COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN

1210 W. Dayton

Madison, Wisconsin

SOLUTION OF NONLINEAR TWO-POINT BOUNDARY VALUE PROBLEMS BY LINEAR PROGRAMMING

J. B. Rosen & Robert Meyer

Computer Sciences Technical Report #1

January 1967

COMPUTER SCIENCES DEPARTMENT

UNIVERSITY OF WISCONSIN

SOLUTION OF NONLINEAR TWO-POINT BOUNDARY VALUE PROBLEMS BY LINEAR PROGRAMMING

J. B. Rosen & Robert Meyer

Computer Sciences Technical Report # 1

January 1967

1210 W. Dayton Street
Madison, Wisconsin

ABSTRACT

A system of n nonlinear ordinary differential equations is considered on the interval $[a, b]$ with at least one of the n boundary conditions specified at each end of the interval. In addition, any available a priori bounds on the solution vector may be imposed. An iterative method for solution is described which is essentially a Newton-Raphson method with a linear programming solution at each iteration. Every iterate is a minimax solution to a linearized finite difference approximation to the original system, and also satisfies the boundary conditions and the a priori bounds. The method will always converge at least as fast as Newton-Raphson, and may converge when Newton-Raphson fails. A number of computational examples are described.

SOLUTION OF NONLINEAR TWO-POINT BOUNDARY VALUE PROBLEMS BY LINEAR PROGRAMMING

J. B. Rosen & Robert Meyer

1. INTRODUCTION

The general problem considered here is that of a system of n nonlinear ordinary differential equations on the interval $[a, b]$, with at least one of the n boundary conditions specified at each end of the interval. In addition, any available a priori bounds on the solution vector may be imposed. It is also useful to impose reasonable upper and lower bounds on the unknown boundary values.

The computational solution to be described is an iterative method in which all the boundary conditions and other imposed bounds are satisfied at every iteration. This is accomplished by what is essentially a Newton-Raphson method with a linear programming solution at each iteration. The method reduces to a Newton-Raphson method if none of the imposed bounds become active (satisfied as an equality) during the process. If one or more of the imposed bounds do become active then the new method may converge faster than the Newton-Raphson method or it may succeed where the latter would fail.

This approach was suggested in a previous paper [11] dealing with discrete optimal control problems. The fundamental recurrence relations to be used here were developed in an earlier report on discrete optimal

---

Sponsored by NSF Research Grant GP 6070.

control [10]. These relations serve to decouple the system and to reduce the dimension of the linear programming problem solved at each iteration.

In all but two cases tested, rapid convergence was obtained using the basic linear programming method. It is possible under some conditions, however, that cycling may occur. Two special cases were constructed to illustrate this, and a special technique was devised to prevent cycling and force convergence. The normal convergent cases are illustrated by Examples 1, 2, and 3, and the special cases by Examples 4 and 5.

2. PROBLEM STATEMENT

We will first consider a class of continuous problems of the following kind: determine an n-dimensional function $y(t)$ such that

$$\dot{y}(t) = f(y, t) \quad , \quad 0 \leq t \leq 1 \tag{2.1}$$

$$y_i(0) = q_{i0} \quad , \quad i = 1, \ldots, \ell < n \tag{2.2}$$

and

$$y_i(1) = q_{im} \quad , \quad i = 1, \ldots, n-\ell \tag{2.3}$$

where f is assumed to be a continuous vector function of y and t with continuous first partial derivatives with respect to the components y_i, $i = 1, \ldots, n$ of y. The boundary conditions (2.3) are specified for the first $n-\ell$ components for the sake of notational simplicity below; the method can be applied to any properly posed problem.

We approximate (2.1) by the finite difference scheme

$$\frac{x_{j+1} - x_j}{\Delta t} = \tfrac{1}{2}\left[f(x_{j+1}, t_{j+1}) + f(x_j, t_j)\right] \qquad (2.4)$$

$$j = 0, \ldots, m-1$$

where $\Delta t = \frac{1}{m}$, $t_j = j\Delta t$, and the x_j, $j = 0, \ldots, m$ are to be determined so that (2.4) and the boundary conditions are satisfied.

We will consider below problems in which (2.1) is replaced by an equation in which a control term is added to the arguments of the function on the right hand side, and (2.2) and (2.3) are replaced by equations involving linear combinations of initial and terminal values. Linear inequality constraints will also be imposed on the x_j and the control terms.

3. ITERATIVE SOLUTION

In order to linearize the nonlinear system (2.4) we use the approximation

$$f(x_j, t_j) \cong f(x_j^k, t_j) + F_j^k\left[x_j - x_j^k\right] \qquad (3.1)$$

where the $n \times n$ matrix F_j^k is the Jacobian with respect to x of $f(x, t)$ evaluated at (x_j^k, t_j) and x_j^k is the k'th approximation to x_j. The initial approximate trajectory $(k = 0)$ is usually taken to be a constant or linear function, and need not satisfy the boundary conditions. The approximation (3.1) is used to transform (2.4) into the linear system of mn equations in mn unknowns

$$\frac{x_{j+1} - x_j}{\Delta t} = \tfrac{1}{2} F_{j+1}^k x_{j+1} + \tfrac{1}{2} F_j^k x_j + \tfrac{1}{2}\left[s_{j+1}^k + s_j^k\right] \qquad (3.2)$$

where $s_j^k = f(x_j^k, t_j) - F_j^k x_j^k$, a constant vector. In order to generalize the Newton-Raphson algorithm, we introduce a control term on the right hand side of (3.2) to obtain

$$\frac{x_{j+1} - x_j}{\Delta t} = \tfrac{1}{2} F_{j+1}^k x_{j+1} + \tfrac{1}{2} F_j^k x_j + \tfrac{1}{2} [s_{j+1}^k + s_j^k] + \mu_j s \quad (3.3)$$

$$j = 0, \ldots, m-1$$

where the μ_j, $j = 0, \ldots, m-1$, are scalars to be determined and s (the sum vector) $\in E^n$ is a constant vector with every component equal to 1. If at any iteration there is no solution of the system (3.2) satisfying the boundary conditions and the inequality constraints to be specified below, there may be such a solution of (3.3) with some $\mu_j \neq 0$. The scalars μ_j play the role of a scalar control (representing an error in the finite difference equation), and are introduced to allow intermediate solutions to be obtained even when the linear system from a Newton-Raphson method might not have a solution. Denoting by X the set $\{x_j \mid j = 0, \ldots, m\}$, by U the set $\{\mu_j \mid j = 0, \ldots, m-1\}$, and by $\|U\|$ the $\max_j |\mu_j|$, we will determine by linear programming the pair (X^*, U^*) which minimizes $\|U\|$ over all pairs (X, U) which satisfy (3.3), the boundary conditions, and the inequality constraints below.

The size of this linear programming problem is reduced by partitioning it into smaller problems. This reduction is accomplished in two stages. The first stage consists of solving (3.3) for x_{j+1} to obtain

$$x_{j+1} = K_j x_j + \mu_j v_j + \bar{s}_j \quad (3.4)$$

where

$$K_j = [I - \frac{1}{2m} F_{j+1}^k]^{-1} [I + \frac{1}{2m} F_j^k] \tag{3.5}$$

$$v_j = \frac{1}{m} [I - \frac{1}{2m} F_{j+1}^k]^{-1} s \tag{3.6}$$

and

$$\bar{s}_j = \frac{1}{2m} [I - \frac{1}{2m} F_{j+1}^k]^{-1} [s_{j+1}^k + s_j^k] \tag{3.7}$$

In the second stage, the recursion relation (3.4) is used to express the terminal vector in terms of x_0 and the μ_j, giving

$$x_m = V_0 x_0 + \sum_{j=0}^{m-1} \mu_j h_j + g \tag{3.8}$$

where

$$g = \sum_{j=0}^{m-1} V_{j+1} \bar{s}_j \tag{3.9}$$

and

$$h_j = V_{j+1} v_j , \quad j = 0, \ldots, m-1 \tag{3.10}$$

The matrices V_j, $j = m-1, \ldots, 0$ are determined by the recursion relations

$$V_m = I , \quad V_j = V_{j+1} K_j \tag{3.11}$$

We now specify upper and lower bounds on the unknown initial and terminal conditions to obtain the inequality constraints cited above. (It should be noted that all of these bounds are not necessary in order to employ the method. However, whatever information on bounds is available should be used in setting up the problem, because it facilitates the solution. Furthermore, upper or lower bounds on any components of any

of the x_j can easily be handled, as shown in Example 5 of section 5. For notational simplicity we consider below only the case of upper and lower bounds at the boundaries.)

We thus specify bounds $\underline{q}_{i0}$ and $\bar{q}_{i0}$ and require

$$\underline{q}_{i0} \le \xi_i \le \bar{q}_{i0} , \quad i = \ell + 1, \ldots, n \tag{3.12}$$

where the ξ_i denote the components of x_0 . If in addition we let

$$\underline{q}_{i0} = \bar{q}_{i0} = q_{i0} , \quad i = 1, \ldots, \ell \tag{3.13}$$

then we can write (2.2) and (3.12) in vector notation as

$$\underline{q}_0 \le x_0 \le \bar{q}_0 . \tag{3.14}$$

Similarly, if we have bounds $\underline{q}_{im}$ and $\bar{q}_{im}$ on the unknown terminal values, we require

$$\underline{q}_m \le x_m \le \bar{q}_m \tag{3.15}$$

where the first $(n - \ell)$ components of (3.15) express the boundary conditions (2.3).

We want to find x_0 so that the initial and terminal bounds are satisfied, and so that $\mu_j = 0$, $j = 0, \ldots, m-1$. If such a solution exists, it will be found by solving the following linear programming problem

$$\min_{x_0, \mu_j, \gamma} \left\{ \gamma \;\middle|\; \begin{array}{l} \underline{q}_0 \le x_0 \le \bar{q}_0 \\ \underline{q}_m \le V_0 x_0 + \sum_{j=0}^{m-1} \mu_j h_j + g \le \bar{q}_m \\ -\gamma \le \mu_j \le \gamma , \quad j = 0, \ldots, m-1 \end{array} \right\} \tag{3.16}$$

If no solution with $\mu_j = 0$ for all j exists, then (3.16) finds a solution for which $\max_j |\mu_j|$ = minimum, as noted above. The state vectors x_j^{k+1}, $j = 1, \ldots, m$ are then immediately determined by (3.4). These new values are the (k+1)st approximation to the solution.

If these new values differ from those obtained in the previous iteration by less than a specified tolerance, we consider the solution to have converged, and terminate the iterations. Otherwise, we use the (k+1)st approximation to determine a new V_0, g, and new h_j, and repeat the process. Note that each approximation thus satisfies the boundary conditions and the upper and lower bounds, but does not in general satisfy (2.4) or (3.2). If the original set of equations (2.4) has a solution satisfying (3.14) and (3.15), we can expect, however, that only the first few iterates will not satisfy (3.2).

