SAND95-2433C
CONF-960662--1

A Fast and Robust Algorithm for General Inequality/Equality Constrained Minimum Time Problems

B. Driessen* and N. Sadegh

School of Mechanical Engineering
Georgia Institute of Technology, Atlanta, GA 30332

Abstract

This paper presents a new algorithm for solving general inequality/equality constrained minimum time problems. The algorithm's solution time is linear in the number of Runge-Kutta steps and the number of parameters used to discretize the control input history.

The method is being applied to a three link redundant robotic arm with torque bounds, joint angle bounds, and a specified tip path. It solves case after case within a graphical user interface in which the user chooses the initial joint angles and the tip path with a mouse. Solve times are from 30 to 120 seconds on a hewlett packard workstation. A zero torque history is *always* used in the initial guess, and the algorithm has never crashed, indicating its robustness.

The algorithm solves for a feasible solution for large trajectory execution time t_f and then reduces t_f by a small amount and re-solves. The fixed time re-solve uses a new method of finding a near-minimum-2-norm solution to a set of linear equations and inequalities that achieves quadratic convergence to a feasible solution of the full nonlinear problem.

1. Introduction

One of the primary attributes of an intelligent control system is its ability to automatically generate the desired state and/or output trajectory that the plant or plants under its control must follow. The determination of the desired trajectory is often performed in order to minimize a certain performance index while satisfying the required state and/or input constraints. A frequently used performance index in many motion control applications is the "time" required to perform a certain task.

Most of the work in minimum-time trajectory planning has been for rigid, non-redundant, completely actuated manipulators, where non-redundant means the end effector path completely determines the path in joint space (to within a finite number of possibilities). Exemplary is the work of Bobrow [2] and Shin and McKay [13, 12, 16, 14]. Here, the trajectory of the system along this path can be completely specified using a single path parameter "s(t)." Bobrow [2] first constructed an infeasible region in the state space (s, $\dot{s}$) (regions for which the appropriate input torques for keeping the system on the path are not available) and then utilized a trial and error procedure for finding switching curves in the state space. Shin and McKay extended Bobrow's work, and later removed trial and error from the method by doing dynamic programming while utilizing a discretization of the state space (s, $\dot{s}$) [14]. Since there are only two state variables, Bellman's curse of dimensionality (see Larson and Casti [9]) was not too burdensome computationally speaking.

Clearly the above approach will not be useful for a redundant robotic arm.

There has been some work for the case where there are no path constraints and only the terminal state is specified. This scenario allows for the use of only switching times as parameters if the state equation of the dynamic system is linear in the inputs and the inputs have explicit upper and lower bounds. (This can be proved with the Pontryagin Minimum Principle {see Ho and Bryson [8] and Lewis [10]}.) Exemplary is the work of Meier and Bryson [11] and Byers and Vadali [4].

However the method of using switching times first requires a very good initial guess, and one must know not only where to guess the switching times but also how many there are, and what to switch between, i.e., which vertices of a hyper cube or polytope in input space. Such guesses require the use of a discretized input history as part of the optimization variables before switching times can be used exclusively as optimization variables.

Wie et al. [16] solved a minimum time pointing control problem using the necessary conditions that can be found in (Ho and Bryson [8], Lewis [10], etc.). The optimization parameters used were the initial values of the costate variables. These researchers noted the fact that this approach requires an extremely good initial guess for the costate variables.

Geering et. al [7] is yet another example of attempts to use the shooting method, and they reported that they were unable to obtain solutions for an IBM robot using the shooting method alone.

On the other hand, our method, applied to the three-link-arm path constrained problem described in the Abstract and Section 3.2.8, solves case after case, each within 30 to 120 seconds, within a graphical user interface in which the user selects the tip path with a mouse, and the initial guess *always* has a zero torque history.

An interesting contrast to the above methods was presented by Zimmerman and Layton [18]. To generate a

*This work was completed in part at Sandia National Laboratories, Albuquerque, NM

This work was supported by the United States Department of Energy under Contract DE-AC04-94AL85000.

sub-optimal minimum time trajectory, they used a genetic algorithm with the actual system in the optimization loop. Since they were dealing with a system with noisy sensors (a flexible system with strain gauge sensors), a gradient based approach presented difficulty. They reportedly achieved good results. Unfortunately, this approach does not easily handle general constraints.

Byers et al. [3] investigated near minimum time closed loop slewing maneuvers of flexible spacecraft. These researchers first neglected the flexibility of the craft to obtain the switching (min/max) input history for a rigid model of the craft and smoothed the history out by replacing the sign function by a hyperbolic tangent approximation. Their work was based on the observation that such an approach yielded a final state that was very close to the desired final state for large angle maneuvers of craft with fairly small flexibility. Then more standard control methods were used to remove the residual error in the final state.

