Design of robotic discrete minimum energy regulator

Y. Bestaoui

Indexing terms: Robotics, discrete optimal control, differential dynamic programming

Abstract: The optimal control of manipulators is a key to the success of automated manufacturing. The problem considered here is an energy minimisation problem with given dynamics and is subject to actuator constraints. A differential dynamic programming algorithm is developed to solve the discrete-time optimal control problem. This method allows calculation of the joint reference trajectories and the design of a proportional derivative regulator. The characteristics of this new method are exposed and the simulation results shown.

1 Introduction

The current generation of manipulators can complete a typical positioning task in seconds. The most frequent task is to transfer the system state from one bounded region of initial states into another bounded region in the state space within a finite settling time. The system state should belong to a bounded region in the state space during the transfer. A better manipulator performance can be achieved by improving their mechanical construction and by using more effective controllers. In this paper, we are only concerned with the latter. The optimal control of manipulators is a key to the success of automated manufacturing. It is therefore essential to design an optimal manipulator system with a suitable performance criterion which is consistent with the foregoing goals.

In many cases, manipulators are desired to move from one point to another using as little energy as possible. Consequently, it is important to design an efficient controller which requires less energy, thus pushing the manipulators to be operated at their maximum efficiency. This consideration naturally leads to a constrained optimal control problem of robotic manipulators with a minimum energy criterion.

The industrial robot is highly nonlinear, which is one of the reasons that makes optimum control a difficult problem. A second reason is the strict constraints imposed on the system. Because of the difficulty, few authors [1, 2] have attempted to minimise the time or quadratic performance index. When such problems are tackled by the maximum principle, nonlinear two points boundary value problems appear and may be solved using several successive appproximation methods [2]. For example, Kahn and Roth [1] investigate the minimum time solution along an operational point, using a linearised continuous model, which represents a gross simplification. There has also been much interest in the development of hierarchical algorithms [3, 4].

This paper presents a control design methodology for robotic manipulators considered as discrete-time systems. The dynamics of the system are then represented as an augmented equation composed of the states of the system, interconnected dynamics and constraints on the control variables. The design of a manipulator feedback control system on the basis of the minimum energy criterion, is presented. This controller provides a feedback control algorithm with a simple updating structure so that it can be implemented in real time on mini or microcomputers. Primarily, this controller is intended to provide fast operation speed using less energy and with reasonable setting accuracy.

Paper 8220D (C8, C15), received 9th November 1990

The author is with the Laboratoire de Robotique et d'Informatique Industrielle, 3 rue du Maréchal Joffre, 44041 Nantes, France

2 Problem formulation

For a manipulator with n joints, the dynamic model can be expressed using a Lagrangian equation as:

$$D(q)q'' + h(q, q') + g(q) = u(t) \tag{1}$$

where the $n \times 1$ vectors q, q' and q'' are, respectively, the joint position, joint velocity and joint acceleration, the $n \times 1$ vector $u(t)$ is the joint input torque, $g(q)$ is the $n \times 1$ gravitational force vector, $h(q, q')$ is the $n \times 1$ Coriolis and centrifugal force vector and $D(q)$ is the $n \times n$ inertial matrix. The inputs in the model are the forces/torques, and the outputs are selected as the positions and the velocities of the manipulator joints.

A $2n \times 1$ state vector of the manipulator system can be defined as:

$$\begin{aligned} x(t) &= (q^T(t), q'^T(t))^T \\ &= (q_1, \ldots, q_n, q'_1, \ldots, q'_n)^T \\ &= (x_1, \ldots, x_{2n})^T \end{aligned} \tag{2}$$

where T denotes the transpose. From eqn. 1, the state-space description of the system [5, 6] can be given by:

$$x'(t) = A(t)x(t) + B(x(t))u(t) + C(x(t)) \tag{3}$$

where

$$\begin{aligned} A &= \begin{bmatrix} 0 & I_n \\ 0 & 0 \end{bmatrix} \\ B(x(t)) &= \begin{bmatrix} 0 \\ D^{-1}(q) \end{bmatrix} \\ C(x(t)) &= \begin{bmatrix} 0 \\ -D^{-1}(q)(h(q, q') + g(q)) \end{bmatrix} \end{aligned} \tag{4}$$

with I_n the $n \times n$ identity matrix. The pair (A, B) is controllable [5]. The state-space model reflects the fact that robots are variable-inertia mechanical systems.

