ABSTRACT

A Successive Approximation Algorithm to Solve Motion Planning Problems

by
Cheng-Ming Chang

A successive approximation algorithm is presented to solve motion planning problems and optimal control problems. A motion planning problem is included in an optimal control problem and due to the heavily nonlinear and coulped differential equations describing the dynamics of the manipulator and several constraints imposed on the system, few efficient methods exist to solve this kind of problems.

The successive approximation algorithm can decompose the global nonlinear problem into several nonlinear subproblems and each subproblem is solved by the nonlinear programming method with a highly convergent rate. Because the limited constraints and variables are included in each subproblem and only the trajectory and control sequence are needed to be stored in each iteration, the algorithm can be very easily implemented on the motion planning problems.

The convergence and optimality of the algorithm are also discussed and the implementations on motion planning problems and optimal control problems are presented with excellent results. The computational results show its promise for future applications.

A SUCCESSIVE APPROXIMATION ALGORITHM TO SOLVE MOTION PLANNING PROBLEMS

by
Cheng-Ming Chang

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science

Department of Electrical and Computer Engineering

January, 1993

APPROVAL PAGE

A Successive
Approximation Algorithm
to Solve Motion Planning Problems

Cheng-Ming Chang

Dr. Meng-Chu Zhou, Thesis Adviser
Assistant Professor of Electrical and Computer Engineering, NJIT

Dr. Marshall Kuo, Committee Member
Professor of Electrical and Computer Engineering, NJIT

Dr. Yun-Qing Shi, Committee Member
Assistant Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Cheng-Ming Chang

Degree: Master of Science in Electrical and Computer Engineering

Date: January, 1993

Undergraduate and Graduate Education:

- Master of Science in Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, 1993
- Bachelor of Engineering in Mechanical Engineering, Tamkang University, Tamsui, Taiwan, R.O.C., 1987

Major: Electrical and Computer Engineering

This thesis is dedicated to
my father and my mother

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my advisor, Professor Meng-Chu Zhou, for his guidance, friendship, and moral support throughout this research.

Special thanks to Professors Marshall Kuo and Yun-Qing Shi for serving as members of the committee. Both of them suggested many valuable opinions for this research and professor Kuo gave me much of knowledge on optimal control problems.

Many of my problems have been solved through discussions and assistances with my friends: Tsan–Horng Lay, Huang–Lien Yang and Shih–Yi Huang. Many thanks to them.

A work could not have been successfully accomplished without a proper environment. A deep sense of gratitude is felt for all who created such an environment during my stay at NJIT.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 What is Motion Planning

Robots are the most important tools for automation in industries today. Many factories applied a large number of robots in their production lines or other fields. Hence the topics of the robotic research become very popular recently. One of the most important topics in the robotic research is the motion control problem. A good controller for the robot can produce the following characteristics: [1]

(1) *The robot can successfully and precisely accomplish the tasks which are assigned by the people.*

(2) *The robot can execute the versatile tasks in the different environments.*

(3) *The robot can finish the tasks in the best way.*

Therefore, the controller is a very important element for the robot.

In traditional, the motion control problem was divided into three smaller subproblems due to the highly nonlinear and coupled nature of the differential equations that describe the dynamics of the manipulator and several constraints imposed on the system which include the limitations on the amounts of torques/forces available from the actuators and the presence of the obstacles in the environment. The three subproblems have been described by Singh [1] as follows:

Path Planning: Given a manipulator and a description of its environment, plan a path between two specified positions, such that the path avoids collision with obstacles in the environment and is optimal with regard to a *geometric* performance index, such as shortest path.

Trajectory Planning: Given a path to be followed by the end-effector, the differential equations describing the manipulator dynamics and the constraints on the torques/forces avaiable from the actuators, find the time history of the positions and velocities along the path so that a given performance index is minimized.

Trajectory Tracking: Given a reference trajectory to be tracked and the dynamics of the manipulator, design a feedback controller to accurately track the given trajectory.

This division is not very reasonable, because the path planning problem considers only the geometric configurations in the environment but completely ignores the dynamics of the manipulator. On the other hand, the trajectory planning problem is concerned with only the dynamics of the manipulator by simply assuming that the path has already been planned and its task is only to plan the timming and velocity along this path. According to the above reasons, we can know that the path planning cannot account for the optimal control problems and the trajectory planning cannot obtain the real results for the optimal control problems.

In recent research [1,2], the path planning problem and the trajectory planning problem have just been combined as a motion planning problem. Then the overall motion control problem of the robotic manipulator can be separated into two stages — *the motion planning stage* (planner) and *the trajectory tracking stage* (tracker). The idea of the overall motion control problem may be explained in Figure 1.1. [1]

In the motion planning stage, the planner accepts overall informations which include the capabilities of the manipulator, the descriptions of the task and the conditions of the environment at the beginning. Then the planner plans the complete task to obtain the desired trajectory which is the time history of the positions and velocities and goes to the second stage. In the trajectory tracking stage, the tracker accepts the desired trajectory as its input and precisely tracks this trajectory until the task accomplishment, but the planner does not accept any informations during

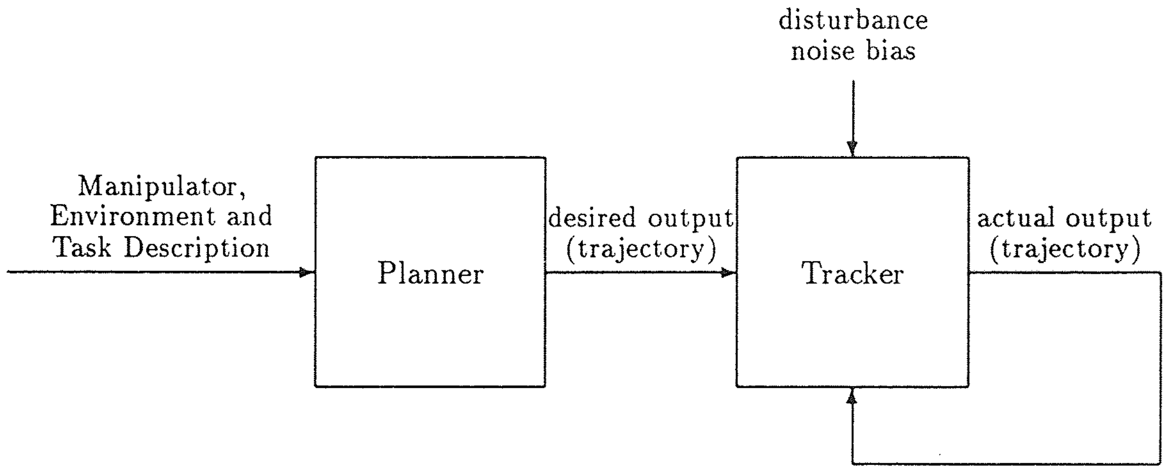
disturbance
noise bias
Manipulator, Environment and Task Description
Planner
desired output (trajectory)
Tracker
actual output (trajectory)

Figure 1.1 The Idea of the Motion Control Problem. [1]

the tracker execution.

The motion planning problem may be identified as follows.

Motion Planning: Given a manipulator and the differential equations that describe the dynamics of the manipulator, consider the constraints including the limitations on the torques/forces available from the actuators and the presence of obstacles in the environment, find the time history of the positions and velocities between two specified positions so that a given performance index is minimized.

1.2 Motion Planning and Optimality

The idea of motion planning is extremely close to that of optimality. Because when we plan a task, we always hope the task will be accomplished in the most efficient way. "The most efficient way" is equivalent to the "optimality", such as the minimum time, the minimum energy and the shortest path. Therefore, we can say that the overall motion planning problem can be considered as a classic optimal control problem which may be formulated as follows: [1,3,4]

The performance measure

$$J = h(\bar{x}(0)) + \int_0^T V(\bar{x}(t), \bar{u}(t), t)\, dt + l(\bar{x}(T)), \tag{1.1}$$

subject to the dynamics

$$\dot{\bar{x}}(t) = F(\bar{x}(t), \bar{u}(t), t) \tag{1.2}$$

and the constraints

$$(\bar{x}(t), \bar{u}(t), t) \in S(t), \tag{1.3}$$

where

$$S(t) = \{(\bar{x}(t), \bar{u}(t), t) : G(\bar{x}(t), \bar{u}(t), t) \geq 0, H(\bar{x}(t), \bar{u}(t), t) = 0\}, \tag{1.4}$$

$$G(\bar{x}(t), \bar{u}(t), t) = (g_1(\bar{x}(t), \bar{u}(t), t), \ldots, g_r(\bar{x}(t), \bar{u}(t), t)) \tag{1.5}$$

$$H(\bar{x}(t), \bar{u}(t), t) = (h_1(\bar{x}(t), \bar{u}(t), t), \ldots, h_s(\bar{x}(t), \bar{u}(t), t)) \tag{1.6}$$

and

$$\bar{x}(t) \in R^n, \bar{u}(t) \in R^m$$

$$h : R^n \longrightarrow R$$

$$l : R^n \longrightarrow R$$

$$V : R^n \times R^m \times R \longrightarrow R$$

$$F : R^n \times R^m \times R \longrightarrow R^n$$

$$g_p : R^n \times R^m \times R \longrightarrow R$$

$$h_q : R^n \times R^m \times R \longrightarrow R$$

It should be noted that the boundary conditions are also included in the constraints.

1.3 The Methods for Solving Motion Planning Problem

In the last section, we mentioned that the motion planning problem can be considered as an optimal control problem. Therefore, any methods, which can solve the optimal control problems, can also solve the motion planning problems. Today, there are three main methods for solving the optimal control problems, which are Pontryagin's minimum principle [5], dynamic programming [6] and nonlinear programming [7]. Each of them has the researchers to implement on the motion planning problems, but owing to the complex dynamics and heavy constraints, it is still extremely difficult to apply these methods to solve this kind of problems.

In the following, we will discuss the advantages and disadvantages of each method. [3,8]

- **Pontryagin's Minimum Principle**

  Advantage: This method can get the accurate and continuous solutions for the motion planning problems.

  Disadvantage: Due to the complex dynamics and heavy constraints, it can solve some simplified motion planning problems only.

- **Dynamic programming**

  Advantage: This method can handle the complex dynamics and heavy constraints by using numeric methods.

  Disadvantage: When we consider the dynamic programming problems, high dimensionality and need for large storage capacity in digital computer often produce severe difficulty in computation.

- **Nonlinear programming**

  Advantage: This method can also handle the complex dynamics and heavy constraints. In addition, it doesn't need for large storage capacity in digital computer.

Disadvantage: When the number of the stages in the problem is increased, the constraints are also increased proportionally. This makes the computation more complicated.

1.4 Successive Approximation Algorithm

The successive approximation algorithm [3] is developed from the two-stage algorithm [9]. Both of them adopt the same principle; that is the two algorithms deal with the problems considering only two stages in each iteration. Among many techniques to tackle the difficulty of high dimensionality when the dynamic programming problems are considered, the two-stage algorithm is the most efficient, conceptually simple and easy to implement.

For solving the optimal control problems, the two-stage algorithm applys the theory of the classic dynamic programming which is Bellman's principle of optimality [6]. By using the Bellman's principle of optimality, Howson and Sancho [9] have proved that the two-stage algorithm is convergent if the criterion function of a problem is strictly convex and if the dynamics can be inverted so that control was a function of state. On the other hand, the successive approximation algorithm does not apply the theory of the dynamic programming but applys the theory of the nonlinear programming. In this study, we can prove that by using the theory of the nonlinear programming, the successive approximation algorithm can solve a class of deterministic multistage decision problems with equality and inequality constraints on both state and control variables and with a general performance index that can cover both separable and nonseparable objective functions. Therefore, we can say that the successive approximation algorithm is different from the two-stage algorithm.

Although the successive approximation algorithm is different from the two-stage algoritm, it still preserves the advantages of the two-stage algorithm. For example, it can also tackle the difficulties of the classic dynamic programming problems. In addition, because the algorithm deals with the problems considering only two stages in each iteration, the constraints will remain invariant when the computation proceeds.

In the next chapter, we will describe the proposed algorithm and discuss the convergence and optimality of this algorithm.

1.5 Previous Works in Motion Planning Problem

The motion planning problems have been extensively discussed by many researchers. In the earlier days, most planners did not account for the dynamics of the manipulators, mainly because of the highly nonlinear and coupled nature of the dynamics and several constraints imposed on the system. They deal with only the geometric problems in the environment, such as the shortest path. This kind of problems have been widely researched and some results have been reported [10,11,12].

By using the Pontryagin's minimum principle, Kahn and Roth [13] studied the minimum time planning problem. They simplified this problem by considering the manipulator travelled from an initial point to a target point with the maximum torques/forces which were assumed to be constant. Niv and Auslander [14] applied the minimum principle to a more general problem which included the state and control constraints of the form

$$C(\bar{x}(t), \bar{u}(t), t) \leq 0.$$

They simplified the problem by assuming that for the minimum time problem, the control would be extremal (bang-bang) all the time and the only unknowns would be the switching time for each joint actuator. Then the only task is to find the switching time for each joint actuator.

By using the dynamic programming, Shin and Mckay [15] applied this method to the paths which could be parameterized. Singh [8] applied this method to the general motion planning problems which are described in Section 1.2. In his study, he solved the minimum time, minimum energy and weighted time and energy minimization problems for the manipulators of one, two or three links.

An attempt to introduce optimal planning with obstacle avoidance was made in Dubowsky et al [2]. Singh and Leu [16] used the distance function approach to solve the motion planning problem with obstacle avoidance. They first applied the dynamic programming to get a coarse solution and applied the nonlinear programming to get the accurate solution. But when the nonlinear programming was considered, there were more than three hundred constraints imposed on the system. They used the active set strategy to reduce the inequality constraints, but the maximum number of the inequality constraints at any time during the search was 36.

1.6 Summary

Robots are the most popular and primary tools for automation in industries. Many researchers have devoted themselves to robotic research. One of the most important topics in the robotic research is the motion planning problem which is considered as a classic optimal control problem. However, due to the highly nonlinear and coupled dynamics of the manipulator and several constraints imposed on the system, most of the methods are extremely difficult to solve this kind of problems. In this study, we propose a successive approximation algorithm to solve the motion planning problem and this method is efficient, conceptually simple and easy to implement.

This thesis is organized as follows. Chapter 2 will describe the successive approximation algorithm and discuss the convergence and optimality of this algorithm. Chapter 3 will implement on an optimal control problem to explain the algorithm. Chapter 4 will simulate on the robotic manipulators. Chapter 5 will present our conclusions and some remarks for future research directions.

CHAPTER 2

SUCCESSIVE APPROXIMATION ALGORITHM

2.1 Discrete Approximation of the Continuous system

In Chapter 1, we have metioned that the motion planning problem can be considered as the optimal control problem which is formulated as Equations (1.1)–(1.6). In order to apply the numeric methods to solving this kind of problems, we must convert a continuous system into a discrete system. This can be done by decomposing the time interval [0,T] into N intervals, and the discrete points may be defined as $\{t_0, t_1, \ldots, t_N\}$, where

$$0 = t_0 < t_1 < \cdots < t_N = T.$$

Then the discrete optimal control problem can be formulated as follows: [1,3,4] The performance measure

$$J = h(\bar{x}(0)) + \sum_{k=0}^{N-1} \int_{t_k}^{t_{k+1}} V(\bar{x}(k), \bar{u}(k), k)\, dt + l(\bar{x}(N)), \tag{2.1}$$

subject to the dynamics

$$\bar{x}(k+1) = \bar{x}(k) + \int_{t_k}^{t_{k+1}} F(\bar{x}(k), \bar{u}(k), k)\, dt; k = 0, 1, \ldots, N-1 \tag{2.2}$$

and the constraints

$$(\bar{x}(k), \bar{u}(k), k) \in S(k); k = 0, 1, \ldots, N. \tag{2.3}$$

In order to simplify these formulas, we assume

$$V_d(\bar{x}(k), \bar{u}(k), k) = \int_{t_k}^{t_{k+1}} V(\bar{x}(k), \bar{u}(k), k)\, dt \tag{2.4}$$

and

$$F_d(\bar{x}(k), \bar{u}(k), k) = \bar{x}(k) + \int_{t_k}^{t_{k+1}} F(\bar{x}(k), \bar{u}(k), k)\, dt. \tag{2.5}$$

Substituting Equation (2.4) into (2.1) and (2.5) into (2.2), the overall discrete optimal control problem can be reformulated as follows:

The performance measure

$$J = h(\bar{x}(0)) + \sum_{k=0}^{N-1} V_d(\bar{x}(k), \bar{u}(k), k) + l(\bar{x}(N)), \tag{2.6}$$

subject to the dynamics

$$\bar{x}(k+1) = F_d(\bar{x}(k), \bar{u}(k), k); k = 0, 1, \ldots, N-1 \tag{2.7}$$

and the constraints

$$(\bar{x}(k), \bar{u}(k), k) \in S(k); k = 0, 1, \ldots, N,$$

where

$$S(k) = \{(\bar{x}(k), \bar{u}(k), k) : G(\bar{x}(k), \bar{u}(k), k) \geq 0, H(\bar{x}(k), \bar{u}(k), k) = 0\};$$

$$k = 0, 1, \ldots, N, \tag{2.8}$$

$$G(\bar{x}(k), \bar{u}(k), k) = (g_1(\bar{x}(k), \bar{u}(k), k), \ldots, g_r(\bar{x}(k), \bar{u}(k), k)) \tag{2.9}$$

$$H(\bar{x}(k), \bar{u}(k), k) = (h_1(\bar{x}(k), \bar{u}(k), k), \ldots, h_s(\bar{x}(k), \bar{u}(k), k)) \tag{2.10}$$

and

$$\bar{x}(k) \in R^n, \bar{u}(k) \in R^m$$

$$h : R^n \longrightarrow R$$

$$l : R^n \longrightarrow R$$

$$V_d : R^n \times R^m \times R \longrightarrow R$$

$$F_d : R^n \times R^m \times R \longrightarrow R^n$$

$$g_p : R^n \times R^m \times R \longrightarrow R$$

$$h_q : R^n \times R^m \times R \longrightarrow R$$

It should be noted that in the continuous system, $\bar{x}$ is a vector function, but in the discrete system, the state is a finite dimensional vector $\overline{X}$, which is given by

$$\overline{X} = \{\bar{x}(0), \bar{x}(1), \ldots, \bar{x}(N)\} \in R^{n(N+1)} \tag{2.11}$$

Similarly, the control vector $\overline{U}$ is given by

$$\overline{U} = \{\bar{u}(0), \bar{u}(1), \ldots, \bar{u}(N-1)\} \in R^{mN} \tag{2.12}$$

For the variable-time problems where the final time T is not fixed, the control vector $\overline{U}$ should be

$$\overline{U} = \{u_0(0), u_1(0), \ldots, u_m(0), \ldots, u_0(N-1), u_1(N-1), \ldots, u_m(N-1),$$

$$\delta t(0), \delta t(1), \ldots, \delta t(N-1)\} \in R^{(m+1)N} \tag{2.13}$$

This will be considered in the minimum time or weighted time and quadratic acceleration minimization control problems.

2.2 The Successive Approximation Algorithm

Consider the following deterministic multistage decision problem, which minimize the following general performance index

$$J\{[\bar{x}(0), \bar{u}(0)], \ldots, [\bar{x}(N-1), \bar{u}(N-1)], [\bar{x}(N)]\} \tag{2.14}$$

and satisfy the dynamics and constraints

$$\bar{x}(k+1) = F_d(\bar{x}(k), \bar{u}(k), k); k = 0, 1, \ldots, N-1, \tag{2.7}$$

$$(\bar{x}(k), \bar{u}(k), k) \in S(k); k = 0, 1, \ldots, N, \tag{2.3}$$

where

$$S(k) = \{(\bar{x}(k), \bar{u}(k), k) : G(\bar{x}(k), \bar{u}(k), k) \geq 0, H(\bar{x}(k), \bar{u}(k), k) = 0\};$$

$$k = 0, 1, \ldots, N,$$

$$G(\bar{x}(k), \bar{u}(k), k) = (g_1(\bar{x}(k), \bar{u}(k), k), \ldots, g_r(\bar{x}(k), \bar{u}(k), k)),$$

$$H(\bar{x}(k), \bar{u}(k), k) = (h_1(\bar{x}(k), \bar{u}(k), k), \ldots, h_s(\bar{x}(k), \bar{u}(k), k)).$$

Notice that the Equation (2.14) includes the Equation (2.6).

