To the Graduate Council :

I am submitting herewith a thesis written by Siamak Esmailzadeh Khadem entitled " A Global Redundant Robot Control Scheme For Obstacle Avoidance." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Mechanical Engineering.

Rajiv Dubey

Rajiv V. Dubey, Major Professor

We have read this thesis and recommend its acceptance :

Accepted for the council :

Vice Provost
and Dean of The Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master of Science degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotation from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature S.E. Kmoch

Date April 25, 88

A GLOBAL REDUNDANT ROBOT CONTROL SCHEME FOR OBSTACLE AVOIDANCE

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Siamak Esmailzadeh Khadem

June 1988

To my grandmother and

in memory of my grandfather

ABSTRACT

Optimal control of kinematically redundant manipulators, those with more than the minimum number of degrees of freedom required to do a specific task, is an important and complicated problem in robotics. The issue is to use the extra degrees of freedom to meet the secondary objectives such as energy optimization, singularity avoidance, obstacle avoidance, improved dynamic response, higher dexterity and flexibility, etc., in addition to the primary objective of tracking a specified path for the end-effector.

Among the secondary objectives, obstacle avoidance is important for a robot to be able to work in a cluttered environment. Using the extra degrees of freedom, we can control the redundant manipulator such that it avoids workspace obstacles and at the same time tracks the specified end-effector path. Some local optimization schemes for obstacle avoidance have been presented which are not able to avoid obstacle over long trajectories. In this paper we deal with the obstacle avoidance problem by using modern control theory and choosing an integral type performance index which results in a global optimization scheme. Obstacles are expressed as state space constraints . The state constraint function and control effort are minimized globally as a performance index. The control effort which maximizes the Hamiltonian and minimizes the performance index

is used to find the homogenous solution by utilizing the null space of the Jacobian. A simulation for a three degrees of freedom planar robot is presented to demonstrate the effectiveness of the scheme.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

A robotic manipulator is called redundant if it has more than the minimum number of joints or degrees of freedom required to perform its task. For arbitrary positioning and orienting a workpiece in the three dimensional space we need six degrees of freedom, three for position and three for orientation. The human arm without considering the degrees of freedom of fingers, which provide extra dexterity for fine motions, has seven degrees of freedom and is kinematically redundant (Figure 1). Using the extra degrees of freedom, we can either meet additional limitation on environment, defining some forbidden region in the joint space, or optimize a performance criterion. In reality, a major portion of the available space for robot manuever becomes unutilizable due to singular configurations or obstacles. In a singular configuration, a robot arm gets mechanically locked and is unable to move in at least one direction. In the vicinity of singular configurations, extremly high joint velocities are required for robot end-effector motion. Therefore, this limits robot movement in certain directions. Depending on the required task, extra degrees of freedom may be either desirable or necessary. If the problem of interest is to reach a specified end point, it is possible to find a path to the final point which avoids obstacles and singularities. Thus, nonredundant robot can track this path and reach the final point. In addition to reaching the final point, if it is

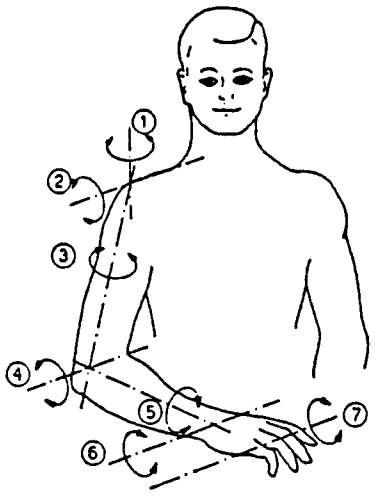
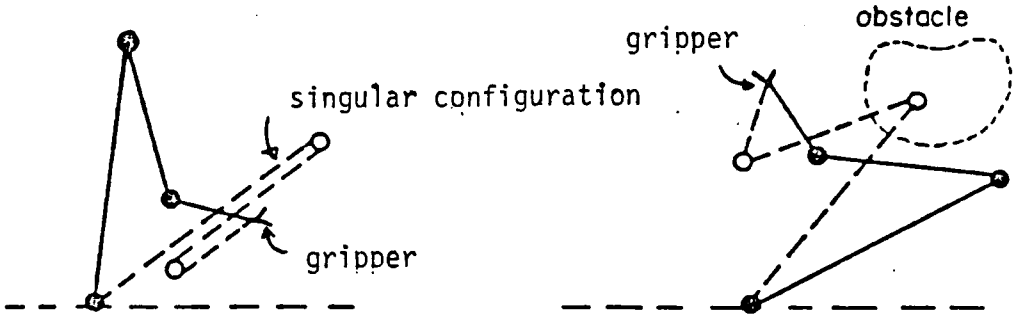


Figure 1. Degrees of Freedom of a Human Arm

required to maintain accurate position and orientation of the end-effector throughout a given trajectory, and at the same time avoid singularity portions of the environment and obstacles (Figure 2), additional joints become inevitable. Extra degrees of freedom provide higher maneuverability and dexterity to the manipulator. This is usually achieved by optimizing various performance measures. This results in energy optimization, singularity avoidance, obstacle avoidance, improved dynamic response, joint motion constraint within their mechanical limits and higher manipulability. Optimizing various performance measures may be achieved locally or globally. The global optimization schemes determine a joint trajectory from the complete description of the desired end-effector trajectory. In local optimization schemes joint trajectory is determined from the instantaneous joint motion required to follow a desired end-effector trajectory. In these schemes, as compared to the global optimization schemes, the performance measure does not reach to its optimum value for the entire trajectory.

A large number of tasks require the robot manipulator to move the end-effector from a given position and orientation to a specified position and orientation along a specified trajectory in a cluttered environment. The workspace of the robot is usually occupied by obstacles, such as mechanical parts, fixtures, links of another robot, etc. which create forbidden area in the



Alternative configuration of a planar redundant manipulator to avoid a singularity configuration

Alternative configuration of a planar redundant manipulator to avoid an obstacle

Figure 2. Alternative configurations of a redundant robot

workspace. Therefore, a control scheme is required to determine the joint motions resulting in robot configurations which avoid these obstacles.

Problem Statement

The desired translational and rotational motion of the robot end-effector is obtained by resolving the end-effector velocity into required joint velocities. For a nonredundant robot, joint velocities required for a specified end-effector velocity are unique. But for a redundant robot this projection from the end-effector space onto the joint space is not unique, and there are infinite number of solutions. The joint velocities may be chosen such that it is possible to optimize a performance criterion in addition to tracking the desired path and moving the end-effector with a desired linear and angular velocity.

This thesis studies the obstacle avoidance problem for redundant robots. Modern control theory with an integral type performance index will be used which results in a global optimization scheme. This problem will be addressed in the following sequence:

(a) defining state constraint function and performance index,

(b) applying modern control theory and Pontryagin's maximum principle [Pontryagin]

(c) finding the optimum joint velocities which result in obstacle avoidance

(d) simulation of a three degrees of freedom planar robot to demonstrate the effectiveness of the scheme.

Organization

This thesis is organized as follows; Chapter II is the background which discusses the State-of-the-Art in obstacle avoidance schemes. Chapter III describes the development of a global control scheme for obstacle avoidance. Simulation results for a redundant planar robot are presented in Chapter IV. Finally, Chapter V presents conclusions and recommendations for future research.

II. BACKGROUND

The position and orientation of a robot manipulator is uniquely defined by the robot configuration which can be described by

$$\mathbf{x}_e = f(\Theta(t)), \tag{1}$$

where $\mathbf{x}_e$ is the $mx1$ end-effector position and orientation vector ($m \leq 6$; 3 translational and 3 rotational degrees of freedom), and $\Theta(t)$ is an $nx1$ joint angles vector. The end-effector velocity is found by taking the derivative of equation (1) with respect to time

$$\dot{\mathbf{x}}_e(t) = J(\theta(t))\ \dot{\Theta}, \tag{2}$$

where $\dot{\mathbf{x}}_e$ is the $mx1$ end-effector velocity vector, $\dot{\Theta}$ is the $nx1$ joint velocity vector, and $J(\Theta) = \partial f/\partial\Theta$, is an mxn matrix, called the Jacobian. If J is nonsingular and a square matrix, the rank of J is equal to m, then $\dot{\Theta}$ can be determined as follows:

$$\dot{\Theta} = J^{-1}\dot{\mathbf{x}}_e. \tag{3}$$

If J is singular, i.e. the rank of $J < m$, or if J is rectangular, $m < n$,

we have

$$\dot{\Theta} = J_e^+ \dot{x}_e + (I - J^+J)\,\alpha \ , \qquad (4)$$

where J^+_e is the Moore-Penrose generalized inverse of the Jacobian which is equal to $J^T(JJ^T)^{-1}$ if J is full ranked. I is an nxn identity matrix, and $(I-J^+J)$ is the null space projection matrix(Albert, 1972). The first term on the right hand side, $J^+_e\dot{x}_e$, is the least norm solution while the second term, $(I-J^+J)\alpha$, is the projection of the arbitrary joint velocity vector α, onto the null space of the Jacobian. It is the homogeneous solution which can be used to enhance a performance criterion. The homogeneous solution does not contribute to the end-effector motion, but may be used to control the robot manipulator to avoid obstacles. Using the homogenous solution we can improve various performance criteria. Several researchers have used this property to optimize certain performance criteria.

Pieper (1968) , Loef and Soni (1975), Udupa (1977) , Khatib and Lemaitre (1978) , Lozano-Perez (1981), Lee and Chien (1987), have presented different approaches for the task of approaching given goal configuration from the known end-effector initial configuration while avoiding obstacles. In these approaches path is not important and any joint trajectory which results in reaching the goal configuration and at the same time avoiding obstacles is satisfactory. In fact these approaches are

concerned with the path planning problem and seeking for safe path to avoid obstacles and reach the desired final point, and are ideal for pick and place tasks.

Udupa (1977), presented a solution for the problem of planning safe trajectories. Using this solution, the trajectory planning system plans a trajectory that will avoid obstacles and maneuver the end-effector into the goal configuration.

Lozano-Perez (1981), focused on the geometric aspects of the pick and place problem. For this synthesis the following data are required:

- geometric description of the environment,
- initial and final configuration of the manipulator and the grasped object.

As a planning problem, Lee and Chien (1987), presented solutions for the time-varying obstacle avoidance problem by changing the path and the trajectory of the manipulator for the purpose of collision avoidance.

Khatib (1985), defined the goal position as an attractive pole and obstacles as repulsive surfaces, and introduced the concept of artificial potential field which is a positive continuous and differentiable function. When the end-effector reaches the final goal this function attains its zero minimum. In a cluttered environment, an artificial potential field can reach to its local minimum while the end-effector is still far from the desired final point. Although linking these local minima can cure this problem,

it can be a highly time consuming process.

In the scheme presented by Maciejewski and Klein (1985), at each instant of time, the collidable point is the point which is the closest one to the obstacle, and it is desired to prevent that point to approach the obstacle. The following equation was used to avoid obstacles:

$$\dot{\theta} = \dot{J}_e^{\dagger}\dot{X}_e + \left[J_o(I - \dot{J}_e^{\dagger}J_e)\right]^{\dagger}(\dot{X}_o - J_o\dot{J}_e^{\dagger}\dot{X}_e) , \qquad (5)$$

where

J_e= end-effector Jacobian

J_o= obstacle avoidance point Jacobian

$\dot{X}_e$= specified end-effector velocity, and

$\dot{X}_o$= specified obstacle avoidance point velocity.

This approach is an instantaneous optimization scheme and it solves the obstacle avoidance problem temporarily, as another point on the same link may become the collidable point. On the other hand, monitoring the motion of the collidable point and its distance to the obstacle is not always possible.

Baillieul (1986), introduced the concept of extended Jacobian, J_e, to find joint rate motions. By maximizing some type of distance functions, he tried to avoid obstacles. In his approach as he pointed out, switching from one constraint to another results in a kinematic singularity. Also, his scheme does not put any bound

on joint velocities.

Yoshikawa (1984), presented an obstacle avoidance scheme for a redundant robot. He used the homogeneous solution to avoid obstacles while the end-effector tracked the path. He assigned the first priority to the path tracking and the second priority to the obstacle avoidance. Using the performance criterion P defined as follows:

$$P = g(\Theta) = \frac{1}{2}(\Theta - \Theta_r)^T H_2 (\Theta - \Theta_r) , \qquad (6)$$

where $H_2 = diag(h_{2i}) \in R^{n \times n}$, and $h_{2i} > 0$ are constants, the arm is brought as close as possible to the safe cofiguration, Θ_r arm posture, which is known. His scheme is an instantaneous optimization scheme which deals with the end-effector obstacle avoidance problem. In a cluttered environment, it is usually difficult to find a universal safe joint vector Θ_r.

Nakamura and Hanafusa (1987), presented a scheme for optimal redundancy control of robot manipulators which is a global optimization scheme. They defined the first manipulation variable $r_1 \in R^{m_1}$ as the constrained variable for the position and orientation of the end-effector, and r_2 ($r_2 \in R^{m_2}$, and $m_2 = 1$) as the performance index :

$$r_1(t) = f_1(\Theta)\,, \tag{7}$$

$$r_2 = \int_{t_0}^{t_1} (k\,P_o(\Theta) + \dot{\Theta}^T \dot{\Theta})\,dt\,. \tag{8}$$

Using the configurational space state, $\Theta(t)$, and applying the Pontryagin's maximum principle, they presented the optimal solution as the system of differential equations (9) -(11):

$$\dot{\Theta} = g\,, \tag{9}$$

$$\dot{\psi} = (\frac{\partial g}{\partial \Theta})^T (2g - \psi) + K(\frac{\partial P_o}{\partial \Theta})\,, \tag{10}$$

$$g = J_1^+ \dot{r}_1(t) + \frac{1}{2}(I - J_1^+ J_1)\,\psi\,, \tag{11}$$

which has to be solved numerically. As may be seen in equation 10, we need to compute partial derivatives of $J_1^+$and $J_1^+J_1$ with respect to Θ, which is the configurational state. Therefore, $(\partial g/\partial \Theta)^T$ will be an $n \times n$ matrix which will require a large amount of time-consuming calculations.

In this thesis we deal with the obstacle avoidance problem by using modern control theory and choosing an integral type performance index which results in a global optimization scheme. Obstacles are expressed as state space constraints. The state constraint function and control effort are minimized globally as a performance index. This scheme results in obstacle avoidance for

the collidable link. The control effort which maximizes the Hamiltonian and minimizes the performance index, is used to find the homogenous solution by utilizing the null space of the Jacobian. A simulation for a three-degree-of-freedom planar robot is presented to demonstrate the effectiveness of the scheme.