4. SOLUTION OF THE LINEAR PROGRAMMING PROBLEM

To put (3.16) into the format of a standard dual linear programming problem, let $u \in E^m$ be the vector with components μ_j, let $V = V_0'$, and let H be the $(m \times n)$ matrix whose j^{th} row is h_j'. Define a vector $c \in E^{2m+4n}$ and a $(m+n+1) \times (2m+4n)$ matrix A as follows

$$c' = (\underline{q}_0' \mid -\bar{q}_0' \mid \underline{q}_m' - g' \mid g' - \bar{q}_m' \mid 0 \ldots 0 \mid 0 \ldots 0)$$

$$A = \begin{bmatrix} I_n & -I_n & V & -V & 0 & 0 \\ 0 & 0 & H & -H & I_m & -I_m \\ 0 & 0 & 0 & 0 & 1 \ldots 1 & 1 \ldots 1 \end{bmatrix}$$

Also let $w,\ b \in E^{m+n+1}$ be given by

$$w = \begin{pmatrix} x_0 \\ u \\ \gamma \end{pmatrix} \quad , \qquad b = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \quad .$$

The problem (3.15) is in the form of an (unsymmetric) dual linear programming problem [4] , and can now be written as

$$\min_{w} \{b'w \mid A'w \geq c\} \quad . \tag{4.1}$$

Computationally, it is convenient to actually solve the corresponding primal problem

$$\max_{z} \{c'z \mid Az = b,\ z \geq 0\} \tag{4.2}$$

where $z \in E^{2m+4n}$. With the problem in this form a standard linear programming code (such as CDM4) can be used to obtain the dual solution w as the shadow price vector when the optimal z is obtained. That is, if B is the optimal basis and $\bar{c}$ is the vector of cost coefficients associated with the elements in the basis, then the solution of (4.1) is given by $w = (B^{-1})'\bar{c}$, where the matrix B^{-1} is obtained from the simplex tableau. As indicated above, the remaining x_j are then computed from the relations (3.4), allowing a check to be made on round-off error, since the terminal vector thus obtained should satisfy the boundary conditions.

In the Appendix we show that an initial feasible basis for the primal problem can always be constructed. For certain types of problems, as discussed in the Appendix, it is possible to show that the dual problem has a solution, which means that both the primal and dual problems have

optimal solutions. If the primal problem has an infinite solution, then there is no feasible solution to the dual, and a different linearization must be tried. In this case, the Newton-Raphson algorithm would also have to be restarted. On the other hand, in the next section we present some examples for which the Newton-Raphson method failed, but the linear programming approach succeeded.

5. COMPUTATIONAL RESULTS

Example 1

Consider the equation

$$\ddot{y} = 2y^3, \quad 0 \le t \le 1 \tag{5.1}$$

with boundary conditions

$$y(0) = 1 , \quad y(1) = \tfrac{1}{2} \tag{5.2}$$

The unique solution is $y(t) = (t+1)^{-1}$, as may be readily verified. In system form, (5.1) becomes

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} y_2 \\ 2y_1^3 \end{pmatrix} \tag{5.3}$$

with the corresponding boundary conditions

$$y_1(0) = 1 , \quad y_1(1) = \tfrac{1}{2} \ . \tag{5.4}$$

The linearized discrete system obtained by using formula (3.3) is

$$\frac{x_{j+1}-x_j}{\Delta t} = \tfrac{1}{2}\begin{bmatrix} 0 & 1 \\ 6\left(x^k_{1,j+1}\right)^2 & 0 \end{bmatrix} x_{j+1} + \tfrac{1}{2}\begin{bmatrix} 0 & 1 \\ 6\left(x^k_{1,j}\right)^2 & 0 \end{bmatrix} x_j$$

$$+ \tfrac{1}{2}\begin{bmatrix} 0 \\ -4\left(x^k_{1,j+1}\right)^3 \end{bmatrix} + \tfrac{1}{2}\begin{bmatrix} 0 \\ -4\left(x^k_{1,j}\right)^3 \end{bmatrix} + \mu_j s \tag{5.5}$$

$$j = 0, \ldots, m-1$$

where $x_j = \begin{pmatrix} x_{1,j} \\ x_{2,j} \end{pmatrix} \in E^2$ and $s = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \in E^2$.

Note that the values $x_{2,j}^{k}$, $j = 0, \ldots, m$, which constitute the k'th approximation to the derivative of $y(t)$ at the grid points, do not appear in the system (5.5). Hence, every initial choice for the values of the derivative at the grid points leads to the same linearized problem on the first iteration. In this example, the constraints on the initial conditions were

$$\begin{bmatrix} 1 \\ -5 \end{bmatrix} \leq x_0 \leq \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{5.6}$$

The length of the time interval Δt was 0.1 $(m = 10)$, and the initial approximate trajectory (IAT) was given by

$$x_j^0 = \begin{pmatrix} 1 - j \cdot \Delta t/2 \\ 0 \end{pmatrix}, \quad j = 0, \ldots, 10 \tag{5.7}$$

The procedure was to be halted when two successive iterates satisfied the following convergence test

$$\sum_{i=1}^{2} \sum_{j=0}^{m} \left| x_{i,j}^{k+1} - x_{i,j}^{k} \right| < 0.01 \, . \tag{5.8}$$

Because of the quadratic convergence of Newton's method, the convergence criterion (5.8) can be expected to imply that further iterations would not change the first four decimal places.

With the linear IAT (5.7), the convergence test (5.8) was satisfied after only three iterations. Neither of the derivative constraints contained in (5.6) was ever active, and thus we had $\mu_j = 0$, $j = 0, \ldots, m-1$ for each iteration. Thus, for this example the iterates were the same as those that would have been obtained if the ordinary Newton-Raphson

algorithm had been used. The maximum relative error between the true and computed solutions at the grid points was approximately 0.2%, indicating an accurate approximation in spite of the small value of m used.

Example 2

We use the same differential equation as in the previous example

$$\ddot{y} = 2y^3 \tag{5.9}$$

but modify the boundary conditions to

$$y(0) + y'(0) = 0 , \quad y(1) = \tfrac{1}{2} . \tag{5.10}$$

The same solution as above, $y(t) = (t+1)^{-1}$ satisfies these boundary conditions also.

If the IAT is $x_{1,j}^{0} = 0$, $j = 0, \ldots, m$, then the system (5.5) has no solution satisfying (5.10) with all $\mu_j = 0$. Note that for this IAT, (5.5) implies that $x_{2,j}$ is a constant independent of j if we require that $\mu_j = 0$, $j = 0, \ldots, m-1$. Hence if $x_{2,j} = c$, $j = 0, \ldots, m$, we see from (5.5) that

$$x_{1,j} = x_{1,0} + j \cdot c \cdot \Delta t , \quad j = 0, \ldots, m . \tag{5.11}$$

However, from the first equation of (5.10), $x_{1,0} = -c$, so that (5.11) may be written

$$x_{1,j} = c(-1 + j \cdot \Delta t) , \quad j = 0, \ldots, m . \tag{5.12}$$

This means that $x_{1,m} = 0$, violating the second boundary condition of (5.10). Geometrically, the above is equivalent to trying to connect the points $(0, -c)$ and $(1, \frac{1}{2})$ with a straight line of slope c. Such a line

must pass through 0 at $t = 1$. Thus, the Newton-Raphson algorithm would fail in this case, and the process would have to be restarted with a different IAT.

Since the generalization of the Newton-Raphson method that we are considering allows for $\mu_j \neq 0$, it succeeds with the given IAT. The boundary condition represented by the first equation of (5.10) is of a different sort than those considered in (2.2), but is easily handled. The equality is converted into two inequalities which are added to the set of inequalities in (3.15); and the vectors $\underline{q}_0$ and $\bar{q}_0$ then represent lower and upper bounds on both of the components of x_0. In this example we chose

$$\underline{q}_0 = \begin{bmatrix} 0 \\ -5 \end{bmatrix} \quad \text{and} \quad \bar{q}_0 = \begin{bmatrix} 5 \\ 0 \end{bmatrix} .$$

Clearly, boundary conditions involving linear combinations of any of the boundary components may be similarly handled.

The convergence test was satisfied in eight iterations, and the final iterate differed from the solution obtained in Example 1 by at most one digit in the third place. The maximum error ($\max |\mu_j|$) in the first iteration was 0.166, and the errors were zero in all later iterations.

By specifying better bounds on the initial conditions, it is possible to accelerate convergence. For lower bounds of -5, -2, -1.5, and -1 on the initial derivative and analogous upper bounds on the initial state, the number of iterations required were respectively 8, 6, 5, and 3. Plots of the results for lower bounds of -1.5 and -1 are presented in

Figs. 1 and 2. The final computed value of the initial derivative in all cases was -0.9990 .

Example 3.

We consider the equation

$$\ddot{y} = e^{-3y} \tag{5.13}$$

with boundary conditions

$$y(0) = y(1) = 0 \ . \tag{5.14}$$

This problem has exactly two solutions:

$$y(t) = \frac{2}{3}\left[\log \cosh \left[\frac{a}{2}\left(t - \frac{1}{2}\right)\right] - \log \cosh \frac{a}{4}\right] \tag{5.15}$$

where a is either of the two solutions of

$$\frac{a}{\cosh \frac{a}{4}} = \sqrt{6} \tag{5.16}$$

For the sake of brevity, the two solutions of (5.15) will be referred to below as the upper and lower (due to their relative positions in the (t, y) plane) solutions.

Case 1

If the method is applied using the IAT

$$x_j^{\,0} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \ , \ j = 0, \ldots, m = 20 \tag{5.17}$$

and the constraint

$$-5 \leq x_{2,\,0} \leq 0 \tag{5.18}$$

the iterates converge to an approximation of the upper solution with the

convergence test (5.8) being satisfied after four iterations. All of the error terms are 0 in each iteration, so the same iterates could have been obtained using the ordinary Newton-Raphson approach. The iterates are shown in Figure 3.

Case 2

With the IAT

$$x_j^0 = \begin{pmatrix} 2\left|t_j - \frac{1}{2}\right| - 1 \\ 0 \end{pmatrix}, \quad j = 0, \ldots, m = 20 \qquad (5.19)$$

and the same constraint (5.18), the iterates converge to an approximation of the lower solution with the convergence test satisfied after five iterations as shown in Fig. 4. Again, the error terms were all 0 in every iteration.

Case 3

Now consider the case in which the IAT (5.19) is again used, but the constraint (5.18) is replaced by

$$-1 \leq x_{2,0} \leq 0 \qquad (5.20)$$

This constraint excludes the solution obtained in Case 2 (lower solution) since the final computed value for $x_{2,0}$ in that case was -2.09. It does not, however, exclude the upper solution obtained in Case 1.

As shown in Fig. 5, the upper solution was obtained in seven iterations. The values for the maximum errors ($e^k = \max_j |\mu_j|$ on k^{th} iteration) were $e^1 = 0.620$, $e^2 = 0.380$, $e^3 = 0.181$, $e^4 = \ldots = e^7 = 0$.

Therefore, when confronted with a problem in which the existence of two or more solutions is suspected, it is possible with this method to

search for the suspected solutions with the assurance that convergence to a solution already obtained cannot occur if the constraints are chosen properly. Alternatively, if a particular solution is desired, the constraints may be chosen to prevent convergence to any other solution.