If a trajectory satisfies the dynamics (2.7) and constraints (2.3), then the trajectory is called to be feasible, and we define

$$D = \{\overline{W} : \overline{W} \text{ is a feasible trajectory}\}.$$

<u>Successive Approximation Algorithm:</u> [3]

(i) Arbitrarily select $\overline{W}^0 = \{[\bar{x}^0(0), \bar{u}^0(0)], \ldots, [\bar{x}^0(N-1), \bar{u}^0(N-1)], [\bar{x}^0(N)]\} \in D$ and set $i = 0, k = 0$.

(ii) **CASE I:** Solve the following nonlinear subproblem $(NLP)^i_k$

$$min \quad J\{[\bar{x}^i(0), \bar{u}^i(0)], \ldots, [\bar{x}^i(k-1), \overbrace{\bar{u}(k-1)], [\bar{x}(k), \bar{u}(k)]},$$

$$[\bar{x}^i(k+1), \bar{u}^i(k+1)], \ldots, [\bar{x}^i(N)]\}, \tag{2.15}$$

subject to

$$\bar{x}(k) = F_d(\bar{x}^i(k-1), \bar{u}(k-1), k-1),$$

$$\bar{x}^i(k+1) = F_d(\bar{x}(k), \bar{u}(k), k),$$

$$(\bar{x}^i(k-1), \bar{u}(k-1), k-1) \in S(k-1),$$

$$(\bar{x}(k), \bar{u}(k), k) \in S(k).$$

Here the superscript i means that the values of the variables are to be fixed in the ith iteration. Assuming $\hat{u}(k-1)$, $\hat{x}(k)$ and $\hat{u}(k)$ are the optimal solutions of this nonlinear subproblem, set $\bar{u}^i(k-1) = \hat{u}(k-1)$, $\bar{x}^i(k) = \hat{x}(k)$ and $k = k+1$.

CASE II: If in case I, the values of the vectors $\bar{u}(k-1), \bar{x}(k)$ and $\bar{u}(k)$ have only one possibility; i.e. these values are also to be fixed, then we can extend the subproblem $(NLP)^i_k$ for one more stage as follows

$$min \quad J\{[\bar{x}^i(0), \bar{u}^i(0)], \dots, [\bar{x}^i(k-1), \overbrace{\bar{u}(k-1)], [\bar{x}(k), \bar{u}(k)], [\bar{x}(k+1), \bar{u}(k+1)]}, \\ [\bar{x}^i(k+2), \bar{u}^i(k+2)], \dots, [\bar{x}^i(N)]\}, \tag{2.16}$$

subject to

$$\bar{x}(k) = F_d(\bar{x}^i(k-1), \bar{u}(k-1), k-1),$$

$$\bar{x}(k+1) = F_d(\bar{x}(k), \bar{u}(k), k),$$

$$\bar{x}^i(k+2) = F_d(\bar{x}(k+1), \bar{u}(k+1), k+1),$$

$$(\bar{x}^i(k-1), \bar{u}(k-1), k-1) \in S(k-1),$$

$$(\bar{x}(k), \bar{u}(k), k) \in S(k),$$

$$(\bar{x}(k+1), \bar{u}(k+1), k+1) \in S(k+1).$$

Assumming $\hat{u}(k-1)$, $\hat{x}(k)$, $\hat{u}(k)$, $\hat{x}(k+1)$ and $\hat{u}(k+1)$ are the optimal solutions of this nonlinear subproblem, set $\bar{u}^i(k-1) = \hat{u}(k-1)$, $\bar{x}^i(k) = \hat{x}(k)$ and $k = k+1$.

CASE III: If in case II, the values of the vectors $\bar{u}(k-1), \bar{x}(k), \bar{u}(k), \bar{x}(k+1)$ and $\bar{u}(k+1)$ are also to be fixed, then we can continuously extend the subproblem $(NLP)^i_k$ for one more stage until these vectors can be varied.

(iii) When $k = N+1$ in case I or $k = N$ in case II, set $\overline{W}^{i+1} = \overline{W}^i$, $i = i+1$, $k = 0$ and go to (ii); otherwise go to (ii) directly.

If J^i denotes the value of criterion function at the end of the ith iteration, the algorithm may terminate when $|J^{i+1} - J^i| < \varepsilon$, where $\varepsilon > 0$ is small enough and preselected.

When the criterion function J takes the form

$$J = h(\bar{x}(0)) + \sum_{k=0}^{N-1} V_d(\bar{x}(k), \bar{u}(k), k) + l(\bar{x}(N)), \tag{2.17}$$

we can use the following function

$$V_d(\bar{x}^i(k-1), \bar{u}(k-1), k-1) + V_d(\bar{x}(k), \bar{u}(k), k) \tag{2.18}$$

instead of the function

$$J\{[\bar{x}^i(0), \bar{u}^i(0)], \ldots, [\bar{x}^i(k-1), \overbrace{\bar{u}(k-1)], [\bar{x}(k), \bar{u}(k)}],$$

$$[\bar{x}^i(k+1), \bar{u}^i(k+1)], \ldots, [\bar{x}^i(N)]\} \tag{2.15}$$

for case I or use the following function

$$V_d(\bar{x}^i(k-2), \bar{u}(k-2), k-2) + V_d(\bar{x}(k-1), \bar{u}(k-1), k-1) + V_d(\bar{x}(k), \bar{u}(k), k) \tag{2.19}$$

instead of the function

$$J\{[\bar{x}^i(0), \bar{u}^i(0)], \ldots, [\bar{x}^i(k-2), \overbrace{\bar{u}(k-2)], [\bar{x}(k-1), \bar{u}(k-1)], [\bar{x}(k), \bar{u}(k)}],$$

$$[\bar{x}^i(k+1), \bar{u}^i(k+1)], \ldots, [\bar{x}^i(N)]\} \tag{2.16}$$

for case II, to solve the nonlinear subproblem $(NLP)_k^i$. By this technique, a trajectory sequence $\{\overline{W}^i\}$ can be obtained. Obviously, $\overline{W}^i \in D$ and

$$J(\overline{W}^{i+1}) \leq J(\overline{W}^i).$$

Here we can see that the original problem is decomposed into a group of simple subproblems and in case I each subproblem $(NLP)_k^i$ contains only one state vector $\bar{x}(k)$ and two control vectors $\bar{u}(k-1), \bar{u}(k)$; in case II each subproblem $(NLP)_k^i$ contains only two state vector $\bar{x}(k), \bar{x}(k+1)$ and three control vectors $\bar{u}(k-1), \bar{u}(k)$, $\bar{u}(k+1)$. After understanding the idea of this algorithm, it is not difficult to give a more general procedure in which a problem can be broken into a group of subproblems; each of them may contain more than two state vectors or three control vectors and the number of state or control vectors in one subproblem may differ from that in another subproblem.

2.3 Discussions on Constraint Qualification

In this section, we discuss a constraint qualification for the constraints belonging to $S(k)$. This constraint qualification is used in Section 2.4 to show the convergence and optimality of the proposed algorithm. In the following, a trajectory $\overline{W}$ is treated as a point in $R^{(n+m)(N+1)}$ when needed.

First, some symbols and definitions are given below.

For $\overline{W} \in R^{(n+m)(N+1)}$, $\overline{W}$ is denoted by

$$\begin{aligned}\overline{W} = \{\bar{w}(k)\} &= \{\bar{w}(0)^T, \bar{w}(1)^T, \ldots, \bar{w}(N)^T\}^T \\ &= \{[\bar{x}(0)^T, \bar{u}(0)^T], [\bar{x}(1)^T, \bar{u}(1)^T], \ldots, [\bar{x}(N-1)^T, \bar{u}(N-1)^T], [\bar{x}(N)^T]\}^T,\end{aligned}$$

where

$$\bar{x}(k) \in R^n, \bar{u}(k) \in R^m \text{ and } \bar{w}(k) \in R^{n+m}.$$

For $\overline{W} \in D$, let

$$I(\overline{W}) = \{(p,k) : g_p(\bar{w}(k), k) = 0\},$$

$$I(\bar{w}(k)) = \{p : g_p(\bar{w}(k), k) = 0\},$$

$$L(\overline{W}) = \{\overline{Z} : \overline{Z} = \{\bar{z}(k)\}, \bar{z}(k)^T \nabla_k g_p(\bar{w}(k), k) \geq 0, (p,k) \in I(\overline{W})\},$$

$$L(\bar{w}(k)) = \{\bar{z}(k) : \bar{z}(k)^T \nabla_k g_p(\bar{w}(k), k) \geq 0, p \in I(\bar{w}(k))\},$$

where

$$k = 0, 1, \ldots, N$$

and

$$\nabla_k = (\partial/\partial w_1(k), \partial/\partial w_2(k), \ldots, \partial/\partial w_{n+m}(k))^T.$$

The closed cone of tangents of a nonempty set A at $\bar{w} \in A$ is defined as

$$T(A, \bar{w}) = \{\bar{y} : \exists\{\bar{w}^i\} \subset A, \lim_{i\to\infty} \bar{w}^i = \bar{w}, and$$

$$a^i \in R, a^i \geq 0, \ni \lim_{i\to\infty} a^i(\bar{w}^i - \bar{w}) = \bar{y}\}.$$

The positively normal cone of set A is

$$A^* = \{\bar{y} : \bar{w} \in A, \bar{y}^T \bar{w} \geq 0\}.$$

The following theorem explains the relationship between the holding for the global problem constraint qulification and that for every subproblem constraint qulification.

Theorem 2.1 *For $\overline{W} \in D$, the constraint qualification of the global problem (NLP) holds; i.e.*

$$[L(\overline{W})]^* = [T(D, \overline{W})]^*$$

if and only if the constraint qualification of every subproblem $(NLP)_k$ holds; i.e.

$$[L(\bar{w}(k))]^* = [T(S(k), \bar{w}(k))]^*; k = 0, 1, \ldots, N.$$

The proof of this theorem is similar to that in [17]. Here we omit the proof to avoid repetition.

2.4 Convergence and Optimality Analysis

In this section, we present the theorems of the convergence and optimality of the successive approximation algorithm.

First, considering the nonlinear programming theory [7], for $\overline{W} \in D$, the holding of the Kuhn–Tucker conditions of the problem discussed here means that there exist vectors λ_p and β_q such that

$$\begin{cases} \nabla J(\overline{W}) - \sum_{k=0}^{N} \sum_{p=1}^{r_k} \lambda_p(k) \nabla g_p(\bar{w}(k)) - \sum_{k=0}^{N} \sum_{q=1}^{s_k} \beta_q(k) \nabla h_q(\bar{w}(k)) = 0, \\ \lambda_p(k) g_p(\bar{w}(k)) = 0, p = 1, \ldots, r_k, k = 0, \ldots, N, \\ \lambda_s(k) \geq 0, \end{cases}$$

where

$$\nabla = (\nabla_0, \ldots, \nabla_N)^T,$$

$$\nabla_k = (\partial/\partial w_1(k), \partial/\partial w_2(k), \ldots, \partial/\partial w_{n+m}(k))^T.$$

Similarly, if $\bar{w}(k)$ is a Kuhn–Tucker point of the subproblem

$$min \quad J\{\bar{w}^i(0), \ldots, \bar{w}^i(k-1), \bar{w}(k), \bar{w}^i(k+1), \ldots, \bar{w}^i(N)\},$$

$$\bar{w}(k) \in S(k),$$

$$k = 0, 1, \ldots, N,$$

it holds that there exist λ_p and β_q such that

$$\begin{cases} \nabla_k (J - \sum_{p=1}^{r_k} \lambda_p(k) g_p(\bar{w}(k), k) - \sum_{q=1}^{s_k} \beta_q(k) h_q(\bar{w}(k), k)) = 0, \\ \lambda_p(k) g_p(\bar{w}(k), k) = 0, p = 1, \ldots, s_k, \\ \lambda_p(k) \geq 0, \end{cases}$$

$$k = 0, 1, \ldots, N.$$

The following theorem indicates the relationship between a global Kuhn–Tucker point and those of subproblems $(NLP)_k$.

Theorem 2.2 *Suppose that J, g_p, h_q are continuously differentiable and, for $\overline{W} \in D$, the constraint qualification of the global problem holds; i.e. $[L(\overline{W})]^* = [T(D, \overline{W})]^*$. Then, the $\overline{W}$ is a Kuhn–Tucker point of the global problem if and only if the $\bar{w}$ is a Kuhn–Tucker point for every subproblem $(NLP)_k; k = 0, 1, \ldots, N$.*

Once again, the proof of this theorem is similar to that in [3]. We omit the proof to avoid repetition.

According to the Theorem 2.1 and 2.2, we can easily prove the Theorems 2.3 -2.5 by using the method similar to that in [17]. These theorems are described as follows:

Theorem 2.3 *Suppose that the following conditions are satisfied*

(i) *The solution to every subproblem $(NLP)_k$ is unique;*

(ii) *For $\overline{W} \in D$ and any real number α, set $\{\overline{W} : J(\overline{W}) \leq \alpha\}$ is bounded;*

(iii) *J, g_p and h_q are continuously differentiable over D.*

Then, for the sequence $\{\overline{W}^i\}$ produced by the algorithm, the following conclusions hold.

(a) *$J(\overline{W}^{i+1}) \leq J(\overline{W}^i)$ and $\overline{W}^i$ is feasible;*

(b) *The accumulation point set, denoted by P, of $\{\overline{W}^i\}$ is not empty;*

(c) *If at $\overline{W} \in P$, the constraint qualification for the Kuhn–Tucker conditions holds, then $\overline{W}$ is a Kuhn–Tucker point of the problem (NLP).*

Theorem 2.4 *Suppose that the assumptions in* Theorem 2.3 *hold. If, in addition, every g_p is quasiconcave, every h_q is both quasiconcave and quasiconvex, and J is pseudoconvex in $\overline{W}$, then, for any $\overline{W} \in P, \widehat{W}$ is an optimum of the problem defined by* (2.6),(2.7),(2.3) *and $\{J(\overline{W}^i)\}$ converges to $J(\widehat{W})$.*

Theorem 2.5 *Suppose that the constraint qualification and the conditions* (ii) *and* (iii) *in* Theorem 2.3 *hold. If, in addition, every g_p is quasiconcave and every h_q is both quasiconcave and quasiconvex, J is strictly pseudoconvex, then, the sequence $\{\overline{W}^i\}$ converges to the unique optimum of the problem defined by* (2.6),(2.7),(2.3).

From Theorem 2.3, we know that the proposed algorithm is convergent under certain conditions. Besides, according to the Theorem 2.4 and 2.5, generalized convexity will ganranutee that an optimum can be obtained by the algorithm.

CHAPTER 3

IMPLEMENTATION ON AN OPTIMAL CONTROL PROBLEM

3.1 Description of the Problem

In this chapter we will take an optimal control problem to illustrate the successive approximation algorithm. This problem is a simplified optimal control problem. Consider a car which is driven in a straight line from point O at time 0 to point E at time T. The distance between the two points is 1000 meters. The maximum speed of the car is 20 meters/second and the maximum acceleration of the car is ± 1 meter/second^2. This problem is shown in Figure 3.1.

In this problem we will consider three different situations — the minimum time problem, the minimum quadratic acceleration problem and the weighted time and quadratic acceleration minimization problem. Each situation can explain some special features for the successive approximation algorithm. Certainly, this algorithm can also solve the problem with a more general performance index such as non-separable function.

According to the Section 1.2, we can formulate this problem as follows: [4,8] The performance measures

(1) for minimum time problem:

$$J = \int_0^T dt, \tag{3.1}$$

(2) for minimum quadratic acceleration problem:

$$J = \int_0^T a^2(t)dt, \tag{3.2}$$

(3) for weighted time and quadratic acceleration minimization problem:

$$J = \int_0^T [\lambda + a^2(t)]dt, \tag{3.3}$$

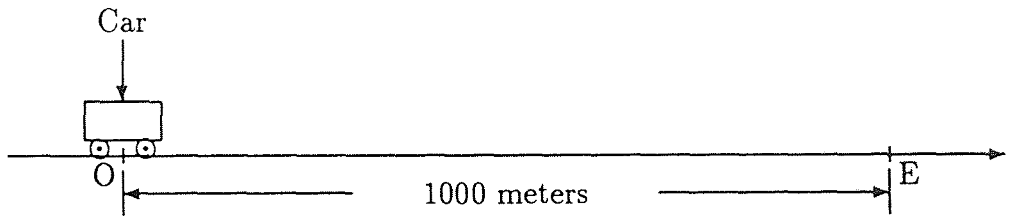

Figure 3.1 An Optimal Control Problem.

subject to the dynamics

$$\dot{x}(t) = v(t), \tag{3.4}$$

$$\dot{v}(t) = a(t), \tag{3.5}$$

and the constraints

$$0 \leq v(t) \leq 20, \tag{3.6}$$

$$-1 \leq a(t) \leq 1, \tag{3.7}$$

where

$$0 \leq t \leq T$$

and

x : the position of the car,

v : the velocity of the car,

a : the acceleration of the car.

λ : any positive real number,

In addition, the boundary conditions for this problem are

$$x(0) = 0, v(0) = 0, \tag{3.8}$$

$$x(T) = 1000, v(T) = 0, \tag{3.9}$$

which will be included in the constraints.

In order to apply the numeric methods to solving this problem, we must convert this system into a discrete system (refer to the Section 2.1 for detail). Then the discrete optimal control problem can be formulated as follows:
The performance measures

(1) for minimum time problem:

$$J = \sum_{k=0}^{N-1} \delta t(k), \tag{3.10}$$

(2) for minimum quadratic acceleration problem:

$$J = \sum_{k=0}^{N-1} a^2(k)\delta t(k), \tag{3.11}$$

(3) for weighted time and quadratic acceleration minimization problem:

$$J = \sum_{k=0}^{N-1} [\lambda + a^2(k)]\delta t(k), \tag{3.12}$$

subject to the dynamics

$$x(k+1) = x(k) + v(k) \times \delta t(k); \tag{3.13}$$

$$v(k+1) = v(k) + a(k) \times \delta t(k); \tag{3.14}$$

$$k = 0, 1, \ldots, N-1$$

and the constraints

$$0 \leq v(k) \leq 20; k = 0, 1, \ldots, N, \tag{3.15}$$

$$-1 \leq a(k) \leq 1; k = 0, 1, \ldots, N-1, \tag{3.16}$$

$$x(0) = 0, v(0) = 0, \tag{3.17}$$

$$x(N) = 1000, v(N) = 0. \tag{3.18}$$

For the variable-time problems, the time interval $\delta t(k)$ will be considered as a control variable. Then the control vector for this problem should include the acceleration and the time interval and the state vector for this problem should include the position and the velocity. These vectors can be expressed as follows:

$$\bar{u}(k) = \begin{bmatrix} a(k) \\ \delta t(k) \end{bmatrix} = \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}; k = 0, 1, \ldots, N-1; \tag{3.19}$$

$$\bar{x}(k) = \begin{bmatrix} x(k) \\ v(k) \end{bmatrix} = \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}; k = 0, 1, \ldots, N. \tag{3.20}$$

In the next three sections, we will solve the three different situations by using the successive approximation algorithm respectively.

3.2 The Minimum Time Problem

3.2.1 Formulation of the Problem

According to the last section, the minimum time problem can be formulated as follows: The performance measure

$$J = \sum_{k=0}^{N-1} u_2(k), \tag{3.21}$$

subject to the dynamics

$$x_1(k+1) = x_1(k) + x_2(k) \times u_2(k); \tag{3.22}$$

$$x_2(k+1) = x_2(k) + u_1(k) \times u_2(k); \tag{3.23}$$

$$k = 0, 1, \ldots, N-1$$

and the constraints

$$0 \le x_2(k) \le 20; k = 0, 1, \ldots, N, \tag{3.24}$$

$$-1 \le u_1(k) \le 1; k = 0, 1, \ldots, N-1, \tag{3.25}$$

$$x_1(0) = 0, x_2(0) = 0, \tag{3.26}$$

$$x_1(N) = 1000, x_2(N) = 0. \tag{3.27}$$

Notice that the performance measure in this problem matchs with the Equation (2.17). Therefore, we can use the Equation (2.18) instead of (2.15) to solve this problem. In addition, we will solve three different stage numbers ($N = 6, N = 11$

and $N = 21$) to compare their results.

3.2.2 The Procedure for Solving the Minimum Time Problem

In this subsection, we will show how to apply the successive approximation algorithm to solving this problem.

First consider the stage number N is equal to 6.