General robotic arm minimum time problems are historically known to be very difficult and historically require good initial guesses. This is due to the highly nonlinear nature of the equations of motion of the arm, the parameters and algorithms used in past research, and the difficult character of the minimum time problem itself.

Yet our algorithm is solving the above mentioned three link arm problem in times comparable to the solve times for much simpler problems solved with the Order(N) method of Wright [16], even though our problem is much more difficult because it is a highly nonlinear problem, with many active inequality constraints at the solution, typical of minimum time problems.

2. Problem Statement and Formulation

Given a nonlinear time invariant dynamic system whose state equation is:

$$\dot{\bar{x}} = \tilde{f}(\bar{x},\bar{u}) \qquad (1)$$

where $\tilde{f}$ is typically linear in $\bar{u}$ and highly nonlinear in $\bar{x}$ for mechanical problems, and with initial condition:

$$\bar{x}(0) = \bar{x}_0 \qquad (2)$$

and equality path constraints:

$$\tilde{g}(\bar{x},\bar{u}) = 0 \qquad (3)$$

and inequality path constraints:

$$\tilde{h}(\bar{x},\bar{u}) \le 0 \qquad (4)$$

and terminal equality constraints:

$$\tilde{G}\left(\bar{x}(t_f),\bar{u}(t_f)\right) = 0 \qquad (5)$$

the objective is to find $\bar{u}(t)$ (and $\bar{x}(t)$) that satisfies constraints (1) through (5), and minimizes the total trajectory execution time t_f.

The terminal equality constraints (5) must be such that $\left(\bar{x}(t_f),\bar{u}(t_f)\right)$ is a feasible equilibrium point of the dynamic system. This will allow us to argue that if the system cannot satisfy all constraints with $t_f = t_f^0$, then it cannot satisfy them for $t < t_f^0$ (because otherwise the system could just stay at its equilibrium point until $t = t_f^0$ and hence satisfy all constraints with $t_f = t_f^0$--a contradiction). This type of terminal constraint then allows us to declare a problem infeasible, for all practical purposes, if it cannot be solved with large t_f.

3. Algorithm Description

By applying a Runge-Kutta formula to Eqn. (1), we obtain:

$$\bar{x}_{k+1} = f(\bar{x}_k,\bar{u}_k),\ (k=1,\ldots,N+1) \qquad (6)$$

and equations (2) through (5) become, respectively:

$$\bar{x}_1 = \bar{x}(0) \qquad (7)$$

$$g(\bar{x}_k,\bar{u}_k) = 0, (k=1,\ldots,N) \qquad (8)$$

$$h(\bar{x}_k,\bar{u}_k) \le 0,\ (k=1,\ldots,N+1) \qquad (9)$$

$$G(\bar{x}_{N+1},\bar{u}_{N+1}) = 0 \qquad (10)$$

where the (N+1)th path equality constraint in Eqn. (8) is taken care of by the set of terminal constraints in Eqn. (10). For example, in the three link arm problem, the terminal position of the tip must be at the endpoint of the tip path; this automatically ensures that the tip is on the path at $t = t_f$. The other terminal constraints for the three link arm problem are that the angular velocities must be zero.

3.1. Overview of the Algorithm

Now that the problem has been discretized, it is approachable with math programming techniques. In the following description, comments in curly braces indicate values used for solving the three-link arm problem described in the **Abstract** above and section **3.2.8. Numerical Results.** Our method starts with a large fixed final time $\{t_f = 30\,\text{sec}$ for the three-link arm$\}$, and solves for a feasible solution. It then reduces the final time by a small amount {5 sec.'s} and tries to re-solve for a feasible solution, given the last torque history as an initial guess, and declaring success if all constraints (in Eqn.'s (8), (9), and (10)) are satisfied to within a tolerance $\{10^{-4}\}$. If it takes too many iterations {10}, i.e., forward integrations, to re-solve, it reduces Δt_f by a factor of two and tries again to re-solve. If a successful re-solve occurs, it doubles Δt_f for the next final time reduction.

3.2. Method of Solving for a Feasible Solution for Fixed Final Time

We will let

$$U \equiv \{\overline{u}_1^T \quad \overline{u}_2^T \quad \ldots \quad \overline{u}_{N+1}^T\}^T \tag{11}$$

and let a change in U be denoted by:

$$\Delta U = \{u_1^T \quad u_2^T \quad \ldots \quad u_{N+1}^T\}^T \tag{12}$$

(thus defining u_k, $(k = 1,\ldots,N+1)$).

The method used is one whose solution time is Order(N) and that converges quadratically to a feasible solution by finding a change ΔU in the torque history that is feasible with respect to a linear approximation of the problem about the current trajectory and where ΔU is close to the minimum 2-norm solution of the corresponding set of linear equations and linear inequalities.

