(NASA-CR-149106) COMPUTATIONAL ALTERNATIVES TO OBTAIN TIME OPTIMAL JET ENGINE CONTROL M.S. Thesis (Notre Dame Univ.) 73 p HC A04/MF A01 CSCL 21E

N77-10060

Unclas G3/07 08034

Department of

ELECTRICAL ENGINEERING

UNIVERSITY OF NOTRE DAME, NOTRE DAME, INDIANA

COMPUTATIONAL ALTERNATIVES TO OBTAIN TIME OPTIMAL JET ENGINE CONTROL

R. J. Basso and R. J. Leake

Department of Electrical Engineering
University of Notre Dame
Notre Dame, Indiana 46556

Technical Report No. EE-7611

October, 1976

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

COMPUTATIONAL ALTERNATIVES TO OBTAIN TIME OPTIMAL JET ENGINE CONTROL

Abstract

This work presents two computational methods to determine an open loop time optimal control sequence for a simple single-spool turbojet engine described by a set of non-linear differential equations. Both methods are modifications of widely accepted algorithms which can solve fixed time unconstrained optimal control problems with a free right end. Constrained problems to be considered have fixed right ends and free time.

Dynamic Programming, originally formulated by Bellman, is defined on a standard problem and it yields a successive approximation solution to the time optimal problem of interest. A feedback control law is obtained and it is then used to determine the corresponding open loop control sequence.

The Fletcher-Reeves Conjugate Gradient Method has been selected for adaptation to solve a non-linear optimal control problem with state variable and control constraints. It uses gradient information to improve the performance index of a nominal trajectory toward the minimum value. The control sequence which produces this minimum trajectory is the open loop time optimal control sequence.

The two above methods are theoretically and computationally extended to include the free time, fixed right end time optimal control problem with state variable and control constraints.

Computer software is developed which can solve a general class of constrained non-linear time optimal control problems. It is shown that the two methods produce similar solutions to the turbojet problem.

TABLE OF CONTENTS

CHAPTER I

PROBLEM DEFINITION

1.1 Introduction

This chapter describes the unconstrained fixed time optimal control problem which well accepted computational methods are able to solve. It is seen that the jet engine presents a tougher problem due to state and control variable constraints and due also to the fixed right end which is the design objective. Initially an optimal control problem is carefully defined and then extended so that the jet engine problem is included in the new class of problems for which the computational methods are to be adapted. Although the jet engine is described by a continuous time model, discrete time systems are studied here because the methods are developed to find time optimal open loop control sequences on a digital computer.

1.2 The Standard Discrete Optimal Control Problem

Consider the n^{th} order, time invariant discrete time system

$$x(t+1) = x(t) + f(x(t),u(t)) \qquad (1.2\text{-}1)$$

with starting time k and terminal time N. In the system with m controls, x(t) is an n-dimensional state variable vector and u(t) is an m-dimensional control vector defined at each sampling instant. In general, various starting times and states are considered, but we always denote

$$x(k) = x \qquad (1.2\text{-}2)$$

as our initial time and state of interest. The terminal time

may be fixed or may be defined as the first instant at which the system state reaches a designated target set S. A performance index

$$J_k(x,\underline{u}) = Y(x(N)) + \sum_{t=k}^{N-1} L(x(t),u(t),t) \qquad (1.2\text{-}3)$$

$$t = k,k+1,\ldots,N-1$$

is to be minimized with $u(t) \in U$, a control set, and $\underline{u}$ is the control sequence

$$\underline{u} = u(k),u(k+1),\ldots,u(N-1) \qquad (1.2\text{-}4)$$

It is understood that in the function $L(x(t),u(t),t)$, $x(t)$ is dependent upon the $u(t)$ choice.

Although there are proven methods which can solve the above problem, we actually wish to solve a class of problems which include free time and fixed right end conditions in addition to state and control constraints of the form

$$A\,x(t) + B\,u(t) \leq C \qquad (1.2\text{-}5)$$

It is this class in which the jet engine control problem lies.

1.3 Jet Engine Control Problem

From an accurate description of the Drone engine in [1], Brennan in [2] has derived a seventh order model and further reduced it to a third order model for simulation on a TR-48 analogue computer. The discrete time version, obtained from the continuous time equations by Euler integration, is shown below and the physical representations of the variables are listed in figure 1.1. In this model, $P_4(t+\Delta t)$ is understood to be $P_4(t+1)$ and $P_4(t)$ is understood to be P_4.

PHYSICAL REPRESENTATION OF VARIABLES

N = rotational speed

P_b = burner density

P_4 = burner pressure

$\dot{w}_f$ = fuel flow

$\dot{w}_3$ = compressor discharge mass flow

T_3 = compressor discharge temperature

Figure 1.1

$$P_4(t+1) = P_4 + \Delta t\left(\left(.93586\,\frac{P_4}{P_b} + 31.486\right)\dot{w}_f + 21.435\dot{w}_3 T_3 - 53.86\,\frac{P_4^2}{P_b}\right) \qquad (1.3\text{-}1)$$

$$P_b(t+1) = P_b + \Delta t(37.78\dot{w}_3 - 38.448P_4 + .66849\dot{w}_f) \qquad (1.3\text{-}2)$$

$$N(t+1) = N + \Delta t(1.258/N)\left(\frac{P_4^2}{P_b} - \dot{w}_3 N^2\right) \qquad (1.3\text{-}3)$$

$$\dot{w}_3 = 1.3009N - .139825P_4 - .13982\sqrt{P_4^2 + .41688N^2 - .0899P_4N} \qquad (1.3\text{-}4)$$

$$T_3 = .64212 + .35788N^2 \qquad (1.3\text{-}5)$$

The two constraints are 1, the surge margin constraint

$$P_4 \leq 1.25N \qquad (1.3\text{-}6)$$

and 2, the turbine inlet temperature constraint .

$$P_4 \leq 1.25P_b \qquad (1.3\text{-}7)$$

The variables in the above equations are normalized in [2] about an equilibrium point ($P_{4\,e}=1, P_{b\,e}=1, N_e=1$) such that

$f(P_{4\ e}, P_{b\ e}, N_e) = 0$. The control objective is to find a discrete open loop fuel sequence which takes the system from an initial state x of windmill (P_4=.5384, P_b=1.774, N=.5461) to a final state x_F of military thrust (P_4=1, P_b=1, N=1) in minimum time such that the surge margain and turbine inlet temperature constraints are satisfied.

1.4 Reduction to Second Order Model

The effect of $\dot{w}_f$ on the system occurs primarily in equation (1.3-1); its effect on equation (1.3-2) is minor. In addition the main single influence on equation (1.3-2) is $\dot{w}_f$, so the assumption that P_4 can be controlled almost directly by $\dot{w}_f$ is made. Therefore, a reduction of the system to a second order problem is made and it is expressed, in the discrete state variable form of section 1.2, as

$$x_1(t+1) = x_1 + \Delta t(37.78\dot{w}_3 - 38.448u_1 + .66849) \tag{1.4-1}$$

$$x_2(t+1) = x_2 + \Delta t(1.258/x_2)\left(\frac{u_1^2}{x_1} - \dot{w}_3 x_2^2\right) \tag{1.4-2}$$

$$\dot{w}_3 = 1.3009x_2 - .139825u_1 - .13982\sqrt{u_1^2 + .41688x_2^2 - .0899u_1x_2} \tag{1.4-3}$$

with the control constraints

$$u_1 \leq 1.25\ x_2 \tag{1.4-4}$$

$$u_1 \leq 1.25\ x_1 \tag{1.4-5}$$

In this second order model, P_4 is now understood to be the control variable with $P_b = x_1$ and $N = x_2$ the state variables.

Equation (1.4-1) includes the constant term because at the equilibrium condition where the constant term would have its largest effect, $\dot{w}_f$ should have a value close to one. The state variable constraints remain identical to the third order model constraints but with the following subtle difference; now, the control is constrained by its position in the state space.

1.5 Scope

Three accepted methods which can solve the unconstrained, fixed time, free right end problem described in section 1.2 are Dynamic Programming, the Discrete Minimum Principle, and ordinary mathematical programming which minimizes a function of many variables. In this thesis, two of these methods are adapted to solve the general class of fixed right end, time optimal control problems with state variable and control constraints.

In Chapter II, a successive approximations technique extends Dynamic Programming to produce a time optimal feedback control law from which an open loop control sequence can be determined. In Chapter III, non-linear programming is used to solve the jet engine problem and produce an open loop control sequence. Constraints are difficult to handle in the Conjugate-Gradient approach, but a simple feasible directions technique is used in conjunction with penalty functions to give satisfactory convergence without jamming. Careful inspection of the respective solutions shows that each method yields the same time optimal control sequence.

CHAPTER II

DYNAMIC PROGRAMMING

2.1 Motivation

Dynamic Programming is a method well suited for solving low order, fixed time, free right end optimal control problems with state and control constraints. The feedback control solution and the cost function are calculated in successive steps as the state and control constraints actually reduce the required number of computations per step. The main difficulty is to include the fixed right end, free time problems into the class of problems which Dynamic Programming can solve. The appropriate modification of the theory is given below.

Pontryagin in [3] has shown that time optimal control problems very often tend to have bang-bang solutions. The Dynamic Programming method is also well suited to the bang-bang solution because the control is quantized and each control candidate is tested at each point in the state space to determine the optimal control. Therefore, regions of similar controls are expected and boundaries separating these regions are not surprising. Suggestions on accurately representing these boundaries are presented later in this chapter.

2.2 Theory

Let K denote the integers, R the real numbers, X an arbitrary nonempty state set, and $S \in X \times K$ an arbitrary nonempty target set. Suppose we also have a control constraint U(x) set

which depends on the state x and is such that controls in U(x) guarantee that the next state is in the state set X.

Consider the discrete time system of section 1.2 and the performance index $J_k(x,\underline{u})$ in (1.2-3) to be minimized. We assume that

$$V_k(x) = \min_{\underline{u}} J_k(x,\underline{u}) \qquad (2.2\text{-}1)$$

exists, where $\underline{u}$ denotes any admissible sequence u(k),u(k+1), ...,u(N-1) with values $u(t) \in U(x(t))$. Only the above assumption is required to prove Bellman's Conditions [4], [5], and [6]

$$V_k(x) = \min_{u \in U(x)} \left[L(x,u,k) + V_{k+1}(x+f(x,u)) \right] \qquad (x,k) \notin S \qquad (2.2\text{-}2)$$

$$V_k(x) = Y(x) \qquad (x,k) \in S$$

which are necessary and sufficient for optimality. These are the usual equations of Dynamic Programming. A particular $u = v(x,k) \in U(x)$ which minimizes the expression above can be used to make up v(x,k) for all $(x,k) \in S$ and comprises an optimal feedback control law.

The optimal open loop control sequence $\underline{u}$ can be determined from v(x,k) as follows. Let $x_v(t)$ be the optimal trajectory obtained using v(x,k) as a feedback control law. Then

$$u(t) = v(x_v(t),t) \qquad (2.2\text{-}3)$$

and

$$\underline{u} = u(k),u(k+1),\ldots,u(N-1) \qquad (2.2\text{-}4)$$

An alternative method for solving problems of this class

called "successive approximations", was suggested by Bellman [4] and later developed by Leake, Liu, and Richardson in [6] and [7].

Let $V_k^n(x)$ be any function such that $V_k^n(x) \geq V_k(x)$ and let $v^n(x,k)$ be a control law which results when performing the minimization

$$\min_{u \in U(x)} \left[L(x,u,k) + V_{k+1}^n(x+f(x,u)) \right] \qquad (2.2\text{-}5)$$

$$(x,k) \notin S$$

Then it is shown in [6] that if $V_k^{n+1}(x)$ is the performance function resulting from $v^n(x,k)$, we have

$$V_k(x) \leq V_k^{n+1}(x) \leq V_k^n(x) \qquad (2.2\text{-}6)$$

and further $V_k^n(x)$ converges monotonically to $V_k(x)$ in a finite number of steps although each (x,k) may require a different number of steps.

In the special case where $V_k(x)$ is independent of k, $V_k(x) = V(x)$, the approximating sequence $V_k^n(x)$ may be taken as independent of k also to yield $V_k^n(x) = V^n(x)$ and (2.2-5) becomes

$$V^{n+1}(x) = \min_{u \in U(x)} \left[L(x,u,k) + V^n(x+f(x,u)) \right]$$

$$(x,k) \notin S \qquad (2.2\text{-}7)$$

Problem of Interest

Recall now that in the jet engine problem (sections 1.3 and 1.4) we wish to find the minimum number of steps (and associated optimal controls) that it takes to drive the system

$$x(t+1) = x(t) + f(x(t),u(t)) \qquad (2.2\text{-}8)$$

from $x(k) = x$ to a fixed state x_F with associated state and control constraints $x(t) \in X$ and $u(t) \in U(x(t))$. Note that the minimum number of steps to reach the target state x_F does not depend on the starting time k. Thus the minimum number of steps can be expressed as

$$V_k(x) = V(x) \tag{2.2-9}$$

a function depending only on x. Similarly the control law $v(x,k) = v(x)$.

Ordinary Dynamic Dynamic Programming Problem

We now define a standard Dynamic Programming problem which yields a successive approximation solution to the time optimal problem of interest. Simply let the system be the same as before with target set

$$S = S_1 \cup S_2 \tag{2.2-10}$$

where S is illustrated in figure 2.1 and

$$S_1 = \{ (x,k) : k = 0, \; x \in X \} \tag{2.2-11}$$

$$S_2 = \{ (x,k) : k < 0, \; x = x_F \}$$

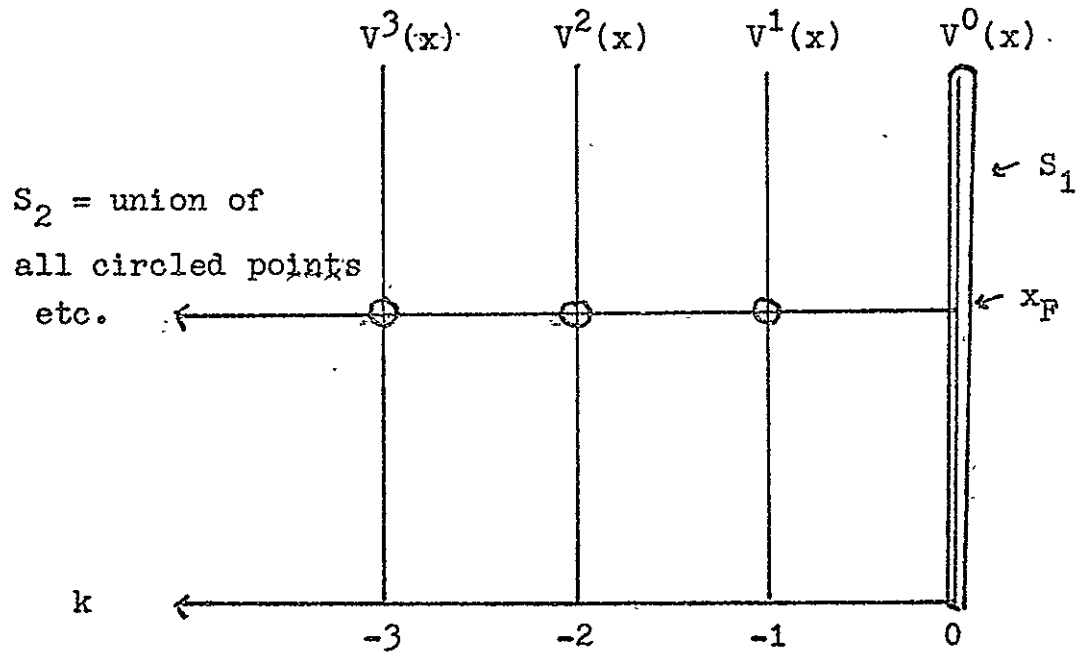
Let X and U(x) be as above, let N=0, let $L(x,u,k) = 1$ to provide a time penalty to the system, andlet Y(x) be chosen such that

$$Y(x) \geq V(x) \tag{2.2-12}$$

$$Y(x_F) = 0$$

Then Bellman's conditions are as in equation (2.2-2) and the problem can be solved as an ordinary Dynamic Programming problem. Keep in mind however, that $(x_F,k) \in S$ so we always have

TARGET SET FOR SUCCESSIVE APPROXIMATIONS DEVELOPMENT



Vertical lines represent the set X
defined at each k.

Figure 2.1

$$V_k(x_F) = 0 \qquad (2.2\text{-}13)$$

for any $k \leq N = 0$.

Successive Approximation Solution

To show that the ordinary Dynamic Programming solution above amounts to a successive approximation solution to the problem of interest, note that the actual target set of interest is simply the set of all (x,k) where $x = x_F$. We can establish a successive approximation of V(x) by choosing

$$V^0(x) = Y(x) \qquad (2.2\text{-}14)$$

Then, if we equate

$$V_k(x) = V^{-k}(x) \quad , \quad k = 0,-1,-2,-3,\ldots \qquad (2.2\text{-}15)$$

we see that Bellman's Conditions are equivalent to

$$V^{n+1} = \min_{u \in U(x)} \left[L(x,u,k) + V^n(x+f(x,u)) \right] \qquad (2.2\text{-}16)$$

$$(x,k) \notin S$$

$$n = 0,1,2,\ldots$$

which is simply the successive approximation equation. This establishes that the Dynamic Programming problem actually gives a successive approximation solution, with

$$V^n(x) \downarrow V(x) \qquad (2.2\text{-}17)$$

and associated feedback control law

$$v^n(x) \longrightarrow v(x) \qquad (2.2\text{-}18)$$

In practice, the state set is discretized and interpolation is used to get approximate solutions, but convergence still occurs. Choosing some k_T as a time when convergence is adequate, we equate

$$V^{-k_T} = V(x) \tag{2.2-19}$$

where V(x) represents the number of time steps needed for

$$v(x,k_T) = v(x) \tag{2.2-20}$$

to take the system from initial state x to final state x_F. V(x) also reveals those initial states x for which a control sequence such that $u(t) \in U(x(t))$ cannot be found to move the system to the final state x_F. The uncontrollable regions are simply found by observing that $U(x) = \emptyset$ for the states in those regions. In the computer program, a penalty cost is arbitrarily assigned to those states and a suitable representation is given to the corresponding feedback control laws to indicate that no admissible control exists.

