$ $m=17$ $\ g\ =8.314 \times 10^{-7}$ Cputime=1 NFE=70
5.0	-	$J^*=0.25531672$ $m=1$ $\ g\ =0.6623269$ Cputime<1 NFE=6	$J^*=0.24446097$ $m=14$ $\ g\ =1.038 \times 10^{-6}$ Cputime<1 NFE=57	$J^*=0.24446097$ $m=13$ $\ g\ =1.032 \times 10^{-6}$ Cputime=1 NFE=55
6.0	-	$J^*=0.26430095$ $m=1$ $\ g\ =1.50813$ Cputime<1 NFE=6	$J^*=0.24446096$ $m=13$ $\ g\ =8.109 \times 10^{-7}$ Cputime<1 NFE=53	$J^*=0.24446096$ $m=13$ $\ g\ =8.109 \times 10^{-7}$ Cputime=1 NFE=53
7.0	-	$J^*=0.24655780$ $m=2$ $\ g\ =2.408 \times 10^{-2}$ Cputime<1 NFE=9	$J^*=0.24455418$ $m=5$ $\ g\ =2.027 \times 10^{-3}$ Cputime<1 NFE=21	$J^*=0.24455418$ $m=5$ $\ g\ =2.027 \times 10^{-3}$ Cputime<1 NFE=21
8.0	-	$J^*=0.24655783$ $m=2$ $\ g\ =2.612 \times 10^{-2}$ Cputime<1 NFE=10	$J^*=0.24455223$ $m=5$ $\ g\ =2.821 \times 10^{-3}$ Cputime<1 NFE=23	$J^*=0.24455223$ $m=5$ $\ g\ =2.821 \times 10^{-3}$ Cputime<1 NFE=23

TABLE (8.3.25):Results of H1 with varying ε and N.

N ε	10	40	78	156
0.1	-	J*=0.13339758 m=38 $\ g\ =9.188 \times 10^{-4}$ Cputime<1 NFE=194	J*=0.13318526 m=35 $\ g\ =3.028 \times 10^{-3}$ Cputime=1 NFE=167	J*=0.13318525 m=35 $\ g\ =3.024 \times 10^{-3}$ Cputime=2 NFE=165
0.12	-	J*=0.13339739 m=55 $\ g\ =1.106 \times 10^{-3}$ Cputime<1 NFE=297	J*=0.13318658 m=35 $\ g\ =3.977 \times 10^{-3}$ Cputime=1 NFE=167	J*=0.13318658 m=35 $\ g\ =3.977 \times 10^{-3}$ Cputime=2 NFE=167
0.13	J*=0.13745473 m=25 $\ g\ =5.025 \times 10^{-2}$ Cputime<1 NFE=127	J*=0.13390099 m=15 $\ g\ =1.566 \times 10^{-2}$ Cputime<1 NFE=84	J*=0.13318403 m=29 $\ g\ =8.832 \times 10^{-4}$ Cputime=1 NFE=149	J*=0.13318403 m=29 $\ g\ =8.832 \times 10^{-4}$ Cputime=2 NFE=149
0.14	-	J*=0.13343967 m=15 $\ g\ =1.542 \times 10^{-2}$ Cputime<1 NFE=73	J*=0.13318703 m=30 $\ g\ =8.937 \times 10^{-4}$ Cputime=1 NFE=141	J*=0.13318703 m=29 $\ g\ =8.937 \times 10^{-4}$ Cputime=2 NFE=141
0.15	-	J*=0.13519949 m=9 $\ g\ =5.593 \times 10^{-2}$ Cputime<1 NFE=50	J*=0.13330998 m=20 $\ g\ =2.778 \times 10^{-2}$ Cputime=1 NFE=94	J*=0.13330998 m=20 $\ g\ =2.778 \times 10^{-2}$ Cputime=2 NFE=94

TABLE (8.3.26):Results of H1 with change in u_0 for $ACC \leq 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
0.9	2	0.24449082	5.70378×10^{-3}	9	5.0	<1
1.0	2	0.24455006	7.23059×10^{-3}	9	6.0	<1
1.2	2	0.24465877	9.56650×10^{-3}	9	6.0	<1
1.4	3	0.24446587	2.87109×10^{-3}	13	6.0	<1
1.6	3	0.24458319	6.63058×10^{-3}	16	2.0	<1
1.8	4	0.24459255	6.08296×10^{-3}	21	2.0	<1
2.0	16	0.13320453	6.90606×10^{-3}	77	0.18	<1
2.2	20	0.13323534	6.36870×10^{-3}	95	0.13	<1
2.4	17	0.13318469	5.46456×10^{-3}	89	0.13	<1
2.6	24	0.24460027	7.40841×10^{-3}	150	0.13	1
2.8	21	0.133419467	8.73069×10^{-3}	108	0.12	1
3.0	16	0.13339801	9.78239×10^{-3}	84	0.13	<1

TABLE (8.3.27):Results of H1 with change in u_0 for $ACC \leq 10^{-3}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
2.0	—	—	—	—	—	—
2.2	30	0.13318481	8.74198×10^{-4}	140	0.13	1
2.4	29	0.13318403	8.83155×10^{-4}	149	0.13	1
2.6	30	0.24446444	8.61432×10^{-4}	203	0.13	1
2.8	—	—	—	—	—	—
3.0	33	0.13317961	7.21462×10^{-4}	161	0.13	1

TABLE (8.3.28):Results of H1 with change in u_0 for $ACC \leq 10^{-4}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
0.9	7	0.24446170	9.32802×10^{-5}	29	5.0	<1
1.0	7	0.24446170	4.72530×10^{-5}	29	6.0	<1
1.2	8	0.24446179	5.66357×10^{-5}	33	6.0	<1
1.4	8	0.24446172	9.73333×10^{-5}	33	6.0	<1
1.6	13	0.24446175	7.55836×10^{-5}	59	2.0	<1
1.8	13	0.24446170	9.73085×10^{-5}	60	2.0	<1
2.0	—	—	—	—	—	—
2.2	—	—	—	—	—	—
2.4	—	—	—	—	—	—
2.6	—	—	—	—	—	—
2.8	—	—	—	—	—	—
3.0	—	—	—	—	—	—

TABLE (8.3.29):Results of H1 with change in u_0 for $ACC \leq 10^{-6}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
0.9	14	0.24446096	3.47414×10^{-7}	57	5.0	<1
1.0	13	0.24446096	8.10990×10^{-7}	53	6.0	<1
1.2	14	0.24446097	9.52095×10^{-7}	57	6.0	<1
1.4	15	0.24446097	8.91567×10^{-7}	62	6.0	<1
1.6	24	0.24446097	4.80018×10^{-7}	104	2.0	<1
1.8	27	0.24446097	1.36489×10^{-7}	117	2.0	<1
2.0	—	—	—	—	—	—
2.2	—	—	—	—	—	—
2.4	—	—	—	—	—	—
2.6	—	—	—	—	—	—
2.8	—	—	—	—	—	—
3.0	—	—	—	—	—	—

TABLE (8.3.30):Results of ATH with varying ϵ and N.

N ϵ	10	40	78	156
3.0	-	J*=0.24585320 m=6 $\ g\ =5.214 \times 10^{-3}$ Cputime<1 NFE=31	J*=0.24446097 m=19 $\ g\ =9.001 \times 10^{-7}$ Cputime<1 NFE=77	J*=0.24446097 m=18 $\ g\ =8.981 \times 10^{-7}$ Cputime=1 NFE=77
4.0	-	J*=0.24945948 m=2 $\ g\ =1.482 \times 10^{-2}$ Cputime<1 NFE=10	J*=0.24446096 m=17 $\ g\ =8.314 \times 10^{-7}$ Cputime<1 NFE=70	J*=0.24446096 m=17 $\ g\ =8.314 \times 10^{-7}$ Cputime=1 NFE=70
5.0	-	J*=0.25531672 m=1 $\ g\ =0.6623269$ Cputime<1 NFE=6	J*=0.24446097 m=14 $\ g\ =1.038 \times 10^{-6}$ Cputime<1 NFE=57	J*=0.24446097 m=13 $\ g\ =1.032 \times 10^{-6}$ Cputime=1 NFE=55
6.0	-	J*=0.26430095 m=1 $\ g\ =1.50813$ Cputime<1 NFE=6	J*=0.24446096 m=13 $\ g\ =8.109 \times 10^{-7}$ Cputime<1 NFE=53	J*=0.24446096 m=13 $\ g\ =8.109 \times 10^{-7}$ Cputime=1 NFE=53
7.0	-	J*=0.24655780 m=2 $\ g\ =2.408 \times 10^{-2}$ Cputime<1 NFE=9	J*=0.24455418 m=5 $\ g\ =2.027 \times 10^{-3}$ Cputime<1 NFE=21	J*=0.24455418 m=5 $\ g\ =2.027 \times 10^{-3}$ Cputime<1 NFE=21
8.0	-	J*=0.24655783 m=2 $\ g\ =2.612 \times 10^{-2}$ Cputime<1 NFE=10	J*=0.24455223 m=5 $\ g\ =2.821 \times 10^{-3}$ Cputime<1 NFE=23	J*=0.24455223 m=5 $\ g\ =2.821 \times 10^{-3}$ Cputime<1 NFE=23

TABLE (8.3.31):Results of ATH with varying ϵ and N.

N ϵ	10	40	78	156
0.1	-	J*=0.13338819 m=45 $\ g\ =1.970 \times 10^{-3}$ Cputime<1 NFE=226	J*=0.13508597 m=15 $\ g\ =4.291 \times 10^{-2}$ Cputime<1 NFE=84	J*=0.13508596 m=15 $\ g\ =4.213 \times 10^{-2}$ Cputime=1 NFE=84
0.11	-	J*=0.13339970 m=45 $\ g\ =2.243 \times 10^{-3}$ Cputime=1 NFE=237	J*=0.13927680 m=10 $\ g\ =0.1413501$ Cputime<1 NFE=61	J*=0.13927679 m=10 $\ g\ =0.1413501$ Cputime=1 NFE=61
0.12	-	J*=0.13374761 m=15 $\ g\ =1.707 \times 10^{-2}$ Cputime<1 NFE=87	J*=0.13317505 m=34 $\ g\ =9.674 \times 10^{-4}$ Cputime=1 NFE=168	J*=0.13317505 m=34 $\ g\ =9.674 \times 10^{-4}$ Cputime=2 NFE=168
0.13	J*=0.13768117 m=15 $\ g\ =3.125 \times 10^{-2}$ Cputime<1 NFE=82	J*=0.13666050 m=10 $\ g\ =4.326 \times 10^{-2}$ Cputime<1 NFE=58	J*=0.13317530 m=34 $\ g\ =9.787 \times 10^{-4}$ Cputime=1 NFE=170	J*=0.13317530 m=34 $\ g\ =9.787 \times 10^{-4}$ Cputime=2 NFE=170
0.15	-	J*=0.13629642 m=10 $\ g\ =4.268 \times 10^{-2}$ Cputime<1 NFE=51	J*=0.13381931 m=15 $\ g\ =2.603 \times 10^{-2}$ Cputime<1 NFE=72	J*=0.13381929 m=15 $\ g\ =2.587 \times 10^{-2}$ Cputime=1 NFE=72

TABLE (8.3.32):Results of ATH with change in u_0 for $ACC \leq 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
0.9	2	0.24449082	5.70378×10^{-3}	9	5.0	<1
1.0	2	0.24455006	7.23059×10^{-3}	9	6.0	<1
1.2	2	0.24465877	9.56650×10^{-3}	9	6.0	<1
1.4	3	0.24446587	2.87109×10^{-3}	13	6.0	<1
1.6	3	0.24458319	6.63058×10^{-3}	16	2.0	<1
1.8	4	0.24459255	6.08296×10^{-3}	21	2.0	<1
2.0	22	0.13318247	8.99351×10^{-3}	104	0.13	1
2.2	15	0.13322805	6.92920×10^{-3}	77	0.12	<1
2.4	15	0.13330087	9.66302×10^{-3}	82	0.12	<1
2.6	17	0.13318977	6.79285×10^{-3}	81	0.13	<1
2.8	24	0.13317827	7.67838×10^{-3}	115	0.12	1
3.0	22	0.13317822	9.77898×10^{-3}	112	0.14	1

TABLE (8.3.33):Results of ATH with change in u_0 for $ACC \leq 10^{-3}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
2.0	38	0.13317530	9.93926×10^{-4}	183	0.13	1
2.2	43	0.13317545	9.94395×10^{-4}	236	0.12	1
2.4	34	0.13317505	9.67378×10^{-4}	168	0.12	1
2.6	42	0.13317526	9.93680×10^{-4}	200	0.13	1
2.8	35	0.13317594	9.99775×10^{-4}	162	0.12	1
3.0	42	0.13317584	9.53071×10^{-4}	201	0.14	1

TABLE (8.3.34):Results of ATH with change in u_0 for $ACC \leq 10^{-4}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
0.9	7	0.24446170	9.32802×10^{-5}	29	5.0	<1
1.0	7	0.24446170	4.72530×10^{-5}	29	6.0	<1
1.2	8	0.24446179	5.66357×10^{-5}	33	6.0	<1
1.4	8	0.24446172	9.73333×10^{-5}	33	6.0	<1
1.6	13	0.24446175	7.55836×10^{-5}	59	2.0	<1
1.8	13	0.24446170	9.73085×10^{-5}	60	2.0	<1
2.0	—	—	—	—	—	—
2.2	—	—	—	—	—	—
2.4	—	—	—	—	—	—
2.6	—	—	—	—	—	—
2.8	—	—	—	—	—	—
3.0	—	—	—	—	—	—

TABLE (8.3.35):Results of ATH with change in u_0 for $ACC \leq 10^{-6}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
0.9	14	0.24446096	3.47414×10^{-7}	57	5.0	<1
1.0	13	0.24446096	8.10990×10^{-7}	53	6.0	<1
1.2	14	0.24446097	9.52095×10^{-7}	57	6.0	<1
1.4	15	0.24446097	8.91567×10^{-7}	62	6.0	<1
1.6	24	0.24446097	4.80018×10^{-7}	104	2.0	<1
1.8	27	0.24446097	1.36489×10^{-7}	117	2.0	<1
2.0	—	—	—	—	—	—
2.2	—	—	—	—	—	—
2.4	—	—	—	—	—	—
2.6	—	—	—	—	—	—
2.8	—	—	—	—	—	—
3.0	—	—	—	—	—	—

TABLE (8.3.36):Results of H3 with varying ϵ and N.

N ϵ	10	40	78	156
3.0	-	$J^*=0.24585320$ $m=6$ $\ g\ =5.214 \times 10^{-3}$ Cputime<1 NFE=31	$J^*=0.24446097$ $m=19$ $\ g\ =9.001 \times 10^{-7}$ Cputime<1 NFE=77	$J^*=0.24446097$ $m=18$ $\ g\ =8.981 \times 10^{-7}$ Cputime=1 NFE=77
4.0	-	$J^*=0.24945948$ $m=2$ $\ g\ =1.482 \times 10^{-2}$ Cputime<1 NFE=10	$J^*=0.24446096$ $m=17$ $\ g\ =8.314 \times 10^{-7}$ Cputime<1 NFE=70	$J^*=0.24446096$ $m=17$ $\ g\ =8.314 \times 10^{-7}$ Cputime=1 NFE=70
5.0	-	$J^*=0.25531672$ $m=1$ $\ g\ =0.6623269$ Cputime<1 NFE=6	$J^*=0.24446097$ $m=14$ $\ g\ =1.038 \times 10^{-6}$ Cputime<1 NFE=57	$J^*=0.24446097$ $m=13$ $\ g\ =1.032 \times 10^{-6}$ Cputime=1 NFE=55
6.0	-	$J^*=0.26430095$ $m=1$ $\ g\ =1.50813$ Cputime<1 NFE=6	$J^*=0.24446096$ $m=13$ $\ g\ =8.109 \times 10^{-7}$ Cputime<1 NFE=53	$J^*=0.24446096$ $m=13$ $\ g\ =8.109 \times 10^{-7}$ Cputime=1 NFE=53
7.0	-	$J^*=0.24655780$ $m=2$ $\ g\ =2.408 \times 10^{-2}$ Cputime<1 NFE=9	$J^*=0.24455418$ $m=5$ $\ g\ =2.027 \times 10^{-3}$ Cputime<1 NFE=21	$J^*=0.24455418$ $m=5$ $\ g\ =2.027 \times 10^{-3}$ Cputime<1 NFE=21
8.0	-	$J^*=0.24655783$ $m=2$ $\ g\ =2.612 \times 10^{-2}$ Cputime<1 NFE=10	$J^*=0.24455223$ $m=5$ $\ g\ =2.821 \times 10^{-3}$ Cputime<1 NFE=23	$J^*=0.24455223$ $m=5$ $\ g\ =2.821 \times 10^{-3}$ Cputime<1 NFE=23

TABLE (8.3.37):Results of H3 with varying ϵ and N.

N ϵ	10	40	78	156
0.1	-	$J^*=0.13339758$ $m=38$ $\ g\ =9.188 \times 10^{-4}$ Cputime<1 NFE=194	$J^*=0.13318526$ $m=35$ $\ g\ =3.028 \times 10^{-3}$ Cputime=1 NFE=167	$J^*=0.13318525$ $m=35$ $\ g\ =3.024 \times 10^{-3}$ Cputime=2 NFE=165
0.12	-	$J^*=0.13339739$ $m=55$ $\ g\ =1.106 \times 10^{-3}$ Cputime<1 NFE=297	$J^*=0.13318658$ $m=35$ $\ g\ =3.977 \times 10^{-3}$ Cputime=1 NFE=167	$J^*=0.13318658$ $m=35$ $\ g\ =3.977 \times 10^{-3}$ Cputime=2 NFE=167
0.13	$J^*=0.13745473$ $m=25$ $\ g\ =5.025 \times 10^{-2}$ Cputime<1 NFE=127	$J^*=0.13390099$ $m=15$ $\ g\ =1.566 \times 10^{-2}$ Cputime<1 NFE=84	$J^*=0.13318403$ $m=29$ $\ g\ =8.832 \times 10^{-4}$ Cputime=1 NFE=149	$J^*=0.13318403$ $m=29$ $\ g\ =8.832 \times 10^{-4}$ Cputime=2 NFE=149
0.14	-	$J^*=0.13343967$ $m=15$ $\ g\ =1.542 \times 10^{-2}$ Cputime<1 NFE=73	$J^*=0.13318703$ $m=30$ $\ g\ =8.937 \times 10^{-4}$ Cputime=1 NFE=141	$J^*=0.13318703$ $m=29$ $\ g\ =8.937 \times 10^{-4}$ Cputime=2 NFE=141
0.15	-	$J^*=0.13519949$ $m=9$ $\ g\ =5.593 \times 10^{-2}$ Cputime<1 NFE=50	$J^*=0.13330998$ $m=20$ $\ g\ =2.778 \times 10^{-2}$ Cputime=1 NFE=94	$J^*=0.13330998$ $m=20$ $\ g\ =2.778 \times 10^{-2}$ Cputime=2 NFE=94

TABLE (8.3.38):Results of H3 with change in u_0 for $ACC \leq 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
0.9	2	0.24449082	5.70378×10^{-3}	9	5.0	<1
1.0	2	0.24455006	7.23059×10^{-3}	9	6.0	<1
1.2	2	0.24465877	9.56650×10^{-3}	9	6.0	<1
1.4	3	0.24446587	2.87109×10^{-3}	13	6.0	<1
1.6	3	0.24458319	6.63058×10^{-3}	16	2.0	<1
1.8	4	0.24459255	6.08296×10^{-3}	21	2.0	<1
2.0	16	0.13320453	6.90606×10^{-3}	77	0.18	<1
2.2	20	0.13323534	6.36870×10^{-3}	95	0.13	<1
2.4	17	0.13318469	5.46456×10^{-3}	89	0.13	<1
2.6	24	0.24460027	7.40841×10^{-3}	150	0.13	1
2.8	21	0.133419467	8.73069×10^{-3}	108	0.12	1
3.0	16	0.13339801	9.78239×10^{-3}	84	0.13	<1

TABLE (8.3.39):Results of H3 with change in u_0 for $ACC \leq 10^{-3}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
2.0	—	—	—	—	—	—
2.2	30	0.13318481	8.74198×10^{-4}	140	0.13	1
2.4	29	0.13318403	8.83155×10^{-4}	149	0.13	1
2.6	30	0.24446444	8.61432×10^{-4}	203	0.13	1
2.8	—	—	—	—	—	—
3.0	33	0.13317961	7.21462×10^{-4}	161	0.13	1

TABLE (8.3.40):Results of H3 with change in u_0 for $ACC \leq 10^{-4}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
0.9	7	0.24446170	9.32802×10^{-5}	29	5.0	<1
1.0	7	0.24446170	4.72530×10^{-5}	29	6.0	<1
1.2	8	0.24446179	5.66357×10^{-5}	33	6.0	<1
1.4	8	0.24446172	9.73333×10^{-5}	33	6.0	<1
1.6	13	0.24446175	7.55836×10^{-5}	59	2.0	<1
1.8	13	0.24446170	9.73085×10^{-5}	60	2.0	<1
2.0	—	—	—	—	—	—
2.2	—	—	—	—	—	—
2.4	—	—	—	—	—	—
2.6	—	—	—	—	—	—
2.8	—	—	—	—	—	—
3.0	—	—	—	—	—	—

TABLE (8.3.41):Results of H3 with change in u_0 for $ACC \leq 10^{-6}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
0.9	14	0.24446096	3.47414×10^{-7}	57	5.0	<1
1.0	13	0.24446096	8.10990×10^{-7}	53	6.0	<1
1.2	14	0.24446097	9.52095×10^{-7}	57	6.0	<1
1.4	15	0.24446097	8.91567×10^{-7}	62	6.0	<1
1.6	24	0.24446097	4.80018×10^{-7}	104	2.0	<1
1.8	27	0.24446097	1.36489×10^{-7}	117	2.0	<1
2.0	—	—	—	—	—	—
2.2	—	—	—	—	—	—
2.4	—	—	—	—	—	—
2.6	—	—	—	—	—	—
2.8	—	—	—	—	—	—
3.0	—	—	—	—	—	—

Table (8.4.1): Summary table for the seven methods

method u_0	GFS	SD	FR	PR	H1	ATH	H3
1.0	$m = 18$ $J^* = 0.24446097$ $\ g\ = 6.03057 \times 10^{-7}$ $\varepsilon = 4.0$ NFE = 19 Cputime < 1	$m = 18$ $J^* = 0.24446096$ $\ g\ = 3.05272 \times 10^{-7}$ $\varepsilon = 6.0$ NFE = 73 Cputime < 1	$m = 13$ $J^* = 0.24446096$ $\ g\ = 8.10990 \times 10^{-7}$ $\varepsilon = 6.0$ NFE = 53 Cputime < 1	$m = 20$ $J^* = 0.24446096$ $\ g\ = 9.3078 \times 10^{-7}$ $\varepsilon = 6.0$ NFE = 81 Cputime < 1	$m = 13$ $J^* = 0.24446096$ $\ g\ = 8.10990 \times 10^{-2}$ $\varepsilon = 6.0$ NFE = 53 Cputime < 1	$m = 13$ $J^* = 0.24446096$ $\ g\ = 8.10990 \times 10^{-7}$ $\varepsilon = 6.0$ NFE = 53 Cputime < 1	$m = 13$ $J^* = 0.24446096$ $\ g\ = 8.10990 \times 10^{-7}$ $\varepsilon = 6.0$ NFE = 53 Cputime < 1
1.4	$m = 18$ $J^* = 0.24446096$ $\ g\ = 7.84019 \times 10^{-7}$ $\varepsilon = 4.0$ NFE = 19 Cputime < 1	$m = 20$ $J^* = 0.24446096$ $\ g\ = 1.51156 \times 10^{-7}$ $\varepsilon = 6.0$ NFE = 82 Cputime = 1	$m = 15$ $J^* = 0.24446097$ $\ g\ = 8.91567 \times 10^{-7}$ $\varepsilon = 6.0$ NFE = 62 Cputime < 1	$m = 24$ $J^* = 0.24446097$ $\ g\ = 2.41687 \times 10^{-2}$ $\varepsilon = 6.0$ NFE = 98 Cputime = 1	$m = 15$ $J^* = 0.24446097$ $\ g\ = 8.91567 \times 10^{-7}$ $\varepsilon = 6.0$ NFE = 62 Cputime < 1	$m = 15$ $J^* = 0.24446097$ $\ g\ = 8.91567 \times 10^{-7}$ $\varepsilon = 6.0$ NFE = 62 Cputime < 1	$m = 15$ $J^* = 0.24446097$ $\ g\ = 8.91567 \times 10^{-7}$ $\varepsilon = 6.0$ NFE = 62 Cputime < 1
2.0	$m = 95$ $J^* = 0.13327228$ $\ g\ = 7.16762 \times 10^{-3}$ $\varepsilon = 0.17$ NFE = 96 Cputime = 1	$m = 55$ $J^* = 0.13317762$ $\ g\ = 5.41984 \times 10^{-5}$ $\varepsilon = 0.5$ NFE = 259 Cputime = 2	$m = 12$ $J^* = 0.13328694$ $\ g\ = 6.60359 \times 10^{-3}$ $\varepsilon = 0.18$ NFE = 63 Cputime < 1	$m = 27$ $J^* = 0.13318594$ $\ g\ = 1.98509 \times 10^{-3}$ $\varepsilon = 0.18$ NFE = 123 Cputime = 1	$m = 16$ $J^* = 0.13320453$ $\ g\ = 6.90606 \times 10^{-3}$ $\varepsilon = 0.18$ NFE = 77 Cputime < 1	$m = 38$ $J^* = 0.13317530$ $\ g\ = 9.94926 \times 10^{-4}$ $\varepsilon = 0.13$ NFE = 183 Cputime = 1	$m = 16$ $J^* = 0.13320453$ $\ g\ = 6.90606 \times 10^{-3}$ $\varepsilon = 0.18$ NFE = 77 Cputime < 1
2.4	$m = 91$ $J^* = 0.13337527$ $\ g\ = 9.99689 \times 10^{-3}$ $\varepsilon = 0.17$ NFE = 92 Cputime = 1	$m = 76$ $J^* = 0.13317500$ $\ g\ = 3.48297 \times 10^{-2}$ $\varepsilon = 0.5$ NFE = 450 Cputime = 3	$m = 28$ $J^* = 0.13317414$ $\ g\ = 8.53505 \times 10^{-4}$ $\varepsilon = 0.12$ NFE = 133 Cputime = 1	$m = 17$ $J^* = 0.13333426$ $\ g\ = 8.16410 \times 10^{-3}$ $\varepsilon = 0.18$ NFE = 87 Cputime < 1	$m = 29$ $J^* = 0.13318403$ $\ g\ = 8.83155 \times 10^{-4}$ $\varepsilon = 0.13$ NFE = 149 Cputime = 1	$m = 34$ $J^* = 0.13317505$ $\ g\ = 9.67378 \times 10^{-4}$ $\varepsilon = 0.12$ NFE = 168 Cputime = 1	$m = 29$ $J^* = 0.13318403$ $\ g\ = 8.83155 \times 10^{-4}$ $\varepsilon = 0.13$ NFE = 149 Cputime = 1

GFS

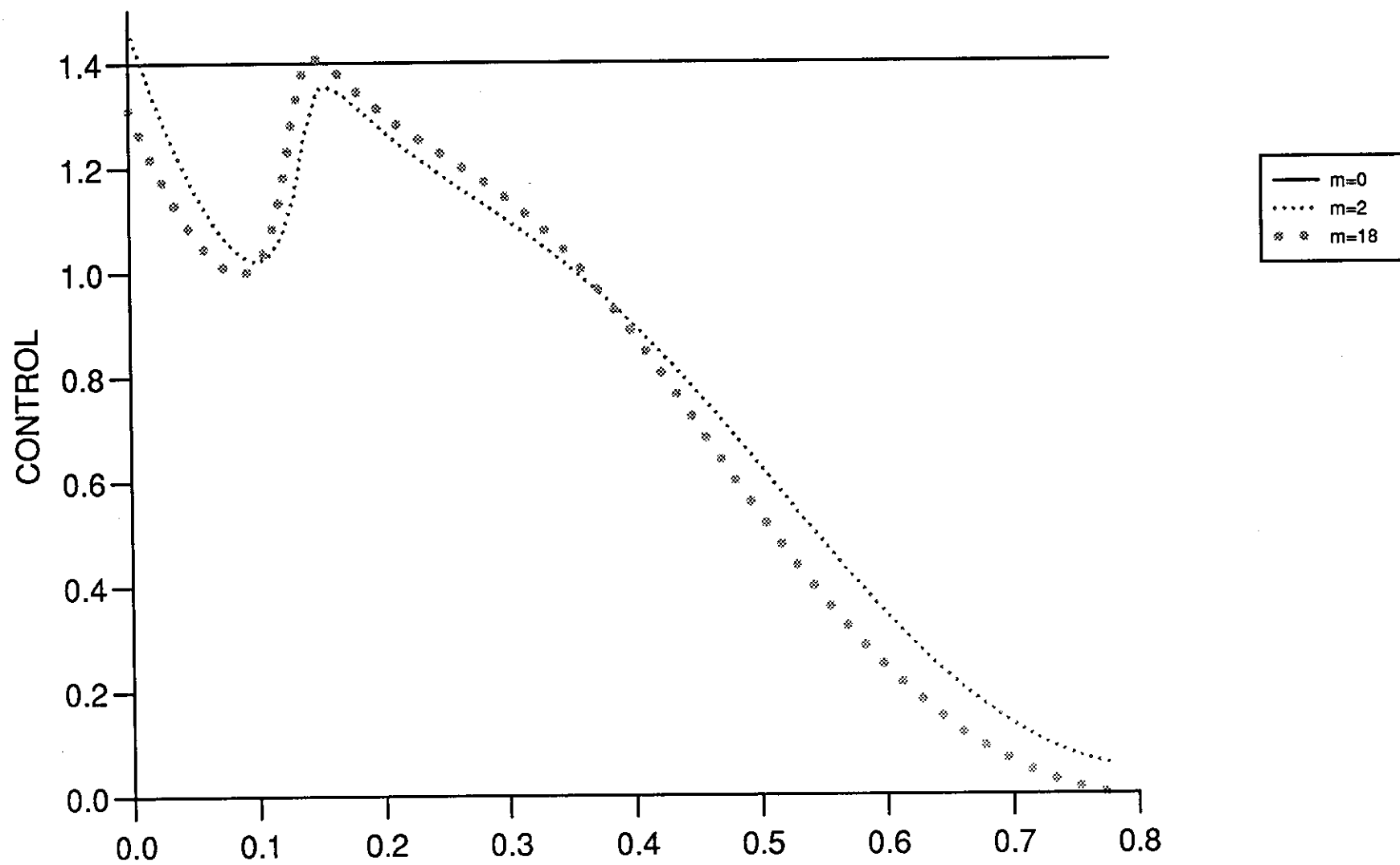


Figure (8.3.1)

GFS

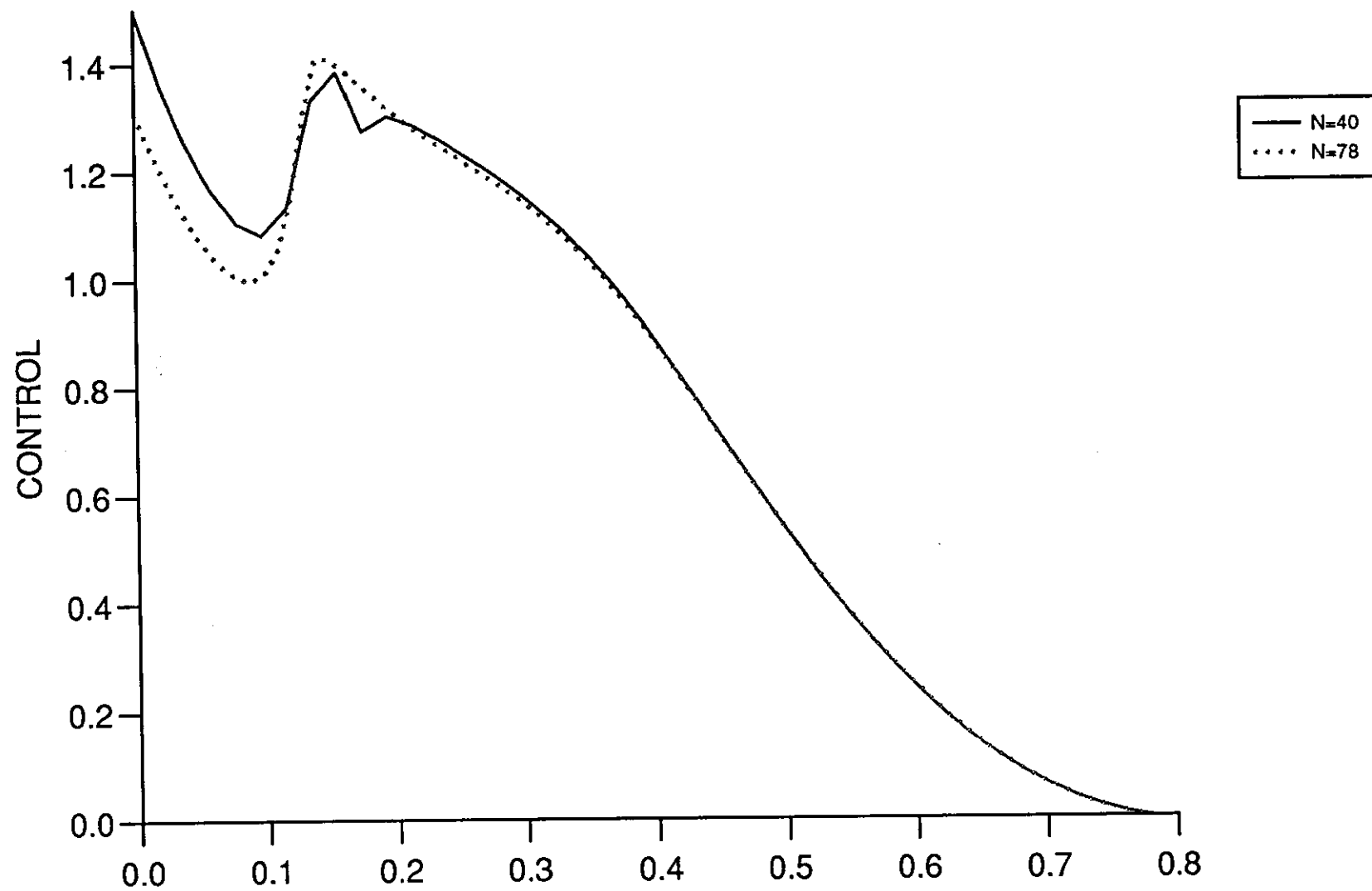


Figure (8.3.2)

GFS

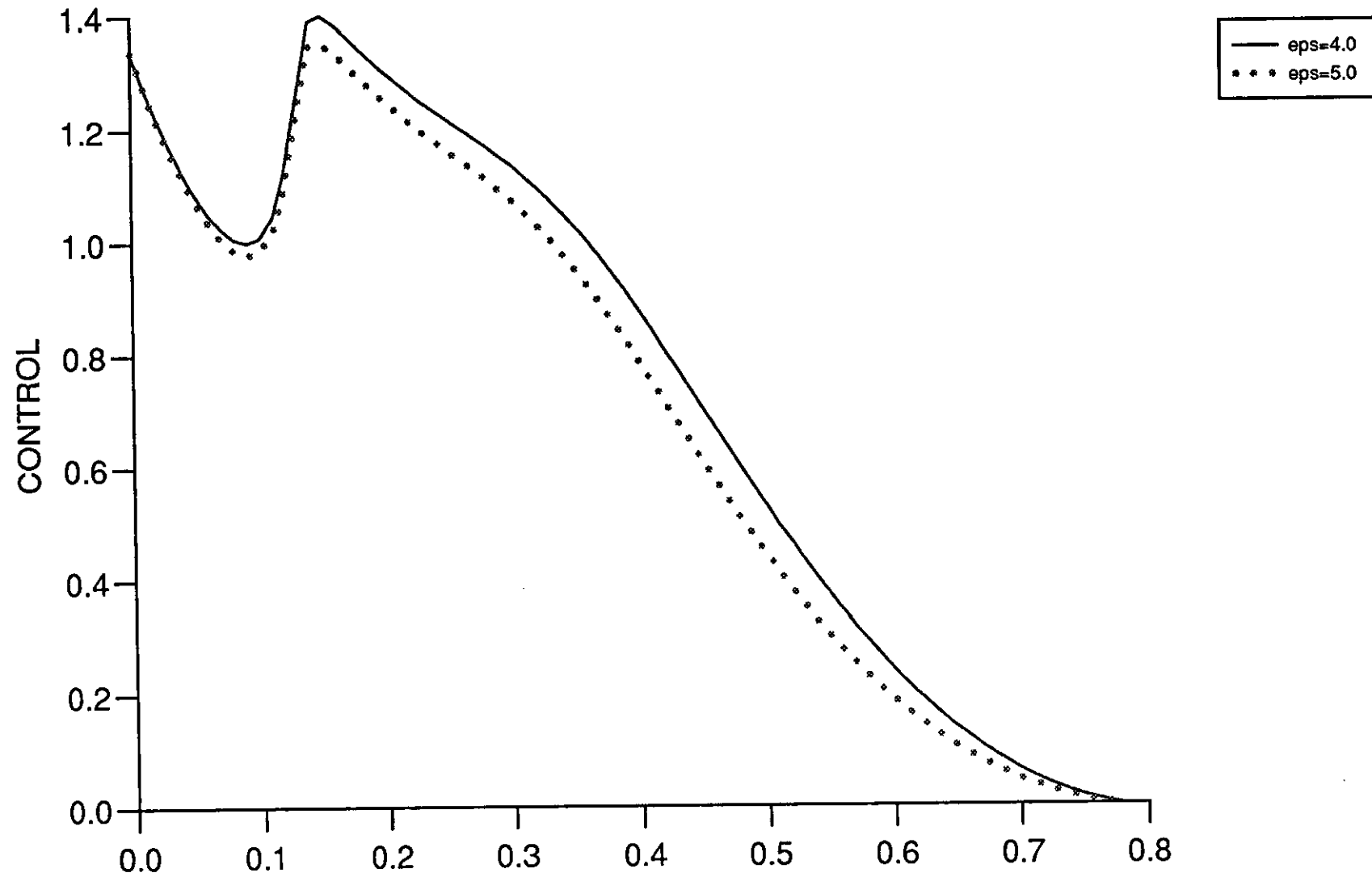


Figure (8.3.3)

GFS

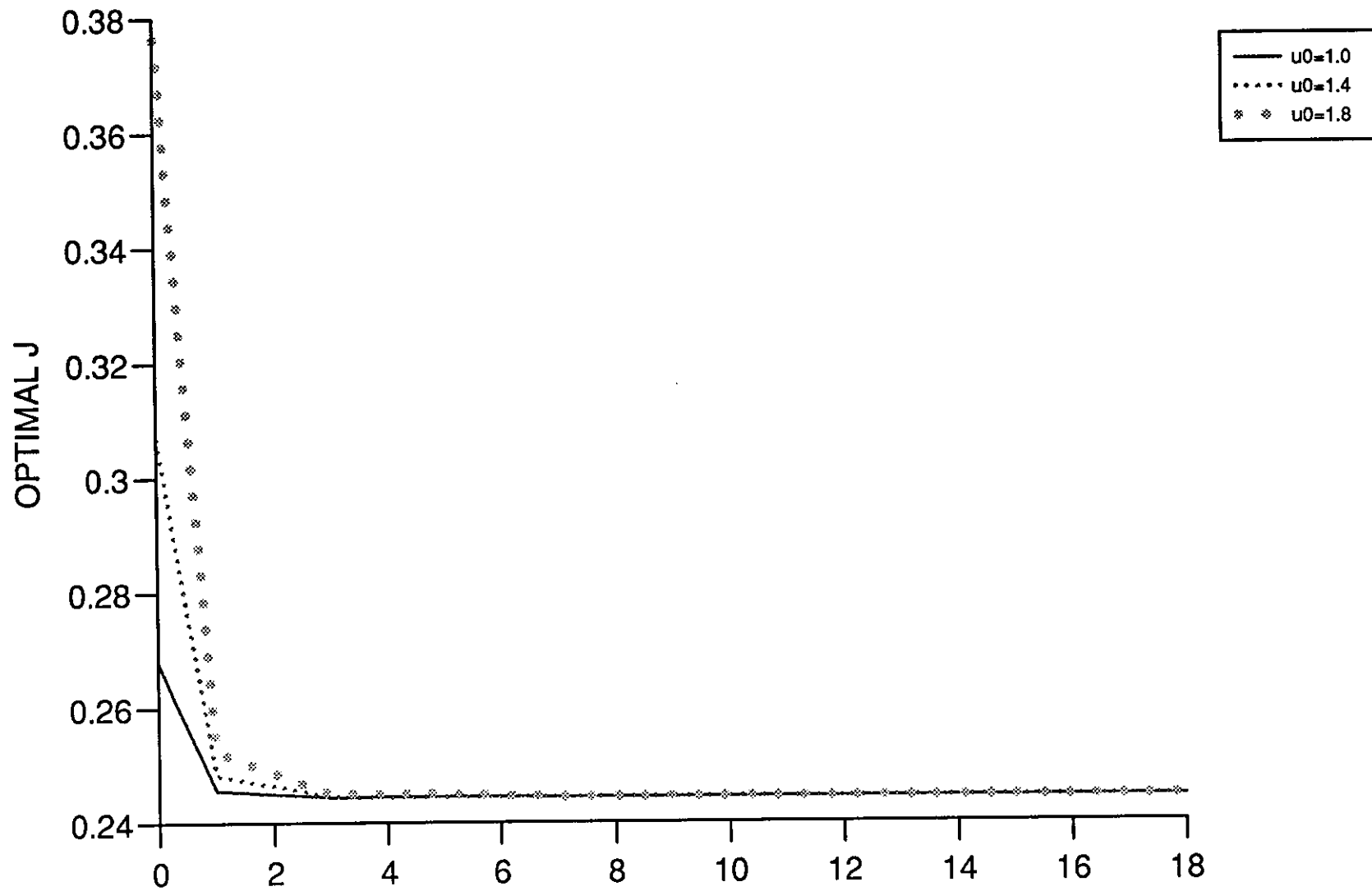


Figure (8.3.4)

GFS

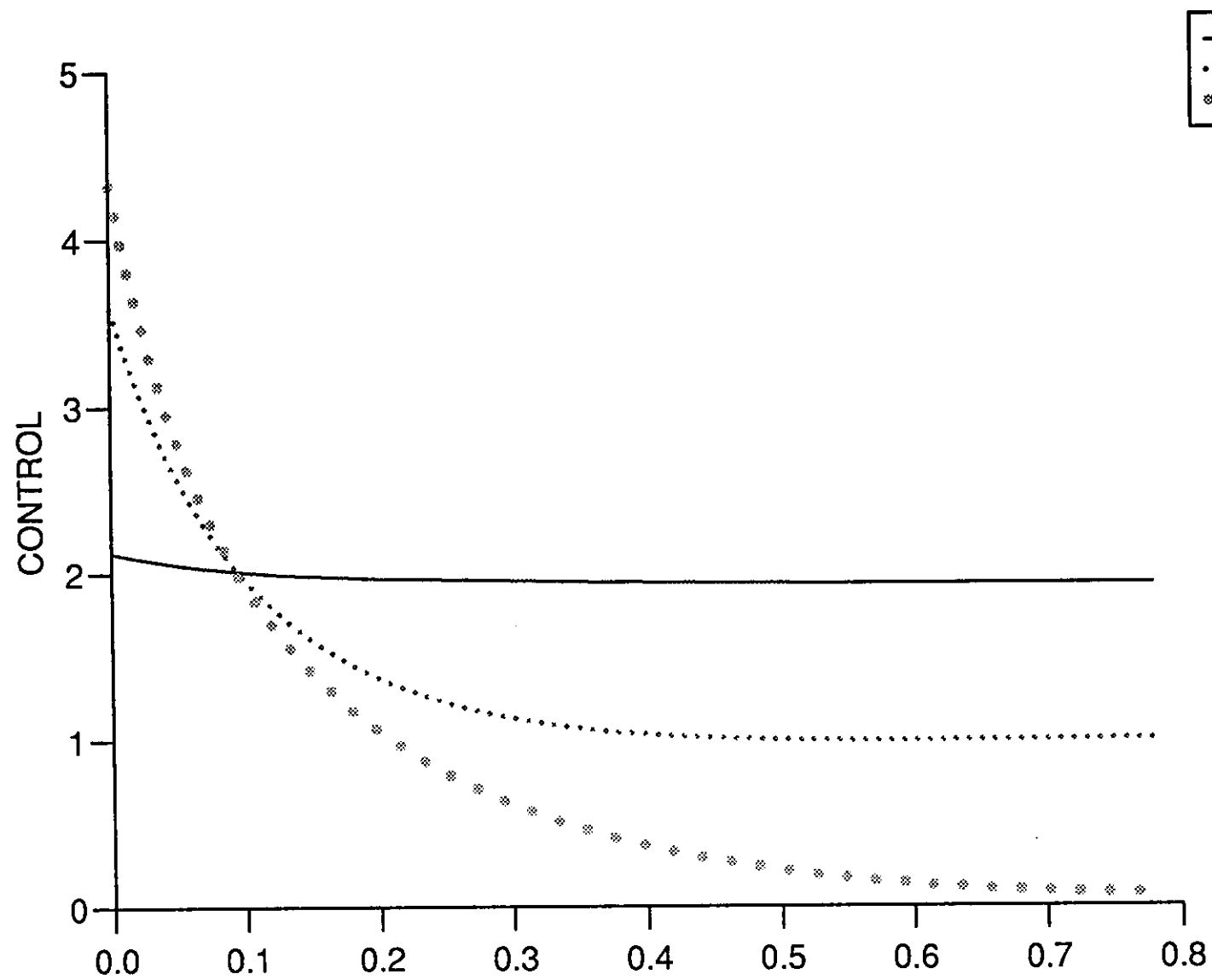


Figure (8.3.5)

GFS

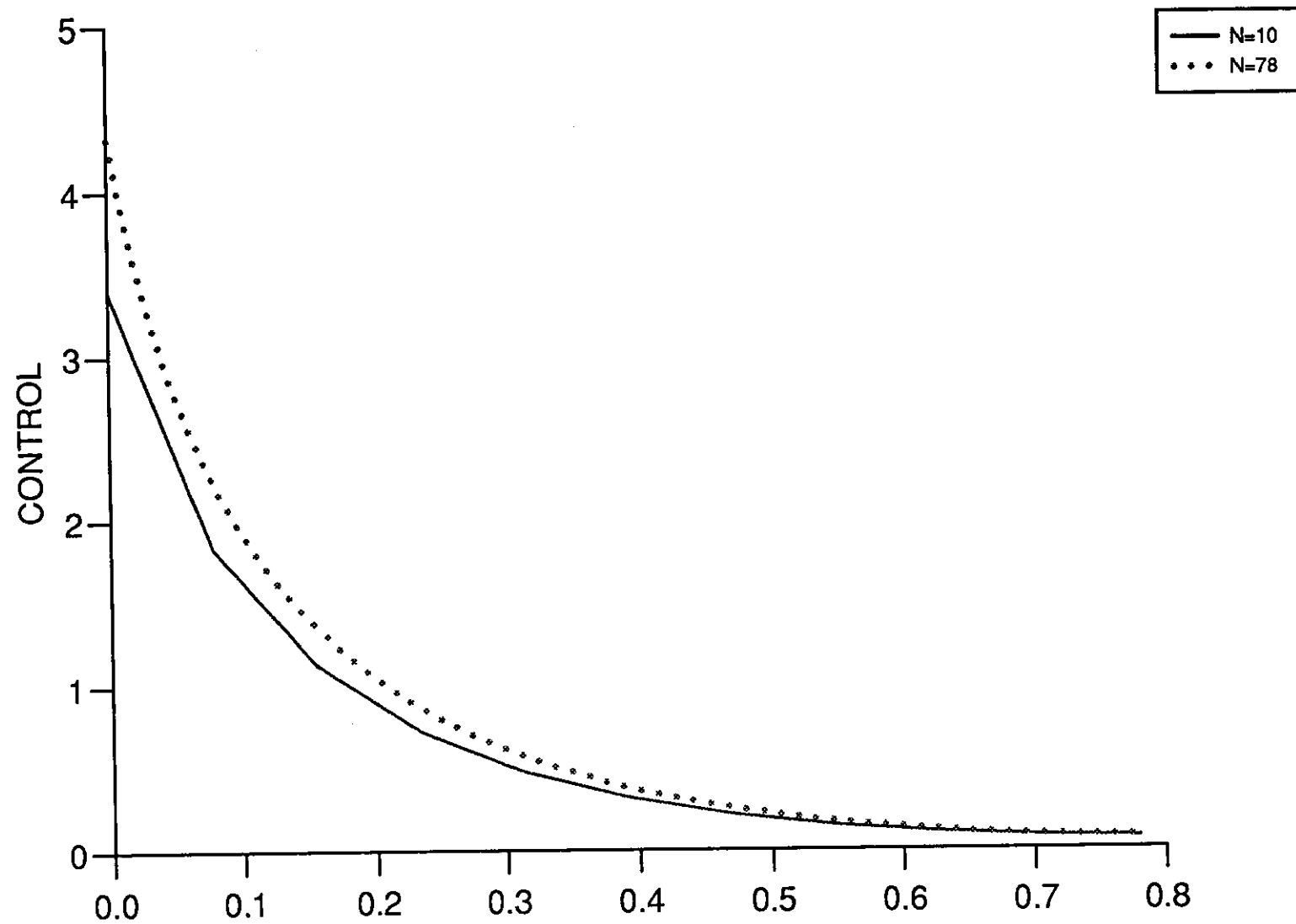


Figure (8.3.6)

GFS

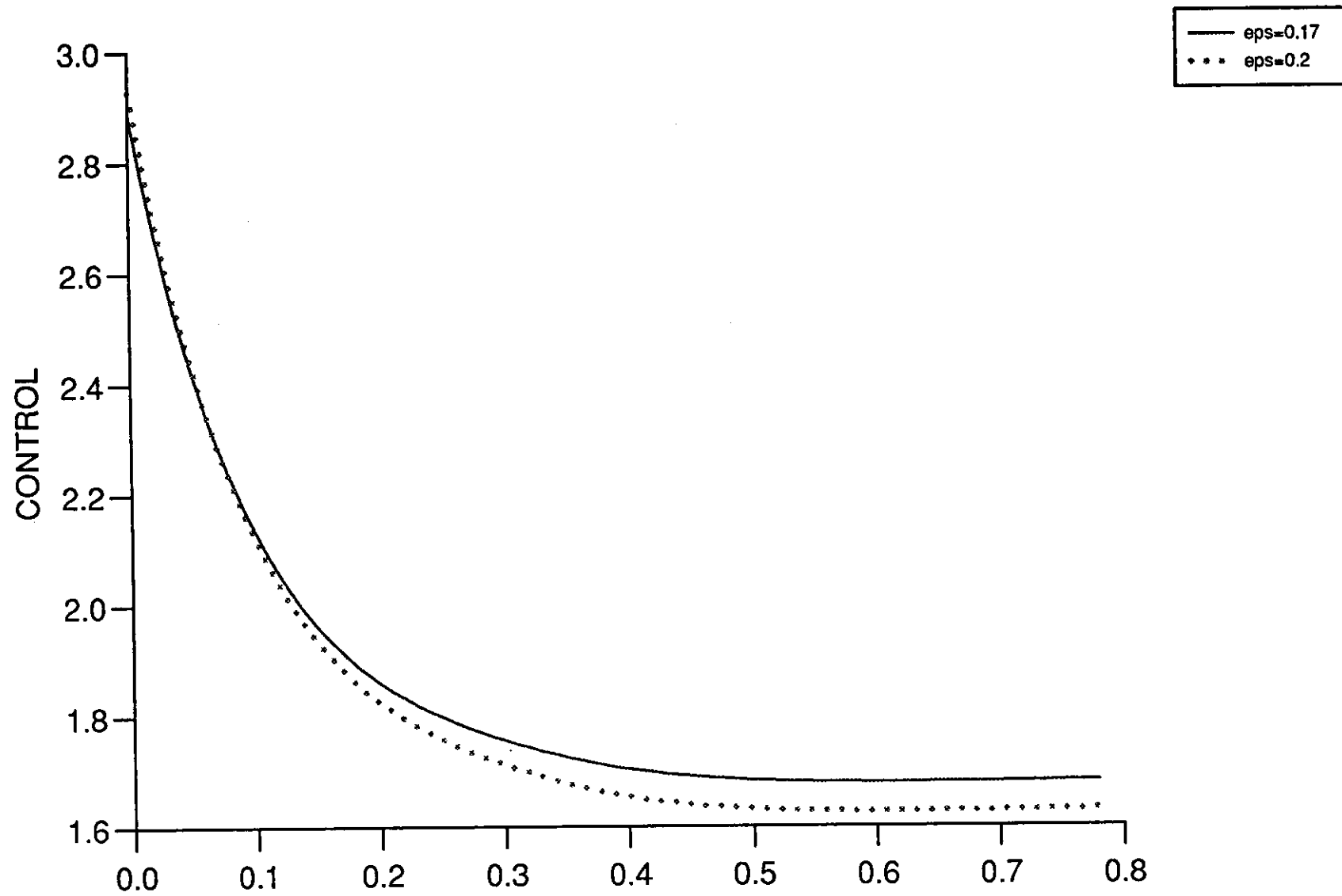


Figure (8.3.7)

GFS

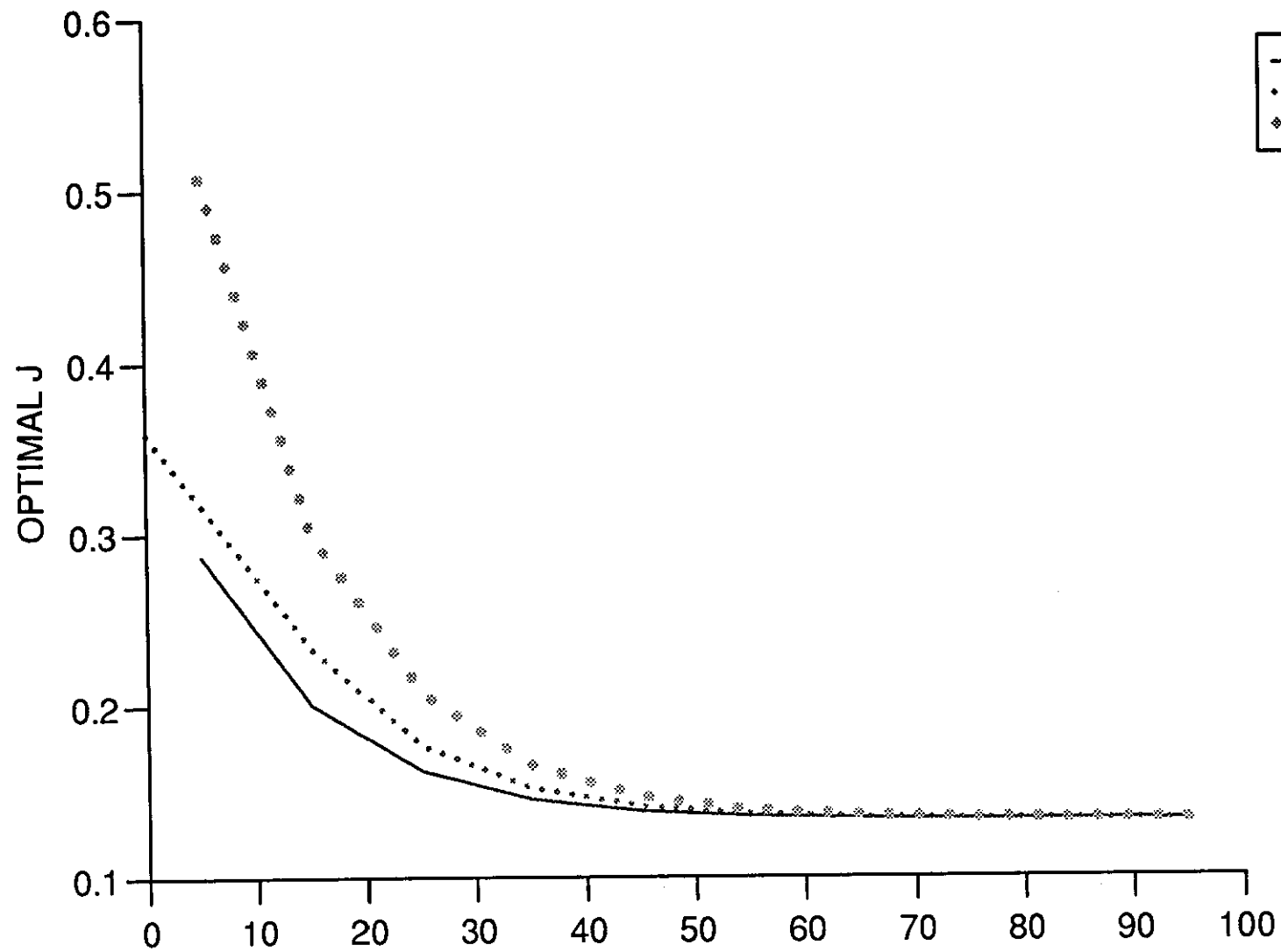


Figure (8.3.8)

SD

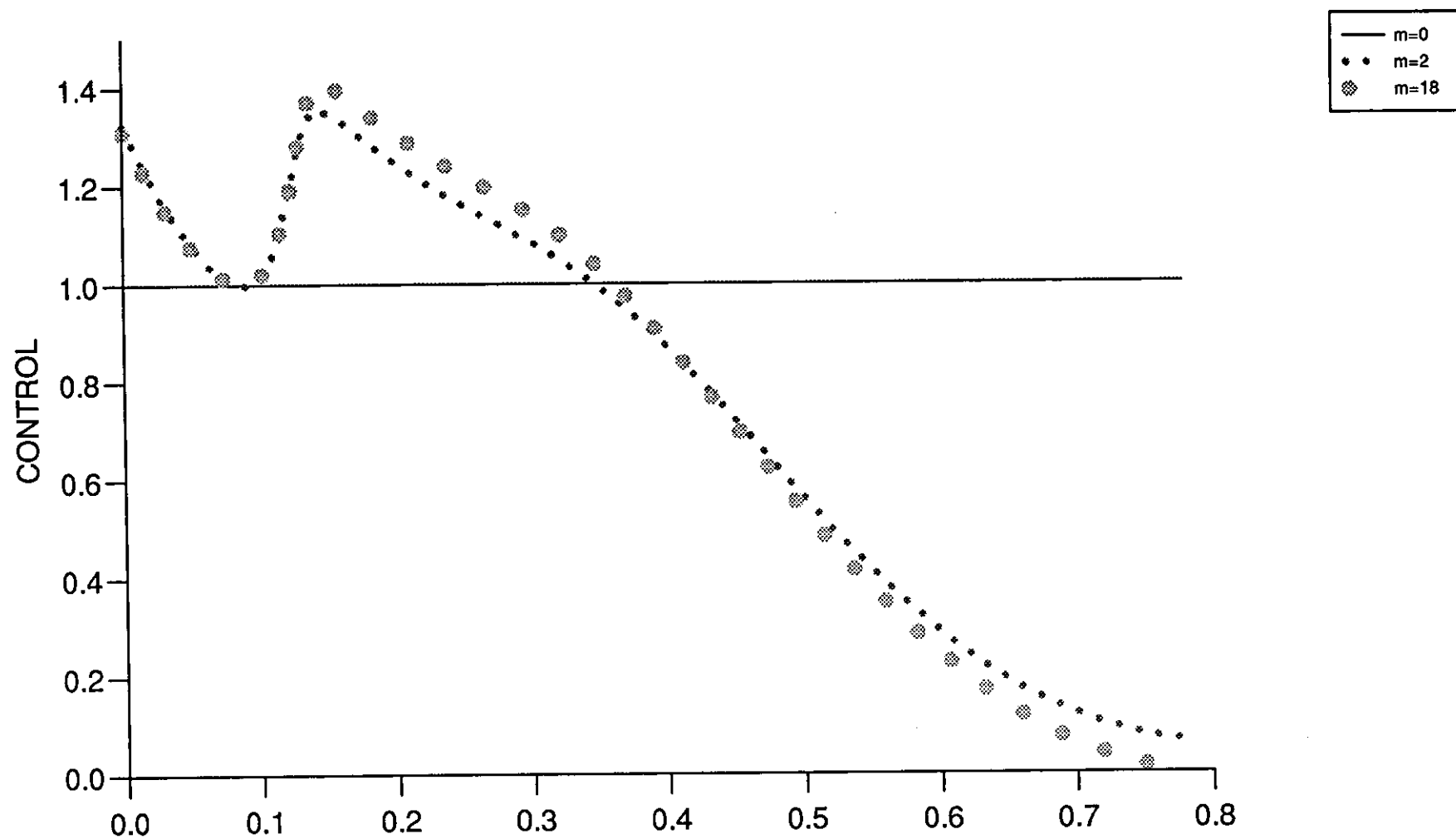


Figure (8.3.9)