The method to obtain this ΔU is described in the sections **3.2.1. Overview of the Method for Obtaining the Quadratically Convergent ΔU** and **3.2.5. Quadratically Convergent ΔU Details (Newton Rhapson Method).**

Since the Newton Rhapson method can sometimes fail to produce quick convergence due to ill-conditioning of the jacobian matrices or large distance from a feasible solution, a backup method for reducing the total constraint violation is used. This method of obtaining this backup ΔU is described in the section **3.2.6. Convergence Assuring ΔU Details.**

In particular, if the banded matrix solver indicates singularity in the process of trying to reduce the total (linear) constraint violation within the linear subproblem, the backup method for obtaining ΔU takes over, and either begins where the first method left off if the first method obtained improvement, or starts over with ΔU=0, otherwise. Also, if the first method takes more than a certain number of iterations {5} without producing improvement within the linear subproblem, the second method begins (with $\Delta U = 0$). This method is also given a maximum number of iterations, except that it must make at least one improving step within the subproblem before ΔU is used on the full problem.

(It is common practice in math programming to not take the subproblem to its entire completion before trying ΔU on the full nonlinear problem, as doing so would often be wasteful of CPU time.)

Once a ΔU is found, regardless of which method it came from, a halve/double step-size selection is used as follows:

$$U_{new} = U_{old} + k_r^{ext}\Delta U,\ k_r^{ext} \in [0,1] \tag{13}$$

The value of k_r^{ext} is set to 1 at the beginning of a fixed-final-time solve; after that, its value is determined by the following rules:

Every time a ΔU fails to produce improvement in the merit function, k_r^{ext} is halved (at which point $\frac{1}{2}\Delta U$ is tried), and every time a $k_r^{ext}\Delta U$ produces improvement, k_r^{ext} is doubled for the next ΔU. Thus we achieve some adaptive step-sizing.

An exception is that if the algorithm has just switched from a second method ΔU to a first method (Newton Rhapson) ΔU, k_r^{ext} is set back to 1.

The merit function used for determining whether a step is good is:

$$Z_{merit}^{ext} = \frac{1}{2}\sum_i (e_i)^2 \tag{14}$$

where the e_i are violations of the ith constraint, where i runs over all constraints in equations (8), (9), and (10). That is, a step must reduce Z_{merit}^{ext} or it is not accepted and is reduced in size by the two factor.

A key point to notice is that if we do not reduce t_f too much, we will still be close to a feasible solution, with the last feasible U as our guess, so that the Newton Rhapson ΔU should converge quickly.

3.2.1. Overview of the Method for Obtaining the Quadratically Convergent ΔU

This is a new method for finding a Near-Minimum-2-Norm solution to a set of linear equations and linear inequalities, that to our knowledge has never been discovered nor presented prior to the work described in this paper.

The details of this method are given in the section **3.2.5. Quadratically Convergent ΔU Details (Newton Rhapson Method).** This method solves the linear equalities exactly while trying to minimize one-half the sum of the squares of the linear inequality constraint violations. But, in each of the steps within this subproblem, the hessian matrices with respect to the u_k (in $\Delta U \equiv \{u_1^T \quad u_2^T \quad \ldots \quad u_{N+1}^T\}^T$) are taken to be the usual hessian plus εI $\{\varepsilon = 10^{-4}\}$ where I is the identity matrix of proper dimension. This εI keeps the hessians positive definite and also prevents too large a move from occurring in each step within the subproblem. The halve-double step-size selection method (same as that described in the section **3.2. Method of Solving for a Feasible Solution for Fixed Final Time**) is employed in the one-dimensional searches within the linear subproblem.

This approach, if taken to convergence, results in a step ΔU that is close to the minimum 2-norm solution to the set of linear equations and linear inequalities, which is especially useful when one is close to a feasible solution in the full problem. A minimum 2-norm solution of the set of linear equalities and linear inequalities is known to produce quadratic convergence to a feasible solution of the full nonlinear problem with nonlinear equality constraints and nonlinear inequality constraints. The "Near-Minimum-2-Norm" solution found by the method above produces the same quadratic convergence.

3.2.2. Achieving Order(N) Solution Time

The algorithm's Order(N) solution time, where again N is the number of input variables used to parametrize the torque history, is achieved through two avenues. One is the use of a banded matrix approach, and the other is the way that the inequalities are treated in the linear subproblems, as is described in the Sections **3.2.5. Quadratically Convergent ΔU Details (Newton Rhapson Method)** and **3.2.6. Convergence Assuring ΔU Details**, namely the lumping together of the squares of all the inequality violations into one quantity.