Case 4

The last case for this example illustrates that some care must be used in excluding one solution of a multiple solution problem. In this case the lower solution was (just barely) excluded by the constraint

$$-2 \leq x_{2,0} \leq 0 \quad . \tag{5.21}$$

Instead of converging to the upper solution it converged (in 4 iterations) to a trajectory close to the lower solution but with some $\mu_j \neq 0$. Thus it was prevented by the bound (5.21), from reaching the global minimum and was caught in a constrained local minimum. This fact was immediately obvious since all the μ_j were not zero when the convergence criterion (5.8) was satisfied. This case is shown in Fig. 6.

Example 4

The equation

$$\ddot{y} = e^{-3.6y} \tag{5.22}$$

has no solution satisfying the boundary conditions

$$y(0) = y(1) = 0 \quad . \tag{5.23}$$

When applied to the discretized version of this problem with $m = 10$, the Newton-Raphson algorithm gave no indication of convergence after

15 iterations, as may be seen in Fig. 7. (It should be noted that one can construct examples in which the continuous problem has no solution, but the discretized problem for a fixed Δt does have a solution to which the Newton-Raphson method converges.)

The basic linear programming approach did not yield a convergent sequence of approximations to this problem either. Therefore, the following inequalities were introduced on the approximations to the initial derivative

$$\max \{-1.3,\ x_2^{(i)} - \eta^{(i)}\} \le x_2^{(i+1)}(0) \le \min \{-1,\ x_2^{(i)} + \eta^{(i)}\} \quad (5.24)$$

where i is an iteration index and $\eta^{(i)}$ is a parameter to be chosen automatically before the start of each iteration in a manner to be described below. This new type of constraint forces each initial derivative to lie in the intersection of a neighborhood of radius $\eta^{(i)}$ about the previous initial derivative with the interval $[-1.3,\ -1]$. The same idea is used in the MAP method of Griffith and Stewart [3].

The basic principles involved in choosing the $\eta^{(i)}$ are analogous to those of the bisection method for determining the roots of an equation. The derivative is allowed to take steps of a given size until there is an indication (in this case, cycling) that we are near our goal, and then the size of the step allowed is decreased by a factor of 4 and the process is repeated. The fundamental difference between the procedure employed here and the bisection-type scheme is the use of the gradient to guide the process.

The starting values used in (5.24) were $x_2^{(0)}(0) = -1.15$ and $\eta^{(0)} = 0.075$. The convergence test was satisfied in 15 iterations, and the final iterate had a minimax error of 0.0106. The iterates are shown in Fig. 8.

Example 5

In order to illustrate how a linear inequality constraint along the whole time interval may be handled, we will consider a true optimal control problem. The basic technique remains unchanged, since the boundary value problems were treated as problems in optimal control. The example below also deals with the problem of cycling.

Consider the constrained brachistochrone problem:

maximize $x_1(T)$ subject to

$$\dot{x}_1 = \sqrt{2gx_2 + c}\ \cos u(t) \qquad (5.25)$$

$$\dot{x}_2 \quad \sqrt{2gx_2 + c}\ \sin u(t) \qquad (5.26)$$

$$x_1(0) = x_2(0) = 0 \qquad (5.27)$$

$$ax_1(t) + b \geq x_2(t) \quad 0 \leq t \leq T \qquad (5.28)$$

where a, b, c, g, and T are given constants, and u(t) is constrained to the interval $[0, \pi/2]$. (In the scaling used, $a = 1/2$, $b = 1/6$, $c = (0.07195)^2$, $g = 1$, and $T = 1.720$.)

The constraint (5.28) is shown in Fig. 10. The state variable x_1 represents horizontal distance and x_2 vertical distance. The control variable in this problem corresponds to the slope of the trajectory rather than the error it represented in the boundary value problems.

The analytic solution to the problem may be obtained by the introduction of multiplier functions, the Euler-Lagrange equations (suitably modified along the boundary (5.28)), and the maximum principle as outlined in Bryson, et al [1]. If we let $[t_1, t_2]$ denote the time interval during which the solution lies on the constraint, then the optimal control $\gamma^*(t)$ is given by the formulae

$$\gamma^*(t) = \begin{cases} \gamma^*(0) - \omega_1 t \ , & t \in [0, t_1] \\ \tan^{-1} a \equiv \theta \ , & t \in [t_1, t_2] \\ \omega_2 (T-t) \ , & t \in [t_2, T] \end{cases} \tag{5.29}$$

where ω_1 is determined by computing the value of r_1 from the nonlinear equations

$$\frac{\cos^2 \theta}{2gr_1^2} - \frac{c}{2g} + \frac{a}{2gr_1^2}\left(\theta - \gamma_0 + \frac{\sin 2\theta}{2} - \frac{\sin 2\gamma_0}{2}\right) - b = 0 \tag{5.30}$$

$$\cos \gamma_0 = -r_1 c^{\frac{1}{2}} \tag{5.31}$$

and using the relation

$$\omega_1 = -g r_1 \ ; \tag{5.32}$$

ω_2 is then determined by the equations

$$t_1 = \frac{\gamma_0 - \theta}{\omega_1} \tag{5.33}$$

$$\ell = g \frac{\left(t_f - t_1 + \frac{\operatorname{ctn} \theta}{\omega_1}\right)^2}{2(\operatorname{ctn} \theta + \theta)} - (b + \frac{c}{2g}) \operatorname{ctn} \theta \tag{5.34}$$

$$\omega_2 = \left(\frac{g}{2}\right)^{\frac{1}{2}} \left(\frac{\theta + \operatorname{ctn}\theta}{(b + \frac{c}{2g})\operatorname{ctn}\theta + \ell}\right)^{\frac{1}{2}} \tag{5.35}$$

The linearization of (5.25) and (5.26) about $x_2^{(0)}(t)$ and $u^0(t)$ is (suppressing for notational simplicity the dependence on t)

$$\dot{x}_1 = [g\cos u^0 (2gx_2^0 + c)^{-\frac{1}{2}}]x_2 - [(2gx_2^0 + c)^{\frac{1}{2}} \sin u^0]u \tag{5.36}$$
$$- g\cos u^0 x_2^0 (2gx_2^0 + c)^{-\frac{1}{2}} + (2gx_2^0 + c)^{\frac{1}{2}} (u^0 \sin u^0 + \cos u^0)$$

$$\dot{x}_2 = [g\sin u^0 (2gx_2^0 + c)^{-\frac{1}{2}}]x_2 + [(2gx_2^0 + c)^{\frac{1}{2}} \cos u^0]u \tag{5.37}$$
$$- g\sin u^0 x_2^0 (2gx_2^0 + c)^{-\frac{1}{2}} + (2gx_2^0 + c)^{\frac{1}{2}} (u^0 \cos u^0 + \sin u^0)$$

These linear equations are then discretized using the finite difference approximation previously described by (2.4) with $\Delta t = T/20 = 0.086$. These yield recurrence relations of the form (3.4), and from those relations we obtain

$$x_i(t_k) = \sum_{j=0}^{k} \alpha_{ijk} u_j + \beta_{ik}, \quad i = 1, 2 \quad k = 1, \ldots m \tag{5.38}$$

where the α_{ijk} and β_{ik} are constants determined by $u^0(t_j)$ and $x_2^0(t_j)$, $j = 0, \ldots, k$, and u_j represents the control at time t_j. Note that since $x_1(0)$ and $x_2(0)$ are known, only the control variables appear as unknowns on the right hand side of (5.38).

By means of (5.38), the constraint (5.28) may be expressed at the grid points by the inequality

$$\sum_{j=0}^{k} (a\alpha_{1jk} - \alpha_{2jk})\, u_j \geq -a\beta_{1k} + \beta_{2k} - b \qquad (5.39)$$

$$k = 1, \ldots, m$$

At each iteration therefore, we solve a linear programming problem of the following form:

$$\text{minimize} \quad - \sum_{j=0}^{m} \alpha_{1jm} u_j \qquad (5.40)$$

subject to the inequalities (5.39) and

$$0 \leq u_j \leq \pi/2 , \quad j = 0, \ldots, m \quad . \qquad (5.41)$$

Note that the only variables in the linear programming problem are the $m + 1$ values of the u_j. Since there are $3m + 2$ inequality constraints, the problem in the primal form thus involves an $(m+1) \times (3m+2)$ matrix. If the linear programming problem has a solution, we solve for the state variables by means of (5.38), compute new values for α_{ijk} and β_{ik}, and repeat the process until the convergence criterion is satisfied or the iteration limit is reached.

If the solution is attempted with the problem in the above form, the iterates begin to cycle after a few iterations, and the convergence criterion is never satisfied. In addition, by the theory of linear programming, at least $(m+1)$ of the $3m+2$ inequalities must be satisfied as equalities by an optimal solution to the linear programming problem. The physical situation dictates that any reasonable trajectory will lie on the constraint (5.39) for only a short time, since the optimal trajectory in the continuous case lies on the constraint (5.28) for only a brief interval of time. This

means that most of the values for the control will be 0 or $\pi/2$, satisfying some of the remaining inequalities (5.41). However, the actual optimal control is definitely not of a bang-bang type. Therefore we could not expect the above procedure to yield a good approximation to the optimal control even if it converged. Both of these problems are solved by replacing the constraints (5.41) by

$$\max\{0,\ u_j^{(i)} - \eta^{(i)}\} \le u_j \le \min\{u_j^{(i)} + \eta^{(i)}, \pi/2\} \qquad (5.42)$$

$$j = 0, \ldots, m$$

In a typical run, the initial approximate trajectory used was $x_1^{(0)} \equiv 0$, $x_2^{(0)} = 0.75\ (t/T)$, $u^{(0)} \equiv 0.9$ radians and $\eta^{(0)} = 0.3$ radians. Cycling was detected by the program after the fifth iteration, and it defined $\eta^{(5)} = \eta^{(4)}/4 = 0.075$ radians, and $x_2^{(5)}$ and $u^{(5)}$ to be the average of their previous two values. Cycling was detected again after the eighth and eleventh iterations, and operations analogous to those following the fifth iteration were performed. After the fourteenth iteration, the procedure was automatically halted when the convergence test was satisfied. At that point the computed values of $x_1(T)$ had begun to cycle between 0.99931 and 0.99932. The optimal value for the given continuous problem is $x_1(T) = 1$.

The first 4 iterations are shown in Fig. 9. Iterations 5 and 6 are shown in Fig. 10. The remaining iterations cannot be distinguished graphically from the 6^{th} iteration. The computed average value $\frac{1}{2}[u_j + u_{j+1}]$

for the interval $t_j \leq t \leq t_{j+1}$ is shown in Fig. 11 together with the continuous optimal control obtained by an exact solution to the problem. The maximum error in the discrete optimal trajectory, as compared to the continuous solution, for both the horizontal and vertical state variables was of the order of one unit in the third decimal place.

6. COMPUTATIONAL EFFICIENCY

Trials were run, using instead of the balanced scheme (3.3), the simple Eulerian approximation

$$\frac{x_{j+1} - x_j}{\Delta t} = F_j^k x_j + s_j^k + \mu_j s \qquad (6.1)$$

$$j = 0, \ldots, m-1$$

On the basis of limited testing it appears that this approach is less efficient in obtaining a solution of given accuracy. The increased number of grid points necessary more than offset the smaller amount of calculation per grid point. However, for large systems of differential equations (large n), the fact that the matrix inversions in (3.5) are not necessary when (6.1) is used, may improve the relative efficiency of (6.1).