SD

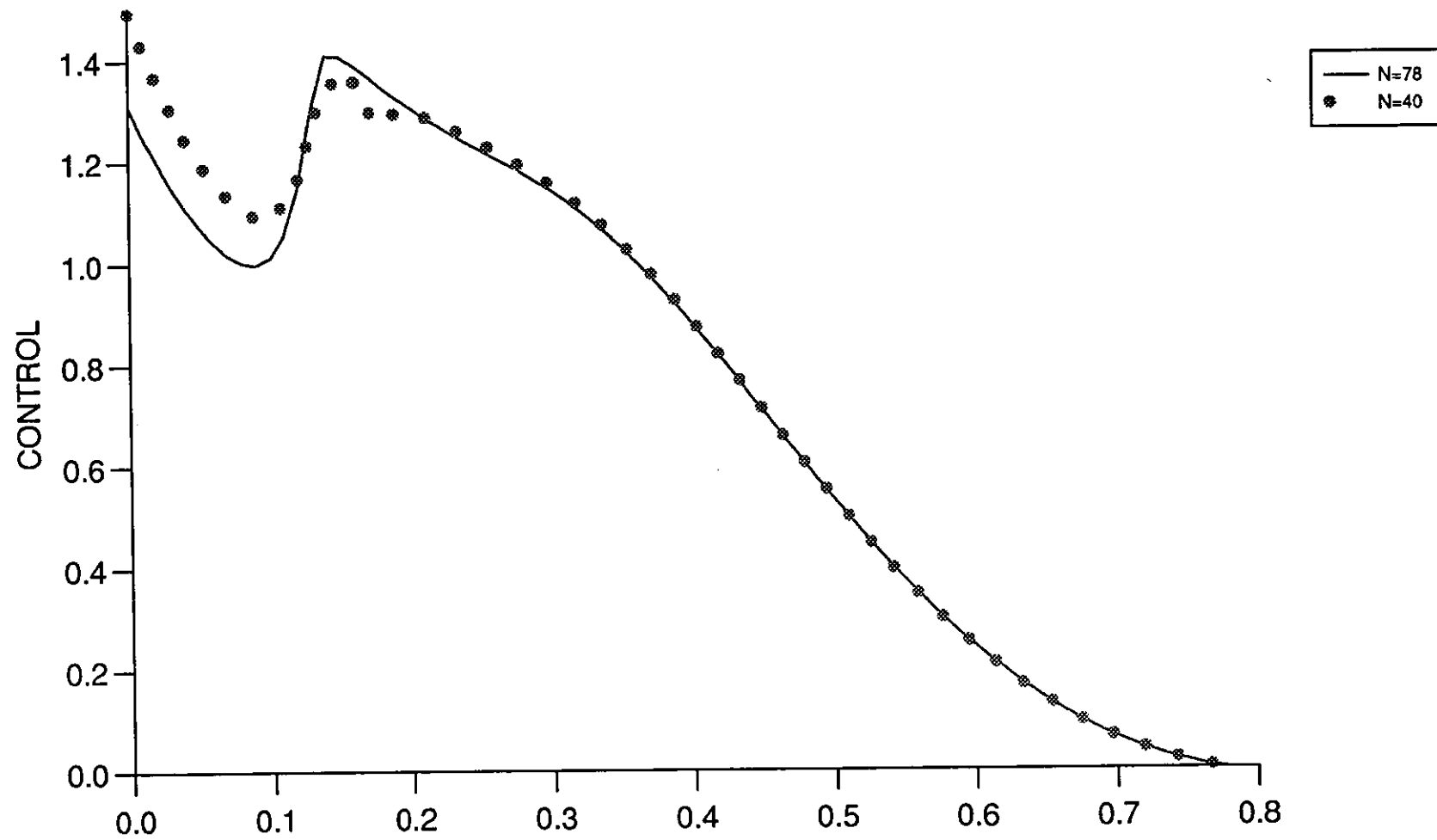


Figure (8.3.10)

SD

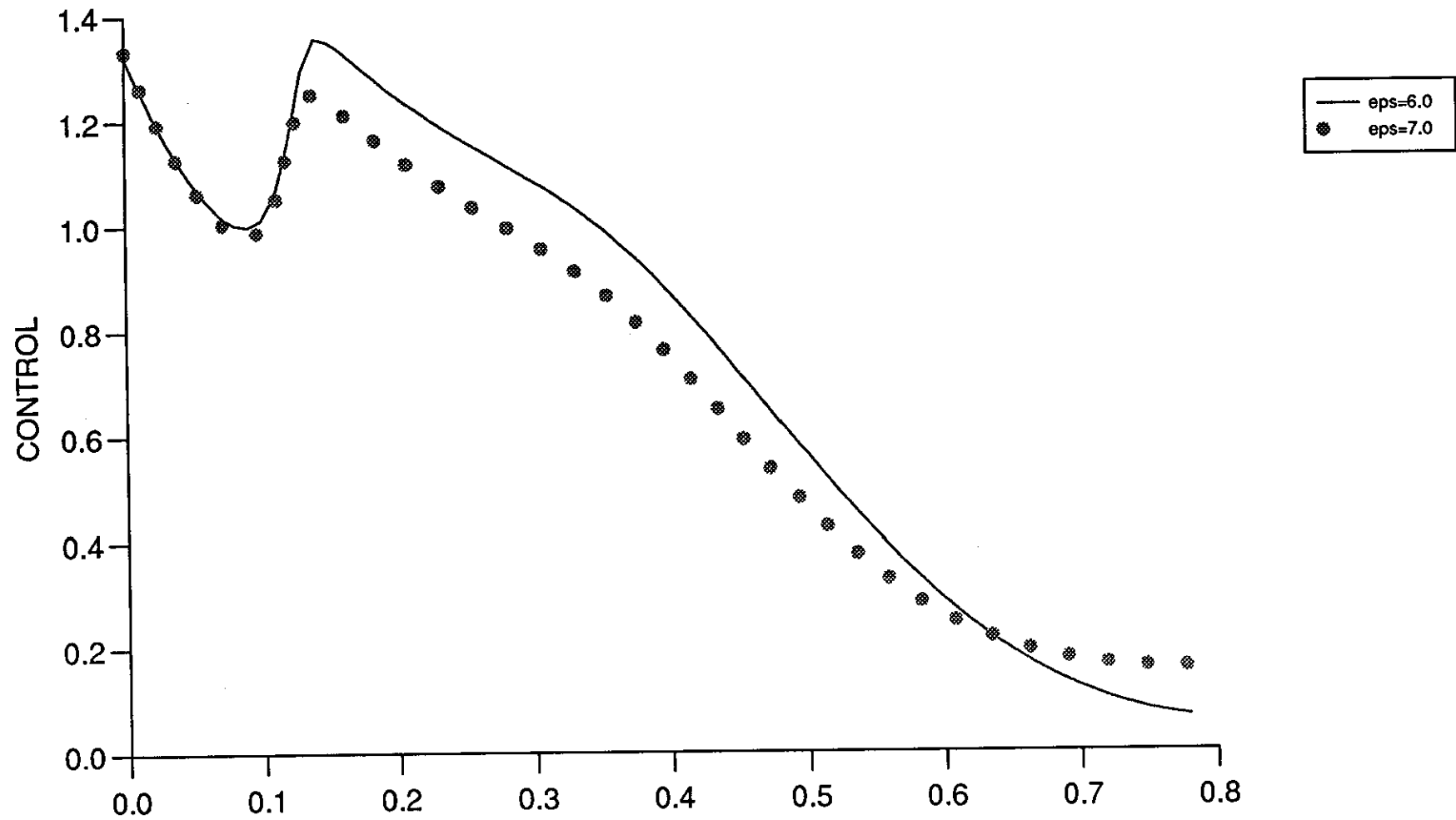


Figure (8.3.11)

SD

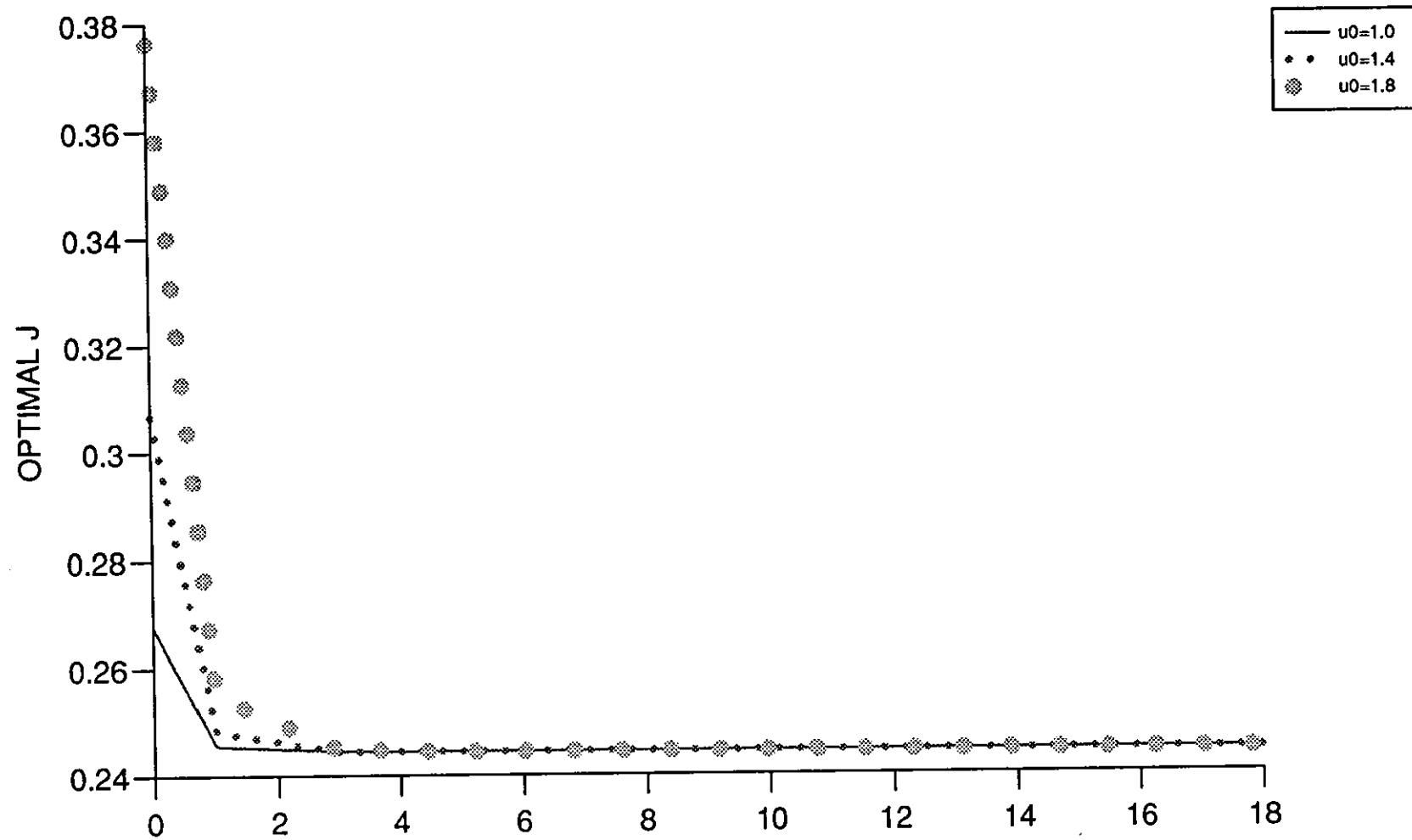


Figure (8.3.12)

SD

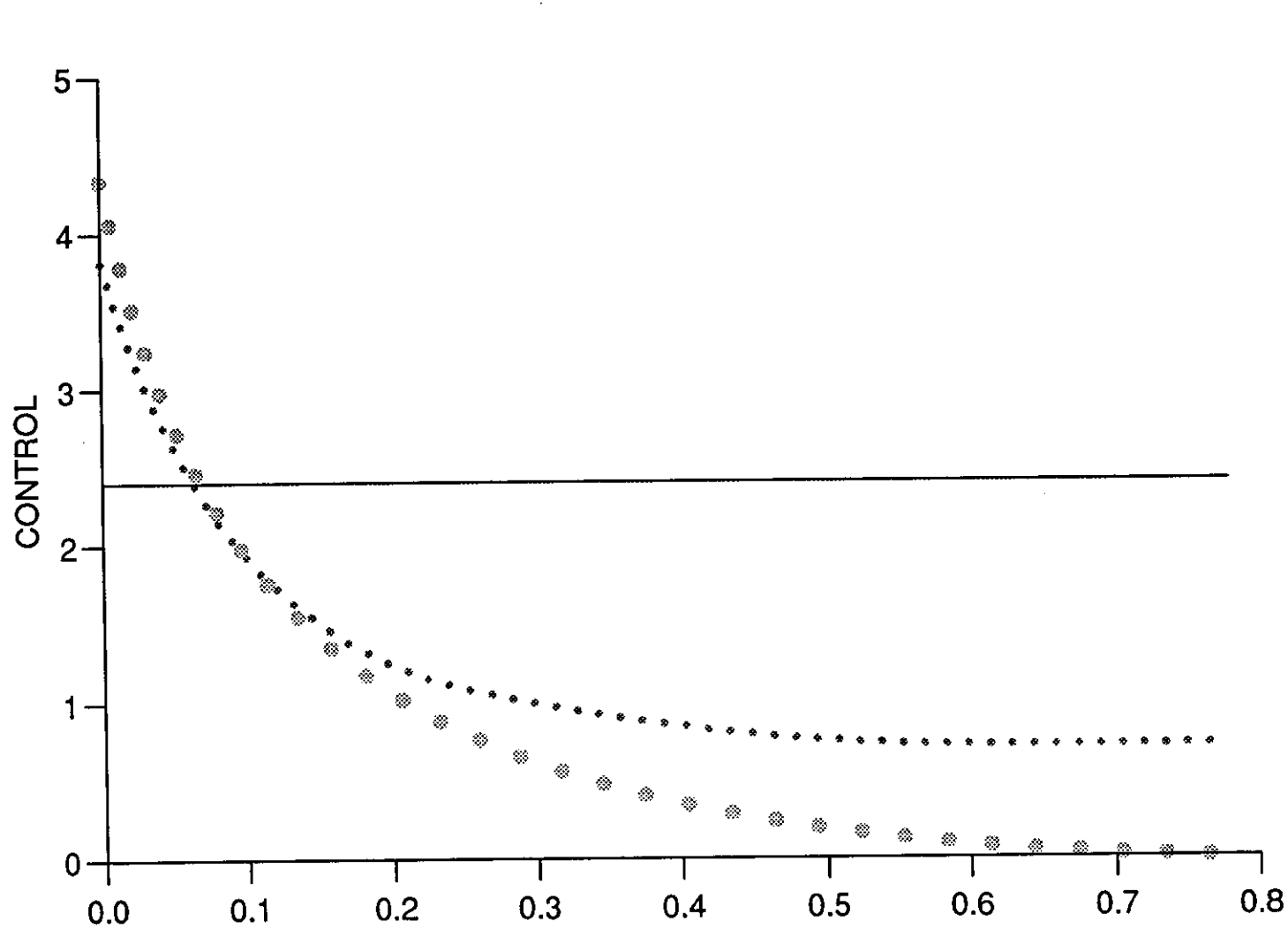


Figure (8.3.13)

SD

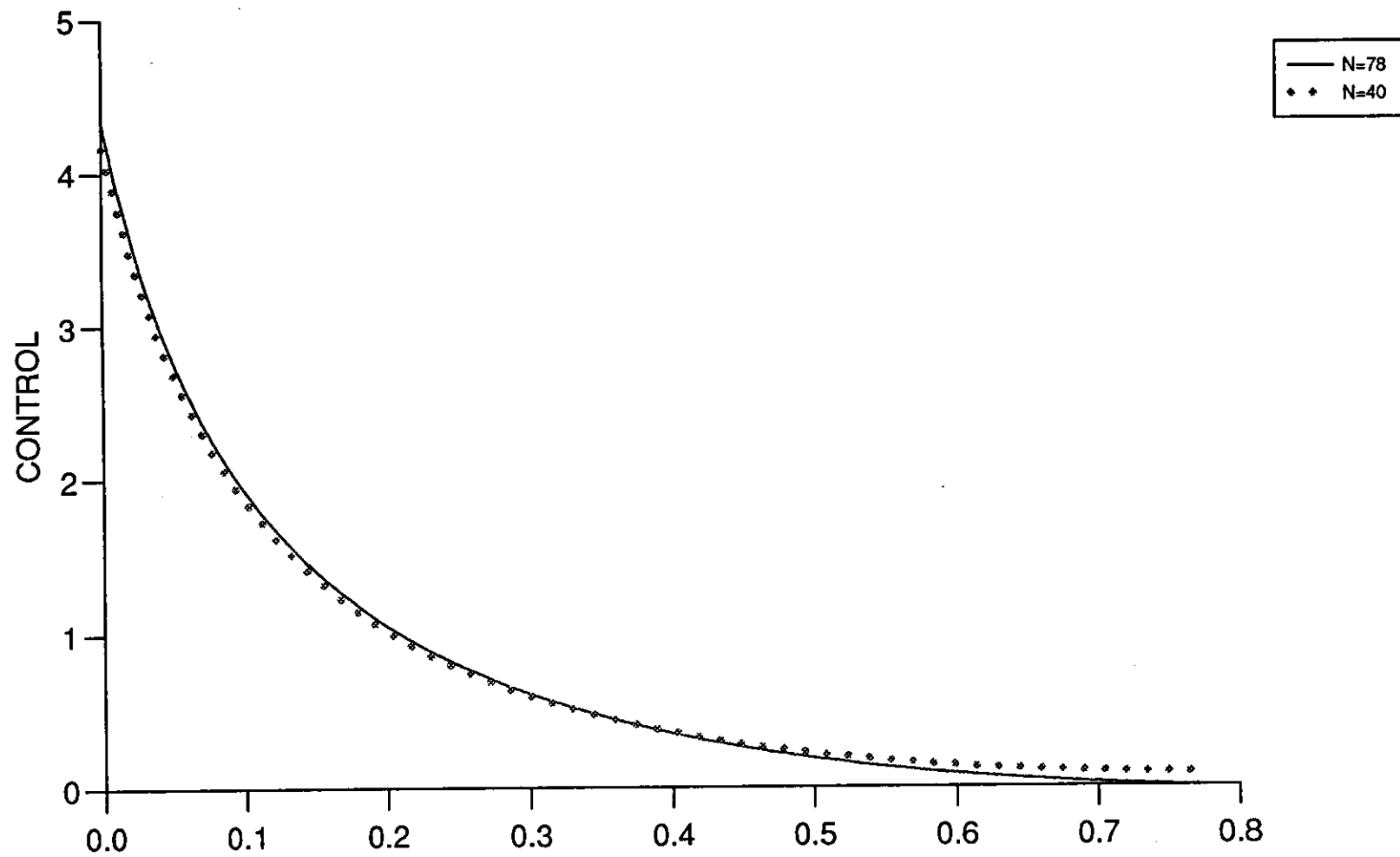


Figure (8.3.14)

SD

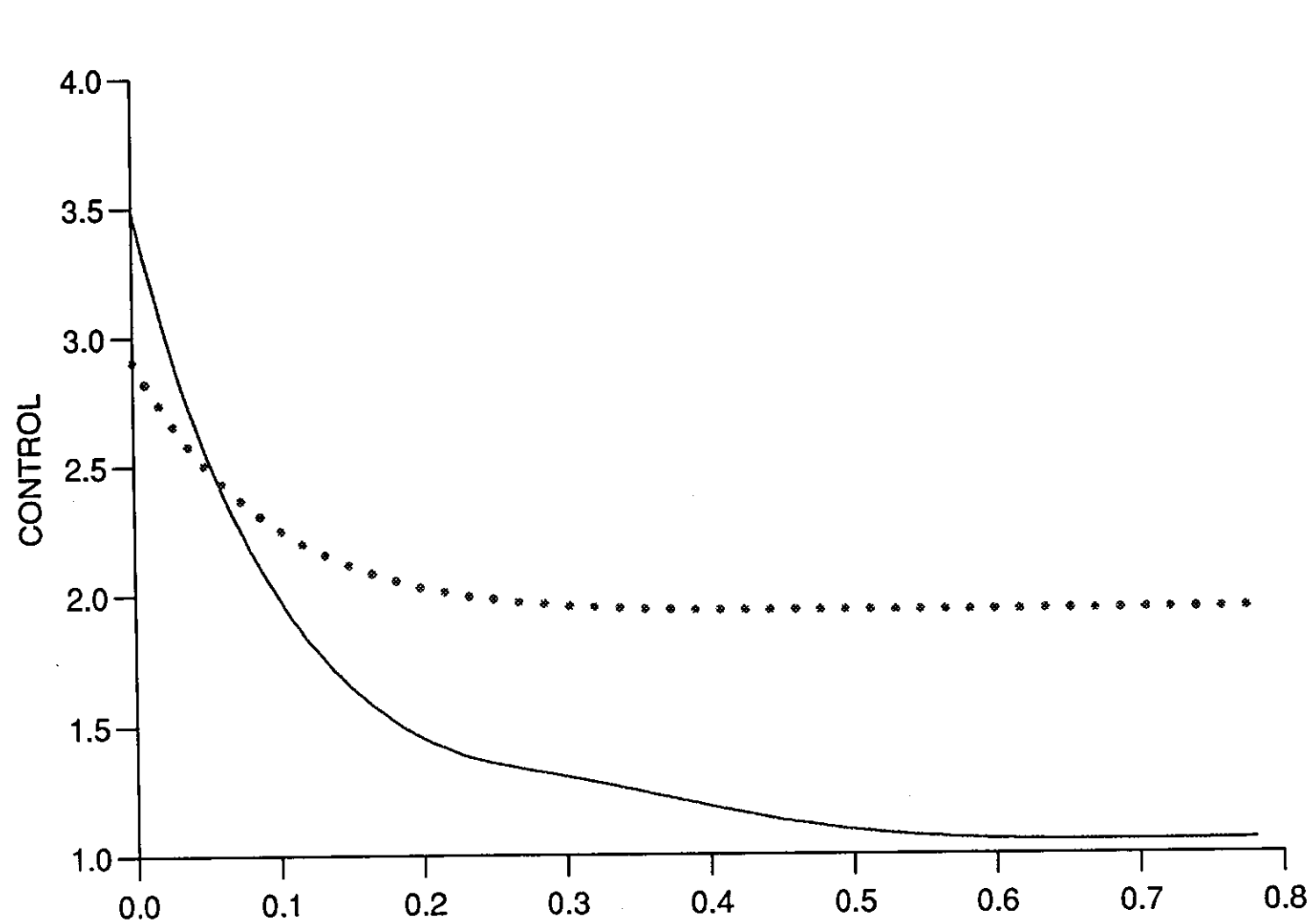


Figure (8.3.15)

SD

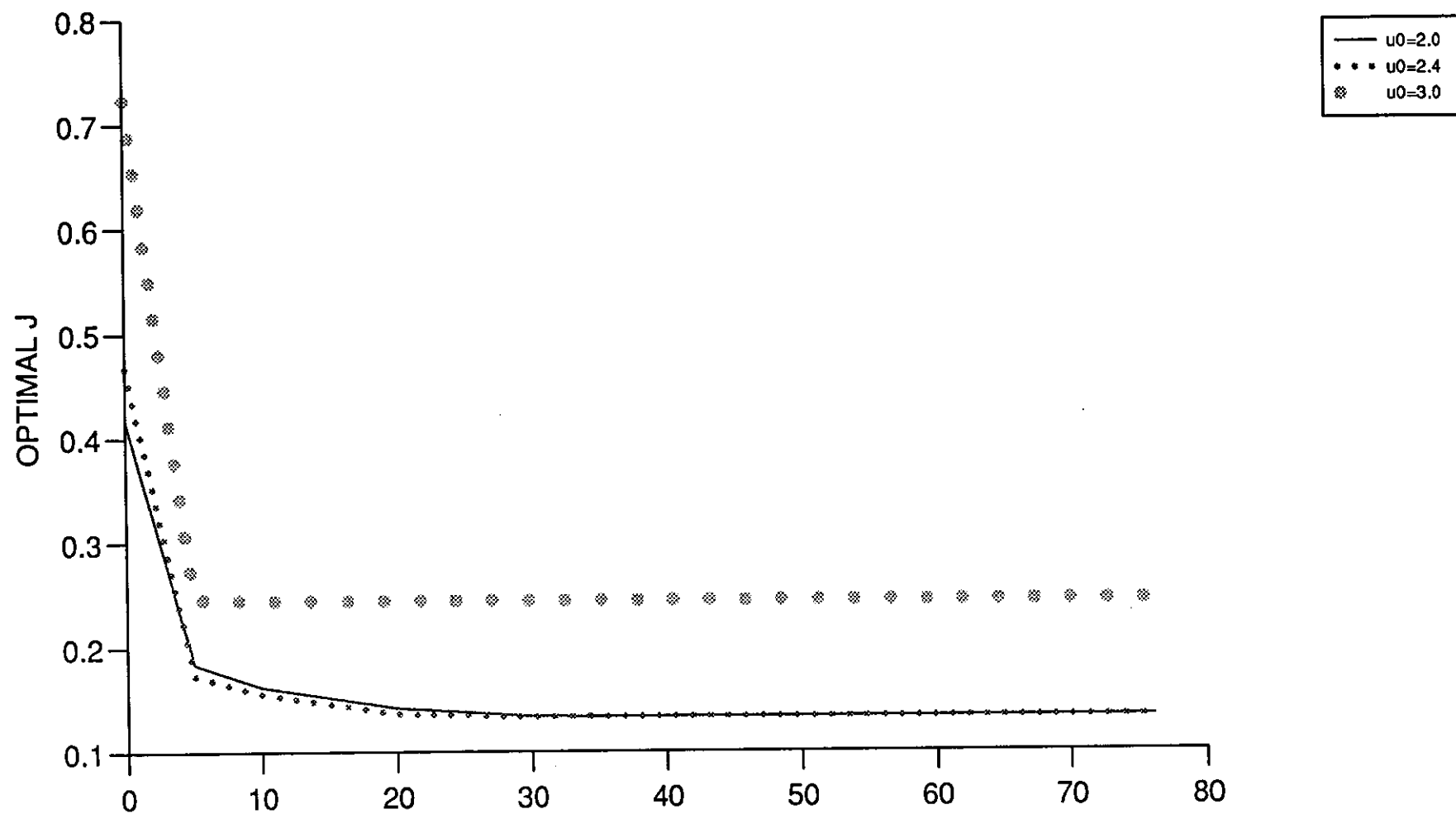


Figure (8.3.16)

FR

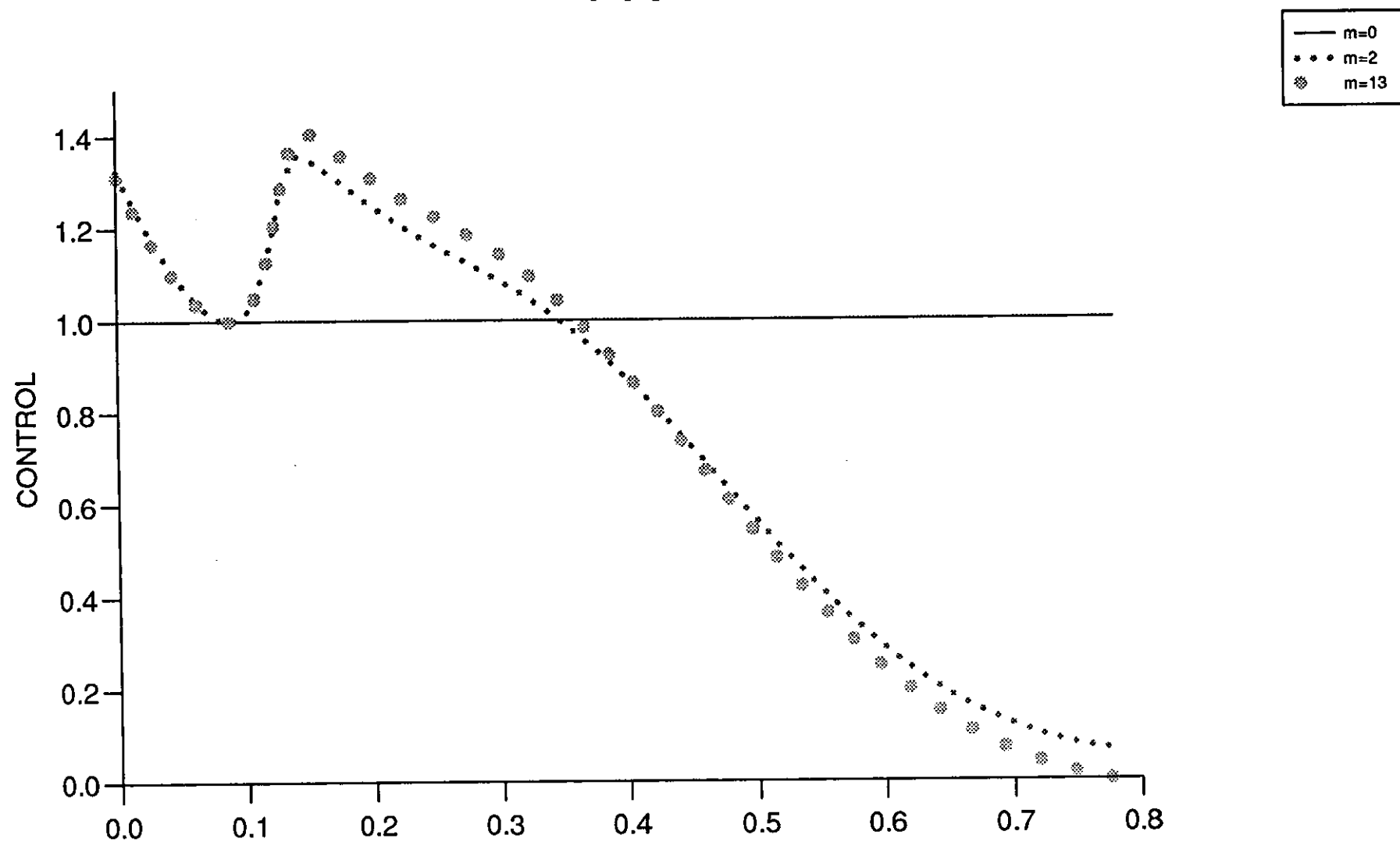


Figure (8.3.17)

FR

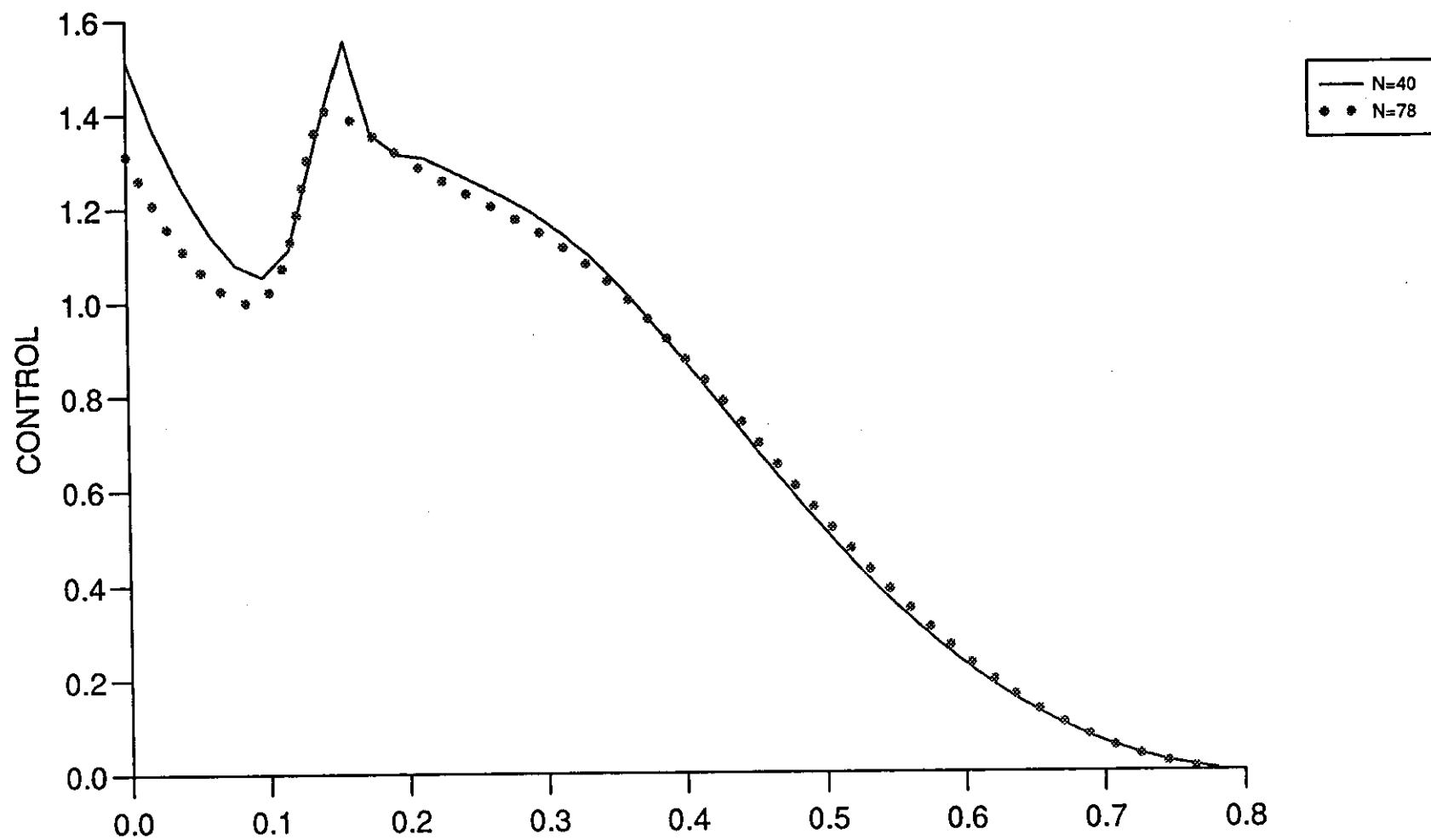


Figure (8.3.18)

FR

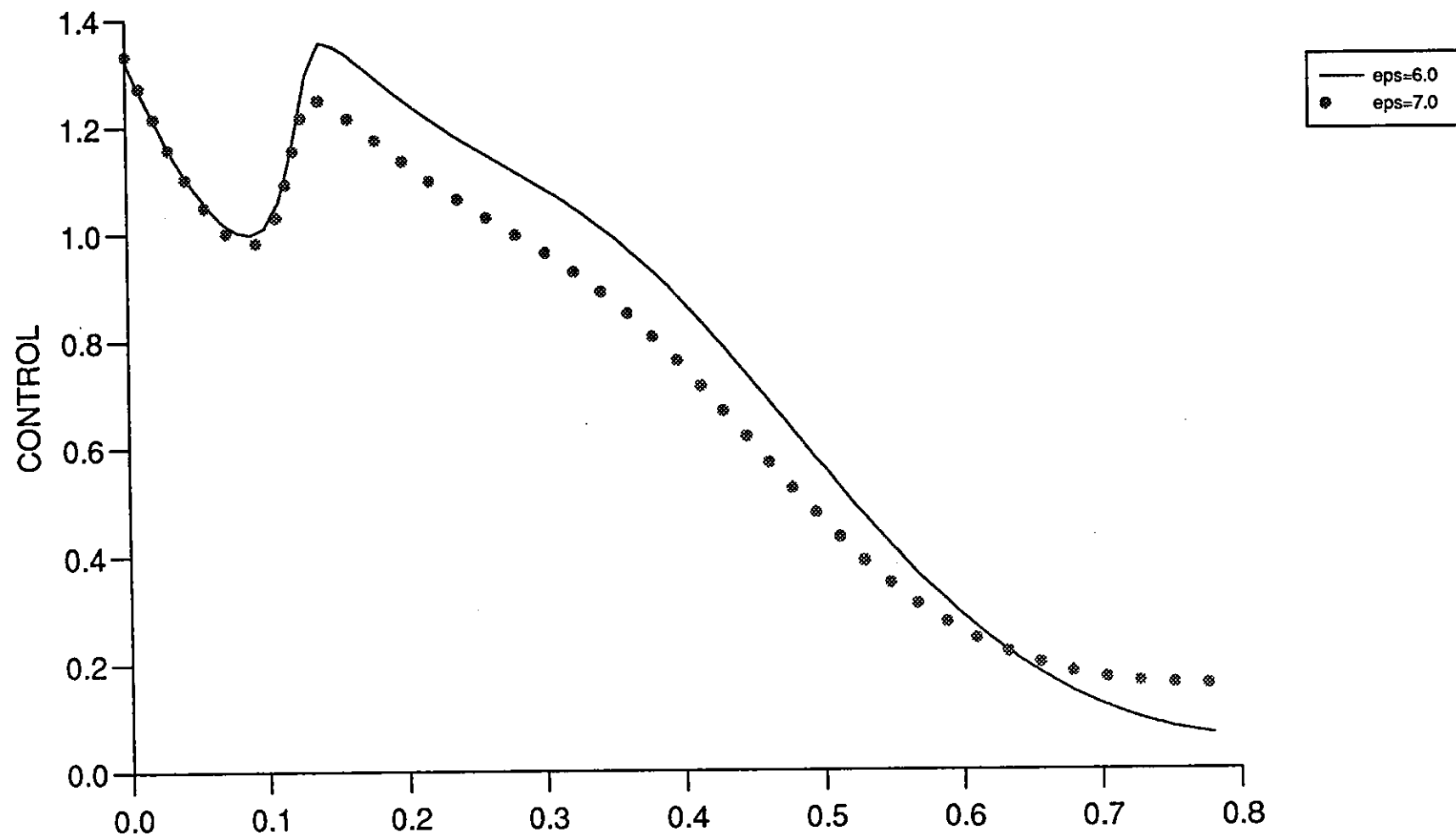


Figure (8.3.19)

FR

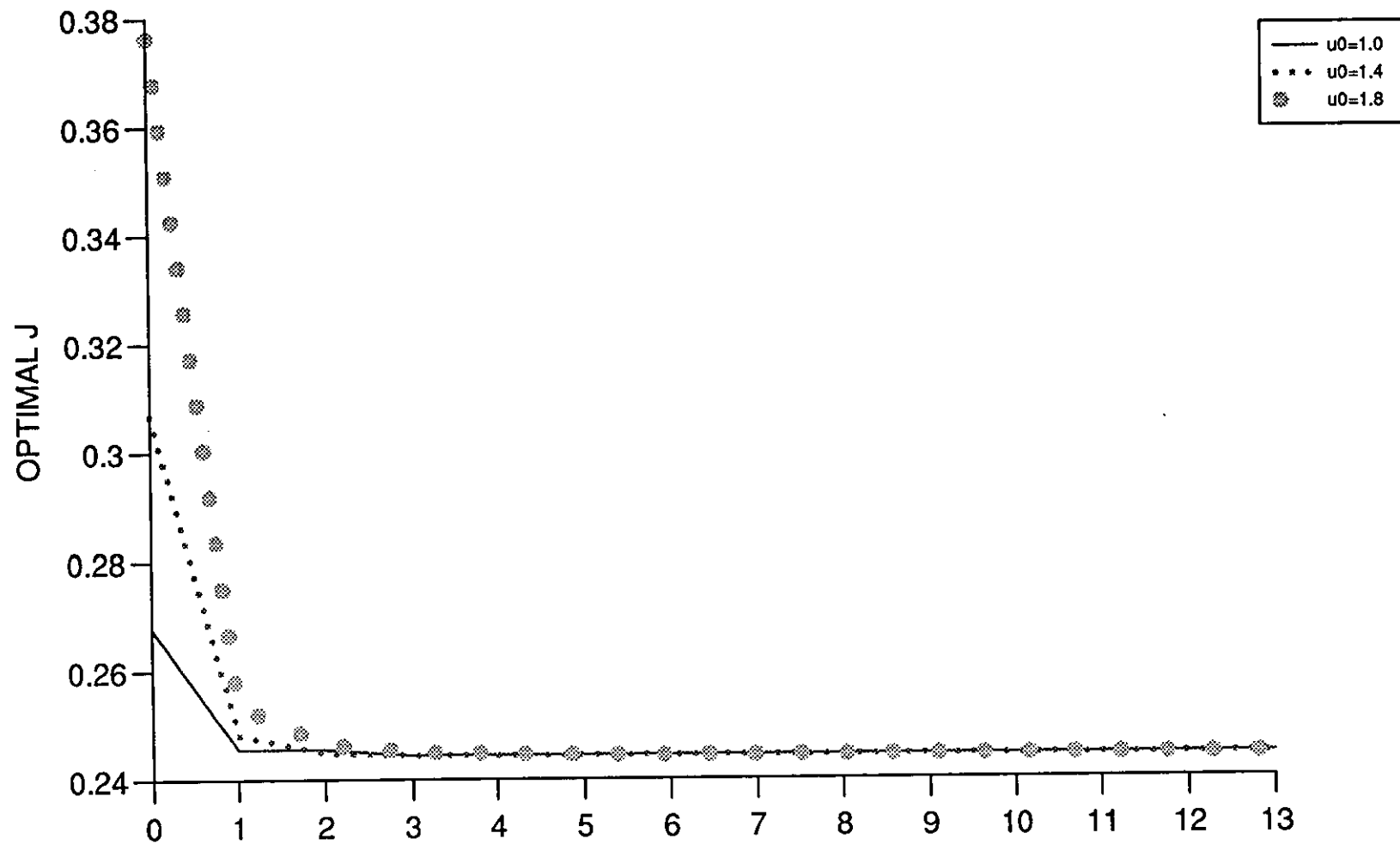


Figure (8.3.20)

FR

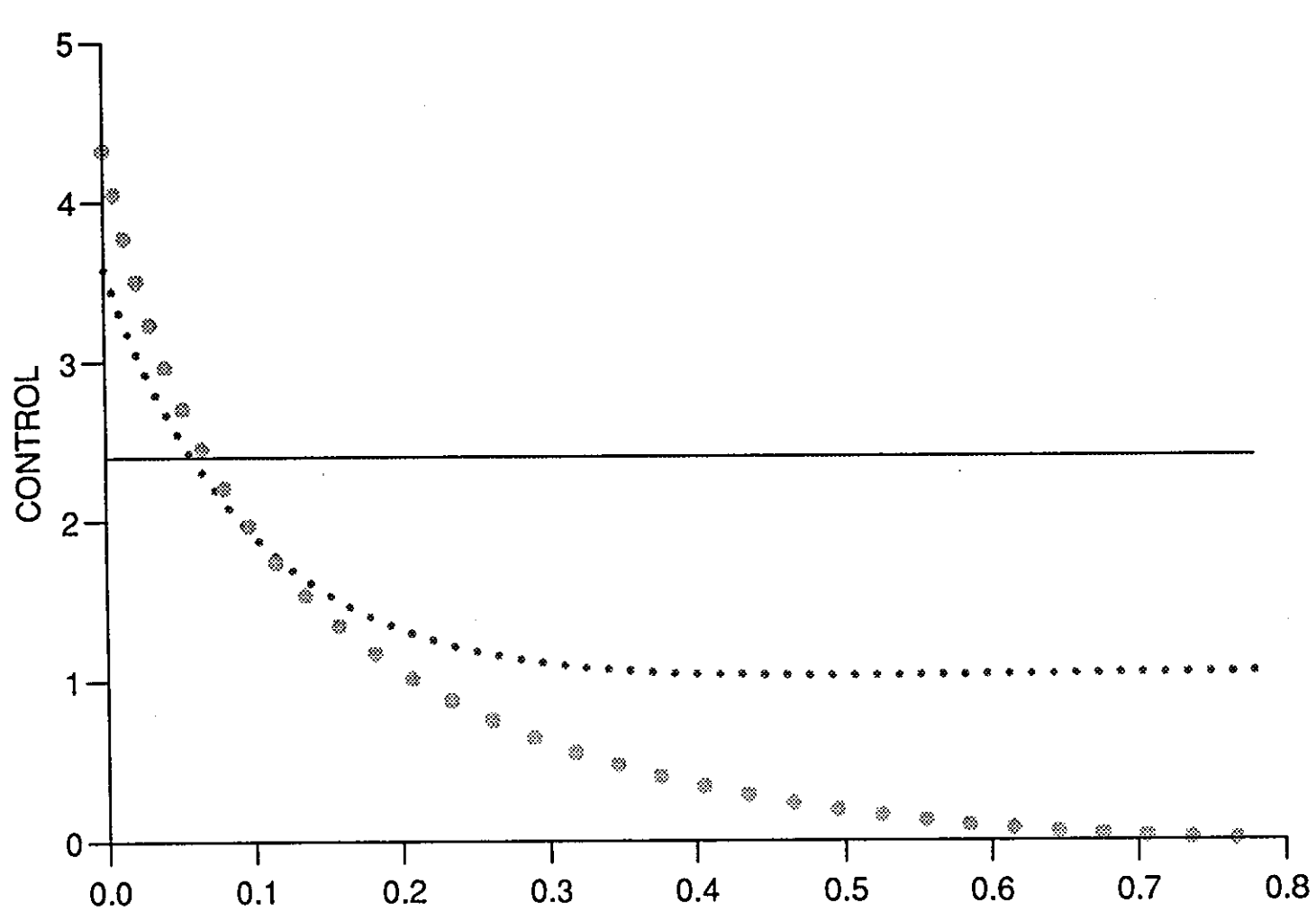


Figure (8.3.21)

FR

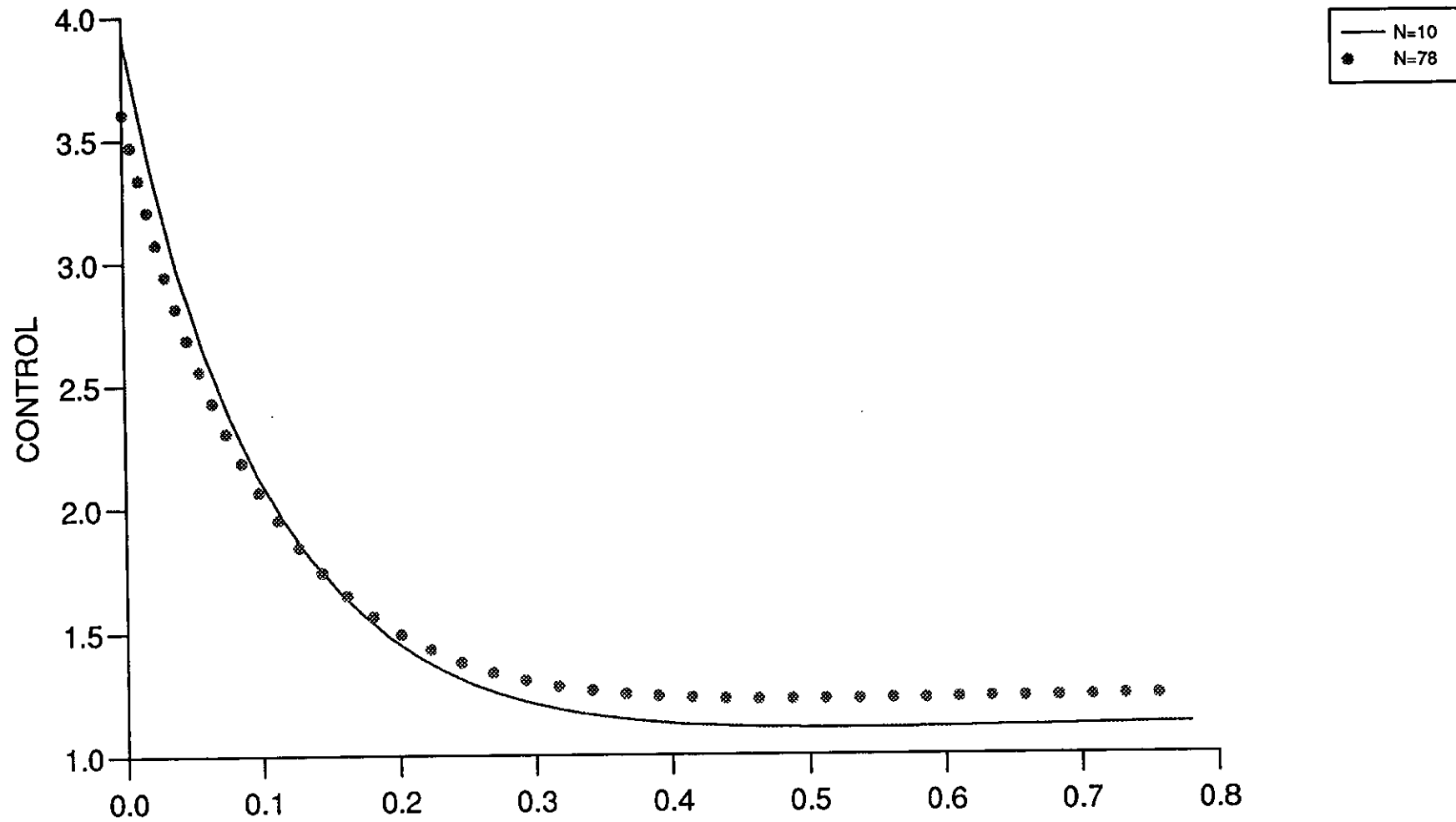


Figure (8.3.22)

FR

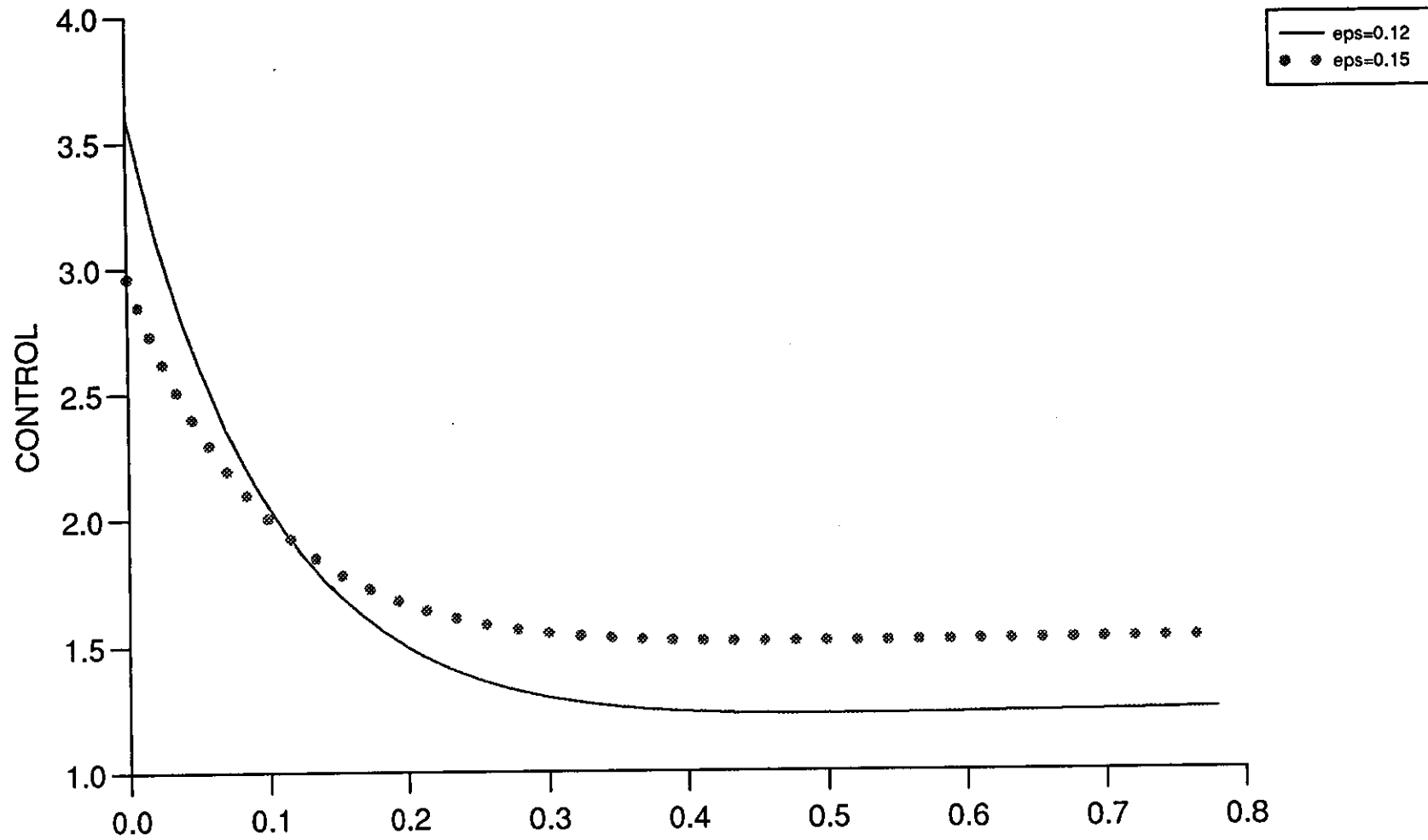


Figure (8.3.23)

FR

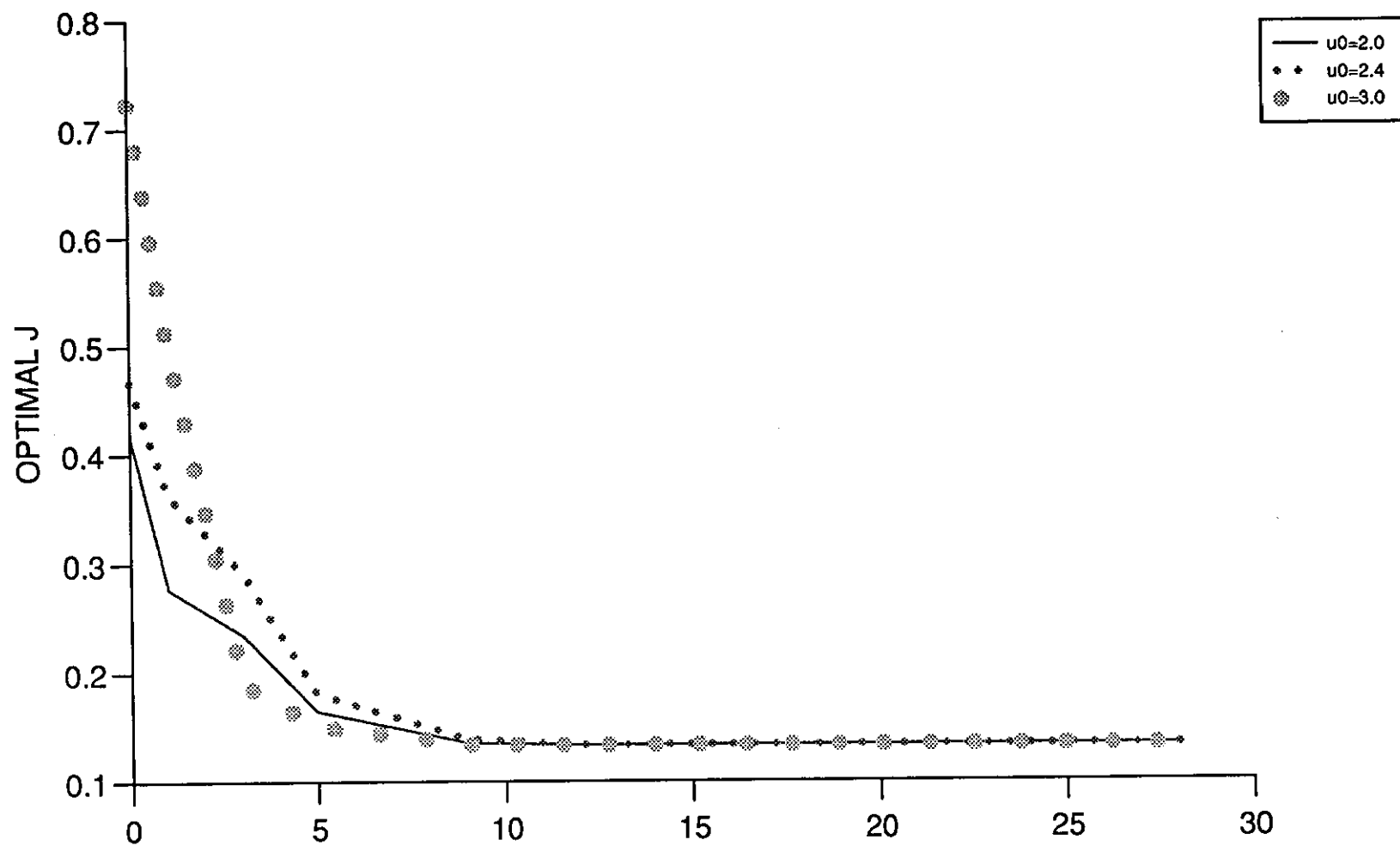


Figure (8.3.24)

PR

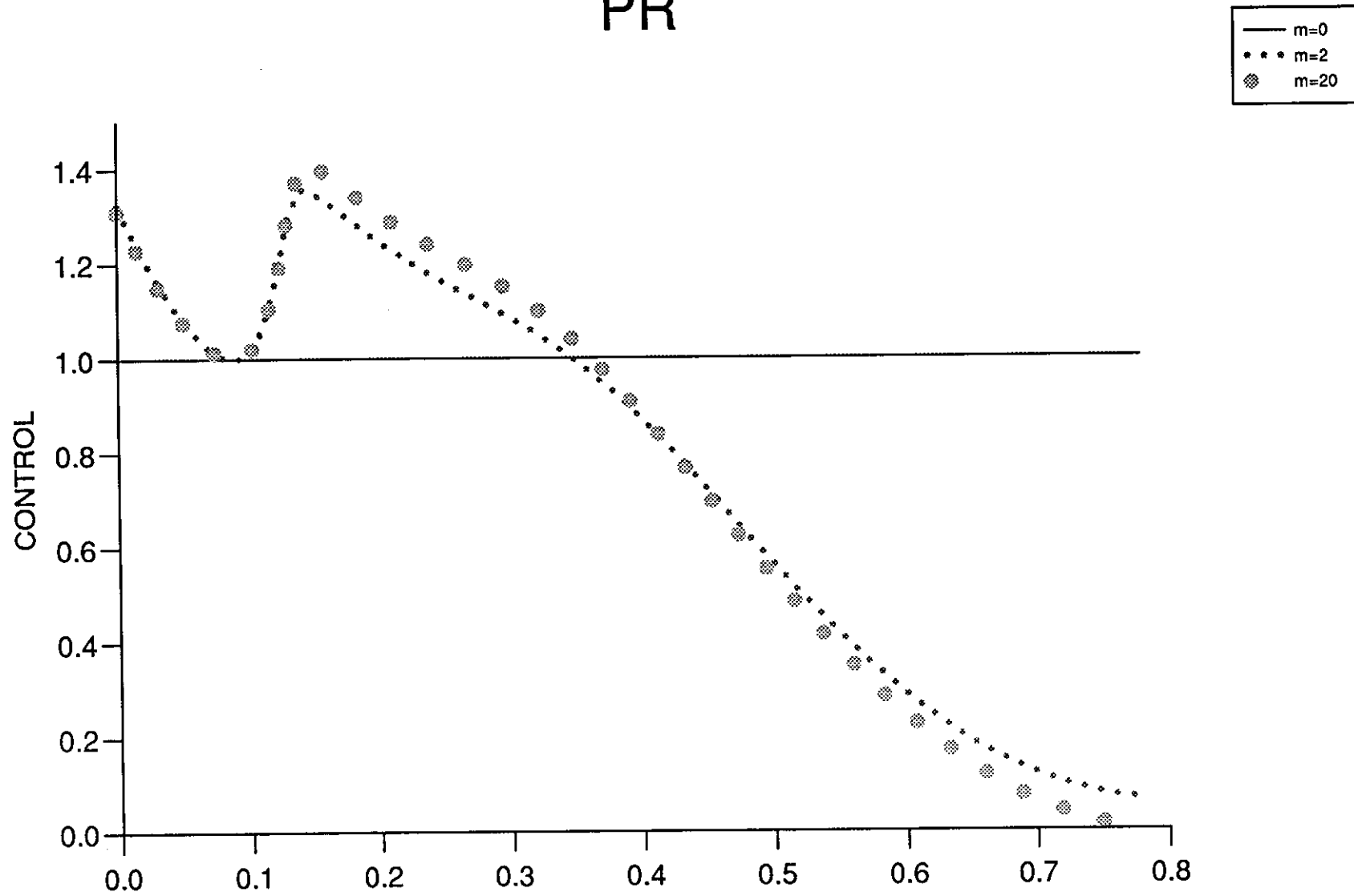


Figure (8.3.25)

