Robust Trajectory Planning for Robotic Manipulators Under Payload Uncertainties

KANG G. SHIN, SENIOR MEMBER, IEEE, AND NEIL D. McKAY, MEMBER, IEEE

Abstract—**A number of trajectory planning algorithms are available for determining the joint torques, positions, and velocities required to move a manipulator along a given geometric path in minimum time. These schemes require knowledge of the robot's dynamics, which in turn depend upon the characteristics of the payload which the robot is carrying. In practice, the dynamic properties of the payload will not be known exactly, so that the dynamics of the robot, and hence the required joint torques, must be calculated for a nominal set of payload characteristics. But since these trajectory planners generate nominal joint torques which are at the limits of the robot's capabilities, moving the robot along the desired geometric path at speeds calculated for the nominal payload may require torques which exceed the robot's capabilities.**

In this paper, bounds on joint torque uncertainties are derived in terms of payload uncertainties. Using these bounds, a new trajectory planner is developed to incorporate payload uncertainties such that all the trajectories generated can be realized with given joint torques. Finally, the trajectory planner is applied to the first three joints of the Bendix PACS arm, a cylindrical robot to demonstrate its use and power.

I. Introduction

BECAUSE their dynamics are highly nonlinear and coupled, optimal control of robots is a very difficult problem. In order to simplify the problem, a common approach is to divide the control of the robot into two sequential segments: off-line trajectory planning and on-line tracking. Using a geometric path as its input, the trajectory planner generates positions, velocities, accelerations, and joint torques as functions of time; the tracker makes adjustments, in real-time, to the nominal inputs to the robot in an attempt to make the robot's actions coincide with those described by the trajectory planner.

Various algorithms are available for performing trajectory planning for robots, i.e., generating desired positions, velocities, accelerations, and torques as functions of time [1]-[3]. These trajectory planners require knowledge of the robot's dynamics, which in turn depend upon the characteristics of the payload being carried. In practice, the exact characteristics of the payload will not be known; since the trajectory planners referenced above need to know the exact dynamics of the robot, the trajectory planning process must be carried out with dynamics which are calculated for a nominal payload. This practice can lead to difficulties. To see why this is the case, note that these trajectory planners generate nominal torques which are at the limits of the robot's capabilities *for the given dynamics*. Moving the robot along the desired path at speeds calculated for the nominal payload may therefore require torques which are beyond the robot's capabilities if the payload differs from the nominal one. If the robot's joints are controlled by independent servos, as is usually the case, then attempting to make the robot move along the nominal trajectory will result in one or more joints "falling behind," so that the robot strays from the desired geometric path. In other words, the trajectory generated by the planner is *realizable* for the nominal payload, but not for the actual payload.

These are a number of adaptive controllers which can compensate for the changes in load, provided that the plant (i.e., the robot joint drive) does not saturate [4], [5]. However, if the plant saturates, as may happen if the actual and nominal payloads differ too much, then these controllers cannot possibly compensate for load changes. It is the objective of this paper to present an analysis of the torque errors caused by payload changes, and incorporate the error information into the trajectory planning process so as to avoid saturation of the individual actuators.

Changes in payload characteristics will be expressed as errors in the *pseudo-inertia* of the payload; the pseudo-inertia is a matrix containing the mass and first and second moments of the payload. It will be shown that bounds on the joint torque errors can be calculated in terms of the norm of the error in the pseudo-inertia of the payload, given the robot's kinematics. A trajectory planning algorithm will then be presented which can handle uncertainties in the dynamics caused by the payload. If the actual and nominal payloads are described by the pseudo-inertias I_A and I_N, respectively, then for a given positive real number E, the algorithm generates a trajectory which is realizable for *all* payloads I_A for which $\|I_A - I_N\| \leq E$.

The rest of the paper is organized as follows. The robust trajectory planning problem is stated in Section II. In Section III, torque errors are calculated in terms of changes to the payload pseudo-inertia. In Section IV bounds on the joint torque errors are derived in terms of bounds on the norm of the pseudo-inertia error. Section V describes a trajectory planning algorithm in detail which takes into account payload uncertainties. Section VI presents a numerical example, and the paper concludes with Section VII.

II. Problem Formulation

In order to determine errors in the torques, the dynamic equations of the robot are required. In tensor notation, the dynamic equations describing the behavior of a robot take the general form

$$u_i = J_{ij}\ddot{q}^j + C_{ijk}\dot{q}^j\dot{q}^k + R_{ij}\dot{q}^j + g_i \tag{2.1}$$

where u_i is the ith generalized force, q^i is the ith generalized coordinate, J_{ij} is the inertia matrix, and C_{ijk} is an array of Coriolis coefficients, defined by

$$C_{ijk} = \frac{1}{2}\left(\frac{\partial J_{ij}}{\partial q^k} + \frac{\partial J_{ik}}{\partial q^j} - \frac{\partial J_{jk}}{\partial q^i}\right). \tag{2.2}$$

The matrix R_{ij} is the viscous friction matrix, and g_i is the gravitational force. The Einstein summation convention has been

Manuscript received July 10, 1986; revised April 30, 1987. This paper is based on a prior submission of May 9, 1985. Paper recommended by Past Associate Editor, W. J. Book. This work was supported in part by the National Science Foundation under Grant ECS-8409938 and by the U.S. Air Force Office of Scientific Research under Contract F33615-85-C-5105.

K. G. Shin is with the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109-2122.

N. D. McKay is with the Computer Science Department, General Motors Research Laboratories, Warren, MI 48090-9055.

IEEE Log Number 8717025.

used here, so that all product terms in (2.1) are summed from 1 to N over repeated indexes, where N is the number of degrees of freedom of the robot. The inertia matrix J_{ij}, the Coriolis array C_{ijk}, and the gravitational loading vector g_i are functions of the position of the robot and the payload pseudo-inertia.

The path which the robot is expected to follow is assumed to be given as a parameterized curve in joint space, i.e., the joint coordinates q^i are given in terms of a single scalar λ by the equations

$$q^i = f^i(\lambda), \qquad 0 \le \lambda \le \lambda_{\max}. \tag{2.3}$$

This allows all of the joint positions, velocities, and accelerations to be expressed in terms of the scalar parameter λ and its time derivatives. Plugging these relations into the dynamic equations (2.1) gives torque in terms of these quantities as follows:

$$u_i = J_{ij}\frac{df^j}{d\lambda}\dot{\mu} + \left(J_{ij}\frac{d^2f^j}{d\lambda^2} + C_{ijk}\frac{df^j}{d\lambda}\frac{df^k}{d\lambda}\right)\mu^2 + R_{ij}\frac{df^j}{d\lambda}\mu + g_i \tag{2.4}$$

where $\mu \equiv \dot{\lambda}$ is the *pseudo-velocity*. (See [2] for a detailed derivation of these equations.) The quantities J_{ij}, C_{ijk}, and g_i depend upon the masses and moments of inertia of the robot's links. The robot's payload is fixed to the last link, and hence must be regarded as part of the last link for purposes of calculating the dynamic coefficient. Therefore, J_{ij}, C_{ijk}, and g_i will change as the characteristics of the payload vary.

We may write (2.4) as

$$u_i = M_i(\lambda, I_N)\dot{\mu} + Q_i(\lambda, I_N)\mu^2 + R_i(\lambda)\mu + S_i(\lambda, I_N) \tag{2.5}$$

where the coefficients M_i, Q_i, R_i, and S_i are given by $M_i \equiv J_{ij}(I_N)df^j/d\lambda$, $Q_i \equiv J_{ij}(I_N)d^2f^j/d\lambda^2 + C_{ijk}(I_N)df^j/d\lambda df^k/d\lambda$, $R_i \equiv R_{ij}df^j/d\lambda$, and $S_i \equiv g_i(I_N)$. The functional dependence of J_{ij}, C_{ijk}, and g_i on the nominal payload I_N has been shown explicitly. Then, the dynamics of the system after the payload has been perturbed may be written

$$u_i' = M_i(\lambda, I_N + \Delta I_N)\dot{\mu} + Q_i(\lambda, I_N + \Delta I_N)\mu^2 + R_i(\lambda)\mu + S_i(\lambda, I_N + \Delta I_N). \tag{2.6}$$

In order to avoid excessive torque requirements, we wish to compute a set of velocities and accelerations μ and $\dot{\mu}$ such that, for all payload inertia errors ΔI_N within the constraint set, if the nominal torques u_i are given by (2.5), then the actual torques u_i' given by (2.6) will be realizable, i.e.,

$$u_i^{\min}(\lambda, \mu) \le u_i' \le u_i^{\max}(\lambda, \mu). \tag{2.7}$$

Also, there may be jerk (or equivalent) constraints in terms of positions, velocities, and accelerations,

$$|\ddot{u}_i| = |F(q, \dot{q}, \ddot{q})| \le K_i \tag{2.8}$$

where $F: R^N \times R^N \times R^N \to R$ is a jerk function, and K_i a constant. Formally, the *robust trajectory planning* (RTP) problem can be stated as follows.

