LEARNING OF A CONTROLLER FOR NON-RECURRING FAST MOVEMENTS

Friedrich Lange and Gerhard Hirzinger

Deutsche Forschungsanstalt für Luft- und Raumfahrt (DLR), Institut für Robotik und Systemdynamik, Postfach 1116, D-82230 Wessling, Germany
email: Friedrich.Lange@dlr.de, Gerd.Hirzinger@dlr.de
www: http://www.op.dlr.de/FF-DR-RS/

Abstract - In this paper a learning method is described which enables a conventional industrial robot to accurately execute the teach-in path in presence of dynamical effects and high speed. After training the system is capable of generating positional commands that in combination with the standard robot controller lead the robot along the desired trajectory. The mean path deviations are reduced to a factor of 20 for our test configuration. For low speed motion the learned controllers' accuracy is in the range of the resolution of the positional encoders. The learned controller does not depend on specific trajectories. It acts as a general controller that can be used for non-recurring tasks as well as for sensor-based planned paths. For repetitive control tasks accuracy can be even increased. Such improvements are caused by a three level structure estimating a simple process model, optimal a posteriori commands, and a suitable feedforward controller, the latter including neural networks for the representation of nonlinear behaviour. The learning system is demonstrated in experiments with a Manutec R2 industrial robot. After training with only two sample trajectories the learned control system is applied to other totally different paths which are executed with high precision as well.

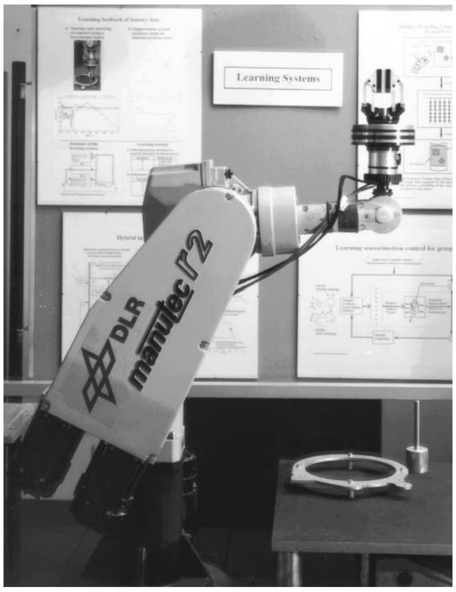
Key words: Feedforward control, path accuracy, positional interface, high speed, nonlinear behaviour, neural networks, generalization

1. INTRODUCTION

Existing industrial robots are able to perform fast and accurate movements. But it takes a lot of effort to teach such paths. In practice, off-line programmed trajectories are modified manually because the trajectory programmed by a static teach-in process deviates from the executed one due to dynamical effects especially in case of high speed movements. After this manual modification procedure the path will be executed with high accuracy since industrial robots are usually stiff.

If the off-line programmed paths are refined by sensor-based path planning as in [2] or [6], the influence of statical inaccuracies due to gravity or mismatch of kinematic parameters usually diappears. This is valid for all sensors which detect the difference between the robot endeffector and the desired position, e. g. robot-mounted sensory devices. Statical path deviations occur only with sensors expressing the target path with respect to a distant reference point.

Other inaccuracies are due to dynamical effects as acceleration, coriolis, or centrifugal forces. Therefore, in this work a learning method is presented which minimizes dynamical path errors. In Figure 2 such path deviations are demonstrated for the robot of Figure 1. The task is programmed as a horizontal and a vertical circle of the tool center



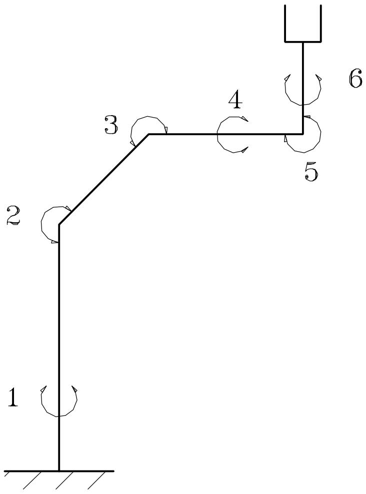


Figure 1. Robot in starting position (x = 500, y = 0, z = 1300) for path of Figure 2

point, to be executed with 500 mm/s while maintaining the orientation. This yields dynamical path errors up to 12 mm.

Improving the path accuracy with learning methods has been worked on before (see e. g. [1], [3], [4], [5], [7], [12], [14], [15], [16]) but was, so far, restricted to repetitive control of the trajectory used for training (trajectory learning) or, to low dimensional problems. In this paper learning is extended to the *behaviour* of a six axis robot, in order to improve different off-line programmed paths after learning a single path thereby acquiring the robot dynamics.