Step 1: Arbitrarily select an initial feasible sequence

$$\begin{aligned}
\overline{W}^0 &= \{[\bar{x}^0(0), \bar{u}^0(0)], \ldots, [\bar{x}^0(5), \bar{u}^0(5)], [\bar{x}^0(6)]\} \\
&= \left\{\left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0.5 \\ 20 \end{pmatrix}\right], \left[\begin{pmatrix} 0 \\ 10 \end{pmatrix}, \begin{pmatrix} 0 \\ 20 \end{pmatrix}\right], \left[\begin{pmatrix} 200 \\ 10 \end{pmatrix}, \begin{pmatrix} 0 \\ 20 \end{pmatrix}\right],\right. \\
&\quad \left[\begin{pmatrix} 400 \\ 10 \end{pmatrix}, \begin{pmatrix} 0 \\ 20 \end{pmatrix}\right], \left[\begin{pmatrix} 600 \\ 10 \end{pmatrix}, \begin{pmatrix} 0 \\ 20 \end{pmatrix}\right], \left[\begin{pmatrix} 800 \\ 10 \end{pmatrix}, \begin{pmatrix} -0.5 \\ 20 \end{pmatrix}\right], \\
&\quad \left.\left[\begin{pmatrix} 1000 \\ 0 \end{pmatrix}\right]\right\} \in D
\end{aligned}$$

and set $i = 0, k = 0$.

Step 2: Solve the following nonlinear subproblem for the zeroth iteration $(NLP)_0^0$:

$$min \quad J^0\{\overbrace{[\bar{x}(0), \bar{u}(0)]}, [\bar{x}^0(1), \bar{u}^0(1)], \ldots, [\bar{x}^0(6)]\}, \tag{3.28}$$

where we use the Equation (2.18) instead of (3.28); i.e.

$$min \quad J^0 = u_2(0),$$

subject to

$$x_1^0(1) = x_1(0) + x_2(0) \times u_2(0),$$

$$x_2^0(1) = x_2(0) + u_1(0) \times u_2(0),$$

$$0 \leq x_2(0) \leq 20,$$

$$-1 \leq u_1(0) \leq 1,$$

$$x_1(0) = 0, x_2(0) = 0,$$

where the superscript 0 means that in the zeroth iteration, the values of these variables are to be fixed and given. Assuming $\hat{x}(0)$ and $\hat{u}(0)$ are the optimal solutions of this nonlinear subproblem, set $\bar{x}^0(0) = \hat{x}(0)$ and $k = 1$.

Step 3: Solve the following nonlinear subproblem for the zeroth iteration $(NLP)_1^0$:

$$min \quad J^0\{[\bar{x}^0(0), \overbrace{\bar{u}(0)], [\bar{x}(1), \bar{u}(1)]}, [\bar{x}^0(2), \bar{u}^0(2)], \ldots, [\bar{x}^0(6)]\}$$

or

$$min \quad J^0 = u_2(0) + u_2(1),$$

subject to

$$x_1(1) = x_1^0(0) + x_2^0(0) \times u_2(0),$$

$$x_2(1) = x_2^0(0) + u_1(0) \times u_2(0),$$

$$x_1^0(2) = x_1(1) + x_2(1) \times u_2(1),$$

$$x_2^0(2) = x_2(1) + u_1(1) \times u_2(1),$$

$$-1 \le u_1(0) \le 1,$$

$$0 \le x_2(1) \le 20,$$

$$-1 \le u_1(1) \le 1.$$

Assuming $\hat{u}(0), \hat{x}(1)$ and $\hat{u}(1)$ are the optimal solutions of this nonlinear subproblem, set $\bar{u}^0(0) = \hat{u}(0), \bar{x}^0(1) = \hat{x}(1)$ and $k = 2$.

$$\vdots$$

Continue this procedure by increasing $k = k + 1$ for each step.

Step 8: Solve the following nonlinear subproblem for the zeroth iteration $(NLP)_6^0$:

$$min \quad J^0\{[\bar{x}^0(0), \bar{u}^0(0)], \ldots, [\bar{x}^0(5), \overbrace{\bar{u}(5)], [\bar{x}(6)]}\}$$

or

$$min \;\; J^0 = u_2(5),$$

subject to

$$x_1(6) = x_1^0(5) + x_2^0(5) \times u_2(5),$$

$$x_2(6) = x_2^0(5) + u_1(5) \times u_2(5),$$

$$-1 \leq u_1(5) \leq 1,$$

$$0 \leq x_2(6) \leq 20,$$

$$x_1(6) = 0, x_2(6) = 0.$$

Assuming $\hat{u}(5)$ and $\hat{x}(6)$ are the optimal solutions of this nonlinear subproblem, set $\bar{u}^0(5) = \hat{u}(5), \bar{x}^0(6) = \hat{x}(6)$ and $i = 1, k = 0$; go to the next iteration.

Step 9: Solve the following nonlinear subproblem for the first iteration $(NLP)_0^1$:

$$min \;\; J^1\{\overbrace{[\bar{x}(0), \bar{u}(0)]}, [\bar{x}^1(1), \bar{u}^1(1)], \ldots, [\bar{x}^1(6)]\}$$

or

$$min \;\; J^1 = u_2(0),$$

subject to

$$x_1^1(1) = x_1(0) + x_2(0) \times u_2(0),$$

$$x_2^1(1) = x_2(0) + u_1(0) \times u_2(0),$$

$$0 \leq x_2(0) \leq 20,$$

$$-1 \leq u_1(0) \leq 1,$$

$$x_1(0) = 0, x_2(0) = 0.$$

Assuming $\hat{x}(0)$ and $\hat{u}(0)$ are the optimal solutions of this nonlinear subproblem, set $\bar{x}^1(0) = \hat{x}(0)$ and $k = 1$.

$$\vdots$$

Step 15: Solve the following nonlinear subproblem for the first iteration $(NLP)_6^1$:

$$min \quad J^1\{[\bar{x}^1(0), \bar{u}^1(0)], \ldots, [\bar{x}^1(5), \overbrace{\bar{u}(5)], [\bar{x}(6)]}\}$$

or

$$min \quad J^1 = u_2(5),$$

subject to

$$x_1(6) = x_1^1(5) + x_2^1(5) \times u_2(5),$$

$$x_2(6) = x_2^1(5) + u_1(5) \times u_2(5),$$

$$-1 \leq u_1(5) \leq 1,$$

$$0 \leq x_2(6) \leq 20,$$

$$x_1(6) = 0, x_2(6) = 0.$$

Assuming $\hat{u}(5)$ and $\hat{x}(6)$ are the optimal solutions of this nonlinear subproblem, set $\bar{u}^1(5) = \hat{u}(5), \bar{x}^1(6) = \hat{x}(6)$ and $i = 2, k = 0$; go to the next iteration.

When we accomplished each iteration, check $|J^{i+1} - J^i|$ is less than ε or not, where $\varepsilon > 0$ is small enough and preselected. If $|J^{i+1} - J^i| < \varepsilon$, then the algorithm may be terminated and get the optimum sequence $\{\overline{W}^{i+1}\}$; otherwise, continue this procedure until $|J^{i+1} - J^i| < \varepsilon$.

According to the Theorem 2.3–2.5, we know that the optimum sequence $\{\overline{W}^{i+1}\}$ for this problem is feasible and converges to the unique optimum sequence.

For the stage number $N = 11$, we select the initial feasible sequence as

$$\begin{aligned}\overline{W}^0 &= \{[\bar{x}^0(0), \bar{u}^0(0)], \ldots, [\bar{x}^0(10), \bar{u}^0(10)], [\bar{x}^0(11)]\} \\ &= \left\{\left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 10 \end{pmatrix}\right], \left[\begin{pmatrix} 0 \\ 10 \end{pmatrix}, \begin{pmatrix} 0 \\ 10 \end{pmatrix}\right], \left[\begin{pmatrix} 100 \\ 10 \end{pmatrix}, \begin{pmatrix} 0 \\ 10 \end{pmatrix}\right], \ldots, \right. \\ &\quad \left. \left[\begin{pmatrix} 800 \\ 10 \end{pmatrix}, \begin{pmatrix} 0 \\ 10 \end{pmatrix}\right], \left[\begin{pmatrix} 900 \\ 10 \end{pmatrix}, \begin{pmatrix} -1 \\ 10 \end{pmatrix}\right], \left[\begin{pmatrix} 1000 \\ 0 \end{pmatrix}\right]\right\} \in D\end{aligned}$$

For the stage number $N = 21$, we select the initial feasible sequence as

$$\begin{aligned}\overline{W}^0 &= \{[\bar{x}^0(0), \bar{u}^0(0)], \ldots, [\bar{x}^0(20), \bar{u}^0(20)], [\bar{x}^0(21)]\} \\ &= \left\{\left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0.5 \\ 10 \end{pmatrix}\right], \left[\begin{pmatrix} 0 \\ 5 \end{pmatrix}, \begin{pmatrix} 0 \\ 10 \end{pmatrix}\right], \left[\begin{pmatrix} 50 \\ 5 \end{pmatrix}, \begin{pmatrix} 0 \\ 10 \end{pmatrix}\right], \ldots, \right. \\ &\quad \left. \left[\begin{pmatrix} 900 \\ 5 \end{pmatrix}, \begin{pmatrix} 0 \\ 10 \end{pmatrix}\right], \left[\begin{pmatrix} 950 \\ 5 \end{pmatrix}, \begin{pmatrix} -0.5 \\ 10 \end{pmatrix}\right], \left[\begin{pmatrix} 1000 \\ 0 \end{pmatrix}\right]\right\} \in D\end{aligned}$$

A single precision Fortran 77 program which applies the successive approximation algorithm to solving the minimum time problem for $N = 6$ is shown in Appendix A. The program is executed on the VAX6430 computer running the VMS 5.3 operating system at New Jersey Institute of Technology. The subroutine NCONF in the IMSL's Math/Library [18] is selected to solve each nonlinear subproblem in the algorithm. NCONF uses a successive quadratic programming method to solve the general nonlinear programming problems. Certainly, other existing subroutines which can solve nonlinear programming problems can also be selected to solve each nonlinear subproblem in the algorithm.

3.2.3 Computational Results

The results for the minimum time problem are almost perfect. This problem is a bang-coast-bang control problem. According to the results, the optimal control policy for the minimum time problem is that the car applies the maximum acceleration 1 m/sec^2 until it attaines the maximum velocity 20 m/sec at the beginning and remaines this velocity to save the elapsed time until it applies the maximum deceleration -1 m/sec^2 and stops at the point E.

The continuous time solution for this problem is list as follows: [4]

$$x_1(t) = \begin{cases} 0.5t^2, & \text{for } 0 \le t \le 20; \\ 20t - 200, & \text{for } 20 \le t \le 50; \\ -0.5t^2 + 70t - 1450, & \text{for } 50 \le t \le 70, \end{cases} \tag{3.29}$$

$$x_2(t) = \begin{cases} t, & \text{for } 0 \le t \le 20; \\ 20, & \text{for } 20 \le t \le 50; \\ -t + 70, & \text{for } 50 \le t \le 70, \end{cases} \tag{3.30}$$

$$u_1(t) = \begin{cases} 1, & \text{for } 0 \le t \le 20; \\ 0, & \text{for } 20 \le t \le 50; \\ -1, & \text{for } 50 \le t \le 70. \end{cases} \tag{3.31}$$

Tables 3.1, 3.2 and 3.3 show the results of the minimum time problem for the stage numbers $N = 6, N = 11$ and $N = 21$, respectively. Figures 3.2, 3.3 and 3.4 show the optimum positions, the optimum velocities and the optimum accelerations for the minimum time problem, respectively.

Notice that for the variable-time problems, the algorithm tends to "stretch" some time intervals and "compress" others. For example, in the Table 3.1 the time interval of the fourth stage is compressed to 0. This makes the fourth stage do not exist practically. This effect is very similar to that of the nonlinear programming. [1]

In addition, there are some errors in the fourth stage and sixteenth stage of the minimum time problem for $N = 21$. The accelerations of the fourth stage and sixteenth stage should be 1 and -1 respectively but they have only 0.543 and -0.646 respectively. These errors are caused by the limited accuracy for solving each nonlinear subproblem. If we increase the accuracy of each nonlinear subproblem, then we can obtain the more accurate results for the global problem.

Table 3.1 Results of the Minimum Time Problem for $N = 6$.

Stage No.	Time Interval (seconds)	Position (meters)	Velocity (m/sec)	Acceleration (m/sec^2)
0	14.143	0	0	1
1	5.857	0	14.143	1
2	15.845	82.834	20	0
3	10	399.730	20	0
4	0	− − −	− − −	− − −
5	20.014	599.730	20	−0.999
6	− − −	1000	0	− − −

The total travel time is 65.858 seconds.

Analytic optimal $J = 70$.

Approximative optimal $J = 65.858$.

Table 3.2 Results of the Minimum Time Problem for $N = 11$.

Stage No.	Time Interval (seconds)	Position (meters)	Velocity (m/sec)	Acceleration (m/sec^2)
0	10	0	0	1
1	7.320	0	10	1
2	2.261	73.205	17.320	1
3	0.418	112.372	19.582	1
4	8.972	120.561	20	0
5	5	299.998	20	0
6	5	399.989	20	0
7	5	499.989	20	0
8	5	599.989	20	0
9	10.001	699.989	20	−1
10	10	900	10	−1
11	— — —	1000	0	— — —

The total travel time is 68.972 seconds.

Analytic optimal $J = 70$.

Approximative optimal $J = 68.972$.

Table 3.3 Results of the Minimum Time Problem for $N = 21$.

Stage No.	Time Interval (seconds)	Position (meters)	Velocity (m/sec)	Acceleration (m/sec^2)
0	7.071	0	0	1
1	4.184	0	7.071	1
2	1.644	29.589	11.256	1
3	3.550	48.095	12.900	1
4	6.532	93.894	16.450	0.543
5	0.883	201.349	20	0.002
6	1.546	219.013	20	0
7	5.169	249.927	20	0
8	0.781	353.299	20	0.001
9	1.562	368.918	20	0
10	5.157	400.160	20	0
11	0.781	503.298	20	0.001
12	1.562	518.919	20	0
13	5.157	550.159	20	0
14	0.781	653.298	20	0.001
15	1.562	668.919	20	0
16	5.100	700.159	20	−0.646
17	1.040	802.156	16.703	−1
18	0.507	819.529	15.663	−1
19	8.079	827.468	15.156	−1
20	7.077	949.917	7.077	−1
21	− − −	1000	0	− − −

The total travel time is 69.726 seconds.

Analytic optimal $J = 70$.

Approximative optimal $J = 69.726$.

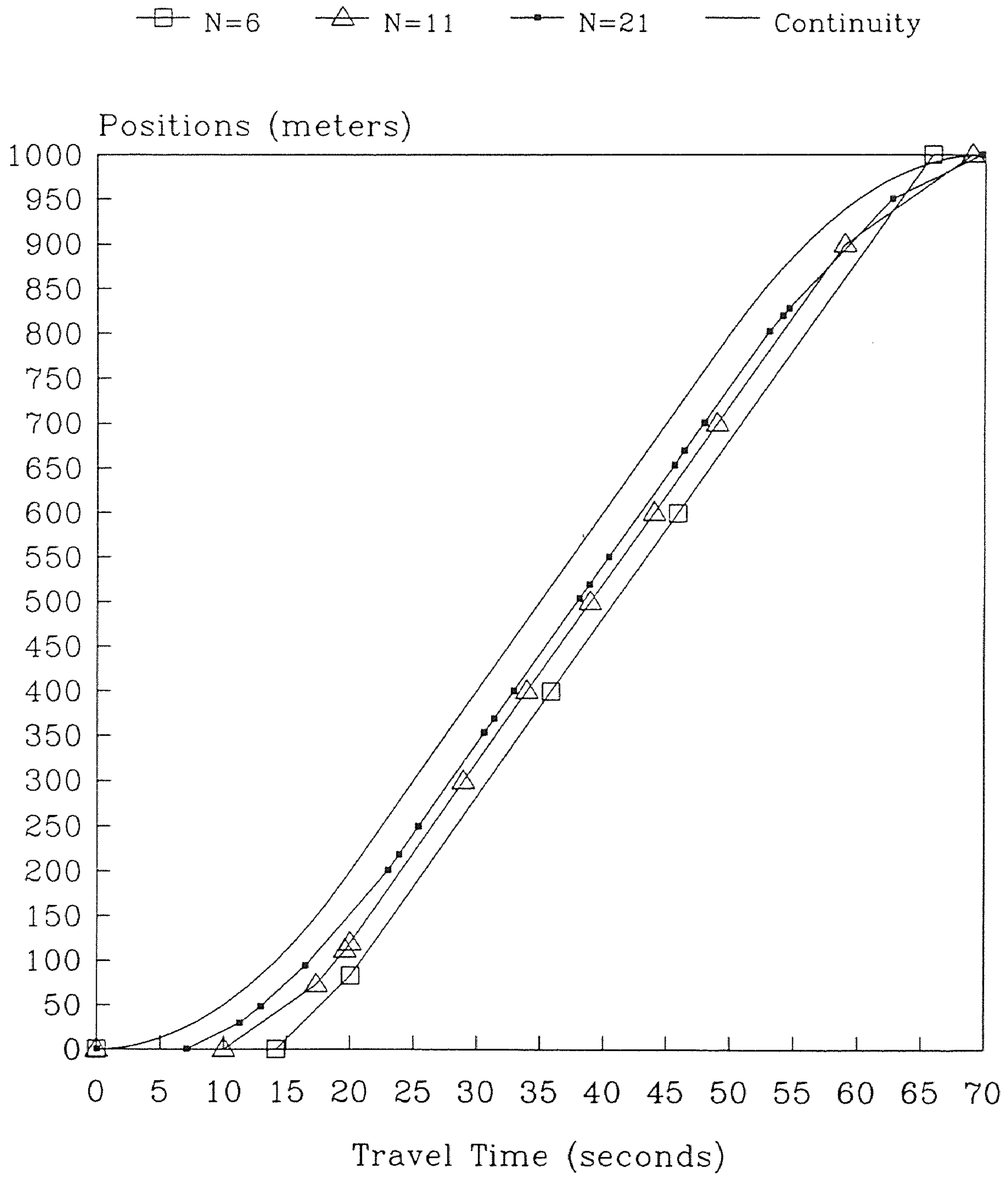

Figure 3.2 Optimum Positions for the Minimum Time Problem.

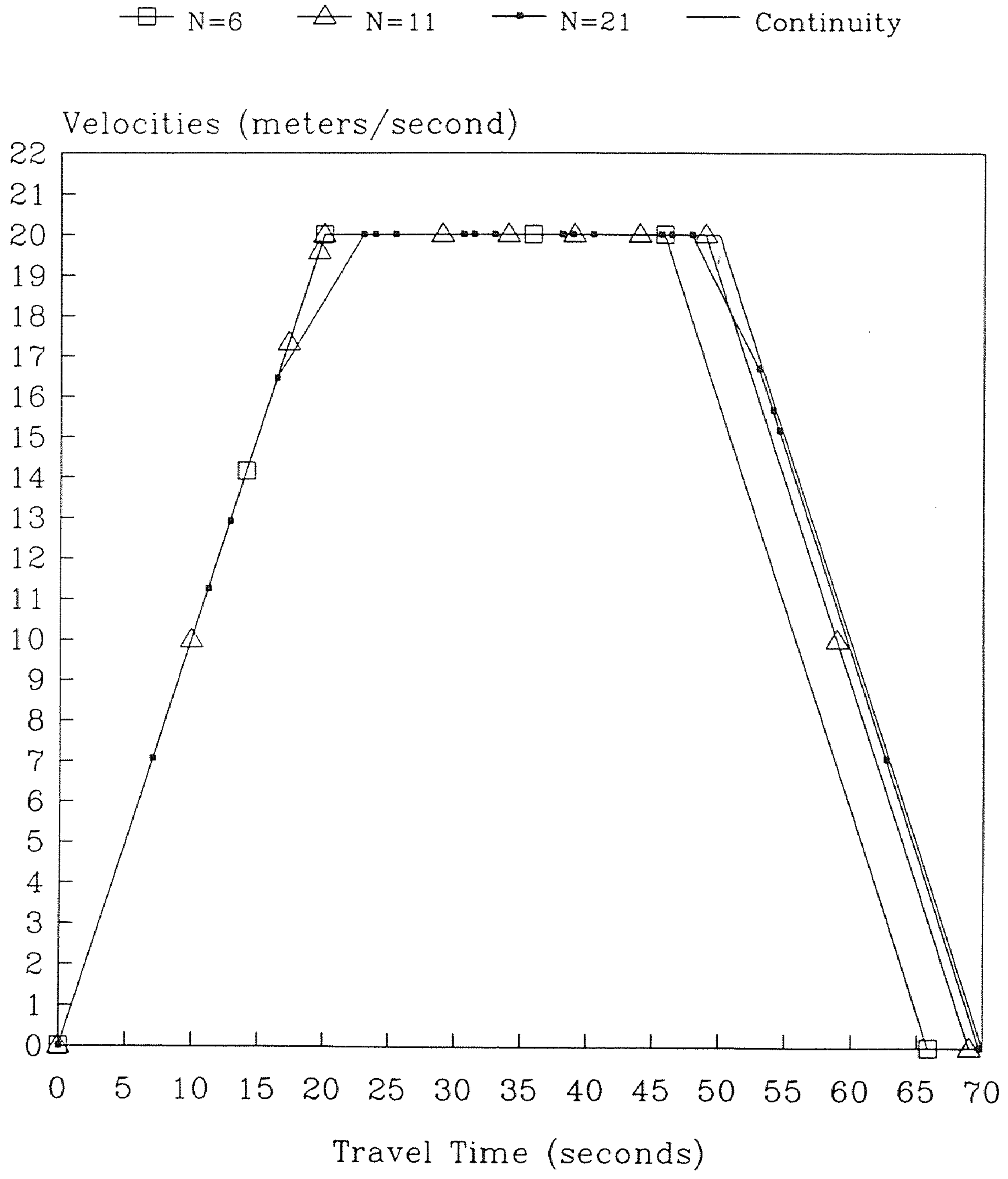

Figure 3.3 Optimum Velocities for the Minimum Time Problem.

