Predictive Functional Control for a Parallel Robot
Oscar Andrès Vivas, Philippe Poignet, François Pierrot

To cite this version:

HAL Id: lirmm-00191950
https://hal-lirmm.ccsd.cnrs.fr/lirmm-00191950
Submitted on 26 Nov 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Abstract: This paper presents an efficient application of a model based predictive control in parallel mechanisms. A predictive functional control control strategy based on a simplified dynamic model is implemented. Experimental results are shown for the H4 robot, a fully parallel structure providing 3 degrees of freedom (dof) in translation and 1 dof in rotation. Predictive functional control, computed torque control and PID control strategies are compared in complex machining tasks trajectories. The tracking performances are enlightened.

1. INTRODUCTION

Parallel mechanisms were introduced by Gough [1] and Steward [2]. Clavel [3] proposed the Delta structure, a parallel robot dedicated to high-speed applications, that has intensively used in industry. This is due to the exceptional simplicity of the Delta 3-dof solution. For most pick-and-place applications, at least four dof are required (3 translations and 1 rotation to arrange the carried object in its final location). For the Delta robot, this is achieved thanks to an additional link between the base and the gripper, but it seems not to be as efficient as a parallel arrangement. On the other hand, 6-dof fully-parallel machines currently used in machining suffer from their complexity (they need at least 6 motors while the cutting process requires only 5 controlled axis plus the spindle rotation) and from their limited tilting angle. As an intermediate solution to these drawbacks, a 4-dof parallel mechanism – the H4 robot - have been proposed [4], [5]. Fig. 1 shows a photography of the H4 parallel robot.

This machine is based on 4 independent active chains between the base and the nacelle; each chain is actuated by a brushless direct drive motor fixed on the base and equipped with an incremental position encoder. Thanks to its design, the mechanism is able to provide high performances. In order to achieve high speed and acceleration for pick-and-place applications or precise motion in machining tasks, advanced model based robust controllers are often required to increase the performances of the robot. In the past decade model predictive control (MPC) has become an efficient control strategy for a large number of process [6]. Several works have shown that predictive control are of great interest when requiring good performances in term of rapidity, disturbances or errors cancellations [6], [7].

In this paper, we focus on the implementation of the predictive functional control (PFC) developed by Richalet [8], [9], on the H4 parallel robot. Basically the procedure will consist in two steps i) the process is first linearized by feedback ii) secondly the model predictive control scheme is computed from a linearized model composed of a set of double integrators firstly stabilized with an inner closed loop structure. Experimental results are compared with those obtained from the model based computed torque control (CTC) [10] and the clasical PID controller.

The paper is organized as follows: Section 2 is dedicated to the geometric, kinematics and dynamics modelling required to implement the control strategy. Section 3 details the model predictive functional control. Section 4 introduces the compared control schemes: predictive functional control, computed torque control and PID. Section 5 exhibits major experimental results in terms of tracking performances in complex machining trajectories. Finally, conclusions are given in section 6.

2. MODELLING

A. Geometric and kinematics modelling

The Jacobian matrix and the forward geometric model are required to compute the dynamic model (see section 2.B) [11]. Therefore we briefly present the way of computing the different relationship necessary to obtain these model and matrix. The design parameters of the robot are described on Fig. 2 where the following parameters have been chosen:

\[ \alpha_1 = 0; \quad \alpha_2 = \pi; \quad \alpha_3 = 3\pi/2; \quad \alpha_4 = 3\pi/2 \]

\[ u_1 = u_2; \quad u_2 = -u_1; \quad u_3 = u_4; \quad u_4 = u_x \]
The angles $\alpha_i$ describe the position of the four motors, $L$ is the length of arms, $l$ is the length of the forearms, $\theta$ the nacelle’s angle, and $d$ and $h$ are the half lengths of the "H" forming the nacelle. $O$ is the origin of the base frame and $D$ is the origin of the nacelle frame. $R$ gives the motor’s position. The $A_i B_i$ segments represent the arms of the robot and $P_i B_i$ the forearm segments. The joint positions are represented by $q_i$.