This paper is structured according to the training procedure in which the following steps have to be executed:

1. Execution of a low speed training trajectory (for which couplings are of little importance)
2. Estimation of coarse (linear) process models for all joints (sect. 2.1)
3. Execution of a high speed training trajectory demonstrating linear and nonlinear behaviour
4. A posteriori estimation of the optimal commands for this path (sect. 2.2)
5. Estimation of independent linear feedforward controllers executing these commands (sect. 3.1)
6. Execution of a high speed training trajectory using these feedforward controllers
7. A posteriori estimation of the optimal commands for this path
8. Training of neural nets for compensation of nonlinear couplings which are not considered in the linear approach of step 5 (sect. 3.2)
9. Execution of a high speed training trajectory using the feedforward controllers and the neural nets

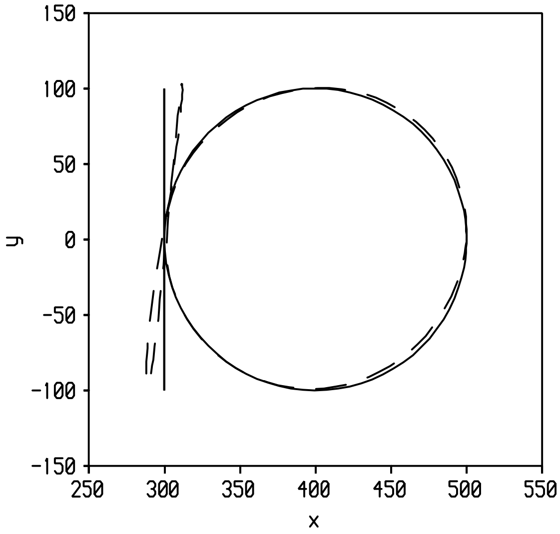


Figure 2. Projection of desired (solid) and actual (dashed) path to the xy-plane when the desired path is commanded without modifications by learning

Learning will be improved if steps 4 to 6 are repeated iteratively before the training procedure is continued. Similarly iterative learning is proposed for steps 7 to 9. In both cases learning is continued until no more improvements are achieved. The learned behaviour turns out to be effective in the whole workspace, at least for the linear part. Please note that this feature goes beyond trajectory learning which is expressed by repeated execution until step 4 (chapt. 2.).

The hardware setup is explained in more detail in [9]. The robot is moved by commanded joint angles of the learned controller stabilized by the internal standard robot cascaded controller. As output the measured values of the motor encoders of the robot are used. In [10] it is proved that the encoder values are sufficient to represent dynamical pose deviations of the tool center point, the measurement errors being about 0.1 mm. The learning controller can be computed outside on another computer or be incorporated in standard industrial robot controllers.

2. LEARNING OF GIVEN ROBOT TRAJECTORIES

There are two possibilities for trajectory learning. Most authors define operators to modify control actions from one execution of the task to the next iteration. These operators have to assure convergence of the learning system. The other approach is to use a known process model being the optimal operator. The method presented here builds such

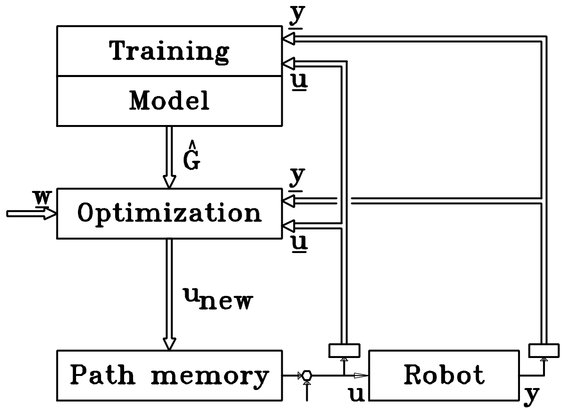


Figure 3. Structure of control (lower part) and trajectory learning of one joint. The small blank blocks collect data of the whole trajectory for off-line learning. So, the vectors there represent the values at different time instants. The arrow from the bottom hints at stochastical modifications of path commands proposed for identification.

a model and then uses it to iteratively estimate positional control actions optimal for the trained trajectory (see Figure 3).

Compared to learning of motor torques this approach might be preferred, because the transfer function is asymptotically stable and insensitive to noise or modelling errors. Because of the iterative procedure a time-discrete, decoupled, and even linear view is allowed for trajectory control. On the other hand the required positional control system is standard and mostly well tuned for industrial robots.

2.1 Identification of a Simple Model

The model is represented by a linear decoupled impulse response function with elements $\hat{g}_i$ for each joint, predicting the actual joint position $y(k+1)$ at time instant $k+1$, caused by positional joint commands u of the preceding time steps.

$$y(k+1) = u(k) + \hat{g}_0(k) + \sum_{i=1}^{n_g} \hat{g}_i \cdot (u(k-i-n_t+1) - u(k)) \tag{1}$$

This equation takes into account a delay-time of n_t sampling instants because of signal processing. It further assumes that the system is stable, so that the impulse response function can be approximated by n_g elements. The resulting error is tolerable since only *changes* in position have consequences. An element $\hat{g}_0$ is provided to prevent inferior learning because of neglecting nonlinear couplings or quantization errors of the robot interface.