PR

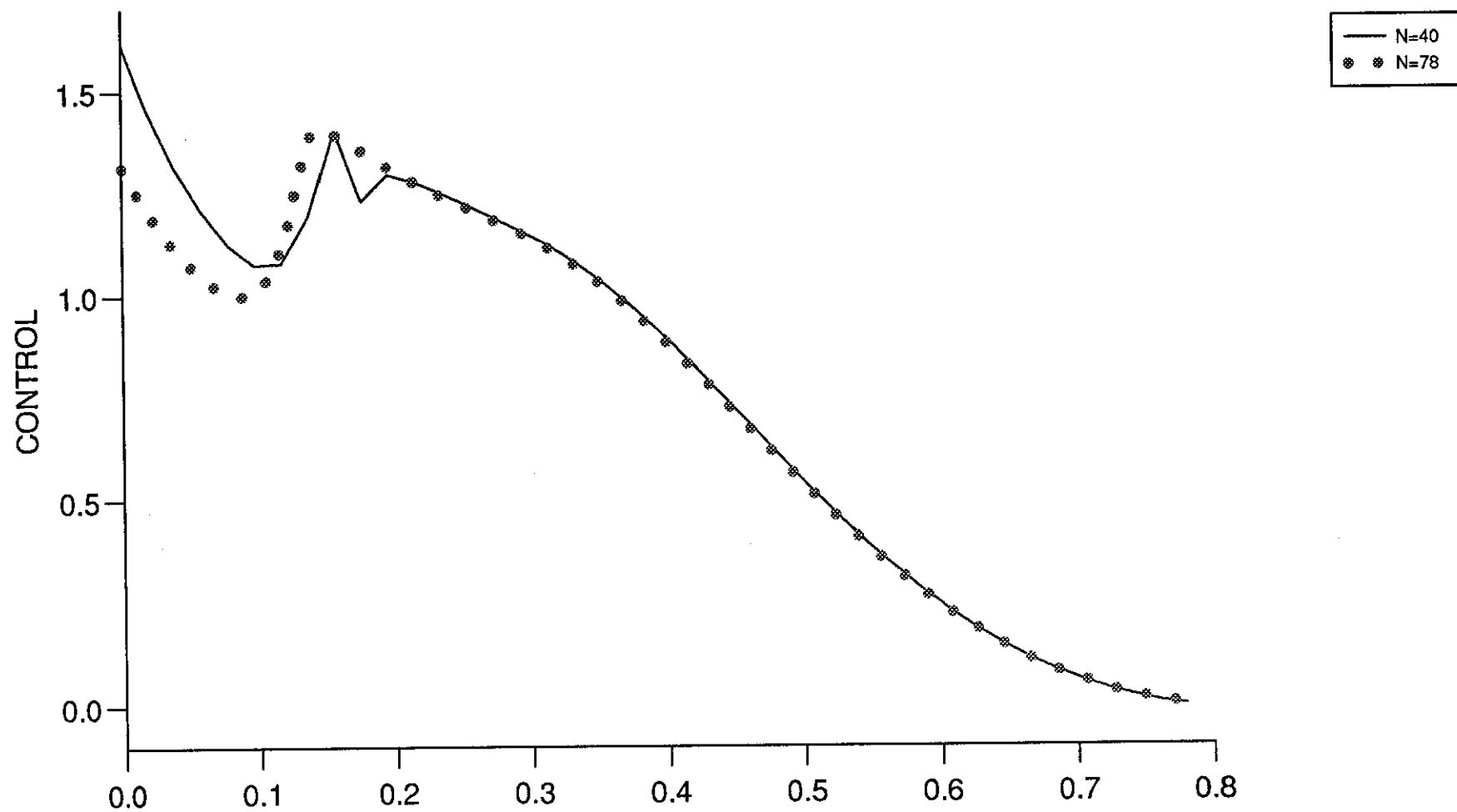


Figure (8.3.26)

PR

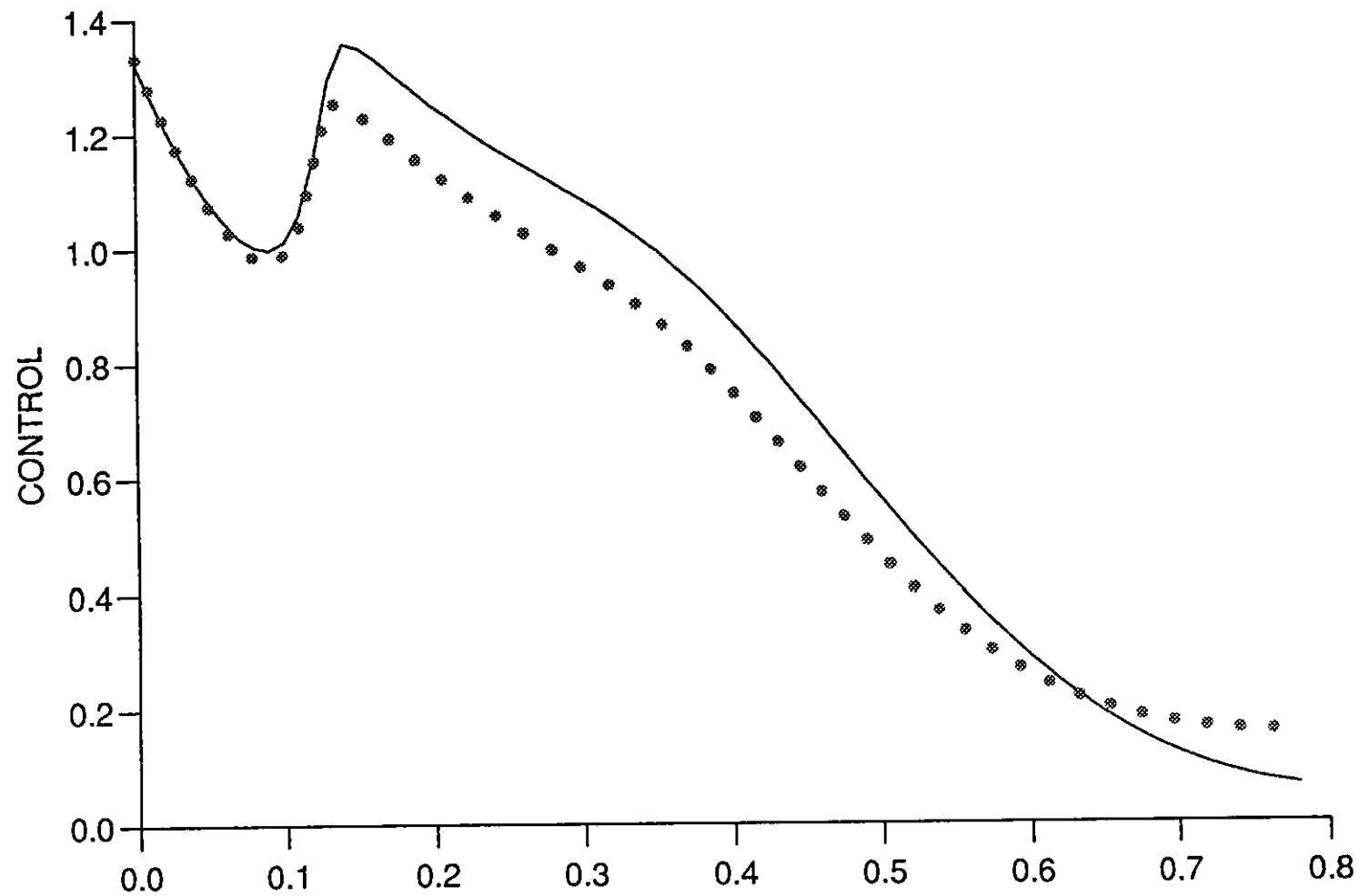


Figure (8.3.27)

PR

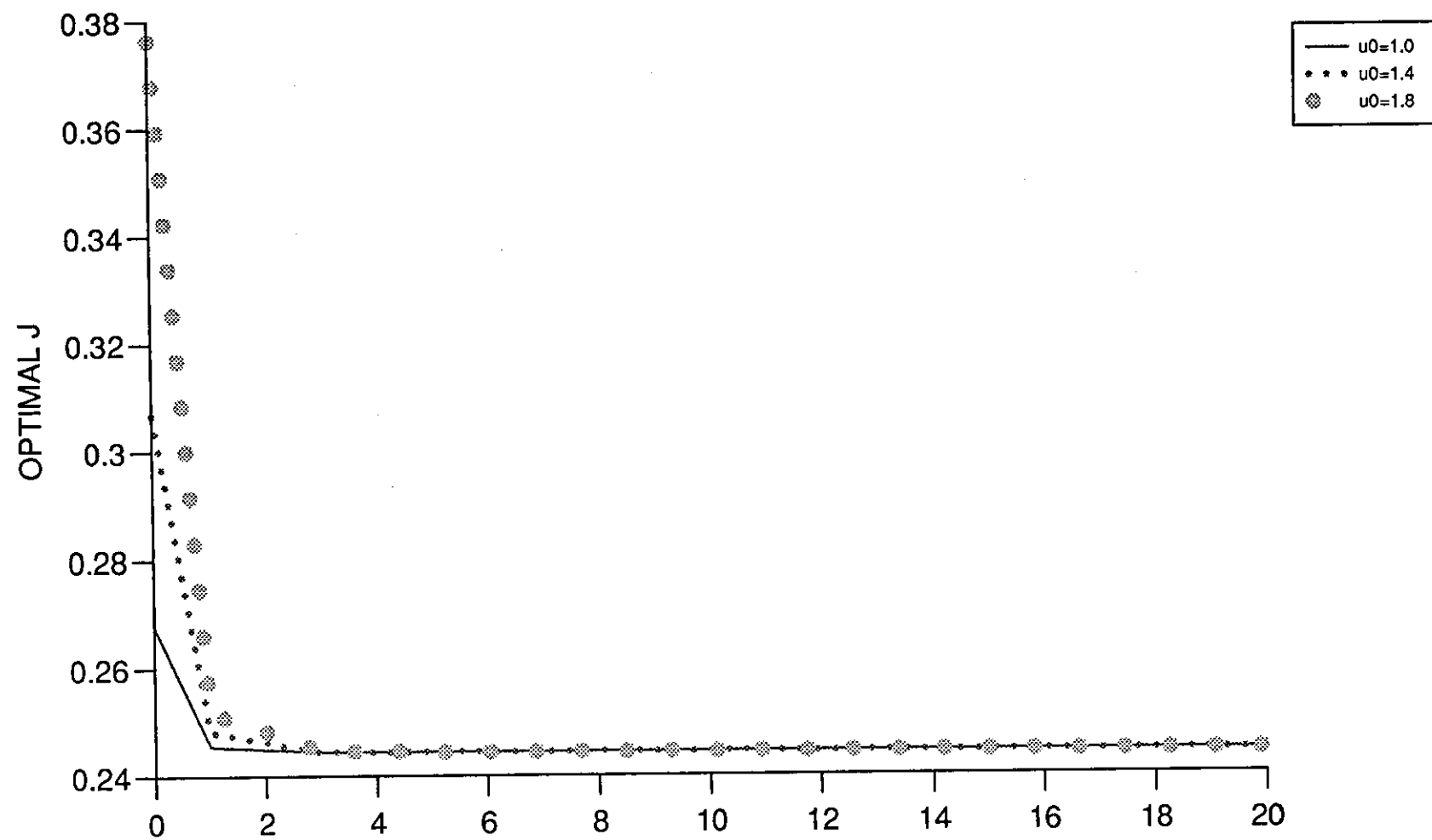


Figure (8.3.28)

PR

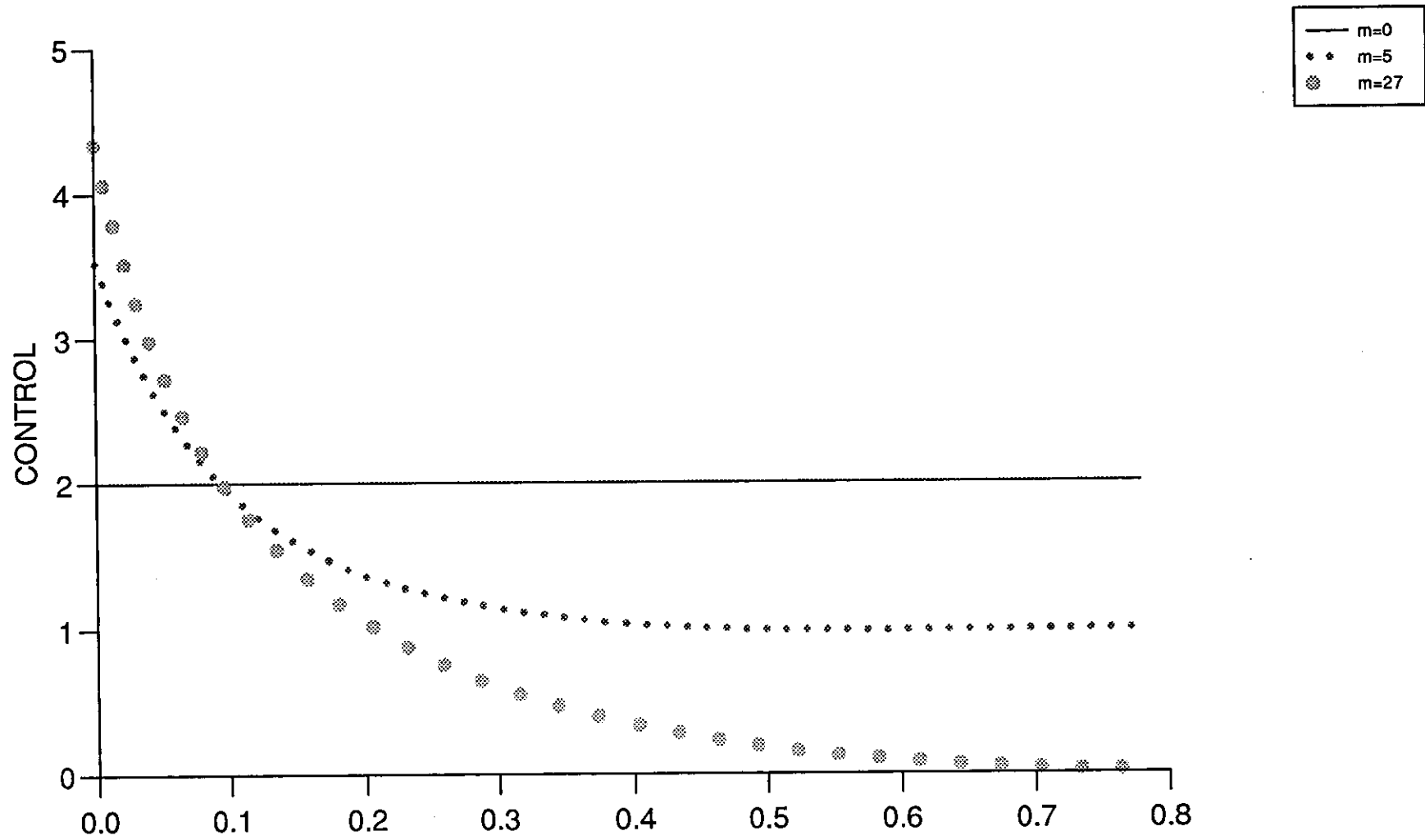


Figure (8.3.29)

PR

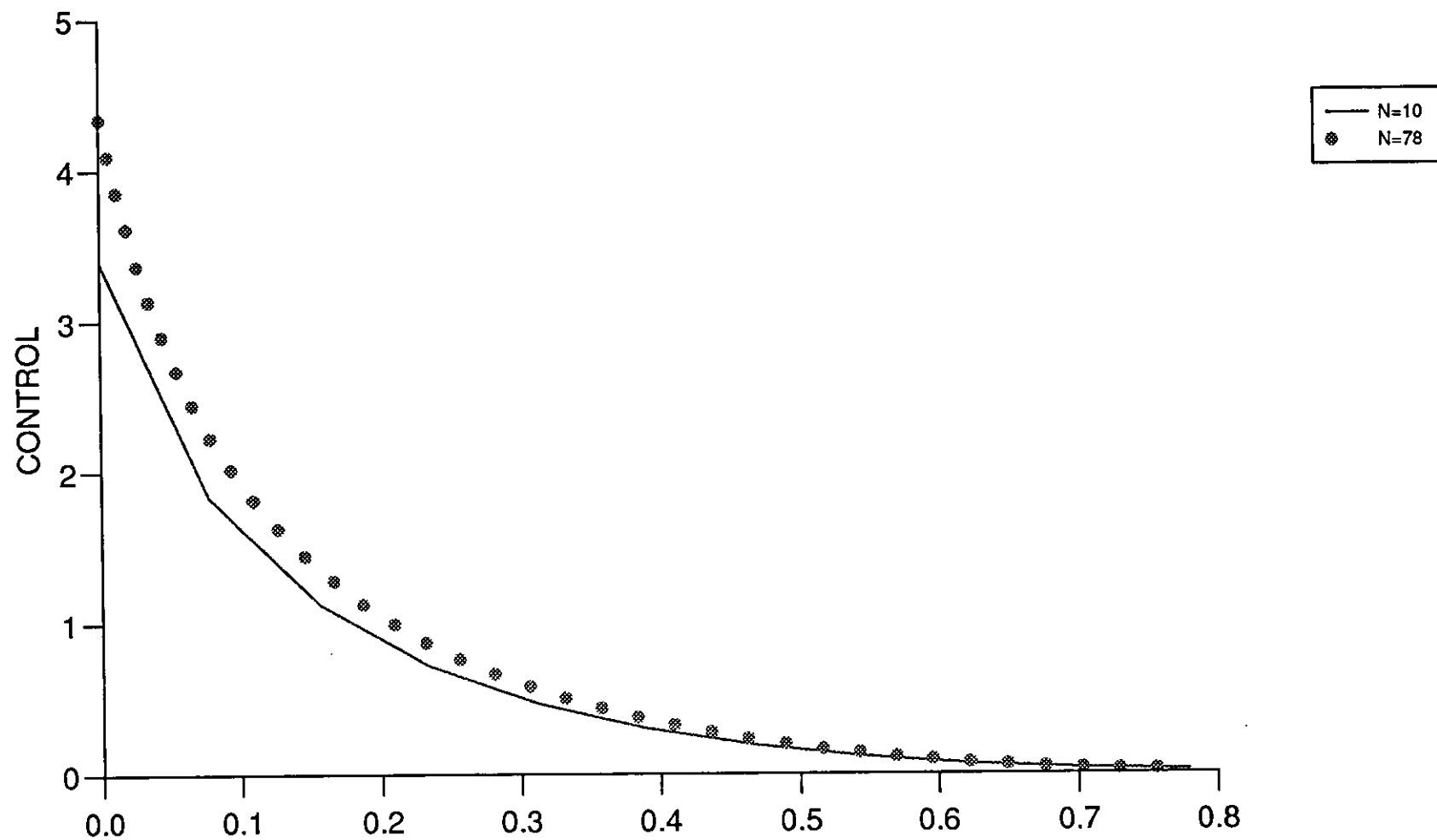


Figure (8.3.30)

PR

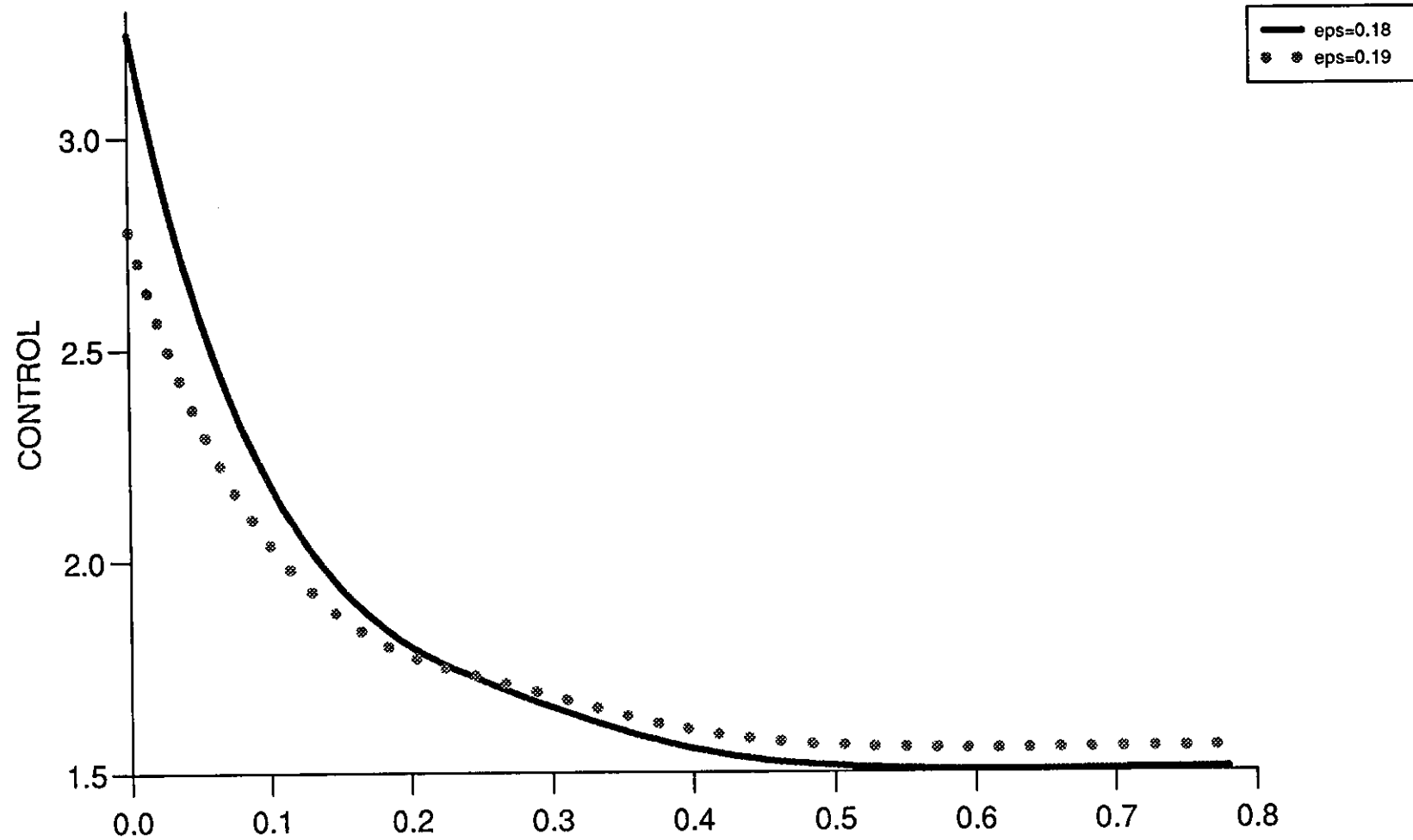


Figure (8.3.31)

PR

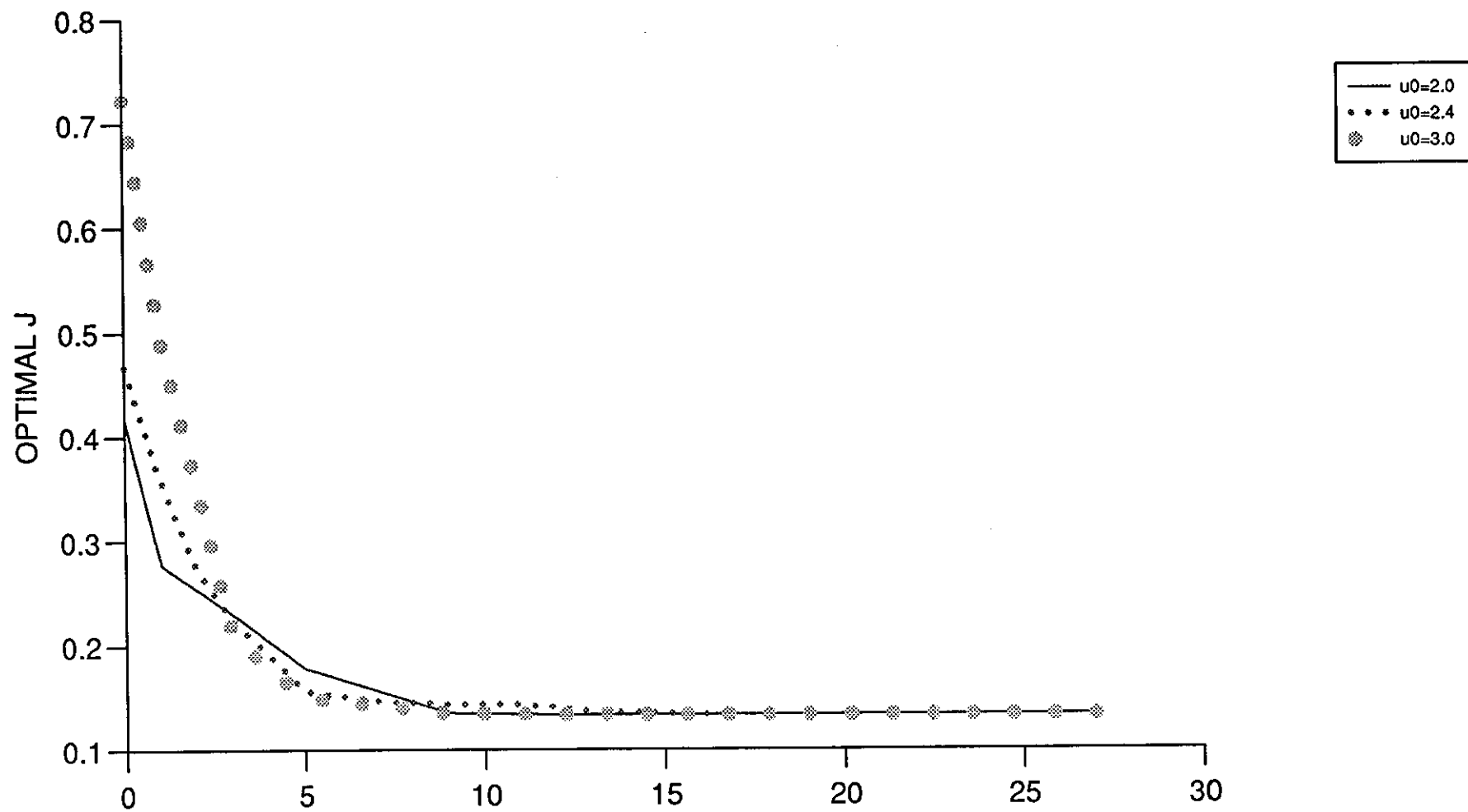


Figure (8.3.32)

H1

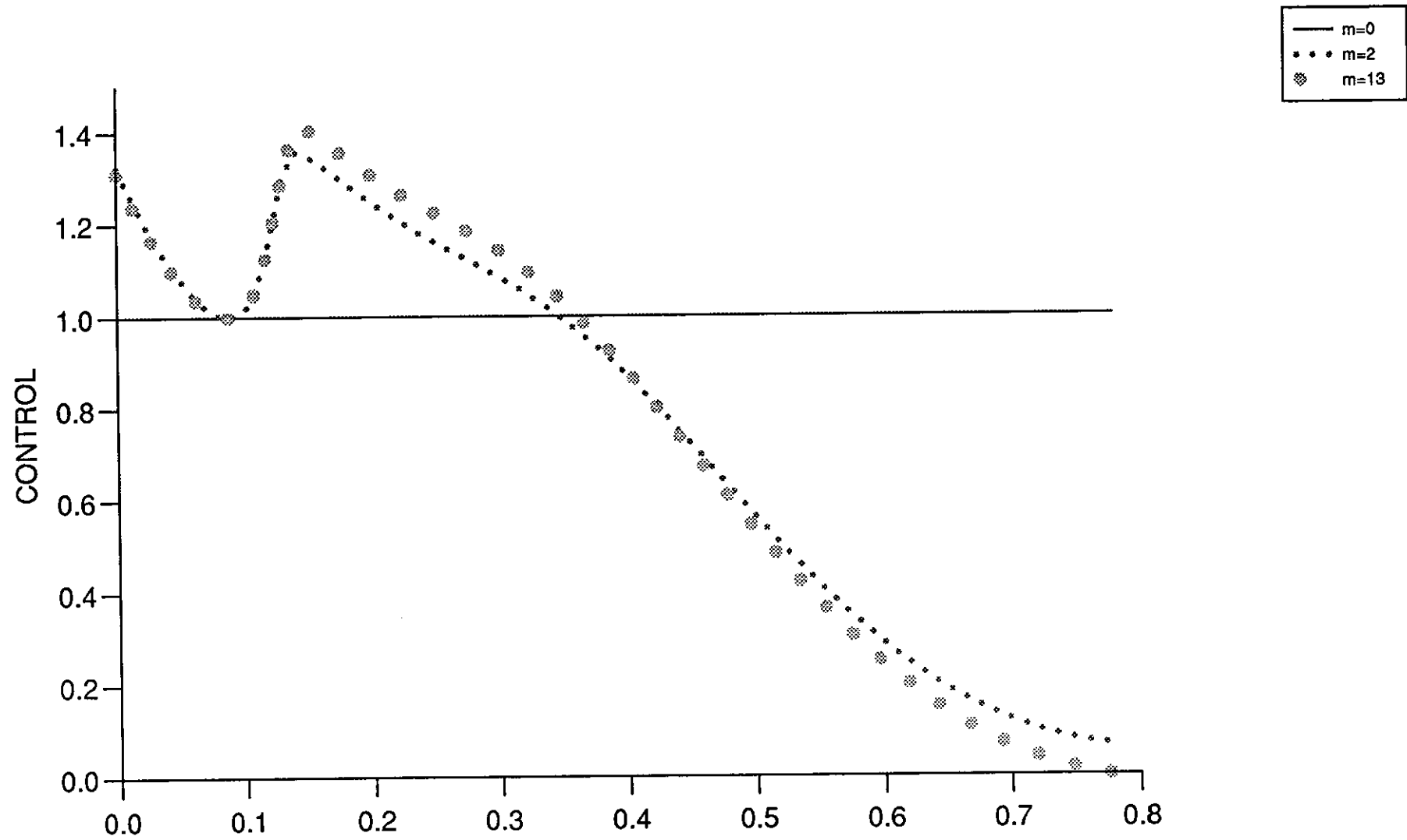


Figure (8.3.33)

H1

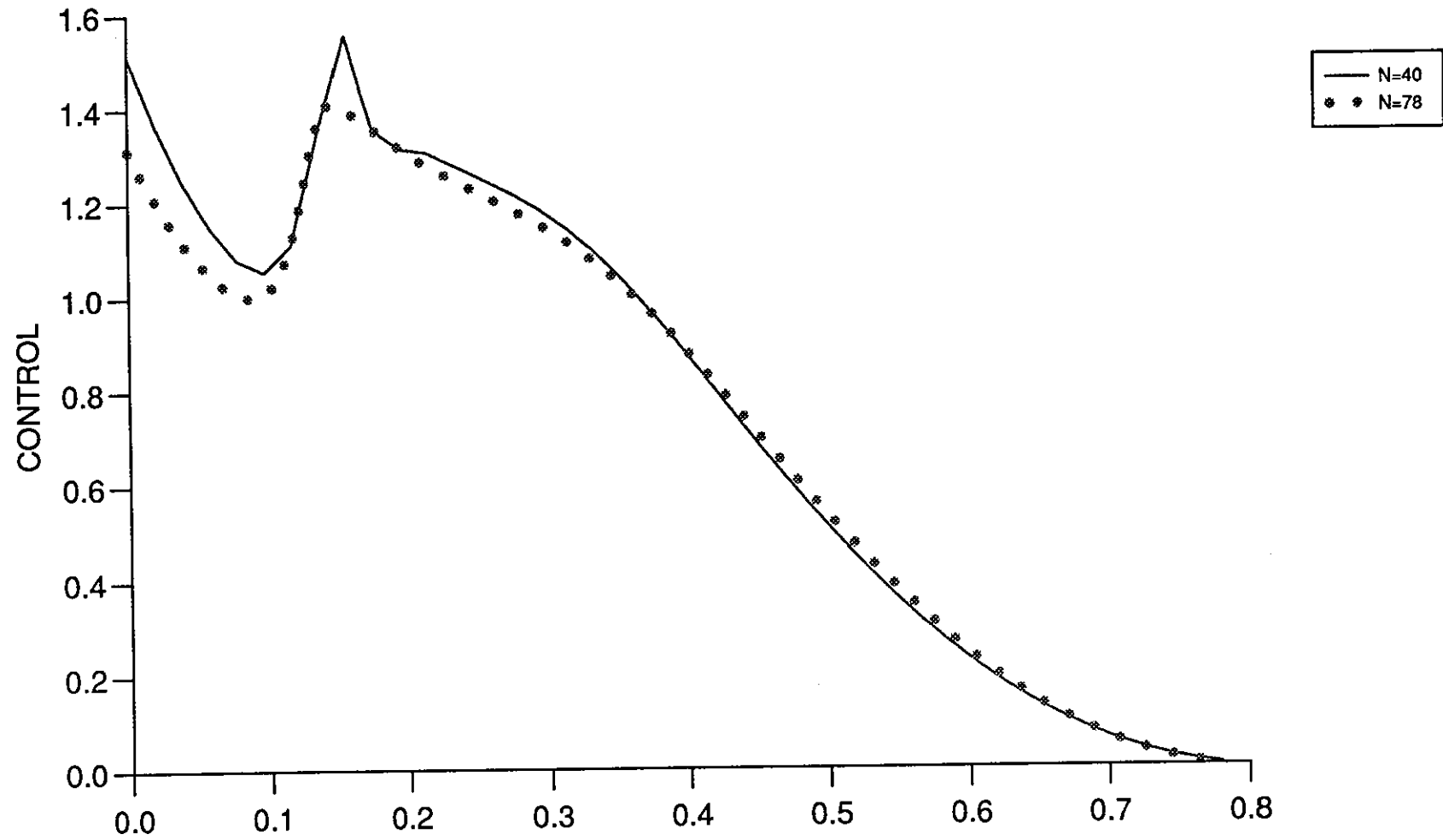


Figure (8.3.34)

H1

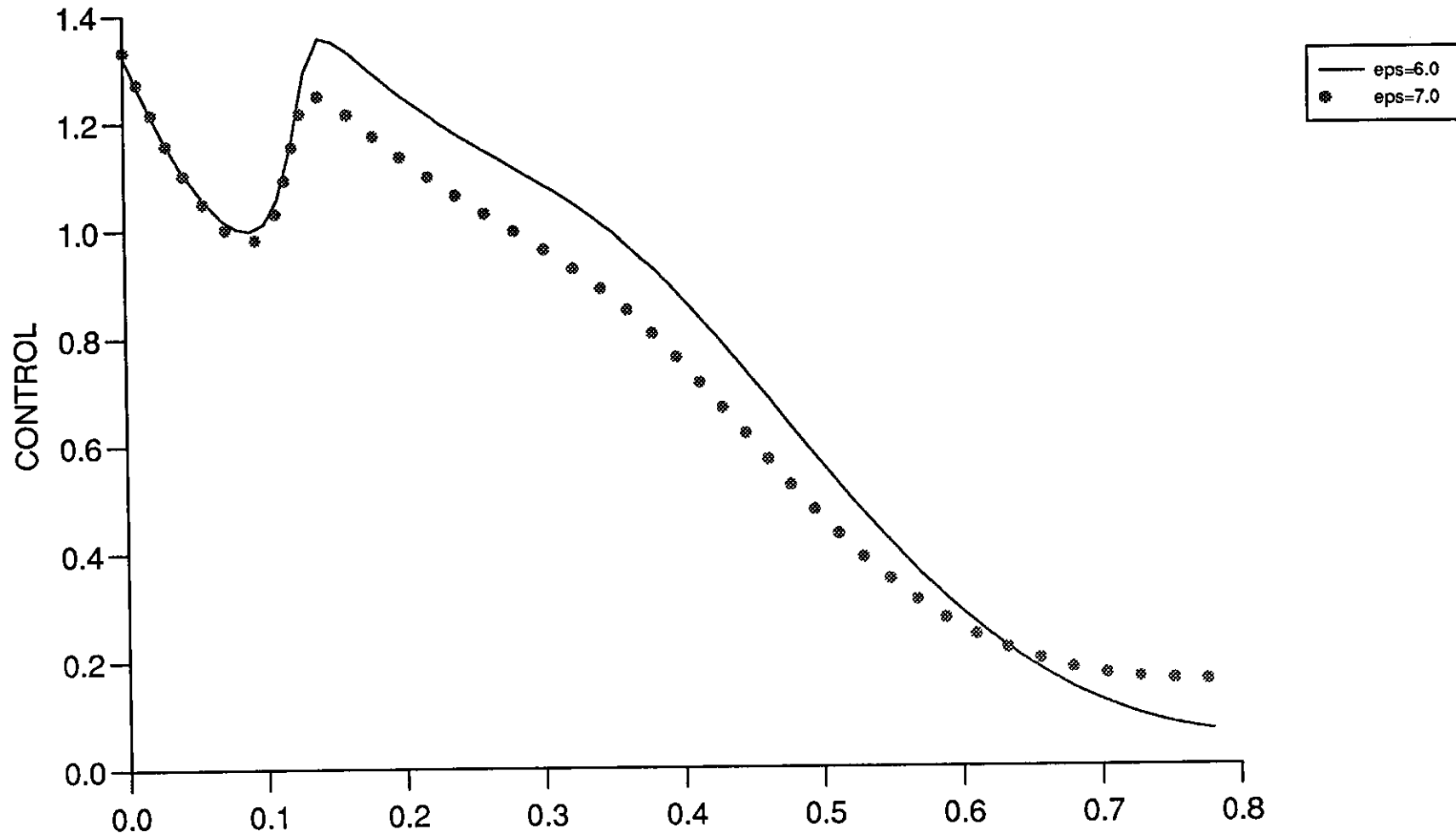


Figure (8.3.35)

H1

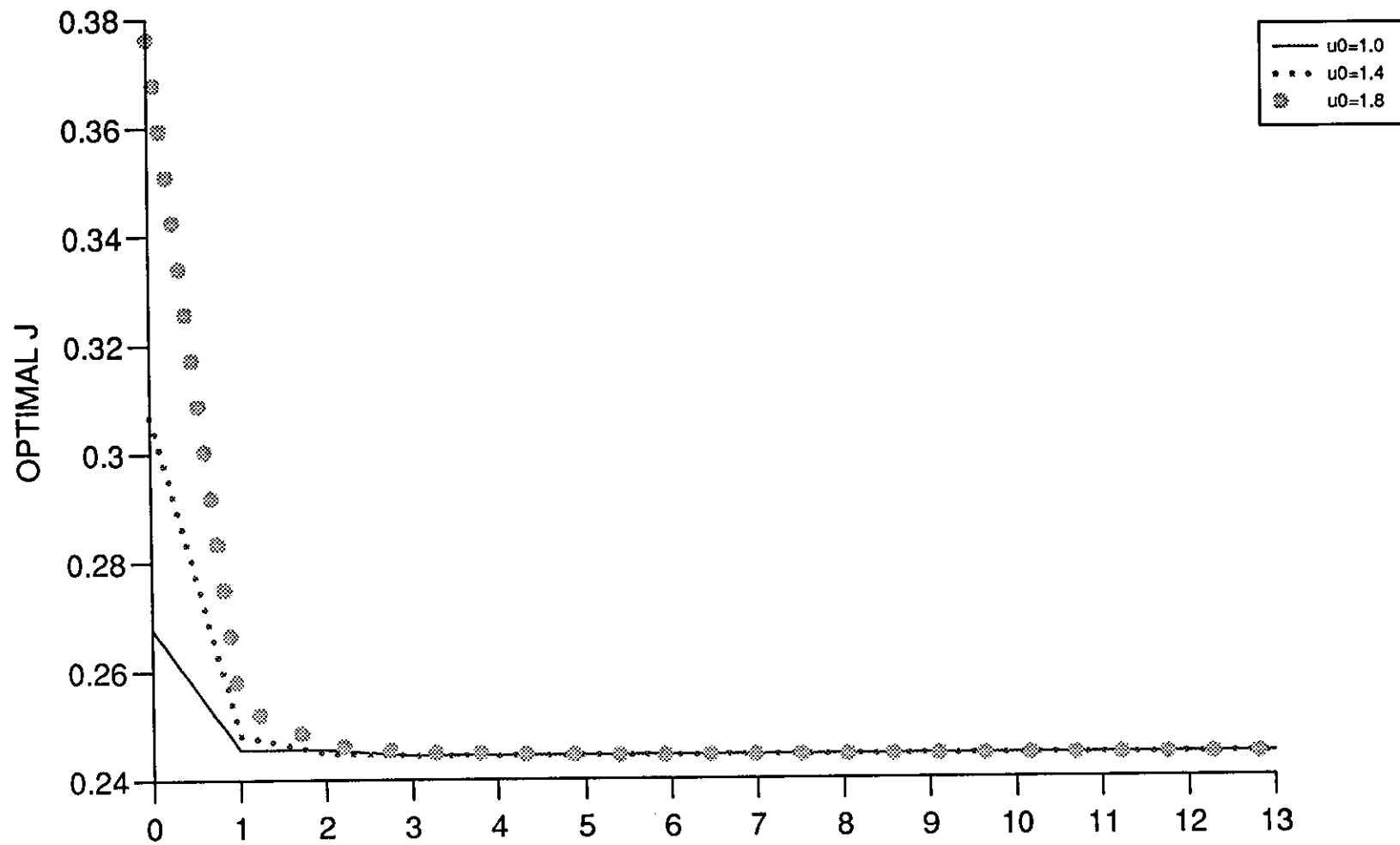


Figure (8.3.36)

H1

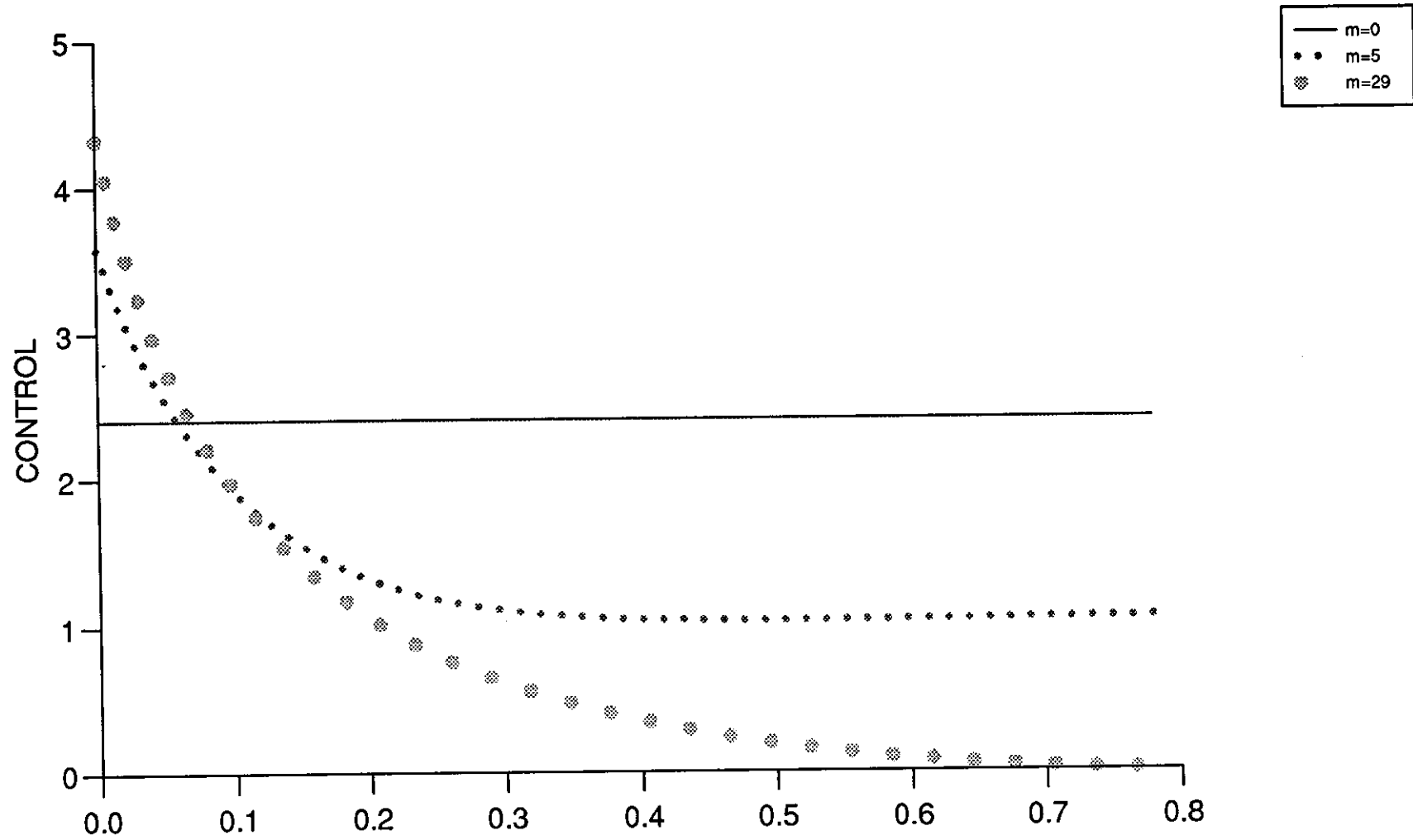


Figure (8.3.37)

H1

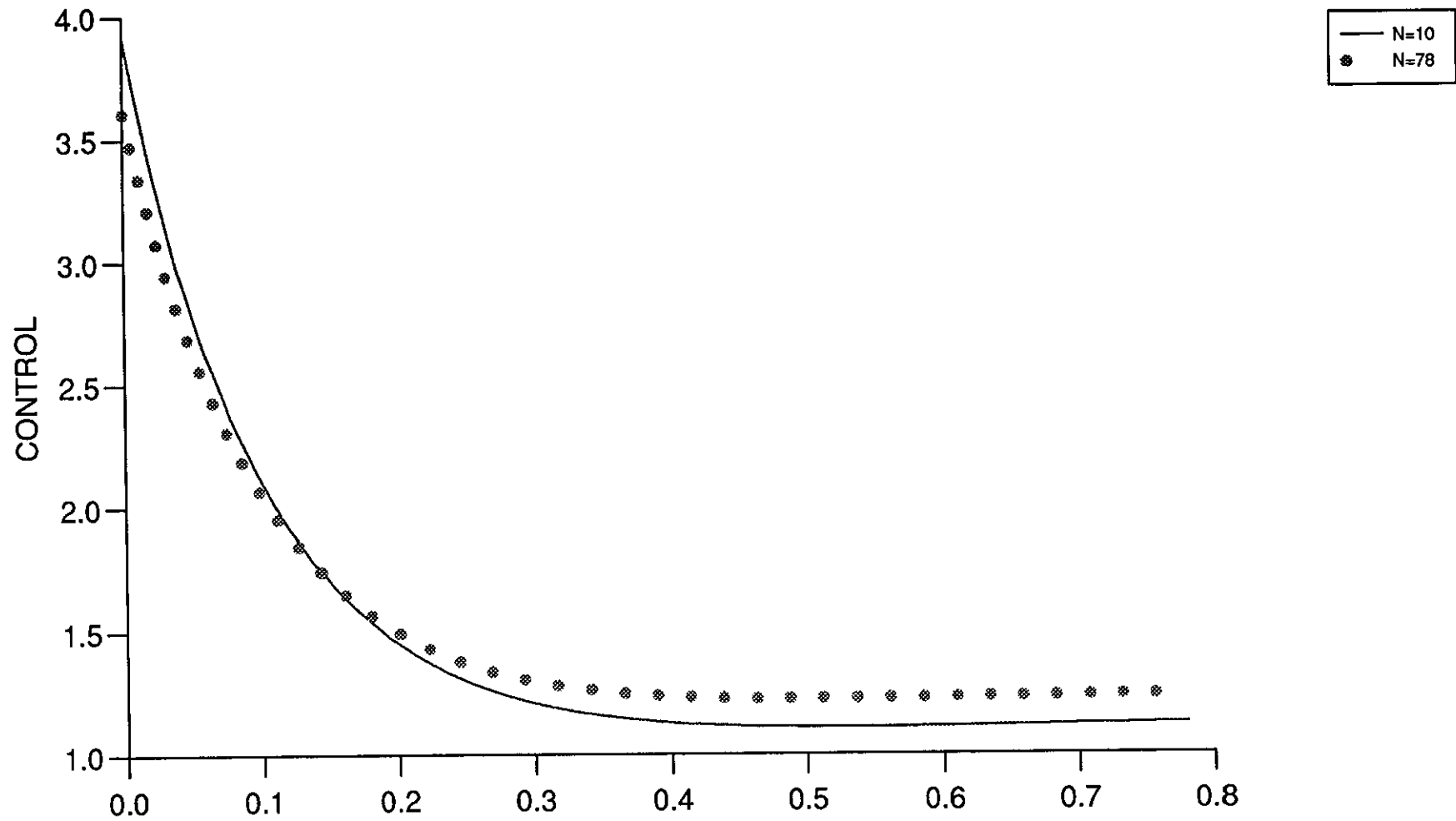


Figure (8.3.38)

H1

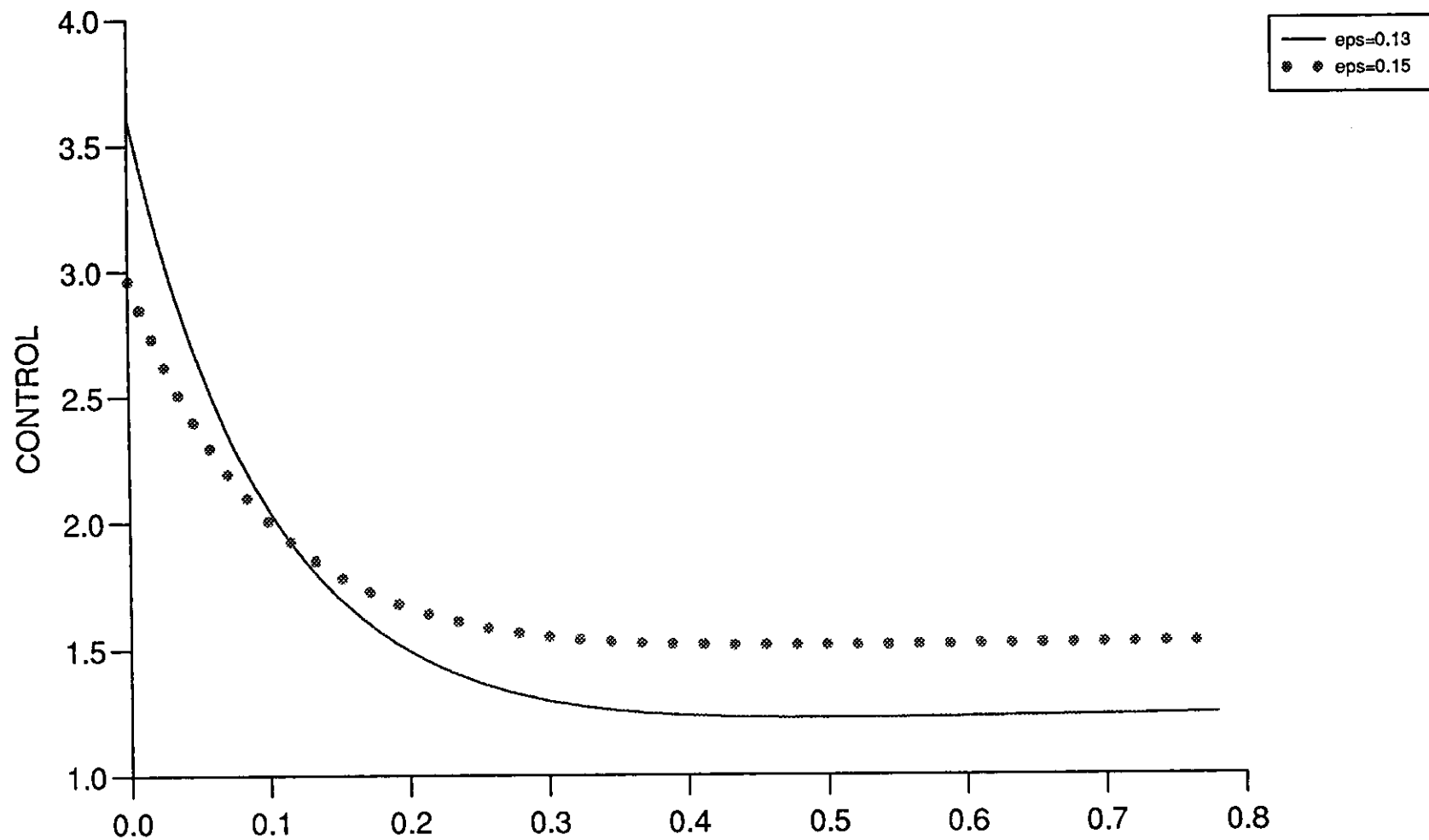


Figure (8.3.39)

H1

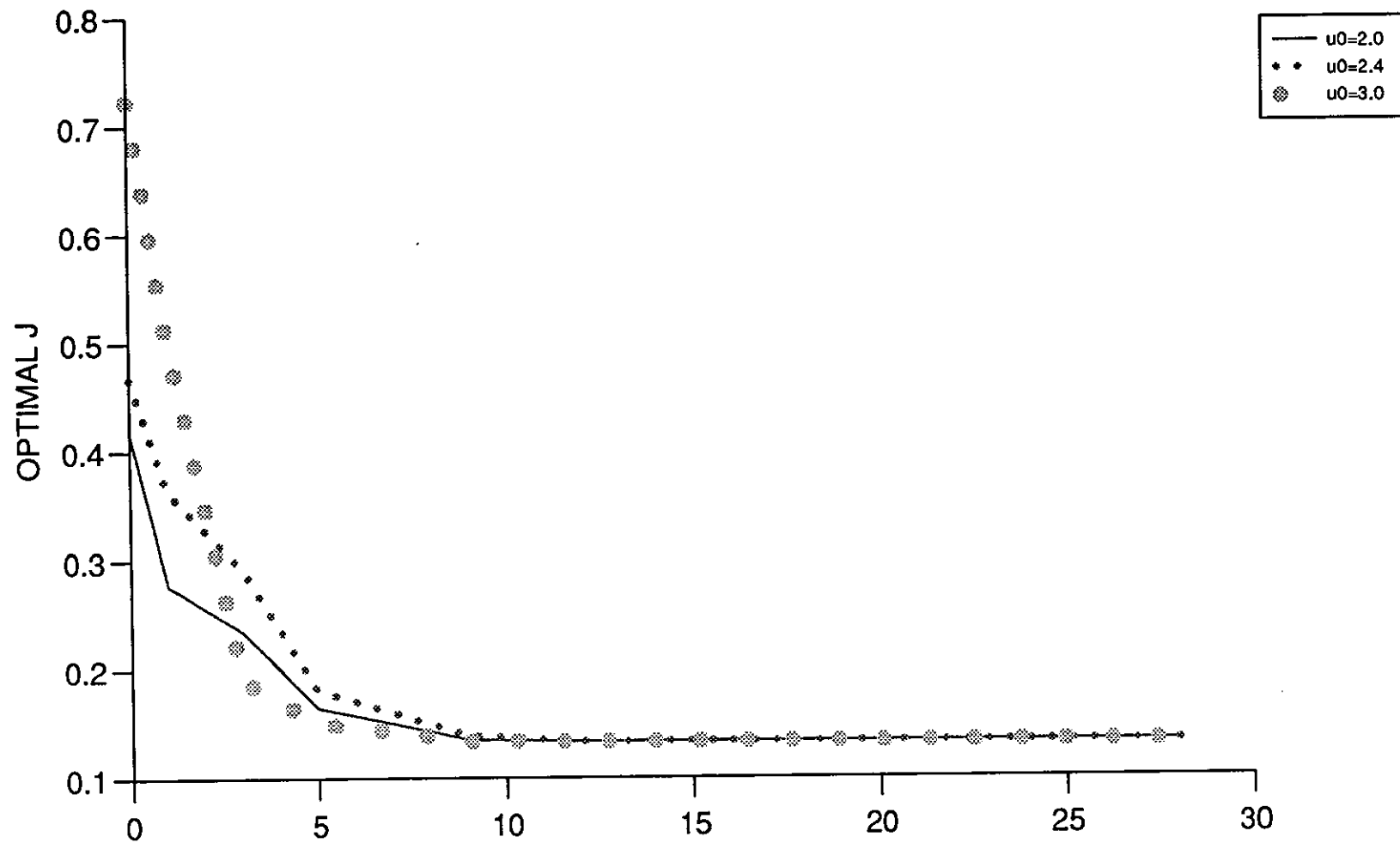


Figure (8.3.40)

ATH

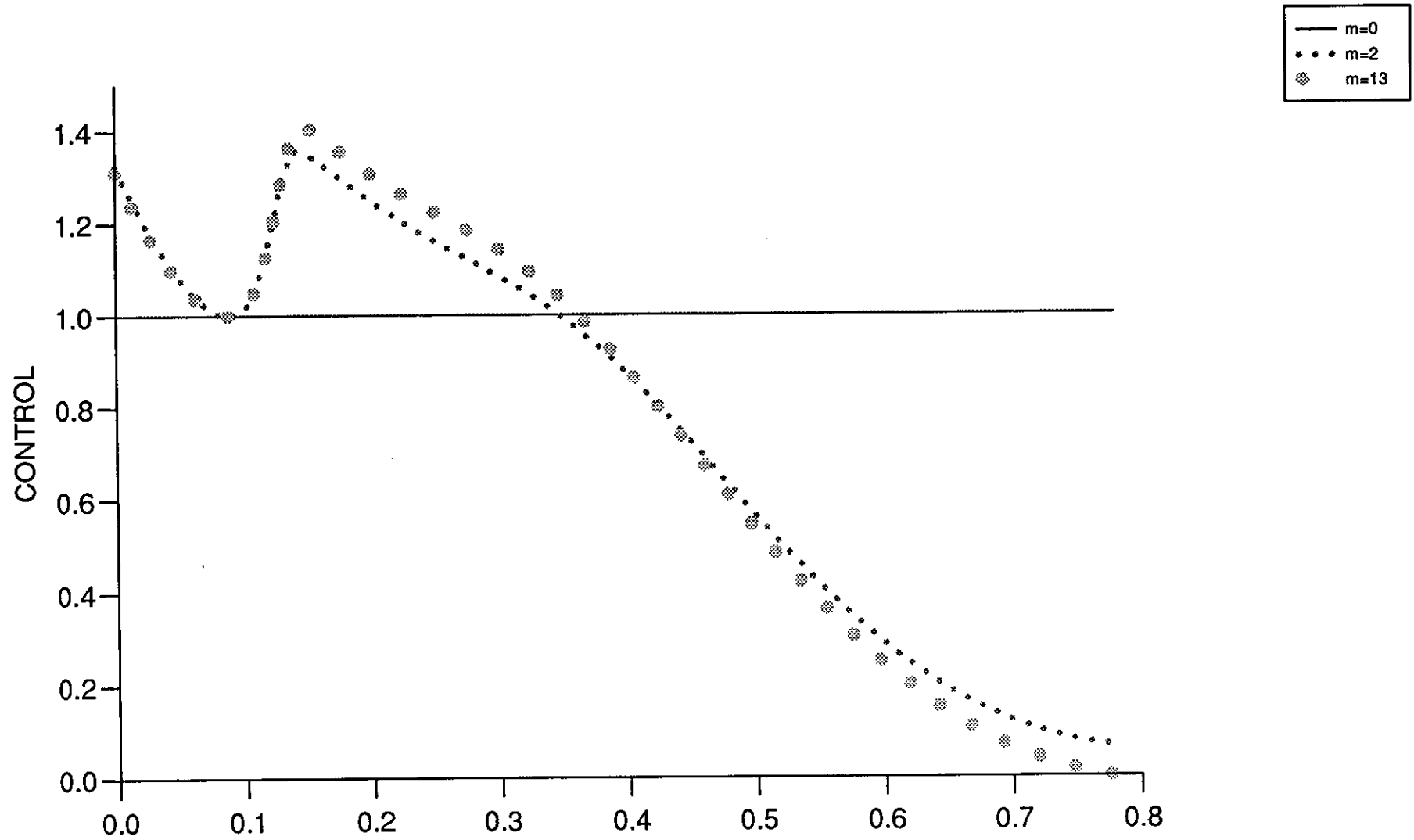


Figure (8.3.41)

ATH

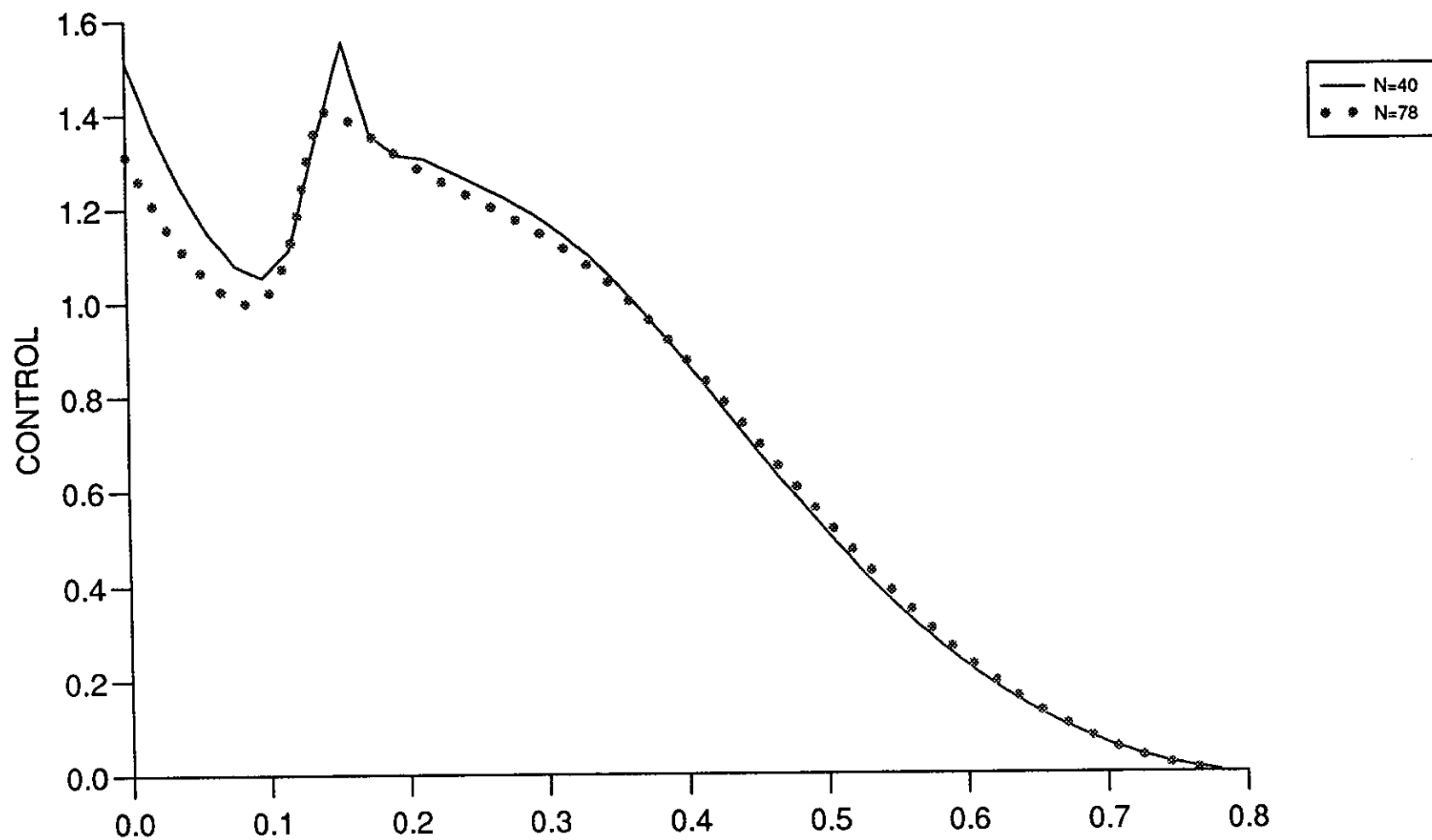


Figure (8.3.42)