First, clearly the solution of a problem $Jv=b$ is Order(N) in the number of elements of the vector v if J is a banded matrix whose band width is independent of N. Second, the treatment of the inequalities by squaring them and lumping them into one sum prevents the number of iterations from growing linearly with N, whereas standard simplex and active set procedures necessarily have Order(N) iterations since they "stop and restart" when a new inequality constraint becomes active. For example, in the simplex method for solving Linear Programs [5] and Quadratic Programs [1, pp. 494-509], the number of pivots is observed to be linear in the number of inequalities. Hence, for these standard methods, the number of iterations being Order (N) and the cost of one iteration being Order(N) yields a total cost that is of course Order(N^2).

3.2.3. Achieving Robustness

Robustness of the algorithm is achieved partly by the way it handles the final time variable t_f. More standard approaches use t_f as one of the variables, and typically what happens is that the SQP algorithm will reduce t_f too far to infeasible values during its search and fail to ever find a feasible solution, much less a minimum time solution. Or, even worse, they crash t_f so small that singular matrices occur and they come back with a singular error message, at which time the program crashes. Our program has never crashed.

Many SQP codes crash when singularity of the constraint jacobian matrix occurs. As will be described shortly, our method includes a backup algorithm for assuring convergence, and this backup algorithm has a coefficient matrix that can be proven to be never singular.

3.2.4. Notation for the Linearization

Let changes in $\bar{x}_k$ be denoted by

$$x_k \equiv \Delta\bar{x}_k ,(k=1,\dots,N+1) \tag{15}$$

Note: This is just for the subproblem notations; we do not use $\Delta\bar{x}_k$ to update $\bar{x}_k$. Instead, we let the forward integration find the $\bar{x}_k ,(k=1,\dots,N+1)$ from the $\bar{u}_k ,(k=1,\dots,N)$.

Linearizing Eqn. (6), we obtain

$$x_{k+1}=A_k x_k + B_k u_k, (k=1,\dots,N) \tag{16}$$

where of course,

$$A_k=\frac{\partial f(\bar{x}_k,\bar{u}_k)}{\partial\bar{x}_k}, B_k=\frac{\partial f(\bar{x}_k,\bar{u}_k)}{\partial\bar{u}_k} \tag{17}$$

Likewise, linearizing Eqn. (8), we obtain

$$D_k x_k + C_k u_k = d_k, (k=1,\dots,N+1) \tag{18}$$

where of course $d_k=-g(\bar{x}_k,\bar{u}_k)$.

and likewise Eqn. (9) gives

$$E_k x_k + F_k u_k \le s_k \tag{19}$$

and Eqn. (10) gives

$$Sx_{N+1}+Tu_{N+1}=z \tag{20}$$

where A_k, B_k, C_k, D_k, d_k, E_k, F_k, s_k, S, and T are of course constants for a given linearization.

The above derivatives for the three link example problem were obtain analytically, with Mathematica. We will note that the algorithm uses only first order derivative information from the model's state equation.

Let changes in u_k within the subproblem be denoted by

$$\tilde{u}_k \equiv \Delta u_k, (k=1,\dots,N+1) \tag{21}$$

and likewise changes in x_k by

$$\tilde{x}_k \equiv \Delta x_k, (k=2,\dots,N+1) \tag{22}$$

3.2.5. Quadratically Convergent ΔU Details (Newton Rhapson Method)

Figure 4 shows the matrix and right hand side used to solve for the $\tilde{x}_k$ and $\tilde{u}_k$ in each iteration in the subproblem. The captions under **Figure 4** explain the notation used in the figure. The λ_s^k are the Lagrange Multipliers associated with the state equation constraints, such as equations 1, 4, and 8 in the figure. The λ_p^k are those associated with the equality path constraints (equations 3 and 7 in **Figure 4**) and λ_{term} that associated with the terminal equality constraint (equation 13 in **Figure 4**).

We note from the figure that we have a right hand side associated with the linear equality constraints (even for the linearized state equation). Ideally, these right hand sides will always be zero, but due to occasional round off errors they can become nonzero. By not blindly assuming they are zero, we effectively provide some iterative refinement to hold down any possible round off errors that would otherwise result in not satisfying the linear equalities accurately. In **Figure 4**, these errors that should ideally be zero are denoted by e_x^k (linear state equation violation), q_k (linear path equality constraint violation), and q_{last} (linear terminal equality constraint violation). We recognize these right hand sides corresponding to linear equality constraints as the negative of the violations of these linear equality constraints. We recognize the $\bar{f}_x^k$ and $\bar{f}_u^k$ as the derivatives of our cost function:

$$Z_1^{int} = \frac{1}{2}\sum_i \left(e_{inequality_i}\right)^2 \qquad (23)$$

with respect to x_k and u_k, respectively, where $e_{inequality_i}$ is the violation of the ith (scalar) linear inequality constraint. The merit function used for the halve/double one-dimensional searches is one half the sum the squares of all linear constraint violations, both equality and inequality:

$$Z_{merit}^{int} = \frac{1}{2}\sum_i \left(e_{inequality_i}{}^2 + e_{equality_i}{}^2\right) \qquad (24)$$

