Robotica (1999) volume 17, pp. 143–157. Printed in the United Kingdom © 1999 Cambridge University Press

Motion control of non-fixed base robotic manipulators

* F.M. Carter, ** D.B. Cherchas

(Received in Final Form: July 6, 1998)

SUMMARY

Robotic manipulators mounted on spacecraft experience a number of kinematic, dynamic, and control problems because the motion of the spacecraft is affected by the robot motion. In this paper, the general three dimensional equations of motion are derived for an *n* link manipulator mounted on a non-fixed base object. Instead of performing a single inverse kinematic calculation at the beginning of a movement to determine the required joint setpoints, multiple inverse kinematic updates are done throughout a movement. The updating sequence is determined by an optimal inverse kinematic updating algorithm. This motion control algorithm is based on experimental simulation results performed in Matlab and a set of performance indices that are used as guidelines. Simple PD joint controllers and a special joint trajectory generator are used for servoing the manipulator joints for a planar robot application. The derived motion control techniques incorporate the base motion without base motion control.

KEYWORDS: Robot, control; Non-fixed base; Inverse kinematics.

1. INTRODUCTION

Most robotic manipulators can be considered to have a base fixed in an inertial frame. There are, however, manipulators for which this is not the case. Two examples would be manipulators mounted on free flying-satellites or submarines. In such cases, the commanded arm motions required to produce a given robot end-effector position (for grasping), calculated assuming a fixed base, will result in position error of the end effector because of the dynamic coupling between the robot and base.

The dynamic coupling that occurs in a moving base robot manipulator system has not been extensively studied until comparatively recently. Early studies[1–3] in this field neglected the dynamic coupling between the robot and the base. One of the first approaches to consider coupling was formulated by Longman, Lindberg, and Zedd.[4] In this work it was assumed the orientation of the spacecraft could be kept constant by using gyroscopes to negate the torques being transmitted at the base. The holonomic conservation of momentum principle was then used to eliminate the remaining three translational degrees of freedom. Modified inverse kinematics and dynamic equations in closed form for a spacecraft with a spherical polar-coordinate manipulator were derived. A new method, termed the Virtual Manipulator Approach, was introduced by Vafa and Dubowsky.[5] In this approach, constant spacecraft orientation is also assumed. The Virtual Manipulator can be considered to be in a fixed inertial frame that uses the center of mass of the system as a Virtual Ground. The position of the Virtual Ground is calculated at the start of each movement and is invariant as long as the system mass properties do not change. The inverse kinematics solution is then found using the holonomic conservation of momentum principle. The problem with these methods is that the center of mass of the system must be accurately calculated to achieve good results.

Some studies[6,7] have demonstrated that when the mass of the base or spacecraft is relatively large compared to that of the manipulator, decoupling between the end-effector motion and the spacecraft motion can be assumed. Using feedforward compensation from the satellite forces and torques, a special feedback control scheme can be used to give approximate results. Various groups[8–10] have researched the scenario of a small, high bandwidth manipulator mounted on a larger, slow bandwidth manipulator (macro/micro manipulator systems). Another spacecraft/manipulator study[11] proposed a control scheme that decoupled end-effector motion and total system momentum. Using the augmented task-space approach, recursive formulations of the kinematic and dynamic equations are performed, and a kinematic redundancy resolution scheme is used to achieve a solution. In the same paper, the manipulator end-effector tracked a desired reference trajectory and a slow gross positioning was used for the satellite. Unfortunately, the satellite, repositioning uses fuel. One paper[12] discusses attitude control of space-craft/manipulator systems using internal joint motions for control. First, the controllability of the system is discussed and proven. Then a basis algorithm for near optimal solutions is formulated for non-holonomic motion planning systems. This approach yields an approximate optimal solution that minimizes the control inputs (energy required).

Unlike singularities for fixed base manipulators that are solely a function of kinematics, space manipulators without base motion control encounter dynamic singularities.[13] These singularities occur when the end-effector cannot move in some inertial direction, and are a function of the dynamic properties of the manipulator and base. The manipulator workspace therefore is a function of the kinematic and dynamic parameters.

In this research, a motion control technique based on multiple inverse kinematic updating was developed. By

* Robotics Engineer, Control Systems Group, International Submarine Engineering Ltd., Port Coquitlam, B.C. (Canada) V3C 2M8

** Professor, Dept. of Mechanical Engineering, Camrol Laboratory, University of British Columbia, Vancouver, B.C. (Canada), V6T 1Z4.

https://doi.org/10.1017/S0263574799001186 Published online by Cambridge University Press

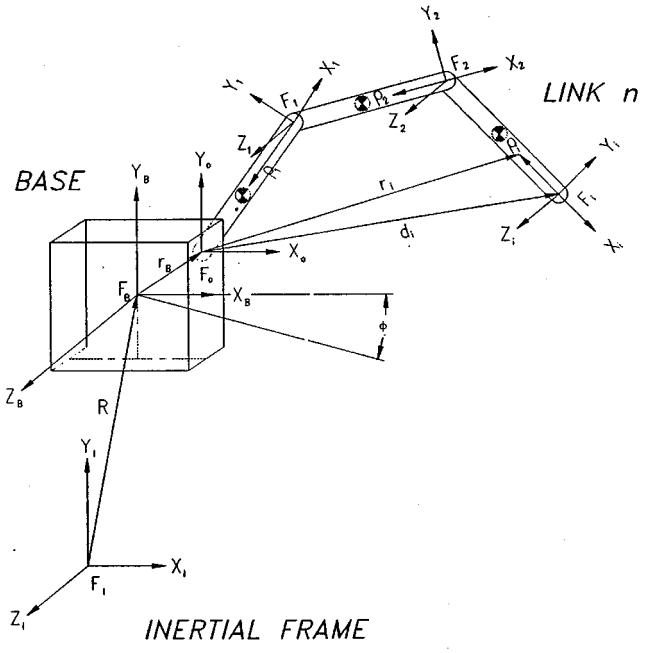

Fig. 1. Robot configuration.

updating the manipulator inverse kinematic solutions throughout the motion, the requirement for attitude servoing can be eliminated. The dynamic parameters of the system need not be known but if they are known approximately, the algorithm performance can be further improved. The only kinematic quantities required are the orientation and position of the base relative to the desired end effector position at certain points during a movement. Such quantities are measureable, for example, by cameras mounted on the space vehicle or the manipulator.

2. MANIPULATOR AND BASE DYNAMICS MODEL

The configuration of the manipulator and non-fixed base is as shown in Figure 1.

In 3 dimensions, the base has 6 degrees of freedom and the robot n degrees of freedom, one for each link. This gives the system $6+n$ degrees of freedom or one degree of freedom for each generalized coordinate. The frames, matrices, and vectors used are identified in the list below.

Reference Frames:

F_I Inertial frame of manipulator – base system.

F_B Base coordinate frame located at the centre of mass of the base.

F_O Zeroth coordinate frame origin on degree of freedom axis at 1st manipulator link.

F_1 Coordinate frame on outboard end of the first link of the manipulator.

F_i Coordinate frame on outboard end of the ith link of the manipulator.

Matrices and Vectors:

$\mathrm{H}_{(i,j)}$ Homogeneous transformation from the jth frame to the ith frame.

$\mathrm{H}_{(I,B)}$ Homogeneous transformation from the Bth frame to the I th frame.

$\mathrm{H}_{(B,0)}$ Homogeneous transformation from the Zeroth frame to the Bth frame.

$\mathrm{H}_{(0,1)}$ Homogeneous transformation from the first frame to the zeroth frame.

$\mathrm{H}_{(0,i)}$ Homogeneous transformation from the ith frame to the zeroth frame.

R Position of frame F_B relative to and projected onto frame F_I.

Φ Rotation matrix of the base in frame F_I.

r_B Position of frame F_O relative to and projected onto frame F_B.

d_i Position of frame F_i relative to and projected onto frame F_O.

ρ_i Position of point on link i relative to and projected onto frame F_i.

r_i Position of point on link i relative to and projected onto frame F_0.

The generalized coordinate for each link will be denoted as q_i. If a link is revolute or prismatic, q_i is represented by either θ_i or d_i respectively. The homogeneous matrix for a link will reflect whether the link is either translational or rotational. The translational coordinates of the base are defined as $\mathbf{R}=(R_x, R_y, R_z, 1)^T$. This corresponds to translations along the X, Y, and Z axes in the $\mathbf{F}_I$ frame. The rotational coordinates are defined as $\Phi=(\Phi_1, \Phi_2, \Phi_3)$. The generalized coordinate for each d.o.f. of the base will be denoted as z_s. The equations of motion of the system are derived using Lagrange's equations. The kinetic energy for the base and the links will first be derived.