When a robot is under computer-control, the inputs $u(t)$ are updated at each sampling instant and maintained constant by digital to analogue convertors (DAC). The DAC are the interfaces between the digital controller and the robot:

$$u(t) = u(kt) = u^k \quad \text{for } kT \leqslant t < (k+1)T \tag{5}$$

The natural (continuous-time) state variables of the robot are the joint co-ordinates and velocities (q and q'). These physical state-variables are accessible and can be measured directly with currently available instrumentation. The analogue to digital convertors (ADC) sample the state variables at each sampling instant to produce the piecewise constant states of the robot:

$$q(t) = q(kT) = q^k \quad \text{for } kT \leqslant t < (k+1)T$$

$$q'(t) = q'(kT) = q'^k \quad \text{for } kT \leqslant t < (k+1)T$$

Considering that the manipulator is controlled by a computer, eqn. 3 needs to be discretised. Setting T as the sampling period and noticing that $u(t)$ is constant in the interval of time $(kT, kT + T)$, the discretised state model can be expressed as:

$$x_i^{k+1} = x_i^k + Tx_{i+n}^k \quad \text{for } 1 \leqslant i \leqslant n \tag{6}$$

$$x_i^{k+1} = x_i^k + T(D^k)^{-1}u^k + T(-(D^k)^{-1}(h^k + g^k)) \quad \text{for } (n+1) \leqslant i \leqslant 2n \tag{7}$$

in which we denote the value of the generic function $f(t)$ at the instant $t = kT$ with f^k

We can set:

$$x^{k+1} = E^k x^k + G^k u^k + d^k \tag{8}$$

where

$$E^k = \begin{bmatrix} I_n & TI_n \\ O_n & I_n \end{bmatrix} \tag{9}$$

$$G^k = TB^k$$

$$d^k = TC^k$$

The dynamic control of an industrial manipulator involves the determination of the inputs for the actuators which operate at the joints so that the set of desired values for the positions and velocities of the manipulator is achieved.

In the feedforward control stage, the driving forces/torques generated by the planner are realised in the active region of the actuators. This control design is accomplished by placing limits on the maximum driving forces/torques and restricting the actuator outputs to lie within these limits. For this task, optimal control approaches can be formulated by augmenting the discrete dynamic robot model with discrete performance criteria. Application of mathematical and dynamic programming can then lead to an optimal feedforward controller.

A generalised version of the statement of the problem is 'Given the robot's dynamic properties and the robot characteristics, what set of signal to the actuators will drive the robot from its current state to a desired final state with a minimum cost?'. This problem can be stated as follows:

$$\min_u L(u) = \left[\sum_k u^{k^T} u^k\right] \Big/ 2 \tag{10}$$

subject to

$$x^{k+1} = T^k(x^k, u^k) = E^k x^k + G^k u + d^k \quad k = 1, \ldots, N \tag{11}$$

$$u^{min} \leqslant u \leqslant u^{max} \tag{12}$$

$$x^0 = x^i,\ x^{N+1} = x^{end} \tag{13}$$

where eqn. 10 represents the objective function to be minimised, eqn. 11 the discrete dynamics of the robot, eqn. 12 the torque constraints and eqn. 13 two points boundary values with x^0 initial point and x^{end} final point. Here, the functions F^k are the real valued 'single-stage' objective functions, and T^k are discrete time state transition functions. They are twice continuously differentiable. This is a nonlinear constrained optimisation problem. Lagrangian and penalty methods are procedures for approximating constrained optimisation problems by unconstrained ones:

$$\max_{v_1, v_2} \min_u \left\{ J(u, v) = \left[\sum_k u^{k^t} u^k\right] \Big/ 2 + v_1^T(-u + u^{min}) + v_2^T(u - u^{max}) + p(x^N - x^{end})^2 = \sum_k F^k(u^k) \right\}$$

subject to

$$x^{k+1} = T^k(x^k, u^k) = E^k x^k + G^k u^k + d^k \quad k = 1, \ldots, N \tag{14}$$

$$x^0 = x^i$$

where v_1, v_2 are Lagrange multipliers and p is a penalty parameter. The Lagrange multiplier can be updated by the projection of the gradient of the objective function onto the tangent subspace at u [7, pp. 257–259]. The penalty parameter p can be chosen arbitrarily (e.g. 10, 100, 1000), depending on the desired precision.