2.3 State Variable Quantization

The first computational requirement in using a dynamic program is to determine an adequate quantization, independent of k, of the state variables and the control variables. In the jet engine problem, the time constants of P_b and N, obtained from linearizing the third order model about the design objective, (see [2] or section 3.5) differ by an order of magnitude. This difference implies that P_b will be able to react more rapidly than N. To complicate matters, a suitable choice for the time step size for the Euler integration must be made to restrict the motion of one time step to one increment in each state variable.

Larson in [5] presents a relation which is useful for determining a proper state variable quantization. If we let

$$F(x(t),u(t)) = f(x(t),u(t))/\Delta t \qquad (2.3\text{-}1)$$

and let $x_i(t)$ represent the i^{th} component of $x(t)$, let $u_j(t)$ represent the j^{th} component of $u(t)$, and let F_i represent the i^{th} discrete time function, then Larson's relation is

$$\Delta t = \min_{\substack{i=1,2,\ldots,n \\ j=1,2,\ldots,m}} \frac{x_i(t)}{|F_i(x_i(t),u_j(t))|} , \qquad (2.3\text{-}2)$$

The above equation requires that $F_i(x_i(t),u_j(t))$ be scanned over all possible states and controls to determine the maximum motion of each state variable per time step. A short computer program easily determines that $F_{1\ max} = P_{b\ max} = 40$ and $F_{2\ max} = N_{max} = 2$ for the jet engine problem. From this information, the values of Δt, ΔP_b, and ΔN are determined.

In problems with state variable and control constraints, each possible control must be in the set $U(x)$ before it is tested to find the best $u(t)$. The quantization of these problems should include a large number of state variable points which land on the boundary of the constraints and it is necessary that those points in S_2 be members of the quantized grid. In the jet engine problem, the discrete quantization must include the point (P_b=1,N=1) and several points which satisfy constraints (1.4-4) and (1.4-5) with equality.

2.4 Initial Cost Function

The initial assigned cost function, $V^0(x)$, is designed to force the system to rapidly approach x_F. One approach is to assign an arbitrary large constant to all $x \neq x_F$ but it is found to decrease the convergence rate of the successive approximations solution because the large initial constant

introduces significant interpolation errors into the algorithm. If V(x) denotes the minimum number of time steps required for the feedback control law to take the system from an initial state x to a final state x_F, equation (2.2-12) establishes a lower bound for the initial constant which is the maximum time expected divided by the time step size. A lower constant function increases the convergence rate of the successive approximations method and also reduces the errors induced by interpolation. The lower bound can be found with a few trial Dynamic Programs or with preliminary equation study. For the jet engine problem, the optimal time required to move from windmill to military thrust is slightly less than one second, or approximately .8 seconds. Consequently the constant is set equal to $1/\Delta t$.

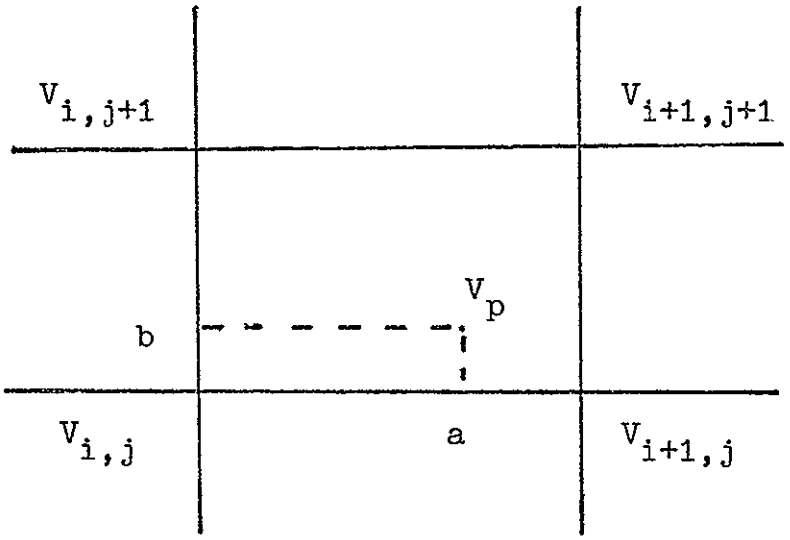
2.5 Interpolation

Recall that $V^n(x)$ will have numerical representations stored on the computer only for those states x which lie on the discrete quantization grid. If a particular control $u \in U(x)$ forces the motion

$$x' = x + f(x,u) \qquad (2.5\text{-}1)$$

to a state x' not defined on the discrete grid, interpolation is required to approximate $V^{n+1}(x')$. Figure 2.2 illustrates the general case and presents an interpolation formula which produces good results for the jet engine problem.

Early results have indicated a tendency for the cost function associated with the members of S_2, $V^{-k}(x_F) = V_k(x_F,k)$,



$$V_p = (1-a)(1-b)V_{i,j} + b(1-a)V_{i,j+1} + a(1-b)V_{i+1,j} + abV_{i+1,j+1}$$

Figure 2.2

to incur with each iteration small positive values which propagate through the algorithm and grow larger. These errors arise from quantizing and interpolating non-linear system equations and cost functions. To minimize this effect, $V^n(x_F)$ is reset to zero after each iteration.

In figure 2.3, a flow chart, describing the general Dynamic Programming Method with the successive approximations extension, is presented. The actual computer program, written in FORTRAN IV for use on the FORTGI compiler in Notre Dame's IBM 370/158 computer system, is listed and documented in appendix A.

2.6 Control Regions for the Turbojet Engine

Recall the reduced jet engine problem of section 1.4

Flow Chart for Dynamic Programming Method

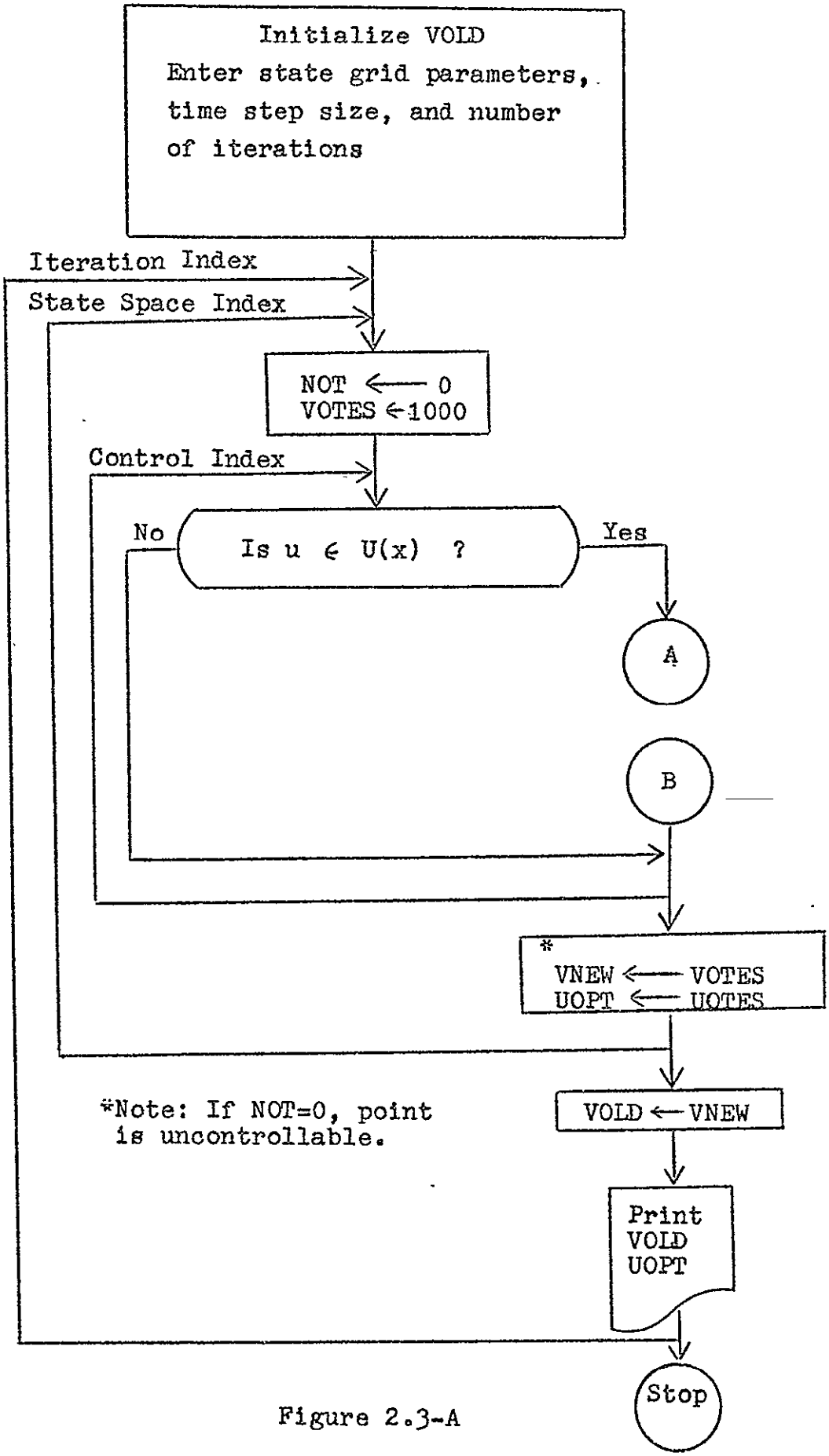


Figure 2.3-A

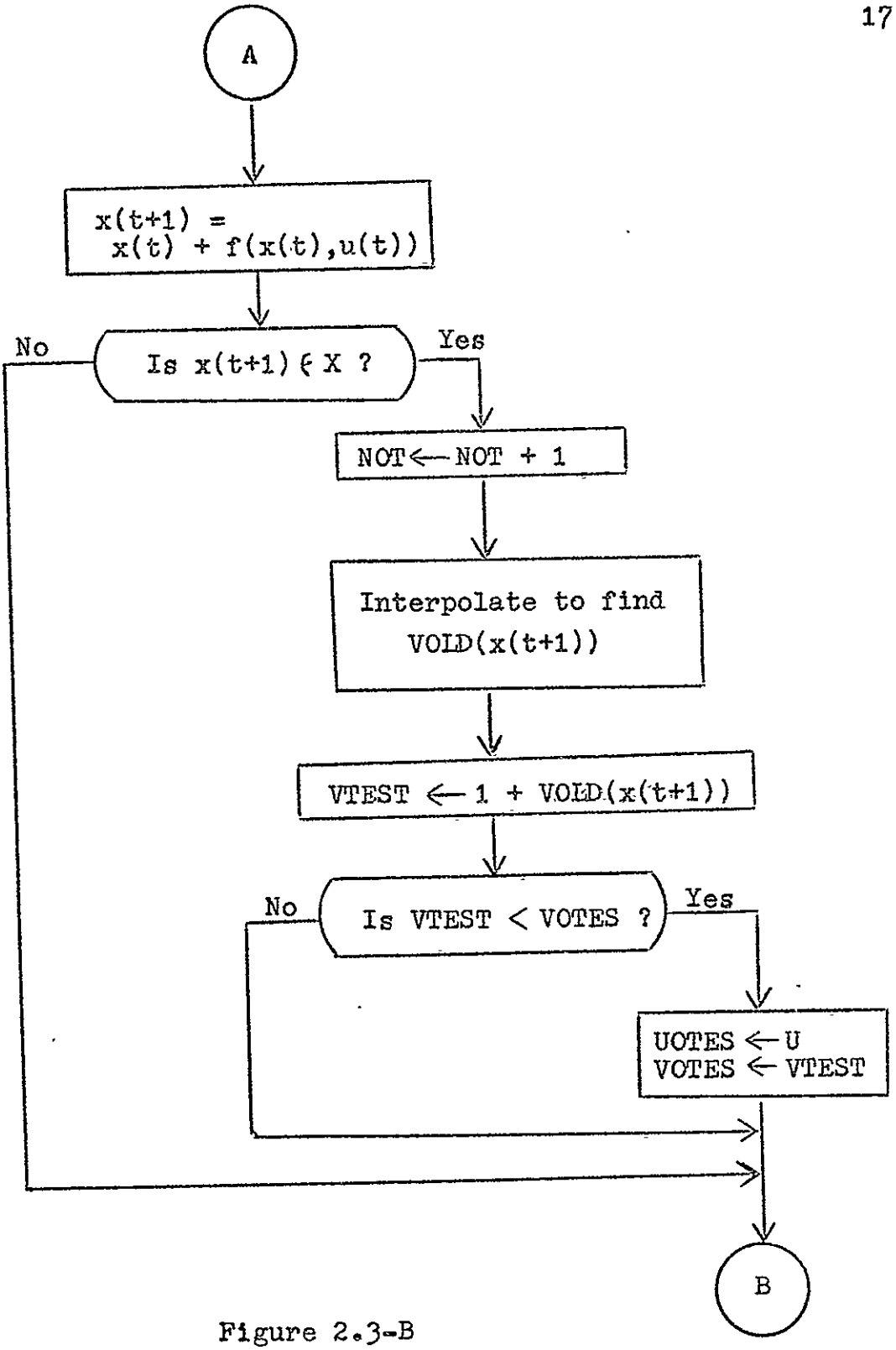

A
x(t+1) =
x(t) + f(x(t),u(t))
No
Is x(t+1) ∈ X ?
Yes
NOT ← NOT + 1
Interpolate to find
VOLD(x(t+1))
VTEST ← 1 + VOLD(x(t+1))
No
Is VTEST < VOTES ?
Yes
UOTES ← U
VOTES ← VTEST
B

Figure 2.3-B

which has one control variable P_4 (or u_1) and two state variables, P_b (or x_1) and N (or x_2). The control set U consists of real numbers which lie in the interval (0.5, 1.25) and various state sets X are considered which have members surrounding the design objective. Recall that the quantization is independent of k so that the design objective can loosely be referred to as the target. As the Dynamic Programming Method is run, the successive approximation solutions show that the controllable region of P_b grows quickly while the controllable region of N grows slowly. This is due to the vast difference in the time constants of P_b (T_{P_b}=.0309) and N (T_N=.352). A state set X consisting of a narrower range of N (0.8-1.1) and a wide range of P_b (0.5-1.8) produces the most meaningful results. For greater detail near control region boundaries, smaller quantization increments and state sets X defining smaller regions can be studied. They yield more information about the boundaries and the feedback control regions are more clearly defined.

The time optimal feedback control solution is illustrated on a state variable map in which three distinct control regions can be seen in figure 2.4. The sharp boundaries which are examined closely in section 2.7 and which separate these regions are indicated by solid lines while the motion of the system, $x_v(x,k_T)$, due to the feedback control law, $v(x,k_T)$, is indicated by dotted lines for several starting states x. The design objective x_F is denoted by a circle and every tenth of a second along the trajectories is marked by crosses.