The use of a vector control term in (3.3) rather than the scalar control multiplying the sum vector was also tested. That is, for any fixed j, the n equations represented by (3.3) were allowed to have different values for their error term. Clearly, this includes as a special case the scalar control method. However, for the small values of n and large values of m used in typical applications, this almost doubles the size of the linear programming problem to be solved at each iteration, and results so far have indicated that it does not reduce the number of iterations required. In a typical trial, using the equation (5.13) with the constraints (5.21), the average time per iteration with the vector control

with $m = 20$ was slightly over 1 minute, whereas the scalar control required less than 15 seconds per iteration. Since the number of iterations required was the same and the final approximations were equally good, the scalar control method was more than four times as efficient in this case.

If one wishes to solve a boundary value problem using a large number of grid points, results have shown that a significant amount of time can be saved by first solving with a much smaller number of grid points and using the solution thus obtained to provide starting values for the original problem. The additional starting values needed are obtained by linear interpolation between the grid points. The computational efficiency of this successive increase in the number m of grid points follows directly from its effect on the number of rows in the primal LP problem (4.2). As given by the definition of the matrix A, this is $(m+n+1)$. The computation time for solving an LP problem is proportional to the q^{th} power of the number of rows where $2 < q < 3$. Thus any scheme which reduces the number of iterations required with large m will certainly reduce the computation time.

For example, in a trial run with equation (5.13), the convergence test was satisfied after four iterations with $m = 10$, but only two iterations were required with $m = 40$ using the interpolated solution for starting values. Since the time required for the four iterations with $m = 10$ was only about $1/3$ of the time required for one iteration with $m = 40$, computing time was almost cut in half by this device.

It should not be difficult to implement an automatic grid-size halving procedure of this type which would improve efficiency and also give additional information on accuracy.

Another improvement in efficiency can probably be obtained by storing information on the current optimal LP basis, for use in the next iteration. The LP solution for each iteration could then be started using the previous optimal basis. This should significantly reduce the computing time per iteration.

7. CONVERGENCE AND DISCRETIZATION ERROR

In order to discuss convergence and to give error estimates, it is necessary that there exist solutions to the given problem and the approximating linear problems. For boundary value problems, existence theorems have been constructed only for certain special classes of problems. McGill and Kenneth [8, 9] show that if the length of the time interval $[a, b]$ is "sufficiently small," the continuous nonlinear problem has a unique solution, and that the Newton-Raphson method converges quadratically to that solution provided the initial guess lies in a certain region containing the line connecting the boundary values. Lees [7] obtains a convergence theorem for second order equations of the following form (which we will call Class L problems)

$$\ddot{y} = f(y, t) , \quad y(a) = A , \quad y(b) = B \tag{7.1}$$

where a, A, b, and B are finite but otherwise arbitrary constants, and $f(y, t)$ is a continuous real-valued function on $R = \{(y, t) \mid a \leq t \leq b , \ -\infty < y < +\infty\}$ that has a continuous partial derivative with respect to y satisfying

$$f_y = \frac{\partial f}{\partial y} \geq -\eta > \frac{-\pi^2}{(b-a)^2} \tag{7.2}$$

Class L problems, as Lees shows, have unique solutions, and under some additional hypotheses concerning the existence of high order derivatives of the solution, the Newton-Raphson method may be shown to converge to the solution of the discretized nonlinear problem provided

the step size is sufficiently small and the initial guess sufficiently good. Lees also gives a bound for the discretization error. Kalaba [6] gives results on differential inequalities which may be used to prove the following theorem:

If a problem is of Class L and if f is a convex function of y for each fixed $t \in [a, b]$, then the Newton-Raphson method applied to the continuous problem yields a sequence of linear problems

$$\ddot{y}_k = f(y_{k-1}, t) + (y_k - y_{k-1}) f_y(y_{k-1}, t) \qquad (7.3)$$

$$y_k(a) = A \ , \ y_k(b) = B$$

each of which has a solution, provided $y_0(t)$ is a continuous function satisfying the boundary conditions. The sequence of functions so generated converges monotonically from above ($y_1(t) \geq y_2(t) \geq \cdots \geq y(t)$) to the solution $y(t)$. (Note that we do not assume $y_0(t) \geq y(t)$.)

It should be emphasized that all of these results are concerned with sufficient conditions for convergence, and that convergence will often take place even when the conditions are not satisfied. Rather crude initial guesses such as linear or constant functions which do not satisfy the sufficient conditions will often be satisfactory. Problems with multiple solutions are not considered in the theoretical investigations mentioned above. Nevertheless, Example 3 of Section 5 demonstrates that the Newton-Raphson method worked well in a problem with two solutions. Not only did we have quadratic convergence in that example,

but as shown in Figure 12, the discretization error was $0((\Delta t)^2)$, just as it would have been if the problem were of Class L.

Finally, as noted above, the method developed in this report will converge whenever the Newton-Raphson method converges (provided the constraints are properly chosen), and it may converge even when Newton-Raphson fails.

APPENDIX

We will first consider three methods of constructing a primal feasible solution for the problem (4.2).

(1) Let the $(2n+1)$st column of A be denoted by a_r. We will put a_r into the basis at level $z_r > 0$ to be determined. Choose n columns from I_n and $-I_n$ to match the first n elements of a_r. Choose m columns from I_m and $-I_m$ to match the next m elements of a_r. If the sum, σ, of these last m activities is not zero, we let $z_r = \frac{1}{\sigma}$, and multiply all other activities by this factor to obtain a basic feasible solution.

(2) Alternatively, an initial feasible basic solution may be constructed by putting the first n columns of A and the first $(m-1)$ columns corresponding to I_m into the basis at 0 level, and putting the m^{th} column corresponding to I_m and the last column of A into the basis at level $1/2$. Note that this method leads to a degenerate solution, so that method (1) is preferable when it can be used.

(3) Finally, instead of one of the explicit constructions above, one may prefer to let the linear programming code do the work by means of a Phase I procedure. Method (2) demonstrates that a primal feasible solution always exists, and method (1) shows that we can generally expect a non-degenerate basic solution to exist, so we can expect good results from a Phase I.

If we were working with a vector control of the same dimension as the system of differential equations, the dual problem (4.1) would always

have a feasible solution. In fact, refering back to equation (3.3), if we replace the term $\mu_j s$ by an arbitrary (error) vector v_j, we can always choose the v_j so as to satisfy (3.3) for any trajectory x_j, $j = 0, \ldots, m$. However, we wish to use a scalar control because of efficiency considerations discussed above, and in this situation it is not clear that (4.1) will always have a solution. If we consider the problem in the form (3.16), we see that a sufficient condition for the existence of a solution to the dual problem is that $m \geq n$ and that n of the h_j, $j = 0, \ldots, m-1$, be linearly independent. In the case $n = 2$, if h_{m-1} and h_{m-2} are well defined (i.e., the required matrix inverses exist) and if the differential equation is given by

$$x = f(x) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} ,$$

where $f(x)$ satisfies $\frac{\partial f_1}{\partial x_1}$, $\frac{\partial f_1}{\partial x_2} \leq 0$ and $\frac{\partial f_2}{\partial x_1}$, $\frac{\partial f_2}{\partial x_2} \geq 0$, with at least one inequality being strict, then h_{m-1} and h_{m-2} are linearly independent. For $n = 3$, we can again conclude that h_{m-1} and h_{m-2} are linearly independent if a similar condition holds, so that if we only place constraints on only 2 components of the terminal vector, we would be guaranteed a dual solution. It appears likely that similar results can be obtained for $n > 3$.

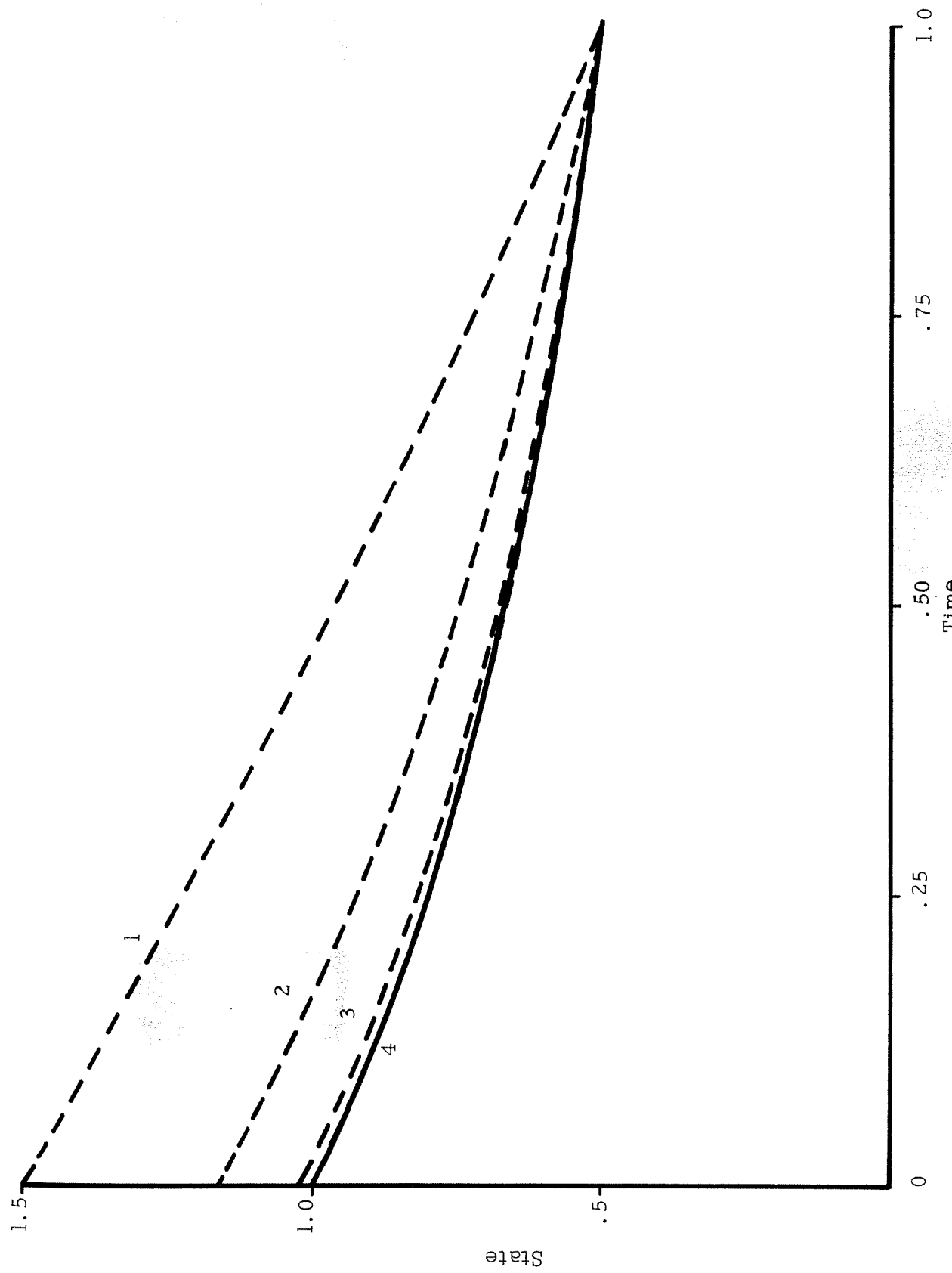

Figure 1. Iterative State Solutions for Example 2. Initial Approximate Trajectory = 0 . Bound on Initial Values = 1.5