The differential dynamic programming method being an initial value unconstrained optimisation technique may be applied to eqn. 14. The resolution method is presented in the following section.

3 Resolution

3.1 *Differential dynamic programming*

This Section introduces an extremely promising decentralised differential dynamic programming algorithm for robotic discrete-time optimal control problems [6]. For such problems, existing dynamic programming algorithms include the state increment dynamic programming [8] and the gradient method [9]. All of these methods share the property that they do not require a discretisation of the state-space. However, these two methods converge only at a linear rate.

Differential dynamic programming is a stagewise implementation of nonlinear programming and is tailored for nonlinear discrete time optimal control problems. It was introduced in Reference 10 as a stagewise successive approximation method for initial value problems. The solution of a nonlinear optimum control problem is approximated by forming a sequence of control laws on the basis of the dynamic programming technique. Given a nominal control u^d (a nonoptimal initial policy), each iteration of the differential dynamic programming consists of two phases: a backward run and a forward run. The analogous phases in conventional

nonlinear programming algorithms [7, 9] are the direction finding step and the line search step, respectively. In the backward run, stagewise value functions (return at each stage plus the cumulative return from succeeding stages) are given by a quadratic. Based on these functions, the forward run generates a successor control u^+[11]. The nominal control, successor control and the previous control are designated by

$$u^d = (u_1^d, \ldots, u_n^d)$$
$$u^+ = (u_1^+, \ldots, u_n^+)$$
$$u^- = (u_1^-, \ldots, u_n^-) \quad (15)$$

respectively. Similarly, the corresponding state trajectory will be denoted by

$$x^d = (x_1^d, \ldots, x_n^d)$$
$$x^+ = (x_1^+, \ldots, x_n^+)$$
$$x^- = (x_1^-, \ldots, x_n^-) \quad (16)$$

The backward run begins at the last stage, denoted N. Here we use notation like dx^k to denote perturbations about the nominal values. Thus

$$dx^k = x^k - x^{dk} \quad (17)$$

At any stage k, we will give the value function, i.e. return due to stage k plus the cumulative return from succeeding stages, by a quadratic function denoted $V^k(dx^k)$. The backward run is initialised by putting $V^{N+1}(dx^{N+1}) = 0$. For stages $k = N, \ldots, 1$, the quadratic value function $V^k(dx^k)$ is obtained recursively. This construction is described next.

Let $V^{k+1}(dx^{k+1})$ be given. Because $V^{k+1}: R^2 \to R$ is a quadratic function, it may be represented as

$$V^{k+1}(dx^{k+1}) = J^{k+1} + (c^{k+1})^T(dx^{k+1}) + 0.5(dx^{k+1})^T Q^{k+1}(dx^{k+1}) \quad (18)$$

Recall that

$$dx^{k+1} = x^{k+1} - x^{dk+1} = T^k(x^k, u^k) - x^d \quad (19)$$

Hence the value function at stage k (return at stage k plus the value function at stage $k+1$) is given by:

$$S^k(x^k, u^k) = F^k(u^k) + V^{k+1}(dx^{k+1}) \quad (20)$$

The quadratic function S^k can be written as:

$$S^k(x^k, u^k) = 0.5u^{k2} + J^{k+1} + (c^{k+1})^T(dx^{k+1}) + 0.5(dx^{k+1})^T Q^{k+1}(dx^{k+1}) + v_1^{kT}(-u^k + u^{k\,min}) + v_2^{kT}(u^k - u^{k\,max}) \quad (21)$$

Using first-order necessary conditions for an extremum of S^k, one obtains the strategy:

$$du^k = \alpha^k + \beta^k\, dx^k \quad (22)$$

where

$$\alpha^k = -(1 + G^{kT}Q^{k+1}G^k)^{-1}(\nabla_u S^{dk}) \quad (23)$$

and

$$\beta^k = -(1 + G^{kT}Q^{k+1}G^k)^{-1}(\nabla_u S^d)^T Q^{k+1}(\nabla_x T^d)^T \quad (24)$$

∇_u, ∇_x denote, respectively, the gradient with respect to u and x.