Given a geometric path described as a parameterized curve (2.3), the torque limits $u_i^{\min}$ and $u_i^{\max}$ as functions of λ and μ, the dynamics of the robot when carrying the nominal payload I_N, and a bound E on the norm of the difference between the pseudo-inertias of the actual and nominal payloads, determine the fastest trajectory (sequence of (λ, μ) pairs) such that the torques u_i' given by (2.6) satisfy the constraints (2.7) for all points on the trajectory and for all payload errors ΔI_N such that $\|\Delta I_N\| \le E$.

We will solve this problem by calculating the worst-case torque error, as a function of λ, μ, and $\dot{\mu}$, for a given payload error, and decreasing the torque limits by this amount when doing trajectory planning.

III. Calculation of Dynamic Coefficient Errors

For a given geometric path, we need to know the changes to the coefficients M_i, Q_i, R_i, and S_i in (2.5) which result from changes in the robot dynamics. In the sequel, changes in dynamics will be assumed to come from changes in payload characteristics, although the methods presented apply equally well to errors caused by uncertainties in the mass distribution of other links and to uncertainties in friction coefficients.

Differences between nominal and actual payload characteristics cause changes in the coefficients J_{ij}, C_{ijk}, and g_i in (2.1). Here we determine the relationship between changes in these coefficients and changes in payload characteristics.

Changes in payload characteristics will result in changes to the pseudo-inertia tensor of the last joint of the robot, i.e., the pseudo-inertia tensor will have the value $I_N + \Delta I_N$ instead of I_N. The coefficients in the inertia matrix are given in [6] as

$$J_{ij}(I_N) = \sum_{p=\max(i,j)}^{N} \operatorname{Tr}\left(\frac{\partial T_p}{\partial q^j} I_p \frac{\partial T_p^T}{\partial q^i}\right) \tag{3.1}$$

where T_p is the 4×4 homogeneous transformation matrix which transforms vectors given in the coordinate system associated with the pth link of the robot to world or base coordinates, and I_p is the pseudo-inertia of the pth link given in the pth link's coordinate frame.

Introducing an error ΔI_N into the pseudo-inertia of the last joint gives

$$J_{ij}(I_N + \Delta I_N) = \sum_{p=\max(i,j)}^{N-1} \operatorname{Tr}\left(\frac{\partial T_p}{\partial q^j} I_p \frac{\partial T_p^T}{\partial q^i}\right) + \operatorname{Tr}\left(\frac{\partial T_N}{\partial q^j}(I_N + \Delta I_N)\frac{\partial T_N^T}{\partial q^i}\right). \tag{3.2}$$

Subtracting (3.1) from (3.2) gives

$$\delta J_{ij}(I_N, \Delta I_N) = \delta J_{ij}(\Delta I_N) = \operatorname{Tr}\left(\frac{\partial T_N}{\partial q^j}\Delta I_N \frac{\partial T_N^T}{\partial q^i}\right). \tag{3.3}$$

Note that the error in the inertia matrix is *linear* in the pseudo-inertia error, and is *independent* of the nominal payload.

Determining the change to M_i, we have

$$\begin{aligned}\delta M_i(I_N, \Delta I_N) &\equiv M_i(I_N + \Delta I_N) - M_i(I_N)\\ &= \left\{J_{ij}(I_N + \Delta I_N) - J_{ij}(I_N)\right\}\frac{df^j}{d\lambda}\\ &\equiv \delta J_{ij}(I_N, \Delta I_N)\frac{df^j}{d\lambda}.\end{aligned} \tag{3.4}$$

To find δM_i, simply plug (3.3) into (3.4), giving

$$\delta M_i = \sum_j \frac{df^j}{d\lambda}\operatorname{Tr}\left(\frac{\partial T_N}{\partial q^j}\Delta I_N \frac{\partial T_N^T}{\partial q^i}\right). \tag{3.5}$$

Computation of the errors δQ_i follows the same pattern as the computation of δM_i. The errors in the Coriolis terms can be determined in much the same way as the errors in the inertia matrix. From [6] we have

$$C_{ijk} = \sum_{p=\max(i,j,k)}^{N} \operatorname{Tr}\left(\frac{\partial^2 T_p}{\partial q^j \partial q^k} I_p \frac{\partial T_p^T}{\partial q^i}\right). \tag{3.6}$$

The errors in the Coriolis terms due to errors in payload characteristics are therefore given by

$$\delta C_{ijk}=C_{ijk}(I_N+\Delta I_N)-C_{ijk}(I_N)=\mathrm{Tr}\left(\frac{\partial^2 T_N}{\partial q^j \partial q^k}\Delta I_N\frac{\partial T_N^T}{\partial q^i}\right). \quad (3.7)$$

The definition of Q_i gives

$$\delta Q_i=Q_i(I_N+\Delta I_N)-Q_i(I_N)$$
$$=\sum_j \mathrm{Tr}\left(\frac{\partial T_N}{\partial q^j}\Delta I_N\frac{\partial T_N^T}{\partial q^i}\right)\frac{d^2 f^j}{d\lambda^2}$$
$$+\sum_j\sum_k \mathrm{Tr}\left(\frac{\partial^2 T_N}{\partial q^j\,\partial q^k}\Delta I_N\frac{\partial T_N^T}{\partial q^i}\right)\frac{df^j}{d\lambda}\frac{df^k}{d\lambda}. \quad (3.8)$$

Now we need to know the error in the gravitational terms. The gravitational forces g_i are given by [6] as

$$g_i=\sum_{k=i}^{N}-m_k G^T\frac{\partial T_k}{\partial q^i}\bar{r}_k \quad (3.9)$$

where $G = [0\ 0\ g\ 0]^T$ is the gravitational force vector, m_k is the mass of the kth link, g is the acceleration due to gravity, and $\bar{r}_k = [\bar{x}\ \bar{y}\ \bar{z}\ 1]^T$ is the center of mass of the kth link given in the coordinates of the kth frame. If we define $w_k \equiv m_k\bar{r}_k$, then we have

$$g_i=-\sum_{k=1}^{N}G^T\frac{\partial T_k}{\partial q^i}w_k. \quad (3.10)$$

But w_k is just the last column of the pseudo-inertia matrix I_k, so that

$$g_i=-\sum_{k=1}^{N}G^T\frac{\partial T_k}{\partial q^i}I_k\begin{bmatrix}0\\0\\0\\1\end{bmatrix}. \quad (3.11)$$

As before, introducing an error into the pseudo-inertia of the last link gives

$$\delta S_i=\delta g_i=-G^T\frac{\partial T_N}{\partial q^i}\Delta I_N\begin{bmatrix}0\\0\\0\\1\end{bmatrix}. \quad (3.12)$$

We may now calculate the error in u_i by adding up the individual components, giving

$$\delta u_i=\delta M_i\ddot{\mu}+\delta Q_i\dot{\mu}^2+\delta S_i. \quad (3.13)$$

IV. Calculation of Torque Error Bounds

If the errors ΔI_N were known exactly, then the torque errors could also be computed exactly. Of course, in practice ΔI_N will not be known exactly. However, if constraints on ΔI_N can be obtained, then we may find bounds on δu_i. We will write these constraints as bounds on the norm of the error ΔI_N.

To obtain these bounds, note that δM_i, δQ_i, and δS_i are all functions of the pseudo-inertia error ΔI_N; in fact, they are *linear* in ΔI_N, so that δu_i is also linear in ΔI_N. If we write

$$\delta u_i=Z(\Delta I_N) \quad (4.1)$$

then we wish to maximize or minimize the linear function Z with respect to ΔI_N, subject to a set of constraints of the form

$$\|\Delta I_N\|\le E. \quad (4.2)$$

At this point, some observations are in order. First, as we noted before, the errors in the δu_i depend linearly upon the pseudo-inertia error ΔI_N. Second, δu_i depends *only* on the kinematics of the robot and on the desired velocity and acceleration, not on the nominal dynamics. These facts are easily proven from the linearity of the Lagrangian equations of motion and the facts that both kinetic and potential energy are linear in mass. One consequence of this linearity in mass is that the errors δu_i do not depend upon the nominal dynamics, as shown above. The implication of this is that much of the error analysis can proceed *without* regard to the nominal dynamics of the robot.

