INFORMATION TO USERS

This reproduction was made from a copy of a manuscript sent to us for publication and microfilming. While the most advanced technology has been used to photograph and reproduce this manuscript, the quality of the reproduction is heavily dependent upon the quality of the material submitted. Pages in any manuscript may have indistinct print. In all cases the best available copy has been filmed.

The following explanation of techniques is provided to help clarify notations which may appear on this reproduction.

1. Manuscripts may not always be complete. When it is not possible to obtain missing pages, a note appears to indicate this.

2. When copyrighted materials are removed from the manuscript, a note appears to indicate this.

3. Oversize materials (maps, drawings, and charts) are photographed by sectioning the original, beginning at the upper left hand corner and continuing from left to right in equal sections with small overlaps. Each oversize page is also filmed as one exposure and is available, for an additional charge, as a standard 35mm slide or in black and white paper format.*

4. Most photographs reproduce acceptably on positive microfilm or microfiche but lack clarity on xerographic copies made from the microfilm. For an additional charge, all photographs are available in black and white standard 35mm slide format.*

*For more information about black and white slides or enlarged paper reproductions, please contact the Dissertations Customer Services Department.

UMI Dissertation Information Service
University Microfilms International
A Bell & Howell Information Company
300 N. Zeeb Road, Ann Arbor, Michigan 48106
Suh, Suk-Hwan

DEVELOPMENT OF AN ALGORITHM FOR A MINIMUM-TIME TRAJECTORY PLANNING PROBLEM UNDER PRACTICAL CONSIDERATIONS

The Ohio State University

University Microfilms International 300 N. Zeeb Road, Ann Arbor, MI 48106

Copyright 1986 by Suh, Suk-Hwan All Rights Reserved
PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark √.

1. Glossy photographs or pages _____
2. Colored illustrations, paper or print ______
3. Photographs with dark background _____
4. Illustrations are poor copy ______
5. Pages with black marks, not original copy ______
6. Print shows through as there is text on both sides of page ______
7. Indistinct, broken or small print on several pages √
8. Print exceeds margin requirements ______
9. Tightly bound copy with print lost in spine ______
10. Computer printout pages with indistinct print ______
11. Page(s) _______ lacking when material received, and not available from school or author.
12. Page(s) _________ seem to be missing in numbering only as text follows.
13. Two pages numbered _______. Text follows.
14. Curling and wrinkled pages ______
15. Dissertation contains pages with print at a slant, filmed as received ________
16. Other ____________________________________________________________

____________________________________________

____________________________________________

University
Microfilms
International
DEVELOPMENT OF AN ALGORITHM FOR A
MINIMUM-TIME TRAJECTORY PLANNING PROBLEM
UNDER PRACTICAL CONSIDERATIONS

DISSERTATION

Presented in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy in the Graduate
School of The Ohio State University

By
Suk-Hwan Suh, B.S., M.S.

The Ohio State University
1986

Reading Committee:
Dr. Albert B. Bishop
Dr. Ronald L. Lewis
Dr. Charles H. Reilly

Approved by

Albert B. Bishop
Adviser
Department of Industrial and Systems Engineering
© Copyright by
Suk-Hwan Suh
1986
To My Parents
I would like to express my sincere appreciation to the members of my committee for their continued support and understanding throughout the research.

Professor Albert B. Bishop, my academic adviser, has always encouraged me with patience and confidence in my work. The many hour discussions with him early in the mornings have been of great help. Especially, I would like to thank him for spending his time in correcting this manuscript.

I also wish to acknowledge Dr. Charles H. Reilly for his concern and interest in my work throughout the general examination and development of this thesis. To Dr. Ronald L. Lewis I want to express my sincere thanks for his warm personal encouragement and friendship through working with him in many courses.

My deepest gratitude goes to my parents whose utmost dedication to my study, not only for the period of my doctoral degree but for my whole life, was the well that I draw my strength from whenever I feel depressed and exhausted.

Last but not least, I wish to express my warmest appreciation to my wife, Eun Sook, and daughter, Jinny, for their love and patience. Because of my night-owl study habit,
I had to take my daughter's bedroom as my study room. One of her dreams has been to have her own bedroom. I am glad that I finally can make her dream come true.
May 24, 1952 .................. Born: Daegu, Korea

1976 .......................... B.S., Industrial Engineering,
Korea University,
Seoul, Korea

1978 .......................... M.S., Industrial Engineering
and Operations Research,
Korea Advanced Institute of
Science and Technology,
Seoul, Korea

1978 - 1981 .................. Project Analyst,
Hyundai Motor Company,
Seoul, Korea

1982 - 1986 .................. Teaching Associate,
The Ohio State University

FIELDS OF STUDY

Major Field: Industrial and Systems Engineering

Robotics
Computer Aided Manufacturing
Optimization
Control Theory
# TABLE OF CONTENTS

ACKNOWLEDGEMENT ......................................... iii
VITA .................................................................... v
LIST OF TABLES ............................................. viii
LIST OF FIGURES ........................................... ix

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Objective</td>
<td>1</td>
</tr>
<tr>
<td>1.2 State of the art</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Organization</td>
<td>7</td>
</tr>
</tbody>
</table>

| II. LITERATURE REVIEW | 10 |

| III. THE MINIMUM-TIME TRAJECTORY PROBLEM | 17 |
| 3.1 Overview | 17 |
| 3.2 Transformation equations | 18 |
| 3.2.1 Kinematics | 18 |
| 3.2.3 Dynamics | 22 |
| 3.3 Feasible space | 23 |
| 3.4 Trajectory primitives | 27 |
| 3.5 Formulation | 29 |

| IV. ALGORITHM DEVELOPMENT | 34 |
| 4.1 Introduction | 34 |
| 4.2 Time scaling | 35 |
| 4.3 Equal time interval | 38 |
| 4.4 Construction scheme of trajectory functions | 39 |
| 4.4.1 Scheme 1 | 41 |
| 4.4.2 Scheme 2 | 43 |
# LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Geometry of two-link robot</td>
<td>83</td>
</tr>
<tr>
<td>2. Ten example paths and their conventional minimum time</td>
<td>85</td>
</tr>
<tr>
<td>3. Computer code names</td>
<td>87</td>
</tr>
<tr>
<td>4. Computer results of the ten examples (M=30)</td>
<td>88</td>
</tr>
<tr>
<td>5. Percent reduction of time to the conventional scheme</td>
<td>91</td>
</tr>
<tr>
<td>6. CPU time comparison</td>
<td>102</td>
</tr>
<tr>
<td>7. Results of experiment 2</td>
<td>111</td>
</tr>
<tr>
<td>8. Minimum time from the presented algorithm for Bobrow's first example</td>
<td>117</td>
</tr>
<tr>
<td>9. Minimum time from the presented algorithm for Bobrow's second example</td>
<td>127</td>
</tr>
<tr>
<td>FIGURE</td>
<td>PAGE</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>1. Typical industrial robot system</td>
<td>2</td>
</tr>
<tr>
<td>2. Minimum-time trajectory problem</td>
<td>6</td>
</tr>
<tr>
<td>3. Representation of end effector coordinate frame</td>
<td>19</td>
</tr>
<tr>
<td>4. Displacement from $R_1$ to $R_2$</td>
<td>20</td>
</tr>
<tr>
<td>5. Conceptual representation of feasible space</td>
<td>24</td>
</tr>
<tr>
<td>6. Comparison of paths adopted in previous work</td>
<td>28</td>
</tr>
<tr>
<td>7. Discretization of reference path</td>
<td>48</td>
</tr>
<tr>
<td>8. General search tree for branch-and-bound method</td>
<td>50</td>
</tr>
<tr>
<td>9. Example search tree for branching rule-1</td>
<td>58</td>
</tr>
<tr>
<td>10. Example search tree for branching rule-2</td>
<td>59</td>
</tr>
<tr>
<td>11. Five directions for two-link manipulator</td>
<td>63</td>
</tr>
<tr>
<td>12. Flow chart of FP</td>
<td>76</td>
</tr>
<tr>
<td>13. Stopping rule 2</td>
<td>78</td>
</tr>
<tr>
<td>14. Two-link robot</td>
<td>82</td>
</tr>
<tr>
<td>15. Conventional trajectory scheme</td>
<td>84</td>
</tr>
<tr>
<td>16. Cartesian velocity magnitude plot (New scheme vs. conventional scheme)</td>
<td>92</td>
</tr>
<tr>
<td>17. The extra boundary condition in scheme 3</td>
<td>94</td>
</tr>
<tr>
<td>18. Boundary condition in scheme 1 and scheme 2</td>
<td>95</td>
</tr>
<tr>
<td>19. b-value plot (example 1, $\xi = 0.05$)</td>
<td>98</td>
</tr>
<tr>
<td>20. Angular velocity plots for the three schemes</td>
<td>99</td>
</tr>
</tbody>
</table>
21. Angular acceleration plot for the three schemes . . 100
22. Cartesian velocity and acceleration plots (example 1, $\xi = 0.05$) ...................... 101
23. Starting and final configuration in experiment 1 . . 105
24. 110 starting solutions in experiment 1 . . . . . . 106
25. Six local optimal solutions in experiment 1 . . . . 108
26. Local optimal solutions reached by each of the 110 starting solutions ................. 109
27. Hollerbach's example ......................... 113
28. Angular acceleration profiles for Hollerbach's example .................................. 115
29. Bobrow's first example .......................... 116
30. Cartesian path plot for Bobrow's first example . . 118
31. $b$-value plot for Bobrow's first example ........ 120
32. Torque plot for Bobrow's first example ........... 121
33. Cartesian velocity and acceleration plots for Bobrow's first example .................. 122
34. Angular position plot for Bobrow's first example . . 123
35. Angular velocity plot for Bobrow's first example . . 124
36. Angular acceleration plot for Bobrow's first example 125
37. Bobrow's second example path .................. 126
38. Cartesian path plot for Bobrow's second example with path constraints ............... 128
39. Cartesian path plot for Bobrow's second example without path constraint ............. 129
40. $b$-value plot without path constraint ............ 129
41. Torque plot without path constraint .............. 130
42. Cartesian velocity magnitude plot without path constraint .................................. 130
43. Cartesian acceleration magnitude plot without path constraint ........................................... 131
44. Angular velocity plot without path constraint .......................................................... 131
45. Angular acceleration plot without path constraint ..................................................... 132
46. $C_{ij}(t)$ plot ............................................................................................................ 155
CHAPTER I

INTRODUCTION

1.1 Objective

For more than a decade, computer controlled industrial robots have been developed and used as a primary means for increasing productivity and improving quality in modern manufacturing systems. The ease of robot programming has made it economical to automate batch type manufacturing and the flexibility of robots has made them attractive for many industrial processes.

Fig. 1 shows a typical robot system consisting of the robot arm, control computer, and power unit. The computer generates commands for the robot control system that will cause a desired motion. These robots are applied in a variety of tasks in manufacturing systems. Part transfer, spray painting, arc welding, and assembly operations are typical examples.

To execute such a variety of tasks, the robot controller must be designed to take into account the characteristics of the tasks and industrial environments. One of the most important problems in the design of robotic controllers is the trajectory planning problem. Trajectory in essence defines the
motion of the robot by specifying the "time schedule of the path" for the robot hand to follow.

Fig. 1 Typical industrial robot system
(Cincinnati Milacron T3 Robot)
There are three important considerations in the planning of trajectories. First of all, the path of the robot hand should be designed properly. In industrial environments where the workspace of the robot is free of obstacles, the path is relatively unimportant compared to other considerations. However, in more complex work environments, the path of the robot hand must be designed to traverse an obstacle free space to avoid any possible collisions.

Secondly, the motion of the robot should be smooth during its entire travel. Since a trajectory once designed may be repeated a vast number of times, nonsmooth motion may well produce stresses on the machinery that are far from desirable, and may shorten the life of the robot.

Thirdly, the required torque or force during the entire motion should be obtainable from the joint actuators. If the actuators attempt to exceed their saturation levels at any point, the manipulator will leave its intended path. Such deviations are highly undesirable and potentially dangerous, particularly in highly structured environments [47].

In addition to these considerations, there is one more important consideration affecting the performance of the robot: minimizing the total travelling time. Minimum-time trajectories can lead to the reduction of the cycle times of the tasks, and the productivity can be improved eventually. However, current commercial robots are too slow to justify their use economically for many applications.
One way to combat this problem can be to increase the speed of the robot by fully utilizing its potential capability. Conventionally, the robot has been programmed to move along its path with constant acceleration and velocities which are heavily dependent on the configuration of the robot. To prevent possible saturation of the actuator, these values are chosen to be the most conservative ones which are imposed at the weakest configuration. As a result, the capability of the robot is underutilized for most of the time during the movement, and travelling time is far from being the shortest possible time.

In this dissertation, the problem of finding minimum-time trajectories is dealt with through the development of an algorithm taking into account the full dynamic model of the robot as well as the important considerations in the trajectory planning. The problem is formulated into an optimization problem by taking the minimization of task execution time as an objective function and three considerations as a set of constraints, namely: path constraints for collision avoidance, motion constraints for the smoothness of motion, and actuator constraints for the capability of robot motion.
1.2 **State of the art**

Minimum-time trajectory planning for robots has long been known to be a complicated problem. The complexities arise mainly from the fact that the robot itself is a highly coupled nonlinear system, but the physical (actuator limit, joint velocity limit, etc.) and environmental (danger of collision) constraints imposed on the motion of the robots can be very strict. Furthermore, the algorithm to be used should be implementable without requiring too much computation for a real-time control. Even in an off-line approach (refer to Chapter II), too much of a computational requirement is often hard to justify in practice.

Nevertheless, the fundamental difficulty in finding the optimal trajectory is due to the incohesiveness of path (spatial) planning and trajectory planning. For instance, suppose the robot is to move from S to D, and we want to find the path as well as the velocity scheme along the path (i.e. trajectory) so that the robot can finish its trip in minimum time while satisfying all the constraints (Fig. 2).

Notice there are two different problems involved in the minimum-time trajectory problem: finding a collision-free path and finding the optimal velocity scheme. The first problem involves purely the spatial (path) planning which has nothing to do with the dynamics of the robot (thus, time), while the reverse situation is true for the second problem.
Obviously, however, the minimum time is considerably affected by the path chosen, but the quality of the path in terms of minimum time cannot be evaluated in the path planning stage. For this reason, an alternative performance index adopted in the spatial planning area is "distance" which is not necessarily closely related with "time". In other words, the minimum-distance path is not necessarily the minimum-time path. Strictly speaking, the true minimum-time trajectory cannot be claimed unless all the possible collision-free paths are evaluated exhaustively.

Thus, for the optimal design of robot motion, both spatial and dynamic problems should be incorporated in an appropriate
manner. However, the inherent nature of these two problems complicates their joint consideration. Thus, the two are usually studied separately in conventional planning of robot applications.

For this reason, previous works in this area have fallen short of a general solution. Most commonly, they have ignored the path constraint, or have assumed that the workspace of the robot is free of obstacles, or they have assumed the collision-free path is precisely given by the path planner. At the same time, the dynamics of the robot have not been fully taken into consideration nor has the smoothness of robot motion been considered in their research.

1.3 Organization

Chapter I presents the introduction of this dissertation including the description of the problem, the objective of this research, and the complexity of the problem through a quick look at the state of the art of the minimum-time trajectory problem.

Chapter II is devoted to the review of literature on the minimum-time trajectory problem including the strong and weak points of previous works from a variety of points of view. In particular, the merits of the off-line, joint-space trajectory planning approach, into which category the presented approach falls, is described in detail.
Chapter III introduces the minimum-time trajectory problem. The concept and definition of the constraints dealt with in the model to be presented, including the assumptions made throughout the dissertation, are described. In particular, a detailed discussion of the concept and the definitions of the feasible space and path constraint is included. Finally, the mathematical representation of the minimum-time trajectory problem is developed.

Chapter IV is dedicated to the development of an algorithm for solving the minimum-time trajectory problem. Fundamental properties involving the time scaling property, which is used as the basis for the development of the algorithm, are described. After which, three schemes for constructing trajectory primitives are described in detail. Finally, a two-phase algorithm consisting of a branch-and-bound search method and a steepest gradient method is presented.

Chapter V compares the performance of the three schemes for the two phases, including the conventional trajectory planning method, through a number of experiments. Then, suggestions are made for the use of the new algorithm based upon the analysis of the results. Finally, the fundamental issues of the new algorithm, such as convergence and optimality are discussed, and then the performance of the new algorithm is compared with other methods available in the literature.
Chapter VI concludes this dissertation. The summary and contributions made in this dissertation are in this chapter, and the issues for further research are discussed.

The appendices include most of the proofs needed to conduct this study.
The earliest attempt at solving the minimum-time trajectory problem used a purely control theory approach. Kahn and Roth [22] derived the necessary conditions for the time-optimal control problem based upon Pontryagin’s minimum principle to find switching points. The resultant conditions turned out to be $4n$ nonlinear coupled differential equations where $n$ denotes the number of degrees-of-freedom of the robot. Solving the nonlinear equations for such two-point boundary value problems is known to be computationally intractable even by numerical evaluation.

Alternatively, some of the authors; Luh, Walker, and Paul [34] simplified the problem by decoupling the nonlinear equations through the linearization of the dynamic equations of the robot. However, the solution turned out to be unsatisfactory because of the drastic approximation. At the same time, their solutions are hard to adapt to industrial practice where the workspace of the robot is not empty, or smooth motion is desired because the typical solution of time-optimal problems involves sudden changes of acceleration at the switching points of the bang-bang control scheme, i.e.,
switching between the maximum acceleration and maximum deceleration one or more times throughout the motion.

Recognizing the difficulties in the direct optimal control approach, many researchers have adopted two-stage optimization; namely, off-line trajectory planning followed by on-line trajectory tracking. By this approach, the direct optimal control problem can be solved off-line where the speed of computation is not of much concern. This approach can be justified by the following reasons:

a) The computational requirement in the direct optimal control approach is too heavy.

b) Other realistic constraints, such as path constraints which are considered in this dissertation, can be incorporated.

c) The same goal that the optimal control approach pursues can be achieved by accurate tracking of the off-line optimal trajectory during on-line control.

The earliest attempt to adopt the off-line optimization approach was made by Luh and Lin [32]. They attempted to find a set of time intervals for a given set of knot points such that the total travelling time is minimized. They implicitly considered the path constraint by assuming that the knot points are selected in a fashion that the end effector of the robot could avoid the obstacles by following the piecewise straight line path connected by these points.
Luh and Lin's assumption involving the actuator constraint is that there are constant bounds on the robot's velocity and acceleration at the hand level. Since the real physical constraints on the manipulator are applied torques/forces at the joints, these bounds are determined based on a number of experiments. In fact, these bounds vary significantly over the position of the robot, so they have to be the lowest values which are realizable in the worst configuration resulting in considerable underutilization of robots.

In addition, the computational requirement during on-line tracking is very high. Since the trajectory is defined in Cartesian space, it is necessary to convert the measured values of joint variables into Cartesian space by direct kinematics, then convert the Cartesian space errors into joint space by inverse kinematics, a usually time-consuming computational procedure. Finally the errors are compensated for in terms of joint torque by solving the inverse dynamics equation. This computational process must be done in every sampling period, which is hard to implement even with parallel processing mechanisms.

To alleviate the computational problem, the joint space trajectory planning method has been adopted by many researchers. Kim and Shin [24] developed a method to find a minimum-time trajectory in joint space. The trajectory primitive adopted in their formulation is the piecewise straight line path in joint space. Like Luh and Lin's [32],
each corner point has been replaced by smooth arcs to remove unnecessary stops at the knot points. The velocity profile they assumed has a trapezoidal shape which is essentially the same as in [32]. Kim and Shin's work, however, is distinguished from [32] in two respects; the trajectory is defined in joint space, and the actuator capability is implicitly considered by computing the approximated values for the maximum accelerations at every segment based upon the dynamics equations. As they pointed out, their method is valid in a collision-free environment.

Lin et al [25] used joint space cubic polynomials as a trajectory primitive by which smooth motion can be obtained. In a splined polynomial trajectory, which consists of a set of piecewise polynomials, smooth motion is obtained by imposing a set of boundary conditions at each knot point, where two local polynomials join, as constraints to be satisfied by the spline function. Each knot point imposes four boundary conditions: two boundary conditions from position, since each local polynomial is required to pass through the point, and two boundary conditions from the continuity of velocity and acceleration (refer to Ahlberg [1] for a detailed discussion on this). Thus, a cubic polynomial can fit the four boundary conditions.

Lin, Chang, and Luh's formulation [25] is based upon this theory. However, their algorithm suffers from shortcomings when dealing with the actuator constraints. In other words,
the dynamics of the robot arm are not considered. The constant bounds on the velocity and acceleration in joint space are not true representations of the physical limits of the robot. Therefore, these bounds will prohibit full utilization of the capability of the robot because the kinematic bounds used in [25] must be the most conservative values which are imposed at the weakest configuration of the robot arm. Moreover, their algorithm cannot be applied in an industrial environment where there exist obstacles because of the assumption that the workspace of robot is empty. This problem has not been removed even in recent work of Lin and Chang [26], although the shortcoming on the actuator constraint has been eliminated.