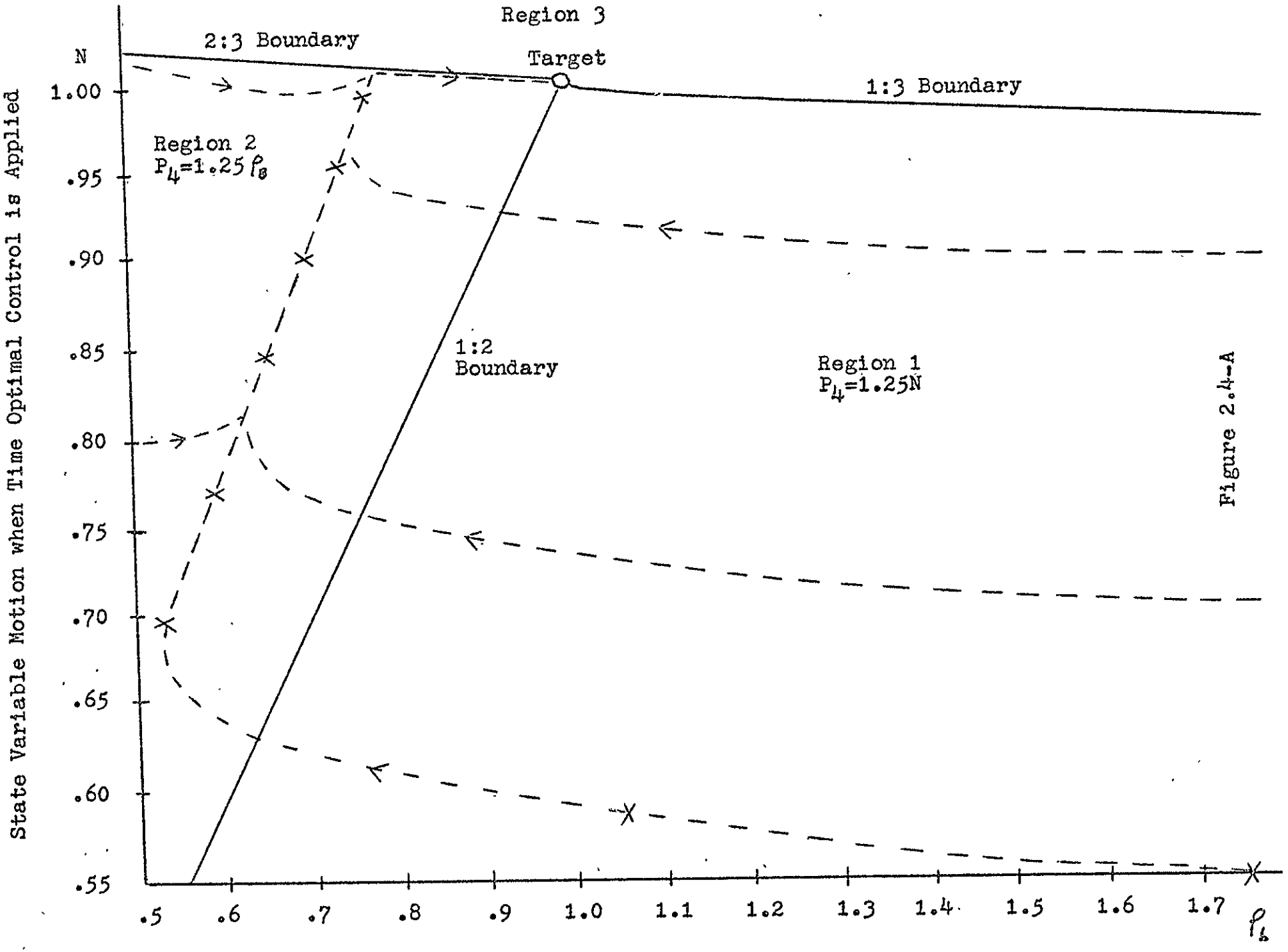
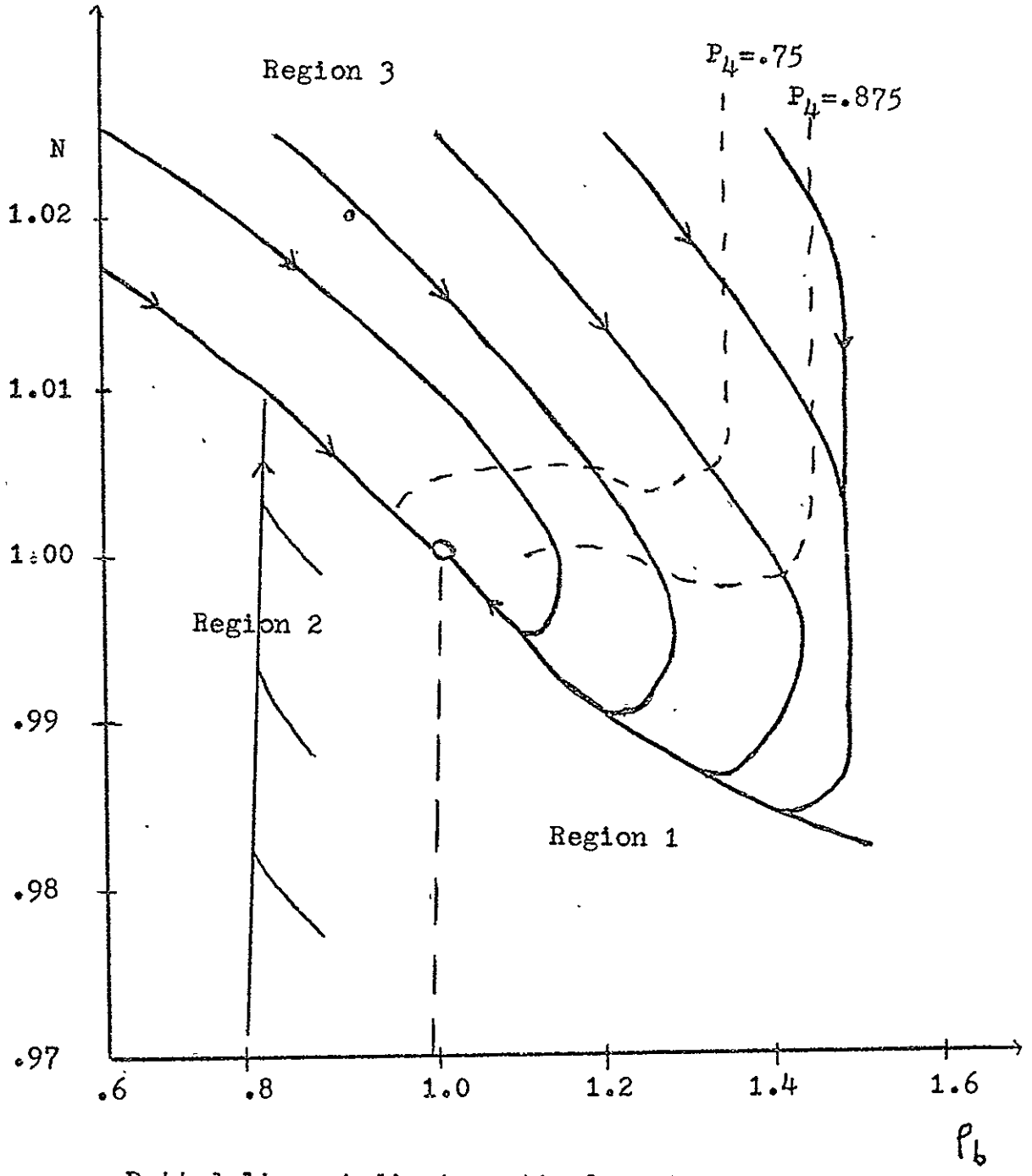

Region 3
2:3 Boundary
Target
1:3 Boundary
N
1.00
.95
.90
.85
.80
.75
.70
.65
.60
.55
Region 2
$P_4=1.25\rho_s$
1:2
Boundary
Region 1
$P_4=1.25N$
State Variable Motion when Time Optimal Control is Applied
.5
.6
.7
.8
.9
1.0
1.1
1.2
1.3
1.4
1.5
1.6
1.7
ρ_s

Figure 2.4-A

Detail of Region 3

$P_{4\ min} = .75$

$\rho_{6\ max} = 1.5$



Dotted lines indicate optimal control contours

Figure 2.4-B

In region 1, the control variable rides the surge margain constraint. This causes a rapid decrease in P_b with a little increase in N. When the motion reaches the 1:2 boundary, the control switches to ride the turbine inlet temperature constraint. In region 2, all starting states have motions which take the system to a common main path which does not lie on a region boundary. Along this path, N increases to its design objective but only a small increase in P_b is observed. At a point determined by the 2:3 boundary, the control jumps to its minimum value and the system motion rapidly approaches the target where the control assumes a value of one. It is this rapid switching action which Pontryagin refers to as a bang-bang control.

Region 3 is the area in which the value of N is higher than the military thrust value. Physically, this area is of little interest, but it reveals an interesting control pattern which has no obvious analytic properties such as those in regions 1 and 2. The results in figure 2.4-B are obtained from a control set $U \in (.75\text{-}1.25)$, and state variable set $N \in (.97\text{-}1.025)$ and $P_b \in (.6\text{-}1.5)$. The time optimal feedback solution reduces N rapidly by reducing the control variable to its minimum value, however the low control simultaneously increases the value of P_b above the design objective. The dotted line contours indicate when the control value is continuously raised to cause the motion of the system to move along the 1:3 boundary, where the control follows the surge margin, directly to the target.

2.7 Accurate Boundary Descriptions

At certain sections in the state map, the time optimal feedback control solution reveals the location of a boundary where the trend of controls changes. Between regions 1 and 3, a line is implied for which states above have an optimal feedback control value $v(x) \doteq 1.25N$ and for which states below have feedback control values which satisfy the surge margain constraint with equality. The exact boundary is the trajectory which, formed by the motion of the system due to controls lying on the surge margain constraint, proceeds directly to the target. This is determined in figure 2.5 by integrating the system forward with a few selected states as initial conditions. In the neighborhood of this boundary, the control contour is continuous, or in other words, there is a smooth transition of feedback control values from region 3 into the boundary.

The 2:3 boundary separates two regions having radically different optimal control strategies. States x which lie in region 2 have $v(x) = 1.25P_b$ and states x which lie on and above this boundary in region 3 have corresponding minimum feedback control values. Figure 2.6 illustrates the resulting boundary as a function of $P_{4\ min}$ which can be chosen arbitrarily in the reduced jet engine problem. Its most realistic value depends upon how quickly P_4 decreases when $\dot{w}_f$ in the third order model is suddenly reduced to zero.

The boundary separating regions 1 and 2 is simply the line $P_b = N$.

Detail of 1:3 Boundary

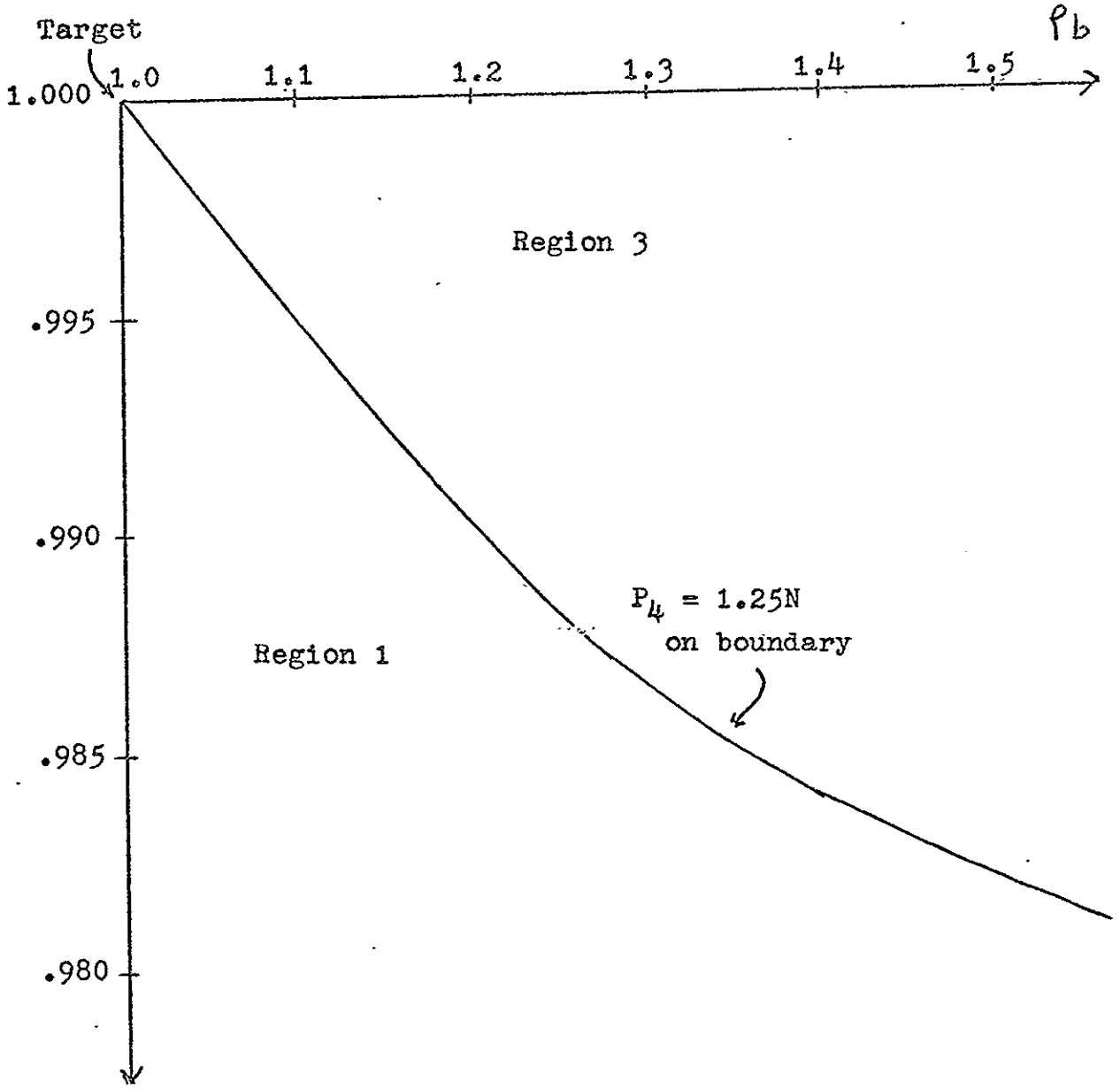


Figure 2.5

Detail of 2:3 Boundary

for several $P_{4\ min}$ values

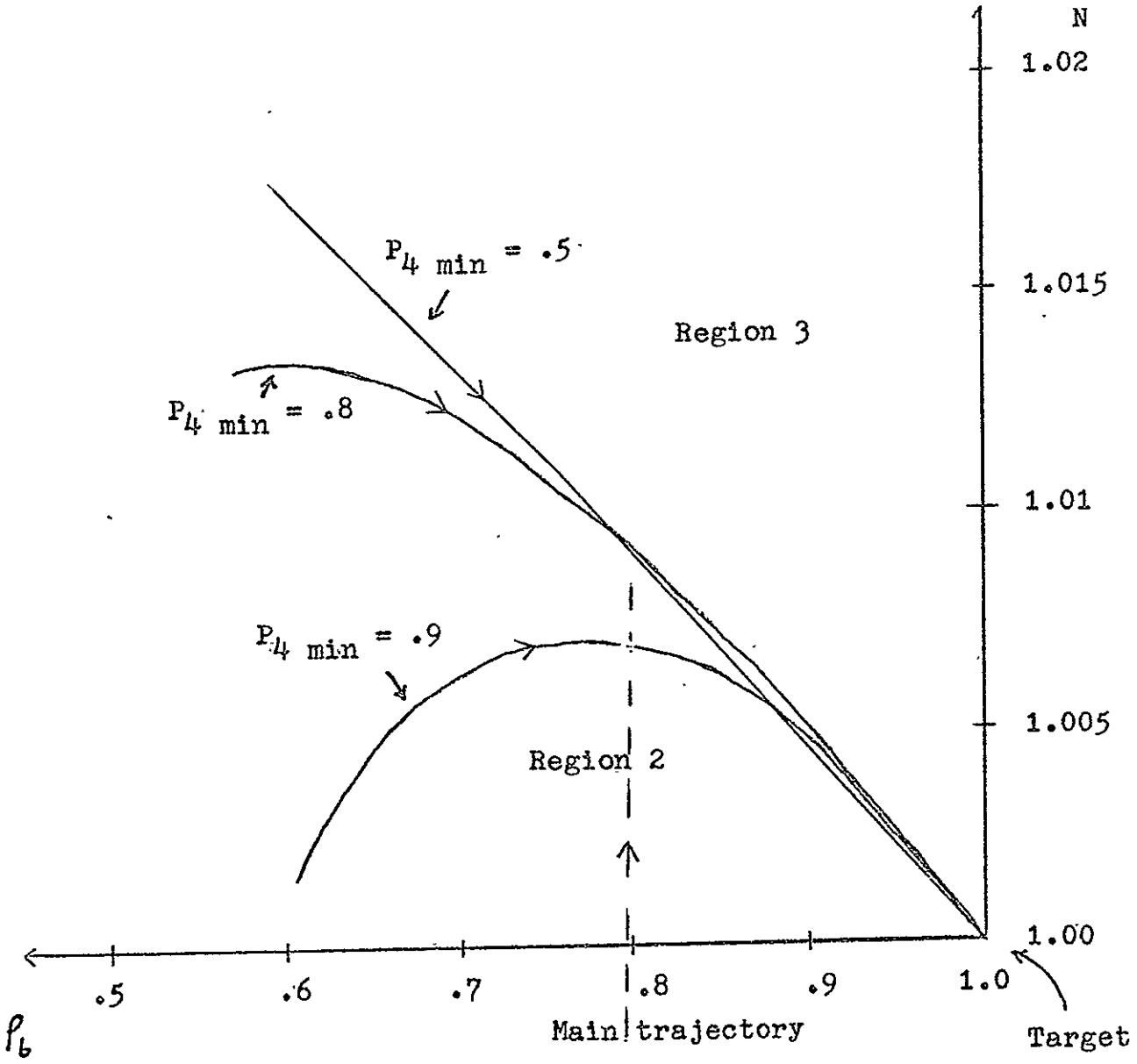


Figure 2.6

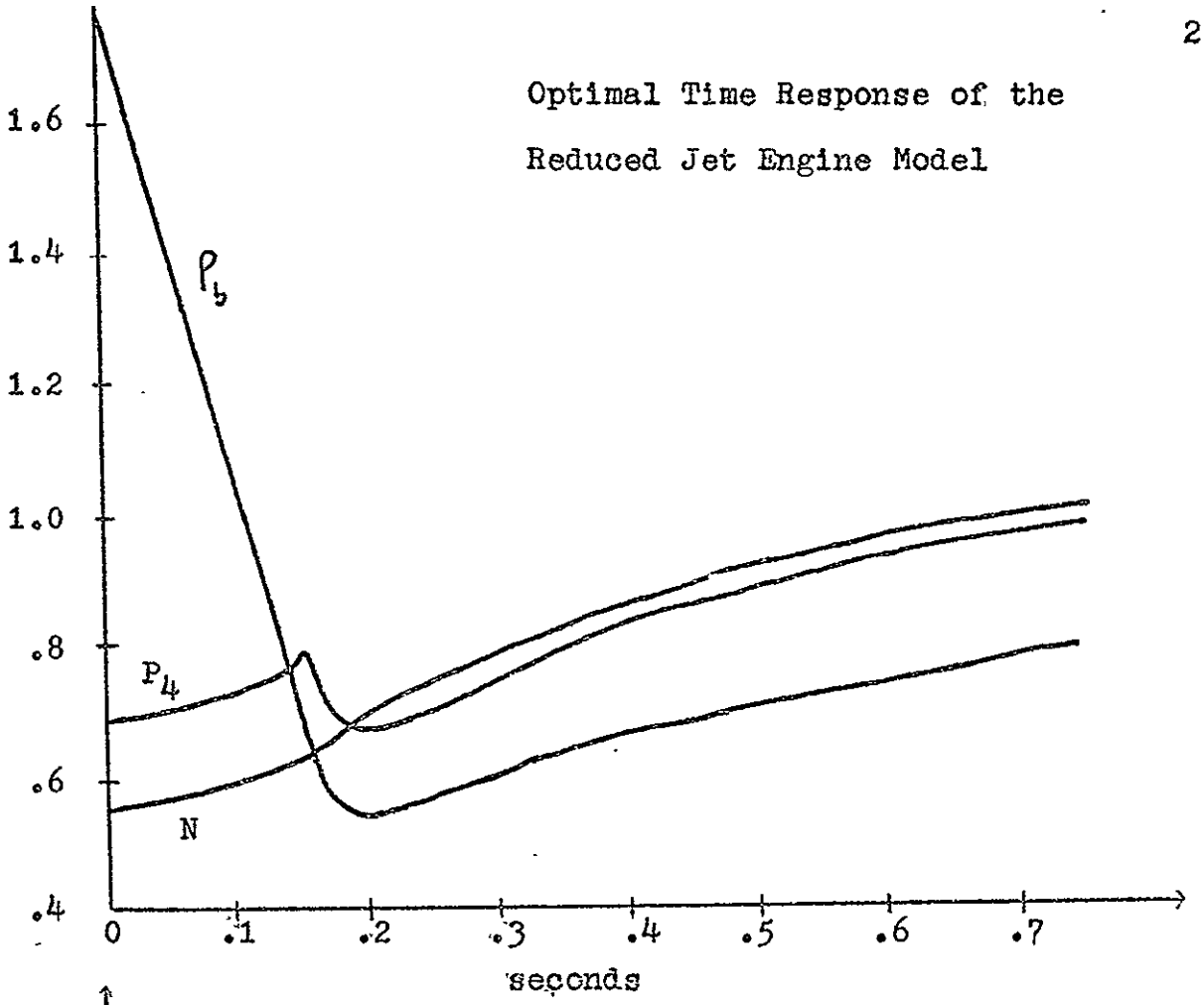
2.8 Summary of Second Order Model Controls

