CONTROL STRATEGIES FOR ROBOTIC MANIPULATORS

J.V. RINGWOOD

School of Electronic Engineering
Dublin City University
Glasnevin, Dublin 9, Ireland

ABSTRACT

The dynamic end-point positioning problem for robotic manipulators is considered. A review of the current approaches to the problem is presented with a critical examination of each technique. A close inspection of the structure of a typical manipulator model is performed to investigate the suitability of the various control approaches. In particular, the PUMA 560 Industrial Manipulator is considered. The advantages and disadvantages of various techniques, including linear and non-linear, fixed and adaptive methods are given and a current approach, based on non-linear self-tuning theory is presented. An attempt is made to give some appreciation of the relative computational complexity of various algorithms and a number of possible hardware architectures for control are discussed. Finally, conclusions regarding the choice of dynamic control algorithms for manipulators are drawn.

1. INTRODUCTION

Currently, the area of robotics poses some of the most challenging problems to the control design engineer. In these days where efficiency in manufacture is so important and improvements in tolerances and standards is a constant requirement, it is desireable to improve both the speed and accuracy of robotic manipulators.

The current range of manipulator-type robots contain either revolute joints, prismatic joints, or a combination of both. A review of the manipulators available would confirm that they fall into a number of 'standard' configurations i.e. although a good variety of manipulators are available, many have similar geometrical structures [1]. Examples of common structures are those of the Puma (revolute) and the Stanford (revolute and

Commence typing
prismatic) manipulators.

The performance of a manipulator-type robot is influenced by a number of factors - dexterity is largely a function of the physical design, and the speed and precision of the end-effector movement depend both on the hardware employed (servomotors and electronic hardware) and the software used to drive it. The mechanical design necessary to achieve dexterity, however, results in a system with complex dynamic properties. To realize the full potential of the robot, the control system must compensate for all the dynamic interactions between different sections of the manipulator and provide the potential for high speed accurate movement of the end-effector position. Since the physical structure of manipulators cannot be improved upon by any large extent, the main performance improvements must be achieved by advanced controller software.

In this paper, the various facets of the robot control problem are outlined with a detailed examination of the dynamic control element. In Section 2, a mathematical model which represents the dynamic interactions in a typical manipulator is described with emphasis on the structural components which present such a difficult control problem. The structure of manipulator controllers and applicable control theory is documented briefly in Sections 3 and 4 respectively. A brief review of current approaches to the dynamic control problem is presented in Section 5, which looks at both adaptive and non-adaptive schemes. In Section 6, a particular solution, based on non-linear self-tuning techniques, is given in algorithmic form. Since two of the limiting factors in manipulator control to date have been the computational complexity of the algorithms and the availability of powerful processing hardware, these aspects are discussed in Section 7. Finally, conclusions are drawn in Section 8.

3) Type in 1½ spacing single column.

2. A TYPICAL ROBOT MODEL

A dynamic model, which relates joint positions, velocities and accelerations to servomotor input voltages is given. This model corresponds to a robot with revolute joints only and, in the spirit of the paper, concentrates on the end effector positioning problem. For this reason, the model focusses on the three principal degrees of freedom, with the tool orientation dynamics omitted. The model is very simply extended, however, to include such dynamics. A typical example of the type of robot under consideration is the PUMA 560, shown in outline in Fig.1.

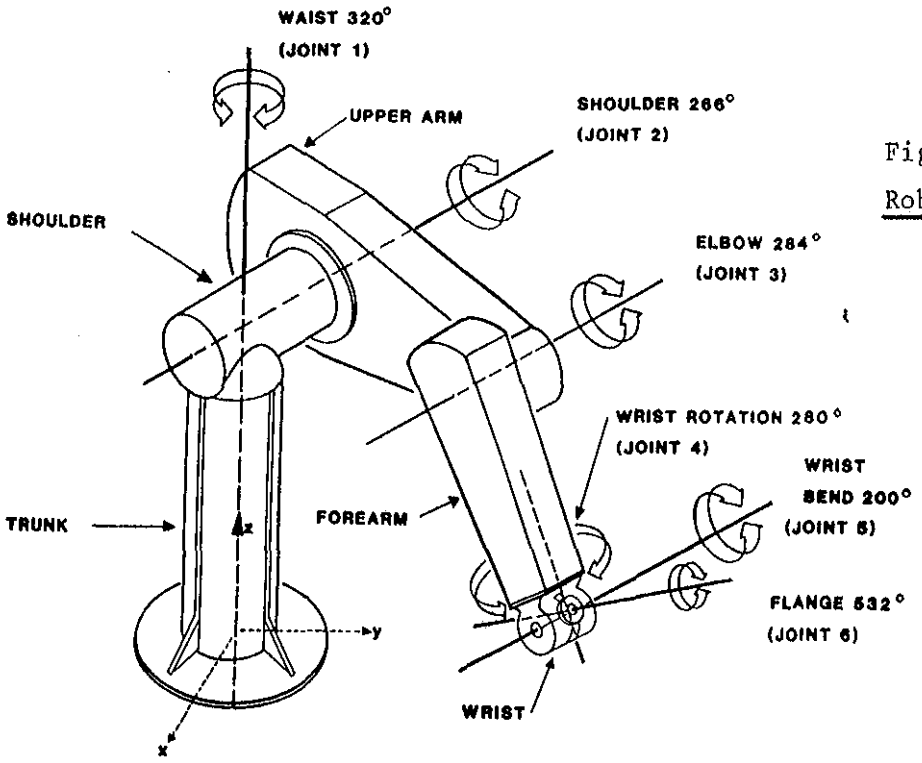

Figure 1: PUMA 560 Robotic Manipulator

The model is given [2] as:

$$
\begin{aligned}
v_i = \frac{L_i}{k_i}\Big[& \sum_{j=1}^{3} D_{ij}\ddot{q}_j + Ia_i\ddot{q}_i + 2\sum_{j=1}^{3}\sum_{k=1}^{3} C_{ijk}\dot{q}_j\dot{q}_k + \sum_{j=1}^{3}\sum_{k=1}^{3} C_{ijk}\dot{q}_j\dot{q}_k \Big] \\
+ \frac{R_i}{k_i}\Big[& \sum_{j=1}^{3} D_{ij}\dot{q}_j + \sum_{j=1}^{3}\sum_{k=1}^{3} C_{ijk}\dot{q}_j\dot{q}_k + Ia_i\dot{q}_i + H_i\dot{q}_i + G_i \Big] \\
+ \frac{L_i}{k_i}\Big[& \sum_{j=1}^{3} \dot{D}_{ij}\dot{q}_j + \sum_{j=1}^{3}\sum_{k=1}^{3} \dot{C}_{ijk}\dot{q}_j\dot{q}_k + H_i\ddot{q}_i + \dot{G}_i \Big] + \ell_i N_i \dot{q}_i \qquad (1)
\end{aligned}
$$

where:

q_i, $\dot{q}_i$, $\ddot{q}_i$ - joint i position, velocity and acceleration

v_i - voltage input to servomotor i

D_{ii}, D_{ij} - effective and coupling inertias for joint i

Ia_i - rotational inertia of servomotor i

C_{ijj}, C_{ijk} - centripedal and coriolis forces for joint i

G_i - gravity loading for joint i

H_i - friction coefficient for joint i

N_i - drive gear ratio for joint i

k_i, ℓ_i - servomotor i torque and voltage constants

R_i, L_i - armature resistance and inductance of motor i

The G, D and C terms may be expanded as:

$$G_i = \sum_{p=i}^{3} -m_p g^T \frac{\delta T_p}{\delta q_i} {}^p\bar{r}_p \tag{2}$$

$$D_{ij} = \sum_{p=i}^{3} \text{trace} \left[\frac{\delta T_p}{\delta q_j} J_p \frac{\delta T_p^T}{dq_i} \right] \tag{3}$$

$$C_{ijk} = \sum_{\max(i,j,k)}^{3} \text{trace} \left[\frac{\delta^2 T_p}{\delta q_j \delta q_k} J_p \frac{\delta T_p^T}{\delta q_i} \right] \tag{4}$$

Note that:

$$g^T = (g_x \; g_y \; g_z \; 1) \quad , \quad |g| = 9.81 \text{ m/s}^2 \tag{5}$$

${}^p\bar{r}_p$ is the centre of mass of link p w.r.t. co-ordinate frame p.
J_p is the Pseudo Inertial Matrix [2], containing inertias about axes of co-ordinate frame p and components of link centres of mass w.r.t. co-ordinate frame p.

T_p is a transformation matrix used to transform a point described w.r.t. link p co-ordinates to base co-ordinates.

For example, given a point ${}^pr$ described with reference to link p co-ordinates, the same point in base co-ordinates is given by:

$$r = T_p {}^p r \tag{6}$$

and the velocity of point r by:

$$\frac{dr}{dt} = \sum_{j=1}^{p} \frac{\delta T_p}{\delta q_j} \dot{q}_j \; {}^p r \tag{7}$$

The above equations indicate the complexity of the robot equations, but a superficial analysis is sufficient to indicate that the model is multivariable, nonlinear and possibly time varying [3]. In addition, the PUMA 560, like most other manipulators has a certain amount of redundancy associated with it - not only is there a large degree of freedom in how the end point is attained, but there are, in general, a number of possible

combinations of final joint angles which will achieve a desired steady-state end effector position. The redundancy problem, however, which is a result of the non-uniqueness of the inverse kinematic solution, will not be treated in any depth in the current analysis.
Equation (1) describes the relationships between the voltages applied to the servomotors and the various joint angles, angle velocities and accelerations. Note that the equation for joint i contains terms involving the other joints as well. This indicates the multivariable nature of the system. The degree of crosscoupling is very significant, not only due to the variables q, $\dot{q}$ and $\ddot{q}$ for other joints appearing in the equation for joint i, but also a result of the matrices G, D and C (see equations (2) -> (4)) which are either directly dependent on the variables corresponding to other links or indirectly, via the transformation matrices.

The nonlinearity of the system is evident from equation (1) due to the product of variables and again indirectly due to the dependance of the matrices G, D and C on the operating point (i.e. nominal value of the system variables).
Given a fixed load, the parameters of the manipulator are constant, and if accurate measurements of the system variables are available, the dynamic behaviour of the system is entirely predictable from the above equations. In addition, the system is time invariant in the sense that given a particular operating point:

$$X(t_1) = (q(t_1) \quad \dot{q}(t_1) \quad \ddot{q}(t_1)) \tag{8}$$

where

$$q(t)^T = (q_1(t) \quad q_2(t) \quad q_3(t)) \tag{9}$$

the dynamics of the system will be exactly the same at time t_2 as they were at time t_1 so long as:

$$X(t_2) = X(t_1)$$

However, changes in the system load (which, in general, is unmeasurable, unless known in advance) will cause variations in the G, D and C matrices (see equations (2) -> (4)) due to the variation in the effective mass of link 3 (the outermost link) and the centre of mass of link 3. These parameters are reflected directly in the m_p, $^p\bar{r}_p$ and J_p terms.

The time-varying nature of a system is the single most important reason for the inclusion of adaptation in the corresponding controller.

Comments here.

3. MANIPULATOR CONTROL SYSTEMS

Robot controllers can be viewed as hierarchical control systems (see Fig.2) where the wider aspects of the system behaviour are dealt with by the upper levels, with the speed requirements increasing as one progresses downwards through the levels.