The analytical forward position relationship is difficult to compute. Up to now, the simplest model we’ve got is a 8th degree polynomial equation. The forward model is then computed iteratively using the classical formula:

$$x_{n+1} = x_n + J(x_n, q_n) [q - q_n]$$

(1)

Where $q$ is the convergence point and $J$ is the robot Jacobian matrix. If the mechanism is not in a singular configuration, this expression is derived as follows [4], [5] :

$$J = J_s^{-1} J_q$$

(2)

Where:

$$J = \begin{bmatrix} AB_1 & AB_2 & AB_3 & (DC_1 \times AB_1) \\ AB_2 & AB_3 & AB_4 & (DC_2 \times AB_2) \\ AB_3 & AB_4 & AB_5 & (DC_3 \times AB_3) \\ AB_4 & AB_5 & AB_6 & (DC_4 \times AB_4) \end{bmatrix}$$

(3)

$$J_q = \text{diag}(PB \times AB) u_{ui}, \quad i=1...4$$

(4)

$DC_i$ is the distance between the center of the nacelle and the center of the half lengths of the "H" that forms the nacelle and $u_{ui}$ the unit vector for each direction.

B. Dynamic modelling

In first approximation, the dynamic model is computed by considering physical dynamics. Indeed, drive torques are mainly used to move the motor inertia, the fore-arms and the arms and the nacelle equipped with a machining tool. Because of the design, the fore-arm inertia can be considered as a part of the motor inertia and the arm (manufactured in carbon materials) effects are neglected [4], [5].

If $\Gamma_{\text{mot}}$ is the (4x1) actuator torque vector, the basic equation of dynamics can be written as :

$$\Gamma_{\text{mot}} = I_{\text{mot}} \ddot{q} + J^T M(\dot{x} - G)$$

(5)

where $I_{\text{mot}}$ represents the motor’s inertia matrix including the forearm’s inertia, $M$ is a matrix containing the mass of the nacelle and its inertia, $x$ the vector of cartesian accelerations, and $G$ the gravity constant. Thanks to the design, the forearm’s inertia is taken into account in the motor’s inertia.

With:

$$I_{\text{mot}} = \text{diag} [I_{\text{mot}1}, I_{\text{mot}2}, I_{\text{mot}3}, I_{\text{mot}4}]$$

(6)

$$M = \text{diag} [M_{\text{nac}}, M_{\text{nac}}, M_{\text{nac}}, I_{\text{bc}}]$$

(7)

The motor position $q = [q_1 \quad q_2 \quad q_3 \quad q_4]^T$ are directly measured, and the velocity $\dot{q}$ and acceleration $\ddot{q}$ are obtained by central derivation. As the acceleration measurement $\ddot{x}$ is not available, $\ddot{x}$ is computed with:

$$\ddot{x} = J \ddot{q} + J_q \dot{q}$$

(8)

where $J$ depends on $x$ and $q$, $\dot{J}$ is computed using a central difference algorithm.

C. Identification

The dynamic parameters are estimated using weighted least square techniques. The estimated values, given in Table 1, will be considered as the nominal value during the experiments. More details concerning the identification procedure may be found in [10], [12], [13].

<table>
<thead>
<tr>
<th>TABLE 1</th>
<th>ESTIMATED PARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical parameters</td>
<td>Estimated values</td>
</tr>
<tr>
<td>$I_{\text{mot}1}$</td>
<td>0.0167 Nm²</td>
</tr>
<tr>
<td>$I_{\text{mot}2}$</td>
<td>0.0164 Nm²</td>
</tr>
<tr>
<td>$I_{\text{mot}3}$</td>
<td>0.0176 Nm²</td>
</tr>
<tr>
<td>$I_{\text{mot}4}$</td>
<td>0.0234 Nm²</td>
</tr>
<tr>
<td>$M_{\text{nac}}$</td>
<td>0.984 Kg</td>
</tr>
<tr>
<td>$I_{\text{bc}}$</td>
<td>0.0029 Nm²</td>
</tr>
</tbody>
</table>
3. PREDICTIVE FUNCTIONAL CONTROL

This section is dedicated to briefly recall the main steps of the model predictive functional control scheme used hereafter for the implementation. This predictive technique has been developed by Richalet and complete details of the computation may be found in [8], [9].

A. Internal Modeling

The model used is a linear one given by:

\[ x_M(n) = F_M x_M(n-1) + G_M u(n-1) \]
\[ y_M(n) = C_M^T x_M(n) \]

where:
- \( x_M \) is the state,
- \( u \) is the input,
- \( y_M \) is the measured model output,
- \( F_M, G_M \) and \( C_M \) are respectively matrices or vectors of the right dimension.

The problem of robustness because of the poles cancellation by the controller if the system is unstable is usually solved by a model decomposition [9].

B. Reference trajectory

The predictive control strategy of the MPC is summarized on Fig. 3. Given the set point trajectory on a receding horizon \([0, h]\), the predicted process output \( \hat{y}_p \) will reach the future set point following a reference trajectory \( y_R \).

\[ \varepsilon(n) = c(n) - y_p(n) \] is the position tracking error at time \( n \), \( c \) is the set point trajectory, \( y_p \) is the process output, CLTR is the closed loop time response.