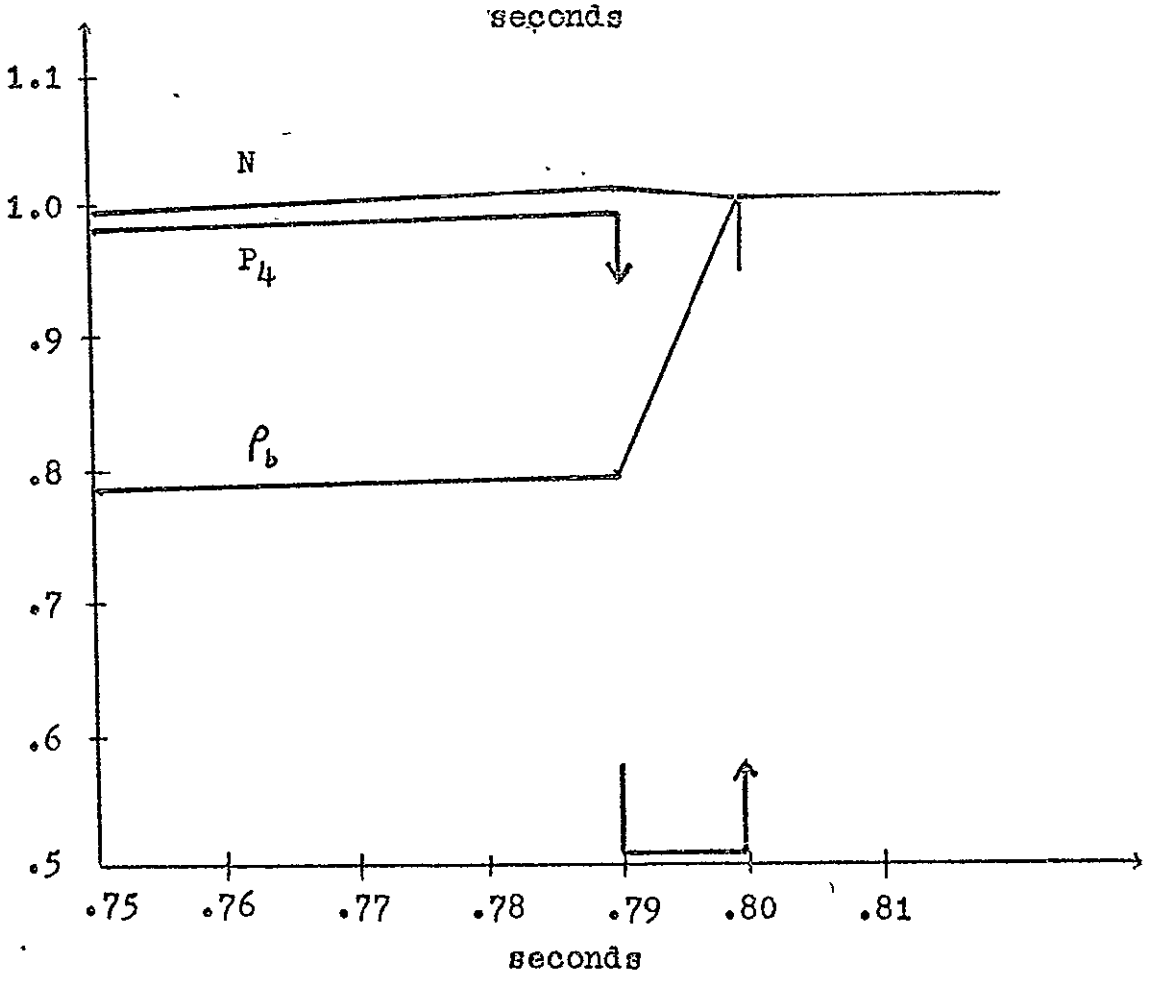
Figure 2.7 shows the optimal time plots of the reduced jet engine problem to which the time optimal feedback control laws obtained from the dynamic programming algorithm are applied. P_4 has three control rules. Initially, P_4 follows the surge margin constraint until P_b is lowered sufficiently to equal N, at which point P_4 will follow the turbine inlet temperature constraint. Physically, these two rules represent the maximum allowable throttle without stalling or overheating the jet engine. At .79 seconds, N has a value slightly above the design speed but P_b will have only 80% of its design value. The point is determined by the 2:3 boundary where P_4 is dropped to $P_{4\ min}$ for an instant and then returned to a value of one. This discontinuous rapid switching action is an example of a bang-bang control and it moves P_b directly to the design value in only .01 seconds. Therefore, the jet engine is controlled from windmill to military thrust in .8 seconds.

2.9 Application to Third Order System

From the reduced system information, the optimal time responses for the third order description of the Drone engine can be calculated. To determine the proper $\dot{w}_f$, it is required to find the desired P_4 from the time optimal feedback control law and calculate the value of $\dot{w}_f$ which will force P_4 to assume values which follow the time optimal control law. When the trajectory reaches the 2:3 boundary, $\dot{w}_f$ is suddenly reduced to a zero value which causes P_4 to decrease and P_b to

Optimal Time Response of the
Reduced Jet Engine Model





increase to its design value. Then, $\dot{w}_f$ must be a positive impulse to raise P_4 to the target rapidly without changing N or P_b. Once this is accomplished, $\dot{w}_f$ is set equal to one and the system is in the military thrust state.

The time responses for the third order model are plotted in figure 2.8. The initial impulse of $\dot{w}_f$ forces P_4 up to the N constraint in one time step.

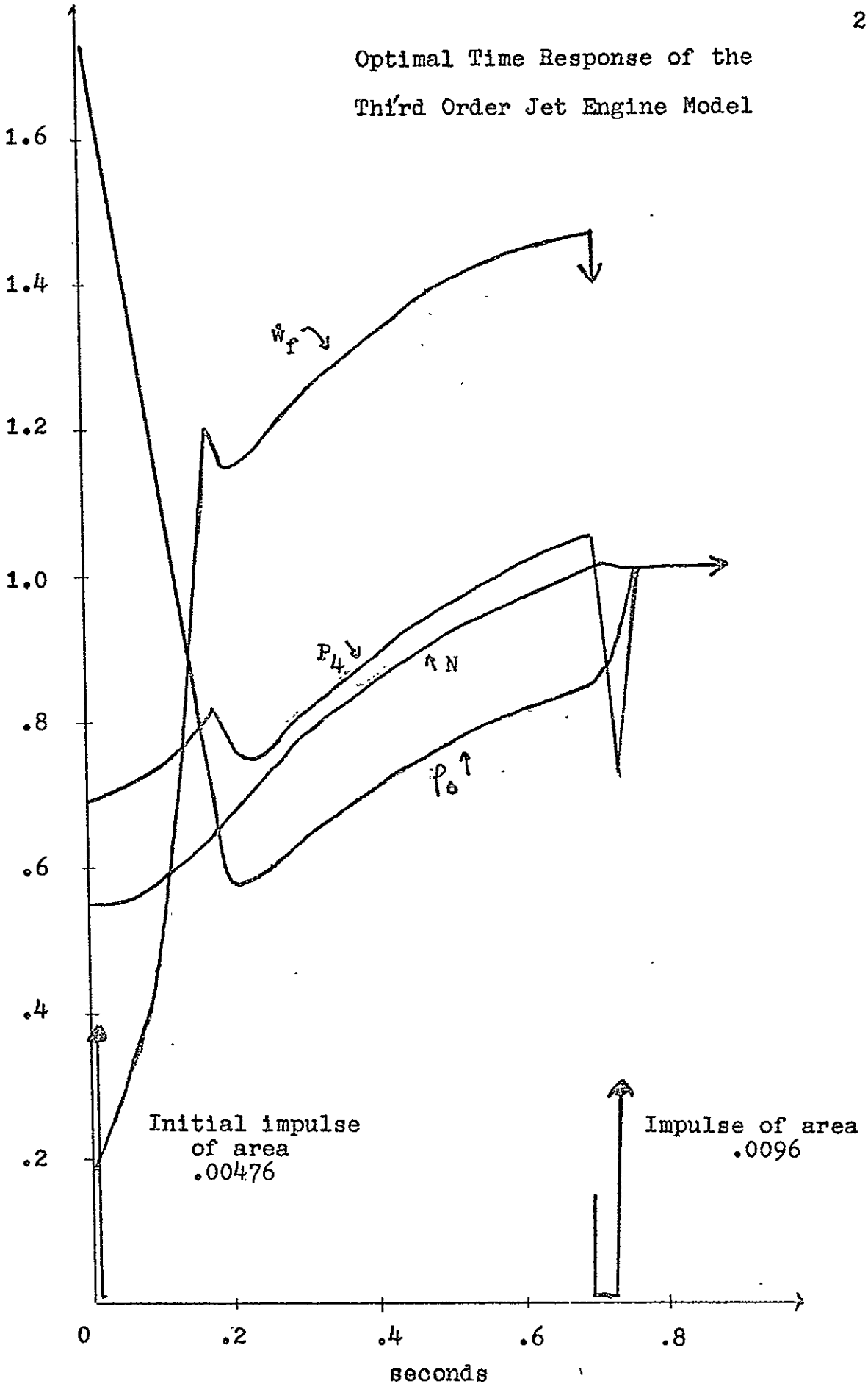

Optimal Time Response of the
Third Order Jet Engine Model
1.6
1.4
1.2
1.0
.8
.6
.4
.2
$\dot{w}_f$
P_4
N
P_0
Initial impulse
of area
.00476
Impulse of area
.0096
0
.2
.4
.6
.8
seconds

Figure 2.8

CHAPTER III

THE MODIFIED FLETCHER-REEVES CONJUGATE GRADIENT METHOD

3.1 Motivation

Ordinary non-linear mathematical programming is a well known method which can solve an unconstrained optimal control problem with fixed time and a free right end by minimizing a performance index, $J(\underline{u})$, which is a function of several variables. This method determines the optimal open loop control sequence and its computational requirements increase only arithmetically with the order of the system under study. In this chapter, a feasible directions idea is presented in conjunction with penalty functions to extend the general class of problems which the Fletcher-Reeves Conjugate Gradient Method can solve to that which includes a constrained time optimal control problem with a fixed right end. The developed computer software is listed and documented in appendix B for use to determine open loop time optimal control sequences for the general class of non-linear free time, fixed right end optimal control problems with state variable and control constraints.

3.2 The Fletcher-Reeves Conjugate Gradient Method

Consider the n^{th} order, time invariant discrete system having m controls

$$x(t+1) = x(t) + f(x(t),u(t)) \qquad (3.2\text{-}1)$$

and

$$x(0) = x \qquad (3.2\text{-}2)$$

with starting time zero, terminal time N, and $f(x(t),u(t))$

is continuously differentiable over the entire state space and control domain. The performance index to be minimized is

$$J(\underline{u}) = K(x(N)) + \sum_{t=0}^{N-1} L(x(t),u(t),t) \qquad (3.2\text{-}3)$$

$$t = 0,1,2,\ldots,N-1$$

which is a function only of $\underline{u}$ given a starting state x where $\underline{u}$ denotes the control sequence

$$\underline{u} = u(0),u(1),\ldots,u(N-1) \qquad (3.2\text{-}4)$$

and

$$u(j) = u_1(j),u_2(j),\ldots,u_m(j) \qquad (3.2\text{-}5)$$

so that $\underline{u}$ is a sequence of Nm real numbers.

This general class of unconstrained free right end fixed time optimal control problems can be solved by the Fletcher-Reeves method which uses a combination of steepest descent and conjugate gradient techniques presented in [8] to minimize a directionally convex multivariable function, $J(\underline{u})$. In figure 3.1, a flow chart for this algorithm is presented and the appropriate equations are described below.

Suppose a nominal control sequence $\underline{u}$ with a resulting state variable trajectory and performance index is selected. The gradient of the trajectory is calculated while the adjoint system equations are solved in reverse time according to

$$y(t) = y(t+1) + y(t+1)\,\nabla_x f(x(t),u(t)) + \nabla_x L(x(t),u(t))$$

$$t = N-1,N-2,\ldots,2,1$$

$$y(N) = \nabla_x K(x(N)) \qquad (3.2\text{-}6)$$

The Fletcher-Reeves Conjugate Gradient Method

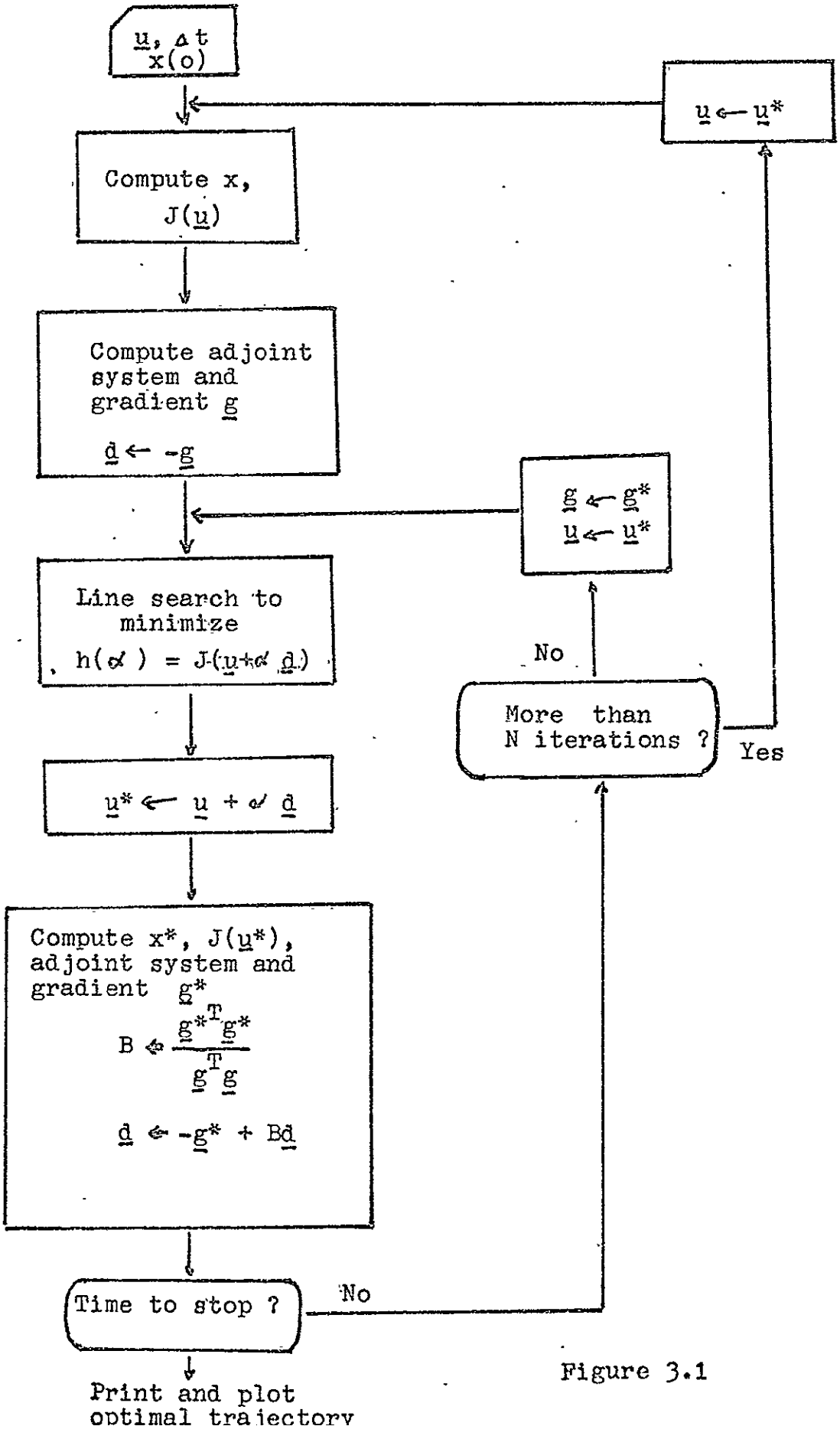


Figure 3.1

with $y(t)$ an array of n length, $\nabla_x f(x(t),u(t))$ is the Jacobian,

$$\nabla_x f(x(t),u(t)) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \qquad (3.2\text{-}7)$$

and

$$\nabla_x L(x(t),u(t)) = \begin{bmatrix} \frac{\partial L}{\partial x_1} & \frac{\partial L}{\partial x_2} & \cdots & \frac{\partial L}{\partial x_n} \end{bmatrix} \qquad (3.2\text{-}8)$$

The gradient is the Nm length sequence

$$g = \left[\nabla_{\underline{u}} J(\underline{u}) = \nabla_{u(0)} J(\underline{u}) \quad \nabla_{u(1)} J(\underline{u}) \cdots \nabla_{u(N-1)} J(\underline{u})\right] \qquad (3.2\text{-}9)$$

and each component can be calculated from the relation

$$\nabla_{u(t)} J(\underline{u}) = y(t+1)\, \nabla_u f(x(t),u(t)) + \nabla_u L(x(t),u(t)) \qquad (3.2\text{-}10)$$

with $\nabla_u f(x(t),u(t))$ and $\nabla_u L(x(t),u(t))$ similarly defined.