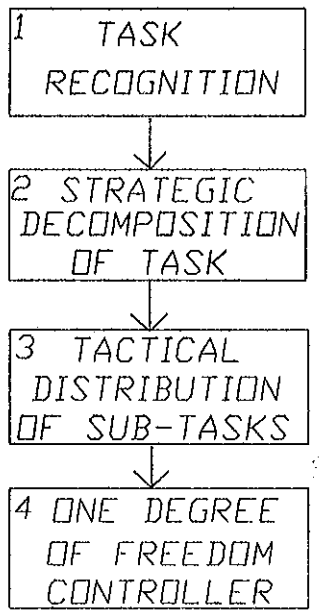

Figure 2: Hierarchical Manipulator Controller

The four levels most commonly encountered [3] are:

(a) Level 1 - which recognises the obstacles in the operating space and makes decisions on how the required task is to be accomplished.

(b) Level 2 - which devides the desired motion (from (a)) into elementary movements.

(c) Level 3 - which distributes the elementary movements to each degree of freedom of the robot.

(d) Level 4 - which executes the required movement of each degree of freedom.

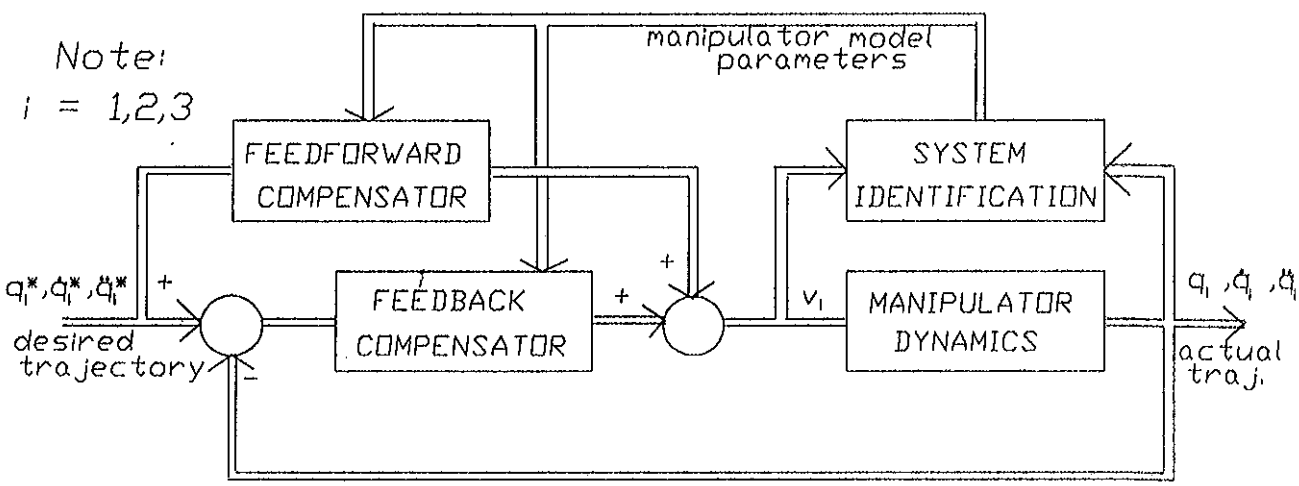

Figure 3: General Manipulator Dynamic Controller

In the current analysis, the emphasis will be placed on level 4 with some consideration given to level 3. Fig. 3 shows a flexible structure for the dynamic control of a manipulator.

4. DYNAMIC CONTROL LAWS

The past 25 years have seen major developments in the variety and sophistication of dynamic control algorithms. Some of these developments have been prompted by particular application requirements (e.g. adaptive control was a solution to problems encountered in aircraft autopilot design) and others due to consistant attempts to improve the performance of control algorithms. A major factor which paralleled this development was the advent of the microprocessor and the availability of cheap computing power.

The two most significant achievements over this period were the development of sound adaptive/self-tuning methods (for a survey see [4]) and the conception of robust control design techniques [5][6].

Dynamic control strategies may be classified as follows:

- Adaptive / non-adaptive - in an adaptive system, the controller parameters may be varied in accordance with some auxiliary measurement (e.g. the system output) or adjusted using some design criteria based on an identified process model.
- Linear / non-linear - in a linear controller the control signal is always a linear function of the system output or error signal.
- Robust / non-robust - using a robust controller formulation, the system may be made optimally insensitive to variations in the plant parameters or disturbances.
- Scalar / Multivariable - in a multivariable controller, the value of a single control signal is determined from a combination of the system outputs.

Given the nature of a robot model (as outlined in Section 2) it would seem appropriate that the controller should be multivariable (to account for the interactions) and either robust or self-adaptive to cater for the varying/non-linear nature of the system.

Unfortunately, robust design techniques for non-linear systems have not yet been

developed, although designs based on linearised system models may be able to cope with the parameter variation due to the nonlinearity. In the case of the robot, apart from the difficulty of coming up with a meaningful linearised model, the severity of the nonlinearity is probably too great for such a design to cope with. It is possible, however, to have a self-tuning version of a robust controller and such algorithms are now beginning to appear in the literature [7]. The benefit of such a scheme is that the adaptor makes the necessary compensation for the parameter variation, with the robust element giving the insensitivity to unmodelled dynamics. Such algorithms, however, are highly complex.

5. APPROACHES TO MANIPULATOR CONTROL

One of the simplest forms of dynamic manipulator control (and the most popular one in commercial manipulator controllers) is fixed gain linear feedback. Schemes employing PID structures are reported in [8] (documenting the Puma 560 controller) and [9]. Luo et al [10] adopt an optimisation approach (Linear Quadratic) to the determination of the P, I and D parameters. In [11], a PID controller is replaced with another simple linear controller of the lead-lag type. Fu et al [8], however, report that such simple controllers do not perform well under varying speeds and payloads and that the Puma arm moves with noticeable vibration at low speeds (using the Unimate controller).

