

A CONTINUUM APPROACH TO MINIMUM TIME CONTROL

by

KOTHANDARAMAN RAJENDRAN

B.E., University of Madras, INDIA, 1984

A MASTERS THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

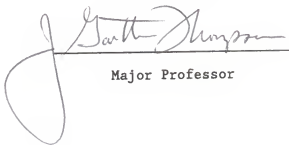
Department of Mechanical Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1988

Approved by:

A handwritten signature in dark ink, appearing to read "Earl Thompson", is written over a horizontal line. Below the line, the text "Major Professor" is printed.

Major Professor

TABLE OF CONTENTS

LIST OF TABLES	ii
LIST OF ILLUSTRATIONS	iii
ACKNOWLEDGEMENTS	iv
Chapter	
I. INTRODUCTION	1
II. OPTIMAL CONTROL THEORY	15
III. THE CONTINUUM APPROACH	26
IV. THE FINITE ELEMENT MODEL	37
V. RESULTS AND CONCLUSIONS	59
.	
LIST OF REFERENCES	70
APPENDICES	73
APPENDIX I	73
APPENDIX II	76

LIST OF TABLES

5.1a	Approximate Solution from the Continuum Approach for the 5x5 Grid	64
5.1b	Actual Solution from the Continuum Approach for the 5x5 Grid	64
5.2a	Approximate Solution from the Continuum Approach for the 10x10 Grid	65
5.2b	Actual Solution from the Continuum Approach for the 10x10 Grid	65
5.3a	Approximate Solution from the Continuum Approach for the 15x15 Grid	66
5.3b	Actual Solution from the Continuum Approach for the 15x15 Grid	67
5.4a	Approximate Solution from the Continuum Approach for the 21x21 Grid	68
5.4b	Actual Solution from the Continuum Approach for the 21x21 Grid	69

LIST OF ILLUSTRATIONS

4.1	Numbering of Nodes	40
4.2a	Basic Element Shape	41
4.2b	Nodal Values of Final Time	41
4.3	Division of Triangle for Integration Purposes	47
5.1	Isochrone Distribution from the 21x21 Finite Element Grid .	63
5.2	Isochrone Distribution from the 41x41 Finite Element Grid .	63

ACKNOWLEDGEMENTS

I would like to thank the following people for their assistance and contributions to this thesis: Dr. Warren N. White, Jr., whose guidance and enthusiasm made this work possible, Dr. J. Garth Thompson for his guidance in optimal control theory concepts and for serving on my committee, Dr. Prakash Krishnaswami for his guidance in optimization techniques and for serving on my committee, Dr. Ruth A. Dyer for serving on my committee, the Department of Mechanical Engineering, Kansas State University, and my fellow students for their encouragement and support.

CHAPTER 1

INTRODUCTION

Human beings have been trying to do things in the most efficient way possible since the beginning of civilization. From the pre-historic period to the present age the accomplishment of tasks with the least expenditure of energy or time, has been a motivating factor for expanding the technological base.

At the beginning of the industrialized age, human labor became an integral part of what was then "the art of manufacturing." A few people were able to produce products at a higher rate than others and they were called "experts." F.W.Taylor [1], in the early 1900's, realized that there were certain motions used by these experts that made them perform their tasks faster than others. Taylor fashioned and promoted the idea of making others follow similar motions to optimize the production rate and, hence, developed the science of work study, which played an important role in the optimization of production in flow line systems of manufacturing.

As technology changed and flow lines gave way to fully automated lines, the need to use the most efficient movements of machines has become a primary concern. The branch of mathematics known as "optimization" has found many new applications in tuning the

production processes. Optimization mathematics facilitated design of manufacturing machinery which operated at increasingly higher rates and efficiencies. However, a drawback of this fixed manufacturing was that in order to manufacture a different product, the entire assembly line had to be re-tooled to accomodate the new product. With the introduction of computers to the manufacturing environment it became possible to route different products through the same line with minimal changes. Although computer control made the transition from one production schedule to another relatively easy, the need to optimize the production of a single product still remained.

There is a cost of manufacturing associated with any production schedule. The first task in optimizing or minimizing cost is to express cost as a function of those quantities which we are free to vary and can control. The two most accessable of these quantities are energy and time consumption. We can minimize consumption of energy by reducing the rate of production or by selecting more efficient drives. Obviously, reducing production rates too greatly has the undesirable consequence of curtailing output. Minimizing the consumption of time can be achieved by minimizing the cycle time of the individual components of the manufacturing system as discussed above. Time of production, subject to the available resources, provides a viable area in which to achieve lower costs. This can be done by optimizing the cycle time of the individual components of the manufacturing system.

One avenue through which the optimization or in this case reduction of cycle time can be achieved is to increase the speed at

which the production machinery or, in the context of this work, robots operate. The problem of cycle time optimization of mechanical manipulators falls into the general category of minimum time control problems with or without path constraints. Parenthetically, production operations have path constraints in order to avoid collisions with the assembled part or other obstacles in the work space. On the other hand loading and unloading tasks in metal working operations and missile interception problems are examples of tasks without path constraints. Extensive work has been done in determining the path constrained, minimum time motion [2] - [5]. This work will deal with the non-path-constrained minimum time problem.

Time optimal control without path constraints can be defined as the control function which forces a system, described by a system equation and having bounded control, from one point to another in minimum time. The system equation is normally a differential equation. Systems which are defined by linear differential equations are called linear systems and those defined by nonlinear differential equations are called nonlinear systems. The early development of time optimal control problems without path constraints was done almost exclusively in terms of linear systems. Nonlinear systems were later solved by using some of the techniques developed in the solution for linear systems. Some of the solution techniques developed for linear and nonlinear systems will be presented here. It should be noted that even though the theory might look simple, in practice it is difficult to apply these solution techniques to all systems.

LITERATURE SURVEY

LINEAR SYSTEMS

The solution techniques developed for linear systems can be broadly classified as analytical and numerical.

ANALYTICAL MODELS

The early development of the solution techniques for linear systems involved finding the analytical solution. Significant among the early analytical models was that presented by Athanassiades and Smith [6]. Athanassiades and Smith developed a minimum time control system for an Nth order linear system with negative real poles. They transformed the state equations into a diagonalized form, solved for time as a logarithmic function and obtained the switching hypersurface. Based on the position of the state point relative to the switching surface, they were able to determine the control. The control they determined would drive the system from the initial state to the phase space origin in the least time possible given the capacity of the available drives. At about the same time, working independently, Desoer and Wing [7] came up with a similar technique for discrete systems.

Later, Ryan [8] developed algebraic expressions for the minimum-time isochronal surfaces for third order systems with real eigenvalues and a single saturating input. He integrated the system equations using simplified controls and solved for the final time. This becomes difficult for higher order and nonlinear systems.

Another interesting approach was presented by Fuller [9]. He computed the time-optimal control required for points where all but one coordinate are zero by backward tracing of the trajectories for a linear plant with distinct real eigenvalues and a scalar saturating control. The trajectory was divided into m time intervals at which the control changes sign. Backward integration of the system equations from the origin then provides the optimal control. He suggested that for state points away from the axes the control can be determined from the dominant state coordinate. However there is a possibility of chatter near the switching curve in this case.

Sebakhy and Abdel-Moneim [10] and Rao and Janakiram [11] presented algorithms to compute the time-optimal control laws to drive closed loop discrete linear systems to the origin of the phase space. Sebaky and Abdel-Moneim defined the system by

$$\underline{x}(t+1) = \underline{A} \underline{x}(t) + \underline{b} \underline{u}(t) \quad (1.1)$$

$$\underline{u}(t) = \underline{F} \underline{x}(t) \quad (1.2)$$

where \underline{x} is an n dimensional state vector, \underline{u} is the control vector, and \underline{A} , \underline{b} and \underline{F} are real matrices. The system was assumed to be controllable. They computed the feedback law \underline{F} from

$$(\underline{A} + \underline{b} \underline{F})^r = 0 \quad (1.3)$$

such that r is minimum.

Rao and Janakiraman presented a solution technique for discrete systems specified in the z - domain of the form,

$$\frac{X(z)}{M(z)} = \frac{\sum_{r=0}^p a_r z^r}{\sum_{i=0}^n b_i z^i} \quad ; p < n \quad (1.4)$$

where $X(z)$ and $M(z)$ are the z - transforms of the output and input respectively. They defined the state variables as the output at n consecutive time intervals and solved for the input as

$$[a] [m] = -[b] [x(0)] \quad (1.5)$$

giving,

$$[m] = -[a]^{-1} [b] [x(0)] \quad (1.6)$$

where $[a]$ and $[b]$ were obtained by converting the system to its difference equations. In cases where there was a saturation limit on the controls, the input was set at that limit.

NUMERICAL MODELS

With the advent of the computer, numerical solution techniques became popular. The solution methodologies for time-optimal control problems were directed towards iterative techniques.

Knudsen [12] developed an iterative procedure for computing the time-optimal controls for the generalized state equations of the form

$$\dot{x} = A x + b u \quad (1.7)$$

where A and b are constant matrices, x the state vector, \dot{x} the rate of change of the state vector, and u is the control driving the system. The control satisfies the inequality constraint of

$$|u| \leq 1. \quad (1.8)$$

He parameterized the minimum time control in terms of the initial conditions of the system's adjoint equations. Knudsen derived a function relating the initial state of the system to the adjoint system's initial condition and to the time required by the optimal control to drive the state to zero. This function was used to solve the problem iteratively. Nagata, Kodama and Kumagai [13] developed an iterative procedure to solve discrete state variable formulations.

Lastman [14] developed a shooting method for two point boundary value problems arising from bang-bang control problems. He guessed the final time and the initial values of the multipliers and iteratively improved the solution by integrating forward in time. He also showed that nonlinear systems can be solved by this method.

Another interesting numerical technique developed by Larson [15] and Yastreboff [16] consisted of iteratively changing the time interval between switchings. Larson assumed arbitrary switching intervals and controls and determined an initial trajectory. He then determined a sequence of trajectories by correcting the previous trajectory so that the respective sequence of end points of the trajectories will approach the desired point in state space. The correction routine used a first order Taylor series approximation about the present switching interval times and assured that each trajectory was time optimal from the initial to whatever final point it reached. The corrections were proportional to the distance by which the final state was missed. It should be noted that the terms neglected in the Taylor series expansion are not necessarily small.

The optimal control was obtained when the correction routine brought the end point of the trajectory near the desired end point in state space. The method was applied to linear systems with complex eigen values and to time varying systems.

Yastreboff [16] formulated his time interval adjustment technique for linear plants with constrained control amplitudes. He arbitrarily chose n switching times and a control function which brought the plant to the desired terminal state. The switching times and control are then adjusted so that the controls approximate a bang-bang form. He adjusted the control by a linearization technique and a logarithmic technique. The linearization technique was actually a variational technique. He took the first variation of the system equation integrated forward in time and set the variation depending upon the final state to zero. The problem was then solved iteratively. The logarithmic technique consisted of approximating the variations with logarithmic functions. The logarithmic approach converged faster, but determining this function for other systems would be a difficult task.

NONLINEAR SYSTEMS

Most physical systems are nonlinear in nature. The solution techniques developed for linear systems are useful in that they can be used to solve nonlinear systems by linearized approximations in the case of analytical solutions or by extending them directly in the case of numerical techniques as in Lastman [14].

LINEARIZED APPROXIMATIONS

Notable among the linearized approximation techniques used was that developed by Kahn and Roth [17]. They investigated a three degree of freedom manipulator and obtained a suboptimal solution to the minimum time problem by using a linearized set of the equations of motion. The equations were decoupled using suitable transformations. Approximations were made to the gravity and velocity terms to reduce the problem to double integrator formulations on each axis. The method gave reasonably good results on those axes which were loosely coupled with others, but on the axes which had a higher degree of coupling, the error between the actual solution and the solution given by the linearized equations were as high as 68%. It should, however, be noted that the state equations were simultaneous coupled nonlinear differential equations.

Wen and Desrochers [18] also presented two linearized solution techniques, namely, the method of averaging dynamics and the method of linear equivalence. The method of averaging dynamics assumed that the non-linear structure was constant in time and used a double integrator solution based on this structure. However, velocity constraints could not be enforced. The method of linear equivalence assumed that the non-linear structure was known and the system was decoupled into double integrators. The torque was calculated by using the non-linear structure.

NUMERICAL TECHNIQUES

Analytical solutions for nonlinear systems are very difficult to obtain. Numerical techniques are easier to apply, however, computational time is a major concern in this area.

Kahn and Roth [17] presented a numerical solution technique to the complete nonlinear problem to verify their linearized equations. The method involved making a guess on the unknown values of the adjoint system variables at the final time and integrating both the state and the costate equations back in time until the initial state was reached. Iteratively changing this guess on the final value of the costate variables produced the exact control. This, however, cannot be used for an on-line control scheme which is required in most modern production systems due to the length of the calculation.

Shetty [19] presented a finite element approach to the minimum time problem of a two degree of freedom robot. He formulated a finite element model of the optimization equations obtained from the Hamiltonian. The position and velocities of the two axes, the multipliers and their first derivatives and the elemental time were used as unknown variables. He used a grid search method to get a reasonably good guess on the final time and used this in the finite element model. The nodal variables and the elemental time were iteratively improved. He used a continuous time method similar to the technique presented by Lastman [14] to verify his results.

Davison and Monro [20] and Wen and Desrochers [21] presented time interval optimization techniques for nonlinear systems. Davison

and Monro presented a computational technique for finding the time-optimal controls of non-linear time varying systems with single inputs. They assumed the number of switchings and the time of each interval and formulated a composite criterion function, which was minimized to get the switching times. They assumed an arbitrary number of switching intervals N for the control input $u(t)$ and made initial guesses on the switching intervals T_i 's and the initial control. The response was calculated based on these values and the values of N and T 's were refined using Rosenbrock's Method [22] and a composite cost function of the form,

$$J = c T + \underline{x}^T(t) \underline{x}(t) \quad (1.9)$$

where c is some weighting factor (usually $c=1$) and

$$T = \sum_{i=1}^N T_i. \quad (1.10)$$

The time interval optimization technique presented by Wen and Desrochers [21] was developed for nonlinear systems with multiple inputs. This is significant due to the fact that references [15], [16], and [20] considered only systems with single inputs. They guessed the N switching times (t_1, t_2, \dots, t_N) where a switching time is defined as the time when any one element of the control vector switches. They also guessed the initial control vector, the final time and the order in which the controls switch. They iteratively improved the solutions by change in the final state due to a change in an earlier switching time. The method worked well when the number of

switches assumed is greater than the actual number and with some correct intuitive guesses.

NOVEL METHODS

A few novel methods have also been presented to reduce the computational effort of the minimum-time problem. Goor [23] converted a three degree of freedom robot problem into three separate problems and defined the path based on the slowest axis. The problem was converted to one in which the 'jerk', which was caused by the switching of the controls, was bounded to reduce wear and tear. The solution to this simplified system was found for each of the axis and the control was defined based on the slowest axis. This does not give us the actual minimum time control or use the non-linear dynamics to advantage.

Luh and Shafran [24] presented another interesting approach. They used two successive least-squares-fit approximations to get the isochrones of the system. The minimum-time isochrones were computed in a discrete region by using some known method. The isochrones were approximated by a hyperellipsoidal surface in terms of the minimal time and the state variables. The coefficients of the hyperellipsoidal surfaces were then ordered according to their corresponding minimal times and they were approximated as a set of continuous functions of time. These approximate functions were used to generate the controls. The first two least-squares-fits were done off line and only the approximate function was used for real time computation purposes. The accuracy of the isochronal hypersurfaces,

however, depends on the number of state points chosen for the initial least-squares-fit.

PRESENT APPROACH

The solution techniques available for time optimal control problems without path constraints have been discussed. However, there is one drawback or another in most of these methods. Analytical solutions are extremely difficult to obtain except for very simple systems. The numerical computational techniques work well with linear systems where the state transition matrix is known. Even then convergence to the absolute minimum requires intuitive guessing. In the case of nonlinear systems, the state transition matrix is not available, the integration interval and stability become greater factors, and intuitive guessing becomes extremely difficult. These factors makes numerical techniques more difficult to apply. Sometimes it takes an enormous amount of time and effort to come up with the optimal control and hence it can only be used in path planning. Another important factor to be considered is that the computations have to be repeated for each initial position.

Due to the problem of computational time and repeated computations required for various initial positions, it is necessary to come up with an efficient method for an online control scheme. Analysis of the problem in the state space domain will reduce unnecessary computations for changes in initial positions. Luh and Shafran [24] presented one such approach. However, as stated before, the accuracy of their analysis depended upon the number of state

points chosen. The purpose of this work is to provide an efficient method to solve the complete minimum time problem for real time applications and to examine the feasibility of a continuum approach towards such a solution.

The continuum approach presented here treats the state variables as independent variables in the state space. The final time is then evaluated as a function of the state variables. This choice of variables enables us to treat both linear and nonlinear systems in essentially the same way. The continuum relations have been derived and an approximate solution has been obtained for a double integrator problem using a finite element analysis.

CHAPTER 2

THE TIME OPTIMAL CONTROL PROBLEM

OPTIMAL CONTROL THEORY

Classical control systems design is based on acceptable performances defined in terms of time and frequency domain criteria such as rise time, settling time, peak overshoot, gain, and phase margin, and bandwidth. Modern technology demands complex multiple-input, multiple-output systems with radically different performance criteria. With the development of the digital computer, optimal control theory has been used in the design of these systems.

The objective of optimal control theory [25] is "to determine the control signals that will cause a process to satisfy the physical constraints and at the same time minimize (or maximize) some performance criterion."

In order to evaluate the performance criterion (or index) and to determine the optimal control, the designer must have a complete knowledge of the mathematical description (or model) of the process to be controlled, a statement of the physical constraints, and a specification of the performance index.