ATH

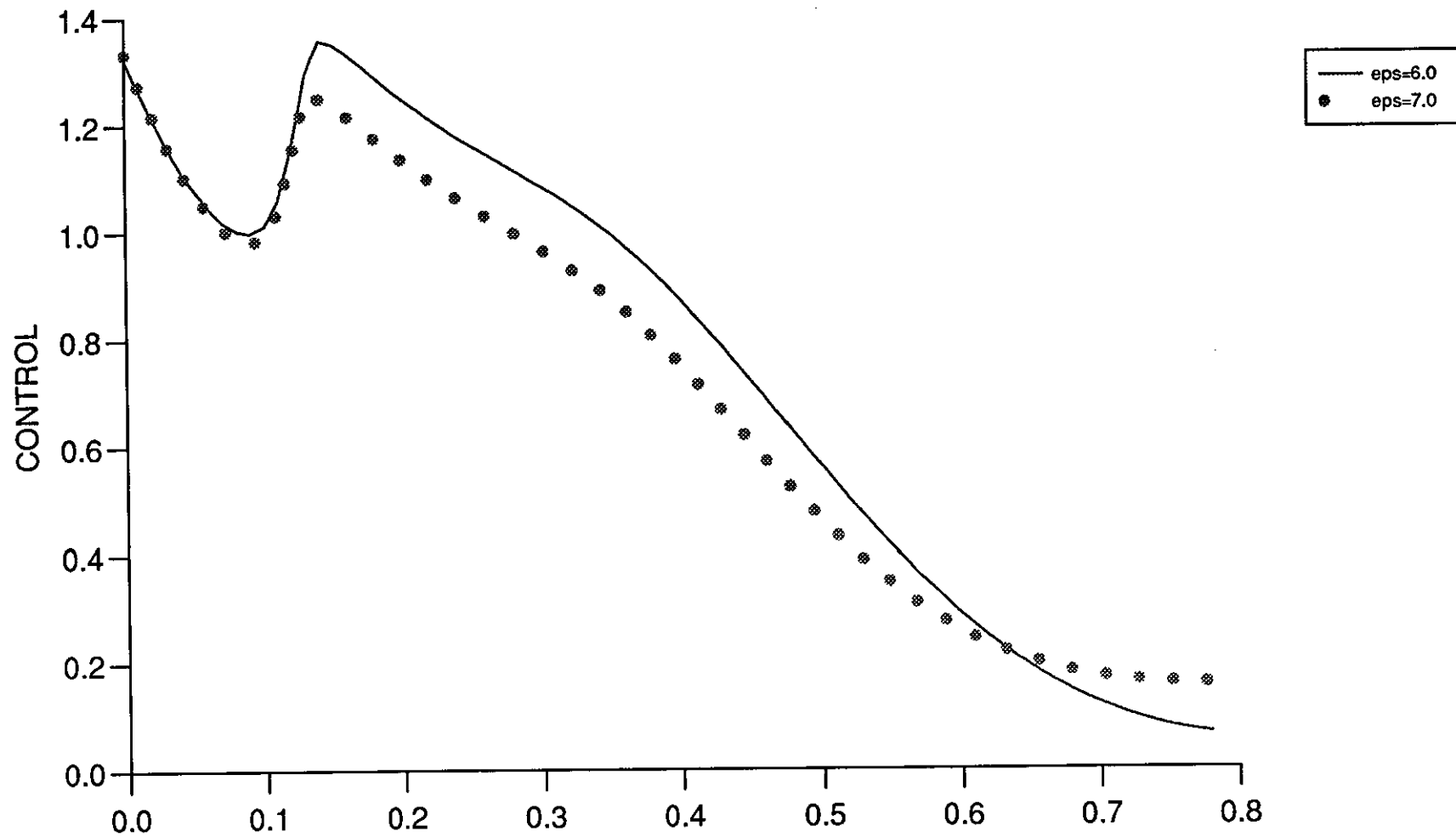


Figure (8.3.43)

ATH

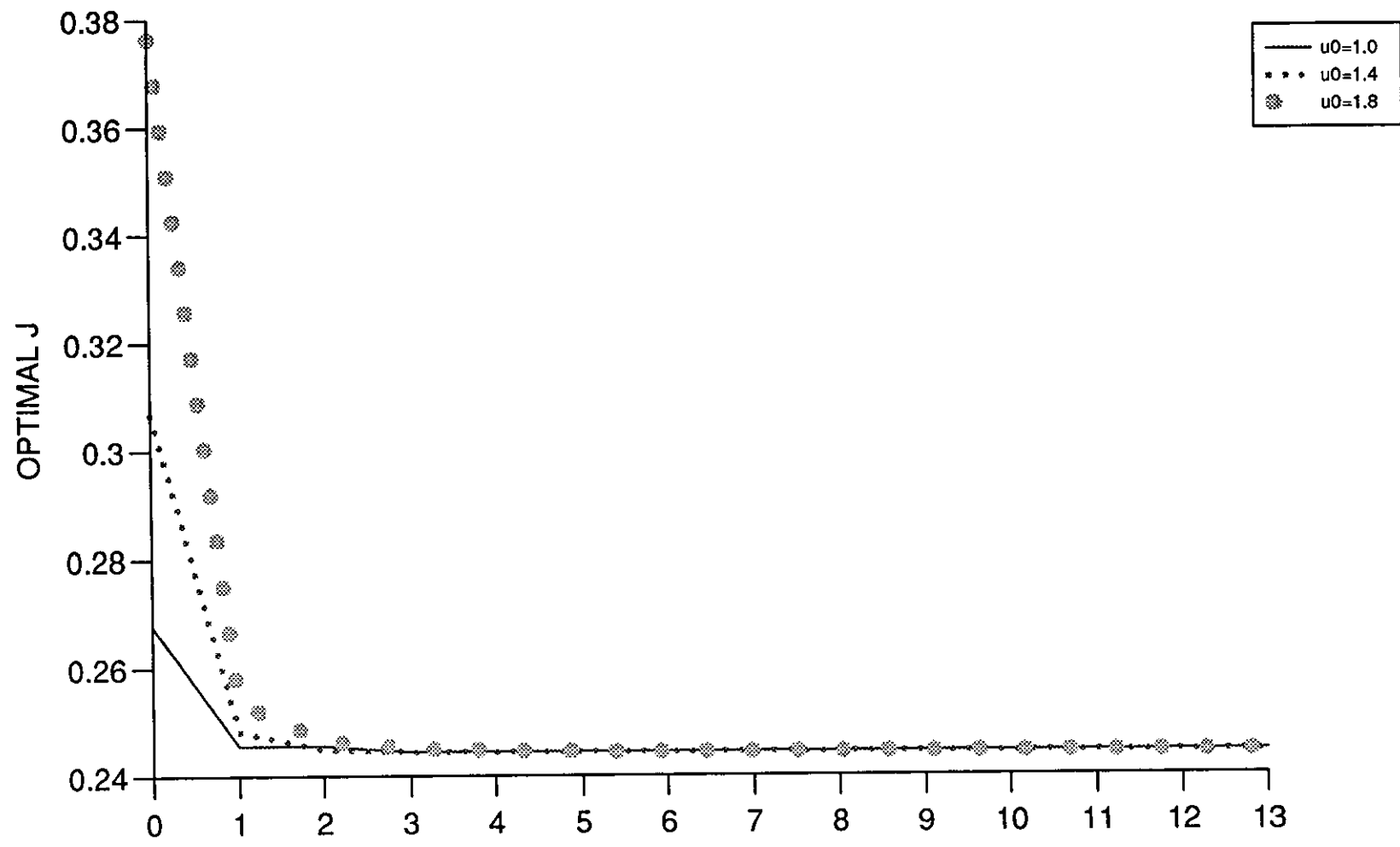


Figure (8.3.44)

ATH

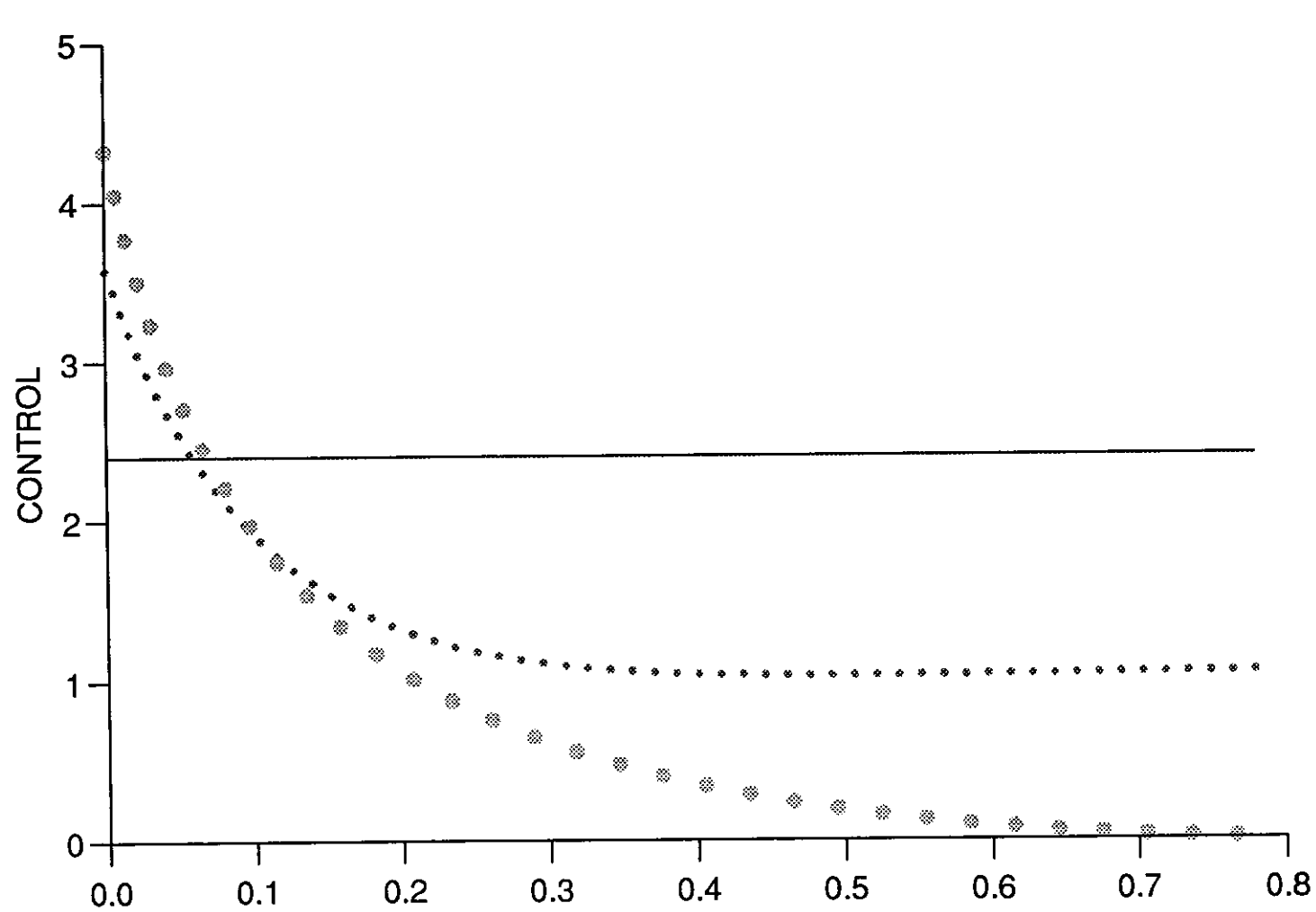


Figure (8.3.45)

ATH

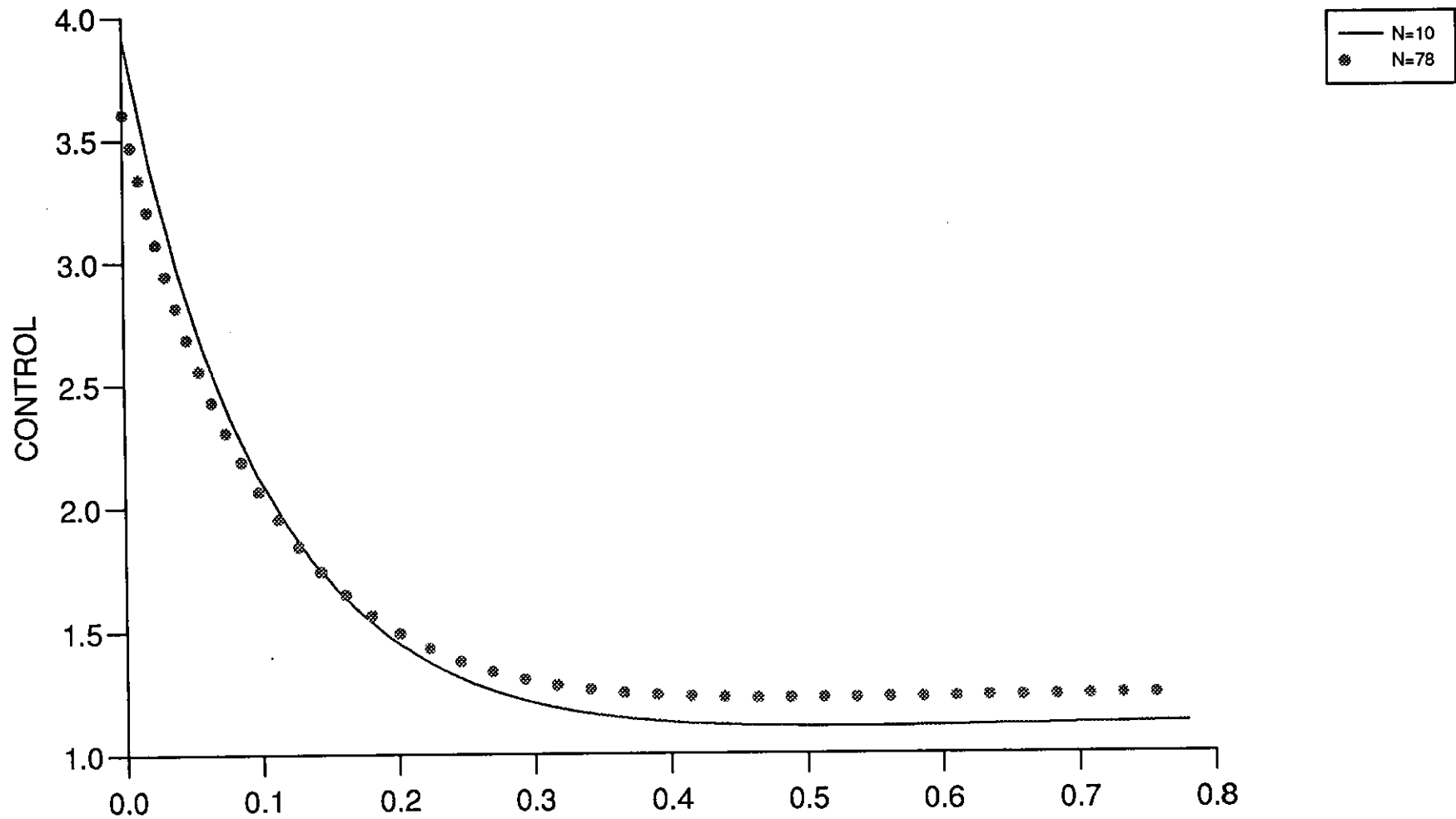


Figure (8.3.46)

ATH

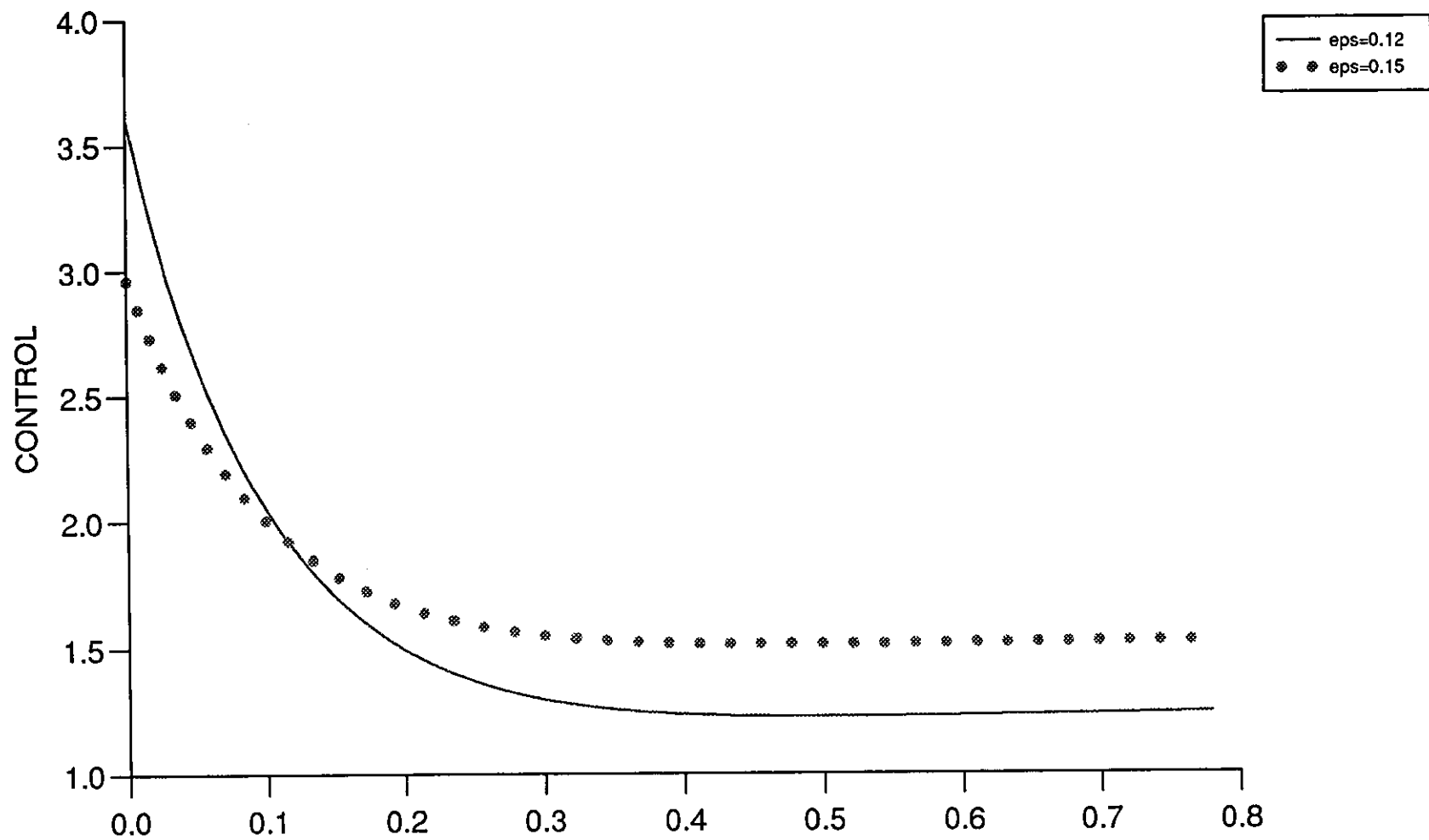


Figure (8.3.47)

ATH

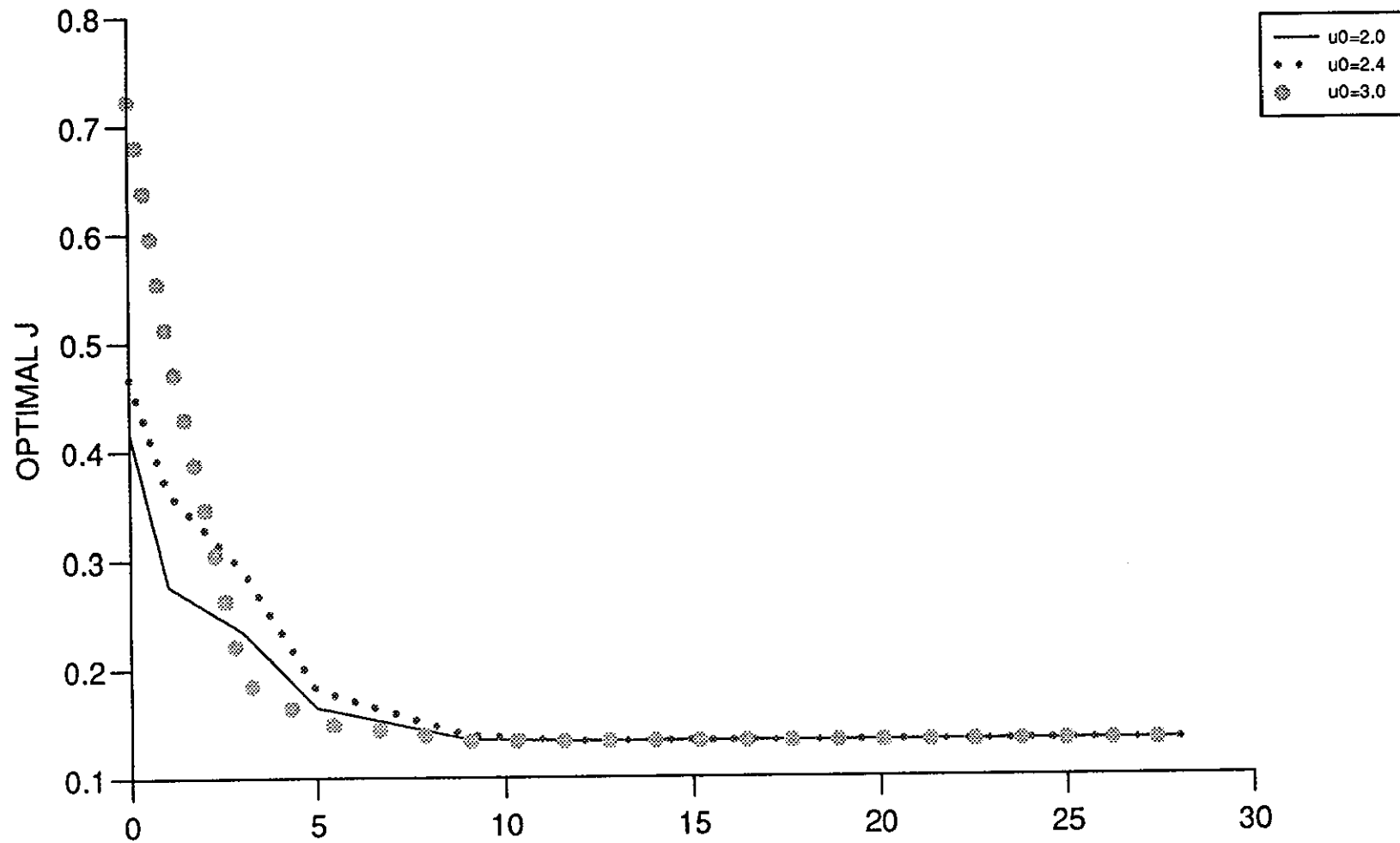


Figure (8.3.49)

H3

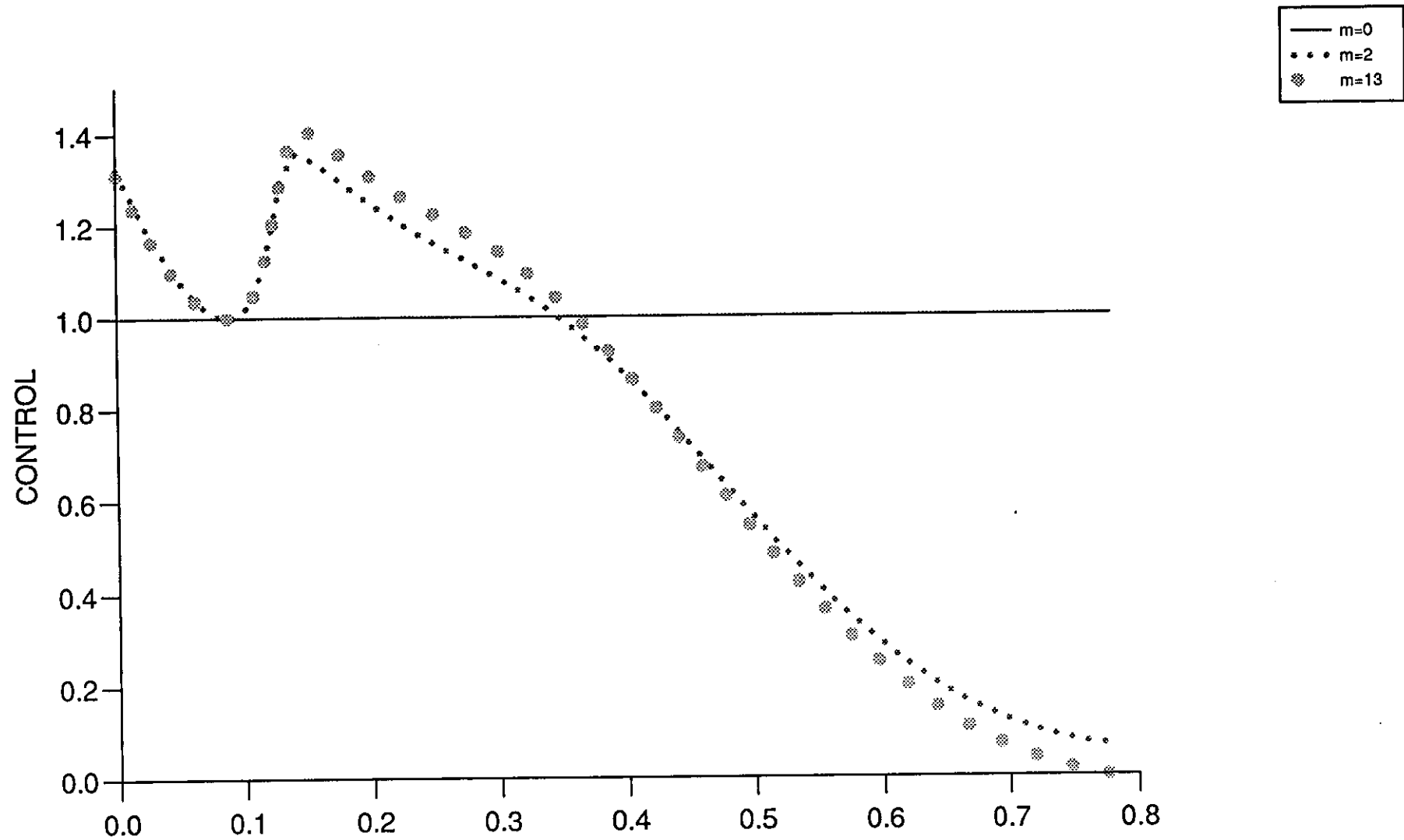


Figure (8.3.49)

H3

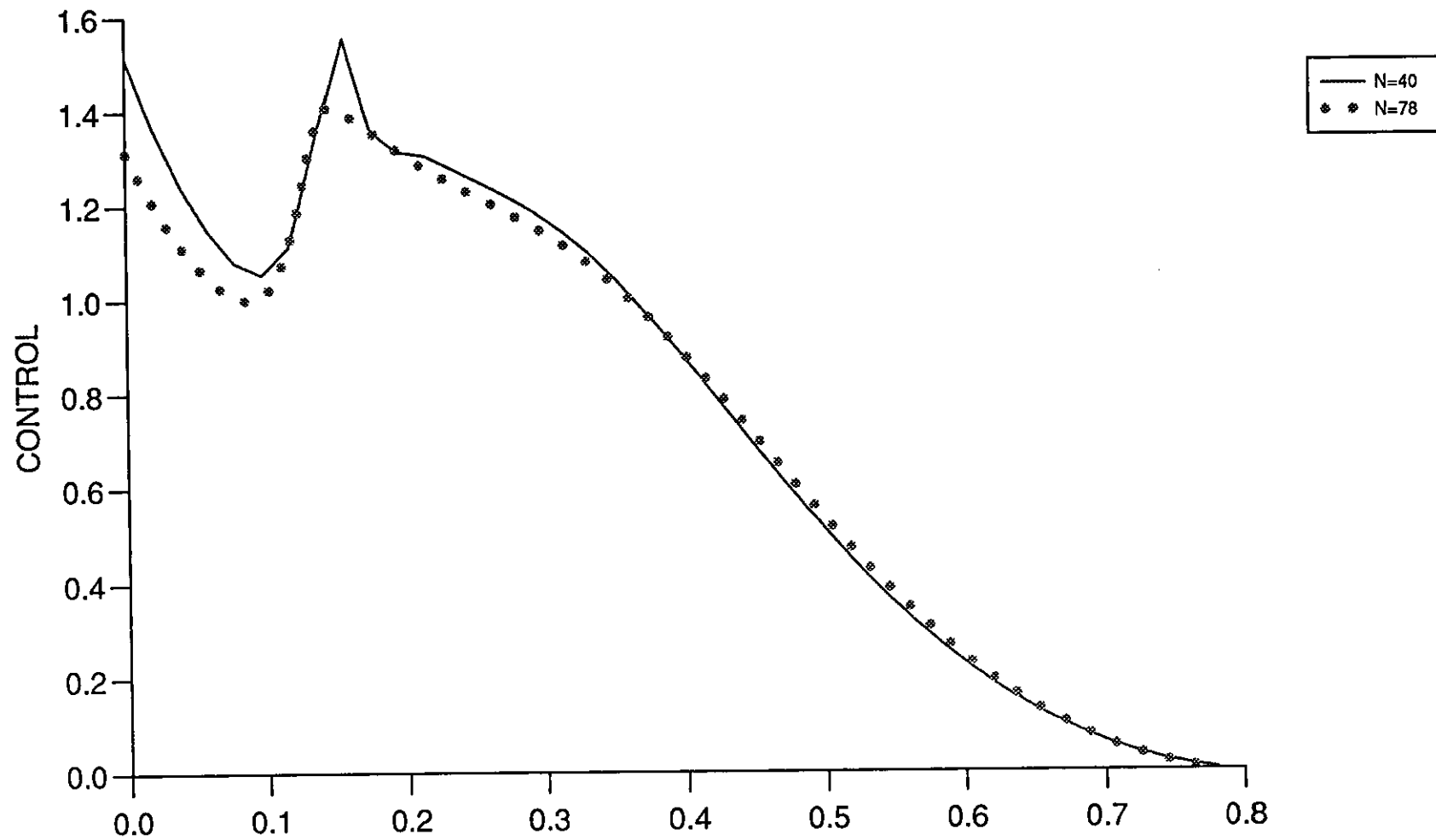


Figure (8.3.50)

H3

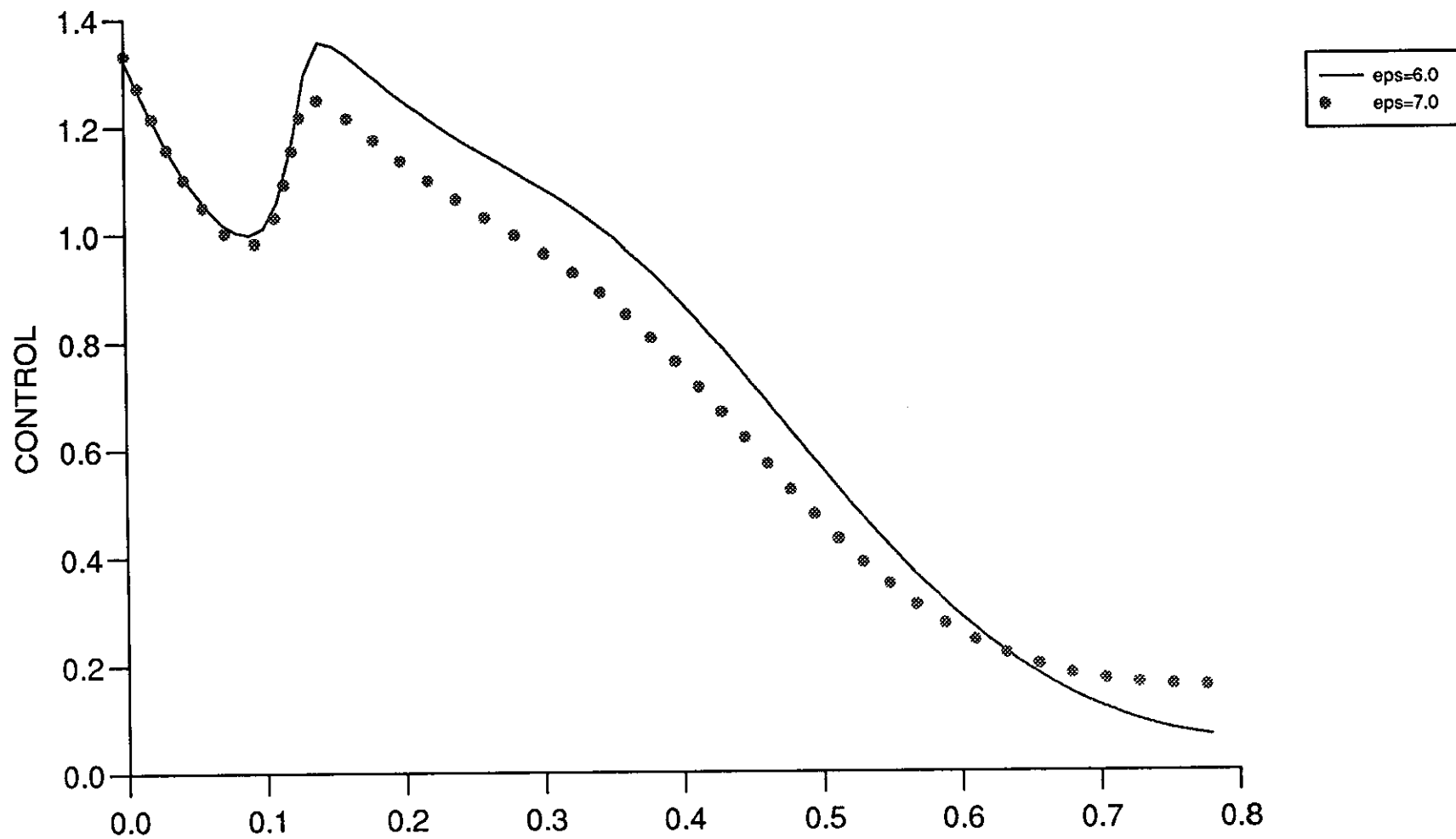


Figure (8.3.51)

H3

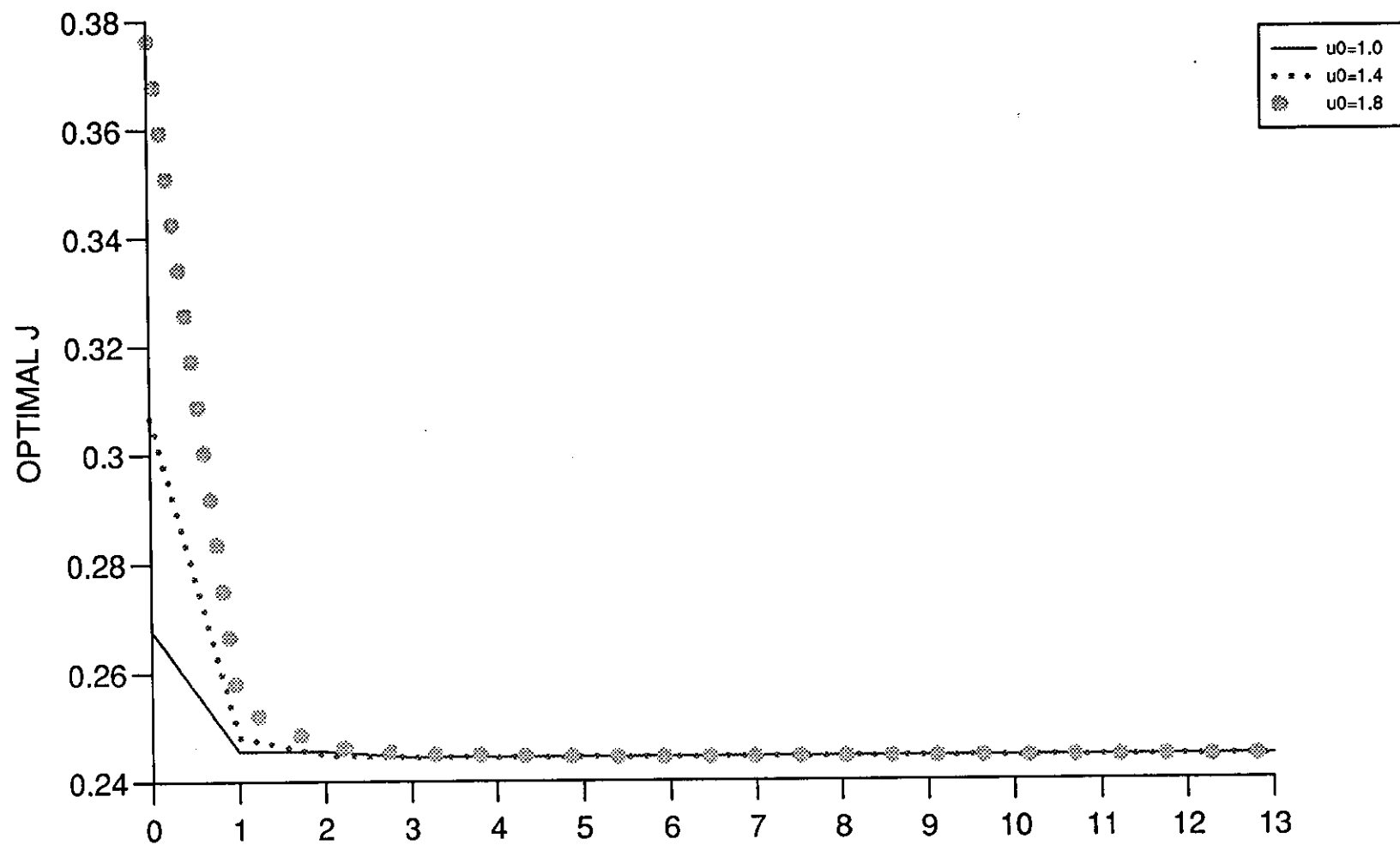


Figure (8.3.52)

H3

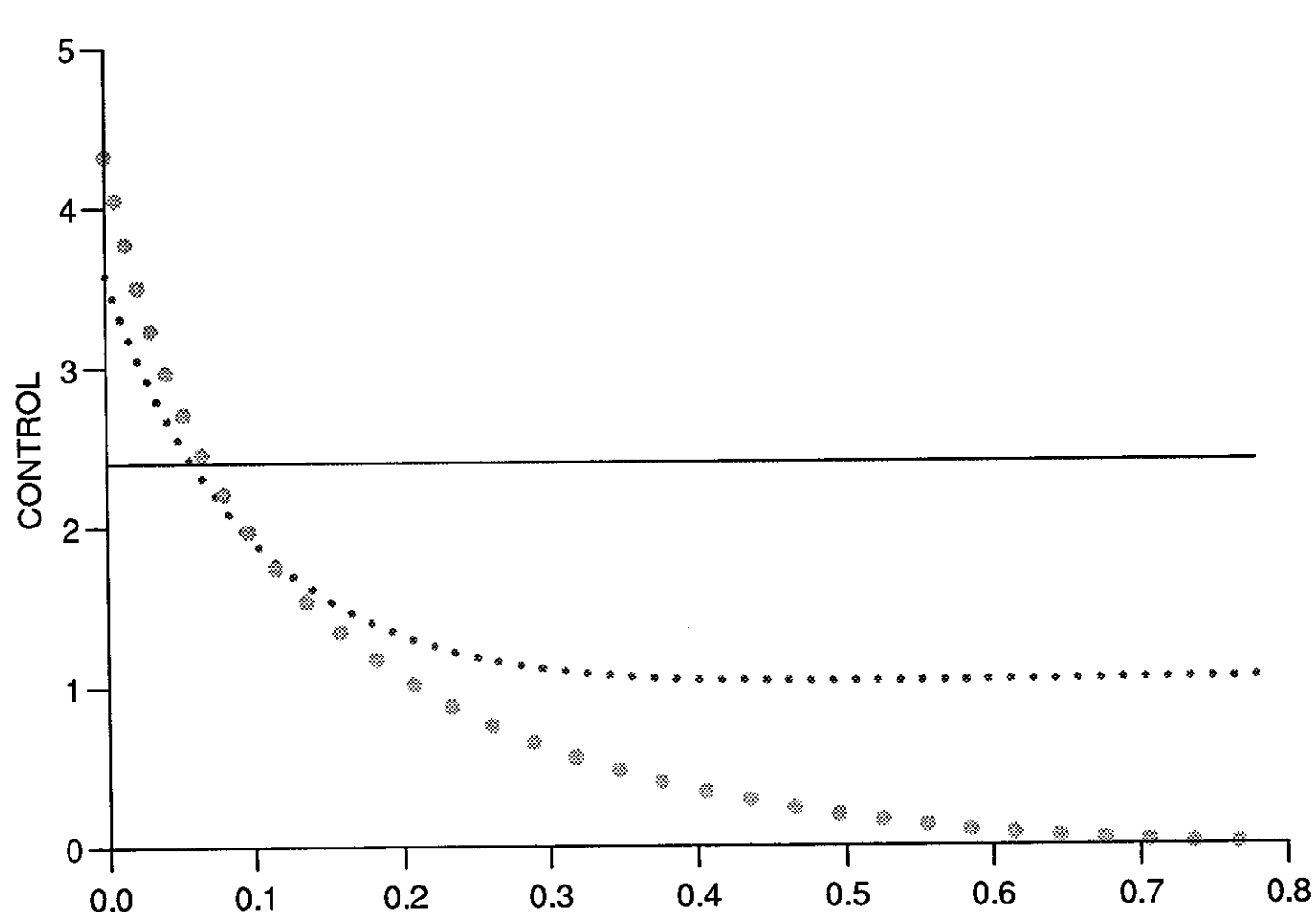


Figure (8.3.53)

H3

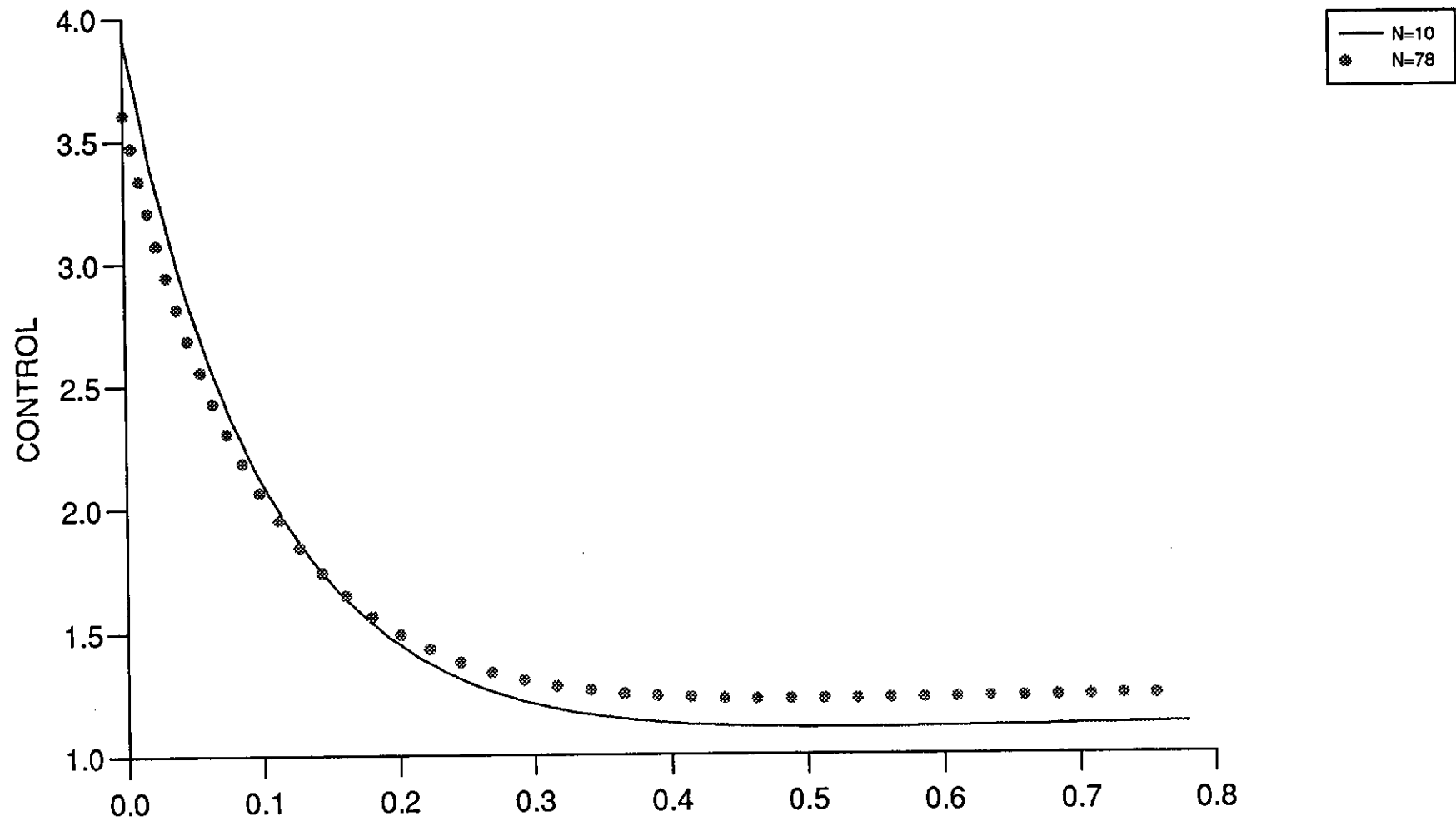


Figure (8.3.54)

H3

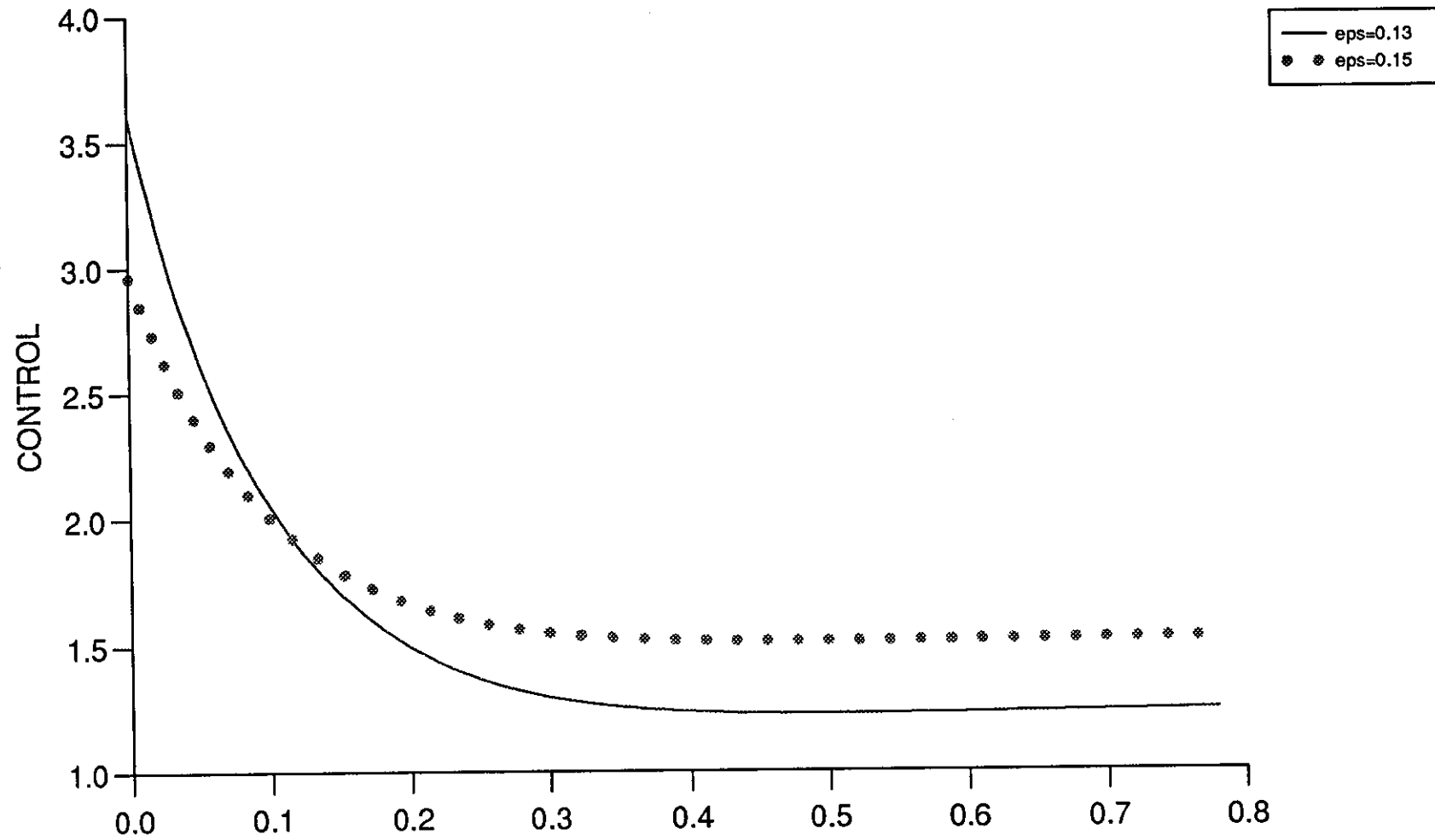


Figure (8.3.55)

H3

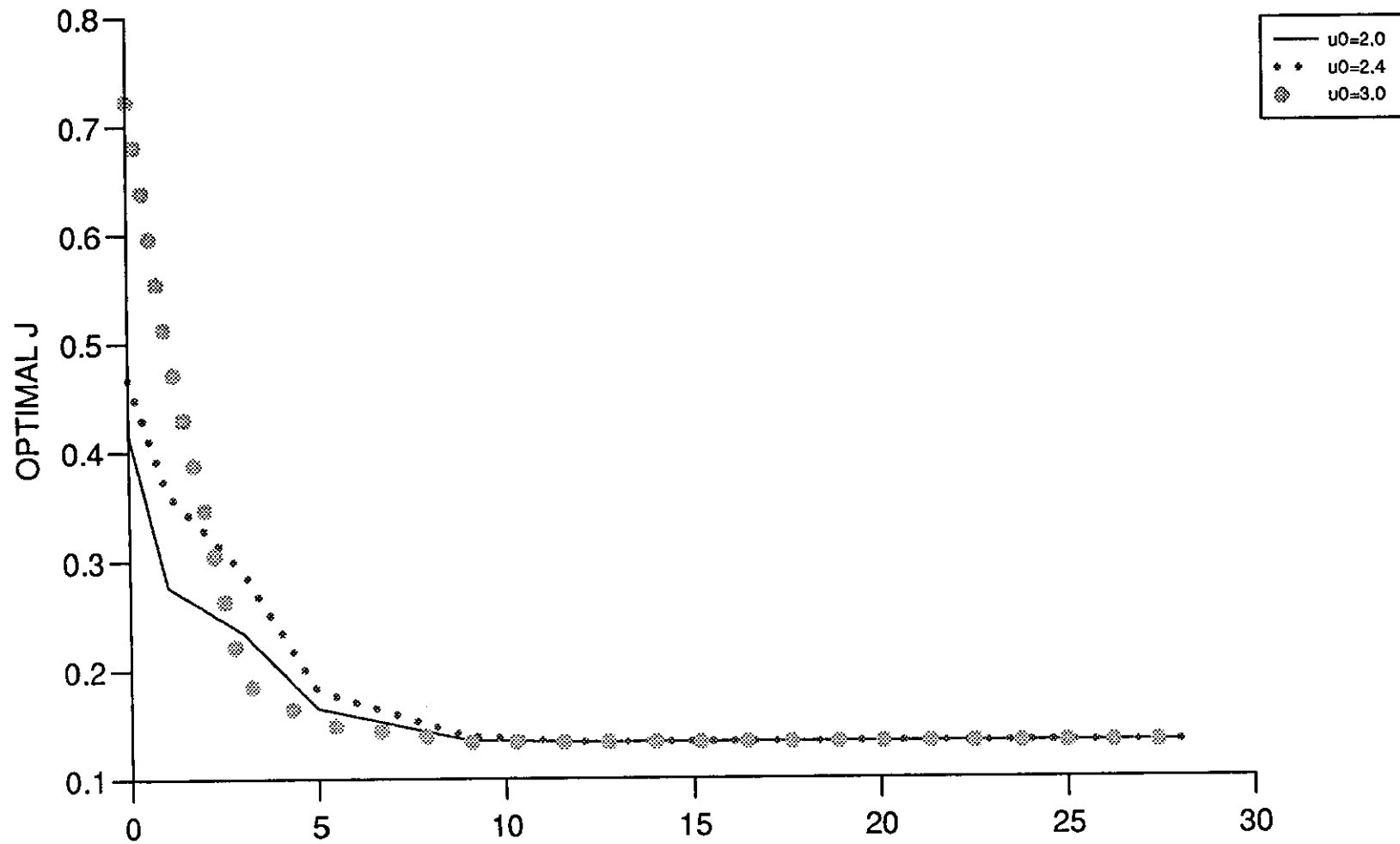


Figure (8.3.56)

COMPARISON OF THE SEVEN METHODS

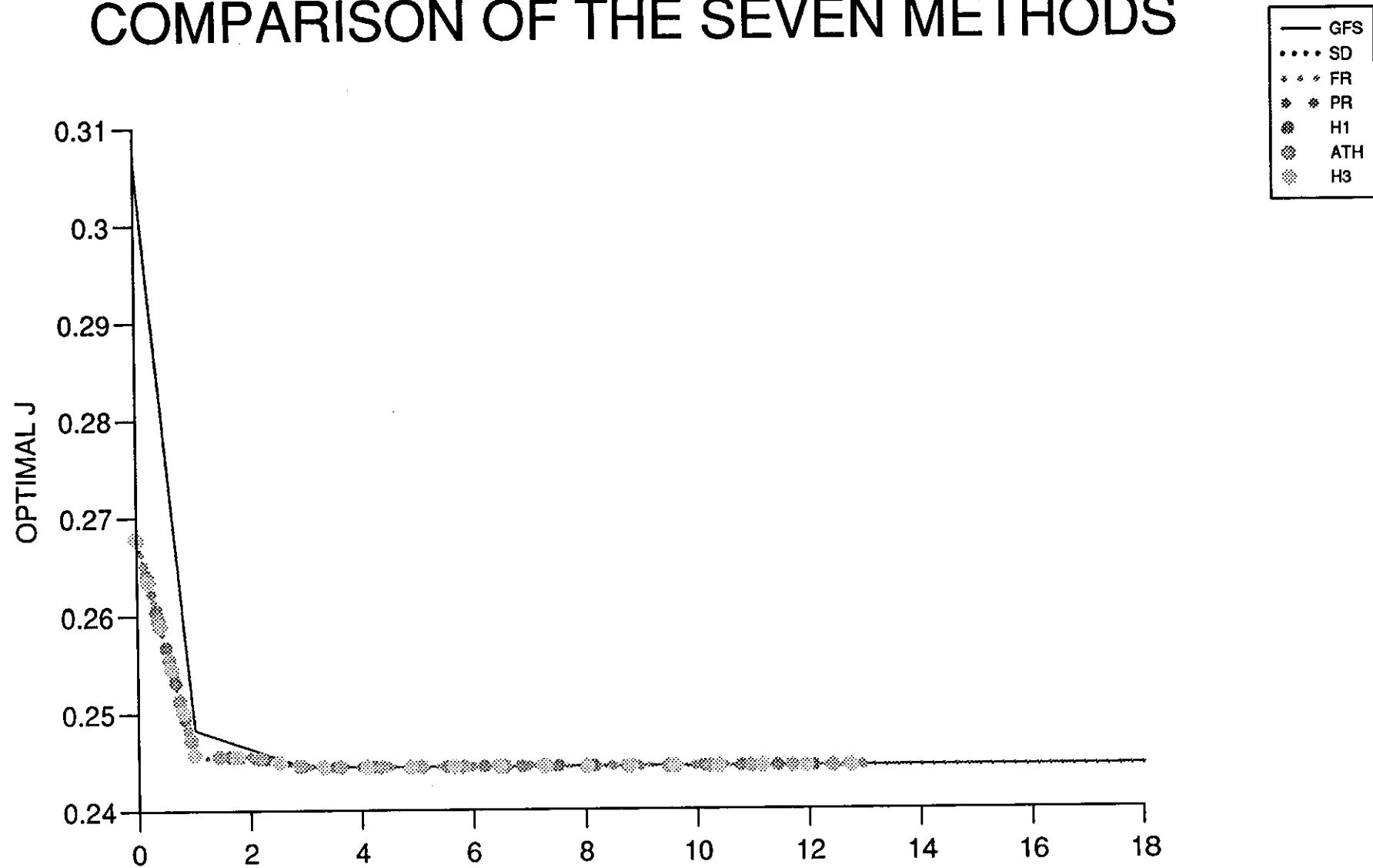


Figure (8.4.1)

COMPARISON OF THE SEVEN METHODS

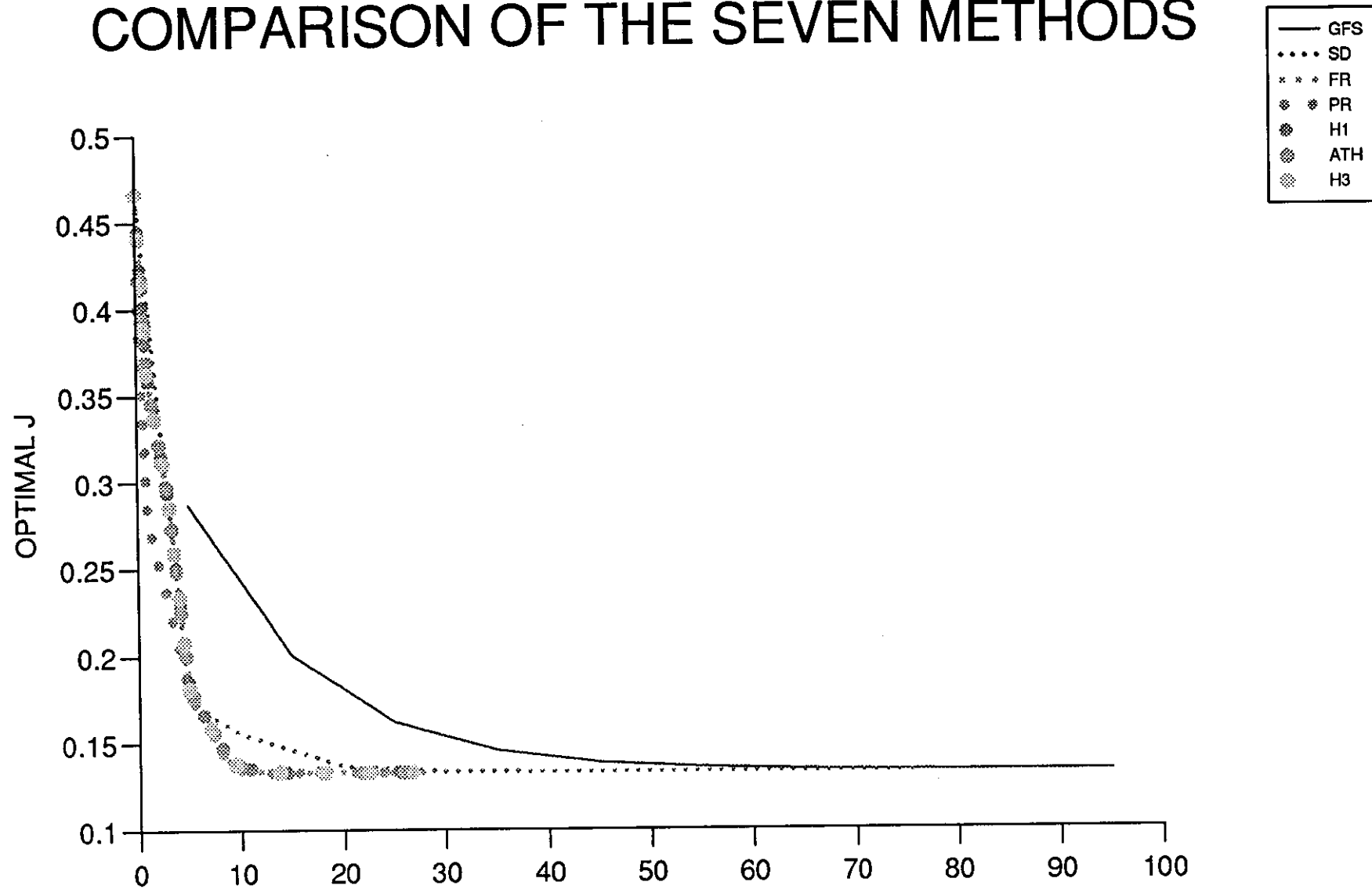


Figure (8.4.2)

Chapter 9

PROBLEM 6

9.1 A Nonlinear Singular Control Problem

Here we consider the problem used by Yeo (1980) [150] to illustrate the use of quasi-linearization to solve non linear control problems and then by R. Laus (1990) [22] using discrete dynamic programming with systematic reduction in grid size.