Referring to Figure 1, the position of a point on link i relative to and projected onto frame F_I is

$$\mathbf{r}_i=\mathbf{H}_{(I,B)}\mathbf{H}_{(B,0)}\mathbf{H}_{(0,1)}\mathbf{H}_{(1,2)}\ldots\mathbf{H}_{(i-1,i)}\,\rho_i \tag{1}$$

$\mathbf{H}_{(I,B)}$ is a function of the six base generalized coordinates z_s, and $\mathbf{H}_{(0,i)}$ is a function of the ith link generalised coordinate (note ρ_i is a vector). Since each generalised coordinate is also a function of time, we can write the velocity as

$$\dot{\mathrm{r}}_i=\mathrm{v}_i=\sum_{s=1}^{6}\frac{\partial H_{(I,B)}}{\partial \mathrm{z}_s}\mathrm{H}_{(B,0)}\mathrm{H}_{(0,i)}\rho_i\dot{z}_s + \sum_{j=1}^{i}\mathrm{H}_{(I,B)}\mathrm{H}_{(B,0)}\frac{\partial \mathrm{H}_{(0,i)}}{\partial q_j}\rho_i\dot{q}_j \tag{2}$$

Knowing that the velocity squared of a vector is

$$\left(\frac{d\mathrm{r}_i}{dt}\right)^2=trace(\dot{\mathbf{r}}\dot{\mathbf{r}}^T) \tag{3}$$

https://doi.org/10.1017/S0263574799001186 Published online by Cambridge University Press

we can write the kinetic energy of a particle of mass dm on link i at ρ_i as

$$dK_i = \frac{1}{2} trace(\dot{\mathbf{r}}_i \dot{\mathbf{r}}_i^T) dm \tag{4}$$

The total kinetic energy of the links is given by integrating over the mass for each link and then summing as follows:

$$K_L = \sum_{i=1}^{n} K_i = \sum_{i=1}^{n} \int dK_i \tag{5}$$

The integral of the term $\rho_i \rho_i^T$ over link i is termed the pseudo inertia matrix

$$\mathbf{J}_i = \int_{linki} \rho_i \rho_i^T dm \tag{6}$$

Similarily, the kinetic energy of the base is summarized for each of the six translational and rotational coordinates, $\mathbf{z}_w$, as

$$K_B = \sum_{w=1}^{6} K_w = \sum_{w=1}^{6} \int dK_w \tag{7}$$

At this point we assume no gravity, and therefore the Lagrangian is simply the system total kinetic energy, and can be written as

$$L = K_S = K_L + K_B \tag{8}$$

To obtain the dynamic equations of motion, the Lagrangian is substituted into the Lagrange's equations as below

$$\frac{d}{dt}\left(\frac{\partial \mathbf{L}}{\partial \dot{\mathbf{p}}_i}\right) - \frac{\partial \mathbf{L}}{\partial \mathbf{p}_i} = \tau_i \tag{9}$$

where p_i is the generalized coordinate and τ_i is the generalized force and torque at joint i.

For the link dynamic equations, we will let the manipulator link generalized coordinates be represented as q_p and the generalized force or torque as τ_p. After many cancellations and index substitutions we obtain the dynamic equations for the manipulator links as below

$$\sum_{i=p}^{n} \sum_{s=1}^{6} tr\left[\frac{\partial \mathbf{H}_{(I,B)}}{\partial \mathbf{z}_s} \mathbf{H}_{(B,0)} \mathbf{H}_{(0,i)} \mathbf{J}_i \frac{\partial \mathbf{H}_{(0,i)}^T}{\partial \mathbf{q}_p} \mathbf{H}_{(B,0)}^T \mathbf{H}_{(I,B)}^T\right] \ddot{z}_s$$

$$+ \sum_{i=p}^{n} \sum_{k=1}^{i} tr\left[\mathbf{H}_{(I,B)} \mathbf{H}_{(B,0)} \frac{\partial \mathbf{H}_{(0,i)}}{\partial \mathbf{q}_p} \mathbf{J}_i \frac{\partial \mathbf{H}_{(0,i)}^T}{\partial \mathbf{q}_k} \mathbf{H}_{(B,0)}^T \mathbf{H}_{(I,B)}^T\right] \ddot{q}_k$$

$$+ \sum_{i=p}^{n} \sum_{s=1}^{6} \sum_{t=1}^{6} tr\left[\frac{\partial \mathbf{H}_{(I,B)}}{\partial \mathbf{z}_t \partial \mathbf{z}_s} \mathbf{H}_{(B,0)} \mathbf{H}_{(0,i)} \mathbf{J}_i \frac{\partial \mathbf{H}_{(0,i)}^T}{\partial \mathbf{q}_p} \mathbf{H}_{(B,0)}^T \mathbf{H}_{(I,B)}^T\right] \dot{z}_s \dot{z}_t$$

$$+ \sum_{i=p}^{n} \sum_{k=1}^{i} \sum_{m=1}^{i} tr\left[\mathbf{H}_{(I,B)} \mathbf{H}_{(B,0)} \frac{\partial \mathbf{H}_{(0,i)}}{\partial \mathbf{q}_p} \mathbf{J}_i \frac{\partial^2 \mathbf{H}_{(0,i)}^T}{\partial \mathbf{q}_m \partial \mathbf{q}_k} \mathbf{H}_{(B,0)}^T \mathbf{H}_{(I,B)}^T\right] \dot{q}_k \dot{q}_m$$

$$+ \sum_{i=p}^{n} \sum_{s=1}^{6} \sum_{i=p}^{n} (2)\ tr\left[\frac{\partial \mathbf{H}_{(I,B)}}{\partial \mathbf{z}_s} \mathbf{H}_{(B,0)} \frac{\partial \mathbf{H}_{(0,i)}}{\partial \mathbf{q}_m} \mathbf{J}_i \frac{\partial \mathbf{H}_{(0,i)}^T}{\partial \mathbf{q}_p} \mathbf{H}_{(B,0)}^T \mathbf{H}_{(I,B)}^T\right] \dot{z}_s \dot{q}_m = \tau_p \tag{10}$$

- $p = 1$ gives the equation for link 1 in q_1
- $p = x$ gives the nth equation for link x ibn q_x
- z_i are the generalized coordinates for the base.

The derivation of the dynamic equation for the base follows similar algebra as for the links, and after simplifications becomes

$$\sum_{i=1}^{n} \sum_{t=1}^{6} tr\left[\frac{\partial \mathbf{H}_{(I,B)}}{\partial \mathbf{z}_w} \mathbf{H}_{(B,0)} \mathbf{H}_{(0,i)} \mathbf{J}_i \mathbf{H}_{(0,i)}^T \mathbf{H}_{(B,0)}^T \frac{\partial \mathbf{H}_{(I,B)}^T}{\partial \mathbf{z}_t} + \frac{\partial \mathbf{H}_{(I,B)}}{\partial \mathbf{z}_w} \mathbf{J}_B \frac{\partial \mathbf{H}_{(I,B)}^T}{\partial \mathbf{z}_t}\right] \ddot{z}_t$$

$$+ \sum_{i=1}^{n} \sum_{j=1}^{i} tr\left[\frac{\partial \mathbf{H}_{(I,B)}}{\partial \mathbf{z}_w} \mathbf{H}_{(B,0)} \mathbf{H}_{(0,i)} \mathbf{J}_i \frac{\partial \mathbf{H}_{(0,i)}^T}{\partial \mathbf{q}_j} \mathbf{H}_{(B,0)}^T \mathbf{H}_{(I,B)}^T\right] \ddot{q}_j$$

$$+ \sum_{i=1}^{n} \sum_{t=1}^{6} \sum_{s=1}^{6} tr\left[\frac{\partial \mathbf{H}_{(I,B)}}{\partial \mathbf{z}_w} \mathbf{H}_{(B,0)} \mathbf{H}_{(0,i)} \mathbf{J}_i \mathbf{H}_{(0,i)}^T \mathbf{H}_{(B,0)}^T \frac{\partial^2 \mathbf{H}_{(I,B)}^T}{\partial z_s \partial z_t} + \frac{\partial \mathbf{H}_{(I,B)}}{\partial \mathbf{z}_w} \mathbf{J}_B \frac{\partial^2 \mathbf{H}_{(I,B)}^T}{\partial \mathbf{z}_s \partial \mathbf{z}_t}\right] \dot{z}_s \dot{z}_t$$

$$+ \sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{m=1}^{i} tr\left[\frac{\partial \mathbf{H}_{(I,B)}}{\partial \mathbf{z}_w} \mathbf{H}_{(B,0)} \mathbf{H}_{(0,i)} \mathbf{J}_i \frac{\partial^2 \mathbf{H}_{(0,i)}^T}{\partial \mathbf{q}_m \partial \mathbf{q}_j} \mathbf{H}_{(B,0)}^T \mathbf{H}_{(I,B)}^T\right] \dot{q}_j \dot{q}_m$$

https://doi.org/10.1017/S0263574799001186 Published online by Cambridge University Press

$$+ \sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{s=1}^{6} \quad (2)$$

$$tr\left[\frac{\partial \mathbf{H}_{(I,B)}}{\partial \mathbf{z}_w} \mathbf{H}_{(B,0)} \mathbf{H}_{(0,i)} \mathbf{J}_i \frac{\partial \mathbf{H}^T_{(0,i)}}{\partial \mathbf{q}_j} \mathbf{H}^T_{(B,0)} \mathbf{H}^T_{(I,B)} z_s \right] \dot{q}_j \dot{z}_s = 0 \quad (11)$$

- $w = 1$ gives the 1st equation for z_w
- $w = 6$ gives the 6th equation for z_6

The above manipulator dynamic equations have been developed in three dimensions for an n link manipulator on a 6 degree of freedom base. These dynamic equations can also be written in matrix form as

$$\mathbf{M}(q)\ddot{q} + \mathbf{V}(q,\dot{q}) = \tau \quad (12)$$

where **M** is the inertia matrix, **V** is the Coriolis-centripetal matrix, τ is the generalized force (torque) vector, and q is the generalized coordinate. If a disturbance term, $\mathbf{F}(\dot{q})$, is added to the above equation the manipulator dynamic equation becomes

$$\mathbf{M}(q)\ddot{q} + \mathbf{V}(q,\dot{q}) + \mathbf{F}(\dot{q}) = \tau \quad (13)$$

or more compactly

$$\mathbf{M}(q)\ddot{q} + \mathbf{N}(q,\dot{q}) = \tau \quad (14)$$

where $\mathbf{N(q,\dot{q})}$ represents all the non-linear terms.