THE MATHEMATICAL MODEL

The mathematical modelling of the system is an important aspect of any control problem. The model should be an accurate, but simple description of the physical system and should predict the response to all anticipated inputs within reasonable accuracy. If the states of the system at any time, t , are $x_1(t)$, $x_2(t)$, $x_3(t)$, . . . , $x_n(t)$ and the control inputs to the system at any time, t , are $u_1(t)$, $u_2(t)$, . . . , $u_m(t)$, the system may be described by n first order differential equations of the form

$$\begin{aligned}\dot{x}_1(t) &= a_1(x_1(t), x_2(t), \dots, x_n(t), \\ &\quad u_1(t), u_2(t), \dots, u_m(t), t), \\ \dot{x}_2(t) &= a_2(x_1(t), x_2(t), \dots, x_n(t), \\ &\quad u_1(t), u_2(t), \dots, u_m(t), t), \\ &\vdots \\ \dot{x}_n(t) &= a_n(x_1(t), x_2(t), \dots, x_n(t), \\ &\quad u_1(t), u_2(t), \dots, u_m(t), t).\end{aligned}\tag{2.1}$$

Then, let

$$\underline{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix} \quad (2.2)$$

be defined as the state vector, and let

$$\underline{u}(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_m(t) \end{bmatrix} \quad (2.3)$$

be defined as the control vector. The system can now be defined in the vector form as

$$\dot{\underline{x}}(t) = \underline{f}(\underline{x}(t), \underline{u}(t), t), \quad (2.4)$$

where \underline{f} is a vector consisting of the functions a_1, a_2, \dots, a_n . This is known as the state space representation of the system.

PHYSICAL CONSTRAINTS

The physical constraints on the state variables and the control inputs have to be specified to limit the state space to the required region. Generally, the initial and final states, the range of controls and states, and time may be specified.

THE PERFORMANCE MEASURE

In general, the performance of a system is evaluated by a measure of the form [25]

$$J = h(\underline{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\underline{x}(t), \underline{u}(t), t) dt, \quad (2.5)$$

where t_0 and t_f are the initial and final times, h and g are scalar functions. The final time t_f may be specified or free.

The performance measure is unique for each set of control and state variable values. The 'h' function is generally used to specify constraints to be satisfied at the final time. The 'g' function is used to specify the time varying and control dependent part of the performance index. The performance index of the minimum time control problem, without constraints, can be written as

$$J = \int_{t_0}^{t_f} 1 dt. \quad (2.6)$$

THE OPTIMAL CONTROL PROBLEM

The optimal control problem is to find the admissible control $\underline{u}^*(t)$ which causes the system described by the set of first order differential equations (2.4) to follow an admissible trajectory $\underline{x}^*(t)$ that minimizes (or maximizes) the performance measure defined by (2.5). The control $\underline{u}^*(t)$ is called an optimal control and $\underline{x}^*(t)$ is called an optimal trajectory.

A history of control input values which satisfies the control constraints during the entire time interval from the initial to the final time $[t_0, t_f]$ is called an admissible control.

A history of state values which satisfies the state variable constraints during the entire time interval from the initial to the final time $[t_0, t_f]$ is called an admissible trajectory.

CALCULUS OF VARIATIONS

The calculus of variations is a branch of mathematics that is extremely useful in solving optimization problems. The optimal control problem is to determine the control, which is a function of time, that minimizes a performance measure, which is a function of functions or better known as a functional.

FUNCTIONALS

A functional is best described as a function of a function or functions. By definition [25], "A functional J is a rule of correspondence that assigns to each function x in a certain class Ω a unique real number. Ω is called the domain of the functional, and the set of real numbers associated with the functions in Ω is called the range of the functional."

For example, if

$$y_1 = f_1(x_1, x_2) \quad (2.7)$$

and
$$y_2 = f_2(x_1, x_2) \quad (2.8)$$

where x_1, x_2 are independent variables and f_1, f_2 are scalar functions, then the quantity

$$J = g(y_1, y_2) \quad (2.9)$$

is a functional.

VARIATION OF A FUNCTIONAL

The variation of a functional is analogous to the differential of a function in that it helps in the determination of the extreme values (maximum or minimum) of the functional. The differential, df , of a function, f , of variables, x_1, x_2, \dots, x_n , is given by

$$df = \frac{\partial f}{\partial x_1} dx_1 + \frac{\partial f}{\partial x_2} dx_2 + \dots + \frac{\partial f}{\partial x_n} dx_n. \quad (2.10)$$

Similarly, the variation, δJ , of a functional, J , of functions y_1, y_2, \dots, y_n is given by the relation

$$\delta J = \frac{\partial J}{\partial y_1} \delta y_1 + \frac{\partial J}{\partial y_2} \delta y_2 + \dots + \frac{\partial J}{\partial y_n} \delta y_n. \quad (2.11)$$

FUNDAMENTAL THEOREM OF CALCULUS OF VARIATIONS

The fundamental theorem states that the variation should be zero on an extremal curve, provided there are no bounds on the curves. Mathematically speaking, if \underline{x} is a vector function of t in the class Ω , and $J(\underline{x})$ is a differentiable function of \underline{x} , then

$$\delta J(\underline{x}^*, \delta \underline{x}) = 0 \quad (2.12)$$

for all admissible $\delta \underline{x}$. An admissible $\delta \underline{x}$ is one which satisfies the condition $(\underline{x} + \delta \underline{x}) \in \Omega$. If Ω is a class of continuous functions, then \underline{x} and $\delta \underline{x}$ must both be continuous.

CONSTRAINED MINIMIZATION OF FUNCTIONALS

So far, we have discussed functionals involving the state vector, $\underline{x}(t)$, with the assumption that they are independent. However, this is not the case in control problems where the state trajectory is determined by the state equations (2.4). This involves $(n+m)$ functions, $\underline{x}(t)$ and $\underline{u}(t)$, of which only the m controls are independent. The state vector is dependent on the controls. Constrained functionals are generally minimized by the Lagrangian multiplier method.

THE LAGRANGIAN METHOD FOR CONSTRAINED MINIMIZATION

The Lagrangian method is used to determine the minimization of a functional, J , of the form

$$J(\underline{x}, \underline{u}) = \int_{t_0}^{t_f} g(\underline{x}(t), \underline{u}(t), t) dt, \quad (2.13)$$

where $\underline{x}(t)$ is the state vector of the order n , and $\underline{u}(t)$ is the control vector of the order m , subject to constraints,

$$\underline{c}(\underline{x}(t), \underline{u}(t), t) = 0, \quad (2.14)$$

where \underline{c} is a vector of equality constraints of order n . The method consists of forming an augmented functional

$$\begin{aligned} \bar{J}(\underline{x}(t), \underline{u}(t), \underline{\lambda}(t), t) = & \int_{t_0}^{t_f} [g(\underline{x}(t), \underline{u}(t), t) \\ & + \underline{\lambda}^T(t) \underline{c}(\underline{x}(t), \underline{u}(t), t)] dt, \end{aligned} \quad (2.15)$$

where $\lambda_i(t)$, $i=1, 2, \dots, n$, are called Lagrangian multipliers. The procedure now allows us to find the function $u(t)$ which extremizes equation (2.15) and satisfies the constraints simultaneously. Now, if the constraints are of the form of equation (2.4), then the augmented functional becomes

$$\bar{J}(x(t), u(t), \lambda(t), t) = \int_{t_0}^{t_f} \{g(x(t), u(t), t) + \lambda^T(t) [\dot{x}(t) - f(x(t), u(t), t)]\} dt. \quad (2.16)$$

When the constraints are satisfied the augmented functional, \bar{J} , equals the unconstrained functional, J , for all values of $\lambda(t)$ and time. It should be noted here that the representation of the state equations in equation (2.16) is not conventional. If we define

$$\bar{g}(x(t), \dot{x}(t), u(t), \lambda(t), t) = g(x(t), u(t), t) + \lambda^T(t) [\dot{x}(t) - f(x(t), u(t), t)] \quad (2.17)$$

then,

$$\bar{J}(x(t), \dot{x}(t), u(t), \lambda(t), t) = \int_{t_0}^{t_f} \bar{g}(x(t), \dot{x}(t), u(t), \lambda(t), t) dt. \quad (2.18)$$

The functional, \bar{J} , can be minimized by setting the variation $\delta \bar{J}$ to zero. The variation, $\delta \bar{J}$, after simplifying [Appendix 1] is given by

$$\begin{aligned}
\delta J = & \frac{\partial \bar{g}}{\partial \underline{x}}^T (\underline{x}(t), \dot{\underline{x}}(t), \underline{u}(t), \underline{\lambda}(t), t) \Big|_{t=t_f} d\underline{x}(t_f) \\
& - \left[\frac{\partial \bar{g}}{\partial \underline{x}}^T (\underline{x}(t), \dot{\underline{x}}(t), \underline{u}(t), \underline{\lambda}(t), t) \dot{\underline{x}}(t) \right] \Big|_{t=t_f} dt_f \\
& + \bar{g}(\underline{x}(t), \dot{\underline{x}}(t), \underline{u}(t), \underline{\lambda}(t), t) \Big|_{t=t_f} dt_f \\
& + \int_{t_0}^{t_f} \left(\frac{\partial \bar{g}}{\partial \underline{x}}^T (\underline{x}(t), \dot{\underline{x}}(t), \underline{u}(t), \underline{\lambda}(t), t) \right. \\
& \left. - \frac{d}{dt} \left(\frac{\partial \bar{g}}{\partial \underline{x}}^T (\underline{x}(t), \dot{\underline{x}}(t), \underline{u}(t), \underline{\lambda}(t), t) \right) \right) \delta \underline{x}(t) \\
& + \frac{\partial \bar{g}}{\partial \underline{u}}^T (\underline{x}(t), \dot{\underline{x}}(t), \underline{u}(t), \underline{\lambda}(t), t) \delta \underline{u}(t) \\
& + \frac{\partial \bar{g}}{\partial \underline{\lambda}}^T (\underline{x}(t), \dot{\underline{x}}(t), \underline{u}(t), \underline{\lambda}(t), t) \delta \underline{\lambda}(t) dt. \quad (2.19)
\end{aligned}$$

For an extremal curve

$$\delta J(\underline{x}^*(t), \dot{\underline{x}}^*(t), \underline{u}^*(t), \underline{\lambda}^*(t), t) = 0, \quad (2.20)$$

where * signifies the optimal values. For a minimal time control problem, as stated before by equation (2.6),

$$g(\underline{x}(t), \underline{u}(t), t) = 1. \quad (2.21)$$

Therefore,

$$\bar{g}(\underline{x}(t), \dot{\underline{x}}(t), \underline{u}(t), \underline{\lambda}(t), t) = 1 + \underline{\lambda}^T(t) [\dot{\underline{x}}(t) - \underline{f}(\underline{x}(t), \underline{u}(t), t)]. \quad (2.22)$$

Substituting equation (2.22) into equation (2.19) produces

$$\begin{aligned}
 \delta J = & \lambda^T(t_f) d\mathbf{x}(t_f) + [1 + \lambda^T(t) (\dot{\mathbf{x}}(t) \\
 & - \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)) - \lambda^T(t) \dot{\mathbf{x}}(t)] \Big|_{t=t_f} dt_f \\
 & + \int_{t_0}^{t_f} \{ [-\lambda^T(t) \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}(t), \mathbf{u}(t), t) - \dot{\lambda}^T(t)] \delta \mathbf{x}(t) \\
 & + [-\lambda^T(t) \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{x}(t), \mathbf{u}(t), t)] \delta \mathbf{u}(t) \\
 & + [\dot{\mathbf{x}}(t) - \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)] \delta \lambda(t) \} dt.
 \end{aligned} \tag{2.23}$$

Since $\delta J = 0$, equation (2.23) gives us the necessary conditions for the optimal control, which are,

$$\dot{\mathbf{x}}^*(t) = \mathbf{f}(\mathbf{x}^*(t), \mathbf{u}^*(t), t), \tag{2.24}$$

$$\dot{\lambda}^*(t) = - \frac{\partial \mathbf{f}^T}{\partial \mathbf{x}}(\mathbf{x}^*(t), \mathbf{u}^*(t), t) \lambda^*(t), \tag{2.25}$$

$$\lambda^{*T}(t) \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{x}^*(t), \mathbf{u}^*(t)) = 0, \tag{2.26}$$

$$\text{and} \quad 1 - \lambda^{*T}(t_f) \dot{\mathbf{x}}^*(t_f) = 0. \tag{2.27}$$

The n equations (2.24) are the state equations. The n equations (2.25) are called the co-state equations or the multiplier equations. The m equations (2.26) are the optimality conditions which specify the control in terms of the state and costate variables. The equation (2.27) constitutes the boundary condition on the multipliers or the

transversality equation. The transversality condition provides one equation for the unknown final time.

The above conditions are not sufficient to solve the time optimal control problem. The above equations would lead to an optimal solution that would approach zero as the controls move towards infinity. This would require that the control be constrained. In most practical systems, however, bounds on the control exist in the form of maximum force or torque that can be applied by the actuator.

CHAPTER 3

THE CONTINUUM APPROACH

In Chapter 2 the general time optimal control problem was formulated. The system state equations, the multiplier equations, the transversality equation, and the optimality conditions were derived for a general system. However, bounds on the control were not specified, whereas physical systems have bounded controls. In this chapter the minimum time problem with bounded control will be introduced and the continuum approach will be developed for the analysis of one such problem in the phase space domain. The solution of the system of equations derived by the continuum approach will be presented in Chapter 4.

MINIMUM TIME MOTION WITH BOUNDED CONTROL

As an illustration of the method by which control bounds are treated consider an n^{th} order linear system in the phase variable canonical form

$$\dot{\mathbf{x}}(t) = \mathbf{A} \mathbf{x}(t) + \mathbf{b} u(t) \quad (3.1)$$

where $\mathbf{x}(t)$ is the state vector of order n , $u(t)$ is the scalar control input, and \mathbf{A} is an $n \times n$ matrix of the form

$$\underline{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & . & . & . & 0 \\ 0 & 0 & 1 & 0 & 0 & . & . & . & 0 \\ 0 & 0 & 0 & 1 & 0 & . & . & . & 0 \\ . & . & . & . & . & . & . & . & 0 \\ -a_1 & -a_2 & -a_3 & -a_4 & -a_5 & . & . & . & -a_n \end{bmatrix}, \quad (3.2)$$

and \underline{b} is an $n \times 1$ vector given by

$$\underline{b} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ . \\ . \\ 0 \\ 1 \end{bmatrix}. \quad (3.3)$$

The control magnitude satisfies the constraints

$$-1 \leq u \leq 1. \quad (3.4)$$

The initial and final conditions, $\underline{x}(0)$ and $\underline{x}(t_f)$, are specified. The system described by equation (3.1) leads to an augmented functional, J , of the form

$$J = \int_0^{t_f} \{1 + \underline{\lambda}^T (\dot{\underline{x}} - \underline{A} \underline{x} - \underline{b} u) + \mu^+(u - 1) + \mu^-(-u-1)\} dt \quad (3.5)$$

where $\underline{\lambda}$ is a vector of Lagrangian multipliers of order n and μ^+ and μ^- are non-negative scalar Lagrangian multipliers used to append the control magnitude constraints to the functional. It can be shown [26] that μ^+ is greater than zero when u is on the $+1$ boundary and zero

otherwise. A similar statement can be made for μ^- with respect to the -1 boundary.

The functional, \bar{J} , can be minimized by setting the variation, $\delta\bar{J}$, to zero. The variation, $\delta\bar{J}$, is given by

$$\begin{aligned} \delta\bar{J} = & [1 + \lambda^T (\dot{\underline{x}} - \underline{A} \underline{x} - \underline{b} u) + \mu^+ (u-1) + \mu^- (-u-1)] \Big|_{t_f} dt_f \\ & + \int_0^{t_f} (\lambda^T \delta\dot{\underline{x}} - \lambda^T \underline{A} \delta\underline{x} + (\mu^+ - \mu^- - \lambda^T \underline{b}) \delta u \\ & + (\dot{\underline{x}} - \underline{A} \underline{x} - \underline{b} u) \delta\lambda + (u-1) \delta\mu^+ + (-u-1) \delta\mu^-) dt. \end{aligned} \quad (3.6)$$

The second, third, and fourth terms of the dt_f terms go to zero for all values of t . Integrating by parts, and using $\delta\underline{x}(t_f) = -\dot{\underline{x}} dt_f$ [26], we get

$$\begin{aligned} \delta\bar{J} = & [1 - \lambda^T \dot{\underline{x}}] \Big|_{t_f} + \int_0^{t_f} (\lambda^T \delta\dot{\underline{x}} - \lambda^T \underline{A} \delta\underline{x} + (\mu^+ - \mu^- - \lambda^T \underline{b}) \delta u \\ & + (\dot{\underline{x}} - \underline{A} \underline{x} - \underline{b} u) \delta\lambda + (u-1) \delta\mu^+ + (-u-1) \delta\mu^-) dt. \end{aligned} \quad (3.7)$$

If we choose λ such that the coefficients of $\delta\underline{x}$ go to zero and since the other variations in equation (3.7) are arbitrary, we get

$$1 - \lambda^T \dot{\underline{x}} \Big|_{t=t_f} = 0, \quad (3.8)$$

$$\dot{\lambda} = -\underline{A}^T \lambda, \quad (3.9)$$

$$\mu^+ - \mu^- - \lambda^T \underline{b} = 0, \quad (3.10)$$

and
$$\dot{x} = A x + b u. \quad (3.11)$$

The last two terms under the integral sign in equation (3.7) indicate that the control u must have a magnitude of

$$|u| = 1. \quad (3.12)$$

Equation (3.10) shows that

$$\lambda_n(t) = \mu^+ - \mu^- \quad (3.13)$$

and it is seen that the variations $\delta\mu^+$ and $\delta\mu^-$ are not independent. Combining equation (3.13) with the conditions on the multipliers, μ^+ and μ^- , we get

$$u(t) = \text{sgn}(\lambda_n(t)), \quad (3.14)$$

which is also demonstrated in optimal control texts [25]-[26].

In this work a double integrator problem has been considered. The double integrator may be defined as one in which the control specifies the acceleration of the system. The position may then be calculated by integrating the control twice. The double integrator problem may be defined by an equation of the form (3.1), where

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad (3.15)$$

and

$$b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \quad (3.16)$$

Equations (3.8), (3.9), (3.11), (3.12) and (3.14) give us

$$1 - \lambda_1(t) x_2(t) - \lambda_2(t) u(t) = 0, \quad (3.17)$$

$$\dot{\lambda}_1(t) = 0, \quad (3.18)$$

$$\dot{\lambda}_2(t) = -\lambda_1(t), \quad (3.19)$$

$$\dot{x}_1(t) = x_2(t), \quad (3.20)$$

$$\dot{x}_2(t) = u(t), \quad (3.21)$$

$$|u(t)| = 1, \quad (3.22)$$

$$\text{and} \quad u(t) = \text{sgn}(\lambda_2(t)). \quad (3.23)$$

Equations [3.8], [3.9], [3.11], [3.12], and [3.14] along with the initial and final conditions on the state variables provide us with a problem similar to the two point boundary value problem [TPBVP] that can be solved by numerical techniques. However we lack boundary conditions on the multiplier values.