A more complex non-adaptive control law is given by the linear optimal control solution [3]. However, most researchers combine the kinematic and dynamic problem in the optimisation framework [12][13], resulting in an optimal path planning type of solution. The use of optimal control techniques for the dynamic problem alone using a conventional quadratic cost function is not likely to be a successful solution since, depending on cost function weights, optimal controllers can be very highly tuned to a nominal plant and could not cater for nonlinear effects and load variations.

An alternative approach is the computed torque method [8], which, using an accurate model of the system, dynamically evaluates the torque required by each servo to track a desired trajectory. Such schemes are widely reported [14], [15]. Computed torque algorithms have the advantages of feedforward control in that improved transient response over feedback systems is possible. Feedforward algorithms, however, are very sensitive to

unmodelled dynamics, which may result from modelling inaccuracies or dynamic load variation. Linear position and velocity feedback is normally employed to compensate for this [16].

Gu and Loh [17] use non-linear feedback to parameterise the system in terms of an "imaginary" linear robot model. A PD type control law is then used to control the modified system. This method can becompared with the computed torque technique in which nonlinear compensation (an inverse dynamic model of the system) is placed in the *feedforward* path. However, despite a number of seemingly successful computed torque implementations, Leahy et al [15] conclude that 'computed torque performance is unacceptable as a real-time gross motion controller'.

The bulk of the recent literature on dynamic manipulator control has concentrated on adaptive systems. The simplest form of adaptive technique uses a gain scheduling technique to switch in different controller (e.g. PID) parameters in response to different operating conditions (positions, velocities and accelerations)[4]. Once commissioned, however, these schemes do not have the capability for further adaptation and hence cannot adjust for load variations. One further problem is the initial derivation of the relationship between the parameters and the operating conditions. One technique for automatically deriving P, I and D parameters is described in [18].

Linear adaptive regulators [19][20] attempt to fit a linear autoregressive model to the input/output data obtained from the robot. A set of control design equations are then used to transform the plant model parameters to controller parameters. Such adaptive schemes are very flexible, in that a wide variety of both identification and controller design methods exist, resulting in a large number of possible combinations. A scheme which uses extended least squares identification with an LQG controller formulation is documented in [21]. However, methods based on linear models carry the assumption that the model coefficients vary slowly compared to the system variables (q, $\dot{q}$, $\ddot{q}$). Modern manipulators move so fast that the effective inertia at a given joint may change by up to 300% in a fraction of a second [22], thus very fast sampling rates in conjunction with fast-converging identification algorithms should be used.

An adaptive scheme with a very strong intuitive apppeal consists of an adaptive feedforward (computed torque) where the robot model and payload parameters are identified on-line. To account for model inaccuracies and the poor disturbance rejection

properties of open-loop control, a linear feedback scheme using PD compensators may be used. Such schemes are reported in [22], [23] and [24].

Liu [25] draws an interesting comparison between two methods which both contain nonlinear feedforward, but one having constant PD feedback while the other identifies a 2nd order ARMA model and applies variational optimal control. No significant difference between the results is noted. A similar study is performed by Lee and Chung [26], who conclude that, for all of the cases examined, the adaptive controller was superior both in trajectory tracking and the final position errors.

A popular adaptive method which has been used sucessfully with many other types of dynamic system is *Model Reference Adaptive Control* (MRAC). In this philosophy, an updating mechanism is used in conjunction with the controller parameters such that the overall system has a response similar to a "reference" model (with a desired response). Han et al [27] propose a non-linear reference model which includes identification of an unknown load and a nonlinear controller. Linear reference models are specified in [28] and [29]. Lim and Eslami [28] use position and velocity feedback and position feedforward to achieve an overall multivariable state-space model which is stable and controllable. Seraji [29], on the other hand, uses single joint feedback (PD) and feedforward (q, $\dot{q}$, $\ddot{q}$) controllers, with a disturbance term accounting for the interacting forces. The reference model is specified in terms of a resonant frequency and damping factor.

Two similar adaptive schemes are presented by Liu et al [30] and Lee and Chung [26] which use a combination of nonlinear feedforward and linear feedback, both multivariable. A dynamic robot model is used to determine nominal torques which compensate for interaction forces along a nominal trajectory. Linearised *perturbation* (or incremental) models are then determined about the nominal trajectory upon which linear feedback schemes are based. It is claimed that the perturbation models take account of dynamic interaction. The linear feedback scheme in [30] is based on a generalised minimum variance (implicit) control law, whereas in [26] least squares is used to explicitly identify a model upon which a one-step ahead optimal control signal is calculated.

Other approaches to the dynamic robot control problem include the application of singular perturbation methods [31] and the treatment of a robot as a variable structure

system [8][32]. The latter methodology is interesting in that an accurate model of the system is not required - the bounds of the model parameters are sufficient to construct the controller. Furthermore, the controller forces the manipulator into a *sliding mode*, where the interactions among the joints are completely eliminated. However, the controller produces a discontinuous signal which changes sign rapidly which may produce "chattering".

An area of control systems which has received much attention recently is intelligent control or the application of A.I. techniques to controllers. In some of these techniques, it is not necessary to have an accurate system model and in other cases it is not even necessary to know the structure of the model. In [33], a learning algorithm is used to reproduce the relationship between sensor outputs and system command variables both for repetitive and non-repetitive tasks. No *a priori* knowledge of the system is required. [34] specifies an impulse response model type where the closed loop behaviour is defined by the reference tarjectory. Control computation is performed using a model predictive heuristic procedure. Another iterative learning control method is described in [35], where a linear state-space model of the system is used, the coefficients of the system matrices assumed to be periodic functions of time. Bondi et al [36] also present an iterative learning method, but follow a strict mathematical argument.