The linearity of the δu_i in the pseudo-inertia has some other practical consequences as well. Consider the maximization which must be performed in order to evaluate δu_i. This maximization requires that the space 4×4 symmetric matrices with norm less than E be searched, which in general is a rather formidable problem. However, by limiting the form of the error constraints, i.e., choosing a particular class of matrix norms, the problem can be made quite simple; in fact, it can be transformed into a linear programming problem.

To see how this transformation can be performed, consider the problem of maximizing the function Z in (4.1), namely

Problem A:

$$\text{maximize } Z(M)=\sum_i\sum_j \beta_{ij}M_{ij} \quad (4.3)$$

$$\text{subject to } \|M\|\le E \text{ and } M=M^T. \quad (4.4)$$

Treatment of the minimization problem proceeds analogously to Problem A. We will show that Problem A transforms into a linear programming problem if the norm used to constrain the matrix M in (4.4) is chosen properly. This will be accomplished by eliminating some absolute values from the constraints.

$Z(M)$, the function to be maximized, is a linear function of M. It remains to be shown that the constraints can be made linear. Of course, if the norm used in Problem A is arbitrary, then in general the constraints will not be linear. However, there is a set of norms, all very easy to calculate, which will yield linear constraints. Consider the class of functions $F{:}R^{4\times4}\to R^+$ given by $F(M) = \max_i \sigma_i(M)$ where $\sigma_i(M) = \Sigma_{j=1}^4 \Sigma_{k=1}^4 \alpha_{ijk}|M_{jk}|$. The matrix 1-norm and ∞-norm, $\max_{i,j}|M_{ij}|$ and $\Sigma_{i,j}|M_{ij}|$, are all functions of this form. It is shown in the lemma of the Appendix that if $\alpha_{ijk} \ge 0$ for all i, j, and k, and if for every pair of indexes (j, k) there is an i such that $\alpha_{ijk} \ne 0$, then $F(M)$ is a norm. Note that this class of norms determines balls in the underlying space which are convex polyhedra; since any convex set can be approximated to any desired degree by a polyhedron, this assumption does not result in any significant loss of generality.

Problem A with this class of norms becomes the following.

Problem B:

$$\text{maximize } Z(M)=\sum_i\sum_j \beta_{ij}M_{ij}$$

$$\text{subject to } \max_i\left(\sum_{j=1}^4\sum_{k=1}^4 \alpha_{ijk}|M_{jk}|\right)\le E \text{ and } M_{jk}=M_{kj}. \quad (4.5)$$

This problem obviously is equivalent to the following.

Problem C:

$$\text{maximize } Z(M)=\sum_i\sum_j \beta_{ij}M_{ij}$$

$$\text{subject to } \sum_{j=1}^4\sum_{k=1}^4 \alpha_{ijk}|M_{jk}|\le E \quad \text{for all } i \text{ and } M_{jk}=M_{kj}. \quad (4.6)$$

Problem C may be transformed into a standard linear programming problem by making the substitutions $M_{ij} = P_{ij} - N_{ij}$ and $|M_{ij}| = P_{ij} + N_{ij}$, where P_{ij} and N_{ij} are nonnegative real numbers. The theorem in the Appendix proves that this transformation gives the correct result.

Now that torque error bounds can be obtained from pseudo-inertia errors, these results must be incorporated into the trajectory planning process. In the next section, we will describe a trajectory planner which can handle the uncertainties in u_i which result from payload variations and can meet constraints on jerk as well as constraints on torque.

V. Robust Trajectory Planning Algorithm

As a solution to the RTP problem, we will develop a simple trajectory planning algorithm called the *perturbation trajectory improvement algorithm* (PTIA).

In practice, a trajectory planner must deal with a variety of arbitrary parametric curves; two representations for curves which immediately suggest themselves are splines and simple sequences of (interpolation) points. We choose to use the latter representation, i.e., the curve (2.3) is represented as an ordered sequence of points $(\lambda_{(k)}, q_{(k)})$; this proves to be the most natural representation for the application of the PTIA.

The trajectory planning process consists of assigning values of the "velocity" μ and "acceleration" $\dot{\mu}$ at each point. For the sake of simplicity, consider only those constraints which can be expressed in terms of position- and velocity-dependent bounds on the torque, i.e., ignore jerk constraints for the time being. Then all constraints can ultimately be given as λ- and μ-dependent constraints on $\dot{\mu}$, or equivalently constraints on $d\mu/d\lambda$ as shown in [2]. In terms of the (λ, μ) plot, each point is assigned a range of allowable slopes. The phase trajectories must, at every point of the phase plane, point in a direction which lies within this range.

In the discrete approximation, having the maximum and minimum slope sets limits on the differences between the values of μ at adjacent interpolation points. The process of trajectory planning requires that the initial and final points of the curve have zero velocity (or some other fixed velocity) and that the velocities at all the intermediate points be as large as possible, consistent with the slope constraint that the velocities at neighboring points not differ too much.

One approach to the solution of this problem is to try to push the speed higher at each individual point. The value of μ can be pushed higher at each point in succession until none of the velocities can be made any larger. If we call this Algorithm A, then we have the following.

Algorithm A:

A1: Set all velocities to values which are realizable (usually all zeros).

A2: Push each intermediate point of the curve as high as possible consistent with the slope constraints.

A3: If any of the velocities were changed in step A2, go back to step A2, otherwise exit.

As a practical matter, the search required to find the highest possible velocity in step A2 of Algorithm A may be fairly expensive, especially since it may be repeated many times for a single point. A simpler approach is to just try adding a particular increment to each velocity, and then make the increment smaller on successive passes of the algorithm. This gives the following.

Algorithm A′:

A1′: Set all velocities to values which are realizable (usually all zeros).

A2′: Set the current velocity increment to the robot's maximum speed.

A3′: Push each intermediate point of the curve up by an amount equal to the current velocity increment, if this is consistent with the slope constraints.

A4′: If any of the velocities were changed in Step A3', go back to step A3'.

A5′: If the current velocity increment is smaller than the desired tolerance, stop. Otherwise halve the increment and go to A3'.

Algorithm A′ is really just a combination of gradient and binary search techniques. The direction in which the curve must move (i.e., the gradient direction) is known *a priori*, since increasing the velocity always decreases the traversal time, and the amount of the change is successively halved, as in a binary search, until some desired accuracy is achieved. Clearly, this algorithm will terminate in a finite number of steps. Algorithm A′ is very simple, except possibly for the slope constraint check required in step A3'. This requires a knowledge of the dynamics and actuators characteristics of the robot. However, this check is a simple "go/no go" check, and can be isolated as a single function call. (This function will henceforth be called the *constraint function.*)

An important characteristic of the constraint function is *locality*. In the case discussed above, the constraints are expressed in terms of λ, μ, and $d\mu/d\lambda$. We need two points to determine the slope $d\mu/d\lambda$, so the constraint depends only upon two points. Therefore, when a point of the curve has its μ value changed, it is constrained only by the two adjacent points (due to the slope constraints); the rest of the curve has no influence. This allows much calculation to proceed *in parallel.* Step A3' of Algorithm A′ can be divided into two sequential steps, one which increments the odd numbered points and one which increments the even numbered ones.[1] Since the even numbered points stay the same while the odd numbered ones are being incremented, and vice versa, the points either side of the incremented points remain stationary, so that the constraint checks are valid. (If *all* points were tested simultaneously, then it is possible, for example, to increment two adjacent points; since in each case the constraint check would be made on the assumption that the other point was remaining stationary, it is possible that the new configuration would not meet the required constraints.)

It is easily seen that the process in Algorithm A′ can be extended to more complicated constraints. For example, constraints on the jerk (the derivative of the torque or acceleration) only require a more complicated constraint function, i.e., both (2.7) and (2.8). Of course, in this case the constraint function needs three points to calculate second derivatives of the speed. Thus, the constraints on a single point will be functions of *two* points either side of the point being checked, rather than one point (see [7] for more details). This affects the degree of parallelism which can be achieved; step A3' would require three passes instead of two.