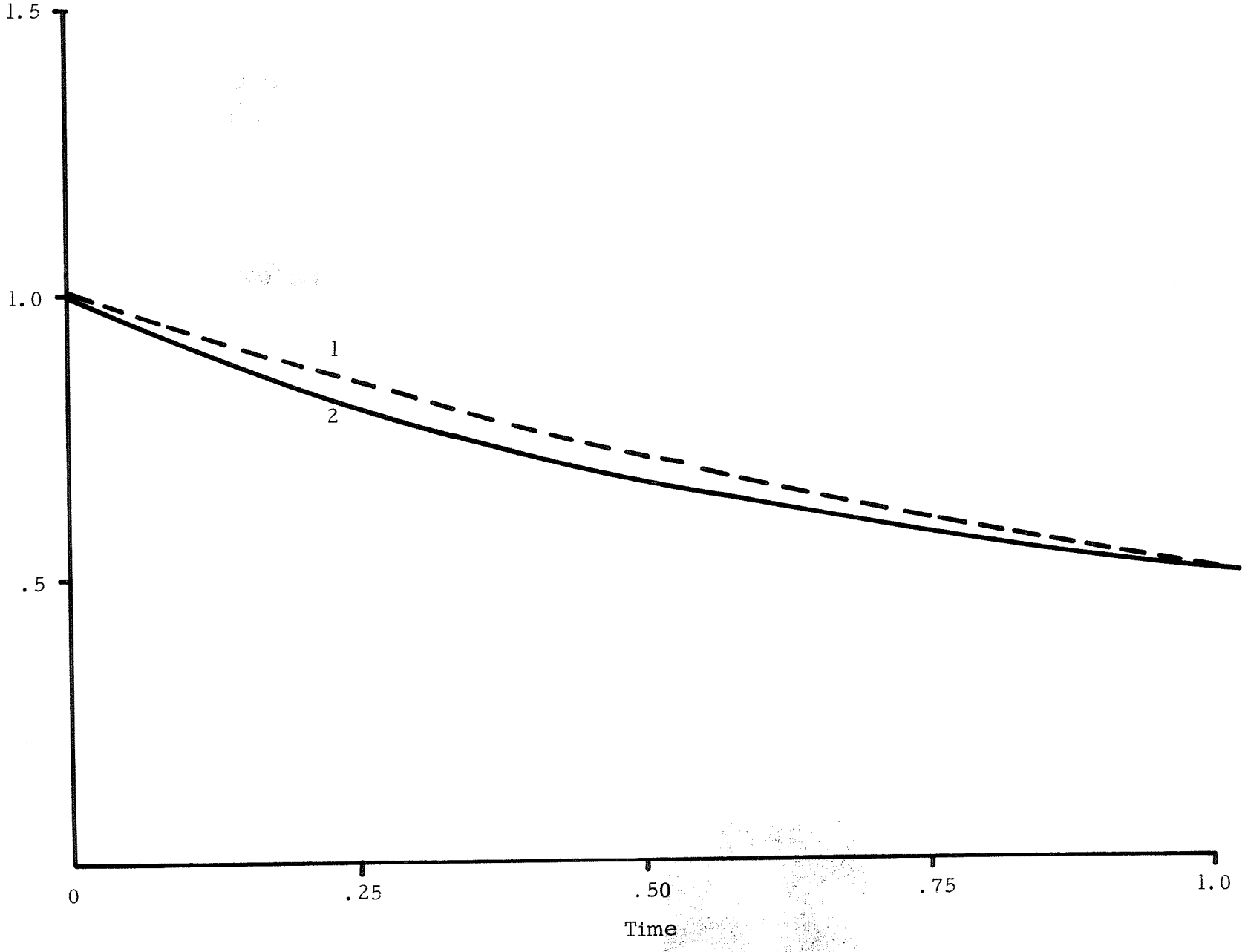

Figure 2. Iterative State Solutions for Example 2. Initial Approximate Trajectory = 0 . Bound on Initial Values = 1.0.