Bobrow et al [5,6] developed a time-optimal solution using the optimality criterion of conventional optimal control theory; i.e. a bang-bang control scheme. They found a set of switching rules in the phase plane defined in distance-velocity space so that minimum-time is accomplished. Note that Bobrow's "distance" is a distance along the predefined path. Interestingly, the optimal torque profile showed that one of the joints is always at its bound. They proved the optimality of this algorithm. Incidentally, Shin and Mckay [48] come to the same conclusions. They independently developed their minimum-time trajectory planning problem in joint space, while the others worked in Cartesian space.

However, both of their methods in [5,6] and [48] are valid when the path of the robot is precisely given by a path
planner. Thus their solutions are optimal for the predefined path. Consequently, a considerable portion of optimization effort is transferred to the path planner. In reality, there are an infinite number of paths that could be selected. Unless these paths are evaluated exhaustively, global optimality would be hard to claim.

Moreover, as Shin and McKay noted in the publication [48], the path defined should be a smooth curve between the starting and destination points. In addition, they assumed that the torque at each of the actuators can be arbitrarily varied between the torque limits. In fact, the true minimum-time trajectory would involve sudden changes of torque at the switching points, which makes it hard to apply when smooth motion is preferred.

Hollerbach et al [45] attempted to find true minimum-time trajectories by using Bobrow's optimality criterion. They discretized the entire joint space in terms of a state-space tree and performed an exhaustive search, which prohibits its application to even off-line trajectory planning because of the computational burden. The computation time for a two-link manipulator simulated on the Lisp machine turned out to be more than an hour. Their concern, however, was to find the true minimum-time trajectory without any constraints except the actuator constraint. They observed an interesting phenomenon called kick-effect, i.e., bending one of the joints slightly backward to obtain faster forward movement. Based upon this
observation, they concluded the true minimum-time path is not a straight line, either in joint space or in Cartesian space.
3.1 Overview

In this chapter, the minimum-time trajectory problem is formulated. The main constraints included in the problem are:

a) Path constraint,

b) Actuator constraint, and

c) Motion constraint.

The path constraint is for the avoidance of collision with obstacles existing in the work space of the robot. Collision free space (feasible space) is represented as a tube, and the tube parameters are assumed to be given by a path planner.

The actuator constraint is due to the torque limit existing in the joint actuator. The motion constraint is for the smoothness of motion throughout the travelling time.

Smoothness of motion is defined as motion for which the governing trajectory is everywhere continuous through its second derivatives. Other constraints considered are the limits in the angular position and velocity of the joints.

The trajectory function adopted is a set of joint space polynomials in which the smoothness of motion is readily
handled. The joint space trajectory polynomial functions are characterized by three decision variables: the number of knot points, the location of the knot points, and the transition time between the knot points. Thus, the problem of finding the minimum-time trajectory involves finding the best set of these decision variables. In this chapter, we do not discuss the method of how to find the optimal decision variables, but only formulate the problem through the discussion of the underlying concept of the problem.

The minimum-time trajectory problem formulated in this chapter is valid for any manipulator so long as the transformation equations, such as the kinematics and the dynamics of the robot, are known.

3.2 Transform equations

3.2.1 Kinematics

The configuration of the robot hand in Cartesian space can be represented by a six-element vector \( R = [x, y, z, \theta, \phi, \gamma]^T \). The first three elements are for the positional displacement of the end effector coordinate frame with respect to the reference coordinate frame. The other three elements involve the orientation, i.e., the rotational displacement, of the end effector coordinate frame. The rotational displacement of the end-effector coordinate frame can also be represented as a
right-handed twist by an angle $s$ about an axis $S$ called the screw axis (Fig. 3):

$$R^f = \text{Rot } [S, s] \quad (3.1)$$

where

$R^f$ is the rotational part of $R$, i.e., $[0, \theta, y]^T$,

$S$ is a three-element vector representing the screw axis, and $s$ is a scalar indicating screw angle\(^1\).

---

1 Refer to [38] for finding $S, s$ of a given end effector frame.
In Cartesian space, the positional and rotational distances between coordinate frame $R_1$ and $R_2$ are defined as:

\[
|R_2^p - R_1^p| = [(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2]^{1/2}
\]  \hspace{1cm} (3.2)

\[
|R_2^r - R_1^r| = |\text{Rot}[S_1,s_1]| = |[R_1^r]^{-1} R_2^r|
\]  \hspace{1cm} (3.3)

where the superscript $p(r)$ indicates the positional (rotational) part of the coordinate frame, and $\text{Rot}[S_1,s_1]$ is a rotation by $s_1$ about axis $S_1$ required to reorient $R_1^r$ into $R_2^r$ (Fig. 4).

Fig. 4 Displacement from $R_1$ to $R_2$
For linear motion from $R_1$ to $R_2$, the configuration at $\lambda$ where $\lambda$ is the normalized time, taking time 0 at $R_1$ and time 1 at $R_2$:

\begin{align*}
R^P(\lambda) &= R_1^P + \lambda(R_2^P - R_1^P) \\
R^S(\lambda) &= R_1^S \cdot \text{Rot}(S_1, \lambda S_1)
\end{align*}

where $0 \leq \lambda \leq 1$.

The configuration of the robot hand can also be represented by the joint space vector $\theta = [\theta_1, \ldots, \theta_n]^T$.

Notation used for mapping from one space to the other is:

\begin{align*}
\text{DIR[.]} \quad \text{(direct kinematics)} & \quad R \\
\text{INV[.]} \quad \text{(inverse kinematics)} & \quad \theta
\end{align*}

For a two-link manipulator, the direct kinematics equation is as follows [8]:

\begin{align*}
\begin{bmatrix}
  x \\
  y
\end{bmatrix} &= \begin{bmatrix}
  l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) \\
  l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2)
\end{bmatrix}
\end{align*}

(3.6)

and, the inverse kinematics equation is,

\begin{align*}
\begin{bmatrix}
  \cos \theta_2 \\
  \theta_1
\end{bmatrix} &= \begin{bmatrix}
  \frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1 l_2} \\
  \frac{\tan^{-1} \frac{y}{x} - \tan^{-1} \frac{l_2 \sin \theta_2}{l_1 + l_2 \cos \theta_2}}{l_1 + l_2 \cos \theta_2}
\end{bmatrix}
\end{align*}

(3.7)

where $l_1$, $l_2$ are the lengths of link 1 and link 2, and $\theta_1$ and $\theta_2$ are the angles of joint 1 and joint 2, respectively.
3.2.2 Dynamics

The torque required to be derived from the joint actuator to realize a planned trajectory is based on the dynamics equation. Ignoring the payload term in the end effector, the dynamics equation in general has the following form:

$$T(t) = I(\dot{\theta}(t))\ddot{\theta}(t) + \dot{\theta}'(t)^T Z(\theta(t))\dot{\theta}'(t) + g(\theta(t)).$$  \hspace{1cm} (3.8)

Simply,

$$T(t) = \text{TOR}[\theta(t), \dot{\theta}(t), \ddot{\theta}(t)]$$  \hspace{1cm} (3.9)

where $T(t)$ is the torque required at time $t$ and is an $n$-element vector, $I$ is an $n \times n$ inertia matrix, $Z$ is $n \times n \times n$ tensor, and $g$ is a gravity vector. $\theta(t), \dot{\theta}(t), \ddot{\theta}(t)$ are angular position, velocity, and acceleration at time $t$, respectively. $\text{TOR}[.]$ is an operator representing the dynamics equation.

The dynamics equations for the two-link manipulator are [8]:

$$\begin{align*}
\tau_1 &= \dot{\theta}_1^2 \left[ I_1 + I_2 + m_2 l_1 l_2 \cos \theta_2 + \frac{m_1 l_1^2 + m_2 l_2^2}{4} + m_2 l_1^2 \right] \\
&+ \dot{\theta}_2 \left[ I_2 + \frac{m_2 l_2^2}{4} + \frac{m_1 l_1 l_2}{2} \cos \theta_2 \right] \\
&- \frac{m_2 l_1 l_2}{2} \theta_2' \sin \theta_2 - m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \sin \theta_2 \\
&+ \frac{m_2 l_2}{2} \cos(\theta_1 + \theta_2) + l_1 \frac{m_1}{2} + m_2 \cos \theta_1 g_2 
\end{align*}$$  \hspace{1cm} (3.10)
\[ \tau_2 = \theta''_1 \left[ I_2 + \frac{m_1 l_1 l_2}{2} \cos \theta_2 + \frac{m_2 l_2^2}{4} \right] + \theta''_2 \left[ I_2 + \frac{m_2 l_2^2}{4} \right] + \frac{m_2 l_1 l_2}{2} \theta'_1 \sin \theta_2 + \frac{m_2 l_2 g_2}{2} \cos (\theta_1 + \theta_2) \] 

(3.11)

where \( m_1, m_2, I_1, I_2 \) are the mass and inertia of links 1 and 2, respectively.

### 3.3 Feasible space

In an industrial environment where the task is performed under the existence of obstacles, a desired path is programmed by a path planner. A common method for obtaining a desired path is to program several points through which the tip of the robot is to pass. These points, representing position and orientation of the tip, are chosen in such a manner that the resulting path, consisting of piecewise straight lines by connecting these points, can avoid collisions with the existing obstacles.

In reality, however, some deviations from the path may not cause any collision. The magnitude of the deviation would depend on the configuration and location of obstacles along the path. In other words, the desired path is just one possible path avoiding collision, and at best it is thought of as a reference path. In this context, the desired path is renamed as the reference path, and the corner points forming the desired path are renamed as reference points. In the same context, collision free-space is defined as feasible space.
The feasible space of the robot hand can be derived by the reference path and maximum deviations from it (path tolerance). Conceptually, the feasible space can be represented as a tube with axis symmetry as the reference path, and the cross sectional radius as path tolerance. For instance, the feasible space shown in Fig. 5 can be represented as:

\[ R_1, \xi_1 - R_2, \xi_2 - R_3, \xi_3 - R_4. \]

where

- \( R_i \) \( i = 1, \ldots, N \) are the reference points, and
- \( \xi_i \) \( i = 1, \ldots, N-1 \) are the path tolerances where \( N \) denotes the number of reference points.

![Fig. 5 Conceptual representation of feasible space](image-url)
In general, the reference path defined in Cartesian space describes both the positional and rotational displacement of the robot hand along the path. For positional displacement, the reference path is simply a set of connecting straight lines describing the positional element of the end effector coordinate frame in terms of X,Y,Z coordinates. Likewise, the rotational reference path can be defined by a set of straight lines in terms of screw angles along the rotational element of the path.

Let

\[ R_i = [ R_i^P, R_i^r ]^T \quad i = 1, \ldots, N, \]  
\[ \xi_i = [ \xi_i^P, \xi_i^r ]^T \quad i = 1, \ldots, N-1. \]  

(3.12)

Then, the feasible space for positional displacement \( F^P \) is:

\[ F^P = \bigcup_{i=1}^{N-1} F_i^P \]  

(3.13)

where

\[ F_i^P = \{ V | <V-L_i> \leq \xi_i^P \}, \]  

(3.14)

\[ L_i = \bigcup_{\lambda \in [0,1]} [(1-\lambda)R_i^P + R_{i+1}^P], \]  

(3.15)

\[ <V-L_i> = \min_{\lambda \in [0,1]} |V - [(1-\lambda)R_i^P + R_{i+1}^P]|, \]  

(3.16)

and \( V \) is any point in X,Y,Z space.

Note that \( <V-L_i> \) indicates the shortest distance from \( V \) to the line segment defined between \( R_i^P \) and \( R_{i+1}^P \).
Rotational feasible space can be represented as:

\[ F^r = U_{i=1}^{N-1} F_i^r \]  

(3.17)

where

\[ F_i^r = \bigcup_{\lambda \in \{0,1\}} W \in \{ |H_i^{-1}(\lambda) \cdot W| \leq \varepsilon_i \}, \]  

(3.18)

\[ H_i(\lambda) = R_i^r \cdot \text{Rot}[S_i, \lambda S_i], \]  

(3.19)

\[ \text{Rot}[S_i, S_i] = [R_i^r]^{-1} \cdot R_{i+1}^r, \]  

(3.20)

and \( W \) is any coordinate frame rotated with respect to the reference coordinate frame.

Note that \( |H_i^{-1}(\cdot) \cdot W| \) indicates rotational distance between \( W \) and the \( H_i(\cdot) \), and \( \text{Rot}[S_i, S_i] \) is the rotational displacement from \( R_i^r \) to \( R_{i+1}^r \).

Practically, a task planner could find approximate values of the tube parameters simply by leading the robot arm around the workspace and measuring the free space. Alternatively, spatial planning algorithms\(^2\) may be used if more precise values are desired. In this research, it is assumed that these values are given by a task planner by any appropriate procedure.

---

2 Such as [28], [29], [30], etc.
3.4 **Trajectory Primitive**

In the feasible space, there is an infinite number of paths that connect the starting and destination points. Obviously, the reference path adopted for the conventional trajectory scheme is one of them. In such a scheme, the robot stops at the end of each line segment of the path. The resulting motion will, therefore, not be smooth, and the task execution time will not be the shortest possible time because of these unnecessary stops.

To remove the unnecessary stops, the corner points have been replaced by smooth arcs [32]. Taylor's bounded deviation algorithm adds sufficient knot points at the Cartesian space midpoints so that the maximum deviation between the Cartesian and the joint space straight line path is less than a predefined path tolerance [53]. Fig. 6 compares the paths implied in these schemes.

Notice that there exists a kind of rule to form the path in the previous schemes. Unless they are shown to be the optimal path(s), the rule may over-constrain the path in the minimum time trajectory planning problem. The only necessary constraint in the formation of a path is keeping the path inside of the tube.
In this study, the trajectory function adopted is a set of joint space polynomial functions. The path is defined in terms of joint angles based on the polynomial functions. The Cartesian path is obtained by a transform (direct kinematics), and is constrained to stay inside of the tube. If the Cartesian path is out of the tube, the trajectory polynomial must be modified in some manner. To handle this problem,

3 Note that there is no longer an explicit governing rule in the formation of the path.
intermediate knot points can be introduced through which the trajectory functions must pass. In so doing, a feasible path can be obtained in joint space.

To ensure smooth motion, the trajectory polynomials are constrained to be continuous through the second derivatives for the entire interval. Thus, each additional knot point imposes four more boundary conditions: two positional boundary conditions, as each of the trajectory polynomials is required to pass through the knot point, and two boundary conditions for the continuity of velocity and acceleration. The robot rests at its starting point before it moves, and stops at its destination. Therefore, there are a total of $4M-2$ boundary conditions for $M$ knot points including starting and destination points.

3.5 Formulation

Joint space trajectory polynomials are characterized by three decision variables:

a) The number of knot points ($M$),

b) The location of the knot points ($\theta_i, i=1,\ldots,M$), and

c) The transition time between the knot points ($T_i, i=1,\ldots,M-1$).

---

4 Refer to Chapter II for a discussion on this.
Suppose a set of joint space trajectory polynomials, \( C_{ij}(.) \), \( i=1,\ldots,M-1 \), \( j=1,\ldots,n \), where \( n \) denotes the number of degrees-of-freedom of the robot, are obtained by choosing a set of the decision variables. Then, \( C_{ij}(t) \), \( C'_{ij}(t) \), \( C''_{ij}(t) \) specify angular position, velocity, and acceleration of the \( j^{th} \) joint at time \( t \), where \( t \in [0,T_i] \). Note that once the decision variables are chosen, the trajectory functions which contain all the trajectory information are defined.

Now, the problem to find the minimum-time trajectory can be stated as finding the best set of the decision variables so that the sum of the \( T_i \), \( i=1,\ldots,M-1 \), is minimized while satisfying the path constraint, actuator constraint, boundary conditions for motion constraint, and other constraints due to physical limits in the angular position and velocity of the joints.

Mathematical representation of the problem is summarized as follows:

**Objective Function**

\[
\text{Min} \sum_{i=1}^{M-1} T_i \quad (3.21)
\]

5 Specifically, \( C_{ij}(t) \) specifies angular position for joint \( j \) at time \( t \) in segment \( i \), and time is reset at the beginning of every segment.
Boundary Conditions

\[ C_j(0) = \theta_j, \quad C_{M-1}(T_{M-1}) = \theta_M, \quad C_i(T_1) = C_{i+1}(0) \]

\[ C'_j(0) = 0, \quad C'_{M-1}(T_{M-1}) = 0, \quad C'_i(T_1) = C'_{i+1}(0) \] (3.22)

\[ C''_j(0) = 0, \quad C''_{M-1}(T_{M-1}) = 0, \quad C''_i(T_1) = C''_{i+1}(0) \]

for \( i = 1, \ldots, M-2 \), and \( j = 1, \ldots, M-1 \).

Path Constraint

\[ D_i(t) \leq \varepsilon_{j_i(t)}^p \] for \( i = 1, \ldots, M-1 \), \( t \in [0, T_1] \) (3.23)

\[ G_i(t) \leq \varepsilon_{j_i(t)}^r \]

where

\[ D_i(t) = \min_j D_{i,j}(t), \quad j = 1, \ldots, N-1 \] (3.24)

and

\[ D_{i,j}(t) = \min_{\lambda \in [0,1]} |\text{DIR}^p[C_i(t)] - [(1-\lambda)R_j^p + R_{j+1}^p]|^7 \] (3.25)

\[ J_i(t) \in \{ j \mid D_i(t) = D_{i,j}(t) \} \] (3.26)

\[ G_i(t) = \left| \left[ R_{j_i(t)}^r \text{Rot}(S_{j_i(t)}, t S_{j_i(t)} / T_{j_i(t)}) \right]^{-1} \right| \text{DIR}^r[C_i(t)] \] (3.27)

6 Note that \( C_i(t) = [C_{i1}(t), C_{i2}(t), \ldots, C_{in}(t)]^T \),

\[ C'_{ij}(x) = \frac{d}{dt} C_{ij}(t) \bigg|_{t=x}, \quad C''_{ij}(x) = \frac{d^2}{dt^2} C_{ij}(t) \bigg|_{t=x}. \]

Note also that there are a total of \( 4M-2 \) boundary conditions for each joint.

7 \( \text{DIR}^p[.], \text{DIR}^r[.]) \) indicate direct kinematics for positional and rotational displacement of the end effector coordinate frame.
Note that $D_i(t)$, $G_i(t)$ are the positional and rotational deviations at time $t$ in the $i^{th}$ segment of the trajectory function, $C_i(t)$, and $D_{ij}(t)$ is the positional distance in Cartesian space between $C_i(t)$ and the $j^{th}$ segment of the reference path (i.e., line segment connecting $R_i$ and $R_{i+1}$). Further note that $J_i(t)$ indicates the segment number of the reference path which contains the minimum-distanced point from $C_i(t)$.

**Actuator constraint**

$$\tau_j^- \leq \tau_{ij}(t) \leq \tau_j^+$$

for $i=1,\ldots,M-1$, $j=1,\ldots,n$, $t \in [0, T_i]$, where

$$\tau_{ij}(t) = \text{TOR}[C_i(t), C'_i(t), C''_i(t)]$$

$\tau_j^-$, $\tau_j^+$ is torque minimum(maximum) of joint actuator $j$.

**Joint limit constraint**

$$\theta_j^- \leq \theta_{ij}(t) \leq \theta_j^+$$

for $i=1,\ldots,M-1$, $j=1,\ldots,n$, $t \in [0, T_i]$, where

$\theta_j^-$, $\theta_j^+$ are angular position limits of joint $j$. 
Joint velocity limit constraint

\[ \theta_{j}^- \leq \dot{\theta}_{i,j}^*(t) \leq \theta_{j}^+ \]

for \( i=1, \ldots, M-1, \quad j=1, \ldots, n, \quad t \in [0, T_i] \),

where

\( \theta_{j}^- (\theta_{j}^+) \) is angular velocity minimum (maximum)

of joint \( j \).
4.1 Introduction

The algorithm for solving this problem essentially involves the following three steps:

a) Generation of decision variables,

b) Construction of trajectory functions based on a trajectory construction scheme, and

c) Evaluation of the trajectory functions.

However, because of the inherent complexities of the problem, finding the global optimal solution seems hard to expect. Note that the optimal solution is represented in terms of decision variables by which the resulting trajectory is feasible and the total time is less than any other feasible trajectories constructed by different sets of decision variables.