It is incidental that the Fletcher-Reeves method under appropriate convexity assumptions actually solves the discrete minimum principle for the above class of problems. If the Hamiltonian of the system is defined by

$$H(x(t),u(t),y(t+1)) = y(t+1)f(x(t),u(t)) + L(x(t),u(t)) \qquad (3.2\text{-}11)$$

then

$$\nabla_{u(t)} J(\underline{u}) = \nabla_u H(x(t),u(t),y(t+1)) \qquad (3.2\text{-}12)$$

which is zero along the optimal trajectory, corresponding to

the Hamiltonian being minimized.

During each first inner iteration, a line search is performed along the conjugate direction of the gradient $\underline{d} = -\underline{g}$ to find the best α such that

$$h(\alpha^*) = \min_{\alpha \geq 0} J(\underline{u} + \alpha \underline{d}) \qquad (3.2\text{-}13)$$

For each successive inner iteration, the direction of the line search is

$$\underline{d}' = -\underline{g}^* + B\underline{d} \qquad (3.2\text{-}14)$$

a linear combination of the present gradient and previous direction with

$$B = \frac{\underline{g}^{*T}\,\underline{g}^*}{\underline{g}^T\,\underline{g}} \qquad (3.2\text{-}15)$$

and $\underline{g}^*$ is the present gradient, $\underline{g}$ the previous gradient, and $\underline{d}$ is the previous direction. After N inner iterations, the direction is again set equal to the conjugate direction of the gradient and the present performance index is compared to the previous one to determine if another set of inner iterations is required. If, during a set of inner iterations, the line search produces an $\alpha^* = 0$, the performance index can not be improved with further inner iterations so the algorithm skips to the next outer iteration. The control sequence $\underline{u}$ which minimizes $J(\underline{u})$ is the optimal open loop control sequence.

3.3 The Extended General Class of Problems

We actually wish to solve a constrained time optimal control problem with a fixed right end. Recall the reduced

jet engine problem of section 1.4 with control constraints and a design objective $x_F = (P_b=1, N=1)$. A combination of a simple feasible directions idea used in conjunction with penalty functions will extend the capabilities of the Fletcher-Reeves Conjugate Gradient Method to solve the general constrained time optimal control problem with a fixed right end.

3.4 Feasible Directions Modification

Several barrier functions were tried which would heavily penalize the system if a control constraint is violated. All attempts to find a continuously differentiable function to force the control variable, P_4, to obey its constraints failed, so great is the tendency of the system to overshoot them. The overshoot path always yields a lower performance index than any legal control sequence would yield. As the barrier functions are steepened, computer overflows appear when the performance index is differentiated, and therefore barrier functions do not provide a solution to the overshoot problem in the reduced jet engine model.

Hence, barrier functions are abandoned and a simple feasible directions idea is added to the algorithm. If a member of a control sequence violates its constraints, the control is reset at the nearest constraint boundary, making it impossible for any control member to violate the control constraints. This subroutine is called from all trajectory calculations, and especially from those in the line search. The restricted algorithm still uses gradient

information to compute the steepest direction, but it restricts the magnitude of that direction which can be added to each member of the nominal control sequence. It should be noted that the gradients are no longer zero along the optimal path and also that a jamming possibility exists.

Recall in the reduced jet engine problem that there are two control constraints. In addition we constrain the value of P_4 to be at least .5 to prevent negative values which would increase P_b and N arbitrarily. This constraint provides an equivalent problem to that solved by Dynamic Programming in chapter II.

3.5 Performance Index Function

A performance index function is used to convert the Fletcher-Reeves Method to solve a free time, fixed right end optimal control problem. Since, in the optimal time problem, the state variables are desired to reach x_F in minimum time, a performance index function, which penalizes the system for not being at x_F, takes the form

$$\sum_{t=0}^{N-1} L(x(t),u(t)) = \sum_{j=1}^{n} \sum_{t=0}^{N-1} c_j(x_j(t) - T_j)^2 \qquad (3.5\text{-}1)$$

where $x_F = (T_1,T_2,\ldots,T_n)$ and $c = (c_1,c_2,\ldots,c_n)$ are selected weighting constants. The squared criterion is chosen to provide a symmetric convex penalty function about the state x_F which is continuously differentiable. Initially the c_j's were assigned the same value but this choice is found to produce an oscillating time optimal solution, a condition

not suggested by the Dynamic Programming solution of chapter II.

To remove the oscillation effect and obtain an accurate time optimal open loop control sequence, the c_j's are chosen so they are in proportion to the time constants of the state variables, which can be obtained by linearizing the system about x_F. If we represent the A,B linear state description of small deviations about the design objective as

$$\frac{\partial \delta x}{\partial x} = A \delta x + B \delta u \qquad (3.5\text{-}2)$$

with

$$A = \frac{\partial f}{\partial x}(x_F, u_F) \qquad (3.5\text{-}3)$$

then the time constants are the inverse of the real parts of the eigenvalues of the Jacobian evaluated at x_F and u_F. In the jet engine problem, $T_N = .352$; $T_{P_b} = .0309$, and $T_{P_4} = .0123$ and the resulting c_j ratios become 1000:90:35. The slowest state variable, N, should have the higher penalty because it requires the most time to react. In the reduced problem, P_4 is the control and is not assigned a penalty to find a time optimal control sequence for the problem.

A prior knowledge of the approximate optimal time is needed to choose a good time step size and state variable array length. Chapter II has already shown that the jet engine can move from windmill to military thrust in .8 seconds so a $t = .005$ and an $N = 200$ are used in the program.

For the optimal time problem, the early states in the trajectory are given a small penalty with respect to that given to later states. This gives the system freedom to

reach the design objective. If we let t be the time step of interest and let N time steps exist, then the performance index functions become

$$L(x(t),u(t)) = N(\frac{t}{N})^9 \sum_{j=1}^{n} c_j(x_j(t) - T_j)^2 \quad (3.5\text{-}4)$$

and

$$K(x(N)) = N \sum_{j=1}^{n} c_j(x_j(N) - T_j)^2 \quad (3.5\text{-}5)$$

A point 90% along the trajectory is given a penalty only 38% of that received by the endpoint. Computational experience has revealed that for problems in which the time optimal control sequence lies along the constraint boundaries for most of the state variable trajectory, a much lower exponent must be used in (3.5-4) to provide higher earlier penalties. In run I of section 3.7, an original exponent of 9 produces a solution in which the jet engine idles at windmill for most of the allowed time and only approaches the design value near the end. When the exponent is lowered to two, the solution is a control sequence which rapidly accelerates the engine.

3.6 Parameter Sensitive Cubic Fit Line Search

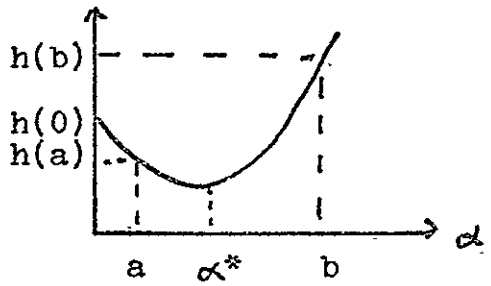
A quick, accurate line search to find

$$h(\alpha^*) = \min_{\alpha \geq 0} J(\underline{y} + \alpha \underline{d}) \quad (3.6\text{-}1)$$

is a vital component of the modified Fletcher-Reeves method. Lasdon in [9] has proposed a quadratic fit technique to find the best alpha. Also considered is a golden section line search but neither approach simultaneously allows a wide

range of possible α^* values and requires relatively few trajectory calculations to determine a test point h(a). The line search proposed here fits a cubic polynomial through three test points and with derivative information algebraically solves for α^* in one step.

Consider the general situation pictured in figure 3.2.



Conditions
1) $h(a) < h(0)$
2) $h(b) > h(0)$
3) $h'(0) < 0$
4) $0 \leq a < b$

Figure 3.2

Suppose that $h(\alpha)$ can be described by

$$h(\alpha) = h(0) + h'(0)\alpha + A\alpha^2 + B\alpha^3 \qquad (3.6\text{-}2)$$

and

$$\left.\frac{\partial h(\alpha)}{\partial \alpha}\right|_{\alpha=\alpha^*} = 0 \qquad (3.6\text{-}3)$$

gives α^* in terms of A,B,h(0), and h'(0) from the solution of the following equation.

$$h'(\alpha) = h'(0) + 2A\alpha + 3B\alpha^2 = 0 \qquad (3.6\text{-}4)$$

The positive solution of the quadratic formula solves the above equation to give

$$\alpha^* = \frac{-A + \sqrt{A^2 - 3Bh'(0)}}{3B} \qquad (3.6\text{-}5)$$

To find A and B, note that

$$h(a) = h(o) + h'(0)a + Aa^2 + Ba^3 \qquad (3.6\text{-}6)$$

$$h(b) = h(0) + h'(0)b + Ab^2 + Bb^3 \qquad (3.6\text{-}7)$$

and the expression for A is then obtained by multiplying $(a/b)^3$ to both sides of (3.6-7) and subtracting the result from (3.6-6).

$$A = h(a) - \frac{a^3}{b^3} h(b) + h(0)\left(\frac{a^3}{b^3} - 1\right) + h'(0)\left(\frac{a^3}{b^2} - a\right) \qquad (3.6\text{-}8)$$

$$B = \frac{h(a) - h(0) - h'(0)a - Aa^2}{a^3} \qquad (3.6\text{-}9)$$

The flow chart for the parameter sensitive cubic fit line search is illustrated in figure 3.3. The components of the line search are explained in the remainder of this section. Recall from the Fletcher-Reeves flow chart that $h(0)$ is already known and $h'(0)$ can be expressed as $\underline{g}^T\underline{d}$.

1) Selection of "a" Value

As an initial guess for "a", let

$$a\,\|\underline{d}\| = \frac{1}{10}\,\|\underline{u}\| \Longrightarrow a = \frac{1}{10}\,\frac{\|\underline{u}\|}{\|\underline{d}\|} \qquad (3.6\text{-}10)$$

with $\|\ \|$ denoting the usual norm of a vector. If the resulting h(a) satisfies condition 1 of figure 3.2, let $b = 5a$ and proceed to find a suitable "b" value; if not, divide "a" by ten and try condition 1 again. Sometimes when $J(\underline{u})$ is very close to the minimum value, "a" can get minutely small with no decrease in h(a). If this happens, set α^* to zero and continue with the main program.

Parameter Sensitive

Cubic Fit Line Search

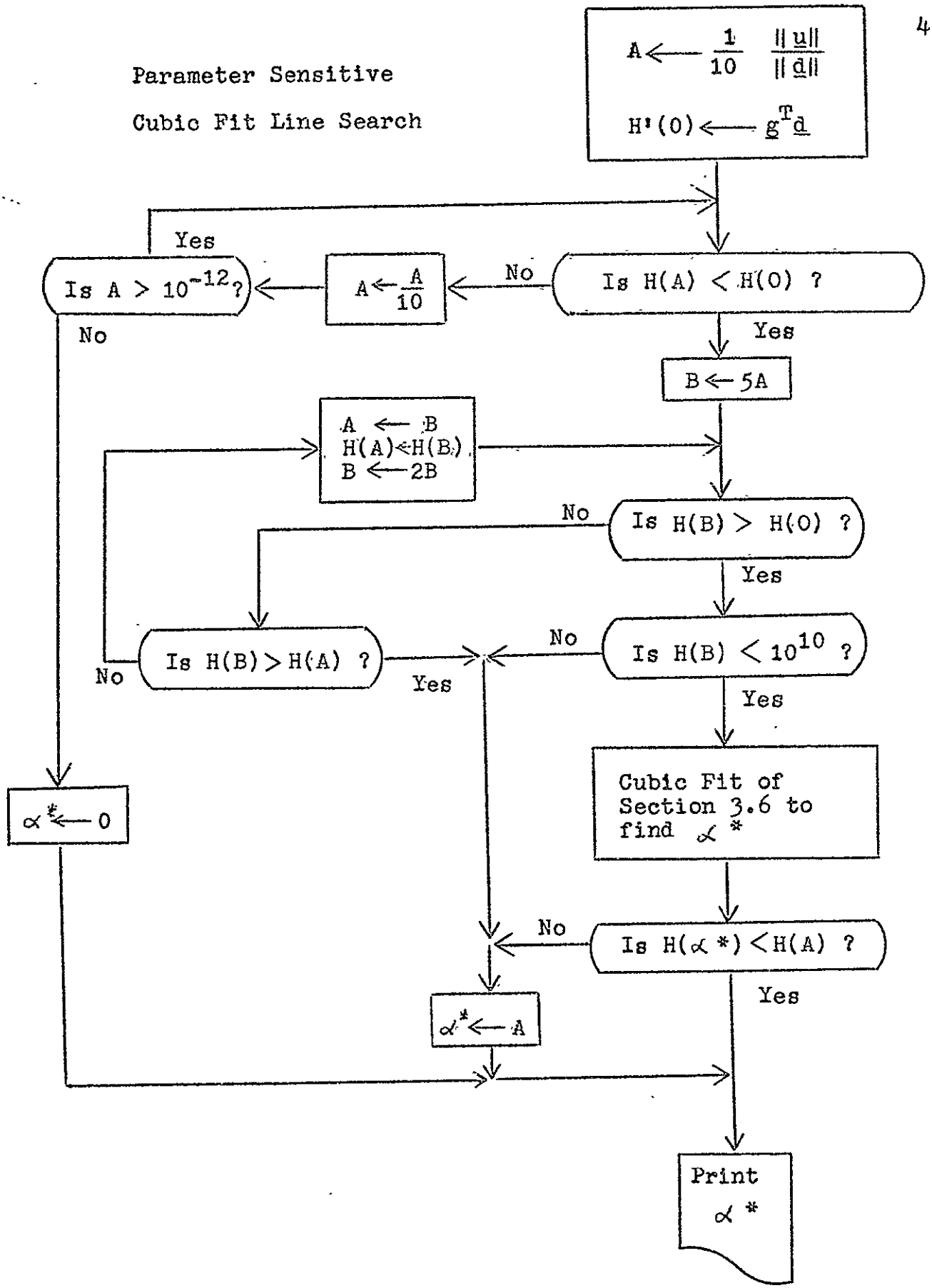


Figure 3.3

2) Selection of "b" Value

If the resulting h(b) satisfies condition 2 and has a reasonable value, use the cubic fit routine to find α^* with equations (3.6-5),(3.6-8), and (3.6-9). If h(b) is less than both h(0) and h(a), move "b" values into "a", let b = 2b and try condition 2 again. This step moves the value of "a" closer to the minimum point and improves the accuracy of the cubic fit. If condition 2 is not satisfied and h(b) is greater than h(a), let $\alpha^* = a$. Experience indicates that, in this case, the function $h(\alpha)$ has the shape indicated in figure 3.4. Since much computer time is spent to find a

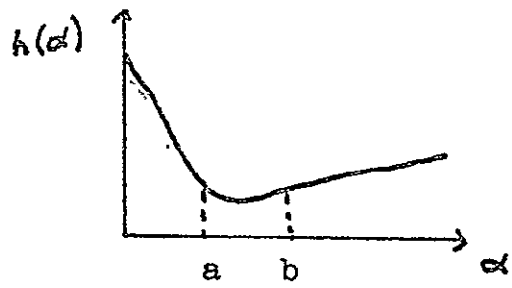


Figure 3.4

"b" which will satisfy condition 2, and since both of the values of "a" and "b" are close to the minimum point, let $\alpha^* = a$. It must be noted that in the jet engine problem, this case only appears at the first iteration when the trajectory approaches the control constraints. After the cubic fit, there is a final test to check that $h(\alpha^*)$ is actually lower than h(a), which will be the usual case.

In the jet engine problem, this line search works well and always improves the performance index with each iteration.

There seem to be many tests in the line search of figure 3.3, however the answer is yes to all tests in the usual case and only three or four trajectory calculations are required to find an accurate α^*. Another advantage is the wide range of values which α^* can assume. It ranges from 10^{-11} to .0015 in the jet engine problem or eight orders of magnitude.

3.7 Second Order Jet Engine Study

In solving the jet engine problem and comparing the two computational methods of this thesis, trajectories resulting from two important sets of initial conditions are studied by the Modified Fletcher-Reeves Conjugate Gradient Method. From chapter II, it is already learned that the jet engine can be controlled from windmill to military thrust in close to .8 seconds, so for this case, the state variable arrays will have a length of 200 and a time step size of .005. From figure 2.4-A, an initial state (P_b=1.774,N=.9) is seen to require about .3 seconds so a state variable array length of 200 is used with a time step of .002 for this problem.