For estimation of the elements $\hat{g}_i$ a recursive Kalman filter is used which, in extension to usual least-square algorithms, can take into account the variance of noise (see [13]).

The trajectory used for identification is restricted to moderate speed to reduce the influence of coriolis or centrifugal forces. In addition a stochastical excitation (arrow from the bottom in Figure 3) is added to the path commands improving the high frequency response of the model.

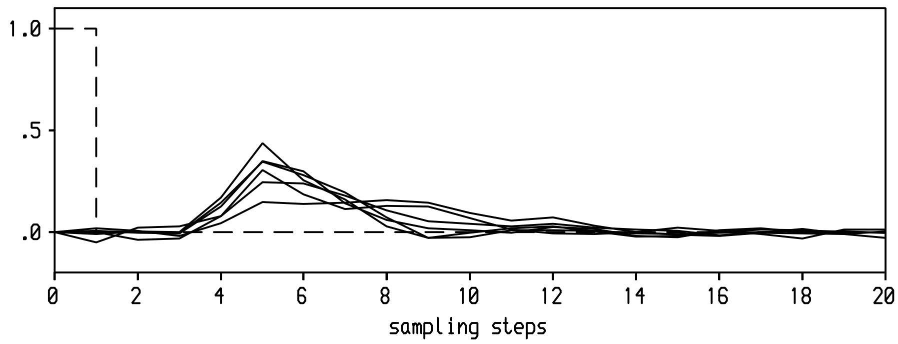


Figure 4. Impulse (dashed) and impulse responses (solid) for different joints

Experiments with a Manutec R2 industrial robot yield the impulse response functions of Figure 4 with sampling intervals of 16 ms. It can be seen, that a delay time of $n_t = 3$ steps and a length of the function of $n_g = 7$ steps characterize the robot sufficiently thus neglecting the signals of the 11th and further steps.

The mean representational error for prediction from $g_0(k)$ to $y(k + n_g + 1)$ is about 0.5 % of the related mean movement. The model is therefore well suited for generation of optimal commands in the next section, although the trajectory used for training had only 200 sampling steps.

2.2 A Posteriori Generation of Optimized Trajectory Commands

Trajectory learning is proposed *after* execution of a path (a posteriori), since the required control error

$$e(k) = w(k) - y(k) \tag{2}$$

from the desired path $w(k)$ is not available, when the control actions $u(k-i)$ with $i > 0$, causing $y(k)$, are commanded. On the other hand there is no generalization intended from the modification of $u(k-i)$ to future commands.

Learning assumes knowledge about which control action has to be modified in case of a control error at time instant k. This knowledge is provided by the model distributing the modifications in relation to the elements of the estimated impulse response function. For multiple control errors at all N time steps this yields a system of equations

$$
\begin{bmatrix}
\hat{g}_1 & & & & \\
\cdots & \cdots & & & \\
\hat{g}_{n_g} & & \cdots & & \\
 & \cdots & & \cdots & \\
 & & \hat{g}_{n_g} & \cdots & \hat{g}_1
\end{bmatrix}
\cdot
\begin{bmatrix}
\Delta u(0) \\ \cdots \\ \cdots \\ \cdots \\ \Delta u(N-n_t-1)
\end{bmatrix}
=
\begin{bmatrix}
e(n_t+1) \\ \cdots \\ \cdots \\ \cdots \\ e(N)
\end{bmatrix}
\tag{3}
$$

with the modifications[1])

$$\Delta u(k) = u_{new}(k) - u(k). \tag{4}$$

Instead of solving equation (3) estimation of the modifications is proposed using an inverse Kalman filter (see e. g. [13]). This allows consideration of noise in the measurements and of model errors, both being roughly calculated by comparing real measurements and predictions by the model.

Another advantage is, that a version of the inverse Kalman filter could be derived with a computational effort being only proportional to the length N of the trajectory. This yields essential acceleration of the optimization since $N >> n_g$.

Learning is performed iteratively, i. e. after modification the new path commands are executed yielding less control errors than before. The experiments with the robot of Figure 1 are demonstrated in Figure 5. The improvements are not limited by the model accuracy, instead the model has to detect only the required direction of modifications. So the resulting precision of the trajectory is limited only by the resolution of the encoders for low speed. For high speed, model errors become noticeable.

For typical paths the initial error is approximately the joint angle covered in 80 ms. For high speed this corresponds to 50 mrad or so. After 20 iterations it is reduced to about 0.1 mrad. Expressed as cartesian path error meaning the deviation perpendicular to the direction of movement, the mean error is about 0.1 mm or even 0.05 mm as in the 3rd column of Table 1 for the speed of 375 mm/s. This corresponds to an improvement by a factor of 40.

Comparing Figure 6 and Figure 2 illustrates that learning results in modifications of commands in the opposite direction of the errors observed without learning. So before learning commands and desired positions coincide whereas after training actual and desired position are almost identical.