The non-fixed base control algorithms in this work were developed initially for a 2 dimensional case of base and manipulator motion. In 2 dimensions, the base is capable of moving in 2 translational directions as well as one rotational direction. Therefore the base possesses 3 degrees of freedom and has 3 generalized coordinates. A 2 link revolute manipulator is used and has 2 degrees of freedom, bringing the total number of degrees of freedom to 5 for the planar case. The symbolic algebra program Maple was used for equation expansion and rearrangement into state space form. The state vector q for the system, referring to Figure 1, is defined below

$$q = [R_x \; R_y \; \Phi_1 \; \Theta_1 \; \Theta_2]^T \quad (15)$$

The state space form of the system dynamics equations for the simulations is achieved by rearranging equation (14) so that it is in first order form. The Maple command *C[eqn]* converts the Maple output into C code which is identical to Matlab code except for the line breaking. The result is that the dynamic equations of motion are in state space form, ready to be used in the Matlab simulations. The complete statement of the equations of motion is given in Carter.[14]

3. FIXED BASE INVERSE KINEMATICS

When a robot is mounted on a fixed base the joint parameters are calculated and passed to the controller which servos the joints to produce the desired end-effector position. The desired parameters or angles are calculated once at the beginning of the movement and they do not change throughout a movement. This is in contrast to the moving base case where the desired endpoint parameters are changing as the base moves (relative to the robot frame). For a 2 link manipulator the Geometric Approach is used to calculate the endpoint joint parameters. In this simple case the solution reduces to simple trigonometric equations. Let (x, y) be the desired end-effector position in the X-Y plane and let the 2 revolute links have lengths l_1 and l_2 as seen in Figure 2. The following equations then yield the necessary joint angles:

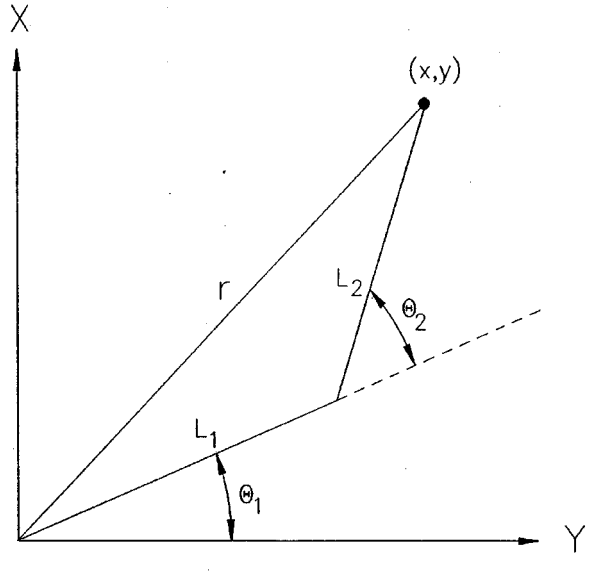

Fig. 2. Inverse kinematics for 2-link robot.

$$r^2 = x^2 + y^2 \quad (16)$$

$$C = \cos(\theta_2) = r^2 - l_1^2 - l_2^2 \quad (17)$$

$$D = \pm\sqrt{1 - C^2} \quad (18)$$

$$\theta_2 = \arctan 2(D, C) \quad (19)$$

$$\theta_1 = \arctan 2(x,y) - \arctan 2(l_2 \sin(\theta_2), l_1 + l_2 \sin(\theta_2)) \quad (20)$$

It is clear that there are two solutions as long as

$$l_1^2 + l_2^2 < x^2 + y^2 \quad (21)$$

If this is not true then the desired point is not in the robot workspace, and a singularity occurs. Note that the ± in Equation (18) above reveals that the solution is not unique. For this simple 2-D case there are two solutions, each of which corresponds to an elbow up or elbow down configuration. In the elbow up configuration both robot links lie above the vector **r**. In the elbow down configuration the links lie below the vector **r**. This shows that the inverse kinematics problem generally has a non-unique solution, the number of possible solutions increasing as the robots degrees of freedom increases. This property can be beneficial for collision avoidance where any number of solutions can provide correct end-effector positioning. The drawback is determining which solution to use if many solutions are possible.

4. INVERSE KINEMATICS UPDATING ALGORITHM FOR MOVING BASE

For the moving base problem, if the required joint angles are calculated at the start of the movement, dynamic coupling

https://doi.org/10.1017/S0263574799001186 Published online by Cambridge University Press

will produce motion of the base resulting in an end-effector position error. In the algorithm proposed herein, the required joint angles are updated at specified times throughout the movement taking into account the translation and rotation of the base that has occurred since the last update or the start of the movement. These updates are done on-line, assuming the desired end-effector position in the manipulator frame can be measured relatively quickly. Fast measurement feedback would mean that the base does not translate or rotate much during this delay. It is then assumed that the base is fixed at this position for the inverse kinematics calculation which gives new endpoint angles. A new joint trajectory is then calculated (one for each link) and the robot joints are servoed to the new desired final setpoints. This new trajectory has matching boundary conditions to the previous one. If the desired end-effector position is no longer in the local robot workspace (kinematic singularity), the movement is terminated.

At the end of the movement, if the end-effector is not within the allowable error of the desired position, further end-point servoing is done to complete the movement. When the desired position is reached, the movement is complete.

Computationally, the updating algorithm is fairly simple and could probably be done at several kHz on a fast digital signal processor. The computational time depends upon the speed at which a manipulators inverse kinematics and trajectory generation can be completed in. For a planar 2 link robot on a 3 degree of freedom base, the CPU time between the start of an update (measurement of end-effector position in manipulator frame), and the time when the joints begin servoing to the new angular setpoints, was measured. The time was 0.0078 seconds for all required calculations in a Matlab simulation (inverse kinematics, updating sequence, trajectory generation). The PC used was a Pentium 100 with 32 MB or RAM. In the computer simulations the updates take only one control cycle or one simulation step to complete. In this sense the update time is instantaneous in the simulation. This is because the inverse kinematics and update algorithm calculations are done before the next integration step starts.

5. JOINT TRAJECTORY UPDATING

To move a robotic end-effector from point A to point B in Cartesian space, each robot joint degree of freedom must follow a prescribed trajectory in joint space. In this work we are not concerned with obstacles and hence can use smooth interpolating polynomials for joint space trajectory generation which satisfy the given boundary conditions. A number of polynomial trajectories were evaluated, and a simple type, a quintic polynomial trajectory was chosen. A polynomial of this order is the minimum required to satisfy the endpoint boundary conditions that match continuously the joint angle, angular velocity, and angular accelerations. A discussion of polynomial trajectories can be found in An, Atkeson, and Hollerbach.[15] The quintic trajectory is of the form

$$c_i^d = a_0 + a_1 t + a_2 t + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{22}$$

The desired velocity trajectory is of the form

$$\dot{c}_i^d = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 \tag{23}$$

and the desired acceleration trajectory is of the form

$$\ddot{c}_i^d = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3 \tag{24}$$

These equations satisfy the boundary conditions

$$c_i^d(t_0) = c_0; \quad c_i^d(t_f) = c_f \tag{25}$$

$$\dot{c}_i^d(t_0) = \dot{c}_0; \quad \dot{c}_i^d(t_f) = \dot{c}_f \tag{26}$$

$$\ddot{c}_i^d(t_0) = \ddot{c}_0; \quad \ddot{c}_i^d(t_f) = \ddot{c}_f \tag{27}$$

where c_0 is the initial trajectory angle and c_f is the final joint angle. For example, if at the start of a movement the original joint angle required for joint i is calculated to be 120 degrees, the robot link begins following a trajectory that will complete this. If at 80 percent of the way through the movement the trajectory is updated and a new final angle of 175 degrees is required, the controller then servos the link along the new trajectory to achieve this. The new endpoint joint angles require that the trajectory polynomials be pieced together with matching boundary conditions. An example of these angle and angular velocity trajectories (θ_1, $\dot{\theta}_1$) for joint 1 can be seen in Figure 3 and Figure 4.

Joint trajectory velocity time scaling is used to satisfy the joint velocity constraints. Each robotic joint has an upper velocity and torque limit that the motor can produce, regardless of whether there is a gear reduction. If the desired joint velocity trajectory exceeds the maximum joint velocity the movement may not result in the desired final setpoint. In the simulations the user may specify a maximum joint velocity that must not be exceeded so that the end-effector velocity or acceleration will not be exceeded. There may also be a minimum specified velocity so that the movement is completed in a specified time. Actual space manipulators typically can move extremely large payloads but at very slow velocities. The CANADARM, for example, can move a payload many times heavier than itself but with maximum

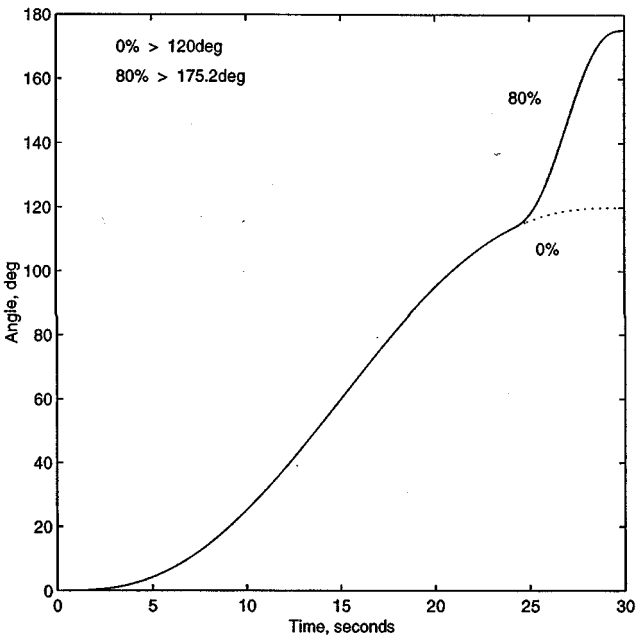

Fig. 3. Single joint trajectory update – θ_1 vs. time.

https://doi.org/10.1017/S0263574799001186 Published online by Cambridge University Press

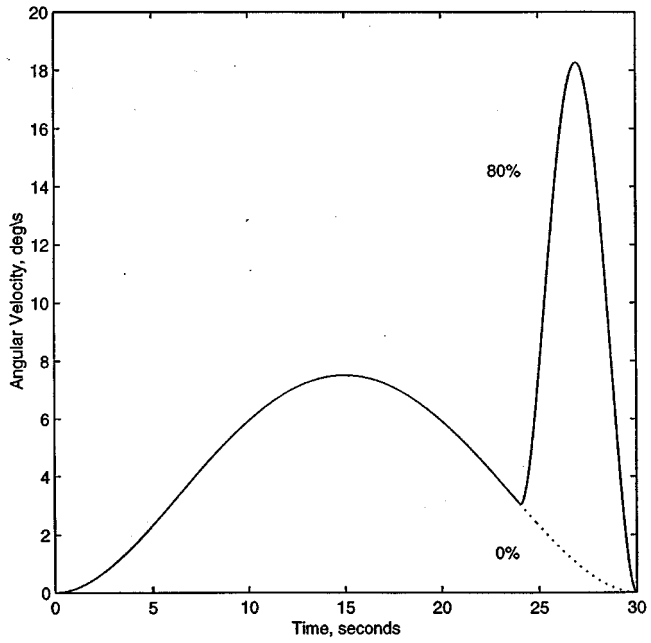

Fig. 4. Single joint trajectory update – θ_1 vs. time.

joint velocities of less than 5 deg/sec. To make the simulations as realistic as possible time scaling was implemented.