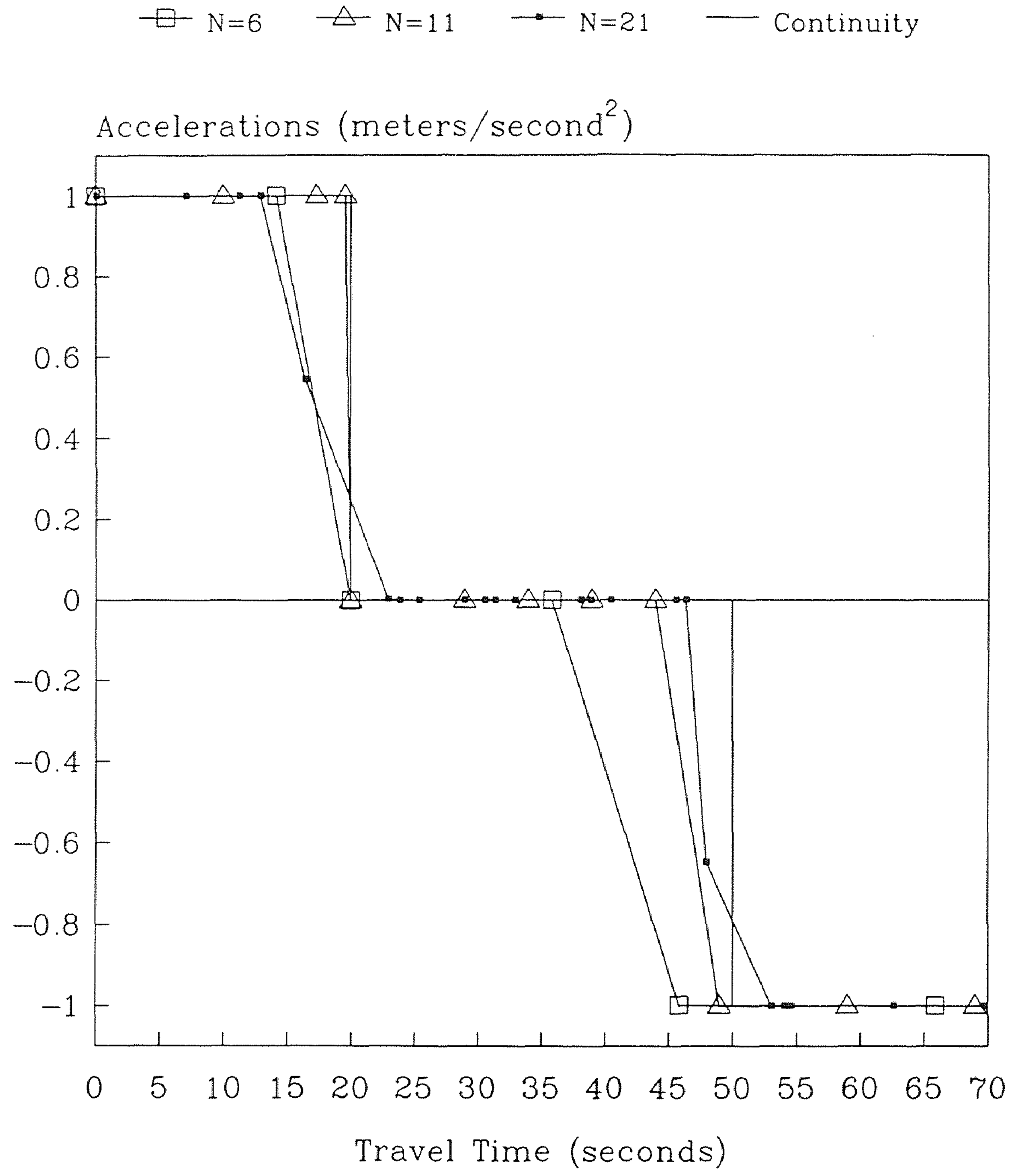

Figure 3.4 Optimum Accelerations for the Minimum Time Problem.

3.3 The Minimum Quadratic Acceleration Problem

3.3.1 Formulation of the Problem

The minimum quadratic acceleration problem is not a variable-time problem where the total travel time T is to be fixed and the time interval of each stage should also be fixed to T/N seconds. Here we arbitrarily select the final time $T = 100$ seconds and the time interval of each stage should be fixed to $100/N$ seconds. According to the Section 3.1, the minimum quadratic acceleration problem can be formulated as follows:

The performance measure

$$J = \sum_{k=0}^{N-1} u_1^2(k), \tag{3.32}$$

subject to the dynamics

$$x_1(k+1) = x_1(k) + x_2(k) \times 100/N; \tag{3.33}$$

$$x_2(k+1) = x_2(k) + u_1(k) \times 100/N; \tag{3.34}$$

$$k = 0, 1, \ldots, N-1$$

and the constraints

$$0 \leq x_2(k) \leq 20; k = 0, 1, \ldots, N,$$

$$-1 \leq u_1(k) \leq 1; k = 0, 1, \ldots, N-1,$$

$$x_1(0) = 0, x_2(0) = 0,$$

$$x_1(N) = 1000, x_2(N) = 0.$$

Notice that the performance measure in this problem is also matchable to the Equation (2.17). However, we will use the Equation (2.19) instead of (2.16) to solve this problem. In addition, the time interval of each stage is fixed to $100/N$ seconds. Therefore, the control vector should exclude the time interval $u_2(k)$ and include the acceleration $u_1(k)$ only.

In this section, we will also solve three different stage numbers ($N = 6, N = 11$ and $N = 21$) for comparison.

3.3.2 Special Remark for the Problem

For this problem, we cannot use Case I in the successive approximation algorithm to solve each nonlinear subproblem. Because when we use Case I to solve the nonlinear subproblems, we will quickly find that the values of the vectors $\bar{u}(k-1), \bar{x}(k)$ and $\bar{u}(k)$ are also to be fixed; that means their values have only one possibility. Then we must use Case II instead of Case I to solve this problem.

For example, consider the following nonlinear subproblem $(NLP)_1^0$ for $N = 6$:

$$min \ \ J^0 = u_1^2(0) + u_1^2(1),$$

subject to

$$x_1(1) = x_1^0(0) + x_2^0(0) \times 100/6,$$

$$x_2(1) = x_2^0(0) + u_1(0) \times 100/6,$$

$$x_1^0(2) = x_1(1) + x_2(1) \times 100/6,$$

$$x_2^0(2) = x_2(1) + u_1(1) \times 100/6,$$

$$-1 \leq u_1(0) \leq 1,$$

$$0 \leq x_2(1) \leq 20,$$

$$-1 \leq u_1(1) \leq 1.$$

where the values of the variables $x_1^0(0), x_2^0(0), x_1^0(2)$ and $x_2^0(2)$ are to be fixed and given by:

$$x_1^0(0) = 0, x_2^0(0) = 0,$$

$$x_1^0(2) = \frac{400}{3}, x_2^0(2) = \frac{44}{3}.$$

Then we will quickly find that the values of the other variables $u_1(0), x_1(1), x_2(1)$ and $u_1(1)$ are also to be fixed; that means the values of these variables have only one possibility; i.e.

$$u_1(0) = 0.48, u_1(1) = 0.4,$$

$$x_1(1) = 0, x_2(1) = 8.$$

Therefore, we cannot use Case I to find the optimal solution for this subproblem and we must use Case II instead of Case I to solve this problem.

3.3.3 Computational Results

The results for the minimum quadratic acceleration problem present the continuous time functions. In order to minimize the quadratic acceleration and accomplish this travel in the fixed time, the car applies the acceleration 0.6 m/sec^2 at the beginning and reduces the acceleration to -0.6 m/sec^2 linearly until it stops at the point E.

The continuous time solution for this problem is list as follows: [4]

$$x_1(t) = -0.002t^3 + 0.3t^2, \tag{3.35}$$

$$x_2(t) = -0.006t^2 + 0.6t, \tag{3.36}$$

$$u_1(t) = -0.012t + 0.6, \tag{3.37}$$

where

$$0 \leq t \leq 100.$$

Tables 3.4, 3.5 and 3.6 show the results of the minimum quadratic acceleration problem for the stage numbers $N = 6, N = 11$ and $N = 21$, respectively. Figures 3.5, 3.6 and 3.7 show the optimum positions, the optimum velocities and the optimum accelerations for the minimum quadratic acceleration problem, respectively.

Table 3.4 Results of the Minimum Quadratic Acceleration Problem for $N = 6$.

Stage No.	Position (meters)	Velocity (m/sec)	Acceleration (m/sec^2)
0	0	0	0.514
1	0	8.570	0.311
2	142.833	13.756	0.108
3	372.096	15.557	−0.110
4	631.386	13.726	−0.320
5	860.151	8.391	−0.503
6	1000	0	− − −

The time interval for each stage is 16.667 seconds.

Analytic optimal $J = 12$.

Approximative optimal $J = 12.348$.

Table 3.5 Results of the Minimum Quadratic Acceleration Problem for $N = 11$.

Stage No.	Position (meters)	Velocity (m/sec)	Acceleration (m/sec^2)
0	0	0	0.596
1	0	5.421	0.430
2	49.284	9.328	0.286
3	134.085	11.930	0.169
4	242.538	13.467	0.091
5	364.962	14.290	0.012
6	494.874	14.401	−0.068
7	625.789	13.781	−0.166
8	751.074	12.269	−0.295
9	862.614	9.584	−0.446
10	949.746	5.528	−0.608
11	1000	0	− − −

The time interval for each stage is 9.091 seconds.

Analytic optimal $J = 12$.

Approximative optimal $J = 12.250$.

Table 3.6 Results of the Minimum Quadratic Acceleration Problem for $N = 21$.

Stage No.	Position (meters)	Velocity (m/sec)	Acceleration (m/sec^2)
0	0	0	0.762
1	0	3.631	0.604
2	17.289	6.506	0.464
3	48.271	8.713	0.341
4	89.764	10.339	0.238
5	138.997	11.474	0.154
6	193.633	12.209	0.091
7	251.771	12.644	0.046
8	311.980	12.862	0.025
9	373.226	12.979	0.004
10	435.031	12.996	0.006
11	496.916	13.026	0.012
12	558.942	13.084	−0.002
13	621.248	13.074	−0.037
14	683.505	12.899	−0.090
15	744.929	12.469	−0.162
16	804.307	11.696	−0.250
17	860	10.507	−0.363
18	910.032	8.779	−0.476
19	951.835	6.511	−0.610
20	982.838	3.604	−0.757
21	1000	0	− − −

The time interval for each stage is 4.762 seconds.

Analytic optimal $J = 12$.

Approximative optimal $J = 13.197$.

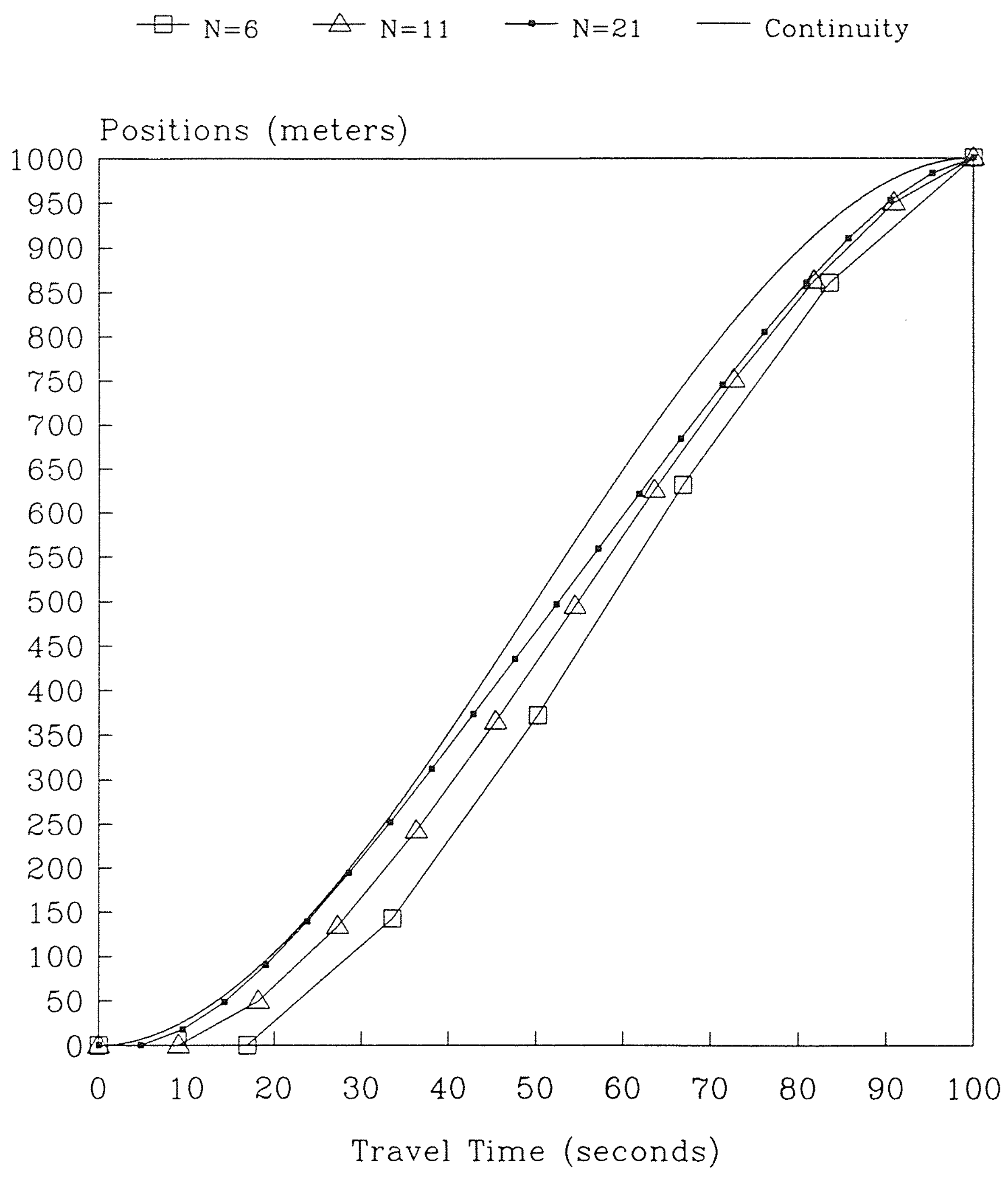

Figure 3.5 Optimum Positions for the Minimum Quadratic Acceleration Problem.

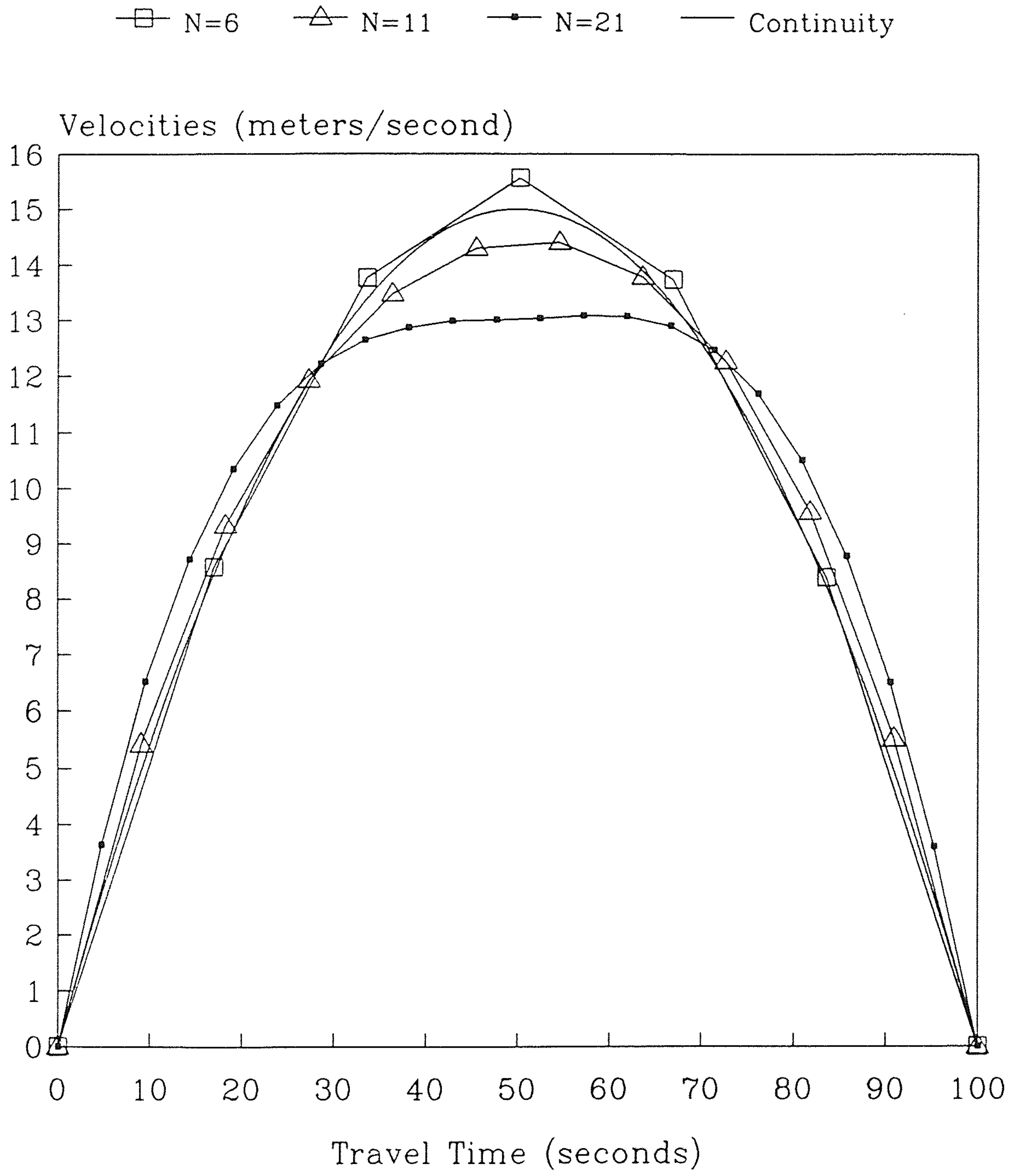

Figure 3.6 Optimum Velocities for the Minimum Quadratic Acceleration Problem.