Therefore, unless the property of the optimal solution is found at step a), the optimal solution can be found only by enumeration of all combinations of the decision variables. In spite of a variety of attempts, it is concluded that defining optimality at step a) suffers from the lack of generality.
Since two of the decision variables (location of knot points and transition time) are continuous variables, enumeration methods involve the discretization of the continuous variables. Even if each continuous variable is assumed to have a few possible values, straightforward enumeration methods are computationally intractable due to the huge combinatorial relationships among the variables. Although the computation is done off-line, too much computation time may not allow its use in practice. Recognizing this, one of the major concerns in this research has been to reduce the amount of computation by the use of appropriate schemes for approximation.

4.2 Time scaling

Def. 4.1 (Time scaling property)

Consider two functions in which time is an independent variable,

\[ C(t), \, 0 < t < T \]  \hspace{1cm} (4.1)
\[ \varphi(t), \, 0 < t < T/b, \, b > 0 \]  \hspace{1cm} (4.2)

If the following conditions hold, \( \varphi(t) \) is a time-scaled function of \( C(t) \), and defined to have the time scaling property:

\[ \varphi(t) = C(bt) \]  \hspace{1cm} (4.3)
\[ \varphi'(t) = b C'(bt) \]  \hspace{1cm} (4.4)
\[ \varphi''(t) = b^2 C''(bt), \, \forall \, t \in [0, T/b]. \]  \hspace{1cm} (4.5)
Lemma 4.1 (Extension of time scaling property)

Consider two sets of functions satisfying boundary conditions (3.22): 

\[ C = \{ C_i(t) \}, \quad 0 \leq t \leq T_1 \]
\[ V = \{ V_i(t) \}, \quad 0 \leq t \leq T_1/b_i, \quad b_i > 0, \quad i = 1, \ldots, M-1 \]

where \( V_i(.) \) is the time-scaled function of \( C_i(.) \), scaled by the scale factor \( b_i \).

Then, the necessary and sufficient condition that \( V \) has the time scaling property with respect to \( C \) is:

\[ b_i = b, \quad i = 1, \ldots, M-1. \]

**Proof:** See Appendix A.

**Corollary 4.1**

If \( V \) is a uniformly time-scaled function of \( C \) (\( b_i = b \) for all \( i = 1, \ldots, M-1 \)), checking of the path constraints (3.23) for \( V \) is not necessary if checking of the path constraints for \( C \) is made.

**Proof:** From Lemma 4.1, \( V \) has the time scaling property.

From Eqn. (4.3), the positional behavior of \( V \) is the same as \( C \). Q.E.D.

---

8 Note that Eqn. (3.22) is using vector notation for \( C_i \). In this Lemma, we can disregard the vector notation, since this Lemma can be applied to any joint.
Lemma 4.2 (Dynamic scaling property)\(^9\)

If the time scaling property holds between \(V=\{\nu_{ij}(.)\}\) and \(C=\{c_{ij}(.)\}, i=1,...,M-1, j=1,...,n,\)
then the minimum-time of \(C, T_c^*\) due to the velocity and torque limits of the joint actuator can be found as follows:

\[
T_c^* = \sum_{i=1}^{M-1} \frac{T_i}{b^*}
\]  
(4.6)

where

\(T_i\) is the transition time for segment \(i\) in \(C,\)

\[
b^* = \min_{i} b_i, \quad i=1,...,M-1,
\]  
(4.7)

\[
b_i = \min_{j,t} b_{ij}(t), \quad j=1,...,n, \quad t \in [0, T_i],
\]  
(4.8)

\[
b_{ij}(t) = \min \left[ \left( \frac{\theta'_j}{\sigma_1} \right), \left( \frac{\tau_j^{\sigma_2} - g_{ij}(t)}{C'_{ij}(t)} \right)^{1/2} \right]
\]  
(4.9)

and

\[
\sigma_1 = \text{sign of } \{C'_{ij}(t)\}^{10}\]
\[
\sigma_2 = \text{sign of } \{\tau_{ij}(t) - g_{ij}(t)\}
\]  
(4.10)

\(g_{ij}(t)\) is the gravity acting on the \(j^{th}\) joint at time \(t\) in segment \(i.\)

Proof: See Appendix B.

---

\(^9\) Consult Hollerbach [21] for details.

\(^{10}\) \(\sigma_1\) of \(-(+)\) indicates \(\theta'_j^- (\theta'_j^+)\), and \(\sigma_2\) of \(-(+)\) indicates \(\tau_{ij}^- (\tau_{ij}^+)\). Note that these adjustments are necessary because the values of the scaling factor must be positive.
Collorary 4.2

The minimum-time of any $V=\{V_{ij}(\cdot)\}$ which is uniformly time-scaled from $C=\{C_{ij}(\cdot)\}$, $i=1,\ldots,M-1$, $j=1,\ldots,n$, is the same as the minimum time of $C$.

Proof: See Appendix C.

4.3 Equal time interval

Collorary 4.2 assures that the minimum-time is dependent only on the ratio of intersegment time intervals, not the individual magnitudes of the time intervals, which considerably reduces the complexity of finding $T_i^*$ for a given set of knot points. Nevertheless, there still exists a combinatorial problem in finding the optimal ratio of intersegment time intervals, and it becomes very difficult to solve as the number of knot points increases. To solve this problem, the equal-time-interval, i.e., $T_i = T$ for all $i$, is assumed based on the following intrinsic properties of the polynomial trajectories.

Lemma 4.3 (Multiple optimal solution)

Suppose $C_i^*(t)$ is a set of the optimal trajectory polynomials constructed by $T_i^*\theta_k^*$, $i=1,\ldots,M'-1$, $k=1,\ldots,M'$, then the same trajectory can be constructed by a number of different sets of $T_p^*\theta_q^*$, $p=1,\ldots,m'-1$, $q=1,\ldots,m'$.

Proof: See Appendix D.
Collorary 4.3 (Existence of equal-time-interval optimal solution)

The optimal set of trajectories $\mathcal{C}_i^*(t)$ constructed by

$$T_i^*, \theta_k^*, i=1, \ldots, M'-1, \ k=1, \ldots, M',$$

can be constructed by $T_p^*, \theta_q^*$,

where

$$T_p^* = \sum_{i=1}^{M'-1} T_i^* / (M'-1),$$

and

$$p=1, \ldots, m^*-1, \ q=1, \ldots, m'^*.$$

Proof: See Appendix E.

Now, the minimum-time trajectory problem is reduced to finding the best location of knot points and the optimal number of knot points which are equally spaced in time. For convenience, a transition time of 1 second is preferred for computational purposes.

4.4 Construction scheme of trajectory functions

A trajectory made up entirely of cubic polynomials for all $M-1$ segments fits $4(M-1) = 4M-4$ boundary conditions. Since there are $4M-2$ boundary conditions to fit, two additional boundary conditions must be considered in some fashion.
The use of quartic polynomials for the first and last segments resulting in a 4/3/...3/4 trajectory is a scheme often adopted. In this scheme, trajectory functions are found by solving a matrix equation of dimension \((4M-2)\times(4M-2)\). Despite the existence of efficient algorithms for solving tridiagonal matrices, e.g., see references [1] and [14], the computational burden is considerably heavy, and numerical error becomes significant as \(M\) increases.

In addition, the matrix equation must be resolved for any changes in the location of knot points and/or the number of knot points since the coefficients of the entire set of trajectory functions are governed by the matrix equation. Moreover, the trajectory functions cannot be constructed unless the location and the number of the knot points are defined completely.

Alternate schemes construct the trajectory functions sequentially from the first segment to the last segment considering a fixed, limited number of knot point(s) ahead. These schemes do not involve the solving of matrix equations; hence, the computation cost is lower. Furthermore, the performance of partially constructed trajectory functions can be evaluated. Thus, it may not be necessary to construct the whole trajectory if the performance of a partial trajectory is not satisfactory, which saves computation time.

Depending on the method used to handle the final boundary conditions, three different schemes are developed.
In scheme 1, the final conditions are considered only in the last segment resulting in 3/3/...3/5 as the order of the polynomials. In scheme 2, the final boundary conditions are considered in the last two segments resulting in 3/3/...3/4/4. Finally, scheme 3 considers the conditions implicitly throughout the entire set of segments resulting in all quartic polynomials.

4.4.1 **Scheme 1 (3/3/.../3/5)**

As for scheme 1, cubic polynomials fit for the first \( (M-2) \) segments, and one quintic polynomial fits the last segment. Let

\[
C_{ij}(t) = a_{0ij} + a_{1ij}t + a_{2ij}t^2 + a_{3ij}t^3
\]

for \( 0 \leq t < T_i \).

be the cubic polynomial describing the trajectory between knot points \( \theta_{ij} \) and \( \theta_{i+1,j} \). For segment \( i \), the cubic polynomial for joint \( j \) can be constructed by the following four boundary conditions:

\[
C_{ij}(0) = \theta_{ij} \quad (4.12)
\]

\[
C_{ij}(T_i) = \theta_{i+1,j} \quad (4.13)
\]

\[
C'_{ij}(0) = \theta'_{ij} \quad (4.14)
\]

\[
C''_{ij}(0) = \theta''_{ij} \quad (4.15)
\]

where \( \theta'_{ij}, \theta''_{ij} \) are the angular velocity and acceleration at \( \theta_{ij} \) which are defined from the final boundary conditions of the preceding cubic polynomial \( C_{i-1,j} \):
If \( i \) equals 1, \( \theta'_{1j} \) and \( \theta''_{1j} \) are zero, since the robot rests at its initial location before it moves smoothly. Using the boundary conditions (4.12) through (4.15) the coefficients of the cubic polynomial in (4.11) can be obtained:

\[
\begin{align*}
    a_{0i,j} &= \theta_{ij} \\
    a_{1i,j} &= \theta'_{ij} \\
    a_{2i,j} &= \theta''_{ij}/2 \\
    a_{3i,j} &= (\theta_{i+1,j} - \theta_{ij})/T_i^3 - \theta'_{ij}/T_i^2 - \theta''_{ij}/2T_i.
\end{align*}
\]

For the last segment, the quintic polynomial,

\[
C_{M-1,j}(t) = a_{0i,j} + a_{1i,j}t + a_{2i,j}t^2 + a_{3i,j}t^3 + a_{4i,j}t^4 + a_{5i,j}t^5
\]

is constructed using six boundary conditions.

The coefficients of quintic polynomials are obtained using boundary conditions (4.12) through (4.15) replacing \( i \) by \( M-1 \), and two more boundary conditions; \( \theta'_{M,j} = \theta''_{M,j} = 0 \), which are from the final motion constraints:

\[
\begin{align*}
    a_{0M-1,j} &= \theta_{M-1,j} \\
    a_{1M-1,j} &= \theta'_{M-1,j} \\
    a_{2M-1,j} &= \theta''_{M-1,j}/2 \\
    a_{3M-1,j} &= 10(\theta_{M,j} - \theta_{M-1,j}) - 12\theta'_{M-1,j} - 3/2\theta''_{M-1,j} \\
    a_{4M-1,j} &= -15(\theta_{M,j} - \theta_{M-1,j}) + 8\theta'_{M-1,j} + 3/2\theta''_{M-1,j} \\
    a_{5M-1,j} &= 6(\theta_{M,j} - \theta_{M-1,j}) - 3\theta'_{M-1,j} - 1/2\theta''_{M-1,j}.
\end{align*}
\]
4.4.2 **Scheme 2** (3/.../3/4/4)

In this scheme, cubic polynomials fit for the first 
(M-3) segments, and two quartic polynomials fit for the last 
two segments. Cubic polynomials can be constructed in the same 
manner as scheme 1:

\[ C_{i,j}(t) = a_{0ij} + a_{1ij}t + a_{2ij}t^2 + a_{3ij}t^3 \] \hspace{1cm} (4.11)

for \( 0 \leq t \leq T_i \), \( i=1,...,M-3 \), \( j=1,...,n \).

The coefficients of the polynomials are computed from Eqns. 
(4.18) through (4.21). For the last two segments, two quartics 
are constructed from ten boundary conditions.

At \( \theta_{M-2,j} \), three boundary conditions imposed for \( C_{M-2,j}(t) \) 
are:

\[ C_{M-2,j}(0) = \theta_{M-2,j} \] \hspace{1cm} (4.29)
\[ C'_{M-2,j}(0) = \theta'_{M-2,j} = C'_{M-3,j}(T_{M-3}) \] \hspace{1cm} (4.30)
\[ C''_{M-2,j}(0) = \theta''_{M-2,j} = C''_{M-3,j}(T_{M-3}) \] \hspace{1cm} (4.31)

At \( \theta_{M-1,j} \), four boundary conditions are:

\[ C_{M-2,j}(T_{M-2}) = \theta_{M-1,j} \] \hspace{1cm} (4.32)
\[ C_{M-1,j}(0) = \theta_{M-1,j} \] \hspace{1cm} (4.33)
\[ C'_{M-2,j}(T_{M-2}) = C'_{M-1,j}(0) \] \hspace{1cm} (4.34)
\[ C''_{M-2,j}(T_{M-2}) = C''_{M-1,j}(0) \] \hspace{1cm} (4.35)
Finally, three boundary conditions at $\theta_{Mj}$ are:

\begin{align*}
C_{M-1,j}(T_{M-1}) &= \theta_{Mj} \\
C'_{M-1,j}(T_{M-1}) &= 0 \\
C''_{M-1,j}(T_{M-1}) &= 0
\end{align*}

(4.36)  
(4.37)  
(4.38)

Applying these ten boundary conditions for

\begin{equation}
C_{ij}(t) = a_{0ij} + a_{1ij}t + a_{2ij}t^2 + a_{3ij}t^3 + a_{4ij}t^4
\end{equation}

(4.39)

for $0 < t \leq T_i$, $i = M-2, M-1, j=1, \ldots, n$,

we get

\begin{align*}
a_{0ij} &= \theta_{ij} \\
a_{ij} &= \theta'_{ij} \\
a_{2ij} &= \theta''_{ij}/2 \\
a_{3ij} &= (-\theta'_{ij} + \theta'_{i+1,j})/T_i^2 - (2\theta''_{ij} + \theta''_{i+1,j})/3T_i \\
a_{4ij} &= (-\theta'_{ij} + \theta'_{i+1,j})/2T_i^3 - (5\theta''_{ij} + \theta''_{i+1,j})/12T_i^2
\end{align*}

(4.40)  
(4.41)  
(4.42)  
(4.43)  
(4.44)

where

\begin{align*}
i &= M-2, M-1, \quad \theta'_{Mj} = \theta''_{Mj} = 0,
\end{align*}

and

\begin{align*}
\theta'_{M-1,j} &= -\left[ \frac{T_{M-1}}{T_{M-2} + T_{M-1}} \right] \left[ \frac{2\theta_{M-2,j}}{T_{M-2}} + \theta'_{M-2,j} + T_{M-2}\theta''_{M-2,j} \right] \\
&\quad + 2 \left[ \frac{T_{M-2} - T_{M-1}}{T_{M-2} T_{M-1}} \right] \theta_{M-1,j} + 2 \left[ \frac{T_{M-2}}{T_{M-1}(T_{M-2} + T_{M-1})} \right] \theta_{M,j}
\end{align*}

(4.45)
4.4.3 Scheme 3 (4/4/.../4/4)

In this scheme, quartic polynomials are fit for all the trajectory segments. Each quartic is constructed by five boundary conditions. Three of them are from the position of three successive knot points, and two boundary conditions are from the continuity of velocity and acceleration.

For the first \((M-3)\) segments, the \(i^{th}\) quartic polynomial for joint \(j\):

\[
C_{ij}(t) = a_{0ij} + a_{1ij}t + a_{2ij}t^2 + a_{3ij}t^3 + a_{4ij}t^4
\]

is constructed based upon following five boundary conditions:

\[
\begin{align*}
C_{ij}(0) &= \theta_{i,j} \\
C'_{ij}(0) &= \theta'_{i,j} = C'_{i-1,j}(T_{i-1}) \\
C''_{ij}(0) &= \theta''_{i,j} = C''_{i-1,j}(T_{i-1}) \\
C_{ij}(T_i) &= \theta_{i+1,j} \\
C_{ij}(T_i + T_{i+1}) &= \theta_{i+2,j}
\end{align*}
\]

\[
(4.46)
\]

\[
\begin{align*}
\theta^{n_{M-1},j} &= \left[ \frac{6}{T_{M-2}+T_{M-1}} \right] \left[ \frac{2}{T_{M-2}} \theta_{M-2,j} + \theta'_{M-2,j} + T_{M-2} \theta''_{M-2,j} \right] \\
 &+ \left[ \frac{-12}{T_{M-2}T_{M-1}} \right] \theta_{M-1,j} + \left[ \frac{12}{T_{M-1}(T_{M-2}+T_{M-1})} \right] \theta_{M,j}
\end{align*}
\]
The coefficients from these boundary conditions are

Equations (4.40), (4.41), (4.42) for \( a_{0ij}, a_{1ij}, a_{2ij}, \) and

\[
a_{3ij} = \left( \frac{T_i}{T_{i+1} + T_{i+1}} \right) (\theta_{ij} - \theta_{i+2,j}) + \left( \frac{T_i + T_{i+1}}{T_i T_{i+1}} \right) (\theta_{i+1,j} - \theta_{ij})
\]

\[
- \left[ \left( \frac{T_i}{T_{i+1} + T_{i+1}} \right)^2 + T_{i+1} \right] \theta'_{ij} - \left[ \frac{T_i}{T_{i+1} + T_{i+1}} + 2T_i + T_{i+1} \right] \theta''_{ij},
\]

\[
a_{4ij} = \left( \frac{1}{T_i^3 T_{i+1}} \right) (\theta_{ij} - \theta_{i+1,j}) + \left( \frac{1}{T_i (T_i + T_{i+1})^3} \right) (\theta_{ij} - \theta_{i+2,j})
\]

\[
+ \left[ \frac{2T_i + T_{i+1}}{T_i^2 (T_i + T_{i+1})^2} \right] \theta'_{ij} + \left[ \frac{1}{2T_i (T_i + T_{i+1})} \right] \theta''_{ij}.
\]

(4.52)

(4.53)

For the last two segments, two quartics are constructed in the same way as for scheme 2. Ten boundary conditions and the coefficients of two quartic polynomials are Equations (4.29) through (4.46).

Note that the three schemes satisfy the boundary conditions (3.23). Thus, these three schemes hold the time scaling property if and only if uniform scaling is performed. This is due to Lemma 4.2.
4.5 **Two-phase algorithm**

Up to this point, we have described the schemes for the generation of $T_i$, and construction of polynomials assuming that the knot points are given. By adopting equal-time-intervals as a scheme for time scheduling, the minimum-time trajectory problem is reduced to finding the best location for an optimal number of knot points ($\theta_i^*, i=1,...,M^*$). Basically, any point in the feasible space can be considered as a knot point. To consider the location of knot points schematically, the space needs to be discretized in some fashion.

If the entire space is discretized by $m$ equally spaced points, including starting and destination points, the total number of possible paths using all possible ordered combinations of the $m$ points is $2^{m-2}$. Evaluating all these paths is computationally impractical for even moderate values of $m$. Note that using smaller values of $m$ limits the potential quality of the solution.

Instead of discretizing the entire space and evaluating all the possible paths, the algorithm presented partially discretizes the space and finds suboptimal solutions by a two-phase method.

Phase 1 restricts the location of knot points to the center of the feasible space (reference path), and finds the best solution by a branch-and-bound search technique after discretizing the reference path. However, because of the
restriction on knot point location, the phase 1 solution may decrease in quality as the path tolerance increases. The phase 2 algorithm relaxes the reference-path restriction and finds improved solutions.

Phase 2 uses the phase 1 solution as an initial feasible solution and finds the best number and location of knot points by methods described as modification, doubling, and halving processes. In this sense, phase 1 is used for the finding of a reasonably attractive feasible solution, and phase 2 is for the fine tuning of the phase 1 solution.

4.5.1 Phase 1 algorithm

Discretization of Reference Path

Let M be the total number of possible knot points equally spaced on the reference path as shown in Fig. 7.

Fig. 7 Discretization of reference path
Examples for possible paths are those which pass through:

1 - M ; no intermediate knot points,
1 - 4 - (M-1) - M, and
1 - 2 - 3 - 4 .....(M-2) - (M-1) - M.

Thus, there is a total of $2^{M-2}$ possible paths since points 1 and M are starting and destination positions, respectively, through which all paths are required to pass. Note the assumption that the path is always constructed in the order of increasing knot-point indexing number. This assumption is realistic since any paths containing backward movement are unlikely to be optimal paths.

Problem Definition of Phase 1

Find a subset of the set of M equally spaced knot points on the reference path so that the trajectory functions constructed by the knot points selected provide a minimum-time trajectory while satisfying all the constraints.
Search Tree

Branch-and-bound (BB) search

To minimize computing time, the proposed BB method tries to limit the number of explicitly-enumerated nodes in the search tree (see Fig. 8) by the use of fathoming rules. Early experiments indicated that a breadth-first branching rule is more efficient for construction schemes 1 and 2 while a depth-first branching rule is preferable for scheme 3. Fathoming rules are based upon feasibility and lower-bound estimation.
4.5.1.A. **Lower bounds**

A partial trajectory function \( \overline{C} \) is constructed using \( k \) knot points with indices corresponding to the knot points of \( Q \),

\[
Q = \{ q_1, q_2, \ldots, q_k \} \tag{4.54}
\]

where \( q_1 = 1 \) and \( k < M \).