With infinite precision arithmetic, this is the same as (23), but not when round off errors exist. As mentioned in the section **2.1.1. Method of Solving for a Feasible Solution for a Fixed Final Time**, if after a certain number of iterations {5}, the x_k and u_k fail to produce a solution that has a smaller Z_{merit}^{int} than $(x_k = 0, \forall k; u_k = 0, \forall k)$, (or if singularity ever occurs in the matrix of **Figure 4**), the second method, i.e. the Convergence Assuring method for ΔU takes over.

3.2.6. Convergence Assuring ΔU Details

Here, instead of solving the linear equalities exactly, we simply include the sum of the squares of their violations into the cost function Z^{int}, so that now Z^{int} is:

$$Z_2^{int} = \frac{1}{2}\sum_i \left(e_{inequality_i}{}^2 + e^{*}_{equality_i}{}^2\right) \qquad (25)$$

where the $e^{*}_{equality_i}$ means all linear equalities except the linearized state equations. Z_{merit}^{int} is the same as in the method of section 3.2.5. The matrix and right hand sides for these iterations within the subproblem are given in **Figure 5** for the case of N=3. **Figure 5** can be derived similarly to **Figure 4**.

Again, the same halve/double step-size selection method that was described in section 3.2.1 is employed with the $\tilde{x}_k$ and $\tilde{u}_k$ when updating the x_k and u_k.

It can be shown that the matrix of coefficients for this method is *never* singular.

3.2.7. Termination Criterion

Two criterion are used for determining when the final time t_f is small enough. There are two parameters $\Delta t_{f_1}^{max}$ {=.01} and $\Delta t_{f_2}^{max}$ {=.0001}. If $|\Delta t_f| < \Delta t_{f_1}^{max}$ and the theoretical saturation of the inputs have been achieved, we terminate. Also, if $|\Delta t_f| < \Delta t_{f_2}^{max}$, we terminate.

3.2.8. Numerical Results--Three Link Redundant Arm

Figure 1 shows a schematic of the problem.

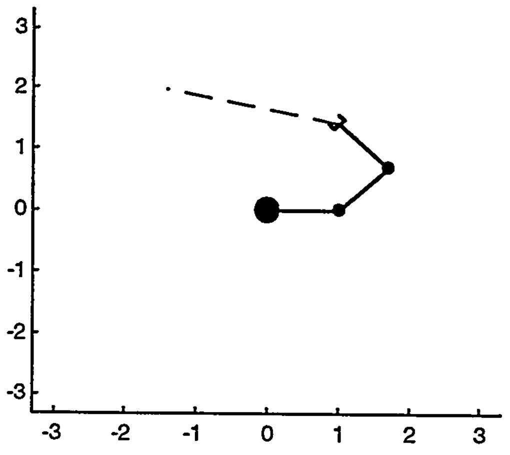

Figure 1

Joint angles are measured counterclockwise (ccw), and their values for **Figure 1** are: $\theta_1 = 0$, θ_2=1.57/2 rad, θ_3=1.57 rad, which represent the initial position of the arm. The arm is initially at rest. The dashed line in **Figure 1** represents the straight line tip path that the arm's tip must stay on. Torques are also measured ccw and their bounds are: $u_{1_{max}} = .25Nm$, $u_{2_{max}} = .09Nm$, $u_{3_{max}} = .05Nm$, and $u_{min_i} = -u_{max_i} \forall i$. The tip must move from its current position if **Figure 1** to the endpoint of the line segment. The joint angle bounds are: $\theta_{1_{max}} = 2.618 rad$, $\theta_{2_{max}} = \frac{\pi}{2} rad$, $\theta_{3_{max}} = 2.2 rad$ and $\theta_{min_i} = -\theta_{max_i} \forall i$. The final required position of the arm tip is:

x = -1.4108695652173910m

y = 1.9576271186440690m

The maneuver is required to terminate at rest, i.e., zero terminal velocity.

Gravity does not act in the plane of the arm.

The parameters of the arm are: $m_i = 1kg\forall i$, $L_i = 1m, \forall i$, $\bar{I}_{G_i} = \frac{1}{12} m_i L_i^2 \; \forall i$, $L_{G_i} = L_i / 2\forall i$, where $\bar{I}_{G_i}$ is the centroidal rotational inertia of the ith link and L_{G_i} is the centroidal position of link i, measured from the base of link i.

This problem was solved for N=50 in 70 seconds on a hewlett packard workstation, and the solve time for N=100 was 135 seconds, as illustrated in **Table 1** below, thus demonstrating the Order(N) complexity of the algorithm.