The non-linear singular control problem is described by the state equations:

$$\dot{x}_1 = x_2, \quad x_1(0) = 0, \quad (9.1.1)$$

$$\dot{x}_2 = -x_3u + 16t - 8, \quad x_2(0) = -1, \quad (9.1.2)$$

$$\dot{x}_3 = u, \quad x_3(0) = -\sqrt{5}. \quad (9.1.3)$$

The performance index is

$$J = \int_0^1 [x_1^2 + x_2^2 + 0.0005(x_2 + 16t - 8 - 0.1x_3u^2)]^2 dt. \quad (9.1.4)$$

We are to find the control u in the time interval $0 \leq t \leq 1$ such that the performance index J is minimized. The control is bounded by

$$-4 \leq u \leq 10 \quad (9.1.5)$$

Now by introducing new state variables x_4 and x_5 we transform the problem into,

$$\min J = \phi(x_4(1)), \quad (9.1.6)$$

Subject to,

$$f_1 = \dot{x}_1 = x_2, \quad x_1(0) = 0, \quad (9.1.7)$$

$$f_2 = \dot{x}_2 = -x_3u + 16x_5 - 8, \quad x_2(0) = -1, \quad (9.1.8)$$

$$f_3 = \dot{x}_3 = u, \quad x_3(0) = -\sqrt{5}, \quad (9.1.9)$$

$$f_4 = \dot{x}_4 = x_1^2 + x_2^2 + 0.0005(x_2 + 16x_5 - 8 - 0.1x_3u^2)^2, \quad x_4(0) = 0, \quad (9.1.10)$$

$$f_5 = \dot{x}_5 = 1, \quad x_5(0) = 0, \quad (9.1.11)$$

with,

$$0 \leq t \leq 1.$$

The Hamiltonian is,

$$H = \sum_{i=1}^5 \lambda_i f_i = \lambda_1 f_1 + \lambda_2 f_2 + \lambda_3 f_3 + \lambda_4 f_4 + \lambda_5 f_5. \quad (9.1.12)$$

The adjoint equation are,

$$\dot{\lambda}_i = - \sum_{j=1}^5 \lambda_j \frac{\partial f_j}{\partial x_i},$$

with,

$$\lambda_i(t_f) = \left. \frac{\partial \phi}{\partial x_i} \right|_{t=t_f}, \quad i = 1, 2, 3, 4, 5.$$

i.e.,

$$\begin{aligned} \dot{\lambda}_1 &= - \left[\lambda_1 \frac{\partial f_1}{\partial x_1} + \lambda_2 \frac{\partial f_2}{\partial x_2} + \lambda_3 \frac{\partial f_3}{\partial x_1} + \lambda_4 \frac{\partial f_4}{\partial x_1} + \lambda_5 \frac{\partial f_5}{\partial x_1} \right], \\ &= - [2\lambda_4 x_1], \end{aligned} \quad (9.1.13)$$

$$\begin{aligned} \dot{\lambda}_2 &= - \left[\lambda_1 \frac{\partial f_1}{\partial x_2} + \lambda_2 \frac{\partial f_2}{\partial x_2} + \lambda_3 \frac{\partial f_3}{\partial x_2} + \lambda_4 \frac{\partial f_4}{\partial x_2} + \lambda_5 \frac{\partial f_5}{\partial x_2} \right], \\ &= - \{ \lambda_2 + \lambda_4 \{ 2x_2 + 0.0005 [2x_2 - 0.2x_3 u^2 + 32x_5 - 16] \} \}, \end{aligned} \quad (9.1.14)$$

$$\begin{aligned} \dot{\lambda}_3 &= - \left[\lambda_1 \frac{\partial f_1}{\partial x_3} + \lambda_2 \frac{\partial f_2}{\partial x_3} + \lambda_3 \frac{\partial f_3}{\partial x_3} + \lambda_4 \frac{\partial f_4}{\partial x_3} + \lambda_5 \frac{\partial f_5}{\partial x_3} \right], \\ &= - \{ -\lambda_2 u + 0.0005 \lambda_4 [-3.2x_5 u^2 + 1.6u^2 + 0.02u^4 x_3 - 0.2x_2 u^2] \}, \end{aligned} \quad (9.1.15)$$

$$\begin{aligned} \dot{\lambda}_4 &= - \left[\lambda_1 \frac{\partial f_1}{\partial x_4} + \lambda_2 \frac{\partial f_2}{\partial x_4} + \lambda_3 \frac{\partial f_3}{\partial x_4} + \lambda_4 \frac{\partial f_4}{\partial x_4} + \lambda_5 \frac{\partial f_5}{\partial x_4} \right], \\ &= 0, \end{aligned} \quad (9.1.16)$$

$$\begin{aligned} \dot{\lambda}_5 &= - \left[\lambda_1 \frac{\partial f_1}{\partial x_5} + \lambda_2 \frac{\partial f_2}{\partial x_5} + \lambda_3 \frac{\partial f_3}{\partial x_5} + \lambda_4 \frac{\partial f_4}{\partial x_5} + \lambda_5 \frac{\partial f_5}{\partial x_5} \right], \\ &= - \{ 16\lambda_2 + 0.0005 \lambda_4 [-3.2x_3 u^2 + 512x_5 + 32x_2 - 256] \}, \end{aligned} \quad (9.1.17)$$

with,

$$\lambda_1(t_f) = \left. \frac{\partial \phi}{\partial x_1} \right|_{t=t_f=1} = \left. \frac{\partial (x_4(1))}{\partial x_1} \right|_{t=1} = 0, \quad (9.1.18)$$

$$\lambda_2(t_f) = \left. \frac{\partial \phi}{\partial x_2} \right|_{t=t_f=1} = \left. \frac{\partial (x_4(1))}{\partial x_2} \right|_{t=1} = 0, \quad (9.1.19)$$

$$\lambda_3(t_f) = \left. \frac{\partial \phi}{\partial x_3} \right|_{t=t_f=1} = \left. \frac{\partial (x_4(1))}{\partial x_3} \right|_{t=1} = 0, \quad (9.1.20)$$

$$\lambda_4(t_f) = \left. \frac{\partial \phi}{\partial x_4} \right|_{t=t_f=1} = \left. \frac{\partial (x_4(1))}{\partial x_3} \right|_{t=1} = 1, \quad (9.1.21)$$

$$\lambda_5(t_f) = \left. \frac{\partial \phi}{\partial x_5} \right|_{t=t_f=1} = \left. \frac{\partial (x_5(1))}{\partial x_5} \right|_{t=1} = 0. \quad (9.1.22)$$

Here, before proceeding to solve the problem numerically, we should note that, singular arc occurs at $x_3 = 0$, where $\det(H_{uu})$ vanishes at any point along it. Here also on the bounded control if $-4 \leq u \leq 10$, the following conditions must be satisfied;

$$\left. \begin{aligned} u = 10 &\Rightarrow \frac{\partial H}{\partial u} \leq 0, \\ -4 < u < 10 &\Rightarrow \frac{\partial H}{\partial u} = 0, \\ u = -4 &\Rightarrow \frac{\partial H}{\partial u} \geq 0. \end{aligned} \right\} \quad (9.1.23)$$

On these intervals, u is truncated, such that if $u > 10$, u is set equal to 10, and if $u < -4$, u is set to equal -4 .

9.2 Numerical Solutions

9.2.1 The state and adjoint equations

The state equations are;

$$\left. \begin{aligned} f_1 = \dot{x}_1 &= x_2, & x_1(0) &= 0, \\ f_2 = \dot{x}_2 &= -x_3 u + 16x_5 - 8, & x_2(0) &= -1, \\ f_3 = \dot{x}_3 &= u, & x_3(0) &= -\sqrt{5}, \\ f_4 = \dot{x}_4 &= x_1^2 + x_2^2 + 0.0005(x_2 + 16x_5 - 8 - 0.1x_3 u^2)^2, & x_4(0) &= 0, \\ f_5 = \dot{x}_5 &= 1, & x_5(0) &= 0, \end{aligned} \right\} \quad (9.2.1)$$

$$0 \leq t \leq 1.$$

Using the Runge-Kutta 4th order method for numerical solution of (9.2.1) we get,

$$\begin{aligned} x_{1,n+1} &= x_{1,n} + \frac{1}{6} \{u_1 + 2u_2 + 2u_3 + u_4\}, \\ x_{2,n+1} &= x_{2,n} + \frac{1}{6} \{v_1 + 2v_2 + 2v_3 + v_4\}, \\ x_{3,n+1} &= x_{3,n} + \frac{1}{6} \{w_1 + 2w_2 + 2w_3 + w_4\}, \\ x_{4,n+1} &= x_{4,n} + \frac{1}{6} \{P_1 + 2P_2 + 2P_3 + P_4\}, \\ x_{5,n+1} &= x_{5,n} + \frac{1}{6} \{Q_1 + 2Q_2 + 2Q_3 + Q_4\}, \end{aligned}$$

where,

$$\begin{aligned} u_1 &= hf_1(x_{1,n}, x_{2,n}, x_{3,n}, x_{4,n}, x_{5,n}) = hx_{2,n}, \\ v_1 &= hf_2(x_{1,n}, x_{2,n}, x_{3,n}, x_{4,n}, x_{5,n}) = h(-x_{3,n}u_n + 16x_{5,n} - 8), \\ w_1 &= hf_3(x_{1,n}, x_{2,n}, x_{3,n}, x_{4,n}, x_{5,n}) = hu_n, \\ P_1 &= hf_4(x_{1,n}, x_{2,n}, x_{3,n}, x_{4,n}, x_{5,n}) = h \left[x_{1,n}^2 + x_{2,n}^2 \right. \\ &\quad \left. + 0.0005(x_{2,n} + 16x_{5,n} - 8 - 0.1x_{3,n}u_n^2)^2 \right], \\ Q_1 &= hf_5(x_{1,n}, x_{2,n}, x_{3,n}, x_{4,n}, x_{5,n}) = h, \end{aligned}$$

$$\begin{aligned}
u_2 &= hf_1(x_{1,n} + \frac{1}{2}u_1, x_{2,n} + \frac{1}{2}v_1, x_{3,n} + \frac{1}{2}w_1, x_{4,n} + \frac{1}{2}P_1, \\
&\quad x_{5,n} + \frac{1}{2}Q_1) = h \left[x_{2,n} + \frac{1}{2}v_1 \right], \\
v_2 &= hf_2(x_{1,n} + \frac{1}{2}u_1, x_{2,n} + \frac{1}{2}v_1, x_{3,n} + \frac{1}{2}w_1, x_{4,n} + \frac{1}{2}P_1, \\
&\quad x_{5,n} + \frac{1}{2}Q_1) = h \left\{ - \left[x_{3,n} + \frac{1}{2}w_1 \right] u_n + 16(x_{5,n} + \frac{1}{2}Q_1) - 8 \right\}, \\
w_2 &= hf_3(x_{1,n} + \frac{1}{2}u_1, x_{2,n} + \frac{1}{2}v_1, x_{3,n} + \frac{1}{2}w_1, x_{4,n} + \frac{1}{2}P_1, \\
&\quad x_{5,n} + \frac{1}{2}Q_1) = hu_n, \\
P_2 &= hf_4(x_{1,n} + \frac{1}{2}u_1, x_{2,n} + \frac{1}{2}v_1, x_{3,n} + \frac{1}{2}w_1, x_{4,n} + \frac{1}{2}P_1, \\
&\quad x_{5,n} + \frac{1}{2}Q_1) = h \left\{ (x_{1,n} + \frac{1}{2}u_1)^2 + (x_{2,n} + \frac{1}{2}v_1)^2 \right. \\
&\quad \left. + 0.0005 \left[(x_{2,n} + \frac{1}{2}v_1) + 16(x_{5,n} + \frac{1}{2}Q_1) - 8 \right. \right. \\
&\quad \left. \left. - 0.1(x_{3,n} + \frac{1}{2}w_1)u_n^2 \right] \right\}, \\
Q_2 &= hf_5(x_{1,n} + \frac{1}{2}u_1, x_{2,n} + \frac{1}{2}v_1, x_{3,n} + \frac{1}{2}w_1, x_{4,n} + \frac{1}{2}P_1, \\
&\quad x_{5,n} + \frac{1}{2}Q_1) = h, \\
u_3 &= hf_1(x_{1,n} + \frac{1}{2}u_2, x_{2,n} + \frac{1}{2}v_2, x_{3,n} + \frac{1}{2}w_2, x_{4,n} + \frac{1}{2}P_2, \\
&\quad x_{5,n} + \frac{1}{2}Q_2) = h(x_{2,n} + \frac{1}{2}v_2), \\
v_3 &= hf_2(x_{1,n} + \frac{1}{2}u_2, x_{2,n} + \frac{1}{2}v_2, x_{3,n} + \frac{1}{2}w_2, x_{4,n} + \frac{1}{2}P_2, \\
&\quad x_{5,n} + \frac{1}{2}Q_2) = h \left\{ - \left[x_{3,n} + \frac{1}{2}w_2 \right] u_n + 16(x_{5,n} + \frac{1}{2}Q_2) - 8 \right\}, \\
w_3 &= hf_3(x_{1,n} + \frac{1}{2}u_2, x_{2,n} + \frac{1}{2}v_2, x_{3,n} + \frac{1}{2}w_2, x_{4,n} + \frac{1}{2}P_2, \\
&\quad x_{5,n} + \frac{1}{2}Q_2) = hu_n, \\
P_3 &= hf_4(x_{1,n} + \frac{1}{2}u_2, x_{2,n} + \frac{1}{2}v_2, x_{3,n} + \frac{1}{2}w_2, x_{4,n} + \frac{1}{2}P_2, \\
&\quad x_{5,n} + \frac{1}{2}Q_2) = h \left\{ (x_{1,n} + \frac{1}{2}u_2)^2 + (x_{2,n} + \frac{1}{2}v_2)^2 \right. \\
&\quad \left. + 0.0005 \left[(x_{2,n} + \frac{1}{2}v_2) + 16(x_{5,n} + \frac{1}{2}Q_2) - 8 \right. \right. \\
&\quad \left. \left. - 0.1(x_{3,n} + \frac{1}{2}w_2)u_n^2 \right] \right\}, \\
Q_3 &= hf_5(x_{1,n} + \frac{1}{2}u_2, x_{2,n} + \frac{1}{2}v_2, x_{3,n} + \frac{1}{2}w_2, x_{4,n} + \frac{1}{2}P_2, \\
&\quad x_{5,n} + \frac{1}{2}Q_2) = h, \\
u_4 &= hf_1(x_{1,n} + u_3, x_{2,n} + v_3, x_{3,n} + w_3, x_{4,n} + P_3, x_{5,n} + Q_3), \\
&= h(x_{2,n} + v_3), \\
v_4 &= hf_2(x_{1,n} + u_3, x_{2,n} + v_3, x_{3,n} + w_3, x_{4,n} + P_3, \\
&\quad x_{5,n} + Q_3) = h \left\{ - [x_{3,n} + w_3] u_n + 16[x_{5,n} - Q_3] - 8 \right\},
\end{aligned}$$

$$\begin{aligned} w_4 &= hf_3(x_{1,n} + u_3, x_{2,n} + v_3, x_{3,n} + w_3, x_{4,n} + P_3, x_{5,n} + Q_3), \\ &= hu_n, \end{aligned}$$

$$\begin{aligned} P_4 &= hf_4(x_{1,n} + u_3, x_{2,n} + v_3, x_{3,n} + w_3, x_{4,n} + P_3, x_{5,n} + Q_3), \\ &= h \{ (x_{1,n} + u_3)^2 + (x_{2,n} + v_3)^2 + 0.0005 [(x_{2,n} + v_3) \\ &\quad + 16(x_{5,n} + Q_3) - 8 - 0.1(x_{3,n} + w_3)u_n^2]^2 \}, \end{aligned}$$

$$\begin{aligned} Q_4 &= hf_5(x_{1,n} + u_3, x_{2,n} + v_3, x_{3,n} + w_3, x_{4,n} + P_3, \\ &\quad x_{5,n} + Q_3) = h. \end{aligned}$$

The adjoint equations are:

$$\left. \begin{aligned} f_1 &= \dot{\lambda}_1 = -[2\lambda_4 x_1], \lambda_1(1) = 0, \\ f_2 &= \dot{\lambda}_2 = -\{\lambda_1 + \lambda_4 \{2x_2 + 0.0005 [2x_2 - 0.2x_3 u^2 + 32x_5 - 16]\}\}, \lambda_2(1) = 0, \\ f_3 &= \dot{\lambda}_3 = -\{-\lambda_2 u + 0.0005\lambda_4 [-3.2x_5 u^2 + 1.6u^2 + 0.02u^4 x_3 - 0.2x_2 u^2]\}, \lambda_3(1) = 0, \\ f_4 &= \dot{\lambda}_4 = 0, \lambda_4(1) = 1, \\ f_5 &= \dot{\lambda}_5 = -\{16\lambda_2 + 0.0005\lambda_4 [-3.2x_3 u^2 + 512x_5 + 32x_2 - 256]\}, \lambda_5(1) = 0. \end{aligned} \right\} \quad (9.2.2)$$

Using the Runge-Kutta 4th order method for numerical solution of (9.2.2) we get,

$$\begin{aligned} \lambda_{1,n} &= \lambda_{1,n+1} + \frac{1}{6} \{u_1 + 2u_2 + 2k_3 + k_4\}, \\ \lambda_{2,n} &= \lambda_{2,n+1} + \frac{1}{6} \{z_1 + 2z_2 + 2z_3 + z_4\}, \\ \lambda_{3,n} &= \lambda_{3,n+1} + \frac{1}{6} \{y_1 + 2y_2 + 2y_3 + y_4\}, \\ \lambda_{4,n} &= \lambda_{4,n+1} + \frac{1}{6} \{a_1 + 2a_2 + 2a_3 + a_4\}, \\ \lambda_{5,n} &= \lambda_{5,n+1} + \frac{1}{6} \{b_1 + 2b_2 + 2b_3 + b_4\}, \end{aligned}$$

where,

$$\begin{aligned} k_1 &= -hf_1(\lambda_{1,n+1}, \lambda_{2,n+1}, \lambda_{3,n+1}, \lambda_{4,n+1}, \lambda_{5,n+1}), \\ &= -h \{-2\lambda_{4,n+1}x_{1,n+1}\} = h \{2\lambda_{4,n+1}x_{1,n+1}\}, \\ z_2 &= -hf_2(\lambda_{1,n+1}, \lambda_{2,n+1}, \lambda_{3,n+1}, \lambda_{4,n+1}, \lambda_{5,n+1}), \\ &= -h \{-\{\lambda_{1,n+1} + \lambda_{4,n+1} \{2x_{2,n+1} + 0.0005 [2x_{2,n+1} \\ &\quad - 0.2x_{3,n+1}u_{n+1}^2 + 32x_{5,n+1} - 16]\}\}\}, \\ y_1 &= -hf_3(\lambda_{1,n+1}, \lambda_{2,n+1}, \lambda_{3,n+1}, \lambda_{4,n+1}, \lambda_{5,n+1}), \\ &= -h \{-\{-\lambda_{2,n+1}u_{n+1} + 0.0005\lambda_{4,n+1} [-3.2x_{5,n+1}u_{n+1}^2 + \\ &\quad 1.6u_{n+1}^2 + 0.02u_{n+1}^4 x_{3,n+1} - 0.2x_{2,n+1}u_{n+1}^2]\}\}\}, \\ &= h \{\{-\lambda_{2,n+1}u_{n+1} + 0.0005\lambda_{4,n+1} [-3.2x_{5,n+1}u_{n+1}^2 + \\ &\quad 1.6u_{n+1}^2 + 0.02u_{n+1}^4 x_{3,n} - 0.2x_{2,n+1}u_{n+1}^2]\}\}\}, \\ a_1 &= -hf_4(\lambda_{1,n+1}, \lambda_{2,n+1}, \lambda_{3,n+1}, \lambda_{4,n+1}, \lambda_{5,n+1}), \\ &= -h(0) = 0, \\ b_1 &= -hf_5(\lambda_{1,n+1}, \lambda_{2,n+1}, \lambda_{3,n+1}, \lambda_{4,n+1}, \lambda_{5,n+1}), \\ &= -h \{-\{16\lambda_{2,n+1} + 0.0005\lambda_{4,n+1} [-3.2x_{3,n+1}u_{n+1}^2 + \\ &\quad 512x_{5,n+1} + 32x_{2,n+1} - 256]\}\}\}, \\ &= h \{\{16\lambda_{2,n+1} + 0.0005\lambda_{4,n+1} [-3.2x_{3,n+1}u_{n+1}^2 + 512x_{5,n+1} \\ &\quad + 32x_{2,n+1} - 256]\}\}\}, \end{aligned}$$

$$\begin{aligned}
k_2 &= -hf_1(\lambda_{1,n+1} + \frac{1}{2}k_1, \lambda_{2,n+1} + \frac{1}{2}z_1, \lambda_{3,n+1} + \frac{1}{2}y_1, \\
&\quad \lambda_{4,n+1} + \frac{1}{2}a_1, \lambda_{5,n+1} + \frac{1}{2}b_1), \\
&= h \left\{ 2(\lambda_{4,n+1} + \frac{1}{2}a_1)x_{1,n+1} \right\}, \\
z_2 &= -hf_2(\lambda_{1,n+1} + \frac{1}{2}k_1, \lambda_{2,n+1} + \frac{1}{2}z_1, \lambda_{3,n+1} + \frac{1}{2}y_1, \\
&\quad \lambda_{4,n+1} + \frac{1}{2}a_1, \lambda_{5,n+1} + \frac{1}{2}b_1) = h \left\{ \left\{ (\lambda_{1,n+1} + \frac{1}{2}k_1) \right. \right. \\
&\quad + (\lambda_{4,n+1} + \frac{1}{2}a_1) \{ 2x_{2,n+1} + 0.0005 [2x_{2,n+1} \\
&\quad - 0.2x_{3,n+1} + u_{n+1}^2 + 32x_{5,n+1} - 16] \} \} \}, \\
y_2 &= -hf_3(\lambda_{1,n+1} + \frac{1}{2}k_1, \lambda_{2,n+1} + \frac{1}{2}z_1, \lambda_{3,n+1} + \frac{1}{2}y_1, \\
&\quad \lambda_{4,n+1} + \frac{1}{2}a_1, \lambda_{5,n+1} + \frac{1}{2}b_1) \\
&= h \left\{ \left\{ -(\lambda_{2,n+1} + \frac{1}{2}z_1)u_{n+1} + (\lambda_{4,n+1} + \frac{1}{2}a_1)0.0005 \right. \right. \\
&\quad \left[-3.2x_{5,n+1}u_{n+1}^2 + 1.6u_{n+1}^2 + 0.02u_{n+1}^4x_{3,n+1} - \right. \\
&\quad \left. \left. 0.2x_{2,n+1}u_{n+1}^2 \right] \right\} \}, \\
a_2 &= -hf_4(\lambda_{1,n+1} + \frac{1}{2}k_1, \lambda_{2,n+1} + \frac{1}{2}z_1, \lambda_{3,n+1} + \frac{1}{2}y_1, \\
&\quad \lambda_{4,n+1} + \frac{1}{2}a_1, \lambda_{5,n+1} + \frac{1}{2}b_1) = -h(0) = 0, \\
b_2 &= -hf_5(\lambda_{1,n+1} + \frac{1}{2}k_1, \lambda_{2,n+1} + \frac{1}{2}z_1, \lambda_{3,n+1} + \frac{1}{2}y_1, \\
&\quad \lambda_{4,n+1} + \frac{1}{2}a_1, \lambda_{5,n+1} + \frac{1}{2}b_1) = h \left\{ \left\{ 16(\lambda_{2,n+1} + \frac{1}{2}z_1) \right. \right. \\
&\quad + (\lambda_{4,n+1} + \frac{1}{2}a_1)0.0005 \left[-3.2x_{3,n+1}u_{n+1}^2 + 512x_{5,n+1} \right. \\
&\quad \left. \left. + 32x_{2,n+1} - 256 \right] \right\} \}, \\
k_3 &= -hf_1(\lambda_{1,n+1} + \frac{1}{2}k_2, \lambda_{2,n+1} + \frac{1}{2}z_2, \lambda_{3,n+1} + \frac{1}{2}y_2, \\
&\quad \lambda_{4,n+1} + \frac{1}{2}a_2, \lambda_{5,n+1} + \frac{1}{2}b_2) = h \left\{ 2(\lambda_{1,n+1} + \frac{1}{2}a_2)x_{1,n+1} \right\}, \\
z_3 &= -hf_2(\lambda_{1,n+1} + \frac{1}{2}k_2, \lambda_{2,n+1} + \frac{1}{2}z_2, \lambda_{3,n+1} + \frac{1}{2}y_2, \\
&\quad \lambda_{4,n+1} + \frac{1}{2}a_2, \lambda_{5,n+1} + \frac{1}{2}b_2), \\
&= h \left\{ \left\{ (\lambda_{1,n+1} + \frac{1}{2}k_2) + (\lambda_{4,n+1} + \frac{1}{2}a_2) \right. \right. \\
&\quad \left\{ 2x_{2,n+1} + 0.0005 [2x_{2,n+1} - 0.2x_{3,n+1}u_{n+1}^2 + 32x_{5,n+1} - 16] \right\} \} \}, \\
y_3 &= -hf_3(\lambda_{1,n+1} + \frac{1}{2}k_2, \lambda_{2,n+1} + \frac{1}{2}z_2, \lambda_{3,n+1} + \frac{1}{2}y_2, \\
&\quad \lambda_{4,n+1} + \frac{1}{2}a_2, \lambda_{5,n+1} + \frac{1}{2}b_2) = h \left\{ \left\{ -(\lambda_{2,n+1} + \frac{1}{2}z_2)u_{n+1} \right. \right. \\
&\quad + (\lambda_{4,n+1} + \frac{1}{2}a_2)0.0005 \left[-3.2x_{5,n+1}u_{n+1}^2 + 1.6u_{n+1}^2 \right. \\
&\quad \left. \left. + 0.02u_{n+1}^4x_{3,n+1} - 0.2x_{2,n+1}u_{n+1}^2 \right] \right\} \},
\end{aligned}$$

$$\begin{aligned}
a_3 &= -hf_4(\lambda_{1,n+1} + \frac{1}{2}k_2, \lambda_{2,n+1} + \frac{1}{2}z_2, \lambda_{3,n+1} + \frac{1}{2}y_2, \\
&\quad \lambda_{4,n+1} + \frac{1}{2}a_2, \lambda_{5,n+1} + \frac{1}{2}b_2) = 0, \\
b_3 &= -hf_5(\lambda_{1,n+1} + \frac{1}{2}k_2, \lambda_{2,n+1} + \frac{1}{2}z_2, \lambda_{3,n+1} + \frac{1}{2}y_2, \\
&\quad \lambda_{4,n+1} + \frac{1}{2}a_2, \lambda_{5,n+1} + \frac{1}{2}b_2) = h \left\{ \left\{ 16(\lambda_{2,n+1} + \frac{1}{2}z_2) \right. \right. \\
&\quad + (\lambda_{4,n+1} + \frac{1}{2}a_2)0.0005 \left[-3.2x_{3,n+1}u_{n+1}^2 + 512x_{5,n+1} \right. \\
&\quad + \left. \left. 32x_{2,n+1} - 256 \right) \right\} \left. \right\}, \\
k_4 &= -hf_1(\lambda_{1,n+1} + k_3, \lambda_{2,n+1} + z_3, \lambda_{3,n+1} + y_3, \lambda_{4,n+1} + a_3, \\
&\quad \lambda_{5,n+1} + b_3) = h \{ 2(\lambda_{4,n+1} + a_1)x_{1,n+1} \}, \\
z_4 &= -hf_2(\lambda_{1,n+1} + k_3, \lambda_{2,n+1} + z_3, \lambda_{3,n+1} + y_3, \\
&\quad \lambda_{4,n+1} + a_3, \lambda_{5,n+1} + b_3) = h \{ \{ (\lambda_{1,n+1} + k_3) \\
&\quad + (\lambda_{4,n+1} + a_3) \{ 2x_{2,n+1} + 0.0005 [2x_{2,n+1} - 0.2x_{3,n+1}u_{n+1}^2 \\
&\quad + 32x_{5,n+1} - 16] \} \} \}, \\
y_4 &= -hf_3(\lambda_{1,n+1} + k_3, \lambda_{2,n+1} + z_3, \lambda_{3,n+1} + y_3, \lambda_{4,n+1} + a_3, \\
&\quad \lambda_{5,n+1} + b_3) = h \{ \{ -(\lambda_{2,n+1} + z_3)u_{n+1} \\
&\quad + (\lambda_{4,n+1} + a_3)0.0005 [-3.2x_{5,n+1}u_{n+1}^2 + 1.6u_{n+1}^2 \\
&\quad + 0.02u_{n+1}^4x_{3,n+1} - 0.2x_{2,n+1}u_{n+1}^2] \} \}, \\
a_4 &= -hf_4(\lambda_{1,n+1} + k_3, \lambda_{2,n+1} + z_3, \lambda_{3,n+1} + y_3, \\
&\quad \lambda_{4,n+1} + a_3, \lambda_{5,n+1} + b_3) = 0, \\
b_4 &= -hf_5(\lambda_{1,n+1} + k_3, \lambda_{2,n+1} + z_3, \lambda_{3,n+1} + y_3, \\
&\quad \lambda_{4,n+1} + a_3, \lambda_{5,n+1} + b_3) = h \{ \{ 16(\lambda_{2,n+1} + z_3) \\
&\quad + (\lambda_{4,n+1} + a_3)0.0005 [-3.2x_{3,n+1}u_{n+1}^2 + 512x_{5,n+1} \\
&\quad + 32x_{2,n+1} - 256] \} \}.
\end{aligned}$$

9.3 Results and Discussion

9.3.1 Gradient method

The algorithm for the gradient in function space applied to problems the same as that described in chapter 4, section 4.4.1. Here, the gradient $\left(g = \frac{\partial H}{\partial u} \right)$ is obtained as follows:

from, (9.1.12),

$$\begin{aligned}
H &= \lambda_1 x_2 + \lambda_2 (-x_3 u + 16x_5 - 8) + \lambda_3 u + \lambda_4 \\
&\quad [x_1^2 + x_2^2 + 0.0005(x_2 + 16x_5 - 8 - 0.1x_3 u^2)^2] + \lambda_5, \\
&= \lambda_1 x_2 + \lambda_2 (-x_3 u + 16x_5 - 8) + \lambda_3 u + \lambda_4 \{ x_1^2 + x_2^2 \\
&\quad + 0.0005 [x_2^2 + (16x_5)^2 + 64 + 10.1x_3 u^2]^2 \\
&\quad - 2(0.1x_3 u^2)x_2 - 2(16x_5)(0.1x_3 u^2) \\
&\quad + 2(8)(0.1x_3 u^2) + 2x_2(16x_5) - 2(8)x_2 \\
&\quad - 2(8)(16x_5) \} + \lambda_5. \\
g &= \frac{\partial H}{\partial u} = -\lambda_2 x_3 + \lambda_3 + 0.0005\lambda_4 [0.04x_3^2 u^3 \\
&\quad - 0.4x_3 x_2 u - 6.4x_5 x_3 u + 3.2x_3 u].
\end{aligned} \tag{9.3.1}$$

The efficiency of the method is examined by varying the critical parameters ϵ , N and u_0 . Here the stopping conditions are taken as $\|g\| < Acc$, where Acc is set to 1.628×10^{-2} , 2.0×10^{-2} and 5.0×10^{-2} . It should be pointed out that selecting $Acc \leq 1.628 \times 10^{-2}$ is because, that even by increasing the number of iterations no improvement can be found for smaller values of Acc .

Table (9.3.1), shows that effect of ϵ and N in obtaining the optimal J with the best Acc possible, and best starting control $u_0 = 4.0$. The best J^* obtained was 0.12000835 to 8 decimal places, with $\|g\| = 1.628 \times 10^{-2}$ and the corresponding parameter values, $\epsilon = 1.9$, $N = 400$ or 800 , and $m = 244$. Selecting N sufficiently large, i.e. 100 or 400 produces consistent results, but with N too small, i.e., 10, the stopping condition for $Acc \leq 1.628 \times 10^{-2}$ was not met, even by increasing the number of iterations, and best Acc obtained was $Acc \leq 10^{-1}$ for ϵ in the range, $1.0 \leq \epsilon \leq 1.9$. Here selecting N as 800 as opposed to 400 may result in slightly better minimum J^* for some ϵ 's, but at the cost of more computing time.

Tables (9.3.2), (9.3.3) and (9.3.4), show the effect of u_0 on J when the best N is selected, i.e. $n = 400$ for $Acc \leq 5.0 \times 10^{-2}$, $\leq 2.0 \times 10^{-2}$ and $\leq 1.628 \times 10^{-2}$ respectively.

From Table (9.3.2), it can be seen that selecting u_0 in the range, $2.0 \leq u_0 \leq 4.0$ achieves the required $Acc \leq 5.0 \times 10^{-2}$, in fewer iterations than $u_0 = 1.0$ or $u_0 = 5.0$. Selecting $u_0 = 4.0$, produces the best overall result. Also, from Table (9.3.3) it can be seen that selecting u_0 in the range $2.0 \leq u_0 \leq 4.0$ achieves the required $Acc \leq 2.0 \times 10^{-2}$ in fewer iterations than $u_0 = 1.0$ or $u_0 = 5.0$ and the best overall result is obtained with $u_0 = 4.0$.

From Table (9.3.4), it can be seen that, selecting u_0 in the range, $2.0 \leq u_0 \leq 4.0$, achieves the required, $Acc \leq 1.628 \times 10^{-2}$, where $u_0 = 1.0$ and $u_0 = 5.0$, do not meet the requirement, for the $Acc \leq 1.628 \times 10^{-2}$. But selecting $u_0 = 4$, again gives the best result.

Various aspects of the effect of the critical parameters, are shown graphically in figures (9.3.1) to (9.3.4).

Figure (9.3.1), effect of m on u with $u_0 = 4.0$, $N = 400$ and $\epsilon = 1.9$, where $m = 1.0, 100$ and 244 . As can be seen from the plots, all the curves of control start at 10, for the 3 different iterations, but then each curve behaves differently along time axis.

Figure (9.3.2), effect of N on u , with $u_0 = 4.0$, $m = 210$, $N = 100$, with $\epsilon = 1.0$ and $N = 400$ with $\epsilon = 1.9$. As can be seen there are some differences in the behaviour of the curves of control between $N = 100$, and $N = 400$, e.g. at approximate time of 0.1 to 0.25, but then from there, they behaved similarly for a while, till the approximate time of 0.32, that from there to the final time of 1, they behaved differently.

Figure (9.3.3), effect of ϵ on u , with $u_0 = 4.0$, $m = 244$, $N = 400$ and $\epsilon = 1.0$ and 1.9 . Here from the graph we can see there are some differences between the curves of $\epsilon = 1.0$ and 1.9 . Although at start, both curves behaved similarly for a while, but from approximate time of 0.16 onwards, the difference is quite eminent from the plots.

Figure (9.3.4), effect of J^* against m , with $N = 400$, $u_0 = 1.0$ with $\epsilon = 0.9$, $u_0 = 3.0$ with $\epsilon = 1.9$ and $u_0 = 4.0$ with $\epsilon = 1.9$. As can be seen from the graph, as m increases the value of optimal J for all starting controls converges to the same value.

The correct choice of initial control in this case $u_0 = 4.0$, can give a better J^* with a more accurate Acc . Also for higher Acc , e.g., 1.628×10^{-2} , selecting distant u_0 's will never achieve that. The interaction between ϵ and N , shows that the best combination to give a better optimal J , in fewer iterations, less computing time and better Acc , is given by selecting sufficiently large N , i.e., 400 and ϵ in the range $1.8 \leq \epsilon \leq 2.0$.

9.3.2 Steepest descent

The algorithm for steepest descent applied to the problem 6, is as described in chapter 2, section 2.2.2. The line search technique used for this method is linear search at constant step, which was described in chapter 3, section 3.4.1. The g is as GFS in this chapter, section 9.3.1.

Here again we investigate the effect of step length factor, integration step and initial control on the solution. Stopping condition are taken as $\|g\| \leq 1.6263 \times 10^{-2}$, 2.0×10^{-2} and 5.0×10^{-2} .

Table (9.3.5) shows the effect of ϵ and N , in obtaining the optimal J , with the best Acc possible, and best u_0 in this case $u_0 = 4.0$. The best J^* obtained was 0.11993895 to 8 decimal places with $\|g\| = 1.621 \times 10^{-2}$ and the corresponding parameter values, $\epsilon = 2.0$, $N = 400$ or 800 and $m = 213$. For N large enough say, 400 selecting ϵ in the range $1.9 \leq \epsilon \leq 2.0$ can produce better values for optimal J than selecting ϵ in the range $1.0 \leq \epsilon \leq 1.5$ or $\epsilon = 2.1$, although it might take more number of iterations to achieve those.

For N smaller, say 100, selecting ϵ in the range $1.9 \leq \epsilon \leq 2.1$ can achieve their best $Acc \leq 1.628 \times 10^{-2}$ in fewer iterations than selecting ϵ in the range $1.0 \leq \epsilon \leq 1.5$. For N too small, say, 10, the best Acc obtained was $Acc \leq 10^{-1}$, which is not accurate enough, and even an increase in m made no improvement. Here also selecting N as 800 as opposed to 400 may result in slightly better minimum J^* for some ϵ 's but at the cost of more computing time.

Tables (9.3.6), (9.3.7) and (9.3.8), show that effect of initial control on J^* , when the best N is selected, i.e. $N = 400$, for $Acc \leq 5.0 \times 10^{-2}$, $\leq 2.0 \times 10^{-2}$ and $\leq 1.6263 \times 10^{-2}$.

From Table (9.3.6), it can be seen that selecting u_0 in the range $1.0 \leq u_0 \leq 2.0$, achieves $Acc \leq 5.0 \times 10^{-2}$ in fewer iterations than u_0 , in the range $3.0 \leq u_0 \leq 5.0$. From Table (9.3.7), it can be seen that selecting u_0 , in the range $2.0 \leq u_0 \leq 4.0$, achieves, the required $Acc \leq 2.0 \times 10^{-2}$, in fewer iterations than $u_0 = 1.0$ or $u_0 = 5.0$. But the best J^* was achieved with $u_0 = 2.0$.

From Table (9.3.8), it can be seen that selecting u_0 in the range $2.0 \leq u_0 \leq 4.0$ achieves the required $Acc \leq 1.6263 \times 10^{-2}$, in fewer iterations, than $u_0 = 1.0$. Also at $u_0 = 5$, the required Acc could never be achieved.

Various aspects of the effect of m , N , ϵ and u_0 are shown graphically in Figures (9.3.5) to (9.3.8).

Figure (9.3.5), effect of m on u with $u_0 = 4.0$, $N = 400$ and $\epsilon = 2.0$, where $m = 10$, $m = 100$ and $m = 213$.

As can be seen from the plots the behaviour of the curves of control are different from each other, along time axis, but as m increase the pattern of behaviour gets closer.

Figure (9.3.6), effect of N on u , with $u_0 = 4.0$, $N = 100$, with $\epsilon = 1.5$, $m = 344$ and $N = 400$ with $\epsilon = 2.0$ and $m = 213$. Difference can be seen from the plots of control, between $N = 100$ and 400 . Although they both started at $u = 10$, but soon at approximate time of 0.09 , the pattern of behaviour started to differ and it continued till the final time.

Figure (9.3.7), effect of ϵ on u , with $u_0 = 4.0$, $m = 200$, $N = 400$ and $\epsilon = 1.0$ and 2.0 . Here again some differences can be seen from the graphs of controls between $\epsilon = 1.0$ and $\epsilon = 2.0$. Here, although the pattern of behaviour are similar, and at time 0.0 , they both started at $u = 10.0$ and behaved similarly for a while up to approximate time of 0.15 , but from there, some differences can be observed, the the behaviour of the curves, that continued up to the approximate time of 0.81 , and then again from there to the final time, they behaved similarly. Figure (9.3.8), effect of J^* against m , with $N = 400$, $u_0 = 1.0$, with $\epsilon = 0.9$, $u_0 = 4.0$ with $\epsilon = 2.0$ and $u_0 = 5.0$ with $\epsilon = 0.9$. As can be seen, by increasing m , the value of optimal J , for all starting controls, converges to the same one.

The above results show that a proper choice of initial control is an important factor in achieving the best J^* in fewer iterations, function evaluations, and finally computing time. Also selecting a distant u_0 may never achieve a high Acc .

The interaction between ϵ and N , shows that the best combination in order to produce a better optimal J , in fewer iterations, less computing time and better Acc , exists with selecting sufficiently large enough N , i.e. 400 with ϵ in the range $1.9 \leq \epsilon \leq 2.0$.

9.3.3 Fletcher-Reeves

The algorithm for the Fletcher-Reeves method, applied to problem 5, is as described in chapter 2, section 2.4. The calculation of the norms of the gradients are as given in chapter 2, section 2.1.1.

The line search technique is the same as that for steepest descent in this chapter, section 9.3.2. The gradient (g) is the same as the one obtained in this chapter, section 9.3.1.

Table (9.3.9), shows the effect of ϵ and N , in obtaining the optimal J , with best Acc possible and best u_0 , in this case $u_0 = 3.0$.

The best J^* obtained was 0.12009654 to 8 decimal places, with $\|g\| = 1.589 \times 10^{-2}$ and the corresponding parameter values $\epsilon = 1.4$, $N = 400$ or 800 and $m = 67$.

For N , large say 400 , selecting ϵ in the range $1.0 \leq \epsilon \leq 1.4$ can meet the requirement for the $Acc \leq 1.60 \times 10^{-2}$, in fewer iterations, also fewer function evaluations and finally less computing time than selecting ϵ and $\epsilon = 0.5$ or $\epsilon = 1.5$. For N large enough say 1.0 , selecting ϵ in the range $1.3 \leq \epsilon \leq 1.4$ can meet the requirement for the $Acc \leq 1.6 \times 10^{-2}$ in fewer iterations and function evaluations, also less computing time than selecting ϵ in the range $0.9 \leq \epsilon \leq 1.0$ and $\epsilon = 1.5$.

For N too small, say 10 , none of the ϵ 's can achieve an accurate Acc and even increasing the number of iterations would not improve the value of J . As for previous methods selecting N as 800 as opposed to 400 may result in

slightly better minimum J^* for some ϵ 's but at the cost of more computing time.

Tables (9.3.10), (9.3.11) and (9.3.12), show the effect of initial control on J , when the best N is selected, i.e., 400 for $Acc \leq 5.0 \times 10^{-2}$, $\leq 2.0 \times 10^{-2}$ and $\leq 1.60 \times 10^{-2}$.

From Table (9.3.10) it can be seen that, selection of u_0 , in the range $1.0 \leq u_0 \leq 3.0$, achieves $Acc \leq 5.0 \times 10^{-2}$, in fewer iterations than u_0 in the range $4.0 \leq u_0 \leq 5.0$. From Table (9.3.11), it can be seen that selecting u_0 as $u_0 = 1.0$ or $u_0 = 3.0$ can achieve the required $Acc \leq 2.0 \times 10^{-2}$, in fewer iterations, than selecting $u_0 = 2.0$ or u_0 in the range $4.0 \leq u_0 \leq 5.0$.

From Table (9.3.12), it can be seen that selecting u_0 , in the range $3.0 \leq u_0 \leq 4.0$, achieves the required $Acc \leq 1.6 \times 10^{-2}$, in fewer iterations than selecting $u_0 = 2.0$ or u_0 in the range $4.0 \leq u_0 \leq 5.0$. The best optimal J was achieved with $u_0 = 3.0$.

Various aspects of effect of m, N, ϵ and u_0 are shown graphically, in Figures (9.3.9) to (9.3.12).

Figure (9.3.9), effect of m on u , with $u_0 = 3.0, N = 400$ and $\epsilon = 1.4$, where $m = 1, 10$ and 67 . As can be seen from the plots, there are some differences between the curves of control for various m 's and as m increases to 67 we can see the behaviour of the curve is completely different from $m = 1$ or $m = 10$.

Figure (9.3.10), effect of N on u , with $u_0 = 3.0, m = 31, N = 100$ with $\epsilon = 1.0$ and $N = 400$ with $\epsilon = 1.4$. Here, although the curves of control for $N = 100$ and $N = 400$, both started at 10 , and behaved similarly for a little while, but from approximate time of 0.08 the difference is quite eminent and it continued to the final time.

Figure (9.3.11), effect of ϵ on u with $u = 3.0, m = 67, N = 400$ and $\epsilon = 0.5$ and $\epsilon = 1.4$. Here, we can see some similarities in the pattern of behaviour between the curves of control for $\epsilon = 0.5$ and 1.4 , but there are differences in the curvature between the two. Figure (9.3.12) effect of J^* against m , with $N = 400, u_0 = 1.0$ with $\epsilon = 1.4, u_0 = 3.0$ with $\epsilon = 1.4$ and $u_0 = 5.0$ with $\epsilon = 0.9$. As can be seen from the plots by increasing m the value of optimal J for all starting controls converges to the same value. In view of the results obtained a proper choice of initial control is an important factor in achieving the best J^* , in fewer iterations in this case $u_0 = 3.0$.

Also the interaction between ϵ and N shows that the best combination in order to produce a better optimal J in fewer iterations less computing time and better Acc , exists with selecting sufficiently large N , i.e., 400 with ϵ in the range $1.0 \leq \epsilon \leq 1.4$.

9.3.4 Polak-Ribière

The algorithm for Polak-Ribière method is described in chapter 2, section 2.5. The line search technique and calculation of the norms are as described for FR in this chapter, section 9.3.1. The gradient (g) is the same as GFS in this chapter, section 9.3.1. Table (9.3.13) shows the effect of ϵ and N in obtaining the optimal J with the best Acc possible and best initial control in this case $u_0 = 4.0$. The best J^* obtained was 0.11999489 to 8 decimal

places, with $\|g\| = 1.624 \times 10^{-2}$ and the corresponding parameter values $\epsilon = 0.11$, $N = 400$ or 800 and $m = 144$.

For N large, e.g. 400 selecting ϵ in the range $0.09 \leq \epsilon \leq 0.12$ can meet the requirements for $Acc \leq 1.63 \times 10^{-2}$, but, selecting $\epsilon = 0.2$, can not meet this requirement, even when m is increased. The best optimal J for $N = 400$, considering the Acc and number of iterations taken, obtained with ϵ in the range $0.11 \leq \epsilon \leq 0.12$.

For smaller N , say 100 , selecting ϵ in the range $0.1 \leq \epsilon \leq 0.11$ can achieve optimal J with a better Acc than $\epsilon = 0.09$ or $\epsilon = 0.2$, since with $\epsilon = 0.09$ or 0.2 even with increasing m no more improvement in Acc could be obtained.

For N too small e.g. 10 , none of the ϵ 's can meet an accurate Acc , and even increasing the number of iterations could not improve the value of J . Here also selecting N as 800 as opposed to 400 may result in slightly better minimum J^* for some ϵ 's but at the cost of more computing time.

Tables (9.3.14), (9.3.15) and (9.3.16) show that effect of selecting u_0 on J^* when the best N is selected, i.e. $N = 400$ for $Acc \leq 5.0 \times 10^{-2}$, $\leq 2.0 \times 10^{-2}$ and $\leq 1.63 \times 10^{-2}$.

From Table (9.3.14) it can be seen that selecting u_0 in the range $3.0 \leq u_0 \leq 5.0$ can achieve the required $Acc \leq 5.0 \times 10^{-2}$, in fewer iterations and also function evaluations than u_0 in the range $1.0 \leq u_0 \leq 2.0$. From Table (9.3.15) it can be seen that selecting u_0 in the range $3.0 \leq u_0 \leq 5.0$ can achieve the required $Acc \leq 2.0 \times 10^{-2}$ in fewer iterations also NFE and Cputime than u_0 in the range $1.0 \leq u_0 \leq 2.0$.

From Table (9.3.16), it can be seen that selecting u_0 in the range $4.0 \leq u_0 \leq 5.0$ can achieve the required $Acc \leq 1.63 \times 10^{-2}$ in fewer iterations and also NFE than u_0 in the range $1.0 \leq u_0 \leq 3.0$.

Various aspects of effect of critical parameters, are shown graphically, in Figures (9.3.13) to (9.3.16).

Figure (9.3.13), effect of m on u with $u_0 = 4.0$, $N = 400$ and $\epsilon = 0.11$, where $m = 1, 20$ and 144 . Here as can be seen from the graphs of control there are quite considerable differences, between various m 's and as m increases the behaviour of the curves of control along time axis changes more.

Figure (9.3.14), effect of N on u with $u_0 = 4.0$, $m = 80$, $N = 100$ with $\epsilon = 0.10$ and $N = 400$ with $\epsilon = 0.11$. As can be seen from the graphs, although the curves of control for both $N = 100$ and 400 , started similarly but there are some minor differences in their behaviours along time axis that can be observed.

Figure (9.3.15) effect of ϵ on u , with $u_0 = 4.0$, $m = 100$, $N = 400$ and $\epsilon = 0.11$ and $\epsilon = 0.2$. Here also, the curves of control for both $\epsilon = 0.11$ and 0.2 , started similarly but as time increases we can see there are differences in their curvature behaviour.

Figure (9.3.16) effect of J^* against m , with $N = 400$, $u_0 = 1.0$ with $\epsilon = 0.11$, $u_0 = 4.0$ with $\epsilon = 0.11$ and $u_0 = 5.0$, with $\epsilon = 0.11$. Here as can be seen as m increases, the value of optimal J for all starting control, converges to the same value.

In view of all above a proper choice of initial control is an important factor in achieving the best J^* in fewer iterations in this case $u_0 = 4.0$.

Also, the interaction between ϵ and N shows that the best combination in order to produce a better optimal J in fewer iterations less computing time

and better Acc , exists with selecting sufficiently large N , i.e. 400 with ϵ in the range $0.11 \leq \epsilon \leq 0.12$.

9.3.5 Hybrid 1

The algorithm for the Hybrid 1 method is described in chapter 2, section 2.6.1. The line search technique and calculation of the norms are the same as for FR in this chapter, section 9.3.5. The gradient (g) is also obtained in the same way as for GFS in section 9.3.1.

Table (9.3.17), shows the effect of ϵ and N in obtaining the optimal J with the best Acc possible and best initial control in this case $u_0 = 3.0$.

The best J^* obtained was 0.11979908 to 8 decimal places with $\|g\| = 1.595 \times 10^{-2}$ and the corresponding parameter values $\epsilon = 1.5, N = 400$ or 800 and $m = 120$.

For N , large say 400 selecting ϵ in the range $1.0 \leq \epsilon \leq 1.5$ can produce fairly better J than selecting ϵ in the range $1.6 \leq \epsilon \leq 2.0$.

For smaller N say 100 selecting ϵ in the range $1.4 \leq \epsilon \leq 1.5$ can obtain optimal J for the $Acc \leq 1.60 \times 10^{-2}$ in fewer iterations than $1.6 \leq \epsilon \leq 2.0$ or $\epsilon = 1.0$. But it does not necessarily achieve a better minimum J^* .

For N too small e.g. 10 with most ϵ 's the optimal J converges with a poor $Acc \leq 10^{-1}$ and even by increasing m the Acc or convergency of J does not improve. Selecting N as 800 as opposed to 400 may result in slightly better minimum J^* for some ϵ 's but at the cost of more computing time.

Tables (9.3.18), (9.3.19) and (9.3.20) show that effect of selecting u_0 on J when the best N is selected, i.e. $N = 400$ for $Acc \leq 5 \times 10^{-2}, \leq 2 \times 10^{-2}$ and $\leq 1.6 \times 10^{-2}$. From Table (9.3.18) it can be seen that selecting $u_0 = 2.0$ of $u_0 = 4.0$ can achieve the required $Acc \leq 5.0 \times 10^{-2}$ in fewer iterations than other starting control $u_0 = 1.0$ or 4.0. Here although selecting $u_0 = 5.0$ can achieve $Acc \leq 5.0 \times 10^{-2}$ for the same iterations as $u_0 = 2.0$ or 4.0 but the J obtained is not as good as the one obtained for the other two starting controls.

From Table (9.3.19) it can be seen that selecting u_0 in the range $3.0 \leq u_0 \leq 5.0$ can achieve the required $Acc \leq 2.0 \times 10^{-2}$ in fewer iterations than selecting u_0 in the range $1.0 \leq u_0 \leq 2.0$.

From Table (9.3.20) selecting $u_0 = 3.0$ can achieve the required $Acc \leq 1.60 \times 10^{-2}$ in fewer iterations than other starting controls in terms of achieving better optimal J and although $u_0 = 5.0$ can achieve $Acc \leq 1.60 \times 10^{-2}$ in fewer iterations than $u_0 = 3.0$, the J obtained is not as good as the one obtained with $u_0 = 3.0$.

Various aspects of critical parameters are shown graphically, in Figures (9.3.17) to (9.3.20).

Figure (9.3.17) effect of m on u with $u_0 = 3.0, N = 400$ and $\epsilon = 1.5$ where $m = 1, 20$ and 120. Here as can be seen from the plots of controls for different m 's as m increases the behaviour of the curves get more complex along the time axis.

Figure (9.3.18) effect of N on u with $u_0 = 3.0, m = 39, N = 100$ with $\epsilon = 1.4$ and $N = 400$ with $\epsilon = 1.5$. Here as can be seen from the curves of control although at start both curve of $N = 100$ and $N = 400$ behaved

similarly but at approximate time of 0.08 some minor differences started to emerge and it continued along the time axis to the end of time 1.0.

Figure (9.3.19) effect of ϵ on u with $u_0 = 3.0, m = 100, N = 400$ and $\epsilon = 1.5$ and $\epsilon = 2.0$. Here also we can see some differences between the curve of control for $\epsilon = 1.5$ and 2.0 . The curves started similarly up to approximate time of 0.09 and from there onwards to the final time they behaved differently.

Figure (9.3.20) effect of J^* against m with $N = 400, u_0 = 1.0$ with $\epsilon = 1.5, u_0 = 5.0$ with $\epsilon = 1.5$ and $u_0 = 5.0$ with $\epsilon = 1.2$. As can be seen from the plots as m increases the value of optimal J for all starting controls converges to the similar value.

In view of the above results an appropriate choice of initial control is an important factor in achieving best J^* in fewer iterations in this case $u_0 = 3.0$.

The interaction between ϵ and N shows that the best combination in order to produce a better optimal J in fewer iterations less computing time and better Acc , exists with sufficiently large N , i.e. 400 with ϵ in the range $1.0 \leq \epsilon \leq 1.5$.

9.3.6 Angle test hybrid

The algorithm for the angle test hybrid method is described in chapter 2, section 2.7. The line search technique and the calculation of the norms are the same as FR in this chapter, section 9.3.3. The gradient (g) is obtained in the same way as GFS in this chapter, section 9.3.1. For this method we had to consider the parameter $\tau > 0$ as well. The method was tested in the same way as the previous ones with the new parameter τ taken as 0.01, 0.0001 and 0.000001.

Table (9.3.21) shows the effect of ϵ and N in obtaining the optimal J with the best Acc possible and best initial control in this case $u_0 = 3.0, \tau = 0.000001$.

The best J^* obtained was 0.11984135 to 8 decimal places, with $\|g\| = 1.587 \times 10^{-2}$ and the corresponding parameter values $\epsilon = 1.3, N = 400$ or 800 and $m = 119$.

For N large say 400 selecting ϵ in the range $1.2 \leq \epsilon \leq 1.3$ can produce a better J with $Acc \leq 1.60 \times 10^{-2}$ in fewer iterations than other ϵ 's.

For smaller N say 100 selecting ϵ as 2.0 can obtain J for $Acc \leq 1.60 \times 10^{-2}$ in fewer iterations than other ϵ 's. For N too small say 10 with most ϵ 's the optimal J , converges with a poor $Acc \leq 10^{-1}$ and even by increasing the number of iterations the Acc of convergency of J does not improve. Here selecting N as 800 as opposed to 400 may result in slightly better minimum J^* for some ϵ 's but at the cost of more computing time.

Tables (9.3.22), (9.3.23) and (9.3.24) show the effect of selecting u_0 on J when the best N is selected, i.e. $N = 400$ and $\tau = 0.000001$, for $Acc \leq 5 \times 10^{-2}, \leq 2.0 \times 10^{-2}$ and $\leq 1.60 \times 10^{-2}$.

From Table (9.3.22) it can be seen that selection of $u_0 = 4.0$ can achieve the required $Acc \leq 5.0 \times 10^{-2}$, in fewer iterations than other u_0 's.

From Table (9.3.23) we can see that selecting $u_0 = 3.0$, can get the $Acc \leq 2.0 \times 10^{-2}$ in fewer iterations than other starting controls.

From Table (9.3.24) selecting u_0 in the range $2.0 \leq u_0 \leq 3.0$ can satisfy the $Acc \leq 1.60 \times 10^{-2}$ in fewer iterations than other starting controls.

Various aspects of the critical parameters are shown graphically in Figures (9.3.21) to (9.3.24) where τ is taken as 0.000001.

Figure (9.3.21) effect of m on u with $u_0 = 3.0$, $N = 400$ and $\epsilon = 1.3$ where $m = 1, 20$ and 119 . Here we can see that by increasing m the curvature of the curves of control differ from each other along the time axis.

Figure (9.3.22) effect of N on u with $u_0 = 3.0$, $m = 46$, $N = 100$ with $\epsilon = 1.4$ and $N = 400$ with $\epsilon = 1.3$. Here the curves of control start similarly but as time increases we can see some minor differences in the behaviour between $N = 100$ and $N = 400$.

Figure (9.3.23) effect of ϵ on u , with $u_0 = 3.0$, $m = 100$, $N = 400$ and $\epsilon = 1.3$ and $\epsilon = 2.0$. As can be seen from the graphs of $\epsilon = 1.3$ and 2.0 , the start of the curves of control are similar but at approximate time of 0.09, the behaviour of the curve starts to differ from each other along the time axis.

Figure (9.3.24) effect of J^* against m with $N = 400$, $u_0 = 1.0$, with $\epsilon = 1.3$, $u_0 = 3.0$ with $\epsilon = 1.3$ and $u_0 = 5.0$ with $\epsilon = 1.0$. Here the value of optimal J for all starting controls converges to the same one, as m increases.

In view of the results obtained a proper choice of initial control is an important factor in achieving best J^* in fewer iterations in this case $u_0 = 3.0$.

The interaction between ϵ and N show that the best combination in order to produce a better optimal J , in fewer iterations less computing time and better Acc exists with sufficiently large N i.e. 400 with ϵ in the range $1.2 \leq \epsilon \leq 1.3$. Here the effect of τ on minimizing J was practically negligible.

9.3.7 Hybrid 3

The algorithm for Hybrid 3, can also be found in chapter 2, section 2.8. The calculation of the norms and line search are the same as FR in this chapter section 9.3.3. The gradient (g) is obtained in the same way as for GFS in this chapter, section 9.3.1.

The effect of the new parameters, $\lambda > 0$ and $\mu < \frac{1}{2}$, had to be considered for this method so test carried out as before with the addition of λ taken as 0.01, 0.0001 and 0.0000001 and μ as 0.15, 0.35 and 0.49999.

Table (9.3.25) shows the effect of ϵ and N in obtaining the optimal J with the best Acc possible and best starting control in this case $u_0 = 3.0$.

Values of λ and μ were taken as 0.0000001 and 0.49999 respectively.

The best J^* obtained was 0.11981238 to 8 decimal places with $\|g\| = 1.585 \times 10^{-2}$ and the corresponding parameter values $\epsilon = 1.4$, $N = 400$ or 800 and $m = 113$.

For N large say 400 selecting ϵ in the range $1.3 \leq \epsilon \leq 1.5$ can produce J^* with $Acc \leq 1.60 \times 10^{-2}$ in fewer iterations than selecting ϵ as 1.0 or 2.0. For smaller N say 100 selecting ϵ in the range $1.4 \leq \epsilon \leq 2.0$ can achieve the $Acc \leq 1.60 \times 10^{-2}$, in fewer iterations than ϵ in the range $1.0 \leq \epsilon \leq 1.3$. For N too small say 10, with most ϵ 's J is found with a poor $Acc \leq 10^{-1}$ and even by increasing the number of iterations the Acc or convergency of J does not improve. Here also selecting N as 800 as opposed to 400 may result in slightly better minimum J^* for some ϵ 's but at the cost of more computing time.