III. A GLOBAL CONTROL SCHEME FOR OBSTACLE AVOIDANCE

We consider an n degree-of-freedom manipulator. The corresponding joint coordinate vector, $\Theta = [\ \theta_1, \theta_2, \ldots\ldots\ldots \theta_n\ ]^T$, is defined in configuration space as $\Theta = \{\ \theta : \theta_{i\ min} <= \theta <= \theta_{i\ max}, i = 1, 2, \ldots n\ \}$ where $\theta_{i\ min}$ and $\theta_{i\ max}$ are the minimal and maximal quantities for the ith joint, defined by constructional constraints of the robot. Each point in the workspace is defined by a vector $\mathbf{x} = [\ x, y, z\ ]^T \in R^3$ where x, y, z are Cartesian coordinates with respect to the base coordinate frame. Let us designate by M the set of points in the workspace which have been occupied by the robot itself. This set is uniquely defined by the robot configuration at each instant of time (in terms of the joint coordinate vector). We assume there is a spherical safety zone around a stationary obstacle and we denote the corresponding set of points which have been occupied by the obstacle and its safety zone by O. We also consider a general case when the coordinates of the end-effector in the Cartesian space are $\mathbf{x}_e = [\ x_{e1}, x_{e2}, \ldots\ldots\ldots x_{em}\ ] \in R^m, m < n$. The first three component of the end-effector coordinates specify the position of the end-effector, while the rest of (m - 3) components specify completely (for m=6) or partially (for m<6) the orientation of the end-effector. Since the number of joints or degrees of

freedom n, is greater than the number of coordinates m, required to specify the position and orientation of the end-effector, the time history of the joint angle vector $\Theta(t)$ which is related to a specified $\mathbf{x}_e(t)$ is not uniquely defined.

The problem is to control the robot in such a way that it avoids obstacles and tracks the specified path at the same time. Therefore, the main purpose is to control the collidable link to avoid obstacles while performing the goal of tracking the path by the end-effector.

We define the minimum distance between the collidable point and the center of the safety zone as:

$$d(\mathbf{x}, \mathbf{c}) = \min || \mathbf{x} - \mathbf{c} ||, \tag{12}$$

where $\mathbf{x} \in M$ is the coordinate vector for the collidable point, $\mathbf{c} \in O$ is the coordinate vector for the center of the safety zone, and $||\ ||$ denotes the Euclidean norm. The manipulator will avoid the obstacle described in Figure 3, if the following condition is satisfied :

$$d(\mathbf{x}, \mathbf{c}) \geq R \tag{13}$$

where R is the radius of the safety zone.

It may be noted that controlling just a collidable point does not guarantee the obstacle avoidance for the whole collidable link

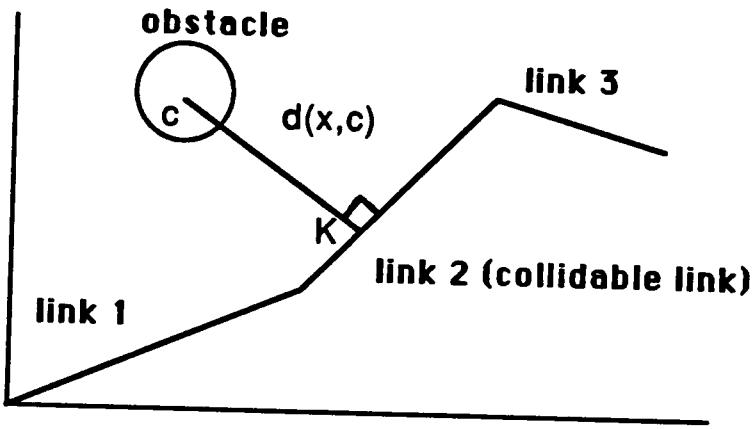


Figure 3. Minimum distance between the collidable link and obstacle

since at any time another point of the same link may become a collidable point. Since we are looking for a collision avoidance scheme for the whole collidable link, we select $d(\mathbf{x}, \mathbf{c})$ as the minimum distance between the center of the safety zone and the collidable link at each instant of time.

Since our manipulator is redundant, we can utilize the redundancy to optimize a performance criterion. To avoid an obstacle, we use the extra degrees of freedom such that $d(\mathbf{x}, \mathbf{c})$ is maximized as a performance criterion.

Let us assume that link L of the robot is the collidable link. We define the collidable point k on link L to be the intersection of the perpendicular drawn from the center of the obstacle (assumed to be spherical by adding a safety zone around the obstacle) to the collidable link L. The velocity of point k will be:

$$\dot{x}_k = f(x_k, u, t) \quad , \tag{14}$$

$$\dot{x}_k = J_k(t)\,\dot{\Theta} = J_k(t)\,\mathbf{u} \quad , \tag{15}$$

where x_k is $m \times 1$ vector and J_k, the Jacobian for point k, is $m \times n$ matrix. Θ or $\mathbf{u}$ is the $n \times 1$ angular velocity vector which we refer to as the input vector.

We define the state constraint function as the square of the minimum distance between the collidable point and the surface of the safety zone as:

$$G(x) = R^2 - (x_k - c)^T (x_k - c) . \qquad (16)$$

To avoid obstacle, the state constraint function $G(x)$, should be :

$$G(x) < 0 , \qquad (17)$$

and when :

$$G(x) = 0 , \qquad (18)$$

the collidable link will be tangent to the safety zone of the obstacle, as shown in Figure 4.

Since our aim is obstacle avoidance, if we minimize the state constraint function $G(x)$, it means we are avoiding obstacle. On the other hand, we desire to spend the least possible amount of control effort, or kinetic energy. Therefore, we select a quadratic integral type performance index as :

$$\pounds = \frac{1}{2}\int_{t_0}^{t_1}\left[R^2 - (x_k - c)^T Q_1 (x_k - c) + u^T R_1 u \right] dt , \qquad (19)$$

where Q_1 and R_1 are symmetric and positive definite weighting matrices. In general, it may be desired to weigh elements of the joint velocity vector $\dot{\Theta}$ or u, differently. It is the case when we

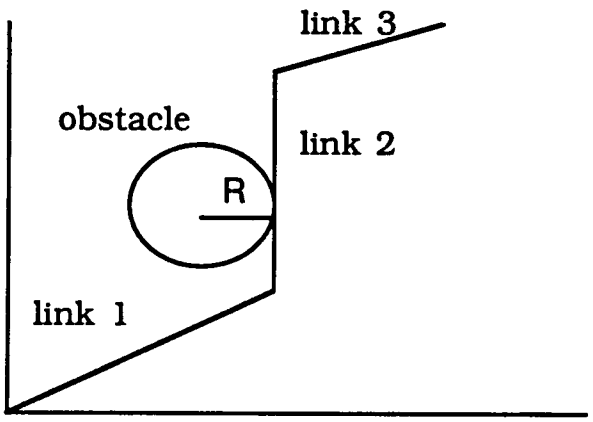


Figure 4. The collidable link is tangent to the obstacle

wish to have different rate of motions for heavier parts of the robot. For example, if we need precise motion, we should penalize the motion of the heavier joints in favor of the other joints which are closer to the wrist. Therefore, R_1 can be used for this task. The elements of Q_1 assign weight to the obstacle avoidance related part of the performance index. The first two terms are state constraint function and the third term is the kinetic energy (or control effort) which is consumed to perform the task of avoiding obstacle. The problem is to minimize £, which is equivalent to the maximization of the vertical distance between the center of the obstacle and the collidable link at each instant of time, and the control effort while the end-effector tracks the specified path.

Using the Pontryagin's method [Pontryagin] for controlling the velocity of collidable point, we define the Hamiltonian H as:

$$H = -\frac{1}{2}R^2 + \frac{1}{2}(\mathbf{x}_k - \mathbf{c})^T Q_1 (\mathbf{x}_k - \mathbf{c}) - \frac{1}{2}\mathbf{u}^T R_1 \mathbf{u} + \lambda^T J_k \mathbf{u} \ , \qquad (20)$$

where λ is an m x 1 adjoint or costate vector for the state vector $\mathbf{x}_k$. Therefore, we have

$$\dot{\mathbf{x}}_{i\,k} = \frac{\partial H}{\partial \lambda_i}, \qquad i=1, 2, ..., m \qquad (21\text{-a})$$

and

$$\dot{\lambda}_i = -\frac{\partial H}{\partial x_{i\,k}} \cdot \qquad i=1, 2, ..., m \qquad (21\text{-b})$$

Note that equation (21 - a) is reduced to the original equation (15) after using equation (20). Our problem is a fixed time problem (t_0 and t_1 are given) with fixed left-hand endpoint x_0, and a free right-hand endpoint [Appendix B]. Therefore, solution to the system equations (21) should satisfy the two-point boundary conditions :

$$x_k(t_0) = x_0 \tag{22}$$

$$\lambda(t_1) = \left[\lambda_1, \ldots\ldots, \lambda_m \right] = \left[0, \ \ \ldots\ldots, 0 \right] \tag{23}$$

which give us $2m$ boundary conditions.

By using the initial value adjusting method [Appendix A], we are able to solve multipoint boundary value problems. To solve the two-point boundary value problem, first we estimate the unknown left-hand boundary conditions. Then combining the estimated and the given boundary conditions we solve the differential equations. Comparing the computed and estimated values for λ at the right-hand point, we modify the left-hand boundary conditions. This process is repeated until the computed values match with the known values for λ at the right-hand point. There are four commonly used numerical techniques for solving the nonlinear two-point boundary value differential equations: Steepest descent, Variation of extremals, Quasi-linearization, and gradient projection [kirk]. In Chapter IV, we use the Variation of

extremals technique to solve this two-point boundary value problem.

In order to find the extreme value for H, we set the partial derivative of H with respect to **u** equal to zero. Thus we have

$$-R_1 \mathbf{u} + J_k^T \lambda = 0 . \tag{24}$$

On rearranging terms in (24) we obtain

$$\mathbf{u} = R_1^{-1} J_k^T \lambda . \tag{25}$$

Taking partial derivative of H with respect to $\mathbf{x}_k$, we have

$$\dot{\lambda} = -\frac{\partial H}{\partial \mathbf{x}_k} = -Q_1(\mathbf{x}_k - \mathbf{c}) , \tag{26}$$

The above equation may be solved using Euler's method. The value of $\lambda(t + \Delta t)$ is obtained as follows:

$$\lambda(t + \Delta t) \approx \lambda(t) + \dot{\lambda}(t)\Delta t . \tag{27}$$

Thus using (27) and the initial condition $\lambda(t_0)$, we can determine

$\lambda(t_1)$. For accurate results, it is required to select a relatively small value of Δt.

Since the end-effector is required to track the desired path, the joint velocities are determined using

$$\dot{\Theta} = J^{+} \dot{x}_e + (I - J^{+}J)\,\alpha\,, \tag{4}$$

where $\dot{x}_e$ is the desired end-effector velocity, and α is an arbitrary joint velocity vector. We now determine α such that $\dot{\Theta}$ in (4) is equal to **u** in (25) in the least square sense. This requires us to solve the following set of equations in the least-square sense:

$$R_1^{-1} J_k^{T} \lambda = J_e^{+} \dot{x}_e + (I - J_e^{+} J_e)\,\alpha\,. \tag{28}$$

Solving for α, we get

$$\alpha = \left[I - J_e^{+} J_e \right]^{+} \left[-J_e^{+} \dot{X}_e + R_1^{-1} J_k^{T} \lambda \right]. \tag{29}$$

This equation is the least mean square solution which minimizes the expression

$$|| R_1^{-1} J_k^T \lambda - J_e^+ \dot{x}_e - (I - J_e^+ J_e)\alpha ||.$$

We now use the following properties of the pseudoinverse of J [Rao]:

(a) $(I - J_e^+ J_e)^+ = I - J_e^+ J_e$

(b) $(I - J_e^+ J_e) J_e^+ = 0$

(c) $(I - J_e^+ J_e)^2 = (I - J_e^+ J_e)$.

Applying property (a) in (29), we obtain the following :

$$\alpha = (I - J_e^+ J_e)(- J_e^+ \dot{X}_e + R_1^{-1} J_k^T \lambda). \tag{30}$$

We now apply property (b) to equation (30) to obtain

$$\alpha = (I - J_e^+ J_e) R_1^{-1} J_k^T \lambda. \tag{31}$$

Substituting α in equation (4), and applying property (c) we have

$$\dot{\Theta} = J_e^+ \dot{X}_e + (I - J_e^+ J_e) R_1^{-1} J_k^T \lambda. \tag{32}$$

By using the above equation, we can determine the required joint velocities to track the desired path and avoid obstacles. We determine the joint position by numerically integrating the joint velocities obtained in (32).

IV. SIMULATION

We consider a three degree of freedom planar robot, shown in Figure 5 for simulation. In order to systematically determine the expression for the Jacobian (J_e and J_k) of this robot, we define coordinate frames corresponding to each joint. The convention we use to locate frames on the links is as follows[Craig]: the z axis of frame i, called z_i, is coincident (as shown in Figure 6) with the joint axis i. The origin of the frame i is located where the a_i perpendicular intersects the joint i axis. x_i points along a_i in the direction from joint i to joint i+1. Therefore,

a_i = The distance from z_i to z_{i+1} measured along x_i,

α_i = The angle between z_i and z_{i+1} measured about x_i,

d_i = the distance from x_{i-1} to x_i measured along z_i; and

Θ_i = the angle between x_{i-1} and x_i measured about z_i.