On the prediction horizon, the reference trajectory \( y_R \), which is the path towards the future set point, is given by:

\[ c(n+i) - y_p(n+i) = \alpha^i (c(n) - y_p(n)) \] for \( 0 \leq i \leq h \)

where \( \alpha (0 < \alpha < 1) \) is a scalar which has to be chosen in function of the desired closed loop response time.

The predictive essence of the control strategy is completed included in Eq. 10. Indeed, the objective is to track the set point trajectory following the reference trajectory. This trajectory may be considered as the desired closed loop behaviour.

C. Performance index

The performance index may be a quadratic sum of the errors between the predicted process output \( \hat{y}_p \) and the reference trajectory \( y_R \). It is defined as follows:

\[ D(n) = \sum_{j=1}^{n_h} \left[ y_p(n+h) - y_p(n+h) \right] \]

where:
- \( n_h \) is the number of coincident time point,
- \( h \) are the coincidence time point on the prediction horizon.
- The predicted output \( \hat{y}_p \) is usually defined as:

\[ \hat{y}_p(n+i) = y_p(n+i) + \hat{e}(n+i) \] for \( 1 \leq i \leq h \)

where:
- \( y_p \) is the model output,
- \( \hat{e} \) is the predicted future output error.

It may be convenient to add a smoothing control term in the performance index. The index becomes:

\[ D(n) = \sum_{j=1}^{n_h} \left[ y_p(n+h) - y_p(n+h) \right] + \lambda [u(n) - u(n-1)] \]

where \( u \) is the control variable.

D. Control variable

The future control variable is assumed to be composed of a priori known functions:

\[ u(n+i) = \sum_{i=1}^{n_h} \mu_i(n) u_{base}(i) \] for \( 0 \leq i \leq h \)

where:
- \( \mu_i \) are the coefficients to be computed during the optimization of the performance index, \( u_{base} \) are the base functions of the control sequence, \( n_h \) is the number of base functions.

The choice of the base functions depends on the nature of the set point and the process. Hereafter we will use:

\[ u_{base}(i) = \begin{cases} 1 & i=1 \\ 0 & \text{otherwise} \end{cases} \quad \forall k \]

In fact, the only first term is effectively applied for the control, that is:

\[ u(n) = \sum_{i=1}^{n_h} \mu_i(n) u_{base}(0) \]
The model output is composed in two parts:

\[ y_{x}(n+i) = y_{ce}(n+i) + y_{e}(n+i) \quad 1 \leq i \leq h \]  
(17)

where:

\( y_{ce} \) is the free output response \((u = 0)\), \( y_{e} \) is the forced output response to the control variable given by Eq. 14.

Given Eq. 9 and Eq. 14, it follows:

\[ h_{n+i} = \sum_{i=0}^{h} B_{n+i} y_{e}(i) \quad 0 \leq i \leq h \]

where \( y_{e} \) is the model output response to \( u_{nc} \). Assuming that the predicted future output error is approximated by a polynomial, it follows:

\[ \hat{e}(n+i) = e(n) + \sum_{m=1}^{d_{p}} e_{m}(n) w_{m} \quad \text{for} \quad 1 \leq i \leq h \]  
(19)

where:

\( d_{p} \) is the degree of the polynomial approximation, \( e_{m} \) are some coefficients calculated on line knowing the past and present output error.

The minimization of the performance index without smoothing control term, in the case of the polynomial base functions, leads to the applied control variable:

\[ u(n) = k_{p}[e(n) - y_{e}(n)] + \sum_{m=1}^{d_{p}} k_{m}(e_{m}(n) - e_{m}(n)) + V_{f}^{T} x_{m}(n) \]  
(20)

where the gains are calculated off-line (see Appendix).

Therefore the control variable is composed of three terms:

- the first one is due to the tracking position error,
- the second one is placed especially for disturbance rejection, and
- the last one corresponds to a model compensation.

4. COMPARED CONTROL STRATEGIES

A. PID Controller

A typical PID controller for the H4 robot is shown in Fig 4. The gains tuning leads to \( K_p = 500, \ K_v = 5000, \) and \( K_\beta = 6 \) in order to guarantee the best behavior in tracking situation.