In other words,

\[
\overline{C} = \{ C_1(t), C_2(t), \ldots, C_{k-1}(t) \},
\]

which is constructed based on \( \theta_{q_{i}} \) \( i=1, \ldots, k \), and equal transition times for all segments; i.e.

\[
T_p = 1 \quad \forall \ p = 1, \ldots, k-1^{11}.
\]

Let \( b_{i} \) be the time scaling factor of the \( i^{th} \) segment trajectory function \( C_i(t) \), \( i=1, \ldots, k-1 \), of the \( \overline{C} \):

\[
b_i = \min_{t \in [0,1]} \min_{1 \leq j \leq n} b_{i,j}(t) \tag{4.55}
\]

where \( b_{i,j}(t) \) is defined in equation (4.9).

From Lemma 4.2, the minimum time for \( \overline{C} \), \( \text{MINT}(\overline{C}) \) is:

\[
\text{MINT}(\overline{C}) = (k-1) / \min_{1 \leq i \leq k-1} b_i
\]

\[
= (k-1) / B_{k-1} \tag{4.56}
\]

\[\text{11 Notice the use of equal-time-interval of 1 sec for computational purposes, i.e., } 0 \leq t \leq 1 \text{ for each segment.}\]
The lower bound of total travelling time for $\overline{c}$, $LB(\overline{c})$:

$$LB(\overline{c}) = MINT(\overline{c}) + A'(\overline{c}) \quad (4.57)$$

where $A'(\overline{c})$ is a lower-bound estimate of time for the remaining travel.

$A'(\overline{c})$ is the maximum of the following two lower-bounds.

A.1 Lower bound-1 (LB-1)

LB-1 is the minimum time for remaining travel ignoring all the constraints except torque and angular velocity limits. LB-1 is based on the following reasoning:

The partial trajectory function needs to travel at least one more segment to reach the destination position. Suppose the maximum time-scaling factor for the one segment, though unknown, is $b^k$. Then the minimum time for this segment is $1/b^k$. However, due to Lemma 4.2 (i.e., the time scaling factor should be uniform to possess the time scaling property), the minimum time for the segment is $1/B_{k-1}$, even though $b^k$ is bigger than $B_{k-1}$.

Thus, the lowest estimate of the minimum time for the remaining travel is:

$$A^*_1(\overline{c}) = 1/B_{k-1} \quad (4.58)$$
A.2 Lower bound-2 (LB-2)

LB-2 is the minimum time to reach the destination point ($\theta_\text{f}$) from the final position of the partial trajectory (i.e., $\theta_k$) ignoring all the constraints except the angular velocity limits:

$$\dot{A}_2^*(\overline{c}) = \max_j \frac{\theta_{M_j} - \theta_{k_j}}{\theta_{r_j} \sigma}$$

(4.59)

where

$$\sigma = \text{sign of } [\theta_{M_j} - \theta_{k_j}]$$

Thus, $A^*(\overline{c}) = \max [A_1^*(\overline{c}), A_2^*(\overline{c})]$. (4.60)

Note that $A^*(\overline{c})$ is computed easily just based on the previous history of the partial path, and does not require the path information of the remaining portions.

4.5.1.B. Fathoming rules

Let $\overline{c}$ be a partial trajectory function with knot point index set (4.54), and let the knot point index for the next node be $q_{k+1}$. The augmented trajectory $\overline{c}_\text{a}$ is fathomed if any of the $C_{k_j}(t)$ $j=1,\ldots,n$ satisfies any of the following fathoming rules at any time $t$, where $C_{k_j}(t)$ is the trajectory polynomial for the $j^{th}$ joint between $\theta_{q_k}$ and $\theta_{q_{k+1}}$. 
B.1 Fathoming rule-1 (FR-1)

FR-1 is due to the joint limit constraints.

\[ C_{k_j}(t) \subseteq [\theta^-_j, \theta^+_j] \quad \forall \ t \in [0,1]. \]  (4.61)

B.2 Fathoming rule-2 (FR-2)

FR-2 is due to the path constraint.

1. \( D(t) > \xi^P_{j_1(t)} \) \quad \forall \ t \in [0,1] \quad (4.62)
2. \( G(t) > \xi^f_{j_1(t)} \) \quad (4.63)

where

\[ D(t) = \min_{i \in J(t)} D_i(t) \quad i = 1, \ldots, N-1 \]  (4.64)

and

\[ D_i(t) = \min_{\lambda \in [0,1]} |\text{DIR}^P[C_k(t)] - [(1-\lambda)R_i^p + R_{i+1}^p]| \]  (4.65)

\[ J(t) = \{i | D_i(t) = D(t)\} \quad (4.66) \]

\[ G(t) = |[R^f_{j_1(t)} \cdot \text{Rot}(S_{j_1(t)}, ts_{j_1(t)})]^{-1} \cdot \text{DIR}^f[C_k(t)]| \]  (4.67)

B.3 Fathoming rule-3 (FR-3)

\[ LB(t) > UB, \quad \forall \ t \in [0,1] \]  (4.68)

where

UB is the lowest minimum time of the complete trajectory functions found so far.

The lower bound at time \( t \) in the \( k^{th} \) segment, \( LB(t) \), is the sum of the following four times:
a) The minimum time taken to reach the knot point $k$, $\Theta_k$ from the starting position, $\Theta_1$ -- ($\text{MINT}_1(t)$)

b) The minimum time taken to reach the current position, $\mathcal{C}_k(t)$ from the $\Theta_k$ measured from the beginning of the current segment $k$ -- ($\text{MINT}_2(t)$)

c) The lowest estimate of the minimum time for finishing the current segment $k$ (i.e., the minimum time to reach $\Theta_{k+1}$ from $\mathcal{C}_k(t)$) -- ($\text{MINT}_3(t)$)

d) The lowest estimate of the minimum time for finishing the rest of segments (i.e., the minimum time to reach the destination point $\Theta_M$ from $\Theta_{k+1}$) -- ($\text{MINT}_4(t)$).

$\text{MINT}_1$ is the minimum time for partial trajectory $\mathcal{C}$ and is very similar to the minimum time shown in Eqn. (4.56):

$$\text{MINT}_1(t) = (k-1) / B(t)$$  \hspace{1cm} (4.69)

where

$$B(t) = \text{Min} \ [B_{k-1}, b_k(t)]$$  \hspace{1cm} (4.70)

and

$B_{k-1}$ is defined as in Eqn. (4.56),

$b_k(t)$ is the maximum time-scaling factor at time $t$ for $\mathcal{C}_k(t)$ defined as $b_k(t) = \text{Min} \ b_{k,j}(t)$, and

$b_{k,j}(t)$ is computed from Eqn. (4.9).

$\text{MINT}_2(t)$ is found as:

$$\text{MINT}_2(t) = t / B(t).$$  \hspace{1cm} (4.71)
\[ MINT_3(t) \text{ is found as:} \]
\[
MINT_3(t) = \max \left[ \frac{1-t}{B(t)}, \max_j \frac{\theta_{k+1,j} - C_{k_j}(t)}{\theta_j} \right] \quad (4.72)
\]

where \( \sigma \) is the sign of \( \theta_{k+1,j} - C_{k_j}(t) \).

Note that the second term in the bracket is the minimum time for reaching knot point \( \theta_{k+1} \) from the current position \( C_k(t) \) based only upon the joint velocity limits.

\( MINT_4(t) \) is zero if \( \theta_{k+1} \) is the destination point.

Otherwise:

Since it is necessary to travel at least one more segment to reach the destination point, the lowest estimate of the minimum time for the segment is found by using a method similar to that used in \( MINT_3(t) \):

\[
MINT_4(t) = \max \left[ \frac{1}{B(t)}, \max_j \frac{\theta_{k,j} - \theta_{k+1,j}}{\theta_j} \right] \quad (4.73)
\]

Hence,

\[
LB(t) = \sum_{i=1}^{4} MINT_i(t) \quad (4.74)
\]

If there are no limits in the joint velocity:

\[
LB(t) = \frac{(k+u)}{B(t)}, \quad (4.75)
\]

where \( u = 0 \) if \( q_{k+1} = M \) (i.e., \( \theta_{k+1} \) is the destination point), otherwise \( u = 1 \).
4.5.1.C. **Branching rules**

The efficiency of a branch-and-bound algorithm is largely dependent on how quickly good solutions are found. Since these solutions provide upper bounds, many partial paths can be fathomed at an early stage. This section describes two branching rules.

**C.1 Branching rule-1 (BR-1)**

Branching rule-1 is adopted for schemes 1 and 2 based on the early experiments which indicated an optimal solution is generally found in shallow levels. With this rule, the next node for branching is chosen as the last node of the partial path having the lowest lower bound in the previous level. An example search tree with M of 5 is shown in the following figure. In this example, it is assumed that none of partial paths is fathomed to show the search sequence of the BR-1. Numbers on the solid lines indicate the sequence numbers of the partial paths to be evaluated. Notice that the search starts from the last node through the next node of the branching node in the same level, which accelerates finding of an upper bound in an early stage.
C.2 **Branching rule-2 (BR-2)**

BR-2 is essentially a backward depth-first search, and is used for scheme 3. This is based on the early experiments indicating the optimal solution in general exists in deep levels. As in BR-1, the branching node is chosen as the last node of the partial path having the lowest lower-bound in the previous level.

---

**Fig. 9** Example search tree for branching rule-1
Fig. 10 Example search tree for branching rule-2
4.5.1.D. **Summary of phase-1 algorithm**

**Step 1**
Set initial upper bound as 9999 seconds.

**Step 2**
Select one of the remaining partial paths.
Construct all possible augmented paths from the branching node based on the branching rule chosen.

**Step 3**
For each augmented path, construct polynomials for the augmented segment based on the construction scheme.

**Step 4**
Apply the fathoming rules to the augmented paths.
If an augmented path is a complete path, and the minimum-time is lower than the current upper bound,
   a) update the upper bound with the new minimum time,
   b) record the complete path as the optimal path, and
   c) apply FR-3 to all remaining partial paths.

**Step 5**
If there are no remaining paths, stop.
The current upper bound is the minimum-time upper bound with the path recorded in the optimal path buffer. Otherwise, go to Step 2.
4.5.2 **Phase 2 algorithm**

Phase 2 relaxes the restriction on the location of knot points imposed in phase 1, and finds the minimum-time trajectory by adjusting the number of knot points and the locations of the knot points. The phase 2 algorithm consists of three processes; modification, doubling, and halving.

4.5.2.A. **Modification Process (MP)**

The modification process is to find the optimal location of the given number of knot points, and is essentially based upon gradient search methods. Let \( \Theta \) be a matrix of dimension \( n \times M \) representing the locations of the \( M \) knot points in joint space,

\[
\Theta = [ \Theta_1, ..., \Theta_M ].
\]  

(4.76)

By the procedure\(^1\) finding the minimum time, the minimum time \( T' \) for \( \Theta \) can be uniquely determined. For convenience, let \( f \) be the procedure transforming \( \Theta \) to \( T' \),

\[
T' = f(\Theta).
\]  

(4.77)

Let \( V' \) be a matrix of dimension \( n \times M \) representing the steepest gradients (refer to the subsection A.1 of this section for the detailed discussion on this) of \( \Theta \) with respect to \( T' \),

\[
V' = [V'_1, V'_2, ..., V'_M].
\]  

(4.78)

---

12 After constructing polynomial functions by \( \Theta \) with \( T_i = 1 \) for all \( i \), then apply Lemma 4.2 to find the minimum time \( T' \).
Then, the modification process can be described by the following three steps.

**Step 1  Initialization**

Let \( \mathcal{O}_i \) be the initial location matrix, 
\( T_i \) be the minimum-time of \( \mathcal{O}_i \).

**Step 2  Finding steepest gradients**

Find \( \mathcal{V}'(\mathcal{O}_i) \).

If \( |\mathcal{V}'(\mathcal{O}_i)| = 0 \), stop.

\( T_0 = T_i, \mathcal{O}_0 = \mathcal{O}_i, M_0 = M_i \) are optimal values.

Otherwise, go to Step 3.

**Step 3  Line search**

Find \( \lambda \in \lambda \) minimizing \( f(\mathcal{O}_i + \lambda \mathcal{V}'(\mathcal{O}_i)) \).

Let \( T_i = f(\mathcal{O}_i + \lambda \mathcal{V}'(\mathcal{O}_i)) \),
and \( \mathcal{O}_i = \mathcal{O}_i + \lambda \mathcal{V}'(\mathcal{O}_i) \).

Go to Step 2.

### A.1 How to find \( \mathcal{V}'(\mathcal{O}) \)

Since \( f \) cannot be represented in closed form but is only a symbolic transform of numerical methods, the \( \mathcal{V}' \) can be found only numerically by discretizing the direction of the gradient. The direction of the gradient at each knot point is discretized by an increment along each axis. Since there are two directions for each axis, the total number of directions for \( n \) joint is \( 2n+1 \) including zero (i.e., do not move it at all).
For a two-link manipulator, there are five directions at each knot point, one of which corresponds to no change at all (Fig. 11).

Fig. 11 Five directions for two-link manipulator
Let $\mathbf{v}_i(d)$ be the $d^{th}$ direction vector (or gradient) of knot point $i$, and $\mathbf{e}_j$ be the unit vector along the $j^{th}$ axis.

Define

$$
\mathbf{v}_i(d) = \begin{cases} 
0 & \text{for } d = 0 \\
\Delta \mathbf{e}_{d-1} & \text{for } d = 2,\ldots,n+1 \\
-\Delta \mathbf{e}_{d-n-1} & \text{for } d = n+2,\ldots,2n+1 
\end{cases} \quad \text{for } i=2,\ldots,M-1
$$

Since a different direction vector at each knot point yields a different gradient matrix, there are a total of $(2n+1)^{M-1}$ possible gradient matrices for $M$ knot points.

Now, the problem to find the steepest gradient is to find $\mathbf{v}^*(\mathcal{O}_1)$ where $\mathbf{v}^*(\mathcal{O}_1)$ is the optimal solution to the problem to minimize $f(\mathbf{v}(\mathcal{O}_1)+\mathcal{O}_1)$ subject to $\mathbf{v}(\mathcal{O}_1) \in \mathcal{\Omega}$ where $\mathcal{\Omega}$ is the set of $(2n+1)^{M-2}$ possible gradient matrices. This problem can be solved by using similar methodology to that used in phase 1.

Define the stage as the index for the knot point and the state as the index for the gradient vector, then the angular position of state $d$ at stage $s$:

$$
P(s,d) = \theta_s + \mathbf{v}(d). \quad (4.80)
$$

---

13 Because the gradient vector for any knot points except the first and last knot points are $\mathbf{v}_i(d) = \mathbf{v}(d)$, we will drop the subscript $i$ for notational convenience.

14 Note that the first and last knot points should not be changed.
Let $C_k^{**}(.)$ be the trajectory polynomial for the $k^{th}$ interval passing through $P(k,m)$ and $P(k+1,e)$, and $I=\{d(1),\ldots,d(k)\}$ be the state history up to stage $k$ where $d(i)$ is the state at stage $i^{15}$. Then, the performance of $C_k^{**}(.)$ can be evaluated by applying the fathoming rules described in Section 4.5.1.B under the following substitutions:

\[
\begin{align*}
\theta_k &= P(k,m) \\
\theta_{k+1} &= P(k+1,e) \\
C_k(t) &= C_k^{**}(t)
\end{align*}
\] (4.81)

Since the number of knot points is fixed in this case, $MINT_4(t)$ of $LB(t)$ in FR-3 is slightly modified as follows:

\[
MINT_4(t) = \max \left[ \frac{M-k-1}{B(t)}, \max_j \frac{\theta_{M,j} - P_j(k+1,e)}{\theta_j^C} \right]
\] (4.82)

where $B(t)$ is defined in Eqn. (4.70),

$P_j(k+1,e)$ is the $j^{th}$ element value of the $P(k+1,e)$ defined in Eqn. (4.80),

and

$C = \text{sign of } [\theta_{M,j} - P_j(k+1,e)]$.

Note that $(M-k-1)$ in the numerator of the first term in Eqn. (4.82) is the number of remaining segments to travel after finishing the current segment $k$.

15 Note that $d(1)=1$, $d(k)=m$, $d(k+1)=e$, and $d(M)=1$.
Also note that $1 \leq d(i) \leq 2n+1$ for all $i = 2, \ldots, M-1$. 
If there are no limits in the angular velocity, LB(t) is simplified as:

\[ LB(t) = \frac{(M-1)}{B(t)} \]  \hspace{1cm} \text{(4.83)}

If \( C_k^{m*}(.) \) satisfies any of the fathoming rules at any time \( t \in [0,1] \), then entire gradient matrices whose first \((k+1)\) gradient vectors are \( I' = \{ I, d(k+1) \} \) in terms of the state index where \( d(k+1) = e \), are eliminated from further consideration, because they are either infeasible gradient matrices or inferior gradient matrices.

The optimization procedure starts from the first interval, and proceeds through the last interval by repeating forward trace and backward trace until all the states are fathomed.

In the forward trace, let the beginning state for \( k^{th} \) interval is \( m \). A trajectory polynomial for the interval \( (C_k^{m*}(t)) \) is constructed with the \( e^{th} \) state of the \((k+1)^{st}\) knot point based upon the construction scheme. Then, the lower bound of the trajectory polynomial,

\[ LB(k,e) = \text{Max}_{t \in [0,1]} LB(t), \text{ where } LB(t) \text{ is defined in Eqn. (4.74),} \]

Eqn. (4.74) with the modified \( \text{MINT}_e(t) \) defined in Eqn. (4.82), is computed, if none of the fathoming rule is satisfied. If any of the fathoming rules are satisfied for any \( t \in [0,1] \), let \( LB(k,e) = 9999 \), indicating the state \( e \) is fathomed at stage \( k \). Repeat the process for all the possible \( e \). The decision state at the stage \( k \) \( (d^*(k+1)) \) is the state that has the lowerest lower bound; \( d'(k+1) = \{ i | LB(k,i) = \text{Min} LB(k,e) \} \).
There are three possible cases at this point.

Case 1: If LB(k,e) = 9999 for all e indicating none of the following state is feasible, the backward trace is executed with k=k-1.

Case 2: If k=M-1 indicating the current interval is the last interval; the upper bound (UB) and decision index (INDEX) is updated if the maximum of the lower bounds,

$$T = \max_{1 \leq i \leq M-1} \text{LB}(i, d^*(i+1))$$

is less than the current upper bound. Then, the backward trace is executed with k=k-1.

Case 3: Otherwise, the forward trace is continued with k=k+1.

In the backward trace, there are two possible cases.

Case 1: If the stage (k) is zero; indicating the current upper bound is the lowest minimum time, and the decision index (INDEX) is the index for the direction of the steepest gradient matrix, then the search procedure is terminated.

Case 2: Otherwise, let the lower bound of the state d^*(k+1) for the k^th stage,

$$\text{LB}(k,d^*(k+1)) = 9999,$$

to indicate the state d^*(k+1) at stage k is fathomed. If all the LB(k,e) for e=1,...,2n+1
are 9999, then repeat backward trace with 
k=k-1. Otherwise, let \( d'(k+1) \) be the state 
that has the lowest lower bound, 
\[
d'(k+1) = \{ i \mid \text{LB}(k,i) = \text{Min } \text{LB}(k,e) \}\,.
\]
then forward trace is executed with \( k=k+1 \).

The following steps summaries the procedure.

**Step 1 (Initialization)**

\[
k = 0
\]
\[
\text{UB} = T_1 \quad \text{where } T_1 = f(\emptyset_1)
\]
\[
d'(1) = 1, \quad d'(M) = 1
\]
Go to Step 2

**Step 2 (Forward trace)**

\[
k = k + 1
\]
\[
m = d'(k)
\]
For \( e = 1, \ldots, 2n+1 \)

1. Compute \( C_k^{**}(t) \)

2. Apply fathoming rules. If fathomed, let 
\[
\text{LB}(k,e) = 9999
\]
where \( \text{LB}(k,e) = \text{Max}_{t\in\{0,1\}} \text{LB}(t) \), 
and \( \text{LB}(t) \) is defined in Eqn. (4.74) with 
modified \( \text{MINT}_i(t) \) defined in Eqn. (4.82).