For the second link as the collidable link, we assume that the collidable point to be point k. We also assign a coordinate frame to the collidable point in order to be able to compute the velocity of that point (as shown in Figure 7). As we use a systematic approach used in robotics, the proposed scheme can be applied to any other robot. To find the Jacobian for the end-effector and the collidable point, we need to construct the rotation matrices $R_{i+1}{}^i$ and $R^{i+1}{}_i$. $R_{i+1}{}^i$ transforms a vector in i+1 coordinate frame to a

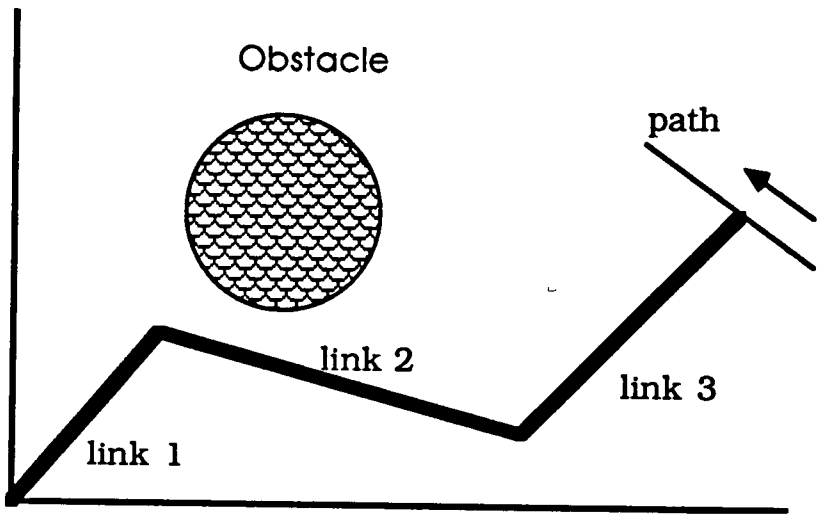


Figure 5. A three degree-of-freedom planar redundant robot

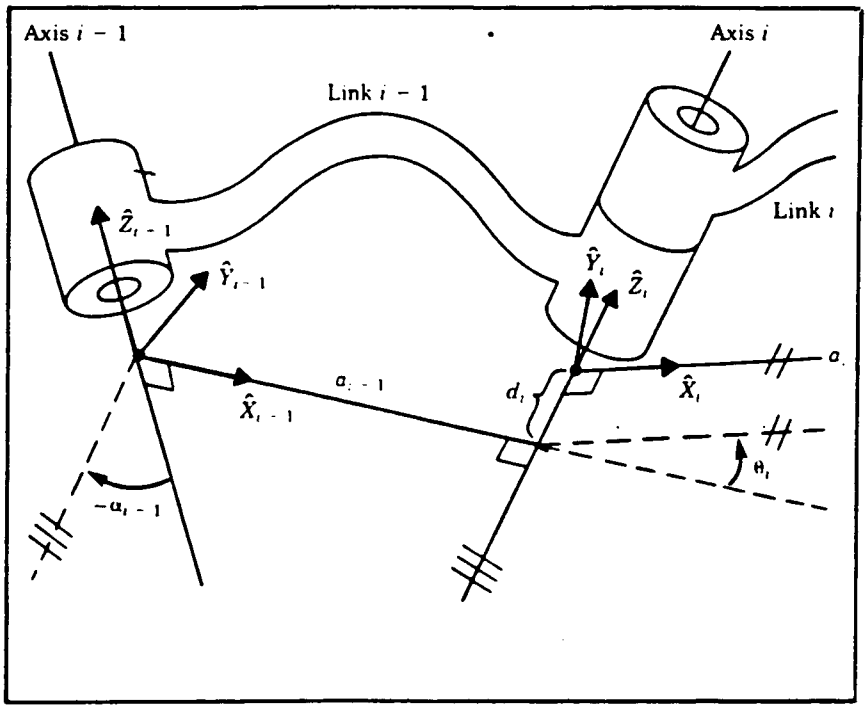


Figure 6. Link frames are attached so that frame {i} is attached rigidly to link i [Craig]

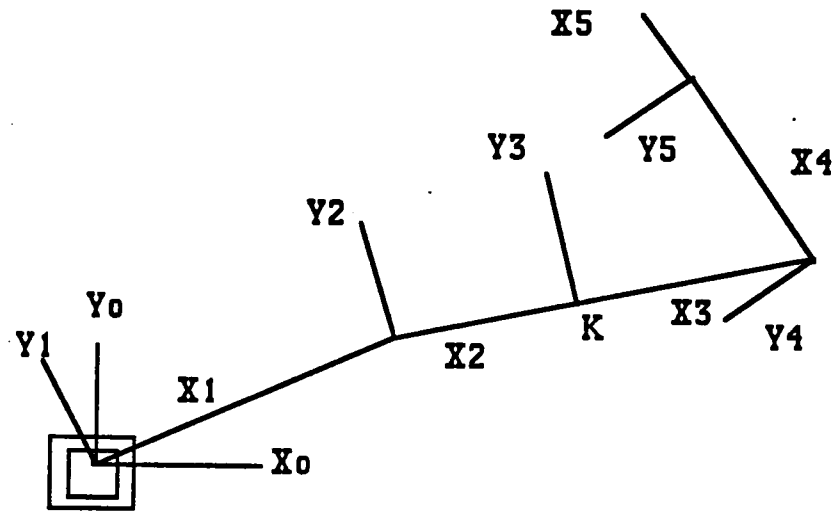


Figure 7. Assigned coordinate frames for each joint and the collidable point

vector in i coordinate frame. The rotation axis z_i for all joints is perpendicular to the plane of the robot. Therefore, rotation matrices R_{i+1}^{i} and R^{i+1}_{i} may be written as:

$$R_{i+1}^{i} = \begin{bmatrix} C_{i+1} & -S_{i+1} & 0 \\ S_{i+1} & C_{i+1} & 0 \\ 0 & 0 & 1 \end{bmatrix}, R_{i}^{i+1} = \begin{bmatrix} C_{i+1} & S_{i+1} & 0 \\ -S_{i+1} & C_{i+1} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (33)$$

where $C_{i+1} = \cos(\theta_{i+1})$, and $S_{i+1} = \sin(\theta_{i+1})$. The velocity of the origin of each coordinate frame is found by the equation

$$V_{i+1}^{i+1} = R_{i}^{i+1}(V_{i}^{i} + \omega_{i}^{i} \times P_{i+1}^{i}), \quad (34)$$

where V_{i+1}^{i+1} is the linear velocity of the coordinate frame i+1 with reference to the coordinate frame i+1, P_{i+1}^{i} is the position vector joining origin i to i+1, ω_i^i is the angular velocity of the joint i with reference to the coordinate frame i, and

$$\omega_{i}^{i} \times = \begin{bmatrix} 0 & \dot{\theta}_i & 0 \\ \dot{\theta}_i & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Using equation (34), we can find the linear velocity for each joint

and the collidable point, in the following manner.

$$i=0,\ V^1_1=0 \tag{35}$$

$$i=1\ ,\ V_2^2=R_1^2(V_1^1+\omega_1^1 \times P_2^1)=\begin{bmatrix} C_2 & S_2 & 0 \\ -S_2 & C_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & -\dot{\theta}_1 & 0 \\ \dot{\theta}_1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} l_1 \\ 0 \\ 0 \end{bmatrix}=\begin{bmatrix} S_2\dot{\theta}_1 l_1 \\ C_2\dot{\theta}_1 l_1 \\ 0 \end{bmatrix} \tag{36}$$

$$i=2\ ,\ V_3^3=R_2^3(V_2^2+\omega_2^2 \times P_3^2) \tag{37}$$

$$R_2^3=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},\ \omega_2^2 \times P_3^2=\begin{bmatrix} 0 & -(\dot{\theta}_1+\dot{\theta}_2) & 0 \\ (\dot{\theta}_1+\dot{\theta}_2) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} l_k \\ 0 \\ 0 \end{bmatrix},$$

$$V_3^3=\begin{bmatrix} S_2\dot{\theta}_1 l_1 \\ C_2\dot{\theta}_1 l_1+l_k(\dot{\theta}_1+\dot{\theta}_2) \\ 0 \end{bmatrix}. \tag{38}$$

$$i=3,\ V_4^4= R_3^4(V_3^3 + \omega_3^3 \times P_4^3) \quad , \tag{39}$$

$$R_3^4=\begin{bmatrix} C_3 & S_3 & 0 \\ -S_3 & C_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} , \tag{40}$$

$$\omega_3^3 \times P_4^3=\begin{bmatrix} 0 & -(\dot\theta_1+\dot\theta_2) & 0 \\ (\dot\theta_1+\dot\theta_2) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} l_2-l_k \\ 0 \\ 0 \end{bmatrix} , \tag{41}$$

$$V_4^4=\begin{bmatrix} S_{23}l_1\dot\theta_1+ S_3l_2(\dot\theta_1+\dot\theta_2) \\ C_{23}l_1\dot\theta_1+ C_3l_2(\dot\theta_1+\dot\theta_2) \\ 0 \end{bmatrix} . \tag{42}$$

$$i=4,\ V_5^5= R_4^5(V_4^4 + \omega_4^4 \times P_5^4) \quad , \tag{43}$$

$$R_4^5=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} , \tag{44}$$

$$\omega_4^4 \times P_5^4 = \begin{bmatrix} 0 & -(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3) & 0 \\ (\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} l_3 \\ 0 \\ 0 \end{bmatrix}, \quad (45)$$

$$V_5^5 = \begin{bmatrix} (S_{23}l_1+S_3l_2)\dot{\theta}_1+S_3l_2\dot{\theta}_2 \\ (C_{23}l_1+C_3l_2+l_3)\dot{\theta}_1+(C_3l_2+l_3)\dot{\theta}_2+l_3\dot{\theta}_3 \\ 0 \end{bmatrix}. \quad (46)$$

In order to determine the collidable point and end-effector velocities with reference to the base coordinates, we do the following:

$$R_3^0 = R_1^0 R_2^1 R_3^2 , \quad (47)$$

$$R_3^0 = \begin{bmatrix} C_{12} & -S_{12} & 0 \\ S_{12} & C_{12} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (48)$$

$$V_3^0 = R_3^0 V_3^3 = \begin{bmatrix} (-S_1l_1-S_{12}l_k)\dot{\theta}_1-S_{12}l_k\dot{\theta}_2 \\ (C_1l_1+C_{12}l_k)\dot{\theta}_1+C_{12}l_k\dot{\theta}_2 \\ 0 \end{bmatrix}. \quad (49)$$

In other words, the velocity of the collidable point k in base coordinates is :

$$V_3^0 = J_k\dot{\theta} = \begin{bmatrix} -S_1 l_1 - S_{12} l_k & -S_{12} l_k & 0 \\ C_1 l_1 + C_{12} l_k & C_{12} l_k & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}, \qquad (49)$$

where J_k is the Jacobian for point k.

$$R_5^0 = R_3^0 R_4^3 R_5^4 = \begin{bmatrix} C_{123} & -S_{123} & 0 \\ S_{123} & C_{123} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad (50)$$

$$V_5^0 = R_5^0 V_5^5 . \qquad (51)$$

Therefore, the end-effector velocity in base coordinates is:

$$V_5^0 = J_e \dot{\Theta} = \begin{bmatrix} (-S_1 l_1 - S_{12} l_2 - S_{123} l_3) & (-S_{12} l_2 - S_{123} l_3) & -S_{123} l_3 \\ (C_1 l_1 + C_{12} l_2 + C_{123} l_3) & (C_{12} l_2 + C_{123} l_3) & C_{123} l_3 \end{bmatrix} \dot{\Theta}, \qquad (52)$$

where J_e is the Jacobian for the end-effector point. The Pseudoinverse of the J_e in equation (32)

$$\dot{\Theta} = J_e^+ \dot{x}_e + (I - J_e^+ J_e) R_1^{-1} J_k^T \lambda, \qquad (32)$$

is computed using the following equation

$$J_e^+ = J_e^T (J_e J_e^T)^{-1}, \tag{53}$$

where J_e is assumed to have a full rank.

To compute λ for each point of the path, we need to solve the following two-point boundary value differential equations (reduced state and costate):

$$\dot{x}_k = \frac{\partial H}{\partial \lambda} = J_k \, u = J_k \, R_1^{-1} \, J_k^T \lambda \ , \tag{54}$$

$$\dot{\lambda} = -\frac{\partial H}{\partial x_k} = -Q_1 \, (x_k - c) \ . \tag{26}$$

In equation (26), the location of the collidable point, x_k, for each path interval, is found by considering the point of intersection of the perpendicular line drawn from the center of obstacle to the collidable link (2nd link), which passes through the points F & H (as shown in Figure 8).

The equation for the line along the second link may be written as:

$$y = m_1 x + b_1 \ , \tag{55}$$

where b_1 is a constant, and m_1 is the slope of the line, and it is related to the joint angles by

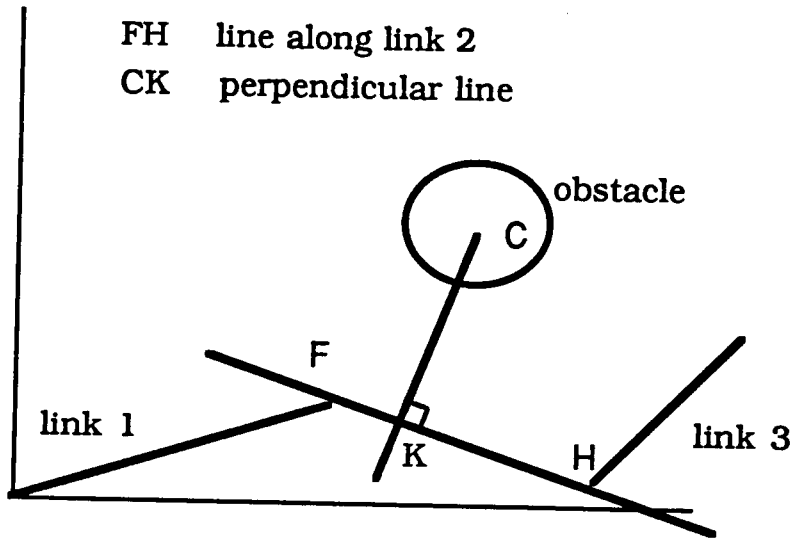


Figure 8. The collidable point is the point of intersection lines CK and FH

$$m_1 = \tan(\theta_1 + \theta_2). \tag{56}$$

Substituting the coordinates of point F in equation (55), we can calculate the constant b_1 as follows:

$$F = (l_1\cos(\theta_1), l_1\sin(\theta_1)), \tag{57}$$

$$b_1 = -\frac{l_1 S_2}{C_{12}} \cdot \tag{58}$$

Therefore, the equation for the line along the collidable link will be:

$$y = \tan(\theta_1 + \theta_2)\, x - \frac{l_1 S_2}{C_{12}} \cdot \tag{59}$$

The slope of the line passing through the center of the obstacle and perpendicular to the collidable link is equal to $-1/m_1$. Therefore, the equation of this line will be:

$$y = -\frac{C_{12}}{S_{12}} x + b_2 \ , \tag{60}$$

where b_2 is a constant. By substituting the coordinates of the center of obstacle in equation (60), we have

$$C = (x_c, y_c), \tag{61}$$

$$b_2 = y_c + \frac{C_{12}}{S_{12}} x \ . \tag{62}$$

Substituting b_2 in (60), we get

$$y = -\frac{C_{12}}{S_{12}}(x - x_c) + y_c \ . \tag{63}$$

Combining equations (59) and (63), we get

$$\begin{aligned} x_k &= C_{12}^2 x_c + y_c C_{12} S_{12} + S_{12} S_2 l_1 \ , \\ y_k &= S_{12} C_{12} x_k + S_{12}^2 y_c + \frac{S_2 S_{12}^2 l_1}{C_{12}} - \frac{l_1 S_2}{C_{12}} \ . \end{aligned} \tag{64}$$

The distance between the collidable point and the origin of the second joint, point F, which is used in calculation of J_k, will be

$$l_k = \sqrt{(x_f - x_k)^2 + (y_f - y_k)^2}$$

$$l_k = \sqrt{(l_1 C_1 - C_{12}^2 x_c - y_c C_{12} S_{12} - S_{12} S_2 l_1)^2 + (l_1 S_1 - S_{12} C_{12} x_c - S_{12}^2 y_c - \frac{S_{12}^2 S_2 l_1}{C_{12}} + \frac{S_2 l_1}{C_{12}})^2} \ . \tag{65}$$

To solve system equations (26), and (54), we have the following boundary conditions:

$$x_k(t_0) = x_0 \ , \tag{22}$$

$$\lambda\,(t_1) = \left[\ \lambda_1, \ldots\ldots\ldots, \lambda_m \right] = \left[\ 0, \quad \ldots\ldots, \ 0\ \right] \ . \tag{23}$$