Tables (9.3.26) to (9.3.28) show that effect of selecting u_0 on J when the

best N is selected i.e. $N = 400$ with $\lambda = 0.0000001$ and $\mu = 0.49999$ for $Acc \leq 5.0 \times 10^{-2}$, $\leq 2.0 \times 10^{-2}$ and $\leq 1.60 \times 10^{-2}$.

From Table (9.3.26) it can be seen that selecting u_0 in the range $2.0 \leq u_0 \leq 3.0$ can achieve the required $Acc \leq 5.0 \times 10^{-2}$ in fewer iterations than other u_0 's.

From Table (9.3.27) we can see that selecting $u_0 = 1.0$ or u_0 in the range $3.0 \leq u_0 \leq 4.0$ can achieve the required $Acc \leq 2.0 \times 10^{-2}$ in fewer iterations than $u_0 = 2.0$ or $u_0 = 5.0$. but the best was achieved with $u_0 = 3.0$.

From Table (9.3.28) selecting u_0 in the range $3.0 \leq u_0 \leq 5.0$ can achieve the $Acc \leq 1.60 \times 10^{-2}$, in fewer iterations than $1.0 \leq u_0 \leq 2.0$.

Various aspects of the critical parameters were shown graphically in Figures (9.3.25) to (9.3.28) where $\lambda = 0.0000001$ and $\mu = 0.49999$.

Figure (9.3.25) effect of m on u with $u_0 = 3.0$, $N = 400$ and $\epsilon = 1.4$ where $m = 1, 20$ and 113 . We can see that difference in the behaviour of the curves of control as m increases along the time axis, and this difference is more eminent between $m = 1$ and $m = 113$.

Figure (9.3.26) effect of N on u with $u_0 = 3.0$, $m = 39$, $N = 100$ with $\epsilon = 1.4$ and $N = 100$ also with $\epsilon = 1.4$. As can be seen from the graphs of control there is not much difference in the behaviour of the curves of $N = 100$ with $N = 400$ except in the approximate time intervals of $[0.08, 0.15]$ and $[0.32, 0.58]$.

Figure (9.3.27) effect of ϵ on u with $u_0 = 3.0$, $m = 100$, $N = 400$ and $\epsilon = 1.4$ and $\epsilon = 2.0$.

Figure (9.3.28) effect of J^* against m with $N = 400$, $u_0 = 1.0$ with $\epsilon = 1.0$. As can be seen from the graphs as m increases the value of optimal J from all starting controls converges to the same value.

In view of the results obtained a proper choice of initial control is an important factor in achieving the J^* in fewer iterations in this case $u_0 = 3.0$.

The interaction between ϵ and N shows that the best combination in order to produce a better optimal J in fewer iterations less computing time and better Acc exists with sufficiently large N , i.e. $N = 400$ with ϵ in the range $1.3 \leq \epsilon \leq 1.5$. Here the effect of λ and μ on minimizing J^* was practically negligible.

9.4 Summary of the Results

A summary of the results can be found in Table (9.4.1) and also a comparison of the methods can be seen in Figure (9.4.1) when N is taken as 400 for ATH, $\tau = 0.000001$ and for H3, $\lambda = 0.0000001$ and $\mu = 0.49999$. Also in Figure (9.4.1) the u_0 is taken as 4.0 for GFS, SD and PR and u_0 is taken as 3.0 for FR, H1, ATH and H3.

Considering all the aspects of convergency for minimum J , i.e. the number of iterations Acc and numerical stability the methods performed as follows;

At $u_0 = 1.0$ the best J^* obtained was by H1 but in terms of m , Cputime and NFE, FR performed the best. But overall taking all the parameters Acc , m , Cputime and NFE into consideration, H3 performed the best. At $u_0 = 2.0$ the best J^* was found by H1, but considering m , Cputime and

NFE, FR performed the best. But overall taking the Acc , m , Cputime and NFE into consideration H3 performed the best of all.

At $u_0 = 3.0$ the best J^* obtained was by H1, but considering m , Cputime and NFE, FR performed the best. But overall taking the Acc , m , Cputime and NFE into consideration, H3 performed the best of all.

At $u_0 = 4.0$ the best J^* was found by H3 but considering m , Cputime and NFE, FR performed the best. But overall taking the Acc , m , Cputime and NFE into consideration again H3 performed the best of all.

At $u_0 = 5.0$ the best J^* was found by H3, but considering m , Cputime, NFE and also an accurate Acc , H1 performed the best. But overall taking the Acc , m , Cputime and NFE into consideration again H3 performed the best of all.

9.5 Conclusion

In this chapter we have tested the problem numerically, investigated the efficiency of seven gradient and conjugate gradient techniques with respect to the critical parameters, ϵ , N and u_0 .

The performance of the methods are compared based on the following factors:

- i. Best minimum J^* achieved.
- ii. Level of Acc .
- iii. Number of iterations.
- iv. Number of function evaluations.
- v. Computing time.

As can be seen from the results and also from the summary table (9.4.1) the GFS, SD and PR methods achieve their best J^* with $u_0 = 4.0$ and the FR, H1, ATH and H3 methods achieved their best by selecting $u_0 = 3.0$.

Also it can be seen from Table (9.4.1) that overall, taking the 5 mentioned important factors into account, H3 performed relatively better with various starting controls than other techniques.

In all cases selecting a good initial control, facilitates finding the optimal J in fewer iterations with better Acc and less computing time. Also always selecting large enough integration step number (N) with appropriate step length factor (ϵ) can help to achieve a better value for the optimal J .

TABLE (9.3.1):Results of GFS with varying ϵ and N.

N ϵ	10	100	400	800
1.0	$J^*=0.15864126$ $m=13$ $\ g\ =9.941 \times 10^{-2}$ Cputime<1 NFE=14	$J^*=0.12150251$ $m=210$ $\ g\ =1.628 \times 10^{-2}$ Cputime=1 NFE=211	$J^*=0.12000934$ $m=472$ $\ g\ =1.628 \times 10^{-2}$ Cputime=14 NFE=473	$J^*=0.12000932$ $m=472$ $\ g\ =1.628 \times 10^{-2}$ Cputime=28 NFE=473
1.5	$J^*=0.15889294$ $m=9$ $\ g\ =9.954 \times 10^{-2}$ Cputime<1 NFE=10	$J^*=0.12151104$ $m=140$ $\ g\ =1.628 \times 10^{-2}$ Cputime=1 NFE=141	$J^*=0.12000860$ $m=312$ $\ g\ =1.628 \times 10^{-2}$ Cputime=9 NFE=313	$J^*=0.12000860$ $m=311$ $\ g\ =1.628 \times 10^{-2}$ Cputime=17 NFE=312
1.8	$J^*=0.15748745$ $m=11$ $\ g\ =9.932 \times 10^{-2}$ Cputime<1 NFE=12	$J^*=0.12151168$ $m=110$ $\ g\ =1.628 \times 10^{-2}$ Cputime=1 NFE=111	$J^*=0.12000857$ $m=258$ $\ g\ =1.628 \times 10^{-7}$ Cputime=8 NFE=259	$J^*=0.12000857$ $m=258$ $\ g\ =1.628 \times 10^{-7}$ Cputime=15 NFE=259
1.9	$J^*=0.15854882$ $m=11$ $\ g\ =9.932 \times 10^{-2}$ Cputime<1 NFE=12	$J^*=0.12151168$ $m=110$ $\ g\ =1.628 \times 10^{-2}$ Cputime=1 NFE=111	$J^*=0.12000835$ $m=244$ $\ g\ =1.628 \times 10^{-2}$ Cputime=7 NFE=245	$J^*=0.12000835$ $m=244$ $\ g\ =1.628 \times 10^{-2}$ Cputime=13 NFE=245
2.0	$J^*=0.16079366$ $m=11$ $\ g\ =0.1080021$ Cputime<1 NFE=12	$J^*=0.12151942$ $m=105$ $\ g\ =1.628 \times 10^{-2}$ Cputime<1 NFE=106	$J^*=0.12001007$ $m=251$ $\ g\ =1.628 \times 10^{-2}$ Cputime=7 NFE=252	$J^*=0.12001007$ $m=251$ $\ g\ =1.628 \times 10^{-2}$ Cputime=15 NFE=252
2.3	$J^*=0.1712356$ $m=9$ $\ g\ =0.2340131$ Cputime<1 NFE=10	$J^*=0.12160342$ $m=115$ $\ g\ =1.630 \times 10^{-2}$ Cputime=1 NFE=116	$J^*=0.12001191$ $m=255$ $\ g\ =1.628 \times 10^{-2}$ Cputime=8 NFE=256	$J^*=0.12001009$ $m=254$ $\ g\ =1.628 \times 10^{-2}$ Cputime=15 NFE=255

TABLE (9.3.2):Results of GFS with change in u_0 for $ACC \leq 5.0 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	19	0.15018608	4.90165×10^{-2}	20	0.9	<1
2.0	10	0.15728053	4.93213×10^{-2}	11	1.9	<1
3.0	9	0.15361698	4.90792×10^{-2}	10	1.9	<1
4.0	10	0.14851412	4.66974×10^{-2}	11	1.9	<1
5.0	19	0.15018608	4.90165×10^{-2}	20	0.9	<1

TABLE (9.3.3):Results of GFS with change in u_0 for $ACC \leq 2.0 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	111	0.12376547	1.99736×10^{-2}	112	0.9	3
2.0	50	0.12344520	1.98793×10^{-2}	51	1.9	1
3.0	48	0.12347513	1.99591×10^{-2}	49	1.9	1
4.0	48	0.12337772	1.98493×10^{-2}	49	1.9	1
5.0	98	0.12350924	1.99507×10^{-2}	99	0.9	3

TABLE (9.3.4):Results of GFS with change in u_0 for $ACC \leq 1.628 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	—	—	—	—	—	—
2.0	417	0.12000845	1.62760×10^{-2}	418	1.9	13
3.0	286	0.12000847	1.62760×10^{-2}	287	1.9	8
4.0	244	0.12000835	1.62759×10^{-2}	245	1.9	7
5.0	—	—	—	—	—	—

TABLE (9.3.5):Results of SD with varying ϵ and N.

N ϵ	10	100	400	800
1.0	J*=0.15654120 m=13 $\ g\ =9.877 \times 10^{-2}$ Cputime<1 NFE=61	J*=0.12149884 m=457 $\ g\ =1.628 \times 10^{-2}$ Cputime=1 NFE=1943	J*=0.12009685 m=215 $\ g\ =1.627 \times 10^{-2}$ Cputime=21 NFE=1162	J*=0.12009683 m=215 $\ g\ =1.627 \times 10^{-2}$ Cputime=40 NFE=1162
1.5	J*=0.15572529 m=13 $\ g\ =9.792 \times 10^{-2}$ Cputime<1 NFE=57	J*=0.12150063 m=344 $\ g\ =1.628 \times 10^{-2}$ Cputime=10 NFE=1436	J*=0.11998067 m=224 $\ g\ =1.626 \times 10^{-2}$ Cputime=21 NFE=1122	J*=0.11998067 m=223 $\ g\ =1.626 \times 10^{-2}$ Cputime=40 NFE=1117
1.9	J*=0.16629115 m=8 $\ g\ =9.873 \times 10^{-2}$ Cputime<1 NFE=416	J*=0.12150366 m=272 $\ g\ =1.628 \times 10^{-2}$ Cputime=7 NFE=1137	J*=0.11993905 m=219 $\ g\ =1.625 \times 10^{-2}$ Cputime=25 NFE=1067	J*=0.11993905 m=219 $\ g\ =1.625 \times 10^{-2}$ Cputime=43 NFE=1067
2.0	J*=0.16001064 m=9 $\ g\ =9.852 \times 10^{-2}$ Cputime<1 NFE=38	J*=0.12150429 m=249 $\ g\ =1.628 \times 10^{-2}$ Cputime=5 NFE=1041	J*=0.11993895 m=213 $\ g\ =1.621 \times 10^{-2}$ Cputime=21 NFE=1026	J*=0.11993895 m=213 $\ g\ =1.621 \times 10^{-2}$ Cputime=40 NFE=1026
2.1	J*=0.17200524 m=4 $\ g\ =0.10370511$ Cputime<1 NFE=19	J*=0.12150429 m=240 $\ g\ =1.628 \times 10^{-2}$ Cputime=5 NFE=1002	J*=0.11994712 m=218 $\ g\ =1.627 \times 10^{-2}$ Cputime=20 NFE=1048	J*=0.11994709 m=218 $\ g\ =1.627 \times 10^{-2}$ Cputime=40 NFE=1048
2.3	J*=0.17201021 m=4 $\ g\ =0.10418521$ Cputime<1 NFE=19	J*=0.12150429 m=245 $\ g\ =1.628 \times 10^{-2}$ Cputime=5 NFE=1035	J*=0.11994801 m=219 $\ g\ =1.627 \times 10^{-2}$ Cputime=20 NFE=1069	J*=0.11994799 m=218 $\ g\ =1.627 \times 10^{-2}$ Cputime=40 NFE=1061

TABLE (9.3.6):Results of SD with change in u_0 for $ACC \leq 5.0 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	7	0.15750064	4.96722×10^{-2}	49	0.9	<1
2.0	8	0.15415585	4.95863×10^{-2}	36	2.0	1
3.0	10	0.14593692	4.34585×10^{-2}	44	2.0	1
4.0	9	0.14647810	4.55623×10^{-2}	39	2.0	1
5.0	10	0.14670852	4.58038×10^{-2}	54	0.9	1

TABLE (9.3.7):Results of SD with change in u_0 for $ACC \leq 2.0 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	31	0.12380461	1.95794×10^{-2}	206	0.9	3
2.0	30	0.12342590	1.93410×10^{-2}	140	2.0	3
3.0	30	0.12364164	1.95637×10^{-2}	137	2.0	3
4.0	29	0.12388872	1.98149×10^{-2}	132	2.0	3
5.0	32	0.12373973	1.95782×10^{-2}	197	0.9	3

TABLE (9.3.8):Results of SD with change in u_0 for $ACC \leq 1.6263 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	223	0.12019060	1.62620×10^{-2}	1262	0.9	25
2.0	214	0.11994554	1.62196×10^{-2}	1048	2.0	24
3.0	213	0.11999889	1.62529×10^{-2}	1043	2.0	21
4.0	213	0.11993895	1.62098×10^{-2}	1026	2.0	21
5.0	—	—	—	—	—	—

TABLE (9.3.9):Results of FR with varying ϵ and N.

N ϵ	10	100	400	800
0.5	$J^*=0.16288812$ $m=20$ $\ g\ =0.1790719$ Cputime<1 NFE=88	$J^*=0.12189481$ $m=43$ $\ g\ =1.598 \times 10^{-2}$ Cputime=1 NFE=219	$J^*=0.12034753$ $m=100$ $\ g\ =1.598 \times 10^{-2}$ Cputime=9 NFE=478	$J^*=0.12034753$ $m=99$ $\ g\ =1.598 \times 10^{-2}$ Cputime=18 NFE=469
1.0	$J^*=0.14294314$ $m=10$ $\ g\ =0.1389834$ Cputime<1 NFE=45	$J^*=0.12184268$ $m=31$ $\ g\ =1.597 \times 10^{-2}$ Cputime<1 NFE=144	$J^*=0.12026827$ $m=80$ $\ g\ =1.596 \times 10^{-2}$ Cputime=7 NFE=364	$J^*=0.12026826$ $m=80$ $\ g\ =1.596 \times 10^{-2}$ Cputime=14 NFE=364
1.3	$J^*=0.17496419$ $m=10$ $\ g\ =0.1885805$ Cputime<1 NFE=43	$J^*=0.12194301$ $m=27$ $\ g\ =1.598 \times 10^{-2}$ Cputime<1 NFE=124	$J^*=0.12035177$ $m=78$ $\ g\ =1.598 \times 10^{-2}$ Cputime=7 NFE=341	$J^*=0.12035177$ $m=78$ $\ g\ =1.598 \times 10^{-2}$ Cputime=13 NFE=341
1.4	$J^*=0.17465729$ $m=4$ $\ g\ =0.1870243$ Cputime<1 NFE=19	$J^*=0.12216884$ $m=29$ $\ g\ =1.598 \times 10^{-2}$ Cputime<1 NFE=126	$J^*=0.12009654$ $m=67$ $\ g\ =1.589 \times 10^{-2}$ Cputime=5 NFE=296	$J^*=0.12009654$ $m=67$ $\ g\ =1.589 \times 10^{-2}$ Cputime=9 NFE=296
1.5	$J^*=0.18363695$ $m=4$ $\ g\ =0.1983155$ Cputime<1 NFE=27	$J^*=0.12204595$ $m=30$ $\ g\ =1.598 \times 10^{-2}$ Cputime<1 NFE=131	$J^*=0.12044837$ $m=95$ $\ g\ =1.598 \times 10^{-2}$ Cputime=8 NFE=398	$J^*=0.12044837$ $m=95$ $\ g\ =1.598 \times 10^{-2}$ Cputime=15 NFE=398
1.7	$J^*=0.18591032$ $m=3$ $\ g\ =0.2145631$ Cputime<1 NFE=24	$J^*=0.12210391$ $m=30$ $\ g\ =1.598 \times 10^{-2}$ Cputime<1 NFE=131	$J^*=0.12044941$ $m=97$ $\ g\ =1.598 \times 10^{-2}$ Cputime=8 NFE=404	$J^*=0.12044939$ $m=97$ $\ g\ =1.598 \times 10^{-2}$ Cputime=16 NFE=404

TABLE (9.3.10):Results of FR with change in u_0 for $ACC \leq 5.0 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	5	0.13999659	4.38504×10^{-2}	29	1.4	<1
2.0	6	0.13407625	2.96083×10^{-2}	29	1.4	1
3.0	5	0.13999659	4.38504×10^{-2}	29	1.4	<1
4.0	7	0.13390277	3.98363×10^{-2}	33	1.4	1
5.0	8	0.13204935	3.30970×10^{-2}	44	0.9	1

TABLE (9.3.11):Results of FR with change in u_0 for $ACC \leq 2.0 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	8	0.12245834	1.82184×10^{-2}	46	1.4	1
2.0	11	0.12181071	1.79332×10^{-2}	55	1.4	1
3.0	8	0.12245834	1.82184×10^{-2}	46	1.4	1
4.0	12	0.12196881	1.79913×10^{-2}	58	1.4	1
5.0	13	0.12216973	1.88267×10^{-2}	75	0.9	2

TABLE (9.3.12):Results of FR with change in u_0 for $ACC \leq 1.60 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	100	0.12039648	1.59399×10^{-2}	417	1.4	9
2.0	86	0.12020817	1.59114×10^{-2}	372	1.4	8
3.0	67	0.12009654	1.58964×10^{-2}	296	1.4	5
4.0	70	0.12028246	1.59270×10^{-2}	301	1.4	6
5.0	100	0.12036894	1.59386×10^{-2}	440	0.9	9

TABLE (9.3.13):Results of PR with varying ϵ and N.

N ϵ	10	100	400	800
0.09	J*=0.15379918 m=20 $\ g\ =0.1035159$ Cputime<1 NFE=297	J*=0.12135162 m=70 $\ g\ =1.713 \times 10^{-2}$ Cputime=7 NFE=1976	J*=0.12002120 m=145 $\ g\ =1.626 \times 10^{-2}$ Cputime=77 NFE=5542	J*=0.12002120 m=144 $\ g\ =1.626 \times 10^{-2}$ Cputime=145 NFE=5534
0.10	J*=0.15375738 m=20 $\ g\ =0.1031916$ Cputime<1 NFE=277	J*=0.12134460 m=80 $\ g\ =1.693 \times 10^{-2}$ Cputime=6 NFE=1908	J*=0.11999519 m=145 $\ g\ =1.624 \times 10^{-2}$ Cputime=69 NFE=5259	J*=0.11999518 m=145 $\ g\ =1.624 \times 10^{-2}$ Cputime=135 NFE=5259
0.11	J*=0.15381491 m=20 $\ g\ =0.1045886$ Cputime<1 NFE=261	J*=0.12134523 m=81 $\ g\ =1.693 \times 10^{-2}$ Cputime=6 NFE=1768	J*=0.11999489 m=144 $\ g\ =1.624 \times 10^{-2}$ Cputime=64 NFE=4809	J*=0.11999489 m=144 $\ g\ =1.624 \times 10^{-2}$ Cputime=121 NFE=4809
0.12	J*=0.15382610 m=20 $\ g\ =0.1052193$ Cputime<1 NFE=245	J*=0.12134628 m=70 $\ g\ =1.693 \times 10^{-2}$ Cputime=6 NFE=1584	J*=0.11999510 m=144 $\ g\ =1.624 \times 10^{-2}$ Cputime=61 NFE=4442	J*=0.11999510 m=144 $\ g\ =1.624 \times 10^{-2}$ Cputime=118 NFE=4442
0.2	J*=0.15424785 m=15 $\ g\ =0.1061884$ Cputime<1 NFE=153	J*=0.12134916 m=55 $\ g\ =1.699 \times 10^{-2}$ Cputime=3 NFE=1015	J*=0.12016363 m=110 $\ g\ =1.723 \times 10^{-2}$ Cputime=33 NFE=2161	J*=0.12016361 m=110 $\ g\ =1.723 \times 10^{-2}$ Cputime=59 NFE=2161
0.23	J*=0.15425181 m=14 $\ g\ =0.1072443$ Cputime<1 NFE=151	J*=0.12135013 m=53 $\ g\ =1.711 \times 10^{-2}$ Cputime=3 NFE=1003	J*=0.12016521 m=110 $\ g\ =1.725 \times 10^{-2}$ Cputime=33 NFE=2173	J*=0.12016519 m=110 $\ g\ =1.724 \times 10^{-2}$ Cputime=59 NFE=2173

TABLE (9.3.14):Results of PR with change in u_0 for $ACC \leq 5.0 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	18	0.15978053	4.75269×10^{-2}	285	0.11	4
2.0	15	0.15721001	4.69841×10^{-2}	235	0.11	3
3.0	13	0.15627567	4.69734×10^{-2}	207	0.11	3
4.0	12	0.15077874	4.63935×10^{-2}	218	0.11	3
5.0	13	0.15087337	4.65923×10^{-2}	212	0.11	3

TABLE (9.3.15):Results of PR with change in u_0 for $ACC \leq 2.0 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	43	0.12398575	1.98555×10^{-2}	1049	0.11	14
2.0	40	0.12367667	1.96831×10^{-2}	1009	0.11	14
3.0	37	0.12365513	1.96397×10^{-2}	965	0.11	13
4.0	35	0.12347949	1.95800×10^{-2}	957	0.11	13
5.0	35	0.12360186	1.96063×10^{-2}	925	0.11	13

TABLE (9.3.16):Results of PR with change in u_0 for $ACC \leq 1.63 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	150	0.12008077	1.63135×10^{-2}	4957	0.11	65
2.0	146	0.12001346	1.62958×10^{-2}	4920	0.11	64
3.0	148	0.1200043	1.62870×10^{-2}	4844	0.11	64
4.0	144	0.11999489	1.62358×10^{-2}	4809	0.11	64
5.0	144	0.11999571	1.62823×10^{-2}	4843	0.11	64

TABLE (9.3.17):Results of H1 with varying ϵ and N.

N ϵ	10	100	400	800
1.0	J*=0.17733932 m=10 $\ g\ =9.908 \times 10^{-2}$ Cputime<1 NFE=44	J*=0.12232688 m=62 $\ g\ =1.597 \times 10^{-2}$ Cputime=1 NFE=285	J*=0.11982405 m=121 $\ g\ =1.597 \times 10^{-2}$ Cputime=12 NFE=576	J*=0.11982405 m=120 $\ g\ =1.597 \times 10^{-2}$ Cputime=24 NFE=566
1.4	J*=0.17278498 m=6 $\ g\ =9.918 \times 10^{-2}$ Cputime<1 NFE=29	J*=0.12164238 m=39 $\ g\ =1.596 \times 10^{-2}$ Cputime<1 NFE=178	J*=0.11982345 m=120 $\ g\ =1.597 \times 10^{-2}$ Cputime=11 NFE=535	J*=0.11982345 m=119 $\ g\ =1.597 \times 10^{-2}$ Cputime=20 NFE=525
1.5	J*=0.17451821 m=6 $\ g\ =9.970 \times 10^{-2}$ Cputime<1 NFE=28	J*=0.12191905 m=41 $\ g\ =1.597 \times 10^{-2}$ Cputime<1 NFE=175	J*=0.11979908 m=120 $\ g\ =1.595 \times 10^{-2}$ Cputime=11 NFE=551	J*=0.11979908 m=120 $\ g\ =1.595 \times 10^{-2}$ Cputime=20 NFE=551
1.6	J*=0.17530963 m=6 $\ g\ =9.973 \times 10^{-2}$ Cputime<1 NFE=28	J*=0.12185869 m=76 $\ g\ =1.597 \times 10^{-2}$ Cputime=1 NFE=319	J*=0.11987383 m=121 $\ g\ =1.598 \times 10^{-2}$ Cputime=11 NFE=548	J*=0.11987383 m=121 $\ g\ =1.598 \times 10^{-2}$ Cputime=20 NFE=548
2.0	J*=0.20098841 m=5 $\ g\ =0.106987$ Cputime<1 NFE=21	J*=0.12166820 m=130 $\ g\ =1.596 \times 10^{-2}$ Cputime<1 NFE=180	J*=0.11983945 m=122 $\ g\ =1.597 \times 10^{-2}$ Cputime=11 NFE=519	J*=0.11983943 m=122 $\ g\ =1.597 \times 10^{-2}$ Cputime=20 NFE=519

TABLE (9.3.18):Results of H1 with change in u_0 for $ACC \leq 5.0 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	7	0.13855714	3.76910×10^{-2}	37	1.2	1
2.0	6	0.13432360	3.72048×10^{-2}	29	1.5	1
3.0	6	0.13941722	3.88949×10^{-2}	28	1.5	1
4.0	7	0.13449875	3.72148×10^{-2}	32	1.5	1
5.0	6	0.15048155	4.91634×10^{-2}	27	1.2	1

TABLE (9.3.19):Results of H1 with change in u_0 for $ACC \leq 2.0 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	17	0.12410783	1.97710×10^{-2}	91	1.2	1
2.0	14	0.12395976	1.97248×10^{-2}	70	1.5	2
3.0	11	0.12261068	1.96016×10^{-2}	58	1.5	1
4.0	12	0.12328072	1.96879×10^{-2}	56	1.5	1
5.0	12	0.12381171	1.96996×10^{-2}	65	1.2	1

TABLE (9.3.20):Results of H1 with change in u_0 for $ACC \leq 1.60 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	128	0.11981111	1.59994×10^{-2}	595	1.2	12
2.0	127	0.11980949	1.59986×10^{-2}	588	1.5	12
3.0	120	0.11979908	1.59549×10^{-2}	551	1.5	11
4.0	134	0.11981637	1.59995×10^{-2}	605	1.5	12
5.0	99	0.11999296	1.59996×10^{-2}	464	1.2	9

TABLE (9.3.21):Results of ATH with varying ϵ and N.

N ϵ	10	100	400	800
1.0	$J^*=0.16733932$ $m=10$ $\ g\ =9.918 \times 10^{-2}$ Cputime<1 NFE=44	$J^*=0.12191387$ $m=62$ $\ g\ =1.597 \times 10^{-2}$ Cputime=1 NFE=285	$J^*=0.11984896$ $m=127$ $\ g\ =1.595 \times 10^{-2}$ Cputime=12 NFE=590	$J^*=0.11984894$ $m=127$ $\ g\ =1.595 \times 10^{-2}$ Cputime=24 NFE=590
1.2	$J^*=0.16878048$ $m=8$ $\ g\ =9.922 \times 10^{-2}$ Cputime<1 NFE=37	$J^*=0.12168302$ $m=48$ $\ g\ =1.597 \times 10^{-2}$ Cputime=1 NFE=223	$J^*=0.11984319$ $m=119$ $\ g\ =1.595 \times 10^{-2}$ Cputime=11 NFE=558	$J^*=0.11984319$ $m=118$ $\ g\ =1.595 \times 10^{-2}$ Cputime=20 NFE=548
1.3	$J^*=0.1644566$ $m=8$ $\ g\ =9.914 \times 10^{-2}$ Cputime<1 NFE=37	$J^*=0.12169886$ $m=39$ $\ g\ =1.597 \times 10^{-2}$ Cputime<1 NFE=180	$J^*=0.11984135$ $m=119$ $\ g\ =1.587 \times 10^{-2}$ Cputime=12 NFE=38	$J^*=0.11984135$ $m=119$ $\ g\ =1.587 \times 10^{-2}$ Cputime=24 NFE=38
1.4	$J^*=0.17272498$ $m=6$ $\ g\ =9.918 \times 10^{-2}$ Cputime<1 NFE=29	$J^*=0.12166746$ $m=46$ $\ g\ =1.595 \times 10^{-2}$ Cputime=1 NFE=208	$J^*=0.11988682$ $m=120$ $\ g\ =1.592 \times 10^{-2}$ Cputime=11 NFE=541	$J^*=0.11988682$ $m=120$ $\ g\ =1.592 \times 10^{-2}$ Cputime=20 NFE=541
2.0	$J^*=0.20380440$ $m=4$ $\ g\ =0.1066074$ Cputime<1 NFE=19	$J^*=0.12224316$ $m=31$ $\ g\ =1.599 \times 10^{-2}$ Cputime<1 NFE=127	$J^*=0.11985070$ $m=122$ $\ g\ =1.589 \times 10^{-2}$ Cputime=11 NFE=521	$J^*=0.11985070$ $m=121$ $\ g\ =1.589 \times 10^{-2}$ Cputime=20 NFE=510

TABLE (9.3.22):Results of ATH with change in u_0 for $ACC \leq 5.0 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	6	0.13663711	3.07048×10^{-2}	32	1.3	<1
2.0	6	0.12894283	2.48550×10^{-2}	31	1.3	<1
3.0	7	0.14223580	3.08488×10^{-2}	35	1.3	<1
4.0	5	0.13624345	3.07033×10^{-2}	26	1.3	<1
5.0	6	0.14255868	4.18792×10^{-2}	30	1.0	<1

TABLE (9.3.23):Results of ATH with change in u_0 for $ACC \leq 2.0 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	16	0.12392616	1.93348×10^{-2}	87	1.3	1
2.0	14	0.12419740	1.98015×10^{-2}	72	1.3	1
3.0	9	0.12361266	1.92913×10^{-2}	45	1.3	1
4.0	14	0.12389185	1.93153×10^{-2}	73	1.3	1
5.0	18	0.12422641	1.99855×10^{-2}	97	1.0	1

TABLE (9.3.24):Results of ATH with change in u_0 for $ACC \leq 1.60 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	126	0.11992437	1.58894×10^{-2}	577	1.3	11
2.0	121	0.11986701	1.58883×10^{-2}	550	1.3	10
3.0	119	0.11984135	1.58782×10^{-2}	535	1.3	10
4.0	124	0.11985712	1.58859×10^{-2}	566	1.3	11
5.0	121	0.11985118	1.58853×10^{-2}	575	1.0	11

TABLE (9.3.25):Results of H3 with varying ϵ and N.

N ϵ	10	100	400	800
1.0	J*=0.16733932 m=10 $\ g\ =9.916 \times 10^{-2}$ Cputime<1 NFE=44	J*=0.12189907 m=62 $\ g\ =1.599 \times 10^{-2}$ Cputime=1 NFE=285	J*=0.11982405 m=121 $\ g\ =1.588 \times 10^{-2}$ Cputime=12 NFE=576	J*=0.11982403 m=121 $\ g\ =1.588 \times 10^{-2}$ Cputime=24 NFE=565
1.3	J*=0.16944566 m=8 $\ g\ =9.917 \times 10^{-2}$ Cputime<1 NFE=37	J*=0.12169143 m=56 $\ g\ =1.598 \times 10^{-2}$ Cputime=1 NFE=254	J*=0.11983722 m=117 $\ g\ =1.588 \times 10^{-2}$ Cputime=12 NFE=545	J*=0.11983721 m=116 $\ g\ =1.588 \times 10^{-2}$ Cputime=24 NFE=534
1.4	J*=0.17272498 m=6 $\ g\ =9.918 \times 10^{-2}$ Cputime<1 NFE=29	J*=0.12164238 m=39 $\ g\ =1.598 \times 10^{-2}$ Cputime<1 NFE=178	J*=0.11981238 m=113 $\ g\ =1.585 \times 10^{-2}$ Cputime=10 NFE=506	J*=0.11981238 m=113 $\ g\ =1.585 \times 10^{-2}$ Cputime=20 NFE=506
1.5	J*=0.17451821 m=6 $\ g\ =9.993 \times 10^{-2}$ Cputime<1 NFE=28	J*=0.12191905 m=41 $\ g\ =1.599 \times 10^{-2}$ Cputime<1 NFE=175	J*=0.11981452 m=118 $\ g\ =1.585 \times 10^{-2}$ Cputime=10 NFE=543	J*=0.11981452 m=118 $\ g\ =1.585 \times 10^{-2}$ Cputime=20 NFE=543
2.0	J*=0.2009004 m=5 $\ g\ =0.106987$ Cputime<1 NFE=21	J*=0.12166820 m=43 $\ g\ =1.598 \times 10^{-2}$ Cputime<1 NFE=180	J*=0.11983082 m=121 $\ g\ =1.589 \times 10^{-2}$ Cputime=10 NFE=515	J*=0.11983082 m=120 $\ g\ =1.589 \times 10^{-2}$ Cputime=20 NFE=505

TABLE (9.3.26)::Results of H3 with change in u_0 for $ACC \leq 5.0 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	8	0.13074902	3.00439×10^{-2}	43	1.4	<1
2.0	6	0.13568401	3.25334×10^{-2}	30	1.4	<1
3.0	6	0.13096619	3.01442×10^{-2}	36	1.4	<1
4.0	7	0.13931906	4.68011×10^{-2}	33	1.3	<1
5.0	8	0.14000118	4.78633×10^{-2}	40	1.0	<1

TABLE (9.3.27):Results of ATH with change in u_0 for $ACC \leq 2.0 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	12	0.12319517	1.86955×10^{-2}	66	1.4	1
2.0	14	0.12353310	1.91408×10^{-2}	72	1.4	1
3.0	10	0.12250342	1.84277×10^{-2}	75	1.3	2
4.0	12	0.12334539	1.93354×10^{-2}	58	1.3	1
5.0	20	0.12397129	1.97044×10^{-2}	108	1.0	2

TABLE (9.3.28):Results of H3 with change in u_0 for $ACC \leq 1.60 \times 10^{-2}$.

u_0	m	J^*	$\ g\ $	NFE	eps	Cputime
1.0	126	0.11981779	1.58632×10^{-2}	567	1.4	11
2.0	125	0.11981299	1.58545×10^{-2}	567	1.4	12
3.0	113	0.11981238	1.58536×10^{-2}	506	1.4	10
4.0	121	0.11982717	1.58771×10^{-2}	549	1.3	11
5.0	118	0.11984933	1.59185×10^{-2}	578	1.0	11

Table (9.4.1): Summary table for the seven methods

method u_0	GFS	SD	FR	PR	H1	ATH	H3
1.0	$m = 111$ $J^* = 0.12376547$ $\ g\ = 1.99736 \times 10^{-2}$ $\varepsilon = 0.9$ NFE = 112 Cputime = 3	$m = 223$ $J^* = 0.12019060$ $\ g\ = 1.62220 \times 10^{-2}$ $\varepsilon = 0.9$ NFE = 1262 Cputime = 25	$m = 100$ $J^* = 0.12039648$ $\ g\ = 1.59399 \times 10^{-2}$ $\varepsilon = 1.4$ NFE = 417 Cputime = 9	$m = 150$ $J^* = 0.12008077$ $\ g\ = 1.63135 \times 10^{-2}$ $\varepsilon = 0.11$ NFE = 4957 Cputime = 65	$m = 128$ $J^* = 0.1198049$ $\ g\ = 1.59994 \times 10^{-2}$ $\varepsilon = 1.2$ NFE = 595 Cputime = 12	$m = 126$ $J^* = 0.11992437$ $\ g\ = 1.58894 \times 10^{-2}$ $\varepsilon = 1.3$ NFE = 577 Cputime = 11	$m = 126$ $J^* = 0.11981779$ $\ g\ = 1.58632 \times 10^{-2}$ $\varepsilon = 1.4$ NFE = 567 Cputime = 11
2.0	$m = 417$ $J^* = 0.12000845$ $\ g\ = 1.62760 \times 10^{-2}$ $\varepsilon = 1.9$ NFE = 418 Cputime = 13	$m = 214$ $J^* = 0.11994554$ $\ g\ = 1.62196 \times 10^{-2}$ $\varepsilon = 2.0$ NFE = 1048 Cputime = 24	$m = 86$ $J^* = 0.12020817$ $\ g\ = 1.59114 \times 10^{-2}$ $\varepsilon = 1.4$ NFE = 372 Cputime = 8	$m = 146$ $J^* = 0.12001346$ $\ g\ = 1.62958 \times 10^{-2}$ $\varepsilon = 0.11$ NFE = 4920 Cputime = 64	$m = 127$ $J^* = 0.1198049$ $\ g\ = 1.59986 \times 10^{-2}$ $\varepsilon = 1.5$ NFE = 588 Cputime = 12	$m = 121$ $J^* = 0.11986701$ $\ g\ = 1.58831 \times 10^{-2}$ $\varepsilon = 1.3$ NFE = 550 Cputime = 10	$m = 125$ $J^* = 0.11981299$ $\ g\ = 1.58545 \times 10^{-2}$ $\varepsilon = 1.4$ NFE = 567 Cputime = 12
3.0	$m = 286$ $J^* = 0.12000847$ $\ g\ = 1.62760 \times 10^{-2}$ $\varepsilon = 1.9$ NFE = 287 Cputime = 8	$m = 213$ $J^* = 0.11999889$ $\ g\ = 1.62529 \times 10^{-2}$ $\varepsilon = 2.0$ NFE = 1043 Cputime = 21	$m = 67$ $J^* = 0.12009654$ $\ g\ = 1.58964 \times 10^{-2}$ $\varepsilon = 1.4$ NFE = 296 Cputime = 5	$m = 148$ $J^* = 0.1200043$ $\ g\ = 1.62870 \times 10^{-2}$ $\varepsilon = 0.11$ NFE = 4844 Cputime = 64	$m = 120$ $J^* = 0.11979908$ $\ g\ = 1.59549 \times 10^{-2}$ $\varepsilon = 1.5$ NFE = 551 Cputime = 11	$m = 119$ $J^* = 0.11984135$ $\ g\ = 1.58782 \times 10^{-2}$ $\varepsilon = 1.3$ NFE = 535 Cputime = 10	$m = 113$ $J^* = 0.11981238$ $\ g\ = 1.58536 \times 10^{-2}$ $\varepsilon = 1.4$ NFE = 506 Cputime = 10
4.0	$m = 244$ $J^* = 0.12000835$ $\ g\ = 1.62759 \times 10^{-2}$ $\varepsilon = 1.9$ NFE = 245 Cputime = 7	$m = 214$ $J^* = 0.11993895$ $\ g\ = 1.62098 \times 10^{-2}$ $\varepsilon = 2.0$ NFE = 1026 Cputime = 21	$m = 70$ $J^* = 0.12028246$ $\ g\ = 1.59270 \times 10^{-2}$ $\varepsilon = 1.4$ NFE = 301 Cputime = 6	$m = 144$ $J^* = 0.11999489$ $\ g\ = 1.62358 \times 10^{-2}$ $\varepsilon = 0.11$ NFE = 4809 Cputime = 64	$m = 134$ $J^* = 0.11981637$ $\ g\ = 1.5995 \times 10^{-2}$ $\varepsilon = 1.5$ NFE = 605 Cputime = 12	$m = 124$ $J^* = 0.11985712$ $\ g\ = 1.58859 \times 10^{-2}$ $\varepsilon = 1.3$ NFE = 566 Cputime = 11	$m = 121$ $J^* = 0.11982717$ $\ g\ = 1.58771 \times 10^{-2}$ $\varepsilon = 1.3$ NFE = 549 Cputime = 11
5.0	$m = 98$ $J^* = 0.12350924$ $\ g\ = 1.99507 \times 10^{-2}$ $\varepsilon = 0.9$ NFE = 99 Cputime = 3	$m = 32$ $J^* = 0.12373973$ $\ g\ = 1.95782 \times 10^{-2}$ $\varepsilon = 0.9$ NFE = 197 Cputime = 3	$m = 100$ $J^* = 0.12036894$ $\ g\ = 1.59386 \times 10^{-2}$ $\varepsilon = 0.9$ NFE = 440 Cputime = 9	$m = 144$ $J^* = 0.11999571$ $\ g\ = 1.62823 \times 10^{-2}$ $\varepsilon = 0.11$ NFE = 4843 Cputime = 64	$m = 99$ $J^* = 0.11999296$ $\ g\ = 1.59996 \times 10^{-2}$ $\varepsilon = 1.2$ NFE = 464 Cputime = 9	$m = 121$ $J^* = 0.11985118$ $\ g\ = 1.5883 \times 10^{-2}$ $\varepsilon = 1.0$ NFE = 575 Cputime = 11	$m = 118$ $J^* = 0.11984933$ $\ g\ = 1.59185 \times 10^{-2}$ $\varepsilon = 1.0$ NFE = 578 Cputime = 11

GFS

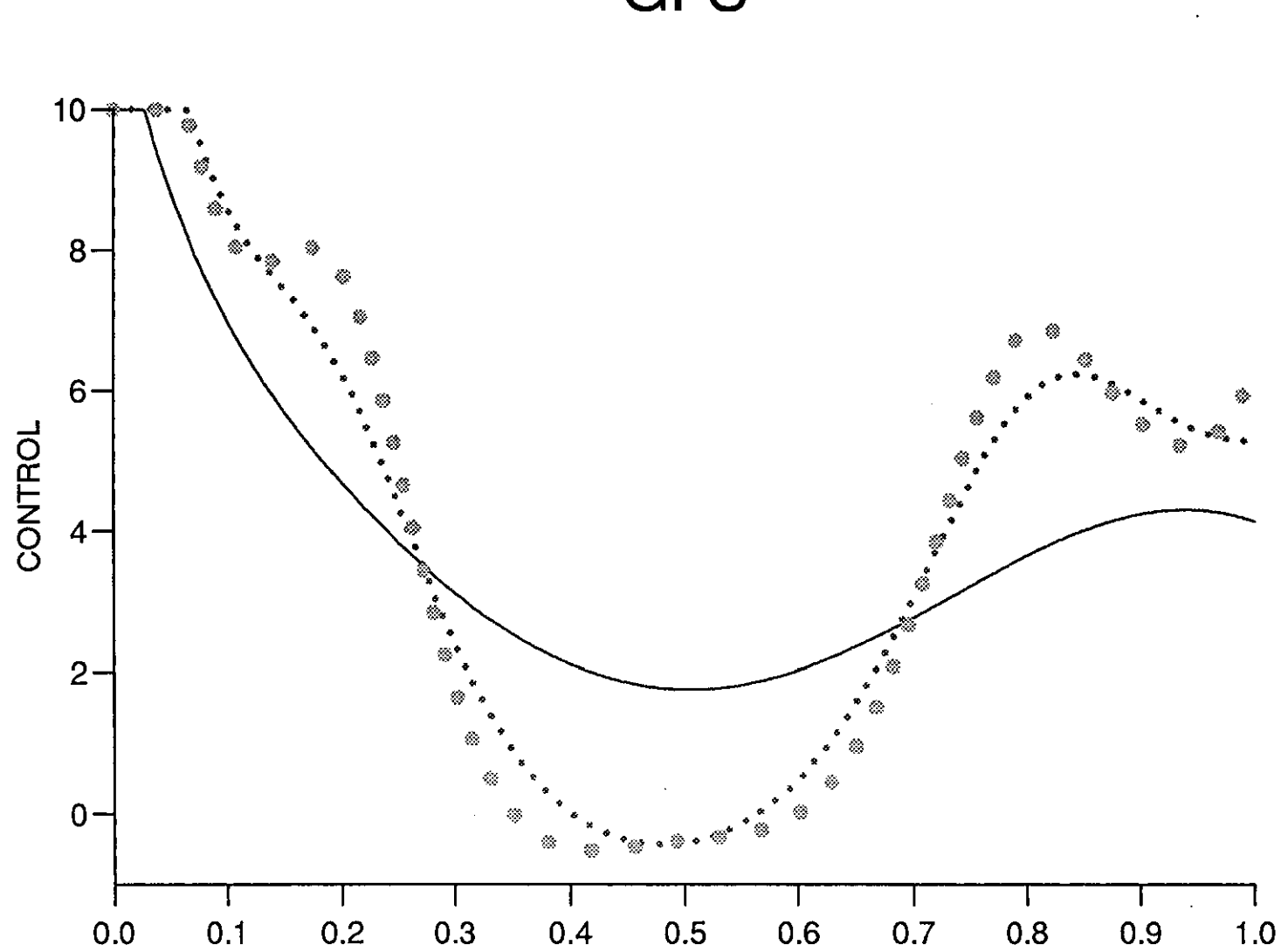


Figure (9.3.1)

GFS

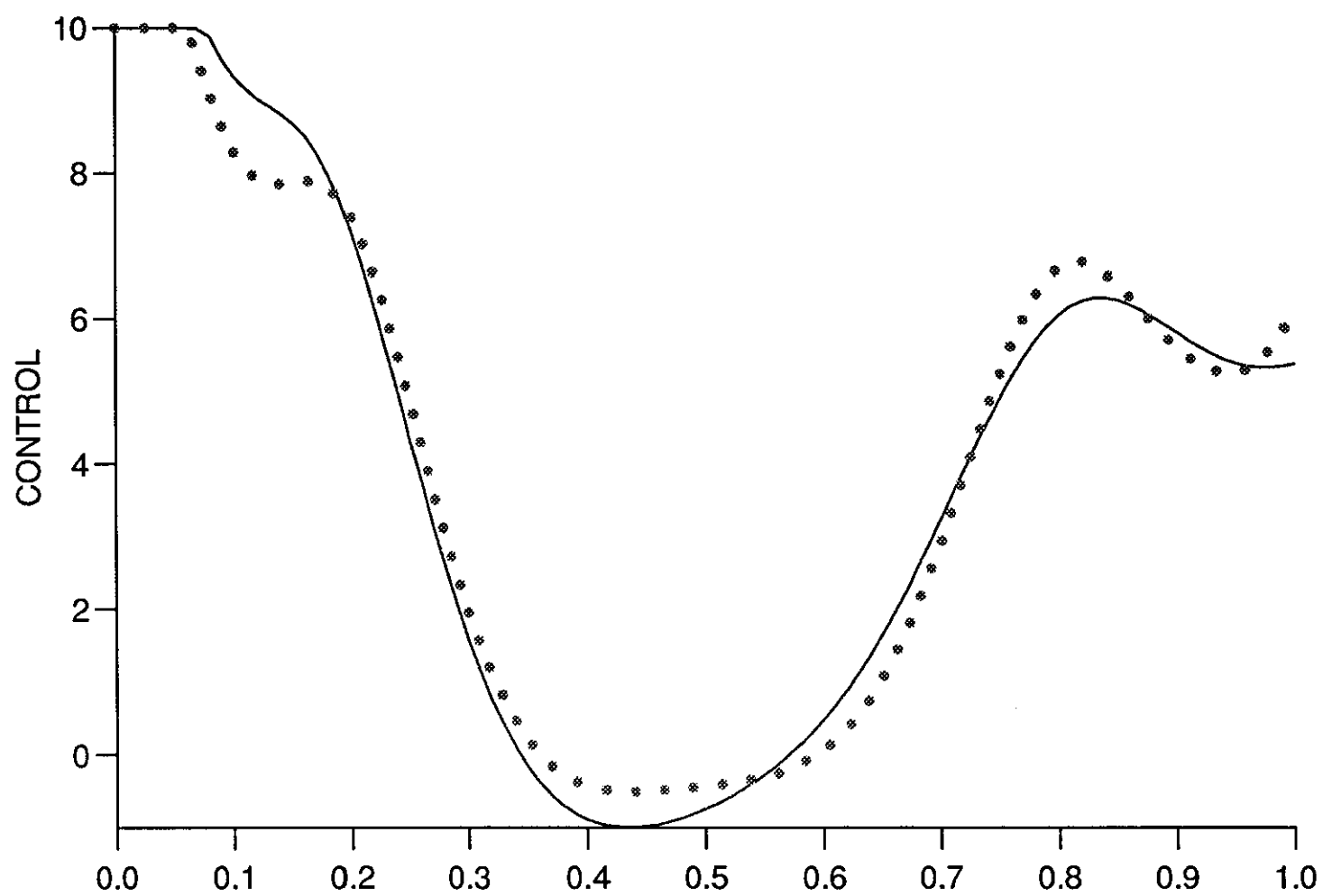


Figure (9.3.2)

GFS

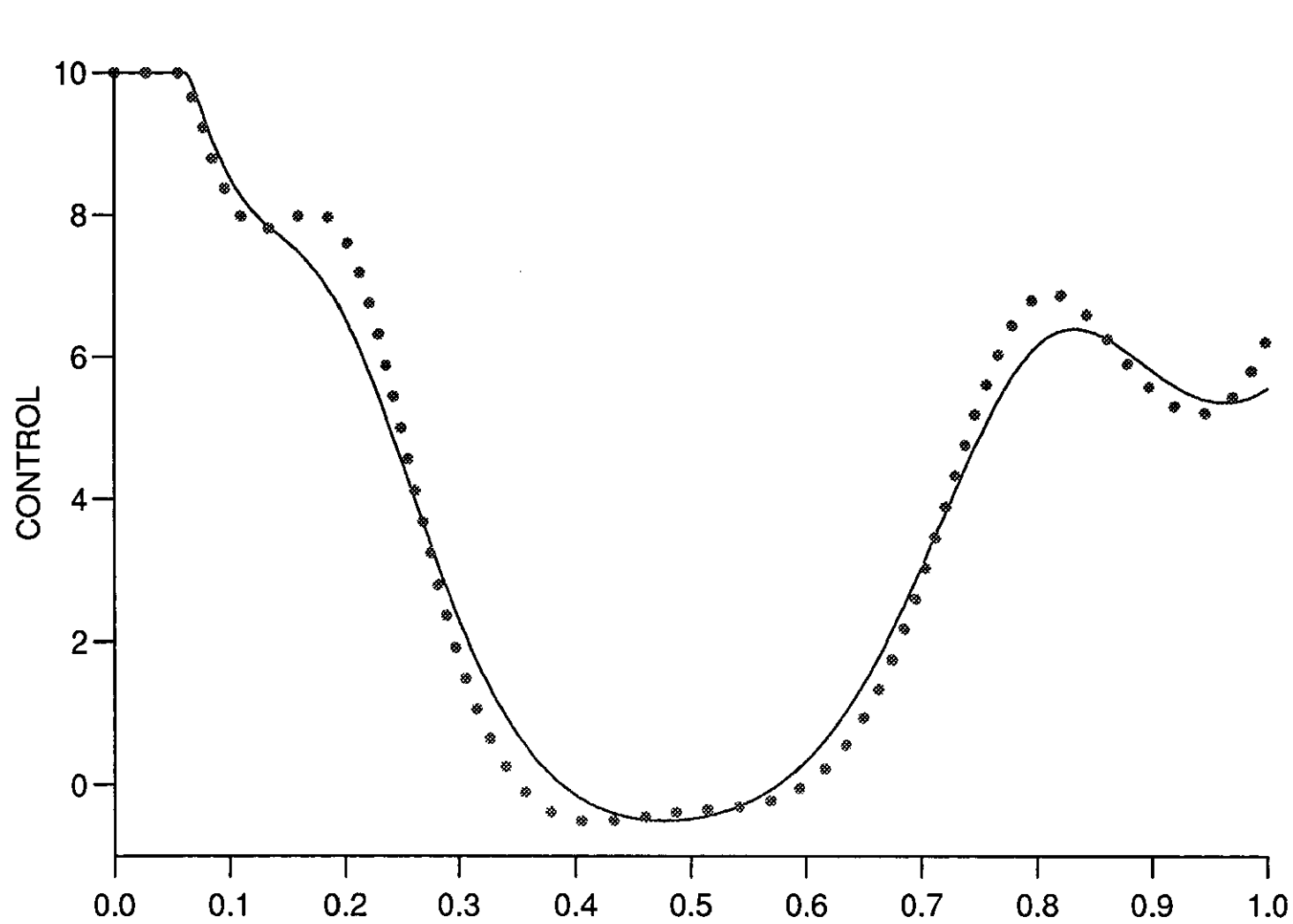


Figure (9.3.3)

GFS

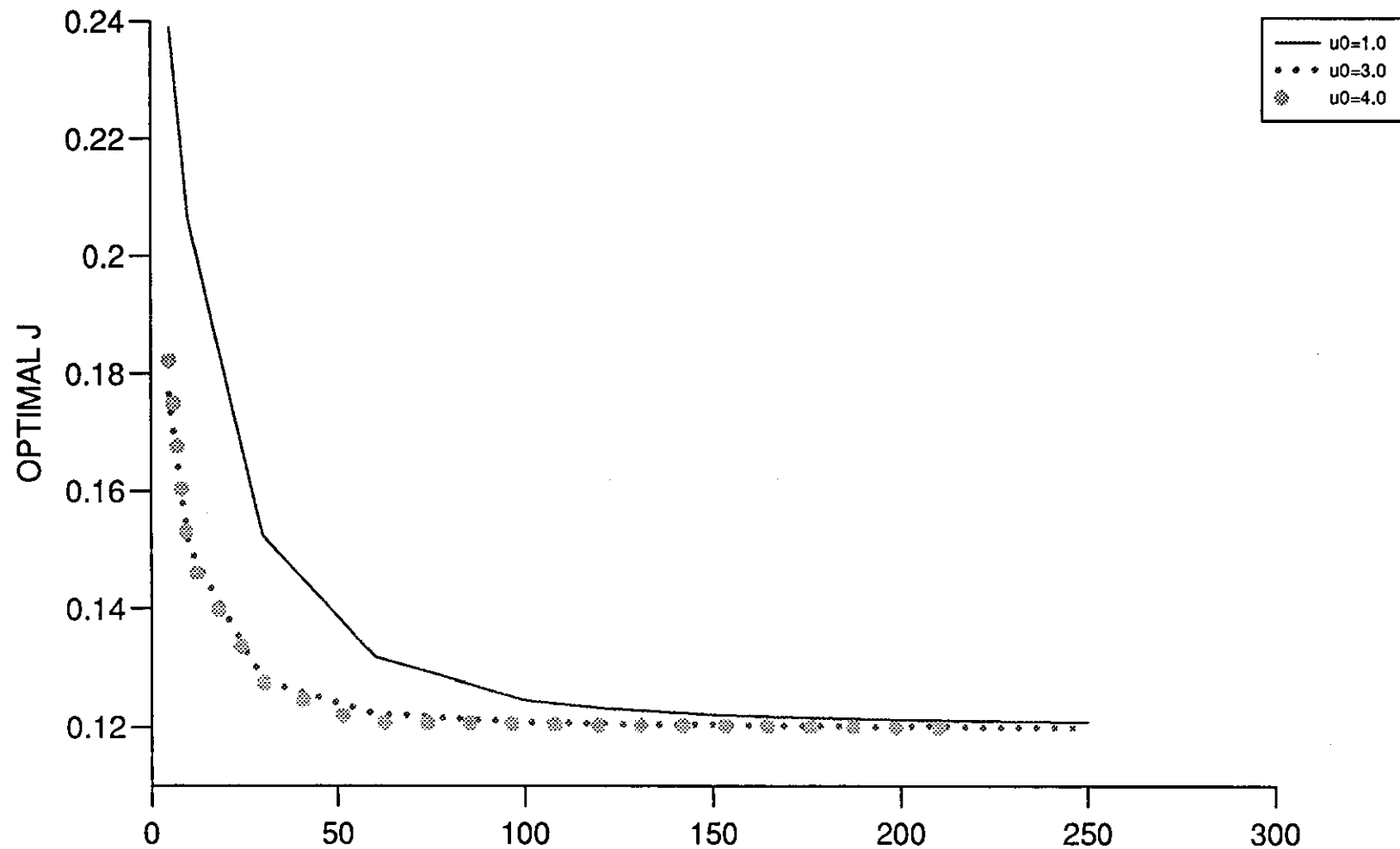


Figure (9.3.4)

SD

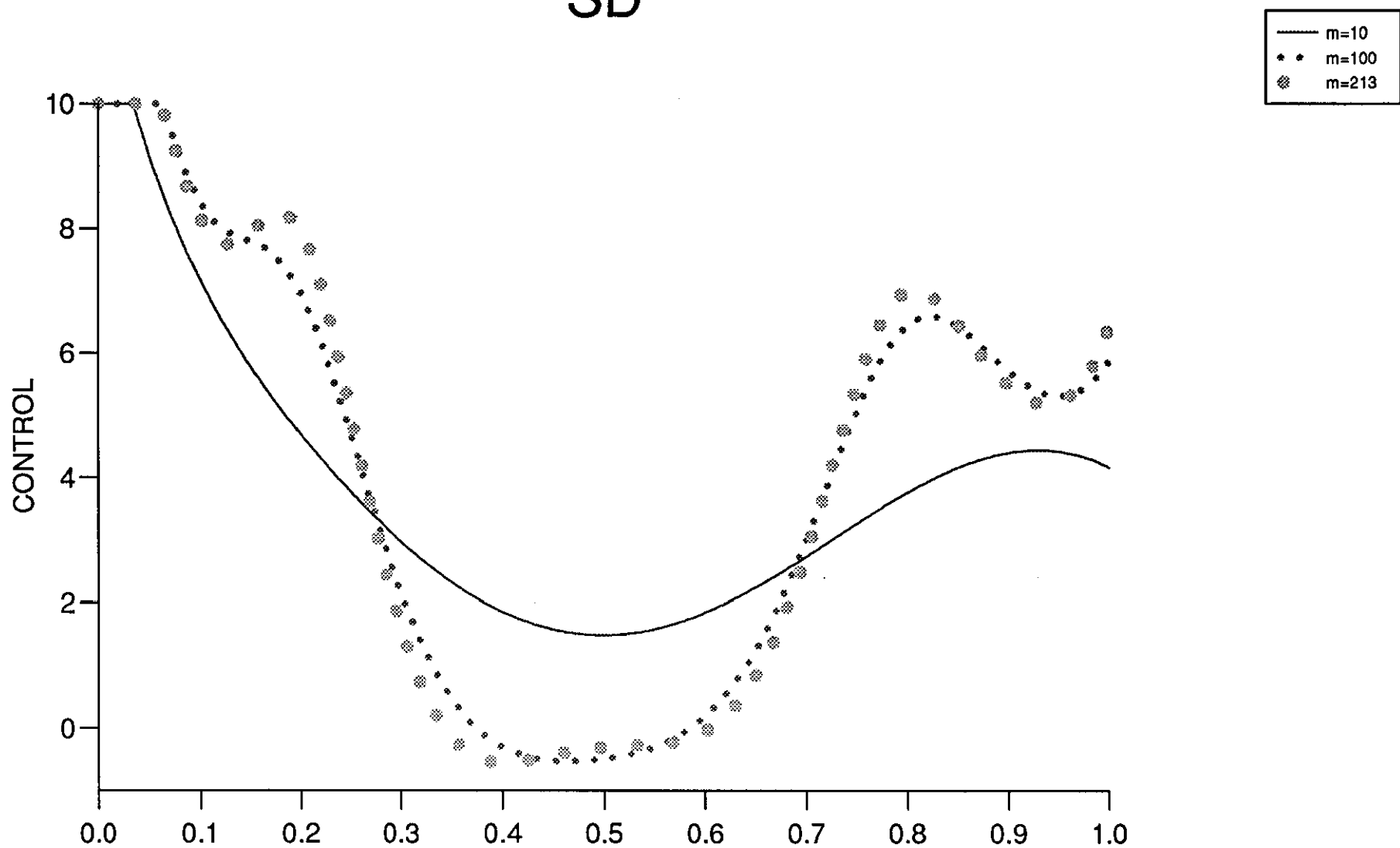


Figure (9.3.5)

