without making any approximations. They derived the digital optimal regulator and tracker for linear time-varying systems.

The digital optimal regulator permits the design of a digital optimal perturbation controller for non linear systems that have to track reference trajectories. Important applications are e.g. a robot performing a prescribed motion or a batch fermentation process, where in both cases the linearized dynamics about the trajectory constitute a *time-varying* system. Until now only the computation of the digital optimal regulator for *time-invariant* systems has been considered (Van Loan 1978). The digital optimal tracker, based on dynamic optimization, has never been considered in the literature before. This is remarkable because it can be applied in all situations were a linear system, controlled by a computer, has to track a reference trajectory, e.g. a cartesian type robot performing a prescribed motion.

Van Willigenburg and De Koning (1990a) derived the digital optimal regulator and tracker, in case of deterministic systems and complete state information at the sampling instants, via both static and dynamic optimization. In a second paper the authors solved the problems in case of stochastic systems and incomplete state information (Van Willigenburg and De Koning, 1990b), which is only possible using dynamic optimization. The solution is given in feedback form. Although the authors solved the problems, they did not specify numerical procedures to compute the digital optimal regulator and tracker. In this paper we will present such procedures, based on both static and dynamic optimization. For several examples it is shown that the solutions based on both approaches are identical, which they of course should be. Finally it is demonstrated that the computational procedure based on dynamic optimization is superior with respect to accuracy computation time and the use of computer memory.

2. The digital optimal regulator and tracker

2.1 The digital optimal regulator and tracking problem

For convenience, and since the problems are certainty equivalent (Van Willigenburg and De Koning, 1990b), we will treat the digital optimal regulator and tracking problem for deterministic systems and complete state information at the sampling instants. However, in section 3 where we present numerical procedures, we will treat the case of stochastic systems and incomplete state information as well. Consider the deterministic continuous-time linear time-varying system

$$\dot{x}(t) = A(t)\, x(t) + B(t)\, u(t), \tag{1a}$$

with known initial state

$$x(t_0) = x_0, \tag{1b}$$

where A(t) and B(t) are the system matrices. The control is piecewise constant, i.e.

$$u(t) = u(t_k), \quad t\in[t_k,t_{k+1}), \quad k=0,1,2,3\ldots, \tag{1c}$$

where t_k are the, not necessarily equidistant, sampling instants. We assume complete state information at the sampling instants, so $x(t_k)$, k=0,1,2,3,... are available.

The *digital optimal regulator problem* for this system is to minimize

$$J = \int_{t_0}^{t_f} [x^T(t)\, Q(t)\, x(t) + u^T(t)\, R(t)\, u(t)]\, dt + x^T(t_f)\, H\, x(t_f) \Big), \tag{2}$$

where $Q(t)\geq 0$, $H\geq 0$ and $R(t)\geq 0$. Furthermore

$$t_f = t_N, \tag{3}$$

where N is a positive integer.

The *digital optimal tracking problem* takes the following form. Given the system (1) and a reference trajectory

$$x_r(t), \quad t_0 \le t \le t_f, \tag{4}$$

minimize

$$J = \int_{t_0}^{t_f} [(x(t)-x_r(t))^T Q(t) (x(t)-x_r(t)) + u^T(t) R(t) u(t)] dt + (x(t_f) - x_r(t_f))^T H (x(t_f) - x_r(t_f)) \tag{5}$$

where again (3) holds and we assume $Q(t) \ge 0$, $H \ge 0$ and $R(t) \ge 0$.

2.2 Solution to the digital optimal regulator and tracking problem via static optimization.

The solution to the digital optimal regulator and tracking problem is uniquely determined by the control sequence

$$u_k = u(t_k), \qquad k=0,1,2,\ldots,N-1. \tag{6}$$

Introducing so called block pulse functions $v_k(t)$ defined by

$$\begin{aligned} v_k(t) &= 1, \qquad t \in [t_k, t_{k+1}), \\ v_k(t) &= 0, \qquad \text{elsewhere}, \end{aligned} \tag{7}$$

the control at each time t, $t_0 \le t \le t_f$ is given by

$$u(t) = \sum_{k=0}^{N-1} v_k(t)\, u_k \tag{8}$$

or

$$u(t)=[v_0(t)I_m\ v_1(t)I_m\ v_2(t)I_m \cdots v_{N-1}(t)I_m] \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \cdot \\ \cdot \\ u_{N-1} \end{bmatrix}, \tag{9}$$

where I_m are identity matrices of dimension m where m is the dimension of the control vector u(t). Equation (9) can be written as

$$u(t) = V(t)\ U, \tag{10}$$

where

$$U = [\ u_0^T\ u_1^T\ u_2^T\ \ldots\ldots u_{N-1}^T\ ]^T \tag{11}$$

is a vector of length mN and

$$V(t) = [\ v_0I_m\ \ v_1I_m\ \ v_2I_m\ \ldots\ldots v_{N-1}I_m\ ] \tag{12}$$

is a matrix with dimensions m x mN. The solution to the digital optimal regulator problem is given by (Van Willigenburg a Koning, 1990a)

$$U = -\ R'^{-1}\ M'\ x(t_0), \tag{13a}$$

and the minimum cost are

$$J = x^T(t_0)(H'-M'^TR'^{-1}M')x(t_0), \tag{13b}$$

where

$$H' = \Phi^T(t_f,t_0)\ H\ \Phi(t_f,t_0) + \int_{t_0}^{t_f} \Phi^T(t,t_0)\ Q(t)\ \Phi(t,t_0)\ dt \tag{14a}$$

which is a n x n matrix,

$$M' = \Gamma_v^T(t_f,t_0)\ H\ \Phi(t_f,t_0) + \int_{t_0}^{t_f} \Gamma_v^T(t,t_0)\ Q(t)\ \Phi(t,t_0)\ dt \qquad (14b)$$

which is a mN x n matrix,

$$R'=\Gamma_v^T(t_f,t_0)\ H\ \Gamma_v(t_f,t_0) + \int_{t_0}^{t_f} [\Gamma_v^T(t,t_0)Q(t)\Gamma_v(t,t_0)+V^T(t)R(t)V(t)]\ dt \qquad (14c)$$

which is a mN x mN matrix,

$$\Gamma_v(t,t_0) = \int_{t_0}^{t} \Phi(t,s)\ B(s)\ V(s)\ ds \qquad (14d)$$

which is a n x mN matrix, where n equals the dimension of system (1), and Φ is the state transition matrix of system (1). Except for $t=t_0$ solution (13a) is not in feedback form. To obtain a solution in feedback form, at every sampling instant t_k, one has to solve a new static optimization problem (Van Willigenburg and De Koning, 1990a). The solution in feedback form is given by

$$U_k = -R_k'^{-1}\ M_k'\ x(t_k), \qquad k=0,1,2,..,N-1, \qquad (15)$$

where

$$U_k = [\ u_k^T\ u_{k+1}^T\ u_{k+2}^T,....u_{N-1}^T\ ]^T \qquad (16)$$

of which only u_k is used for the actual control. M_k' and R_k' are of dimension m(N-k) x n and m(N-k) x m(N-k) and given by (14) with t_0 replaced by t_k. Note that for calculation of the optimal control (15) it is sufficient to know R' and M'.

The solution to the digital optimal tracking problem is given by

$$U = -R'^{-1}\ (M'x(t_0) - L'), \tag{17a}$$

and the minimum cost are,

$$J = x^T(t_0)(H'-M'^T R'^{-1} M')x(t_0) + 2x^T(t_0)M'^T R'^{-1} L' - L'^T R'^{-1} L' + J_1 \tag{17b}$$

where H', R', M' and Γ_v are given by (14), and

$$L' = \Gamma_v^T(t_f,t_0)\ H\ x_r(t_f) + \int_{t_0}^{t_f} \Gamma_v^T(t,t_0)\ Q(t)\ x_r(t)\ dt, \tag{18a}$$

and finally,

$$J_1(x(t_0),x_r(t)) = \int_{t_0}^{t_f} \left[x_r^T(t)Q(t)x_r(t) - 2x_r^T(t)Q(t)\Phi(t,t_0)x(t_0)\right]dt$$

$$+x_r^T(t_f)Hx_r(t_f) - 2x_r^T(t_f)H\Phi(t_f,t_0)x(t_0). \tag{18b}$$

As in the regulator case, except for $t=t_0$, the solution is not in feedback form. To obtain a solution in feedback form one again has to solve a static optimization problem at each sampling instant t_k. The solution in feedback form is given by (Van Willigenburg and De Koning, 1990a)

$$U_k = -R_k'^{-1}\ (M_k'\ x(t_k) - L_k'), \qquad k=0,1,2,..,N-1, \tag{19}$$

where U_k, R_k', M_k' are the same as in the regulator case and L_k', equals (18a), with t_0 replaced by t_k. Note that for calculation of the optimal control (19) it is sufficient to know R_k', M_k' and L_k'. Finally we should remark that clearly the solutions only exist if $R_k'>0$, $k=0,1,2,..,N-1$. If $Q(t)\geq 0$, $H\geq 0$ and $R(t)\geq 0$, as

assumed in paragraph 2.1, this condition is nearly always satisfied (Van Willigenburg and De Koning, 1990a).

Now very important, and opposite to what is mentioned by Nour Eldin (1971), except for R'_k which equals the final m(N-k) rows and columns of R', the matrix M'_k, which is involved in the solution of the digital optimal regulator problem, and both M'_k and L'_k which are involved in the solution of the digital optimal tracking problem, are not sub matrices of M' and L' respectively. They have to be computed again at every sampling instant which presents a serious drawback compared to the dynamic optimization approach, in case of both the digital optimal regulator and tracker given in feedback form (see paragraph 2.3).

2.3 Solution to the digital optimal regulator and tracking problem via dynamic optimization.

The solution of the digital optimal regulator and tracking problem via dynamic optimization involves two stages. First the digital optimal regulator and tracking problem, with the piecewise constant constraint on the control, are transformed into so called equivalent discrete-time problems, with unconstrained control, since these can be solved using "standard" techniques (Van Willigenburg and De Koning, 1990a). The original digital optimal regulator problem (1), (2), (3) can be transformed the following equivalent discrete-time problem with unconstrained control. Given the so called equivalent discrete-time system

$$x_{k+1} = \Phi_k x_k + \Gamma_k u_k \tag{20a}$$

which represents the continuous-time system behavior at the sampling instants where

$$x_k = x(t_k), \tag{20b}$$
$$u_k = u(t_k), \tag{20c}$$
$$\Phi_k = \Phi(t_{k+1}, t_k), \tag{20d}$$

$$\Gamma_k = \Gamma(t_{k+1}, t_k), \tag{20e}$$

with Φ being the state transition matrix of system (1) and

$$\Gamma(t,t_k) = \int_{t_k}^{t} \Phi(t,s)\ B(s)\ ds \tag{21}$$

minimize

$$J = \sum_{k=0}^{N-1} \left(x_k^T\ Q_k\ x_k + 2\ x_k^T\ M_k\ u_k + u_k^T\ R_k\ u_k \right) + x_N^T\ H\ x_N \tag{22}$$

where

$$Q_k = \int_{t_k}^{t_{k+1}} \Phi^T(t,t_k)\ Q(t)\ \Phi(t,t_k)\ dt, \tag{23a}$$

$$M_k = \int_{t_k}^{t_{k+1}} \Phi^T(t,t_k)\ Q(t)\ \Gamma(t,t_k)\ dt, \tag{23b}$$

$$R_k = \int_{t_k}^{t_{k+1}} [R(t) + \Gamma^T(t,t_k)\ Q(t)\ \Gamma(t,t_k)]\ dt. \tag{23c}$$

If $R_k>0$, $Q_k\geq 0$ and $H\geq 0$, which holds in almost any case if $R(t)\geq 0$, $Q(t)\geq 0$ and $H\geq 0$, as assumed in paragraph 2.1 (Van Willigenburg and De Koning, 1990a), the solution of the equivalent discrete-time problem (20), (22) exists. The solution can be presented in several forms (Van Willigenburg and De Koning, 1990a). We will present the form that will be used to numerically compute the digital optimal regulator.

$$u_k = -(K_k' + R_k^{-1} M_k^T)\, x_k, \qquad k=0,1,2,..,N-1, \tag{24a}$$

$$K_k' = (\Gamma_k^T S_{k+1} \Gamma_k + R_k)^{-1}\ \Gamma_k^T S_{k+1} \Phi_k', \tag{24b}$$

$$S_k = (\Phi_k' - \Gamma_k K_k')^T S_{k+1} (\Phi_k' - \Gamma_k K_k') + K_k'^T R_k K_k' + Q_k', \quad S_N = H, \tag{24c}$$

$$\Phi_k' = \Phi_k - \Gamma_k R_k^{-1} M_k^T, \tag{24d}$$

$$Q_k' = Q_k - M_k R_k^{-1} M_k^T. \tag{24e}$$

Equations (24b), (24c) constitute a Riccati type recursion, written in the so called Joseph stabilized form which is known for its good numerical performance (see section 3).

The digital optimal tracking problem (1), (3), (4), (5) can be transformed into the following equivalent discrete-time problem (Van Willigenburg and De Koning, 1990a). Given the equivalent discrete-time system (20) minimize

$$J = \sum_{k=0}^{N-1} \left(x_k^T Q_k x_k + 2x_k^T M_k u_k + u_k^T R_k u_k - 2L_k x_k - 2T_k u_k + X_k \right) + x_N^T H x_N - 2x_r^T(t_f) H x_N + x_r^T(t_f) H x_r(t_f), \tag{25}$$

where Q_k, M_k, R_k are given by (23) and

$$L_k = \int_{t_k}^{t_{k+1}} x_r^T(t)\, Q(t)\, \Phi(t,t_k)\, dt, \tag{26a}$$

$$T_k = \int_{t_k}^{t_{k+1}} x_r^T(t)\, Q(t)\, \Gamma(t,t_k)\, dt, \tag{26b}$$

$$X_k = \int_{t_k}^{t_{k+1}} x_r^T(t)\, Q(t)\, x_r(t)\, dt. \tag{26c}$$

Again if $R_k > 0$, $Q_k \geq 0$ and $H \geq 0$, which holds in almost any case if $R(t) \geq 0$, $Q(t) \geq 0$ and $H \geq 0$, as assumed in paragraph 2.1 (Van Willigenburg and De Koning, 1990a), the solution of the equivalent discrete-time problem (20), (25) exists. The solution can be written in the following form which will be used to numerically compute the digital tracker (Van Willigenburg and De Koning, 1990a)

$$u_k=-(K_k'+R_k^{-1}M_k^T)x_k+K_kv_{k+1}+K_kT_k', \qquad k=0,1,2,..,N-1, \tag{27a}$$

$$K_k'=(\Gamma_k^TS_{k+1}\Gamma_k+R_k)^{-1}\Gamma_k^TS_{k+1}\Phi_k', \tag{27b}$$

$$K_k^1=(R_k+\Gamma_k^TS_{k+1}\Gamma_k)^{-1}\Gamma_k^T, \tag{27c}$$

$$K_k^2=(R_k+\Gamma_k^TS_{k+1}\Gamma_k)^{-1}, \tag{27d}$$

$$S_k=(\Phi_k'-\Gamma_kK_k')^TS_{k+1}(\Phi_k'-\Gamma_kK_k')+K_k'^TR_kK_k'+Q_k', \quad S_N=H, \tag{27e}$$

$$v_k=(\Phi_k'-\Gamma_kK_k')^Tv_{k+1}-K_k'^TT_k^T+L_k'^T, \qquad v_N=Hx_r(t_f), \tag{27f}$$

$$\Phi_k'=\Phi_k-\Gamma_kR_k^{-1}M_k^T, \tag{27g}$$

$$Q_k'=Q_k-M_kR_k^{-1}M_k^T, \tag{27h}$$

$$L_k'=L_k-T_kR_k^{-1}M_k^T. \tag{27i}$$

Note that equation (27b,e,g,h) are equal to (24b,c,d,e), which implies that the feedback for both the digital optimal regulator and tracker is the same, which matches the result obtained from static optimization, since equation (15) and (19) imply equal feedback as well. Note furthermore that we did not give an expression for the cost of the digital optimal regulator and tracker. This will be done in section 3 where we discuss numerical solutions for stochastic systems and incomplete state information.

2.4 Advantages of dynamic optimization

Using static optimization, at every sampling instant one has to compute a new optimization problem and corresponding solution, to obtain the digital optimal regulator and tracker in feedback form. Furthermore the dimensions of the matrices constituting the solutions (15), (19) obtained from static optimization, increases linearly with the horizon of the problem. For problems with large horizon the result obtained fromstatic optimization therefore becomes impractical considering computation time, and the use of computer memory. Dynamic optimization directly results in solutions given in feedback form and the dimensions of the matrices which determine the solutions (24),

(27) are small and independent of the horizon. Only the number of recursions depends linearly on the horizon of the problem. Finally in section 4 the result obtained from dynamic optimization will also appear to be numerically more reliable. Summarizing the dynamic optimization approach is superior with respect to accuracy, computation time and the use of computer memory.

3. Numerical procedures to compute the digital optimal regulator and tracker

3.1 Computational tasks

Consider the digital optimal regulator given by (15), (13b) and (14) and the digital tracker given by (19), (17b), (18) and (14) based on static optimization. Furthermore consider (20), (21), (23), and (26) which appear in the recursions (24) and (27) that constitute the digital optimal regulator and tracker, based on dynamic optimization. To compute these equations we need an algorithm to compute the state transition matrix of system (1a) and a numerical integration procedure. We will present an algorithm to compute the state transition matrix of a general linear time-varying system, which is a natural extension of the scaling and squaring algorithm (Moler and Van Loan, 1978) used to compute the state transition matrix of linear time-invariant systems. This algorithm fits very nicely a numerical integration procedure based on the trapezium rule, where the integrands are evaluated at equidistant times, as will be shown. The result will be that the equations mentioned above can all be computed recursively, forward in time, where the error due to numerical integration is of the same order as the error due to the numerical computation of the state transition matrix. Finally we will deal with the computation of the recursions (24), (27) and the cost of the digital optimal regulator and tracker based on dynamic optimization.

3.2 Computation of the state transition matrix

In the sequel we will use the following well known facts concerning the state transition matrix of the general continuous-time linear systems (1a).

$$\Phi(t_1,t_3) = \Phi(t_1,t_2)\ \Phi(t_2,t_3),\ \forall\ t_1,t_2,t_3,\ t_0 \le t_1 \le t_2 \le t_3 \le t_f, \tag{28}$$

and

$$\Phi(t_1,t_1) = I,\quad \forall\ t_1,\quad t_0 \le t_1 \le t_f, \tag{29}$$

where I is the identity matrix. Furthermore if

$$A(t) = A_i \qquad t\in[t_1,t_2], \qquad t_0 \le t_1 \le t_2 \le t_f, \tag{30}$$

i.e. A(t) is a constant matrix within $[t_1,t_2]$ then

$$\Phi(t_1,t_2) = \exp(A_1(t_2-t_1)). \tag{31}$$

To numerically compute the state transition matrix Φ, A(t), $t_0 \le t \le t_f$ in system (1a) is approximated by a series of constant matrices in the following way (Rao and Ganapathy, 1979)

$$A'(t) = A_i, \qquad t\in[t_i,t_{i+1}), \qquad i=0,1,2,..,S-1, \tag{32a}$$

where S is a positive integer and t_i are equidistant i.e.

$$t_i = t_0 + i\ \Delta t, \tag{32b}$$

and

$$A_i = \int_{t_i}^{t_{i+1}} A(t)\ dt. \tag{32c}$$

Given (1a), (32a), and (32b)

$$\Delta t = t_f / S. \tag{32d}$$

The piecewise constant approximation (32a) of A(t) is such that over each time interval $[t_i, t_{i+1})$ A(t) is approximated by its average value A_i, given by (32c). By reducing Δt, the length of the time intervals $[t_i, t_{i+1})$, the approximation can be made arbitrarily close (Rao and Ganapathy, 1979). Summarizing we approximate the system (1a) by

$$\dot{x}(t) = A'(t)\, x(t) + B(t)\, u(t). \tag{33}$$

Since A'(t) is piecewise constant the state transition matrix of (33) according to (32), (30) and (31) satisfies

$$\Phi(t_{i+1}, t_i) = \exp(A_i \Delta t), \tag{34}$$

and according to (32),(34), and (28)

$$\Phi(t_i, t_0) = \prod_{k=0}^{i-1} \exp(A_k \Delta t). \tag{35}$$

Note that when the time-varying system (1a) belongs to the class of commutative systems, i.e.

$$A(t_1)\, A(t_2) = A(t_2)\, A(t_1), \qquad \forall\, t_1, t_2 \in [t_0, t_f], \tag{36}$$

then (32c), (34), and (35) also hold for the original system (1a) since for a commutative system (Wu and Sheriff, 1976)

$$\Phi(t_1, t_2) = \exp\left(\int_{t_1}^{t_2} A(t)\, dt \right), \qquad t_0 \le t_1 \le t_2 \le t_f. \tag{37}$$

Only if the system (1a) does not belong to this class, which according to (36) is very restrictive, (32c), (34), and (35) constitute approximations with regard to system (1a). So in that case it is necessary to chose Δt sufficiently small so that (33)

approximates (1a) close enough. Summarizing (32), (34) and (35) are used to approximate the state transition matrix of system (1a). Note that the computation (35) can be done recursively, forward in time. If $\Delta t \ll 1$ then (34) can be very well approximated by a second order Taylor expansion (Moler and Van Loan, 1978)

$$\exp(A_i \Delta t) \approx I + A_i \Delta t + 0.5 A_i^2 \Delta t^2. \tag{38}$$

The error using (38) is of the order Δt^3. As will turn out later, the error using the trapezium integration rule is of the same order. If the requirement $\Delta t \ll 1$ is to severe considering the approximation of system (1a) by system (33), then (38) could be replaced by another method to compute $\exp(A_i \Delta t)$. A well known method is the scaling and squaring method given by (Moler and Van Loan, 1978)

$$\exp(A_i \Delta t) = (\exp(A_i \Delta t/m))^m, \tag{39}$$

where m is a positive, sufficiently large integer so that $\Delta t/m \ll 1$ and

$$\exp(A_i \Delta t/m) \approx I + A_i \Delta t/m + 0.5 A_i^2 (\Delta t/m)^2. \tag{40}$$

Note however that (39), (40) is exactly the same as (35), (38) with Δt replaced by $\Delta t/m$ and where A_k, $k=0,1,2,..,i$ are kept constant and equal to A_i. If it does not take much effort to evaluate A(t), clearly choosing $\Delta t \ll 1$ immediately and using (35), (38) is preferable, from the point of view of accuracy and simplicity. In the sequel we will chose $\Delta t \ll 1$, and such that (33) approximates (1a) properly, and use (32), (35), and (38) to compute the state transition matrix of the time varying system (1a). The error in (38) is then of the order Δt^3. This procedure is a natural extension of the scaling and squaring method, originally designed to compute the state transition matrix for time-invariant systems. In section 4 examples are presented that demonstrate the accuracy of this method.