As a simple illustration of how the algorithm works, consider a simple one-dimensional problem. Suppose we wish to move an object of mass m from $x = 0$ to $x = 4$. Further, suppose that there is no friction, and that there are constant bounds on the magnitude of the applied force. There will be only one parametric function f_i which may be taken to be the identity function, so that $\lambda = x$. We then have

$$F=m\frac{d^2x}{dt^2}=m\frac{d^2\lambda}{dt^2}=m\frac{d\mu}{dt}=m\frac{d\mu}{d\lambda}\frac{d\lambda}{dt}=m\mu\frac{d\mu}{d\lambda}. \tag{5.1}$$

If we consider λ-intervals of length 1, then the discrete approximation to the parameterized "curve" will have 5 points. The acceleration $\dot{\mu} = \mu d\mu/d\lambda$ can be approximated as

$$\mu\frac{d\mu}{d\lambda}\approx\mu\frac{\mu_{i+1}-\mu_i}{\lambda_{i+1}-\lambda_i}\approx\frac{\mu_{i+1}+\mu_i}{2}\cdot\frac{\mu_{i+1}-\mu_i}{\lambda_{i+1}-\lambda_i}=\frac{\mu_{i+1}^2-\mu_i^2}{2(\lambda_{i+1}-\lambda_i)}. \tag{5.2}$$

The torque constraints then become

$$F_{\max}\geq|F|=m|\dot{\mu}|=m\left|\frac{\mu_{i+1}^2-\mu_i^2}{2(\lambda_{i+1}-\lambda_i)}\right|. \tag{5.3}$$

If we use $m = 1$, $F_{\max} = 2$, and $\lambda_{i+1} - \lambda_i = 1$ for all i, this

[1] Thus, step A3' requires two passes.

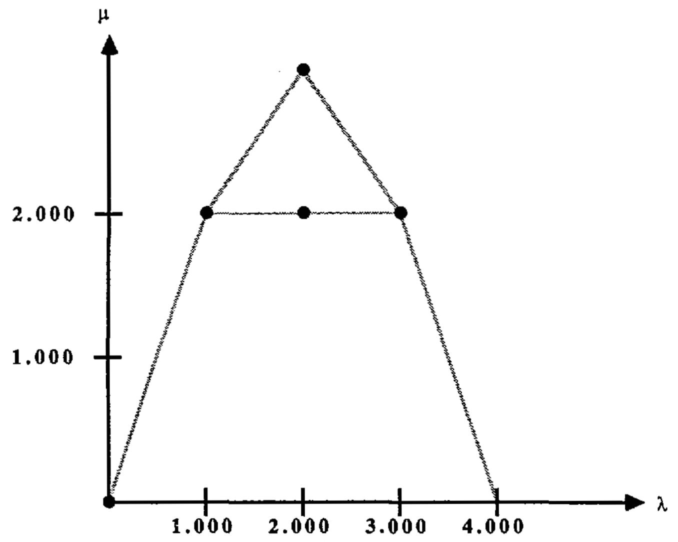


Fig. 1

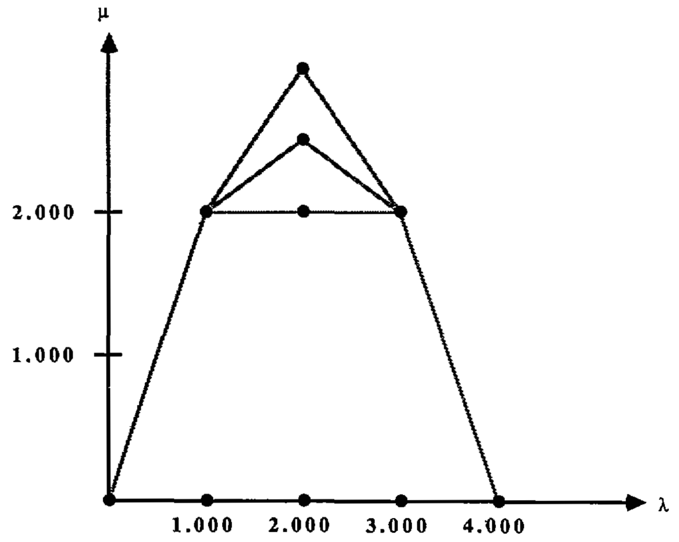


Fig. 2

reduces to

$$|\mu_{i+1}^2 - \mu_i^2| \le 4 . \tag{5.4}$$

Now consider what happens if Algorithm A is applied. We may look at the intermediate points of the curve in sequence. First, point 1 can be raised by 2, since the adjacent points have μ values of zero (i.e., $\mu_0 = \mu_2 = 0$), and $|2^2 - 0^2| = 4$. Raising the middle point, point 2, we are constrained by the fact that $\mu_3 = 0$, which limits μ_2 to 2 also. Likewise, we may change μ_3 to 2. This completes step 2 of Algorithm A. Since some of the μ values changed, we try to increase them again. This time only point 2 can be raised, giving a value of $\mu_2 = 2\sqrt{2}$. On the next pass, no μ values change, so Algorithm A terminates. It is easily verified that the solution obtained from Algorithm A is indeed the optimal solution to the discretized problem. Fig. 1 shows the discretized trajectory after passes zero, one, and two of Algorithm A.

Now look at what happens when we use Algorithm A′. Say we start with an increment of 2. Then the result of the first pass of Algorithm A′ is the same as the result of the first pass of Algorithm A, namely $\mu_1 = \mu_2 = \mu_3 = 2$. If the increment is cut to 1, then there is no change. Cutting the increment to 1/2, we may raise the middle point to 2.5. Continuing in this fashion, the middle point gets closer and closer to $2\sqrt{2}$, the correct result. Fig. 2 shows the trajectory after passes zero, one, three, and four of Algorithm A′.

The robust trajectory planner only needs to have available a test function which determines whether or not a given $(\lambda, \mu, \dot{\mu})$ triple requires excessive torque; in effect, they automatically perform the numerical search for the allowable values of $\dot{\mu}$. But such a function is easily constructed, since for a given $(\lambda, \mu, \dot{\mu})$ we can easily minimize or maximize δu_i in (3.13), and see if $u_i + \delta u_i^{\max}$ exceeds $u_i^{\max}(\lambda, \mu)$ or $u_i + \delta u_i^{\min}$ falls below $u_i^{\min}(\lambda, \mu)$. In particular, the following algorithm checks to see if a particular $(\lambda, \mu, \dot{\mu})$ triple meets all the torque constraints:

It should be noted that this function is called for *each* (λ, μ) pair; it does not, for example, reject a $(\lambda, \mu, \dot{\mu})$ triple based on an error which is computed for *all* positions or *all* velocities. As a consequence, speed is sacrificed only when absolutely necessary to guarantee that the trajectory will be realizable for all payloads within the allowable range.

The PTIA, unlike dynamic programming [3], requires relatively little memory; it requires only one floating point number per interpolation point. However, computation of the CPU time requirements is interesting.

Obviously, the computation time must increase at least linearly with the number of interpolation points on the curve, that is, the size of λ intervals. In fact, the time increases as the square of the number of interpolation points. To see why this is so, consider what happens when the number of interpolation points is doubled. Since there are twice as many points to check on each pass of the algorithm, the computation time must increase by a factor of two. Recalling that the torque constraints translate into slope constraints, it is clear that the ratio of the amount by which a μ-value may be raised to the distance between λ-values will be approximately constant. Therefore, halving the spacing of the interpolation points halves the size of the steps which can be taken in the μ direction, thus doubling the number of steps. This factor of two times the factor of two which results directly from doubling the number of points gives a factor of four increase in computation time. If doubling the number of interpolation points quadruples the computation time, then the time dependence is quadratic in the number of points, i.e., $O(N_\lambda^2)$ where N_λ is the number of the λ intervals.

It is obvious from the discussion that the fineness of the λ intervals will have a significant impact on the running time of the algorithm. It will also affect the accuracy of the results. Similarly to the convergence proof of the trajectory planning with dynamic programming in [3], we can treat the effect of the grid density on the accuracy of our solution in a quantitative manner.

```
for each joint $i$ do
  begin
    compute $u_i^N = M_i(\lambda)\dot{\mu} + Q_i(\lambda)\mu^2 + R_i(\lambda)\mu + S_i(\lambda)$
    compute $\delta u_i^{\max} = \max_{|\Delta I_N| \le E} \{\delta M_i(\lambda, \Delta I_N)\dot{\mu} + \delta Q_i(\lambda, \Delta I_N)\mu^2 + \delta S_i(\lambda, \Delta I_N)\}$
    compute $\delta u_i^{\min} = \min_{|\Delta I_N| \le E} \{\delta M_i(\lambda, \Delta I_N)\dot{\mu} + \delta Q_i(\lambda, \Delta I_N)\mu^2 + \delta S_i(\lambda, \Delta I_N)\}$
    if $u_i^N + \delta u_i^{\min} < u_i^{\min}(\lambda, \mu)$, then return REJECT
    if $u_i^N + \delta u_i^{\max} > u_i^{\max}(\lambda, \mu)$, then return REJECT
  end
return ACCEPT.
```