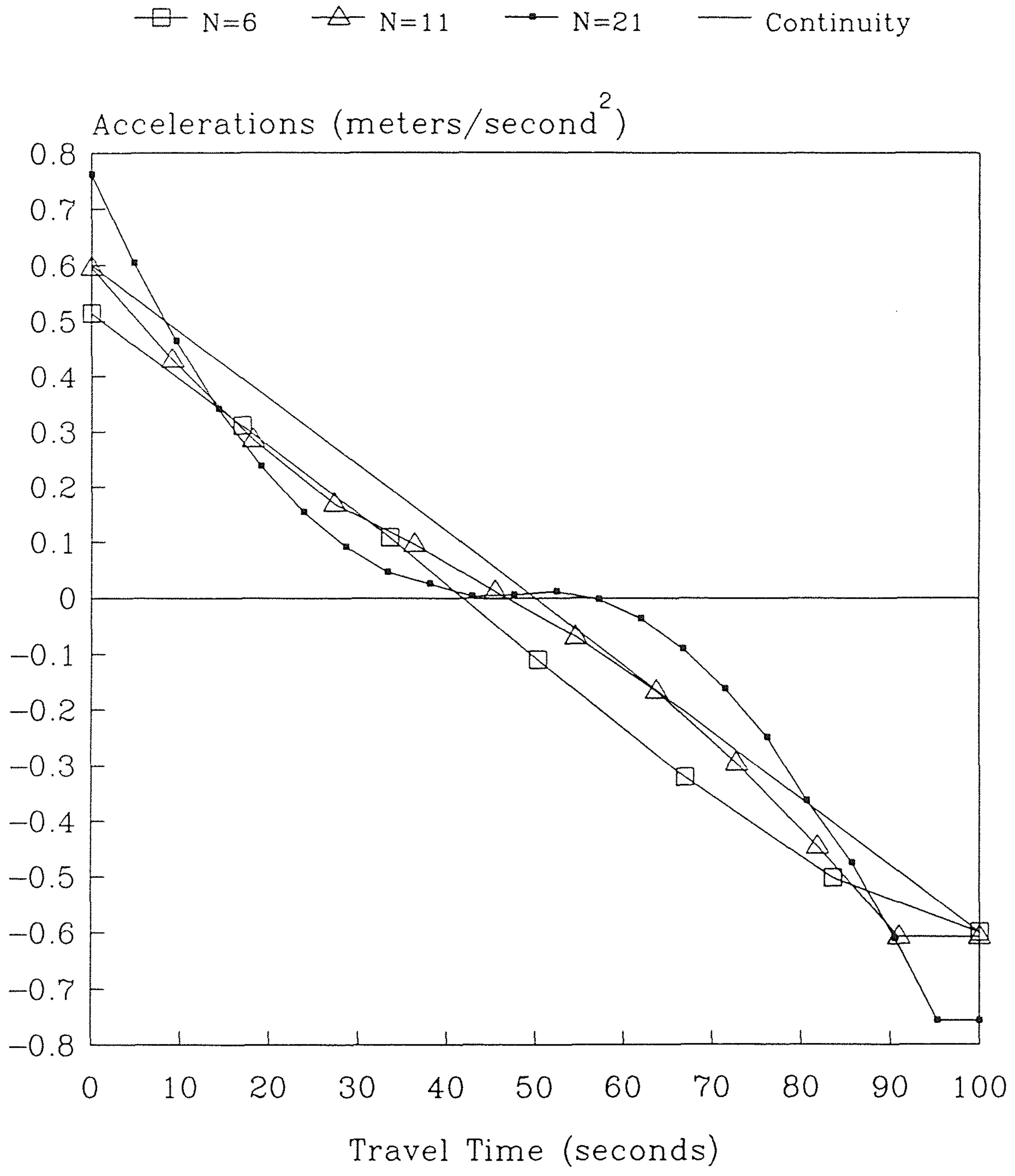

Figure 3.7 Optimum Accelerations for the Minimum Quadratic Acceleration Problem.

3.4 The Weighted Time and Quadratic Acceleration Minimization Problem

According to the minimum time problem and minimum quadratic acceleration problem, we can conjecture that the quadratic acceleration is almost inversely proportional to the elapsed time. One technique for handling this situaion is to include both elapsed time and quadratic acceleration in the performance measure which has been shown in Equation (3.3). For $\lambda \to 0$ the optimal system will resemble a free-final-time, quadratic-acceleration-optimal control problem, whereas for $\lambda \to \infty$ the optimal system will resemble a time-optimal control problem. For this problem we arbitrarily choose $\lambda = 1$ and according to the Section 3.1, the weighted time and quadratic acceleration minimization problem can be formulated as follows:
The performance measure

$$J = \sum_{k=0}^{N-1} [1 + u_1^2(k)]u_2(k), \tag{3.38}$$

subject to the dynamics

$$x_1(k+1) = x_1(k) + x_2(k) \times u_2(k);$$

$$x_2(k+1) = x_2(k) + u_1(k) \times u_2(k);$$

$$k = 0, 1, \ldots, N-1$$

and the constraints

$$0 \leq x_2(k) \leq 20; k = 0, 1, \ldots, N,$$

$$-1 \leq u_1(k) \leq 1; k = 0, 1, \ldots, N-1,$$

$$x_1(0) = 0, x_2(0) = 0,$$

$$x_1(N) = 1000, x_2(N) = 0.$$

Because this problem is also a variable-time problem, we will consider the control vector as

$$\bar{u}(k) = [u_1(k), u_2(k)]^T; k = 0, 1, \ldots, N-1$$

and the state vector as

$$\bar{x}(k) = [x_1(k), x_2(k)]^T; k = 0, 1, \ldots, N$$

From the results of this problem, we can know that this car applies the maximum acceleration 1 m/sec^2 at the beginnig and reduces the acceleration to -1 m/sec^2 linearly to minimize the elapsed time and quadratic acceleration until it stops at the point E.

The continuous time solution for this problem is list as follows: [4]

$$x_1(t) = -0.0043t^3 + 0.5t^2, \tag{3.39}$$

$$x_2(t) = -0.0129t^2 + t, \tag{3.40}$$

$$u_1(t) = -0.0258t + 1, \tag{3.41}$$

where

$$0 \leq t \leq 77.460.$$

Tables 3.7, 3.8 and 3.9 show the results of the weighted time and quadratic acceleration minimization problem for the stage numbers $N = 6, N = 11$ and $N = 21$, respectively. Figures 3.8, 3.9 and 3.10 show the optimum positions, the optimum velocities and the optimum accelerations for the weighted time and quadratic acceleration minimization problem, respectively.

Table 3.7 Results of the Weighted Time and Quadratic Acceleration Minimization Problem for $N = 6$.

Stage No.	Time Interval (seconds)	Position (meters)	Velocity (m/sec)	Acceleration (m/sec^2)
0	7.875	0	0	1
1	9.014	0	7.875	0.731
2	10.073	70.985	14.460	0.454
3	20.134	216.648	19.034	−0.083
4	0	− − −	− − −	− − −
5	23.056	599.863	17.355	−0.753
6	− − −	1000	0	− − −

The total travel time is 70.153 seconds.

Analytic optimal $J = 103.241$.

Approximative optimal $J = 98.118$.

Table 3.8 Results of the Weighted Time and Quadratic Acceleration Minimization Problem for $N = 11$.

Stage No.	Time Interval (seconds)	Position (meters)	Velocity (m/sec)	Acceleration (m/sec^2)
0	5.562	0	0	1
1	6.335	0	5.562	0.801
2	6.836	35.239	10.640	0.592
3	6.074	107.971	14.688	0.427
4	11.714	197.187	17.281	0.100
5	5.421	399.627	18.458	−0.037
6	5.478	499.692	18.258	−0.174
7	5.780	599.712	17.303	−0.317
8	6.472	699.731	15.469	−0.475
9	0	− − −	− − −	− − −
10	16.146	799.855	12.396	−0.768
11	− − −	1000	0	− − −

<u>The total travel time is 75.820 seconds.</u>

Analytic optimal $J = 103.241$.

Approximative optimal $J = 100.806$.

Table 3.9 Results of the Weighted Time and Quadratic Acceleration Minimization Problem for $N = 21$.

Stage No.	Time Interval (seconds)	Position (meters)	Velocity (m/sec)	Acceleration (m/sec^2)
0	4.039	0	0	1
1	4.566	0	4.039	0.861
2	4.985	18.442	7.971	0.715
3	7.864	58.176	11.534	0.447
4	3.367	148.881	15.050	0.365
5	3.073	199.550	16.279	0.291
6	2.913	249.584	17.173	0.209
7	2.813	299.605	17.781	0.141
8	2.751	349.622	18.178	0.077
9	2.719	399.634	18.391	0.016
10	2.713	449.644	18.433	−0.044
11	2.731	499.653	18.313	−0.104
12	2.774	549.661	18.028	−0.165
13	2.846	599.670	17.571	−0.227
14	2.955	649.678	16.926	−0.292
15	3.113	699.687	16.062	−0.362
16	3.349	749.696	14.933	−0.440
17	3.715	799.703	13.461	−0.527
18	4.358	849.717	11.504	−0.628
19	0	− − −	− − −	− − −
20	11.423	899.855	8.767	−0.768
21	− − −	1000	0	− − −

The total travel time is 79.067 seconds.

Analytic optimal $J = 103.241$.

Approximative optimal $J = 102.563$.

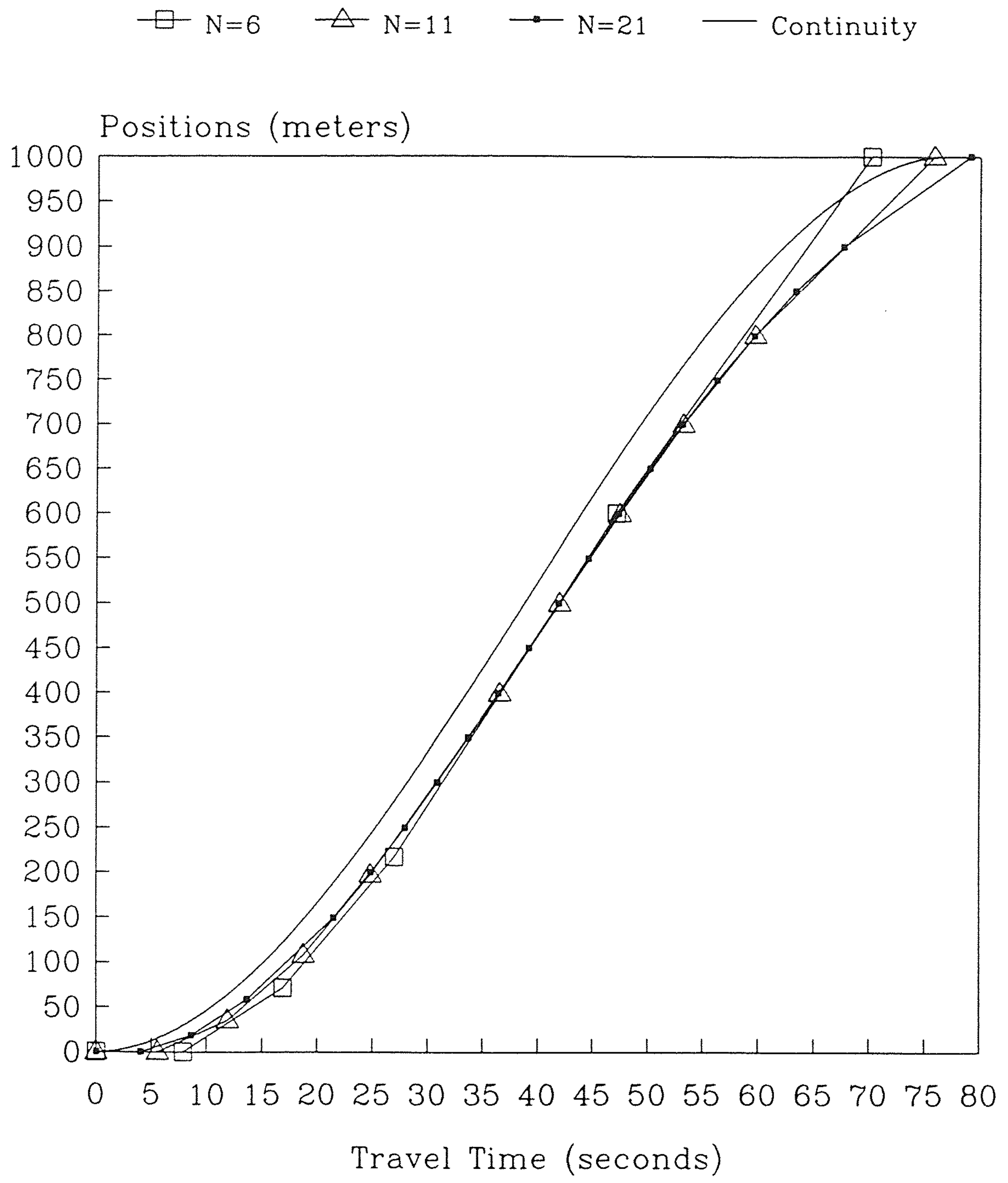

Figure 3.8 Optimum Positions for the Weighted Time and Quadratic Acceleration Minimization Problem.

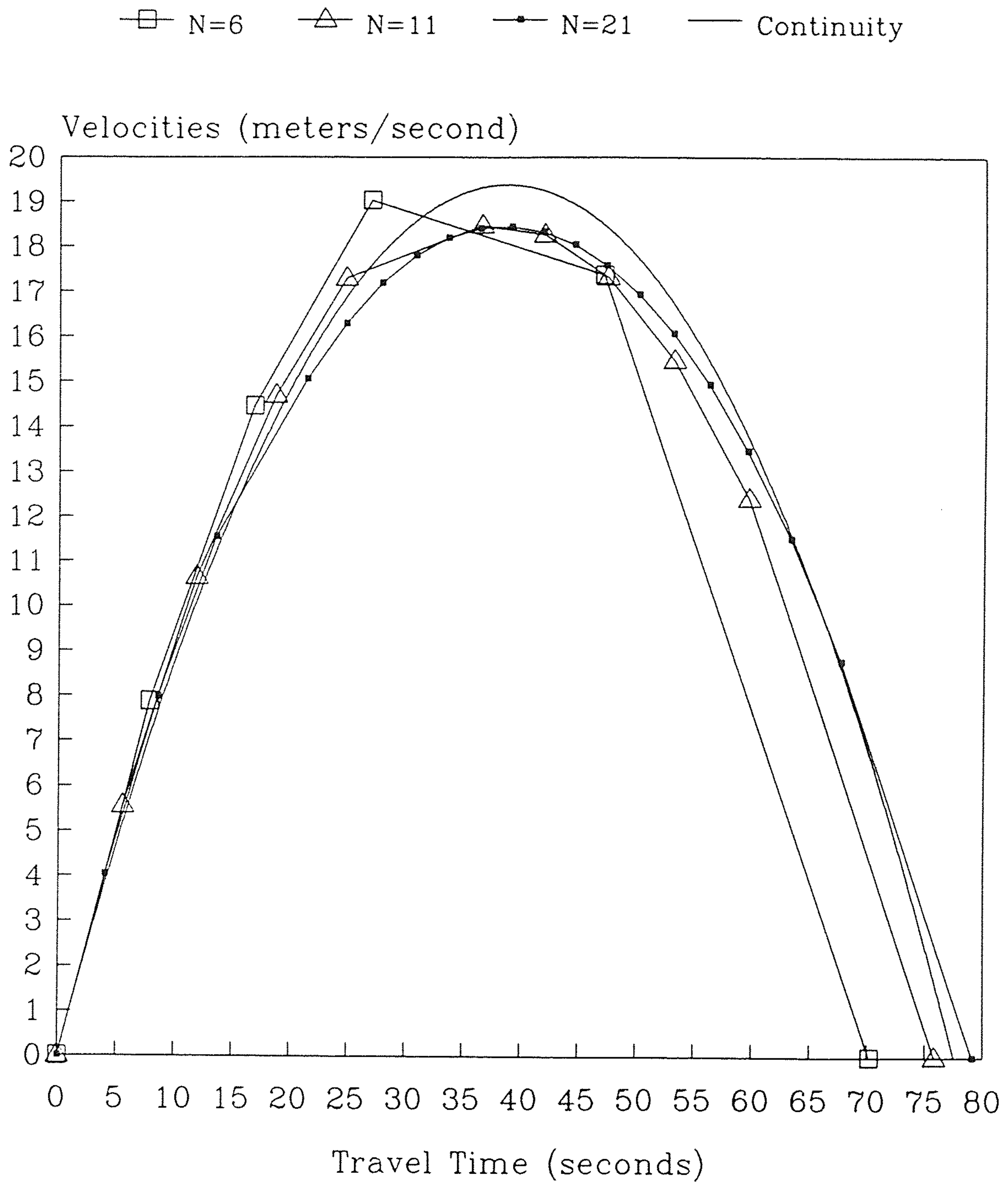

Figure 3.9 Optimum Velocities for the Weighted Time and Quadratic Acceleration Minimization Problem.

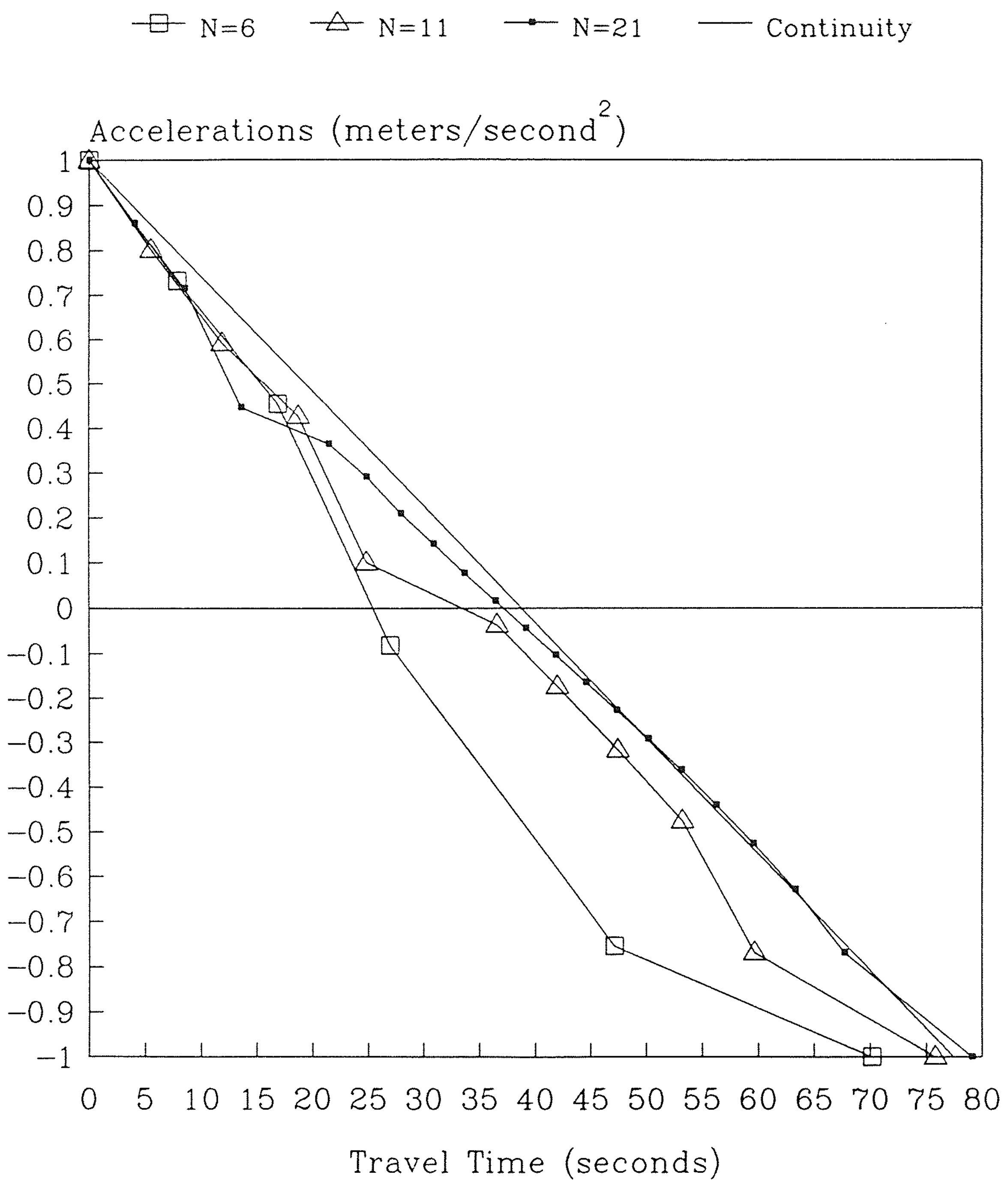

Figure 3.10 Optimum Accelerations for the Weighted Time and Quadratic Acceleration Minimization Problem.

3.5 Concluding Remarks

In this section, we will conclude some remarks for appling the successive approximation algorithm.

1. The error of the global nonlinear optimal control problem is proportional to the number of stages in the problem. For example, if the global nonlinear optimal control problem has 100 stages and the accuracy of each nonlinear subproblem is $10e - 4$, then the accuracy of the global nonlinear problem is $100 \times 10e - 4 = 10e - 2$. Therefore, if we want to increase the number of stages in the problem, we must increase the accuracy of each nonlinear subproblem proportionally.

2. The computation time for solving the nonlinear optimal control problem is almost proportional to the number of stages in the problem.[3] According to the computational experience, the more the number of stages, the more the computation time.

3. For the variable-time problems, the successive approximation algorithm tends to "stretch" some time intervals and "compress" others. This effect makes some stages disappear practically and has been explained in the Subsection 3.2.3.