3.3 Numerical integration.

Consider a general matrix function

$$F(t), \qquad t\in[t_0,t_f]. \tag{41}$$

The trapezium numerical integration rule, where the integrand is evaluated at equidistant times t_i, given by (32b), is based on the following approximation

$$\int_{t_i}^{t_{i+1}} F(t)\,dt \approx \Delta t/2\,(F(t_i) + F(t_{i+1})) \qquad i=0,1,2,..,S-1. \tag{42}$$

The error caused by the approximation (42) is of the order Δt^3 (Collatz, 1966), so of the same order as the error caused by the approximation (38). In the sequel integral signs indexed by N, refer to the values of the integral, computed using the trapezium rule as just described. So (42) may be rewritten as

$$\int_{N\,t_i}^{t_{i+1}} F(t)\,dt = \Delta t/2\,(F(t_i) + F(t_{i+1})) \qquad i=0,1,2,..,S-1. \tag{43}$$

Since

$$\int_{t_0}^{t_i} F(t)\,dt = \sum_{k=0}^{i-1} \int_{t_k}^{t_{k+1}} F(t)\,dt, \qquad i=1,2,..,S, \tag{44}$$

given (43), (44)

$$\int_{N\,t_0}^{t_{i+1}} F(t)\,dt = \int_{N\,t_0}^{t_i} F(t)dt+\Delta t/2(F(t_i)+F(t_{i+1})), \quad i=0,1,2,..,S-1. \tag{45}$$

From (45) it can be seen that each integral, using the trapezium integration rule, can be computed recursively forward in time, using successive equidistant function evaluations of the integrand. All matrices appearing in the integrands of the equations that make up the digital optimal regulator and tracker, based on static optimization, and the recursions that make up the digital optimal regulator and tracker based on dynamic optimization, are a priori known, except from Φ, Γ and Γ_v. As shown in paragraph 3.2 Φ can also be computed recursively, forward in time. Consider Γ_v, given by (14d). From (43), (45) we have

$$\int_{t_0}^{t_i} F(t)\,dt = \Delta t/2 \left(F(t_0) + F(t_i) + 2\sum_{k=1}^{i-1} F(t_k)\right), \quad i=2,3,..,S. \qquad (46)$$

Note that for i=1 (43) holds. Denoting the value of Γ_v, obtained from numerical integration, using the trapezium rule, by Γ_v^N given (14d), (46) we have

$$\Gamma_v^N(t_i,t_0) = \Delta t/2 \left[\Phi(t_i,t_0)B(t_0)V(t_0)+\Phi(t_i,t_i)B(t_i)V(t_i) + 2\sum_{k=1}^{i-1} \Phi(t_i,t_k)B(t_k)V(t_k)\right], \quad i=2,3,..,S \qquad (47)$$

and

$$\Gamma_v^N(t_{i+1},t_0) = \Delta t/2 \left[\Phi(t_{i+1},t_0)B(t_0)V(t_0)+\Phi(t_{i+1},t_{i+1})B(t_{i+1})V(t_{i+1}) + 2\sum_{k=1}^{i} \Phi(t_{i+1},t_k)B(t_k)V(t_k)\right], \quad i=1,2,..,S-1. \qquad (48)$$

Using (47, (48), (28) and (29) the following may be written

$$\Gamma_v^N(t_{i+1},t_0) =$$

$$\Phi(t_{i+1},t_i)\Gamma_V^N(t_i,t_0)+\Delta t/2\left[\Phi(t_{i+1},t_i)B(t_i)V(t_i)+B(t_{i+1})V(t_{i+1})\right] \quad (49)$$

From (49) it follows that $\Gamma_V(t_i,t_0)$ can also be computed recursively forward in time. Note that $\Phi(t_{i+1},t_i)$ in (49) was also used to compute (35) recursively forward in time. Finally Γ, given by (21), can be computed recursively forward in time in exactly the same manner. When we skip all appearances of V in (49) it holds for Γ^N, the numerical value obtained for Γ using the trapezium integration rule.

Summarizing, the matrices R', M', L', given by (14c), (14b), (18a), and in case we are interested in the minimum cost, the matrices H' and J_1, given by (14a) and (18b) which make up the solution to the digital optimal regulator and tracking problem, based on static optimization, can be computed off-line recursively forward in time using the trapezium rule. The integrands are evaluated at equidistant times t_i, given by (32b), where $\Delta t \ll 1$ and such that (33) properly approximates (1a). Except for Φ and Γ_V the integrands contain a priori known matrices. Φ is computed recursively forward in time using (32c) and (35), where (32c) is approximated according to (43). Γ_V is computed forward in time using (49) with the initial condition determined by (43). Using this method the error made during each integration step Δt is of the order Δt^3.

Consider the digital optimal regulator and tracker (24), (27) based on dynamic optimization. The equivalent discrete-time criterion matrices R_k, Q_k, M_k and L_k, given by (23), (26), except from Φ and Γ, contain a priori known matrices. Φ can be computed recursively, forward in time as described above, and so can Γ, which is computed using (49), with the initial condition determined by (43) where all appearances of V are left out. The equivalent discrete time system matrices (20d) and (20e) are simply the values of Φ and Γ at the end of the integration interval. Again the error made during each integration step Δt is of the order Δt^3.

3.4 Computation of the cost and the recursions that constitute the digital optimal regulator and tracker based on dynamic optimization.

The digital optimal regulator and tracker for stochastic systems and incomplete state information were derived by Van Willigenburg and De Koning (1990b). They can only be derived from dynamic optimization. The problems are certainty equivalent, so the solution to the deterministic problems hold, where the state is replaced by its estimate generated by the discrete-time Kalman one step ahead predictor. Expressions for the cost of the digital optimal regulator and tracker, explicit in the system, criterion and covariance matrices, were derived. These expressions can be easily computed using the previous results. However, for convenience and since the problems are certainty equivalent, in this paper we considered deterministic systems and complete state information. Therefore we here present the deterministic versions of the expressions for the cost of the digital optimal regulator and tracker based on dynamic optimization.. The cost of the digital optimal regulator, for deterministic systems and complete state information is given by

$$J=x_0^T S_0 x_0. \tag{50}$$

The cost of the digital tracker for deterministic systems and complete state information is given by

$$J=x_0^T S_0 x_0-2x_0^T v_0+x_r^T(t_f)Hx_r(t_f)+\sum_{k=0}^{N-1} X_k-(K_k^1 v_{k+1})^T(2T_k^T+\Gamma_k^T v_{k+1})-T_k K_k^2 T_k^T. \tag{51}$$

Finally we will treat aspects concerning the computation of the recursions (24), (27). Equations (24b,c) which equal (27b,e) constitute the solution of a standard discrete-time regulator problem (Van Willigenburg and De Koning, 1990a). The solution is written in the so called Joseph stabilized form which is known for

its good numerical performance (Lewis, 1986, Bierman, 1977). Due to computer round-off errors, errors in the computation of the solution of a standard discrete-time regulator problem may arise. In particular the symmetry and positive definiteness of the matrices S_k, k=0,1,2,..,N-1 may be lost. A thorough study on this subject is presented by Bierman (1977). The work considers discrete sequential estimation, however this problem is dual to a standard discrete-time regulator problem. Although the Joseph stabilized form is to be preferred above other forms we should mention here that computational methods based on matrix factorization are superior (Bierman, 1977). We are currently working on their implementation, however the examples presented in section 4 are all computed using the recursions (24) and (27).

4 Numerical examples

The digital optimal regulator and tracker, based on both static and dynamic optimization, have been programmed by the author using PC-Matlab. The software is available on request. In this paragraph we will present a numerical example which demonstrates the superiority of the result obtained from dynamic optimization, considering accuracy. Furthermore, using yet another optimization approach, we will demonstrate the correctness of the result obtained from both static and dynamic optimization, using examples with small horizon. For problems with large horizon the static optimization approach becomes cumbersome (see paragraph 2.4). In fact we ran out of computer memory. The first example however, will deal with the computation of the state transition matrix, which is equally important in both the static and dynamic optimization approach. Two approximations appear in the proposed computation scheme to compute the state transition matrix of system (1a). We actually compute the state transition matrix of system (33), which is an approximation of system (1a), using the approximation (35). However when the time varying system (1a) is commutative the state transition matrix for both systems is the same at times t_i, given by (32) (See paragraph 3.2). So we present an example of a linear time varying system that is not commutative

and of which an analytical expression for the state transition matrix is known (Wu, Horowitz and Dennisson, 1975). Given the linear time varying system (1a) with

$$A(t) = \begin{pmatrix} -3t^2 & 0 \\ 3t^5 & -6t^2 \end{pmatrix}, \tag{52}$$

The state transition matrix of this system is given by (Wu, Horowitz and Dennisson, 1975)

$$\Phi(t,0) = \begin{pmatrix} \exp(-t^3) & 0 \\ t^3\exp(-t^3) & \exp(-t^3) \end{pmatrix}. \tag{53}$$

Figure 4.1a shows the values of the components of $\Phi(t,0)$ computed using the exact solution (53) and the numerical results based on (33), (35) and (38) where Δt, given by (32) equals 0.1333. Figure 4.1b shows the result for Δt equal to 0.02. As can be seen if Δt is sufficiently small the approximation is accurate.

The second example deals with a digital optimal regulator problem for a multi input time-varying system, and a time-varying cost criterion. Consider the system (1) with

$$x_0 = \begin{pmatrix} 10 \\ 10 \end{pmatrix} \tag{54a}$$

$$A(t) = \begin{pmatrix} t^2-1 & 0 \\ 5 & 2(t^2-1) \end{pmatrix}, \tag{54b}$$

$$B(t) = \begin{pmatrix} \sin(3t) & 1 \\ -1 & \cos(3t) \end{pmatrix}, \tag{54c}$$

$$t_{k+1}-t_k=0.5,\ k=0,1,2,..,N-1,\ N=5 \tag{54d}$$

then it can be easily seen that this system is not commutative. Consider furthermore the regulator criterion (2) with

$$Q(t) = \begin{pmatrix} 2+\sin(2t) & 0 \\ 0 & 2+\sin(2t) \end{pmatrix}, \tag{55a}$$

$$R(t) = \begin{pmatrix} 2+\cos(2t) & 0 \\ 0 & 2+\cos(2t) \end{pmatrix}, \tag{55b}$$

$$H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{55c}$$

We solved the digital optimal regulator problem (1), (2), (3) (54), (55) using both static and dynamic optimization, where in both cases Δt, the step size of the numerical integration and the computation of the state transition matrix was taken equal to 0.02. From section 3 we know that dynamic optimization results in a solution in feedback form (24). The minimum cost was computed using (50). Given the feedback and using the equivalent discrete-time system matrices (20d,e) which are part of the solution (24), we simulated the response of the system (1), (54) at the sampling instants and computed the control. The result is presented in the first two columns of table 4.1. Static optimization does not directly result in a solution given in feedback form. It presents the optimal control in an open loop fashion given by equation (17a). The optimal control obtained from static optimization and the corresponding response of system (1), (54) at the sampling instants, as well as the value of the minimum cost, computed from (13b), equals the result obtained from the dynamic optimization approach, within four decimals (See table 4.1). Finally, given the control obtained from dynamic optimization, a Runge Kutta fifth and sixth order numerical integration algorithm IVPRK, from the IMSL library, was used to

numerically integrate the system (1), (54), and the cost (2). This alternative computation allows us to check the system response at the sampling instants, and the value of the minimum cost. Furthermore it permits us to compute the system response in between the sampling instants. All results are summarized in figure 4.2 and table 4.1.

Finally we have used an alternative optimization procedure to check if both static and dynamic optimization actually generate an optimal solution. Given the initial state and the system dynamics (1a,b) the cost (2) is a complex function of the finite dimensional control vector (11). So we can compute the minimum cost using a routine that minimizes a general, possibly discontinues function, of a finite number of variables. We used the minimization routine ACPOL from the IMSL library and given any control (11) we computed the cost, i.e. the function value, using again the Runge Kutta fifth and sixth order numerical integration algorithm. We initiated the minimization with the control generated by the static and dynamic optimization approach, and found no improvement. This clearly indicates that both the static and dynamic approach have actually generated optimal solutions.

Consider the system (1) with

$$x_0 = \begin{pmatrix} 0 \\ 10 \end{pmatrix} \tag{56}$$

and where furthermore (54b,c) hold. Consider the criterion (5) with

$$x_r(t) = 10\,[\sin(t)\ \cos(t)]^T \qquad ,t_0 \leq t \leq t_f, \tag{57}$$

and where furthermore (55) and (3) hold. We solved this digital optimal tracking problem where again Δt equals 0.02 using bo static and dynamic optimization approach. The computations were performed in a way similar to those performed for the

digital optimal regulator problem just described. The cost obtained dynamic optimization was computed from (51). Again the solutions obtained from both approaches are equal within four decimals, and closely match the results obtained from numerical integration. The results are summarized in figure 4.3 and table 4.2. Again no improvement was found using the alternative optimization procedure, which demonstrates that the optimal solution is found.

In paragraph 2.4 the advantages of the result obtained from dynamic optimization were discussed. We will finally show an example that illustrates the superior numerical reliability of the result obtained from dynamic optimization. As mentioned in paragraph 2.4 the dimensions of the matrices constituting the solution of the static optimization approach (13), (17) depend linearly on the horizon of the problem. The matrices in fact reflect the system and cost behavior over the complete time interval $[t_0,t_f]$. Depending on the system dynamics (1) and the cost criteria (2), (4) within $[t_0,t_f]$, these matrices may become ill conditioned, which seriously affects the accuracy of the solution. In case of the dynamic optimization approach the matrices which are involved in the solutions (24), (27) only reflect the system and cost behavior over one sample interval, so will not as easily become ill conditioned. Consider the system (1) with

$$x_0 = \begin{pmatrix} 0 \\ 10 \end{pmatrix} \tag{58a}$$

$$A(t) = \begin{pmatrix} 4t-1 & 0 \\ 5 & 8t-2 \end{pmatrix}, \tag{58b}$$

$$B(t) = \begin{pmatrix} \sin(3t) & 1 \\ -1 & \cos(3t) \end{pmatrix}, \tag{58c}$$

$$t_{k+1}-t_k=0.2,\ k=0,1,2,..,N-1,\ N=10 \tag{58d}$$

and the criterion (5) where (3), (55) and (57) hold. Table 4.4 shows the solutions of this digital optimal tracking problem obtained from static and dynamic optimization. The cost is computed from (13b) and (51) respectively and Δt equals 0.001. Furthermore the results obtained when both solutions are numerically integrated using the Runge Kutta fifth and sixth order numerical integration procedure are shown. Clearly the dynamic optimization procedure performs superior.

5 Conclusions

Two numerical procedures, one based on static the other on dynamic optimization, have been presented to compute the digital optimal regulator and tracker for linear time varying systems. For two examples it was shown that both procedures gave exactly the same result. A third example demonstrated the numerical superiority of the method based on dynamic optimization. Considering computation time, and the use of computer memory, the method based on dynamic optimization is also superior. The method based on static optimization was merely used as a check. Furthermore, using a standard numerical integration algorithm besides an alternative optimization procedure, we demonstrated the accuracy and optimality of the solutions generated by both procedures. It is believed that the digital optimal regulator result permits for the first time the computation of an optimal perturbation controller for non linear systems, controlled by computers, that have to track reference trajectories, e.g. a robot performing a prescribed motion or a batch fermentation process. The linearized dynamics about the trajectory constitute a time-varying system. Until now only the digital regulator for time-invariant systems could be computed. The digital tracker, based on dynamic optimization, has never been considered in the literature. This is remarkable since it can be applied in all situations where a linear system, controlled by a computer, has to track a reference trajectory, e.g. a cartesian type robot performing a prescribed motion.

References

ACKERMANN J., 1985, 'Sampled data control-systems', Springer Verlag.

ASTROM K.J., WITTENMARK B., 1984, Computer controlled systems, Englewood Cliffs, NJ: Prentice Hall

BIERMAN G.J., 1977, Factorization methods for discrete sequential estimation, Academic Press.

COLLATZ, 1966, 'The numerical treatment of differential equations', Springer Verlag.

FRANKLIN, G.F., POWELL D., Digital control of dynamic systems, Addison and Wesley.

DE KONING W.L., 1980, 'Equivalent discrete optimal control problem for randomly sampled digital control systems', Int.J.Sys.Sci., 11, 841-855.

LEVIS A.H., SCHLUETER R.A., ATHANS M., 1971, 'On the behavior of optimal linear sampled data regulators', Int. J. Contr., 13, 343-361.

LEWIS F.L., 1986, Optimal Control, Wiley.

MOLER C., VAN LOAN C., 1978 'Nineteen dubious ways to compute the exponential of a matrix', Siam Review 20, 801-836

NOUR ELDIN H.A., 1971, 'Optimierung linearer regelsysteme mit quadratischer zielfunction', Lect. Notes Oper. Res. Math. Sys., 47.

RAO B.R., GANAPATHY S.,'Linear time-varying systems-state transition matrix', Proc. IEE 126, 1331-1335.

VAN LOAN, C.F., 1978, 'Computing Integrals Involving the Matrix Exponential', IEEE Trans. Aut. Contr., AC-23, 3, 395-404.

VAN WILLIGENBURG L.G., 1989, 'True digital tracking for an orthogo robot manipulator', Proc. ICCON '89, TP 3-5.

VAN WILLIGENBURG L.G., DE KONING W.L., 1990a, 'The digital optimal regulator and tracker for deterministic time-varying systems', submitted for publication.

VAN WILLIGENBURG L.G., DE KONING W.L., 1990b, 'The digital optimal regulator and tracker for stochastic time-varying systems', submitted for publication.

WU M, SHERIF A., 1976, 'On the commutative class of linear time varying systems' Int. J. Contr. 23, 433-444.

WU M., HOROWITZ I.M., DENNISON J.C., 1975, 'On solution, stability and transformation of linear time-varying systems', Int. J. Contr. 22, 169-180.

Table 4.1

	dynamic opt.		static opt.		Runge Kutta 5-6	
time	control	state	control	state	control	state
0.0	-1.9480 -14.1410	10.0000 10.0000	-1.9480 -14.1410	10.0000 10.0000	-1.9480 -14.1410	10.0000 10.0000
0.5	-0.3496 -3.4061	0.0711 8.4691	-0.3496 -3.4061	0.0711 8.4691	-0.3496 -3.4061	0.0711 8.4682
1.0	1.8251 2.6368	-1.6413 4.7682	1.8581 2.6368	-1.6413 4.7682	1.8251 2.6368	-1.6414 4.7672
1.5	-1.0015 1.5790	-1.1337 0.8508	-1.0015 1.5790	-1.1337 0.8508	-1.0015 1.5790	-1.1339 0.8481
2.0		-0.9235 0.0600		-0.9235 0.0600		-0.9239 0.0327
cost	550.4561		550.4561		550.4936	

Table 4.2

	dynamic opt.		static opt.		Runge Kutta 5-6	
time	control	state	control	state	control	state
0.0	-0.8113 0.2843	0.0000 10.0000	-0.8113 0.2843	0.0000 10.0000	-0.8113 0.2843	0.000 10.000
0.5	0.1926 -1.1941	-0.1015 4.2786	0.1926 -1.1941	-0.1015 4.2786	0.1926 -1.1941	-0.1015 4.2783
1.0	6.2027 5.4588	-0.5765 2.3027	6.2027 5.4588	-0.5765 2.3027	6.2027 5.4588	-0.5765 2.3023
1.5	0.0013 3.5828	0.6965 -2.7626	0.0013 3.5828	0.6965 -2.7626	0.0013 3.5828	-0.6963 -2.7646
2.0		5.3612 -2.9382		5.3612 -2.9382		5.3610 -2.9536
cost	368.5580		368.5580		368.5674	

Table 4.3a

	static opt.		RK5-6
time	control	state	state
0.0	-0.9632 -6.4736	0.0000 10.0000	0.0000 10.0000
0.2	1.0974 -4.5575	-1.1818 6.0092	-1.1818 6.0091
0.4	3.2240 -0.7184	-1.9977 3.9991	-1.9977 3.9991
0.6	4.6468 3.0190	-1.8908 2.7031	-1.8908 2.7031
0.8	3.7794 4.3942	-0.9995 1.5027	-0.9995 1.5027
1.0	2.2346 3.0226	0.0670 4.0488	0.0670 4.0478
1.2	2.1357 0.1810	0.6655 -0.2647	0.6655 -0.2651
1.4	2.8524 -2.1026	1.1598 -0.4498	1.1598 -0.4519
1.6	2.3304 -2.4548	1.4748 -0.7348	1.4748 -0.7497
1.8	-0.8097 -1.4447	2.9340 -2.6357	2.9340 -2.7874
2.0		10.1720 -16.6105	-10.1720 -18.7363
cost	-2.1176e+7		640.859

Table 4.3b

	dynamic opt.		RK5-6
time	control	state	state
0.0	-0.9632 -6.4735	0.0000 10.0000	0.0000 10.0000
0.2	1.0974 -4.5575	-1.1818 6.0092	-1.1818 6.0091
0.4	3.2240 -0.7183	-1.9977 3.9991	-1.9977 3.9991
0.6	4.6468 3.0190	-1.8908 2.7032	-1.8908 2.7031
0.8	3.7794 4.3942	-0.9995 1.5029	-0.9995 1.5029
1.0	2.2346 3.0226	0.0699 0.4054	0.0699 0.4053
1.2	2.1357 0.1810	0.6656 -0.2629	0.6656 -0.2627
1.4	2.8524 -2.1026	1.1601 -0.4383	1.1601 -0.4394
1.6	2.3304 -2.4548	1.4755 -0.6482	1.4755 -0.6565
1.8	-0.8097 -1.4447	2.9362 -1.7499	2.9363 -1.8347
2.0		10.1802 -4.1806	-10.1804 -5.3686
cost	418.976		420.454

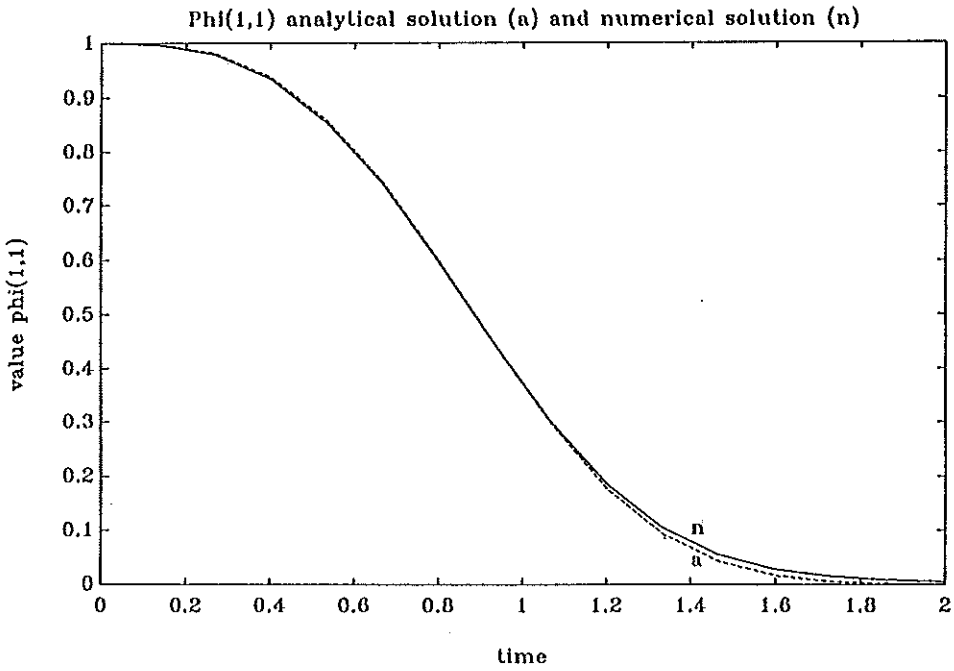
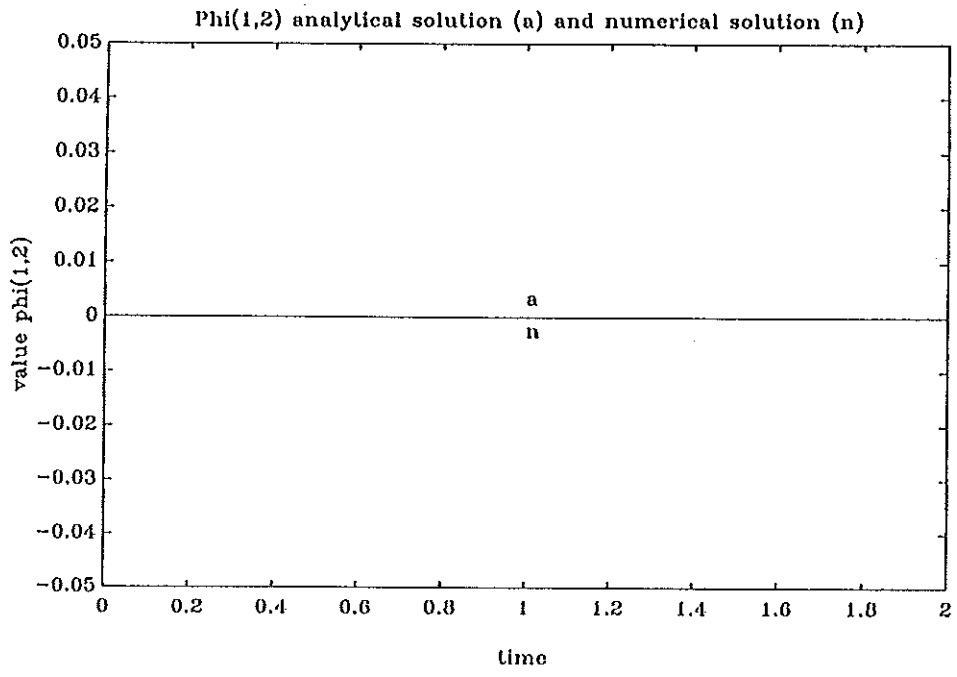

