European Control Conference ECC'99
Karlsruhe, Germany, August/September 1999

ADAPTIVE MINIMIZATION OF THE MAXIMAL PATH DEVIATIONS OF INDUSTRIAL ROBOTS

Friedrich Lange, Gerhard Hirzinger

Deutsches Zentrum für Luft- und Raumfahrt e. V. (DLR)
Oberpfaffenhofen, D-82234 Weßling, Germany
Fax : +49-8153-28-1134 e-mail: Friedrich.Lange@dlr.de

Keywords : robot, feedforward control, path accuracy, adaptive, learning

Abstract

A learning system is presented which uses feedforward control to improve the accuracy of standard position controlled robots. The method is executed on joint level since in this case there are less couplings than in the cartesian space. On the other side the main goal is to reduce the maximal deviation from a given cartesian path. This requires extended algorithms which are derived and examined using a KUKA KR6/1 industrial robot. The universal controller is adapted to minimize the maximal path error and then shows significantly better performance when repeating the training path or a similar trajectory.

1 Introduction

Reduction of dynamical path errors has been extensively discussed. But still the accuracy is not always satisfactory for industrial tasks. The problem is intensified since task execution without sensor corrections requires very high accuracy.

Known approaches use model based feedforward control as the computed torque approach [1, 7]. We do not have a detailed model of the robot and its feedback control loops. So in contrast to model-based approaches we learn the controllers.

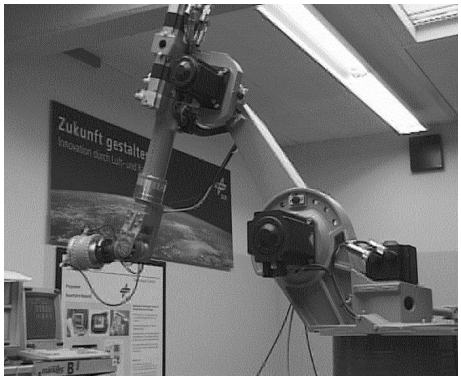

Figure 1: Industrial robot used for experiments

Some authors use neural networks to learn the optimal control inputs (joint torques) for a certain task (see e.g. [8]). In contrast this paper considers improvements of control for standard industrial robots which in most cases have no interface for advanced model-based algorithms. So we leave stabilization and feedback control of the motor positions unchanged as a standard cascaded control system (see Figure 2).

Nevertheless this paper presents a learned part of the controller which almost globally improves the usual robot accuracy. It turns out that for the improvements feedforward control is sufficient for common robots since the positional control loop normally is well tuned so that additional information is required to be able to improve control. This additional knowledge is represented by the desired speed and acceleration or by future desired positions since the actual desired position is used in the cascaded control system, anyhow.

Experiments show that for most robots the joint dynamics are different but fairly decoupled. In contrast, the cartesian description of motion is highly coupled. Therefore learning is executed on joint level. This means that joint commands $\mathbf{q_c}$ are sent every 12 ms and the joint positions $\mathbf{q}$ are available within the same rate.

The paper first explains the learning system for minimization of the mean joint errors. This has been shown comprehensively in [3]. Training can be performed using one or several general training paths which promise a good overall performance of the controller which therefore will be called a universal controller. For repetitive applications it may be refined or replaced using the respective path. The result will be called an adapted controller.

Section 3 then extends the learning method, first, to the reduction of the maximal errors instead of the rms values and, second, to the consideration of the cartesian values instead of the joint values. The latter includes that normally only errors perpendicular to the direction of motion are of interest. Finally all features are verified in experiments using an industrial robot (see Figure 1).