In addition, we need to estimate the missing boundary conditions on λ at the initial point such that when we reach the right-hand end-point, the computed boundary conditions on λ for the endpoint are equal to the given conditions in equation (23). As previously mentioned there are four commonly used numerical techniques to solve the nonlinear two-point boundary value differential equations: Steepest descent, Variation of extremals, Quasi-linearization, and Gradient projection. We use the Variation of extremals technique [Appendix A] to solve the two-point boundary value problem represented by the following differential equations:

$$\dot{x}_k = \frac{\partial H}{\partial \lambda} = J_k\, u = J_k\, R_1^{-1}\, J_k^T \lambda \ , \tag{54}$$

$$\dot{\lambda} = -\frac{\partial H}{\partial x_k} = -Q_1\,(x_k - c) \ . \tag{26}$$

According to this technique, we need to compute the following expressions:

$$\frac{\partial^2 H}{\partial x_k \partial \lambda}, \frac{\partial^2 H}{\partial \lambda^2}, \frac{\partial^2 H}{\partial x_k^2},$$

which one used to compute

$$\lambda^{(i+1)}(t_0), \dot{\mathbf{P}}_x \text{ and } \dot{\mathbf{P}}_P,$$

while considering the equations (A.3-18) and (A.3-26a),

where P_x and P_P are the state and costate influence matrices respectively. Assuming R_1 as an identity matrix, equation (54) takes the form:

$$\dot{x}_k = J_k J_k^T \lambda , \tag{66}$$

where $J_k J_k^T$ is equal to

$$J_k J_k^T = \begin{bmatrix} l_1^2 S_1^2 + 2 l_k^2 S_{12}^2 + 2 l_1 l_k S_1 S_{12} & -l_1^2 S_1 C_1 - l_1 l_k S_1 C_{12} - 2 S_{12} C_{12} l_k^2 - C_1 S_{12} l_1 l_k \\ -l_1^2 S_1 C_1 - l_1 l_k S_1 C_{12} - 2 S_{12} C_{12} l_k^2 - C_1 S_{12} l_1 l_k & l_1^2 C_1^2 + 2 l_k^2 C_{12}^2 + 2 C_1 C_{12} l_1 l_k \end{bmatrix} , \tag{67}$$

or for convenience let

$$A = J_k J_k^T = \begin{bmatrix} A(1,1) & A(1,2) \\ A(2,1) & A(2,2) \end{bmatrix} . \tag{68}$$

Therefore, if we take the second partial derivatives of system equations (54) and (26) with respect to x_k and λ we have

$$\frac{\partial^2 H}{\partial x_k \partial \lambda} = 0 , \tag{69}$$

$$\frac{\partial^2 H}{\partial x_k^2} = Q_1 , \tag{70}$$

$$\frac{\partial^2 H}{\partial \lambda^2} = A = \begin{bmatrix} A(1,1) & A(1,2) \\ A(2,1) & A(2,2) \end{bmatrix} . \tag{71}$$

Substituting these expressions in equation (A.3-26a) for P_x and P_P, we get

$$\dot{P}_x = \frac{\partial^2 H}{\partial \lambda^2} P_P = A\, P_P , \tag{72}$$

$$\dot{P}_P = - Q_1 P_x , \tag{73}$$

$$\text{with boundary conditions} \quad \begin{cases} P_x(\lambda^{(i)}(t_0), t_0) = 0 \;\; 2 \times 2 \text{ matrix} \\ P_P(\lambda^{(i)}(t_0), t_0) = I \;\; 2 \times 2 \text{ matrix} \end{cases} \tag{74}$$

Solving simultaneously equations (54), (26), and (72) - (74) along

with using

$$\lambda^{(i+1)}(t_0) = \lambda^{(i)}(t_0) - \left[P_P(\lambda^{(i)}(t_0), t_1) \right]^{-1} \lambda^{(i)}(t_1), \qquad (75)$$

and considering the stopping criterion for perturbation part (equations (72)-(75)) as:

$$|| \lambda(t_1) || \approx 0$$

we get the optimal costate vector, λ, which can be substituted in the following equation

$$\dot{\Theta} = J_e^+ \dot{x}_e + (I - J_e^+ J_e) R_1^{-1} J_k^T \lambda, \qquad (32)$$

to calculate the safe joint trajectories.

The simulation results for the planar robot with equal links and for horizontal, vertical, and 45 degree paths are shown in Figures 9 through 14 and Tables 1 through 6. In all figures, the circle represents an obstacle and the second link is the collidable link. The end-effector starts its motion from point A and goes forward along the path to reach the final point B. In Figures 9, 11, and 13, only the least norm solution is used, and the robot goes through the circle while following the desired end-effector trajectories. On the other hand, in Figures 10, 12, and 14, we are applying the proposed scheme for the same trajectories, and the robot avoids

completely the obstacle while the end-effector remains perfectly on the path and tracks the desired trajectories. As can be seen from Figures 8,10, and 12, the robot motion is very smooth, and the collidable link avoids obstacle with good margin. The components of λ (t_1), in all cases, were very small and the stopping criterion was satisfied.

Table 1. Results for planar redundant robot motion along AB, least norm solution (Horizontal path)*

End-Effector X	End-Effector Y	Theta One	Theta Two	Theta Three	G(X)
13.16259	9.004515	35	325	68	-1.987629
12.65014	8.976815	34.19301	324.3519	75.10395	-2.388696
12.15571	8.980766	33.62367	324.2019	80.98966	-2.6523
11.65844	8.981377	33.14687	324.3216	86.40223	-2.858696
11.16074	8.982089	32.79557	324.6589	91.37109	-2.995937
10.66265	8.982664	32.57954	325.1708	95.9635	-3.060718
10.16427	8.983149	32.50739	325.8268	100.2268	-3.048852
9.665674	8.983576	32.58619	326.6044	104.1974	-2.956233
9.166902	8.983959	32.82209	327.4868	107.9031	-2.779333
8.667988	8.984324	33.2207	328.4605	111.3654	-2.515901
8.168928	8.984679	33.78708	329.5147	114.6007	-2.165897
7.669755	8.985051	34.52572	330.6407	117.621	-1.732666
7.170432	8.985428	35.44035	331.8305	120.4353	-1.22425
6.670968	8.98584	36.53379	333.0772	123.0492	-.6547368
6.171344	8.986269	37.80746	334.3739	125.4657	-.04570675
5.671544	8.986731	39.26125	335.7136	127.6857	+.5727916
5.171548	8.987206	40.89292	337.0892	129.7079	+1.161634
4.671353	8.987699	42.69784	338.4925	131.529	+1.673025
4.170943	8.988178	44.66853	339.9147	133.1444	+2.052078
3.670319	8.988632	46.79452	341.3456	134.5481	+2.239805

*DT, C_1, A_1, A_2 = .001, 1, 650, 650

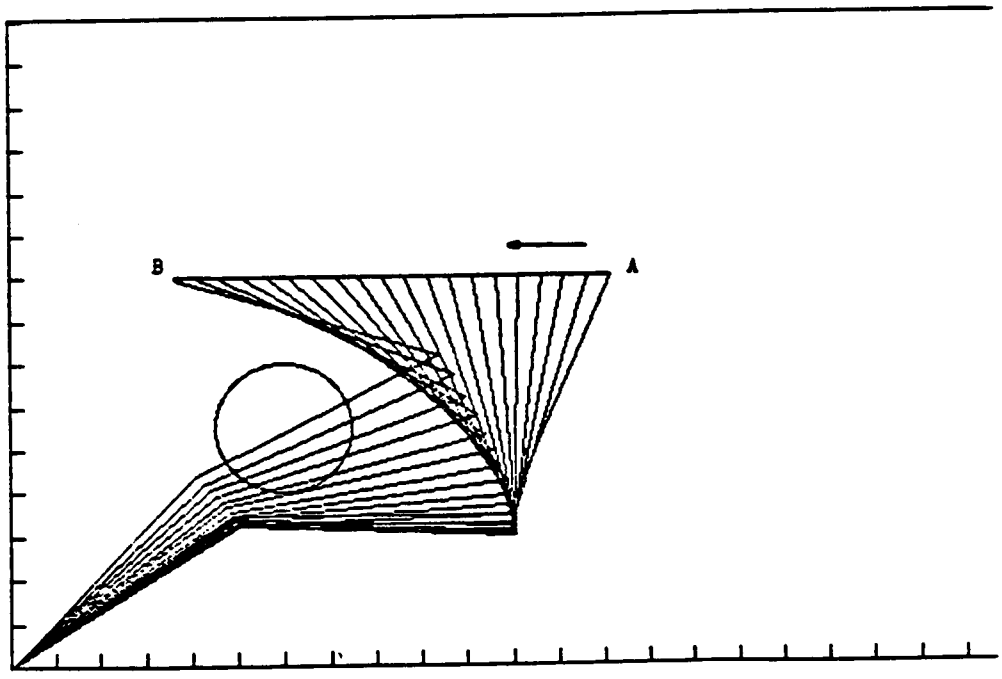


Figure 9. Planar redundant robot motion along AB, least norm solution (Horizontal path)

Table 2. Results for planar redundant robot motion along AB, null space used to avoid obstacle (Horizontal path)*

End-Effector X	End-Effector Y	Theta One	Theta Two	Theta Three	G(X)
13.16259	9.004515	35	325	68	-1.987629
12.63459	8.949032	31.22372	329.0654	75.19663	-3.439679
12.14485	8.964572	28.12408	333.7648	79.9042	-4.756135
11.64933	8.969501	25.35795	338.5101	83.82945	-6.071503
11.15302	8.974198	22.98454	343.146	87.08321	-7.286886
10.656	8.977762	20.98874	347.5748	89.86132	-8.340248
10.15845	8.980458	19.3553	351.7455	92.29662	-9.177553
9.660519	8.98248	18.0672	355.6374	94.48392	-9.759802
9.162259	8.983989	17.10751	359.249	96.48897	-10.06372
8.663725	8.985119	16.46005	362.5911	98.35527	-10.0809
8.164953	8.985983	16.10967	365.6817	100.1098	-9.816129
7.665972	8.986644	16.04228	368.5426	101.7675	-9.285782
7.166784	8.987169	16.2449	371.1968	103.334	-8.515976
6.667421	8.987601	16.70546	373.6661	104.8089	-7.541674
6.167878	8.987974	17.41267	375.9707	106.1877	-6.405362
5.668168	8.988305	18.35573	378.1275	107.4629	-5.156313
5.168285	8.988604	19.52415	380.1503	108.6256	-3.849107
4.668236	8.988887	20.90732	382.0491	109.6664	-2.542004
4.16803	8.989144	22.49418	383.8306	110.5755	-1.294777
3.667671	8.989386	24.27286	385.4983	111.3436	-.1656205

*First estimation for Lambda:Lambda(1,1)= Lambda(2,1)= 0, Optimal estimation after 45 iterations: Lambda(1,1)=11.81582,Lambda(2,1)=-33.14992, Costate vector for final point:Lambda(1,1)=5.960465E-06,Lambda(2,1)=-2.264977E-06,DT,C_1,A_1,A_2=.001,1,650,650

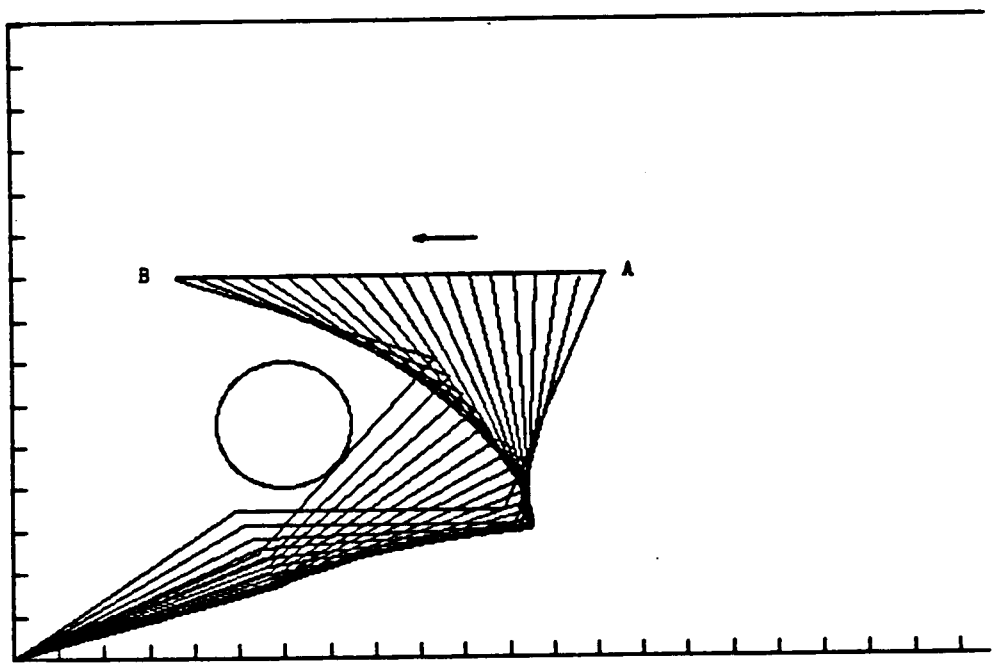


Figure 10. Planar redundant robot motion along AB, null space used to avoid obstacle(Horizontal path)

Table 3. Results for planar redundant robot motion along AB, least norm solution (Vertical path)*

End-Effector X	End-Effector Y	Theta One	Theta Two	Theta Three	G(X)
11.41505	8.895637	35	320	90	-2.352258
11.40274	9.39291	37.16132	320.9395	85.92606	-1.417182
11.40207	9.891459	39.35304	321.9435	81.38748	-.5195971
11.4013	10.39	41.56424	323.0437	76.46205	+.3096109
11.40021	10.88818	43.79741	324.2766	71.08125	+1.041
11.39859	11.38582	46.05496	325.6976	65.1415	+1.639667
11.39614	11.88256	48.33563	327.3937	58.48566	+2.061366
11.39216	12.37775	50.62818	329.5157	50.86	+2.247138