6. JOINT SERVO CONTROL

For robot joint control it is necessary to determine the control algorithms to servo all the joints of the manipulator to allow accurate tracking of a time-based trajectory. The simplest type of robot joint control is independent or classical joint control. In this type of joint control the n joint manipulator to be controlled will be controlled by n decoupled individual joint controllers whose control input is based on the locally measured joint variables (errors). Independent error driven joint control has long been popular since there is no need to solve complicated nonlinear robot inverse dynamics on-line. It also allows a decoupled analysis of the closed loop system using single-input single-output classical techniques. It has been proven that this type of control is very suitable for following a desired trajectory if the manipulator motion is relatively low speed and in fact is representative of how many industrial robots are controlled today.[16] It has been proven[17] that if PD control is applied to each joint and $e(0)=0$ and $\dot{e}(0)=0$ that the position and velocity errors are bounded within a circle whose radius decreases approximately (for large K_v) as $1/\sqrt{K_v}$. However, K_v cannot be increased without reaching the actuator torque limits. If we define the tracking error of the ith joint variable to be

$$e_i(t)=q_{d_i}(t)-q_i(t) \tag{28}$$

where q_d is the desired joint trajectory and q is the actual trajectory of the ith joint, we can define the velocity error to be

$$\dot{e}_i(t)=\dot{q}_{d_i}(t)-\dot{q}_i(t) \tag{29}$$

For PD control we would then select our control signal $u(t)$ as proportional-plus-derivative feedback or

$$u_i=-K_{v_i}\dot{e}_i-K_{p_i}e_i \tag{30}$$

To account for external disturbances an integral control term of the form $K_I\varepsilon$ is added where

$$\dot{\varepsilon}_i=e_i \tag{31}$$

A robot link control torque then becomes

$$\tau_i=-u_i=K_{v_i}\dot{e}_i+K_{p_i}e_i+K_{I_i}\varepsilon_i \tag{32}$$

If each joint actuator were modelled as having lumped inertia and damping values of J and B, the closed loop characteristic polynomial obtained from the error dynamics is

$$s^2+K_{v_i}s+K_{p_i}=0 \tag{33}$$

The standard form for the characteristic polynomial is

$$s^2+2\zeta_{n_i}\omega_{n_i}s+\omega_{n_i}^2=0 \tag{34}$$

where ω_{n_i} is the joint error natural frequency and ζ_{n_i} is the damping coefficient. Therefore the desired joint control performance can be achieved by selecting the appropriate gains. Equating the two polynomials gives

$$K_{p_i}=\omega_{n_i}^2 \tag{35}$$

and

$$K_{v_i}=2\zeta_{n_i}\omega_{n_i} \tag{36}$$

To obtain critical damping (no overshoot) for example then

$$\zeta_{n_i}=1 \tag{37}$$

and the gains would be related by

$$K_{v_i}=2\sqrt{K_{p_i}} \tag{38}$$

The upper limit of the gains are related to the structural frequency of the link, ω_r. The first flexible resonant mode of the link is given as

$$\omega_{r_i}=\sqrt{\frac{k_{r_i}}{J_{r_i}}} \tag{39}$$

where J_{r_i} and k_{r_i} are the link inertia and stiffness respectively. Therefore to avoid exciting the links resonant mode the joint error frequency should be chosen to be less than half the maximum link frequency (configuration dependent usually) or

$$\omega_{n_i}<\frac{\omega_{r_i}}{2} \tag{40}$$

Of course the joint torque upper limit will also limit the gains.

A PD position controller was implemented as developed above for each link in the simulations, ignoring the integral term because gravity was not used and no disturbance terms were introduced into the system (when the integral term was incorporated, as verification, it made virtually no difference in the tracking). The PD controller is very effective in trajectory tracking, with large gains. Equations (35) to (40) were used for setting the gains. A block diagram of the controller can be seen in Figure 5. Torque limits are

https://doi.org/10.1017/S0263574799001186 Published online by Cambridge University Press

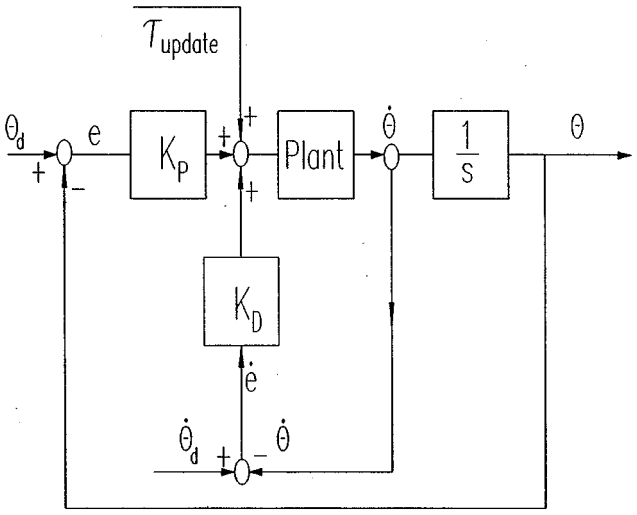

Fig. 5. Position controller.

incorporated to simulate actuator constraints present in all robot actuators.

One thing to note is the torque added to the error torque labelled τ_{update} in the block diagram. This torque is added to the torque term immediately after a trajectory update occurs. This is because at the beginning of an update (and at $t=0$) the joint errors are zero. At the first update step the actual joint trajectories are equal to the (new) desired trajectories and hence there is no torque. Prior to the update there were joint errors since the controller is not perfect and no system can realize infinite gains. What this update torque does is add a constant torque term (for the rest of the movement) equal to the last torque value before the update. The closed loop controller takes care of this offset. If this torque offset is not added, immediately after the torque being zero, there is a very large correction torque which results in trajectory discontinuities.

7. SPACE MANIPULATOR WORKSPACES AND DYNAMIC SINGULARITIES

Unlike singularities for fixed-base manipulators that are solely a function of kinematics, space manipulators without base motion control encounter dynamic singularities.[13,18] These singularities occur when the end-effector cannot move in some inertial direction, and are a function of the dynamic properties of the manipulator and base. Dynamic singularities occur when the system generalized Jacobian, J^* becomes rank deficient or non-invertible. The Jacobian for a manipulator-base system relates the end-effector's linear and angular velocities in inertial space, to the controlled manipulator joint angles. Unlike fixed-base Jacobians, this generalized Jacobian depends upon dynamic parameters as well as kinematic ones.

Dynamic singularities are path dependent, depending upon the history of the spacecraft attitude. And because of the dynamic coupling in the system, the spacecraft attitude depends upon the history of the manipulator motion. This path dependence is due to the non-integrability of the angular momentum of the system. Physically, each point in the manipulator workspace can be reached with an infinite number of system configurations. Therefore a point may or may not be singular depending upon the path used to get there.

The workspace for a space manipulator is therefore related to the system's dynamic singularities. When the spacecraft attitude is controlled (e.g. with gyroscopes), the manipulator workspace is termed the *Reachable Workspace*,[13,18] and is a sphere centered at the system centre of mass. This represents a maximum workspace for the manipulator-base system. The workspace in which dynamic singularities can occur is termed the *Path Dependent Workspace* (*PDW*). Points in this space can only be reached if a suitable path is taken. The difference between the *Reachable Workspace* boundary and *Path Dependent Workspace* is termed the *Path Independent Workspace* (*PIW*). Points in this workspace can always be reached, and can never lead to dynamic singularities. Graphical examples of these terms are illustrated in Figure 6.

To illustrate these terms, points A through D are labelled in Figure 6. This figure and the following example are taken from Papadopoulos and Dubowsky.[13] Because points B and D both lie in the *PIW*, either point can be reached from the other, or from any point found in the region labelled *PIW*. Point A or C can also be reached from any point (movement) in the *PIW*. But a point such as A in the *PDW* may or may not be reachable directly from point C, or from another point in the *PDW*. When the term directly is used here, it means directly as in a straight line or direct type path. But this does not mean that the point can never be reached. The manipulator may have to perform repetitive movements (i.e. where the end-effector traces out a closed loop path repetitively) to re-orient the base. The manipulator may then be able to move to point A directly from this new configuration. At a dynamic singularity, the end-effector may still be able to move, just not in certain directions. To maximize the *PIW*, manipulator redundancy may be utilized if it exists.

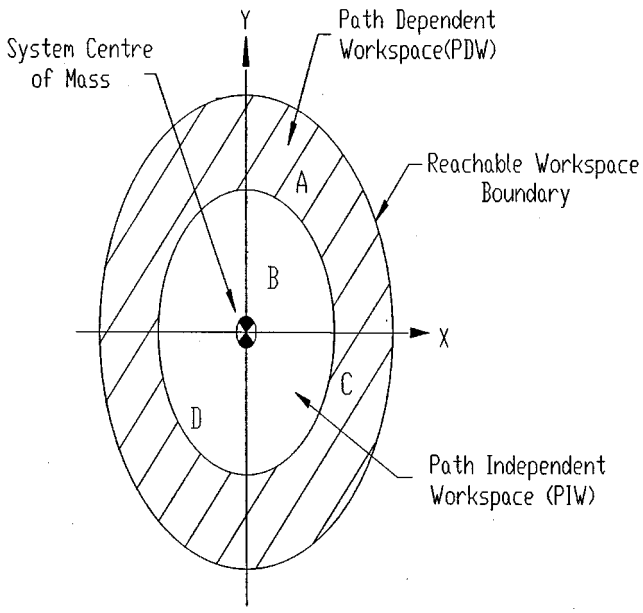

Fig. 6. Space manipulator workspaces.

https://doi.org/10.1017/S0263574799001186 Published online by Cambridge University Press

Because calculation of the dynamic singularities and the various workspaces depend upon all the dynamic parameters of the system, these concepts were not incorporated into the motion control algorithm or the trajectory generator developed in this thesis. The main principle behind developing the motion control or updating algorithm in this work, is that the dynamic parameters of a system *need not be known*. This is true, except for the problem of dynamic singularities. And as discussed previously, for many systems in space, values for these parameters may be very difficult to obtain.