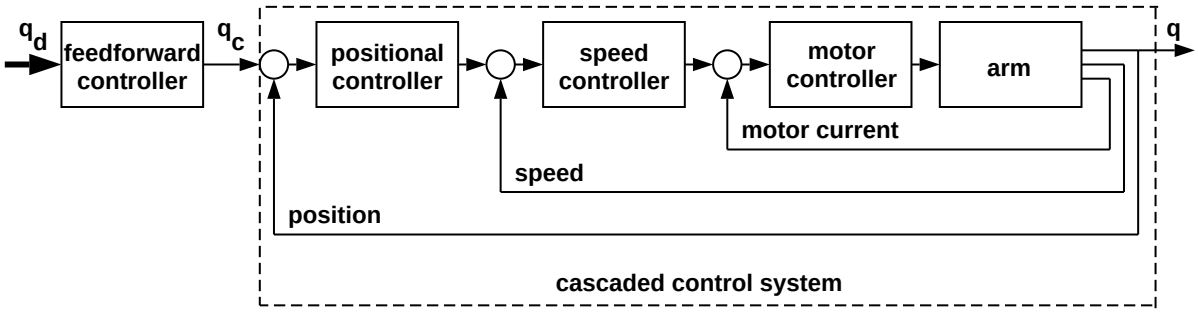

Figure 2: Cascaded positional robot control structure

2 Learning feedforward control to minimize the rms joint errors

Learning of the compensation of joint errors is performed hierarchically in three levels (see Figure 3). First a coarse model of the robot is built (section 2.1). Then this model is used to modify the commands of the training trajectory such that the control error is minimized (section 2.2). These modifications can be learned in a feedforward controller which thereafter is able to improve tracking of other paths as well (section 2.3). Learning of the optimal commands and of the feedforward controllers is repeated iteratively with application of the suboptimal controllers in the meantime. So the resulting controller is better than the model. This is a fundamental difference to other methods as e.g. [2].

The two lower blocks of Figure 3 represent online control of a specified path. The small blocks collect data of the whole trajectory allowing off-line learning in the rest of the figure. So the bold face lines represent the values at different time instants. The robot block includes the cascaded control system since joint torques are not accessible within usual positional interfaces.

2.1 Identification of a simple model

Identification of a coarse model means learning of a mapping between the positional commands $q_c(k,j)$ of joint j at time instant k and the measured joint positions $q(k,j)$. The experiments show that a linear decoupled mapping is sufficient for this task so that different but scalar models for the individual joints are appropriate. The approach for each of these models is

$$q(k+1,j) = q_0(k,j) + \hat{g}_0(k,j) + \sum_{i=1}^{n_g} \hat{g}_i(j) \cdot (q_c(k-i+1,j) - q_0(k,j)). \quad (1)$$

A recursive Kalman filter as a method for linear parameter estimation determines the estimated values $\hat{g}_i(j)$ of the impulse response functions. The theoretical number of elements of this function is infinite, but for asymptotically stable processes it can be limited to e.g. $n_g = 10$ elements that are sufficient when using sampling intervals of 12 ms.

The additional functions $\hat{g}_0(k,j)$ are necessary to represent a bias. To ensure convergence in case of a smooth trajectory it is further necessary to add some stochastical excitation to the commanded values.

2.2 Learning to follow the training trajectory

The models of Equation (1) are used to modify the commands of the training path so that the control errors are minimized. To do so, Equation (1) is used twice, first, to represent the observed motion $\mathbf{q} = \mathbf{f}(\mathbf{q}_c)$, and second, to describe the desired motion $\mathbf{q}_d = \mathbf{f}(\mathbf{q}_c^*)$. The differences of both types of applications of the model are summarized for each joint by

$$\begin{pmatrix} \hat{g}_1(j) & & \\ \vdots & \ddots & \\ \hat{g}_{n_g}(j) & & \ddots \\ & \hat{g}_{n_g}(j) \ \cdots \ \hat{g}_1(j) & \end{pmatrix} \cdot \begin{pmatrix} \Delta q_c(0,j) \\ \vdots \\ \vdots \\ \Delta q_c(N-1,j) \end{pmatrix} = \begin{pmatrix} e(1,j) \\ \vdots \\ \vdots \\ e(N,j) \end{pmatrix} \quad (2)$$

with the control errors $e(k,j) = q_d(k,j) - q(k,j)$ and $\Delta q_c(k,j)$ denoting the required modifications with respect to the previously commanded path $q_c(k,j)$.