To guarantee convergence of the iterative procedure the condition

$$\left| 1 - \hat{G}^{-1}(\omega) \cdot G(\omega) \right| < 1 \tag{5}$$

has to be met where $G(\omega)$ is the real transfer function of one joint of the controlled robot and $\hat{G}^{-1}(\omega)$ is the operator representing the estimation of $\Delta u(\omega)$ from $e(\omega)$ according to equation (3). Violation of equation (5) only happens for frequencies near the sampling rate which are not as well represented by the model. It turns out however, that in most cases weakening of high frequency actions is not optimal either. That is because smoothing affects desired modifications as well and therefore should be used carefully.

1) Notice that the variable k stands for the time step and not for the iteration. To distinguish executed commands and the modified ones, which are executed in the next iteration, the latter are marked with the index $_{new}$.

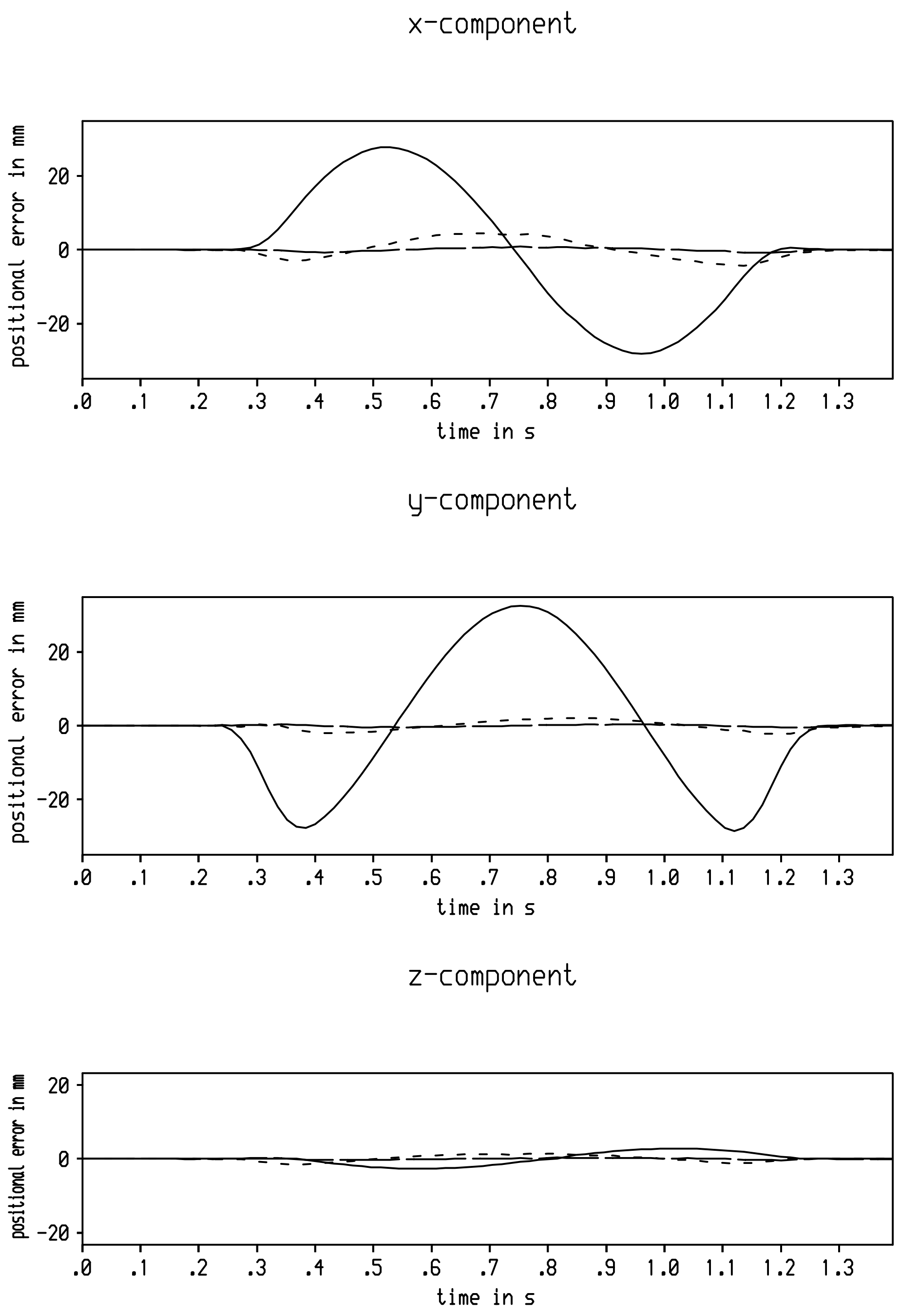


Figure 5. Pose error of a horizontal circular path before learning (solid), after the first iteration (dotted), and after the second iteration (dashed)