TABLE I

Parameter	Description	Value
τ_θ^{sat}	Saturation torque of θ motor	2.0 Nt.-M.
τ_r^{sat}	Saturation torque of r motor	0.05 Nt.-M.
τ_z^{sat}	Saturation torque of z motor	2.0 Nt.-M.
$V_\theta^{\min}$	Lower voltage limit for θ joint	-40 v.
$V_r^{\min}$	Lower voltage limit for r joint	-40 v.
$V_z^{\min}$	Lower voltage limit for z joint	-40 v.
$V_\theta^{\max}$	Upper voltage limit for θ joint	40 v.
$V_r^{\max}$	Upper voltage limit for r joint	40 v.
$V_z^{\max}$	Upper voltage limit for z joint	40 v.
k_θ^g	Gear ratio for θ drive	0.01176
k_r^g	Gear ratio for r drive	0.00318 Meters/radian
k_z^g	Gear ratio for z drive	0.00318 Meters/radian
k_θ^m	Motor constant for θ joint	0.0397 Nt.-M./amp
k_r^m	Motor constant for r joint	0.79557×10^{-3} Nt.-M./amp
k_z^m	Motor constant for z joint	0.0397 Nt.-M./amp
R_θ^m	Motor and power supply resistance, θ joint	1 Ω
R_r^m	Motor and power supply resistance, r joint	1 Ω
R_z^m	Motor and power supply resistance, z joint	1 Ω
k_θ	Friction coefficient of θ joint	8.0 Kg.-$M.^2$/sec./rad.
k_r	Friction coefficient of r joint	4.0 Kg./sec.
k_z	Friction coefficient of z joint	1.0 Kg./sec.
M_t	Mass of r joint	10.0 Kg.
M_z	Mass of z joint	40.0 Kg.
J_t	Moment of inertia around θ axis	12.3183 Kg.-$M.^2$
K	Moment of inertia offset term	3.0 Kg.-M.

VI. Numerical Examples

As an example, we will apply the methods of the previous section to the first three joints of the Bendix PACS robot arms. This arm is cylindrical in configuration and is driven by DC servos. Its dynamics and actuator characteristics are described in Table I.

The kinematics of the PACS arm are quite simple. If the coordinate frames of the base and hand are as shown in Fig. 3, then the coordinate transform T_3 is easily shown to be

$$T_3 = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & -r\sin\theta \\ \sin\theta & 0 & \cos\theta & r\cos\theta \\ 0 & -1 & 0 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.1}$$

The partial derivatives of T_3 are easily computed, and are then used to compute the δM_i. If we let $H = \Delta I_N$ and define m_{ij} = Tr $(\partial T_N/\partial q^j H \partial T_N^T/\partial q^i)$, then we have $\delta M_i = m_{ij} dq^j/dL$. The nonzero m_{ij} are then found to be

$$m_{zz} = \mathrm{Tr}\left(\frac{\partial T_3}{\partial z} H \frac{\partial T_3^T}{\partial z}\right) = H_{44}$$

$$m_{\theta\theta} = \mathrm{Tr}\left(\frac{\partial T_3}{\partial \theta} H \frac{\partial T_3^T}{\partial \theta}\right) = H_{11} + H_{33} + 2rH_{34} + r^2 H_{44}$$

$$m_{\theta r} = m_{r\theta} = \mathrm{Tr}\left(\frac{\partial T_3}{\partial \theta} H \frac{\partial T_3^T}{\partial r}\right) = H_{14}$$

$$m_{rr} = \mathrm{Tr}\left(\frac{\partial T_3}{\partial r} H \frac{\partial T_3^T}{\partial r}\right) = H_{44}.$$

We can now find the δM_i. We have

$$\delta M_z = H_{44} \frac{dz}{d\lambda}$$

$$\delta M_\theta = (H_{11} + H_{33} + 2rH_{34} + r^2 H_{44}) \frac{d\theta}{d\lambda} + H_{14} \frac{dr}{d\lambda}$$

$$\delta M_r = H_{14} \frac{d\theta}{d\lambda} + H_{44} \frac{dr}{d\lambda}.$$

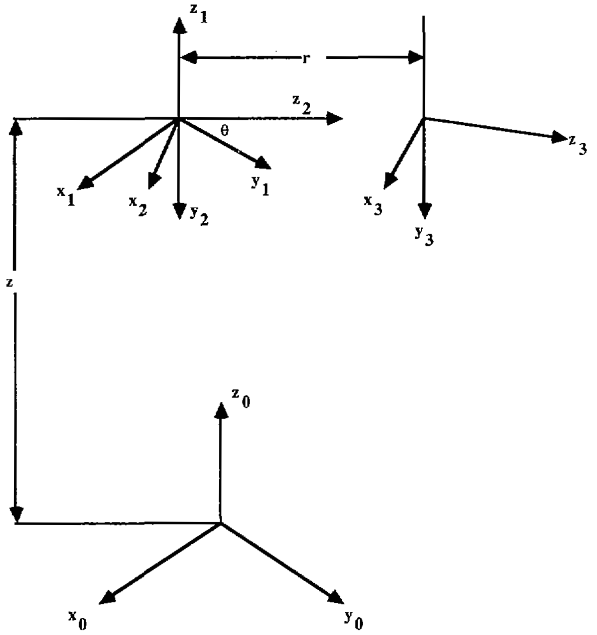


Fig. 3

Similarly, if we define c_{ijk} by c_{ijk} = Tr $(\partial^2 T_N/\partial q^j \partial q^k H \partial T_N^T/\partial q^i)$, then we have

$$\delta Q_i = m_{ij} \frac{d^2 f^j}{d\lambda^2} + c_{ijk} \frac{df^j}{d\lambda} \frac{df^k}{d\lambda}.$$

We now compute the c_{ijk}. Only six cases need to be considered, since all but two of the second partials of T_3 are zero. Of these, only two of the c_{ijk} are nonzero

$$c_{r\theta\theta} = \mathrm{Tr}\left(\frac{\partial^2 T_3}{\partial \theta^2} H \frac{\partial T_3}{\partial r}\right) = -H_{34} - H_{44} r$$

$$c_{\theta\theta r} = \mathrm{Tr}\left(\frac{\partial^2 T_3}{\partial \theta \partial r} H \frac{\partial T_3}{\partial \theta}\right) = H_{34} + H_{44} r.$$

Calculating the δQ_i,

$$\delta Q_z = H_{44} \frac{d^2 z}{d\lambda^2}$$

$$\delta Q_\theta = (H_{11} + H_{33} + 2rH_{34} + r^2 H_{44}) \frac{d^2\theta}{d\lambda^2} + H_{14} \frac{d^2 r}{d\lambda^2} + 2(H_{34} + rH_{44}) \frac{dr}{d\lambda} \frac{d\theta}{d\lambda}$$

$$\delta Q_r = H_{14} \frac{d^2\theta}{d\lambda^2} + H_{44} \frac{d^2 r}{d\lambda^2} - (H_{34} + rH_{44}) \left(\frac{d\theta}{d\lambda}\right)^2.$$

The δS_i, the gravitational error coefficients, are easily found to be $\delta S_z = H_{44} g$ and $\delta S_\theta = \delta S_r = 0$, where g is the acceleration due to gravity.

We will use the norm $\|H\| = \Sigma_{i=1}^4 \Sigma_{j=1}^4 \alpha_{ij} |H_{ij}|$, where $\alpha_{ij} > 0$. This makes the problem of finding the error bounds very simple. It can be shown that if the functional to be maximized is $Z = \Sigma_i \Sigma_j \beta_{ij} H_{ij}$, then the maximum over H for $\|H\| \le E$ occurs when all the H_{ij} are zero except for those values of i and j for which $|\beta_{ij}/\alpha_{ij}|$ is a maximum; this number times E is also the maximum value of Z. If we use

$$\alpha_{ij} = \begin{cases} 1 & i = j \\ 1/2 & i \ne j \end{cases}$$

then the resulting bounds on the $\|\delta u_i|$ are

$$|\delta u_z| \le \left|\frac{dz}{d\lambda}\dot{\mu}+\frac{d^2z}{d\lambda^2}\mu^2+g\right| \|\Delta I_N\|$$

$$|\delta u_\theta| \le \max\left\{\left|\frac{d\theta}{d\lambda}\dot{\mu}+\frac{d^2\theta}{d\lambda^2}\mu^2\right|, \left|\frac{dr}{d\lambda}\dot{\mu}+\frac{d^2r}{d\lambda^2}\mu^2\right|,\right.$$

$$\left|2r\frac{d\theta}{d\lambda}\dot{\mu}+2r\frac{d^2\theta}{d\lambda^2}\mu^2+2\frac{dr}{d\lambda}\frac{d\theta}{d\lambda}\mu^2\right|,$$

$$\left.\left|r^2\frac{d\theta}{d\lambda}\dot{\mu}+r^2\frac{d^2\theta}{d\lambda^2}\mu^2+2r\frac{dr}{d\lambda}\frac{d\theta}{d\lambda}\mu^2\right|\right\} \|\Delta I_N\|$$

$$|\delta u_r| \le \max\left\{\left|\frac{d\theta}{d\lambda}\dot{\mu}+\frac{d^2\theta}{d\lambda^2}\mu^2\right|, \left|\left(\frac{d\theta}{d\lambda}\right)^2\mu^2\right|,\right.$$

$$\left.\left|\frac{dr}{d\lambda}\dot{\mu}+\frac{d^2r}{d\lambda^2}\mu^2-r\left(\frac{d\theta}{d\lambda}\right)^2\mu^2\right|\right\} \|\Delta I_N\| .$$