Instead of the solution of the system of equations we prefer estimation of the $\Delta q_c(k,j)$ since estimation smoothes if $e(k,j)$ is noisy. This estimation is performed using the inverse Kalman filter (see e.g. [6]), which can be modified to reduce the computing effort to be only proportional to the length N of the trajectory. This yields essential acceleration of the optimization, since $N \gg n_g$.

Convergence is improved if the learning parameters of the inverse Kalman filter, as e.g. the assumed noise on the

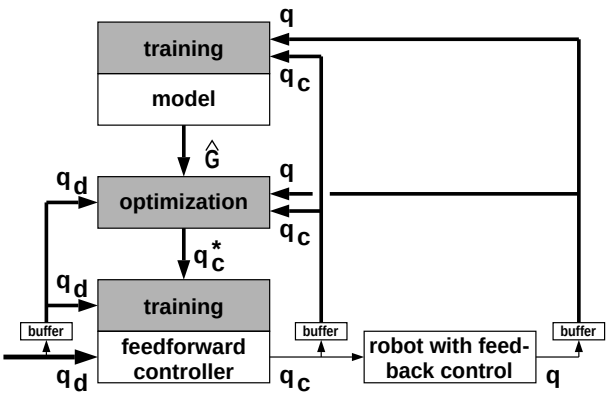

Figure 3: Structure of the learning system

measured values, are determined in advance by comparing real data and the predictions of the models.

The optimal commands for the training path

$$q_c^*(k,j) = \Delta q_c(k,j) + q_c(k,j) \tag{3}$$

are calculated, to be used as target values for the learning of the controller in section 2.3. Before, the commands are smoothed to assure convergence in spite of model inaccuracies.

2.3 Feedforward control for arbitrary paths

In the first approach linear feedforward filters are trained using some subsequently desired positions $q_d(k+i,j)$ as input.

$$q_c(k,j) = q_d(k,j) + \sum_{i=1}^{n_d} r_d(i,j) \cdot (q_d(k+i,j) - q_d(k,j)) \tag{4}$$

For the experiments only $n_d = 15$ predicted steps of the path are necessary. Training is performed by estimating the parameters $r_d(i,j)$ of Equation (4) according to the target signals $q_c^*(k,j)$ using a recursive Kalman filter as for the identification in section 2.1. So the adaptation of learning parameters is possible as well.

There are robots and applications for which this approach is sufficient. For higher requirements the controller structure has to be extended to represent couplings between the joints. These couplings are caused by the non-diagonal terms of the mass matrix and by the centrifugal and coriolis terms. Because of the highly nonlinear mapping the representation is stored in a multilayer perceptron for each joint. The inputs of the nets differ from joint to joint according to the geometric relations. This yields an extension of Equation (4)

$$\begin{aligned} q_c(k,j) = \cdots + net_j(&\mathbf{q}_d(k), \\ &q_d(k+1,j_1) - q_d(k,j_1), q_d(k+2,j_1) - q_d(k,j_1), .. \\ &q_d(k+1,j_2) - q_d(k,j_2), q_d(k+2,j_2) - q_d(k,j_2), ..) \end{aligned} \tag{5}$$

where j_1 and j_2 are indices of joints that are seen to influence joint j for the robot on hand. $\mathbf{q}_d(k)$ is the whole desired joint vector which describes the current geometrical relations.

The solution with linear part and neural net is chosen because Equation (4) has to be computed with high accuracy which is a problem for most neural net architectures. On the other side a multilayer perceptron is adequate for the representation of influences which do not fit to the linear approach. Unfortunately, the generalization of the neural net is restricted to paths which are similar to the training path.