The one type of singularity that is implemented in this thesis, is the standard fixed-base manipulator kinematic singularity. If this type of singularity is reached, the movement is terminated. When a dynamic singularity is encountered, this is not explicitly known by the algorithm. Consequently, joint servoing continues until a kinematic singularity is reached. The point may still be reachable using a suitable path in conjunction with the workspace principles discussed in this section.

8. DYNAMIC SIMULATION RESULTS

8.1 Base-Manipulator Mass Ratio

For the simulations presented herein, the robot links are both 2 meters long, have masses of 100 kilograms each, and are modeled as solid bars for inertia matrix calculations. The base is 2 meters by 2 meters square. The mass of the base varies in the simulations, but is set to 600 kilograms in most of the simulations. The entire system is assumed to be at rest at the beginning of the simulations, and therefore comes to rest at the end of the movement (link velocities of zero). Non zero robot or base initial conditions can be specified but were not used in any simulations shown in this paper. The simulations were performed using SI units, but for the results plots, the lengths are shown in cm.

When a movement occurs without any trajectory updating, the end-effector of the robot will (if the controller is tuned properly) reach the desired position in the robot coordinate frame (fixed) but not in the inertial frame. The amount by which the end-effector misses the desired position (inertial frame) will depend on the size of the commanded joint movements and the base-arm mass (and inertia) ratio, amongst other things. If the mass ratio of the base to arm (termed R_M) is very large then the system is essentially a fixed base system. The amount of base disturbance will depend on the R_M as well as on how fast the movement is completed. Figure 7 shows the end-effector trajectories for mass ratios, R_M, varying between 3 and 100.

8.2 Multiple updates and update frequency

If updating is done throughout a movement the variables to be chosen are when to begin updating and how frequently should updates be performed. A large number of simulations were done and several observations were made. One observation was that early updating yielded better results. Another observation was that frequent updating was better than infrequent updating. Figure 8 shows the system for a movement with and without multiple updating, the frequency being 2.5% of total time starting at 80% (two updates). Without an update, the robot achieves its commanded position in the robot frame but in the inertial frame misses the desired point by approximately 190 cm. If updating begins at 80% of the total movement time, the error is reduced to 12 cm as opposed to 100 cm for a single update at 80%. Clearly there is a significant improvement in even two updates (over one) if performed near the end of the movement.

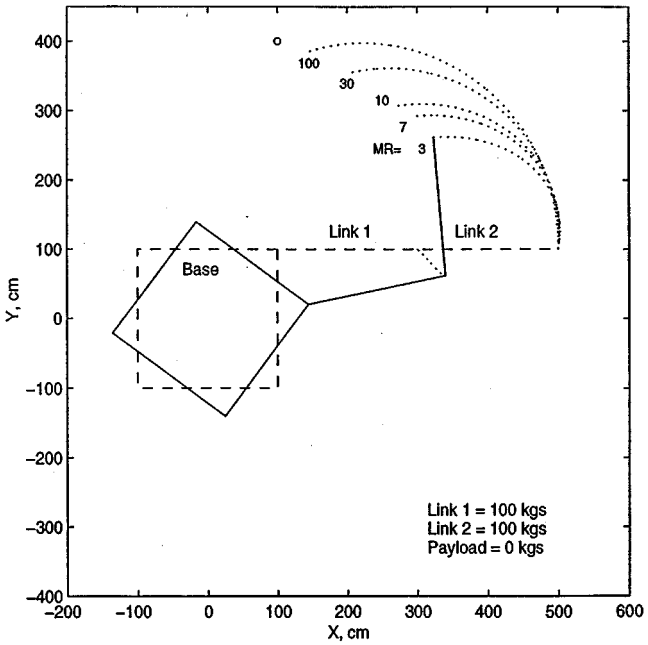

Fig. 7. End-effector trajectories – variable mass ratio.

Figure 9 shows end-effector trajectories of movements at 80%, 60%, and 0% (no update). It is observed here that when updating was started at 60% of the movement the end-effector error dropped to 23 cm, in contrast to an error of 41 cm with multiple updates starting at 80%. This shows

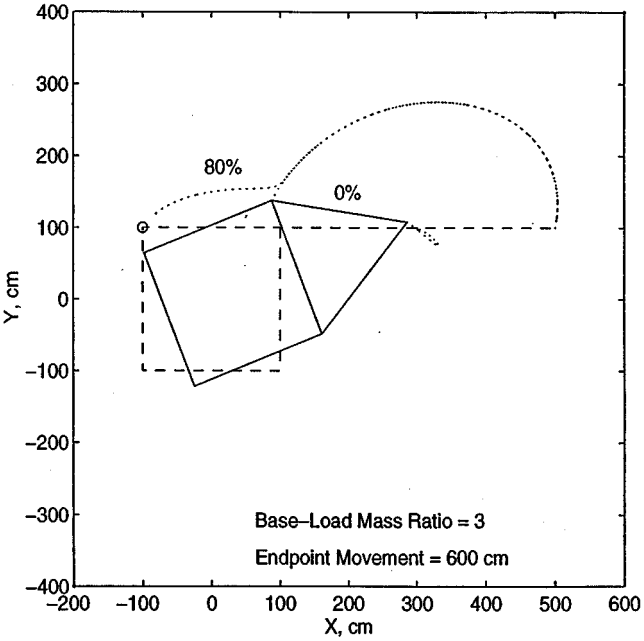

Fig. 8. Multiple update end-effector trajectories.

https://doi.org/10.1017/S0263574799001186 Published online by Cambridge University Press

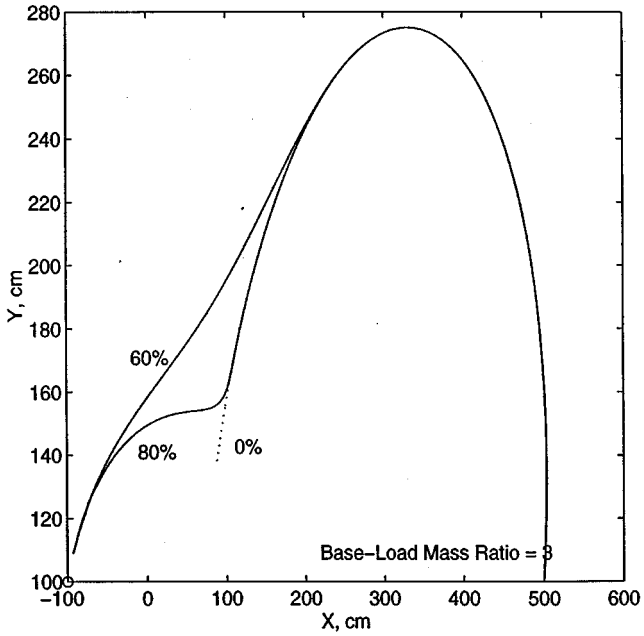

Fig. 9. Multiple update end-effector trajectories close up.

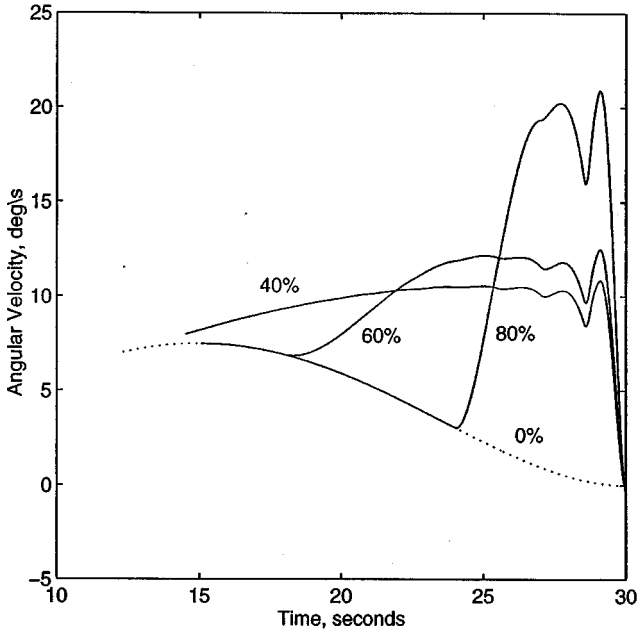

Fig. 11. Multiple update $\dot{\theta}_1$ vs. time.

that early updating results in less error. This is the opposite of single updates, where a later update is better.

Figure 10 shows the joint angles for the different update times. In this plot at each update the final desired angle increases. Figure 11 and 12 show the joint angular velocity and torque for joint 1 only. The earlier the updating the smaller the required velocities and torques (the opposites of single updates) and hence less base disturbance. This is because there is no requirement for a large error correction at the last moment.