* DT, C_1, A_1, A_2= .0027,1, 950, 950

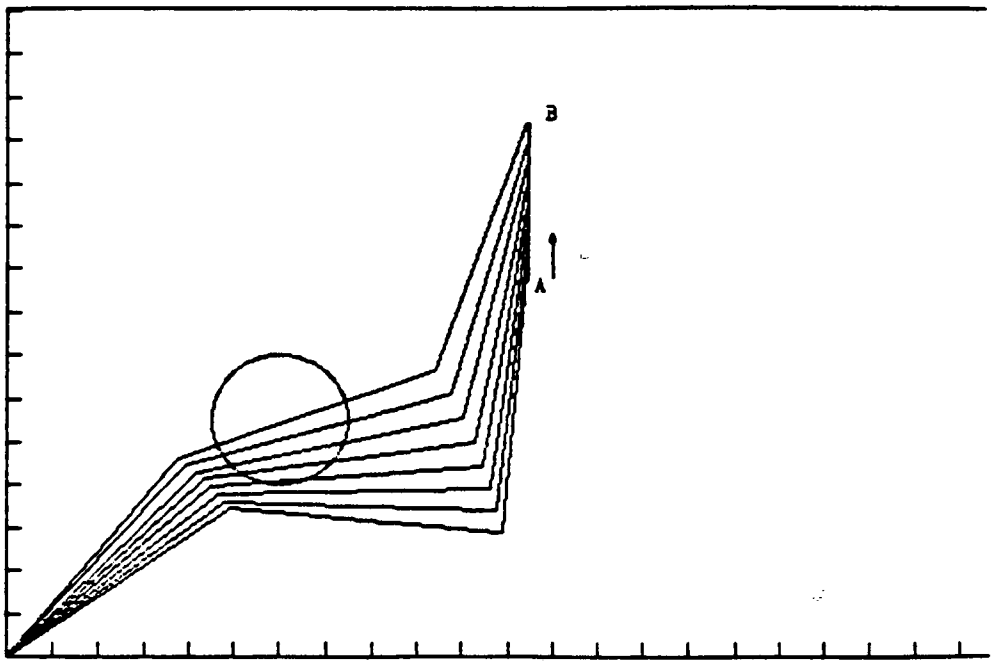


Figure 11. Planar redundant robot motion along AB, least norm solution (Vertical path)

Table 4. Results for planar redundant robot motion along AB, null space used to avoid obstacle (Vertical path)*

End-Effector X	End-Effector Y	Theta One	Theta Two	Theta Three	G(X)
11.41505	8.895657	35	320	90	-2.352258
11.30758	9.364189	29.04372	335.2756	84.92557	-4.115402
11.34062	9.881849	25.31983	347.3519	76.72566	-5.236135
11.3722	10.38191	23.49392	355.9137	68.58218	-5.401955
11.39048	10.88195	23.42568	361.3302	61.07346	-4.621373
11.39824	11.38178	24.77271	364.2287	54.27316	-3.24338
11.39969	11.88033	27.23707	265.2023	47.95543	-1.620614
11.39748	12.37659	30.64682	364.6846	41.7584	-.02171588

*First estimation for Lambda:Lambda(1,1)= Lambda(2,1)= 0, Optimal estimation after 21 iterations:Lambda(1,1)=12.34198,Lambda(2,1)=-41.09684, Costate vector for final point:Lambda(1,1)=-1.430512E-06,Lambda(2,1)=4.29153E-06,DT,C_1,A_1,A_2=.0027,1,950,950

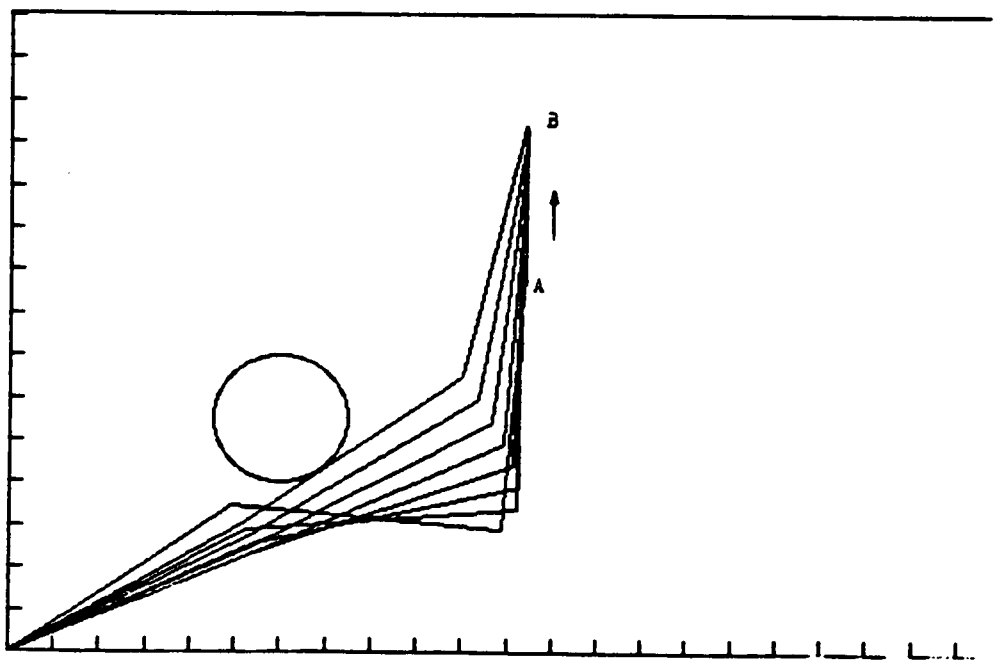


Figure 12. Planar redundant robot motion along AB, null space used to avoid obstacle (Vertical path)

Table 5. Results for planar redundant robot motion along AB, least norm solution (45 degree path)*

End-Effector X	End-Effector Y	Theta One	Theta Two	Theta Three	G(X)
13.16259	9.004515	35	325	68	-1.987629
12.80334	9.349558	35.92162	325.5491	69.60049	-1.552502
12.4498	9.704035	36.96572	326.2033	70.72023	-1.07122
12.0963	10.05807	38.08227	326.9096	71.55145	-.5767481
11.74272	10.41207	39.27232	327.6551	72.10426	-.07898641
11.38908	10.76601	40.53469	328.4287	72.38816	+.4094452
11.03539	11.1199	41.86784	329.2214	72.41051	+.8741528
10.68167	11.47373	43.2701	330.0256	72.17647	+1.299273
10.32793	11.82749	44.73978	330.8356	71.68916	+1.667671
9.974167	12.18119	46.27544	331.6469	70.94973	+1.96127
9.620408	12.53481	47.87595	332.4559	69.95695	+2.161188
9.266636	12.88834	49.54073	333.2608	68.70721	+2.24794

* DT, C_1, A_1, A_2= .0018,1, 920, 920

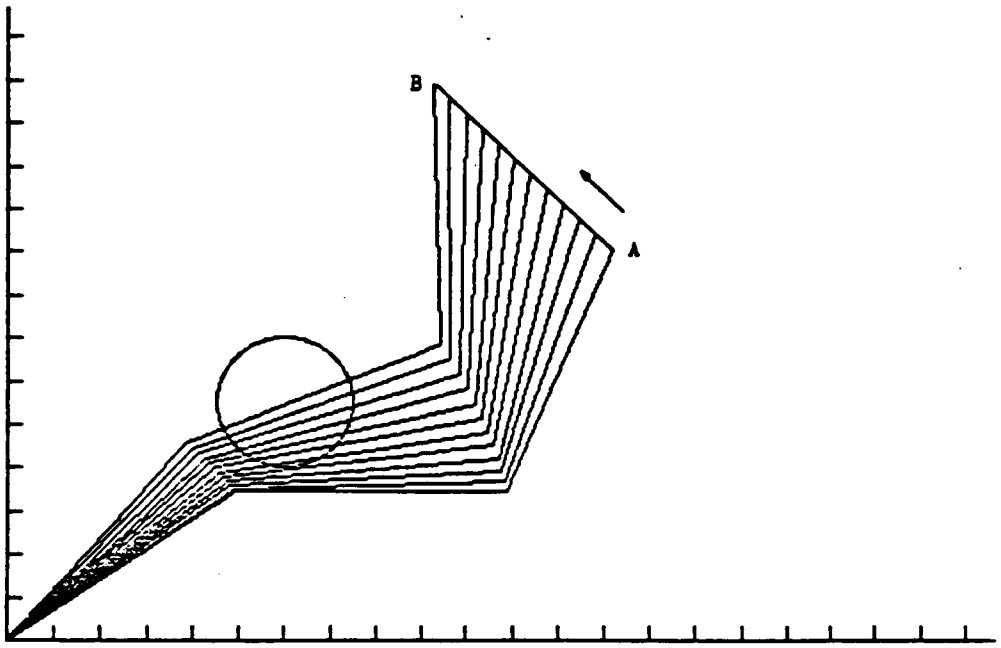


Figure 13. Planar redundant robot motion along AB, least norm solution (45 degree path)

Table 6. Results for planar redundant robot motion along AB, null space used to avoid obstacle (45 degree path)*

End-Effector X	End-Effector Y	Theta One	Theta Two	Theta Three	G(X)
13.16259	9.004515	35	325	68	-1.987629
12.73737	9.293058	29.02454	336.4976	69.8158	-4.021901
12.4043	9.686051	25.23411	346.6022	67.55705	-5.379221
12.06809	10.04855	22.67069	354.6891	64.93052	-6.182626
11.72605	10.40673	21.20881	360.9584	62.32517	-6.334786
11.37973	10.76284	20.62671	365.7082	59.96687	-5.954037
11.03037	11.11794	20.73151	369.2478	57.90423	-5.20427
10.67913	11.47246	21.37168	371.8402	56.103	-4.233499
10.32678	11.82663	22.43478	373.6895	54.50225	-3.157082
9.973798	12.18058	23.83988	374.9471	53.03931	-2.060433
9.620443	12.53436	25.52974	375.7224	51.6586	-1.006642
9.266872	12.888	27.46423	376.0939	50.31266	-.04361141

*First estimation for Lambda:Lambda(1,1)= Lambda(2,1)= 0, Optimal estimation after 18 iterations:Lambda(1,1)=17.67001,Lambda(2,1)=-40.82936, Costate vector for final point:Lambda(1,1)=-1.430512E-05,Lambda(2,1)=-2.3126E-05,DT,C_1,A_1,A_2=.0018,1,920,920

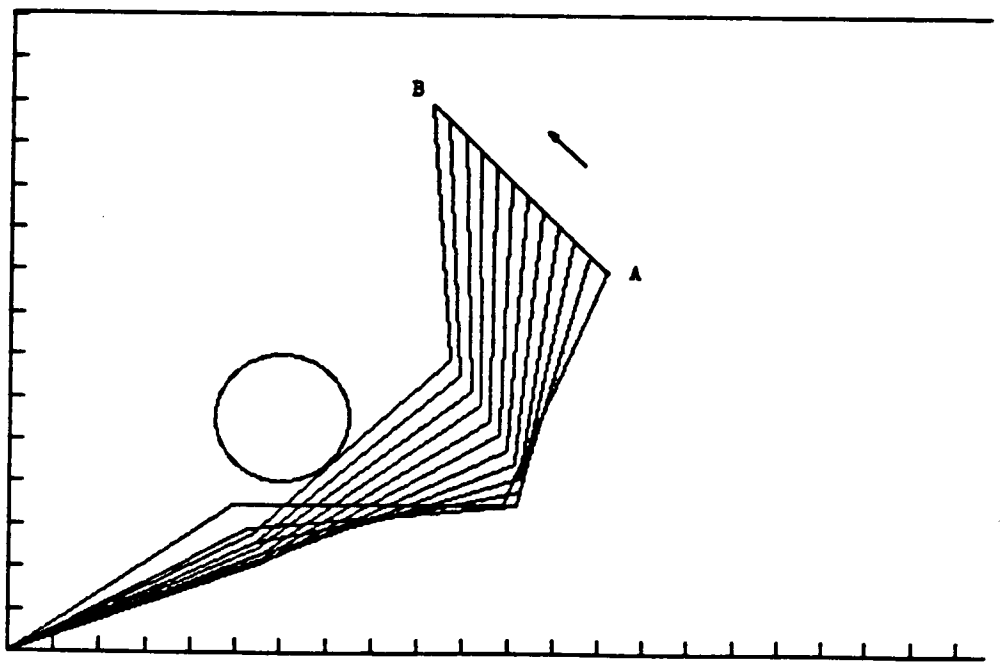


Figure 14. Planar redundant robot motion along AB, null space used to avoid obstacle (45 degree path)

V. CONCLUSIONS AND RECOMMENDATIONS

A global optimization scheme for obstacle avoidance problem was presented. An integral type of performance index was defined for obstacle avoidance. The Cartesian coordinates of the end-effector position were used as the state variable. By applying modern control theory, the redundancy of the robot manipulator was utilized to avoid obstacles.

This scheme has several advantages as compared to the presently available schemes. Unlike the schemes available in the literature, the proposed scheme minimizes the control effort required to perform the task of avoiding obstacles. This results in bounded joint velocities. This scheme maximizes the distance between the collidable link and the safety zone of the obstacle. Therefore, it performs the obstacle avoidance for the whole collidable link. It does not require the computation of the gradient of complex expressions in symbolic form. Thus, it is well suited for robots with seven or more degrees of freedom. To use numerical techniques, we need to compute different partial derivatives with respect to the state and costate vectors. These expressions get simple form in the way that this problem has been set up. As seen from the simulation results, this numerical technique converges well.

As recommendations for the future research, this scheme

should be implemented for robots with seven or more degrees of freedom. It can be expanded to moving obstacles. It may also be applied to the cluttered environment with multiple obstacles.

REFERENCES

REFERENCES

Albert, A. , Regression and the Moore-Penrose pseudoinverse, Academic press, 1972.

Baillieul, J. , Avoiding Obstacles and Resolving Kinematic Redundancy, Proc. of IEEE conf. on Robotics and Automation,1698-1704. Son Francisco. 1986

Barnett,S. , Matrices in control theory,Krieger Publishing Co.,FL.,1984.

Barnett, S. , Introduction to mathematical control theory, Clarendon Press. Oxford 1975.

Bronson, R. , Matrix Methods, Academic Press 1969.

Craig, J. J. , Introduction to Robotics: Mechanics and Control, Addison- Wesley Publishing company, 1986.

Dubey, R., and Luh, J. Y.S., Redundant Robot Control for higher Flexibility, in proc. IEEE conf. Robotics and Automation, Raleigh, North Carolina, 1066-72 , 1987.

Kirk, E. D. , Optimal control theory, Prentice-Hall, Inc. 1970.

Khatib, O. and Lemaitre, J.F. , 1978 september 12-15, Udine, Italy, Dynamic control of manipulators operating in a complex environment. Proc. 3rd CISM-IFT. MM Symp. Theory practice Robots manipulators, 267-282. Elsevier. 1979.

Khatib, O. 1985, Real-time obstacle avoidance for manipulators and mobil Robots. IEEE conf. on Robotics and Automation, St. Louis. March 25-28, pp. 500-505.

Lozano-Perez, T. 1981, Automatic planning of manipulator transfer movements. IEEE Trans. Sys. Man,Cyber. SMC-11(10):681-698.

Lee, B. H. and Chien, Y. P. 1987,Time-Varying Obstacle Avoidance for Robot Manipulators: Approaches and Difficulties, IEEE conf. on Robotics and Automation,1610-1615. Raleigh.

Lee, B. E., and Markus, L. , Foundations of optimal control theory, John Willey & Sons, Inc. 1967.