As stated in Chapter 1 most workers used some sort of a shooting technique to solve the problem. However most of these techniques are time consuming and are unfeasible for the design of an online controller. Shetty [19] has shown that conventional numerical techniques in the time domain are unstable unless the initial estimates are close to the actual solution. The various novel methods discussed in Chapter 1 either require a linear system or do not solve the problem completely. The switching curve, which is a map of the sign of the control at any point in the state space, is the fastest technique and still the best control method but is not possible for

higher order systems because it is hard to visualize and only of limited use for nonlinear systems.

THE CONTINUUM APPROACH

Luh and Shafraan [24] developed a method to determine the control from a least-squares-fit of the distribution of the minimum time isochrones in the phase space. The isochrones are surfaces of constant final time. This type of an analysis provides a convenient method to determine the control for an infinite number of different problems in an efficient manner. Linear systems require a single distribution of isochrones, however, nonlinear systems require one isochrone distribution for each exclusive destination. Whereas Luh and Shafraan determined the distribution from many different solutions of a linear system, the goal in this work is to determine if this isochrone distribution could be determined from some balance relation involving the gradients of the final time.

There are several points which promote this idea and provide an incentive for the work. These are [27]

1. Any exclusive distribution is a function of the state variables exclusively, thus eliminating time as the independent parameter.
2. Since the state variables are the independent parameters, the steps in finding the isochrones are the same for both linear and nonlinear systems.

3. If the isochrone information can be determined quickly then it may be possible to develop a controller based on these principles.

A continuum approach in the phase plane is presented for a double integrator problem.

GOVERNING EQUATIONS

The necessary and sufficient conditions for a minimum time control problem are given by equations (3.8), (3.9), (3.11), (3.12) and (3.13). The co-state vector λ at the initial time is related to the final time by [26],

$$\lambda(t_1) = -\nabla t_f(\mathbf{x}(t_1)), \quad (3.24)$$

where ∇ is the vector gradient operator given by

$$\nabla^T = \left(\frac{\partial}{\partial x_1} \quad \frac{\partial}{\partial x_2} \quad \dots \quad \frac{\partial}{\partial x_n} \right), \quad (3.25)$$

for an n^{th} order system. Equation (3.25) holds over all the permissible regions in the phase space where the gradient exists. Equation (3.24) will be derived for a general system defined by equation (2.4).

The augmented functional, \tilde{J} , for a minimum time control problem driving a system to the origin can be written as

$$\tilde{J} = \int_{t_1}^{t_f} [1 + \lambda^T (\dot{\mathbf{x}} - \mathbf{f}(\mathbf{x}, \mathbf{u}))] dt = \tilde{J}(\mathbf{x}(t_1)) = t_f(\mathbf{x}(t_1)). \quad (3.25.1)$$

This is a continuous function of $\mathbf{x}(t_1)$. Integrating equation (3.25.1) by parts produces

$$\dot{J}(\mathbf{x}(t_i)) = \dot{\lambda}^T \mathbf{x} \left|_{t_i}^{t_f} + \int_{t_i}^{t_f} [1 - \dot{\lambda}^T \mathbf{x} - \dot{\lambda}^T \mathbf{f}(\mathbf{x}, \mathbf{u})] dt. \quad (3.25.2)$$

Applying the gradient operator then produces

$$\nabla J = \nabla t_f(\mathbf{x}(t_i)) = -\dot{\lambda}^T(t_i) + \int_{t_i}^{t_f} \nabla [1 - \dot{\lambda}^T \mathbf{x} - \dot{\lambda}^T \mathbf{f}(\mathbf{x}, \mathbf{u})] dt. \quad (3.25.3)$$

Since ∇ consists of partials with respect to the components of $\mathbf{x}(t_i)$ then the integrand in equation (3.25.3) vanishes for all t not equal to t_i . For the case when t is equal to t_i we have

$$\nabla [1 - \dot{\lambda}^T \mathbf{x} - \dot{\lambda}^T \mathbf{f}(\mathbf{x}, \mathbf{u})] \Big|_{t_i} = -\dot{\lambda}^T - \dot{\lambda}^T \nabla \mathbf{f}(\mathbf{x}, \mathbf{u}) \Big|_{t_i}. \quad (3.25.4)$$

Rearranging equation (3.25.4) produces

$$\dot{\lambda} + [\nabla \mathbf{f}(\mathbf{x}, \mathbf{u})]^T \dot{\lambda} = 0. \quad (3.25.5)$$

which shows that the integrand of equation (3.25.5) vanishes for all t in the interval $(0, t_f)$. Hence we get equation (3.24) from equation (3.25.3). It should be noted that due to the choice of signs of multipliers in equation (3.25), the vector $\dot{\lambda}(t_i)$ points in the direction of the greatest decreasing final time.

The transversality condition equation (3.8) is usually applied only at the final time, but it is true everywhere in the phase space where the gradient exists. The transversality condition evaluated at the initial state is

$$1 + \dot{\mathbf{x}}^T(t_i) \nabla t_f(\mathbf{x}(t_i)) = 0. \quad (3.26)$$

Equation (3.26) is the projection of the gradient of the final time on the time derivative of the state vector. Equation (3.26) is insufficient to uniquely specify the components of the gradient of t_f for systems of order two or greater.

NECESSARY CONDITIONS

In order to develop a method to determine the final time field, $t_f(\mathbf{x}(t_i))$, it is necessary to develop a means of uniquely specifying the gradient of the final time.

The cost function of the minimum time problem is given by

$$t_f = \int_0^{t_f} 1 \, dt, \quad (3.27)$$

which can be written in the augmented form as

$$t_f = \int_0^{t_f} (1 - \lambda^T \dot{\mathbf{x}} + \lambda^T \dot{\mathbf{x}}) dt. \quad (3.28)$$

It should be noted that the final time value is the same as the augmented functional. The first two terms of the integral constitute the transversality equation. Equation (3.28) then reduces to

$$t_f = \int_0^{t_f} \lambda^T \dot{\mathbf{x}} \, dt. \quad (3.29)$$

As stated earlier in this chapter, the double integrator will be used to illustrate the problem. Integrating equation (3.29) by parts for the double integrator and using state and costate equations, produces

$$t_f = \lambda^T \mathbf{x} \Big|_0^{t_f} - \int_0^{t_f} \dot{\lambda}^T \mathbf{x} \, dt,$$

$$\text{or} \quad t_f = \lambda^T x \Big|_0^{t_f} - \int_0^{t_f} (x_1 \dot{\lambda}_1 + x_2 \dot{\lambda}_2) dt.$$

The first term under the integral is zero, and the second term under the integral can be rewritten using the state and costate equations. This gives

$$t_f = \lambda^T x \Big|_0^{t_f} + \int_0^{t_f} \dot{x}_1 \lambda_1 dt.$$

Integrating again, we have

$$t_f = \lambda^T x \Big|_0^{t_f} + x_1 \lambda_1 \Big|_0^{t_f} - \int_0^{t_f} \dot{\lambda}_1 x_1 dt.$$

The term under the integral is zero by equation (3.18). Therefore, we have

$$t_f = (2 x_1 \lambda_1 + x_2 \lambda_2) \Big|_0^{t_f}. \quad (3.30)$$

If the destination is the origin of the phase plane, equation (3.30) becomes

$$t_f = -2 x_1(0) \lambda_1(0) - x_2(0) \lambda_2(0),$$

which gives

$$t_f = [2 x_1(0) \quad x_2(0)] \nabla t_f(x(0)). \quad (3.31)$$

Equation (3.31) is the additional, necessary condition to uniquely specify the gradient.

As stated before, the minimum time problem is solved in the time domain approach by the integration of the functional along the phase trajectory and the functional value is the difference between the final time and the initial time, i.e. the endpoints of the path of integration are the initial and final points. Equation (3.30) shows

that the final time can be described as a function of the local variables at each end point. Now, equation (3.29) can be written as

$$t_f = \int_0^{t_f} \Delta^T \dot{\mathbf{x}} dt,$$

or,

$$t_f = \int_{\mathbf{x}(0)}^{\mathbf{x}(t_f)} \Delta^T d\mathbf{x},$$

or,

$$t_f = - \int_{\mathbf{x}(0)}^{\mathbf{x}(t_f)} \nabla^T t_f d\mathbf{x}, \quad (3.32)$$

which shows that t_f is a potential function. The path of integration is arbitrary and the value depends only on the endpoints of the path of integration.

It is instructive to solve equations (3.26), and (3.30) simultaneously for the components of the gradient of t_f . The solution is

$$\nabla t_f = \frac{1}{(2x_1 u - x_2^2)} \begin{bmatrix} u & -x_2 \\ -x_2 & 2x_1 \end{bmatrix} \begin{bmatrix} t_f \\ -1 \end{bmatrix} \quad (3.33)$$

where it can be seen that the coefficient matrix is singular on the switching curve [26]. Equations (3.26) and (3.30) are used to determine the final time field. These equations are solved by a finite element method [28]. The details of the finite element method are given in Chapter 4.

CHAPTER 4

FINITE ELEMENT MODEL

INTRODUCTION

Engineering problems are invariably nonlinear in nature and closed form solutions are not available. This necessitates the use of some form of numerical technique. Two methods which are commonly used are the finite difference and the finite element methods. The finite element method uses piecewise continuous approximations of the region under consideration. The finite element method is more versatile and easier to use with problems that have irregular geometry or unusual boundary conditions. The finite element method was used in this work because of the success of the method in treating other problems based upon a continuum formulation.

The finite element method discretizes the solution region of a continuum problem into a finite number of elements. The unknown solutions within each element are then expressed in terms of approximating functions. The problem is reduced to one with a finite number of unknowns, the nodal values, and solved.

The advantage of such a formulation is that it reduces the complex problem to a greatly simplified problem at an elemental level. The method provides a variety of different ways in which the elemental

* problem can be formulated. The least-squares method has been used in this work to produce the element equations.

The finite element method has five basic steps. These are,

1. Discretization of the continuum.
2. Specification of the interpolation function.
3. Development of the element equations.
4. Assembly of the element equations to get the system of equations.
5. Application of the boundary conditions and obtaining the solution of the system of equations.

In this Chapter the finite element model of the double integrator problem, for which the continuum relations were derived in Chapter 3, will be presented. The presentation of the results and the pertinent discussions and conclusions will be deferred until Chapter 5.

FINITE ELEMENT ANALYSIS

As stated before, the finite element method basically consists of five steps. In this work the analysis was done with the help of FORTRAN programs executed on both an IBM 370 VM/CMS computer and an IBM PC microcomputer. The program provides for changes in the size of the region under consideration and the size of the grid, but it allows only a rectangular grid with triangular elements. The least squares technique was used in this analysis so as to provide symmetrical element matrices for economical storage and faster solution time. The analysis was performed using single precision arithmetic and is suitable for implementation on microprocessors. The steps in the

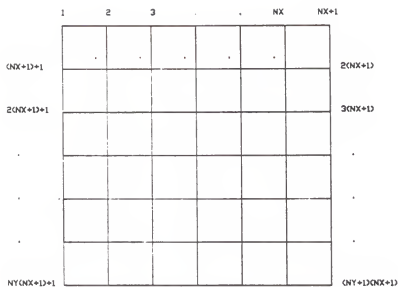
formulation and analysis of the finite element model will be presented. The programs are listed in Appendix 2.

DISCRETIZATION OF THE CONTINUUM

The discretization of the continuum has been done with two FORTRAN subroutines, namely, NODE and GRIDLT. The domain selected is a square region with the origin at the center. The subroutine NODE places equispaced nodes in two directions and numbers them. The nodes are numbered horizontally if the number of nodes in the y-direction is greater than or equal to the number of nodes in the x-direction or vertically downwards otherwise (Figure (4.1)). This reduces the memory required when the element equations are stored in banded matrix form. It should be noted that there is no overall advantage to any method when the number of nodes is the same in both directions. The nodes divide the region into a regular and uniform grid. The subroutine GRIDLT divides the grid into triangular elements of the same size and forms a table, called the node table, which contains the node numbers of each element. The node table has its elements specified in the counterclockwise direction. This is shown in Figure (4.2). The triangular element was chosen for ease of application.

INTERPOLATION FUNCTION

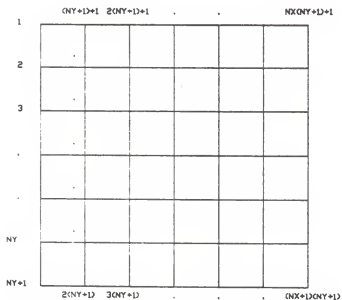
The next step is to choose the interpolation function. The nodal values of the final time, t_f , are t_{f_i} , t_{f_j} , and t_{f_k} and the nodal coordinates are (x_{1_i}, x_{2_i}) , (x_{1_j}, x_{2_j}) , and (x_{1_k}, x_{2_k}) . A linear



NX = Number of Rectangular Regions in the x axis

NY = Number of Rectangular Regions in the y axis

Figure 4.1a: Numbering of Nodes
Case i: $NY \geq NX$



NX = Number of rectangular regions in the x axis

NY = number of rectangular regions in the y axis

Figure 4.1b: Numbering of Nodes
Case ii: $NY < NX$

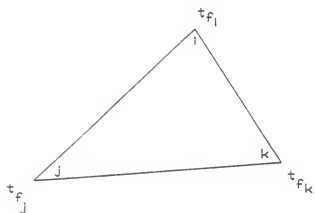


Figure 4.2a: Basic Element Shape

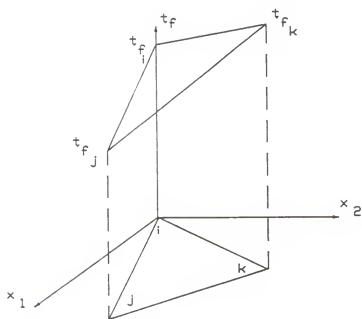


Figure 4.2b: Nodal Values of Final Time

interpolation formula [28] of the form

$$t_f = \alpha_1 + \alpha_2 x_1 + \alpha_3 x_2 \quad (4.1)$$

is chosen. The nodal conditions give

$$\begin{aligned} t_{f_i} &= \alpha_1 + \alpha_2 x_{1_i} + \alpha_3 x_{2_i}, \\ t_{f_j} &= \alpha_1 + \alpha_2 x_{1_j} + \alpha_3 x_{2_j}, \\ t_{f_k} &= \alpha_1 + \alpha_2 x_{1_k} + \alpha_3 x_{2_k}, \end{aligned} \quad (4.2)$$

which yields

$$\begin{aligned} \alpha_1 &= \frac{1}{2A} [(x_{1_j} x_{2_k} - x_{1_k} x_{2_j}) t_{f_i} + (x_{1_k} x_{2_i} - x_{1_i} x_{2_k}) t_{f_j} \\ &\quad + (x_{1_i} x_{2_j} - x_{1_j} x_{2_i}) t_{f_k}], \\ \alpha_2 &= \frac{1}{2A} [(x_{2_j} - x_{2_k}) t_{f_i} + (x_{2_k} - x_{2_i}) t_{f_j} + (x_{2_i} - x_{2_j}) t_{f_k}], \\ \alpha_3 &= \frac{1}{2A} [(x_{1_k} - x_{1_j}) t_{f_i} + (x_{1_i} - x_{1_k}) t_{f_j} + (x_{1_j} - x_{1_i}) t_{f_k}], \end{aligned}$$

where the determinant

$$\begin{vmatrix} 1 & x_{1_i} & x_{2_i} \\ 1 & x_{1_j} & x_{2_j} \\ 1 & x_{1_k} & x_{2_k} \end{vmatrix} = 2A, \quad (4.3)$$

and A is the area of the triangle.

Substituting for $\alpha_1, \alpha_2, \alpha_3$ in equation (4.1) and rearranging, we get

$$t_f = N_i t_{f_i} + N_j t_{f_j} + N_k t_{f_k}, \quad (4.4)$$

where

$$N_i = \frac{1}{2A} [a_i + b_i x_1 + c_i x_2], \quad (4.5)$$

$$N_j = \frac{1}{2A} [a_j + b_j x_1 + c_j x_2], \quad (4.6)$$

$$N_k = \frac{1}{2A} [a_k + b_k x_1 + c_k x_2], \quad (4.7)$$

and

$$a_i = x_{1j} x_{2k} - x_{1k} x_{2j}, \quad b_i = x_{2j} - x_{2k}, \quad c_i = x_{1k} - x_{1j},$$

$$a_j = x_{1k} x_{2i} - x_{1i} x_{2k}, \quad b_j = x_{2k} - x_{2i}, \quad c_j = x_{1i} - x_{1k},$$

$$a_k = x_{1i} x_{2j} - x_{1j} x_{2i}, \quad b_k = x_{2i} - x_{2j}, \quad c_k = x_{1j} - x_{1i}.$$

$N_i, N_j,$ and N_k are called the shape functions. It may be noted that each shape function is one at its node and zero at the others, the sum of the three shape functions at any point is one, and the shape function varies linearly between its node and the other two nodes but is zero on the side opposite its node. The selection of this type of an interpolation function gives us the final time, t_f , as a linear function of x_1 and x_2 and also gives constant first derivatives in all elements. The first derivatives of t_f are given by

$$\frac{\partial t_f}{\partial x_1} = \frac{1}{2A} [b_i t_{f_i} + b_j t_{f_j} + b_k t_{f_k}], \quad (4.8)$$

and

$$\frac{\partial t_f}{\partial x_2} = \frac{1}{2A} [c_i t_{f_i} + c_j t_{f_j} + c_k t_{f_k}]. \quad (4.9)$$

The element nodal coordinates can be transformed into a dimensionless local coordinate system. Such a transformation is useful only in that it provides us with a integral relation involving the shape functions. The details of the transformation are given in reference [28] and it gives an integral of the form

$$\int_A N_i^p N_j^q N_k^r dA = \frac{p! q! r!}{(p+q+r+2)!} 2A. \quad (4.10)$$

This integral is useful in the determination of the element properties.