Performing multiple updating may give acceptable end-effector accuracy depending on the requirements. An example of this is shown in the next plots for a desired position of (0,0) cm. Figure 13 shows the end-effector error for 10%, 5%, and 2.5% frequency updating on a single plot. Each circle represents one 30 second simulation. The most important observation inferred from this (and many other) plots is that there is an optimum time to start updating. Since all the curves are relatively flat between 20% and 60%, updating any earlier than 60% results in very little change in the final end-effector error. And because the curves (2.5, 5, or 10%) begin increasing sharply after about 75%, it is now possible to conclude that the optimum time to begin updating is approximately between 50 and 65% of the total movement time depending on the distance between the desired and initial end-effector positions. This can be

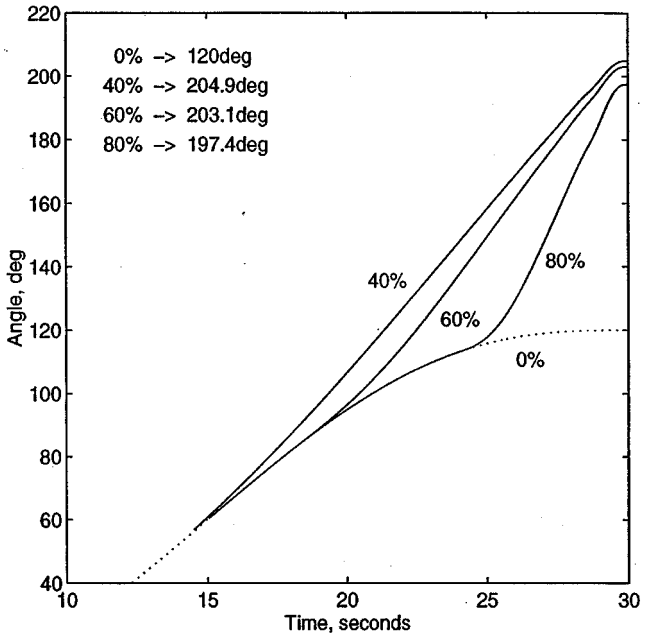

Fig. 10. Multiple update θ_1 vs. time.

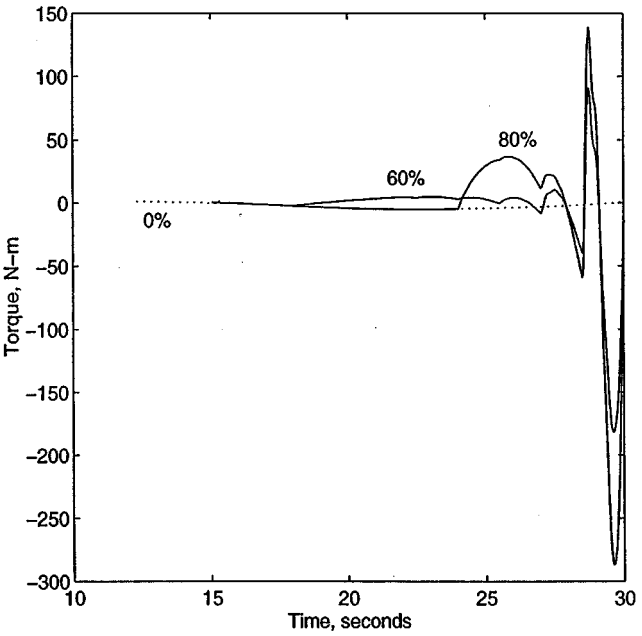

Fig. 12. Multiple update joint 1 torque vs. time.

https://doi.org/10.1017/S0263574799001186 Published online by Cambridge University Press

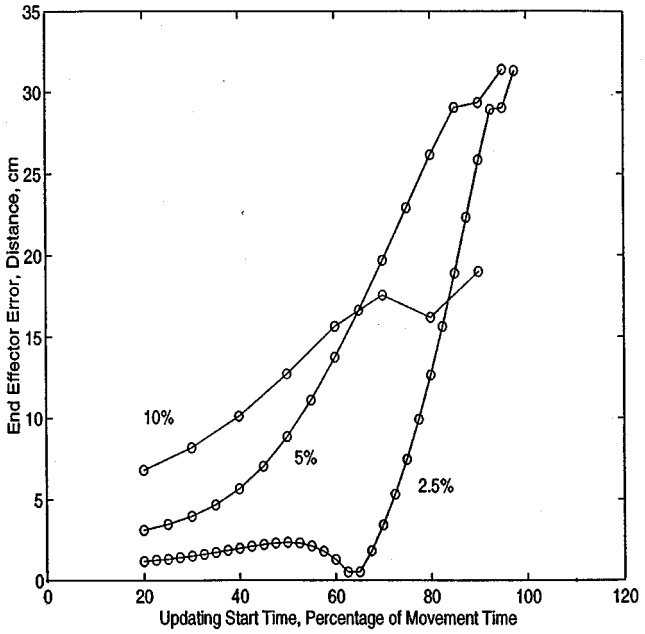

Fig. 13. Multiple update end-effector errors – $x=0$, $y=0$.

thought of as when the robot begins closing in on the desired position.

If the desired accuracy is less than say 2 cm, beginning updating at 65% with a 2.5% frequency will result in the desired accuracy. For some desired points however using frequent updating starting at an optimum time may still not yield sufficient accuracy. Then one of the methods, endpoint over-compensation and/or endpoint servoing, will have to be incorporated.

8.3 Endpoint overcompensation and endpoint servoing

If the accuracy obtained by performing an optimum set of multiple updates during a movement is not high enough, endpoint overcompensation (*EOC*) can be performed during the last part of the movement. In this method the desired end-effector position is moved slightly beyond the original one for the inverse kinematics recalculation. This is done linearly, such that if the desired position was (0,0) and the end-effector was at (1,1), if the update occurred at this point, the algorithm would attempt to servo to $(-1, -1)$ (or some smaller distance in the direction of $(-1, -1)$). This could be done for each update or only for the last couple of updates. In one simulation, endpoint error was 42.6 cm without *EOC* and 8.7 cm with *EOC* enabled. It should be noted that endpoint overcompensation appears to improve performance in only a few configurations, particularly when the desired point is almost but not quite out of the manipulator's workspace.

If further accuracy is required after multiple updates and possibly end-point overcompensation, then one or more smaller manipulator movements are performed and are referred to as endpoint servoing (*EPS*). At this point the manipulator end-effector will be relatively close to the desired position. With the system now at rest, another movement will be performed not unlike the one that previously occurred in that multiple updating. But this movement will take less time to complete than the first large movement, and the end-effector can be positioned where desired with as high a precision as required. If very close positioning is necessary, then multiple movements may be required. It should be noted that since the manipulator is already close to the desired point thereby requiring small joint angular movements, there will be much less base disturbance than with the first large movement. As an example, in one simulation end-effector error was 24.2 cm without *EPS* at the end of a 30 second multiple updating movement. After the first 6 second *EPS*, the error was 6.3 cm, and after the second 6 second servo, the error is reduced to 1.9 cm (total movement 12 cm). The algorithm is set up so either a set number of *EPS* will be performed or alternatively, a maximum allowed error can be specified and *EPS* will continue until the error is less than this value.

9. INVERSE KINEMATICS GUIDELINES AND PERFORMANCE INDICES

To evaluate the performance for the results of a manipulator movement, certain criteria must be specified with which to judge the inverse kinematics updating algorithm performance. Some typical specifications that are listed when describing a (fixed base) robot are maximum payload, maximum joint and/or Cartesian speed, end-effector accuracy and repeatability, and Cartesian tracking error (adherence to a straight line). Specifications such as maximum joint velocity, payload, and repeatability are functions of the mechanical properties of the manipulator in question. Cartesian tracking error and accuracy (and repeatability) are dependent not only upon the mechanical properties of the robot but also on the performance of the control system. But because the manipulator is simulated and controller gains are chosen large enough to result in very accurate joint angular control, the manipulator is assumed perfect. Therefore manipulator end-effector accuracy in the inertial frame, as opposed to in the manipulator base frame, is used as a performance index in this thesis. Another performance index often used is the energy expended by the manipulator during a movement. This is a variable to be minimized or optimized when a movement is performed. Obviously performing manipulator – base positioning without the use of thrusters or gyroscopes is pointless if more energy is consumed by the manipulator using an updating algorithm than with thrusters or gyroscopes. Energy consumed will not be used as a performance index, but it was shown in Carter[14] that performing multiple updates throughout a movement uses less energy than performing one movement without updating, then another without updating, etc.

Base disturbance or the robot base dynamic coupling factor is another criteria by which a particular manipulator movement can be related to system performance. The lower the base acceleration and movement the better the movement is judged to be. An extreme example would be a very large, fast movement with a heavy payload performed by the CANADARM attached to Space Station Alpha. If this type of movement were possible and a large base disturbance resulted it would disturb and possibly be detrimental to people and equipment on board. Because of the nature of the on-line updating that this algorithm uses, the only way

https://doi.org/10.1017/S0263574799001186 Published online by Cambridge University Press

in which base disturbance can be minimized is by specifying low joint velocities and hence time scaling, which will result in minimal base disturbance. To actually use this as a variable, pre-movement off-line calculations that require all manipulator and base dynamic parameter are required. A good discussion and application of this is given by Xu.[19] Here the author discusses the use of a performance index in planning robot motion, evaluating robot trajectories in minimizing base motion, and for optimizing the robot configuration design and robot base location on the base. The number of updates required to complete a movement with the desired accuracy can also be viewed as a type of performance index. Each time an update is performed there are inverse kinematics and possibly time scaling calculations to be done. These require CPU time and take some length of time to complete. Therefore the number of updates should be minimized, although accuracy is higher priority than update minimization. And for expended joint power minimization as discussed in the previous section, it was shown that more updates are preferred. Therefore the number of updates will be minimized if reducing the number has no effect on (primarily) accuracy and (secondarily) energy expended.