Maciejewski, A. A., and Klein, C. A. 1985, Obstacle avoidance for kinematically redundant manipulators in dynamically varynig environments. In Robtics Research,V.4,No.3, pp. 109-117.

Nakamura,Y. , and Hanafusa,H. ,1987, Optimal Redundancy Control of Robot Manipulators. Robotics Research, Vol.6, No.1, pp.32-42, MIT Press.

Nobel, B. and Daniel,J.W., Applied linear Algebra, Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1977.

Pontryagin, L. S., et al. 1962, The mathematical theory of optimal process, New York: Wiley.

Ra0,C.R. and Mitra, S. K. 1971, Generalized inverse of matrices and its applications. New York: Wiley.

Udupa, S. 1977, Collision detection and avoidance in computer controlled manipulators, 5`th Int. Joint Conf. Artificial Intell. Cambridge: MIT.

Yoshikawa, T. 1984, Analysis and control of robot manipulators with redundancy. In Robotics Research, eds. M. Brady and R. Paul, pp.735-747, Cambridge: MIT Press.

APPENDICES

APPENDIX A

VARIATION OF EXTREMALS+

The iterative numerical technique that we shall discuss in this section is called *variation of extremals*, because every trajectory generated by the algorithm satisfies Eqs. (A.1-1) through(A.1-3) and hence is an extremal.

$$\dot{\mathbf{x}}^*(t)=\frac{\partial \wp}{\partial \mathbf{P}}=\mathbf{a}\left(\mathbf{x}^*(t),\mathbf{u}^*(t),t\right) \tag{A.1-1}$$

$$\dot{\mathbf{P}}^*(t)=\frac{\partial \wp}{\partial \mathbf{x}}=-\left[\frac{\partial \mathbf{a}}{\partial \mathbf{x}}\left(\mathbf{x}^*(t),\mathbf{u}^*(t),t\right)\right]^T\mathbf{P}^*(t)-\frac{\partial g}{\partial \mathbf{x}}\left(\mathbf{x}^*(t),\mathbf{u}^*(t),t\right) \tag{A.1-2}$$

$$\mathbf{0}=\frac{\partial \wp}{\partial \mathbf{u}}=\left[\frac{\partial \mathbf{a}}{\partial \mathbf{u}}\left(\mathbf{x}^*(t),\mathbf{u}^*(t),t\right)\right]^T\mathbf{P}^*(t)+\frac{\partial g}{\partial \mathbf{u}}\left(\mathbf{x}^*(t),\mathbf{u}^*(t),t\right) \tag{A.1-3}$$

$$\mathbf{x}^*(t)=\mathbf{x}_0 \tag{A.1-4a}$$

$$\mathbf{P}^*(t_f)=\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\left(\mathbf{x}^*(t_f)\right) \tag{A.1-4b}$$

To illustrate the basic concept of the algorithm, let us consider a simple example.

A First-order Optimal Control Problem

Suppose that a first-order system

+ This material has been taken completely from : *Donald E. Kirk,* optimal control theory.

$$\dot{x}(t) = a(x(t), u(t), t) \tag{A.3-1}$$

is to be controlled to minimize a performance measure of the form

$$J = \int_{t_0}^{t_f} g(x(t), u(t), t)\, dt \tag{A.3-2}$$

where $\mathbf{x}(t_0)=\mathbf{x}_0$ is given, t_0 and t_f are specified, and the admissible state and control values are not constrained by any boundaries. If the equation [corresponding to (A.1-3)]

$$\frac{\partial \mathcal{H}}{\partial u} = 0 \tag{A.3-3}$$

is solved for the control in terms of the state and costate and substitute in the state and costate equations, the reduced differential equations

$$\begin{aligned} \dot{x}(t) &= a(x(t), P(t), t) \\ \dot{P}(t) &= d(x(t), P(t), t) \end{aligned} \tag{A.3-4}$$

are obtained. In general, d is a nonlinear function of $\mathbf{x}(t)$, $\mathbf{P}(t)$, and t. Since h=0 in the performance measure, Eq. (A.1-4b) gives $\mathbf{P}(t_f)=0$. To determine an optimal trajectory, we must find a solution of Eq. (A.3-4) that satisfies the boundary conditions $x(t_0)=x_0$, $\mathbf{P}(t_f)=0$.

If $\mathbf{P}(t_0)$ were known, Eq. (A.3-4) could be solved by using

numerical integration. Since this is not the case, suppose we guess a value $\mathbf{P}^{(0)}(t_0)$ for the initial costate, and use this initial value to integrate numerically (A.3-4) from t_0 to t_f; we denote the costate solution obtained by this integration as $\mathbf{P}^{(0)}$. If we should guess an initial costate value that causes $\mathbf{P}^{(0)}(t_f)$ to be zero, the tw0-point boundary value problem is solved. In general, however, it will turn out that the final costate will not equal zero. Notice that the value obtained for the terminal costate, $\mathbf{P}^{(0)}(t_f)$, will depend on the number chosen for $\mathbf{P}^{(0)}(t_0)$; in other words, $\mathbf{P}(t_f)$ is a function of $\mathbf{P}(t_0)$. Unfortunately, an analytical expression for this function is not known, nor is it readily determined; however, *values* of the function [such as $\mathbf{P}^{(0)}(t_f)$] can be found by using selected values of $\mathbf{P}(t_0)$ to integrate numerically the reduced state-costate equations. The method of variation of extremals is an algorithm that uses the observed values of $\mathbf{P}(t_f)$ to adjust systematically the guessed values of $\mathbf{P}(t_0)$. One technique for making systematic adjustments of the initial costate values is based on Newton's method for finding roots of nonlinear equations. In general, using this algorithm gives

$$P^{(i+1)}(t_0) = P^{(i)}(t_0) - \left[\frac{dP(t_f)}{dP(t_0)}\Big|_{P^{(i)}(t_0)}\right]^{-1} P^{(i)}(t_f) \qquad \text{(A.3-10a)}$$

as the expression for the (i+1)st trial value of $\mathbf{P}(t_0)$. We note that

if the desired value of the final costate $\mathbf{P}(t_f)$ is some nonzero constant $\mathbf{P}_f$, then Eq.(A.3-10a) must be modified to read

$$P^{(i+1)}(t_0) = P^{(i)}(t_0) - \left[\frac{dP(t_f)}{dP(t_0)}\Big|_{P^{(i)}(t_0)}\right]^{-1}\left[P^{(i)}(t_f) - P_f\right] \qquad \text{(A.3-17)}$$

If we have 2n first-order differential equations (n state equations and n costate equations), the matrix generalization of Eq.(A.3-10a) is

$$\mathbf{P}^{(i+1)}(t_0) = \mathbf{P}^{(i)}(t_0) - \left[\mathbf{P}_P(\mathbf{P}^{(i)}(t_0), t_f)\right]^{-1}\mathbf{P}^{(i)}(t_f) \qquad \text{(A.3-18)}$$

where $\mathbf{P}_p(\mathbf{P}^{(i)}(t_0), t)$ is the nxn matrix of partial derivatives of the components of $\mathbf{P}(t)$ with respect to each of the components of $\mathbf{P}(t_0)$, evaluated at $\mathbf{P}^{(i)}(t_0)$; that is,

$$\mathbf{P}_P(\mathbf{P}^{(i)}(t_0), t) = \begin{bmatrix} \frac{\partial P_1(t)}{\partial P_1(t_0)} & \cdots & \frac{\partial P_1(t)}{\partial P_n(t_0)} \\ \vdots & \cdots & \vdots \\ \frac{\partial P_n(t)}{\partial P_1(t_0)} & \cdots & \frac{\partial P_n(t)}{\partial P_n(t_0)} \end{bmatrix}_{P^{(i)}(t_0)} \qquad \text{(A.3-19)}$$

The $\mathbf{P}_P$ matrix indicates the influence of changes in the initial costate on the costate trajectory at time t; hence, we shall call $\mathbf{P}_P$

the *costate influence function matrix.* Notice that (A.3-18) requires that $\mathbf{P}_P$ be known only at the terminal time t_f. Equation (A.3-18) is appropriate only if the desired value of the final costate is zero, which occurs if the term $h(\mathbf{x}(t_f))$ is missing from the performance measure. If, however, h is not absent from the performance measure, it can be shown that the appropriate equation for the iterative procedure is

$$\mathbf{P}^{(i+1)}(t_0) = \mathbf{P}^{(i)}(t_0) + \left\{ \left[\left[\frac{\partial^2 h}{\partial \mathbf{x}^2}(\mathbf{x}(t_f)) \right] \mathbf{P}_x(\mathbf{P}(t_0), t_f) - \mathbf{P}_P(\mathbf{P}(t_0), t_f) \right]_i \right\}^{-1} \cdot \left[\mathbf{P}(t_f) - \frac{\partial h}{\partial \mathbf{x}}(\mathbf{x}(t_f)) \right]_i \tag{A.3-20}$$

where $\mathbf{P}_x(\mathbf{P}^{(i)}(t_0), t_f)$ is the nxn *state influence function matrix*

$$\mathbf{P}_x(\mathbf{P}^{(i)}(t_0), t) = \left[\begin{array}{ccc} \frac{\partial x_1(t)}{\partial P_1(t_0)} & \cdots & \frac{\partial x_1(t)}{\partial P_n(t_0)} \\ \vdots & & \vdots \\ \frac{\partial x_n(t)}{\partial P_n(t_0)} & & \frac{\partial x_n(t)}{\partial P_n(t_0)} \end{array} \right]_{\mathbf{P}^{(i)}(t_0)} \tag{A.3-21}$$

evaluated at $t=t_f$. The notation $[\ ]_i$ means that the enclosed terms are evaluated on the ith trajectory, and

$$\frac{\partial^2 h}{\partial \mathbf{x}^2}(\mathbf{x}(t_f))$$

is the matrix whose jkth element is

$$\left[\frac{\partial^2 h}{\partial \mathbf{x}^2}(\mathbf{x}(t_f))\right]_{Jk} = \frac{\partial^2 h}{\partial x_J \partial x_k}(\mathbf{x}(t_f))$$

Notice that if h=0, (A.3-20) reduces to Eq. (A.3-18).

Determination Of The Influence Function Matrices

Conceptually we may think of finding the $\mathbf{P}_P$ and $\mathbf{P}_x$ matrices at $t=t_f$ by the following finite difference procedure:

1. Using $\mathbf{P}(t_0)=\mathbf{P}^{(i)}(t_0)$ and $\mathbf{x}(t_0)=\mathbf{x}_0$, integrate the reduced state and costate equations from t_0 to t_f, and store the resulting values of $\mathbf{P}^{(i)}(t_f)$ and $\mathbf{x}^{(i)}(t_f)$.

2. Perturb the first component of the vector $\mathbf{P}^{(i)}(t_0)$ by an amount $\delta P_1(t_0)$; again integrate the reduced state and costate equations from t_0 to t_f. The first column of $\mathbf{P}_p(\mathbf{P}^{(i)}(t_0), t_f)$ is found from the relationship

$$\frac{\partial \mathbf{P}(t_f)}{\partial P_1(t_0)}\Big|_{\mathbf{P}^{(i)}(t_0)} \doteq \frac{\delta \mathbf{P}(t_f)}{\delta P_1(t_0)}, \qquad \text{(A.3–22)}$$

where $\delta\mathbf{P}(t_f)$ is found by subtracting the value of $\mathbf{P}^{(i)}(t_f)$ generated in step 1 from the value of $\mathbf{P}(t_f)$ generated by using the perturbed value of $\mathbf{P}(t_0)$. Similarly, the first column of $\mathbf{P}_x(\mathbf{P}^{(i)}(t_0),t_f)$ is found from

$$\frac{\partial \mathbf{x}(t_f)}{\partial P_1(t_0)} \Big|_{P^{(i)}(t_0)} \doteq \frac{\delta \mathbf{x}(t_f)}{\delta P_1(t_0)} \qquad \text{(A.3–23)}$$

3. The remaining columns of the influence function matrices at $t=t_f$ are generated by perturbing each of the components of $\mathbf{P}(t_0)$, with all other components at their initial settings, and evaluating the ratios of the changes in $\mathbf{P}(t_f)$ and $\mathbf{x}(t_f)$ to the change in the appropriate component of $\mathbf{P}(t_0)$.

If the influence function matrices are computed by using this procedure, the integrations of steps 1, 2, and 3 would probably be performed simultaneously.

One rather apparent difficulty with this finite difference procedure is the selection of the perturbations. Relatively large perturbations may cause the difference approximations of the partial derivatives to be inaccurate, whereas, on the other hand, very small perturbations may significantly increase the effects of inaccuracies caused by numerical integration, and trunction and round-off errors. These difficulties can be avoided, however, by using a different method to evaluate the influence function matrices; let us now discuss this alternative procedure.

Assume that $\partial \wp / \partial \mathbf{u}$ has been solved for $\mathbf{u}(t)$ and used to obtain the reduced state and costate differential equations

$$\dot{\mathbf{x}}(t) = \frac{\partial \wp}{\partial \mathbf{P}}(\mathbf{x}(t), \mathbf{P}(t), t)$$

$$\dot{\mathbf{P}}(t) = -\frac{\partial \wp}{\partial \mathbf{x}}(\mathbf{x}(t), \mathbf{P}(t), t) \qquad \text{(A.3-24)}$$

Taking the partial derivatives of these equations with respect to the initial value of the costate vector gives

$$\frac{\partial\left[\dot{\mathbf{x}}(t)\right]}{\partial \mathbf{P}(t_0)} = \frac{\partial\left[\frac{\partial \wp}{\partial \mathbf{P}}(\mathbf{x}(t), \mathbf{P}(t), t)\right]}{\partial \mathbf{P}(t_0)}$$

$$\frac{\partial\left[\dot{\mathbf{P}}(t)\right]}{\partial \mathbf{P}(t_0)} = \frac{\partial\left[-\frac{\partial \wp}{\partial \mathbf{x}}(\mathbf{x}(t), \mathbf{P}(t), t)\right]}{\partial \mathbf{P}(t_0)} \qquad \text{(A.3-25)}$$

If it is assumed that $\partial[d\mathbf{x}/dt]/\partial\mathbf{P}(t_0)$ and $\partial[d\mathbf{P}/dt]/\partial\mathbf{P}(t_0)$ are continuous with respect to $\mathbf{P}(t_0)$ and t, the order of differentiation can be interchanged on the left side of (A.3-25); doing this and using the chain rule on the right-hand side, we obtain

$$\frac{d\left[\frac{\partial \mathbf{x}(t)}{\partial \mathbf{P}(t_0)}\right]}{dt} = \left[\frac{\partial^2 \wp}{\partial \mathbf{P} \partial \mathbf{x}}(\mathbf{x}(t), \mathbf{P}(t), t)\right]\frac{\partial \mathbf{x}(t)}{\partial \mathbf{P}(t_0)} + \left[\frac{\partial^2 \wp}{\partial \mathbf{P}^2}(\mathbf{x}(t), \mathbf{P}(t), t)\right]\frac{\partial \mathbf{P}(t)}{\partial \mathbf{P}(t_0)}$$

$$\frac{d\left[\frac{\partial \mathbf{P}(t)}{\partial \mathbf{P}(t_0)}\right]}{dt} = \left[-\frac{\partial^2 \wp}{\partial \mathbf{x}^2}(\mathbf{x}(t), \mathbf{P}(t), t)\right]\frac{\partial \mathbf{x}(t)}{\partial \mathbf{P}(t_0)} - \left[\frac{\partial^2 \wp}{\partial \mathbf{x} \partial \mathbf{P}}(\mathbf{x}(t), \mathbf{P}(t), t)\right]\frac{\partial \mathbf{P}(t)}{\partial \mathbf{P}(t_0)} \qquad \text{(A.3-26)}$$