6. A NON-LINEAR SELF-TUNING APPROACH

A solution to the dynamic manipulator control problem is considered, where single loop non-linear quadratic gaussian (NLQG) compensators are employed [37]. Leahy et al [15], having performed a range of tests on a Puma robot, conclude that the effects of Coriolis and centripedal forces are negligable, which comprise some of the main potential interaction effects in a multi-link robot (see equations (1) and (4)). For decentralised (single-loop) control, Seraji [29] proposes the following model decomposition:

$$m_{ii}(\Theta)\ddot{\Theta}_i(t) + d_i(\Theta, \dot{\Theta}, \ddot{\Theta}) = T_i(t) \tag{10}$$

where

$$d_i(\Theta,\dot{\Theta},\ddot{\Theta}) = \sum_{\substack{j=1 \\ j\neq i}}^{3} m_{ij}(\Theta)\ddot{\Theta}_j(t) + c_i(\Theta,\dot{\Theta}) + g_i(\Theta) + h_i(\dot{\Theta}) \tag{11}$$

with the obvious identification of terms from equation (1). d_i is the effective disturbance

to joint i due to coupling effects. Note that equation (10) is written in terms of a torque input, there being a first-order relationship between voltage (as in eq (1)) and torque inputs.

In the current approach, the objective is to apply non-linear quadratic gaussian control to each individual joint (eq (10)) and reject the disturbance in eq (11). Two model structures are considered, which are both non-linear extensions of the basic discrete-time ARMAX model:

$$A(z^{-1})y(k) = B(z^{-1})x(k) + C(z^{-1})\xi(k) \tag{12}$$

where A, B and C are scalar n^{th} order polynomials in the delay operator (z^{-1}), y is the system output, ξ is a white noise sequence and x is an intermediate system input, where:

$$x(k) = f_0 + u(k) + f_1u^2(k) + \ldots\ldots + f_mu^m(k) \tag{13}$$

for a Hammerstein Plant [38], and

$$x(k) = \mathbf{u}^T B_m \mathbf{u} \tag{14}$$

where

$$\mathbf{u}^T = (u(k) \quad u(k-1) \quad \ldots . u(k-m)) \tag{15}$$

and

$$B_m = \begin{bmatrix} \beta_{oo} & \beta_{o1} & \ldots . & \beta_{om} \\ 0 & \beta_{11} & \ldots . & \beta_{1m} \\ 0 & 0 & \ldots . & \beta_{2m} \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots . & \beta_{mm} \end{bmatrix} \tag{16}$$

for a Volterra type plant [39].

A third order ARMAX model (n = 3) is used, from considerations of equation (1). Leahy et al [15] recommend the consideration of actuator dynamics, including inertias.

Since the system equations are linear in the parameters [39], the parameters of these models may be identified using recursive *Extended Least Squares* (RELS) in a 2 stage procedure for the Hammerstein model (n+m parameters) and a 3 stage algorithm for the Volterra model (n+[m/2][m+1] parameters).

In RELS, a disturbance model (the C polynomial coefficients) is identified, allowing disturbance compensation to be performed, if desired. Also note that in equation (13) the f_0 parameter corresponds to an external (uncontrolled) d.c. input, for example a gravity

term (see equation (11)). Leahy et al [15] report that gravity forces are significant and should be modelled.

The Volterra model, due to its extra complexity, provides a more flexible model (it incorporates a 2nd order Hammerstein model) and a greater degree of dynamic nonlinearity.

The solution to the NLQG problem for Hammerstein and Volterra plants is documented in [37]. To illustrate the type of computations involved, one pass through the self-tuning algorithm for a Hammerstein plant is outlined here as:

(a) Perform one recursion of the RELS algorithm to give updated estimates for the A, B and C polynomial coefficients and the f_0 -> f_m nonlinear parameters.

(b) Determine the controller parameters by solving for the spectral factor D from:

$$D\,D^* = B\,B^*\,Q + A\,A^*\,R \tag{17}$$

and computing the G and H polynomial coefficients from the diophantine equation:

$$A\,H + B\,G = D\,C \tag{18}$$

where A, B and C in eq (12) are given from step (a), and Q and R are weighting matrices in the quadratic cost function:

$$J = E\,(\,Q\,e^2(k) + R\,u^2(k)) \tag{19}$$

where E( ) denotes the expectation operator and e(k) is the angle error in the system (desired angle minus actual angle).

(c) Update the signal x(k) using:

$$x(k) = G(z^{-1})/H(z^{-1})\;e(k) \tag{20}$$

(d) Compute the optimal control signal u(k) as the minimum magnitude root of the equation:

$$(f_0 - x(k)) + u(k) + f_2u^2(k) + \ldots + f_mu^m(k) = 0 \tag{21}$$

Note that the weighting matrix Q may be dynamical, an integral term ensuring that the system has zero steady-state error. Note also that, since the Q and R matrices appear directly in the update equations (equation (17)), they may be varied during the movement of the robot arm along desired trajectory. This allows for the possibility of energy saving (large R => small control signals) during the early part of the movement, with more accurate control (larger Q) as the end point is reached.

Simulation results are reported in [21] for a *linear* version of the above algorithm (i.e. LQG) used with a Puma 560 robot model. Initial simulation experiences with the nonlinear version seem to be a considerable improvement.