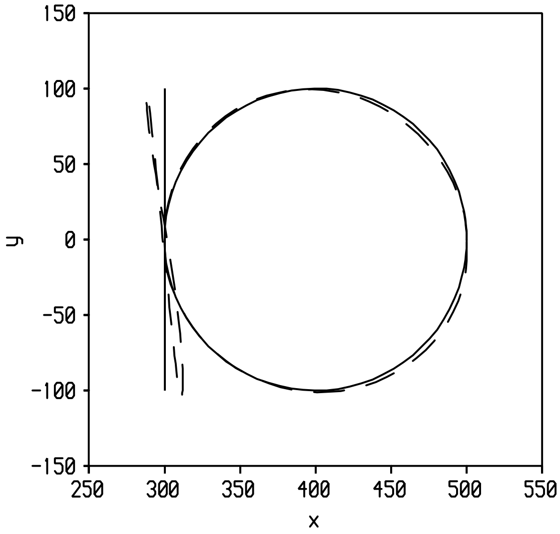


Figure 6. Projection of desired (solid) and commanded (dashed) path to the xy-plane when actual and desired path coincide due to learning of commands (compare to Figure 2)

3. LEARNING OF GENERAL ROBOT BEHAVIOUR

So far, a variation of the well known trajectory learning is demonstrated. The advantage of the approach presented is the possibility to extend it to the learning of the robot behaviour instead of a single trajectory. So, also for trajectories which are still unknown during the learning phase, the commands can be on-line modified during execution thus reducing the path error substantially.

For a behavioural reaction, modifications of commands have to be represented independently of the current trajectory. It turns out to be successful if the modifications are mapped in relation to the future desired path. This means that feedforward control is sufficient, not demanding feedback control. The changed structure of the whole learning system is shown in Figure 7.

It is similar to Figure 3. Just the lower left part is changed, i. e. the memory of modified path commands is replaced by a feedforward controller which outputs a summation of the original path and a learned feedforward signal due to the dynamics.

Figure 7 can be summarized as a hierarchical structure with three levels for learning, but only one for control. The levels of learning, except for the intermediate one, consist of a training module and a memory, each. In contrast the second learning level just processes information and passes it to the lowermost module for updating after each learning step.

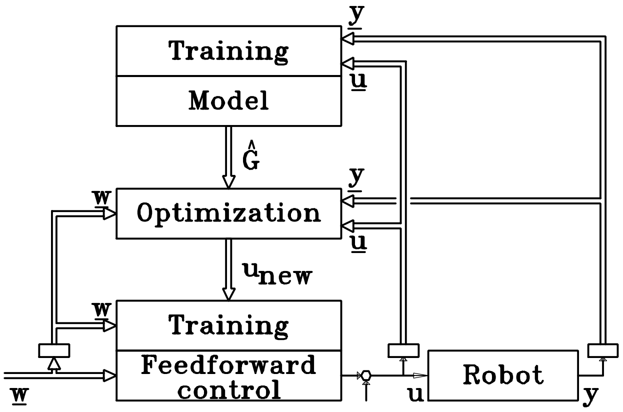


Figure 7. Structure of feedforward control (lower part) and learning of robot behaviour. For notation see Figure 3

In contrast to Figure 3 the system is no longer regarded to be decoupled. So simple lines do not represent single joint values but the whole set. By that means the controller of joint j can compensate effects that are caused by other joints, too.

Different approaches are possible for the controller. The simplest case are independent linear feedforward controllers for the individual joints. This is implemented in sect. 3.1. A better solution may be independent nonlinear controllers. This is not discussed here because the remaining errors after linear compensation seem to stem from nonlinear couplings between the individual joints. So nonlinear controllers which take into account several joints are introduced in sect. 3.2.

3.1 Linear Feedforward Control

The simplest approach has a controller structure similar to the model in equation (1). So the commanded positions of one joint are

$$u(k) = w(k) + \sum_{i=1}^{n_w} r_i \cdot (w(k + n_t + i) - w(k)) \tag{6}$$

with n_w controller parameters r_i weighing future desired positions with respect to the actual value.

This approach can be seen as an anticausal inversion of the model. So it may be superior to any causal controller which only considers actual or past information about the desired or the actual path. Knowledge of future path data is absolutely necessary for compensation of dynamical delays.

Learning of this controller is quite simple because of the parameter-adaptive setup. The task is to estimate the parameters r_i of equation (6) with $u(k)$ being the modified command $u_{new}(k)$. A recursive Kalman filter estimates the controller parameters, similar to the identification of the model.

This estimation can represent u_{new} only as good as it can be fitted in the structure of equation (6). So the error of estimation has to be examined to adapt the structure. It turns out to be sufficient, if the number of considered subsequent sampling points of the desired path n_w is identical to the regarded length of the impulse response function n_g. Information of past time instants as the history of the desired or the actual path is not advantageous for our robot as the internal control reduces all deviations in an efficient way. Only for large contact forces special considerations prove to be useful (see [10]).

Unlike the identification of the model there is no excitation required for the estimation of the controller parameters to improve convergence. That is because the total controller output has to be represented after training, whereas the internal parameters are of no importance. It is sufficient to learn from one or several trajectories covering the scope of possible movements.