ELEMENT PROPERTIES

The element properties have been derived with the least-squares technique. Using equations (3.24) and (3.29), we seek the solution, $t_f(x_1, x_2)$, that minimizes the integral

$$I_{fe} = \int_A \left\{ \frac{1}{2} \left[(1 + \mathbf{x}^T \nabla t_f)^2 + K (t_f - [2x_1 \quad x_2] \nabla t_f)^2 \right] \right. \\ \left. + \mu^+(u - 1) + \mu^-(-u - 1) \right\} dA, \quad (4.11)$$

where I_{fe} is the performance index measuring the error in the solution and K is a constant of unit magnitude that insures consistency of dimensional units. Other formulations were tried,

however, equation (4.11) produced the results which fit the requirements and by which the solution could be obtained.

The control can be determined to minimize the performance index, I_{fe} . Setting the partial derivative of I_{fe} with respect to the control, u , to zero, we get

$$\mu^+ = \lambda_2(1 - x_2 \lambda_1) - \lambda_2^2; u = 1, \quad (4.12)$$

$$\text{and} \quad \mu^- = -\lambda_2(1 - x_2 \lambda_1) - \lambda_2^2; u = -1. \quad (4.13)$$

Since the multipliers, μ^+ and μ^- , are always non-negative, we have

$$\lambda_2(1 - x_2 \lambda_1) \geq \lambda_2^2; u = 1, \quad (4.14)$$

$$\text{and} \quad -\lambda_2(1 - x_2 \lambda_1) \geq \lambda_2^2; u = -1. \quad (4.15)$$

Equations (4.14) and (4.15) can be combined to give

$$u \lambda_2 (1 - x_2 \lambda_1) \geq 0. \quad (4.16)$$

Solving equation (4.16) for u produces

$$u = \text{sgn} (\lambda_2 (1 - x_2 \lambda_1)). \quad (4.17)$$

Equation (4.17) is a discrete form of the control. The control, u , can change for different values of x_2 within the element and the multipliers, λ_1 and λ_2 , are constant inside each element but vary between elements. This gives us a distinct control at each point in the phase space region chosen. The control is chosen such that equation (4.11) is always reduced.

Now, expanding the first term of equation (4.11) gives

$$I_{fe}^1 = \frac{A}{2} + \int_A (x_2 u) dA \nabla t_f + \frac{1}{2} \nabla^T t_f \int_A \begin{bmatrix} x_2^2 & x_2 u \\ x_2 u & u^2 \end{bmatrix} dA \nabla t_f, \quad (4.18)$$

where the integrals are computed over the area of the element. Since the control may either be constant or vary inside each element, the integrals involving the control, u , need to be determined separately. If the control, u , is constant within the element then the integrals are easily evaluated. However, if the control changes sign within the element then the integration requires another procedure. Assuming that the distribution of the final time, t_f , is known, it is possible to determine whether there is a sign change within the element by evaluating the control equation (4.17) at each node. If there is a sign change, then at some value of x_2 the argument of equation (4.17) must vanish. This value of x_2 will be denoted as x_{2s} and is given by,

$$x_{2s} = \frac{1}{\lambda_1} = \frac{-1}{\partial t_f / \partial x_1} \quad (4.19)$$

which is a function of the nodal values of the final time. Figure (4.3) shows the two possible distribution of the control, u , inside an element in this case. It should be noted that the grid was completely made up of these two types of elements exclusively.

There are two integrals in equation (4.18) that depend upon the control, u . With reference to Figure (4.3)

$$\int_A u dA = u_j \int_{A_1} dA - u_j \int_{A-A_1} dA = u_j (2A_1 - A). \quad (4.20)$$

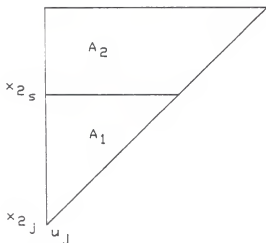


Figure 4.3a: Division of Triangle for Integration Purposes
Case i

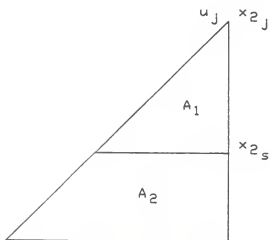


Figure 4.3b: Division of Triangle for Integration Purposes
Case ii

The area A_1 is a function of the location of the switching line given by equation (4.19) and is, thus, a function of the nodal values of t_f .

The area A_1 is given by

$$A_1 = A (x_{2_s} - x_{2_j}) / (x_{2_k} - x_{2_j}) \quad (4.21)$$

It should also be noted that the quantity u_j in equation (4.20) is the nodal control value that has a unique sign. The other integral in equation (4.18) that depends upon the control, u , is

$$\int_A x_2 u \, dA = u_j \int_{A_1} x_2 \, dA - u_j \int_{A-A_1} x_2 \, dA,$$

$$\text{or} \quad \int_A x_2 u \, dA = 2 u_j \int_{A_1} x_2 \, dA - u_j \int_A x_2 \, dA. \quad (4.22)$$

Now, each of the integrals in equation (4.22) is the first moment of area of the triangles about the x_1 axis, therefore, equation (4.22) becomes

$$\int_A x_2 u \, dA = u_j [2 A_1 (x_{2_j} + 2 x_{2_s}) - A (x_{2_i} + x_{2_j} + x_{2_k})] / 3 \quad (4.23)$$

where x_{2_j} is the x_2 coordinate of the node having the control u_j while x_{2_i} and x_{2_k} are the x_2 coordinates of the remaining nodes. The remaining integrals in equation (4.18) give

$$\int_A x_2 \, dA = A (x_{2_i} + x_{2_j} + x_{2_k}) / 3, \quad (4.24)$$

$$\text{and} \quad \int_A u^2 \, dA = \int_A \, dA = A. \quad (4.25)$$

Defining the vectors and the matrix, \underline{d} , as

$$\underline{N}^T = [N_i \ N_j \ N_k], \quad (4.26)$$

$$\underline{t}_f^T = [t_{f_i} \ t_{f_j} \ t_{f_k}], \quad (4.27)$$

$$\underline{x}_1^T = [x_{1_i} \ x_{1_j} \ x_{1_k}], \quad (4.28)$$

$$\underline{x}_2^T = [x_{2_i} \ x_{2_j} \ x_{2_k}], \quad (4.29)$$

and

$$\underline{d}^T = \begin{bmatrix} b_i & b_j & b_k \\ c_i & c_j & c_k \end{bmatrix} \quad (4.30)$$

we have

$$\int_A x_2^2 \, dA = \int_A \underline{N}^T \underline{x}_2 \underline{N}^T \underline{x}_2 \, dA,$$

or

$$\int_A x_2^2 \, dA = \int_A \underline{x}_2^T \underline{N} \underline{N}^T \underline{x}_2 \, dA,$$

or

$$\int_A x_2^2 \, dA = \underline{x}_2^T \int_A \underline{N} \underline{N}^T \, dA \underline{x}_2. \quad (4.31)$$

Now

$$\int_A \underline{N} \underline{N}^T \, dA = \int_A \begin{bmatrix} N_1^2 & N_1 N_2 & N_1 N_3 \\ N_1 N_2 & N_2^2 & N_2 N_3 \\ N_1 N_3 & N_2 N_3 & N_3^2 \end{bmatrix} \, dA.$$

Using equation (4.10) we have

$$\int_A \underline{N} \underline{N}^T \, dA = \frac{A}{12} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}. \quad (4.33)$$

Defining the matrix \underline{G} as

$$\underline{G} = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}, \quad (4.33)$$

then equation (4.31) gives

$$\int_A x_2^2 dA = \frac{A}{12} \underline{x}_2^T \underline{G} \underline{x}_2. \quad (4.34)$$

Now, the second part of equation (4.11) gives

$$\begin{aligned} I_{fe}^{ii} &= \int_A \frac{1}{2} t_f^2 dA + \frac{1}{2} \underline{v}^T t_f \int_A \begin{bmatrix} 2 x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} 2 x_1 & x_2 \end{bmatrix} dA \underline{v} t_f \\ &\quad - \int_A \begin{bmatrix} 2 x_1 t_f & x_2 t_f \end{bmatrix} dA \underline{v} t_f, \\ \text{or } I_{fe}^{ii} &= \int_A \frac{1}{2} t_f^2 dA + \frac{1}{2} \underline{v}^T t_f \int_A \begin{bmatrix} 4 x_1^2 & 2 x_1 x_2 \\ 2 x_1 x_2 & x_2^2 \end{bmatrix} dA \underline{v} t_f \\ &\quad - \int_A \begin{bmatrix} 2 x_1 t_f & x_2 t_f \end{bmatrix} dA \underline{v} t_f. \end{aligned} \quad (4.35)$$

The integral in the first term of equation (4.35) gives

$$\begin{aligned} \int_A \frac{1}{2} t_f^2 dA &= \frac{1}{2} \int_A \underline{N}^T \underline{t}_f \underline{N}^T \underline{t}_f dA, \\ \text{or } \int_A \frac{1}{2} t_f^2 dA &= \frac{1}{2} \underline{t}_f^T \int_A \underline{N} \underline{N}^T dA \underline{t}_f, \\ \text{or } \int_A \frac{1}{2} t_f^2 dA &= \frac{1}{2} \frac{A}{12} \underline{t}_f^T \underline{G} \underline{t}_f. \end{aligned} \quad (4.36)$$

The integrals in the second term of equation (4.35) give

$$\begin{aligned} \int_A 4 x_1^2 dA &= 4 \int_A \underline{N}^T \underline{x}_1 \underline{N}^T \underline{x}_1 dA, \\ \text{or } \int_A 4 x_1^2 dA &= 4 \underline{x}_1^T \int_A \underline{N} \underline{N}^T dA \underline{x}_1, \\ \text{or } \int_A 4 x_1^2 dA &= 4 \frac{A}{12} \underline{x}_1^T \underline{G} \underline{x}_1, \\ \text{and } \int_A 2 x_1 x_2 dA &= 2 \int_A \underline{N}^T \underline{x}_1 \underline{N}^T \underline{x}_2 dA, \end{aligned} \quad (4.37)$$

$$\begin{aligned}
\text{or} \quad & \int_A 2 x_1 x_2 dA = 2 \mathbf{x}_1^T \int_A \mathbf{N} \mathbf{N}^T dA \mathbf{x}_2, \\
\text{or} \quad & \int_A 2 x_1 x_2 dA = 2 \frac{A}{12} \mathbf{x}_1^T \mathbf{G} \mathbf{x}_2.
\end{aligned} \tag{4.38}$$

The integrals in the third term of equation (4.35) give

$$\begin{aligned}
& \int_A 2 x_1 t_f dA = 2 \int_A \mathbf{N}^T \underline{t}_f \mathbf{N}^T \mathbf{x}_1 dA, \\
\text{or} \quad & \int_A 2 x_1 t_f dA = 2 \underline{t}_f^T \int_A \mathbf{N} \mathbf{N}^T dA \mathbf{x}_1, \\
\text{or} \quad & \int_A 2 x_1 t_f dA = 2 \frac{A}{12} \underline{t}_f^T \mathbf{G} \mathbf{x}_1,
\end{aligned} \tag{4.39}$$

$$\begin{aligned}
& \text{and} \quad \int_A x_2 t_f dA = \int_A \mathbf{N}^T \underline{t}_f \mathbf{N}^T \mathbf{x}_2 dA, \\
\text{or} \quad & \int_A x_2 t_f dA = \underline{t}_f^T \int_A \mathbf{N} \mathbf{N}^T dA \mathbf{x}_2, \\
\text{or} \quad & \int_A x_2 t_f dA = \frac{A}{12} \underline{t}_f^T \mathbf{G} \mathbf{x}_2.
\end{aligned} \tag{4.40}$$

Now, from equations (4.8) and (4.9)

$$\nabla t_f = \frac{1}{2A} \underline{d}^T \underline{t}_f, \tag{4.41}$$

$$\text{and} \quad \underline{\nabla} t_f^T = \frac{1}{2A} \underline{t}_f^T \underline{d}. \tag{4.42}$$

If we define

$$\mathbf{P}_1 = \int_A [x_2 \quad u] dA, \tag{4.43}$$

$$\mathbf{P}_2 = \int_A \begin{bmatrix} x_2^2 & x_2 u \\ x_2 u & u^2 \end{bmatrix} dA, \tag{4.44}$$

$$\mathbf{P}_3 = \int_A \begin{bmatrix} 4 x_1^2 & 2 x_1 x_2 \\ 2 x_1 x_2 & x_2^2 \end{bmatrix} dA, \tag{4.45}$$

and
$$P_4 = \frac{A}{12} \begin{bmatrix} 2 G x_1 & G x_2 \end{bmatrix}. \quad (4.46)$$

Then, equations (4.18) and (4.19) can be combined to give

$$\begin{aligned} I_{fe} = & \frac{A}{2} + \frac{1}{2A} P_1 d^T \underline{t}_f + \frac{1}{2} \frac{1}{4A^2} \underline{t}_f^T d P_2 d^T \underline{t}_f \\ & + \frac{1}{2} \frac{A}{12} \underline{t}_f^T G \underline{t}_f + \frac{1}{2} \frac{1}{4A^2} \underline{t}_f^T d P_3 d^T \underline{t}_f \\ & - \frac{1}{2A} \underline{t}_f^T P_4 d^T \underline{t}_f. \end{aligned} \quad (4.47)$$

In order to minimize the error in the element we have to set the partial of I_{fe} with respect to the nodal values \underline{t}_f to zero. This can be done by making use of the vector differential properties.

$$\frac{\partial I_{fe}^T}{\partial \underline{t}_f} = 0,$$

which gives

$$\begin{aligned} & \frac{1}{2A} d P_1^T + \frac{1}{2A} \left[\frac{\partial P_1}{\partial \underline{t}_f} d^T \underline{t}_f \right]^T + \frac{1}{4A^2} d P_2 d^T \underline{t}_f \\ & + \frac{A}{12} G \underline{t}_f + \frac{1}{4A^2} d P_3 d^T \underline{t}_f + \frac{1}{2} \frac{1}{4A^2} \left[\underline{t}_f^T d \frac{\partial P_2}{\partial \underline{t}_f} d^T \underline{t}_f \right]^T \\ & - \frac{1}{2A} P_4 d^T \underline{t}_f - \frac{1}{2A} d P_4^T \underline{t}_f = 0. \end{aligned} \quad (4.48)$$

The second term in equation (4.48) gives

$$\frac{1}{2A} \left[\frac{\partial P_1}{\partial \underline{t}_f} d^T \underline{t}_f \right]^T = \frac{1}{2A} \left[\frac{\partial P_1}{\partial x_{2s}} d^T \underline{t}_f \frac{\partial x_{2s}}{\partial \underline{t}_f} \right]^T \quad (4.49)$$

Evaluating the partial differential terms in equation (4.49) separately, we have

$$\frac{\partial P_1}{\partial x_{2s}} = \left[0 \quad \frac{2 u_1 A}{(x_{2k} - x_{2j})} \right], \quad (4.50)$$

and
$$\frac{\partial x_{2s}}{\partial t_f} = \left(-1/\lambda_1^2 \right) \left(-\frac{1}{2A} \right) b^T,$$

or
$$\frac{\partial x_{2s}}{\partial t_f} = c_1 b^T, \quad (4.51)$$

where
$$c_1 = \frac{1}{2A\lambda_1^2}. \quad (4.52)$$

Equation (4.49) can be rewritten as

$$\frac{1}{2A} \left[\frac{\partial P_1}{\partial t_f} d^T t_f \right]^T = \frac{1}{2A} c_1 b P_5 d^T t_f \quad (4.53)$$

where

$$P_5^T = \frac{\partial P_1}{\partial x_{2s}}. \quad (4.54)$$

The sixth term in equation (4.48) gives

$$\begin{aligned} \frac{1}{2} \frac{1}{4A^2} \left[t_f^T d \frac{\partial P_2}{\partial t_f} d^T t_f \right]^T &= \frac{1}{2} \frac{1}{4A^2} \left[t_f^T d \frac{\partial P_2}{\partial x_{2s}} d^T t_f \frac{\partial x_{2s}}{\partial t_f} \right]^T, \\ &= \frac{1}{2} \frac{1}{4A^2} \left[t_f^T d P_6 d^T t_f c_1 b^T \right]^T, \\ &= \frac{1}{2} \frac{1}{4A^2} c_1 b t_f^T d P_6 d^T t_f, \quad (4.55) \end{aligned}$$

where

$$P_6 = \frac{\partial P_2}{\partial x_{2s}}$$

$$\text{or } P_6 = \begin{bmatrix} 0 & \frac{\partial P_{2(1,2)}}{\partial x_{2s}} \\ \frac{\partial P_{2(1,2)}}{\partial x_{2s}} & 0 \end{bmatrix}. \quad (4.56)$$

The partial differentials in equation (4.56) can be evaluated as

$$\begin{aligned} \frac{\partial P_{2(1,2)}}{\partial x_{2s}} &= \frac{u_1}{3} \left[2 A \frac{(x_{2j} + x_{2s})}{(x_{2k} - x_{2j})} + 4 A_1 \right], \\ \text{or } &= \frac{2 u_1}{3} \left[A \frac{(x_{2j} + x_{2s})}{(x_{2k} - x_{2j})} + 2 A_1 \right]. \end{aligned} \quad (4.57)$$

Equation (4.48) can now be rewritten as

$$[EM(\underline{t}_f)] \underline{t}_f + V(\underline{t}_f) = [0 \ 0 \ 0]^T = [RES]^T \quad (4.58)$$

where

$$\begin{aligned} [EM(\underline{t}_f)] &= \frac{1}{2 A} c_1 b \underline{P}_5^T \underline{d}^T + \frac{1}{4 A^2} \underline{d} \underline{P}_2 \underline{d}^T + \frac{A}{12} \underline{G} \\ &+ \frac{1}{4 A^2} \underline{d} \underline{P}_3 \underline{d}^T - \frac{1}{2 A} \underline{P}_4 \underline{d}^T - \frac{1}{2 A} \underline{d} \underline{P}_4^T \end{aligned} \quad (4.59)$$

$$\text{and } [V(\underline{t}_f)] = \frac{1}{2 A} \underline{d} \underline{P}_1^T + \frac{1}{2} \frac{1}{4 A^2} c_1 b \underline{t}_f^T \underline{d} \underline{P}_6 \underline{d}^T \underline{t}_f \quad (4.60)$$

This gives us three elemental equations for the three nodal unknowns. However, the nodal values are also associated with other

nodes and the problem has to be solved on a global basis. The element equations are computed in the FORTRAN routine LINTRI.