Substituting eqn. 22 into eqn. 21 we define

$$V^k(dx^k) = J^k + (c^k)^T dx^k + 0.5(dx^k)^T Q^k dx^k \quad (25)$$

where

$$J^k = S^{dk} + \beta^k \nabla_u S^d \alpha^k \quad (26)$$

$$c^k = \nabla_x S^{dk} + \beta^k \nabla_u S^{dk} \quad (27)$$

$$Q^k = E^{kT}Q^{k+1}E^k + \beta^k G^{kT}Q^{k+1}G^k \quad (28)$$

with the gradients (row vectors) given by

$$\nabla_x S^k = E^T(c^{k+1} + Q^{k+1}(Ex^k + G^k u^k + d^k - x^{dk})) \quad (29)$$

and

$$\nabla_u S^k = u + G^{kT}(c^{k+1} + Q^{k+1}(Ex^k + G^k u^k + d^k - x^{dk})) - v_1^k + v_2^k \quad (30)$$

This completes the computation of the backward run for stage k. Because eqn. 25 has the same form as eqn. 18, computations for stage $k-1$ may now begin. This procedure continues until we obtain du^1 (eqn. 22) for stage $k = 1$. At this point, the backward run of a differential dynamic programming iteration is complete.

The forward run computes the successor differential dynamic programming control. Because x^1 is fixed, setting $dx^1 = 0$ yields $du^1 = \alpha^1$ or $u^{1+} = u^{d1} + du^1$. Next, we recursively compute

$$x^{+k} = T^{k-1}(x^{+k-1}, u^{+k-1})$$
$$du^k = \alpha^k + \beta^k\, dx^k \quad (31)$$

These forward run computations are standard for differential dynamic programming, such a strategy does not ensure global convergence. Hence, to ensure global convergence, some line search scheme is necessary. Towards this end, let $\rho > 0$ and define

$$du^k(\rho) = \rho\alpha^k + \beta^k\, dx^k \quad (32)$$

Initially, ρ is set to 1 (in relation with the Newton method). If we find that

$$L(u(\rho)) - L(u^-) < c\rho \quad (33)$$

where c is a constant, then $u(\rho)$ is accepted as the successor policy and replaces u^- in the next iteration. Otherwise, ρ is redefined to be one half its former value and the policy $u(\rho)$ is again computed. This process of having ρ and testing $u(\rho)$ continues until acceptance occurs. Most times, $\rho = 1$. Other techniques of computing the steplength are available as the Goldstein's or Armijo's rule [11].

The description of the backward run and the forward run is now complete. For stability, the function should always be decreasing (see Table 1). The algorithm converges quadratically because all the functions involved are convex [10, 11].

3.2 Regulator design

This method allows the design of a proportional-derivative regulator eqn. 22 to 24 and 32

$$u = u^* + K_p(q - q^*) + K_v(q' - q'^*) \quad (34)$$

where the proportional and derivative gains matrices are given by

$$(K_p, K_v)^T = -\beta \quad (35)$$

with

$$K_p = \text{diag}\,(K_{pi}) \quad \text{and} \quad K_v = \text{diag}\,(K_{vi}) \quad (36)$$

The coefficients q^*, q'^* and u^* are the solutions obtained by the differential dynamic programming method.