4. Because the discrete system is applied to the optimal control problem. We can conjecture that the larger the number of stages, the more accurate the results and the limiting case of infinite stages would yield the continuous time solution.

CHAPTER 4
SIMULATION ON ROBOTIC MANIPULATOR

4.1 The Motion Planning Problem

In Chapter 1, we have stated that the most important and difficult part for the motion planning problem is the dynamics describing the manipulator. Considering a n-degree-of-freedom manipulator, the dynamics of the manipulator can be calculated by iterative Newton-Euler dynamics algorithm [20] or Lagrangian equation [19] and has the general form: [1,20]

$$M(\bar{q}_p)\ddot{\bar{q}}_p + h(\bar{q}_p, \dot{\bar{q}}_p) + g(\bar{q}_p) = \bar{u}, \tag{4.1}$$

where

$\bar{q}_p$	: the position vector of the generalized coordinate;
$M(\bar{q}_p)$	: symmetric inertia matrix;
$h(\bar{q}_p, \dot{\bar{q}}_p)$	: coriolis and centripetal torque or force vector;
$g(\bar{q}_p)$	: gravity torque or force vector;
$\bar{u}$	: torque or force vector.

Assuming that the control vector $\bar{u}$ represents the torque/force applied at each joint and the state vector correspond to $(\bar{q}_p, \bar{q}_v)$, where $\bar{q}_p$ and $\bar{q}_v$ represent the vector of joint positions and velocities respectively. Then the state equations describing the dynamics of the system are given by: [19]

$$\dot{\bar{q}}_p = \bar{q}_v, \tag{4.2}$$

$$\dot{\bar{q}}_v = [M(\bar{q}_p)]^{-1}[-h(\bar{q}_p, \bar{q}_v) - g(\bar{q}_p) + \bar{u}], \tag{4.3}$$

where

$$\bar{q}_p \in R^n, \bar{q}_v \in R^n, \bar{u} \in R^n.$$

In addition, there are several inequality constraints imposed on the system. Typically, these inequality constraints include the limitations on the actuator torques/forces and on the joint velocities. For the purpose of generality, these inequality constraints can be expressed as

$$\bar{v}(\bar{q}_p) \leq \bar{q}_v \leq \bar{w}(\bar{q}_p), \tag{4.4}$$

$$\bar{r}(\bar{q}_p, \bar{q}_v) \leq \bar{u} \leq \bar{s}(\bar{q}_p, \bar{q}_v), \tag{4.5}$$

where the constraints on the joint velocities have been represented as functions of joint positions and the constraints on the actuator torques/forces have been represented as functions of joint velocities and positions . The constraints on the joint velocities are expressed independently with the torque/force constraints, because the motion of the manipulator can become unstable at high speeds.

The performance measure to be minimized can be expressed as

$$J = \int_0^T V(\bar{q}_p, \bar{q}_v, \bar{u})dt. \tag{4.6}$$

For the minimum time problems, $V(\bar{q}_p, \bar{q}_v, \bar{u}) = 1$.

Then the overall motion planning problem can be formulated as follows:
The performance measure

$$J = \int_0^T V(\bar{q}_p, \bar{q}_v, \bar{u})dt,$$

subject to the dynamics

$$\dot{\bar{q}}_p = \bar{q}_v,$$

$$\dot{\bar{q}}_v = [M(\bar{q}_p)]^{-1}[-h(\bar{q}_p, \bar{q}_v) - g(\bar{q}_p) + \bar{u}]$$

and the constraints

$$G(\bar{q}_p, \bar{q}_v, \bar{u}) \geq 0, H(\bar{q}_p, \bar{q}_v, \bar{u}) = 0,$$

where

$$\bar{v}(\bar{q}_p) \leq \bar{q}_v \leq \bar{w}(\bar{q}_p),$$

$$\bar{r}(\bar{q}_p, \bar{q}_v) \leq \bar{u} \leq \bar{s}(\bar{q}_p, \bar{q}_v).$$

4.2 Simulation on One-Link Manipulator

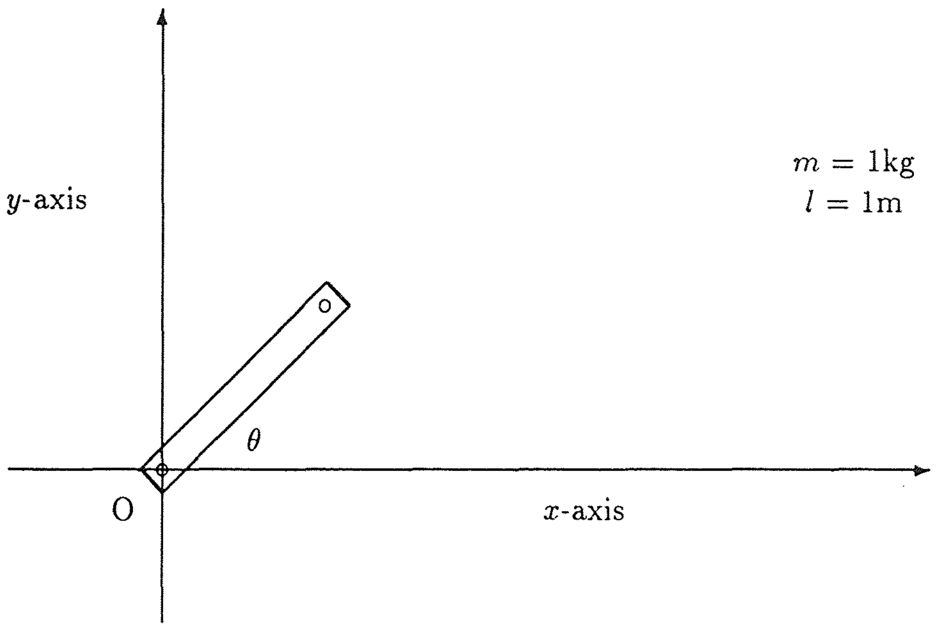

Figure 4.1 One-Link Manipulator.

The one-link manipulator is shown in Figure 4.1, where the manipulator can arbitrarily rotate at the joint O and the link has unit length and unit mass. Suppose that the torque is applied at the joint O and the gravity is applied along the inverse direction of the y-axis. Then the dynamics describing the system can be calculated as: [19,20]

$$u = 0.8274\ddot{\theta} + 4.9\cos\theta. \tag{4.7}$$

In this system, we assume the maximum angular velocity for this manipulator is ± 30 degrees/second and the maximum torque available for this manipulator is ± 5 Newton-meter.

Here, we want the manipulator to start from the y-axis ($\theta = 90°$) and stop on the x-axis ($\theta = 0°$) in the minimum time and the overall motion planning problem for one-link manipulator can be formulated as follows:

The performance measure

$$J = \int_0^T dt,$$

Table 4.1 Results of the Motion Planning Problem for One-Link Manipulator.

Stage No.	Time Interval (seconds)	Position (degrees)	Velocity (degs/sec)	Torque (Newton-meter)
0	0.087	90	0	−5
1	0.338	90	−30	0
2	0.505	79.845	−30	0.864
3	0.487	64.706	−30	2.095
4	0.487	50.105	−30	3.145
5	0.371	35.496	−30	3.992
6	0.812	24.363	−30	5
7	− − −	0	0	− − −

The total travel time is 3.087 seconds.

subject to the dynamics

$$\dot{\theta} = \omega,$$

$$\dot{\omega} = 1.2086u - 5.9259$$

and the constraints

$$-30 \leq \omega \leq 30,$$

$$-5 \leq u \leq 5,$$

$$\theta(0) = 90, \omega(0) = 0,$$

$$\theta(T) = 0, \omega(T) = 0.$$

Table 4.1 shows the results of the motion planning problem for one-link manipulator. The manipulator applies the maximum torque −5 Newton-meter at the beginning and attains the maximum velocity −30 degrees/second to save time. In order to remain this velocity, the torques must be applied at each place to balance the gravity which is along the inverse direction of y-axis. Finally, the manipulator uses the maximum torque 5 Newton-meter to stop on the x-axis.

4.3 Simulation on Two-Link Manipulator

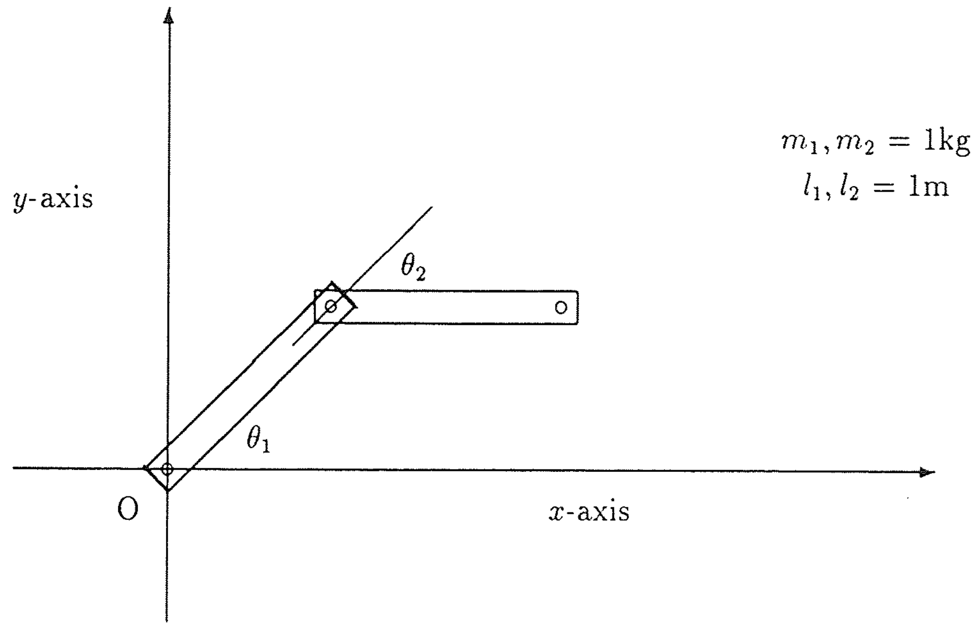

Figure 4.2 Two-Link Manipulator.

The two-link manipulator is shown in Figure 4.2. The both links are assumed with unit lengthes and unit masses and link 2 is attached at the end of link 1. The link 1 and link 2 can arbitrarily rotate at the joint 1 and joint 2, respectively. The gravity is also applied along the inverse direction of y-axis. Then the dynamics of the two-link manipulator can be calculated as: [19,20]

$$\begin{aligned} u_1 &= (2.6547 + \cos\theta_2)\ddot{\theta}_1 + (0.8274 + 0.5 \times \cos\theta_2)\ddot{\theta}_2 - \sin\theta_2 \times \dot{\theta}_1\dot{\theta}_2 \\ &\quad -0.5\sin\theta_2 \times \dot{\theta}_2^{\,2} + 14.7\cos\theta_1 + 4.9\cos(\theta_1 + \theta_2), \end{aligned} \tag{4.8}$$

$$u_2 = (0.8274 + 0.5\cos\theta_2)\ddot{\theta}_1 + 0.8274\ddot{\theta}_2 + 0.5\sin\theta_2 \times \dot{\theta}_1^{\,2} + 4.9\cos(\theta_1 + \theta_2). \tag{4.9}$$

From the Equations (4.8) and (4.9), we can know that the dynamics of the two-link manipulator is nonlinear and coupled differential equations and the more complex the manipulator, the more nonlinear and coupled the dynamics.

The maximum angular velocity for link 1 is ±30 degrees/second and for link 2 is ±45 degrees/second. The maximum torque available for link 1 is ±20 Newton-meter and for link 2 is ±10 Newton-meter.

In this problem, we also want the manipulator to start from the y-axis ($\theta_1 = 90°, \theta_2 = 0°$) and stop on the x-axis ($\theta_1 = 0°, \theta_2 = 0°$) in the minimum time. However, there is a disk obstacle in the environment. The manipulator must avoid the obstacle to accomplish this task. The obstacle can be described in the Cartesian coordinate system by:

$$(x-1.5)^2 + (y-1.5)^2 \leq 0.484^2. \tag{4.10}$$

In order to make sure the manipulator lies outside the obstacle, the obstacle avoidance conditions must be included in the motion planning problem. These conditions can be obtained by using the method described in [1] for detail and list as follows:

$$[\cos\theta_1 + 0.25 \times \cos(\theta_1+\theta_2)]^2 + [\sin\theta_1 + 0.25 \times \sin(\theta_1+\theta_2)]^2 - 0.5^2 \geq 0, \tag{4.11}$$

$$[\cos\theta_1 + 0.50 \times \cos(\theta_1+\theta_2)]^2 + [\sin\theta_1 + 0.50 \times \sin(\theta_1+\theta_2)]^2 - 0.5^2 \geq 0, \tag{4.12}$$

$$[\cos\theta_1 + 0.75 \times \cos(\theta_1+\theta_2)]^2 + [\sin\theta_1 + 0.75 \times \sin(\theta_1+\theta_2)]^2 - 0.5^2 \geq 0, \tag{4.13}$$

$$[\cos\theta_1 + \cos(\theta_1+\theta_2)]^2 + [\sin\theta_1 + \sin(\theta_1+\theta_2)]^2 - 0.5^2 \geq 0. \tag{4.14}$$

Then the motion planning problem for two-link manipulator can be formulated as follows:

The performance measure

$$J = \int_0^T dt,$$

subject to the dynamics

$$\dot{\theta}_1 = \omega_1,$$

$$\begin{aligned}\dot{\omega}_1 \quad = \quad & u_1 - (1 + 0.6043\cos\theta_2)u_2 \\ & +(0.5\sin\theta_2 + 0.1511\sin 2\theta_2)\omega_1^2 + \sin\theta_2 \times \omega_1\omega_2 + 0.5\sin\theta_2 \times \omega_2^2 \\ & -14.7\cos\theta_1 + 2.9611\cos\theta_1\cos^2\theta_2 - 1.4806\sin\theta_1\sin 2\theta_2,\end{aligned}$$

$$\dot{\theta}_2 = \omega_2,$$

$$\begin{aligned}\dot{\omega}_2 &= -(1+0.6043\cos\theta_2)u_1 + (2.2086+1.2086\cos\theta_2+0.3652\cos^2\theta_2)u_2 \\ &\quad -(1.15\sin\theta_2+0.3022\sin 2\theta_2+0.0457\sin 3\theta_2)\omega_1^2 \\ &\quad -(\sin\theta_2+0.3022\sin 2\theta_2)\omega_1\omega_2-(0.5\sin\theta_2+0.1511\sin 2\theta_2)\omega_2^2 \\ &\quad +14.7\cos\theta_1+8.8832\cos\theta_1\cos\theta_2 \\ &\quad -(5.9222+2.9611\cos\theta_2+1.7894\cos^2\theta_2)\cos(\theta_1+\theta_2)\end{aligned}$$

and the constraints

$$-30 \le \omega_1 \le 30, -45 \le \omega_2 \le 45,$$

$$-20 \le u_1 \le 20, -10 \le u_2 \le 10,$$

$$\theta_1(0) = 90, \theta_2(0) = 0, \omega_1(0) = 0, \omega_2(0) = 0,$$

$$\theta_1(T) = 0, \theta_2(T) = 0, \omega_1(T) = 0, \omega_2(T) = 0,$$

$$[\cos\theta_1 + 0.25 \times \cos(\theta_1+\theta_2)]^2 + [\sin\theta_1 + 0.25 \times \sin(\theta_1+\theta_2)]^2 - 0.5^2 \ge 0,$$

$$[\cos\theta_1 + 0.50 \times \cos(\theta_1+\theta_2)]^2 + [\sin\theta_1 + 0.50 \times \sin(\theta_1+\theta_2)]^2 - 0.5^2 \ge 0,$$

$$[\cos\theta_1 + 0.75 \times \cos(\theta_1+\theta_2)]^2 + [\sin\theta_1 + 0.75 \times \sin(\theta_1+\theta_2)]^2 - 0.5^2 \ge 0,$$

$$[\cos\theta_1 + \cos(\theta_1+\theta_2)]^2 + [\sin\theta_1 + \sin(\theta_1+\theta_2)]^2 - 0.5^2 \ge 0.$$

Figure 4.3 shows the motion strategy of the two-link manipulator to avoid the disk obstacle. Tables 4.2 and 4.3 show the results of the motion planning problem for link 1 and link 2, respectively.

Notice that the initial estimated trajectory is made by observation. First the link 2 is folded from 0° to −90° to avoid obstacle while the link 1 keeps stationary and then the link 1 is folded from 90° to 0° completely while holding the link 2 stationary. Finally, the link 2 is folded from −90° to 0° and stops on the x-axis.

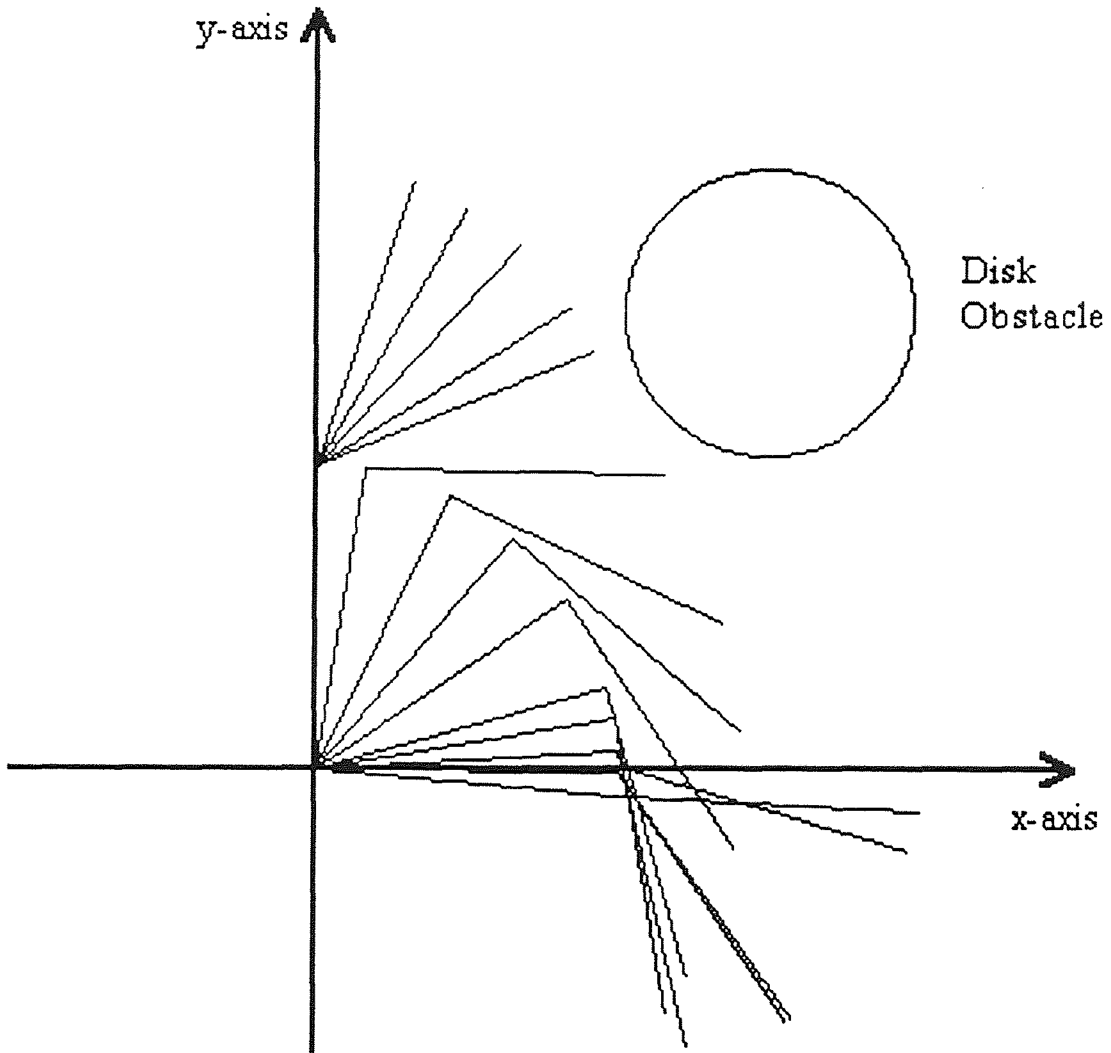

Figure 4.3 Motion Strategy of the Two-Link Manipulator to Avoid Disk Obstacle.

Table 4.2 Results of the Motion Planning Problem for Link 1.