Phi(1,1) analytical solution (a) and numerical solution (n)
value phi(1,1)
time
n
a
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0.2
0.4
0.6
0.8
1.2
1.4
1.6
1.8
2

Figure 4.1a


Phi(1,2) analytical solution (a) and numerical solution (n)
value phi(1,2)
time
a
n
0.05
0.04
0.03
0.02
0.01
0
−0.01
−0.02
−0.03
−0.04
−0.05
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

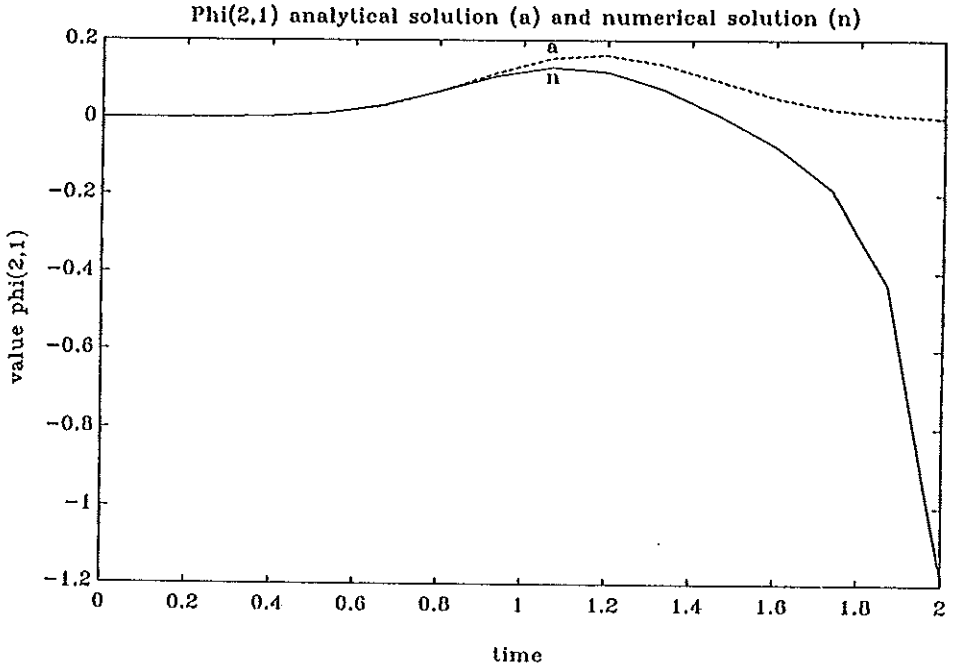
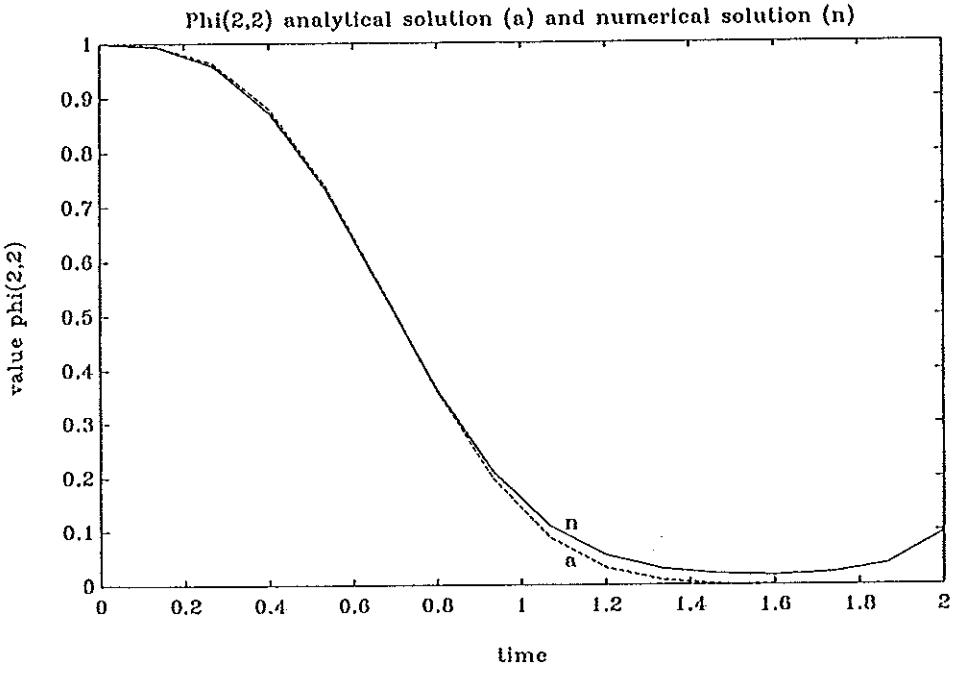

Phi(2,1) analytical solution (a) and numerical solution (n)
value phi(2,1)
time
a
n
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

Figure 4.1a


Phi(2,2) analytical solution (a) and numerical solution (n)
value phi(2,2)
time
n
a
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

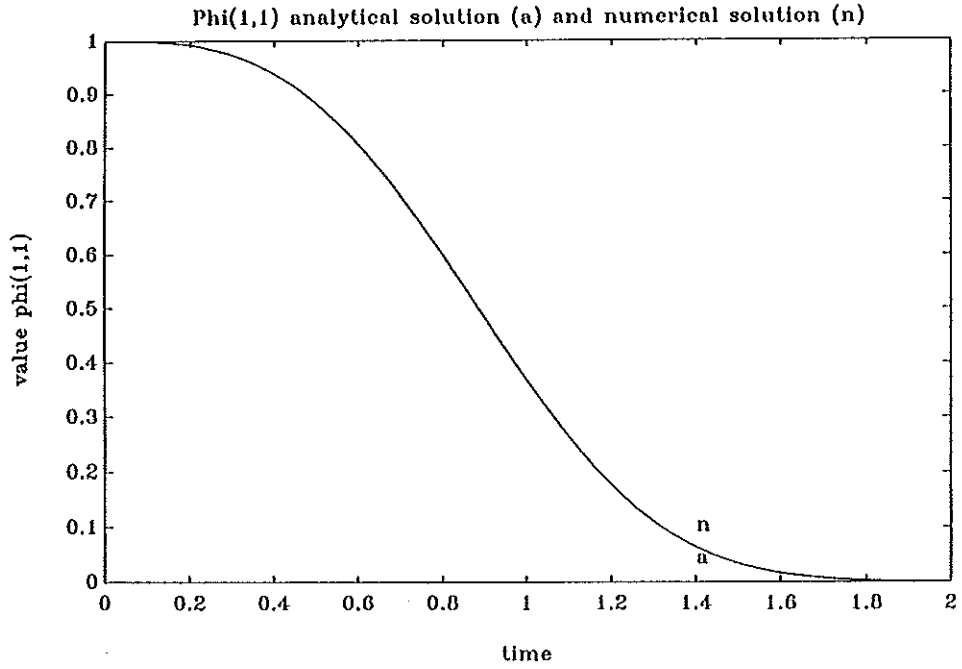
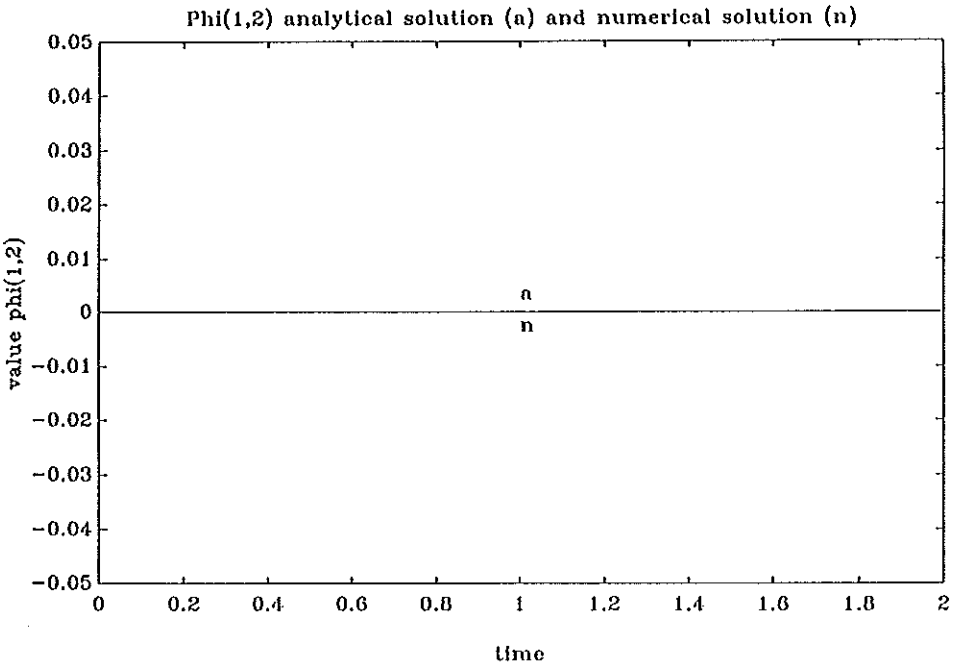

Phi(1,1) analytical solution (a) and numerical solution (n)
value phi(1,1)
time
n
a
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0.2
0.4
0.6
0.8
1.2
1.4
1.6
1.8
2

Figure 4.1b


Phi(1,2) analytical solution (a) and numerical solution (n)
value phi(1,2)
time
a
n
0.05
0.04
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.05
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

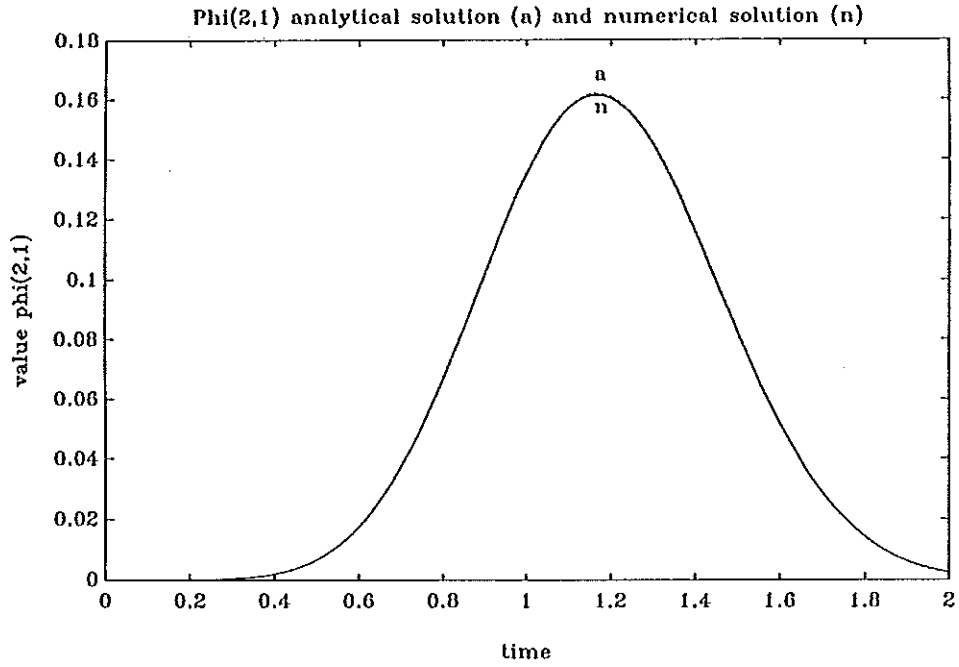
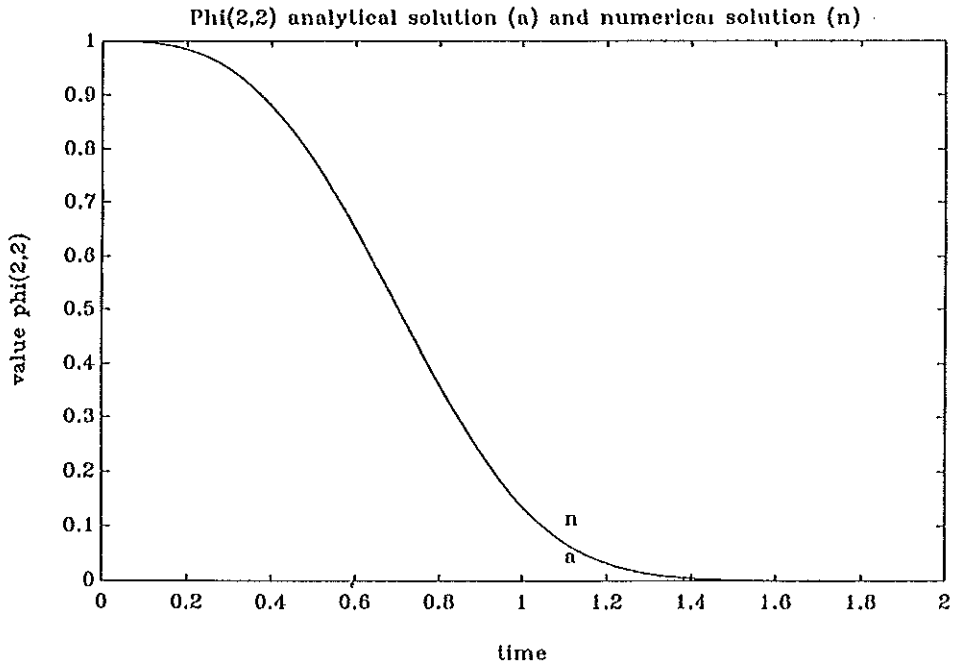

Phi(2,1) analytical solution (a) and numerical solution (n)
value phi(2,1)
time
a
n
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

Figure 4.1b


Phi(2,2) analytical solution (a) and numerical solution (n)
value phi(2,2)
time
n
a
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

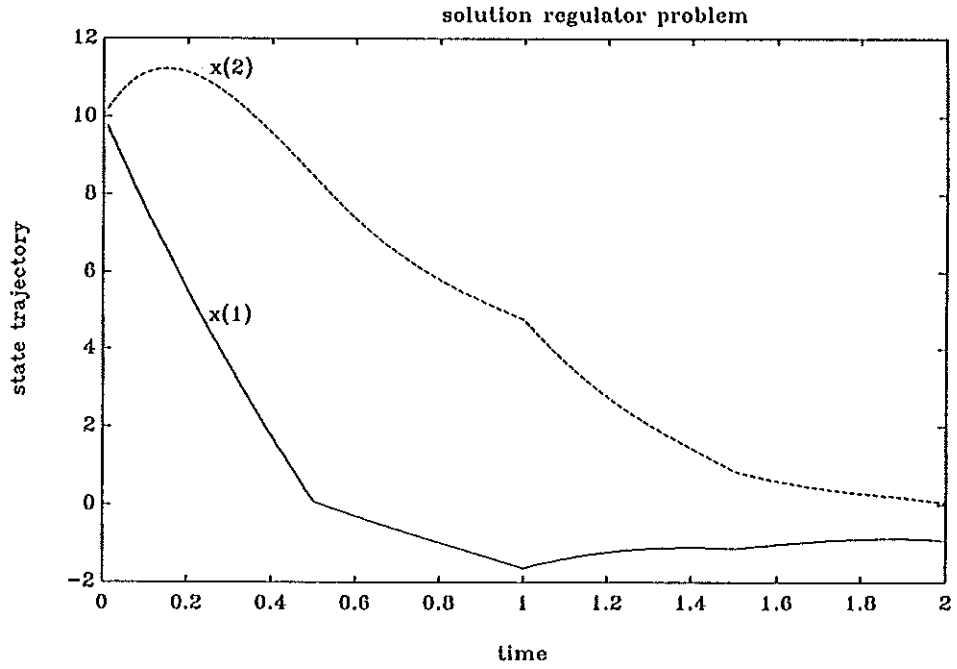
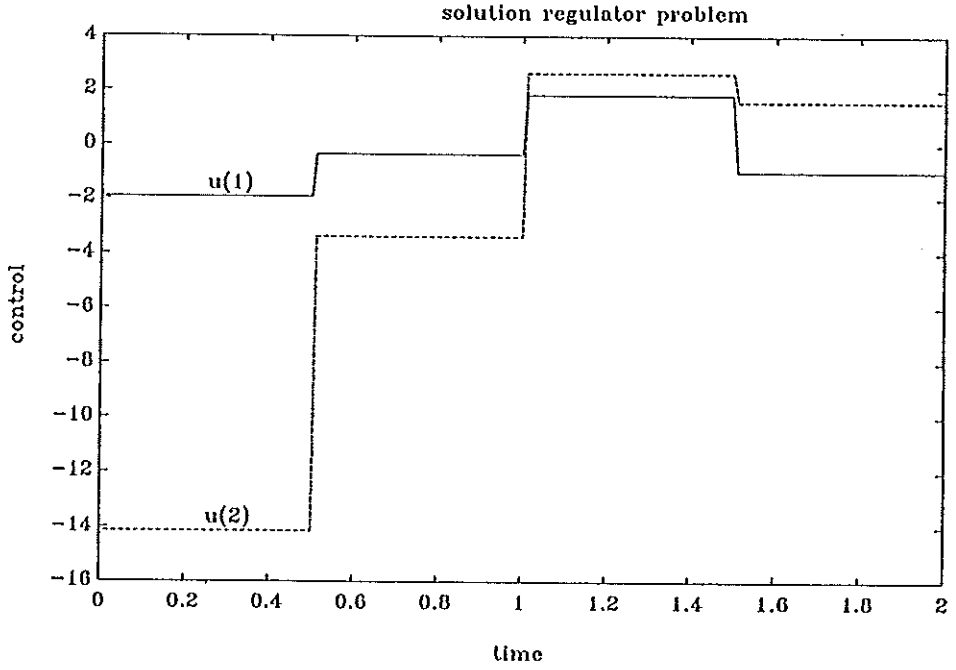

solution regulator problem
x(2)
x(1)
state trajectory
time
12
10
8
6
4
2
0
−2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

Figure 4.2


solution regulator problem
u(1)
u(2)
control
time
4
2
0
−2
−4
−6
−8
−10
−12
−14
−16
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

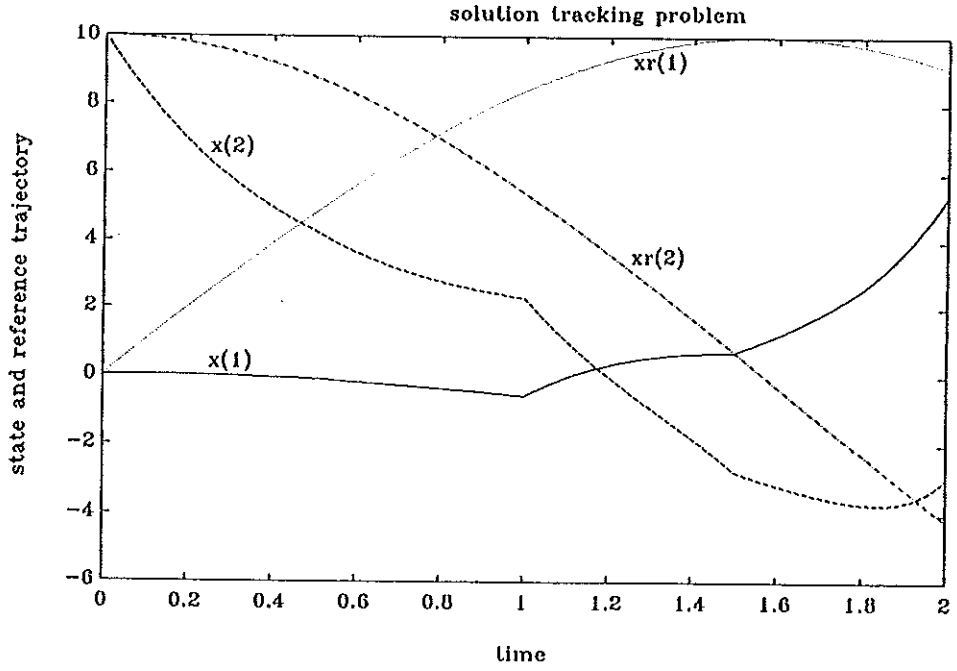
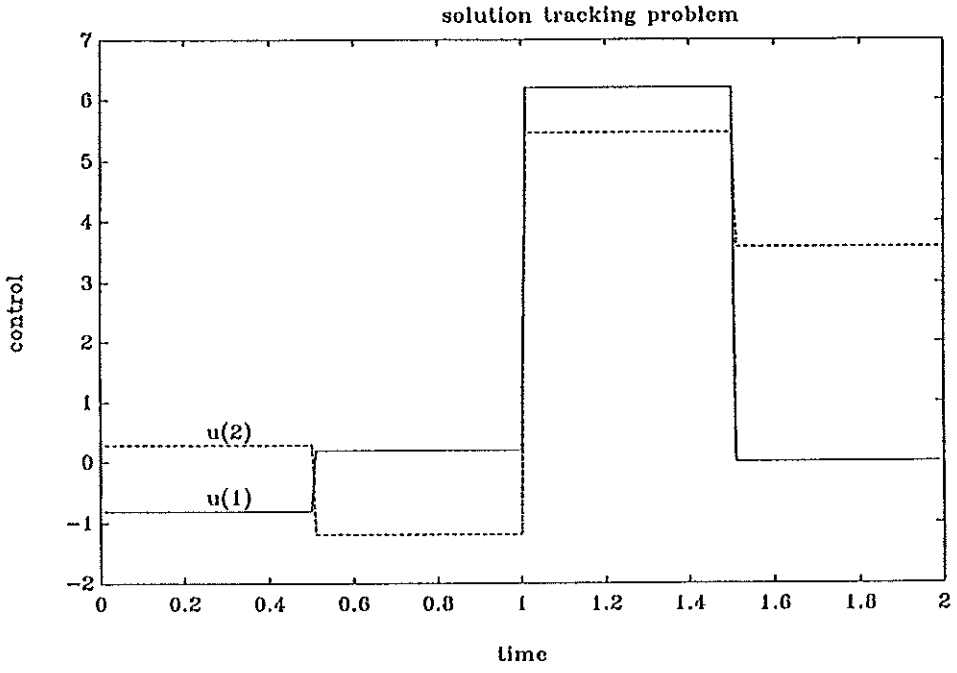

solution tracking problem
state and reference trajectory
time
xr(1)
x(2)
xr(2)
x(1)
10
8
6
4
2
0
-2
-4
-6
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

Figure 4.3


solution tracking problem
control
time
u(2)
u(1)
7
6
5
4
3
2
1
0
-1
-2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

CHAPTER 2

COMPUTATION OF TIME-OPTIMAL MOTIONS FOR AN INDUSTRIAL X-Y ROBOT SUBJECTED TO ACTUATOR AND PATH CONSTRAINTS

Abstract

The time-optimal control problem is solved for an industrial X-Y robot, subjected to two types of actuator constraints, while the motion is constrained to an arbitrary path. A numerical procedure is presented to compute the solution. The robot dynamics include both viscous and coulomb friction. Although not treated in this paper, the extension of the solution and the numerical procedure, to general rigid manipulators, including both viscous and coulomb friction, is straight forward. Numerical solutions involving several paths are presented. The solutions involve a continuous-time state trajectory besides a continuous-time open loop control. Since robots are controlled by digital computers a digital tracking controller is used to track the computed state trajectories. In a companion paper a recently developed procedure to compute digital optimal tracking controllers is applied to the solutions computed in this paper, and experimental results with the implemented digital time-optimal controllers are presented.