Table 1

Recursion number	Value L(u) ×10	ρ	Difference L(u(ρ)) − L(u) ×10
1	29879.89	1	—
2	9987.672	1	19892.22
3	4736.469	1	5251.203
4	3195.438	1	1541.031
5	2574.639	1	620.799
6	2525.727	1	48.912
7	2521.012	1	4.715
8	2519.024	1	1.988
9	2518.122	1	0.902
10	2517.694	1	0.428
11	2517.485	1	0.209
12	2517.382	1	0.103
13	2517.332	0.5	0.05
14	2517.306	1	0.026
15	2517.293	1	0.013
16	2517.287	0.5	0.006
17	2517.284	0.5	0.003

These regulator parameters may consist of trigonometrical elements, but their evolution is very smooth, so these terms can be approximated by simple functions. This step is robot-dependent. Each term of β ($2n$ in all) may be plotted against the normalised time τ:

$$\tau = t/t_{sim}$$

t is the current time and t_{sim} the simulation time.

In general, linear interpolation is sufficient (see Fig. 6, 7, 13 and 14):

$$K_{pi} = \delta_i \tau + \gamma_i \tag{37}$$

where the coefficients δ and γ are computed as follows: Let K_p^0 and K_p^1 be, respectively, the values of the proportional gains at the beginning and the end of the off-line simulation, then

$$\delta = K_p^0 \quad \gamma = (K_p^1 - K_p^0) \tag{38}$$

The same computations are done for the derivative gains. The parameters δ and γ are computed offline. Online, it will suffice to calculate eqn. 37. The term α (see Fig. 8) depends strongly on the initial choice of Q and c. The term u^* represents the desired torques: solution of the optimisation problem eqns. 10–13 (see Figs. 5 and 11), q^* and q'^* the corresponding joint position and velocity. They can be stored, then used online as reference trajectories.

4 Simulation results

This Section presents the application of the described method to the cartesian manipulator presented in Fig. 1

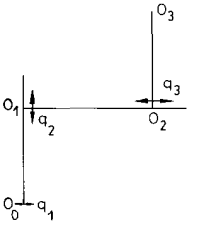

Fig. 1 *Configuration of cartesian manipulator*

and the MA23 robot arm presented in Fig. 2. Numerous numerical simulations are illustrated here to test the efficiency of the proposed control scheme. For the cartesian manipulator, to obtain comprehensive information about the tracking capabilities of this technique, the algorithm has been evaluated over two different operational environments. The choice of these parameters is made arbitrary for the sake of numerical demonstration.

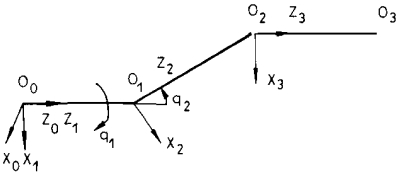

Fig. 2 *Configuration of MA23 manipulator*

4.1 Small amplitude

$q_0 = (0.1, 0.1, 0.1)$ and $q_{goal} = (0.5, 0.5, 0.5)$ m, simulation time 0.4 s. The bounds on the control input are assumed to be $-100\ \text{N} \leqslant u \leqslant 100\ \text{N}$. The mass of the third axis and the load is 4 Kg.

To show the importance of the sampling rate, four different experiments were done: $\delta t = 2.5$, 10, 20 and 50 ms. Figs. 3, 4 and 5 show, respectively, the third joint posi-

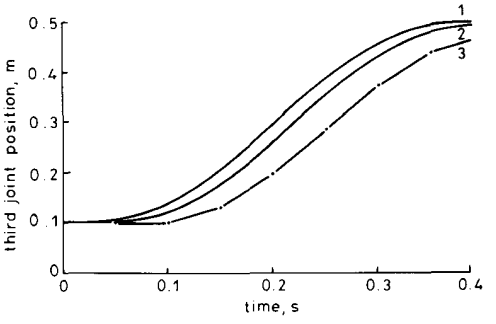

Fig. 3 *Third joint position for different sampling periods: cartesian manipulator*