The joint torques that can be applied to the PACS arm are limited by saturation of the drive motors, which give a constant torque or force limit for each joint. In addition, there are limits on the voltages which can be applied to the motors, so we need to know how the errors in the joint torques translate into errors in the motor voltages. It will be assumed that the back-EMF constant, winding resistance, and voltage source resistance are known exactly, although this is not necessary. Since for a given speed voltage is a linear function of torque, i.e., $V_i = A_i u_i + B_i$, the change in voltage will be $\delta V_i = A_i \delta u_i$. These changes in voltage can then be added to the nominal voltage and tested against the motor voltage limits in much the same way that the torques are checked against the motor torque saturation limits.

The perturbation to the nominal dynamics of the manipulator will be caused by placing a cube with edges of length L and uniform mass density ρ in the gripper of the robot, with its center of mass coincident with the origin of the end effector coordinate system. The pseudo-inertia of this cube is

$$\Delta I_3 = \begin{bmatrix} \frac{1}{12}\rho L^5 & 0 & 0 & 0 \\ 0 & \frac{1}{12}\rho L^5 & 0 & 0 \\ 0 & 0 & \frac{1}{12}\rho L^5 & 0 \\ 0 & 0 & 0 & \rho L^3 \end{bmatrix} .$$

The norm of this "error" is $\|\Delta I_3\| = \rho L^3 + 1/4\ \rho L^5$. The maximum torque error for a given range of pseudo-inertia errors occurs when the error bound E is precisely equal to the norm of the actual pseudo-inertia error. Therefore, the most stringent test of the results of the previous section is to use a tight error bound, i.e., $E = \|\Delta I_3\|$. This has been done for a cube with sides of 5 cm and densities of 0, 6, 12, 18, 24, and 30 g/cc. The path traversed is a straight line from the (Cartesian) point (0.7, 0.7, 0.1) to (0.4,–0.4, 0.4). For comparison, the true optimal solution has been calculated, using the actual dynamics (including the effects of the cube in the gripper). The results are summarized in Tables II–VI. Table II gives traversal time for the true optimal solution and for the case in which errors are included. The "percent difference" column gives the percentage by which the true optimal traversal time is exceeded. Tables III and IV give minimum and maximum voltages, respectively. The actual and nominal values are both computed for the "nomial" trajectory, i.e., the trajectory which is calculated with errors included. The

TABLE II

Density	Time (Seconds)		Percent Difference
	Nominal	Optimal	
0	1.789	1.789	0
6	1.934	1.844	4.9%
12	2.076	1.898	9.4%
18	2.213	1.950	13.5%
24	2.340	2.002	16.9%
30	2.459	2.054	19.7%

TABLE III

Minimum Voltages						
Density	z joint		θ joint		r joint	
	Nominal	Actual	Nominal	Actual	Nominal	Actual
0	29.86	29.86	-39.93	-39.93	-40.00	-40.00
6	30.34	30.91	-38.26	-38.52	-37.94	-40.00
12	30.54	31.67	-33.05	-33.51	-36.12	-39.98
18	30.67	32.39	-24.92	-25.37	-34.47	-39.99
24	30.78	33.07	-19.94	-20.36	-33.01	-40.00
30	30.86	33.73	-16.80	-17.18	-31.70	-39.96

TABLE IV

Maximum Voltages						
Density	z joint		θ joint		r joint	
	Nominal	Actual	Nominal	Actual	Nominal	Actual
0	37.64	37.64	39.86	39.86	40.00	40.00
6	37.40	38.03	37.14	37.73	32.62	32.60
12	36.79	38.04	17.64	18.25	27.01	27.48
18	35.78	37.61	10.61	11.21	23.79	24.94
24	35.12	37.54	7.30	7.89	21.39	23.41
30	34.69	37.69	5.39	5.98	19.51	22.39

TABLE V

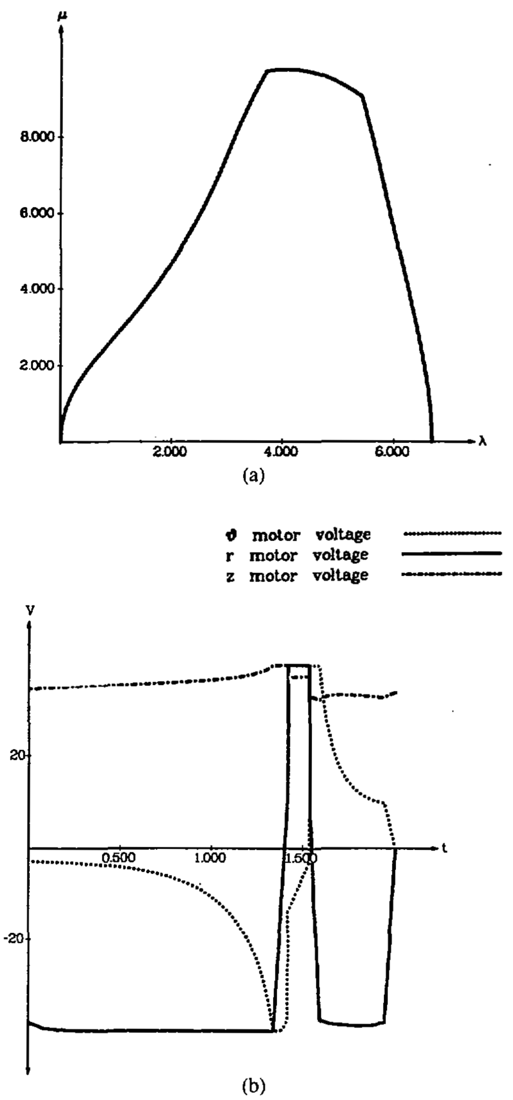
Minimum Torques/Forces						
Density	z joint (Newtons)		θ joint (Newton-Meters)		r joint (Newtons)	
	Limit = -629 Nt.		Limit = -170 Nt.-M.		Limit = -15.7 Nt.	
	Nominal	Actual	Nominal	Actual	Nominal	Actual
0	333.52	333.52	-112.29	-112.29	-9.99	-9.99
6	335.91	342.20	-104.78	-105.71	-9.47	-9.99
12	363.02	376.62	-87.70	-89.26	-9.02	-9.99
18	373.03	394.00	-62.91	-64.43	-8.61	-9.99
24	377.84	406.17	-48.15	-49.55	-8.24	-9.99
30	380.11	415.72	-39.14	-40.43	-7.92	-9.98

TABLE VI

Maximum Torques/forces						
Density	z joint (Newtons)		θ joint (Newton-Meters)		r joint (Newtons)	
	Limit = 629 Nt.		Limit = 170 Nt.-M.		Limit = 15.7 Nt.	
	Nominal	Actual	Nominal	Actual	Nominal	Actual
0	421.31	421.31	161.72	161.72	10.03	10.03
6	418.68	426.52	150.39	152.36	8.15	8.18
12	414.50	430.04	78.43	80.48	6.76	6.89
18	406.68	429.55	51.39	53.41	5.96	6.25
24	402.17	432.31	38.18	40.19	5.36	5.87
30	400.42	437.96	30.20	32.20	4.89	5.61

actual voltages are those required to move the robot with the cube in the gripper, while the nominal values are those which are required without the cube, i.e., with the nominal payload. The minimum and maximum voltages available are -40 and 40 V, and it is easily seen that these limits are not exceeded for any joint or for either payload. Tables V and VI give the minimum and maximum torques or forces for each joint. The torque or force limits are given at the head of the column for the appropriate joint; again, the limits are not exceeded.