Experiments with behavioural learning using always the same training trajectory show that pose errors for this path are reduced as well as during trajectory learning in Figure 5. About 10 iterations are sufficient.

For the spatial path of Figure 2 appropriate commands are executed as well after training with this trajectory. In this case however, compensation of dynamical effects is only possible for a setup in joint space, not in cartesain space. The optimal modifications to the desired path according to Figure 6 are maximal in the x-direction when moving in the y-z-plane. So in the cartesian space such path errors cannot be reduced by independent feedforward controllers. In joint space however, the compensation can be represented by equation (6) without consideration of couplings. This shows that learning in cartesian space is inferior to the proposed approach.

Table 1 compares the original robot behaviour (2nd column) with the learned trajectory commands of chapter 2. (3rd column) and the compensated robot behaviour of this section (4th column) using different criteria of performance. All training begins with the unlearned state. Improvements are substantial, trajectory learning exceeding the performance of behavioural learning by a factor of two for the training path. This is not

error criterion	without learning	mod. of commands	feedforward control
pose error of joint 1	38.736 mrad	0.109 mrad	0.250 mrad
pose error of joint 2	50.270 mrad	0.054 mrad	0.215 mrad
pose error of joint 3	61.572 mrad	0.075 mrad	0.242 mrad
pose error of joint 4	0.604 mrad	0.103 mrad	0.767 mrad
pose error of joint 5	18.171 mrad	0.093 mrad	0.364 mrad
pose error of joint 6	43.050 mrad	0.252 mrad	0.669 mrad
cartesian pose error	23.091 mm	0.069 mm	0.216 mm
cartesian path error	2.107 mm	0.046 mm	0.114 mm
cartesian trailing error	22.994 mm	0.051 mm	0.183 mm

Table 1. Comparison of different error criteria during execution of a horizontal circle with 375 mm/s after training at the same path (Trailing error means the projection of the pose error to the direction of movement, path error covers the directions perpendicular to this.)

path	speed	cartesian pose error without	with feedforward control
hor. and vert. circles, gripper upside	500 mm/s 250 mm/s	39.900 mm 21.523 mm	0.187 mm 0.130 mm
hor. circle, gripper downside	375 mm/s 188 mm/s	23.091 mm 13.546 mm	0.216 mm 0.134 mm
vert. circle, gripper downside	375 mm/s 188 mm/s	24.053 mm 14.126 mm	0.182 mm 0.121 mm
hor. circles, gripper upside, from x=100 mm to x=650 mm	375 mm/s 188 mm/s	32.074 mm 16.709 mm	0.160 mm 0.123 mm
hor. rectangles, gripper upside, from x=125 mm to x=650 mm	375 mm/s 188 mm/s	27.221 mm 16.218 mm	0.185 mm 0.155 mm

Table 2. Pose errors at different paths after behavioural learning using alternating the two uppermost paths.

sursprising since so far the compensation is reduced to linear, position independent, and decoupled behaviour (equation (6)).

On the other hand after training of one trajectory, other paths can be compensated, too, even with a different speed as shown in Table 2.

It turns out that after initial learning, which normally would be executed after installation or servicing of the robot, all pose errors are reduced substantially, regardless of the shape of the path (circles or rectangles), the orientation of the gripper (upside as in Figure 1 or opposite), or the speed.[2])

Regarding the range of paths it can be said that the acceleration limits of the robot are reached, so that faster movements are not possible. The two lowermost paths in Table 2 cover maximal changes of the inertia of joint 1, from small values of x according to maximal joint speed to an almost singular configuration with big values of x.

Orientation of the gripper was held constant for all experiments. So joint 4 was not moved, thus disabling any improvement by feedforward control.

The final pose accuracy of about 0.2 mm (path accuracy is about half as much) is limited by the trajectories used for learning and by the structure of the controller. So the 4th column of Table 1 as well as Table 2 show the result of a training that was performed using the two uppermost paths of Table 2. Training with the first path only leads to about 20% higher errors resulting from too specialized learning.

Concerning the controller structure nonlinear independent controllers for the individual joints promise only little improvement since paths with large changes in position and inertia are performed quite well. On the other hand couplings are noticeable at least for the 4th joint. The joint is affected by accelerations of the first joint (see Figure 1) causing deteriorations which cannot be compensated by independent behavioural learning. During acceleration and deceleration this creates ramps of the pose error deformed by the integrative feature of the internal controller. So deviations of joint 4 due to acceleration of

[2]) Speed is adapted to the permissible accelerations of the robot yielding lower speed close to edges or the end of the path.