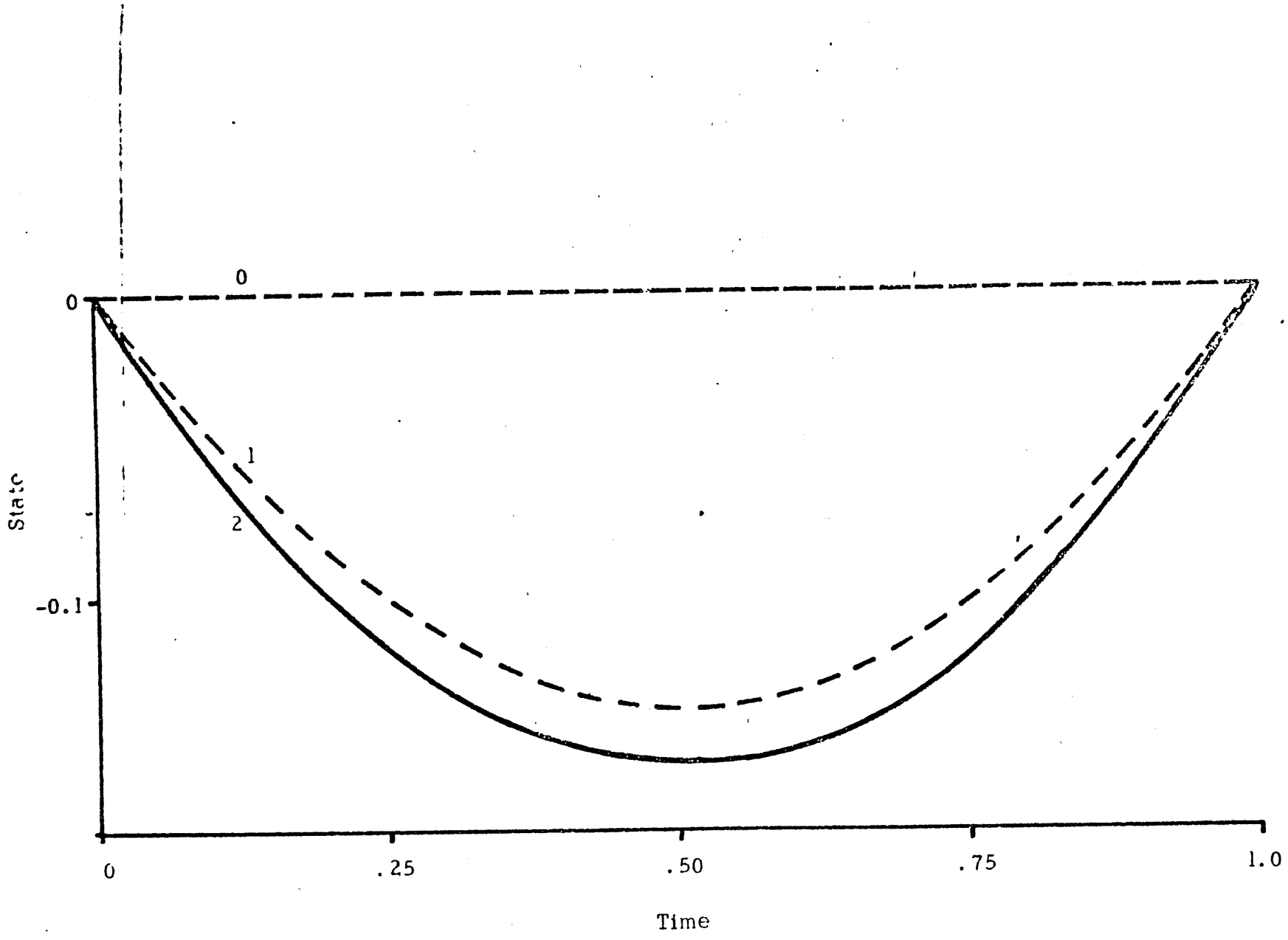

Figure 3. Iterative State Solutions for Example 3, Case 1

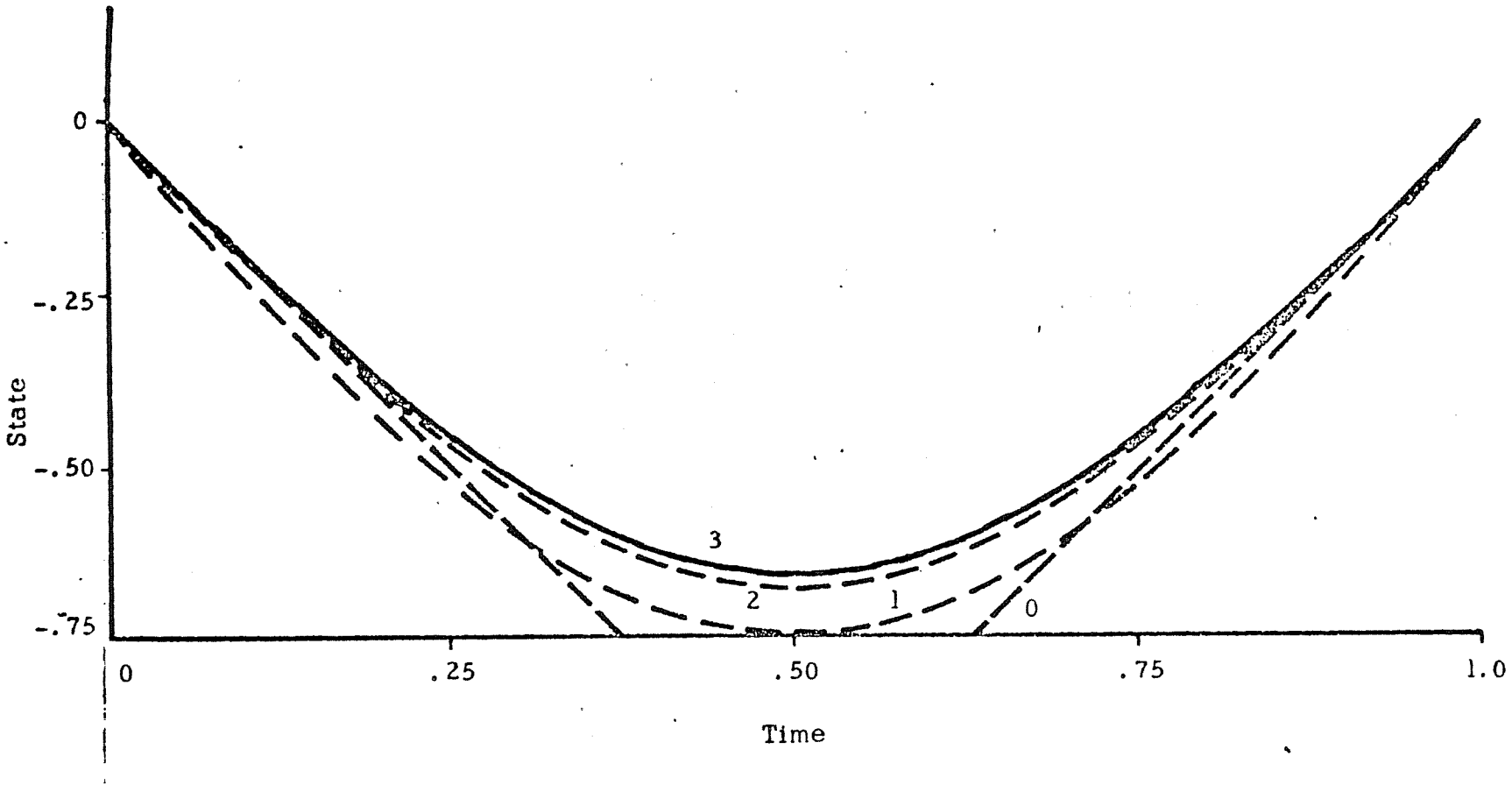

Figure 4. Iterative State Solutions for Example 3, Case 2

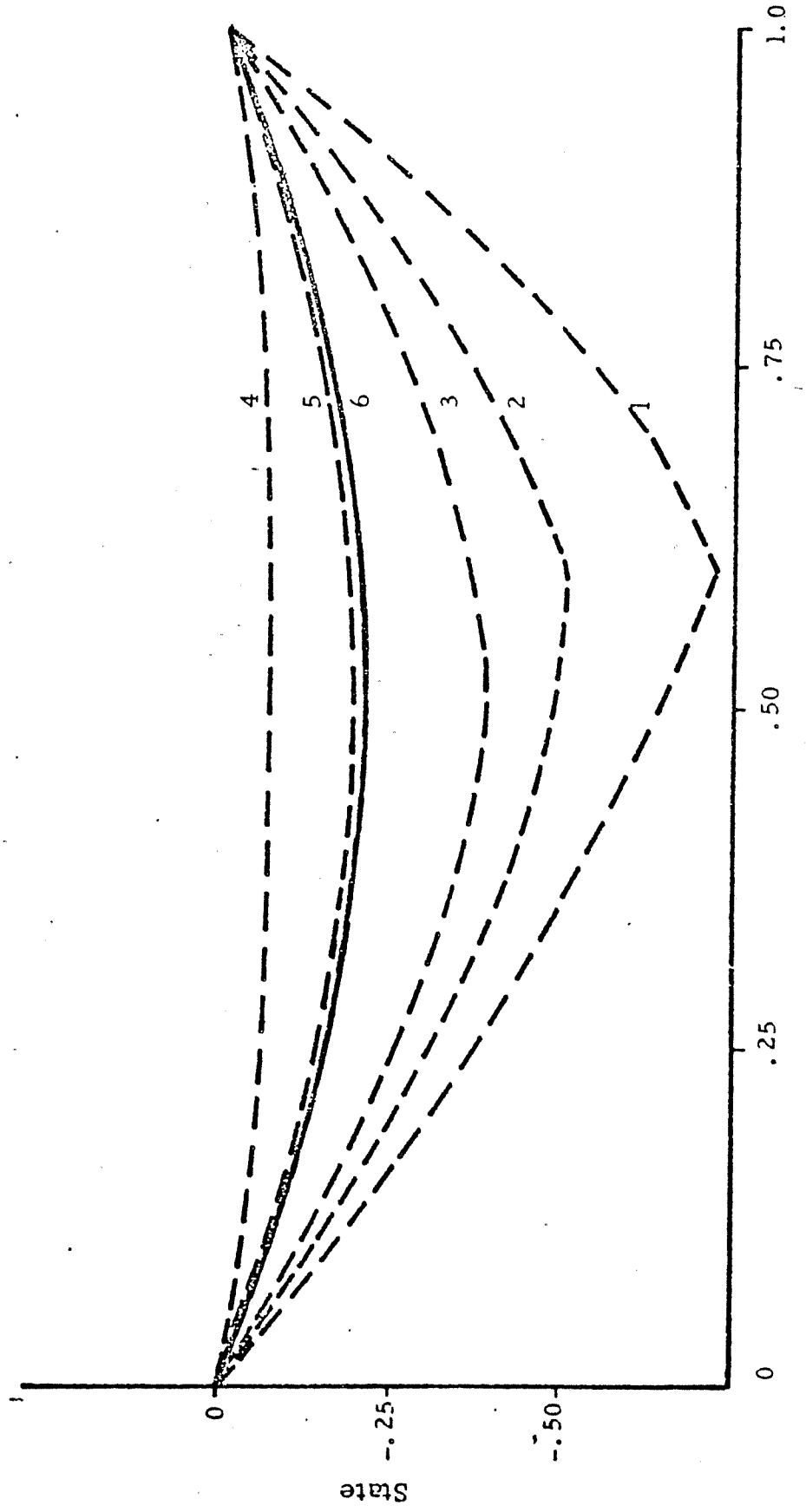

Figure 5. Iterative State Solutions for Example 3, Case 3

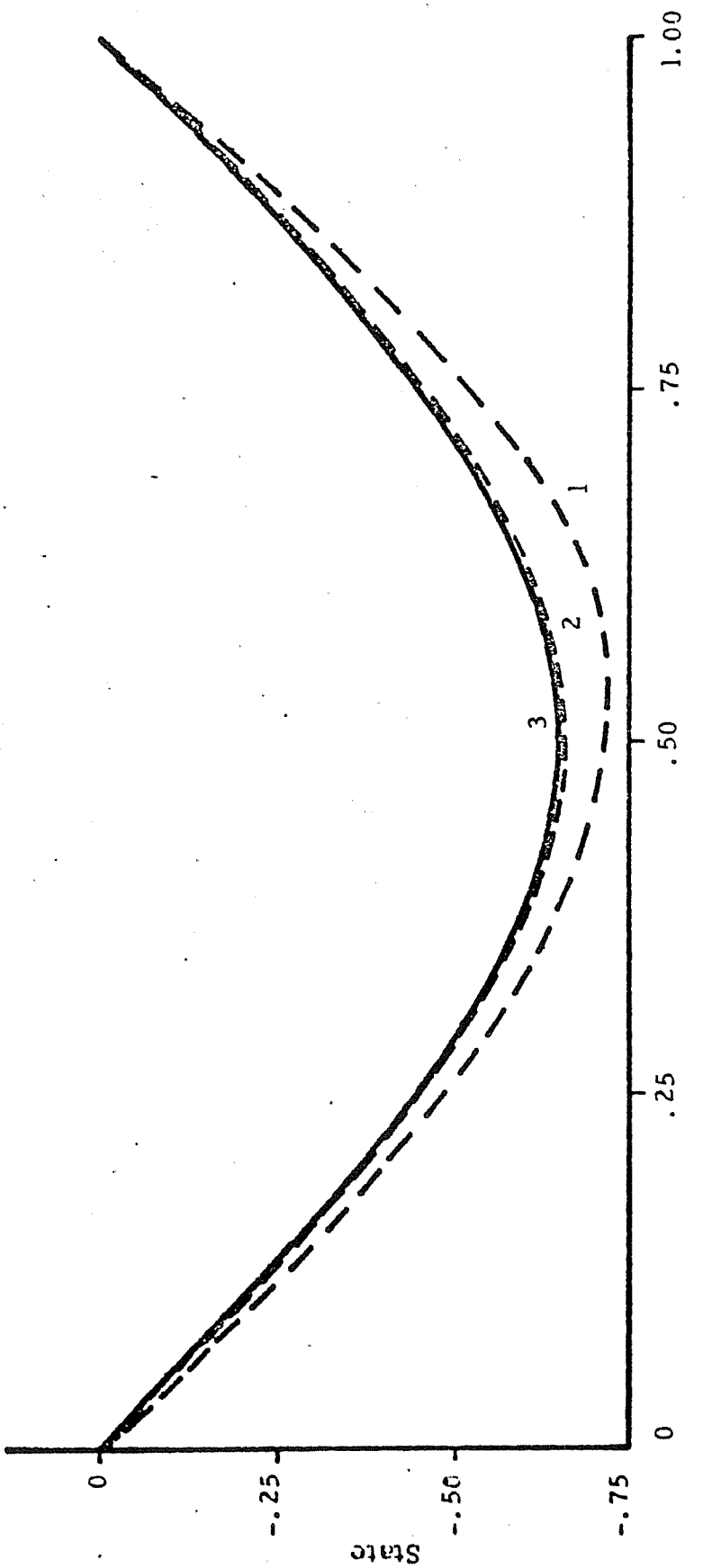

Figure 6. Iterative State Solutions for Example 3, Case 4

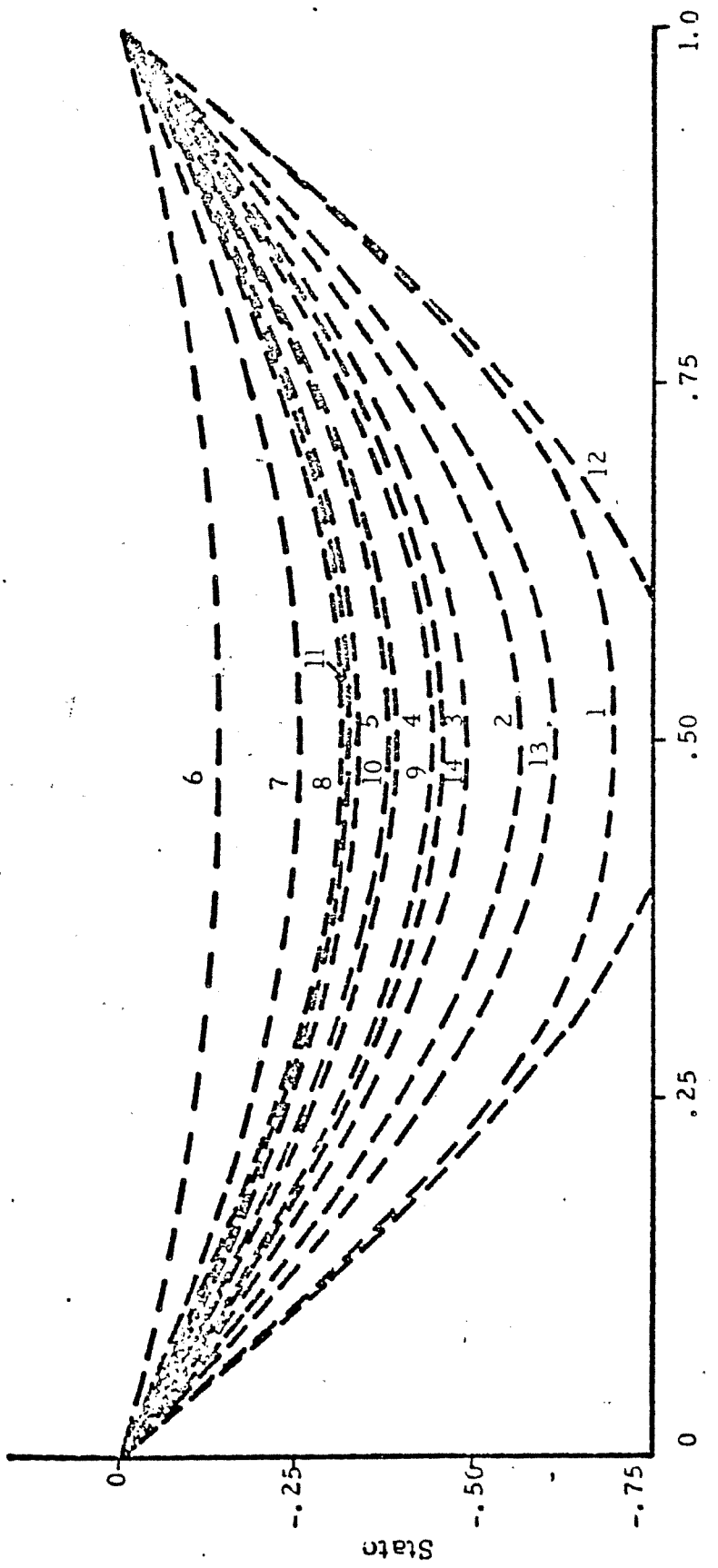

Figure 7. Iterative State Solutions for Example 4. No Convergence.