A poor nominal constant control of P_4 = .6 which decreases the rotational speed and increases the burner density is used in the non-linear programs because this nominal control, unable to influence the solution, yields a true test of the feasible directions modification and free time, fixed right end extension. The number of inner iterations per outer iteration is equal to the array size of the variables. Outer iterations are run until the performance index no longer decreases. If the line search produces an α^* equal to zero,

no further improvement can be made until a new gradient direction is calculated. Therefore, significant c.p.u. time can be conserved by skipping the remaining inner iterations when an α^* equal to zero is produced. Table 3.1 tabulates the action of the performance index as outer iterations of the three significant programs are run. The vast improvement in the first iteration represents the rapid action away from the poor initial control sequence toward a much improved open loop control law. Further iterations refine the optimal solution and only slightly lower the performance index. An illustration of this is the bang-bang portion of run III, in which the time plot of the control sequence assumes a rounded shape in the earlier iterations and refines later to a rectangular shape.

Run I directly attacks the jet engine problem posed in section 1.4 and produces very encouraging results. A close inspection of the time response in figure 3.5 reveals that the control sequence follows the control constraints throughout most of the trajectory. Near the end, the control is reduced and increased but not with the rapid switching action suggested by the dynamic programming solution. The author hypothesizes that the algorithm has jammed here because the optimal direction is calculated with no awareness of the control constraints and also the gradients along those constraints do not approach zero. However, a great part of the solution has been calculated with run I and it agrees with that in chapter II. Figure 3.6 reveals that run II

ITERATIONS REQUIRED AND RESULTING

PERFORMANCE INDEX

	OUTER ITERATION #	PERFORMANCE INDEX
Run I $x = (P_b=1.774, N=.5461)$		$t = .005$
	0	2639895.00
	1	107942.56
	2	107791.62
	3	107725.00
	4	107724.44
	5	no change
Run II $x = (P_b=1.774, N=.9)$		$t = .002$
	0	2556673.00
	1	333.52
	2	328.07
	3	327.53
	4	no change
Run III $x = (P_b=.7833, N=.9853)$		$t = .001$
	0	88460.69
	1	101.49
	2	90.17
	3	89.22
	4	no change

Table 3.1

Time Response of Reduced Jet Engine

Obtained from Run I

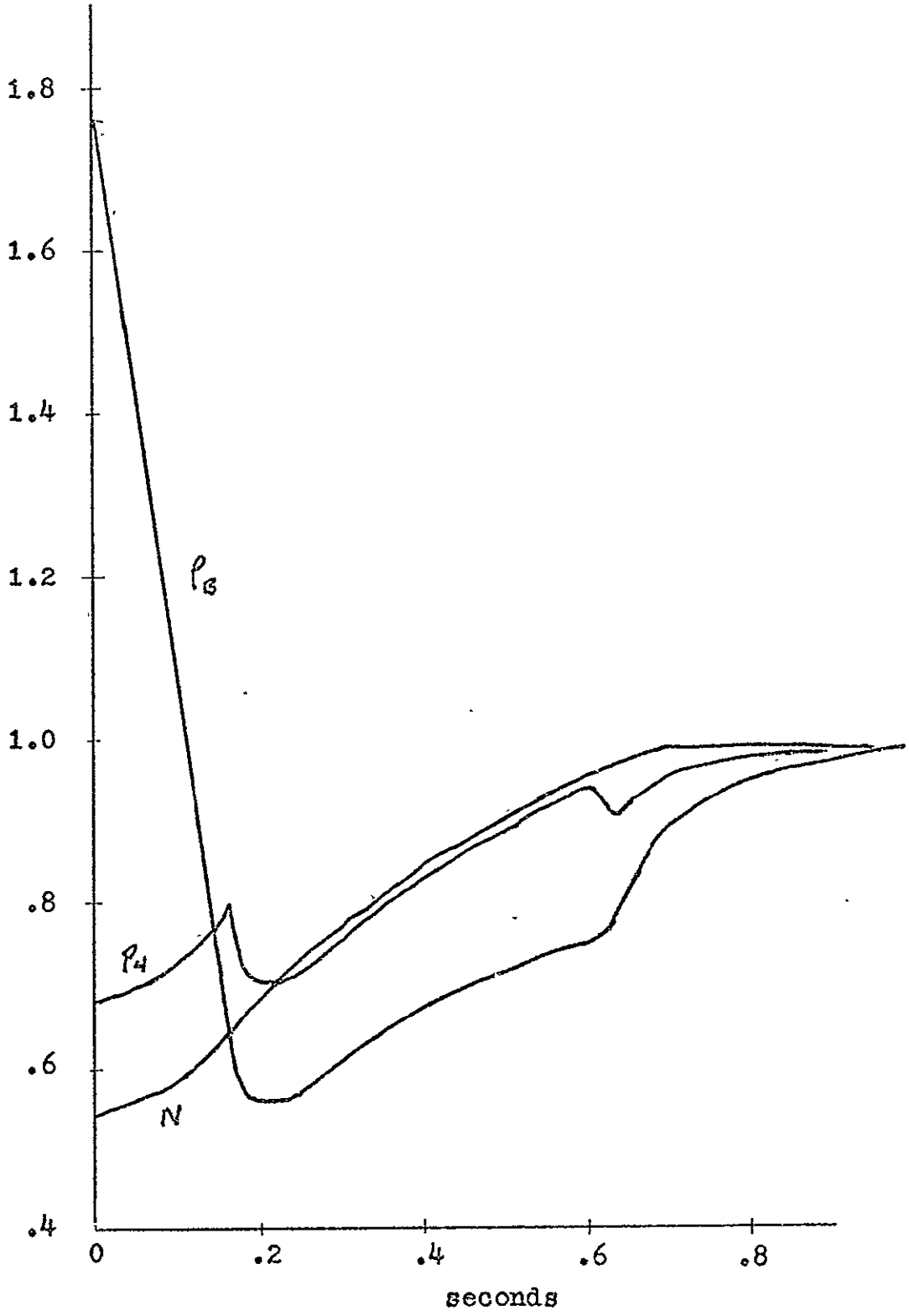


Figure 3.5

Time Response of Reduced Jet Engine

Obtained from Run II

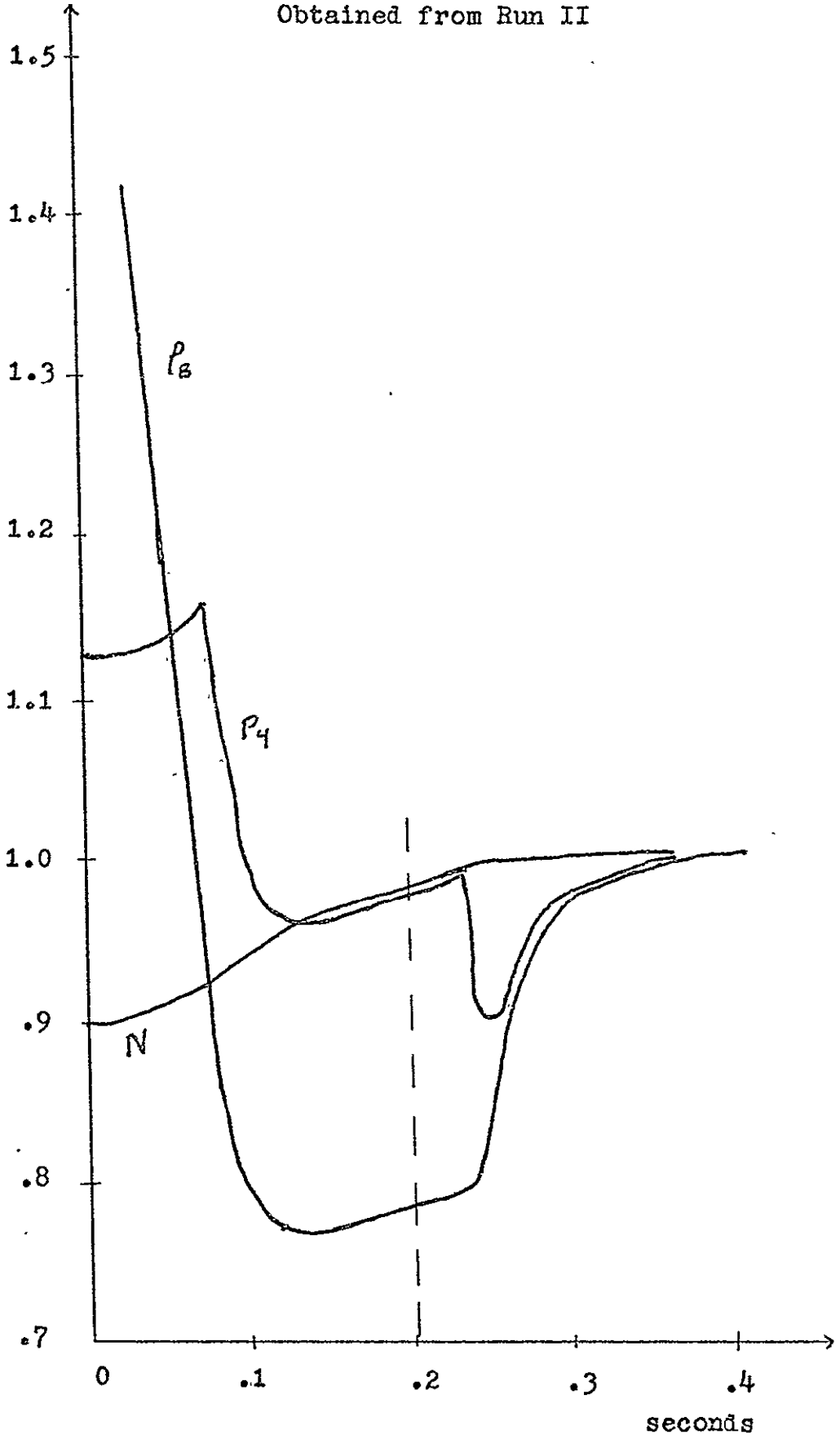


Figure 3.6

has produced a similar situation and has also defined a state variable path which joins that formed in run I. This occurrence verifies the existence of the common trajectory of Dynamic Programming's region 2 which is seen in figure 2.4-A.

The initial conditions of run III are obtained by assuming that the portion of the solution, of which the control values lie on the constraints, is valid and by choosing a point near the end of the boundary as an initial condition for another run. The new state variable array dimension is 100 and the time step size is .001. The time response plot in figure 3.7 reveals a bang-bang control sequence which is faster than that in figure 3.5 and 3.6 and is similar to the solution of chapter II. It shows that P_4 switches directly to the $P_{4\ min}$ constraint from the turbine inlet temperature constraint. This later sequence added to the control sequence obtained from the first part of run I produces an open loop control sequence which is most similar to that of chapter II.

The methods and results of section 2.9 can again be employed to calculate the third order time optimal control sequence which is already graphed in figure 2.8. Since the reduced solutions are identical, so are the third order solutions.

In figure 3.8, the discontinuous control law for the two methods is shown.

Time Response of Reduced Jet Engine

Obtained from Run III

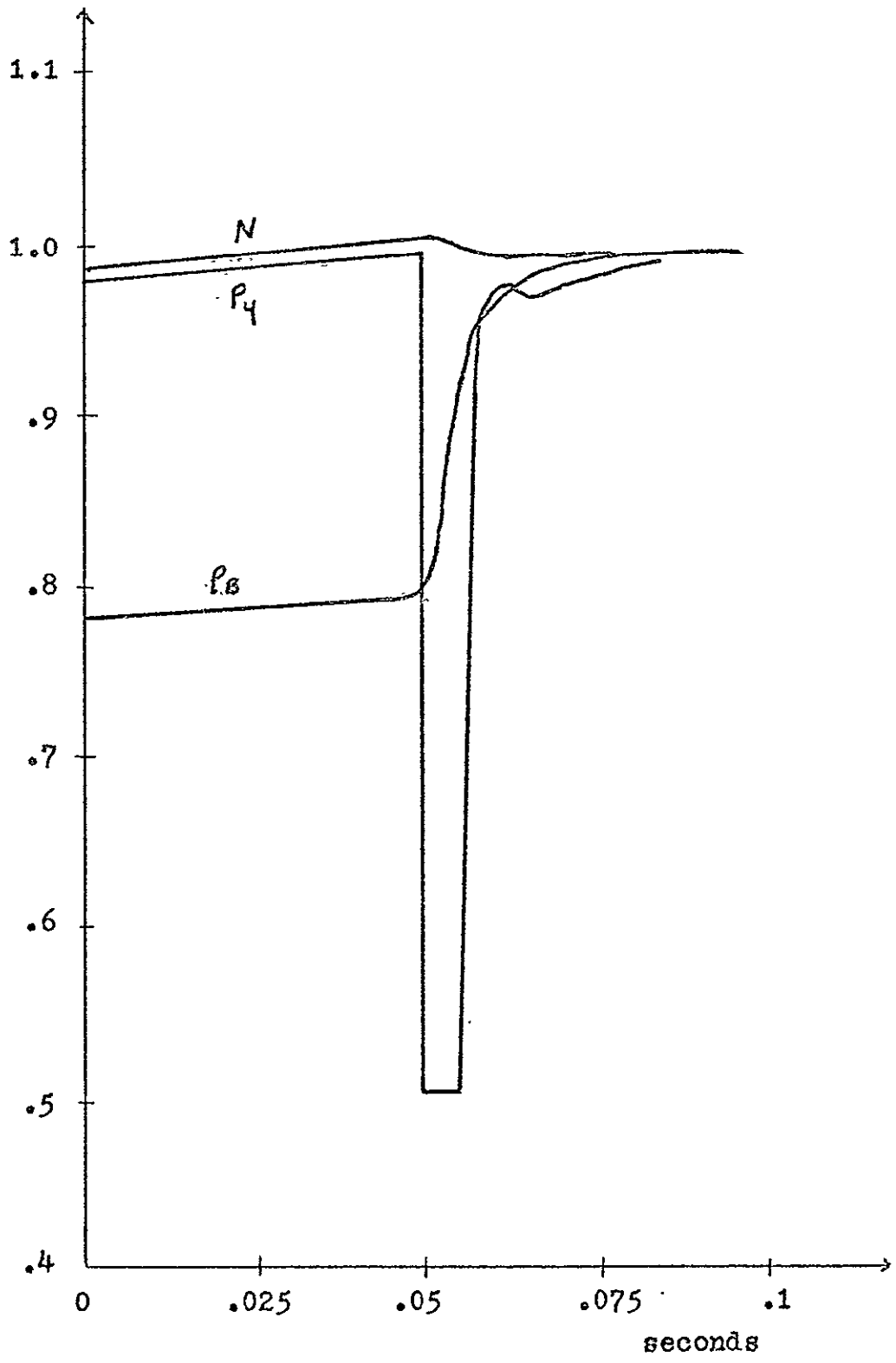


Figure 3.7

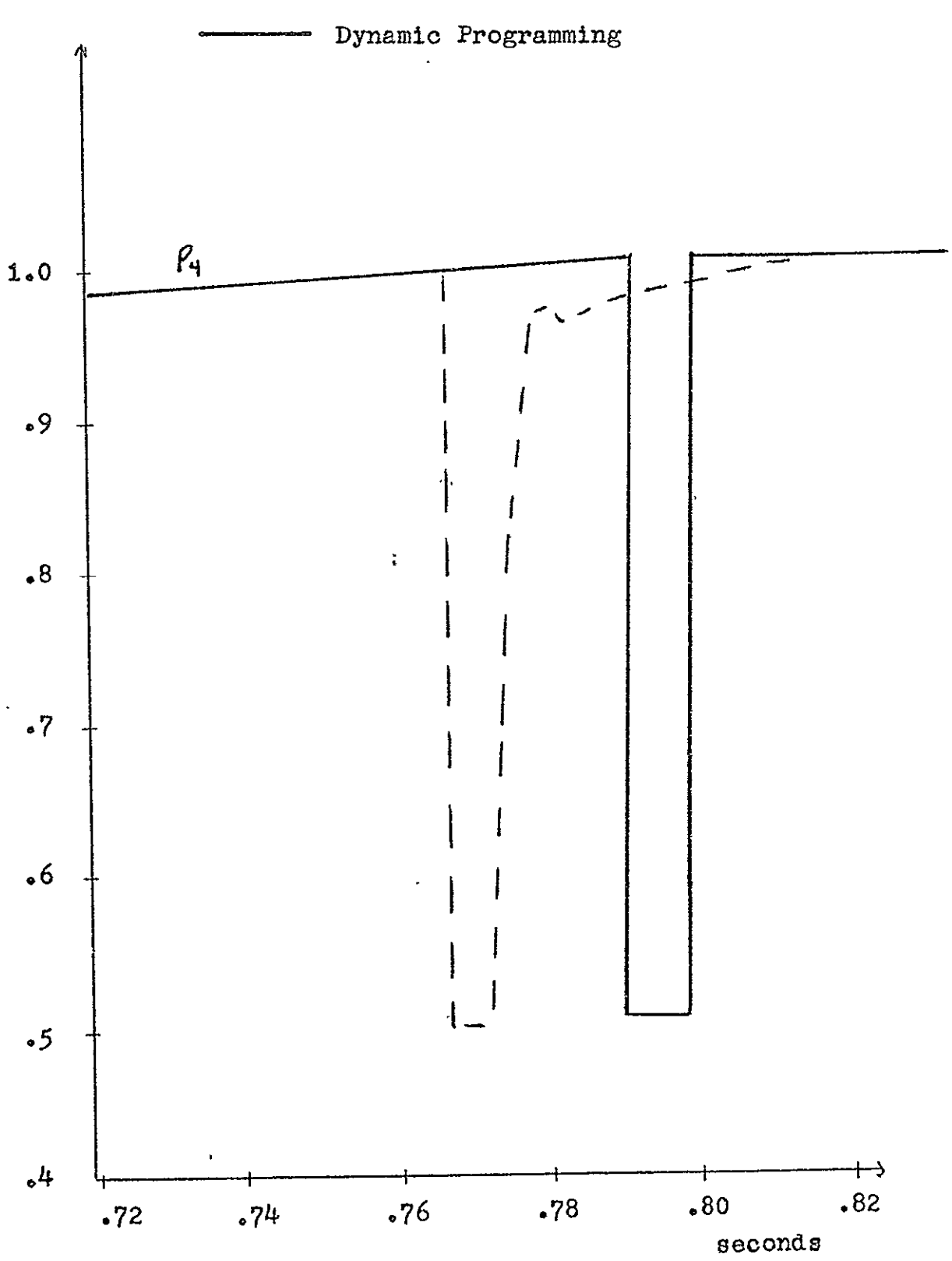

Two Bang-Bang Control Laws
Fletcher-Reeves Method
Dynamic Programming
p_4
1.0
.9
.8
.7
.6
.5
.4
.72
.74
.76
.78
.80
.82
seconds