ASSEMBLY OF THE ELEMENTAL EQUATIONS

In order to solve for the unknown nodal values of the final time in the region under consideration. It is necessary to combine or assemble the element matrix equations to form the global system of equations. The basis for the assembly procedure is that all the elements are interconnected at the nodes with adjacent elements and the value of the unknowns are the same for each element sharing that node. The assembly procedure is carried out by means of the routine BUILD.

SOLUTION OF THE SYSTEM OF EQUATIONS

In order to solve the system of equations, the values of final time, t_f , were specified as boundary conditions on the outer boundaries. This was performed in routine BC. The solution was then obtained iteratively by using Newton's method. In order to solve the system of equations the Jacobian has to be calculated. The Jacobian, $[J]$, can be computed from equation (4.58) and is of the form

$$[J] = [EM(t_f)] + \frac{\partial}{\partial t_f} [EM(t_f)] t_f + \frac{\partial}{\partial t_f} [V(t_f)]. \quad (4.61)$$

The second term in the equation (4.61) gives

$$\frac{\partial}{\partial t_f} [EM(t_f)] t_f = \frac{1}{2 A} b P_5^T d^T t_f \frac{\partial c_1}{\partial t_f} + \frac{1}{4 A^2} d \frac{\partial P_2}{\partial t_f} d^T t_f$$

$$\begin{aligned}
& - \frac{1}{2A} \mathbf{b} \mathbf{P}_5^T \mathbf{d}^T \mathbf{t}_f \frac{c_1}{A\lambda_1} \mathbf{b}^T + \frac{1}{4A^2} \mathbf{d} \frac{\partial \mathbf{P}_2}{\partial \mathbf{x}_{2s}} \mathbf{d}^T \mathbf{t}_f c_1 \mathbf{b}^T \\
& - \frac{1}{2A} \mathbf{b} \mathbf{P}_5^T \mathbf{d}^T \mathbf{t}_f \frac{c_1}{A\lambda_1} \mathbf{b}^T + \frac{1}{4A^2} \mathbf{d} \mathbf{P}_6 \mathbf{d}^T \mathbf{t}_f c_1 \mathbf{b}^T. \quad (4.62)
\end{aligned}$$

The third term of equation (4.61) gives

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{t}_f} [V(\mathbf{t}_f)] &= \frac{1}{2A} \frac{\partial \mathbf{P}_1^T}{\partial \mathbf{t}_f} + \frac{1}{2} \frac{1}{4A^2} \left[2 \mathbf{b} \mathbf{t}_f^T \mathbf{d} \mathbf{P}_6 \mathbf{d}^T c_1 \right. \\
&\quad \left. + \mathbf{b} \mathbf{t}_f^T \mathbf{d} \frac{\partial \mathbf{P}_6}{\partial \mathbf{t}_f} \mathbf{d}^T \mathbf{t}_f c_1 + \mathbf{b} \mathbf{t}_f^T \mathbf{d} \mathbf{P}_6 \mathbf{d}^T \mathbf{t}_f \frac{\partial c_1}{\partial \mathbf{t}_f} \right] \\
\text{or} \quad &= \frac{1}{2A} \mathbf{d} \mathbf{P}_5 c_1 \mathbf{b}^T + \frac{1}{2} \frac{1}{4A^2} \left[2 \mathbf{b} \mathbf{t}_f^T \mathbf{d} \mathbf{P}_6 \mathbf{d}^T c_1 \right. \\
&\quad \left. + \mathbf{b} \mathbf{t}_f^T \mathbf{d} \frac{\partial \mathbf{P}_6}{\partial \mathbf{x}_{2s}} \mathbf{d}^T \mathbf{t}_f c_1 \frac{\partial \mathbf{x}_{2s}}{\partial \mathbf{t}_f} \right. \\
&\quad \left. + \mathbf{b} \mathbf{t}_f^T \mathbf{d} \mathbf{P}_6 \mathbf{d}^T \mathbf{t}_f \frac{c_1}{A\lambda_1} \mathbf{b}^T \right] \quad (4.63)
\end{aligned}$$

The third term in equation (4.63) can be evaluated as

$$\begin{aligned}
& \frac{1}{2} \frac{1}{4A^2} \mathbf{b} \mathbf{t}_f^T \mathbf{d} \frac{\partial \mathbf{P}_6}{\partial \mathbf{x}_{2s}} \mathbf{d}^T \mathbf{t}_f c_1 \frac{\partial \mathbf{x}_{2s}}{\partial \mathbf{t}_f} \\
&= \frac{1}{2} \frac{1}{4A^2} \mathbf{b} \mathbf{t}_f^T \mathbf{d} \mathbf{P}_7 \mathbf{d}^T \mathbf{t}_f c_1^2 \mathbf{b}^T \quad (4.64)
\end{aligned}$$

where

$$\mathbf{P}_7 = \frac{\partial \mathbf{P}_6}{\partial \mathbf{x}_{2s}},$$

or

$$P_7 = \begin{bmatrix} 0 & \frac{\partial^2 P_{2(1,2)}}{\partial x_{2s}^2} \\ \frac{\partial^2 P_{2(1,2)}}{\partial x_{2s}^2} & 0 \end{bmatrix}. \quad (4.65)$$

The non-zero terms in equation (4.65) can be evaluated as

$$\frac{\partial^2 P_{222}}{\partial x_{2s}^2} = \frac{8}{3} u_j \frac{A}{(x_{2k} - x_{2j})}. \quad (4.66)$$

Equation (4.63) can now be written as

$$\begin{aligned} \frac{\partial}{\partial \underline{x}_f} [V(\underline{x}_f)] &= \frac{1}{2A} \underline{d} P_5 c_1 \underline{b}^T + \frac{1}{2} \frac{1}{4A^2} \left[2 \underline{b} \underline{x}_f^T \underline{d} P_6 \underline{d}^T c_1 \right. \\ &\quad + \underline{b} \underline{x}_f^T \underline{d} P_7 \underline{d}^T \underline{x}_f c_1^2 \underline{b}^T \\ &\quad \left. + \underline{b} \underline{x}_f^T \underline{d} P_6 \underline{d}^T \underline{x}_f \frac{c_1}{A \lambda_1} \underline{b}^T \right]. \end{aligned} \quad (4.67)$$

The Jacobian is now given by

$$\begin{aligned} [J] &= \frac{1}{2A} c_1 \underline{b} P_5^T \underline{d}^T + \frac{1}{4A^2} \underline{d} P_2 \underline{d}^T + \frac{A}{12} \underline{G} + \frac{1}{4A^2} \underline{d} P_3 \underline{d}^T - \frac{1}{2A} P_4 \underline{d}^T \\ &\quad - \frac{1}{2A} \underline{d} P_4^T + \frac{1}{2A} \underline{b} P_5^T \underline{d}^T \underline{x}_f \frac{c_1}{A \lambda_1} \underline{b}^T + \frac{1}{4A^2} \underline{d} P_6 \underline{d}^T \underline{x}_f c_1 \underline{b}^T \\ &\quad + \frac{1}{2A} \underline{d} P_5 c_1 \underline{b}^T + \frac{1}{2} \frac{1}{4A^2} \left[2 \underline{b} \underline{x}_f^T \underline{d} P_6 \underline{d}^T c_1 \right. \\ &\quad \left. + \underline{b} \underline{x}_f^T \underline{d} P_7 \underline{d}^T \underline{x}_f c_1^2 \underline{b}^T + \underline{b} \underline{x}_f^T \underline{d} P_6 \underline{d}^T \underline{x}_f \frac{c_1}{A \lambda_1} \underline{b}^T \right]. \end{aligned} \quad (4.68)$$

It should be noted that the Jacobian is symmetric. The symmetry is obvious in terms two, three, four, seven, nine, eleven, and twelve of equation (4.68). Terms five and six, and terms one and nine are the sum of a matrix and its transpose, which is symmetric. Since the Jacobian was obtained from the element matrix it has to be calculated along with the element properties and assembled.

In order to solve the problem iteratively the residual equation is given by

$$[J] \delta \underline{u}_f = [RES], \quad (4.69)$$

where

$$\underline{u}_f^{i+1} = \underline{u}_f^i - \delta \underline{u}_f. \quad (4.70)$$

The superscripts in equation (4.70) denote the iteration numbers. Equation (4.69) can now be rewritten as

$$[J] \underline{u}_f^{i+1} = [J] \underline{u}_f^i - [RES]. \quad (4.71)$$

The assembled form of equation (4.71) can now be solved iteratively. The system of equations in each iteration were solved by a FORTRAN routine SOLVE. The results and conclusions are presented and analysed in Chapter 5. Recommendations for further study are also presented.

CHAPTER 5

RESULTS AND RECOMMENDATIONS

In this work a continuum approach to the minimum time problem has been investigated to approximately determine the isochrone distribution in the phase plane of a double integrator problem. The continuum relations were derived and an approximate solution was obtained using the finite element method. The calculations were done by FORTRAN 77 programs implemented on an IBM 370 VM/CMS computer and an IBM PC microcomputer.

A four by four square region was selected to test this technique. A number of different grids were used in the calculation ranging from very coarse to very fine. Two such cases are presented here. Figure (5.1) shows the isochrone distribution for a 21 by 21 grid while figure (5.2) shows the same distribution for a 41 by 41 grid.

The abrupt change in the direction of the isochrones in these two figures allows one to approximately trace the switching curve. The development of the switching line, x_{2s} , causes the switching curve to have a tendency to prefer a horizontal distribution. The true switching curve passes through the lower righthand and upper lefthand corners of the grid. The 41 by 41 grid shows that as the element

sizes is reduced the switching curve approaches the expected orientation.

The solutions are also presented in tabular form for four cases along with the actual solutions. A comparison of the actual solution to the approximate solution shows that the solutions with fine grids give very good results.

Better isochrone distributions are possible if the grid is made to conform to the switching curve, however, this defeats the purpose of the investigation.

The boundary conditions on the outer grid surface were varied to study the performance. The only set of boundary conditions which provided a reliable solution was to specify the true final time at each boundary node. That this is a necessary condition for the solution was verified by deriving the differential equations describing the variations of the final time with respect to the state variables, x_1 and x_2 , from equation (3.31). These two differential equations are separable and each requires two boundary conditions before a unique solution is possible.

An interesting condition occurs when all the boundary conditions are removed from the problem except the boundary condition at the origin. The solution which results is

$$t_f(x_1, x_2) = -ux_2 \quad (5.1)$$

where u is negative for positive x_2 and positive otherwise. Equation (5.1) produces the control necessary to reduce the velocity to zero in

minimum time regardless of the initial and final value of x_1 . An examination of equation (5.1) and equation (4.11) shows that this simple functional expression causes equation (4.11) to vanish entirely, thus rendering equation (4.11) an absolute minimum.

Conclusions

Conclusions reached in this investigation are:

1. The present formulation is necessary but not sufficient to uniquely specify the minimum time control; other information is necessary in order to obtain a solution.
2. Coarser grids require very little time to produce an approximate solution, a point which has provided motivation to continue the work in this area.
3. Linear and nonlinear problems present the same level of complexity when approached from this point of view.
4. The method can be used to provide closed loop minimum time controls by determining the position of the system at each instant of time and calculating the control at that point.

Problem areas requiring additional investigation include:

1. The determination of additional continuum relations which would provide a unique solution without the need to specify the solution on the boundary.
2. Developing a means to treat systems of order greater than two.
3. Determining an alternative analysis by which the continuum condition is determined directly from the minimum time functional.

4. Testing the control produced by this or similar methods to see if there is not excessive chattering.

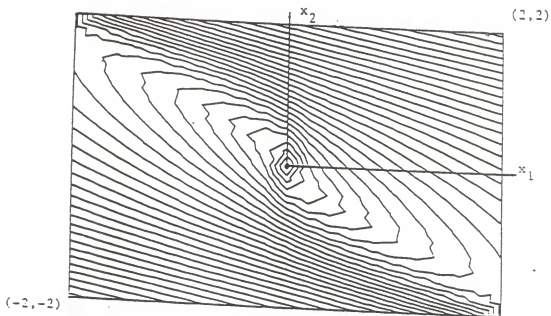


Figure 5.1: Isochrone Distribution from the 21x21 Finite Element Grid

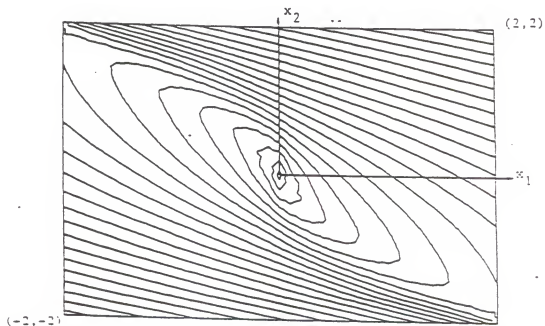


Figure 5.2: Isochrone Distribution from the 41x41 Finite Element Grid

2.0	4.0	4.8	5.5	6.0
2.2	1.4	2.4	3.4	4.2
2.8	2.0	0.0	2.0	2.8
4.2	3.4	2.4	1.4	2.2
6.0	5.5	4.8	4.0	2.0

Table 5.1a: Approximate Solution from the Continuum Approach
For the 5x5 Grid

2.0	4.0	4.8	5.5	6.0
2.2	1.9	2.2	3.4	4.2
2.8	1.9	0.0	1.9	2.8
4.2	3.4	2.2	1.9	2.2
6.0	5.5	4.8	4.0	2.0

Table 5.1b: Actual Solution from the Continuum Approach
For the 5x5 Grid

2.0	3.3	3.9	4.3	4.7	5.0	5.3	5.5	5.8	6.0
2.0	2.2	2.5	2.9	3.4	3.9	4.3	4.6	4.9	5.1
2.1	1.9	1.8	1.9	2.3	2.9	3.4	3.7	4.1	4.3
2.3	2.0	1.6	1.3	1.3	1.9	2.5	3.0	3.3	3.6
2.6	2.3	1.9	1.4	0.8	1.2	1.9	2.3	2.7	3.1
3.1	2.7	2.3	1.9	1.2	0.8	1.4	1.9	2.3	2.6
3.6	3.3	3.0	2.5	1.9	1.3	1.3	1.6	2.0	2.3
4.3	4.1	3.7	3.4	2.9	2.3	1.9	1.8	1.9	2.1
5.1	4.9	4.6	4.3	3.9	3.4	2.9	2.5	2.2	2.0
6.0	5.8	5.5	5.3	5.0	4.7	4.3	3.9	3.3	2.0

Table 5.2a: Approximate Solution from the Continuum Approach
For the 10x10 Grid

2.0	3.3	3.9	4.3	4.7	5.0	5.3	5.5	5.8	6.0
2.0	1.8	2.2	3.0	3.5	3.9	4.3	4.6	4.9	5.1
2.1	1.8	1.5	1.2	2.4	2.9	3.4	3.7	4.1	4.3
2.3	2.0	1.6	1.2	0.7	2.0	2.6	3.0	3.3	3.6
2.6	2.3	1.9	1.4	0.8	1.2	1.9	2.4	2.7	3.1
3.1	2.7	2.4	1.9	1.2	0.8	1.4	1.9	2.3	2.6
3.6	3.3	3.0	2.6	2.0	0.7	1.2	1.6	2.0	2.3
4.3	4.1	3.7	3.4	2.9	2.4	1.2	1.5	1.8	2.1
5.1	4.9	4.6	4.3	3.9	3.5	3.0	2.2	1.8	2.0
6.0	5.8	5.5	5.3	5.0	4.7	4.3	3.9	3.3	2.0

Table 5.2b: Actual Solution from the Continuum Approach
For the 10x10 Grid

2.0	3.1	3.5	3.9	4.1	4.4	4.6	4.8	5.0	5.2	5.4	5.5	5.7	5.9	6.0
2.0	2.2	2.5	2.8	3.2	3.5	3.8	4.1	4.4	4.6	4.8	4.9	5.1	5.3	5.4
2.0	1.9	1.9	2.1	2.3	2.7	3.0	3.4	3.7	3.9	4.2	4.4	4.6	4.7	4.9
2.1	1.9	1.7	1.7	1.7	1.9	2.2	2.7	3.1	3.3	3.6	3.8	4.0	4.2	4.4
2.2	2.0	1.8	1.6	1.4	1.4	1.5	1.9	2.4	2.8	3.1	3.3	3.5	3.7	3.9
2.4	2.2	1.9	1.7	1.4	1.1	1.0	1.2	1.9	2.3	2.6	2.9	3.1	3.3	3.5
2.6	2.4	2.1	1.9	1.6	1.2	0.8	0.6	1.4	1.8	2.2	2.5	2.7	2.9	3.1
2.8	2.6	2.4	2.1	1.8	1.5	1.0	0.0	1.0	1.5	1.8	2.1	2.4	2.6	2.8
3.1	2.9	2.7	2.5	2.2	1.8	1.4	0.6	0.8	1.2	1.6	1.9	2.1	2.4	2.6
3.5	3.3	3.1	2.9	2.6	2.3	1.9	1.2	1.0	1.1	1.4	1.7	1.9	2.2	2.4
3.9	3.7	3.5	3.3	3.1	2.8	2.4	1.9	1.5	1.4	1.4	1.6	1.8	2.0	2.2
4.4	4.2	4.0	3.8	3.6	3.3	3.1	2.7	2.2	1.9	1.7	1.7	1.7	1.9	2.1
4.9	4.7	4.6	4.4	4.2	3.9	3.7	3.4	3.0	2.7	2.3	2.1	1.9	1.9	2.0
5.4	5.3	5.1	4.9	4.8	4.6	4.4	4.1	3.8	3.5	3.2	2.8	2.5	2.2	2.0
6.0	5.9	5.7	5.5	5.4	5.2	5.0	4.8	4.6	4.4	4.1	3.9	3.5	3.1	2.0