7. COMPUTATIONAL CONSIDERATIONS

In addition to the difficulty of controlling the complex dynamics of a manipulator, the speed of response of the dynamics places great demands on a real-time control system. The benefits of digital control are well known (e.g. flexibility, ease of implementation of complex algorithms, accuracy), but to successfully implement a digital control scheme, it must be possible to perform all of the control computations with the sampling period appropriate to a particular manipulator. This may include inverse kinematics, recursive computation of the Newton–Euler equations, system identification and adaptive controller calculations. Fortunately, cheap, high-performance processing power is available in the form of general-purpose microprocessors (μP's) and support chips (floating point units (FPU's), memory management units (MMU's) and interrupt control units (ICU's)), dedicated digital signal processing (DSP) chips and a host of array and parallel processing machines.

As a typical example, consider the Puma robot - Nigam and Lee [40] found the natural resonant frequency of the Puma 560 to be 15 Hz, thus a sampling frequency of 300Hz was chosen, giving a sampling period of about 3mS. Current practice in the commercial Unimate Puma controller involves a loop sampling period of 0.875 mS with new position setpoints being provided every 28 mS [8] (otherwise joints jerk erratically when servod [41]).

Comp

7.1 Algorithm complexity

A measure of the complexity of a control algorithm, and one which determines the hardware requirements, is the number of additions and multiplications required per sampling period. Given the nature of the control problem, it is assumed that all numbers are in floating point format. It is necessary to distinguish between additions and multiplications due to the vastly different computational time involved for each on machines such as general purpose μP's.

A reasonable amount of documentation is available in the computational aspects of the computed torque technique. In addition, research has been undertaken in formulating efficient computational algorithms for evaluating manipulator dynamics [42][43]. Nigam and Lee [40] estimate the number of operations for computed torque to be 662(+) and 792(x) for 6 joint control using the Newton-Euler equations. In contrast, for the Lagrange-Euler formulation, the number of operations soared to 78000(+) and 102000(x). Liu and Chen [44] implement a computed torque scheme on the Stanford manipulator and achieve a 3.3mS sampling period (uning a 68020/68881 combination). Significantly, they claim that this is less complicated than a Puma implementation due to the Stanford's prismatic joint. The same authors evaluate an adaptive feedback controller (with inverse dynamic feedforward) as documented in [25] but no extra computational information is provided (apart from the inverse dynamic section), other than to say that an extra 68020/68881 combination is required for each adaptive joint controller. Seraji [29] compares computations required for control of p joints using multivariable ($5n^2$ + n adaptive gains) and decentralised (6n gains) versions of his adaptive algorithm (see Section 5).

Lee and Chung [26] provide details of the computations involved in an adaptive feedforward (computed torque) and linear adaptive scheme (see Section 5) as:

Section	Mults.	Adds.
N-E eqtns.	$117n - 24$	$103n - 21$
RLS (I.D.)	$30n^2+5n+1$	$30n^2+3n-1$
Control	$8n^3+2n^2+39$	$8n^3-n^2-n+18$
Total	$8n^3+32n^2+5n+40$	$8n^3+29n^2+2n+17$

For a three joint manipulator this works out at 559(x) and 500(+). For six joints the total is 2950(x) and 2801(+) for comparison with Nigam and Lee [40] above.

For the proposed algorithm (Section 6), using a third order dynamic model (A, B and C) and a fourth order nonlinearity, the number of computations per joint is:

Section	Mults.	Adds.	
RELS(A,B,C)	1162	900	
RLS (f_0->f_4)	157	100	
Solve (17)	1512	1512	*
Solve (18)	266	165	
Solve (21)	91	63	*
Total	3188	2740	

Note that the calculations marked (*) are iterative - an average number of required iterations is used.

7.2 Computational Hardware Options

Computational hardware for a robot controller must be capable of performing all of the functions indicated in Fig.2. The upper two layers of this robot controller must be performed in sequence and, in general, will be performed by a single processor. The lower two layers (but principally the bottom one) implement the dynamic control. This can be performed using a distributed system (decentralised control) or a single processor.

A wide variety of schemes are reported in the literature. Probably the simplest is that used in the commercial Unimate Puma robot controller [8]. This consists of a central LSI-11 processor (DEC PDP-11/23) with six Rockwell 6503 (8-bit μP's) as the individual joint controllers. Another approach [45] replaces the Unimate controller with an 80286-based Intel System 310. While providing more flexability (the user is not tied to VAL II), this modification would seem to provide little improvement, if any, in computational power. Seraji [29] replaces the Unimate controller with a μVAX which implements all 4 layers of Fig.2 on the three major joints. Two alternative architectures for a Puma are reported in [41] - (a) a SIERA [46] system, based around multiple 68000-based single board computers, a custom developed Armstrong multiprocessor system and two SUN 3/260 computers and (b) a TUNIS [47] system, consisting of one ZP1632 master processor board and up to four similar slave boards. The ZP1623 is based on Nat. Semi's 32000 chip set comprising 32016 CPU, MMU, ICU and FPU.

An architecture which is very much tailored to the control algorithm is proposed by Liu and Chen [44] who use 1 μP for inverse kinematics and path planning, 2 μP's for the inverse dynamics (feedforward) and one μP per joint to implement the adaptive controllers. The μP's used are either 80286/80287 or 68020/68881 (68020/68881 is preferred) combinations.

As a departure from the more traditional computing architectures, Khosla and Kanade [14] consider the use of a Marinco array processor in conjunction with a 68000-based system plus TMS320 controllers for each joint. This would appear to be one of the most powerful processor arrangement and indicates the difficulty of implementing the computed torque technique. A sampling period of 2mS was achieved in this application.

Another application which implements a dynamic robot model [40] applies pipelining techniques. Normally, in feedback controllers, pipelining is not possible since the control input must be evalued based on the current error in the system. In this case (a feedforward situation), the setpoints are known in advance from the path-planning stage, so the current control input is based on a setpoint received a number of steps (sampling periods) previously. In this example, six stages of pipelining are used, deriving maximum benefit from the serial nature of the Newton–Euler equations of motion. The pipeline implementation uses either 80286 or 68020 μP's, although a number of other architectures are considered, all the way up to a ($60,000) Floating Point Systems AP-120B array processor coupled to a VAX!