Figure 3.8

Optimal Time Responses

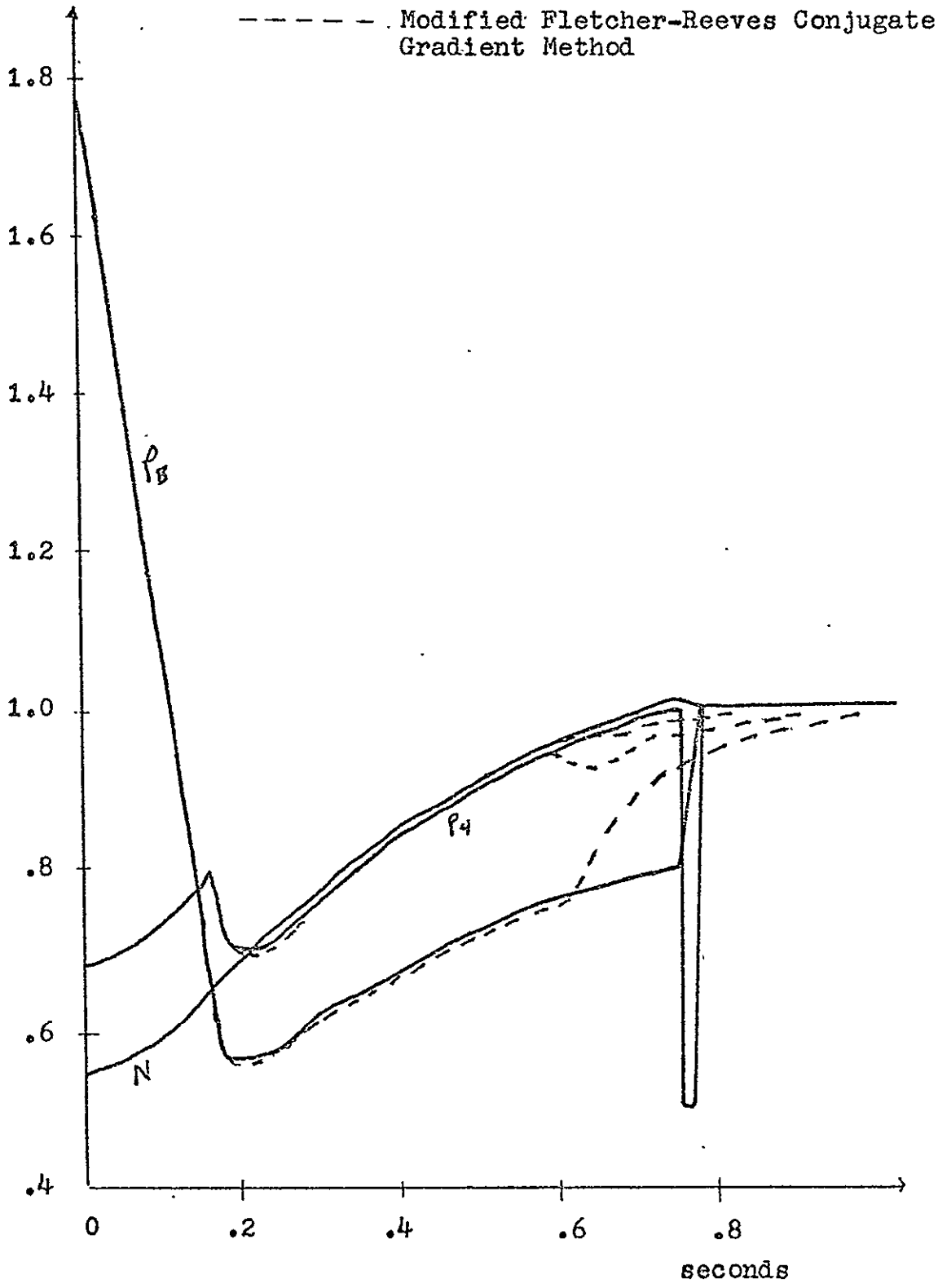


Figure 3.9

CHAPTER IV

CONCLUSION

Two well known computational methods, Dynamic Programming and the Fletcher-Reeves Conjugate Gradient Method, are successfully adapted in this thesis to solve the general class of fixed right end, free time optimal control problems with state variable and control constraints. Both methods have determined similar time optimal open loop control sequences for the discretized version of the reduced jet engine problem. These solutions have bang-bang features and also control values satisfying constraints with equality.

A successive approximations technique extends the capabilities of Dynamic Programming to produce a time optimal feedback control law which has described, in the jet engine problem, several regions in the state space map for which the controls follow a common rule. Separating these regions are distinct boundaries whose locations are approximately indicated on the discretized state variable grid. Unfortunately, the number of required computations increases geometrically with the order of the system and there is a problem with picturing a multi-dimensional feedback control law.

A feasible directions idea, used in conjunction with penalty functions, extends the Fletcher-Reeves Conjugate Gradient Method to produce a time optimal open loop control sequence for a fixed right end optimal control problem with state variable and control constraints. The solution of the reduced jet engine problem again shows the control values

lying on the control constraints for most of the state variable trajectory. Using a point near the end of the constraint region as an initial condition, a discontinuous bang-bang control sequence, identical to that suggested by the feedback control law of Dynamic Programming, is obtained. This method is well suited to higher order problems because the number of computations required to solve higher order systems increases only arithmetically with the system order.

The computer software which produced the main results of this thesis has been generalized to solve any general problem in this class. In the appendix, the software programs are listed and documented for general use to solve fixed right end, free time optimal control problems with state variable and control constraints.

APPENDIX A

This appendix lists and documents the software which implements the successive approximations technique to Dynamic Programming. It is written with the reduced jet engine equations but it is carefully documented to describe a straightforward extension to the general constrained time optimal control problem. The flow chart appears in figures 2.3-A and 2.3-B.

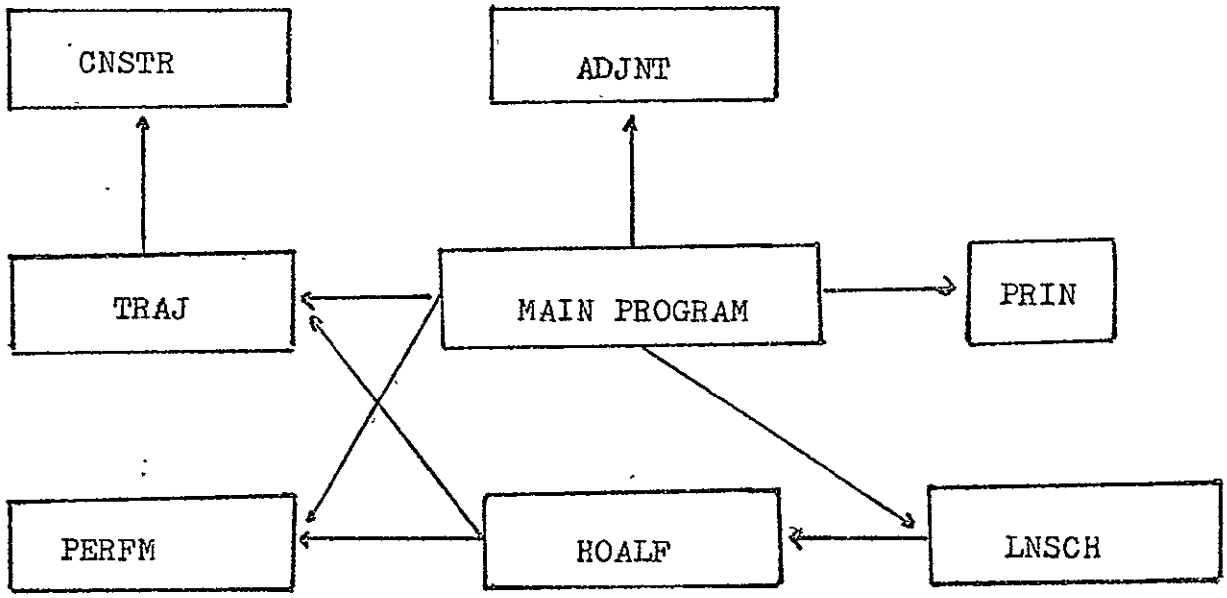
```
C
CCC     DYNAMIC PROGRAMMING SOLUTION FOR SECOND ORDER SYSTEM
CCC     THIS PROGRAM ALLOWS AN ARBITRARY SELECTION OF THE STATE SPACE
CCC     TO BE STUDIED
C
        DIMENSION VNEW(30,30),VOLD(30,30),UOPT(30,30),ALIST(30)
C
CCC     NITT=DESIRED NO. OF ITERATIONS
CCC     DELTA IS THE DESIRED TIME STEP
CCCCC
CCCCC   PB IS THE BURNER DENSITY STATE VARIABLE X1
CCCCC   EN IS THE ROTOR SPEED STATE VARIABLE X2
CCCCC   P4 IS THE BURNER PRESSURE USED AS THE CONTROL IN THIS SYSTEM.
CCCCCC
CCCCC   FST IS THE FIRST POINT IN THE ALLOWABLE STATE SPACE
CCCCC   LST IS THE LAST POINT IN THE ALLOWABLE STATE SPACE
CCCCC   INC IS THE INCREMENT IN THE STATE SPACE
CCCCC   NOTE, RESTRAIN TO 30 POINTS PER VARIABLE,
CCC     THE TARGET POINTS MUST BE ON THE GRID
C
        ENFST=0.98
        ENLST=1.05
        ENINC=0.0025
        PBFST=0.6
        PBLST=1.5
        PBINC=0.05
        P4FST=0.75
        P4LST=1.25
        P4INC=0.125
        P4INC=P4INC/2.
        DELTA=0.001
        NITT=130
C
CCC     ENTER THE TARGET POINTS
C
        PBTAR=1.
        ENTAR=1.
        EP=1.E-6
        NEN=IFIX((ENLST-ENFST+EP)/ENINC)+1
        NPB=IFIX((PBLST-PBFST+EP)/PBINC)+1
        NP4=IFIX((P4LST-P4FST+EP)/P4INC)+1
        NEN1=IFIX((ENTAR-ENFST+EP)/ENINC)+1
        NPB1=IFIX((PBTAR-PBFST+EP)/PBINC)+1
        NNEN=NEN+1
        NNPB=NPB+1
C
CCC     NSP IS THE STARTING POINT FOR THE PB PRINTOUT
CCC     NEP IS THE END POINT FOR THE PB PRINTOUT
C
        NSP=1
        NEP=NPB
C
CCC     INITIALIZE THE PENALTY FUNCTION - MAKE AS LOW AS POSSIBLE
C
        DO 10 I=1,NNEN
        DO 10 J=1,NNPB
   10   VOLD(I,J)=1./DELTA
        VOLD(NEN1,NPB1)=0.
        DO 15 I=1,NPB
   15   ALIST(I)=PBINC*(FLOAT(I-1))+PBFST+EP
C
CCC     ITERATION INDEX
C
        DO 20 N=1,NITT
C
CCC     STATE VARIABLE INDEX
C
        DO 30 I=1,NEN
        EN=ENINC*(FLOAT(I-1))+ENFST
        DO 30 J=1,NPB
        PB=PBINC*(FLOAT(J-1))+PBFST
```

```
      NOT=0
      VOTES=1./DELTA+10.
C
CC    CONTROL INDEX
CC    DISCARD UNALLOWABLE CONTROLS
C
      DO 40 K=1,NP4
      P4=P4INC*(FLOAT(K-1))+P4FST
C
CC    THE CONTROL CONSTRAINTS ARE INDICATED HERE
C
      IF(P4.GT.1.25001*EN) GO TO 40
      IF(P4.GT.1.25001*PB) GO TO 40
C
CC    CALCULATE THE NEXT STATE AND DISCARD UNALLOWABLE CONTROLS
CC    THE STATE EQUATIONS HERE ARE FOR THE REDUCED JET ENGINE PROBLEM
C
      Z=SQRT(P4*P4+0.41688*EN*EN-0.0899*P4*EN)
      W3DOT=1.3009*EN-0.139825*P4-0.13982*Z
      PBT=PB+DELTA*(37.78*W3DOT-38.448*P4+0.66849)
      IF(PBT.GT.PBLST+EP) GO TO 40
      IF(PBT.LT.PBFST-EP) GO TO 40
      ENT=EN+DELTA*1.258/EN*(P4*P4/PB-W3DOT*EN*EN)
      IF(ENT.GT.ENLST+EP) GO TO 40
      IF(ENT.LT.ENFST-EP) GO TO 40
      NOT=NOT+1
C
CC    INTERPOLATE TO FIND CORRESPONDING PENATY
C
      AI=(FLOAT(NEN-1)*ENT-ENFST*FLOAT(NEN)+ENLST)/(ENLST-ENFST)+EP
      BJ=(FLOAT(NPB-1)*PBT-PBFST*FLOAT(NPB)+PBLST)/(PBLST-PBFST)+EP
      ICOR=AI
      JCOR=BJ
      A=AI-FLOAT(ICOR)
      B=BJ-FLOAT(JCOR)
      ONE=VOLD(ICOR,JCOR)
      TWO=VOLD(ICOR,JCOR+1)
      THREE=VOLD(ICOR+1,JCOR)
      FOUR=VOLD(ICOR+1,JCOR+1)
      V=(1.-A)*(1.-B)*ONE+(1.-A)*B*TWO+A*(1.-B)*THREE+A*B*FOUR
C
CC    DETERMINE IF THIS CONTROL IS BETTER
C
      VTEST=1.+V
      IF(VTEST.GT.VOTES) GO TO 40
      UOTES=P4
      VOTES=VTEST
   40 CONTINUE
C
CC    FORM VNEW MATRIX FROM NEW DATA
C
      IF(NOT)31,32,31
   31 IF(VOTES.GT.1./DELTA-EP) GO TO 32
      UOPT(I,J)=UOTES
      VNEW(I,J)=VOTES
      GO TO 30
C
CC    NOTATION FOR THE UNCONTROLLABLE STATE AND FEEDBACK CONTROL LAW
C
   32 UOPT(I,J)=0.
      VNEW(I,J)=1./DELTA
   30 CONTINUE
C
CC    RESET TARGET STATE PENALTY TO 0
C
      VNEW(NEN1,NPB1)=0.
C
CC    MOVE VNEW INTO VOLD FOR NEXT ITERATION
C
      DO 50 I=1,NEN
      DO 50 J=1,NPB
```

```
   50 VOLD(I,J)=VNEW(I,J)
   20 CONTINUE
C
C     PRINT ROUTINE
C
      WRITE(6,100)NITT
      WRITE(6,101)(ALIST(I),I=NSP,NEP)
      WRITE(6,103)
      DO 33 K=1,NEN
      KK=NEN+1-K
      XD=ENLST-ENINC*FLOAT(K-1)
      WRITE(6,102)XD,(VOLD(KK,I),I=NSP,NEP)
      WRITE(6,101)(UOPT(KK,I),I=NSP,NEP)
      WRITE(6,103)
   33 CONTINUE
      WRITE(6,101)(ALIST(I),I=NSP,NEP)
  100 FORMAT(5X,'TIME STEP NUMBER',I4)
  101 FORMAT(6X,19F6.3)
  102 FORMAT(1X,F5.3,19F6.1)
  103 FORMAT(1X,'-')
      STOP
      END
```