The phase plane (λ versus μ) plot and motor voltage versus time plot for the zero-density case are shown in Fig. 4(a) and (b). Since the error is zero in this case, the results are exact. For a density of 12 g/cc, the optimal and nominal (i.e., with errors included) phase plane plots are shown in Fig. 5(a). (The optimal phase trajectory is the minimum-time phase trajectory which would be obtained if we knew the exact payload characteristics.) Fig. 5(b) gives joint positions versus time; z and r are in meters, θ in radians. Fig. 5(c) through (e) gives nominal and actual motor



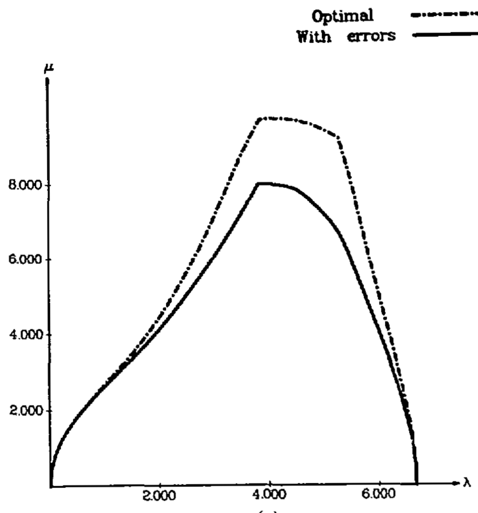
(a)

(b)
Fig. 4

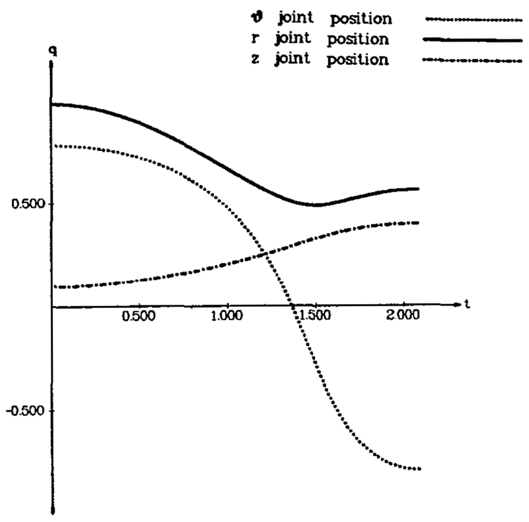
voltages required to drive the robot along the nominal trajectory for the z, θ, and r joints, respectively. The nominal voltages are those which would be required if the actual payload were identical to the nominal payload. The actual torques are the torques required to keep the robot with the perturbed payload on the nominal trajectory. The torque plots look very much like the voltage plots, and are therefore omitted. The same plots for various payload densities can be found in [7].

It was noted above that none of the joint torque or voltage constraints was violated. However, the minimum voltage for the r joint at one point meets the lower voltage limit. This indicates that the trajectory which is generated when payload errors are included is indeed the fastest possible trajectory for the given range of possible payloads; for this particular point, the worst-case payload happens to have the same characteristics as the actual payload. A larger payload would have resulted in violation of a voltage constraint.

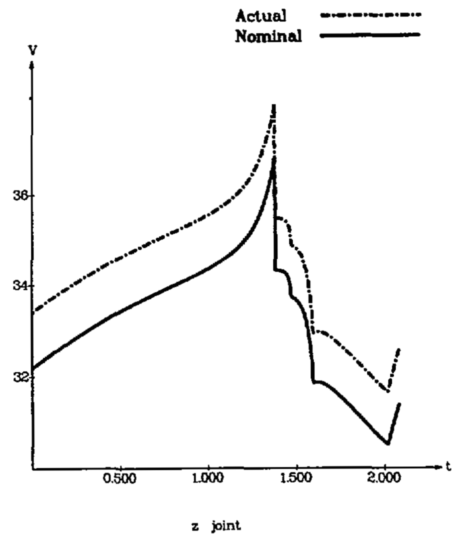
Another point to consider is the relationship between the nominal and optimal phase trajectories. It is expected that the nominal phase trajectory will be lower than the optimal trajectory; a nominal trajectory which was higher than the optimal one would lead to a contradiction of the optimality of the optimal trajectory. Also, the difference between the optimal and nominal trajectories increases as the payload error bound increases. This would be expected, since the nominal trajectory must accommodate *all* payloads within a given range; as the range of payloads increases,


Optimal
With errors
μ
8.000
6.000
4.000
2.000
2.000
4.000
6.000
λ

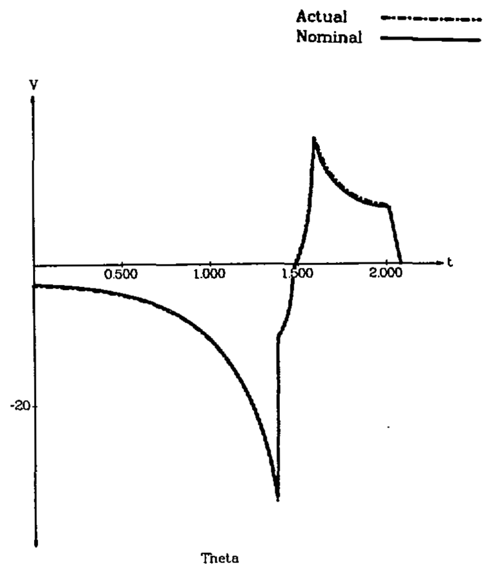
(a)


ϑ joint position
r joint position
z joint position
q
0.500
-0.500
0.500
1.000
1.500
2.000
t

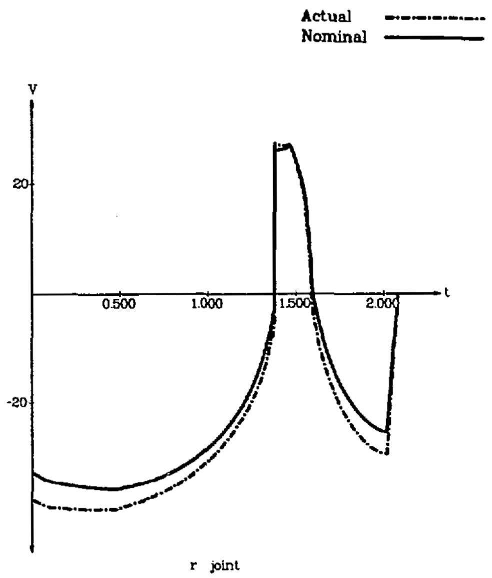
(b)


Actual
Nominal
V
36
34
32
0.500
1.000
1.500
2.000
t
z joint

(c)


Actual
Nominal
V
-20
0.500
1.000
1.500
2.000
t
Theta

(d)


Actual
Nominal
V
20
-20
0.500
1.000
1.500
2.000
t
r joint

(e)

Fig. 5

the worst-case errors also increase, resulting in more restrictive limits on the nominal torques, and hence longer trajectory traversal times.

VII. Conclusions

A method for including payload inertia errors in the manipulator trajectory planning process has been presented. Errors in the payload inertia are characterized by bounds on the norm of the difference between the actual and nominal pseudo-inertias of the payload. Given such a bound, it has been shown that a trajectory can be constructed which meets all torque and force constraints for *all* actual payloads, provided that the norm of the difference of the pseudo-inertias of the actual and nominal payloads differs by less than the given error bound. The method may also be extended to accommodate jerk constraints. This technique was applied to the Bendix PACS robot for a number of different payloads, and the resulting trajectories were shown not to violate any joint torque or motor voltage constraints. In the worst case, in which the actual payload mass differs from the nominal mass by approximately one third of the robot's rated maximum load, the traversal time was less than 20 percent over the optimal value.

Appendix

Lemma: If $\alpha_{ijk} \geq 0$ for all i, j, and k, and if for every pair of indexes (j, k) there is an i such that $\alpha_{ijk} \neq 0$, then $F(M)$ is a norm.

Proof: In order to prove that F is a norm, we must show that

1) $F(M) \geq 0$ for all M,
2) $F(M) = 0$ iff $M = 0$, 3) $F(\gamma M) = |\gamma| F(M)$ for all scalars γ and all matrices M, and
4) $F(X + Y) \leq F(X) + F(Y)$ for all matrices X and Y.