The indicated partial derivatives are nxn matrices having jkth

elements

$$\left[\frac{\partial^2 \wp}{\partial P \partial x}\right]_{Jk} = \frac{\partial^2 \wp}{\partial P_J \partial x_k}, \text{ and } \left[\frac{\partial x(t)}{\partial P(t_0)}\right]_{Jk} = \frac{\partial x_J(t)}{\partial P_k(t_0)}, \text{ etc.}$$

Notice that if the definitions of the influence function matrices given by (A.3-19) and (A.3-21) are used, Eq. (A.3-26) becomes

$$\frac{d\left[P_x(P^{(i)}(t_0), t)\right]}{dt} = \left[\frac{\partial^2 \wp}{\partial P \partial x}(t)\right]_i P_x(P^{(i)}(t_0), t) + \left[\frac{\partial^2 \wp}{\partial P^2}(t)\right]_i P_P(P^{(i)}(t_0), t)$$

(A.3-26a)

$$\frac{d\left[P_P(P^{(i)}(t_0), t)\right]}{dt} = \left[-\frac{\partial^2 \wp}{\partial x^2}(t)\right]_i P_x(P^{(i)}(t_0), t) + \left[\frac{\partial^2 \wp}{\partial x \partial P}(t)\right]_i P_P(P^{(i)}(t_0), t)$$

where $[\ ]_i$ indicates that the enclosed matrices are evaluated on the trajectory $\mathbf{x}^{(i)}$, $\mathbf{P}^{(i)}$ obtained by integrating the reduced state-costate equations with initial conditions $\mathbf{x}(t_0)=\mathbf{x}_0$, $\mathbf{P}(t_0)=\mathbf{P}^{(i)}(t_0)$. Notice that (A.3-26a) represents a set of $2n^2$ first-order differential equations involving the influence function matrices. The matrices $\mathbf{P}_x(\mathbf{P}^{(i)}(t_0),t_f)$ and $\mathbf{P}_P(\mathbf{P}^{(i)}(t_0),t_f)$ can be obtained by integrating these differential equations simultaneously with the reduced state-costate differential equations. The appropriate initial conditions for the influence function equations are

$$\mathbf{P}_x(\mathbf{P}^{(i)}(t_0), t_0) = \frac{\partial \mathbf{x}(t_0)}{\partial \mathbf{P}(t_0)} \Big|_{\mathbf{P}^{(i)}(t_0)} = 0 \quad (\text{the n x n zero matrix}) \qquad \text{(A.3-27a)}$$

$$\mathbf{P}_P(\mathbf{P}^{(i)}(t_0), t_0) = \frac{\partial \mathbf{P}(t_0)}{\partial \mathbf{P}(t_0)} \Big|_{\mathbf{P}^{(i)}(t_0)} = \mathbf{I} \quad (\text{the n x n identity matrix}) \qquad \text{(A.3-27b)}$$

Equation (A.3-27a) follows because a change in any of the components of $\mathbf{P}(t_0)$ does not affect the value of $\mathbf{x}(t_0)$; the state values are specified at time t_0. A change in the jth component of $\mathbf{P}(t_0)$ changes only $P_j(t_0)$; hence the result shown in (A.3-27b) is obtained.

The Variation Of Extremals Algorithm

1. Form the reduced differential equations by solving $\partial \wp / \partial \mathbf{u} = \mathbf{0}$ for $\mathbf{u}(t)$ in terms of $\mathbf{x}(t)$, $\mathbf{P}(t)$, and substituting in the state and costate equations [which then contain only $\mathbf{x}(t)$, $\mathbf{P}(t)$, and t].

2. Guess $\mathbf{P}^{(0)}(t_0)$, an initial value for the costate, and set the iteration index i to zero.

3. Using $\mathbf{P}(t_0) = \mathbf{P}^{(i)}(t_0)$ and $\mathbf{x}(t_0) = \mathbf{x}_0$ as initial conditions, integrate the reduced state-costate equations and the influence function equations (A.3-26a), with initial conditions (A.3-27), from t_0 to t_f. Store only the values $\mathbf{P}^{(i)}(t_f)$, $\mathbf{x}^{(i)}(t_f)$, and the nxn matrices $\mathbf{P}_p(\mathbf{P}^{(i)}(t_0), t_f)$ and $\mathbf{P}_x(\mathbf{p}^{(i)}(t_0), t_f)$.

4. Check to see if the termination criterion $\| \mathbf{P}^{(i)}(t_f) - \partial h(\mathbf{x}^{(i)}(t_f))/\partial \mathbf{x} \| < \gamma$ is satisfied. If it is, use the final iterate of $\mathbf{P}^{(i)}(t_0)$ to reintegrate the state and costate

equations and print out (or graph) the optimal trajectory and the optimal control. If the stopping criterion is not satisfied, use the iteration equation (A.3-20) to determine the value for $\mathbf{P}^{(i+1)}(t_0)$, increase i by one, and return to step 3. Notice that steps 1 and 2 are performed off-line by the user or programmer; the computer program consists of steps 3 and 4.

APPENDIX B

PONTRYAGIN'S MAXIMUM PRINCIPLE *

According to modern control theory and Pontryagin's maximum principle, which is a generalization of the classical calculus of variations, for the fundamental system of equations :

$$\frac{dx^i}{dt} = f^i(x,u) \qquad \text{for } i = 0, 1, \ldots\ldots n \qquad (B.1)$$

we have the following :

Theorem 1 : Let u(t), $t_0 <= t <= t_1$ be an admissible control such that the corresponding trajectory x(t) which begins at the point x_0 at the time t_0 passes, at some time t_1, through the point x_1. In order that u(t) and x(t) be optimal it is necessary that there exist a nonzero continuous vector function $\psi(t) = (\psi_0(t), \ldots\ldots \psi_n(t))$ corresponding to u(t) and x(t) such that :

1- for every t, $t_0 <= t <= t_1$, the function $\wp (\psi(t), x(t), u)$ of the variable $u \in U$ attains its maximum at the point u = u(t)

$$\wp (\psi(t), x(t), u(t)) = m (\psi(t), x(t)) \qquad (B.2)$$

2- at the terminal time t_1 the relations

---

* This material has been used completely from : *Pontryagin, L. S., et al.* 1962, The matematical theory of optimal process.

$$\psi_0(t_1) <= 0 \quad , \quad m(\psi(t_1),\ x(t_1)\ ) = 0 \tag{B.3}$$

are satisfied. Furthermore it turns out that if $\psi(t)$, $x(t)$ and $u(t)$ satisfy systems

$$\frac{dx^i}{dt} = \frac{\partial \wp}{\partial \psi_i} \qquad i = 0,1, \ldots\ldots n \tag{B.4}$$

$$\frac{d\psi_i}{dt} = -\frac{\partial \wp}{\partial x^i} \qquad i = 0,1, \ldots\ldots n \tag{B.5}$$

and condition 1, the time functions $\psi_0(t)$ and $m(\psi(t),\ x(t)\ )$ are constant. Thus condition 2 may be verified at any time t, $t_0 <= t <= t_1$ and not just at t_1.

Theorem 2 : Let $u(t)$, $t_0<=t<=t_1$ be an admissible control which transfers the phase point from x_0 to x_1 and let $x(t)$ be the corresponding trajectory so that $x(t_0)=x_0$, $x(t_1)=x_1$. In order that $u(t)$ and $x(t)$ be time-optimal it is necessary that there exist a nonzero continuous vector function $\psi(t)= (\ \psi_1(t), \ldots\ldots \psi_n(t)\ )$ corresponding to $u(t)$ and $x(t)$ such that :

1- for all t , $t_0<=t<=t_1$ the function $\wp\ (\ \psi(t),x(t),u\ )$ of the variable $u \in U$ attains its maximum at the point $u=u(t)$

$$\wp\ (\psi(t),\ x(t),\ u(t)\ ) = m\ (\psi(t),\ x(t)\ ) \tag{B.6}$$

2- at the terminal time t_1 the relation $m(\psi(t_1), x(t_1)) >= 0$ is satisfied. Furthermore, it turns out that if $\psi(t)$, $x(t)$ and $u(t)$ satisfy system

$$\frac{dx^i}{dt} = \frac{\partial \wp}{\partial \psi_i} \qquad i = 0,1, \ldots\ldots n \qquad \text{(B.7)}$$

$$\frac{d\psi_i}{dt} = -\frac{\partial \wp}{\partial x^i} \qquad i = 0,1, \ldots\ldots n \qquad \text{(B.8)}$$

and condition 1, the time function $m(\psi(t), x(t))$ is constant thus condition 2 may be verified at any time t, $t_0 <= t <= t_1$ and not just at t_1.

And for nonautonomous systems with the fixed times :

Theorem 6 : Let $u(t)$, $t_0 <= t <= t_1$, be an admissible control which transfers the phase point from the position x_0 to the position x_1, and let $x(t)$ be the corresponding trajectory so that $x(t_0)=x_0$ and $X(t_1)=x_1$

(the times t_0 and t_1 are fixed). In order that $u(t)$ yield a solution of the given optimal problem of :

$$\frac{dx^i}{dt} = f^i(x,u,t) = f^i(x,u,x^{n+1}) \qquad i = 0, 1, \ldots\ldots n \tag{B.9}$$

$$\frac{dx^{n+1}}{dt} = 1 \tag{B.10}$$

$$\begin{cases} x^{n+1}(t_0) = t_0 \\ x^{n+1} = t \end{cases} \tag{B.11}$$

$$\pounds = \int_{t_0}^{t_1} f^0(x, u, t)\, dt \tag{B.12}$$

with fixed time it is necessary that there exist a nonzero continuous vector function $\psi(t) = (\psi_0(t), \ldots\ldots\ldots \psi_n(t))$ corresponding to the function u(t) and x(t) such that :

1- for all t, $t_0 <= t <= t_1$, the function $\wp (\psi(t), x(t), t, u)$ of the variable $u \in U$ attains its maximum at the point u=u(t)

$$\wp (\psi(t), x(t), t, u(t)) = m (\psi(t), x(t), t) \tag{B.13}$$

2- the function $\psi_0(t)$ is nonpositive (which need only be verified at any one point of the interval $t_0 <= t <= t_1$, since from

$$\frac{d\psi_i}{dt} = -\frac{\partial \wp}{\partial x^i} \qquad (B.14)$$

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

$$\frac{d\psi_{n+1}}{dt} = -\frac{\partial \wp}{\partial t} \qquad (B.15)$$

ψ_0= constant).

Theorem 7 : In order that the admissible control u(t), $t_0 <= t <= t_1$, and the corresponding trajectory x(t) yield a solution of the optimal problem :

$$\frac{dx^i}{dt} = f^i(x,u,t) = f^i(x,u,x^{n+1}) \qquad i = 0, 1, \ldots\ldots n \qquad (B.16)$$

$$\frac{dx^{n+1}}{dt} = 1 \qquad (B.17)$$

$$\begin{cases} x^{n+1}(t_0) = t_0 \\ x^{n+1} = t \end{cases} \qquad (B.18)$$

$$\pounds = \int_{t_0}^{t_1} f^0(x, u, t)\, dt \qquad (B.19)$$

with a fixed left-hand end point x_0 and a free right-hand end point (the times t_0 and t_1 are given) it is necessary that there exist a

nonzero continuous vector function $\psi(t) = (\psi_0(t), \ldots\ldots, \psi_n(t))$ corresponding to the functions u(t) and x(t) such that :

1- for all t, $t_0 <= t <= t_1$ the function $\wp(\psi(t), x(t), t, u)$ of the variable $u \in U$ attains its maximum at the point u=u(t)