For many robots there are couplings that are known to be linear. This concerns the hand axes whose motors are concentrated near the 3rd joint (see Figure 1). In this case every rotation of joint 4 causes changes of the orientations of joints 5 and 6. All rotations of joint 5 further influence joint 6. For this type of robot kinematics, control of the motor coordinate is learned instead of the joint coordinate. So q_j means the position of motor j, not of the corresponding joint. For the compensation of this kind of couplings no neural net is required. A linear extension of Equation (4) (with almost global relevance) is sufficient:

$$\begin{aligned} q_c(k,j) = &\cdots \\ &+ \sum_{i=1}^{n_c} r_c(i,j,j_1) \cdot (q_d(k+i,j_1) - q_d(k,j_1)) \\ &+ \sum_{i=1}^{n_c} r_c(i,j,j_2) \cdot (q_d(k+i,j_2) - q_d(k,j_2)) \end{aligned} \tag{6}$$

with j_1 and j_2 as above.

Special consideration is required concerning contact forces because they influence the robot trajectory. This yields a further extension of Equation (4)

$$q_c(k,j) = \cdots + \sum_{i=1}^{n_\tau} r_\tau(i,j) \cdot (\tau_d^c(k+i,j) - \tau_d^c(k,j)). \tag{7}$$

Note that $\tau_d^c = \mathbf{J}^t \cdot \mathbf{F}_d$ with the Jacobian $\mathbf{J}$ is only that part of the desired joint torque vector τ_d that corresponds to the desired contact force $\mathbf{F}_d$. The total joint torque is not considered since joint accelerations are covered by Equations (4) to (6) and gravity is compensated by an integrative part in the standard feedback control system of Figure 2.

All parameters r and one neural net are learned for each joint (see Figure 4). Equations (4) to (7) mean that a fictive path $\mathbf{q}_c$ is commanded in order to execute the desired path $\mathbf{q}_d$.

This kind of feedforward control is superior to any algorithm implemented in the robot controller, because information about the future path is provided. On the other side the necessity of knowledge of the future path can be a restriction, especially for sensor guided movements. This disadvantage can be compensated by a predictive sensor

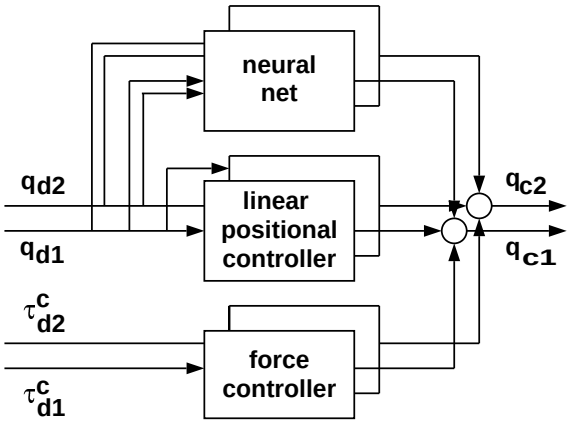

Figure 4: Structure of the feedforward controller of a two-axis robot with linear couplings for joint 2 only

evaluation as in [5] which in combination with the proposed feedforward control allows to make fast and accurate sensor controlled movements.

3 Reduction of the maximal path deviations for specific applications

The performance of the method of section 2 is limited by the fact that the optimal solution of Equation (2) (which would yield $e = 0$) can neither be represented by Equations (4) to (7) nor be executed by real motors. So a compromise has to be found. The estimation of Equation (2) is such a compromise which minimizes the *mean joint* errors. In contrast, for industrial purposes the *maximal path* errors are considered since they yield the accuracy of the product.

Therefore this section reduces the *maximal* deviations instead of the *mean* errors and considers the *cartesian* deviations instead of the *joint* errors. Besides, this method is advantageous for sensor controlled motion, since the calculation of joint errors is ambiguous with low dimensional cartesian sensors (see e.g. [4]).

Unfortunately, the resulting controller is more dependent on the training path. So it may be inferior for trajectories which differ very much from this path.

3.1 Reduction of the maximal joint errors instead of the rms values

The kind of estimation of $\Delta\mathbf{q}_c$ from Equation (2) determines whether the root mean square errors or the maximal errors are minimized. Using the standard inverse Kalman filter yields optimization with respect to the mean square error which is identical to minimization with respect to the root of the mean square (rms) error.