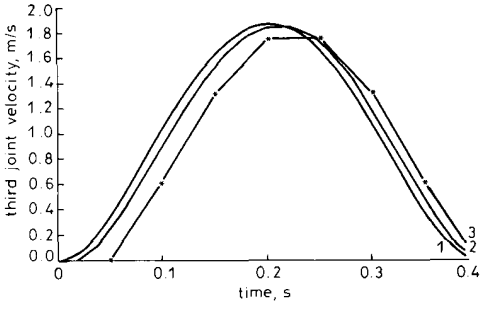

Fig. 4 *Third joint velocity for different sampling periods: cartesian manipulator*

tion, velocity and torque, for these sampling rates. For $\delta t = 2.5$ and 10 ms see curve 1, for $\delta t = 25$ ms see curve 2 and for $\delta t = 50$ ms, see curve 3. Sampling extracts a discrete-time signal from a continuous time one. The sampling frequency must be selected properly to permit accurate representation of the assigned signal by the resulting sample sequence. Experiments seem to indicate

that the choice of the sampling rate should be no less than 60 Hz ($T \leqslant 11$ ms) to achieve a sufficiently smooth control for most motions. In the following, the sampling period chosen is $\delta t = 10$ ms.

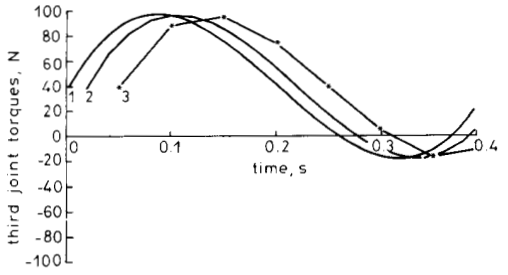

Fig. 5 *Third joint torque for different sampling periods: cartesian manipulator*

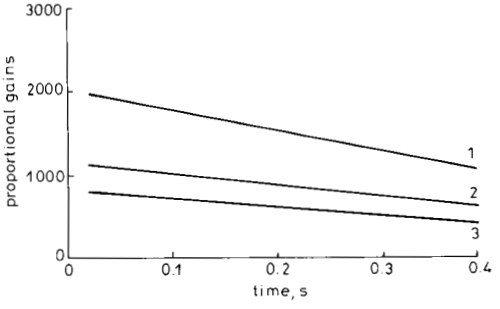

Fig. 6 *Proportional gains against time: cartesian manipulator*

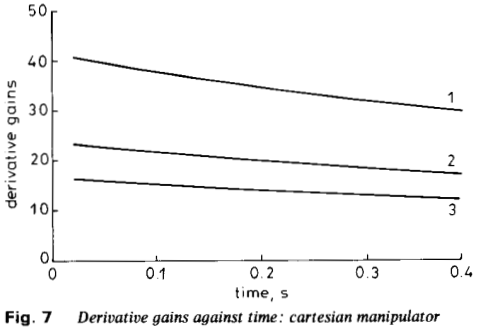

Fig. 7 *Derivative gains against time: cartesian manipulator*

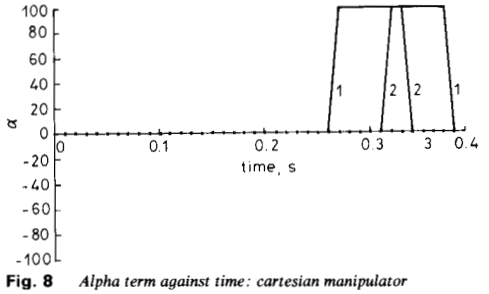

Fig. 8 *Alpha term against time: cartesian manipulator*

Figs. 6 and 7 give the evolution of the regulator gains against the time for the three joints. It can be seen that the evolution is linear. Fig. 8 shows the evolution of the α term, for the three joints.

Fig. 9 gives the norm of the error between the optimisation problem solution x^* and real position x in cartesian space, respectively, of tests 1 to 3.