Stage No.	Time Interval (seconds)	Position (degrees)	Velocity (degs/sec)	Torque (Newton-meter)
0	0.065	90	0	−16.043
1	0	− − −	− − −	− − −
2	0.419	90	0	0
3	0.254	90	0	1.681
4	0.274	90	0	2.624
5	0.338	90	0	3.525
6	0.215	90	0	−1.539
7	0.311	90	−30	7.060
8	0.563	80.682	−30	7.691
9	0.476	63.794	−30	11.027
10	0.522	49.525	−30	13.308
11	0.604	33.854	−30	14.946
12	0.191	15.738	−30	18.876
13	0.224	10.005	−30	18.442
14	0.545	3.278	−7.405	16.700
15	0.033	−0.756	3.272	15.030
16	0.836	−0.647	1.219	17.051
17	0.402	0.372	−13.086	20
18	0.401	−4.891	12.181	18.144
19	− − −	0	0	− − −

The total travel time is 6.674 seconds.

Table 4.3 Results of the Motion Planning Problem for Link 2.

Stage No.	Time Interval (seconds)	Position (degrees)	Velocity (degs/sec)	Torque (Newton-meter)
0	0.065	0	0	−10
1	0	− − −	− − −	− − −
2	0.419	0	−45	0
3	0.254	−18.833	−45	1.582
4	0.274	−30.250	−45	2.469
5	0.338	−42.593	−45	3.316
6	0.215	−57.809	−45	1.487
7	0.311	−67.500	−45	5.917
8	0.563	−81.476	−12.383	5.018
9	0.476	−88.447	−2.479	4.374
10	0.522	−89.626	−0.574	3.624
11	0.604	−89.926	−0.117	2.600
12	0.191	−89.997	−0.005	4.590
13	0.224	−89.998	44.961	2.172
14	0.545	−79.917	45	1.437
15	0.033	−55.400	45	1.530
16	0.836	−53.904	45	2.507
17	0.402	−16.278	45	4.360
18	0.401	1.817	−4.527	4.354
19	− − −	0	0	− − −

The total travel time is 6.674 seconds.

4.4 Simulation on Six-Degree-of-Freedom Manipulator

The six-degree-of-freedom manipulator is found in [19] and called Stanford Manipulator which is described by Paul in detail. The joints of the manipulator are all rotational except joint 3 which is prismatic. Table 4.4 lists the link parameters for the Stanford Manipulator, where d_2 and d_3 are supposed to be 0.1524 and 1 meter, respectively.

A simplified dynamics describing this system is given by: [19]

$$\begin{aligned} u_1 &= (1.422 + 2.51\sin^2\theta_2 - 5.48\sin^2\theta_2 d_3 + 6.47\sin^2\theta_2 d_3^2 + 0.23\sin^2\theta_2\cos\theta_5 d_3)\ddot{\theta}_1 \\ &\quad -(0.986\cos\theta_2 d_3)\ddot{\theta}_2 - (0.986\sin\theta_2)\ddot{d}_3, \end{aligned} \tag{4.15}$$

$$\begin{aligned} u_2 &= (4.721 - 5.48 d_3 + 6.47 d_3^2 + 0.23\cos\theta_5 d_3)\ddot{\theta}_2 + 26.869\sin\theta_2 + 63.446\sin\theta_2 d_3 \\ &\quad -1.128(\cos\theta_2\cos\theta_4\sin\theta_5 + \sin\theta_2\cos\theta_5), \end{aligned} \tag{4.16}$$

$$u_3 = 7.252\ddot{d}_3 + 63.446, \tag{4.17}$$

$$u_4 = (0.107 + 0.0003\sin^2\theta_5)\ddot{\theta}_4 + 1.128\sin\theta_2\sin\theta_4\sin\theta_5, \tag{4.18}$$

$$u_5 = 0.113\ddot{\theta}_5 + 1.128(-\cos\theta_2\sin\theta_5 + \sin\theta_2\cos\theta_4\cos\theta_5), \tag{4.19}$$

$$u_6 = 0.0203\ddot{\theta}_6. \tag{4.20}$$

The maximum velocities for link 1 through link 6 are arbitrarily selected as ±15, ±20, ±0.3 (meter/second), ±30, ±35 and ±40 degrees/second, respectively and the maximum torques available for link 1 through link 6 are arbitrarily selected as ±20, ±120, ±75, ±10, ±5 and ±2 Newton-meter, respectively.

In this problem, we want the manipulator to start from the initial point

$$\text{the initial point vector} = [0^\circ, 0^\circ, 0, 0^\circ, 0^\circ, 0^\circ]$$

and stop at the final point

$$\text{the final point vector} = [90^\circ, 108^\circ, 0.9, 144^\circ, 162^\circ, 180^\circ]$$

Table 4.4 Link Parameters for Stanford Manipulator. [19]

Link	Variable	α	a	d	$\cos\alpha$	$\sin\alpha$
1	θ_1	$-90°$	0	0	0	-1
2	θ_2	$90°$	0	d_2	0	1
3	d_3	$0°$	0	d_3	1	0
4	θ_4	$-90°$	0	0	0	-1
5	θ_5	$90°$	0	0	0	1
6	θ_6	$0°$	0	0	1	0

in the minimum time.

Tables 4.5, 4.6, 4.7 and Figures 4.4, 4.5, 4.6 show the optimum positions, the optimum velocities and the optimum torques of the motion planning problem for Stanford Manipulator.

From the results of this problem, we find a very interesting conclusion; i.e. the optimal policy for this problem is that the slowest link applies the maximum velocity and the other links apply the equal but not maximum velocities to accomplish this task. Here we call the slowest link as a "control" link, because it can control the system behavior in the minimum time problems. Certainly, if we change the maximum velocity of the slowest link, then the "control" link will become another slowest link. For example, if we change the maximum velocity of link 1 in this problem as ± 25 degrees/second, then the "control" link should be link 2 but not link 1.

Table 4.5 Optimum Positions of the Motion Planning Problem for Stanford Manipulator.

Stage No.	Time Interval	Optimum Positions (degrees)					
	(seconds)	joint 1	joint 2	joint 3*	joint 4	joint 5	joint 6
0	0.094	0	0	0	0	0	0
1	0.726	0	0	0	0	0	0
2	0.649	10.891	13.069	0.109	17.426	19.604	21.782
3	0.659	20.619	24.743	0.206	32.990	37.114	41.238
4	0.664	30.502	36.603	0.305	48.804	54.904	61.004
5	0.668	40.459	48.551	0.405	64.734	72.827	80.918
6	0.673	50.475	60.570	0.505	80.761	90.856	100.951
7	0.682	60.565	72.677	0.606	96.903	109.016	121.128
8	0.790	70.801	84.961	0.708	113.281	127.441	141.602
9	0.490	82.647	99.178	0.826	132.238	148.767	165.294
10	− − −	90	108	0.9	144	162	180

<u>The total travel time is 6.094 seconds.</u>

* The unit of the optimum positin for joint 3 is meter.

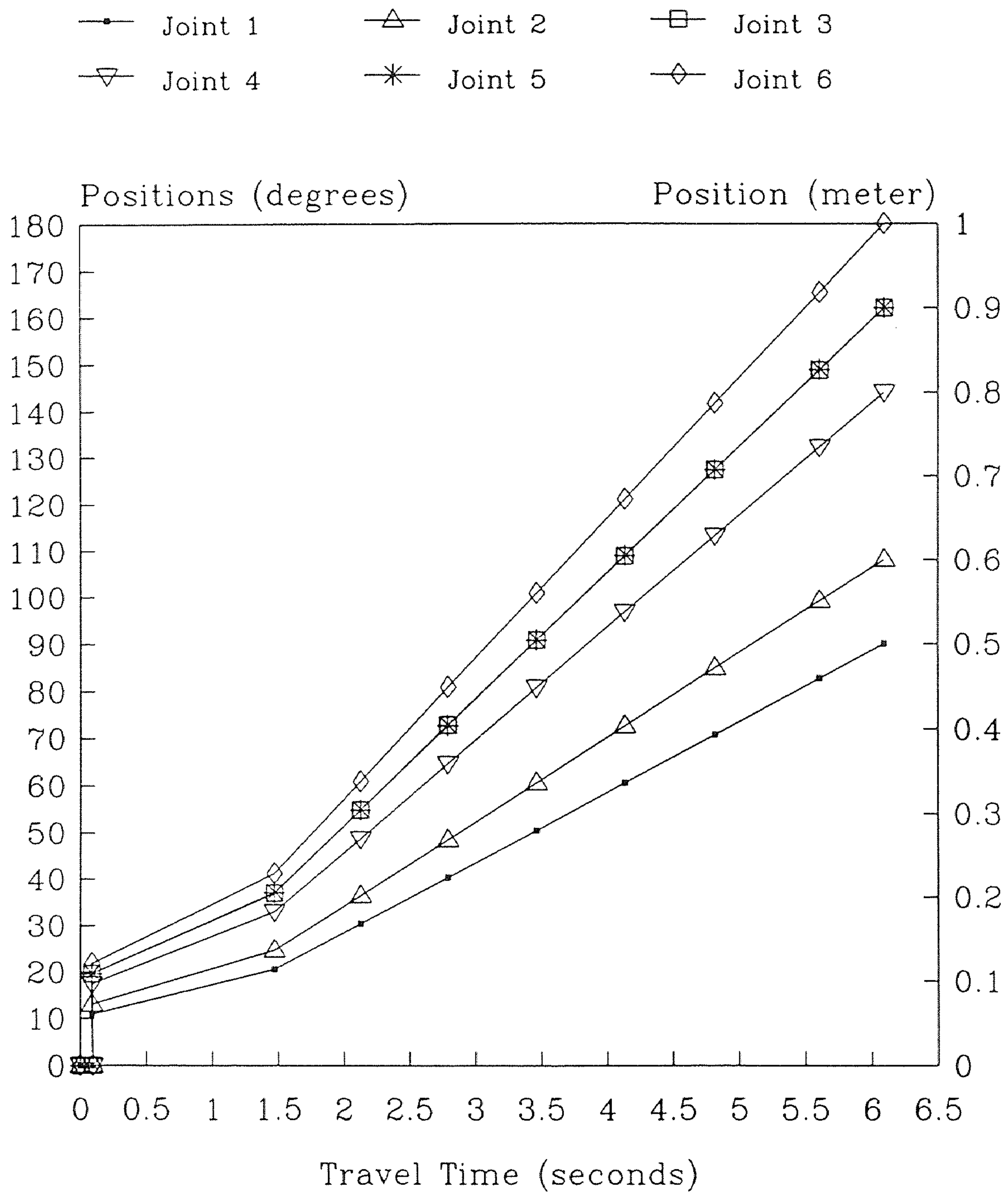

Figure 4.4 Optimum Positions of the Motion Planning Problem for Stanford Manipulator.

Table 4.6 Optimum Velocities of the Motion Planning Problem for Stanford Manipulator.

Stage No.	Time Interval (seconds)	Optimum Velocities (degrees/second)					
		joint 1	joint 2	joint 3*	joint 4	joint 5	joint 6
0	0.094	0	0	0	0	0	0
1	0.726	15	18	0.150	24	27	30
2	0.649	15	18	0.150	23.999	27	30
3	0.659	15	18.001	0.150	24.001	27.001	30
4	0.664	15	18	0.150	23.999	27	30
5	0.668	15	18	0.150	24.001	27	30
6	0.673	15	17.999	0.150	23.999	26.999	29.998
7	0.682	15	18	0.150	24	27.001	30.002
8	0.790	15	18.002	0.150	24.004	27.003	30
9	0.490	15	17.997	0.150	23.994	26.994	30
10	— — —	0	0	0	0	0	0

The total travel time is 6.094 seconds.

* The unit of the optimum velocity for joint 3 is meter/second.

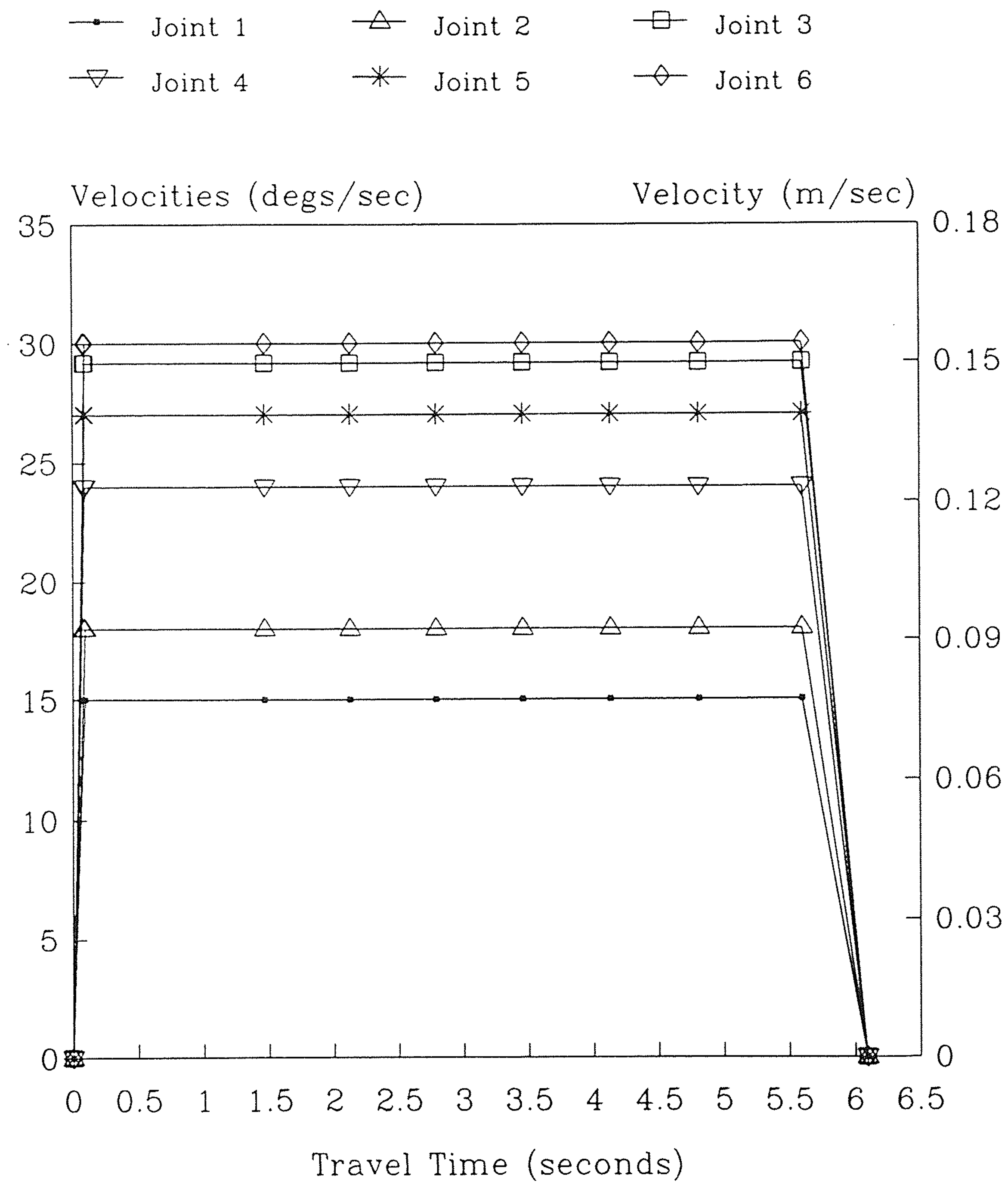

Figure 4.5 Optimum Velocities of the Motion Planning Problem for Stanford Manipulator.

Table 4.7 Optimum Torques of the Motion Planning Problem for Stanford Manipulator.

Stage No.	Time Interval (seconds)	Optimum Torques (Newton-meter)					
		joint 1	joint 2	joint 3	joint 4	joint 5	joint 6
0	0.094	3.955	15.758	75	0.476	0.566	0.113
1	0.726	0	0	63.443	0	0	0
2	0.649	0	7.046	63.443	0.026	−0.139	0
3	0.659	0	15.826	63.443	0.155	−0.302	0
4	0.664	0	26.685	63.443	0.414	−0.486	0
5	0.668	0	38.826	63.443	0.731	−0.607	0
6	0.673	0	51.220	63.443	0.970	−0.557	0
7	0.682	0	62.722	63.443	1.011	−0.275	0
8	0.790	0	72.225	63.443	0.819	0.191	0
9	0.490	−1.710	76.330	61.224	0.336	0.625	−0.022
10	− − −	− − −	− − −	− − −	− − −	− − −	− − −

The total travel time is 6.094 seconds.

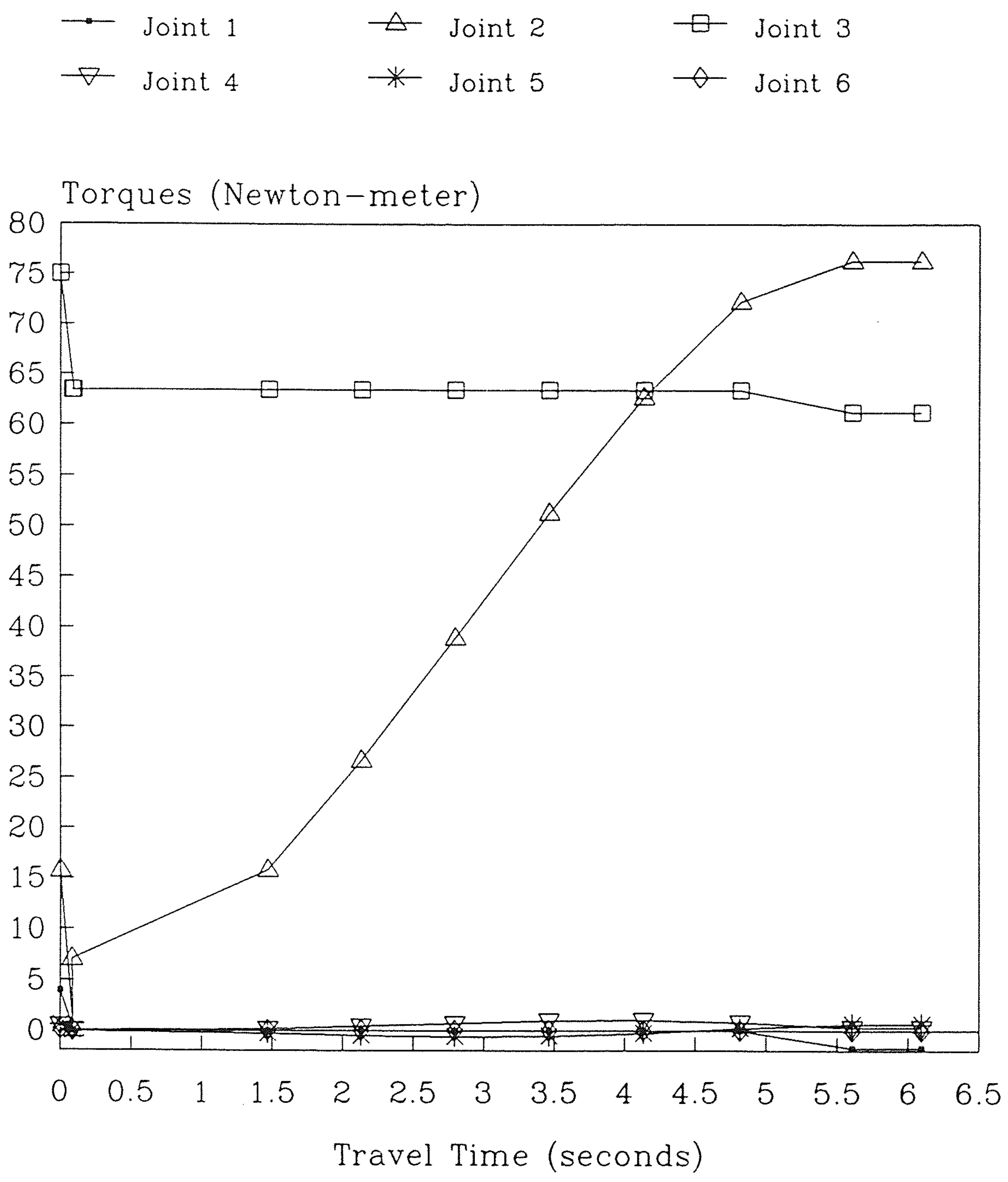

Figure 4.6 Optimum Torques of the Motion Planning Problem for Stanford Manipulator.

CHAPTER 5
CONCLUSION

In this study, we propose a successive approximation algorithm to solve a class of complex optimal control problems or deterministic multistage decision problems, such as motion planning problems.

For the motion planning problems, the planner accepts many informations to plan out the "best" trajectory. These informations include the capabilities of the manipulator, the descriptions of the task and the conditions of the environment and can be expressed as heavily nonlinear and coupled differential equations and several inequality and equality constraints imposed on the system. Therefore, many methods are extremely difficult to solve the motion planning problems.