N	Solve Time
50	70 seconds
100	135 seconds

Table 1

The minimum time was found to be $t_f = 11.42$ seconds. The minimum time torque history is plotted in **Figure 2** and the joint angle and joint angle velocities plotted in **Figure 3.** As expected from the Pontryagin Maximum Principle, two out of the three torques are saturated at each time point for which the joint angle bounds are not saturated. The second joint angle bound saturates at about time=5 seconds, as seen in **Figure 3**, explaining why only one of the three torques is saturated at that time, as seen in **Figure 2.**

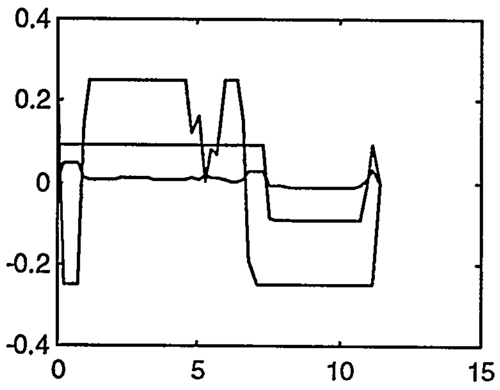

Figure 2 Torques(Nm) Versus Time (Sec)

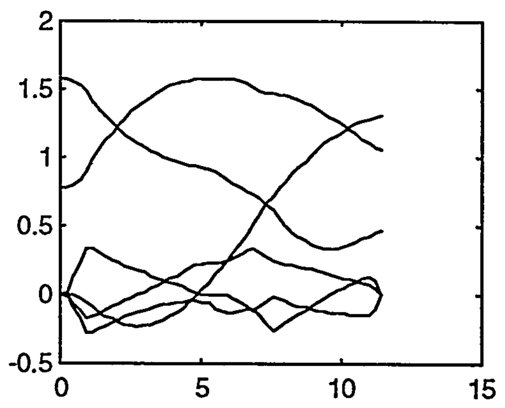

Figure 3: Joint Angles(rad) and Joint Rates (rad/s) Versus Time (Sec)

4. Conclusion

We have presented and demonstrated a new fast and robust algorithm for solving general inequality/equality constrained minimum time problems. The method has solution time that is linear in the number of Runge-Kutta steps and the number of parameters used to parametrize the control history. Its robustness has also been demonstrated.

The speed is achieved by the quadratic convergence of the fixed-final-time re-solves and the Order(N) complexity of the algorithm. The quadratic convergence is achieved in turn by a new Order(N) algorithm that finds a near-minimum-2-norm solution to a set of linear equations and inequalities. The Order (N) solution time is due to the banded matrix approach and the special way that the inequalities are handled by lumping them into a single sum-of-the-squares error quantity.

The robustness of the algorithm is achieved by the way it treats the final time variable to be minimized, in particular by reducing the final time a small amount and re-solving for a feasible solution given the last torque history for the previous (larger) final time. Robustness is also achieved by the backup approach which assures convergence of the fixed-final-time iterations.

The algorithm conveniently uses only first order derivative information from the state equation of the dynamic system.

References

[1] Bazarra, M.S. and Shetty, C.M., *Nonlinear Programming*, New York: John Wiley and Sons, 1979.

[2] Bobrow, J.E. et al., "On the Optimal Control of Robotic Manipulators with Actuator Constraints," *Proceedings of the 1983 American Control conference*, vol. 2, pp. 782-787, 1983.

[3] Byers, et al., "Near-Minimum Time, Closed Loop Slewing of Flexible Spacecraft," *Journal of Guidance, Control, and Dynamics*, vol. 13, No. 1, Jan.-Feb., 1990.

[4] Byers, R.M. and Vadali, S.R., "Quasi-Closed-Form Solution to the Time-Optimal Rigid Spacecraft Reorientation Problem," *Journal of Guidance, Control, and Dynamics*, vol. 16, No.3, May-June 1993.

[5] Chvatal, V., *Linear Programming*, New York: W.H. Freeman and Company, 1983.

[7] Geering, H.P. et al. 1986 *IEEE Transactions on Automatic Control*, vol. AC-31, No. 6, June 1986, "Time-Optimal Motions of Robots in Assembly Tasks."

[8] Ho, Y. and Bryson, A.E., *Applied Optimal Control*, New York: Hemisphere Publishing Corporation, 1975.

[9] Larson, R.E. and Casti, J.L., *Principles of Dynamic Programming, Part II, Advanced Theory and Applications*. New York: Marcel Dekker, Inc., 1982.

[10] Lewis, F.L., *Optimal Control*, New York: John Wiley and Sons, Inc., 1986.