If \( \text{LB}(k,e) = 9999 \) for all \( e \), go to step 4.
If \( k = M-1 \), go to step 3.
Otherwise, let \( d'(k+1) = \{ i \mid \text{LB}(k,i) = \text{Min } \text{LB}(k,e) \} \),

---

16 Note that \( e = 1 \) only when \( k = M-1 \).
then go to step 2.

**Step 3 (Last segment)**

\[
T = \max_{1 \leq i \leq M-1} \text{LB}(i, d'(i+1))
\]

If \( T < UB \), replace UB by T, and let INDEX = \( \{d'(i), i=1,\ldots,M\} \).

Go to step 4.

**Step 4 (Backward trace)**

\[ k = k-1 \]

If \( k = 0 \), go to step 5.

Let \( \text{LB}(k, d'(k+1)) = 9999 \).

If \( \text{LB}(k, e) = 9999 \) for all \( e=1,\ldots,2n+1 \),

\[ \text{go to step 4.} \]

Otherwise,

let \( d'(k+1) = \{i | \text{LB}(k, i) = \min_e \text{LB}(k, e) \} \)

and go to step 2.

**Step 5 (Optimal solution)**

Form \( V'(\emptyset_i) \) based on INDEX:

\[
V'(\emptyset_i) = [V(d'(1)), V(d'(2)), \ldots, V(d'(M))]
\]

where \( V(d'(1)) \) and \( V(d'(M)) \) are 0.

If \( d'(i) = 1 \) for all \( i=1,\ldots,M \), then \( V'(\emptyset_i) \) is a null matrix indicating \( \emptyset_i \) is the optimal location for \( M \) knot points and UB is the minimum time.
(A.2) How to find $\lambda^*$

Let $E(\emptyset)$ be the factored matrix of $V(\emptyset)$ by $A$;

$$E(\emptyset) = A^{-1} V(\emptyset), \text{ and let } f(A) = f(\emptyset + AE(\emptyset)),$$

then

$$f(\kappa) = f(\emptyset + \kappa E(\emptyset)) = f(\emptyset + \kappa V(\emptyset)), \text{ where } \kappa = \lambda A^{17}.$$

Thus, the problem to find $\lambda^*$ is the same as that to find $\kappa^*$. A one-dimensional search is needed. There are many one-dimensional search techniques, such as Fibonacci, Golden Section, and so on. However, it is hard to apply these techniques directly to our problem because of the following:

a) Unimodality of $f(\kappa)$ is not guaranteed, and

b) The initial search interval is hard to find$^{18}$.

We solved these problems by modifying the Golden Section method based on the characteristics of our problem. The proposed method utilizes the values of $f(0)$ and $f(\Delta)^{19}$, and earlier experimental results indicating that $\kappa^*$, in general, is not located far from 0.

Let the lower (upper) value of the initial search interval be $S_0 (S_2)$, and $S_1$ be a value in-between the interval. In the procedure for finding the initial search interval, a set of values for $S_0$, $S_1$, and $S_2$ is found which satisfies the

17 Note that $A$ is the step size which is given.

18 One might set the initial interval as $[-\infty, \infty]$, but this costs computation time to find $\kappa^*$ within a specified accuracy.

19 $f(0)$ and $f(\Delta)$ are obtained when $V^*(\emptyset)$ is found.
following with the smallest value of $S_1$:

$$f(S_1) < f(S_0), \text{ and } f(S_1) < f(S_2)$$

where

$$S_1 = S_0 + G(S_2 - S_0), \text{ and } G \text{ is either } 0.382 \text{ or } 0.618$$

depending on the search counter (ICOUNT).

In the procedure for Golden Section search, the initial search interval found by the procedure described above is iteratively reduced until the interval of uncertainty is less than a specified magnitude $\epsilon$.

The details of the procedures for finding initial search interval and $\delta'$ are explained in the following steps:

**Procedure for finding initial search interval $[S_0, S_2]$**

**Step 1 (Initialization)**

Let $S_0=0$, $S_1=\Delta$, and ICOUNT=0.

**Step 2 (Determination of $S_2$)**

Let ICOUNT=ICOUNT+1.

If ICOUNT is odd,

let ICASE=1,

$$S_2=S_1+\Delta(0.618/0.382).$$

Otherwise, let ICASE=2

$$S_2=S_1+\Delta.$$ 

Go to step 3.
Step 3 (Termination)

If \( f(S_2) > f(S_1) \), stop and go to the procedure for Golden Section.

Otherwise, let \( S_0 = S_1 \) and \( S_1 = S_2 \),
then go to step 2.

Procedure for Golden Section Search

Step 1 (Determination of \( S_3 \))

If ICASE=1, \( S_3 = 0.618(S_2 - S_1) + S_0 \),
go to step 2.

Otherwise, \( S_3 = 0.382(S_2 - S_0) + S_0 \),
go to step 3.

Step 2

If \( f(S_3) < f(S_1) \), let \( S_0 = S_1 \) , \( S_1 = S_3 \), and
ICASE=1.

Otherwise, let \( S_2 = S_3 \) and ICASE=2.
Go to step 4.

Step 3

If \( f(S_3) < f(S_1) \), let \( S_2 = S_1 \) , \( S_1 = S_3 \), and
ICASE=2

Otherwise, let \( S_0 = S_3 \) and ICASE=1.
Go to step 4.

Step 4 (Termination)

If \( (S_2 - S_0) < \delta \) where \( \delta \) is a specified accuracy,
then stop, \( \lambda' = S_1 \) or \( \lambda' = S_1 / \Delta \).

Otherwise, go to step 1.
4.5.2.B. Number of knot points (NKP)

In the modification process, we have assumed the NKP is fixed and that optimization is made only for the location of knot points. The performance of the trajectory function (minimum time), however, is affected by the NKP according to the results of experiments.

In general, the performance is improved as NKP increases. This may be explained because the polynomial trajectory with bigger NKP has a better chance to fit the unknown true-minimum-time trajectory (TMTT), or the TMTT can be more closely approximated by the polynomial trajectory with bigger NKP. In this section, we describe the methodology for adjusting the NKP.

B.1 Doubling process (DP)\textsuperscript{20}

A change in NKP yields a new problem for finding an initial feasible solution, particularly due to the path constraint\textsuperscript{21}. Note that the modification process always starts with a feasible solution and finds the optimal location of knot points.

\textsuperscript{20} 'Doubling', and 'Halving' are in terms of the number of segments rather than NKP.

\textsuperscript{21} Infeasible solutions due to actuator and/or angular velocity limits can be made feasible by increasing travelling time without modifying the location of knot points. In this context, the infeasibility in this discussion is due to joint limits and/or path constraints.
If the initial solution is infeasible, the search should start from scratch to find an initial feasible solution, which is very time consuming. One way to avoid this problem is using a previously found feasible solution as an initial solution. The method presented for finding the initial feasible solution is based upon the property described in Corollary 4.3.

Suppose a known feasible trajectory is formed by NKP of \( M_i \) with locations of \( \theta_i \), \( i=1,\ldots,M_i \) and the trajectory functions

\[
C_i(t), \quad i=1,\ldots,M_i-1, \quad 0 \leq t \leq 1.
\]

Then, almost the same trajectory can be formed by using an NKP of \((2M_i-1)\) with location of \( \bar{\theta}_i \), \( i=1,\ldots,M_0 \), where

\[
\begin{align*}
\bar{\theta}_{2i-1} &= \theta_i, \quad i=1,\ldots,M_i \\
\bar{\theta}_{2i} &= C_i(1/2), \quad i=1,\ldots,M_i-1 \\
M_0 &= 2M_i - 1,
\end{align*}
\]

if the order of the polynomials for the entire set of segments is not changed by the change of NKP.

The new polynomial trajectory functions \( D_i(t) \), \( i=1,\ldots,M_0-1 \), are:

\[
\begin{align*}
D_{2i-1}(t) &= C_i(t) \\
D_{2i}(t) &= C_i(t+1/2)
\end{align*}
\]

where \( 0 \leq t \leq 1/2, \quad i=1,\ldots,M_i-1. \)
The doubling process is explained by the following two steps:

**Step 1**

Read $C_i(t)$, $0 \leq t \leq 1$, $i = 1, \ldots, M_i - 1$

and $\theta_i$, $i = 1, \ldots, M_i$.

**Step 2**

Find $\varnothing_i$ $i = 1, \ldots, 2M_i - 1$ as follows:

1. $\varnothing_{2j-1} = \varnothing_j$ $j = 1, \ldots, M_i$
2. $\varnothing_{2k} = C_k(1/2)$ $k = 1, \ldots, M_i - 1$.

Note, however, that none of our trajectory schemes can exactly fit the original trajectory by the knot points found from the doubling process, but do fit closely. This might cause failure to find an initial feasible solution in case the path tolerance is very tight. For this case, a feasibility process is applied.

**B.2 Feasibility process (FP)**

FP is similar to the method for finding $V'(\emptyset)$. The differences are:

a) FR-3 is not applied for FP

b) When a feasible solution is found, FP stops

c) If FP fails to find a feasible solution, it is repeated up to maximum number of times (MAX) specified with the increase in the step size $\Delta$.

A flow chart of FP is shown in Fig. 12.
Read $\emptyset$ from DP

Form $C_i \forall i$

Apply FR-1 & FR-2

Feasible?

No

$ICOUNT=1$

$\Delta=ICOUNT\times\Delta$

Apply the procedure for finding $v'(\emptyset)$ w/o FR-3

Is Step 3 of the procedure reached?

No

$ICOUNT=ICOUNT+1$

$ICOUNT:\text{MAX}$

Yes

Success

Stop

Fail

Fig. 12 Flow chart of FP
B.3. **Halving process (HP)**

HP is for the stopping rule of the phase 2 algorithm. The phase 2 algorithm stops when the following are met:

a) No further reduction of minimum time is realized by increasing the NKP, or the magnitude of reduction is less than the specified magnitude $z$,

b) Minimum time with maximum NKP is stabilized within $z$.

Suppose $M_i$ is the NKP and $\theta$ is the location matrix consisting of $\theta_i$, $i=1,..,M_i$ when the stopping rule a) is met. Then the maximum NKP is $(M_i+1)/2$. For stopping rule b), HP generates an initial solution for the location of $(M+1)/2$ knot points as follows:

$$\phi_i = \theta_{2i-1} \quad i=1,\ldots,M_o$$

where $M_o = (M_i+1)/2$. (4.85)

As previously discussed in DP, this initial solution can form a trajectory which is very close to the original trajectory formed by $\theta$. For the case of infeasibility, FP is applied. Then, MP is applied. Note that the starting solution of the MP (5 in Fig. 13) which is preceded by HP may be different from the starting solution of the MP before DP (1 in Fig. 13) even though the NKP is the same. This can reduce the error due to discretization$^{22}$ when $T'$ is computed.

---

$^{22}$ Note that we limited the number of directions for each gradient vector to five, for example, for the two-link manipulator, but it is essentially continuous space.
in MP, which is what stopping rule 2 is for. In other words, HP is used as a means for generating a different initial feasible solution. Fig. 13 explains stopping rule 2.23

![Diagram showing stopping rule 2]

**Legend**

1 = Starting solution #1 at Terminal
2 = Final solution #1 at Terminal (MP)
3 = Starting solution at Doubling Station (DP/FP)
4 = Final solution at Doubling Station (HP)
5 = Starting solution #2 at Terminal (HP/FP)
6 = Final solution #2 at Terminal (MP)

**Fig. 13 Stopping rule 2**

23 Note that the NKP of the terminal in Fig. 13 is the maximum (the optimal) number of knot points through which the reduction of the minimum time is accomplished by increasing the NKP, since stopping rule 1 is met when the stopping rule 2 is applied.
4.5.2.C. **Summary of the phase 2 algorithm**

**Step 1 (Initialization)**

1. Read $T^*$, $M^*$, $\emptyset^*$ from phase 1 (① in Fig. 13)
2. Let FLAG=1 to indicate the search process continues without checking in Step 2
3. If $M^*$=2, go to step 2.
4. Otherwise:
   a. Let $T_i=T^*$, $M_i=M^*$, $\emptyset_i=\emptyset^*$.
   b. Call MP (the result of MP is ② in Fig. 13).
   c. If $T_0 < T^*$,
      let $T'=T_0$, $M'=M_0$, $\emptyset'=\emptyset_0$.
   d. Compute $\text{DIFF} = (T_o-T^*) / T^*.$
   e. If $\text{DIFF} \leq z$, the reduction in terms of the minimum time is not significant:
      stop; $T^*$, $M^*$, $\emptyset^*$ are optimal.
   Otherwise, go to step 2.

**Step 2 (Terminal)**

If FLAG=1, search continues. Record the following solution as ② in Fig. 13 before leaving the Terminal:

let $T_{\text{LEAVE}} = T^*$, $M_{\text{LEAVE}} = M^*$, $\emptyset_{\text{LEAVE}} = \emptyset^*$,
then go to step 3.

If FLAG=0, search continues if the reduction made in the previous iteration is significant:

compute $\text{DIFF} = |T_{\text{LEAVE}} - T_{\text{ARRV}}| / T_{\text{LEAVE}}.$
If DIFF $\leq z$, the reduction is not significant:
stop; $T^*, M^*, \emptyset^*$ are optimal values.

Otherwise, the reduction is significant. Record the following solution as $\text{2}$ in Fig. 13 before leaving the Terminal):
let $T_{\text{LEAVE}} = T_{\text{ARIV}}, M_{\text{LEAVE}} = M_{\text{ARIV}},$ and
$\emptyset_{\text{LEAVE}} = \emptyset_{\text{ARIV}},$ then go to Step 3.

Step 3 (DP & FP & MP)
1. Let $M_i = M_{\text{LEAVE}}, \emptyset_i = \emptyset_{\text{LEAVE}}.$
2. Get an initial location of knot points by (4.84).
3. Call FP and let the result be $T_o, M_o, \emptyset_o$.

If success, an initial feasible solution at doubling station has been found. Record the following solution as $\text{3}$ in Fig. 13.

a. Let $\text{FLAG}=1,$ and let $T_i = T_o, M_i = M_o, \emptyset_i = \emptyset_o$.

b. Apply 4.b, 4.c, 4.d in Step 1. The result of MP in 4.b of Step 1 is $\text{4}$ in Fig. 13.

c. If DIFF $\geq z$, the reduction is significant.

Go to Step 2 (note: Doubling Station will become a new Terminal).

Otherwise, the reduction is not significant.

Thus, go to Step 4 to return to Terminal.

If failure, an initial feasible solution at doubling station has not been found:
Go to Step 4 to return to Terminal.
Step 4 (HP & FP & MP)

1. Let $T_1 = T_0$, $M_1 = M_0$, $\emptyset_1 = \emptyset_0$.

2. Get an initial location of knot points by (4.85).

3. Call FP, and let the result be $T_o, M_o, \emptyset_o$.

If success, an initial feasible solution at terminal has been found. Record the following solution as 5 in Fig. 13.

a. Let $T_1 = T_0$, $M_1 = M_0$, $\emptyset_1 = \emptyset_0$.

b. Apply 4.b, 4.c, 4.d in Step 1. The result of MP in 4.b of Step 1 is 6 in Fig. 13.

c. If $\text{DIFF} \geq z$, the reduction is significant.

   Let $\text{FLAG}=1$, then go to Step 2
   (note: Terminal will not be changed).

   Otherwise, the reduction is not significant.

   Let $T_{\text{ARIV}} = T_0$, $M_{\text{ARIV}} = M_o$, $\emptyset_{\text{ARIV}} = \emptyset_o$,
   $\text{FLAG}=0$, then go to Step 2
   (note: Terminal will not be changed).

If failure, an initial feasible solution at Terminal has not been found. In this case, 6 is the same as 2 in Fig. 13.

Let $T_{\text{ARIV}} = T_{\text{LEAVE}}$, $M_{\text{ARIV}} = M_{\text{LEAVE}}$, $\emptyset_{\text{ARIV}} = \emptyset_{\text{LEAVE}}$,
$\text{FLAG}=0$, go to Step 2
(note: Terminal will not be changed).
5.1 Two-link robot

The algorithms developed are tested using a two-link robot simulated on an IBM 3081 computer at the Ohio State University.

Fig. 14 Two-link robot

---

24 Refer to Section 3.2 for the transform equations; direct kinematics, inverse kinematics, and dynamics equations.
The geometry of the robot is shown in Table 1.

Table 1 Geometry of two-link robot

<table>
<thead>
<tr>
<th></th>
<th>Link 1</th>
<th>Link 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mass  (Kg)</td>
<td>14.6</td>
<td>14.6</td>
</tr>
<tr>
<td>Length (m)</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>Inertia (Kg·m²)</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Actuator limits for joint motors 1 and 2 are:

\[-95 \text{ N·m} \leq \tau_1 \leq 95 \text{ N·m}\]
\[-27 \text{ N·m} \leq \tau_2 \leq 27 \text{ N·m}.\]

The limits of the velocity and acceleration for this robot in a conventional trajectory scheme which follows constant maximum acceleration, constant velocity, and constant deceleration along a straight line path in Cartesian space (refer to Fig. 15) are 0.75 m/sec and 1.2 m/sec², respectively.

25 These values are taken from Bobrow [5] with the following conversion factors:
1 lb = 0.454 Kg, 1 ft = 0.304 m, 1 ft·lb = 1.3558 N·m.
5.2 Example paths

To compare the performance of the three schemes for representing the trajectory functions for the two phases, ten examples were chosen. Examples one through eight were chosen for the basic motion of the robot arm on vertical, horizontal, and diagonal paths. Examples nine and ten are for motions on a combination of paths.

Reference paths and reference points of ten examples including the conventional minimum time (from the trapezoidal velocity profile) for these paths are shown in Table 2.
Table 2  Ten example paths and their conventional minimum time

<table>
<thead>
<tr>
<th>Ex #</th>
<th>Reference Path</th>
<th>Reference Points</th>
<th>Conventional Minimum Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>( R_1 = (0.3, 0.5) )</td>
<td>1.96 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( R_2 = (0.3, -0.5) )</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>( R_1 = (0.3, -0.5) )</td>
<td>1.96 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( R_2 = (0.3, 0.5) )</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>( R_1 = (0.6, 0.0) )</td>
<td>1.00 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( R_2 = (0.3, 0.0) )</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>( R_1 = (0.3, 0.0) )</td>
<td>1.00 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( R_2 = (0.6, 0.3) )</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>( R_1 = (0.1, 0.5) )</td>
<td>1.93 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( R_2 = (0.5, -0.3) )</td>
<td></td>
</tr>
</tbody>
</table>
(Table 2 continued)

<table>
<thead>
<tr>
<th>Ex #</th>
<th>Reference Path</th>
<th>Reference Points</th>
<th>Conventional Minimum time</th>
</tr>
</thead>
</table>
| 6    | ![Diagram](image)
|      | $R_1 = (0.5, -0.3)$ |
|      | $R_2 = (0.1, 0.5)$ |
|      | 1.93 sec       |
| 7    | ![Diagram](image) |
|      | $R_1 = (0.1, -0.5)$ |
|      | $R_2 = (0.5, 0.3)$ |
|      | 1.93 sec       |
| 8    | ![Diagram](image) |
|      | $R_1 = (0.5, 0.3)$ |
|      | $R_2 = (0.1, -0.5)$ |
|      | 1.93 sec       |
| 9    | ![Diagram](image) |
|      | $R_1 = (0.0, 0.5)$ |
|      | $R_2 = (0.4, 0.2)$ |
|      | $R_3 = (0.4, -0.2)$ |
|      | $R_4 = (0.0, -0.5)$ |
|      | 3.74 sec       |
| 10   | ![Diagram](image) |
|      | $R_1 = (0.0, -0.5)$ |
|      | $R_2 = (0.4, -0.2)$ |
|      | $R_3 = (0.4, 0.2)$ |
|      | $R_4 = (0.0, 0.5)$ |
|      | 3.74 sec       |
5.3 Results

Six computer programs have been developed in FORTRAN, one for each phase and trajectory scheme combination. The names of the codes are listed in Table 3.

Table 3  Computer code names

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Phase 1</th>
<th>Phase 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheme 1</td>
<td>PGM 135</td>
<td>PGM 235</td>
</tr>
<tr>
<td>Scheme 2</td>
<td>PGM 134</td>
<td>PGM 234</td>
</tr>
<tr>
<td>Scheme 3</td>
<td>PGM 144</td>
<td>PGM 244</td>
</tr>
</tbody>
</table>

The performance of these codes for the ten example paths with various path tolerances is summarized in Table 4. Note that $T_1^*$ and $T_2^*$ indicate the minimum time (in seconds) of Phase 1 and Phase 2, respectively, and $\&$ indicates the path tolerance (in meters).