B. Feedback linearization

In order to compute the PFC control strategy [14] as well as for the CTC controller, it is basically required to linearize the non linear dynamic model of the robot. Let's consider the non linear dynamic equations for an m-link robot expressed as follows:

\[ \dot{x} = Ax + B\beta^{-1}(x)[I - a(x)] \]  
(22)

where:

\[ A = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}, \quad B = [0 \ \ 1], \quad \beta(x) = M(x), \quad a(x) = \mathbf{H}(x, x) \]  
(23)

Considering a nonlinear feedback given by (Fig. 5):

\[ \mathbf{I} = \mathbf{a}(x) + \beta(x) w \]  
(24)

The transfer between \( w \) and \( y \) is equivalent to:

\[ \dot{y} = w \]  
(25)

This is known as the feedback linearized system. It corresponds to the familiar inverse dynamics control scheme which transforms the direct dynamic model into a double set of integrator equations.

Classical control techniques based in a linear model can now be used to design a tracking controller. Computed torque control and predictive functional control that use this linearization will be explained in the next sections.

C. Computed torque control

Assuming that the motion is completely specified with the desired position \((q^d)\), velocity \((\dot{q}^d)\) and acceleration \((\ddot{q}^d)\), the classical computed torque control [10] computes the required arm torque. An integrator gain is added to this classical scheme for obtain a diminishment of tracking error.
due to the differences between the used dynamic model and the real system. The control law becomes:

\[ w = K_p(q^d - q) + K_v(\dot{q}^d - \dot{q}) + K_i \int (q^d - q) dt + \dot{q} \]

(26)

where \( K_p, K_v, K_i \) are the controller gains.

Fig. 5 illustrates the computed torque control scheme. The gains tuning leads to \( K_p=7000, K_v=60, K_i=60000 \) in order to guarantee the torque and dynamic actuator constraints in tracking situation.

D. Predictive functional control

The MPC is implemented with a second order internal model issued from a weighted least square identification technique [12]. The model identified \( (G(s) = \frac{2.7}{s^2 + 2s + 54}) \) is unstable. An inner closed loop in velocity is added, with a gain \( K_v=70 \), for stabilize the system which will be the internal model for the predictive control.

5. EXPERIMENTAL RESULTS

These experiments are running within 1.5 ms sampling period. Complex machining tasks trajectories are used as operational set point: a 20 mm radius circle and a change in the direction of a line (55°); paths done in 3 and 6 seconds, velocities of 2 rad/s and 0.012 m/s respectively. These trajectories show the different performances of the three controllers applied in this parallel mechanism. Other results can be found in [16].

Fig. 7 shows the circular path and Fig. 8 the tracking performance. Fig. 9 shows the line path in the angle’s change zone and Fig. 10 their tracking error performance.
The results show the good performances of the PFC controller in comparison to the CTC and PID control. For both cases, PFC responses are most fast and the tracking error lower. Table II presents this, showing the average tracking errors.

### Table II: AVERAGE TRACKING ERRORS

<table>
<thead>
<tr>
<th>Trajectory</th>
<th>PID (m)</th>
<th>CTC (m)</th>
<th>PFC (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circle (w=2 rads/s)</td>
<td>1.4882e-4</td>
<td>1.6735e-4</td>
<td>9.1518e-5</td>
</tr>
<tr>
<td>Right (v=0.012 m/s)</td>
<td>4.7068e-5</td>
<td>5.4752e-5</td>
<td>3.1372e-5</td>
</tr>
</tbody>
</table>

6. CONCLUSION

This paper exhibits relevant results of the application of model-based control strategies. We compared a predictive scheme with the commonly used computed torque control and PID in terms of tracking performances in complex machining trajectories. The behavior of the PFC strategy is better than the CTC and PID controllers. Further works will concern also the analysis with internal and external disturbances as soon as their implementation in machining tasks.

7. REFERENCES


APPENDIX

\[
k_0 = v^1 \begin{pmatrix} 1 - \alpha h^1 & \vdots & h^m \\ \vdots & \ddots & \vdots \\ 1 - \alpha h^m \end{pmatrix} ,

k_m = v^1 \begin{pmatrix} h^1_m \\ \vdots \\ h^m_m \end{pmatrix} ,

v = - \begin{pmatrix} C^T (F^0 M - I) \end{pmatrix} v

= R^T u_0(0) \text{ where:}

R = \sum_{i=1}^{n} \begin{pmatrix} y_i(h_1) y_i(h_2) \cdots y_i(h_n) \end{pmatrix}

y_i = \begin{pmatrix} y_n \ y_{n-1} \cdots y_{n-i} \end{pmatrix} , u_0 = \begin{pmatrix} u_n \ u_{n-1} \cdots u_{n-i} \end{pmatrix} \]