[11] Meier, E. and Bryson, A.E., "Efficient Algorithm for Time-Optimal Control of a Two-Link Manipulator," *Journal of Guidance, Control, and Dynamics*, vol. 13, No. 5, Sept.-Oct., 1990.

[12] Shin, K.G and McKay, N.D., "Minimum-Time Control of Robotic Manipulators with Geometric Paths," *IEEE Transactions on Automatic Control*, vol. AC-30, No. 6, June 1985.

[13] Shin, K.G and McKay, N.D., "Open-Loop Minimum-Time Control of Mechanical Manipulators and its Application," *Conference Proceedings of the 1984 American Control Conference*, vol. 3, pp. 1231-1236.

[14] Shin, K.G. and McKay, N.D., "A Dynamic Programming Approach to Trajectory Planning of Robotic Manipulators," *IEEE Transactions on Automatic Control*, vol. AC-31, No. 6, June 1986.

[15] Shin, K.G. and McKay, N.D., "Selection of Near-Minimum Time Geometric Paths for Robotic Manipulators," *IEEE Transactions on Automatic Control*, vol. AC-31, NO. 6, June 1986.

[16] Wie, B. and Sunkel, J. 1990 *Journal of Guidance, Control, and Dynamics*, vol. 13, No. 5, Sept.-Oct.,1990, pp. 867-873, "Minimum-Time Pointing Control of a Two-Link Manipulator."

[17] Wright, S., "Structure Interior Point Methods for Optimal Control," Proceedings of the 30th Conference on Decision and Control, Brighton England, Dec., 1991, pp. 1711-1716.

[18] Zimmerman, D.C. and Layton, D.S., "Large Angle Slewing Maneuvers Using Performance Driven Darwinian Learning Controllers: Theory and Experiment," *The 34th AIAA / ASME / ASCE / AHS / ASC / Structures, Structural Dynamics, and Materials Conference, AIAA/ASME Adaptive Structures Forum, Part 6*, April 19-22, 1993, pp. 3540-3550.

	$\bar{u}_1$	λ_s^1	$\bar{x}_2$	$\bar{u}_2$	λ_p^2	λ_s^2	$\bar{x}_3$	$\bar{u}_3$	λ_p^3	λ_s^3	$\bar{x}_4$	$\bar{u}_4$	λ_{term}				
1	B_1	0	$-I$											$=$	$-(B_1u_1 - x_2)$	$\equiv$	e_s^1
2	$-H_{uu}^1$	B_1^T	0											$=$	$F_1^T(F_1u_1 - s_1)_+$	$\equiv$	$\bar{f}_u^1$
3			D_2	C_2										$=$	$-(D_2x_2 + C_2u_2 - d_2)$	$\equiv$	q_2
4			A_2	B_2	0	0	$-I$							$=$	$-(A_2x_2 + B_2u_2 - x_3)$	$\equiv$	e_s^2
5		$-I^T$	$(-E_2^TE_2)_{++}$	$(-E_2^TF_2)_{++}$	D_2^T	A_2^T	0							$=$	$E_2^T(E_2x_2 + F_2u_2 - s_2)_+$	$\equiv$	$\bar{f}_x^2$
6			$(-F_2^TE_2)_{++}$	$-H_{uu}^2$	C_2^T	B_2^T	0							$=$	$F_2^T(F_2u_2 + E_2x_2 - s_2)_+$	$\equiv$	$\bar{f}_u^2$
7							D_3	C_3						$=$	$-(D_3x_3 + C_3u_3 - d_3)$	$\equiv$	q_3
8							A_3	B_3	0	0	$-I$			$=$	$-(A_3x_3 + B_3u_3 - x_4)$	$\equiv$	e_s^3
9						$-I^T$	$(-E_3^TE_3)_{++}$	$(-E_3^TF_3)_{++}$	D_3^T	A_3^T	0			$=$	$E_3^T(E_3x_3 + F_3u_3 - s_3)_+$	$\equiv$	$\bar{f}_x^3$
10							$(-F_3^TE_3)_{++}$	$-H_{uu}^3$	C_3^T	B_3^T	0			$=$	$F_3^T(F_3u_3 + E_3x_3 - s_3)_+$	$\equiv$	$\bar{f}_u^3$
11										$-I^T$	$(-E_4^TE_4)_{++}$	$(-E_4^TF_4)_{++}$	S^T	$=$	$E_4^T(E_4x_4 + F_4u_4 - s_4)_+$	$\equiv$	$\bar{f}_x^4$
12											$(-F_4^TE_4)_{++}$	H_{uu}^4	T^T	$=$	$F_4^T(F_4u_4 + E_4x_4 - s_4)_+$	$\equiv$	$\bar{f}_u^4$
13											S	T	0	$=$	$-(Sx_4 + Tu_4 - z)$	$\equiv$	q_{last}