1. Introduction

Operations performed by robots are often characterized by the fact that the desired robot motion is determined in space, for instance in the case of spraying, cutting, or welding. However freedom is left to how this motion should be executed in time. Since generally one wants the robot to operate as fast as possible, the problem of executing a motion, specified in space, called a path, in minimum time, is of vital practical importance. Furthermore the solution of the problem may be used to select paths, with equal starting and end points, according to their minimum traveling time, in an attempt to solve the minimum-time

robot motion control problem. This problem considers the minimum-time motion of a robot from one configuration into another, while subjected to actuator constraints (Ailon and Langholtz, 1985, Geering et. al., 1986, Sontag and Sussman, 1986, Van Willigenburg, 1990a,b), and possibly gripper, payload and obstacle constraints (Shiller and Dubowsky, 1989).

The problem has been treated by several authors (Shiller and Dubowsky, 1989, Bobrow, Dubowsky and Gibson, 1985, Shin and Mc Kay, 1985), but except for Shin and Mc Kay (1985), who consider viscous friction, friction is not included in the robot dynamics. Furthermore, only one type of actuator constraint is considered. In practice however friction effects often play a significant role in the dynamic behavior of robots, and two types of actuator constraints are generally present, as explained in section 2 and 3. Furthermore a special situation occurs if the path is such that at some part all links, except for one, stand still. This situation has not been given attention except by Shin and Mc Kay (1985). The proof of the time-optimality of the solution in all cases relied on a number of unproved properties of figures. Finally numerical procedures to compute the solution for general paths have not been presented.

We will present a solution to the time-optimal control problem for an industrial X-Y robot, where the motion is constrained to an arbitrary path and *both* types of actuator constraints are considered. The robot dynamics include both *viscous and coulomb friction*. A *numerical procedure* to compute the solution will be presented together with *numerical examples* involving several paths. We will discuss in detail the complications that occur if all of the links, except for one, stand still. Finally we will give a *rigorous* proof of the solution. Although not treated in this paper the extension of the solution and the numerical procedure to general rigid manipulators, where both coulomb and viscous friction may be included in the manipulator dynamics, is straight forward.

The solution to the problem involves a continuous-time state trajectory and a continuous-time open loop control. Since robots are controlled by computers, in a companion paper (Van Willigenburg, 1990d), a recently developed numerical procedure to compute digital optimal tracking controllers (Van Willigenburg, 1990c) is applied to the time-optimal state trajectories computed in this paper. This companion paper furthermore presents results obtained with the *implemented digital time-optimal controllers*, obtained in this way.

2. Actuators and actuator constraints

The industrial X-Y robot that we consider is actuated by DC motors. The dynamics of a DC motor can be represented by

$$U = L\, dI/dt + R\, I + K\, \omega, \tag{1a}$$

$$T = K\, I, \tag{1b}$$

where

L : the induction of the motor (H),
R : the electrical resistance of the motor (Ohm),
I : the motor current (A),
U : the voltage applied to the motor (V),
ω : the rotation speed of the motor (rad/s),
T : torque generated by the motor (Nm/rad),
K : Voltage constant of the motor (Vs/rad).

From both a practical and a modeling viewpoint it will turn out convenient to chose the motor-currents to be the control variables of the robot. The capabilities of a DC-motor are mainly limited by the heat generation and dissipation characteristics. The heat generation is represented by the second term in (1a), so is proportional to the motor-current. DC-motor controllers are build around a motor-current controller, which is used to limit the

motor-current to prevent overheating. If we consider the current controller to be ideal, which is a reasonable assumption (Van Willigenburg, 1990d), and send our actual control signals to the inputs of the current controllers we may consider the motor-currents to be the control variables. An advantage of this approach is that we only need the motor-current controller and not other parts.

Very often, for instance in all references cited, dynamic models of robots consider forces and/or torques applied to the robot links to be the control variables. From (1) it can be seen that the motor-current is proportional to the torque. Since we assume the transmission from the DC-motor to the robot to be ideal, the force applied to each robot link is proportional to the motor-current. Assuming the motor-currents to be the control variables therefore does not affect the structure of the robot dynamics.

We will adopt the assumptions mentioned above and consider the DC motor-currents to be the control variables. One actuator constraint consists of the limitation of the absolute value of the motor-current to prevent overheating

$$|I| \leq I_{max}, \tag{2}$$

where I_{max} is the maximum absolute value of the motor-current, which may be either positive or negative, where overheating will not occur. In practice to control the motor-current the voltage supplied to the DC motor is rapidly switched on and off, which may be considered as varying the voltage supplied to the motor. Clearly the voltage supplied to the motor is constrained to a maximum determined by the value of the voltage supply. So another actuator constrained is given by

$$|U| \leq U_{max} \tag{3}$$

where U_{max} is the maximum absolute value of the voltage supply

which may be either positive or negative. Now in our case, and in fact very often, we may neglect the first term in equation (1). Doing so from (1) it is obvious that

$$|\omega| \leq \omega_m \tag{4a}$$

where

$$\omega_m = U_{max} / K \tag{4b}$$

Since V is used to control the motor-current, from (1) it is also obvious that the extreme values of the motor-current (2) can only be reached as long as

$$|\omega| \leq \omega_{max} \tag{5a}$$

where

$$\omega_{max} = (U_{max} - R\, I_{max}) / K \tag{5b}$$

We will use constraint (5) although it is somewhat more conservative than (4), since this allows for a constant bound (2) for the motor-current, i.e. the control variable. However if the bound (5) is considered to be a serious drawback, increasing the value of the positive and negative voltage supply will increase the bound (5) without violating other DC motor constraints.

Summarizing we will consider the motor-currents to be the control variables of the robot and consider (2) and (5) as the actuator constraints.

3. Dynamic model of the X-Y robot

Very often (Ailon and Langholtz, 1985, Geering et. al., 1986, Bobrow, Dubowsky and Gibson, 1985, Shiller and Dubowsky, 1989) friction effects are neglected in robot dynamics. Experiments with

our X-Y robot have demonstrated that friction effects play a significant role in the dynamic behavior (Van Willigenburg, 1990d). In our robot model friction is represented by both viscous and coulomb friction. We assume that the motion of one link does not influence the motion of the other link. This assumption is valid if the links move perpendicular and if the friction in each link is not influenced by the motion of the other link. Only the latter assumption is not true in case of an X-Y robot. However, accurate modeling of the friction in one link, if the other is not moving, is already impossible because of its partially stochastic nature (Van Willigenburg 1990d). Therefore we assumed independent dynamics for each link and used "average" values for the friction coefficients. With these assumptions the following robot model is obtained,

$$\ddot{x}_p = -v_x \dot{x}_p - c_x \, \mathrm{sign}(\dot{x}_p) + b_x u_x, \tag{6a}$$

$$\ddot{y}_p = -v_y \dot{y}_p - c_y \, \mathrm{sign}(\dot{y}_p) + b_y u_y, \tag{6b}$$

where x_p represents the translation of the x-link with respect to a reference position, y_p represents the translation of the y-link with respect to a reference position, v_x and v_y are the viscous friction coefficients, c_x and c_y the coulomb friction coefficients and finally b_x and b_y the sensitivity coefficients to the control variables u_x and u_y, which are equal to the motor-currents. Both b_x and b_y depend on the mass of the link and the payload which we assume to be known. We may write the dynamics (6) in state-space form,

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ \ddot{x}_p \\ \ddot{y}_p \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -v_x & 0 \\ 0 & 0 & -v_y & 0 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ \dot{x}_p \\ \dot{y}_p \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ b_x & 0 \\ 0 & b_y \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ c_x \mathrm{sign}(\dot{x}_p) \\ c_y \mathrm{sign}(\dot{y}_p) \end{bmatrix}. \tag{7}$$

The actuator constraint (2) in terms of (6) and (7) becomes a constraint on the control

$$|u_x| \leq u_{max}, \tag{8a}$$

$$|u_y| \leq u_{max}. \tag{8b}$$

where

$$u_{max} = I_{max} \tag{8c}$$

The actuator constraint (5) can be transformed into a state constraint on the state variable $\dot{x}_p$. Since we already assumed the transmission from the DC motor to the robot to be ideal, $\dot{x}_p$ is linearly related to ω.

$$\dot{x}_p = c_t \, \omega, \tag{9}$$

where the constant c_t is determined by the transmission from the DC motor to the robot. The constraint (5) is therefore equivalent to

$$|\dot{x}_p| \leq s_{max}, \tag{10a}$$

where

$$s_{max} = \omega_{max} / c_t, \tag{10b}$$

and ω_{max} is determined by (5). The same constraint also holds for $\dot{y}_p$ the speed of the y-link,

$$|\dot{y}_p| \leq s_{max}. \tag{10c}$$

The actuator constraint (8) limits the force (or torque) applied to each link. This limitation will result in limitations of the link speeds and accelerations, as demonstrated in the next section. The state constraint (10), which resulted from the

actuator constraint (5), directly limits the link speeds. Although we considered DC motors, other actuators generally present the same constraints since usually both the torque and the maximum speed of the actuators are bounded.

4. The time optimal control problem.

We consider a time-optimal control problem where the robot dynamics are given by (6) and the state and control constraints by (8) and (10). Furthermore the state is constrained to follow a so called prescribed path, defined by a parameterized curve.

$$x_p = f(\lambda),\ y_p = g(\lambda), \qquad 0 \le \lambda \le \lambda_{max}, \tag{11}$$

where f and g are continuous functions of λ with continuous first derivatives. The path (11) describes the robot motion in space. Many robot applications such as welding, spraying and cutting are characterized by the fact that the desired robot motion in space is known, but freedom is left to how this motion is executed in time. Since in general the objective is to operate as fast as possible we consider the problem of executing the motion in minimum time.

In practice the functions f and g in (10) are seldom directly available. Usually the path is specified as a finite number of coordinate pairs (x_p, y_p) which have to be passed in a given order. In section 6 we will use cubic splines to interpolate the x_p and y_p coordinates. So both f and g will be a piecewise polynomial of order four, and therefore a piecewise analytic function, that possesses a continuous first derivative (De Boor, 1978).

Given the dynamics (6) and the control and state constraints (8), (10) and (11) *the time-optimal control problem comes down to determining* λ *in* (11) *as a non decreasing continuous function of time*, i.e.

$$\lambda(t) \qquad 0 \le t \le t_{max}, \tag{12a}$$

where

$$\lambda(0) = 0, \tag{12b}$$

$$\lambda(t_{max}) = \lambda_{max}, \tag{12c}$$

such that t_{max} is minimal. Once (12) is known the time-optimal trajectory is given by (10) and (12) i.e.

$$x_p = f(\lambda(t)) \qquad 0 \le t \le t_{max}, \tag{13a}$$

$$y_p = g(\lambda(t)) \qquad 0 \le t \le t_{max}. \tag{13b}$$

Since $\lambda(t)$ in (12a) is a non decreasing continuous function of time, minimizing t_{max} is equivalent to maximizing $d\lambda/dt$ at all times t, $0 \le t \le t_{max}$. *The solution to the time-optimal control problem will be based on converting the dynamics* (6) *and the control and state constraints* (8), (10) and (11) *into constraints on* $d\lambda/dt$ *and* $d^2\lambda/dt^2$. At this stage we will adopt the following notations

$$f_1 = df/d\lambda \tag{14a}$$
$$f_2 = d^2f/d\lambda^2 \tag{14b}$$
$$g_1 = dg/d\lambda \tag{14c}$$
$$g_2 = d^2g/d\lambda^2 \tag{14d}$$
$$\dot{\lambda} = d\lambda/dt \tag{14e}$$
$$\ddot{\lambda} = d^2\lambda/dt^2 \tag{14f}$$

The quantity $\dot{\lambda}$ is referred to as the path velocity, since it determines the speed by which the path is travelled in time. Accordingly the quantity $\ddot{\lambda}$ is referred to as the path acceleration. From (13) given the notation (14) we establish the following facts determined by well known rules for differentiation

$$\dot{x}_p = f_1\dot{\lambda}, \tag{15a}$$

$$\ddot{x}_p = f_1\ddot{\lambda} + f_2\dot{\lambda}^2, \tag{15b}$$

$$\dot{y}_p = g_1\dot{\lambda}, \tag{15c}$$

$$\ddot{y}_p = g_1\ddot{\lambda} + g_2\dot{\lambda}^2. \tag{15d}$$

From (7) it is obvious that a state trajectory determined by $x_p(t)$, $y_p(t)$, $t_0 \le t \le t_{max}$ can only be realized if both functions are continuous and have continuous first derivatives. Now from (13) and (15a,c) and the properties of f, g, and $\lambda(t)$ we observe that indeed this is the case. Since the control is bounded, not all state trajectories possessing these properties can actually be realized. However we will show that by adjusting $\lambda(t)$ we can always satisfy the state and control constraints (8), (10), and (11).

From (15) and (6) we obtain

$$f_1\ddot{\lambda} + f_2\dot{\lambda}^2 = -v_x\, f_1\dot{\lambda} - c_x\, \mathrm{sign}(f_1\dot{\lambda}) + b_x\, u_x \tag{16a}$$

$$g_1\ddot{\lambda} + g_2\dot{\lambda}^2 = -v_y\, g_1\dot{\lambda} - c_y\, \mathrm{sign}(g_1\dot{\lambda}) + b_y\, u_y \tag{16b}$$

Note that since $\lambda(t)$ is non decreasing the path velocity $\dot{\lambda}(t)$ is everywhere positive or zero. If we assume f_1 and g_1 everywhere unequal to zero from (16) we obtain

$$\ddot{\lambda} = -f_2\dot{\lambda}^2/f_1 - v_x\,\dot{\lambda} - c_x/|f_1| + b_x\, u_x/f_1 \tag{17a}$$

$$\ddot{\lambda} = -g_2\dot{\lambda}^2/g_1 - v_y\,\dot{\lambda} - c_y/|g_1| + b_y\, u_y/g_1 \tag{17b}$$

The state constraint (10) using (15) may be written as

$$\dot{\lambda} \le |s_{max}/f_1| \tag{18a}$$

$$\dot{\lambda} \leq |s_{max}/g_1| \tag{18b}$$

We are now in a position to restate the time-optimal control problem. The problem comes down to maximizing the path velocity $\dot{\lambda}$ at all t, $0 \leq t \leq t_{max}$ given the constraints (8), (17) and (18). Since f, f_1, f_2, g_1, and g_2 only depend on λ, equation (17) constitutes two differential equations in λ of which the evolution is the same and uniquely determined by $\lambda(0)$ and $\dot{\lambda}(0)$ and the values of the control variables $u_x(t)$ and $u_y(t)$, $0 \leq t \leq t_{max}$. Since the evolution of both differential equations is the same *only one of the control variables can be chosen independently, the other (others) have to be adjusted to stay on the path.* The control constraint (8a) restricts the admissible values of the path acceleration $\ddot{\lambda}(t)$ in (17a),

$$-f_2\dot{\lambda}^2/f_1 - v_x\dot{\lambda} - c_x/|f_1| - |b_x u_{max}/f_1| \leq \ddot{\lambda} \leq -f_2\dot{\lambda}^2/f_1 - v_x\dot{\lambda} - c_x/|f_1| + |b_x u_{max}/f_1|. \tag{19a}$$

The same holds for (8b) and (17b),

$$-g_2\dot{\lambda}^2/g_1 - v_y\dot{\lambda} - c_y/|g_1| - b_y u_{max}/|g_1| \leq \ddot{\lambda} \leq -g_2\dot{\lambda}^2/g_1 - v_y\dot{\lambda} - c_y/|g_1| + b_y u_{max}/|g_1|. \tag{19b}$$

The left and right hand side arguments in (19) besides known constants only depend on λ and $\dot{\lambda}$ and (19) may therefore be written as

$$\alpha_x(\lambda,\dot{\lambda}) \leq \ddot{\lambda} \leq \beta_x(\lambda,\dot{\lambda}), \tag{20a}$$

$$\alpha_y(\lambda,\dot{\lambda}) \leq \ddot{\lambda} \leq \beta_y(\lambda,\dot{\lambda}), \tag{20b}$$

where

$$\alpha_x(\lambda,\dot{\lambda}) = -f_2\dot{\lambda}^2/f_1 - v_x\dot{\lambda} - c_x/|f_1| - b_x u_{max}/|f_1|, \tag{20c}$$

$$\beta_x(\lambda,\dot{\lambda}) = -f_2\dot{\lambda}^2/f_1 - v_x\dot{\lambda} - c_x/|f_1| + b_x u_{max}/|f_1|, \tag{20d}$$

$$\alpha_y(\lambda,\dot{\lambda}) = -g_2\dot{\lambda}^2/g_1 - v_y\dot{\lambda} - c_y/|g_1| - b_y u_{max}/|g_1|, \tag{20e}$$

$$\beta_y(\lambda,\dot{\lambda}) = -g_2\dot{\lambda}^2/g_1 - v_y\dot{\lambda} - c_y/|g_1| + b_y u_{max}/|g_1|. \tag{20f}$$

Equation (20) not only restricts the admissible values for the path acceleration $\ddot{\lambda}$, but also restricts the admissible values of the path velocity $\dot{\lambda}$. If for some λ and $\dot{\lambda}$, $\alpha_x > \beta_y$ or $\alpha_y > \beta_x$ the value of $\dot{\lambda}$ is inadmissible. In general if for some λ the largest α, exceeds the smallest β, $\dot{\lambda}$ is inadmissible. For each λ the admissible values for $\dot{\lambda}$ are determined by the equation

$$\alpha_{max}(\lambda,\dot{\lambda}) - \beta_{min}(\lambda,\dot{\lambda}) \leq 0. \tag{21a}$$

where

$$\alpha_{max} = \text{MAX}(\alpha_x,\alpha_y) \tag{21b}$$

$$\beta_{min} = \text{MIN}(\beta_x,\beta_y) \tag{21c}$$

In (21) MAX refers to the maximum operation applied to the values within the brackets and accordingly MIN to the minimum operation. As just mentioned, for the X-Y robot (21a) is equivalent to

$$\alpha_x - \beta_y \leq 0 \ \wedge \ \alpha_y - \beta_x \leq 0. \tag{22}$$

In general if the robot has n links (21a) is equivalent to n(n-1) inequalities. Using (20) we may rewrite (22) as

$$p_1(\dot{\lambda}) \leq 0 \wedge p_2(\dot{\lambda}) \leq 0, \tag{23a}$$

where

$$p_1(\dot{\lambda}) = -a\,\dot{\lambda}^2 - b\,\dot{\lambda} - c - d, \tag{23b}$$
$$p_2(\dot{\lambda}) = a\,\dot{\lambda}^2 + b\,\dot{\lambda} + c - d, \tag{23c}$$

where

$$a = f_2 / f_1 - g_2 / g_1, \tag{23d}$$
$$b = v_x - v_y, \tag{23e}$$
$$c = c_x/|f_1| - c_y/|g_1| \tag{23f}$$
$$d = (b_x/|f_1| + b_y/|g_1|)\ u_{max}. \tag{23g}$$

Equation (23) contains quadratic inequalities which may be visualized by parabolas (figure 4.1). Now since f_1 and g_1 are continuous the robot is always capable of following a path if it is moving slowly enough. In terms of the path (11) and the motion along the path (13) this implies that each constraint in (23) must allow values of the path velocity $\dot{\lambda}$ within an interval $[0,\varepsilon)$, where ε is a small positive constant. With this in mind we can visualize the two possibilities by which (23) may put constraints on $\dot{\lambda}$. From figure 4.1 we observe that the admissible values of $\dot{\lambda}$, for each λ, may consist of either a single interval, or of two distinct intervals. If we draw the admissible values for $\dot{\lambda}$, against λ, two typical situations shown in figure 4.2 may arise. If the admissible values of $\dot{\lambda}$, for each λ, consist of a single interval, the admissible region in figure 4.2 will contain no "islands of inadmissibility". Otherwise the admissible region will contain at least one "island of inadmissibility". The border $T(\lambda)$ of the admissible region in figure 4.2 according to (21) is determined by

$$\alpha_{max}(\lambda,\dot{\lambda}) - \beta_{min}(\lambda,\dot{\lambda}) = 0. \tag{24}$$

For convenience in the sequel, although sometimes formally incorrect, we will assume the border $T(\lambda)$ to belong to the admissible region. Since in practice (see section 6) we always have to chose conservative bounds for the control variables this assumption is legitimate. Note that the $(\lambda,\dot{\lambda})$ plane is often referred to as the *phase plane* (Shin and Mc Kay, 1985).

The admissible and inadmissible region in figure 4.2 and 4.3

illustrate the bounds on the path velocity $\dot{\lambda}$ caused by the control constraint (8), the dynamics (6) and the state constraint (11), as a function of λ. Furthermore the constraints (6), (8) and (11) caused the bounds (20) on the path acceleration $\ddot{\lambda}$. We may represent these bounds in figure 4.3 by introducing

$$\mu = d\dot{\lambda}/d\lambda. \tag{25}$$

The quantity μ in (25) represents the slope of the path velocity evolution as a function of λ. From (25) we have

$$\mu = d\dot{\lambda}/d\lambda = \frac{d\dot{\lambda}/dt}{d\lambda/dt} = \ddot{\lambda}/\dot{\lambda} \tag{26}$$

From (26) we observe that the admissible values of μ at each point in figure 4.3 are bounded by (20). At some points in figure 4.3 this is represented by arrows. The upper arrow at each point represents μ, the slope of the the path velocity evolution, when the path acceleration $\ddot{\lambda}$ is maximized, given the bounds (20). The lower arrow represents μ when the path acceleration $\ddot{\lambda}$ is minimized, given the bounds (20). At each point of figure 4.3 the actual slope of the path velocity evolution should lie between these arrows. Note that on the border $T(\lambda)$ of the admissible region in figure 4.3, the upper and lower arrow exactly overlap, since there only one value of $\ddot{\lambda}$, and because of (26), only one value of μ is allowed. If the only admissible value of μ at some point on $T(\lambda)$ points outwards the admissible region we call this an inadmissible point on $T(\lambda)$, since from this point we always move out of the admissible region.

Definition 4.1

An inadmissible point on $T(\lambda)$, the border of the admissible region, is a point on $T(\lambda)$ which only allows a continuation, forward in the λ domain, outside the admissible region.

Definition 4.2

Any point on $T(\lambda)$ that is not inadmissible is called an *admissible point on $T(\lambda)$*.

In terms of figure 4.3 the time-optimal control problem now comes down to maximizing the path velocity $\dot{\lambda}$, for each λ, $0 \leq \lambda \leq \lambda_{max}$, where $\dot{\lambda}$ should stay inside the admissible region and furthermore everywhere μ via (25) is bounded by (20). Shin and Mc Kay (1985) developed an algorithm to maximize the path velocity $\dot{\lambda}$ in figure 4.3, given $\dot{\lambda}(0)$ and $\dot{\lambda}(t_{max})$, and thereby solved the time-optimal control problem if the bounds (10) are disregarded. However their algorithm and the proof that it generates a time-optimal solution relied on a number of assumptions based on a figure similar to figure 4.3, which were not justified. We will present essentially the same algorithm, however in a different rigorous and compact form, which also allows extension to include the bounds (10). Finally we will present a rigorous proof that the algorithm generates the time-optimal solution. Like Shin and Mc Kay (1985) we will limit the discussion to situations where no "islands of inadmissibility" occur. However, our algorithm can be extended in exactly the same manner as theirs to include situations were "islands of inadmissibility" do occur.