Table 5.3a: Approximate Solution from the Continuum Approach
For the 15x15 Grid

2.0	3.1	3.5	3.9	4.1	4.4	4.6	4.8	5.0	5.2	5.4	5.5	5.7	5.9	6.0
2.0	1.9	2.1	2.9	3.3	3.6	3.9	4.1	4.4	4.6	4.8	4.9	5.1	5.3	5.4
2.0	1.9	1.7	1.5	2.2	2.8	3.1	3.4	3.7	4.0	4.2	4.4	4.6	4.7	4.9
2.1	1.9	1.7	1.5	1.3	1.7	2.4	2.8	3.1	3.4	3.6	3.8	4.0	4.2	4.4
2.2	2.0	1.8	1.6	1.4	1.1	1.4	2.1	2.5	2.8	3.1	3.3	3.5	3.7	3.9
2.4	2.2	2.0	1.7	1.4	1.1	0.8	1.4	1.9	2.3	2.6	2.9	3.1	3.3	3.5
2.6	2.4	2.1	1.9	1.6	1.3	0.9	0.7	1.4	1.9	2.2	2.5	2.7	2.9	3.1
2.8	2.6	2.4	2.1	1.9	1.5	1.1	0.0	1.1	1.5	1.9	2.1	2.4	2.6	2.8
3.1	2.9	2.7	2.5	2.2	1.9	1.4	0.7	0.9	1.3	1.6	1.9	2.1	2.4	2.6
3.5	3.3	3.1	2.9	2.6	2.3	1.9	1.4	0.8	1.1	1.4	1.7	2.0	2.2	2.4
3.9	3.7	3.5	3.3	3.1	2.8	2.5	2.1	1.4	1.1	1.4	1.6	1.8	2.0	2.2
4.4	4.2	4.0	3.8	3.6	3.4	3.1	2.8	2.4	1.7	1.3	1.5	1.7	1.9	2.1
4.9	4.7	4.6	4.4	4.2	4.0	3.7	3.4	3.1	2.8	2.2	1.5	1.7	1.9	2.0
5.4	5.3	5.1	4.9	4.8	4.6	4.4	4.1	3.9	3.6	3.3	2.9	2.1	1.9	2.0
6.0	5.9	5.7	5.5	5.4	5.2	5.0	4.8	4.6	4.4	4.1	3.9	3.5	3.1	2.0

Table 5.3b: Actual Solution from the Continuum Approach
For the 15x15 Grid

2.0	2.9	3.3	3.5	3.8	4.0	4.2	4.4	4.5	4.7	4.8	5.0	5.1	5.2	5.3	5.5	5.6	5.7	5.8	5.9	6.0
2.0	2.2	2.4	2.7	3.1	3.3	3.6	3.8	4.0	4.2	4.3	4.5	4.6	4.8	4.9	5.0	5.2	5.3	5.4	5.5	5.6
2.0	2.0	2.0	2.1	2.3	2.6	2.9	3.2	3.4	3.6	3.9	4.0	4.2	4.3	4.5	4.6	4.7	4.9	5.0	5.1	5.2
2.1	1.9	1.8	1.8	1.9	2.0	2.3	2.5	2.8	3.1	3.4	3.6	3.7	3.9	4.1	4.2	4.4	4.5	4.6	4.7	4.9
2.1	2.0	1.8	1.7	1.6	1.7	1.8	1.9	2.2	2.5	2.8	3.1	3.3	3.5	3.7	3.8	4.0	4.1	4.2	4.4	4.5
2.2	2.0	1.9	1.7	1.6	1.5	1.4	1.5	1.7	2.0	2.3	2.6	2.9	3.1	3.3	3.4	3.6	3.8	3.9	4.0	4.2
2.2	2.1	2.0	1.8	1.6	1.4	1.3	1.2	1.3	1.5	1.8	2.2	2.5	2.7	2.9	3.1	3.3	3.4	3.6	3.7	3.8
2.4	2.2	2.1	1.9	1.7	1.5	1.3	1.1	1.0	1.0	1.3	1.8	2.1	2.4	2.6	2.8	2.9	3.1	3.3	3.4	3.6
2.5	2.3	2.2	2.0	1.8	1.7	1.4	1.2	0.9	0.7	0.8	1.4	1.8	2.0	2.3	2.5	2.7	2.8	3.0	3.1	3.3
2.6	2.5	2.3	2.2	2.0	1.8	1.6	1.4	1.1	0.8	0.4	1.1	1.5	1.8	2.0	2.2	2.4	2.6	2.7	2.9	3.0
2.8	2.7	2.5	2.4	2.2	2.0	1.8	1.5	1.2	0.9	0.0	0.9	1.2	1.5	1.8	2.0	2.2	2.4	2.5	2.7	2.8
3.0	2.9	2.7	2.6	2.4	2.2	2.0	1.8	1.5	1.1	0.4	0.8	1.1	1.4	1.6	1.8	2.0	2.2	2.3	2.5	2.6
3.3	3.1	3.0	2.8	2.7	2.5	2.3	2.0	1.8	1.4	0.6	0.7	0.9	1.2	1.4	1.7	1.8	2.0	2.2	2.3	2.5
3.6	3.4	3.3	3.1	2.9	2.8	2.6	2.4	2.1	1.8	1.3	1.0	1.0	1.1	1.3	1.5	1.7	1.9	2.1	2.2	2.4
3.8	3.7	3.6	3.4	3.3	3.1	2.9	2.7	2.5	2.2	1.8	1.5	1.3	1.2	1.3	1.4	1.6	1.8	2.0	2.1	2.2
4.2	4.0	3.9	3.8	3.6	3.4	3.3	3.1	2.9	2.6	2.3	2.0	1.7	1.5	1.4	1.5	1.6	1.7	1.9	2.0	2.2
4.5	4.4	4.2	4.1	4.0	3.8	3.7	3.5	3.3	3.1	2.8	2.5	2.2	1.9	1.8	1.7	1.6	1.7	1.8	2.0	2.1
4.9	4.7	4.6	4.5	4.4	4.2	4.1	3.9	3.7	3.6	3.4	3.1	2.8	2.5	2.3	2.0	1.9	1.8	1.8	1.9	2.1
5.2	5.1	5.0	4.9	4.7	4.6	4.5	4.3	4.2	4.0	3.9	3.6	3.4	3.2	2.9	2.6	2.3	2.1	2.0	2.0	2.0
5.6	5.5	5.4	5.3	5.2	5.0	4.9	4.8	4.6	4.5	4.3	4.2	4.0	3.8	3.6	3.3	3.0	2.7	2.4	2.2	2.0
6.0	5.9	5.8	5.7	5.6	5.5	5.3	5.2	5.1	5.0	4.8	4.7	4.5	4.4	4.2	4.0	3.8	3.5	3.3	2.9	2.0

Table 5.4a: Approximate Solution from the Continuum Approach
For the 21x21 Grid

-0	2.9	3.1	3.5	3.8	4.0	4.2	4.4	4.5	4.7	4.8	5.0	5.1	5.2	5.3	5.5	5.6	5.7	5.8	5.9	6.0
-0	1.9	2.1	2.7	3.1	3.4	3.6	3.8	4.0	4.2	4.3	4.5	4.6	4.8	4.9	5.0	5.2	5.3	5.4	5.5	5.6
2.0	1.9	1.8	1.7	2.2	2.7	3.0	3.2	3.5	3.7	3.9	4.0	4.2	4.3	4.5	4.6	4.7	4.9	5.0	5.1	5.2
2.1	1.9	1.8	1.7	1.6	1.4	2.2	2.6	2.9	3.2	3.4	3.6	3.7	3.9	4.1	4.2	4.4	4.5	4.6	4.7	4.9
2.1	2.0	1.8	1.7	1.6	1.4	1.3	1.9	2.3	2.6	2.9	3.1	3.3	3.5	3.7	3.8	4.0	4.1	4.2	4.4	4.5
2.2	2.0	1.9	1.8	1.6	1.4	1.3	1.1	1.6	2.1	2.4	2.7	2.9	3.1	3.3	3.4	3.6	3.8	3.9	4.0	4.2
2.2	2.1	2.0	1.8	1.7	1.5	1.3	1.1	0.9	1.5	1.9	2.2	2.5	2.7	2.9	3.1	3.3	3.4	3.6	3.7	3.8
2.4	2.2	2.1	1.9	1.7	1.6	1.4	1.2	0.9	0.6	1.4	1.8	2.1	2.4	2.6	2.8	2.9	3.1	3.3	3.4	3.6
2.5	2.3	2.2	2.0	1.9	1.7	1.5	1.2	1.0	0.7	1.0	1.5	1.8	2.0	2.3	2.5	2.7	2.8	3.0	3.1	3.3
2.6	2.5	2.3	2.2	2.0	1.8	1.6	1.4	1.1	0.7	0.5	1.1	1.5	1.8	2.0	2.2	2.4	2.6	2.7	2.9	3.0
2.8	2.7	2.5	2.4	2.2	2.0	1.8	1.5	1.3	0.9	0.0	0.9	1.3	1.5	1.8	2.0	2.2	2.4	2.5	2.7	2.8
3.0	2.9	2.7	2.6	2.4	2.2	2.0	1.8	1.5	1.1	0.5	0.7	1.1	1.4	1.6	1.8	2.0	2.2	2.3	2.5	2.6
3.3	3.1	3.0	2.8	2.7	2.5	2.3	2.0	1.8	1.5	1.0	0.7	1.0	1.2	1.5	1.7	1.9	2.0	2.2	2.3	2.5
3.6	3.4	3.3	3.1	2.9	2.8	2.6	2.4	2.1	1.8	1.4	0.6	0.9	1.2	1.4	1.6	1.7	1.9	2.1	2.2	2.4
3.8	3.7	3.6	3.4	3.3	3.1	2.9	2.7	2.5	2.2	1.9	1.5	0.9	1.1	1.3	1.5	1.7	1.8	2.0	2.1	2.2
4.2	4.0	3.9	3.8	3.6	3.4	3.3	3.1	2.9	2.7	2.4	2.1	1.6	1.1	1.3	1.4	1.6	1.8	1.9	2.0	2.2
4.5	4.4	4.2	4.1	4.0	3.8	3.7	3.5	3.3	3.1	2.9	2.6	2.3	1.9	1.3	1.4	1.6	1.7	1.8	2.0	2.1
4.9	4.7	4.6	4.5	4.4	4.2	4.1	3.9	3.7	3.6	3.4	3.2	2.9	2.6	2.2	1.4	1.6	1.7	1.8	1.9	2.1
5.2	5.1	5.0	4.9	4.7	4.6	4.5	4.3	4.2	4.0	3.9	3.7	3.5	3.2	3.0	2.7	2.2	1.7	1.8	1.9	2.0
5.6	5.5	5.4	5.3	5.2	5.0	4.9	4.8	4.6	4.5	4.3	4.2	4.0	3.8	3.6	3.4	3.1	2.7	2.1	1.9	2.0
6.0	5.9	5.8	5.7	5.6	5.5	5.3	5.2	5.1	5.0	4.8	4.7	4.5	4.4	4.2	4.0	3.8	3.5	3.3	2.9	2.0

Table 5.4b: Actual Solution from the Continuum Approach
For the 21x21 Grid

REFERENCES

- 1) Taylor, F.W., "Shop Management," New York, 1919.
- 2) Luh, J.Y.S., and Lin, C.S., "Optimal path planning for Mechanical Manipulators," Transactions of the ASME, Journal of Dynamic Systems and Controls - June 1981.
- 3) Lin, C.S., Chang, P.R., and Luh, J.Y.S., "Formulation and optimization Of Cubic Polynomial Joint Trajectories for Industrial Robots," IEEE Transactions on Automatic Controls Vol AC 28, No. 12 December 1983.
- 4) Bobrow, J.E., Dubowsky, S., and Gibson, J.S., "Time Optimal control of robotic manipulators along specified paths," International Journal of Robotics Research, Vol. 4 no. 3 Fall 1985.
- 5) Shin, K.G., and McKay, N., "Minimum time control of robotic manipulators with geometric path constraints," IEEE Transactions on Automatic Control, AC 30 No. 6, June 1985.
- 6) Athanassiades, M., and Smith, O.J.M., "Theory and Design of Higher Order Bang Bang Control Systems," IRE Transactions on Automatic Controls, May 1961.
- 7) Desoer, C.A., and Wing, J., "A Minimal Time Discrete System," IRE Transactions on Automatic Control, May 1961.
- 8) Ryan, E.P., "Minimum Time isochronal surfaces for certain third order systems" International Journal of Control, September 1977
- 9) Fuller, A.T., "Time optimal control on the state axes," International Journal of Control Vol 32 No 5 1980 pp 771-775.
- 10) Sebhakhy, O.A., and Abdel-Moneim, T.M., "State regulation in linear discrete time systems in minimum time," IEEE Transactions on Automatic Control Vol AC 24 No 1 February 1979.
- 11) Rao, P.V., and Janakiraman, P.A., "Time optimal design of discrete Data systems in the frequency domain," Automatica Vol 10 No 5 September 1974 pp 539-543
- 12) Knudsen, H.K., "An Iterative procedure for computing time optimal controls," IEEE Transactions on Automatic Controls, January 1964

pp 23-30.

- 13) Nagata, A., Kodama, S., and Kumagai, S., "Time optimal Discrete control systems with bounded state variables," IEEE Transactions on Automatic Controls, April 1965, pp 155-164.
- 14) Lastman, G.J., "A Shooting Method for solving two point boundary value problems," International Journal of control, vol 27 No. 4 April 1978 pp 513-524.
- 15) Larson, V.H., "Minimum time control by time interval optimization," International Journal of Control vol. 7 no. 4 1968 pp 381-394.
- 16) Yastreboff, M., "Synthesis of Time Optimal Control by Time Interval Adjustment," IEEE Transactions on Automatic Control Dec. 1969 pp 707-710.
- 17) Kahn, M.E., and Roth, B., "The near minimum time control of open loop articulated kinematic chains," Transactions of the ASME, September 1971.
- 18) Wen, J.T., and Desrochers, A., "Sub-time optimal control strategies for robotic manipulators," IEEE Conference on robotics and automation 1986 pp 402-205.
- 19) Shetty, A., "A Solution Technique for the Minimum Time Control Problem of an R-Theta Manipulator," Master's Thesis, Department of Mechanical Engineering, Kansas State University, 1987.
- 20) Davison, E.J., and Monro, D., M., "A computational technique for finding the time optimal control of nonlinear time varying systems," Joint Automatic Control Conference, Boulder Colorado, 1969 pp 270-280.
- 21) Wen, J.T., and Desrochers, A., "An Algorithm for Obtaining Bang-Bang Control Laws," Journal of Dynamic Systems, Measurement and Control, June 1987 Vol. 109, pp 171-175.
- 22) Rosenbrock, H.H., "An Automatic way of Finding the Greatest or Least Value of a function," Computer Journal, Vol. 3, No. 3, 1963.
- 23) Goor, R.M., "A new approach to minimum time robot control," ASME winter annual meeting, Miami, Florida, November 1985.
- 24) Luh, J.Y.S., and Shafraan, J.S., "An approximate minimal time closed loop controller for processes with bounded control amplitudes and rates," Joint Automatic Control Conference, Boulder, Colorado 1969.
- 25) Kirk, D.E., "Optimal Control Theory," Prentice Hall, 1970.

- 26) Bryson, A.E., and Ho, Y.C., "Applied Optimal Control," John Wiley and Sons, 1975.
- 27) White, W.N. Jr., and Rajendran, K., "A Continuum Approach to Minimum Time," Paper presented for publication.
- 28) Segerlind, L.J., "Applied Finite Element Analysis," John Wiley and Sons, 1984.

APPENDIX 1

CONSTRAINED OPTIMIZATION OF A FUNCTIONAL

Consider the functional, \bar{J} , given by equation (2.18)

$$\bar{J}(\underline{x}(t), \dot{\underline{x}}(t), \underline{u}(t), \underline{\lambda}(t), t) = \int_{t_0}^{t_f} \bar{g}(\underline{x}(t), \dot{\underline{x}}(t), \underline{u}(t), \underline{\lambda}(t), t) dt \quad (A1.1)$$

where the initial time, t_0 , and initial states, $\underline{x}(t_0)$, are fixed and the final time, t_f , is free to vary. The functional, \bar{J} , can be minimized by setting the variation $\delta \bar{J}$ to zero. The variation, $\delta \bar{J}$, is given by

$$\delta \bar{J} = \int_{t_0}^{t_f} \left[\frac{\partial \bar{g}^T}{\partial \dot{\underline{x}}} \delta \dot{\underline{x}} + \frac{\partial \bar{g}^T}{\partial \underline{x}} \delta \underline{x} + \frac{\partial \bar{g}^T}{\partial \underline{u}} \delta \underline{u} + \frac{\partial \bar{g}^T}{\partial \underline{\lambda}} \delta \underline{\lambda} \right] dt + \int_{t_f}^{t_f + dt_f} \bar{g} dt. \quad (A1.2)$$

Integrating the first term of equation (A1.2) by parts, we get

$$\begin{aligned} \delta \bar{J} &= \frac{\partial \bar{g}^T}{\partial \dot{\underline{x}}}(\underline{x}(t), \dot{\underline{x}}(t), \underline{u}(t), \underline{\lambda}(t), t) \bigg|_{t_0}^{t_f} \delta \underline{x}(t_f) \\ &\quad + \bar{g}(\underline{x}(t), \dot{\underline{x}}(t), \underline{u}(t), \underline{\lambda}(t), t) \bigg|_{t=t_f} dt_f \\ &\quad + \int_{t_0}^{t_f} \left(\frac{\partial \bar{g}^T}{\partial \underline{x}}(\underline{x}(t), \dot{\underline{x}}(t), \underline{u}(t), \underline{\lambda}(t), t) \right. \end{aligned}$$

$$\begin{aligned}
& - \frac{d}{dt} \left(\frac{\partial \bar{g}}{\partial \dot{\mathbf{x}}}^T (\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{u}(t), \lambda(t), t) \right) \delta \mathbf{x}(t) \\
& + \frac{\partial \bar{g}}{\partial \mathbf{u}}^T (\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{u}(t), \lambda(t), t) \delta \mathbf{u}(t) \\
& + \frac{\partial \bar{g}}{\partial \lambda}^T (\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{u}(t), \lambda(t), t) \delta \lambda(t) \} dt. \quad (A1.3)
\end{aligned}$$

Since the variation $\delta \mathbf{x}$ is zero at the initial time equation (A1.3) can be rewritten as

$$\begin{aligned}
\delta J = & \frac{\partial \bar{g}}{\partial \dot{\mathbf{x}}}^T (\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{u}(t), \lambda(t), t) \Big|_{t_f} \delta \mathbf{x}(t_f) \\
& + \bar{g} (\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{u}(t), \lambda(t), t) \Big|_{t=t_f} dt_f \\
& + \int_{t_0}^{t_f} \left\{ \left[\frac{\partial \bar{g}}{\partial \dot{\mathbf{x}}}^T (\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{u}(t), \lambda(t), t) \right. \right. \\
& - \frac{d}{dt} \left(\frac{\partial \bar{g}}{\partial \dot{\mathbf{x}}}^T (\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{u}(t), \lambda(t), t) \right) \Big] \delta \mathbf{x}(t) \\
& + \frac{\partial \bar{g}}{\partial \mathbf{u}}^T (\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{u}(t), \lambda(t), t) \delta \mathbf{u}(t) \\
& + \frac{\partial \bar{g}}{\partial \lambda}^T (\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{u}(t), \lambda(t), t) \delta \lambda(t) \Big\} dt. \quad (A1.4)
\end{aligned}$$