Optimization with respect to the maximal error means that only 1 row of Equation (2) is evaluated using the row with the maximal value of e. Unfortunately this means that many trajectories have to be recorded for learning.

So a compromise is programmed using the full system of equations, but with different weights for the estimation. In the Kalman filter algorithms the weight of $e(k,j)$ is expressed by the expected variance (= squared error) $\sigma^2(k,j)$ of the noise of $e(k,j)$ at time instant k. So nonlinear weighing of $e(k,j)$ can be reached e.g. by $\sigma(k,j) = \sigma_0(j)/e(k,j)$ meaning infinite noise for zero control errors and small noise for big $e(k,j)$.

A more practical approach is

$$\sigma(k,j) = \begin{cases} \sigma_0(j) \text{ for } e_{max}(j) \le e(k,j) \\ \sigma_0(j) \cdot (e_{max}(j)/e(k,j))^\alpha \\ \qquad \text{for } e_{min}(j) < e(k,j) < e_{max}(j) \\ \sigma_0(j) \cdot (e_{max}(j)/e_{min}(j))^\alpha \\ \qquad \text{for } e(k,j) \le e_{min}(j) \end{cases} \quad (8)$$

with $\sigma_0(j)$ as a suitable value to assure smooth convergence in spite of real noise and model inaccuracies and $e_{max}(j)$ as the maximal joint error. $e_{min}(j)$ has to prevent a numerical overflow. This algorithm is used in the experiments with $\alpha = 8$.

3.2 Explicit consideration of the cartesian path error

First of all some definitions: The difference between an actual joint value and the corresponding desired value is called the (positional) joint error $e(k,j)$ of joint j at time instant k. The difference between an actual cartesian position which is calculated from the actual joint positions, and the corresponding desired position is called the cartesian positional error. The latter can be divided into the *tracking* error $e_t(k)$ which is the component in the direction of motion and the *path* error $\mathbf{e}_p(k)$ which comprises the two other directions. Including orientation the path error $\mathbf{e}_p(k)$ has $p = 4$ elements for simultaneous translation and rotation, $p = 5$ elements in the case of either translation or rotation, or $p = 6$ elements when no motion is desired.

Experiments show that for a reduction of the cartesian *path* error by a factor of two the *positional* joint errors have to be reduced by a factor of 20 or so. This comes from the fact that the majority of the joint errors compensate each other with respect to the cartesian path error because they mainly contribute to the tracking error. So it is reasonable to reduce the cartesian path error directly.

This can be done by replacing the scalars in Equation (2) by vectors. $e(k,j)$ is replaced by the vector $\mathbf{e}_p(k)$, $\Delta q_c(k,j)$ is replaced by the vector $\mathbf{\Delta q}_c(k)$ and $\hat{g}_i(j)$ is replaced by the matrix $\hat{\mathbf{G}}_i(k)$ with elements $\hat{G}_i(k,l,j) =$

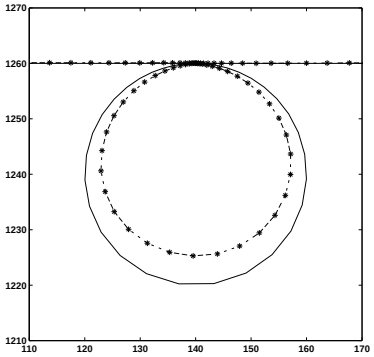

Figure 5: Critical part of path 1 without feedforward control ($*$ = actual path, solid = desired path)

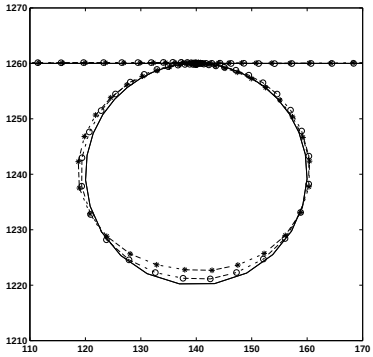

Figure 6: Critical part of path 1 with learned control ($*$ = with universal controller, $\circ$ = with adapted controller for minimized maximal cartesian path errors, solid = desired path, path errors are displayed extended)