26 The M value used for phase 1 in this experiment is 30.
Table 4: Computer results of the ten example paths
(M = 30)

<table>
<thead>
<tr>
<th>Ex #</th>
<th>E</th>
<th>Scheme 1</th>
<th></th>
<th>Scheme 2</th>
<th></th>
<th>Scheme 3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>T&lt;sub&gt;1&lt;/sub&gt;</td>
<td>T&lt;sub&gt;2&lt;/sub&gt;</td>
<td>T&lt;sub&gt;1&lt;/sub&gt;</td>
<td>T&lt;sub&gt;2&lt;/sub&gt;</td>
<td>T&lt;sub&gt;1&lt;/sub&gt;</td>
<td>T&lt;sub&gt;2&lt;/sub&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.01</td>
<td>2.12</td>
<td>2.03</td>
<td>1.70</td>
<td>1.68</td>
<td>1.02</td>
<td>0.84</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>1.12</td>
<td>1.95</td>
<td>1.44</td>
<td>1.08</td>
<td>1.02</td>
<td>0.82</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
<td>2.12</td>
<td>1.94</td>
<td>1.11</td>
<td>1.04</td>
<td>0.91</td>
<td>0.81</td>
</tr>
<tr>
<td></td>
<td>0.04</td>
<td>2.12</td>
<td>1.89</td>
<td>1.11</td>
<td>1.03</td>
<td>0.91</td>
<td>0.80</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>2.12</td>
<td>1.88</td>
<td>1.11</td>
<td>1.03</td>
<td>0.91</td>
<td>0.80</td>
</tr>
<tr>
<td>2</td>
<td>0.01</td>
<td>1.23</td>
<td>1.18</td>
<td>1.46</td>
<td>1.46</td>
<td>1.08</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>1.23</td>
<td>1.16</td>
<td>1.32</td>
<td>1.05</td>
<td>1.08</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
<td>1.23</td>
<td>1.16</td>
<td>1.32</td>
<td>1.05</td>
<td>1.04</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>0.04</td>
<td>1.23</td>
<td>1.16</td>
<td>1.32</td>
<td>1.05</td>
<td>1.04</td>
<td>0.73</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>1.23</td>
<td>1.16</td>
<td>1.26</td>
<td>1.02</td>
<td>1.04</td>
<td>0.73</td>
</tr>
<tr>
<td>3</td>
<td>0.01</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.45</td>
<td>0.47</td>
<td>0.44</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.46</td>
<td>0.47</td>
<td>0.44</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.46</td>
<td>0.47</td>
<td>0.44</td>
</tr>
<tr>
<td></td>
<td>0.04</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.45</td>
<td>0.47</td>
<td>0.44</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>0.51</td>
<td>0.51</td>
<td>0.51</td>
<td>0.45</td>
<td>0.47</td>
<td>0.44</td>
</tr>
<tr>
<td>4</td>
<td>0.01</td>
<td>0.46</td>
<td>0.46</td>
<td>0.50</td>
<td>0.50</td>
<td>0.44</td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>0.46</td>
<td>0.46</td>
<td>0.50</td>
<td>0.50</td>
<td>0.44</td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
<td>0.46</td>
<td>0.46</td>
<td>0.50</td>
<td>0.50</td>
<td>0.44</td>
<td>0.39</td>
</tr>
<tr>
<td></td>
<td>0.04</td>
<td>0.46</td>
<td>0.46</td>
<td>0.50</td>
<td>0.50</td>
<td>0.44</td>
<td>0.39</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>0.46</td>
<td>0.46</td>
<td>0.50</td>
<td>0.50</td>
<td>0.44</td>
<td>0.39</td>
</tr>
<tr>
<td>5</td>
<td>0.01</td>
<td>3.27</td>
<td>3.14</td>
<td>1.80</td>
<td>1.53</td>
<td>1.00</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>2.12</td>
<td>1.75</td>
<td>1.45</td>
<td>1.36</td>
<td>1.00</td>
<td>0.80</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
<td>1.88</td>
<td>1.41</td>
<td>1.45</td>
<td>1.36</td>
<td>1.00</td>
<td>0.80</td>
</tr>
<tr>
<td></td>
<td>0.04</td>
<td>1.88</td>
<td>1.28</td>
<td>1.26</td>
<td>1.06</td>
<td>1.00</td>
<td>0.80</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>1.87</td>
<td>1.28</td>
<td>1.24</td>
<td>1.05</td>
<td>1.00</td>
<td>0.79</td>
</tr>
<tr>
<td>6</td>
<td>0.01</td>
<td>1.21</td>
<td>1.14</td>
<td>1.45</td>
<td>1.31</td>
<td>1.04</td>
<td>0.92</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>1.21</td>
<td>1.10</td>
<td>1.32</td>
<td>1.03</td>
<td>0.98</td>
<td>0.91</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
<td>1.21</td>
<td>1.10</td>
<td>1.32</td>
<td>1.03</td>
<td>0.95</td>
<td>0.83</td>
</tr>
<tr>
<td></td>
<td>0.04</td>
<td>1.21</td>
<td>1.10</td>
<td>1.21</td>
<td>1.01</td>
<td>0.95</td>
<td>0.81</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>1.21</td>
<td>1.10</td>
<td>1.21</td>
<td>1.01</td>
<td>0.95</td>
<td>0.80</td>
</tr>
<tr>
<td>7</td>
<td>0.01</td>
<td>1.81</td>
<td>1.81</td>
<td>0.98</td>
<td>0.94</td>
<td>0.79</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>1.15</td>
<td>1.00</td>
<td>0.98</td>
<td>0.91</td>
<td>0.79</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
<td>1.15</td>
<td>0.99</td>
<td>0.98</td>
<td>0.91</td>
<td>0.79</td>
<td>0.72</td>
</tr>
<tr>
<td></td>
<td>0.04</td>
<td>1.12</td>
<td>0.86</td>
<td>0.98</td>
<td>0.91</td>
<td>0.79</td>
<td>0.70</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>1.12</td>
<td>0.86</td>
<td>0.98</td>
<td>0.91</td>
<td>0.79</td>
<td>0.70</td>
</tr>
</tbody>
</table>
(Table 4 continued)

<table>
<thead>
<tr>
<th>Ex #</th>
<th>$\mathcal{E}$</th>
<th>Scheme 1</th>
<th>Scheme 2</th>
<th>Scheme 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$T_1^*$</td>
<td>$T_2^*$</td>
<td>$T_1^*$</td>
</tr>
<tr>
<td>8</td>
<td>0.01</td>
<td>1.54</td>
<td>1.49</td>
<td>1.21</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>1.54</td>
<td>1.43</td>
<td>1.15</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
<td>1.54</td>
<td>1.40</td>
<td>1.15</td>
</tr>
<tr>
<td></td>
<td>0.04</td>
<td>1.54</td>
<td>1.37</td>
<td>1.15</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>1.50</td>
<td>1.35</td>
<td>1.10</td>
</tr>
<tr>
<td>9</td>
<td>0.01</td>
<td>$t^{27}$</td>
<td>$t^{27}$</td>
<td>3.54</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>5.19</td>
<td>4.89</td>
<td>3.32</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
<td>2.94</td>
<td>2.52</td>
<td>2.19</td>
</tr>
<tr>
<td></td>
<td>0.04</td>
<td>1.96</td>
<td>1.76</td>
<td>1.87</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>1.96</td>
<td>1.73</td>
<td>1.78</td>
</tr>
<tr>
<td>10</td>
<td>0.01</td>
<td>$t^{27}$</td>
<td>$t^{27}$</td>
<td>3.39</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>3.58</td>
<td>3.47</td>
<td>2.93</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
<td>2.32</td>
<td>2.20</td>
<td>2.55</td>
</tr>
<tr>
<td></td>
<td>0.04</td>
<td>1.97</td>
<td>1.78</td>
<td>2.47</td>
</tr>
</tbody>
</table>

*27 I indicates infeasible solution.*
5.4 Analysis of the results

The results show that the proposed schemes provide a lower minimum time than the conventional trajectory scheme. Except for Scheme 1, even the phase 1 solution gives a superior result to the conventional minimum time. The percentage reduction in time, compared to the conventional time (refer to Table 5), is shown to vary from 5% up to 69%, depending on the trajectory scheme and the magnitude of the path tolerance. In particular, the percentage reduction of the phase 2 solution value for scheme 3 turns out to be over 50% in all the cases.

This significant reduction comes from the fact that the conventional method does not fully consider the dynamics of the robot and takes the most conservative maximum of the velocity and acceleration which can be realizable in any configuration of the robot arm throughout the movement. As a result, the robot is underutilized and the travelling time is longer than necessary. The velocity profiles of example 2 for the conventional scheme and the proposed scheme are displayed in Fig. 16.
Table 5 Percent reduction of time to the conventional scheme

<table>
<thead>
<tr>
<th>Ex #</th>
<th>( \xi )</th>
<th>Scheme 2</th>
<th>Scheme 3</th>
<th>Ex #</th>
<th>Scheme 2</th>
<th>Scheme 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Ph1 Ph2</td>
<td>Ph1 Ph2</td>
<td></td>
<td>Ph1 Ph2</td>
<td>Ph1 Ph2</td>
</tr>
<tr>
<td>0.01</td>
<td>13</td>
<td>14</td>
<td>48 57</td>
<td>0.01</td>
<td>25</td>
<td>32</td>
</tr>
<tr>
<td>0.02</td>
<td>27</td>
<td>45</td>
<td>48 58</td>
<td>0.02</td>
<td>32</td>
<td>47</td>
</tr>
<tr>
<td>0.03</td>
<td>43</td>
<td>47</td>
<td>54 59</td>
<td>0.03</td>
<td>32</td>
<td>47</td>
</tr>
<tr>
<td>0.04</td>
<td>43</td>
<td>47</td>
<td>54 59</td>
<td>0.04</td>
<td>37</td>
<td>48</td>
</tr>
<tr>
<td>0.05</td>
<td>43</td>
<td>47</td>
<td>54 59</td>
<td>0.05</td>
<td>37</td>
<td>48</td>
</tr>
</tbody>
</table>

|      |         |          |          |      |          |          |
| 0.01 | 26      | 26       | 45 54    | 0.01 | 49       | 51       |
| 0.02 | 33      | 46       | 45 57    | 0.02 | 49       | 53       |
| 0.03 | 33      | 46       | 47 62    | 0.03 | 49       | 53       |
| 0.04 | 33      | 46       | 47 63    | 0.04 | 49       | 53       |
| 0.05 | 36      | 48       | 47 63    | 0.05 | 49       | 53       |

|      |         |          |          |      |          |          |
| 0.01 | 49      | 53       | 53 56    | 0.01 | 37       | 38       |
| 0.02 | 49      | 54       | 53 56    | 0.02 | 40       | 41       |
| 0.03 | 49      | 54       | 53 56    | 0.03 | 40       | 47       |
| 0.04 | 49      | 55       | 53 56    | 0.04 | 40       | 47       |
| 0.05 | 49      | 55       | 53 56    | 0.05 | 40       | 47       |

|      |         |          |          |      |          |          |
| 0.01 | 50      | 50       | 56 60    | 0.01 | 5        | 11       |
| 0.02 | 50      | 50       | 56 60    | 0.02 | 11       | 13       |
| 0.03 | 50      | 50       | 56 60    | 0.03 | 41       | 57       |
| 0.04 | 50      | 50       | 56 60    | 0.04 | 50       | 53       |
| 0.05 | 50      | 50       | 56 60    | 0.05 | 50       | 53       |

|      |         |          |          |      |          |          |
| 0.01 | 7       | 21       | 48 56    | 0.01 | 9        | 20       |
| 0.02 | 25      | 30       | 48 59    | 0.02 | 22       | 23       |
| 0.03 | 25      | 30       | 48 59    | 0.03 | 32       | 43       |
| 0.04 | 35      | 45       | 48 59    | 0.04 | 34       | 45       |
| 0.05 | 36      | 46       | 48 59    | 0.05 | 52       | 60       |
Fig. 16 Cartesian velocity magnitude plot
(New scheme vs. conventional scheme)
The experimental results indicate that in general the phase 1 solution is not much different from the phase 2 solution when the path tolerance is small. This seems to be obvious considering the center line of the feasible space essentially represents the entire feasible space so that the optimal location of knot points in the phase 2 solution is very close to the reference path. As the path tolerance increases, however, the difference between the phase 1 solution and the phase 2 solution becomes larger.

Without exception, as the path tolerance increases, the minimum time decreases in both the phase 1 and phase 2 solutions of all the trajectory schemes. This can be explained by the fact that the bigger tolerance provides more room to exploit the dynamic effects of the robot for speedy motion. The effect of the path tolerance can be seen from Table 4.

The results also reveal that the minimum-time with scheme 3 is dominant over scheme 1 and scheme 2 in all cases, indicating the significance of the trajectory construction method. The dominance of scheme 3 seems to be due to the following two differences between scheme 3 and the other two schemes:

a) Imposing one more boundary condition (the final position constraint of the following trajectory segment) on the trajectory polynomial of the current segment can prevent abrupt changes in the pattern of robot motion when switching the trajectory polynomial
of the current segment to that of the following segment.

In other words, the extra boundary condition provides predictiveness to the current trajectory segment. The following figure shows the boundary conditions covered by each trajectory segment (refer to section 4.4.3 for details).

![Diagram showing boundary conditions](image)

The solid (broken) bar indicates the boundary conditions covered by the trajectory polynomial for the current (following) segment.

Fig. 17 The extra boundary condition in scheme 3

b) The uniform order in scheme 3 (all quartic polynomials) keeps the pattern of robot motion from abrupt changes throughout the entire travel.
On the other hand, neither scheme 1 nor scheme 2 imposes the extra boundary condition (refer to Fig. 18), and the order of the trajectory polynomials is not uniform.

![Diagram of trajectory segments](image)

**Fig. 18** Boundary condition in Scheme 1 and Scheme 2

The change of the pattern of robot motion due to the lack of predictiveness and the change in the order of polynomial trajectory thus appears to be undesirable. For instance, the drastic change in the order of the polynomial at the last segment in scheme 1 (from cubic to quintic) sometimes makes it hard to find even a feasible solution to the complex path with tight path tolerance (note the infeasibility for examples 9 and 10 with a path tolerance of 0.01 m in Table 4).

Abrupt changes in the pattern of robot motion often require a high amount of torque, resulting in the slowdown of
the entire motion to satisfy the torque constraint due to the
dynamic-scaling property. Thus, except for the last segment(s)
of the trajectory interval, where the high torque is required,
the capability of the joint actuator is not fully utilized.
This can be visualized in the b curve shown in Fig. 19.

A b(t) value of 1 indicates that the torque is fully
utilized, and a b(t) value greater than 1 indicates that the
torque is not fully utilized at time t. A b(t) value less than
1 is not feasible because the torque required at time t is not
realizable from the joint actuator. Fig. 19 plots the b curve
of the minimum-time trajectory for each of the three schemes
for example path 1 with path tolerance of 0.05.

In Fig. 19 the b value of scheme 1 progressively moves
down until it reaches 1 at the last segment indicating that the
capability of the actuator is not utilized much particularly
during the early part of the travel. Scheme 2 shows better
utilization of the capability than scheme 1, and scheme 3
appears to utilize the capability well during most of the
travel. This phenomenon can also be observed in various plots:
angular velocity and acceleration, and Cartesian velocity and
acceleration shown in Figs. 20 through 22.

However, scheme 3 appears to take the longest CPU time,
and scheme 1 in general takes the least CPU time. The CPU time
for the experimental runs is summarized in Table 6.
The following observations can be made from Table 6:

a) Scheme 3 takes the longest CPU time
b) Phase 2 takes a longer time than phase 1
c) The bigger the path tolerance is, the longer CPU time is required.

Observations b) and c) seem to be obvious since the increase in the space to be searched for the location of knot points in general increases the search time.

Observation a) can be explained using Fig. 17 and Fig. 18. In scheme 3, the trajectory polynomial of the current segment is affected by the location of the following knot point. Therefore, it is necessary to construct and evaluate the trajectory polynomial for the current segment as many times as the number of possible locations for the following knot point, while in scheme 1 and scheme 2 this needs to done only once, regardless of the location of the following knot point.

Considering the CPU time and the performance of the solution, the phase 1 algorithm with scheme 2 might be applied for on-line trajectory planning, and the two-phase algorithm with scheme 3 used for off-line trajectory planning. For this reason, the following discussions will focus on the two-phase algorithm with scheme 3.
Fig. 19 b-value plot (example 1, $\varepsilon = 0.05$)
Fig. 20 Angular velocity plots for the three schemes (example 1, $\xi = 0.05$)
Fig. 21 Angular acceleration plot the three schemes (example 1, $\xi = 0.05$)
Fig. 22 Cartesian velocity and acceleration plots (example 1, \( \xi = 0.05 \))
<table>
<thead>
<tr>
<th>Ex #</th>
<th>Scheme 1</th>
<th>Scheme 2</th>
<th>Scheme 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ex</td>
<td>Phase1</td>
<td>Phase2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>0.43</td>
<td>1.12</td>
<td>0.68</td>
</tr>
<tr>
<td>0.02</td>
<td>0.48</td>
<td>1.41</td>
<td>0.82</td>
</tr>
<tr>
<td>1</td>
<td>0.03</td>
<td>0.60</td>
<td>2.19</td>
</tr>
<tr>
<td>0.04</td>
<td>0.73</td>
<td>2.45</td>
<td>1.55</td>
</tr>
<tr>
<td>0.05</td>
<td>0.93</td>
<td>2.68</td>
<td>1.73</td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>0.43</td>
<td>1.03</td>
</tr>
<tr>
<td>2</td>
<td>0.02</td>
<td>0.51</td>
<td>1.36</td>
</tr>
<tr>
<td>0.03</td>
<td>0.56</td>
<td>1.56</td>
<td>1.45</td>
</tr>
<tr>
<td>0.04</td>
<td>0.78</td>
<td>1.82</td>
<td>1.45</td>
</tr>
<tr>
<td>0.05</td>
<td>0.84</td>
<td>1.99</td>
<td>1.45</td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>0.45</td>
<td>0.59</td>
</tr>
<tr>
<td>3</td>
<td>0.02</td>
<td>0.45</td>
<td>1.48</td>
</tr>
<tr>
<td>0.03</td>
<td>0.44</td>
<td>0.95</td>
<td>0.67</td>
</tr>
<tr>
<td>0.04</td>
<td>0.44</td>
<td>0.87</td>
<td>0.67</td>
</tr>
<tr>
<td>0.05</td>
<td>0.44</td>
<td>0.90</td>
<td>0.66</td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>2.10</td>
<td>2.59</td>
</tr>
<tr>
<td>4</td>
<td>0.02</td>
<td>2.10</td>
<td>2.61</td>
</tr>
<tr>
<td>0.03</td>
<td>2.10</td>
<td>2.71</td>
<td>3.96</td>
</tr>
<tr>
<td>0.04</td>
<td>2.10</td>
<td>2.71</td>
<td>3.92</td>
</tr>
<tr>
<td>0.05</td>
<td>2.10</td>
<td>2.82</td>
<td>3.92</td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>0.50</td>
<td>3.88</td>
</tr>
<tr>
<td>5</td>
<td>0.02</td>
<td>0.51</td>
<td>6.68</td>
</tr>
<tr>
<td>0.03</td>
<td>0.51</td>
<td>19.96</td>
<td>3.38</td>
</tr>
<tr>
<td>0.04</td>
<td>0.82</td>
<td>29.87</td>
<td>2.94</td>
</tr>
<tr>
<td>0.05</td>
<td>1.25</td>
<td>9.02</td>
<td>3.79</td>
</tr>
</tbody>
</table>