SD

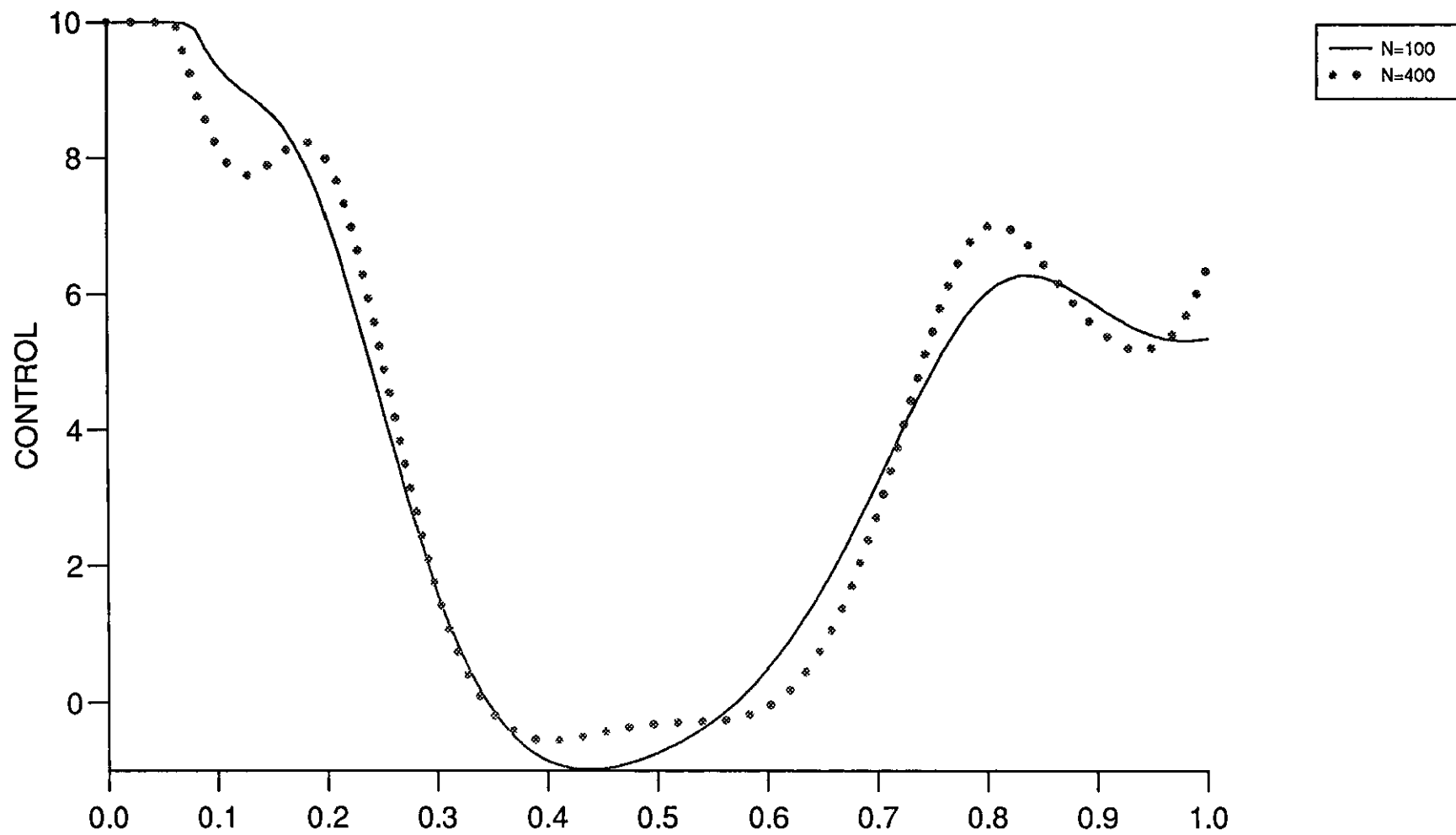


Figure (9.3.6)

SD

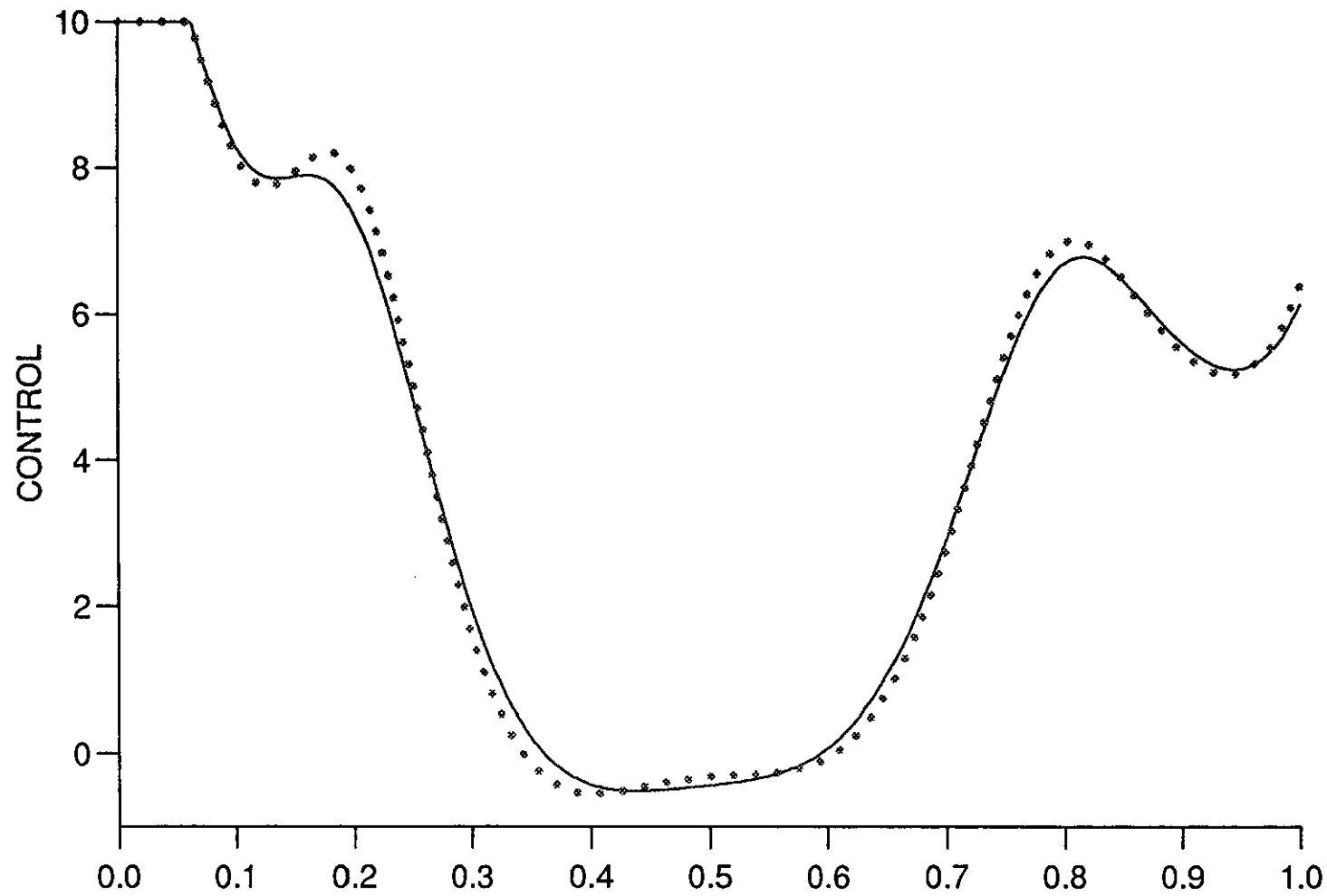


Figure (9.3.7)

SD

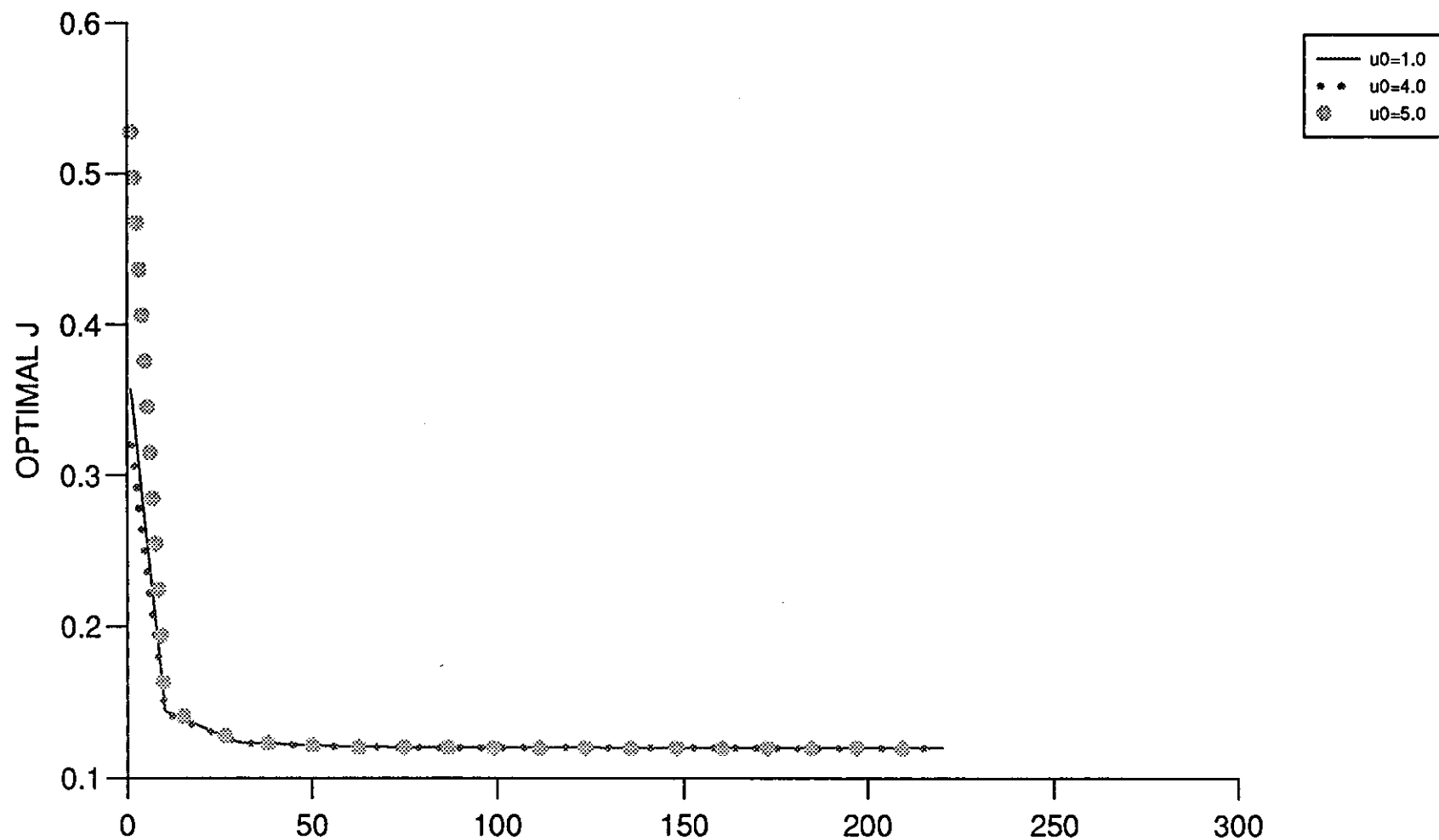


Figure (9.3.8)

FR

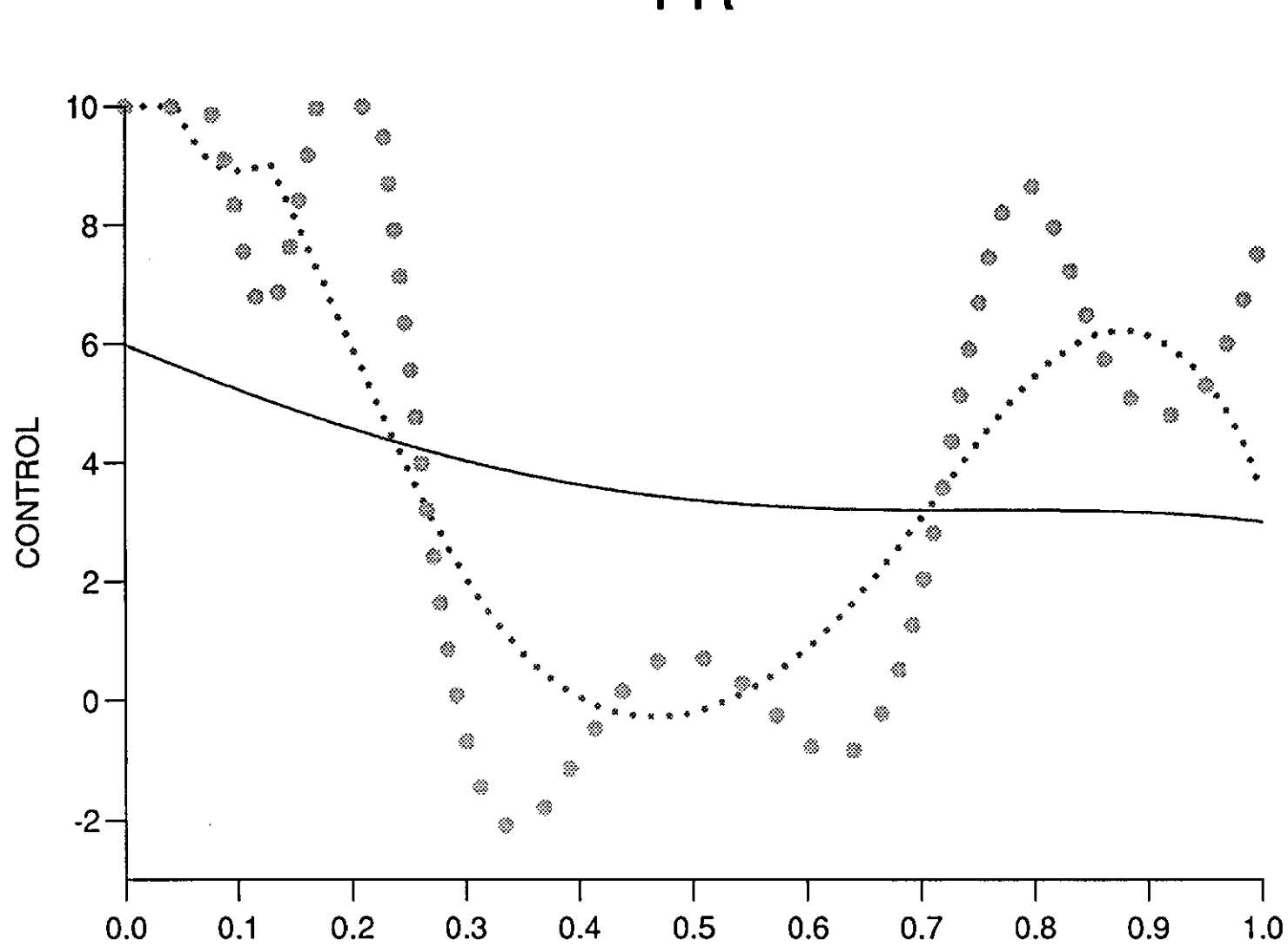


Figure (9.3.9)

FR

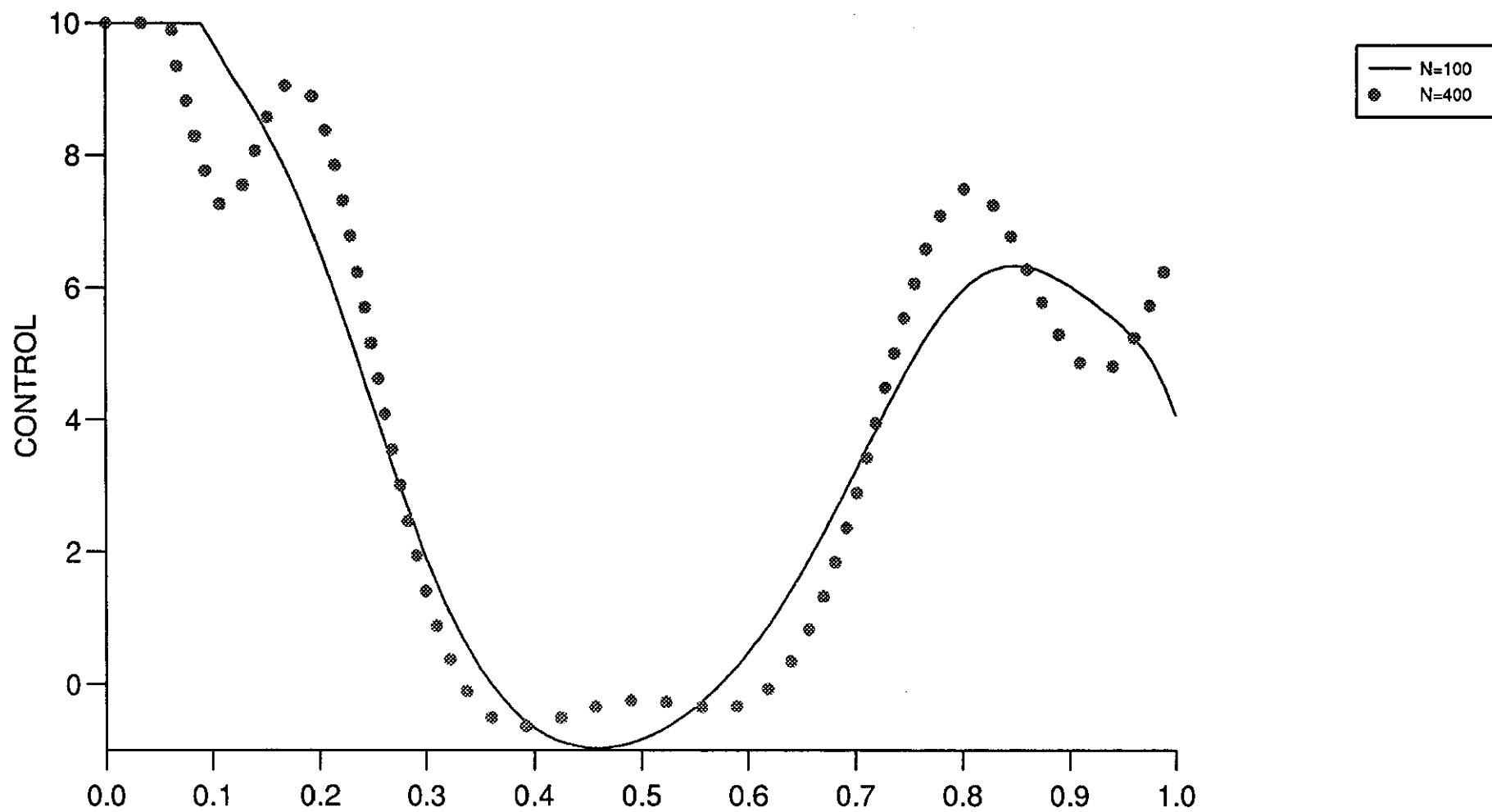


Figure (9.3.10)

FR

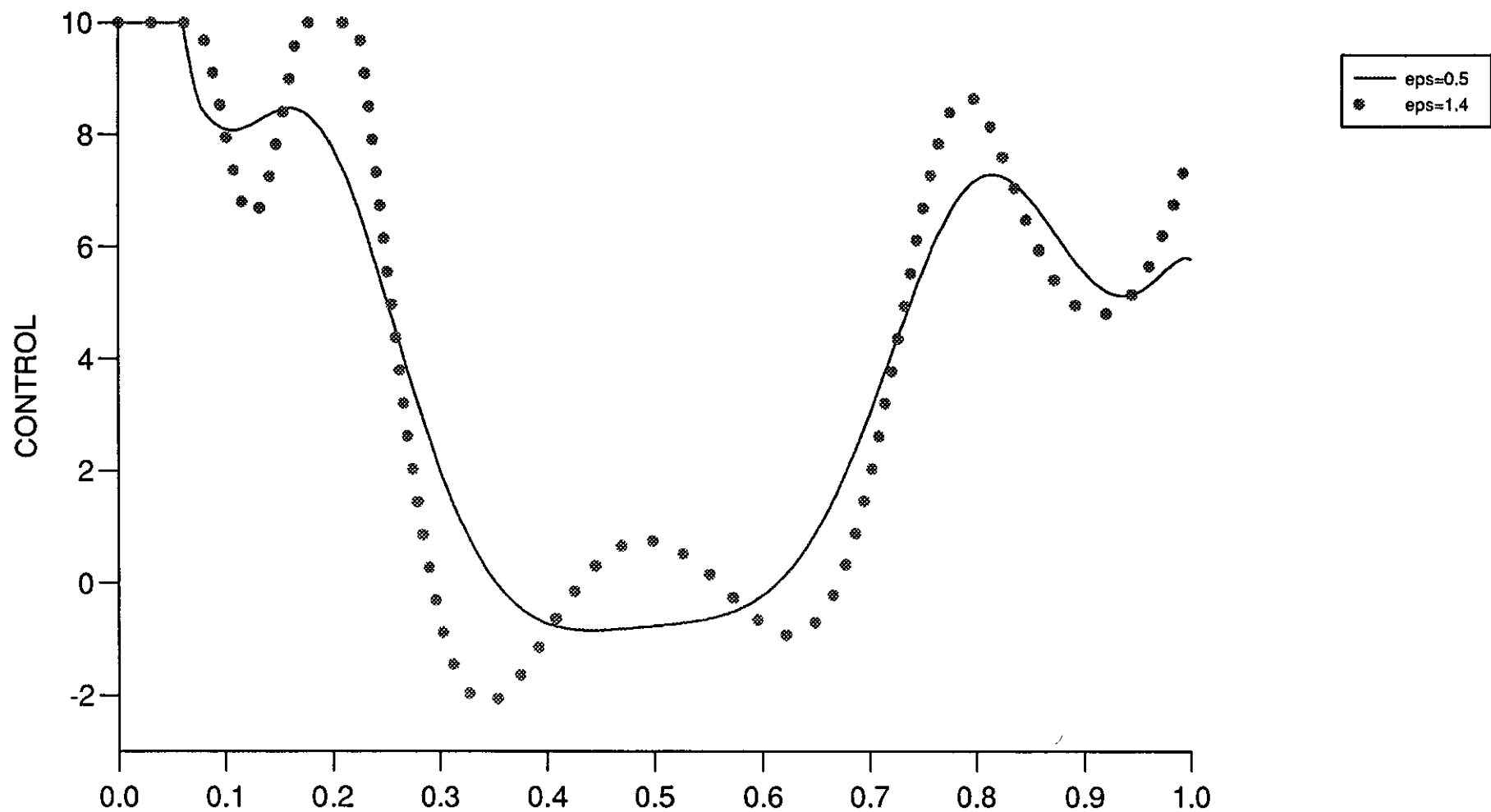


Figure (9.3.11)

FR

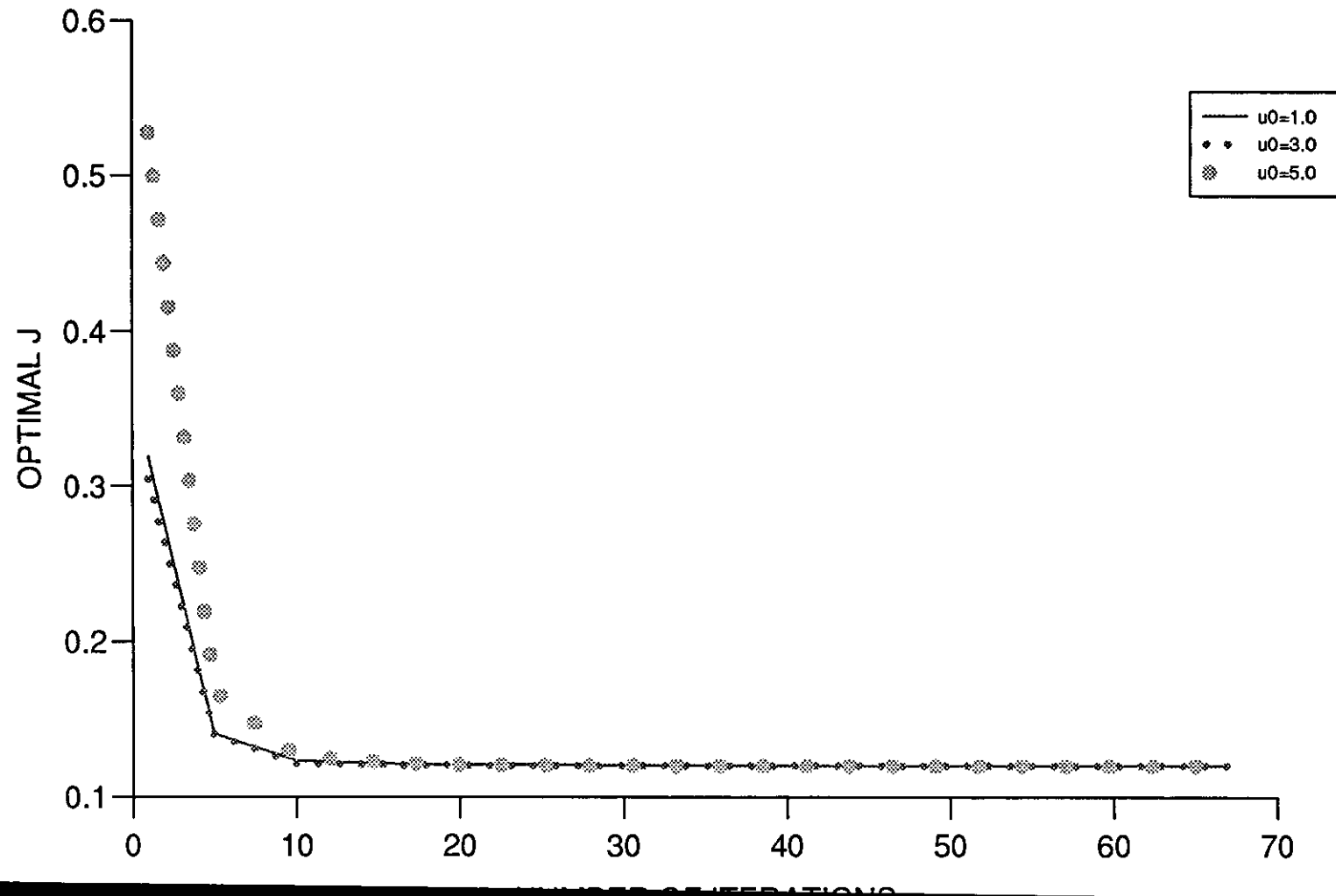


Figure (9.3.12)

PR

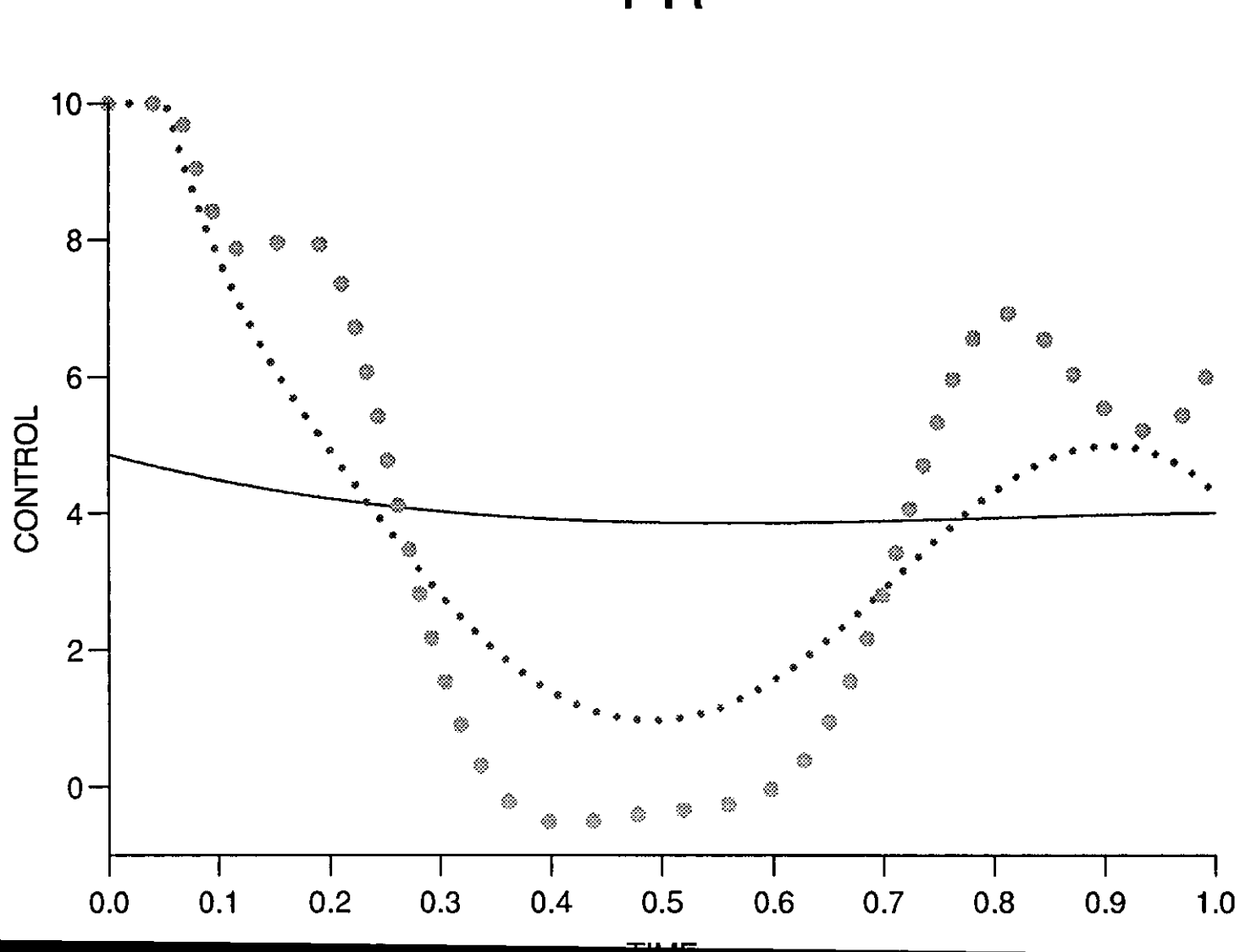


Figure (9.3.13)

PR

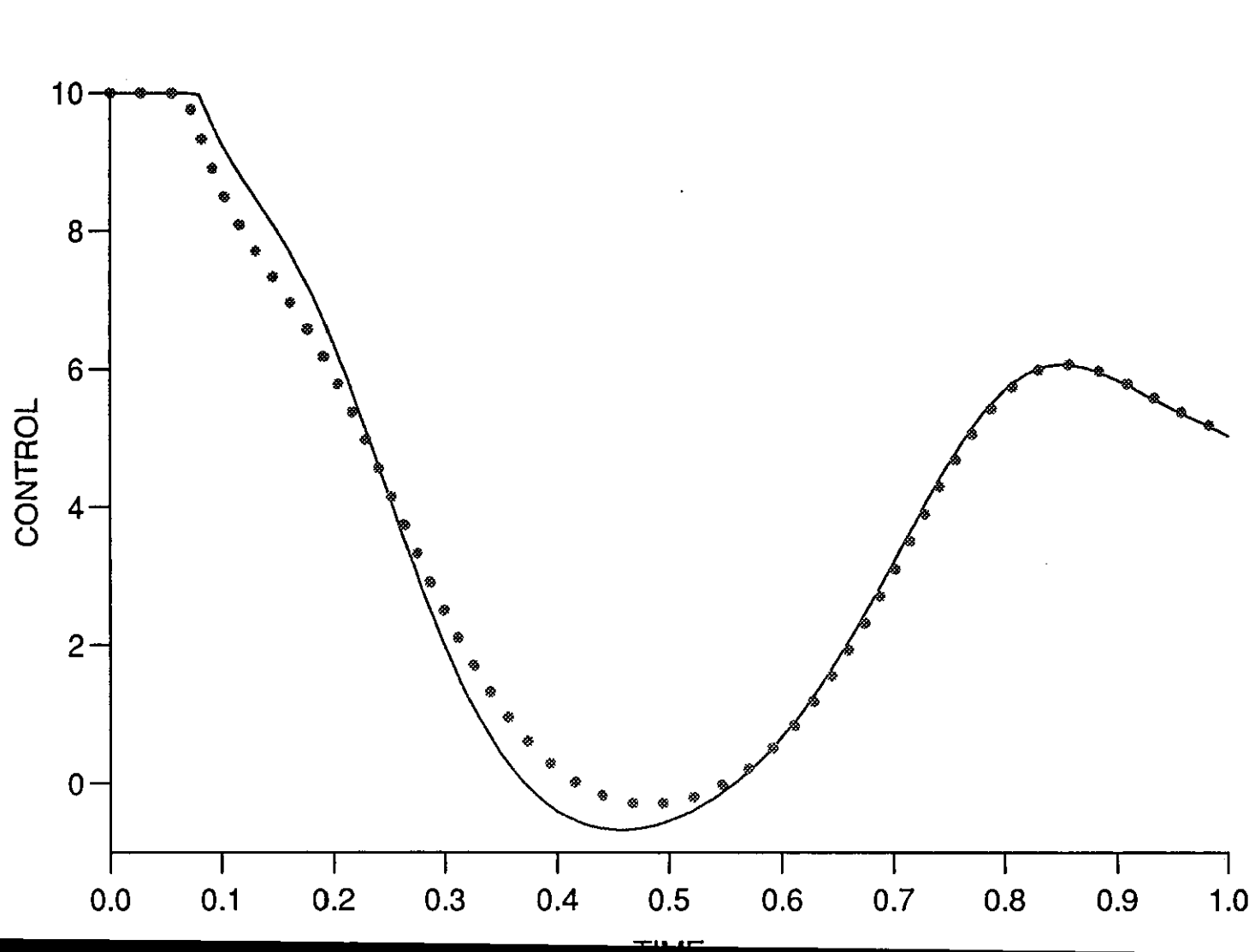


Figure (9.3.14)

PR

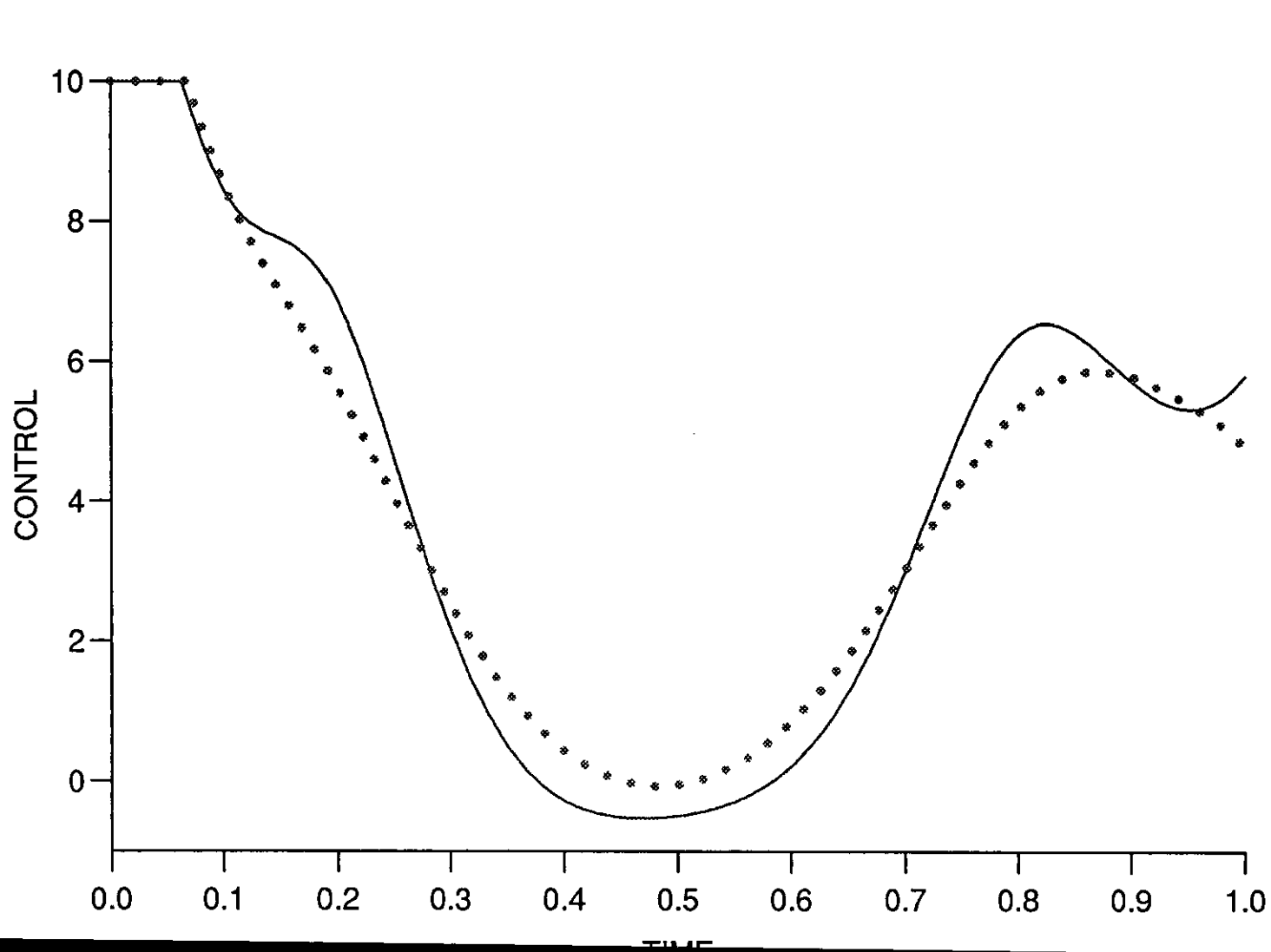


Figure (9.3.15)

PR

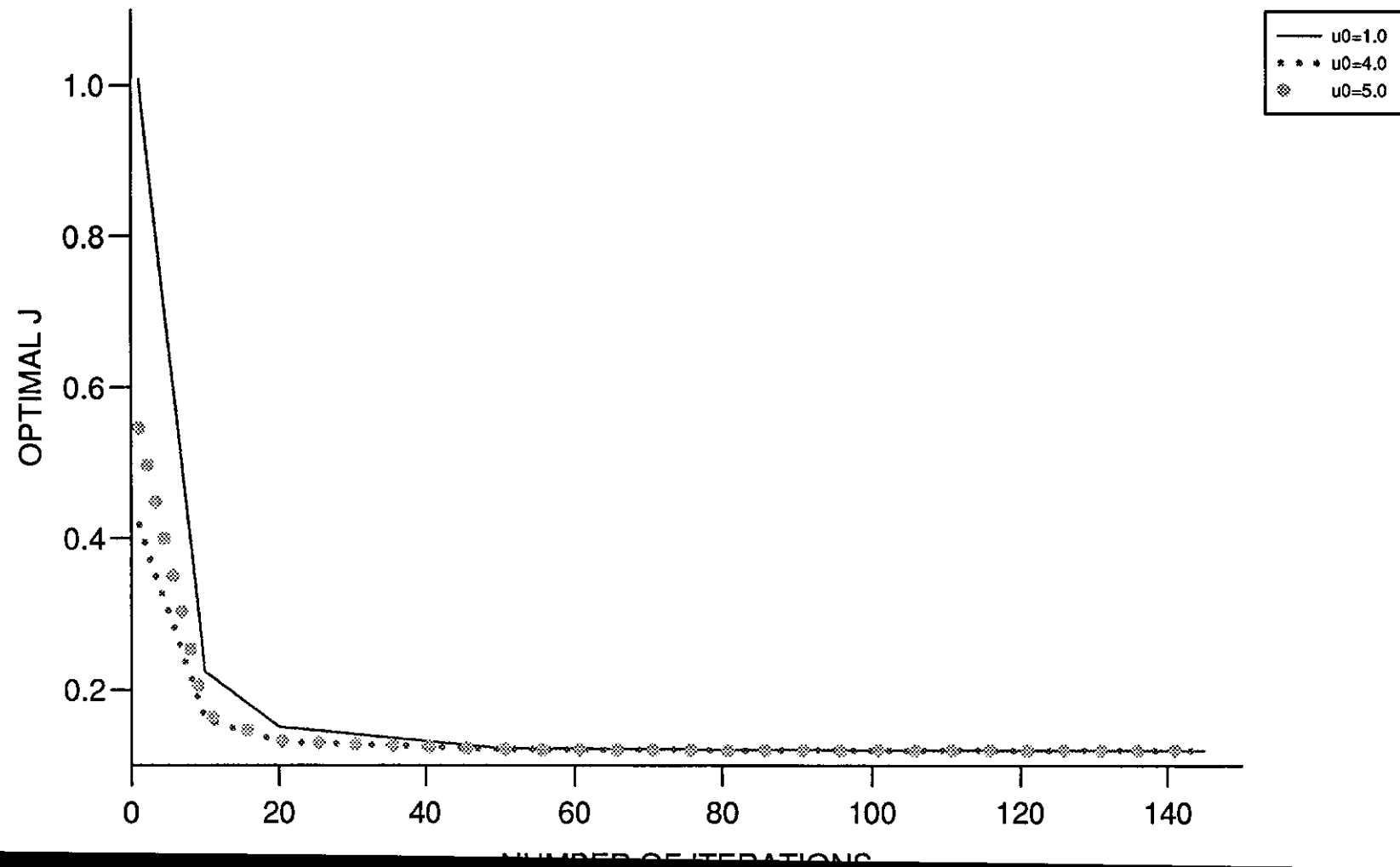


Figure (9.3.16)

H1

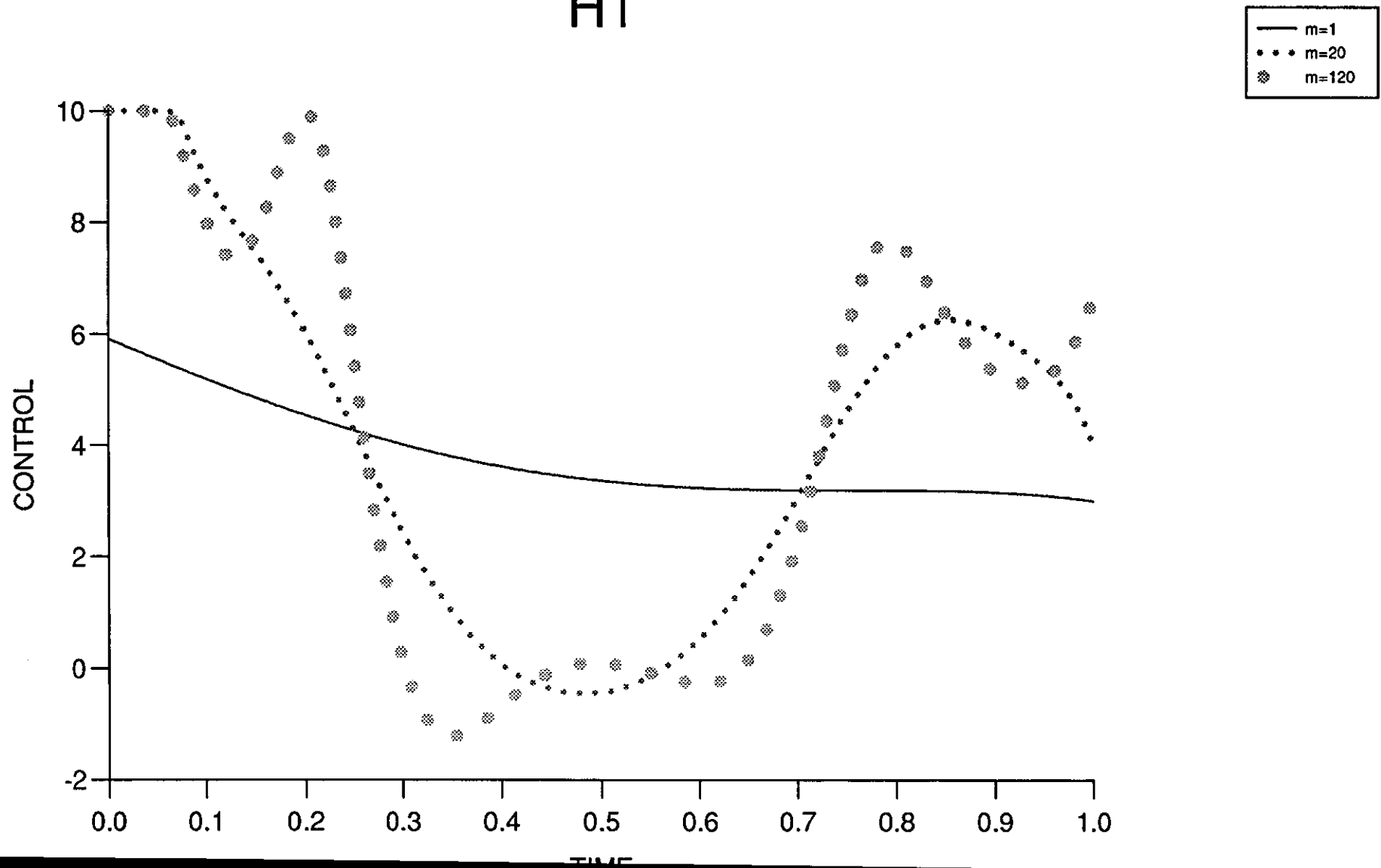


Figure (9.3.17)

H1

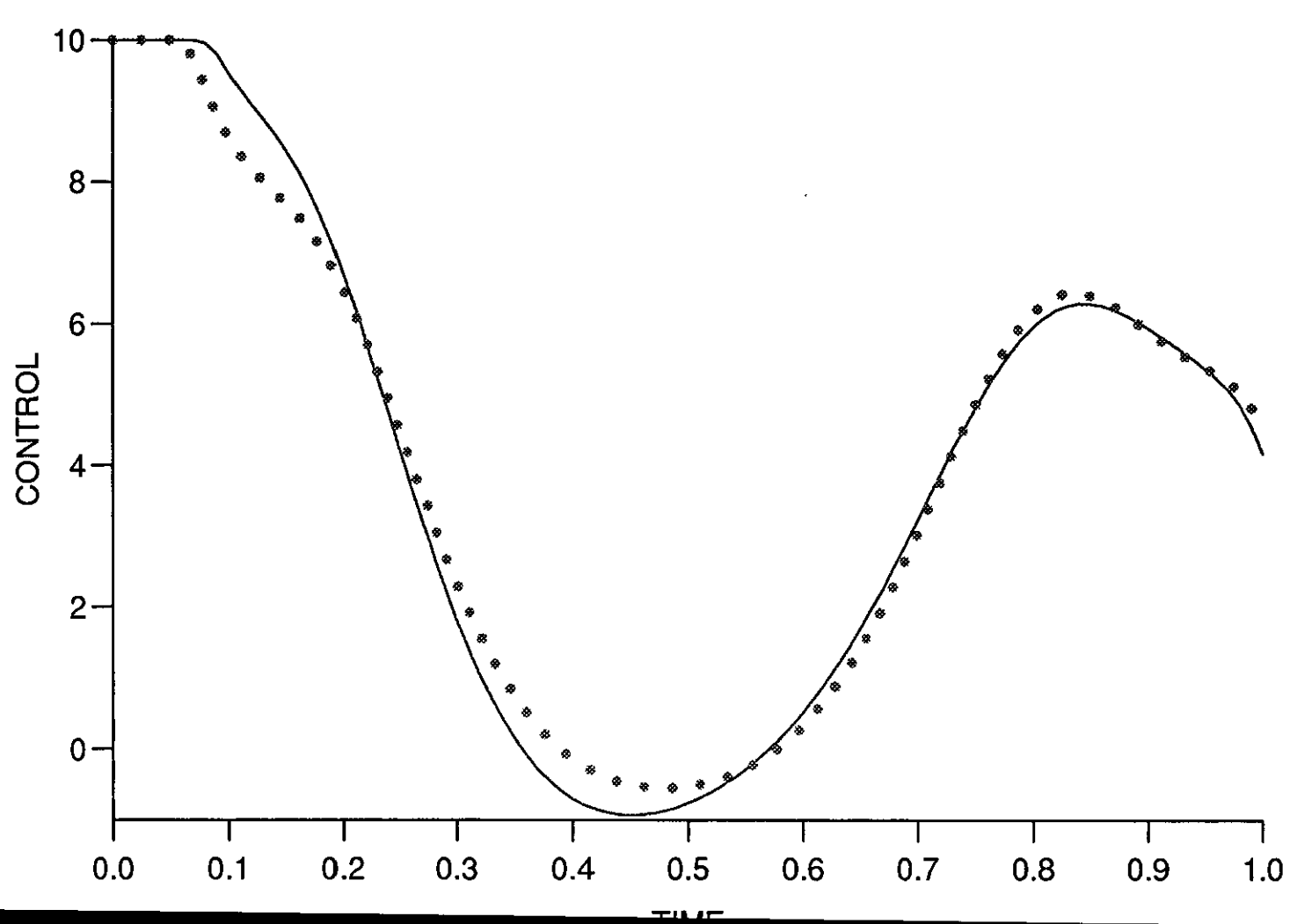


Figure (9.3.18)

H1

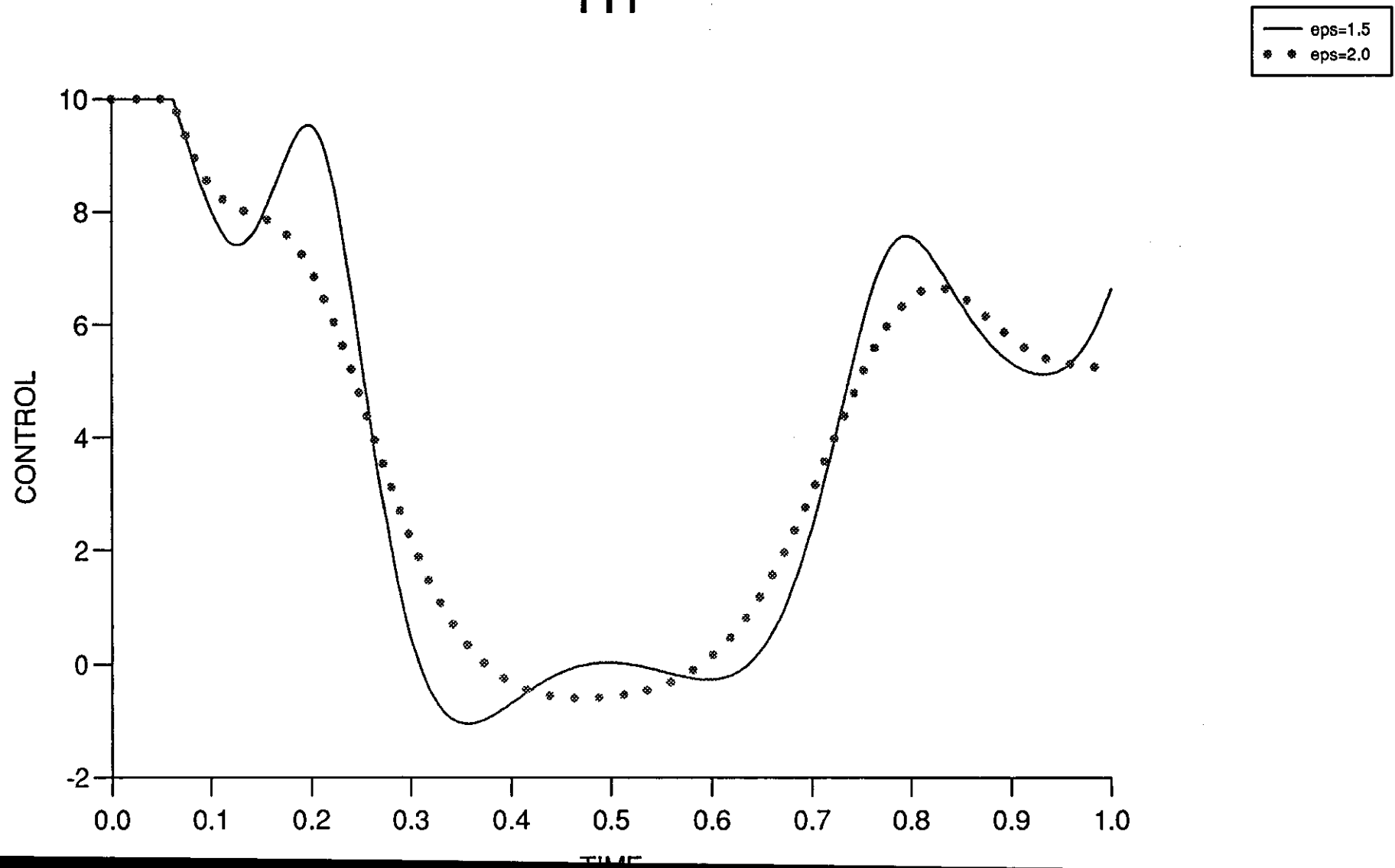


Figure (9.3.19)

H1

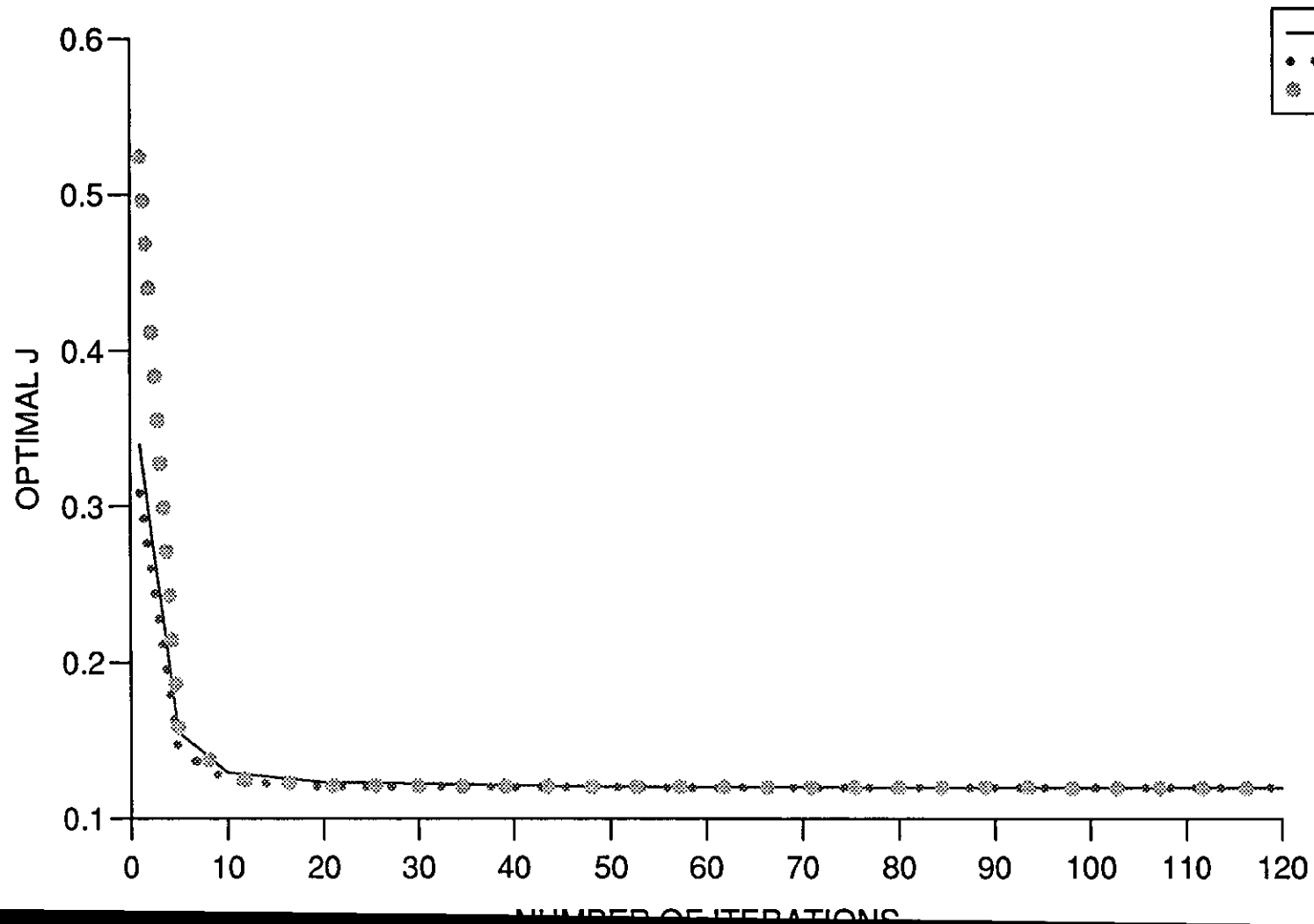


Figure (9.3.20)

ATH

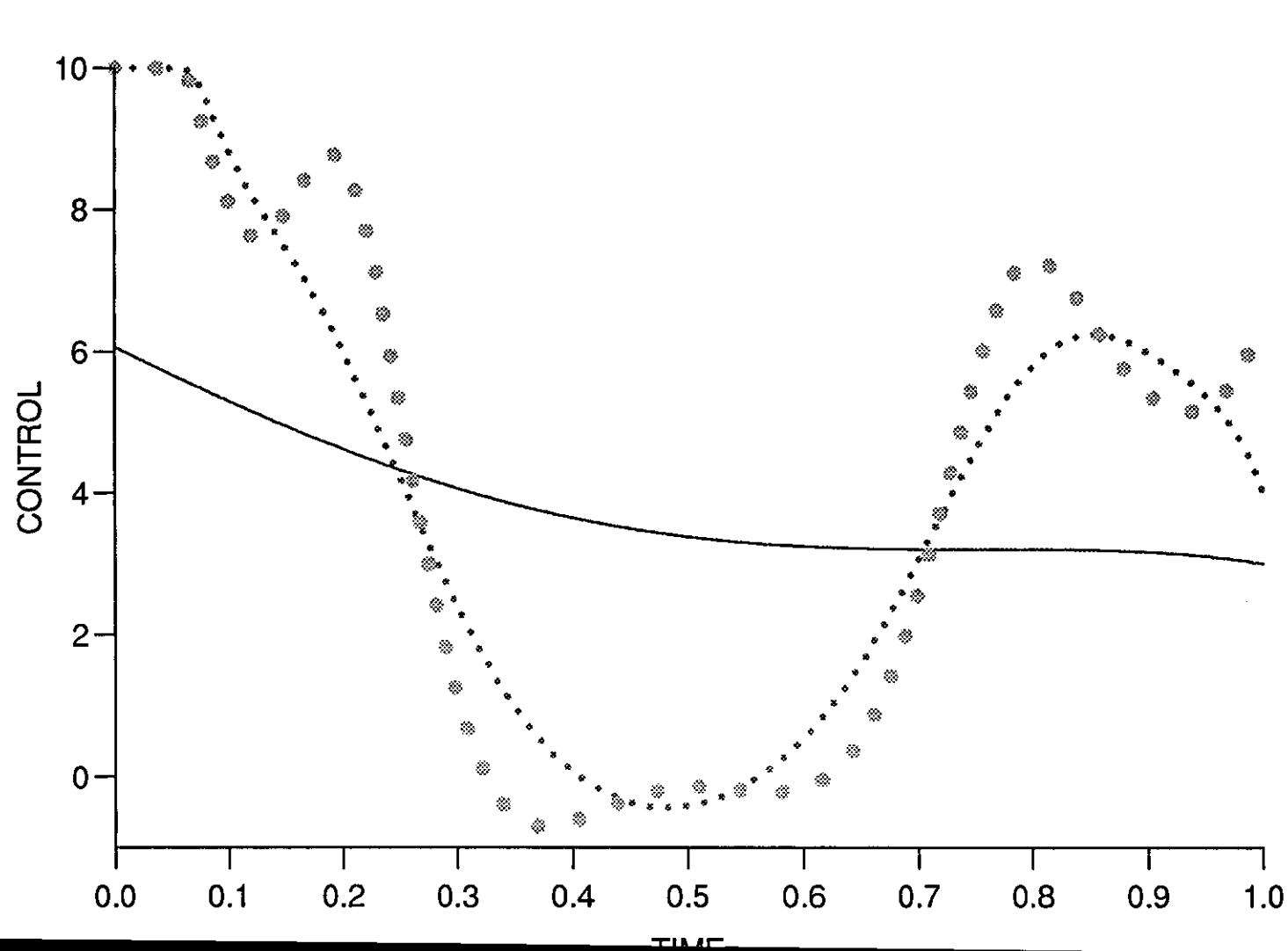


Figure (9.3.21)

ATH

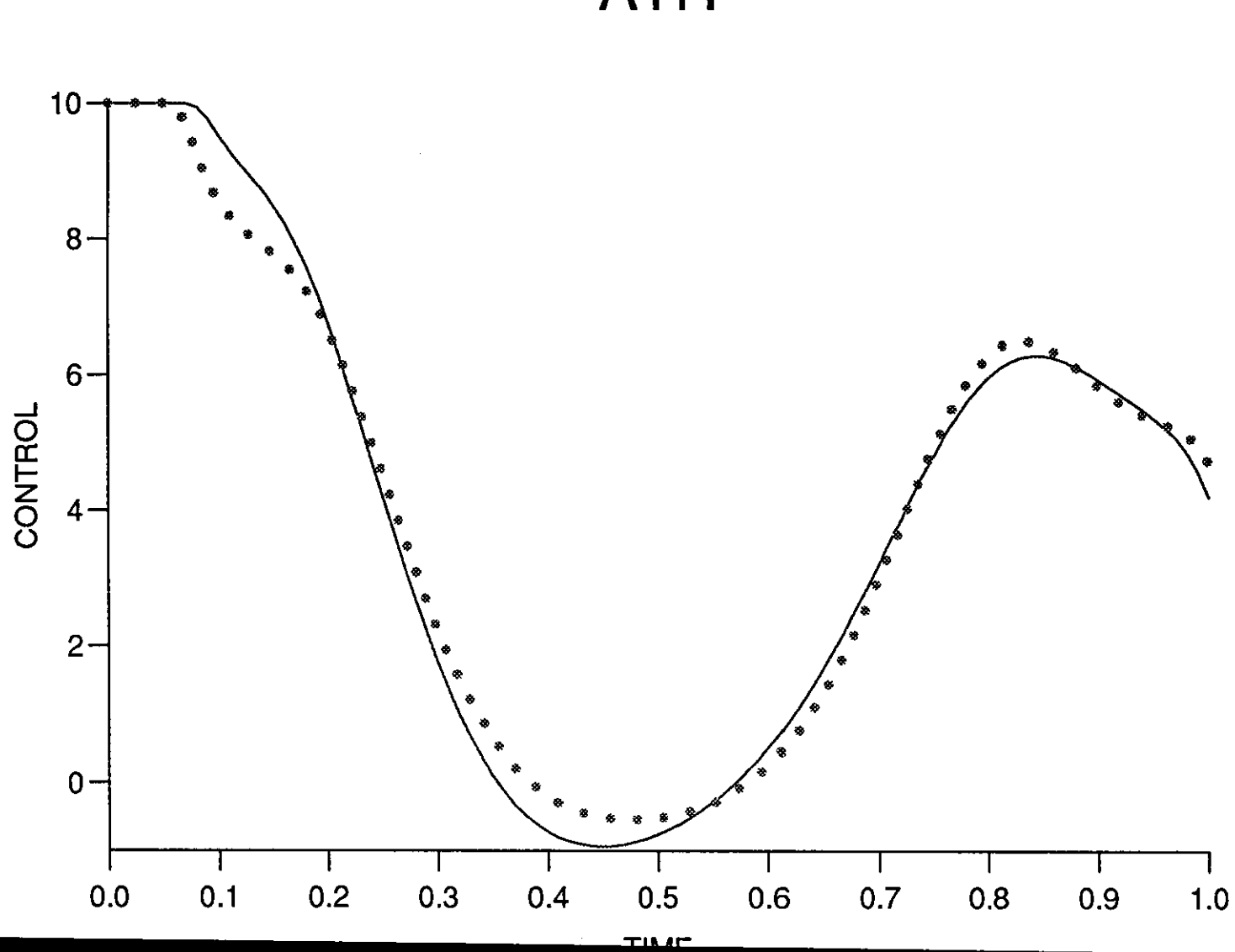


Figure (9.3.22)