If we represent the quantity $d\mathbf{x}(t_f)$ as the difference between $\mathbf{x}(t_f+dt_f)$ and $\mathbf{x}(t_f)$. Then we have

$$d\mathbf{x}(t_f) = \delta\mathbf{x}(t_f) + \dot{\mathbf{x}}(t_f) dt_f$$

$$\text{or} \quad \delta\mathbf{x}(t_f) = d\mathbf{x}(t_f) - \dot{\mathbf{x}}(t_f) dt_f \quad (\text{A1.5})$$

Equation (A1.2) can now be rewritten as

$$\begin{aligned} \delta\bar{J} = & \frac{\partial \bar{g}^T}{\partial \mathbf{x}}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{u}(t), \lambda(t), t) \bigg|_{t=t_f} d\mathbf{x}(t_f) \\ & - \left[\frac{\partial \bar{g}^T}{\partial \mathbf{x}}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{u}(t), \lambda(t), t) \dot{\mathbf{x}}(t) \right] \bigg|_{t=t_f} dt_f \\ & + \bar{g}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{u}(t), \lambda(t), t) \bigg|_{t=t_f} dt_f \\ & + \int_{t_0}^{t_f} \left\{ \left[\frac{\partial \bar{g}^T}{\partial \mathbf{x}}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{u}(t), \lambda(t), t) \right. \right. \\ & \left. \left. - \frac{d}{dt} \left(\frac{\partial \bar{g}^T}{\partial \mathbf{x}}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{u}(t), \lambda(t), t) \right) \right] \delta\mathbf{x}(t) \right. \\ & \left. + \frac{\partial \bar{g}^T}{\partial \mathbf{u}}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{u}(t), \lambda(t), t) \delta\mathbf{u}(t) \right. \\ & \left. + \frac{\partial \bar{g}^T}{\partial \lambda}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{u}(t), \lambda(t), t) \delta\lambda(t) \right\} dt. \quad (\text{A1.6}) \end{aligned}$$

APPENDIX 2

```

*   PROGRAM MAIN FOR FINITE ELEMENT PACKAGE
      PARAMETER (MBAND=43)
      PARAMETER (IMAX=441)
      PARAMETER (JMAX=IMAX*MBAND)
      PARAMETER (NDMAX=6)
      PARAMETER (NELMS=450)
      PARAMETER (NTLEN=NELMS*NDMAX)
      PARAMETER (NJ=NTLEN/3)
      PARAMETER (IELMAX=900)
      REAL*4   X(IMAX),Y(IMAX),A(JMAX),RSV(IMAX),ELM(NDMAX,NDMAX),
+             PRSV(NDMAX),XI(3),YI(3),NT(NJ,3),ERROR,SUMTS,
+             T(IMAX),TEMP,TOLD(IMAX),TTEST
      INTEGER*4 IDEBUG,NDS,NX,NY,NMAX/IMAX/,MAXBND/MBAND/,
+             MAXMAT/JMAX/,MAXEND/NDMAX/,MAXELM/NELMS/,
+             MAXTTL/NTLEN/,NTABLE(NTLEN),NUMEL,NELM,ITYPE,
+             TWIDTH,NBANDW,ITABLE(NDMAX),NEQ,IT,IB(IELMAX)
      CHARACTER*1 ANS
      CHARACTER*80 FORM
      CHARACTER*2 NB
      EQUIVALENCE (T(1),RSV(1))
      COMMON XL,YL,XH,YH
      READ(5,*)XL,YL,XH,YH
10    IDEBUG=0
      WRITE(6,100)
100   FORMAT(1X,' DO YOU WANT THE DEBUG SWITCH ON (Y OR N)?:')
      READ(5,110)ANS
110   FORMAT(A1)
      IF(ANS.EQ.'Y')IDEBUG=1
120   WRITE(6,130)
130   FORMAT(1X,' SPECIFY THE GRID DENSITY (NX BY NY):')
      READ (5,*)NX,NY
      IF(NX.LT.1.OR.NY.LT.1)THEN
140       WRITE(6,140)
          FORMAT(1X,' ERROR - GRID PARAMETER MUST BE 1 OR GREATER '/')
          GOTO 120
          ELSE IF((NX+1)*(NY+1).GT.NMAX)THEN
              WRITE(6,150)
150       FORMAT(1X,' ERROR - REQUESTED GRID EXCEEDS AVAILABLE STORAGE')
          GOTO 120
      ENDIF
      CALL NODE(X,Y,NMAX,NDS,NX,NY,IDEBUG)
      ITYPE=1
      CALL GRIDLT(NX,NY,NTABLE,NELM,NUMEL,IDEBUG)

```

```

ENDIF
CALL BAND(NTABLE, TWIDTH, NUMEL, NBANDW, NELM, IDEBUG)
NEQ=MAXMAT/NBANDW
IF((NBANDW.GT.MAXBND).OR.(NEQ.LT.NDS))THEN
  WRITE(6,*)'ERROR - INSUFFICIENT MEMORY'
  GOTO 10
ENDIF
TTEST=1.
DO 500 I=1,NDS
  T(I)=0.
500 CONTINUE
DO 900 IT=1,20
  IF(TTEST.GT.1.E-6)THEN
    DO 550 I=1,NDS
      TOLD(I)=T(I)
550 CONTINUE
      CALL BUILD(NTABLE, NUMEL, TWIDTH, NELM, A, RSV, NEQ, NBANDW,
+        NDS, ITYPE, X, Y, NMAX, ELM, PRSV, MAXEND, IDEBUG, ITABLE,
+        IB, TOLD, IT)
      CALL BC(A, NBANDW, NEQ, RSV, NMAX, NDS, X, Y, IDEBUG)
      CALL SOLVE(A, NBANDW, NEQ, RSV, NMAX, NDS)
      ERROR=0.0
      SUMTS=0.0
      DO 600 I=1,NDS
        ERROR=ERROR+(T(I)-TOLD(I))**2
        SUMTS=SUMTS+T(I)*T(I)
600 CONTINUE
        TTEST=SQRT(ERROR/SUMTS)
        WRITE(6,610)IT,TTEST
610 FORMAT(1X,I5,4X,'TEST = ',E16.8)
        DO 700 J=1,NY+1
          WRITE(6,950)(T((J-1)*(NX+1)+I), I=1, NX+1)
          FORMAT(1X,24F5.1,10(/1X,24F5.1))
950 CONTINUE
700 CONTINUE
      ENDIF
900 CONTINUE
      NBA=NBANDW-1
      WRITE(NB,25)NBA
25 FORMAT(I2)
      WRITE(6,*)' THE SOLUTION IS : '
      FORM='( '//NB//' (1X,F4.1) /)'
      WRITE(6,FORM)(RSV(I), I=1,NDS)
      DO 210 J=1,3
        DO 220 I=1,NJ
          NT(I,J)=NTABLE(NJ*(J-1)+I)
220 CONTINUE
210 CONTINUE
      ERROR=0.
      SUMTS=0.
      DO 1100 I=1,NDS
        TEMP=GETTEE(X(I),Y(I))

```

```

        ERROR=ERROR+(TEMP-T(I))**2
        SUMTS=SUMTS+TEMP*TEMP
1100  CONTINUE
        ERROR=SQRT(ERROR/SUMTS)
        WRITE(6,1150)ERROR
1150  FORMAT(1X,' ERROR = ',E16.8)
        WRITE(6,2000)
2000  FORMAT(1X,'DO YOU WANT MAKE A PLOT :')
        READ(5,110)ANS
        IF(ANS.EQ.'Y')THEN
2050      WRITE(6,2100)
2100      FORMAT(1X,'HOW MANY CONTOURS DO YOU WANT:')
            READ(5,*)I
            IF(I.LE.0)GOTO 2050
            NUMEL=MAXTL/3
            CALL GRIDLT(NX,NY,NTABLE,NELM,NUMEL,IDEBUG)
            CALL PLT(X,Y,RSV,NMAX,NDS,NTABLE,NELM,NUMEL,A(1),A(MAXMAT/2),
+          MAXMAT/(2*I),IDEBUG,I)
        ENDIF
        WRITE(6,10000)
10000  FORMAT(1X,' DO YOU WANT TO RUN ANOTHER CASE ')
        READ(5,110)ANS
        IF(ANS.EQ.'Y')THEN
            GOTO 10
        ENDIF
        STOP
        END
        REAL FUNCTION GETTEE(X,Y)
        REAL*4 X,Y,U
        U=1.
        IF((Y.GT.0.0 .AND. X.GT.(-Y*Y*0.5)).OR.
+      (Y.LE.0.0 .AND. X.GT.(Y*Y*0.5)))U=-1.
        GETTEE=-U*Y+2.*SQRT(-U*X+Y*Y*0.5)
        RETURN
        END
* SUBROUTINE FOR NUMBERING NODES
        SUBROUTINE NODE(X,Y,NMAX,NDS,NX,NY,IDEBUG)
        REAL*4 X(NMAX),Y(NMAX)
        INTEGER*4 NMAX,NDS,NX,NY,IDEBUG
        COMMON XL,YL,XH,YH
        NDS=(NX+1)*(NY+1)
        DELTAX=(XH-XL)/FLOAT(NX)
        DELTAY=(YH-YL)/FLOAT(NY)
        IF(NY.GE.NX)THEN
            DO 10 I=1,NDS
                KK=MOD(I,(NX+1))
                IF(KK.EQ.0)THEN
                    X(I)=DELTAX*FLOAT(NX)+XL
                ELSE
                    X(I)=DELTAX*FLOAT(KK-1)+XL
                ENDIF
            ENDIF

```

```

        IY=(I-1)/(NX+1)
        Y(I)=YH-DELTAY*FLOAT(IY)
10      CONTINUE
        ELSE
        DO 20 I=1,NDS
            IX=(I-1)/(NY+1)
            KK=MOD(I,(NY+1))
            IF(KK.EQ.0)THEN
                Y(I)=YH-DELTAY*FLOAT(NY)
            ELSE
                Y(I)=YH-DELTAY*FLOAT(KK-1)
            ENDIF
            X(I)=DELTAX*FLOAT(IX)+XL
20      CONTINUE
        ENDIF
        IF(IDEBUG.NE.0)THEN
            WRITE(9,*)'THE X AND Y VALUES ARE'
            DO 30 I=1,NDS
                WRITE(9,40)I,X(I),Y(I)
40          FORMAT(4X,I4,2(4X,F10.6)/)
30      CONTINUE
        ENDIF
        RETURN
    END
*      SUBROUTINE FORMING THE NODE TABLE FOR EACH ELEMENT
        SUBROUTINE GRIDLT(NX,NY,NTABLE,NELM,NUMEL,IDEBUG)
        INTEGER*4 NX,NY,NTABLE(NUMEL,3),IDEBUG,N1,N2,N3,N4
        NELM=0
        DO 10 I=1,NY
            DO 20 J= 1,NX
                IF(NX.GE.NY)THEN
                    N1=(I-1)*(NX+1)+J
                    N2=N1+1
                    N4=I*(NX+1)+J
                    N3=N4+1
                ELSE
                    N1=(J-1)*(NY+1)+I
                    N4=N1+1
                    N2=J*(NY+1)+I
                    N3=N2+1
                ENDIF
                NELM=NELM+1
                IF(NELM.GT.NUMEL)GOTO 50
                NTABLE(NELM,1)=N1
                NTABLE(NELM,2)=N4
                NTABLE(NELM,3)=N2
                NELM=NELM+1
                IF(NELM.GT.NUMEL)GOTO 50
                NTABLE(NELM,1)=N2
                NTABLE(NELM,2)=N4
                NTABLE(NELM,3)=N3
            
```

```

20     CONTINUE
10     CONTINUE
    IF (IDEBUG.NE.0) THEN
        WRITE(9,*) ' THE NUMBER OF ELEMENTS REQD. FOR THE PROBLEM : ',
+           NELM
        WRITE(9,25)
        FORMAT(// ' THE NODE TABLE IS : '//)
25     DO 40 I=1,NELM
        WRITE(9,30) I,NTABLE(I,1),NTABLE(I,2),NTABLE(I,3)
30     FORMAT(1X,I4,3(2X,I4)/)
40     CONTINUE
    ENDIF
    RETURN
50     WRITE(5,*) 'ERROR *** NUMBER OF ELEMENTS REQD. EXCEEDS MEMORY
+           RESERVED'
    STOP
    END
*   SUBROUTINE FOR BUILDING THE GLOBAL MATRIX
    SUBROUTINE BUILD(NTABLE,NUMEL,TWIDTH,NELM,A,RSV,NEQ,NBANDW,NDS,
+           ITYPE,X,Y,NMAX,ELM,PRSV,MAXEND,IDEBUG,ITABLE,
+           IB,TOLD,IT)
    INTEGER*4 TWIDTH,ITABLE(MAXEND),NTABLE(NUMEL,TWIDTH),IT,IB(NUMEL)
    REAL*4 ELM(MAXEND,MAXEND),PRSV(MAXEND),RSV(NMAX),A(NEQ,NBANDW),
+           X(NMAX),Y(NMAX),TOLD(NMAX)
    DO 10 I=1,NDS
        DO 20 J=1,NBANDW
            A(I,J)=0.
20     CONTINUE
        RSV(I)=0.
10     CONTINUE
    DO 30 I=1,NELM
        DO 40 J=1,TWIDTH
            ITABLE(J)=NTABLE(I,J)
40     CONTINUE
        X1=X(ITABLE(1))
        X2=X(ITABLE(2))
        X3=X(ITABLE(3))
        Y1=Y(ITABLE(1))
        Y2=Y(ITABLE(2))
        Y3=Y(ITABLE(3))
        CALL LINTRI(X1,X2,X3,Y1,Y2,Y3,ELM,PRSV,MAXEND,IDEBUG,IB(I),
+           TOLD(ITABLE(1)),TOLD(ITABLE(2)),TOLD(ITABLE(3)),IT)
    DO 60 K=1,TWIDTH
        IJ=ITABLE(K)
        DO 50 J=1,TWIDTH
            II=ITABLE(J)
            IF(II.LT.IJ)GOTO 50
            IK=II-IJ+1
            A(IJ,IK)=A(IJ,IK)+ELM(K,J)
50     CONTINUE
        RSV(IJ)=RSV(IJ)+PRSV(K)

```

```

60      CONTINUE
30      CONTINUE
      IF(IDEBUG.NE.0)THEN
          WRITE(9,*)' THE GLOBAL MATRIX IS : '
          WRITE(9,21)((A(I,J),J=1,TWIDTH),I=1,NDS)
21      FORMAT(3(F8.4,1X)/)
          WRITE(9,22)(RSV(I),I=1,NDS)
22      FORMAT(5(F8.4)/)
      ENDIF
      RETURN
      END
      SUBROUTINE LINTRI(X1,X2,X3,Y1,Y2,Y3,ELM,ERSV,NM,IDEBUG,IB,
&          T1,T2,T3,IT)

C      LINTRI.FOR IS FOR THE SOLUTION OF THE TRANSVERSALITY CONDITION
C      IN THE X1-X2 PHASE PLANE FOR THE DOUBLE INTEGRATOR PROBLEM.  THE
C      EQUATIONS ARE NONLINEAR AND THE SOLUTION IS OBTAINED THROUGH
C      NEWTON METHOD.  THIS SUBROUTINE BUILDS THE ELEMENT JACOBIAN MATRIX
C      AND EVALUATES THE RESIDUALS ON AN ELEMENT BASIS.  INCLUDED WITH THE
C      THE TRANSVERSALITY CONDITION IS A CLOSED FORM EXPRESSION FOR THE
C      MINIMUM TIME FUNCTIONAL.  THE ELEMENT EQUATIONS ARE DETERMINED FROM
C      THE LEAST SQUARES PROCESS.

      INTEGER*4    NM, IDEBUG, IB, IT
      REAL*4       X1, X2, X3, Y1, Y2, Y3, ELM(NM,NM), ERSV(NM)
      REAL*4       T1, T2, T3

C      LOCAL VARIABLES
      INTEGER*4    I, J, IU, IUI
      REAL*4       A(3), B(3), C(3), U(3), X(3), Y(3), L1, L2, DEL2, X2S
      REAL*4       UJ, A1, UINT, X2UINT, DUINT, DDUINT, DX2U, DEL, DD2U
      REAL*4       H11, H12, H22, C1, C2, C3, C4, C5, C6, C7, C8, C9
      REAL*4       T(3), XAVE, YAVE, H(3,3), G11, G12, G22, F(3), XXX

      A(1) = X2*Y3 - X3*Y2
      A(2) = X3*Y1 - X1*Y3
      A(3) = X1*Y2 - X2*Y1
      B(1) = Y2 - Y3
      B(2) = Y3 - Y1
      B(3) = Y1 - Y2
      C(1) = X3 - X2
      C(2) = X1 - X3
      C(3) = X2 - X1
      Y(1) = Y1
      Y(2) = Y2
      Y(3) = Y3
      X(1) = X1
      X(2) = X2
      X(3) = X3
      T(1) = T1
      T(2) = T2