9.1 Guidelines for inverse kinematics updating sequences

To determine the sequence of events that should occur during a movement, the desired results (i.e., accuracy) and the variables which affect accuracy should be stated. As discussed in the last section, the main system performance index will be the inertial frame end-effector accuracy. Other performance indices that will be considered but not strictly adhered to are the number of updates, manipulator energy expended, and manipulator-base dynamic coupling or disturbance. The fixed system factors on which accuracy depends on are the base-arm mass ratio and the total endpoint movement. The total endpoint movement is also related to joint angular displacements. The algorithm factors that are varied to produce the required accuracy are the time that the update occurs, the frequency of the updates or the update sequence, endpoint overcompensation, and endpoint servoing. The secondary performance indices that can be minimized with no effect on accuracy are joint energy expended and dynamic coupling as these depend on total movement time. As discussed in the section on time scaling, the movement time can be lengthened to minimize the energy consumed and the base disturbance, and will have no effect on accuracy. To devise an algorithm that will produce the desired accuracy for any system initial configuration and properties, a sub-algorithm or equation is written that shows the dependencies of these factors. It will be used as a guideline for determining the updating sequence. First all the variables are assigned a notation and their dependencies shown in the general equation below

$$(ACC_e, EN_j, NUM_u, DIST_b)$$

$$= func(R_M, \Delta_e, T_u, S_u, EOC, EPS, T_{total}, V_j, \tau_j) \qquad (41)$$

where

- ACC_e: Accuracy of the end-effector position in the inertial frame
- EN_j: energy expended by a manipulator during a movement
- NUM_u: number of updates
- $DIST_b$: base disturbance during movement
- R_M: base-manipulator mass ratio
- Δ_e: total required end-effector movement distance
- T_u: time during a manipulator movement when an update is performed
- S_u: Update sequence during a manipulator movement
- EOC: endpoint over compensation
- EPS: endpoint servoing at the end of a movement
- T_{total}: total movement time
- V_j: maximum joint velocities
- τ_j: maximum joint torques

The variables that will be incorporated into the algorithm for general cases are shown in the equation below.

$$(ACC_e) = function(R_M, \Delta_e, T_u, S_u, EOC, EPS) \qquad (42)$$

Based on the results of the previous sections, we know that end-effector accuracy and therefore when the updating should start depends largely on the R_M. We write a general analytical equation of updating start time determined through experimentation as a function of R_M as follows:

$$T_u = 55 + 35(1 - 2^{-\frac{R_M}{15}}) \qquad (43)$$

A graph of this function is seen in Figure 14.

The coefficients are varied so that the curve fits the previously determined observations. Updating in any case should not begin sooner than 55% of the way through the movement and no later than approximately at 90% of the way through the movement. The factor of 15 in the decaying exponential component was chosen so that the update time

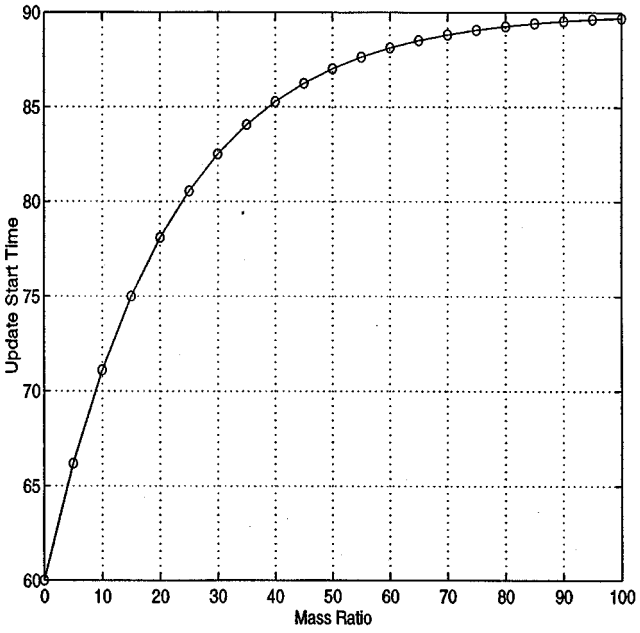

Fig. 14. Update start time vs. mass ratio.

https://doi.org/10.1017/S0263574799001186 Published online by Cambridge University Press

would be most "sensitive" for low mass ratios, and less sensitive for higher mass ratios (larger slope at smaller mass ratios). T_u can also be related to Δ_e or total required end-effector distance knowing that regardless of the mass ratio, if the manipulator end-effector is to move only a small distance relative to the total manipulator length, then updating could start a little later than would be justified by Equation (43). If a very large distance is required (maximum 200% or twice manipulator length) then updating should start sooner. Therefore T_u will be multiplied by a factor k_T related to the end-effector distance. This factor will vary between 0.8 and 1.2 depending on Δ_e, the equation being a linear one and written as

$$k_T = \left(1.2 - 0.002\left(\frac{\Delta_e}{200}\right)\right) \tag{44}$$

This equation was derived to give the desired function slope. A graph of this function is seen in Figure 15.

The new update time is then given by

$$T_u = k_T T_u \tag{45}$$

It has been determined that the higher the update frequency the better the final accuracy. And to reduce the total number of updates required a function will be chosen so that the update frequency increases as the movement progresses. As a guideline for an updating sequence function, a decaying exponential curve is used similar to the one used to determine T_u. The sequence should begin approximately at T_u, and the latest an update should occur is approximately 96% of the total movement time. If time and velocity scaling are enabled an update can occur later and excessive torques and velocities will not result due to the increased movement time. The experimental function is written as

$$S_u = T_u\left(2 - 2^{-\frac{k_{step} - Tu}{c_f}}\right) \tag{46}$$

where c_f is given as

$$c_f = 25 + (T_u - 60) \tag{47}$$

and k_{step} is a linear sequence used as a reference sequence starting at T_u and increasing in steps to approximately the time of the last update or T_{last}. A step size for this sequence is obtained using

$$\Delta k_{step} = \frac{T_u}{20} \tag{48}$$

and is an average size of the actual update sequence interval. The number of steps (N_k) performed is given by result of the following equation rounded up to the next integer value

$$N_k = \frac{T_{last} - T_u}{\Delta k_{step}} \tag{49}$$

Because of the rounding up of the number of steps, if starting at T_u and adding steps of size Δk_{step} to T_u N_k times, the final update time will be slightly larger than T_{last}. Therefore the difference between these values is subtracted from each element in the updating sequence so that the step size is Δk_{step} and so that the last step occurs at T_{last}. The result is that T_u is smaller so that updating begins a little sooner than the original value. The new S_u updating sequence is calculated as shown below

$$S_u = S_u - (S_{old}(last)) - T_{last}) \tag{50}$$

where $S_{old}(last)$ is the last value of the original updating sequence. A graph of this updating sequence is shown in Figure 16. Curves for sequences starting at different T_u values are shown. As stated previously, the updates occur more frequently as the movement progresses.

If further accuracy is required after this sequence, which may or may not incorporate endpoint overcompensation,

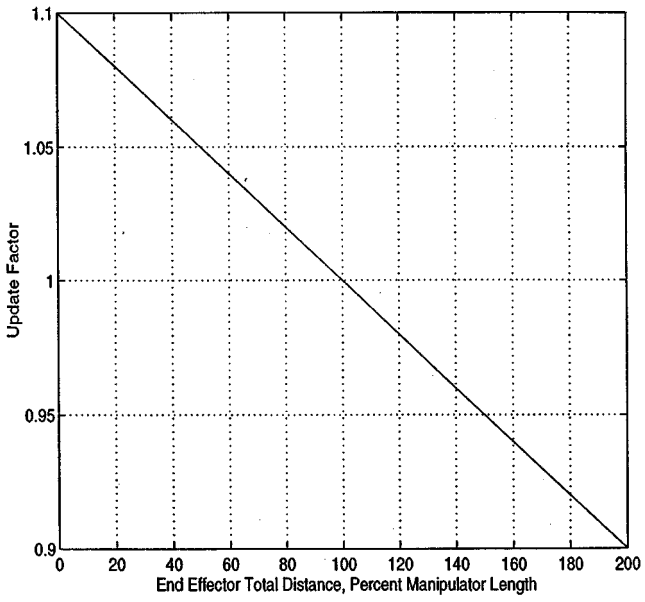

Fig. 15. Update factor vs. end-effector total distance.

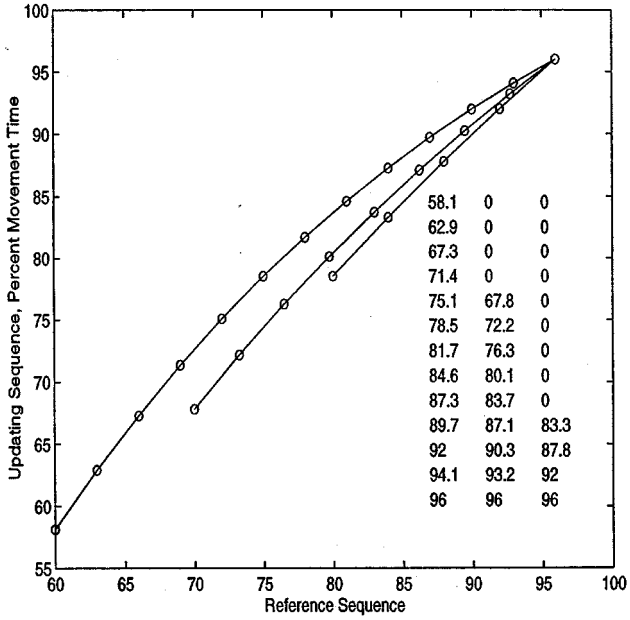

Fig. 16. Updating sequence vs. reference sequence.

https://doi.org/10.1017/S0263574799001186 Published online by Cambridge University Press

then servoing is continued until the desired accuracy is achieved.

10. FURTHER EXAMPLES AND ALGORITHM VERIFICATION

In this section simulations to test the previous algorithms were performed with systems having different mass ratios, desired end-effector locations, and accuracy requirements. From these results an overall performance rating for the updating algorithm can be obtained. The result from a series of simulations with R_M equal to 25 are shown in Table I.

For this mass ratio there were 25 different (x, y) positions that the end-effector was commanded to servo to. The algorithm was set up to use multiple updating for the first movement and up to 5 endpoint servos to achieve a desired end-effector accuracy of 1.0 cm or less. Where there is only one value shown, this is the ACC_e after the first 30 second movement. Where there are 2 values separated by a semicolon (:), the first element is the error after the initial movement and the second value is the error after one 6 second endpoint servo. For a fairly large R_M such as 25, most of the movements achieve the desired accuracy after only one movement, and the others after one endpoint servo. The results from a series of simulations with R_M equal to 3 are shown in Table II.