5. Solution to the time optimal control problem

5.1 Solution to the problem with one actuator constraint

In section 4 we presented the time optimal control problem. When we disregard the constraint (10) the problem can be converted into one where the path velocity $\dot{\lambda}$ has to be maximized for all λ, $0 \leq \lambda \leq \lambda_{max}$, where $\dot{\lambda}$ is constrained to be in the admissible region determined by (23) and shown in figure 4.3. Furthermore μ defined in (25), via (26) is constrained by (20), which in figure 4.3 was represented at some points by arrows. The initial and final state of the system (7) have not been prescribed. However the state constraint (11) prescribes half of the state, i.e. the values of

x_p and y_p at the initial and final time. From (15a,c) we observe that the initial and final value of the remaining state variables $\dot{x}_p$ and $\dot{y}_p$ are determined by the initial and final value of the path velocity $\dot{\lambda}$, since we assumed $f_1(0)$, $g_1(0)$, $f_1(\lambda_{max})$ and $g_1(\lambda_{max})$ unequal to zero. Usually we want the robot to stand still at the beginning and end of the path. Therefore we prescribe the initial and final path velocity to be zero. However without loss of generality we may prescribe the initial and/or final path velocity unequal to zero.

The solution of the problem is based on the following rule. When travelling *forward* in the λ domain, obviously to maximize $\dot{\lambda}$ we should *maximize* μ, given by (25). From (26) we observe that μ is maximized by maximizing the path acceleration $\ddot{\lambda}$. Therefore the curve so constructed is called an *acceleration curve*. Travelling *backwards* in the λ domain obviously to maximize $\dot{\lambda}$ we should *minimize* μ. Accordingly the curve so constructed is called a *deceleration curve*. Note that travelling forward and backwards in the λ domain is equivalent to travelling forward and backwards in time. When started at some point in the admissible region an acceleration curve is no longer of interest if it leaves the admissible region which happens at an inadmissible point on $T(\lambda)$. It may happen that an acceleration curve does not leave the admissible region. However then it is no longer of interest when λ exceeds λ_{max}.

Definition 5.1

An *acceleration curve* is a curve started anywhere inside the admissible region which travels *forward* in the λ domain while μ is *maximized*. It ends at either the first inadmissible point on $T(\lambda)$ it meets or at a point for which $\lambda=\lambda_{max}$.

By inspection of (20) and (21) an acceleration curve is characterized by

$$\ddot{\lambda} = \beta_{min} \tag{27}$$

When started at some point in the admissible region a deceleration curve is no longer of interest if it leaves the admissible region which happens at an admissible point on $T(\lambda)$. It may happen that a deceleration curve does not leave the admissible region. However then it is no longer of interest when $\lambda<0$.

Definition 5.2

A *deceleration curve* is a curve started anywhere inside the admissible region which travels *backwards* in the λ domain while μ is *minimized*. It ends at either the first admissible point on $T(\lambda)$ it meets or at a point for which $\lambda=0$.

By inspection of (20) and (21) a deceleration curve is characterized by

$$\ddot{\lambda} = \alpha_{max} \tag{28}$$

The next two lemmas are basic to the proof that the algorithm to be presented in this paragraph, generates the time-optimal solution. These two lemmas, although implicitly assumed, have not been stated in previous work on this subject.

Lemma 5.1

Deceleration curves do not intersect.

Proof

Consider the point where two deceleration curves intersect. The slope of both curves, i.e. the value of μ, at this point is different. However this contradicts the fact that by definition μ on each deceleration curve is uniquely determined by its minimum admissible value at each point.

Lemma 5.2

An acceleration curve and a deceleration curve have at most one point in common. If this point is not the end point of the deceleration curve, on the right of this point the acceleration curve lies above, on the left underneath the deceleration curve.

Proof

Consider $(\lambda_s,\dot{\lambda}_s)$ to be a point of both an acceleration and a deceleration curve. Consider any point on the acceleration curve on the right of this point i.e. $(\lambda_r,\dot{\lambda}_{ar})$, $\lambda_r>\lambda_s$. Consider any point $(\lambda_r,\dot{\lambda}_{dr})$ on the deceleration curve. Now from (25) we have

$$\dot{\lambda}_{ar} = \dot{\lambda}_s + \int_{\lambda_s}^{\lambda_r} \mu_{max}(\lambda,\dot{\lambda})\, d\lambda, \tag{29a}$$

and

$$\dot{\lambda}_{dr} = \dot{\lambda}_s + \int_{\lambda_s}^{\lambda_r} \mu_{min}(\lambda,\dot{\lambda})\, d\lambda, \tag{29b}$$

where $\mu_{max}(\lambda,\dot{\lambda})$ represents the maximum admissible value of μ at each point and accordingly $\mu_{min}(\lambda,\dot{\lambda})$ the minimum admissible value of μ at each point. Since, except on $T(\lambda)$, $\mu_{max}(\lambda,\dot{\lambda})>\mu_{min}(\lambda,\dot{\lambda})$ it follows that $\dot{\lambda}_{ar}>\dot{\lambda}_{dr}$. Now consider any point on the acceleration curve on the left of (λ,λ_s), i.e. (λ_l,λ_{al}), $\lambda_l<\lambda_s$. Consider the point $(\lambda_l,\dot{\lambda}_{dl})$ on the deceleration curve. Now from (25) we have

$$\dot{\lambda}_{al} = \dot{\lambda}_s - \int_{\lambda_l}^{\lambda_s} \mu_{max}(\lambda,\dot{\lambda})\, d\lambda, \tag{30a}$$

and

$$\dot{\lambda}_{dl} = \dot{\lambda}_s - \int_{\lambda_1}^{\lambda_s} \mu_{min}(\lambda,\dot{\lambda})\, d\lambda, \qquad (30b)$$

and since, except on $T(\lambda)$, $\mu_{max}(\lambda,\dot{\lambda})>\mu_{min}(\lambda,\dot{\lambda})$ it follows that $\dot{\lambda}_{ar}<\dot{\lambda}_{dr}$. Finally consider the intersection to be on $T(\lambda)$. By definition 5.2 a deceleration curve ends as soon as it reaches $T(\lambda)$, so besides possibly its starting point, it will contain at most one point of $T(\lambda)$. Because of the previous an acceleration curve cannot intersect both the starting and end points of a deceleration curve so an intersection on $T(\lambda)$ has to be unique as well.

Definition 5.3

A point $(\lambda_1,\dot{\lambda}_1)$ in the $(\lambda,\dot{\lambda})$ plane *precedes* another point $(\lambda_2,\dot{\lambda}_2)$ if $\lambda_1 \leq \lambda_2$ (Note the equality).

Definition 5.4

An acceleration or deceleration curve *precedes* another acceleration or deceleration curve if the end point of the first precedes the end point of the latter (Note that the end point of a deceleration curve precedes its starting point).

Now after application of the following algorithm, which consists of three steps, we obtain the solution to the time-optimal control problem.

Algorithm

1 From the initial values $\lambda=0$ and $\dot{\lambda}=0$ start an acceleration curve. If the acceleration curve ends on $T(\lambda)$ call this point $T(\lambda_a)$. By definition $T(\lambda_a)$ is an inadmissible point on $T(\lambda)$.

2 From the final values $\lambda=\lambda_{max}$ and $\dot{\lambda}=0$ start a deceleration

curve. If the deceleration curve intersects the acceleration curve generated by step 1, it will be proved that this intersection is unique, and the procedure ends. Else the deceleration curve ends at $T(\lambda)$. Call this point $T(\lambda_c)$. By definition $T(\lambda_c)$ is an admissible point on $T(\lambda)$.

3 Search the border $T(\lambda)$ from $T(\lambda_a)$ for the first admissible point $T(\lambda_b)$ on $T(\lambda)$, $\lambda_a<\lambda_b\leq\lambda_c$. From $T(\lambda_b)$ start a deceleration curve. It will be proved that this curve will always intersect exactly one acceleration curve generated by the algorithm so the computation may be stopped once an intersection is found. From $T(\lambda_b)$ start an acceleration curve. If this curve ends at an inadmissible point on $T(\lambda)$ this point becomes the new $T(\lambda_a)$ and this step is repeated. Else this curve crosses the deceleration curve generated by step 2 and the procedure ends.

Theorem 5.1

The set of acceleration and deceleration curves generated by the algorithm uniquely connects the initial and final conditions. This connecting curve alternately consists of acceleration and deceleration parts and constitutes the solution to the time-optimal control problem.

Proof of uniqueness

The algorithm is such that it will never generate the same curve twice. Therefore from lemma 5.2 we know that if an acceleration and a deceleration curve generated by the algorithm intersect, this intersection is unique. The procedure ends if an acceleration curve intersects the deceleration curve generated by step 2. This acceleration curve is either generated by step 1 or step 3. If the acceleration curve generated by step 1 intersects the deceleration curve generated by step 2 these curves uniquely connect the initial and final conditions. If the deceleration curve generated by step 2 is intersected by an acceleration curve generated by step 3, this acceleration curve started at $T(\lambda)$, where it was

connected to a preceding deceleration curve. The sub proof below proves that by the nature of the procedure this deceleration curve always intersects exactly one preceding acceleration curve generated by the algorithm. If this is the curve generated by step 1 again a unique connection is found, else the acceleration curve started at $T(\lambda)$ where it was connected to a preceding deceleration curve and we start all over again. Thereby we have proved that the the acceleration and deceleration curves generated by the algorithm uniquely connect the initial and final conditions.

Sub proof

Note that by definition a deceleration curve can only intersect preceding acceleration curves. Consider a deceleration curve with starting point $T(\lambda_1)$. This deceleration curve can only end at a point which precedes λ_2, where λ_2 is the largest value of λ_2 such that $T(\lambda_2)$ is a point where a preceding acceleration curve ends. In other words the "latest" preceding acceleration curve ends at $T(\lambda_2)$. This is because by the nature of step 3 all points $T(\lambda)$, $\lambda_2 \leq \lambda < \lambda_1$ are inadmissible points on $T(\lambda)$ and by definition 5.2 a deceleration curve ends at an admissible point on $T(\lambda)$. So either the deceleration curve intersects the "latest" preceding acceleration curve, or it lies everywhere under it. If the latter is the case the "latest" preceding acceleration curve cannot be the curve generated by step 1 since then the boundary condition at $\lambda=0$ is not fulfilled. So the acceleration curve generated by step 1 must be intersected and since it has no preceding acceleration curves except itself, the deceleration curve cannot intersect any other preceding acceleration curves. Assume the deceleration curve intersects a preceding acceleration curve which is not generated by step 1 and starts at $T(\lambda)$, say $T(\lambda_3)$. If the intersection is on $T(\lambda)$ then this is the point where the deceleration curve ends and therefore it cannot intersect any other preceding acceleration curves. Else by lemma 5.2 on the left of the intersection the deceleration curve lies above the acceleration curve and the deceleration curve must end at some point $T(\lambda_4)$, $\lambda_4 > \lambda_3$. Note that the algorithm is such that the λ intervals covered by different

acceleration curves generated by the algorithm are distinct. Therefore the deceleration curve will not intersect any other preceding acceleration curves. Finally if the deceleration curve lies completely under a preceding acceleration curve it will also be completely under the "latest" deceleration curve that precedes this acceleration curve, since they have the same starting point on $T(\lambda)$ and since deceleration curves do not intersect (lemma 5.1). So the next possibility is an intersection of the next preceding acceleration curve and we start all over again.

Proof of optimality

Let Γ be the unique solution generated by the procedure. Note that Γ can be divided into *sections* consisting of one acceleration curve connected to one deceleration curve. All switching points should be considered part of a deceleration curve. Note furthermore that the admissible region above each section is either bounded by $\lambda=0$, $\lambda=\lambda_{max}$, $T(\lambda)$ or Γ itself. If the solution generated by our procedure is not optimal a curve Γ' should exist of which at least one point say, $(\lambda_0,\dot{\lambda}_0)$ lies above Γ. Now consider the section of Γ which contains λ_0. Now one point in this section must exist where Γ' goes up from Γ and one point where it comes down on Γ since otherwise Γ' violates the boundary conditions or leaves the admissible region. Now either Γ' goes up from and comes down on the acceleration curve of the Γ section, Γ' goes up from and comes down on the deceleration curve of the Γ section, or Γ' goes up from the acceleration curve of the Γ section and comes down on the deceleration curve of the Γ section. However, Γ' going up from the acceleration curve of the Γ section violates the fact that μ on Γ is maximized and Γ' coming down on the deceleration curve of the Γ section violates the fact that μ on Γ is minimized, leaving no opportunities for Γ'.

Figure 4.3 illustrates the fact that the time-optimal solution consists of alternately acceleration (a) and deceleration (d) parts, and that it is unique. According to Shin and Mc Kay (1985), in case "islands of inadmissibility" appear in the phase plane

(figure 4.2b) we compute all acceleration and deceleration curves, i.e. the initial acceleration curve, the final deceleration curve, and curves starting at all points on the borders of the admissible region, where a transition from an admissible to an inadmissible point on the border takes place. Because of lemma 5.2 in this case a unique highest connection of the initial and final conditions exist, again consisting of alternately acceleration and deceleration parts, which constitutes the time-optimal solution. This solution can be found by a backtracking procedure. Finally Shin and Mc Kay (1985) proved the algorithm terminates in a finite number of steps if f and g are piecewise analytic functions. Since we will use cubic spline interpolation to obtain f and g the latter is the case.

5.2 Solution to the problem with two actuator constraints

In the previous paragraph we solved the time optimal control problem disregarding the state constraint (10) which resulted from the actuator constraint (5). Using (15a,c) and since we assumed f_1 and g_1 unequal to zero the state constraint (10) can be converted in a constraint on $\dot{\lambda}$.

$$\dot{\lambda} \leq S(\lambda) \tag{31a}$$

where

$$S(\lambda) = \text{MIN}(s_{max}/|f_1|, s_{max}/|g_1|) \tag{31b}$$

where MIN refers to the minimum operation applied to the arguments within the brackets. Clearly the constraint (31) changes the admissible region in figure 5.1 at points where $S(\lambda)<T(\lambda)$ which is illustrated in figure 5.1. Let us define the border of the new admissible region by $V(\lambda)$ which we again assume to be part of the admissible region. Obviously

$$V(\lambda) = \text{MIN}(S(\lambda), T(\lambda)) \tag{32}$$

Compared with the previous paragraph new situations arise when $V(\lambda)$ is equal to $S(\lambda)$. In this case μ is *not* restricted to a single value on $V(\lambda)$ and three typical situations occur illustrated in figure 5.1. $V(0.14)$ is an inadmissible point on $V(\lambda)$ since the admissible values of μ do not allow a continuation in the admissible region. $V(0.35)$ and $V(0.5)$ are admissible points on $V(\lambda)$ since the admissible values of μ allow a continuation within the new admissible region, where $V(0.35)$ allows a continuation on the border $V(\lambda)$. Obviously since the objective still is to maximize $\dot{\lambda}$, in the case of $V(0.35)$ we want to stay on the border $V(\lambda)$. However in that case μ generally *no longer takes on an extreme value*. If we extend the notion of an acceleration curve the procedure described in the previous paragraph also solves the time-optimal control problem including the constraint (31) when we replace $T(\lambda)$ by $V(\lambda)$.

Definition 5.5

An *acceleration curve* is a curve started anywhere in the admissible region which travels forward in the λ domain while μ is *maximized*, except when on the border $V(\lambda)$. On the border $V(\lambda)$ μ is chosen *to stay on the border* ($V(\lambda)$, $0.25<\lambda<0.5$ in figure 5.1). When this is no longer possible either this is an inadmissible point on $V(\lambda)$, where the acceleration curve ends ($V(0.14)$ in figure 5.1), or the acceleration curve continues inside the admissible region where μ is maximized again ($V(0.5)$ in figure 5.1). If the acceleration curve does not meet an inadmissible point on $V(\lambda)$ it ends at the point where $\lambda=\lambda_{max}$.

Given this definition all lemmas and theorems are still valid where the old border of the admissible region $T(\lambda)$ is replaced by the new border $V(\lambda)$. Only the proof of optimality of theorem 5.1 should be slightly adjusted. An acceleration curve, if it stays on the border $V(\lambda)=S(\lambda)$, is no longer characterized by the maximization of μ. However the going up from an acceleration part of Γ, if it stays on the border $V(\lambda)=S(\lambda)$, still leads to a

contradiction, since we move out of the admissible region in this case. Figure 5.1 illustrates that the solution again consists of alternately acceleration (a) and deceleration (d) parts.

6 Computation of time-optimal solutions

6.1 The numerical algorithm

We have developed software to compute the time-optimal solution where f and g, which constitute the path (11), are obtained by cubic spline interpolation of a set of coordinate pairs

$$(x_k, y_k) \qquad k=0,1,2,..,N, \tag{33}$$

which constitute prescribed values of $x_p(t_k)$ and $y_p(t_k)$, where t_k are undetermined but with the property that $t_{k+1}>t_k$. So (33) constitutes a set of points in the X-Y plane, which have to be passed in a given order. The function f is obtained by cubic spline interpolation of the sequence

$$x(\lambda_k) \qquad k=0,1,2,..,N \tag{34a}$$

where

$$x(\lambda_k) = x_k, \tag{34b}$$

$$\lambda_0 = 0,$$

$$\lambda_{k+1} - \lambda_k = 1/N. \tag{34c}$$

Accordingly y is obtained by cubic spline interpolation of the sequence

$$y(\lambda_k) \qquad k=0,1,2,..,N \tag{35a}$$

where

$$y(\lambda_k) = y_k, \tag{35b}$$

We chose Akima's cubic spline interpolation since it combats wiggles in the interpolant (De Boor, 1978), and wiggles in the path are generally undesirable. We used the routine CSAKM from the IMSL library to perform this interpolation. Figure 6.1 shows the points which were used for the example presented in this section, and the order in which they should be passed. Obviously from figure 6.1 this is just an arbitrary example. We are able to deal with *any* path obtained from a limited set of prescribed coordinate pairs (33).

We will now describe in detail how the computation of acceleration and deceleration curves as well the search for a next admissible point of $V(\lambda)$ is performed numerically. We consider the situation where two actuator constraints are active as described in paragraph 5.2. and where no "islands of inadmissibility" occur, i.e. for each λ we consider the border with the smallest value in figure 4.2b. The computation is largely based on the numerical integration of (27) and (28) forward and backwards in time. $(\lambda_n, \dot{\lambda}_n, t_n)$ will refer to a point in the $(\lambda, \dot{\lambda})$ plane which is reached at time t_n. The numerical integration will be evaluated at equidistant times t_n, i.e.

$$t_{n+1} - t_n = \Delta t, \tag{36}$$

for acceleration curves and the search procedure and

$$t_{n+1} - t_n = -\Delta t, \tag{37}$$

for deceleration curves where Δt is a positive constant. For our application we chose

Δt = 1mS.

The numerical computation of an acceleration curve by definition starts with n=0 so $(\lambda_0,\dot{\lambda}_0,t_0)$ is the starting point of the curve. The computational procedure consists of nine steps.

1 From $(\lambda_n,\dot{\lambda}_n,t_n)$ numerically integrate (27) forward in time and evaluate the result $(\lambda_{n+1},\dot{\lambda}_{n+1},t_{n+1})$.

2 If $\lambda_{n+1}>\lambda_{max}$ stop the acceleration curve at $(\lambda_n,\dot{\lambda}_n,t_n)$.

3 Compute $T(\lambda_{n+1})$, $S(\lambda_{n+1})$ and $V(\lambda_{n+1})$.

4 If $\dot{\lambda}_{n+1}<V(\lambda_n)$, set n=n+1, and goto 1

5 If $V(\lambda_{n+1})=T(\lambda_{n+1})$ stop the acceleration curve at $(\lambda_n,\dot{\lambda}_n,t_n)$.

6 Numerically integrate (28) from $(\lambda_n,\dot{\lambda}_n,t_n)$ forward in time and evaluate the result $(\lambda_{n+1},\dot{\lambda}_{n+1},t_{n+1})$.

7 Compute $S(\lambda_{n+1})$.

8 If $\dot{\lambda}_{n+1}>S(\lambda_{n+1})$ stop the acceleration trajectory at $(\lambda_n,\dot{\lambda}_n,t_n)$.

9 Set $\dot{\lambda}_{n+1}=S(\lambda_{n+1})$, n=n+1 and goto 1.

Step 1, 2 and 3 are the general steps taken. Step 4 involves the usual continuation if we are within the admissible region. Step 5 involves the situation where we reached an inadmissible point on the border $V(\lambda)=T(\lambda)$. Step 6 and 7 are taken if we reached a point on the border $V(\lambda)=S(\lambda)$. Step 8 involves an inadmissible point on the border $V(\lambda)=S(\lambda)$, while step 9 involves the continuation on the border $V(\lambda)=S(\lambda)$.

The numerical computation of a deceleration curve by definition starts with n=0 so $(\lambda_0,\dot{\lambda}_0,t_0)$ is the starting point of the curve. The computational procedure consists of six steps.

1 Numerically integrate (28) from $(\lambda_n, \dot{\lambda}_n, t_n)$ backwards in time and evaluate the result $(\lambda_{n+1}, \dot{\lambda}_{n+1}, t_{n+1})$.

2 If $\lambda_{n+1}<0$ stop the deceleration curve at $(\lambda_n, \dot{\lambda}_n, t_n)$.

3 Compute $V(\lambda_{n+1})$.

4 If $\dot{\lambda}_{n+1}>V(\lambda_n)$ stop the deceleration curve at $(\lambda_n, \dot{\lambda}_n, t_n)$.

5 If λ_{n+1} is inside the λ domain of an acceleration curve search for the nearest computed value λ' in this domain and consider the point $(\lambda', \dot{\lambda}')$ of the acceleration curve. If $\dot{\lambda}_{n+1}>\dot{\lambda}'$ an intersection is found and the deceleration curve ends at $(\lambda_n, \dot{\lambda}_n, t_n)$.

6 Set n=n+1 and goto 1

Step 1, 2 and 3 are the general steps taken. Step 4 involves the reaching of $V(\lambda)$. Step 5 involves the situation where a connection to an acceleration curve is found.

The search for a next admissible point on $V(\lambda)$ is performed by the following procedure consisting of nine steps. By definition we start searching at n=0 so at $(\lambda_0, \dot{\lambda}_0, t_0)$, where $\dot{\lambda}_0=V(\lambda_0)$.

1 Numerically integrate (27) from $(\lambda_n, \dot{\lambda}_n, t_n)$ forward in time and evaluate the result $(\lambda_{n+1}, \dot{\lambda}_{n+1}, t_{n+1})$.

2 If $\lambda_{n+1}>\lambda_{max}$ stop the search.

3 Compute $T(\lambda_{n+1})$, $S(\lambda_{n+1})$ and $V(\lambda_{n+1})$.

4 If $\dot{\lambda}_{n+1}<V(\lambda_{n+1})$, $(\lambda_n, \dot{\lambda}_n)$ is the next admissible point on $V(\lambda)$.

5 If $V(\lambda_{n+1})=T(\lambda_{n+1})$ goto step 10.

6 Numerically integrate (28) forward in time from $(\lambda_n, \dot{\lambda}_n, t_n)$ and evaluate the result $(\lambda_{n+1}, \dot{\lambda}_{n+1}, t_{n+1})$

7 Compute $S(\lambda_{n+1})$.

8 If $\dot{\lambda}_{n+1} < S(\lambda_{n+1})$, $(\lambda_n, \dot{\lambda}_n)$ is the next admissible point on $V(\lambda)$.

9 Set $\dot{\lambda}_{n+1} = S(\lambda_{n+1})$.

10 Set n=n+1 and goto 1.

Step 1, 2 and 3 are the usual steps taken. Step 4 involves a continuation from the border inside the admissible region. Step 6 until 9 are taken if the border $V(\lambda)=S(\lambda)$.