28 CPU time is in seconds, and CPU time of phase 2 includes the CPU time of phase 1.
(Table 6 continued)

<table>
<thead>
<tr>
<th>Ex #</th>
<th>( \varepsilon )</th>
<th>Scheme 1 Phase1</th>
<th>Phase2</th>
<th>Scheme 2 Phase1</th>
<th>Phase2</th>
<th>Scheme 3 Phase1</th>
<th>Phase2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.44</td>
<td>0.90</td>
<td></td>
<td>0.76</td>
<td>1.79</td>
<td>11.29</td>
<td>13.94</td>
</tr>
<tr>
<td>0.02</td>
<td>0.48</td>
<td>1.12</td>
<td></td>
<td>1.12</td>
<td>7.68</td>
<td>15.10</td>
<td>27.81</td>
</tr>
<tr>
<td>0.03</td>
<td>0.62</td>
<td>1.41</td>
<td></td>
<td>1.41</td>
<td>9.30</td>
<td>21.52</td>
<td>61.18</td>
</tr>
<tr>
<td>0.04</td>
<td>0.83</td>
<td>1.63</td>
<td></td>
<td>1.40</td>
<td>8.85</td>
<td>29.22</td>
<td>82.20</td>
</tr>
<tr>
<td>0.05</td>
<td>0.94</td>
<td>1.72</td>
<td></td>
<td>1.20</td>
<td>3.48</td>
<td>30.69</td>
<td>90.52</td>
</tr>
<tr>
<td>0.01</td>
<td>1.24</td>
<td>2.92</td>
<td></td>
<td>1.17</td>
<td>2.46</td>
<td>20.08</td>
<td>23.50</td>
</tr>
<tr>
<td>0.02</td>
<td>1.70</td>
<td>4.65</td>
<td></td>
<td>1.59</td>
<td>3.30</td>
<td>25.28</td>
<td>47.31</td>
</tr>
<tr>
<td>0.03</td>
<td>2.18</td>
<td>6.42</td>
<td></td>
<td>2.03</td>
<td>5.15</td>
<td>28.95</td>
<td>77.24</td>
</tr>
<tr>
<td>0.04</td>
<td>2.65</td>
<td>5.66</td>
<td></td>
<td>2.09</td>
<td>5.46</td>
<td>33.02</td>
<td>96.88</td>
</tr>
<tr>
<td>0.05</td>
<td>3.87</td>
<td>6.00</td>
<td></td>
<td>2.09</td>
<td>5.23</td>
<td>31.63</td>
<td>128.67</td>
</tr>
<tr>
<td>0.01</td>
<td>0.45</td>
<td>1.24</td>
<td></td>
<td>0.76</td>
<td>1.29</td>
<td>19.18</td>
<td>44.91</td>
</tr>
<tr>
<td>0.02</td>
<td>0.49</td>
<td>1.93</td>
<td></td>
<td>0.95</td>
<td>2.82</td>
<td>23.51</td>
<td>72.09</td>
</tr>
<tr>
<td>0.03</td>
<td>0.75</td>
<td>2.82</td>
<td></td>
<td>1.31</td>
<td>23.43</td>
<td>32.18</td>
<td>87.43</td>
</tr>
<tr>
<td>0.04</td>
<td>0.70</td>
<td>3.04</td>
<td></td>
<td>1.45</td>
<td>25.27</td>
<td>48.77</td>
<td>114.28</td>
</tr>
<tr>
<td>0.05</td>
<td>0.85</td>
<td>2.34</td>
<td></td>
<td>1.59</td>
<td>8.54</td>
<td>55.85</td>
<td>146.72</td>
</tr>
<tr>
<td>0.01</td>
<td>( I^{29} )</td>
<td>( I^{29} )</td>
<td></td>
<td>1.50</td>
<td>2.01</td>
<td>34.14</td>
<td>91.64</td>
</tr>
<tr>
<td>0.02</td>
<td>2.41</td>
<td>7.49</td>
<td></td>
<td>1.75</td>
<td>3.79</td>
<td>49.48</td>
<td>112.41</td>
</tr>
<tr>
<td>0.03</td>
<td>5.18</td>
<td>8.51</td>
<td></td>
<td>2.39</td>
<td>7.95</td>
<td>62.30</td>
<td>154.66</td>
</tr>
<tr>
<td>0.04</td>
<td>8.71</td>
<td>11.51</td>
<td></td>
<td>2.62</td>
<td>26.50</td>
<td>79.92</td>
<td>189.14</td>
</tr>
<tr>
<td>0.05</td>
<td>9.60</td>
<td>14.58</td>
<td></td>
<td>2.95</td>
<td>3.47</td>
<td>95.24</td>
<td>278.04</td>
</tr>
<tr>
<td>0.01</td>
<td>( I^{29} )</td>
<td>( I^{29} )</td>
<td></td>
<td>1.81</td>
<td>2.55</td>
<td>24.52</td>
<td>74.18</td>
</tr>
<tr>
<td>0.02</td>
<td>3.54</td>
<td>7.00</td>
<td></td>
<td>1.99</td>
<td>4.38</td>
<td>35.40</td>
<td>90.05</td>
</tr>
<tr>
<td>0.03</td>
<td>3.98</td>
<td>6.52</td>
<td></td>
<td>2.09</td>
<td>6.21</td>
<td>52.94</td>
<td>176.39</td>
</tr>
<tr>
<td>0.04</td>
<td>5.11</td>
<td>13.50</td>
<td></td>
<td>1.90</td>
<td>5.63</td>
<td>69.19</td>
<td>210.09</td>
</tr>
<tr>
<td>0.05</td>
<td>8.25</td>
<td>12.86</td>
<td></td>
<td>1.97</td>
<td>4.39</td>
<td>81.78</td>
<td>255.14</td>
</tr>
</tbody>
</table>

29 Infeasible solution.
5.5 Further analysis

In this section, we attempt to examine the convergence and/or the optimality of the presented algorithm. Because of the complexities of the problem, it seems to be very difficult, if not impossible, to handle these problems mathematically. Instead, a number of experiments are made under a variety of starting solutions to investigate the solution space of the problem. The results indicate that there are many local optimal solutions, thus making the final solutions dependent on the starting solutions. In this sense, neither the convergence nor the global optimality can be claimed for the new algorithm presented in Chapter 4.

However, it appears that the local optimal solutions found under good initial feasible solutions (e.g., phase-1 solutions under different values of M) fall within a reasonable boundary in terms of the minimum time. In this regard, the algorithm presented\(^{30}\) must be relatively independent of the value of M in determining the initial feasible solution.

Finally, the performance of the new algorithm is compared with other algorithms using some examples available in the literature. However, due to differences in modelling, an exact comparison cannot be made, but insights have been obtained from indirect comparisons.

---

30 "Algorithm" herein refers the two-phase algorithm with scheme 3, and the "relative independence" means the phase-2 solution of the algorithm is relatively independent of the value of M.
5.5.1 Experiment 1

This simple experiment is designed to observe the configuration of the solution space. The task is to find the optimal location of one intermediate knot point for the construction of two quartic polynomials to minimize the total travelling time to move from S to D (Fig. 23). A total of 110 starting solutions equally spaced throughout the workspace of the robot are chosen as shown in Fig. 24. The path constraint is relaxed for the purpose of the experiment, since many of the starting solutions will be infeasible if it is imposed.

Fig. 23 Starting and final configuration in experiment 1
Fig. 24 110 starting solutions in experiment 1
The results of this experiment show that:

a) There are many local optimal solutions (six in this case (refer to Fig. 25 for the minimum-time and the location of each point)

b) The local optimal solution reached depends on the starting solution (refer to Fig. 26 for the local optimal solutions reached by each of the 110 starting solutions).

In Fig. 26, however, it appears that reasonably good starting solutions located around the straight line between the initial and final point of the robot end effector (boxed region) mostly converge to the best local optimal solution (marked 1), and a few more reached the next best local optimal solution (marked 2). Particularly, all the starting solutions on the straight line between the reference points reached the best local optimal solution. In this respect, the heuristic used for phase 1 of the proposed algorithm (all the knot points are restricted to be on the reference path) and the use of the phase 1 solution as a starting solution seem to be justified.
Fig. 25 Six local optimal solutions
Fig. 26 Local optimal solutions reached by each of the 116 starting solutions
5.5.2 Experiment 2

This experiment was performed to observe the effect of $M$ on the new algorithm. The phase 1 solution obviously is affected by the value of $M$ since $M$ defines the possible locations of the knot points. Thus, different values of $M$ generate different initial solutions for phase 2. Note that the experimental results shown in Table 4 are based on an $M$ of 30.

Similar experiments using the same example paths and path tolerances were also performed using a variety of values of $M$. Some of the results are shown in Table 7. $M_1'$ and $M_2'$ indicate the number of knot points in the phase 1 and phase 2 solutions, respectively. The results of this experiment are as follows:

a) In general, as the value of $M$ increases, the minimum time of the phase-1 solution decreases.

b) The minimum time of the phase-2 solution is not greatly affected by the value of $M$. In particular, the minimum time of the phase-2 solution seems to be quite stable for the values of $M$ between 20 and 35.
Table 7 Results of experiment 2

<table>
<thead>
<tr>
<th>Ex #</th>
<th>$\epsilon$</th>
<th>M</th>
<th>$T_1^*$</th>
<th>$M_1^*$</th>
<th>$T_2^*$</th>
<th>$M_2^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.01</td>
<td>15</td>
<td>1.59</td>
<td>6</td>
<td>1.53</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>1.19</td>
<td>5</td>
<td>0.93</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>1.16</td>
<td>5</td>
<td>0.95</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>1.08</td>
<td>6</td>
<td>0.90</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>35</td>
<td>0.98</td>
<td>6</td>
<td>0.88</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>40</td>
<td>0.97</td>
<td>7</td>
<td>0.90</td>
<td>7</td>
</tr>
<tr>
<td>0.02</td>
<td></td>
<td>15</td>
<td>1.27</td>
<td>5</td>
<td>1.14</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>1.01</td>
<td>5</td>
<td>0.92</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>1.04</td>
<td>5</td>
<td>0.90</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>1.08</td>
<td>6</td>
<td>0.85</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>35</td>
<td>0.98</td>
<td>6</td>
<td>0.84</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>40</td>
<td>0.97</td>
<td>7</td>
<td>0.88</td>
<td>7</td>
</tr>
<tr>
<td>0.03</td>
<td></td>
<td>15</td>
<td>1.22</td>
<td>4</td>
<td>0.91</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>1.01</td>
<td>5</td>
<td>0.80</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>1.02</td>
<td>6</td>
<td>0.79</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>1.03</td>
<td>6</td>
<td>0.75</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>35</td>
<td>0.98</td>
<td>6</td>
<td>0.76</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>40</td>
<td>0.97</td>
<td>7</td>
<td>0.80</td>
<td>7</td>
</tr>
</tbody>
</table>

| 4    | 0.01       | 15 | 0.49     | 4        | 0.44     | 7        |
|      |            | 20 | 0.48     | 4        | 0.39     | 7        |
|      |            | 25 | 0.46     | 4        | 0.40     | 7        |
|      |            | 30 | 0.44     | 5        | 0.40     | 5        |
|      |            | 35 | 0.43     | 4        | 0.40     | 7        |
|      |            | 40 | 0.42     | 4        | 0.40     | 7        |

| 0.02 |            | 15 | 0.49     | 4        | 0.44     | 7        |
|      |            | 20 | 0.49     | 4        | 0.39     | 7        |
|      |            | 25 | 0.45     | 4        | 0.40     | 7        |
|      |            | 30 | 0.44     | 5        | 0.40     | 5        |
|      |            | 35 | 0.42     | 5        | 0.39     | 5        |
|      |            | 40 | 0.42     | 4        | 0.40     | 7        |

| 0.03 |            | 15 | 0.49     | 4        | 0.44     | 7        |
|      |            | 20 | 0.48     | 4        | 0.30     | 7        |
|      |            | 25 | 0.45     | 4        | 0.40     | 7        |
|      |            | 30 | 0.44     | 5        | 0.39     | 5        |
|      |            | 35 | 0.41     | 4        | 0.38     | 7        |
|      |            | 40 | 0.40     | 4        | 0.38     | 7        |
(Table 7 continued)

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.01</td>
<td>15</td>
<td>1.29</td>
<td>5</td>
<td>1.16</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1.21</td>
<td>5</td>
<td>0.95</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1.15</td>
<td>6</td>
<td>0.89</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>1.04</td>
<td>5</td>
<td>0.92</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>1.01</td>
<td>5</td>
<td>0.94</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0.97</td>
<td>6</td>
<td>0.85</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>0.02</td>
<td>15</td>
<td>1.23</td>
<td>4</td>
<td>0.96</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1.02</td>
<td>5</td>
<td>0.88</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1.01</td>
<td>5</td>
<td>0.90</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.98</td>
<td>5</td>
<td>0.91</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>1.01</td>
<td>5</td>
<td>0.91</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0.97</td>
<td>6</td>
<td>0.82</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>0.03</td>
<td>15</td>
<td>1.20</td>
<td>4</td>
<td>0.84</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1.02</td>
<td>5</td>
<td>0.87</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1.00</td>
<td>5</td>
<td>0.88</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.95</td>
<td>6</td>
<td>0.83</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>0.98</td>
<td>6</td>
<td>0.83</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0.94</td>
<td>6</td>
<td>0.80</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.01</td>
<td>15</td>
<td>1.12</td>
<td>6</td>
<td>0.91</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1.16</td>
<td>6</td>
<td>0.82</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>0.99</td>
<td>5</td>
<td>0.82</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.91</td>
<td>7</td>
<td>0.80</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>0.88</td>
<td>6</td>
<td>0.77</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0.81</td>
<td>7</td>
<td>0.75</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>0.02</td>
<td>15</td>
<td>1.12</td>
<td>6</td>
<td>0.77</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1.14</td>
<td>6</td>
<td>0.79</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>0.99</td>
<td>5</td>
<td>0.80</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.91</td>
<td>7</td>
<td>0.79</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>0.88</td>
<td>6</td>
<td>0.77</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0.81</td>
<td>7</td>
<td>0.72</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>0.03</td>
<td>15</td>
<td>1.01</td>
<td>5</td>
<td>0.77</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1.13</td>
<td>7</td>
<td>0.80</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>0.95</td>
<td>7</td>
<td>0.78</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.91</td>
<td>7</td>
<td>0.77</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>0.80</td>
<td>5</td>
<td>0.75</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0.79</td>
<td>6</td>
<td>0.70</td>
<td>11</td>
<td></td>
</tr>
</tbody>
</table>
5.5.3 Comparison with Other Works

In this subsection, the performance of the present algorithm is compared with other solutions available in the literature. However, direct comparison cannot be made because of differences among the concerns being addressed in the various problems. Thus, indirect comparison will be made based on the following discussion.

Hollerbach et al [45] investigated the true minimum-time trajectory with actuator constraints while allowing discontinuities in the acceleration. The following example is taken from Hollerbach's study.

![Figure 27 Hollerbach's example](image)
The optimal path in this example turned out to be a straight line both in joint space and Cartesian space with minimum time of 0.525 second. Incidentally, the optimal path generated by the algorithm developed herein also turns out to be straight line both in joint space and Cartesian space but with minimum time of 0.662 second, which is 26% higher than Hollerbach's solution. The difference is investigated through the comparison of acceleration profiles (Fig. 28).

As illustrated in the figures, Hollerbach's acceleration profile starts with high non-zero values and ends up with high non-zero values with some discontinuities in the middle of the trajectory. If we accept polynomial trajectories as a means for obtaining smooth motion, the 26% difference represents the cost of smoothness. Since "smooth motion" is defined as a motion of which the governing trajectories have continuous acceleration, zero acceleration at both ends of the trajectory should also be imposed because the accelerations before and after the motion are zero.\[31\]

31 Relaxing the boundary condition for the zero acceleration at both ends in the proposed algorithm results in a minimum time of 0.610 second (refer to Fig. 28).
Fig. 28 Angular acceleration profiles for Hollerbach's example
In the literature, Bobrow [5] presents a method to determine a mathematically proven optimal solution for a "predefined path". The two examples presented are solved by the algorithm developed herein by taking the predefined path as a reference path. Then, a comparison is made using various magnitudes of path tolerances and by using the smoothness cost (i.e., 26%) as a discounting factor since Bobrow's solution also involves discontinuity in the second derivatives.

His optimal solution to the first example path of the straight line (Fig. 29) is 0.72 second with one switching point in the phase diagram. Table 8 shows the minimum time given by the present algorithm for various values of path tolerances.

Fig. 29 Bobrow's first example

32 Refer to Chapter II for the discussion on this.
Table 8 Minimum time from the presented algorithm for Bobrow's first example

<table>
<thead>
<tr>
<th>Path Tolerance (m)</th>
<th>Minimum Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.005</td>
<td>1.18</td>
</tr>
<tr>
<td>0.010</td>
<td>1.00</td>
</tr>
<tr>
<td>0.020</td>
<td>0.86</td>
</tr>
<tr>
<td>0.030</td>
<td>0.84</td>
</tr>
<tr>
<td>0.040</td>
<td>0.83</td>
</tr>
<tr>
<td>0.050</td>
<td>0.83</td>
</tr>
</tbody>
</table>

Applying the discount factor for the smoothness, his solution is approximately 0.91 second. With extremely tight tolerances, for instance 0.005 (m), the minimum-time from the proposed method appears to be somewhat higher than the discounted solution. However, as the path tolerance increases, the proposed algorithm provides lower minimum times. The path of the end-effector with different path tolerances is seen in Fig. 30.

According to Fig. 30, the robot seems to take advantage of the dynamics for the speedy motion by traversing away from the minimum-distance reference path, as the path tolerance increases.
Fig. 30 Cartesian path plot for Bobrow's first example

The b curve (Fig. 31) supports this finding. As the path tolerance increases, the shape of the curve gets close to the U shape, indicating at least one of the joints at its torque bound most of the time, the major exceptions being at the start and end of the trajectory. However, the b-curve for the tight-tolerance cases shows an increasing W shape,
indicating neither of the joints is actuator-limited much of the time. This can also be observed in the torque plot (Fig. 32). In fact, Bobrow et al [5,6] proved mathematically that a necessary condition for an optimal trajectory for a predefined path is that at least one of the joint actuators is always at its torque bound.

It is, however, not appropriate to use Bobrow's optimality criterion for our problem, where the path is not predefined but given as a tube. Besides, it is hard to realize the optimality criterion if the continuity of acceleration is imposed at the switching point where the discontinuity of acceleration occurs.

There must be a singular zone where none of the actuators is at the torque bound even for a predefined path. Nonetheless, the closer to a U shape with the bottom of the U at or near one, the better the torque utilization. Note that some portion of underutilization around both ends of the motion is unavoidable in any case due to the boundary condition for zero acceleration at both ends.

Other plots describing the trajectory, profiles for position, velocity and acceleration of the robot end effector and joints with various magnitudes of path tolerances, are shown from Figs. 33 through Fig. 36.
Fig. 31  b-value plot for Bobrow's first example
Fig. 32 Torque plot for Bobrow's first example
Fig. 33 Cartesian velocity and acceleration plots for Bobrow's first example
Fig. 34 Angular position plot for Bobrow's first example
Fig. 35 Angular velocity plot for Bobrow’s first example
Fig. 36 Angular acceleration plot for Bobrow's first example
Bobrow's second example path is shown in Fig. 37. His optimal solution to this path is 0.70 second with two switching points in the phase diagram, and 0.88 sec with the smoothness discount factor. Results of the present algorithm can be seen in Table 9.
Table 9  Minimum time from the presented algorithm to Bobrow's second example

<table>
<thead>
<tr>
<th>Path Tolerance (m)</th>
<th>Minimum Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.005</td>
<td>0.98</td>
</tr>
<tr>
<td>0.010</td>
<td>0.95</td>
</tr>
<tr>
<td>0.020</td>
<td>0.89</td>
</tr>
<tr>
<td>0.030</td>
<td>0.85</td>
</tr>
<tr>
<td>0.040</td>
<td>0.82</td>
</tr>
<tr>
<td>0.050</td>
<td>0.77</td>
</tr>
<tr>
<td>0.060</td>
<td>0.68</td>
</tr>
<tr>
<td>0.070</td>
<td>0.62</td>
</tr>
<tr>
<td>$\infty$</td>
<td>0.42</td>
</tr>
</tbody>
</table>

Again, the new method results in higher minimum time with tight path tolerances, but lower minimum time as the path tolerance increases. If the path constraint is bigger than 0.05 (m), the presented minimum time is even lower than the true minimum time (undiscounted time); and, if there is no obstacle, (no path constraint), a minimum time of 0.42 sec is realized.

As previously observed, the robot takes advantage of the spatial freedom of motion for the faster movement as the path tolerance increases. The robot uses its dynamics by bending and extending the joints within a given tolerance (Fig. 38). This bending and extending phenomenon is also observed even when the robot is allowed to move freely; i.e., with no path constraint. Referring to Fig. 39, the robot bends its elbow joint slightly for faster forward movement. This is known as the 'kick effect', and is also observed in Rajan [41]
and Hollerbach [45]. Plots illustrating the behavior of robot motion without path constraints are displayed in Figs. 40 through Fig. 45. These plots which appear to be a smoothed form of the true minimum-time trajectory seem to describe closely the behavior of a true minimum-time trajectory with smooth motion.

Fig. 38 Cartesian path plot for Bobrow's second example with path constraints
Fig. 39 Cartesian path plot without path constraint

Fig. 40 b-value plot without path constraint
Fig. 41 Torque plot without path constraint

Fig. 42 Cartesian velocity magnitude plot without path constraint
Fig. 43 Cartesian acceleration magnitude plot without path constraint

Fig. 44 Angular velocity plot without path constraint
Fig. 45 Angular acceleration plot without path constraint
6.1 Research summary

The minimum-time trajectory planning problem is to find the optimal trajectory functions so that the total travelling time is minimized while satisfying the following constraints:

a) Path Constraint,

b) Actuator Constraint, and

c) Motion Constraint.

To the author's knowledge, most of the previous work on this problem concentrated on constraint b), and no explicit consideration has been given to constraint a) up to the present time. In reality, however, constraint a) is important especially when the robot works in complex work environments.

In addition, the smoothness of motion, represented by constraint c), is emphasized in this dissertation. Smooth motion is defined as the motion for which the governing trajectory has continuous acceleration throughout the movement. This constraint, however, is in conflict with the minimum-time objective function in the sense that smooth motion often takes longer times than nonsmooth motion, such as bang-bang types of motion. For this reason, the smoothness constraint has not
been included in research involving true minimum-time trajectories.