Clearly the smaller the R_M, the greater the number of iterations that must be performed to obtain the desired accuracy. Of the 25 desired points, only 1 point was achieved on the first iteration, and 19 others movements were successfully completed in less than five endpoint servos. For 2 points, the desired positions were achievable after a few more iterations, and 3 points cannot be achieved. This is because the desired point is no longer in the manipulator kinematic workspace (standard, fixed base type singularity), due to manipulator-base reorientation around the system centre of mass. There is code in the algorithm that detects this singularity condition, and no more movements are performed. If joint range limits are incorporated (± 180 degrees), and a joint angle exceeding the limit is required, then the movement is terminated as the desired point cannot be achieved. Simulations were performed with R_Ms other than presented in the Tables I and II, but are not shown as similar conclusions can be inferred from the previous discussion. In Figure 17 a system movement (desired point 200, −200) is shown in which 5 iterations are required to achieve an accuracy of less than 1 cm. In Figure 18 a system movement (desired point 0–200) is shown in which the desired point is no longer reachable after 4 iterations. The manipulator is in a singular configuration.

To achieve the desired end-effector setpoint and required accuracy with a minimum number of endpoint servos, endpoint overcompensation is utilized for movements with large Δ_e such as (100, 200) and (200, 200). These two points required more than five iterations to achieve the desired accuracy. If the end-effector error is greater than 25% of the total manipulator length after the first movement, *EOC* is used for the remaining iterations. For points that are no longer in the manipulator kinematic workspace, no difference is seen using *EOC*. The results comparing the final error for movements with and without *EOC* are shown in Table III.

Endpoint servoing in conjunction with endpoint overcompensation results in less iterations to achieve the desired accuracy.

From the results presented and from numerous other test simulations performed, it is evident that as long as the desired point is in the manipulators kinematic workspace and the joint range limits are not reached, a point can be reached to any desired accuracy. The updating algorithm also enables the manipulator to achieve this with a high

Table I. End-Effector Error – R_M=25

		X desired position, cm				
		−200	−100	0	100	200
Y desired position, cm	200	6.6:0.5	5.7:0.5	5.5:0.5	4.9:0.5	4.6:0.5
	100	2.4:0.1	2.2:0	2.8:0.3	0.9:0	0.3:0
	0	0.8	0.4	0.9	0.3	0.9
	−100	0.1	0.0	0.1	0.2	1.0
	−200	0.2	0.2	0.3	0.8	1.7:0.20

Table II. End-Effector Error – R_M=3

		X desired position, cm				
		−200	−100	0	100	200
Y cm	200	99:72:67:67	80:44:41:41	61:30:17:12:1	50:25:14:8:4	42:24:15:10:6
	100	51:11:2:0.6	27:5:1.6:0.5	24:6:1.9:0.6	14:7:4:2:1	1.9:0.8
	0	17:11:2:0.6	6:0.7	7:0.1	2:0.2	8:1.3:0.3
	−100	4:0.3	1.0	1.3:0.1	0.8	9:4:1.9:0.6
	−200	11.8:1.6:0.4	5:0.5	6:0.8	10:2:0.6	18:7:3:1.5:0.7

https://doi.org/10.1017/S0263574799001186 Published online by Cambridge University Press

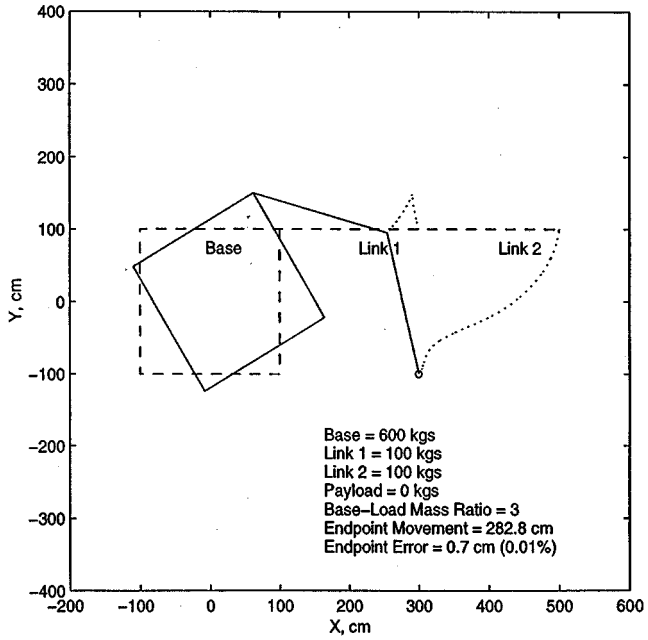

Fig. 17. System movement – five endpoint servos.

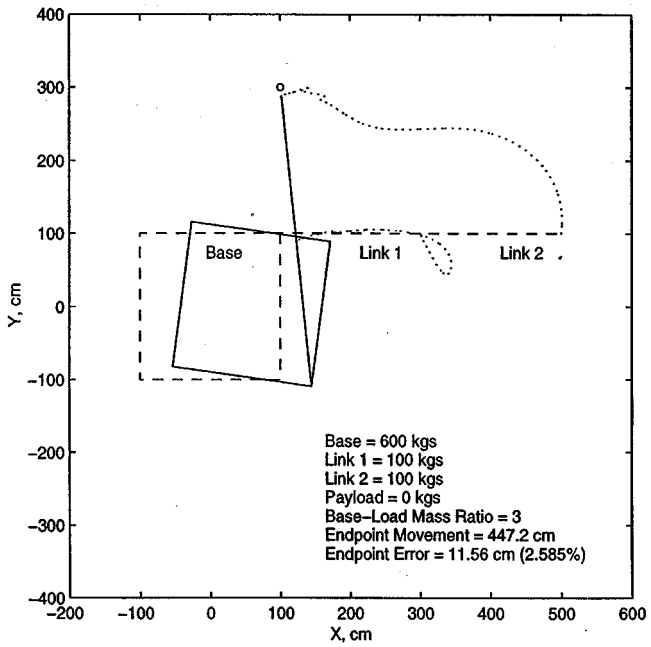

Fig. 18. System movement – manipulator singularity.

Table III. End-Effector Error – *EOC*

	(100, 200)	(200, 200)
No *EOC*	2.6 cm	4.0 cm
With *EOC*	0.8 cm	0.9 cm

degree of stability and a minimal amount of energy expended.

11. CONCLUSIONS

A motion control algorithm was developed for enabling a manipulator mounted on a non-fixed base to achieve a desired position in inertial space. The algorithm is implemented without base control, and the exact dynamic parameters of the system are not required. A joint trajectory generator utilizing velocity time scaling and quintic polynomials was developed and a simple PD position controller was used for controlling the manipulator and was shown to provide good servoing performance. Multiple inverse kinematic updates are done throughout a movement based on the motion control algorithm. The algorithm takes into account such factors as manipulator – base mass ratio, total required end-effector movement, and required accuracy and generates an updating sequence that starts at an optimal time. Updates are performed as specific times that result in better accuracy than multiple single inverse kinematic calculation movements.

References

1. M.A. Bronez, M.M. Clarke, A. Quinn, "Requirements development for a free-flying robot–the 'ROBIN'," *Proc. IEEE Int. Conf. on Robotics and Automation* (1986).
2. R. French, B. Boyce, "Satellite servicing by teleoperators", *J. Engin. Industry* **107**, 49–54 (1985).
3. S. Lee, G. Bekey and A.K. Bejczy, "Computer Control of space-borne teleoperators with sensory feedback", *Proc. IEEE Int. Conf. on Robotics and Automation* (1985).
4. R.W. Longman, R.E. Lindberg and M.F. Zedd, "Satellite-mounted robot manipulators–New kinematics and reaction moment compensation". *Int. J. Robot. Res.* **6**(3), 87–103 (1987).
5. Z. Vafa and S. Dubowsky, "The Kinematics and Dynamics of Space Manipulators: The Virtual Manipulator Approach", *Int. J. Robot. Res.* **9**(4), 3–21 (1990).
6. H.L. Alexander and R.H. Cannon, "Experiments on the control of a satellite manipulator", *Proc. 1987 American Control Conf.* (1987).
7. Y. Umetani and K. Yoshida, *Experimental Study on Two Dimensional Free-flying Robot Satellite Model (NASA Space Telerobotics Workshop, Jet Propulsion Laboratory,* 1989).
8. O. Egeland, "Task-space tracking with redundant manipulators", *IEEE J. Robot. Automation* **RA-3(5)**, 471–475 (1987).
9. O. Khatib, "Augmented object and reduced effective inertia in robot systems", *Proc. American Control Conference* (1988), pp. 2140–2147.
10. J.K. Salisbury and J.D. Abramowitz, Design and control of a redundant mechanism for small motion. *Proc. IEEE Int. Conf. on Robotics and Automation* (1985) pp. 323–328.
11. O. Egeland and J.R. Sagli, "Coordination of motion in a spacecraft-manipulator system", *Int. J. Robot. Res.* **12**(4), 366–379 (1993).
12. C. Fernandes, L. Gurvits and Z.X. Li, "Attitude Control of Space Platform/Manipulator System Using Internal Motion", *IEEE Int. Conf. on Robotics and Automation* 893–898 (1992).
13. E. Papadopoulos and S. Dubowsky, "Dynamic Singularities in Free-Floating Space Manipulators", *ASME J. Dyn. Systems, Measurement and Control* **115**, 44–52 (1993).
14. F.M. Carter, "Motion Control of Non-Fixed Base Robotic Manipulators". *M.A.Sc. Thesis* (Mechanical Engineering, University of British Columbia, 1997).

https://doi.org/10.1017/S0263574799001186 Published online by Cambridge University Press

15. C.H. An, C.G. Atkeson and J.M. Hollerbach, *Model Based Control of a Robot Manipulator* (MIT Press, Cambridge, Mass., 1988).
16. J.J. Craig, *Introduction to Robotics* (Addison-Wesley Publishing Company, Reading, Mass., 1986).
17. D.M. Dawson, "Uncertainties in the control of robot manipulators", *Ph.D. Thesis* (School of Electrical Engineering, Georgia Institute of Technology, 1990).
18. E. Papadopoulos, "On the Dynamics of Control of Space Manipulators". *Ph.D. Thesis* (Department of Mechanical Engineering, MIT, 1990).
19. Y. Xu, "The Measure of Dynamic Coupling of Space Robots", *IEEE Int. Conf. on Robotics and Automation* (1993) pp. 615–620.

https://doi.org/10.1017/S0263574799001186 Published online by Cambridge University Press