For the successive approximation algorithm, it can decompose the global nonlinear problem into several nonlinear subproblems. Each subproblem includes only limited constraints and variables to be solved. Although the details about how to solve each subproblem can be different, the nonlinear programming method with highly convergent rate is a very good choice to solve each subproblem. Therefore, the algorithm is conceptually simple and easy to implement on the motion planning problems.

The successive approximation algorithm is not as perfect as we thought. It still have some drawbacks and limitations on the algorithm.

First, the most difficult part for using the algorithm is how to find the initial feasible sequence $\overline{W}^0$. For a more complex problem, it always takes much time to find the $\overline{W}^0$. Some skills which can be adopted to find the $\overline{W}^0$ are suggested as follows:

1. In some problems, we can find the initial feasible sequence by using observation. For example, in Section 4.3, although the constraints for this problem are very complex, we are still able to find the initial feasible trajectory by using

observation.

2. Use the simpliest numeric pattern to find the initial feasible sequence. For example, in Subsection 3.2.2, we select the initial feasible sequences in partial repetition patterns.

3. Use some trajectory-finding algorithms to find the initial feasible sequence. For example, we can use the dynamic programming method to find a coarse solution first and then use the coarse solution as an initial feasible sequence to get the accurate solution by using the successive approximation algorithm.

Secondly, the algorithm does not quarantee to obtain the global optimum. According to the Theorem 2.4, the algorithm may converge to a local optimum. Therefore, more than one initial estimate sequence has to be used to check and choose the best trajectory. That one can use the dynamic programming method to find the coarse solution and use the successive approximation algorithm to get the accurate solution is also a good method.

Thirdly, the algorithm cannot be applied in a class of problems with the "common" variables in each nonlinear subproblem. For example, in another kind of minimum time problems, the time interval of each stage is equivalent; i.e.

$$\delta t(0) = \delta t(1) = \cdots = \delta t(N) = T/N,$$

where T is not fixed. We have stated that the time intervals in variable-time problems can be considered as control variables. Therefore, since the time intervals are equivalent in this kind of minimum time problems, we can call the time intervals as "common" variables in each subproblem.

Because the time intervals must be equivalent in each subproblem, when some subproblems are solved, all the time intervals should be also changed. But for the other subproblems, the trajectory maybe becomes infeasible because the changed time intervals do not match the original trajectory in these subproblems and these subproblems will have no solutions. Therefore, we cannot solve the problems with the "common" variables by using successive approximation algorithm.

The future research directions about the successive approximation algorithm may focus on these drawbacks and limitations to improve the algorithm. First, we can develop a better and easier algorithm to find the initial feasible sequence. Secondly, we can make sure the results of the algorithm are global optimum. Thirdly, we can develop a new technique to solve the problems with the "common" variables by using successive approximation algorithm.

In conclusion, the algorithm is still a good method to solve the complex optimal control problems or large-scale nonlinear programming problems and promise for future applications.

APPENDIX A

A PROGRAM USING SUCCESSIVE APPROXIMATION ALGORITHM

In this appendix, we will show a single precision Fortran 77 program which uses successive approximation algorithm to solve the minimum time problem for $N = 6$ in Chapter 3. The program is basically divided into three parts. The first part solves the first nonlinear subproblem in each iteration $(NLP)_0^i$ which includes initial boundary conditions. The second part solves two through $N-1$ nonlinear subproblems in each iteration by using the recursive method. The third part solves the last nonlinear subproblem in each iteration $(NLP)_N^i$ which includes final boundary conditions.

The subroutine NCONF in the IMSL's Math/Library is selected to solve each nonlinear subproblem in the algorithm. The NCONF can solve the general nonlinear programming problems as follows: [18]

$$min \quad f(x), x \in R^n$$

subject to

$$g_j(x) = 0, \text{ for } j = 1, \ldots, m_e,$$

$$g_j(x) \geq 0, \text{ for } j = m_e + 1, \ldots, m,$$

$$x_l \leq x \leq x_u.$$

Therefore, we can match each nonlinear subproblem to this pattern and solve these subproblems.

The main arguments of the program can be explained as follows:

FCN	User-supplied subroutine to evaluate the criterion function and constraints at a given point.
M	Total number of constraints in the subproblem.

ME	Number of equality constraints in the subproblem.
N	Number of variables in the subproblem.
XGUESS	Vector of length N containing an initial guess of the computed solution.
IBYPE	Scalar indicating the types of bounds on variables.
XLB	Vector of length N containing the lower bounds on variables. If there is no lower bound for a variable then the corresponding XLB value should be set to $-1.0e6$.
XUB	Vector of length N containing the upper bounds on variables. If there is no upper bound for a variable then the corresponding XUB value should be set to $1.0e6$.
XSCALE	Vector of length N containing the diagonal scaling matrix for the variables.
IPRINT	Parameter indicating the desired output level.
MAXITN	Maximum number of iterations allowed.
X	Vector of length N containing the computed solution.
FVALUE	Scalar containing the value of the criterion function at the computed solution.
NX	Number of state variables.
NU	Number of control variables.
NK	Number of stages.
X(NX,NK+1)	State variables.
U(NU,NK)	Control variables.
FIX(NX*2)	Fixed values for each subproblem.
J	Values of criterion function for each stage.
CJ	Value of criterion function for the current iteration.

LJ	Value of criterion function for the last iteration.
DJ	Difference between CJ and LJ.

In this program, the state variables X(1,K), X(2,K) express the positions and velocities and the control variables U(1,K), U(2,K) express the time intervals and accelerations, respectively.

The program for solving the minimum time problem of stage number $N = 6$ is shown in the following:

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                     C
C     THE PROGRAM USES SUCCESSIVE APPROXIMATION ALGORITHM C
C                                                     C
C        TO SOLVE MINIMUM TIME PROBLEM WITH SIX STAGES    C
C                                                     C
C        Written by:                                  C
C                                                     C
C                      Cheng-Ming Chang               C
C              Electrical and Computer Engineering Dept.  C
C              New Jersey Institute of Technology, Newark, NJ.  C
C                                                     C
C        Date:         September 29, 1992             C
C                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C
C   Assign the numbers of constraints, equality constraints and variables in
C   each subproblem.
C
      INTEGER IBTYPE1,IPRINT1,M1,MAXITN1,ME1,N1
      INTEGER IBTYPE2,IPRINT2,M2,MAXITN2,ME2,N2
      INTEGER IBTYPE3,IPRINT3,M3,MAXITN3,ME3,N3
C
      PARAMETER (IBTYPE1=0,IPRINT1=0,M1=4,MAXITN1=100,ME1=4,N1=4)
      PARAMETER (IBTYPE2=0,IPRINT2=0,M2=4,MAXITN2=100,ME2=4,N2=6)
      PARAMETER (IBTYPE3=0,IPRINT3=0,M3=4,MAXITN3=100,ME3=4,N3=4)
C
      REAL FVALUE1,X1(N1),XGUESS1(N1),XLB1(N1),XSCALE1(N1),XUB1(N1)
      REAL FVALUE2,X2(N2),XGUESS2(N2),XLB2(N2),XSCALE2(N2),XUB2(N2)
      REAL FVALUE3,X3(N3),XGUESS3(N3),XLB3(N3),XSCALE3(N3),XUB3(N3)
C
C   Assign the numbers of state variables, control variables and stages.
C
      INTEGER I,K,NX,NU,NK
      PARAMETER (NX=2,NU=2,NK=6)
      REAL X(NX,NK+1),U(NU,NK),FIX(NX*2)
      REAL J(NK),CJ,LJ,DJ,e
      COMMON FIX
C
      EXTERNAL NCONF,FCN1,FCN2,FCN3
C
C   Input the bounded values of state and control variables.
C
      DATA XSCALE1/N1*1.0/,XSCALE2/N2*1.0/,XSCALE3/N3*1.0/
      DATA XLB1/0.0,-20.0,0.0,-1.0/,
    & XUB1/1000.0,20.0,1.0E6,1.0/
      DATA XLB2/0.0,-1.0,0.0,-20.0,0.0,-1.0/,
    & XUB2/1.0E6,1.0,1000.0,20.0,1.0E6,1.0/
      DATA XLB3/0.0,-1.0,0.0,-20.0/,
    & XUB3/1.0E6,1.0,1000.0,20.0/
C
C   Give the initial feasible sequence.
C
      DATA LJ/999.999/,e/1.0E-6/
      DATA ((X(I,K),K=1,NK+1),I=1,NX) /0.0,0.0,200.0,400.0,600.0,800.0,
    & 1000.0,0.0,5*10.0,0.0/
      DATA ((U(I,K),K=1,NK),I=1,NU) /6*20.0,0.5,4*0.0,-0.5/
C
C   Open an output file.
C
      OPEN (UNIT=20,TYPE='NEW',NAME='OUTPUT.DAT')
C
C   The first part solves the first nonlinear subproblem for each iteration including
C   the initial boundary conditions.
C
  1   DO 110 I=1,NX
      XGUESS1(I)=X(I,1)
      FIX(I)=X(I,2)
 110  CONTINUE
      DO 120 I=1,NU
      XGUESS1(I+NX)=U(I,1)
 120  CONTINUE
C
      CALL NCONF (FCN1,M1,ME1,N1,XGUESS1,IBTYPE1,XLB1,XUB1,
    & XSCALE1,IPRINT1,MAXITN1,X1,FVALUE1)
C
      DO 130 I=1,NK
      X(I,1)=X1(I)
 130  CONTINUE
      DO 140 I=1,NU
      U(I,1)=X1(I+NX)
 140  CONTINUE
C
C   The second part solves two through N-1 nonlinear subproblems for
C   each iteration.
C
      DO 270 K=2,NK
      DO 210 I=1,NU
      XGUESS2(I)=U(I,K-1)
 210  CONTINUE
      DO 220 I=1,NX
      XGUESS2(I+NU)=X(I,K)
      FIX(I)=X(I,K-1)
      FIX(I+NX)=X(I,K+1)
 220  CONTINUE
      DO 230 I=1,NU
      XGUESS2(I+NU+NX)=U(I,K)
 230  CONTINUE
C
      CALL NCONF (FCN2,M2,ME2,N2,XGUESS2,IBTYPE2,XLB2,XUB2,
     & XSCALE2,IPRINT2,MAXITN2,X2,FVALUE2)
C
      DO 240 I=1,NU
      U(I,K-1)=X2(I)
 240  CONTINUE
      DO 250 I=1,NX
      X(I,K)=X2(I+NU)
 250  CONTINUE
      DO 260 I=1,NU
      U(I,K)=X2(I+NU+NX)
 260  CONTINUE
 270  CONTINUE
C
C   The third part solves the last nonlinear subproblem for each iteration
C   including the final boundary conditions.
C
      DO 310 I=1,NU
      XGUESS3(I)=U(I,NK)
 310  CONTINUE
      DO 320 I=1,NX
      XGUESS3(I+NU)=U(I,NK+1)
      FIX(I)=X(I,NK)
 320  CONTINUE
C
      CALL NCONF (FCN3,M3,ME3,N3,XGUESS3,IBTYPE3,XLB3,XUB3,
     & XSCALE3,IPRINT3,MAXITN3,X3,FVALUE3)
C
      DO 330 I=1,NU
      U(I,NK)=X3(I)
 330  CONTINUE
      DO 340 I=1,NX
      X(I,NK+1)=X3(I+NU)
 340  CONTINUE
C
C   Check if the criterion function is convergent to the optimum solution or not.
C
      CJ=0.0
      DO 410 K=1,NK
      J(K)=U(1,K)
      CJ=CJ+J(K)
 410  CONTINUE
      DJ=ABS (CJ-LJ)
      IF (DJ.LT.e) GO TO 11
      LJ=CJ
      GO TO 1
C
C   Output the results to output.dat file.
C
 11   WRITE(20,520)
      DO 510 K=1,NK
      WRITE(20,530)K-1
      WRITE(20,540)U(1,K)
      WRITE(20,550)X(1,K)
      WRITE(20,560)X(2,K)
      WRITE(20,570)U(2,K)
510   CONTINUE
      WRITE(20,530)NK
      WRITE(20,550)X(1,NK+1)
      WRITE(20,560)X(2,NK+1)
      WRITE(20,580)CJ
520   FORMAT(7X,'Results of the Minimum Time Problem for N = 6.',/,/,/)
530   FORMAT(5X,'For the ',I1, 'th stage:')
540   FORMAT(5X,'Time Interval = ',F9.3, 'seconds.')
550   FORMAT(5X,'Position = ',F9.3, 'meters.')
560   FORMAT(5X,'Velocity = ',F9.3, 'm/sec.')
570   FORMAT(5X,'Acceleration = ',F9.3, 'm/sec/sec.',/,/)
580   FORMAT(/,/,5X,'The optimal performance measure J = ',F9.3,'seconds.')
      STOP
      END
C
C   The user-supplied subroutine evaluates the criterion function and constraints
C   at a given point for the first part.
C
      SUBROUTINE FCN1 (M1,ME1,N1,X1,ACTIVE1,F1,G1)
      INTEGER M1,ME1,N1,NX1
      PARAMETER (NX1=2)
      REAL X1(*),F1,G1(*),FIX(NX1*2)
      COMMON FIX
      LOGICAL ACTIVE1(*)
C
      F1=X1(3)
C
      IF(ACTIVE1(1)) G1(1)=X1(2)+X1(4)*X1(3)-FIX(2)
      IF(ACTIVE1(2)) G1(2)=X1(1)+X1(2)*X1(3)-FIX(1)
      IF(ACTIVE1(3)) G1(3)=X1(1)
      IF(ACTIVE1(4)) G1(4)=X1(2)
C
      RETURN
      END
C
C   The user-supplied subroutine evaluates the criterion function and constraints
C   at a given point for the second part.
C
      SUBROUTINE FCN2 (M2,ME2,N2,X2,ACTIVE2,F2,G2)
      INTEGER M2,ME2,N2,NX2
      PARAMETER (NX2=2)
      REAL X2(*),F2,G2(*),FIX(NX2*2)
      COMMON FIX
      LOGICAL ACTIVE2(*)
C
      F2=X2(1)+X2(5)
C
      IF(ACTIVE2(1)) G2(1)=FIX(2)+X2(2)*X2(1)-X2(4)
      IF(ACTIVE2(2)) G2(2)=FIX(1)+FIX(2)*X2(1)-X2(3)
      IF(ACTIVE2(3)) G2(3)=X2(4)+X2(6)*X2(5)-FIX(4)
      IF(ACTIVE2(4)) G2(4)=X2(3)+X2(4)*X2(5)-FIX(3)
C
      RETURN
      END
C
C   The user-supplied subroutine evaluates the criterion function and constraints
C   at a given point for the third part.
C
      SUBROUTINE FCN3 (M3,ME3,N3,X3,ACTIVE3,F3,G3)
      INTEGER M3,ME3,N3,NX3
      PARAMETER (NX3=2)
      REAL X3(*),F3,G3(*),FIX(NX3*2)
      COMMON FIX
      LOGICAL ACTIVE3(*)
C
      F3=X3(1)
C
      IF(ACTIVE3(1)) G3(1)=FIX(2)+X3(2)*X3(1)-X3(4)
      IF(ACTIVE3(2)) G3(2)=FIX(1)+FIX(2)*X3(1)-X3(3)
      IF(ACTIVE3(3)) G3(3)=X3(3)-1000.0
      IF(ACTIVE3(4)) G3(4)=X3(4)
C
      RETURN
      END
```

The results of the program are shown in Figure A.1.

```
Results of the Minimum Time Problem for N = 6.


For the 0th stage:
Time Interval =     14.143 seconds.
Position =      0.000 meters.
Velocity =      0.000 m/sec.
Acceleration =      1.000 m/sec/sec.

For the 1th stage:
Time Interval =      5.857 seconds.
Position =      0.000 meters.
Velocity =     14.143 m/sec.
Acceleration =      1.000 m/sec/sec.

For the 2th stage:
Time Interval =     15.845 seconds.
Position =     82.834 meters.
Velocity =     20.000 m/sec.
Acceleration =      0.000 m/sec/sec.

For the 3th stage:
Time Interval =     10.000 seconds.
Position =    399.730 meters.
Velocity =     20.000 m/sec.
Acceleration =      0.000 m/sec/sec.

For the 4th stage:
Time Interval =      0.000 seconds.
Position =    599.729 meters.
Velocity =     20.000 m/sec.
Acceleration =     -0.480 m/sec/sec.

For the 5th stage:
Time Interval =     20.014 seconds.
Position =    599.730 meters.
Velocity =     20.000 m/sec.
Acceleration =     -0.999 m/sec/sec.

For the 6th stage:
Position =   1000.000 meters.
Velocity =      0.000 m/sec.


The optimal performance measure J =     65.858 seconds.
```

Figure A.1 Results of the Program for the Minimum Time Problem of $N = 6$.

BIBLIOGRAPHY

[1] Singh, S.K. "Motion Planning with Obstacles and Dynamic Constraints." *Ph.D. Thesis, Department of Mechanical Engineering, Cornell University* (Jan. 1988).

[2] Dubowsky, S., M.A. Norris, and Z. Shiller. "Time Optimal Trajectory Planning for Robotic Manipulators with Obstacle Avoidance." *IEEE Conference on Robotics and Automation, Vol. 3. San Francisco, Calif.: IEEE* (1986): 1906–1912.

[3] Zuo, Z.Q., and M.C. Zhou. "Solving Multistage Decision Problems with Non-Separable Performance Indices via Successive Approximation." *Proc. of 1992 IEEE Int. Conf. on Systems, Man & Cybernetics, Chicago, IL.* (Oct. 1992): 1526–1531.

[4] Kirk, D.E. "Optimal Control Theory an Introduction." *Prentice-Hall Inc., Englewood Cliffs, NJ.* (1970).

[5] Rozonoer, L.I. "L.S. Pontryagin's Maximum Principle in the Theory of Optimum Systems I, II, III." *Automation and Remote Control* (1960): 1288–1302, 1405–1421, 1517–1532.

[6] Bellman, R.E. "Dynamic Programming." *Princeton, NJ.: Princeton University Press,* (1957).

[7] Avriel, M. "Nonlinear Programming: Analysis and Methods." *Prentice-Hall, Englewood Cliffs, NJ.* (1976).

[8] Singh, S. "Optimal Trajectory Planning of Robotic Manipulators." *MS Thesis, Department of Mechanical Engineering, Cornell University* (Aug. 1985).

[9] Howson, H.R., and N.G.F. Sancho. "A New Algorithm for the Solution of Multi-State Dynamic Programming Problems." *Mathematical Programming 8* (1975): 104–116.

[10] Lozano-Perer, T., and M. Wesley. "An Algorithm for Planning Collision Free Path Among Polyhedral Obstacles." *Communications of the ACM Vol. 22, No. 10* (Oct. 1979): 560–570.

[11] Brooks, R.A. "Planning Collision Free Motions for Pick and Place Operations." *The International Journal for Robotics Research Vol. 2, No. 4* (1983): 19–44

[12] Schwartz, J.T., and C.K. Yap. "Advances in Robotics." *New York: Lawrence Erlbaum Associates* (1987).

[13] Kahn, M.E., and B. Roth. "The Near-Minimum Time Control of Open Loop Articulated Kinematic Chains." *ASME Journal of Dynamic Systems, Measurement and Control, Vol. 93, No. 3* (Sep. 1971): 164–172.

[14] Niv, M., and D.M. Auslander. "Optimal Control of a Robot with Obstacles." *Proceedings of the American Control Conference* (1983): 280–287.

[15] Shin, K.G., and N.D. Mckay. "A Dynamic Programming Approach to Trajectory Planning of Robotic Manipulators." *IEEE Trans. Auto. Control AC-31(6)* (1986): 491–500.

[16] Singh, S.K., and M.C. Leu. "Manipulator Motion Planning in the Presence of Obstacles and Dynamic Constraints." *The International Journal of Robotics Research, Vol. 10, No. 2* (Apr. 1991): 171–188.

[17] Zuo, Z.Q., and C.P. Wu. "Successive Approximation Technique for a Class of Large Scale NLP Problems and Its Application to Dynamic Programming." *Journal of Optimization Theory Applications 62* (1989): 515–527.

[18] "IMSL, Math/Library-Fortran Subroutines for Mathematical Applications." (1989).

[19] Paul, R.P. "Robot Manipulators: Mathematics, Programming and Control." *MIT Press, Cambridge, MA.* (1981).

[20] Craig, J.J. "Introduction to Robotics, Mechanics & Control." *Addison-Wesley Publishing Company* (1986).