Figure 4: Matrix and Right Hand Side for N=3, for Quadratically Convergent Iterations. Note: $(V)_+$ means: $((V)_+)_i = \left\{ \begin{matrix} 0 \text{ if } V_i < 0 \\ V_i, \text{ otherwise} \end{matrix} \right\}$, $(i = 1, \ldots, \delta)$, where $V \in R^\delta$. Note: $(E_k^TF_k)_{++}$ means treat as identically zero the rows of E_k and F_k that correspond to linear inequalities that are not violated by x_k and u_k, or, equivalently, when treating $E_k^TF_k$

as the sum of vector outer products, leave out those outer products corresponding to unviolated linear inequalities. Likewise for $\left(E_k^T E_k\right)_{++}$, etc. $H_{uu}^k = \left(F_k^T F_k\right)_{++} + \varepsilon I$, $\varepsilon \approx 10^{-4}$, $(k = 1, \ldots, N+1 = 4)$.

	$\tilde{u}_1$	λ_s^1	$\tilde{x}_2$	$\tilde{u}_2$	λ_s^2	$\tilde{x}_3$	$\tilde{u}_3$	λ_s^3	$\tilde{x}_4$	$\tilde{u}_4$				
1	B_1	0	$-I$								$=$	$-(B_1 u_1 - x_2)$	$\equiv$	e_s^1
2	$-H_{uu}^1$	B_1^T	0								$=$	$F_1^T(F_1 u_1 - s_1)_+$	$\equiv$	f_u^2
3			A_2	B_2	0	$-I$					$=$	$-(A_2 x_2 + B_2 u_2 - x_3)$	$\equiv$	e_s^2
4		$-I^T$	$-H_{xx}^2$	$-H_{xu}^2$	A_2^T						$=$	$E_2^T(E_2 x_2 + F_2 u_2 - s_2)_+ + D_2^T(D_2 x_2 + C_2 u_2 - d_2)$	$\equiv$	f_x^2
5			$-H_{ux}^2$	$-H_{uu}^2$	B_2^T						$=$	$F_2^T(F_2 u_2 + E_2 x_2 - s_2)_+ + C_2^T(D_2 x_2 + C_2 u_2 - d_2)$	$\equiv$	f_u^2
6						A_3	B_3	0	$-I$		$=$	$-(A_3 x_3 + B_3 u_3 - x_4)$	$\equiv$	e_s^3
7					$-I^T$	$-H_{xx}^3$	$-H_{xu}^3$	A_3^T			$=$	$E_3^T(E_3 x_3 + F_3 u_3 - s_3)_+ + D_3^T(D_3 x_3 + C_3 u_3 - d_3)$	$\equiv$	f_x^3
8						$-H_{ux}^3$	$-H_{uu}^3$	B_3^T			$=$	$F_3^T(F_3 u_3 + E_3 x_3 - s_3)_+ + C_3^T(D_3 x_3 + C_3 u_3 - d_3)$	$\equiv$	f_u^3
9								$-I^T$	$-H_{xx}^4$	$-H_{xu}^4$	$=$	$E_4^T(E_4 x_4 + F_4 u_4 - s_4)_+ + S^T(S x_4 + T u_4 - d_4)$	$\equiv$	f_x^4
10									$-H_{ux}^4$	$-H_{uu}^4$	$=$	$F_4^T(F_4 u_4 + E_4 x_4 - s_4)_+ + T^T(S x_4 + T u_4 - d_4)$	$\equiv$	f_u^4

Figure 5: Matrix and Right Hand Side for N=3, for Convergence Assuring Iterations. Note: $H_{xx}^k = \left(E_k^T E_k\right)_{++} + D_k^T D_k$ $(k = 1, \ldots, N = 3)$; $H_{xu}^k = \left(E_k^T F_k\right)_{++} + D_k^T C_k$, $(k = 1, \ldots, N = 3)$; $H_{ux}^k = \left(H_{xu}^k\right)^T$, $(k = 1, \ldots, N+1 = 4)$; $H_{uu}^k = \left(F_k^T F_k\right)_{++} + C_k^T C_k + \varepsilon I$, $\varepsilon \approx 10^{-4}$, $(k = 1, \ldots, N = 3)$; $H_{xx}^{N+1} = \left(E_{N+1}^T E_{N+1}\right)_{++} + S^T S$; $H_{xu}^{N+1} = \left(E_{N+1}^T F_{N+1}\right)_{++} + S^T T$; $H_{ux}^{N+1} = \left(H_{xu}^{N+1}\right)^T$; $H_{uu}^{N+1} = \left(F_{N+1}^T F_{N+1}\right)_{++} + T^T T + \varepsilon I$, $\varepsilon \approx 10^{-4}$. See Figure 4 caption for definition of $\left(E_k^T F_k\right)_{++}$,etc. and $(V)_+$.