```

```

T(3) = T3
XAVE = (X1+X2+X3)/3.0
YAVE = (Y1+Y2+Y3)/3.0

DEL2 = A(1) + A(2) + A(3)
IF(DEL2 .LE. 0.0) THEN
  WRITE(*,80) DEL2
80   FORMAT(1X,'ERROR IN GRID : 2 * TRIANGLE AREA =',F12.6)
  STOP
ENDIF

DEL = 0.5 * DEL2
L1 = (-B(1)*T1-B(2)*T2-B(3)*T3)/DEL2
L2 = (-C(1)*T1-C(2)*T2-C(3)*T3)/DEL2
H11 = DEL2*(Y1*Y1+Y2*Y2+Y3*Y3+Y1*Y2+Y1*Y3+Y2*Y3)*2.0/24.0
H22 = DEL
DO 100 I = 1, 3
  U(I) = 1.0
  IF( IT .EQ. 1) THEN
    IF( YAVE .GT. 0.0 .AND. XAVE .GT. (-YAVE*YAVE*0.5) .OR.
      &   YAVE .LE. 0.0 .AND. XAVE .GT. ( YAVE*YAVE*0.5))THEN
      U(I) = -1.0
    ENDIF
  ELSE
    IF( L2*(1.0 - L1*Y(I)) .LT. 0.0 ) U(I) = -1.0
  ENDIF
100 CONTINUE

DO 200 I = 1, 3
  DO 180 J = 1, 3
    H(I,J) = 2.0 * X(I) * B(J) + Y(I) * C(J)
180 CONTINUE
200 CONTINUE
DO 220 I = 1, 3
  F(I) = 0.0
  DO 210 J = 1, 3
    F(I) = F(I) + H(J,I)
210 CONTINUE
220 CONTINUE
G11 = DEL2*(X1*X1+X2*X2+X3*X3+X1*X2+X1*X3+X2*X3)/3.0
G12 = DEL2*(2.0*(X1*Y1+X2*Y2+X3*Y3)+X1*Y2+X2*Y1+X1*Y3+X3*Y1+
  &   X2*Y3+X3*Y2)/12.0
G22 = DEL2*(Y1*Y1+Y2*Y2+Y3*Y3+Y1*Y2+Y1*Y3+Y2*Y3)/12.0

DO 250 I = 1, 3
  DO 240 J = I, 3
    ELM(I,J) = DEL2/24.0 - (H(I,J)+H(J,I)+F(I)+F(J))/24.0
    ELM(I,J) = ELM(I,J) + (B(I)*B(J)*G11+(B(I)*C(J)+B(J)*C(I))
      &   *G12+C(I)*C(J)*G22)/(DEL2*DEL2)
    IF(I .EQ. J) THEN
      ELM(I,I) = ELM(I,I) + DEL2/24.0
    
```

```

ELSE
  ELM(J,I) = ELM(I,J)
ENDIF
240  CONTINUE
250  CONTINUE
DO 270 I = 1, 3
  ERSV(I) = 0.0
  DO 260 J = 1, 3
    ERSV(I) = ERSV(I) - ELM(I,J)*T(J)
260  CONTINUE
270  CONTINUE

IF( U(1)*U(2) .LT. 0.0 .AND. IT .NE. 1 .OR.
&  U(1)*U(3) .LT. 0.0 .AND. IT .NE. 1 .OR.
&  U(2)*U(3) .LT. 0.0 .AND. IT .NE. 1 ) THEN
  IF( U(1) .NE. U(2) .AND. U(1) .NE. U(3) ) IU = 1
  IF( U(2) .NE. U(1) .AND. U(2) .NE. U(3) ) IU = 2
  IF( U(3) .NE. U(1) .AND. U(3) .NE. U(2) ) IU = 3
  X2S = 1.0 / L1
  UJ = U(IU)
  IU1 = IU - 1
  IF(IU1 .EQ. 0) IU1 = 3
  A1 = DEL * (X2S - Y(IU))/(Y(IU1) - Y(IU))
  UINT = UJ * (2.0*A1 - DEL)
  X2UINT = UJ*(2.0*A1*(2.0*X2S+Y(IU))-DEL*(Y1+Y2+Y3))/3.0
  DUINT = UJ/(Y(IU1)-Y(IU))*L1*L1
  DDUINT = DUINT / (DEL * L1)
  DX2U = UJ*(DEL*(2.0*X2S+Y(IU))/(Y(IU1)-Y(IU))+2.0*A1)/
&      (3.0*DEL*L1*L1)
  DDX2U = 2.0*UJ/(3.0*(Y(IU1)-Y(IU))*DEL*(L1**4))+DX2U/(DEL*L1)

  C1 = (DUINT+X2UINT/DEL2)/DEL2
  C2 = 1.0/(DEL2*DEL2)
  C3 = -DX2U/DEL2
  C4 = -L2*DDUINT + L1*L2*DDX2U
  C5 = C1 + L1 * C3
  C6 = C2*H11 + C3*2.0*L2 + C4
  C7 = C2*H22
  C8 = (Y1+Y2+Y3)/6.0-L1*H11/DEL2+(L1*DX2U-X2UINT/DEL2-DUINT)*L2
  C9 = (UINT - L2*H22 - L1*X2UINT)/DEL2

DO 500 I = 1, 3
  ERSV(I) = ERSV(I) - C8*B(I)-C9*C(I)
  DO 300 J = 1, 3
    ELM(I,J)=ELM(I,J)+C5*(C(I)*B(J)+C(J)*B(I))+
&      C6*B(I)*B(J)+C7*C(I)*C(J)
    ELM(J,I)=ELM(I,J)
300  CONTINUE
DO 400 J = 1, 3
  ERSV(I) = ERSV(I) + ELM(I,J)*T(J)
400  CONTINUE

```



```

500      CONTINUE
      ELSE
        IF( IT .EQ. 1) THEN
          UJ = 1.0
          IF( YAVE .GT. 0.0 .AND. XAVE .GT. (-YAVE*YAVE*0.5) .OR.
            &      YAVE .LE. 0.0 .AND. XAVE .GT. ( YAVE*YAVE*0.5))THEN
            UJ = -1.0
          ENDIF
        ELSE
          UJ = U(1)
        ENDIF
        H12 = UJ*DEL*(Y1+Y2+Y3)/3.0
        UINT = UJ*DEL
        DO 700 I = 1, 3
          ERSV(I) = ERSV(I) - B(I)*(Y1+Y2+Y3)/6.0 - C(I)*UINT/DEL2
          DO 600 J = 1, 3
            XXX = (B(I)*B(J)*H11+(C(I)*B(J)+C(J)*B(I))*H12+
            &      C(I)*C(J)*H22)/(DEL2*DEL2)
            ERSV(I) = ERSV(I) - XXX * T(J)
            ELM(I,J) = ELM(I,J) + XXX
          DO 600 J = 1, 3
            CONTINUE
          DO 700 I = 1, 3
            CONTINUE
            DO 800 J = 1, 3
              ERSV(I) = ERSV(I) + ELM(I,J) * T(J)
            CONTINUE
          DO 900 J = 1, 3
            CONTINUE
          ENDIF
        IF(IDEBUG .NE. 0) THEN
          WRITE(*,2000) X1, X2, X3, Y1, Y2, Y3
          FORMAT(1X,'LINEAR TRIANGLE',6F12.5)
          DO 2500 I = 1, 3
            WRITE(*,2200) (ELM(I,J),J=1,3), ERSV(I)
            FORMAT(1X,3F12.5, 10X, F12.5)
          CONTINUE
          ENDIF
        RETURN
      END
    * SUBROUTINE FOR INCLUDING THE BOUNDARY CONDITIONS
      SUBROUTINE BC(A,NBANDW,NEQ,RSV,NMAX,NDS,X,Y,IDEBUG)
      INTEGER*4 NBANDW,NEQ,NMAX,NDS,IDEBUG
      REAL*4 A(NEQ,NBANDW),RSV(NMAX),X(NDS),Y(NDS)
      CHARACTER*80 FORM
      CHARACTER*2 NB
      COMMON XL,YL,XH,YH
      DO 10 I=1,NDS
        IF(((ABS(X(I)-XL)).LE.1.E-3).OR.((ABS(X(I)-XH)).LE.1.E-3).OR.
          + ((ABS(Y(I)-YL)).LE.1.E-3).OR.((ABS(Y(I)-YH)).LE.1.E-3).OR.
          + ((ABS(X(I)).LE.1.E-3).AND.((ABS(Y(I))).LE.1.E-3)))THEN

```

```

      U=1.
      IF(((Y(I).GT.0.0).AND.(X(I).GT.(-Y(I)*Y(I)*0.5))).OR.
+      ((Y(I).LE. 0.0).AND.(X(I).GT.(Y(I)*Y(I)*0.5))))THEN
        U=-1.0
      ENDIF
      SQA=-U*X(I)+Y(I)*Y(I)*0.5
      IF(SQA.LE.1.E-6)SQA=0.
      RSV(I)=-U*Y(I)+2.*SQRT(SQA)
      IF(RSV(I).LT.0.)RSV(I)=-RSV(I)
      RSV(I)=RSV(I)*A(I,1)*1.E20
      A(I,1)=A(I,1)*1.E20
    ENDIF
10  CONTINUE
    WRITE(NB,5)NBANDW
5    FORMAT(I2)
    FORM='('//NB//'(E12.3,1X)/)
    IF(IDEBUG.NE.0)THEN
      WRITE(9,*)' THE GLOBAL MATRIX IS :'
      WRITE(9,FORM)((A(I,J),J=1,NBANDW),I=1,NDS)
      WRITE(9,112)(RSV(I),I=1,NDS)
112  FORMAT(5(E14.4,1X)/)
    ENDIF
    RETURN
  END
*  SUBROUTINE FOR SOLVING THE GLOBAL MATRIX
    SUBROUTINE SOLVE (A, HBANDW, NEQ, RSV, NMAX, NDS)

      INTEGER*4  HBANDW, NEQ, NMAX, NDS, I, J, K, MAXCOL
      REAL*4     A(NEQ,HBANDW), RSV(NMAX), DENOM, TERM, SUM
C  REDUCE MATRIX
C  LOOP ON COLUMNS TO BE REDUCED
      DO 300 I = 1, NDS-1
        DENOM = 1.0/A(I,1)
        MAXCOL = NDS - I + 1
        IF(MAXCOL.GT. HBANDW) THEN
          MAXCOL = HBANDW
        ENDIF
        DO 200 J = 2, MAXCOL
          IF(ABS(A(I,J)).GT. 1.0E-20) THEN
            TERM = A(I,J) * DENOM
            IF(ABS(TERM).GT. 1.0E-25) THEN
              DO 100 K = J, MAXCOL
                IF(ABS(A(I,K)).GT. 1.0E-25) THEN
                  A(I+J-1,K-J+1)=A(I+J-1,K-J+1)-TERM*A(I,K)
                ENDIF
100             CONTINUE
                RSV(I+J-1) = RSV(I+J-1) - TERM * RSV(I)
              ENDIF
            ENDIF
          ENDIF
        CONTINUE
200      CONTINUE
300    CONTINUE

```

```

C BACK SUBSTITUTE
  DO 500 I = NDS, 1, -1
    SUM = RSV(I)
    DO 400 J = 2, HBANDW
      K = I + J - 1
      IF(K .LE. NDS) THEN
        IF(ABS(RSV(K)) .GT. 1.0E-25) THEN
          IF(ABS(A(I,J)) .GT. 1.0E-25) THEN
            SUM = SUM - RSV(K) * A(I,J)
          ENDIF
        ENDIF
      ENDIF
    CONTINUE
  CONTINUE
  RSV(I) = SUM / A(I,1)
400  CONTINUE
500  RETURN
END
* SUBROUTINE FOR DIVIDING THE REGION INTO CONTOURS
SUBROUTINE PLT(X, Y, T, NMAX, NDS, NTABLE, NELM, NUMEL, XS,
&             YS, MAXP, IDEBUG, NLINE)

  INTEGER*4    I, NLINE, NMAX, NPP(100), J, K, K1, K2, IMIN, IMAX
  INTEGER*4    NDS, IDEBUG, NELM, NUMEL, MAXP, NTABLE(NUMEL,3)
  REAL*4       X(NMAX), Y(NMAX), XS(MAXP,NLINE), YS(MAXP,NLINE)
  REAL*4       T(NMAX), TMAX, TMIN, XX, YY, RATIO, TINC, ETMAX
  REAL*4       ETMIN
  CHARACTER*1  ANS
  IF(IDEBUG .NE. 0) THEN
    DO 10 I = 1, NELM
      WRITE(9,5)I,(NTABLE(I,J),J=1,3)
      FORMAT(1X,I4,5X,3I6)
5     CONTINUE
10    DO 20 I = 1, NDS
      WRITE(9,15)I,X(I),Y(I), T(I)
      FORMAT(1X,I5,3F12.5)
15    CONTINUE
20    ENDIF
11    TMAX = T(1)
    TMIN = T(1)
    DO 50 I = 2, NDS
      IF(T(I) .GT. TMAX) TMAX = T(I)
      IF(T(I) .LT. TMIN) TMIN = T(I)
50    CONTINUE
    DO 1000 I = 1, NLINE
      NPP(I) = 0
      TINC = FLOAT(2*I-1)*(TMAX-TMIN)/FLOAT(2*NLINE) + TMIN
      IF(IDEBUG .NE. 0) WRITE(9,55) TINC
      FORMAT(1X,'TINC = ',E16.8)
85    DO 500 J = 1, NELM
      ETMAX = T(NTABLE(J,1))
      ETMIN = T(NTABLE(J,1))

```

```

IMAX = 1
IMIN = 1
DO 100 K = 2, 3
  IF(T(NTABLE(J,K)).GT. ETMAX) THEN
    ETMAX = T(NTABLE(J,K))
    IMAX = K
  ENDIF
  IF(T(NTABLE(J,K)) .LT. ETMIN) THEN
    ETMIN = T(NTABLE(J,K))
    IMIN = K
  ENDIF
CONTINUE
IF(TINC .GE. ETMIN .AND. TINC .LE. ETMAX) THEN
  DO 400 K = 1, 3
    K1 = K
    K2 = K + 1
    IF(K2 .EQ. 4) K2 = 1
    IF(ABS(T(NTABLE(J,K1))-T(NTABLE(J,K2)))
      & .LT. 1.0E-5 * ABS(TINC) .AND. ABS(T(NTABLE(J,K1))-
      & TINC) .LT. 1.0E-5 * ABS(TINC)) THEN
      XS(NPP(I)+1,I) = X(NTABLE(J,K1))
      XS(NPP(I)+2,I) = X(NTABLE(J,K2))
      YS(NPP(I)+1,I) = Y(NTABLE(J,K1))
      YS(NPP(I)+2,I) = Y(NTABLE(J,K2))
      NPP(I) = NPP(I) + 2
    ELSE
      ETMIN = T(NTABLE(J,K1))
      ETMAX = T(NTABLE(J,K2))
      IF(ETMIN .GT. ETMAX) THEN
        XX = ETMIN
        ETMIN = ETMAX
        ETMAX = XX
      ENDIF
      IF(TINC .GT. ETMIN .AND. TINC .LT. ETMAX) THEN
        & RATIO = (TINC-T(NTABLE(J,K1)))/(
        & T(NTABLE(J,K2)) - T(NTABLE(J,K1)))
        & XX = RATIO * (X(NTABLE(J,K2))-
        & X(NTABLE(J,K1))) + X(NTABLE(J,K1))
        & YY = RATIO * (Y(NTABLE(J,K2))-
        & Y(NTABLE(J,K1))) + Y(NTABLE(J,K1))
        NPP(I) = NPP(I) + 1
        XS(NPP(I),I) = XX
        YS(NPP(I),I) = YY
      ENDIF
    ENDIF
  CONTINUE
CONTINUE
ENDIF
CONTINUE
IF(NPP(I) .GT. MAXP) THEN
  WRITE(6,600)
  FORMAT(1X,'MORE DATA POINTS REQUIRED FOR PLOT THAN',

```

```

&      ' AVAILABLE - USE FEWER LINES')
      READ(5,111)ANS
      GOTO 11
    ENDIF
1000  CONTINUE

      CALL PLOT(XS,YS,NPP,NMAX,T,NLINE,MAXP)

112  WRITE(5,*)' DO YOU WANT ANOTHER PLOT:'
      READ(5,111)ANS
111  FORMAT(A1)
      IF(ANS.EQ.'Y')THEN
        WRITE(5,*)' ENTER NO. OF CONTOURS:'
        READ(5,*)NLINE
        GOTO 11
      ELSE IF(ANS.NE.'N')THEN
        GOTO 112
      ENDIF
      RETURN
      END
      SUBROUTINE TO4014
      INTEGER ARRAY(2)/27,49/
      CHARACTER ICHRS
C      ICHRS = ' '
C      CALL KAS2AM(2,ARRAY,ICHRS)
      WRITE(6,555) ICHRS
555  FORMAT(' ', A2)
      RETURN
      END
      SUBROUTINE TOANSI
      INTEGER ARRAY(2)/27,50/
      CHARACTER ICHRS
C      ICHRS = ' '
C      CALL KAS2AM(2,ARRAY,ICHRS)
      WRITE(6,555) ICHRS
555  FORMAT(' ', A2)
      RETURN
      END
*      SUBROUTINE FOR PLOTTING THE ISOCHRONES
      SUBROUTINE PLOT (XS,YS,NPP,NMAX,T,NLINE,MAXP)
      REAL*4 XS(MAXP,NLINE),YS(MAXP,NLINE),T(NMAX)
      INTEGER*4 NLINE,NPP(100)
      COMMON XL,YL,XH,YH
      CALL TO4014
      CALL GRSTRT(4014,1)
      CALL NEWPAG
      CALL WINDOW(XL,XH,YL,YH)
      CALL VWPORT(0.,130.,0.,100.)
      CALL CMOPEN
      CALL MOVE(XL,YL)
      CALL CMCLOS

```

```

CALL CMOPEN
DO 10 I=1,NLINE
  DO 20 J=1,NPP(I)/2
    CALL MOVE(XS(2*J-1,I),YS(2*J-1,I))
    CALL DRAW(XS(2*J,I),YS(2*J,I))
20  CONTINUE
10  CONTINUE
CALL MOVE(XL,YL)
CALL DRAW(XH,YL)
CALL DRAW(XH,YH)
CALL DRAW(XL,YH)
CALL DRAW(XL,YL)
CALL CMCLOS
CALL GRSTOP
CALL TOANSI
RETURN
END

```

A CONTINUUM APPROACH TO MINIMUM TIME CONTROL

by

KOTHANDARAMAN RAJENDRAN

B.E., University of Madras, INDIA, 1984

AN ABSTRACT OF A THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Mechanical Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1988

ABSTRACT

A continuum approach to the minimum time problem has been taken in order to approximately determine the isochrone distribution in the phase plane of a double integrator problem. The approximate solution and the actual solution have been provided.

In this analysis the system state variables have been treated as independent variables and time has been eliminated from the analysis. This choice of variables enables us to treat both linear and nonlinear systems with the same methods of solution. Moreover, the costate variables can be determined directly from the gradient with respect to the state variables of the final time at any point in state space in accordance with the Hamilton-Jacobi-Bellman equation. The continuum relations have been derived and an approximate solution has been obtained for a double integrator problem by using a finite element technique.

The approximate solution has been compared with the actual solution and a plot of the isochrones has been provided.