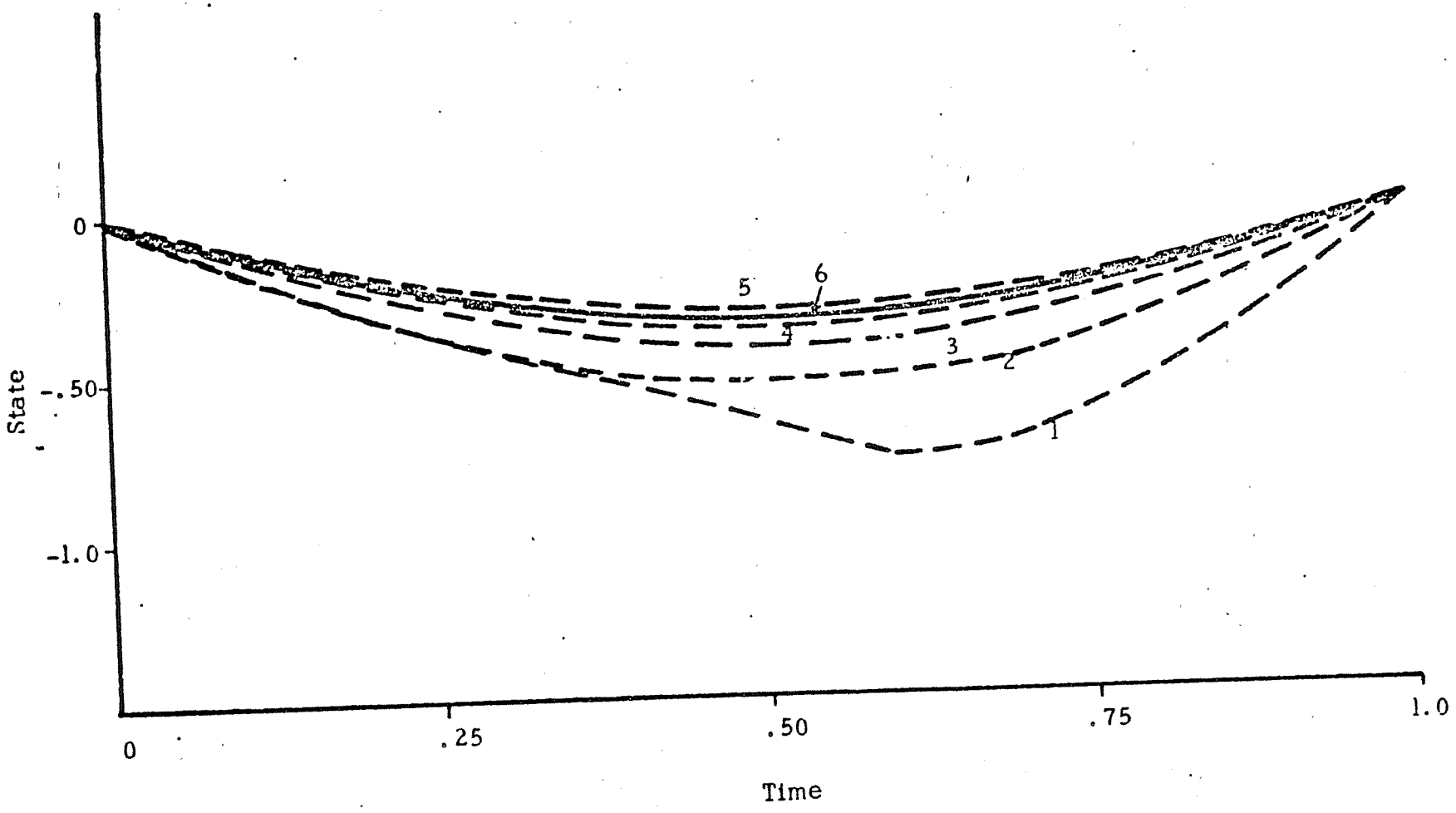

Figure 8. Iterative State Solutions for Example 4. Forced Convergence

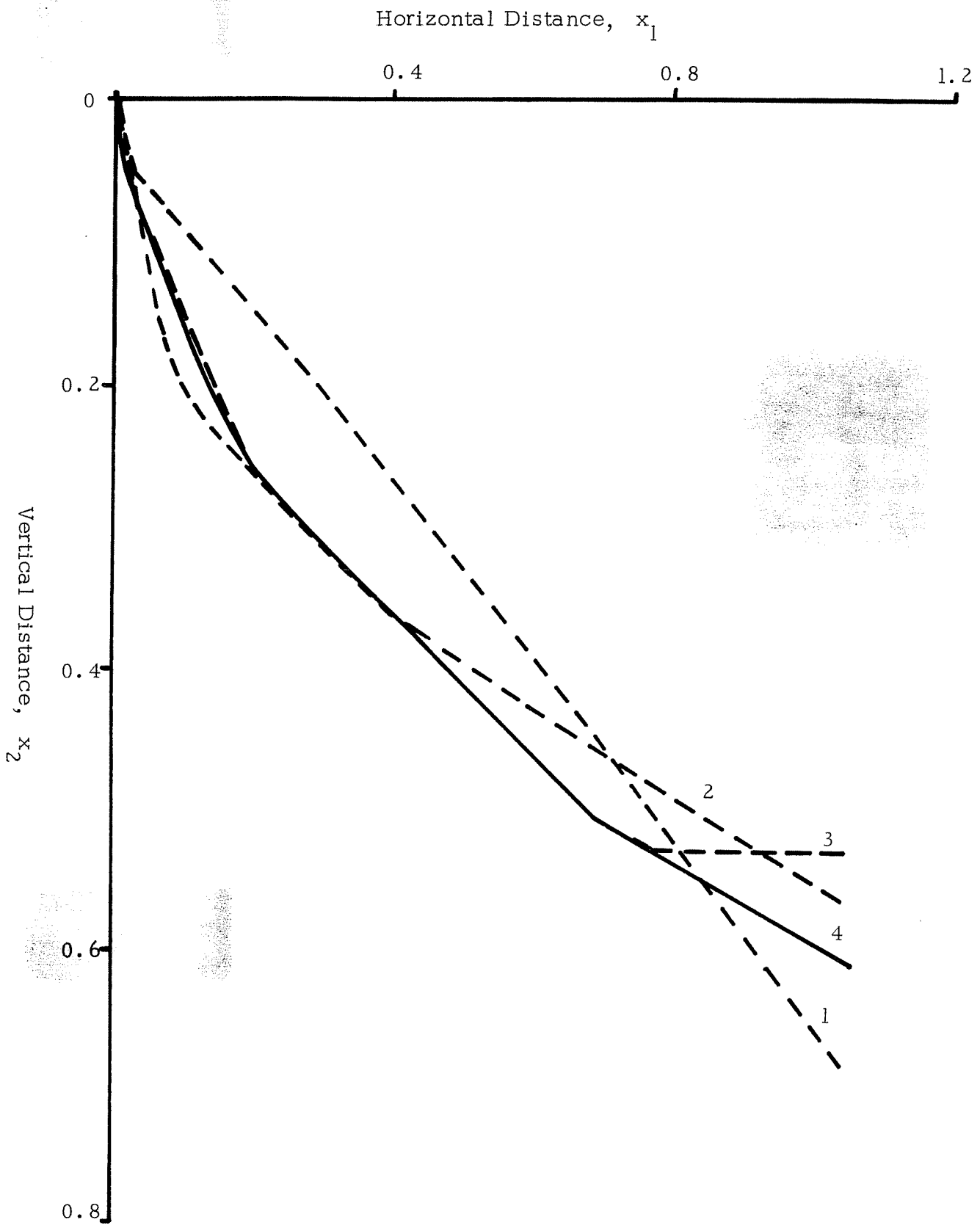

Figure 9. Constrained Brachistochrone Problem Solutions for First 4 Iterations

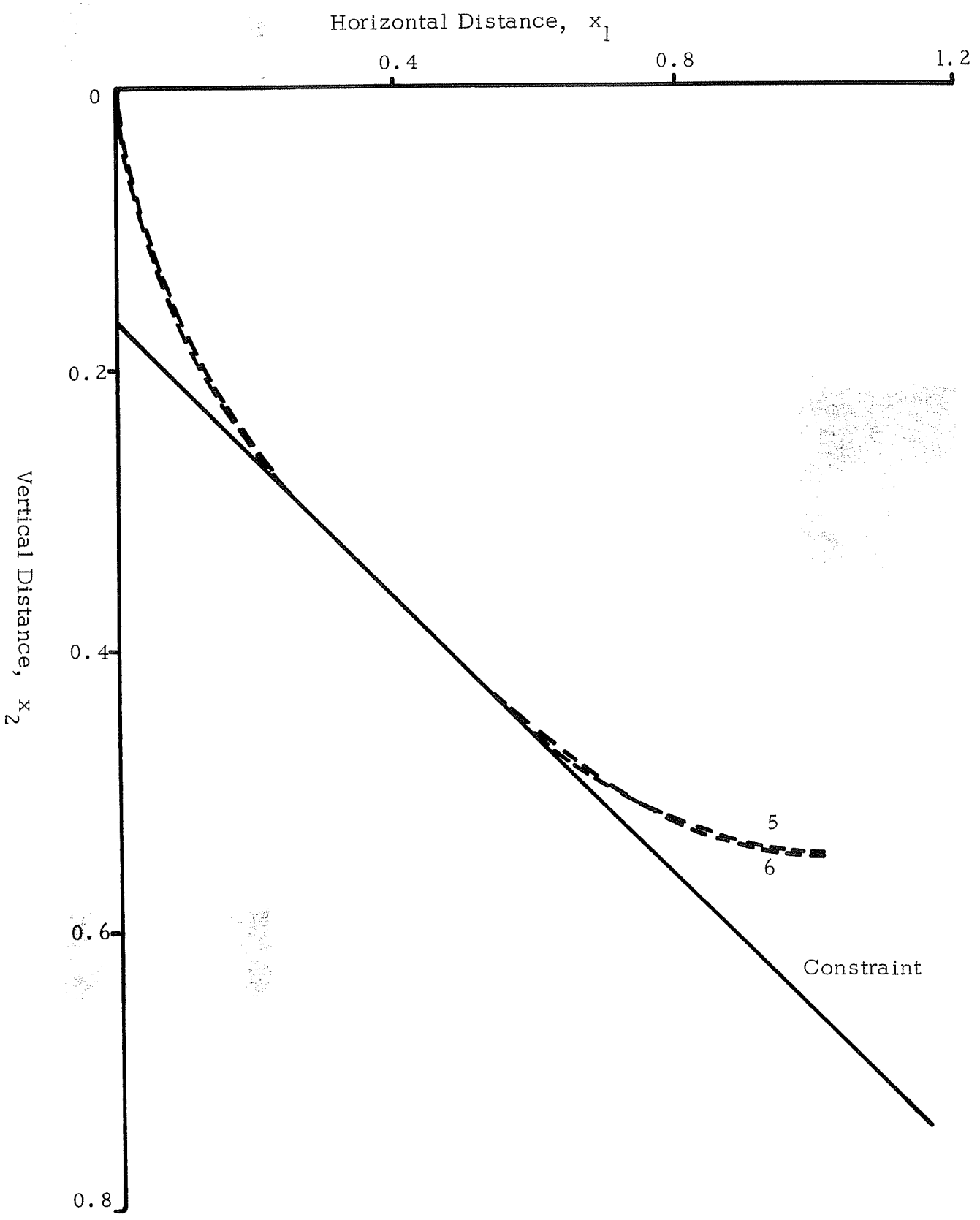

Figure 10. Constrained Brachistochrone Problem. Final Iterations and Optimal Trajectory