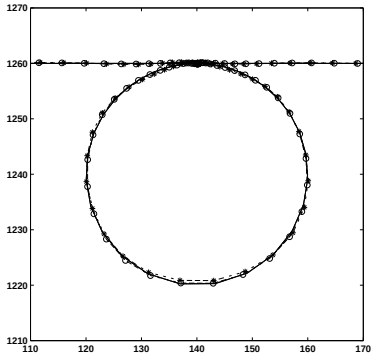

Figure 7: Critical part of path 1 with direct execution of the learned commands ($*$ = with minimized rms joint errors, $\circ$ = with minimized maximal cartesian path errors, solid = desired path, path errors are displayed extended)

$J_p(k,l,j) \cdot \hat{g}_i(j)$.

$$\begin{pmatrix} \hat{\mathbf{G}}_1(1) & & & \\ \vdots & \ddots & & \\ \hat{\mathbf{G}}_{n_g}(n_g) & & \ddots & \\ & \hat{\mathbf{G}}_{n_g}(N) & \cdots & \hat{\mathbf{G}}_1(N) \end{pmatrix} \cdot \begin{pmatrix} \Delta\mathbf{q}_c(0) \\ \vdots \\ \vdots \\ \Delta\mathbf{q}_c(N-1) \end{pmatrix} = \begin{pmatrix} \mathbf{e}(1) \\ \vdots \\ \vdots \\ \mathbf{e}(N) \end{pmatrix} \quad (9)$$

For a robot with n_j joints $\mathbf{J}_p(k)$ is the $p \times n_j$ Jacobian with respect to the time-variant coordinate system with the p components of the tracking error, i.e.

$$de_p(k,l) = \sum_{j=1}^{n_j} J_p(k,l,j) \cdot de(k,j). \quad (10)$$

It has to be used as well for the transformation of $\sigma(k,j)$ or $\sigma_0(j)$ in Equation (8) by

$$\sigma_p(k,l) = \frac{1}{w(l)} \cdot \sum_{j=1}^{n_j} J_p(k,l,j) \cdot \sigma_0(j). \quad (11)$$

$w(l)$ is the weight of the lth component of the path error. $w(l)$ is useful if the orientational DOFs of the path error are less important.

The cartesian approach means that all joint commands are estimated at the same time. So the formulation of n_j times N equations with N variables (Equation (2)) changes to 1 system of $p \cdot N$ equations with $n_j \cdot N$ variables (Equation (9)). This motivates the selection of $p = 2$ for most applications.

4 Experiments

For the experiments a KUKA KR6/1 industrial robot with the standard industrial controller KUKA KRC1 is used. The experiments are executed at the reference path of Figure 8 with full speed according to the path planning algorithm of the KRC1. It turns out that path deviations during the circle are much greater than at the edges. Therefore the circle is zoomed for displaying the learning results. Besides, all performance criteria are summarized in Table 1.

Figure 5 shows the test path (path 1) with the standard controller configuration. In this case the circle is executed too small. Then a feedforward controller according to Equations (4) and (6) is trained using a horizontal and a vertical circle as training path (path 2). It turns out that this (universal) controller is quite good for both paths (see Table 1). Training is then continued, this time using path 1. Training is done in three steps minimizing the mean joint errors, the mean path errors and finally the maximal path errors. Each step is finished when no further improvement can be reached. This procedure is recommended because minimization of the path error does not influence the tracking error and minimization of the maximal errors actually changes only two or three timesteps.