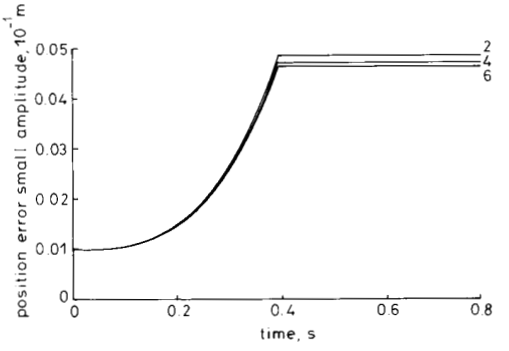

Fig. 9 *Position error against time for different loads: cartesian manipulator*

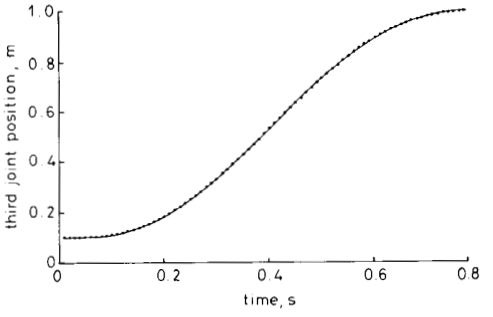

Fig. 10 *Third joint position for large amplitude: cartesian manipulator*

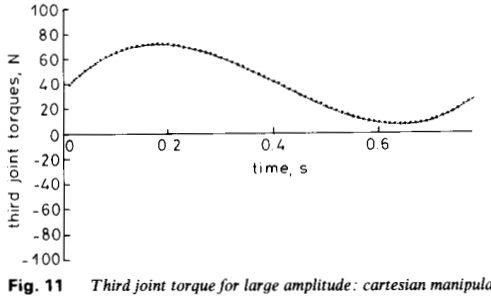

Fig. 11 *Third joint torque for large amplitude: cartesian manipulator*

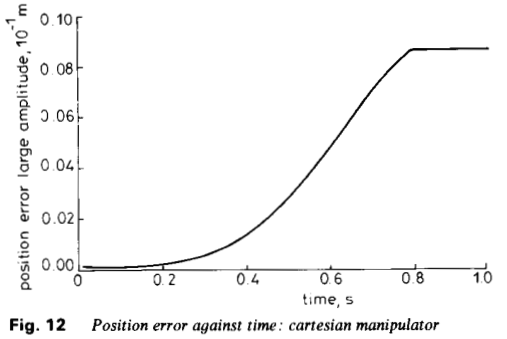

Fig. 12 *Position error against time: cartesian manipulator*

test 1* When the nominal mass of the third axis and the load is exactly estimated ($m_3 = 4$ Kg).

test 2* When the nominal mass of the third axis and the load is overestimated by 50% with respect to the assigned mass ($m_3 = 2$ Kg).

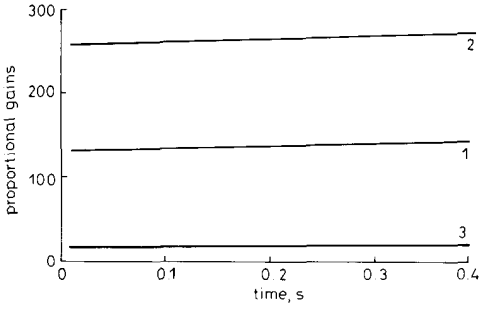

Fig. 13 *Proportional gains against time: MA23 manipulator*

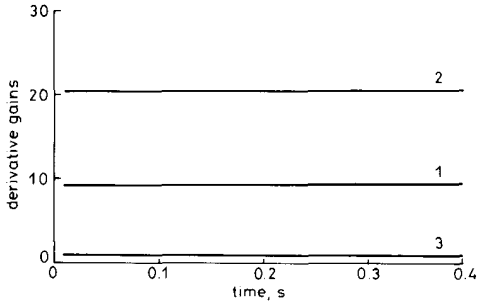

Fig. 14 *Derivative gains against time: MA23 manipulator*

test 3* When the nominal mass of the third axis and the load is underestimated by 50% with respect to the assigned mass ($m_3 = 6$ Kg).

The nominal values were used in the computation of the nonlinear feedback u^* and the parameter gains K_p and K_v (solution of the optimisation problem). The assigned values were used in the computations implementation of the robot arm. The robustness of this new control strategy is well illustrated by these figures. There is a static error: a classical result for a proportional derivative regulator. Table 1 gives the successive results for test 1 when the differential dynamic programming method is applied to the solution of problem (eqns. 10–13).