Since we obtain λ and $\dot{\lambda}$ through numerical integration of equation (27) forward and (28) backwards in time we immediately have available $\lambda(t)$ and $\dot{\lambda}(t)$. Via (13) and (15a,c) we therefore have available the state trajectory of the system (7). This state trajectory will be the basis for computing a *digital* controller in the companion paper (Van Willigenburg, 1990d). During the numerical integration the values of the continuous open loop control may be obtained using (17), as long as we do not move along the border $V(\lambda)=S(\lambda)$. In this case (27) or (28) holds and from (20), (21) we observe that always one of the control variables takes on an extreme value. This result is intuitively expected since to stay on the path only one control variable may be chosen independently. If we are moving on the border $V(\lambda)=S(\lambda)$ μ and $\ddot{\lambda}$ are not determined by (27) or (28). In this case the control may be approximately obtained assuming

$$\ddot{\lambda}_n = (\dot{\lambda}_{n+1} - \dot{\lambda}_n)/\Delta t \qquad (38)$$

The routine IVPRK from the IMSL library, which uses a Runge Kutta fifth and sixth order method, is used to perform the numerical integration. However at some intervals error conditions are

obtained since the equations (27), (28) become stiff. Then an Euler integration routine was used where very small steps were taken to insure a reasonable accuracy. We should mention that the use of the Euler routine seriously decreases the computational efficiency. From (20) which via (21) determines α_{max} and β_{min} in (27), (28) we observe that (27), (28) become stiff if f_2 or g_2 becomes large or if both f_1 and g_1 become close to zero.

In chapter 5 we assumed both f_1 and g_1 everywhere unequal to zero. In practice however, often f_1 or g_1 becomes zero at intervals or isolated points. If f_1 becomes zero from (10), (15) we observe that (31) is completely determined by $|g_1|$. From (16a) we obtain

$$\dot{\lambda}^2 = |b_x u_{max}/f_2|, \tag{39}$$

and when g_1 becomes zero (31) is completely determined by $|f_1|$ and from (16b) we have

$$\dot{\lambda}^2 = |b_y u_{max}/g_2|. \tag{40}$$

If $f_1=0$ at an isolated point, f_2 is unequal to zero, (20b) determines the bounds on $\ddot{\lambda}$, (27), (28) are well defined, and from (39) we have

$$T(\lambda) = |b_x u_{max}/f_2|^{1/2}. \tag{41}$$

If only $g_1=0$ at an isolated point, g_2 is unequal to zero, (20a) determines the bounds on $\ddot{\lambda}$, (27), (28) are again well defined, and from (41) we have

$$T(\lambda) = |b_y u_{max}/g_2|^{1/2}. \tag{42}$$

Since we treat the case of isolated points one may wonder if (41), (42) result in a discontinuity of $T(\lambda)$, because this would have serious consequences for the numerical algorithm. However this is not so.

Lemma 5.4

Consider $f(\lambda_0)$ to be an isolated point where $f_1(\lambda_0)=0$. Then

$$\lim_{\lambda\to\lambda_0} T(\lambda) = T(\lambda_0)$$

Proof

From (42) we have

$$T(\lambda_0) = |b_x u_{max}/f_2(\lambda_0)|^{1/2}. \tag{43}$$

Since f is such that f_1 is continuous and $f(\lambda_0)$ is an isolated point where $f_1(\lambda_0)=0$ if $\lambda\to\lambda_0$, $f_1(\lambda)\to 0$. From (16a) given $f_1(\lambda)\to 0$, we obtain

$$(-b_x u_{max}-f_2(\lambda)\dot{\lambda}^2)/|f_1(\lambda)| \le \ddot{\lambda} \le (b_x u_{max}-f_2(\lambda)\dot{\lambda}^2)/|f_1(\lambda)| \tag{44}$$

Obviously if $\dot{\lambda}>|b_x u_{max}/f_2(\lambda)|^{1/2}$ the upper and lower bound both go to either ∞ or $-\infty$ as $f_1(\lambda)\to 0$. Together with (16b) this leaves no admissible values for $\ddot{\lambda}$. If $\dot{\lambda}<|b_x u_{max}/f_2(\lambda)|^{1/2}$ the upper bound goes to $+\infty$ and the lower bound to $-\infty$ as $f_1(\lambda)\to 0$, with (16b) leaving admissible values of $\ddot{\lambda}$.

For an isolated point $g(\lambda_0)$ where $g_1(\lambda_0)=0$ we obtain analogous results. Although $T(\lambda)$ is continuous it may change rapidly around these points, which may cause numerical problems.

Now consider the situation where $f_1=0$ over a closed interval $[\lambda_0,\lambda_1]$. Then for each $\lambda \in (\lambda_0,\lambda_1)$ $f_2(\lambda)=0$ and (16a) imposes no bounds on $\dot{\lambda}$ or $\ddot{\lambda}$. Therefore $T(\lambda)$ does not exist which can be regarded as $T(\lambda)=\infty$. Therefore $V(\lambda)=S(\lambda)$ which by inspection of (10), (15), and (32) is completely determined by $|g_1|$. If λ equals λ_0 or λ_1 this equals the situation of isolated points described previously. For $g_1=0$ over a closed interval we obtain analogous results. From this we observe that the border $V(\lambda)$ at λ_0 and λ_1,

the edges of the interval, may *jump* from $T(\lambda)$ to $S(\lambda)$, generally causing a discontinuity. These discontinuities present one major problem illustrated in figure 6.2. Assume from $V(0.4)$ until the discontinuity of $V(\lambda)$, at $\lambda=0.5$, $V(\lambda)$ consists of inadmissible points on $V(\lambda)$. To obtain a proper solution $V(\lambda)$ *should be defined as the smallest value of* $\dot{\lambda}$ *involved in the jump, which constitutes an admissible point on* $V(\lambda)$ from which an acceleration and deceleration curve should be started. In figure 6.2 therefore $V(0.5)=3.0$ is an admissible point on $V(\lambda)$. It therefore is essential to continuously search the border for the next admissible point on it, i.e. to start the next integration step of the search procedure at the value of λ where the previous step ended. Searching from $V(0.4)$ in figure 6.2, $V(0.48)$ will then be recognized as the first admissible point on $V(\lambda)$, since taking a time step Δt the numerical integration of (27), and possibly (28), ends inside the admissible region. This is symbolically represented by the arrow in figure 6.2. Our numerical algorithm therefore recognizes $V(0.48)$, somewhat to the left of $V(0.5)$, as an admissible point on $V(\lambda)$ so that the discontinuity in $V(\lambda)$ is properly dealt with. In fact we ran into this problem in an early stage of the software development when we searched the border discontinuously.

Finally if simultaneously $f_1=0$ and $g_1=0$ this amounts to a situation where the path demands the robot to stand still as can be seen from (15). Since in this case both (27) and (28) are undefined the algorithm is no longer well defined, so we do not allow this to happen. Note that since the robot is forced to stand still, we may split up the path at this point, and by demanding the final path velocity of the first path, and the initial path velocity of the second path to be zero, we can force the robot to stand still.

Summarizing we do allow f_1 and g_1 to be zero, at intervals or isolated points but not simultaneously. Furthermore since we want to prescribe the initial and final state of the robot (6), from (15a,c) we see that we do demand a path for which $f_1(0)$, $g_1(0)$,

$f_1(\lambda_{max})$ and $g_1(\lambda_{max})$ are unequal to zero. In that case $\dot{\lambda}(0)$ and $\dot{\lambda}(t_{max})$ and (11) uniquely determine the initial and final state of the robot (6).

We have demonstrated that the numerical computation of the algorithm works in all practical situations. The main problem with the numerical computation of the algorithm is that the equations (27), (28) may be stiff at some intervals. This causes the numerical integration to become computationally expensive and less accurate. This problem can be partially overcome by adjusting the parameterisation (34), (35), i.e. by adjusting the values of λ_k.

6.2 Numerical examples

From the path given in figure 6.1 we computed the time-optimal solution presented in figure 6.3. The following parameter values were used which apply to an industrial robot (Van Willigenburg, 1990d) if x_p, y_p, the link translations are expressed in cm., $\dot{x}_p$, $\dot{y}_p$ the link velocities in cm/s and $\ddot{x}_p$, $\ddot{y}_p$, the link accelerations in cm/s^2,

$$v_x = 0, \quad (45a) \qquad v_y = 3.0, \quad (45b)$$
$$c_x = 70, \quad (45c) \qquad c_y = 96, \quad (45d)$$
$$b_x = 70, \quad (45e) \qquad b_y = 88, \quad (45f)$$
$$u_{max} = 5.00, \quad (45g) \qquad s_{max} = 100.0. \quad (45h)$$

The bound (45g) should be chosen as high as possible, on the other hand leaving enough room for control deviations which will occur due to uncertainty and imperfect modeling. It therefore should be experimentally determined and will always be less than the real achievable value.

To check the accuracy of the time-optimal state trajectories, computed according to the numerical algorithm presented in section 5, where the state is available at equidistant times t_k, we computed the time-optimal control from these state trajectories and the robot dynamics (6) using the following approximations

$$\ddot{x}_p(t_k) = (\dot{x}_p(t_{k+1}) - \dot{x}_p(t_k)) / \Delta T, \qquad (46a)$$

$$\ddot{y}_p(t_k) = (\dot{y}_p(t_{k+1}) - \dot{y}_p(t_k)) / \Delta T, \qquad (46b)$$

where

$$\Delta T = t_{k+1} - t_k, \qquad \forall k. \qquad (47)$$

Using (6) and the approximation (46), (47) we obtain for the control

$$u_x(t_k)=1/b_x\left[(\dot{x}_p(t_{k+1})-\dot{x}_p(t_k))/\Delta T+v_x\dot{x}_p(t_k)+c_x\,\mathrm{sign}(\dot{x}_p(t_k))\right], \qquad (48a)$$

$$u_y(t_k)=1/b_y\left[(\dot{y}_p(t_{k+1})-\dot{y}_p(t_k))/\Delta T+v_y\dot{y}_p(t_k)+c_y\,\mathrm{sign}(\dot{y}_p(t_k))\right]. \qquad (48b)$$

The time-optimal controls presented in figure 6.3e,f were computed according to (47), (48) where $\Delta T \approx 10$mS. From figure 6.3e,f we observe that everywhere one control variable takes on an extreme value. This result is expected since from figure 6.3c,d we observe that the velocity bound (45h) is never active and therefore everywhere $\ddot{\lambda}$ takes on an extreme value.

A second numerical example is presented in figure 6.4 and 6.5. The values of the parameters in (45) again hold, except for (45g) which is replaced by

$$u_{max} = 8.00. \qquad (49)$$

The time-optimal control was again computed according to (47), (48) with $\Delta T \approx 10$mS. In this example we do observe the velocity bound (45h) being active. In the companion paper (Van Willigenburg, 1990d) we will treat the design and implementation of digital optimal tracking controllers, based on these two solutions, and present experimental results.

7. Conclusions

We solved the time-optimal control problem for an industrial X-Y robot subjected to two types of actuator constraints, while the motion is constrained to follow an arbitrary prescribed path. We included both viscous and coulomb friction in the robot dynamics. We gave a rigorous proof of the solution which, opposed to earlier work on this subject, does not rely on unproved properties of figures. The actuator constraints that we considered are constraints which generally appear in practice. One of them constitutes a limitation of the forces (torques) applied to the links, the other a limitation of the speed of the links. The latter constraint has not been considered before in the literature. Furthermore for the first time we have presented and demonstrated a numerical procedure to compute solutions for an arbitrary path. Although we considered an X-Y robot the extension of the solution and the numerical procedure to general rigid manipulators, where both coulomb and viscous friction may be included in the manipulator dynamics, is straight forward. Finally we have demonstrated that a "special situation", which may occur in a general problem, does not affect both our solution or the numerical procedure to compute it. This "special situation" generally did not receive attention in the literature. In a companion paper (Van Willigenburg, 1990d) we will present the design and implementation of digital optimal tracking controllers which will be used to track the time-optimal state trajectories computed in this paper.

References

AILON A., LANGHOLTZ G., 1985, 'On the Existence of Time-Optimal Control of Mechanical Manipulators', Journal of Optimization Theory and Applications, 46, 1, 1-20.

BOBROW J.E., DUBOWSKY S., GIBSON J.S., 1985, 'Time-Optimal Control of Robotic Manipulators Along Specified Path', International Journal of Robotic Research, Vol 4, no 3, pp 3-17.

BOOR DE C., 1978, A Practical Guide to Splines, Applied Mathematic Sciences 27, Springer Verlag, New York.

GEERING H.P., GUZELLA L., HEPNER S.A.R., ONDER C.H., 1986, 'Time-Optimal Motions of Robots in Assembly Tasks', IEEE Transactions on Automatic Control, 31, 6, 512-518.

SHILLER Z., DUBOWSKY S., 1989, 'Robot Path Planning with Obstacles Actuator, Gripper, and Payload Constraints', International Journal of Robotic Research, Vol 8, no 6, pp 3-18.

SHIN K.G., Mc KAY N., 1985, 'Minimum-Time Control of Robotic Manipulators with Geometric Path Constraints', IEEE Transactions on Automatic Control, Vol AC-30, no 6, pp 531-541.

SONTAG E.D., SUSSMAN H.J., 1986, 'Time-optimal control of manipulators', IEEE Conf. on Robotics and Automation, San Francisco.

VAN WILLIGENBURG L.G., 1990a, 'First order controllability and the time-optimal control problem for articulated arm robots with friction', International Journal of Control, Vol 51, no 6, pp 1159-1171.

VAN WILLIGENBURG L.G., LOOP R.P.H., 1990b, 'Computation of time-optimal controls applied to rigid manipulators with friction', submitted for publication.

VAN WILLIGENBURG L.G., 1990c, 'Numerical procedures to compute the digital optimal regulator and tracker for time-varying systems', Submitted for publication.

VAN WILLIGENBURG L.G., 1990d, 'Design, computation and implementation of digital optimal tracking controllers for cartesian robots', submitted for publication.