$$\wp(\psi(t), x(t), t, u(t)) = m(\psi(t), x(t), t) \qquad \text{(B.20)}$$

2- $$\psi(t_1) = (-1, 0, \ldots\ldots, 0)\,. \qquad \text{(B.21)}$$

APPENDIX C

COMPUTER PROGRAMS

```
5  REM   PROGRAM FOR 45 DEGREE PATH
6  DIM  X(2,1,22),DX(2,1,22),XF(2,1,22),JE(2,3,22),TRANPJE(3,2,22),P(2,2,22),PSE
UJE(3,2,22),Q(3,3,22),R(3,3,22),M(3,3,22),N(3,1,22),DTA(3,1,22),TA(3,1,22),W(2,2
,22),D(3,1,22),V(3,1,22),G(22),XO(2,1,22),XA(2,1,22),XB(2,1,22),LAM(2,1,22),PX(2
,2,22)
7  DIM  PP(2,2,22),LAMM(2,1,50),QQ(2,2,22),AA(2,2,22)
8  REM   DT : TIME INTERVAL; C1: SWITHING COEFFICIENT  TO SHIFT FROM CONTROL SCH
EME TO THE LEAST NORM SOLUTION. FOR THE CONTROL SCHEME C1=1, FOR THE LEAST NORM
SOLUTION C1=0; A1,A2 : Q1(1,1) AND Q1(2,2) RESPECTIVELY( ELEMENTS OF THE DIAGONA
L MATRIX)
9 REM Z: THE NUMBER OF POINTS ON THE PATH.
13   INPUT "DT,C1,A1,A2,Z=";DT,C1,A1,A2,Z
14   INPUT  "DIVISION=";DIV
15 COUNTER=1
20   PRINT TAB(5);"END-EFFECTOR ";TAB(24);"THETA ONE";TAB(35);"THETA TWO";TAB(47
);"THETA THREE";TAB(61);"G(X)"
21   PRINT  "X" ;TAB(13);"Y"
30  REM  LPRINT TAB(5);"END-EFFECTOR";TAB(24);"THETA ONE";TAB(35);"THETA TWO";TA
B(47);"THETA THREE";TAB(61);"G(X)"
31  REM  LPRINT  "X" ;TAB(13);"Y"
33 REM   FOR HORIZONTAL AND VERTICAL PATHS: H=DIV
40   H=SQR(DIV^2/2)
70   L1=6: L2=6:L3=6
80   TA(1,1,1)=35*3.14159/180:TA(2,1,1)=325*3.14159/180:TA(3,1,1)=68*3.14159/180
:A=6:B=5.5 :R1=1.5
100  FOR  K=1 TO  Z
103  XF(1,1,K)=L3*COS(TA(1,1,K)+TA(2,1,K)+TA(3,1,K))+L2*COS(TA(1,1,K)+TA(2,1,K))
+L1*COS(TA(1,1,K))
105  XF(2,1,K)=L3*SIN(TA(1,1,K)+TA(2,1,K)+TA(3,1,K))+L2*SIN(TA(1,1,K)+TA(2,1,K))
+L1*SIN(TA(1,1,K))
106  X(1,1,1)=XF(1,1,1) : X(2,1,1)=XF(2,1,1)
107 REM  FOR HORIZONTAL PATH: X(1,1,K+1)=X(1,1,1)-K*H,X(2,1,K+1)=X(2,1,1); FOR
VERTICAL PATH: X(1,1,K+1)=X(1,1,1),X(2,1,K+1)=X(2,1,1)+K*H
110  X(1,1,K+1)=X(1,1,1)-K*H
120  X(2,1,K+1)=X(2,1,1)+K*H
122  XO(1,1,K)=0 : XO(2,1,K)=0
124   XA(1,1,K)=L1*COS(TA(1,1,K))
125   XA(2,1,K)=L1*SIN(TA(1,1,K))
127   XB(1,1,K)=L1*COS(TA(1,1,K))+L2*COS(TA(1,1,K)+TA(2,1,K))
128   XB(2,1,K)=L1*SIN(TA(1,1,K))+L2*SIN(TA(1,1,K)+TA(2,1,K))
150  IF  K=1  THEN 160 ELSE 190
151 REM FOR HORIZONTAL PATH: DX(1,1,1)=-H, DX(2,1,1)=0; FOR VERTICAL PATH: DX(1,
1,1)=0, DX(2,1,1)=H
160  DX(1,1,1)=-H
170  DX(2,1,1)=+H
180  GOTO 210
190  DX(1,1,K)=X(1,1,K+1)-XF(1,1,K)
200  DX(2,1,K)=X(2,1,K+1)-XF(2,1,K)
210  JE(1,1,K)=-L1*SIN(TA(1,1,K))-L2*SIN(TA(1,1,K)+TA(2,1,K))-L3*SIN(TA(1,1,K)
 +TA(2,1,K)+TA(3,1,K))
220  JE(1,2,K)=-L2*SIN(TA(1,1,K)+TA(2,1,K))-L3*SIN(TA(1,1,K)+TA(2,1,K)+TA(3,1,K)
)
230  JE(1,3,K)=-L3*SIN(TA(1,1,K)+TA(2,1,K)+TA(3,1,K))
240  JE(2,1,K)=L1*COS(TA(1,1,K))+L2*COS(TA(1,1,K)+TA(2,1,K))+L3*COS(TA(1,1,K)+
 TA(2,1,K)+TA(3,1,K))
250  JE(2,2,K)=L2*COS(TA(1,1,K)+TA(2,1,K))+L3*COS(TA(1,1,K)+TA(2,1,K)+TA(3,1,K))
260  JE(2,3,K)=L3*COS(TA(1,1,K)+TA(2,1,K)+TA(3,1,K))
270   FOR I=1  TO  3
271       FOR J=1  TO  2
272       LET  TRANPJE(I,J,K)=JE(J,I,K)
```

```
275   FOR  I=1  TO  2
276        FOR  J=1  TO  2
277        LET W(I,J,K)=0
278            FOR  M=1  TO  3
279            LET  W(I,J,K)=W(I,J,K)+JE(I,M,K)*TRANPJE(M,J,K)
280            NEXT  M
281        NEXT J
282   NEXT I
283   LET DETW=W(1,1,K)*W(2,2,K)-W(1,2,K)*W(2,1,K)
284   FOR  I=1  TO  2
285        FOR  J=1  TO  2
286        LET  P(I,J,K)=((-1)^(I+J))*W(3-I,3-J,K)/DETW
287        NEXT J
288   NEXT  I
289   FOR  I=1  TO  3
290        FOR  J=1  TO  2
291        LET  PSEUJE(I,J,K)=0
292             FOR  F=1  TO  2
293             LET  PSEUJE(I,J,K)=PSEUJE(I,J,K)+TRANPJE(I,F,K)*P(F,J,K)
294             NEXT  F
295        NEXT J
296   NEXT I
297   FOR   I=1  TO  3
298         FOR  J=1  TO  3
299         LET  Q(I,J,K)=0
300              FOR  L=1  TO  2
301              LET  Q(I,J,K)=Q(I,J,K)+PSEUJE(I,L,K)*JE(L,J,K)
302              NEXT L
303         NEXT J
304   NEXT  I
305   FOR   I=1  TO  3
306         FOR  J=1  TO  3
307         IF  I=J  THEN  308  ELSE  310
308         LET  R(I,J,K)=1
309         GOTO  311
310         LET  R(I,J,K)=0
311         NEXT J
312   NEXT  I
313   FOR   I=1  TO  3
314         FOR  J=1  TO  3
316         LET  M(I,J,K)=R(I,J,K)-Q(I,J,K)
319         NEXT J
320   NEXT  I
331  B4=L1*COS(TA(1,1,K))-A*COS(TA(1,1,K)+TA(2,1,K))^2-B*COS(TA(1,1,K)+TA(2,1,K)
)*SIN(TA(1,1,K)+TA(2,1,K))-L1*SIN(TA(2,1,K))*SIN(TA(1,1,K)+TA(2,1,K))
332  B5=L1*SIN(TA(1,1,K))-A*SIN(TA(1,1,K)+TA(2,1,K))*COS(TA(1,1,K)+TA(2,1,K))-B*
SIN(TA(1,1,K)+TA(2,1,K))^2-(L1*SIN(TA(2,1,K))*SIN(TA(1,1,K)+TA(2,1,K))^2-L1*SIN(
TA(2,1,K)))/COS(TA(1,1,K)+TA(2,1,K))
333  LU=SQR((B4)^2+(B5)^2)
334  XU1=A*COS(TA(1,1,K)+TA(2,1,K))^2+B*COS(TA(1,1,K)+TA(2,1,K))*SIN(TA(1,1,K)+T
A(2,1,K))+L1*SIN(TA(2,1,K))*SIN(TA(1,1,K)+TA(2,1,K))
335  XU2=A*COS(TA(1,1,K)+TA(2,1,K))*SIN(TA(1,1,K)+TA(2,1,K))+B*SIN(TA(1,1,K)+TA(
2,1,K))^2+(L1*SIN(TA(2,1,K))*SIN(TA(1,1,K)+TA(2,1,K))^2-L1*SIN(TA(2,1,K)))/COS(T
A(1,1,K)+TA(2,1,K))
337   PX(1,1,1)=0 : PX(1,2,1)=0 : PX(2,1,1)=0 : PX(2,2,1)=0
338   PP(1,1,1)=1 : PP(2,2,1)=1
339   PP(1,2,1)=0 : PP(2,1,1)=0
340   AA(1,1,K)=L1^2*SIN(TA(1,1,K))^2 +2*LU^2*SIN(TA(1,1,K)+TA(2,1,K))^2+2*L1*LU
*SIN(TA(1,1,K))*SIN(TA(1,1,K)+TA(2,1,K))
342   AA(1,2,K)=-L1^2*SIN(TA(1,1,K))*COS(TA(1,1,K))-L1*LU*SIN(TA(1,1,K))*COS(TA(
1,1,K)+TA(2,1,K))-2*LU^2*SIN(TA(1,1,K)+TA(2,1,K))*COS(TA(1,1,K)+TA(2,1,K))-L1*LU
*COS(TA(1,1,K))*SIN(TA(1,1,K)+TA(2,1,K))
344   AA(2,1,K)=AA(1,2,K)
345   AA(2,2,K)=L1^2*COS(TA(1,1,K))^2+2*LU^2*COS(TA(1,1,K)+TA(2,1,K))^2+2*L1*LU*
```

```
348    PX(1,1,K+1)=PX(1,1,K)+(AA(1,1,K)*PP(1,1,K)+AA(1,2,K)*PP(2,1,K))*DT
350    PX(1,2,K+1)=PX(1,2,K)+(AA(1,1,K)*PP(1,2,K)+AA(1,2,K)*PP(2,2,K))*DT
352    PX(2,1,K+1)=PX(2,1,K)+(AA(2,1,K)*PP(1,1,K)+AA(2,2,K)*PP(2,1,K))*DT
354    PX(2,2,K+1)=PX(2,2,K)+(AA(2,1,K)*PP(1,2,K)+AA(2,2,K)*PP(2,2,K))*DT
356    PP(1,1,K+1)=PP(1,1,K)-PX(1,1,K)*DT
358    PP(1,2,K+1)=PP(1,2,K)-PX(1,2,K)*DT
360    PP(2,1,K+1)=PP(2,1,K)-PX(2,1,K)*DT
361    PP(2,2,K+1)=PP(2,2,K)-PX(2,2,K)*DT
362   LAMM(1,1,1)=0            :LAMM(2,1,1)=0
363    LAM(1,1,1)=LAMM(1,1,COUNTER)
364    LAM(2,1,1)=LAMM(2,1,COUNTER)
385   LET  LAM(1,1,K+1)=LAM(1,1,K)-A1*(XU1-A)*DT
390   LET  LAM(2,1,K+1)=LAM(2,1,K)-A2*(XU2-B)*DT
399   N(1,1,K)=(-L1*SIN(TA(1,1,K))-LU*SIN(TA(1,1,K)+TA(2,1,K)))*LAM(1,1,K)+(L1*CO
S(TA(1,1,K))+LU*COS(TA(1,1,K)+TA(2,1,K)))*LAM(2,1,K)
400   N(2,1,K)=(-LU*SIN(TA(1,1,K)+TA(2,1,K)))*LAM(1,1,K)+ LU*COS(TA(1,1,K)+TA(2,1
,K))*LAM(2,1,K)
405   N(3,1,K)=0
406    FOR  I=1  TO  3
407         LET O(I,1,K)=0
408         FOR  S=1 TO  3
409         LET  O(I,1,K)=O(I,1,K)+M(I,S,K)*N(S,1,K)*DT
410         NEXT S
411    NEXT  I
412    FOR  I=1  TO  3
413         LET  V(I,1,K)=0
414         FOR  Y=1  TO   2
415         LET  V(I,1,K)=V(I,1,K)+PSEUJE(I,Y,K)*DX(Y,1,K)
416         NEXT  Y
417    NEXT  I
418  LET  B3=(A*SIN(TA(1,1,K)+TA(2,1,K))*COS(TA(1,1,K)+TA(2,1,K))+B*SIN(TA(1,1,K
)+TA(2,1,K))^2+(L1*SIN(TA(2,1,K))*SIN(TA(1,1,K)+TA(2,1,K))^2-L1*SIN(TA(2,1,K)))/
(COS(TA(1,1,K)+TA(2,1,K)))-B)^2
419 G(K)=R1^2-(A*COS(TA(1,1,K)+TA(2,1,K))^2+B*COS(TA(1,1,K)+TA(2,1,K))*SIN(TA(1,
1,K)+TA(2,1,K))+L1*SIN(TA(2,1,K))*SIN(TA(1,1,K)+TA(2,1,K))-A)^2-B3
423        LET  C=C1
426    FOR  I=1  TO  3
427         LET  DTA(I,1,K)=V(I,1,K)+C*O(I,1,K)
428    NEXT  I
431    FOR  I=1  TO  3
432         LET  TA(I,1,K+1)=TA(I,1,K)+DTA(I,1,K)
433    NEXT  I
440  PRINT XF(1,1,K);TAB(12);XF(2,1,K);TAB(24);TA(1,1,K)*180/3.14159;TAB(35);TA(
2,1,K)*180/3.14159;TAB(47);TA(3,1,K)*180/3.14159;TAB(61);G(K),LAM(1,1,K ),LAM(2,
1,K )
450  REM   LPRINT XF(1,1,K);TAB(12);XF(2,1,K);TAB(24);TA(1,1,K)*180/3.14159;TAB(
35);TA(2,1,K)*180/3.14159;TAB(47);TA(3,1,K)*180/3.14159;TAB(61);G(K)
452  REM   LPRINT  XA(1,1,K),XA(2,1,K),XB(1,1,K),XB(2,1,K),XF(1,1,K),XF(2,1,K)
460 NEXT K
461      IF  (LAM(1,1,Z )^2 + LAM(2,1,Z )^2)<1E-17 THEN 500 ELSE 462
462  COUNTER=COUNTER+1
463   DETPP=PP(1,1,Z )*PP(2,2,Z )-PP(2,1,Z )*PP(1,2,Z )
466   FOR  I=1 TO 2
468        FOR J=1 TO 2
470      QQ(I,J,Z )=((-1)^(I+J))*PP(3-I,3-J,Z )/DETPP
472        NEXT  J
474   NEXT I
485  LAMM(1,1,COUNTER)=LAMM(1,1,COUNTER-1)-QQ(1,1,Z )*LAM(1,1,Z )-QQ(1,2,Z )*LAM
(2,1,Z )
490  LAMM(2,1,COUNTER)=LAMM(2,1,COUNTER-1)-QQ(2,1,Z )*LAM(1,1,Z )-QQ(2,2,Z )*LAM
(2,1,Z )
495  GOTO  20
500 END
```

```
! Graphing program
! N : number of points on the path.
Input N
DIM arm(4*N,2)
open #1:printer
MAT READ arm
data
LIBRARY "True basic:Libraries:GraphLib"
INPUT prompt "Draw AXES (y/n) ? ":drawaxes$
IF drawaxes$ = "y" or drawaxes$ = "Y" then
   INPUT prompt "Distance between X-Axes tick marks ? ":distanceX
   INPUT prompt "Distance between Y-Axes tick marks ? ":distanceY
   INPUT prompt "Window minimum X value ? ":lowx
   INPUT prompt "Window maximum X value ? ":highx
   INPUT prompt "Window minimum Y value ? ":lowy
   INPUT prompt "Window maximum Y value ? ":highy
END IF
INPUT prompt "Clear graphics window (y/n) ? ": Clear$
IF clear$ = "Y" or clear$ = "y" then
   CLEAR
END IF
SET window lowx,highx,lowy,highy
IF drawaxes$ = "y" or drawaxes$ = "Y" then
   CALL ticks(distanceX,distanceY)
END IF
LET i = 1
FOR k = 1 to N
    FOR counter = 1 to 4
        PLOT LINES : arm(i,1),arm(i,2);
        LET i = i + 1
    NEXT counter
    PLOT
NEXT k
BOX CIRCLE 4.5,7.5,4.0,7.0
print#1:printer
END
```

VITA

Siamak Esmailzadeh Khadem was born in Mashad, Iran on November 28, 1957. He attended elementary schools in that city and was graduated from Azar High School in June 1976. The following September he entered Sharif University of Technology in Tehran, Iran. In December 1984 he received a Bachelor of Science degree in Mechanical Engineering. In the fall of 1986 he began study toward a Master's degree. He received this degree in Mechanical Engineering from the University of Tennessee, Knoxville in June 1987.