Since most robot motions, once programmed, will be repeated a vast number of times, nonsmooth motions could easily cause the build-up of unnecessary stresses on the machinery which is far from desirable. In other words, if the life span of the robot is affected, the use of true minimum-time with nonsmooth motion trajectories should be reconsidered.

In this dissertation, the actuator and path constraints are explicitly considered, and algorithms for minimum-time trajectory planning are developed mainly for off-line programming.

The feasible space for the avoidance of collision is represented as a tube specified by a set of reference points and path tolerances in terms of the robot hand. These tube parameters are set by the path planner considering the task environment of the robot. Thus, the problem is to find minimum-time trajectory functions which keep the robot hand travel inside of the tube while also satisfying other constraints.

The trajectory function adopted is a set of joint space polynomials. The decision variables for the construction of the trajectory polynomials are:

a) The number of knot points,

b) The location of the knot points, and

c) The transition time between the knot points.
Thus, the problem of finding the minimum-time trajectory functions involves the selection of the best set of design parameters. Because of the complexities of the problem involved, finding the global optimal solution is very difficult, if not impossible. Effort is therefore made to find near-optimal solutions by the reduction of the problem size using heuristics, approximations, etc.

To reduce the problem size, equal-time intervals are assumed based on the intrinsic properties of polynomial trajectories. By this, we can remove design parameter c) from consideration due to the time scaling and dynamic scaling properties.

Recognizing the inefficiencies involved in the splined method for constructing polynomials, three localized construction schemes (namely, scheme 1, scheme 2, scheme 3) are developed. The three schemes are different in the way they handle the final boundary conditions for the robot motion. The resulting order of polynomials for scheme 1 is all cubic except for the last segment which is quintic. Scheme 2 consists of all cubic polynomials except the last two segments which are both quartics, while scheme 3 consists of all quartics. The three schemes are applied to a two-stage optimization algorithm: phase 1, phase 2.

In essence, the phase-1 algorithm is to find a good solution, and the phase-2 algorithm is for the fine tuning of the phase-1 solution. In phase 1, all the knot points are
restricted to be on the reference path, and a branch-and-bound search method is used to find the best location and number of knot points by the discretization of the reference path.

The phase-2 algorithm takes the phase-1 solution as an initial feasible solution, and finds the optimal number and location of knot points by repeating three processes: modification, doubling, and halving. The modification process is to find the optimal locations for the given number of knot points by a gradient search method, while the doubling and halving processes involve finding preferred numbers of knot points.

To test the two-phase algorithm for the three different schemes, six FORTRAN codes are developed, and implemented for a two-link manipulator simulated on an IBM 3081 computer at the Ohio State University. Experiments are made with ten example paths representing basic motions of the robot. The results of the experiments showed that scheme 3 is dominant over scheme 1 and scheme 2. Scheme 1, in general, resulted in the largest minimum-times. Although scheme 3 needs longer computing time than the others, scheme 3 seems to be the most appropriate for off-line trajectory planning.

The algorithms presented indicate that the path tolerance has a significant effect on the minimum-time found. Under tight path tolerances, the robot has no spatial freedom in which to use its dynamics for speedy motion, resulting in higher values of minimum-time. However, as path tolerance
increases, the robot takes advantage of the dynamics by bending and extending its joints resulting in lower minimum times.

Compared with phase-2 solutions, phase-1 solutions, in general, proved to be relatively good, particularly when the path tolerance is small. This is consistent with the common notion that the center-line path of the tube well represents the feasible space having small tolerance.

For on-line purposes, the phase-1 algorithm with scheme 2 appears to be a good alternative considering the CPU time and its performance. However, the algorithm may need to be recast into a faster parallel computation procedure for on-line implementation.

By comparison with the conventional method, the time saving of the presented method is apparent. Even phase-1 solutions turned out to be considerably superior to the conventional method's minimum times. Moreover, the time savings of phase-2 solutions with scheme 3 recorded time reductions of over 50% in all cases.

To examine the fundamental issues of the presented algorithm, such as convergence and optimality, two experiments were made. In the first experiment, the starting solutions are arbitrarily generated, while in the second experiment they are generated by the use of the phase-1 algorithm for a variety of values of the discretization number M. The results of the first experiment showed that there are a number of local optimal solutions. However, the local optimal solutions
reached under reasonable starting solutions appeared to be equal or very close to the global optimal solution thus justifying the use of phase-1 solutions as an initial feasible solution in the presented algorithm.

The results of the second experiment showed that the minimum times obtained by the presented algorithm are relatively independent of the value of M. In particular, the minimum times for M values between 20 and 35 appear to vary very little.

Finally, the performance of the present algorithm was indirectly compared with other methods available in the literature. These methods have been proven to be optimal for predefined paths which allow discontinuities in the acceleration. For one example, the polynomial trajectory appears to take approximately a 26% longer time than the true minimum-time trajectory involving discontinuities in the second derivatives. The difference of 26% is considered as a cost for the smooth motion which results from the continuity of acceleration throughout the entire travelling time as well as at the start and end of the motion.

Taking into consideration the cost of the smoothness, the present algorithm could provide lower minimum times than other methods as the path tolerance increases. For a task requiring very high accuracy in the path of a robot, fine motion planning is applied. In such a situation, there are many more important and compelling requirements to be met than
minimum time. Thus, in reality, there exists a certain
degree of path tolerance allowed, and the present method
can take advantage of it. In this respect, the presented
methodology is considerably more flexible than the algorithm
which is based on a fixed path.

Furthermore, the algorithm presented in this dissertation
can be easily implemented to a robot control system with
little memory space since only the coefficients of the
trajectory polynomials are stored. During on-line control,
computation for desired trajectory parameters involves only
algebraic computation, and hence, accurate path tracking can be
accomplished with a high sampling frequency.

6.2 Further research

6.2.1 Reducing computation time

The presented off-line trajectory planning algorithm can
be easily extended to the six-link robot, but the computation
time is expected to increase considerably. The major hurdle
that causes long computation time is due to the "inseparability
of the optimality" between the stages (knot points). In other
words, the optimal trajectory as measured in terms of minimum
travel time is not necessarily the best trajectory for all the
trajectory segments. Although the quality of the solution
might decrease, an approximation with the separability will
save considerable computation time, since the approximated solution can be used as a fathoming rule which will be more strict than the fathoming rules used in the branch-and-bound search (phase 1) and when finding the steepest gradient in the modification process (phase 2). Regardless, the development of more strict fathoming rules can reduce the computation time, and is a topic for future study.

6.2.2 Tube parameters

In the present model, the tube parameters are assumed to be given by a path planner. Thus, the algorithm finds the optimal trajectory for the feasible space specified by the tube parameters. However, as the configuration of the task environment becomes more complex, there would be a number of alternatives to represent the collision-free space in terms of the tube parameters. The performance of the optimal trajectory would vary over the selections made. In view of this, one could extend the research to the spatial planning problem.

6.2.3 Optimality for smooth motion

Bobrow's optimality criterion involves discontinuities in the acceleration at the switching points. If they are constrained to be continuous, there must exist singular zones where none of the joints is at its torque bound. Based upon
this idea, it may be an interesting research topic to develop the optimality criteria for the trajectory with smooth motion for a predefined path. Furthermore, it will be one of the most tempting researches to develop the optimality criteria when the feasible space is given as a tube.


Sufficient condition

If $b_i = b$, for all $i$ where $i = 1, \ldots, M-1$, we prove $\mathcal{V}$ has the time scaling property w.r.t $C$.

Proof:

If $b_i = b$ for all $i$, the following holds for any $i$ since $\mathcal{V}_i$ is a time-scaled function of $C_i$:

$$\mathcal{V}_i(t) = C_i(bt) \quad \forall \; t \in [0, T_i/b],$$

where $i = 1, \ldots, M-1$.

Let $X$ be a cumulative time for $\mathcal{V}$

$$0 \leq X \leq \sum_{i=1}^{M-1} T_i / b,$$

then the functional value of $\mathcal{V}$ at $X$,

$$\mathcal{V}(X) = \mathcal{V}_k(x)$$

where $k$ is the largest integer that satisfies

$$\sum_{j=1}^{k} \frac{T_j}{b} \leq X,$$

and

$$x = X - \sum_{j=1}^{k} \frac{T_j}{b}.$$
Since the time axis of $V$ is uniformly scaled by $b$ from the time axis of $C$, the time $X$ in $V$ is equivalent to $bX$ in $C$.

Thus, the functional value at $bX$ in $C$ is

$$C(bX) = C_k(bx) \quad \text{(A.3)}$$

where $k$ and $x$ are the same as those defined in Eqn. (A.2).

From Eqn. (A.1), (A.2), and (A.3),

$$V(X) = C(bX). \quad \text{(A.4)}$$

Eqn. (A.4) holds for any $X \in [0, \sum_{i=1}^{M-1} T_i/b]$.

Taking derivatives of both sides of Eqn. (A.4) with respect to $X$, gives

$$V'(X) = bC(bX)$$

$$V''(X) = b^2 C(bX). \quad \text{Q.E.D.}$$

**Necessary condition**

If $V$ has the time scaling property w.r.t $C$, we prove $b_i = b$, for all $i$, $i=1,...,M-1$.

**Proof:**

If $V$ has the time scaling property,

$$V'_i(t) = b_i C'_i(b_i t) \quad \text{(A.5)}$$

$$V''_i(t) = b_i^2 C''_i(b_i t) \quad \text{(A.6)}$$

for any $t \in [0,T_i/b_i]$, for any $i$, $i=1,...,M-1$. 
Since both $C$ and $V$ satisfy the boundary conditions, the following holds for any knot point $i$, $i=1,...,M-1$:

\begin{align}
C_i' (T_i) &= C_{i+1}' (0) \\  
C_i'' (T_i) &= C_{i+1}'' (0) \\
V_i' (T_i/b_i) &= V_{i+1}' (0) \\
V_i'' (T_i/b_i) &= V_{i+1}'' (0). 
\end{align}

From Eqn. (A.5):

At $t=0$, $i=i+1$

\begin{equation}
V_i' (0) = b_{i+1} C_{i+1}' (0) 
\end{equation}

At $t=T_i/b_i$,

\begin{equation}
V_i' (T_i/b_i) = b_i C_i' (T_i) .
\end{equation}

From Eqns. (A.9), (A.11), (A.12),

\begin{equation}
C_i' (T_i) / C_{i+1}' (0) = b_i / b_{i+1}
\end{equation}

Since the left hand side of Eqn. (A.13) is 1 from Eqn. (A.5), $b_i = b_{i+1}$. This holds for any $i$.

Q.E.D.\textsuperscript{33}

\textsuperscript{33} Proof can also be made by using Eqns. (A.6), (A.8), and (A.10).
APPENDIX B

PROOF OF LEMMA 4.2

Proof: The minimum time of trajectory function $C$ is governed by the velocity and torque limits of the joint actuator.

Let $C'_{ij}(t)$ be the angular velocity of joint $j$ at time $t$ in trajectory segment $i$. In the time-scaled trajectory, the velocity is found from Eqn. (4.4):

$$\nabla'_{ij}(t/b) = b\ C'_{ij}(t) \quad (A.14)$$

where $t \in [0,T_i]$.

The maximum-time-scaling-factor (MTSF) due to the velocity limit of joint $j$ at this instant, $b'_{ij}(t)$ is found by replacing the L.H.S of Eqn. (A.14) by the maximum (or minimum depending on the sign of $C'_{ij}(t)$) velocity of joint $j$, $\theta'_{j}$:

$$b'_{ij}(t) = \theta'_{j} \sigma'_{ij}/C'_{ij}(t) \quad (A.15)$$

where $\sigma'_{ij}$ is the sign of $C'_{ij}(t)$.

The required torque at this instant is from Eqn. (3.9):

$$T_i(t) = I(C_i(t))C''_i(t) + C'_i(t)^T Z(C_i(t)) C'_i(t)$$

$$+ g_i(C_i(t)) \quad (A.16)$$
In the time-scaled trajectory,
\[
\ddot{T}_i(t/b_i) = \dot{I}(\ddot{\nu}_i(t/b_i))\dot{\nu}_i(t/b_i) + \dot{\nu}_i(t/b_i)^T \dot{Z}(\ddot{\nu}_i(t/b_i))\nu'_i(t/b_i) + g_i(\nu_i(t/b_i)).
\]

Applying Eqns. (4.3), (4.4), (4.5) to (A.17);
\[
T'_i(t/b_i) = b^2_i [I(C_i(t))C''_i(t) + C'_i(t)^T \dot{Z}(C_i(t))C'_i(t)] + g_i(C_i(t)).
\]

The MTSF due to the torque limit at time t in segment i, $b'_i(t)$ is found by replacing the L.H.S of Eqn. (A.18) by the maximum (or minimum depending on the sign of the denominator in Eqn. (A.21)) torque of joint j, $\gamma^\pm_j$ :
\[
\gamma^\pm_j = \left[ b'_i(t) \right]^2 [I(C_i(t))C''_i(t) + C'_i(t)^T \dot{Z}(C_i(t))C'_i(t)] + g_i(C_i(t)).
\]

Subtracting the gravity term in both Eqns. (A.16) and (A.19),
\[
[b'_i(t)]^2 [\ddot{T}_i(t) - g_i(C_i(t))] = \gamma^\pm_j - g_i(C_i(t)).
\]

For the MTSF due to $j^{th}$ joint torque limit at the instant t, $b'_{i,j}(t)$ is from the $j^{th}$ equation of (A.20):
\[
b'_{i,j}(t) = \left[ \frac{\gamma^\pm_j - g_{i,j}(t)}{g_{i,j}(t)} \right]^{1/2}
\]
where
\[ g_{ij}(t) \] is the gravity imposing on joint \( j \) at time \( t \) in segment \( i \), and is the \( j^{th} \) element of \( g_{i}(C_{i}(t)) \),
and \( \zeta \) is sign of \( \left[ 1_{ij}(t) - g_{ij}(t) \right] \).

The MTSF due to the limit of both angular velocity and torque of joint actuator \( j \) at the instant \( t \), is
\[ b_{ij}(t) = \min \{ b_{i,j}^{v}(t), b_{i,j}^{T}(t) \} \quad (A.22) \]
Since these limits should be kept for any joint and any time during the entire travel, and the time-scaling factor should be uniform to possess the time scaling property (Lemma 4.1), the MTSF for the entire travel \( b' \) is
\[ b' = \min_{i} b_{i}, \quad \text{for} \quad i=1,\ldots, M-1 \quad (A.23) \]
where
\[ b_{i} = \min_{t,j} b_{ij}(t), \quad \text{for} \quad t \in [0, T_{i}], \quad j=1,\ldots, n \]
\( b_{i} \) is the MTSF for segment \( i \).

Q.E.D.
APPENDIX C
PROOF OF COLLORARY 4.2

Proof: From Lemma 4.2, the minimum time of $C$ is

$$T_{c^*} = \sum_{i=1}^{M-1} \frac{T_i}{b^*}.$$  \hspace{1cm} (4.6)

where the original transition time for segment $i$ was $T_i$, $i=1, \ldots, M-1$, and the total transition time was:

$$T_c = \sum_{i=1}^{M-1} T_i.$$ \hspace{1cm} (A.24)

Suppose $V = \{v_{ij}(.), i=1, \ldots, M-1, j=1, \ldots, n\}$ is a uniformly time-scaled function of $C$ with scale factor of $k$; i.e., the transition time for segment $i$ in $V$ is $T_i/k$, $i=1, \ldots, M-1$.

Thus, the total transition time is:

$$T_v = \sum_{i=1}^{M-1} \frac{T_i}{k}.$$ \hspace{1cm} (A.25)

To prove the minimum-time of $V$, $T_v^*$ is the same as $T_c^*$, we show the MTSF for $V$ is $b^*/k$. This can be shown by following the similar method to that used in Appendix B.
MTSF due to the velocity limit of joint \( j \) at time \( t \) in segment \( i \) for \( v_{ij}(t) \),

\[
b^\nu_{ij}(t) = \theta_j \frac{\sigma_1}{\nu'_{ij}(t)} \tag{A.26}
\]

where \( \sigma_1 \) is defined as (A.15).

From Eqn. (4.4),

\[
\nu'_{ij}(t) = k C'_{ij}(kt). \tag{A.27}
\]

Thus,

\[
b^\nu_{ij}(t) = \theta_j \frac{\sigma_1}{k C'_{ij}(kt)}, \quad 0 < t < T_i/k. \tag{A.28}
\]

Following the procedure in Eqns. (A.16) through (A.20) with the relationships shown below,

\[
\nu_{ij}(t) = C_{ij}(kt) \tag{A.29}
\]

\[
\nu''_{ij}(t) = k^2 C''_{ij}(kt) \tag{A.30}
\]

and Eqn. (A.27), we get:

\[
b^\tau_{ij}(t) = \frac{1}{k} \left[ \frac{\gamma_j^{\sigma_2} - g_{ij}(t)}{\gamma_{ij}(t) - g_{ij}(t)} \right]^{1/2} \tag{A.31}
\]

By the same reasoning as made in Appendix 2, the MTSF for \( \nu \) is \( b^* / k \). Q.E.D.
APPENDIX D

PROOF OF LEMMA 4.3

Proof: Suppose \( \{T^*_1, i=1,\ldots,M^*-1\}, \{\theta^*_i, i=1,\ldots,M^*\} \) are the optimal decision variables for constructing \( C^*_i \), \( i=1,\ldots,M^*-1 \).

Suppose the following plot is the optimal trajectory for joint \( j; C^*_{ij}(t), i=1,\ldots,M^*-1 \).

![Fig. 46 C_{ij}(t) plot](image-url)
For the $i^{th}$ segment:

Let $\phi_j$ be the joint angle at time $\beta$ along $C_{ij}^*$;

$$C_{ij}^*(\beta) = \phi_j, \quad 0 \leq \beta \leq T_{i}^*,$$  \hspace{1cm} (A.32)

Dividing the $i^{th}$ segment into two intervals with;

$$1^{st} C_{ij}^*(j) = C_{ij}^*(t), \quad 0 \leq t \leq \beta$$ \hspace{1cm} (A.33)

$$2^{nd} C_{ij}^*(j) = C_{ij}^*(t+\beta) \quad 0 \leq t \leq T_{i}^* - \beta$$ \hspace{1cm} (A.34)

Now, the original optimal trajectory can be constructed from the following decision variables$^{34}$:

$$(T_{1}^*, \ldots, T_{i-1}^*, \beta, T_{i}^* - \beta, T_{i+1}^*, \ldots, T_{M+1}^*)$$  \hspace{1cm} (A.35)

$$(\theta_{1,j}, \ldots, \theta_{i,j}, \phi_j, \theta_{i+1,j}, \ldots, \theta_{M,j})$$  \hspace{1cm} (A.36)

In other words, the original segments $i+1$ through $M-1$ become $i+2$ through $M$, and there are $M+1$ knot points in the new trajectory polynomials.

Since this analogy can be applied for any joint and segment, there are an infinite number of sets of design parameters from which to construct the optimal trajectory.

Q.E.D.

---

$^{34}$ Note that $t$ begins at 0 again in each new segment.
APPENDIX E

PROOF OF COLLORARY 4.3

Proof: Suppose \( \{ T_i^*, i=1, \ldots, M^*-1 \}, \{ \theta_i^*, i=1, \ldots, M^* \} \) are the optimal decision variables for constructing \( C_i^* \), \( i=1, \ldots, M^*-1 \).

Consider Fig. 46 in APPENDIX D.

Let \( u \) be any common denominator\(^{35} \) of \( T_i^* \), where \( i=1, \ldots, M^*-1 \).

Representing \( T_i^* \) in terms of \( u \);
\[
T_i^* = u \gamma_i
\]  
(A.37)

Since \( \beta \) in Eqn. (A.30) can be chosen arbitrarily, we can assume \( \beta = u \). By this, the segment \( i \) will be divided into \( \gamma_i \) segments. The polynomial for the \( a^{th} \) subsegment of the \( i^{th} \) segment is
\[
^a C_{i,j}^*(t) = C_{i,j}^*(t+(a-1)u)
\]  
(A.38)

where
\[
0 \leq t \leq u
\]
\[
a = 1, \ldots, \gamma_i
\]

---

\(^{35}\) Common denominator herein could be any positive real values except 0 (e.g., 0.0001), and is not necessarily a natural number. Thus, there always exist common denominators for any sets of \( T_i^* \), \( i=1, \ldots, M^*-1 \).
This holds for any joint and segment.

Applying the same analogy used in APPENDIX D, the optimal trajectories \( (\mathbf{C}_i^*, i=1, \ldots, M^*-1) \) can be constructed by a total of \( \sum_{i=1}^{M^*-1} \gamma_i + 1 \) knot points with equal-time-interval of \( u \).

Q.E.D.