fig 4.1a: Adm. and inadm. values of dlambda/dt for a single lambda

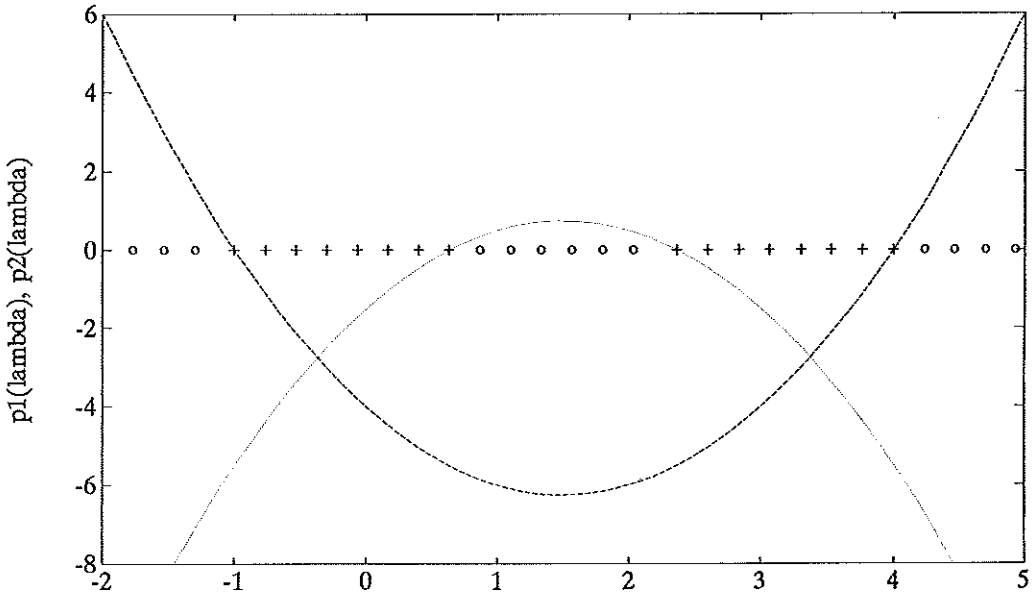


fig 4.1b: Adm. and inadm. values of dlambda/dt for a single lambda

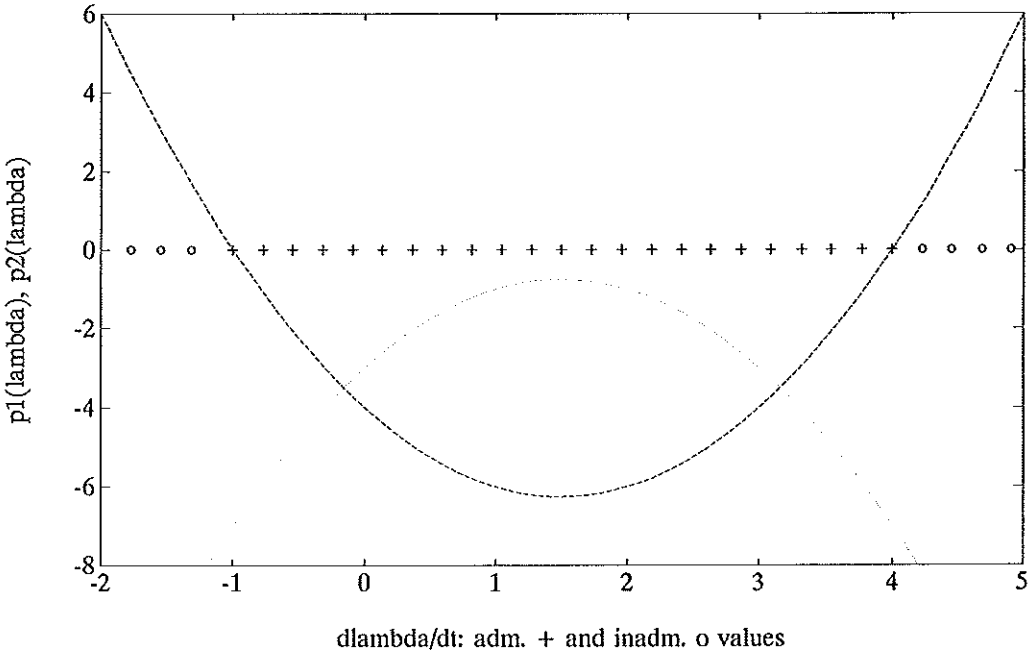


Fig 4.2a: Adm. and inadm. regions in phase plane, no islands

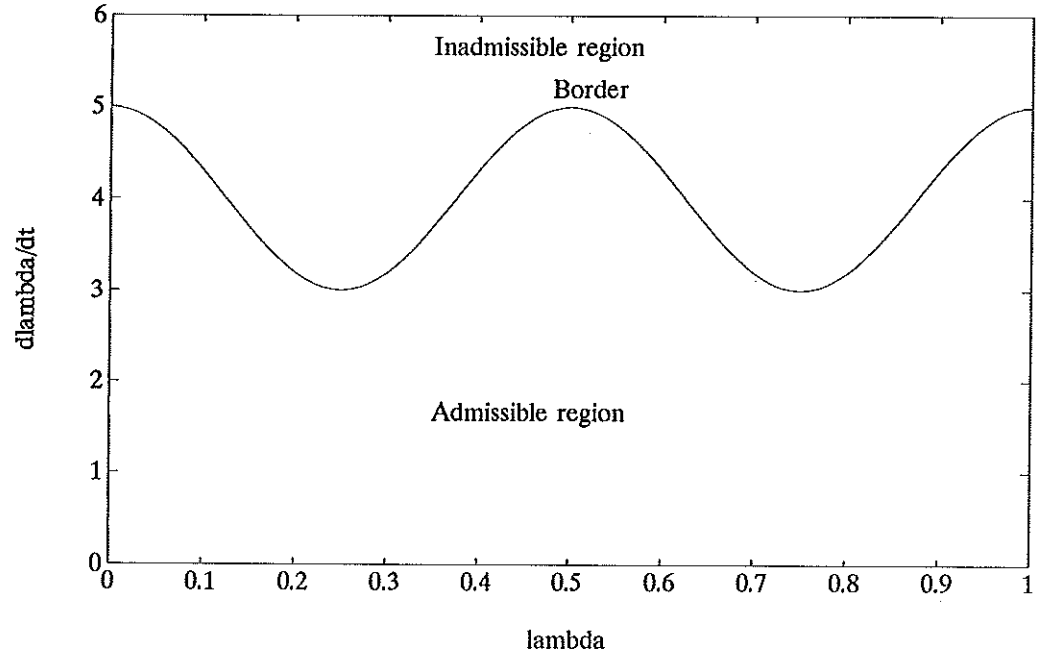


Fig 4.2b: Adm. and inadm. regions in phase plane, island

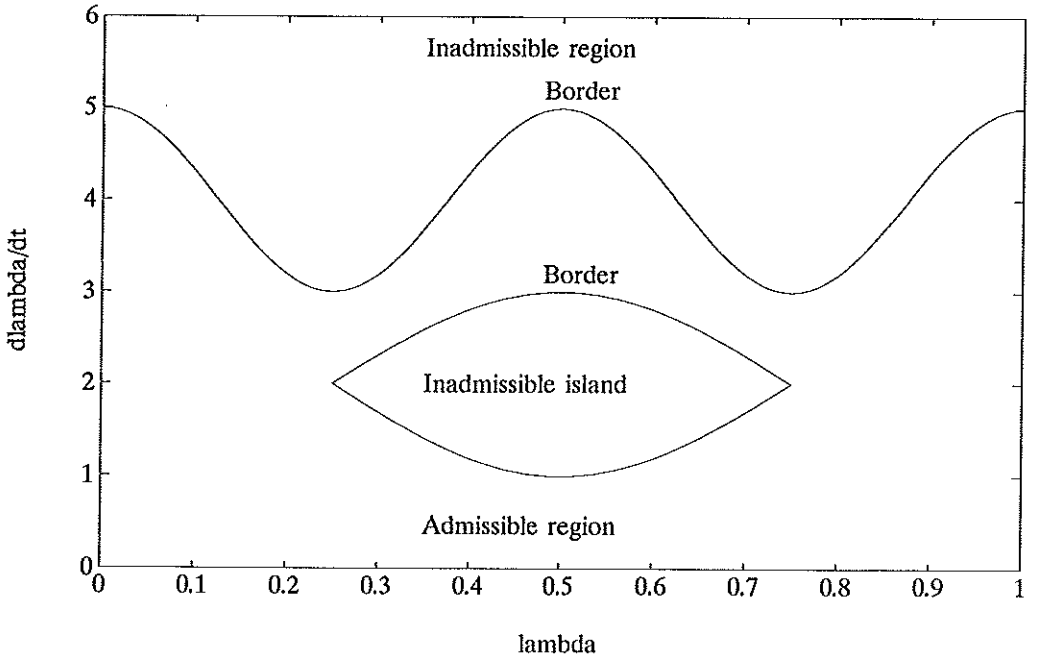


Fig 4.3:Problem and solution in phase plane,1 actuator constraint

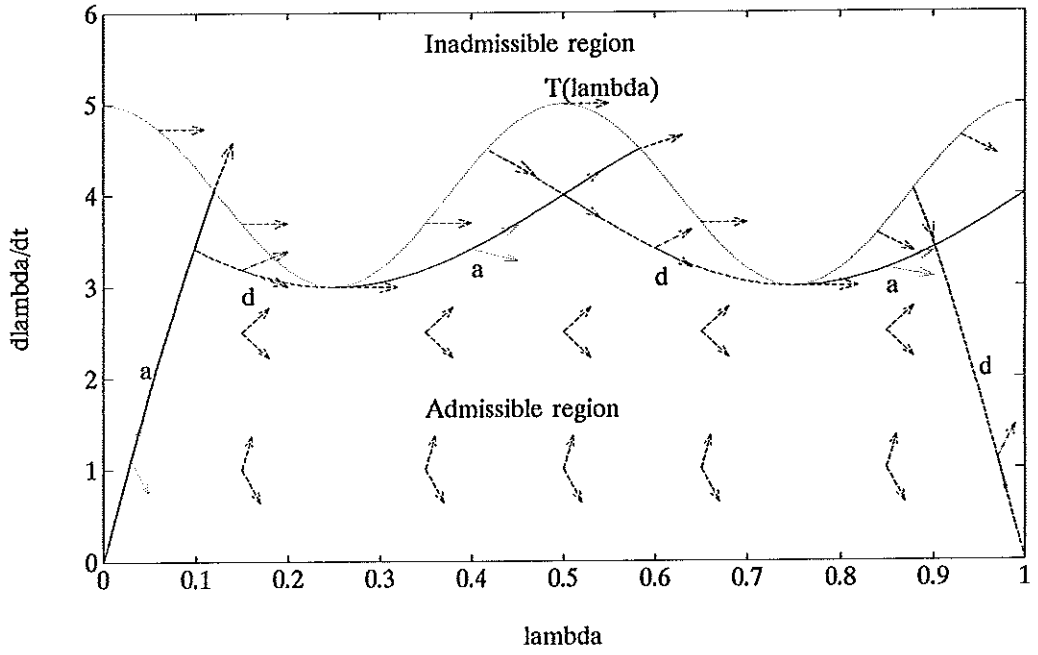


Fig 5.1: Problem and solution in phase plane,2 actuator constraints

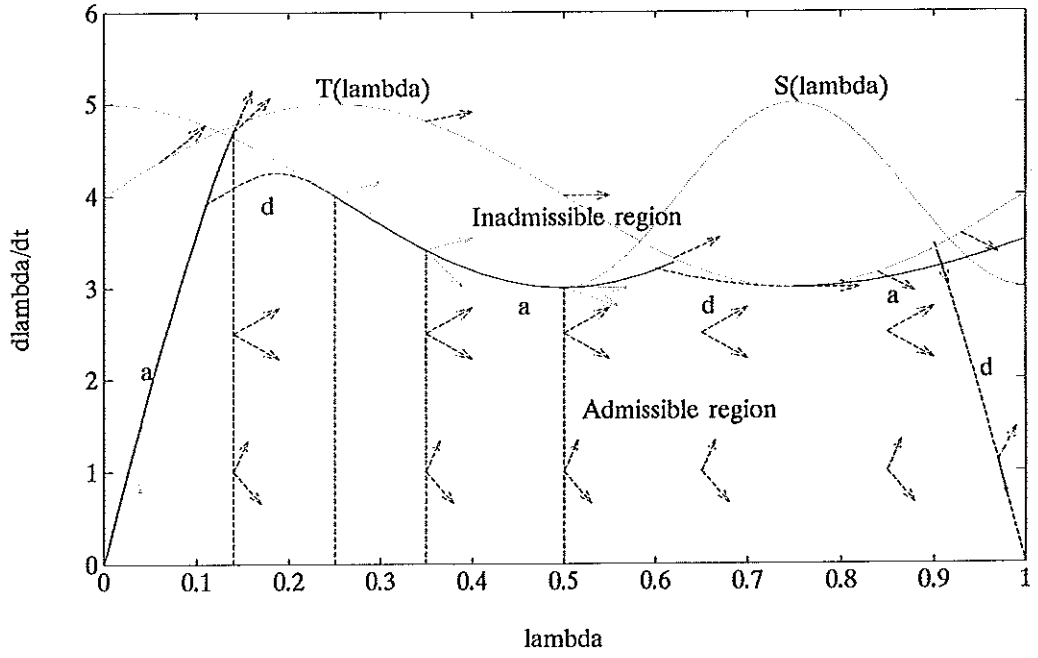


fig 6.1: The path

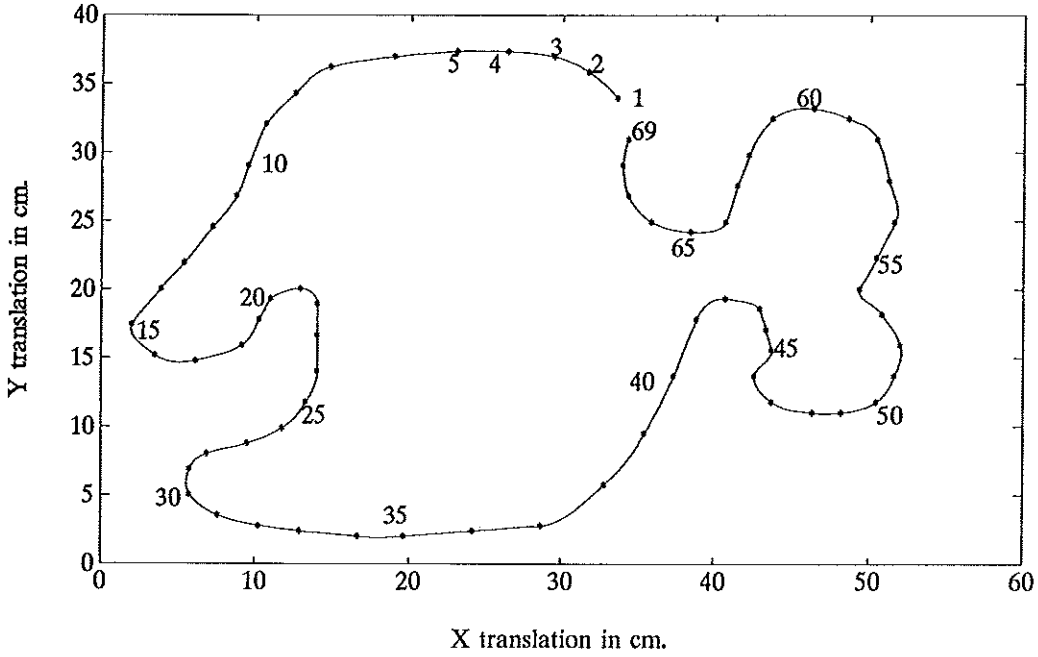


Fig 6.2: Discontinuity in V(lambda)