4.2 Large amplitude

$q_0 = (0.1, 0.1, 0.1)$ m and $q_{goal} = (1, 1, 1)$ m, simulation time 0.8 s. The bounds on the control input are assumed to be: $-75 \text{ N} \leqslant u \leqslant 75$ N.

Figs. 10, 11 and 12 give, respectively, the evolution of the third joint position, torque and position error norm. Figs. 5 and 11 show that the motor torques limits are never exceeded. Figs. 13 and 14 show, respectively, the evolution of the proportional and derivative gains against time, for a three rotate joints manipulator: the MA23 robot: $x_0 = (0.1, 0.1, 0.1)$ rad and $q_{goal} = (0.5, 0.5, 0.5)$ rad. The motor torques limits are $-100 \text{ Nm} \leqslant u \leqslant 100$ Nm. The gains evolution is still linear. The position, velocity and torque evolution is similar to the cartesian manipulator one.

Other simulation results have also shown that a suitable selection of the initial conditions (choice of Q^{N+1} and c^{N+1}) in computing the feedback gains also plays an important role in reducing position and velocity errors in tracking the desired path x^*. The main simulation result is that there is a smooth positioning, never exceeding the motor torques limits.

5 Conclusions

The problem considered first is an energy minimisation problem with given dynamics and subject to the actuator constraints. This resolution cannot take place online, the operations involved being too numerous. However, this method has an important advantage. It allows the design of a proportional derivative regulator with gains determination done by the differential dynamic programming method, as well as the calculation of reference trajectories. The same method may be used for the design of a PID regulator with slight changes in the space vector.

This regulator is designed from the solution of an optimisation problem which takes care of the real physical constraints on the manipulators: the applied torques at the joints. This is an important result since real constraints are introduced in the calculation of the gain parameters, and the planning of the desired positions and velocities.

The recursive algorithm used in the proposed control scheme requires only a small amount of memory space (u^*, q^*. q'^*), the mathematical operations are simple and fast to compute. Simulation studies on the manipulator systems have been presented to demonstrate the applicability of the control scheme.

It should be noted that the energy minimum controller studied in this paper is concerned only with free space motion.

6 References

1 KAHN, M.E., and ROTH, B.: 'The near minimum-time control of open-loop articulated kinematics chains', *J. Dynamic Syst. Meas. & Control*, 1971, pp. 164–172

2 KIM, B.K., and SHIN, K.G.: 'Suboptimal control of industrial manipulators with a weighted minimum-time fuel criterion', *IEEE Trans.*, 1985, **AC-30**, pp. 213–223

3 ABOU KANDIL, H., DROUIN, M. and BERTRAND, P.: 'Two-level control laws for discrete-time nonlinear systems'. Proceedings of the American Control Conference, 1984, pp. 1493–1497

4 BESTAOUI, Y.: 'Adaptive hierarchical control for robotic manipulators', *Robotics & Autonomous Systems*, 1988, **4**, pp. 145–155

5 LIU, M.H., WEI, L., and HUANG, J.F.: 'Pole assignment self-tuning control of robotic manipulators'. Proceedings 16th International Symposium on Industrial Robots, UK, 1986, pp. 289–297

6 NICOSIA, S., and TOMEI, P.: 'A discrete-time MRAS control for industrial robot'. Proceedings IFAC Symposium, 1985, pp. 83–89

7 LUENBERGER, D.G.: 'Introduction to linear and nonlinear programming' (Addison-Wesley, 1973)

8 YAKOWITZ, S.J.: 'Convergence rate analysis of the state-increment dynamic programming method', *Automatica*, 1983, **19**, (1), pp. 53–60

9 McCORMICK, G.P.: 'Nonlinear programming' (J. Wiley, 1983)

10 YAKOWITZ, S., and RUTHERFORD, B.: 'Computational aspects of discrete-time optimal control', *Applied Math. & Comput.*, 1984, **15**, pp. 29–45

11 SEN, S., and YAKOWITZ, S.J.: 'A quasi-Newton differential dynamic programming algorithm for discrete-time optimal control', *Automatica*, 1987, **23**, (6), pp. 749–752