ATH

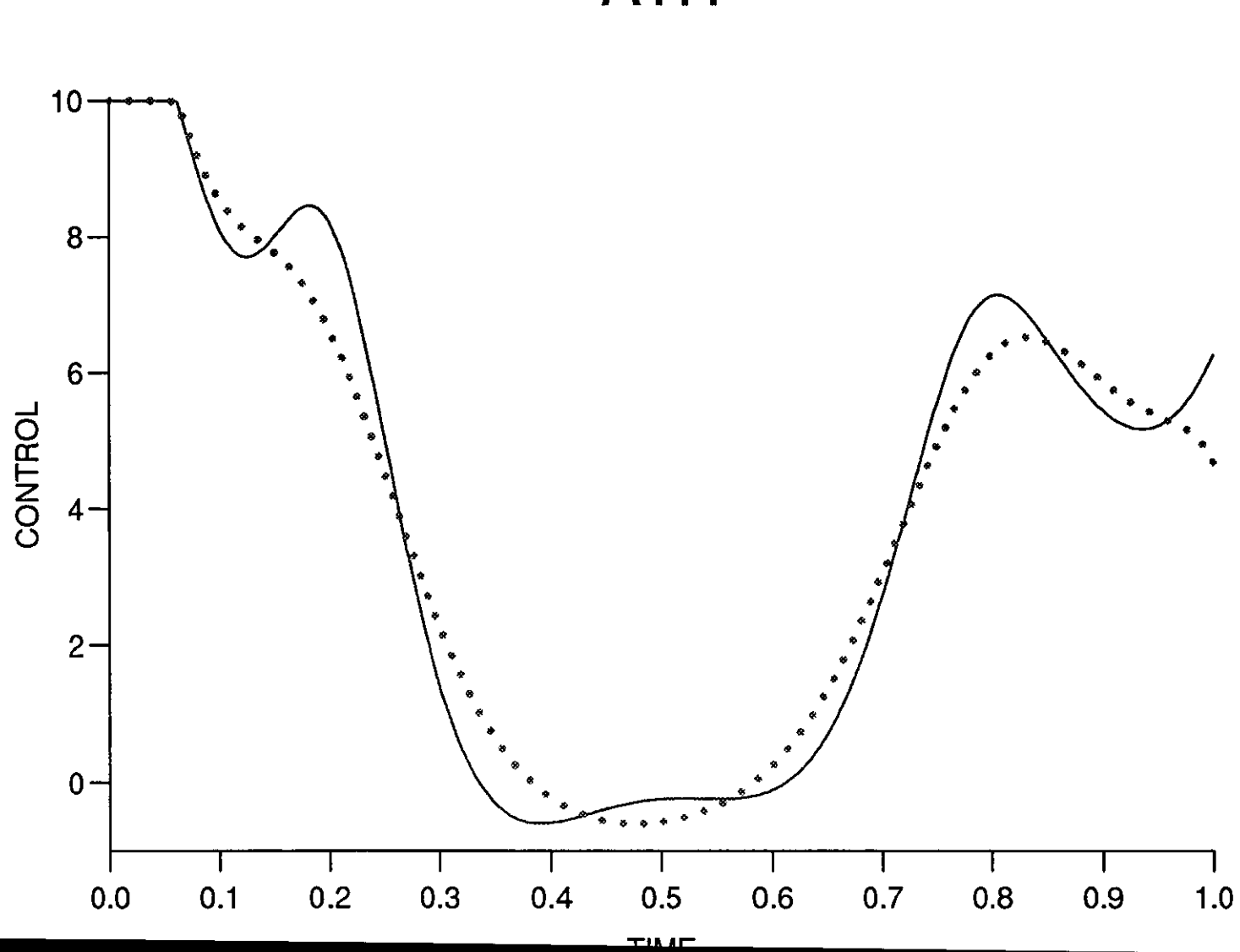


Figure (9.3.23)

ATH

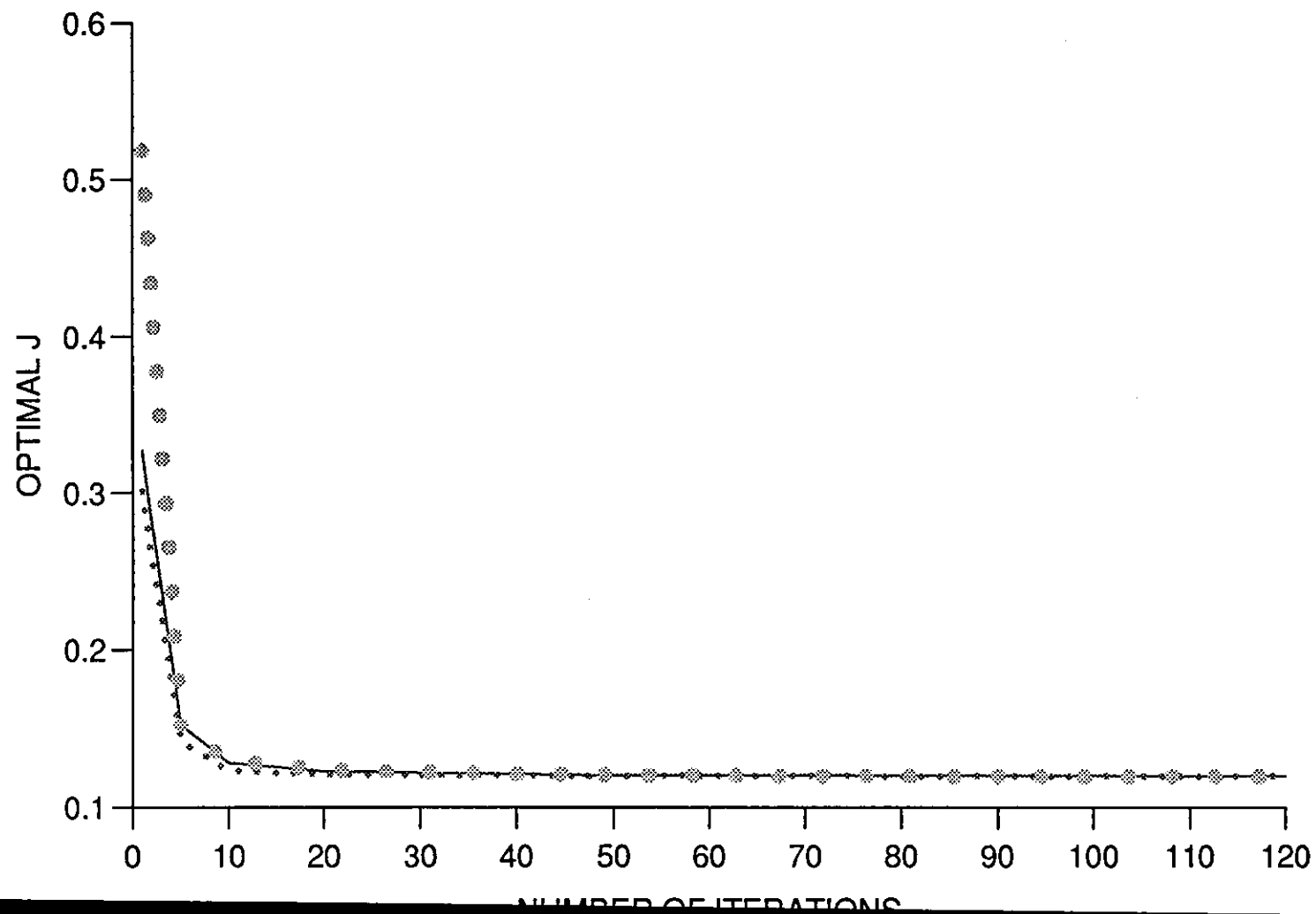


Figure (9.3.24)

H3

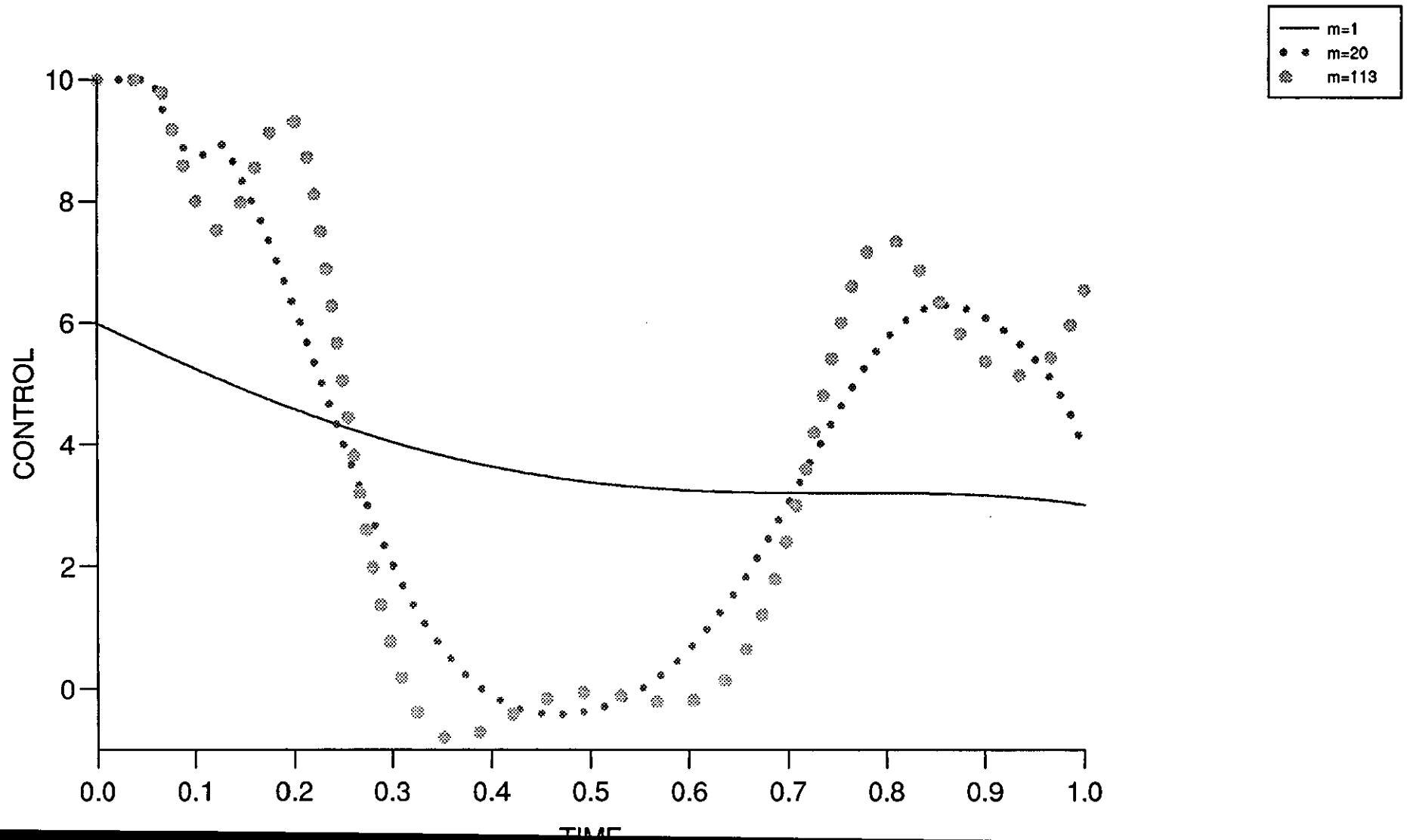


Figure (9.3.25)

H3

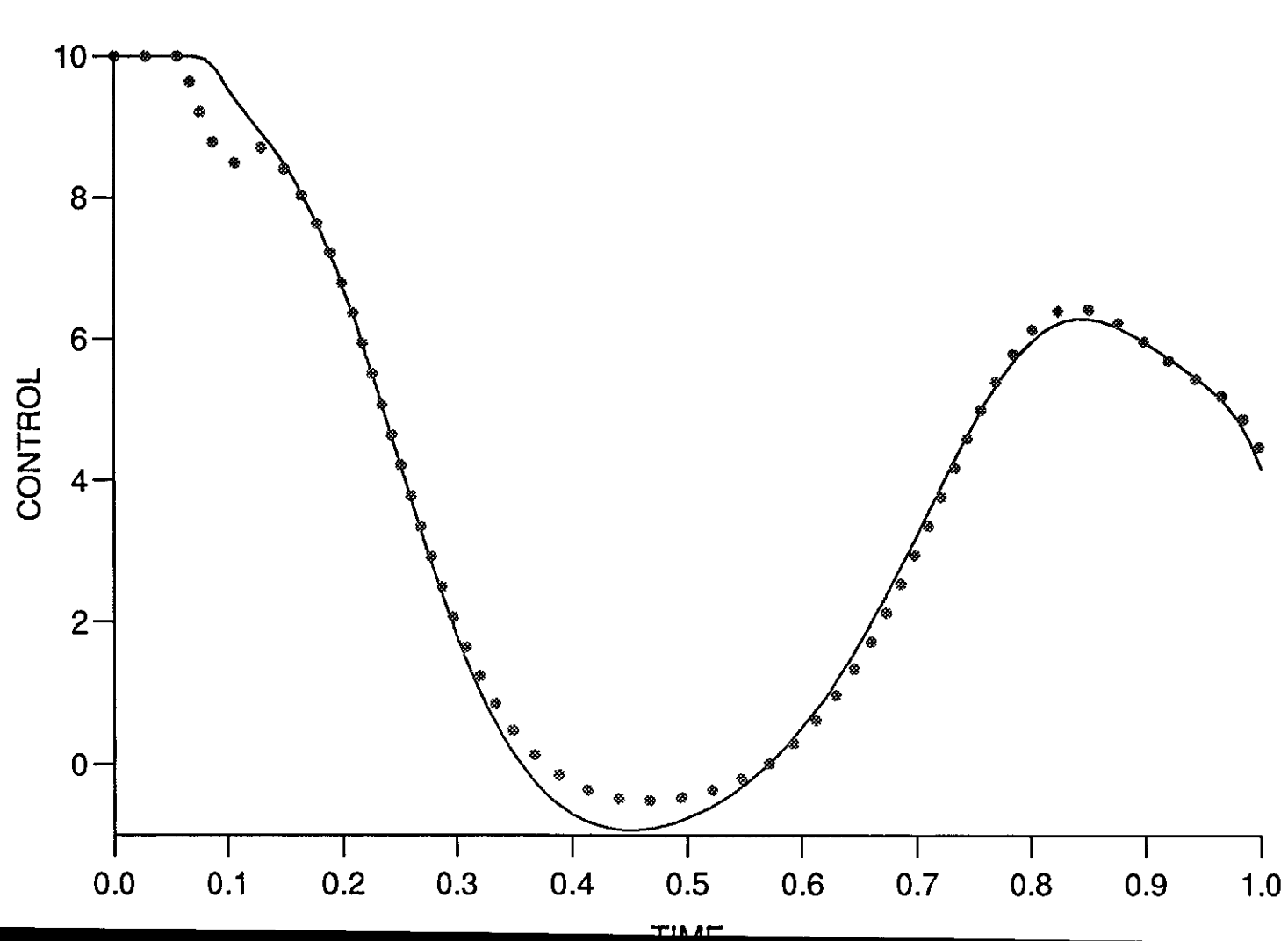


Figure (9.3.26)

H3

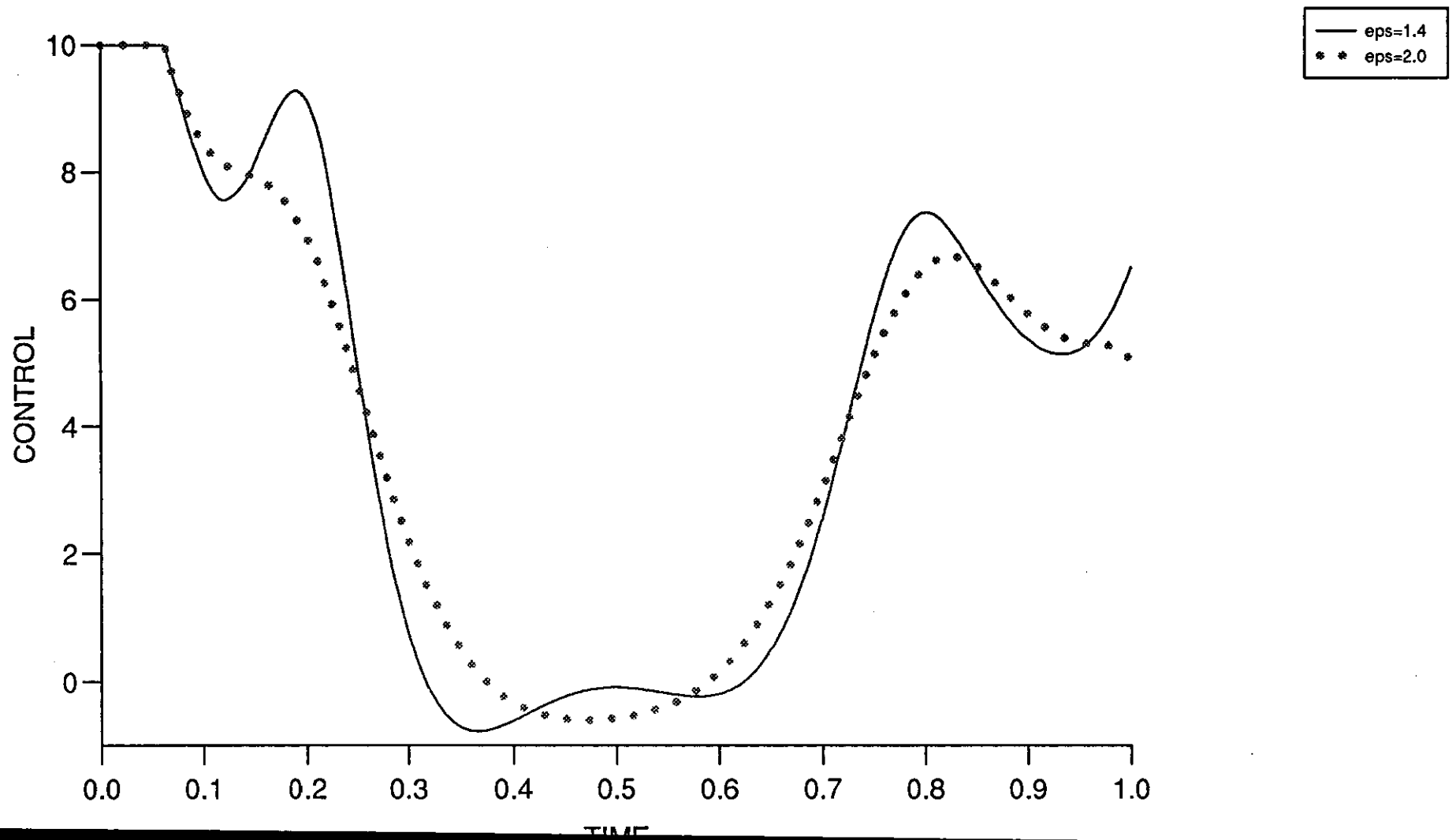


Figure (9.3.27)

H3

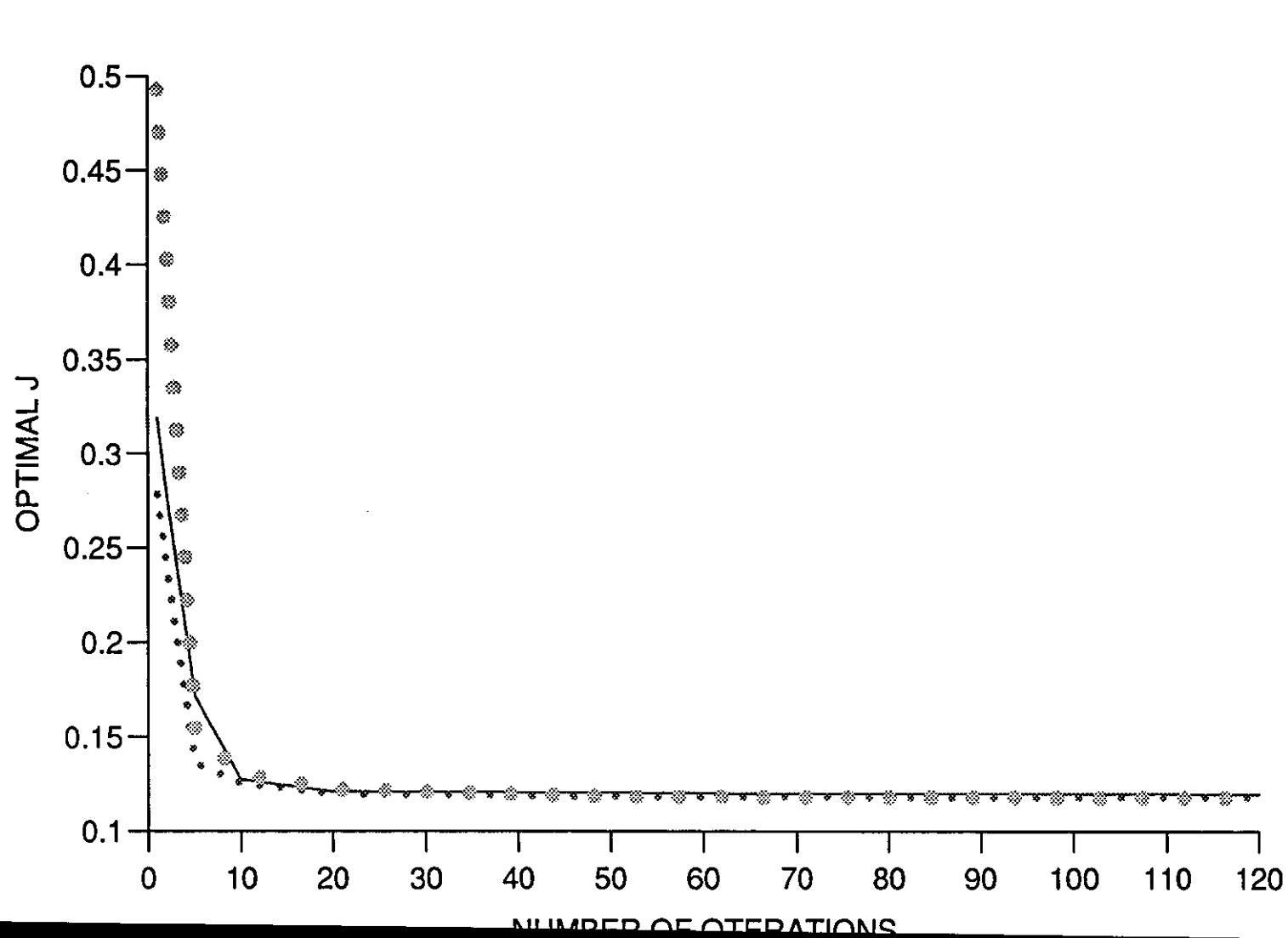


Figure (9.3.28)

COMPARISON OF THE SEVEN METHODS

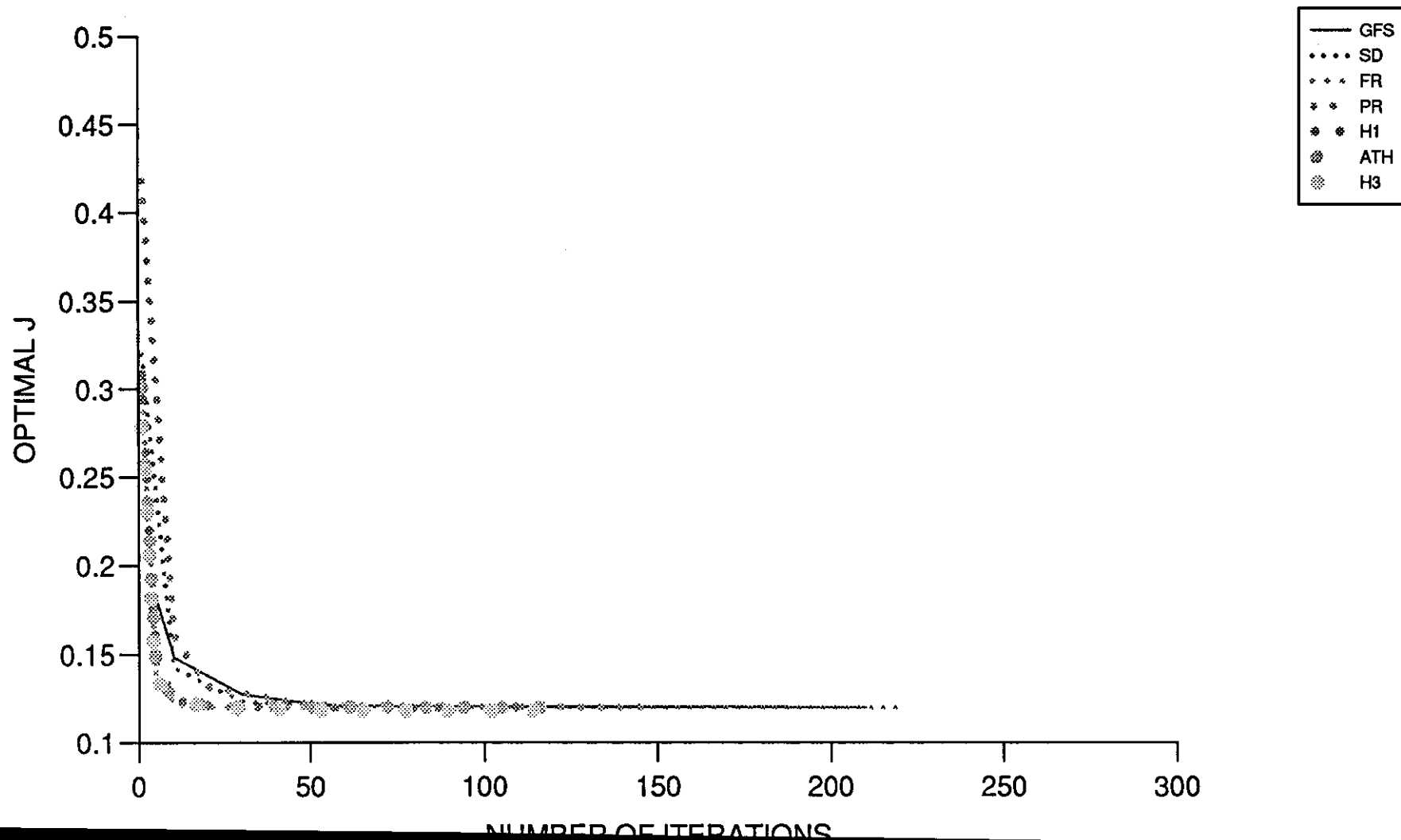


Figure (9.4.1)

Chapter 10

CONCLUSION

In this thesis, we have investigated the performance of a number of Hybrid conjugate gradient methods for solving infinite dimensional, i.e., dynamic optimisation problems and compared them with some existing gradient and conjugate gradient techniques.

The superiority of a method over the others could be decided by taking into consideration the following factors:

- i. Ease of Programming,
- ii. computing time,
- iii. Convergence,
- iv. number of iterations,
- v. number of function evaluations,
- vi. numerical stability,
- vii. versatility.

In Problem 1, the best minimum value of $x_2(t_f)$ was found by FR, H1, ATH and H3. They also took the same number of iterations, amount of computing time and number of function evaluations to achieve this optimum. When a distant initial control from the one that found the optimal $x_2(t_f)$ was selected, FR achieved a better value for $x_2(t_f)$ than the other methods.

Although GFS took more iterations to obtain $x_2(t_f)$, than the other methods, it took fewer function evaluations and also less computing time to achieve this. In this problem for all the methods, although taking N as 400 as opposed to 100 could produce slightly better minimum J for some ϵ 's, but taking into consideration the computing time, it is more economical to select N as 100 rather than 400. Here, concerning problem 1 as a test problem,

due to the simple nature of the problem, the results for some methods were so similar to each other that we could not establish a firm superiority of one technique over the other.

In problem 2, the best value for the Drag coefficient (CD) was found by FR and GFS in the sense of being closest to the analytical minimum value. FR took 50% fewer iterations than GFS to achieve this, but GFS took fewer function evaluations than FR. When a distant initial control was selected all the Hybrid methods, FR and PR performed fairly similarly and were superior to GFS and SD in terms of number of iterations taken to achieve the optimal CD up to 5 decimal places. Here taking N larger, say 1600, produced better minimum J^* for most methods but at the cost of more computing time, therefore the best possible N that could have been selected was $N = 800$. This particular problem showed that SD took a greater number of iterations and consequently much more computing time than all of the other methods.

In problem 3, the best minimum value of J^* , when the coefficient of penalty function (k) was taken as 10, was obtained by H1 and H3. They took the same number of iterations, amount of computing time and number of function evaluations to achieve this. But the advantage of GFS, SD and ATH over the above two methods was that they could achieve a reasonable value for J for distant and different initial controls, where the other two were not successful.

Also ATH and PR are sensitive with respect to the selection of ϵ , even with N sufficiently large very special care should be taken in choosing ϵ , otherwise numerical instability occurs. In terms of stability, the performance of FR was poor, since regardless of N and ϵ after 60, or so, iterations we could get no more improvement in the value of J^* even by increasing the number of iterations or taking larger values for N . For GFS numerical instability happened with ϵ relatively large, say 0.009, regardless of what N was selected. For SD numerical instability happened for $N = 10, 100$ and 200 with ϵ relatively large say 0.009. For PR selecting the proper ϵ for each N was crucial in avoiding numerical instability, and due to its sensitivity special care should have been taken in selecting ϵ . For H_1 numerical instability occurred for $\epsilon \geq 0.006$ and for $N = 50$ numerical instability happened for ϵ in the range $0.003 \leq \epsilon \leq 0.006$. For ATH, selecting N small, say 10, resulted in numerical instability. For $N = 50$, numerical instability occurred with $\epsilon \geq 0.006$ and for $N = 100$ and 200 numerical instability occurred with $\epsilon \geq 0.007$. For H3, when N was taken as 10 numerical instability occurred with $\epsilon \geq 0.006$, for $N = 50$ numerical instability occurred with ϵ in the range $0.003 \leq \epsilon \leq 0.006$ and also $\epsilon = 0.008$. This showed the great sensitivity of H_3 with regard to the selection of proper ϵ . For $N = 100$ and 200 numerical instability occurred for ϵ relatively large say 0.008. In this problem although for all methods taking N as 200 as opposed to 100 could produce slightly better minimum J^* for some ϵ 's, but taking into consideration the computing time taken it is more economical to select N as 100 rather than 200.

In Problem 4, the best value of J^* , when the coefficient of the penalty function was taken as $k = 500$, was found by SD, FR, PR and all three Hybrid

conjugate gradient methods. They all took the same number of iterations, amount of computing time and number of function evaluations to obtain this value. In fact, all the methods performed similarly except GFS, which had two advantages over the other methods. Firstly, even with selecting a distant starting control it achieved, $\text{Acc} \leq 10^{-6}$, whereas with the other techniques this was not possible. Secondly although GFS took more iteration to obtain the optimal J , but took fewer function evaluations compared to the others. Here also taking N as 200 as opposed to 100 produced either slightly better or the same minimum J^* for some ϵ 's, but at the cost of more computing time, therefore the best possible N that could have been selected was $N = 100$. Due to the similarity of the results for different optimization techniques in this particular test problem, we could not establish a definite superiority of a method over the others.

In Problem 5, two distinct local minima were present and with most methods the first minima was found by selecting u_0 in the range $1.0 \leq u_0 \leq 1.8$ and the second with $2.0 \leq u_0 \leq 3.0$. The best value of J^* for the first minima was obtained by FR, H1, ATH and H3. They took the same number of iterations, amount of computing time and number of function evaluations to achieve the minima. The best value of J^* for the second minimum was obtained by FR which took fewer iterations and function evaluations than the other methods to achieve this value. Here for all of the methods considering the first minimum taking N too small resulted in numerical instability. For the second minimum taking N too small resulted in complete numerical instability for SD. Also for PR, FR, H1, ATH and H3, unless ϵ was selected very carefully numerical instability occurred. The only method that even by selecting N too small could produce consistent results was GFS. When distant initial controls selected for the first minima all the techniques could achieve a required high accuracy, but for the second minima, the performance of GFS was very poor and FR was poor, H3, H1 and PR were better, ATH was good and the best was SD. Here taking N as 156 as opposed to 78 produced slightly better minimum J^* for some methods, but at the cost of more computing time.

In problem 6, the best minimum value of J^* up to 8 decimal places was found by H1. But taking into consideration the number of iterations, the number of function evaluations and the computing time, the best value of J^* up to 4 decimal places was obtained by H3. When a distant initial control is selected, the performances of GFS and SD were poor compared to the other techniques. Here also the number of integration steps, should be selected large enough for all the methods, otherwise poor convergence results.

When N was selected as 800 as opposed to 400 the same or slightly better minimum J^* was obtained but at the cost of more computing time.

Now in view of the performance of the Optimization techniques used in this work for the six practical problems, one can see that for each problem, there were one or more techniques that performed better than the others, but overall there was no technique that could claim complete superiority in terms of the seven criteria mentioned earlier in this chapter. In terms of ease of programming, the Gradient method was relatively easy to program and it

required the least storage compared with the other methods.

Steepest descent involved substantially more programming than GFS because of the linear search required as part of the method. For the conjugate and Hybrid conjugate gradient methods, further numerical techniques were necessary to estimate the norm of each gradient trajectory.

For most problems although it took more iterations for GFS to obtain optimality, but due to lack of complexity it took fewer function evaluations and therefore less computing time than other methods. But for more accurate optimal results considering the number of iterations taken FR, H1 and H3 produced better results. When problems were experienced through selecting a remote initial control in most cases, the performance of FR and ATH in achieving optimality was better than other techniques. Concerning the approximation to u^* with comparison to the analytical solutions, we could see that almost for all the problems, the hybrid conjugate gradient methods produced better results than the conventional gradient or conjugate gradient techniques.

Here, for the problems that used penalty functions, it is worth mentioning that, when the coefficient of the penalty function (k), was changed, in effect a different problem was solved, and it was shown analytically for problem 4 that when $k \rightarrow \infty$ only then was the solution using a penalty function equivalent to the correct solution. Overall, the tests provided valuable information on the convergence characteristics of various conjugate and hybrid conjugate gradient algorithms and should serve as a practical guide to the potential user wishing to implement these methods.

Finally, in this work all of the methods used for all the problems, produced consistent results, providing sufficiently large values were taken for the number of integration steps, N , along with appropriate values for ϵ and the initial control u_0 .

Bibliography

- [1] Bliss, G.A. Lectures on the calculus of Variations. Univ. Chicago Press, Chicago, 1946.
- [2] Bryson, A.E., and Ho, Y.C. Applied Optimal Control. Blaisdell, Waltham, Mass, 1969.
- [3] Robbins, H.M. A generalised Legendre-Clebsch condition for the singular cases of Optimal Control. IBM J. Res. Develop., 11, 361-372, 1967.
- [4] Kelley, H.J. A second variation test for singular extremals. ALAA J. 2, 1380-1382, 1964.
- [5] Tait, K.S. Singular problems in optimal control. Ph.D, dissertation, Harvard Univ. Cambridge, Mass, 1965.
- [6] Kopp, R.E., and Moyer, H.G. Necessary conditions for singular extremals. AIAA J. 3, 1439-1444, 1965.
- [7] Kelley, H.J., Kopp, R.E., and Moyer, H.G. Singular extremals, in "Topics in Optimization". G. Leitmann, ed., 63-101. Academic Press, New York, 1967.
- [8] Robbins, H.M. A generalised Legendre-Clebsch condition for the singular cases of optimal control, IBM, J1 Res, Dev, 3, 361-372, 1967.
- [9] Goh, B.S. Necessary conditions for singular extremals involving multiple variables. SIAM J, Control 4, 716-731, 1966.
- [10] Jacobson, D.H. Sufficient conditions for non-negativity of the second variation in singular and non-singular control problems, SIAM, J, Control, 8, 403-423, 1970.
- [11] Mayne, D.Q. Differential dynamic programming. A unified approach to the optimization of dynamic systems, in "Advances in control systems". (C.T. Leondes, ed) Vol.10, 179-254, Academic Press, Newyork and London, 1973.
- [12] Jacobson, D.H. On conditions of optimality for singular control problems. IEEE Trans. Autom. Control, AC-15, 109-110, 1970.
- [13] Gelfand, I.M., and Fomin, S.V. Calculus of variations. Prentice Hall, Englewood Cliffs, N.J, 1963.

- [14] Johansen, D.E. Convergence properties of the method of gradients, in "Advances in control systems", (C.T. Leondes, ed) Vol.4, 279-316, Academic Press, New York and London, 1966.
- [15] Jacobson, D.H. A general sufficiency theorem for the second variation, *J.Math. Analysis Applic*, 34, 578-589, 1971.
- [16] Jacobson, D.H., and Speyer, J.L. Necessary and sufficient conditions for optimality for singular control problems: A limit approach. *J. Math. Analysis Applic*, 34, 239-266, 1971.
- [17] Abdul Wahab Jusoh. Reduction of continuous control problems to parametric representation. Mathematics Department, Loughborough University of Technology, 1979.
- [18] Touati-Ahmed, D., and Storey, C. Efficient conjugate gradient techniques. Department of Mathematics, Loughborough University of Technology, 1987.
- [19] Carg, S.C. Numerical minimization methods for functionals: Comparison and extensions. Institute for Aerospace studies, University of Toronto. UTIAS Report No. 209, CN ISSN 0082-5255, 1977.
- [20] Edge, E.R, and Powers, W.F. Function-space Quasi-Newton algorithm for optimal control problems with bounded control and singular arcs. *Journal of optimization theory and applications*. vol.20, No.4, 1976.
- [21] Luss, R., and Galli, M. Multiplicity of solutions in solving dynamic programming for optimal control. Department of Chemical Engineering, University of Toronto, Toronto, Ontario, M5S 1A4, Canada, presented at the second North America/German workshop on Chemical Engineering, Mathematics and Computation, Göttingen, West Germany. July 1990.
- [22] Luss, R. Optimal control by dynamic programming using systematic reduction in grid size. *Int. J. control*, vol, 51, No.5, 995-1013, 1990.
- [23] Hancock, H. Theory of maxima and minima. Reprinted by Dover Publications, New York, 1960.
- [24] Cooper, L., and Steinberg, D. Introduction to methods of optimization. W.B. Saunders & Co., Philadelphia, 1970.
- [25] Wilde, D.J., and Beightler, C.S. Foundations of optimization. Prentice Hall, 1967.
- [26] Fox, R.L. Optimization methods for the engineering design. Addison-Wesley Publishing Co., 1971.
- [27] Kowlik, J., and Osborne, M.R. Methods for unconstrained optimization problems. American Elsevier Publishing Co., New York, 1968.
- [28] Beltrumi, E.J. An algorithmic approach to nonlinear analysis and optimization. Academic Press. 1970.

- [29] Bryson, A.E., and Ho, Y.C. Applied optimal control. Blaisdell Publishing Co, 1969, Chapter 1, 1969.
- [30] Dantzig, G.B. Linear programming and extensions. Princeton University Press, Princeton, N.J., 1963.
- [31] Box, M.J., Davies, D., and Swann, W.H. Non linear optimization techniques. I.C.I. Monograph No.5, Oliver and Boyd Ltd, Edinburgh, 1969.
- [32] Himmelblau, D.M. Applied non linear programming. McGraw-Hill, Inc., 1972.
- [33] Hadley, G.H. Non linear and dynamic programming. Addison-Wesley Publishing Co., Reading, Mass., 1964.
- [34] Fiacco, A.V., and McCormick, G.P. Non linear programming. John Wiley and Sons, New York. 1968.
- [35] Greenberg, H. Integer programming, Academic Press, 1971.
- [36] Zoutendisk, G. Mathematical programming methods. North-Holland Publishing Co., Amsterdam, 1976.
- [37] Ostrowski, A.M. Solution of equations and systems of equations. 2nd Edition. Academic Press, New York, 1966.
- [38] Wolfe, P. Convergence conditions for Ascent methods. S.I.A.M. Review, 11, 226-235, 1969.
- [39] Sargeant, R.W.H., and Sebastian, D.J. On the convergence of sequential minimization algorithms. J.O.T.A., 12, 567-575, 1973.
- [40] Polak, E., Sargeant, R.W.H., and Sebastian, D.J. On the convergence of sequential minimization algorithms. J.O.T.A., 14, 439-442, 1974.
- [41] Zoutendijk, G. Non linear programming, Computational methods in Integer and non linear programming, Abadie, J, Ed. North Holland Publishing Co., Amsterdam, 1970.
- [42] Fletcher, R. Practical methods of optimization. vol.1. Unconstrained optimization, John Wiley and Sons, 1980.
- [43] Gill, P.E., Murray, W., and Wright, M.H. Practical optimization. Academic Press, London, 1981.
- [44] Polak, E. Computational methods in optimization. A unified approach. Academic Press, New York, 1971.
- [45] Wolfe, M.A. Numerical methods for unconstrained optimization. Van Nastran Reinholds Company, 1978.
- [46] Cauchy, A. Methode Generate pour la Resolution des systemes d'Equations simultanees. C.R. Acad. Sci., 27, 536-538, 1987.

- [47] Hestenes, M., and Stiefel, E. Methods of conjugate Gradients for solving linear systems. *Journal of Research of the National Bureau of Standards.*, 29, 409-439, 1952.
- [48] Fletcher, R, and Reeves, C.M. Function minimization by conjugate gradients. *The computer Journal*, vol.7, 149-153, 1964.
- [49] Daniel, J.W. The conjugate gradient method for linear and non linear operator equations. *SIAM Journal on Numerical Analysis*, vol.4, 10-26.
- [50] Sorenson, H.W. Comparison of some conjugate direction procedures for function minimization. *Journal of The Franklin*, vol.288, 421-441, 1969.
- [51] Polak, E. and Ribière, G. Note sur La Convergence des Methodes de Directions Conjuguees, *Revue Francaise Infor. Rech. Oper.*, 16, R1, 35-43, 1969.
- [52] Dixon, L.C.W. Conjugate gradient algorithms, Quadratic termination without linear searches. *The Hatfield Polytechnic, Numerical optimization Centre T.R.38*, 1972.
- [53] Liu, Y., and Storey, C. Efficient generalized conjugate gradient algorithms, PartI: Theory, *Journal of Optimization Theory and Applications*, vol.69, 129-137, 1991.
- [54] Khoda, K.M., Liu, Y., and Storey, C. Generalized Polak-Ribière Algorithm. *Journal of Optimization Theory and Applications*, vol. 75, No.2, November 1992.
- [55] Powell, M.J.D. Restart procedures for the conjugate gradient method. *Mathematical Programming*, vol.12, 241-254, 1977.
- [56] Hu, Y.F., and Storey, C. On unconstrained gradient optimization methods and their interrelationships, *Mathematics Report Number A129*, Department of Mathematical Sciences, Loughborough University of Technology, England, 1990.
- [57] Powell, M.J.D. Nonconvex minimization calculations and the conjugate gradient method. *Lecture notes in Mathematics*, vol.1066, Springer-Verlag, Berlin, 1984.
- [58] Al-Baali, M. Descent property and global convergence of the Fletcher-Reeves method with inexact line search. *IMA Journal of Numerical Analysis*, vol.5, 121-124, 1985.
- [59] Goldstein, A.A. On steepest descent, *SIAM Journal on Control*, vol.3, 147-151, 1965.
- [60] Powell, M.J.D. Some global convergence properties of a variable metric algorithm for minimization without exact line searches. *SIAM-AMS Proceedings*, vol.IX, eds. Cottle, R.W., and Lemke, C.E. SIAM Publications, Philadelphia, 1976.

- [61] Touati-Ahmed, D., and Storey, C. Efficient Hybrid conjugate gradient techniques. *Journal of optimization Theory and Applications*. vol.64, 379-397, 1990.
- [62] Gilbert, J.C., and Nocedal, J. Global convergence properties of conjugate gradient methods for optimization. *Rapport de Recherche No. 1268*, Institute National de Recherche en Informatique et en Automatique, France, 1990.
- [63] Hu, Y.F., and Storey, C. A global convergence result for conjugate gradient methods. *Journal of Optimization Theory and Applications*. vol. 71, 399-405, 1991.
- [64] Beale, E.M.L. A derivation of conjugate gradients. *Numerical Methods for non linear optimization*, ed., Lootsma, F.A., Academic Press, London, 1972.
- [65] Nazareth, J.L. A conjugate direction algorithm without line searches. *Journal of Optimization Theory and Applications*, vol.23, 373-387, 1977.
- [66] Dahlquist, G., and Bjorck, A. *Numerical methods*. Englewood Cliffs, N.J: Prentice-Hall, 1974.
- [67] Touati-Ahmed, D. Ph.D Thesis in A study of Hybrid conjugate gradient methods. Mathematical Sciences Department. Loughborough University of Technology, 1989.
- [68] Hestenes, M. *Conjugate direction methods in optimization*. Springer-Verlag, 1980.
- [69] Crowder, H., and Wolfe, P. Linear convergence of the conjugate gradient method. *I.B.M. Journal of Research and Development*, 16, 431-433, 1971.
- [70] Powell, M.J.D. Nonconvex minimization calculations and the conjugate gradient method. Report No. DAMPT 1983/NA14, Department of Applied Mathematics and Physics. University of Cambridge, England, 1983.
- [71] Powell, M.J.D. Convergence properties of algorithms for nonlinear optimization. Report No. DAMPT 1985/NA1. Department of Applied Mathematics and Theoretical Physics. University of Cambridge, England, 1985.
- [72] Touati-Ahmed, D., and Storey, C. Globally convergent Hybrid conjugate gradient methods. Department of Mathematical Sciences. Loughborough University of Technology. Research No 196, November 1986.
- [73] Shanno, D.F. Globally convergent conjugate gradient algorithms. *Math. Prog.*, 33, 61-67, 1985.
- [74] Zadeh, L.A., Neustadt, L.W., and Blakrishnan, A.V. Computing methods in optimization problems-2. *Proc. SIAM, Conf. at San Remo, Italy*, Sept 1968, 9-13. Academic Press, New York, 1969.

- [75] Bliss, G.A. The evaluation of problems of the calculus of variations. *American Math. Monthly*, V 43-1936, 598-609, 1936.
- [76] Bliss, G.A. Contributions to the calculus of variations, Department of Maths. University of Chicago, 1933-1946 (4 volumes).
- [76]* Pontryagin, L.S., Boltyanskii, V.G., Gamkrelidze, R.V., and Mischenko, E.F. The Mathematical theory of optimal processes. D.F. Brown, Translate Pergamon Press-Macmillan, New York, 1964.
- [77] McShane, E.J. On multiples for Lagrange problems. *American J. Math.*, v61, 809-819, 1939.
- [78] Bellman, R. Dynamic Programming, Princeton University Press, 1957.
- [79] Dreyfus, S.E. Dynamic programming and the calculus of variations. Academic Press, New York, 1965.
- [80] Hestenes, M.R. Calculus of variations and optimal control theory. John Wiley and Sons, New York, 1966.
- [81] Kopp, R.E., and McGill, R. Several trajectory optimization techniques in computing methods in optimization problems. A.V. Balakrishnan and L.W. Neustadt. Eds. New York Academic, 65-89, 1964.
- [82] Kelley, H.J. Methods of gradients, in optimal techniques. G. Leitmann, Ed. New York: Academic. Ch 6, 1962.
- [83] Denham, W.F., and Bryson, A.E. Optimal programming problems with inequality constraints. II. Solution by Steepest Ascent. *AIAAJ.*, vol.2. 25-34 January 1964.
- [84] McGill, R. Optimal control, inequality, state constraints and the generalized Newton-Raphson algorithm. *J. SIAM on Control*, vol.3, No.2, 291-298, 1965.
- [85] Fletcher, R., and Powell, M.J.D. A rapidly convergent descent method for minimization. *British Computer J.* 163-168, June 1963.
- [86] Kelley, H.J. Gradient theory of optimal flight path. *AM. Rocket Soc. J.* vol.30, 947-953, October 1960.
- [87] McReynolds, S.R., and Bryson Jr., A.E. A successive sweep method for solving optimal programming problems. *JACC*, 1965.
- [88] Mitter, S. Successive approximation method for the solution of optimal control problems. *Automation*, vol.3. 133-149. 1966.
- [89] Box, M.J. A comparison of several current optimization methods and the use of transformations in constrained problems. *The computer J.*, vol.9. 67-78, May 1966.
- [90] Lasdon, L.S., Mitter, S.K., and Waren, A.D. The conjugate gradient method for the optimal control problems. *IEEE Transactions on automatic control*, vol.AC-12, No.2, 132-138, April 1967.

- [91] Rice, J.R. The approximation of functions. Reading Mass Addison-Wesley, Vol 1, 1964.
- [92] Curry, H.B. The method of steepest descent for nonlinear problems. Quart. Appl. Math., Vol 2, 258-261, October 1988.
- [93] Runge, C. Über die numerische Auflösung von differential gleichungen, Math. Ann., 46, 167-178, 1895.
- [94] Kutta, W. Beitrag zur näherungsweise integration totaler differential gleichungen. Z. Math. Phys., 46, 535-453, 1901.
- [95] Heun, K. Neue methode zur approximativen integraion de differential gleichungen einer unabhängigen veränderlichen. Z. Math. Physik, 45, 23-38, 1900.
- [96] Henrici, P. Discrete variable methods in ordinary differential equations. John Wiley and Sons, 1962.
- [97] Lambert, J.D. Computational methods in ordinary differential equations. John Wiley and Sons, 1986.
- [98] Butcher, J.C. Coefficients for the study of Runge-Kutta integration processes. J. Austral. Math. Soc., 3, 185-201, 1963.
- [99] Butcher, J.C. On the integration processes of A. Huťa. J. Austral. Math. Soc., 3, 202-206, 1963.
- [100] Butcher, J.C. Implicit Runge-Kutta processes. Math. Comp., 18, 50-64, 1964.
- [101] Butcher, J.C. Integration processes based on Radau quadrature formulas. Math. Comp., 18, 233-243, 1964.
- [102] Butcher, J.C. On Runge-Kutta processes of higher order. J. Austral. Math. Soc., 4, 179-194, 1964.
- [103] Butcher, J.C. On the attainable order of Runge-Kutta methods. Math. Comp., 408-417, 1965.
- [104] Huťa, A. Contribution à la formule de sixième ordre dans la méthode de Runge-Kutta-Nyström. Acta Fac. Rerum Natur. Univ. Comenian. Math., 2, 21-24, (1957).
- [105] Ralston, A. A first course in numerical analysis. McGraw-Hill, 1965.
- [106] Lothin, M. On the accuracy of Runge-Kutta's method. M.T.A.C., 5, 128-132, 1951.
- [107] Ralston, A. Runge-Kutta methods with minimum error bounds. Math. Comp. 16, 431-437, 1962.
- [108] Bieberbach, L. Theorie der differential gleichungen. Springer, 1930.
- [109] Carr, J.W. III. Error bounds for Runge-Kutta single step integration processes. J. Assoc. Comput. Mach., 5, 39-44, 1958.

- [110] Guller, B.A., and Rozenberg, D.P. A generalization of a theorem of Carr on error bounds for Runge-Kutta procedures. *J. Assoc. Comput. Mach.*, 7, 57-60, 1960.
- [111] Richardson, L.F. The deferred approach to the limit, I-single lattice. *Trans. Roy. Soc. London*, 226, 299-349, 1927.
- [112] Kuntzmann, J. Evaluation de l'erreur sur ou pas dans les methodes a pas séparés, chiffres, 2, 97-102, 1959.
- [113] Scraton, R.E. Estimation of the truncation error in Runge-Kutta and allied processes. *Comput. J.*, 7, 286-288, 1964.
- [114] Merson, R.H. An operational method for the study of integration processes. *Proc. Symp. Data processing, Weapon research establishment, Salisbury, S. Australia*. 1957.
- [115] England, R. Error estimates for Runge-Kutta type solutions to systems of ordinary differential equations. *Comput. J.* 12, 166-170, 1969.
- [116] Shintani, H. Approximate computation of errors in numerical integration of ordinary differential equations by one-step methods. *J. Sci. Hiroshima Univ. Ser A-1 Math.*, 29, 97-120, 1965.
- [117] Shintani, H. On a one-step method of order 4. *J.Sci. Hiroshima Univ. Ser. A-1 Math.*, 30, 91-107, 1966.
- [118] Shintani, H. Two-step processes by one-step methods of order 3 and of order 4. *J. Sci. Hiroshima Univ. Ser. A-1 Maths.*, 30, 183-195, 1966.
- [119] Gröbner, W., Hofreiter, N. *Integraltafel*. 2 Vols, Berlin, Springer Verlag, 1961.
- [120] Walsh, G.R. *Methods of optimization*. John Wiley and Sons, 1975.
- [121] Dennis, J.E. and Schanbel, R.B. *Numerical methods for unconstrained optimization and non linear equations*. Prentice Hall inc. Englewood Cliffs, New Jersey, 07632, 1983.
- [122] Goldstein, A.A. Cauchy's method of minimization. *Numer. Math.*, 4, 146-150, 1962.
- [123] Goldstein, A.A. On Newton's methods. *Numer. Math.*, 7, 391-393, 1965.
- [124] Armijo, L. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific, J. Math.*, 16, 1-3, 1966.
- [125] Al-Baali, M. and Fletcher, R. An efficient line search for nonlinear least-squares, *J.O.T.A.*, 48, No 3, 359-377, 1986.
- [126] Lemaréchal, C. A view of line searches. in *optimization and optimal control, lecture notes in control and information science*, 30, eds. Auslander, A., Oettli, W. and Stoer, J.. Springer-Verlag, 1981.

- [127] Fletcher, R. Practical methods of optimization. John Wiley and sons, Chichester 1987.
- [128] Goldstein, A.A. Constructive methods in optimization. Springer Verlag, 1967.
- [129] Walder, T.J. The numerical solution of an optimal temperature problem. M.Sc. thesis, Loughborough University of Technology, Department of Mathematics, March 1969.
- [130] Powell, M.J.D. An efficient method for finding the minimum of a function of several variables without calculating derivatives. The computer Journal, Vol 7, 155-162, July 1964.
- [131] Rao, S.S. (1979). Optimization theory and applications, Wile eastern limited, 1979.
- [132] Bell, D.J. and Jacobson, D.H. Singular optimal control problems. Mathematics in science and engineering, volume 117, Academic Press, London, New York and San Francisco, 1975.
- [133] Horn, F. and Troltenier, U. Über den optimalen temperature lauf im reaktionsrohr, Chem. Eng. Tech., 32, 382-393, 1960.
- [134] Coarant, R. and Hilbert, D. Methods of mathematical physics, Vol I (Interscience), 222-224, 1960.
- [135] Kelley, H.J. Optimization techniques, G. Leitman, Ed, Academic Press, 206-254, 1962.
- [136] Bryson, A.E. and Denham, W.F. A steepest ascent method for solving optimum programming problems, J. Applied Mechanics, Trans. A.S.M.E., Series E, 29, 247-257, 1962.
- [137] McDanell, J.P. and Powers, W.F. Necessary conditions for joining optimal singular and nonsingular subarcs. SIAM J, Control Vol 9, No 2, 161-173, May 1971.
- [138] Bell, D.J. and Boissard, M. Necessary conditions at the junctin of singular and nonsingular control squares, Internat. J. Control. 29, No 6. 981-990, 1979.
- [139] Bell, D.J. The search for a counter example in a class of optimal control problems, II. Second order systems. Internat. J. Control, 34, No 1, 153-158, 1981.
- [140] Bortins, R. On joining non-singular and singular optimal control subarcs. Internat. J. Control., 37, No. 4, 867-872, 1983.
- [141] Bell, D.J. and Zainuddin, Z. Some further evidence for a counter-example in singular optimal control problems. Internat. J. Control, 46, No 4, 1123-1130, 1987.

- [127] Fletcher, R. Practical methods of optimization. John Wiley and sons, Chichester 1987.
- [128] Goldstein, A.A. Constructive methods in optimization. Springer Verlag, 1967.
- [129] Walder, T.J. The numerical solution of an optimal temperature problem. M.Sc. thesis, Loughborough University of Technology, Department of Mathematics, March 1969.
- [130] Powell, M.J.D. An efficient method for finding the minimum of a function of several variables without calculating derivatives. The computer Journal, Vol 7, 155-162, July 1964.
- [131] Ruu, S.S. (1979). Optimization theory and applicaiotns, Wille eastern limited, 1979.
- [132] Bell, D.J. and Jacobson, D.H. Singular optimal control problems. Mathematics in science and engineering, volume 117, Academic Press, London, New York and San Francisco, 1975.
- [133] Horn, F. and Troltenier, U. Über den optimalen temperature lauf im reaktionsrohr, Chem. Eng. Tech., 32, 382-393, 1960.
- [134] Courant, R. and Hilbert, D. Methods of mathematical physics, Vol I (Interscience), 222-224, 1960.
- [135] Kelley, H.J. Optimization techniques, G. Leitman, Ed, Academic Press, 206-254, 1962.
- [136] Bryson, A.E. and Denham, W.F. A steepest ascent method for solving optimum programming problems, J. Applied Mechanics, Trans. A.S.M.E., Series E, 29, 247-257, 1962.
- [137] McDanell, J.P. and Powers, W.F. Necessary conditions for joining optimal singular and nonsingular subarcs. SIAM J, Control Vol 9, No 2, 161-173, May 1971.
- [138] Bell, D.J. and Boissard, M. Necessary conditions at the junctin of singular and nonsingular control squares, Internat. J. Control. 29, No 6. 981-990, 1979.
- [139] Bell, D.J. The search for a counter example in a class of optimal control problems, II. Second order systems. Internat. J. Control, 34, No 1, 153-158, 1981.
- [140] Bortins, R. On joining non-singular and singular optimal control subarcs. Internat. J. Control., 37, No. 4, 867-872, 1983.
- [141] Bell, D.J. and Zainuddin, Z. Some further evidence for a counter-example in singular optimal control problems. Internat. J. Control, 46, No 4, 1123-1130, 1987.

Appendix A1

The complete derivation of (5.1.5) from Chapter 5.

$$\begin{aligned}\dot{x}_2 &= \frac{dx_2}{dt} = \frac{1}{\ell} \left(\frac{u^3 r}{1+u^2} \right) = \frac{u^3 x_1}{1+u^2} \\ x_2(t) &= \int_0^t \frac{u^3 x_1}{1+u^2} dt.\end{aligned}$$

Now from (5.1.4) we have

$$\frac{D}{4\Pi q} = \frac{1}{2} r^2(\ell) + \int_0^\ell \frac{ru^3}{1+u^2} dx.$$

Since $x_1 = \frac{r(x)}{\ell} = \frac{r(\ell t)}{\ell} \Rightarrow r = \ell x_1$
i.e.,

$$\begin{aligned}\frac{D}{4\Pi q} &= \frac{1}{2} \ell^2 x_1^2(1) + \int_0^\ell \frac{ru^3}{1+u^2} dx \\ &= \frac{1}{2} \ell^2 x_1^2(1) + \int_0^1 \frac{u^3 r}{1+u^2} \frac{dx}{dt} dt.\end{aligned}$$

Since $x = \ell t \Rightarrow \frac{dx}{dt} = \ell$
i.e.,

$$\begin{aligned}\frac{D}{4\Pi q} &= \frac{1}{2} \ell^2 x_1^2(1) + \ell \int_0^1 \frac{u^3 r}{1+u^2} dt \\ &= \frac{1}{2} \ell^2 x_1^2(1) + \ell \int_0^1 \frac{u^3 \ell x_1}{1+u^2} dt \\ &= \frac{1}{2} \ell^2 x_1^2(1) + \ell^2 x_2(1) \\ &= \ell^2 \left[\frac{1}{2} x_1^2(1) + x_2(1) \right] \\ &= \ell^2 \emptyset.\end{aligned}$$

When C_D the drag coefficient in hypersonic flow is defined as $C_D = \frac{D}{\Pi q a^2}$ (Bryson and Ho [2])

$$C_D = \frac{4\Pi q \ell^2 \emptyset}{\Pi q a^2} = 4 \left(\frac{\ell}{a} \right)^2 \emptyset.$$