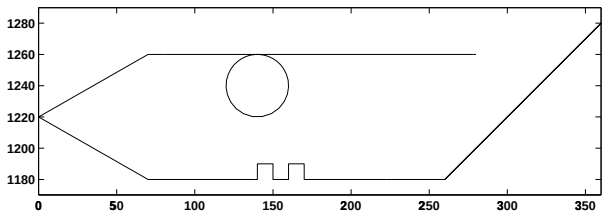

Figure 8: Path 1

	test with path 1			test with path 2	
	rms error	max error	Figure	rms error	max error
standard controller	0.806 mm	5.369 mm	5	1.804 mm	2.451 mm
universal feedforward controller	0.180 mm	1.365 mm	6	0.176 mm	0.527 mm
adapted feedforward controller with					
minimized rms joint error	0.162 mm	0.808 mm		0.252 mm	0.729 mm
minimized max joint error	0.163 mm	0.720 mm		0.312 mm	0.677 mm
minimized rms path error	0.140 mm	0.725 mm		0.293 mm	0.637 mm
minimized max path error	0.140 mm	0.617 mm	6	0.290 mm	0.680 mm
learned commands for					
minimized rms joint error	0.066 mm	0.319 mm	7	-	-
minimized max joint error	0.073 mm	0.241 mm		-	-
minimized rms path error	0.044 mm	0.184 mm		-	-
minimized max path error	0.044 mm	0.120 mm	7	-	-

Table 1: Root mean square and maximal cartesian path deviations for different control strategies. The universal controller has been trained at path 2 (horizontal and vertical circles). All other controllers are trained with path 1.

Finally, the result is superior than with the universal controller (see Figure 6). The performance would be even better if the linear controller could be extended according to Equation (5).

Figure 7 shows control in a different setup. This time no controller is determined but the actions of Equation (3) are commanded directly. This means that training path and test path are identical. Figure 7 demonstrates that the control errors generally are smaller than with feedforward control. This comes from the fact that Equations (4) and (6) are not sufficient to represent the optimal (nonlinear) controller. Unfortunately this type of control is not possible when online reactions to sensor data are required. In that case the adapted feedforward control is adequate because it is insensitive to changes of the desired path (see right hand side of Table 1 for a totally different path).

Figure 7 further displays the difference between a sequence of commands which has been learned to reduce the *mean joint* errors and the commands which minimize the *maximal* cartesian *path* error.

Table 1 shows that explicit reductions of the *maximal* path devitiations automatically mean slightly bigger *mean* path deviations. Velocity inaccuracies or orientational errors may be worse, too, since they are not considered ($p = 2$). But this is acceptable for most applications.

5 Conclusion

It has been shown that performance with respect to the maximal cartesian path deviations during high speed robot motion increases if the learning approach conforms to the performance criterion. For repetitive paths the learned control sequence can be commanded directly, yielding almost zero control errors.

References

[1] C. H. An, C. G. Atkeson, and J. M. Hollerbach. *Model-Based Control of a Robot Manipulator*. The MIT Press, Cambridge, Massachusetts, London, England, 1988.

[2] D. W. Clarke, C. Mohtadi, and P. S. Tuff. Generalized predictive control - part I. the basic algorithm. *Automatica*, 23(2):137–148, 1987.

[3] F. Lange and G. Hirzinger. Learning of a controller for non-recurring fast movements. *Advanced Robotics*, 10(2):229–244, April 1996.

[4] F. Lange and G. Hirzinger. Learning accurate path control of industrial robots with joint elasticity. In *IEEE Int. Conference on Robotics and Automation*, Detroit, Michigan, May 1999.

[5] F. Lange, J. Langwald, and G. Hirzinger. Predictive feedforward control for high speed tracking tasks. In *European Control Conference*, Karlsruhe, Germany, August / September 1999.

[6] P. S. Maybeck. *Stochastic Models, Estimation and Control, volume 2*, volume 141-2 of *Mathematics in Science and Engineering*. Academic Press, 1982.

[7] L. Sciavicco and B. Siciliano. *Modeling and Control of Robot Manipulators*. Electrical and Computer Engineering Series. McGraw-Hill, 1996.

[8] T. Yamada and T. Yabuta. Application of learning type feedforward feedback neural network controller to dynamic systems. In *IEEE Int. Conf. on Intelligent Robots and Systems*, pages 225–231, Yokohama, Japan, Jul. 1993.