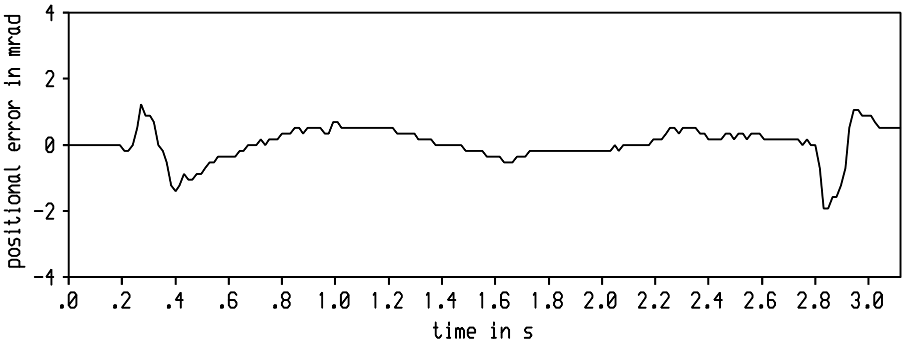


Figure 8. Pose error of joint 4 during independent feedforward control of all joints

joint 1 are reduced after two sampling steps yielding an overshoot when the initial acceleration phase of joint 1 is terminated (see Figure 8).

3.2 Neural Networks for Compensation of Nonlinear Effects

In this section the method to reduce such couplings is described. For a better understanding the analytic description of the dynamic structure is set up first.

$$m_{ii}(\underline{\varphi}) \cdot \ddot{\varphi}_i = \tau_i + v_i(\underline{\varphi}, \underline{\dot{\varphi}}) + \gamma_i(\underline{\varphi}) - \sum_{i \neq j} m_{ij}(\underline{\varphi}) \cdot \ddot{\varphi}_j \tag{7}$$

Here φ_i is a joint angle, influenced by the corresponding joint torque τ_i, by coriolis and centrifugal forces v_i, by gravity γ_i, and by accelerations of the other joint angles φ_j. These influences are highly nonlinear, on one side because of the elements of the mass matrix m_{ii} depending on the overall configuration, on the other side because of nonlinearities in the coupling elements themselves. In our case accelerations of the 1st joint affect the 4th joint in relation to the orientation, thus meaning the opposite influence for a hanging gripper than for a standing configuration (see Figure 1).

It is difficult to learn the compensation of all possible couplings as easy as in the previous section as long as a detailed analytical description of the robot and the internal controller is unknown. Therefore the dominant couplings have to be selected and trained for a small number of trajectories only.

Table 1 shows that main errors after independent feedforward control remain for joints 4 to 6, since the other motors are more powerful and thus insensitive to couplings. On the other hand joint 6 is not influenced by other joints because of the symmetric construction. So decoupling can be reduced to joints 4 and 5.

It can be assumed further that deviations of these joints caused by couplings stem essentially from accelerations and less from coriolis, centrifugal, or gravity forces, because slowly changing influences have no effect due to the integral part of the internal controller.

The robot configuration, expressed by joint angles 2 to 5, and the corresponding accelerations are sufficient to describe the couplings. The structure of the robot (see Figure 1) shows that for joint 4 only the acceleration of joint 1 is significant, whereas joint 5 is affected by accelerations of joint 2 and 3.

For feedforward control as in section 3.1 the values at different time instants are required. Because of slow changes in the configuration this is only valid for the accelerations which can be computed by temporal differences.

This yields

$w_2(k)$, $w_3(k)$, $w_4(k)$, $w_5(k)$,
$(w_1(k+1) - w_1(k))$,$(w_1(k+2) - w_1(k))$, ... ,$(w_1(k+10) - w_1(k))$

and

$w_2(k)$, $w_3(k)$, $w_4(k)$, $w_5(k)$,
$(w_2(k+1) - w_2(k))$,$(w_2(k+2) - w_2(k))$, ... ,$(w_2(k+10) - w_2(k))$,
$(w_3(k+1) - w_3(k))$,$(w_3(k+2) - w_3(k))$, ... ,$(w_3(k+10) - w_3(k))$

as inputs for the nonlinear feedforward controllers of joints 4 and 5, respectively.

As implementation for learning and representation of the nonlinear behaviour performance-adaptive systems are more general than parameter-adaptive methods. Multilayer perceptrons with sigmoid activation functions turn out to be adequate. They are designed here with two hidden layers with 7 (8) and 3 (3) neurons respectively, and one output neuron (values for joint 5 in parentheses). All neurons possess bias terms resulting in 133 and 231 trainable weights for the nonlinear controllers of joint 4 and joint 5, respectively.

These nets are arranged parallel to the controllers of section 3.1 meaning a structure similar to Figure 9. This allows the net output to be mapped to a small range, thereby increasing the resolution. Moreover, since the output range represents an upper bound for the errors that can be produced by the net, the small range will guarantee minimal deteriorations in case of misrepresentation of training data.