The approach used in conjunction with the algorithm presented in Section 6 is a 20MHz 80386/80387 IBM PC/AT together with one NEC μPD77230 - based signal processing board for each individual joint controller. The μPD77230 is designed specifically for 32 bit floating point calculations and is rated at 150 nS for a floating point multiply/ accumulate. A reasonable sampling rate should be achievable, since the computations for the given algorithm should take an estimated 0.889 mS.

Considering the type of calculations common to all classes of dynamic robot controllers, it would seem that signal processing type hardware dedicated to floating point calculations is most suitable. The processing of arrays would also seem to be an integral part of most controller calculations (e.g. computing robot dynamics, identification, etc.) but the addition of most array processors would be highly uneconomic [40]. However, one device from Intel which has recently been announced is the i860 [48], which has a

RISC-based architecture and 1 000 000 transistors. The significant feature of this chip is the provision of vector processing capability at only $750 (for a 33MHz version). In addition, all of the floating point instructions are implemented in a single cycle. It's overall performance has been rated at about half of a CRAY-1 and would seem to be ideally suited to robot control applications.

8. CONCLUSIONS

A number of approaches to the dynamic robot control problem have been researched. Though a wide variety of algorithms are available offering a range of complexity/performance tradeoffs, most still have to find their way into industrial practice. Klafter et al [42] comment that 'robots currently being produced are perceived (by the companies) to be "good enough" for the applications of today'. They also comment that it is fortunate that universities have not been as shortsighted!

The most significant classification of controller algorithms seems to be that of centralised or decentralised control. Centralised would seem to offer better performance (since cross coupling terms are directly compensated) but with greater computational burden (n^2 computations as opposed to 3n for a 3 joint controller). It has been seen, though, that hardware should no longer be a limiting factor. However, the decentralised scheme naturally possesses better integrity characteristics, since an erraneous joint angle measurement will affect only that joint and not be propagated to others.

The chosen method for control should therefore depend on a number of factors, not least the nature of the application itself. For applications where the payload is constant (e.g. welding, spray painting, grinding/deburring), a control technique with fixed parameters may suffice (assuming effects due to wear and manufacturing tolerances are not too significant). On the other hand, for applications with varying loads (e.g. parts handling/transfer, assembly or sorting operations) or where the load itself posesses significant dynamics, a self-tuning/adaptive controller may be required.

ACKNOWLEDGEMENT

The author wishes to acknowledge the support of the EC under the Science Stimulation Programme. Cooperation with our partners at the Industrial Control Unit at the University of Strathclyde, particularly Ms. S. Carr, Prof. M.J. Grimble and Dr. M.A. Johnson is also acknowledged.

REFERENCES

1. Klafter, R.D., Chmielewski, T.A. and Negin, M. "Robotic engineering - an integrated approach", Prectice-Hall, 1989.
2. Anderson, G. "Modelling and simulation of a Puma 560 manipulator for control system appraisal", M.Eng. Thesis, School of Electronic Engineering, NIHE, 1988.
3. Vukobratović, M. and Stokić, D. "Scientific fundamentals of robotics 2 - Control of manipulation robots", Springer-Verlag, 1982.
4. Åstrom, K.J. "Theory and applications of adaptive control - a survey", Automatica, Vol.19, No.5, 1983.
5. Kwakernaak, H. "A polynomial approach to minimax frequency domain optimization of multivariable feedback systems", Int. J. Contr. Vol.44, No.1, 1986.
6. Zames, G. "Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses", IEEE Trans. Aut. Contr. Vol.AC-26, No.4, 1981.
7. Grimble, M.J. and Fairbairn, N.A. "The F-iteration approach to H^{∞} control", Proc. IFAC Symp. on Adaptive Systems in Control and Signal Processing, Glasgow, April 1989.
8. Fu, K.S., Gonzalez, R.C. and Lee, C. "Robotics - control, sensing, vision and intelligence", McGraw-Hill, 1987.
9. McCloy, D. and Harris, M. "Robotics - An introduction", Open University Press, 1986.
10. Luo, G.L. and Saridis, G.N. "L-Q design of PID controllers for robot arms", IEEE Journal of Robotics and Automation, Vol.RA-1, No.3, Sept 1985.
11. Chen, Y. "Replacing a PID controller by a lead-lag compensator for a robot - a frequency response approach", IEEE Trans. Rob. and Auto., Vol.RA-5, No.2, April 1989.