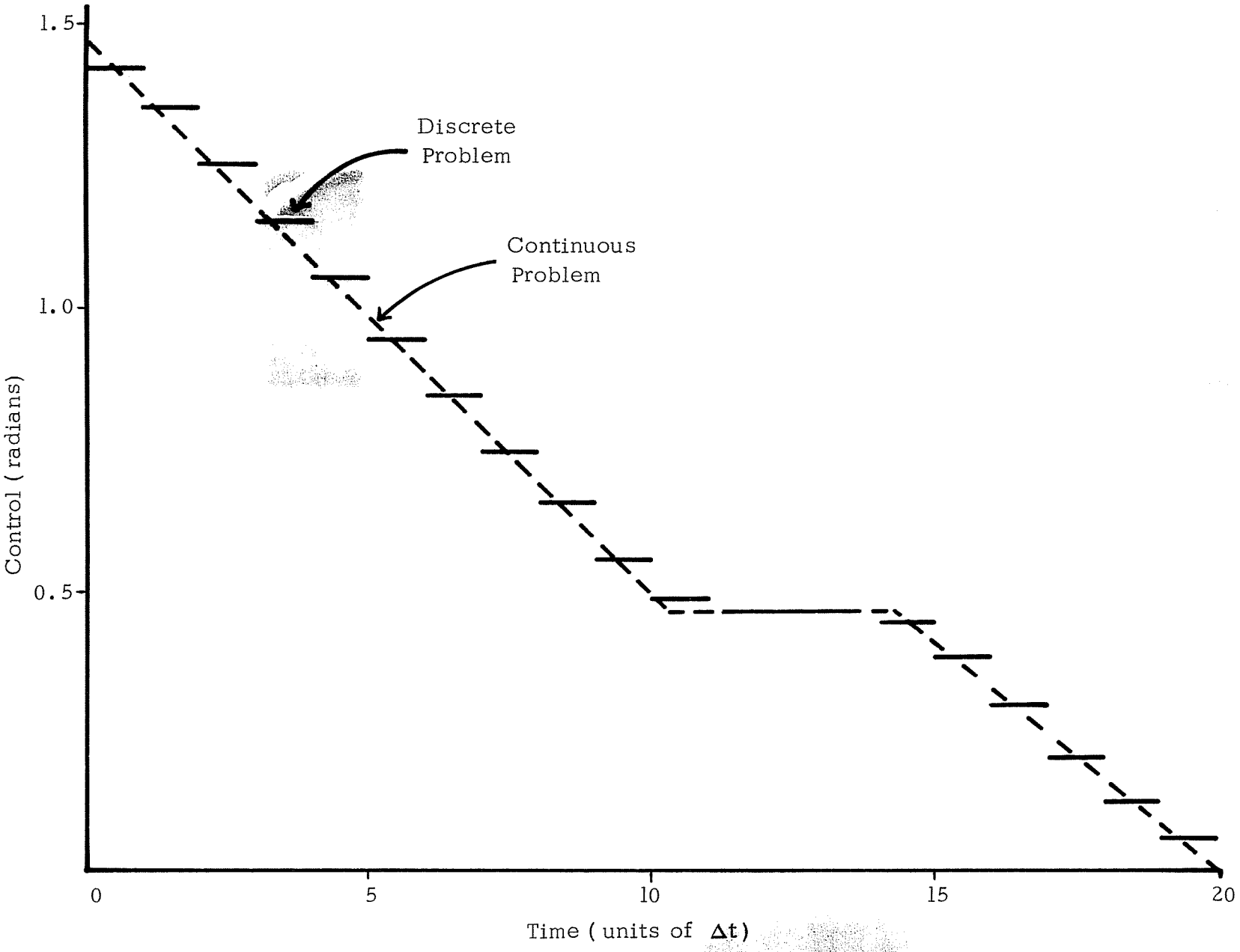

Figure 11. Comparison of Computed Discrete Optimal Control and Continuous Problem Optimal Control

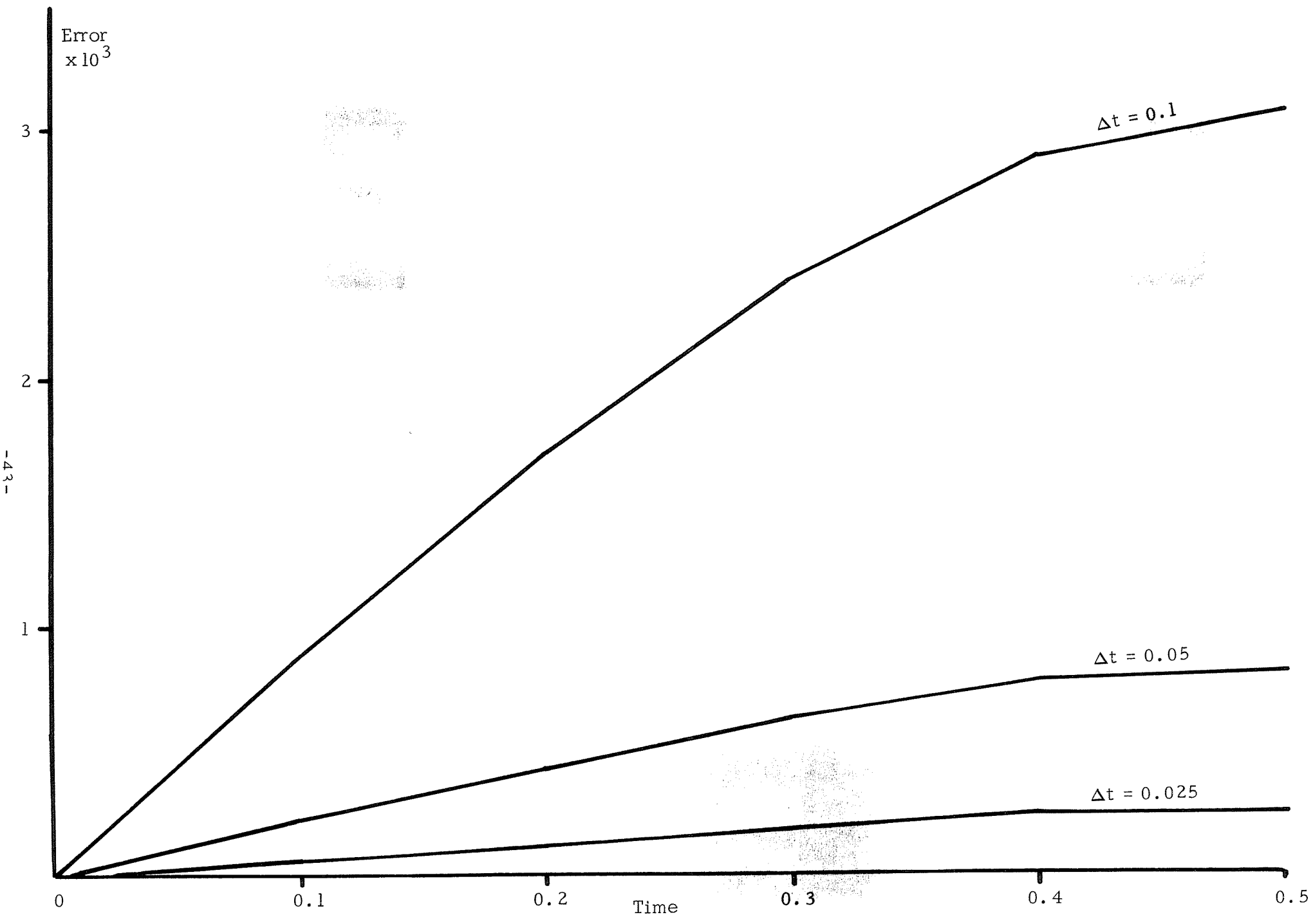

Figure 12. Comparison of Error for Different Grid Sizes. Example 3. Error = Computed - Exact

BIBLIOGRAPHY

1. Bryson, A. E., Denham, W. F., and Dreyfus, S. E., "Optimal Programming Problems with Inequality Constraints I: Necessary Conditions for Extremal Solutions," AIAA J., 1 (1963), pp. 2544-2550.

2. Fox, L., The Numerical Solution of Two-Point Boundary Problems in Ordinary Differential Equations, Oxford Univ. Press, 1957.

3. Griffith, R. E. and Stewart R. A., "A Nonlinear Programming Technique for the Optimization of Continuous Processing Systems," Management Science, 7 (1961), pp. 379-392.

4. Hadley, G., Linear Programming, Addison-Wesley, 1963.

5. Henrici, P., Discrete Variable Methods in Ordinary Differential Equations, John Wiley & Sons, 1962.

6. Kalaba, R., "On Nonlinear Differential Equations, The Maximum Operation, and Monotone Convergence," J. Math. Mech., 8 (1959), pp. 519-574.

7. Lees, Milton, "Discrete Methods for Nonlinear Two-Point Boundary Value Problems," Numerical Solution of Partial Differential Equations, James Bramble (Ed.), Academic Press, 1966.

8. McGill, R. and Kenneth, P., "A Convergence Theorem on the Iterative Solution of Nonlinear Two-Point Boundary Value Systems," XIVth International Astronautical Federation Congress, Paris, France, 1963.

9. McGill, R. and Kenneth, P., "Two-Point Boundary Value Problem Techniques," Advances in Control Systems, Vol. 3, C. T. Leondes (Ed.), Academic Press (1966), pp. 69-109.

10. Rosen, J. B., "Optimal Control and Convex Programming," Proceedings of the IBM Scientific Computing Symposium on Control Theory and Applications, IBM (1966).

11. Rosen, J. B., "Iterative Solution of Nonlinear Optimal Control Problems," J. SIAM Control, Ser. A, 4, pp. 223-244, (1966).