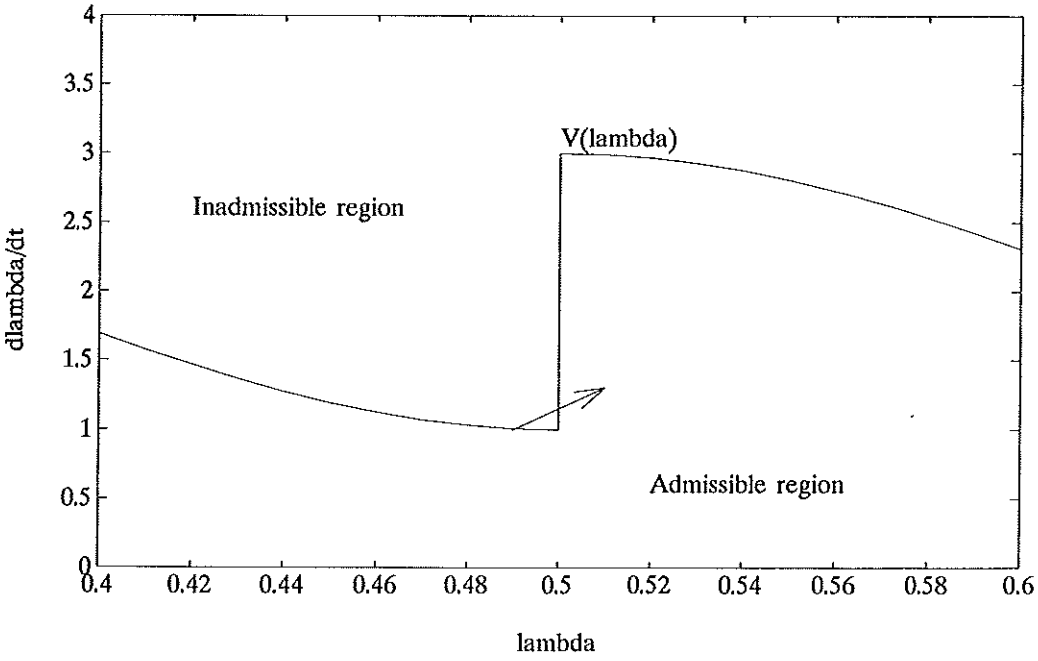


Fig 6.3a: Time optimal solution

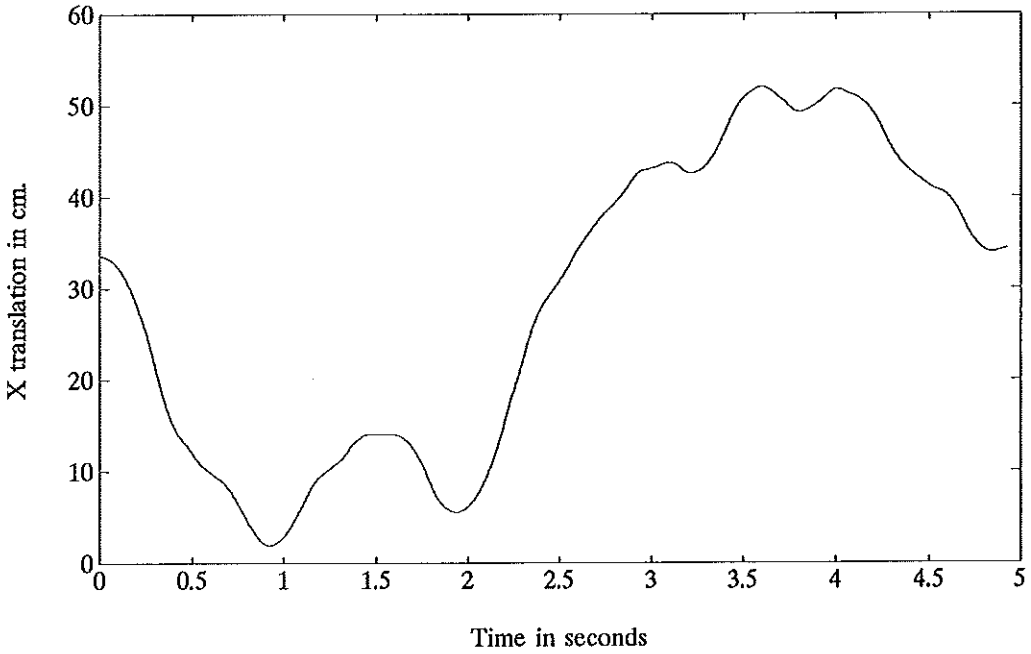


Fig 6.3b: Time optimal solution

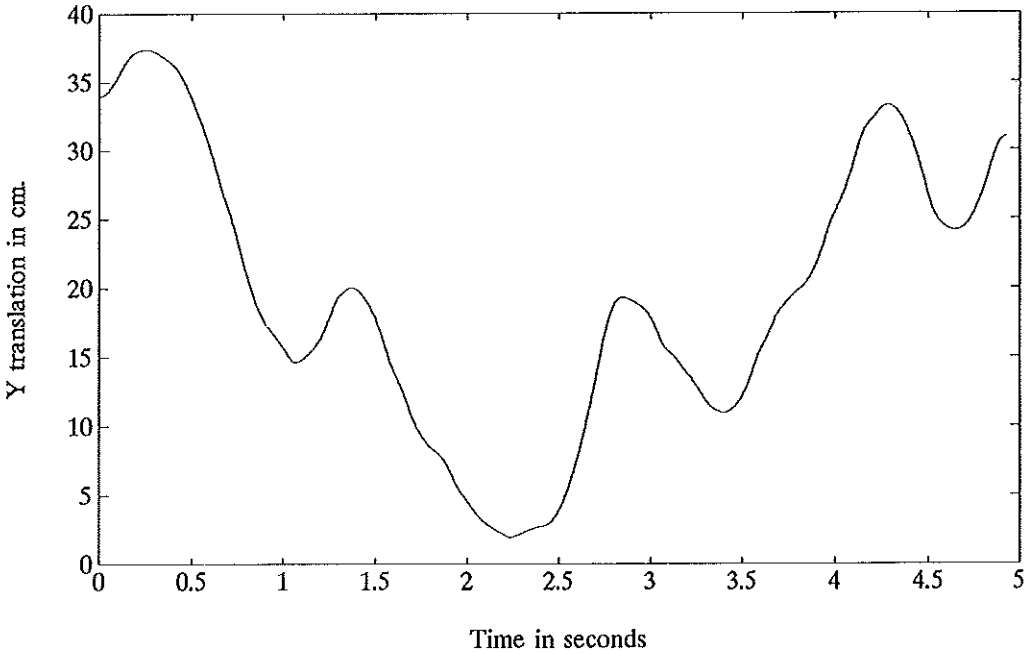


Fig 6.3c: Time optimal solution

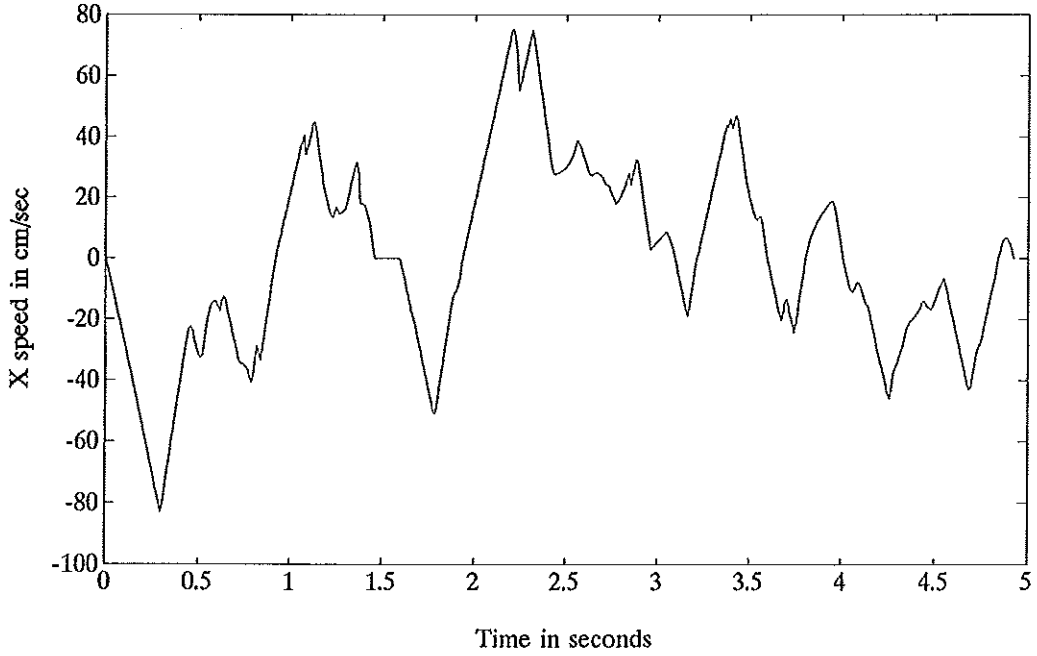


Fig 6.3d: Time optimal solution

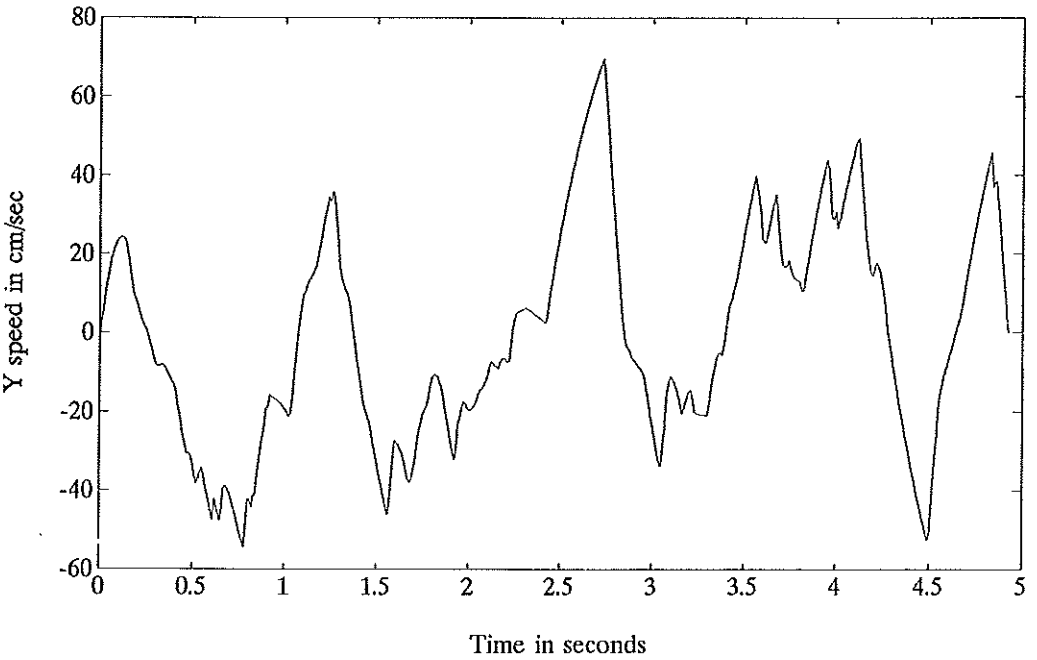


Fig 6.3e: Time optimal solution, control X-link

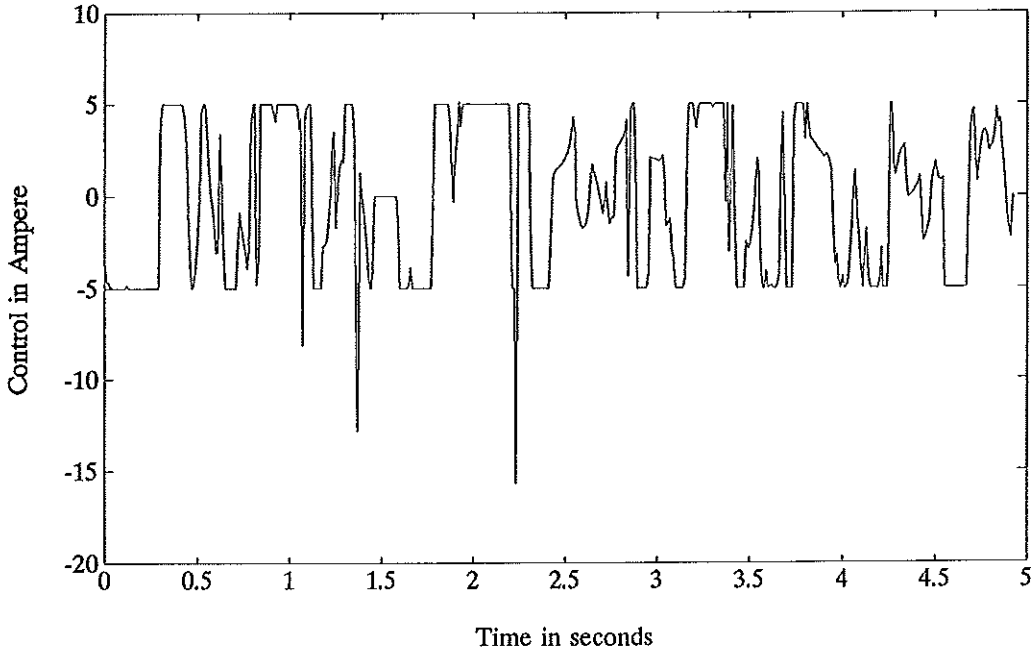


Fig 6.3f: Time optimal solution, control Y-link

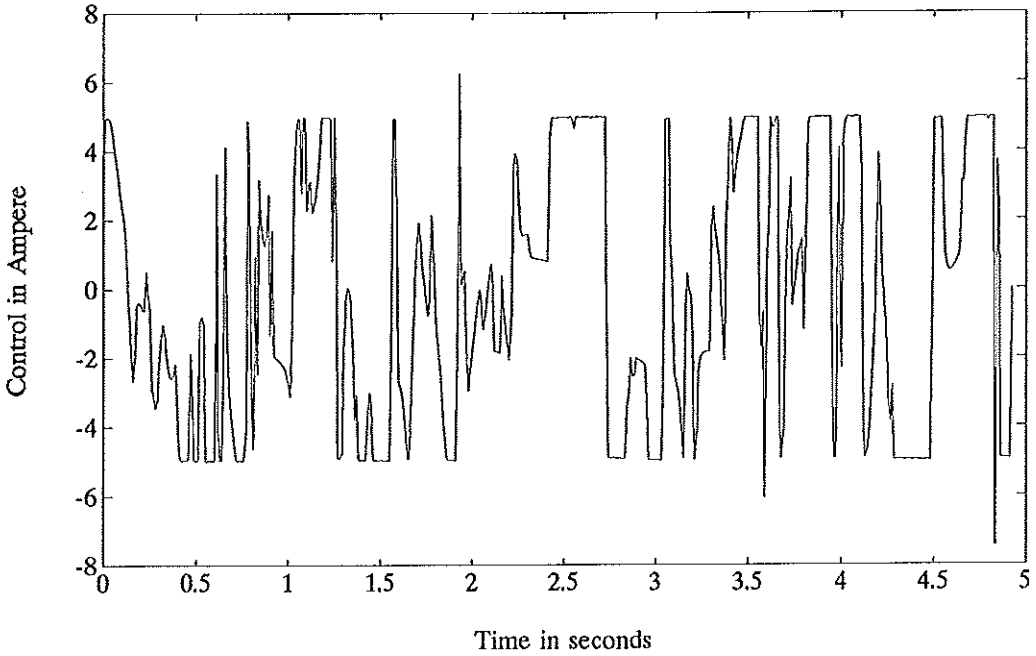


fig 6.4: The path

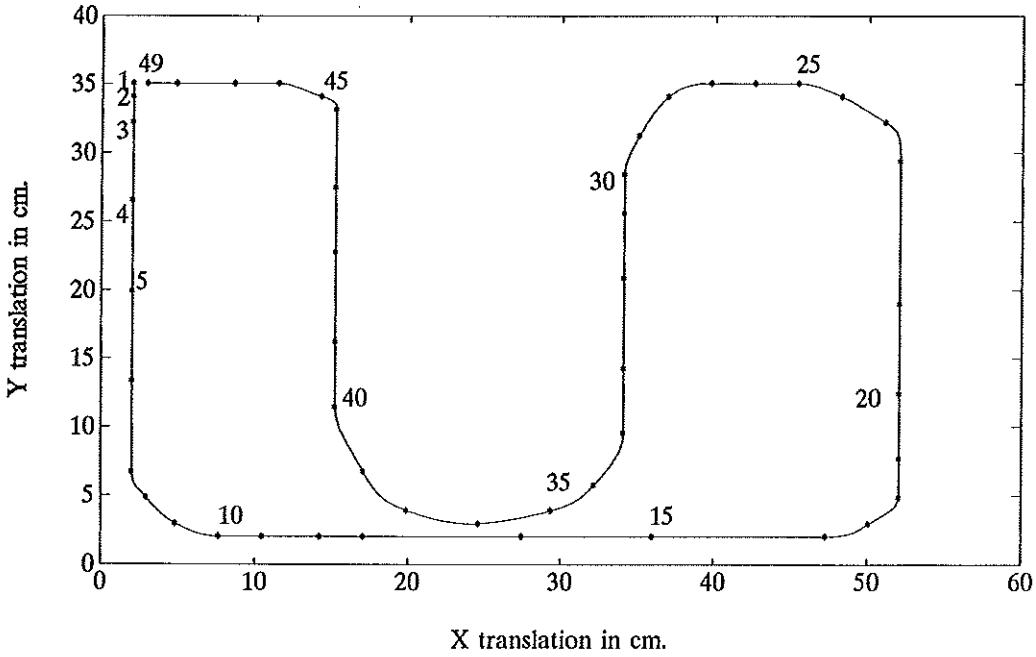


Fig 6.5a: Time optimal solution

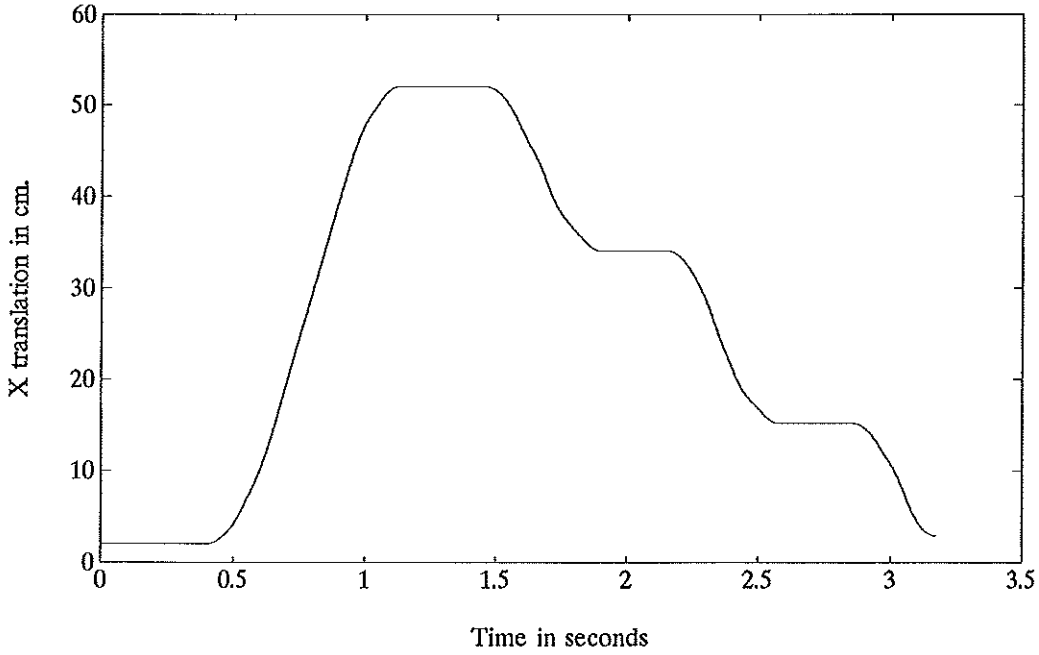


Fig 6.5b: Time optimal solution

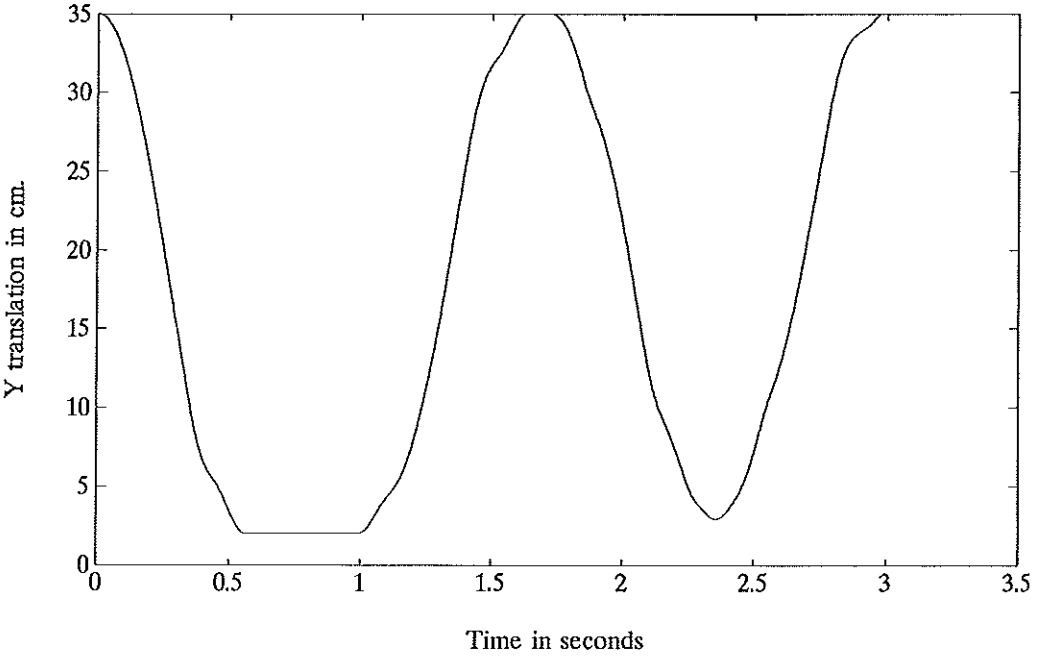


Fig 6.5c: Time optimal solution

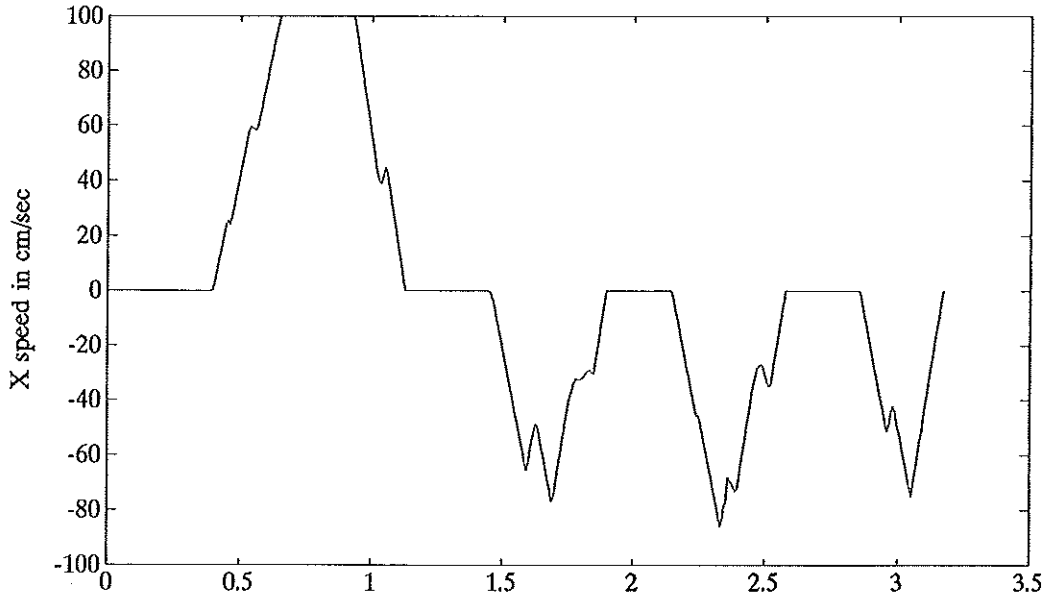


Fig 6.5d: Time optimal solution

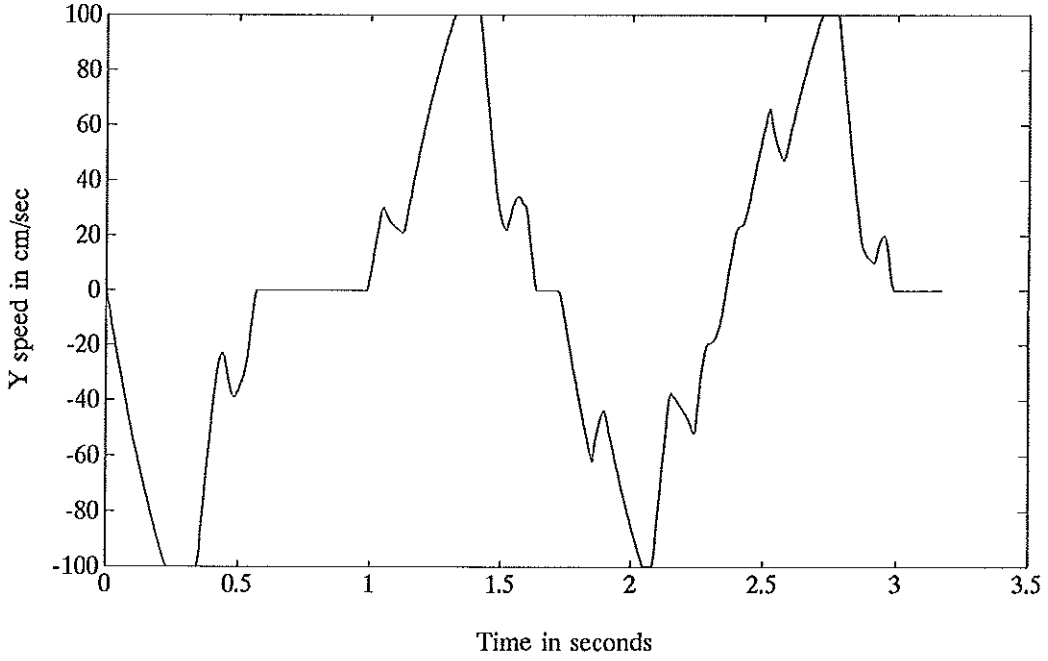


Fig 6.5e: Time optimal solution, control X-link

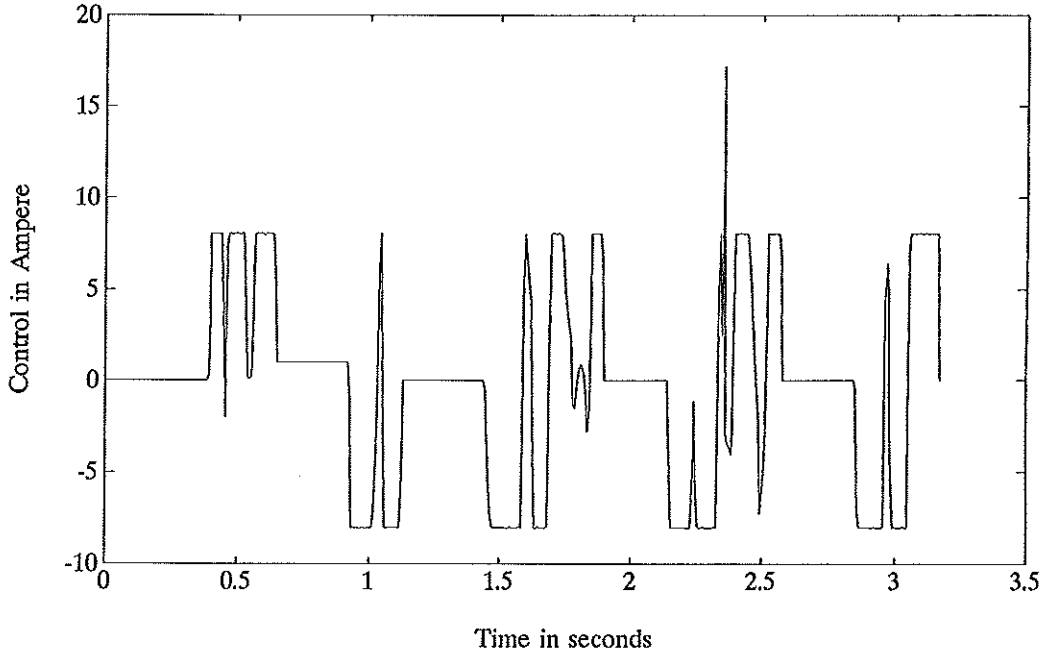
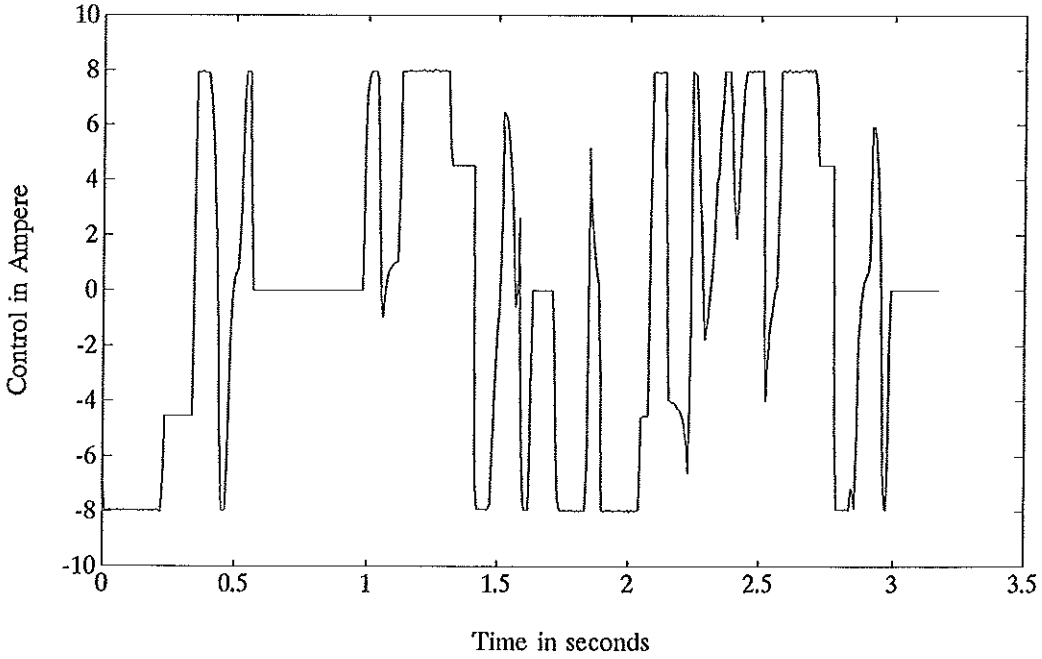


Fig 6.5f: Time optimal solution, control Y-link



L.G. VAN WILLIGENBURG

DESIGN COMPUTATION AND IMPLEMENTATION OF DIGITAL OPTIMAL TRACKING CONTROLLERS FOR CARTESIAN ROBOTS

Abstract

Based on the recently developed digital optimal regulator and tracker for linear time-varying systems we will treat the design, computation and implementation of digital optimal tracking controllers for cartesian robots where friction is modeled by both viscous and coulomb friction and where the influence of gravity is included. The design is characterized by the fact that no approximations are made, i.e. the inter-sample behavior is explicitly considered and therefore "large" sampling times are allowed. Sampling times will vary from 60 to 200mS, while generally sampling times larger than 20mS are considered too large for robot control. Experimental results obtained with *implemented* digital controllers on an industrial cartesian X-Y robot are presented which demonstrate that the accuracy is not seriously affected by the choice of "large" sampling times. Two methods of implementation will be presented. One concerns the "direct" implementation of the digital optimal tracker, which is in feedback form. The other presents the control to the system in an open loop fashion and uses a digital optimal perturbation controller to control deviations from the trajectory. The practical advantage of the latter are discussed.

1. Introduction

The design of digital controllers for cartesian robots has not very often been considered in the literature. Probably this is because cartesian robots have "simple dynamics". Since the links of a cartesian robot move perpendicularly the motion of one link does not influence the motion of the others so the dynamics of the links are decoupled. Furthermore if one neglects friction which is a usual assumption (Asada and Slotine 1986) the dynamics of each

link are determined by a simple second order linear differential equation. However, since cartesian robots are "easy" from the point of view of dynamics and they are also "easy" from the point of view of kinematics, and since the links move parallel to the axes of a cartesian frame, they are "easy to control", i.e. they allow for the use of relatively simple controllers which demand only a relatively small number of on-line computations. So from a practical point of view cartesian robots may be very attractive.

In this paper we will consider cartesian robots where friction is modeled by both viscous and coulomb friction and where we also consider the effect of gravity. We will present a procedure to design digital optimal tracking controllers for such robots. Throughout the paper we assume the desired robot motion as a function of time, to be known in advance. The design procedure is characterized by the fact that it explicitly considers the inter-sample behavior. This allows for the choice of "large" sampling times, which is important since in the case of robot control the computational burden on the computer is generally high. Common design procedures only consider the system behavior at the sampling instants and therefore demand "small" sampling times (Ackermann, 1985, Astrom and Wittenmark, 1984, Franklin and Powell, 1980). Even if the sampling time is "small" it is still favorable to consider the inter-sample behavior explicitly since the continuous-time behavior is of actual interest. To demonstrate our design procedure we will design and compute digital tracking controllers for an industrial cartesian X-Y robot and present experimental results obtained after their implementation . The controllers will be computed for time-optimal motions of the X-Y robot computed by Van Willigenburg (1990a) so that in these cases one may consider the result to be a digital time-optimal controller.

The outline of the paper is as follows. In paragraph 2.1 we will introduce the dynamics of a general rigid cartesian robot including gravity and friction. Based on identification experiments we will present a dynamic model of the industrial X-Y

robot in paragraph 2.2. In section 3 we will present the design procedure based on both the digital optimal regulator and tracker result presented in paragraph 3.2. In paragraph 3.3 we will present numerical examples and in paragraph 3.4 experimental results obtained with the implemented controllers. Finally section 4 contains the conclusions.

2. Dynamics of cartesian robots

2.1 Dynamic model of a rigid cartesian robot with friction

We will consider a rigid cartesian robot, i.e. a robot with rigid links that move in perpendicular directions and where the forces applied to the links are considered to be the control variables. If for instance the links are actuated by current controlled DC motors and the transmission is considered to be ideal the motor current is proportional to the force applied to the links (Van Willigenburg, 1990a). In paragraph 2.2 we will demonstrate that for the X-Y robot this is a reasonable assumption. Since the links move in perpendicular directions the motion of one link does not influence the motion of the others and we may consider the dynamics of each link separately. We will assume each link suffers from both viscous and coulomb friction, which is a reasonable assumption for the X-Y robot treated in paragraph 2.2. Finally some links are considered to move in the direction of gravity. Using the above assumptions the dynamics of such a link become

$$\ddot{x}_c = -v\,\dot{x}_c - c\,\mathrm{sign}(\dot{x}_c) + bu - g, \qquad (1)$$

where x_c represents the translation of the link with respect to a reference position , v is the viscous friction coefficient, c the coulomb friction coefficients and b the sensitivity coefficient to the control variable u which is the actuation force of the link. Finally g represents acceleration due to gravity, the direction being opposite to a positive translation. Note that we can

compensate for gravity by a constant value of the control. By making a change of control variable from (1) we obtain

$$\ddot{x}_c = -v\ \dot{x}_c - c\ \mathrm{sign}(\dot{x}_c) + bu' \tag{2a}$$

where

$$u' = u + g/b \tag{2b}$$

Equation (2) represents a general description of the dynamics of each link, where the second term of (2b) should be dropped if the link does not move in the direction of gravity. By inspection of (2) the dynamics of each link are of the form (2a). Therefore, and since the dynamics of each link are decoupled it can easily beseen that all ideas put forward for the X-Y robot can easily be extended to cartesian robots with more links.

2.2 Dynamic model of an X-Y robot

The industrial X-Y robot that we consider as an example is actuated by DC motors. Since both links move in a direction perpendicular to that of gravity from (1) we obtain the following dynamics for the X-Y robot,

$$\ddot{x}_p = -v_x\ \dot{x}_p - c_x\ \mathrm{sign}(\dot{x}_p) + b_x u_x, \tag{3a}$$

$$\ddot{y}_p = -v_y\ \dot{y}_p - c_y\ \mathrm{sign}(\dot{y}_p) + b_y u_y, \tag{3b}$$

where x_p represents the translation of the X-link with respect to a reference position, y_p represents the translation of the Y-link with respect to a reference position, v_x and v_y are the viscous friction coefficients, c_x and c_y the coulomb friction coefficients and finally b_x and b_y the sensitivity coefficients to the control variables u_x and u_y, which represent the motor-currents. Both b_x and b_y depend on the mass of the link and the payload which we assume to be known. We may write the dynamics (3) in state-space

form,

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ \ddot{x}_p \\ \ddot{y}_p \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -v_x & 0 \\ 0 & 0 & -v_y & 0 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ \dot{x}_p \\ \dot{y}_p \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ b_x & 0 \\ 0 & b_y \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ c_x \text{sign}(\dot{x}_p) \\ c_y \text{sign}(\dot{y}_p) \end{bmatrix}. \qquad (4)$$

We will explain the choice of this model, and then present the parameter values obtained from identification experiments. The model assumes the transmission from the motor to the robot links to be perfectly rigid. Furthermore the motor current controllers, to which the actual control is applied, are assumed to be ideal (Van Willigenburg, 1990a). Experimental results have demonstrated that the latter is a reasonable assumption. Figure 2.1 shows two characteristic step responses of the motor current controller. Since we will use a piecewise constant control and sampling times from 60 to 200mS, the input to the motor current controllers will consist of steps of between 60 and 200mS. Therefore the transient behavior shown in figure 2.1 may very well be neglected.

Experimental results with the X-Y robot revealed that friction plays an important role in the robot dynamics. Figure 2.2a shows speed responses of the X-link while the motor current u_x is kept at a certain constant value u_{x_c} for several initial conditions. The fluctuations in the speed responses are mainly caused by backlash which we have not included in the model. Since the speed responses, apart from the fluctuations caused by backlash, are almost flat, from the dynamics (3a) we observe that a) the coulomb friction in the positive direction is compensated for by the constant value u_{x_c} of the control and b) viscous friction is negligible. So we have

$$v_x = 0. \qquad (5)$$

Figure 2.2b shows a characteristic speed response of the Y-link together with an exponential approximation, when the motor-current

u_y is kept at a certain constant value u_{y_c}. When we approximate the speed-response by the exponential from the dynamics (3b) we conclude that a) the coulomb friction in the positive direction is compensated for by the constant value u_{y_c} of the control b) viscous friction is present and determined by the exponential approximation in figure 2.2b, since if the coulomb friction is compensated for by the control, the second and third term on the right of equation (3b) cancel out. So from figure 2.2b and equation (3b) we have

$$v_y = 3.0. \tag{6}$$

From figure 2.2 we also obtain the values of the control u_{x_c} and u_{y_c} necessary to compensate the coulomb friction. They have been found through experiments and where chosen such that after an initial velocity the links kept just moving in both the positive and negative direction, while applying u_{x_c} and u_{y_c} with appropriate sign.

$$u_{x_c} = 1.0, \tag{7}$$

$$u_{y_c} = 1.1. \tag{8}$$

Figure 2.3a shows a speed-response of the X-link while the control is kept at its maximum value,

$$u_x = u_{max}, \tag{9a}$$

where

$$u_{max} = 10.0. \tag{9b}$$

From (4) and (5) the response is expected to be a straight line. The deviations from the straight line are primarily caused by backlash. The sensitivity b_x to the control u_x may be obtained

from (4) and (7),

$$b_x = s_x/(u_{max}-u_{x_c}) \tag{10}$$

where s_x is the slope of the straight line in figure 2.3a,

$$s_x = 630 \ . \tag{11}$$

From (4) and (5) we also have

$$c_x = b_x \, u_{x_c} . \tag{12}$$

Summarizing, given u_{max}, u_{x_c} and s_x given by (7), (9b) and (11) using (10) and (12) we obtain the parameter values b_x and c_x of the X-link,

$$b_x = 70 \ , \tag{13}$$

$$c_x = 70 \ . \tag{14}$$

Figure 2.3b shows the speed-response of the Y-link while the control is kept at its maximum value

$$u_y = u_{max}, \tag{15}$$

where u_{max} is given by (9b). Consider s_y to be the slope of the speed response in figure 2.3b at t=0. From figure 2.3b we obtain

$$s_y = 783 \ , \tag{16}$$

Since at t=0 the speed and therefore the viscous friction occurring in the Y-link are equal to zero we can derive the parameter values b_y and c_y in exactly the same manner, i.e. using equations (7),(8),..,(12) where the index x must be replaced by the index y. Doing so from (16) we obtain

$$b_y = 88 \ , \tag{17}$$

$$c_y = 96 \ . \tag{18}$$

3. Digital time-optimal controller design and implementation

3.1 Introduction

The dynamic model (4) of the X-Y robot, apart from coulomb friction, constitutes a linear model. In section 2 we demonstrated that coulomb friction can be compensated for by the control, leaving a linear model. The recently developed digital optimal regulator and tracker (Van Willigenburg and De Koning 1990a,b , Van Willigenburg, 1990b) for linear systems were used to compute digital controllers for the X-Y robot. Usual digital controller designs (Ackermann, 1985, Astrom and Wittenmark, 1984, Franklin and Powell, 1980) are based on approximations, and therefore demand "small" sampling times since they do not explicitly consider the inter-sample behavior. In the case of robot control this demand often leads to computational problems (Vukobratovic and Stokic, 1982). The digital optimal regulator and tracker were derived without making any approximations, and therefore do not demand a "small" sampling time. We will use the digital optimal tracker result (Van Willigenburg and De Koning, 1990a,b) to design tracking controllers for the X-Y robot. Since the digital optimal tracker is given in feedback form, we may directly implement the result. The digital control system that results is represented by figure 3.1.