12. Ailon, A. and Langholz, G. "A study of controllability and time-optimal control of a robot model with drive train compliances and actuator dynamics", IEEE Trans. Auto. Cont., Vol.AC-33, No.9, Sept. 1988.
13. Chen, Y.C. "On the structure of the time-optimal controls for robotic manipulators", IEEE Trans. Auto. Cont., Vol.AC-34, No.1, Jan. 1989.
14. Khosla, P.K. and Kanade, T. "Real-time implementation and evaluation of computed torque scheme", IEEE Trans. Rob. and Auto., Vol.RA-5, No.2, April 1989.
15. Leahy, M.B., Valavanis, K.P. and Saridis, G.N. "Evaluation of dynamic models for PUMA robot control", IEEE Trans. Rob. and Auto., Vol.RA-5, No.2, April 1989.
16. Khosla, P.K. and Kanade, T. "Experimental evaluation of nonlinear feedback and feedforward control schemes for manipulators", Robotics Research, Vol.7, No.1, Feb. 1988.
17. Gu, Y.L. and Loh, N.K. "Dynamic modelling and control by using an imaginary robot model", IEEE Jour. of Robotics and Automation, Vol.4, No.5, Oct. 1988.
18. Chen, Y. "Parameter fine-tuning for robots", IEEE Control Systems Magazine, Vol.9, No.2, Feb. 1989.
19. Walters, R.G. and Bayoumi, M.M. "Application of a self- tuning pole placement regulator to an industrial manipulator", Proc. 21st IEEE CDC, Vol.1, Orlando, Florida, Dec. 1982.
20. Koivo, A.J. "Force-position-velocity control with self- tuning for robotic manipulators", IEEE Conf. Robotics and Automation, San Francisco, CA, 1986.
21. Carr, S., Anderson, G.,Grimble, M.J. and Ringwood, J.V. "An LQG approach to self-tuning control with application to robotics", Proc. IEE Intl. Workshop on Robot Control, Oxford, U.K., April 1988.
22. Craig, J.J., Hsu, P. and Sastry, S.S. "Adaptive control of mechanical manipulators", Robotics Research, Vol.6, No.2, Sum. 1987.
23. Slotine, J.E. and Li, W. "Adaptive manipulator control: A case study", IEEE Trans. Auto. Cont., Vol.AC-33, No.11, Nov. 1988.
24. Slotine,J.E. and Li, W. "On the adaptive control of robot manipulators", Robotics Research, Vol.6, No.3, Fall 1987.
25. Liu, C.H. "A comparison of controller design and simulation for an industrial manipulator", IEEE Trans. Ind. Electron., Vol.IE-33, No.1, Feb. 1986.
26. Lee, C.S.G. and Chung, M.J. "Adaptive perturbation control with feedforward compensation for robot manipulators", Simulation, Vol.44, No.3, 1985.

27. Han, J-Y., Hemami, H. and Yurkovitch, S. "Nonlinear adaptive control of an N-link robot with unknown load", Robotics Research, Vol.6, No.3, Fall 1987.
28. Lim, K.Y. and Eslami, M. "Robust adaptive controller designs for robot manipulator systems", IEEE Jour. of Robotics and Automation, Vol.RA-3, No.1, Feb. 1987.
29. Seraji, H. "Decentralised adaptive control of manipulators: Theory, simulation and experimentation", IEEE Trans. on Robotics and Automation, Vol.AC-5, No.2, April 1989.
30. Liu, M.H., Chang, W.S. and Zhang, L.Q. "Multivariable self-tuning control of redundant manipulators", IEEE Jour. of Robotics and Automation, Vol.RA-4, No.5, Oct. 1988.
31. Siciliano, B. and Book, W.J. "A singular perturbation approach to control of lightweight flexible manipulators", Robotics Research, Vol.7, No.4, August 1988.
32. Hashimoto, H, Maruyama, K and Harashima, F. "A microprocessor-based robot manipulator control with sliding mode", IRRR Trans. Ind. Electron., Vol.IE-34, No.1, Feb. 1987.
33. Miller, W.T. "Sensor-based control of robotic manipulators using a general learning algorithm", IEEE Jour. of Robotics and Automation, Vol.RA-3, No.2, April 1987.
34. Kaynak, O., Melancon, P and Rajagopalan, V. "Model predictive hueristic control of a position servo system in robotics application", IEEE Jour. of Robotics and Automation, Vol.RA-3, No.5, Oct. 1987.
35. Oh, S.R., Bien, Z. and Suh, I.H. "An iterative learning control method with application for the robot manipulator",. IEEE Jour. of Robotics and Automation, Vol.RA-4, No.5, Oct. 1988.
36. Bondi, P., Casalino, G. and Gambardella, L. "On the iterative learning control theory for robotic manipulators", IEEE Jour. of Robotics and Automation, Vol.RA-4, No.1, Feb. 1988.
37. Carr, S. and Grimble, M.J. "Nonlinear quadratic gaussian self-tuning control", Research Report ICU/228/October 1988, Industrial Control Unit, University of Strathclyde.
38. Anbumani, K., Sarma, I.G. and Patnaik, L.M. "Self-tuning control of nonlinear systams characterised by Hammerstein models, Proc. 8th IFAC World Congress, Kyoto, Japan, 1981.
39. Haber, R. and Keviczky, L. "Nonlinear structures for system identification", Periodica Politechnica, Vol.18, No.4, 1974.

40. Nigam, R. and Lee, C.S.G. "A multiprocessor-based controller for the control of mechanical manipulators", IEEE Jour. of Robotics and Automation, Vol.RA-1, No.4, Dec. 1985.
41. Goldenberg, A.A. and Chan, L. "An approach to real-time control of robots in task space. Application to control of PUMA 560 without VAL-II", IEEE Trans. Ind. Electron, Vol.IE-35, No.2, May 1988.
42. Kanade, T.K., Khosla, P.K. and Tanaka, N. "Real-time control of the CMU direct-drive arm II using customised inverse dynamics", Proc 23rd CDC, Las Vegas, 1984.
43. Burdick, J.W. "An algorithm for generation of efficient manipulator dynamic equations", Proc. IEEE Int. Conf. Robotics and Automation, San Francisco, 1986.
44. Liu, C. and Chen, Y. "Multi-microprocessor-based cartesian space control techniques for a mechanical manipulator", IEEE Jour. of Robotics and Automation, Vol.RA-2, No.2, June 1986.
45. Bihn, D.G. and Hsia, T.C.S. "Universal six-joint robot controller", IEEE Control Systems Magazine, Vol.8, No.1, Feb 1988.
46. Kazanzides, P., Wasti, H. and Wolovich, W.A. "A multiprocessor system for real-time robotic control: design and applications", Proc. IEEE Int. Conf. Robotics and Automation, 1987.
47. Penny, D. "Control of the PUMA robot without VAL", Univ. Toronto, RAL Tech. Rep., April 1985.
48. Perry, T.S. "Intel's secret is out", IEEE Spectrum, Vol.26, No.4, April 1989.