APPENDIX B

This appendix contains detailed software of the Modified Fletcher-Reeves Conjugate Gradient Method. A chart showing the relation of the main program with seven specialized subroutines is given below and the listings and documentations of the actual software are given on the following pages.



```
      DIMENSION U(200,3),US(200,3),X(201,4),XS(201,4),
     ?G(200,3),GS(200,3),D(200,3),ALIST(201)
      COMMON PCON(4),T(4),DELTA,KOUT,IND,NC,NX,NPA
C
C        READ IN INITIAL DATA
C        CARD 1 - NC   = NUMBER OF CONTROLS
C                 NX   = NUMBER OF STATE VARIABLES
C                 NPA  = NUMBER OF POINTS PER ARRAY
C                 NOUT=MAXIMUM NUMBER OF TOTAL ITERATIONS
C        CARD 2 - INITIAL CONDITIONS FOR EACH STATE VARIABLE
C        CARD 3 - PENALTY CONSTANT FOR EACH STATE VARIABLE
C        CARD 4 - DESIGN OBJECTIVE FOR EACH STATE VARIABLE
C        CARD 5 - TIME STEP SIZE
C
      READ(5,*)NC,NX,NPA,NOUT
      READ(5,*)(X(1,J),J=1,NX)
      READ(5,*)(PCON(J),J=1,NX)
      READ(5,*)(T(J),J=1,NX)
      READ(5,*) DELTA
      DO 20 J=1,NX
   20 XS(1,J)=X(1,J)
C
C        INDEX IS THE TOTAL ITERATION COUNTER
C        KOUT IS THE OUTER ITERATION COUNTER
C
      INDEX=0
      KOUT=0
      PTEST=1.E10
      EP=1.E-4
C
C        ASSIGN A NOMINAL CONTROL
C
      DO 1 K=1,NC
      DO 1 J=1,NPA
    1 U(J,K)=0.7
    2 CONTINUE
C
C        IND IS THE INNER ITERATION COUNTER
C
      IND=0
      KOUT=KOUT+1
C
C        COMPUTE THE STATE VARIABLE TRAJECTORY AND THE PERFORMANCE INDEX
C
      CALL TRAJ(X,U)
      CALL PERFM(X,U,PER)
C
C        COMPUTE THE GRADIENTS
C
      CALL ADJNT(X,U,G)
      CALL PRIN(X,U,G,PER,INDEX)
C
C        DIRECTION IS THE NEGATIVE OF THE GRADIENT
C
      DO 3 K=1,NC
      DO 3 J=1,NPA
    3 D(J,K)=-G(J,K)
    4 CONTINUE
      INDEX=INDEX+1
C
C        LINE SEARCH TO FIND THE ALPHA WHICH MINIMIZES
C        H(ALPHA) = J(U+ALPHA*D)
C
      CALL LNSCH(X,U,D,G,ALPHA,PER)
C
C        IF ALPHA IS VERY SMALL, PROCEED TO THE NEXT CUTER ITERATION
C
      IF(ALPHA.LT.1.E-12) INDEX=KOUT*NPA
      IF(ALPHA.LT.1.E-12) IND=NPA
C
C        US = U + ALPHA*D
C
      DO 5 K=1,NC
      DO 5 J=1,NPA
    5 US(J,K)=U(J,K)+ALPHA*D(J,K)
C
C        COMPUTE NEW TRAJECTORY AND PERFORMANCE INDEX
C
      CALL TRAJ(XS,US)
      CALL PERFM(XS,US,PER)
```

```
      IF(KOUT.GT.1) GO TO 6
      WRITE(6,13) PER
    6 CONTINUE
C
CC       COMPUTE NEW GRADIENTS
C
      CALL ADJNT(XS,US,GS)
C
CC       COMPUTE BETA
C
      BN=0.
      BD=0.
      DO 7 K=1,NC
      DO 7 J=1,NPA
      BN=BN+GS(J,K)*GS(J,K)
    7 BD=BD+G(J,K)*G(J,K)
      BETA=BN/BD
C
CC       ASSIGN NEW DIRECTION FOR LINE SEARCH
C
      DO 8 K=1,NC
      DO 8 J=1,NPA
    8 D(J,K)=-GS(J,K)+BETA*D(J,K)
C
CC       MORE THAN MAXIMUM NUMBER OF TOTAL ITERATIONS ?
C
      IF(NOUT.LT.INDEX) GO TO 11
C
CC       MORE THAN MAXIMUM NUMBER OF INNER ITERATIONS ?
C
      IF(NOUT.LT.NPA) GO TO 9.
C
CC       DOES PERFORMANCE INDEX IMPROVE ?
C
      IF(PTEST-PER.LT.EP) GO TO 11
      PTEST=PER
      GO TO 2
C
CC       MOVE NEW GRADIENT INTO OLD GRADIENT
C
    9 DO 10 K=1,NC
      DO 10 J=1,NPA
   10 G(J,K)=GS(J,K)
      GO TO 4
C
CC       PRINT AND PLOT OPTIMAL TRAJECTORY
C
   11 KOUT=0
      CALL PRIN(XS,US,GS,PER,INDEX)
      DO 12 J=1,NPA
   12 ALIST(J)=FLOAT(J)
      CALL PLOTA(2,ALIST,US,NPA,NPA,NC)
      N=NPA+1
      CALL PLOTA(2,ALIST,XS,N,N,NX)
   13 FORMAT('+',65X,'PERFORMANCE INDEX IS ',F15.4)
      STOP
      END
```

```
      SUBROUTINE ADJNT(X,U,G)
      DIMENSION X(201,4),U(200,3),G(200,3),Y(4),YN(4),DLDX(4),DFX(4,4),
     ?DFU(4,3)
      COMMON PCON(4),T(4),DELTA,KOUT,IND,NC,NX,NPA
      N=NPA+1
C
CC        INITIALIZE Y(K)
C
      DO 1 K=1,NX
    1 Y(K)=2.*PCON(K)*(X(N,K)-T(K))*FLOAT(N)
C
CC        CALCULATE THE ADJOINT SYSTEM IN REVERSE TIME
C
      DO 2 J=1,NPA
      K=NPA+1-J
C
CC        DEFINE ALL DERIVATIVES OF THE FORM
CC        DFX(NX,NX) AND DFU(NX,NC)
CC
CC        FOLLOWING ARE DERIVATIVES FOR THE SECOND ORDER JET ENGINE MODEL
C
      DEN=SQRT(U(K,1)**2+0.41688*X(K,2)**2-0.0899*U(K,1)*X(K,2))
      W=1.3009*X(K,2)-0.139825*U(K,1)-0.13982*DEN
      DWDU=-0.139825-0.06691*(2.*U(K,1)-0.0899*X(K,2))/DEN
      DWDX2=1.3009-0.06691*(0.83376*X(K,2)-0.0899*U(K,1))/DEN
      DFX(1,1)=0.
      DFX(1,2)=37.78*DWDX2
      DFX(2,1)=-1.258*((U(K,1)/X(K,1))**2/X(K,2))
      DFX(2,2)=-1.258*((U(K,1)/X(K,2))**2/X(K,1)+W+X(K,2)*DWDX2)
      DFU(1,1)=37.78*DWDU-38.448
      DFU(2,1)=1.258*(2.*U(K,1)/(X(K,1)*X(K,2))-X(K,2)*DWDU)
C
CC        CALCULATE DLX(NX)
C
      DO 3 I=1,NX
    3 DLDX(I)=2.*PCON(I)*(X(K,I)-T(I))*FLOAT(NPA)*(FLOAT(K)/FLOAT(NPA)**
     ?9)
C
CC        CALCULATE PRESENT GRADIENT
C
      DO 4 I=1,NC
      G(K,I)=0.
      DO 4 M=1,NX
    4 G(K,I)=G(K,I)+Y(M)*DFU(M,I)*DELTA
C
CC        CALCULATE NEXT Y(K)
C
      DO 5 M=1,NX
      YN(M)=Y(M)
      DO 5 I=1,NX
    5 YN(M)=YN(M)+Y(I)*DFX(I,M)*DELTA
      DO 6 I=1,NX
      YN(I)=YN(I)+DLDX(I)+Y(I)
    6 Y(I)=YN(I)
    2 CONTINUE
      RETURN
      END
```

Subroutine ADJNT computes the gradient arrays of a nominal trajectory. The theory of the adjoint system equations is presented in section 3.2. The user is required to calculate and define in ADJNT all derivatives of the form

DFX(J,K) J=1,2,...,NX K=1,2,...,NX

DFU(J,K) J=1,2,...,NX K=1,2,...,NC

where

$$DFX(J,K) = \frac{\partial F_j}{\partial x_k}; \qquad DFU(J,K) = \frac{\partial F_j}{\partial u_k}$$

The derivatives must be expressed in the form which is illustrated by the jet engine problem example. The portion of ADJNT which calculates the adjoint system and the gradients is completely general and need not be altered by the user.

INPUTS:

U - the control arrays

X - the state variable arrays

OUTPUT:

G - the resulting gradient arrays

```
SUBROUTINE CNSTR(X,UTRY,U)
DIMENSION X(4),UTRY(3),U(3)
COMMON PCON(4),T(4),DELTA,KOUT,IND,NC,NX,NPA
IF(UTRY(1).GT.1.25*X(1)) UTRY(1)=X(1)
IF(UTRY(1).GT.1.25*X(2)) UTRY(1)=X(2)
IF(UTRY(1).LT.0.5) UTRY(1)=0.5
U(1)=UTRY(1)
RETURN
END
```

Subroutine CNSTR tests a state variable n-tuple to determine the legality of the control, which is reset at the nearest legal boundary if the control is in an unallowable region.

This is a specialized subroutine which implements the feasible directions modification. The user simply states his linear constraints in a similar manner. CNSTR is called only from the subroutine TRAJ.

INPUTS:

UTRY - the present controls to be tested

X - the present state variable n-tuple

OUTPUT:

U - the resulting allowable controls

```
      SUBROUTINE HOALF(X,U,D,ALP,H)
      DIMENSION X(201,4),U(200,3),D(200,3),UTRY(200,3)
      COMMON PCON(4),T(4),DELTA,KOUT,IND,NC,NX,NPA
      DO 1 K=1,NC
      DO 1 J=1,NPA
    1 UTRY(J,K)=U(J,K)+ALP*D(J,K)
      CALL TRAJ(X,UTRY)
      CALL PERFM(X,UTRY,H)
      IF(KOUT.GT.1) GO TO 2
      IF(IND.GT.5) GO TO 2
      WRITE(6,3) ALP,H
    2 CONTINUE
    3 FORMAT(5X,'ALPHA = ',E15.5,' H(ALPHA) = ',F15.4)
      RETURN
      END
```

Subroutine HOALF calculates $h(a) = J(\underline{u} + a\underline{d})$. It is called from LNSCH to determine the performance index resulting from possible control arrays. This subroutine is completely general and independent of the particular problem to be solved. It need not be altered by the user.

INPUTS:

U - the nominal control array

X - the nominal state variable arrays

D - the direction along which the line search is made

ALP - the present distance tested along D

OUTPUT:

H - the resulting performance index

```
      SUBROUTINE LNSCH(X,U,D,G,ALPHA,PER)
C        LNSCH IS A LINE SEARCH SUBROUTINE WHICH UTILIZES A CUBIC POLYNOMIAL
CC       FIT TECHNIQUE TO DETERMINE A MINIMUM POINT.
CC       IN ADDITION, IT IS SENSITIVE TO INFORMATION OBTAINED FROM
CC       CALCULATING THE PARAMETERS NEEDED FOR THE CUBIC FIT
C
      DIMENSION X(201,4),U(200,3),D(200,3),G(200,3)
      COMMON PCON(4),T(4),DELTA,KOUT,IND,NC,NX,NPA
      IND=IND+1
C
CC       DETERMINE THE DERIVATIVE (H'(0)) AND THE INITIAL 'A' GUESS
C
      HPO=0.
      AN=0.
      AD=0.
      DO 1 K=1,NC
      DO 1 J=1,NPA
      HPO=HPO+G(J,K)*D(J,K)
      AN=AN+U(J,K)*U(J,K)
    1 AD=AD+D(J,K)*D(J,K)
      A=SQRT(AN/AD)/10.
      HO=PER
    2 CONTINUE
      CALL HOALF(X,U,D,A,HA)
C
CCC      TEST THE RELATIONSHIP BETWEEN H(A) AND H(0)
C
      IF(HO.GT.HA) GO TO 4
C
CCC      MAKE A BETTER GUESS FOR 'A'
C
      A=A/10.
      IF(A.LT.1.E-12) GO TO 3
      GO TO 2
    3 ALPHA=0.
      GO TO 8
    4 B=5.*A
    5 CONTINUE
      CALL HOALF(X,U,D,B,HB)
C
CCC      TEST FOR RELATIONSHIP BETWEEN H(B) AND H(0), ALSO H(B) AND H(A)
C
      IF(HB.GT.HO) GO TO 6
      IF(HB+0.1.GT.HA) GO TO 7
C
CCC      MOVE 'B' INTO 'A' AND MAKE A BETTER GUESS FOR 'B'
C
      A=B
      HA=HB
      B=2.*B
      GO TO 5
C
CCC      IF 'B' IS A REASONABLE VALUE, USE THE CUBIC FIT TECHNIQUE
CC       TO FIND THE MINIMUM ALPHA
C
    6 IF(HB.GT.1.E10) GO TO 7
      AC=HA-HB*(A/B)**3+HO*((A/B)**3-1.)+HPO*(A*(A/B)**2-A)
      AC=AC/(A**2-A**3/B)
      BC=(HA-HO-HPO*A-AC*A*A)/A**3
      ALPHA=(SQRT(AC*AC-3.*BC*HPO)-AC)/(3.*BC)
C
CCC      TEST TO DETERMINE IF PERFORMANCE INDEX IS IMPROVED BY ALPHA
C
      CALL HOALF(X,U,D,ALPHA,HS)
      IF(HS.LT.HA) GO TO 8
    7 ALPHA=A
    8 CONTINUE
C
CCC      PRINT RESULTS OF LNSCH DURING THE FIRST OUTER ITERATION
C
      IF(KOUT.GT.1) GO TO 9
      WRITE(6,10) IND
      WRITE(6,11) ALPHA,HPO
    9 CONTINUE
   10 FORMAT(5X,'IND= ',I4)
   11 FORMAT(5X,'THE CHOSEN ALPHA IS ',F10.9,' HPO = ',E15.5)
      RETURN
      END
```

```
      SUBROUTINE PERFM(X,U,PER)
      DIMENSION X(201,4),U(200,3)
      COMMON PCON(4),T(4),DELTA,KOUT,IND,NC,NX,NPA
      WGHT(K,N)=FLOAT(N)*(FLOAT(K)/FLOAT(N))**9
      N=NPA+1
      PER=0.
      DO 1 J=1,N
      PEN=0.
      DO 2 K=1,NX
    2 PEN=PEN+PCON(K)*(X(J,K)-T(K))**2
      PEN=PEN*WGHT(J,K)
    1 PER=PER+PEN
      RETURN
      END
```

Subroutine PERFM calculates the performance index for state variable and control arrays. This subroutine is completely general in nature and need not be altered by the user.

INPUTS:

U - the control arrays

X - the state variable arrays

OUTPUT:

PER - the resulting performance index

```
      SUBROUTINE PRIN(X,U,G,PER,INDEX)
      DIMENSION X(201,4),U(200,3),G(200,3)
      COMMON PCON(4),T(4),DELTA,KOUT,IND,NC,NX,NPA
      WRITE(6,2) PER,INDEX
      IF(KOUT.GT.1) GO TO 4
      DO 1 J=1,NPA
    1 WRITE(6,3)J,(X(J,K),K=1,NX),(U(J,K),K=1,NX),(G(J,K),K=1,NX)
    4 CONTINUE
    2 FORMAT(/,5X,'PERFORMANCE INDEX IS ',F15.5,' ITERATION NUMBER',I5)
    3 FORMAT(I10,10F10.4)
      RETURN
      END
```

Subroutine PRIN prints the state variable and control arrays, and also any other desired information.

INPUTS:

U - the control arrays

X - the state variable arrays

G - the gradient arrays

PER - the performance index

INDEX - the total number of iterations thus far

OUTPUT:

literal computer printout

```
      SUBROUTINE TRAJ(X,U)
      DIMENSION X(201,4),U(200,3),UA(3),UT(3),XT(4)
      COMMON PCON(4),T(4),DELTA,KOUT,IND,NC,NX,NPA
C
C        ENTER THE SYSTEM EQUATIONS HERE IN THE FORM
C
C        F1(X1,...,XNX,U1,....,UNC)
C          .
C          .
C          .
C        FNX(X1,....,XNX,U1,....,UNC)
C
C        THE FOLLOWING EQUATIONS ARE EXAMPLES FROM THE
C        SECOND ORDER JET ENGINE MODEL
C
      F1(X1,X2,U1,W)=37.78*W-38.448*U1+0.66849
      F2(X1,X2,U1,W)=1.258*(U1*U1/(X1*X2)-W*X2)
      V1(U1,X2)=1.3009*X2-0.139825*U1
      V2(U1,X2)=SQRT(U1*U1+0.41688*X2*X2-0.0899*X2*U1)
      DO 1 J=1,NPA
      K=J+1
C
C        CHOOSE STATE VARIABLE N-TUPLES AND CONTROLS TO BE TESTED
C        FOR THE FEASIBLE DIRECTIONS MODIFICATION
C
      DO 2 M=1,NX
    2 XT(M)=X(J,M)
      DO 3 M=1,NC
    3 UT(M)=U(J,M)
      CALL CNSTR(XT,UT,UA)
C
C        THE RESULTING ALLOWABLE CONTROLS
C
      DO 4 M=1,NC
    4 U(J,M)=UA(M)
C
C        EULER INTEGRATION
C
      W=V1(U(J,1),X(J,2))-0.13982*V2(U(J,1),X(J,2))
      X(K,1)=X(J,1)+F1(X(J,1),X(J,2),U(J,1),W)*DELTA
      X(K,2)=X(J,2)+F2(X(J,1),X(J,2),U(J,1),W)*DELTA
    1 CONTINUE
      RETURN
      END
```

Subroutine TRAJ computes the state variable trajectory resulting from the control arrays and a set of initial conditions. The user enters the non-linear differential equations which describe the motion of the system.

Subroutine CNSTR is called to insure that the controls lie in an allowable region.

INPUTS:

U - the nominal control arrays

X(1,) - a set of initial conditions

OUTPUT:

U - the allowable control arrays

X - the resulting state variable trajectory

REFERENCES

1. Seldner, Kurt; Geyser, Lucille C.; Gold, Harold; Walker, Darrell; and Burgner, Gary. Performance and Control Study of a Low-Pressure-Ratio Turbojet Engine for a Drone Aircraft. NASA TM X-2537,1972.

2. Brennen, T.C. "Simplified Simulation Models for Control Studies of Turbojet Engines", Masters Thesis, Department of Electrical Engineering, University of Notre Dame, Notre Dame, Ind., August 1975.

3. Pontryagin, L.S. The Mathematical Theory of Optimal Processes, New York, Interscience 1962.

4. Bellman, R.E. Dynamic Programming, Princeton, Princeton University Press, 1957.

5. Larson, R. State Increment Dynamic Programming, New York, American Elsevier Publishing Co.,1968.

6. Richardson, M.H.; and Leake, R.J. Optimization of Discrete-Time Systems by Successive Approximations, Proc. Allerton Conf. on Circuit and System Theory, University of Illinois, 468-474, 1968.

7. Leake, R.J.; and Liu, Ruey-Wen, "Construction of Suboptimal Control Sequences", J. SIAM Control, Vol. 5, No. 1, 1967, 54-63.

8. Luenberger, D. Introduction to Linear and Nonlinear Programming, Reading Mass: Addison-Wesley, 1965.

9. Lasdon, L.S. The Conjugate Gradient Method for Optimal Control Problems, IEEE Transactions on Automatic Control, April 1967.