Obviously 1) is true, since F is the maximum of a set of nonnegative quantities. $F(0) = 0$, proving the "if" part of 2). To prove the "only if" part, observe that if M is nonzero, then it has some nonzero element M_{jk}. For this particular jk pair, there is some i such that α_{ijk} is nonzero, so that $\sigma_i > 0$ for this i. Therefore, $F > 0$. 3) is true since

$$\sigma_i(\gamma M) = \sum_{j=1}^{4} \sum_{k=1}^{4} \alpha_{ijk} |\gamma M_{jk}| = |\gamma| \sum_{j=1}^{4} \sum_{k=1}^{4} \alpha_{ijk} |M_{jk}| = |\gamma| \sigma_i(M)$$

and hence

$$F(\gamma M) = \max_i \sigma_i(\gamma M) = \max_i |\gamma| \sigma_i(M)$$
$$= |\gamma| \max_i \sigma_i(M) = |\gamma| F(M) .$$

Finally,

$$\sigma_i(X+Y) = \sum_j \sum_k \alpha_{ijk} |X_{jk} + Y_{jk}|$$
$$\leq \sum_j \sum_k \alpha_{ijk} |X_{jk}| + \sum_j \sum_k \alpha_{ijk} |Y_{jk}| = \sigma_i(X) + \sigma_i(Y)$$

so that

$$F(X+Y) = \max_i \{\sigma_i(X+Y)\} \leq \max_i \{\sigma_i(X) + \sigma_i(Y)\}$$
$$\leq \max_i \{\sigma_i(X)\} + \max_i \{\sigma_i(Y)\} = F(X) + F(Y). \quad ■$$

To prove that the transformation of Problem C into a linear programming problem gives the correct result, first eliminate the symmetry constraint, giving the following.

Problem C':

$$\text{maximize } W(M) = \sum_{j=1}^{4} \sum_{k=j}^{4} \beta'_{jk} M_{jk}$$

$$\text{subject to } \sum_{j=1}^{4} \sum_{k=j}^{4} \alpha'_{ijk} |M_{ijk}| \leq E$$

where

$$\alpha'_{ijk} = \begin{cases} \alpha_{ijk} & j=k \\ \alpha_{ijk} + \alpha_{ikj} & j \neq k \end{cases}$$

and

$$\beta'_{ij} = \begin{cases} \beta_{ij} & i=j \\ \beta_{ij} + \beta_{ji} & i \neq j \end{cases} .$$

Then we have the following theorem.

Theorem: Let Problem D be defined as

$$\text{maximize } Z(P, N) = \sum_{j=1}^{4} \sum_{k=j}^{4} \beta'_{jk} (P_{jk} - N_{jk})$$

$$\text{subject to } \sum_{j=1}^{4} \sum_{k=j}^{4} \alpha'_{ijk} (P_{jk} + N_{jk}) \leq E \text{ and } P_{ij} \geq 0,\ N_{ij} \geq 0 .$$

Then the optimal W from Problem C' is equal to the optimal Z from Problem D.

Proof: Let M be a solution of Problem C', and let $W^* = W(M)$. If we make the substitutions

$$P_{jk} = \begin{cases} M^*_{jk} & M^*_{jk} \geq 0 \\ 0 & M^*_{jk} < 0 \end{cases}$$

and

$$N_{jk} = \begin{cases} 0 & M^*_{jk} \geq 0 \\ -M^*_{jk} & M^*_{jk} < 0 \end{cases}$$

then we have $M^*_{ij} = P_{ij} - N_{ij}$, and $|M^*_{ij}| = P_{ij} + N_{ij}$. Making these substitutions in Problem C' gives

$$W^* = W(M^*) = \sum_{j=1}^{4} \sum_{k=j}^{4} \beta'_{jk} (P_{jk} - N_{jk}) = Z(P, N)$$

$$\sum_{j=1}^{4} \sum_{k=j}^{4} \alpha'_{ijk} (P_{jk} + N_{jk}) \leq E \text{ and } P_{ij} \geq 0,\ N_{ij} \geq 0 .$$

The conditions for Problem D are satisfied, so we must have $W^* \leq Z^*$, where Z^* is the optimal value of Z obtained from Problem D.

Likewise, let P^* and N^* be an optimal solution to Problem D. Then for every pair of indexes (j, k) we have $\beta'_{jk} > 0$, $\beta'_{jk} = 0$, or $\beta'_{jk} < 0$, then we must have $N^*_{jk} = 0$. Otherwise, we could substitute $P^*_{jk} + N^*_{jk}$ for P_{jk} and 0 for N_{jk}; these new values still satisfy the required constraints, but increase the objective function, contradicting the fact that (P^*, N^*) is optimal. Similarly, if $\beta'_{jk} < 0$, then we must have $P^*_{jk} = 0$. If $\beta'_{jk} = 0$, then we may take $P^*_{jk} = N^*_{jk} = 0$, since this leaves the constraints satisfied and has no effect on the objective function. Therefore, we always have either $\boldsymbol{P}^*_{jk} = 0$ or $\boldsymbol{N}^*_{jk} = 0$. Taking $\boldsymbol{M}_{ij} = \boldsymbol{P}^*_{ij} - \boldsymbol{N}^*_{ij}$, it follows that $|\boldsymbol{M}_{ij}| = \boldsymbol{P}^*_{ij} + \boldsymbol{N}^*_{ij}$. Making these substitutions in Problem D,

$$Z^* = Z(\boldsymbol{P}^*, \boldsymbol{N}^*) = \sum_{j=1}^{4} \sum_{k=j}^{4} \beta'_{jk} \boldsymbol{M}_{jk} \leq W^*$$

$$\sum_{j=1}^{4} \sum_{k=j}^{4} \alpha'_{ijk} |\boldsymbol{M}_{jk}| \leq E\ .$$

Therefore $W^* \leq Z^* \leq W^*$, proving the theorem. ■

References

[1] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "On the optimal control of robotic manipulators with actuator constraints," in *Proc. 1983 Automat. Contr. Conf.*, June 1983, pp. 782-787.

[2] K. G. Shin and N. D. McKay, "Minimum-time control of a robotic manipulator with geometric path constraints," *IEEE Trans. Automat. Contr.*, vol. AC-30, pp. 531-541, June 1985.

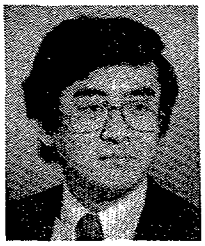
[3] K. G. Shin and N. D. McKay, "Robot path planning using dynamic programming," in *Proc. 23rd CDC*, Dec. 1984, pp. 1629-1635; also in *IEEE Trans. Automat. Contr.*, vol. AC-31, pp. 491-500, June 1986.

[4] S. Dubowsky and D. T. DesForges, "The application of model-referenced adaptive control to robotic manipulators," *ASME J. Dynam. Syst., Measurement, Contr.*, vol. 101, pp. 193-200, Sept. 1979.

[5] A. J. Koivo and T. H. Guo, "Adaptive linear controller for robotic manipulators," *IEEE Trans. Automat. Contr.*, vol. AC-28, pp. 162-170, Feb. 1983.

[6] R. P. C. Paul, *Robot Manipulators: Mathematics, Programming, and Control.* Cambridge, MA: M.I.T. Press, 1981.

[7] N. D. McKay, "Minimum-cost control of robotic manipulators with geometric path constraints," Ph.D. dissertation, The Univ. Michigan, Sept. 1985.



Kang G. Shin (S'75-M'78-SM'83) received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1970, and both the M.S. and Ph.D. degrees in electrical engineering from Cornell University, Ithaca, NY, in 1976 and 1978, respectively.

From 1970 to 1972 he served in the Korean Army as an ROTC Officer and from 1972 to 1974 he was on the Research Staff of the Korea Institute of Science and Technology, Seoul, Korea, working on the design of VHF/UHF communication systems. From 1978 to 1982 he was an Assistant Professor at Rensselaer Polytechnic Institute, Troy, NY. He was also a Visiting Scientist at the U.S. Air Force Flight Dynamics Laboratory in Summer 1979 and at Bell Laboratories, Holmdel, NJ, in Summer 1980. At present he is a Professor in the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, which he joined in 1982. He has been very active and authored/coauthored over 120 technical papers in the areas of fault-tolerant real-time computing, computer architecture, and robotics and automation. In 1986 he founded the Real-Time Computing Laboratory, where he and his students are currently building a 19-node hexagonal mesh multiprocessor to validate various architectures and analytic results in the area of distributed real-time computing.

Dr. Shin was the Program Chairman of the 1986 IEEE Real-Time Systems Symposium (RTSS), the General Chairman of the 1987 RTSS, and the Guest Editor of the August 1987 Special Issue on Real-Time Systems of the IEEE TRANSACTIONS ON COMPUTERS. He is a member of the ACM, Sigma Xi, and Phi Kappa Phi.

Neil D. McKay (M'86) was born in Albany, NY, on May 5, 1958. He received the B.S. degree in electrical engineering from the University of Rochester, Rochester, NY, in 1980, the M.S. degree in 1982 from Rensselaer Polytechnic Institute, Troy, NY, and the Ph.D. degree from the University of Michigan, Ann Arbor, in 1985.

Currently, he is with the Computer Science Department at General Motors Research Laboratories, Warren, MI.