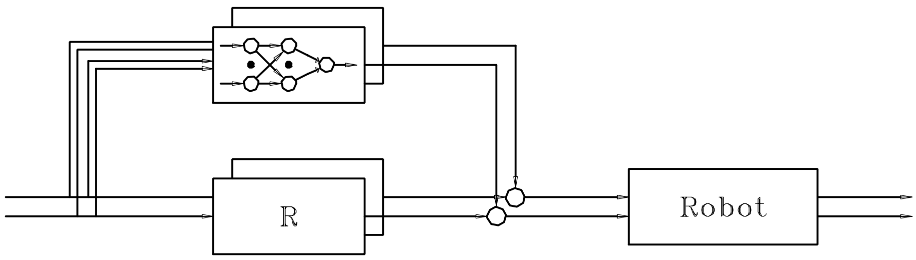


Figure 9. Structure of the feedforward control for a robot with only two joints (R means the linear coefficients r_i of equation (6))

Learning is done after convergence of the independent feedforward controllers, i. e. when the cause for the couplings remains unchanged. It is performed in the same way as for the linear feedforward controller, i. e. a desired command is estimated in the intermediate level of Figure 7 and serves as target value for the training of the controller. For the training of the net the output of the linear part is subtracted from this target.

As training algorithm standard back-propagation is tested but not used. Instead algorithms derived from optimization theory proved to be appropriate, as are a truncated Newton method or an extended Kalman filter [11].

Learning of nonlinear compensation for the path of Figure 2 is reported in Table 3 showing a reduction of the mean pose error up to a factor of 3. The ramps of Figure 8 vanish in Figure 10.

	joint 4	joint 5
mean pose error without learning	0.212 mrad	70.722 mrad
mean pose error without neural net	0.507 mrad	0.427 mrad
mean pose error with neural net	0.095 mrad	0.251 mrad
trajectories used for net learning	9	10
cpu-time used for net training	32 s	150 s

Table 3. Improvement by compensation of couplings using a neural net (cpu-time is stated for a R4000 processor)

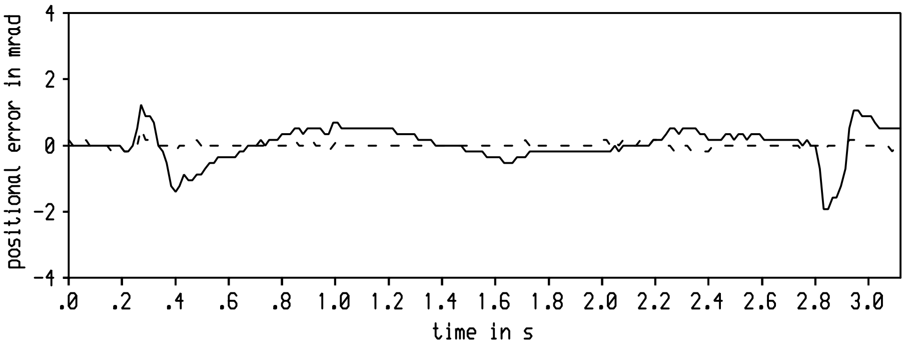


Figure 10. Influence of the neural net to the pose error of joint 4 (dotted: with neural net, solid: without neural net)

Although this figure is impressive it has no practical use. It makes no sense to train neural nets for a special trajectory as the method of chapter 2. will be superior in any case. Feedforward control of nonlinear behaviour will only be clever for untrained paths. Unfortunately the generalization is not as wide as in the linear case. Neural nets are only favourable in vicinity of the trajectory used for training. Therefore they can be recommended only for inaccurately specified paths which are modified on-line according to sensor data.

To demonstrate this, the training for Table 3 is repeated, this time without using the nominal path but with trajectories shifted relating to the cartesian directions by 50 mm (see Figure 11 for the paths shifted in the x-y-plane.) Altogether training data are 6 times as much as for Table 3.

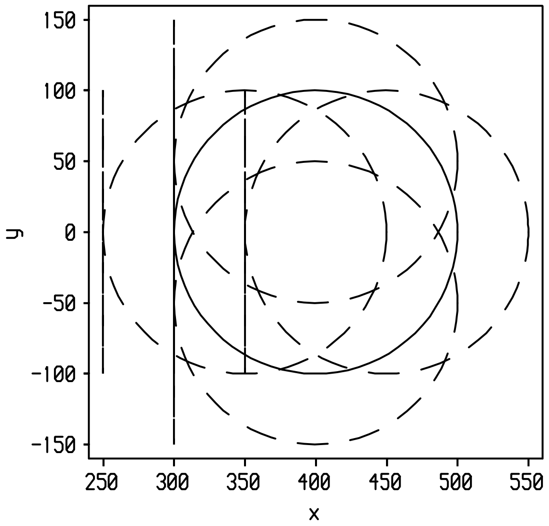


Figure 11. Projections of the nominal path (solid) and shifted duplications in the x-y-plane (dashed)

The learned nonlinear feedforward control for the untrained nominal trajectory proves to be similar to the previous experiment. Pose errors of 0.108 mrad and 0.272 mrad are achieved for the 4th and the 5th joint, respectively.

4. CONCLUSION

The paper presents an off-line learning method for positional control of a robot reducing the path error of arbitrary but realizable trajectories to about 0.1 mm. Training needs to be done only once after installation or servicing of the robot. An extension to on-line learning is possible (see [8]).

The training sequence and procedure are summarized in chapt. 1.. After the training path accuracy is substantially improved for trained and non-trained paths as the dynamical behaviour of the robot can be transferred. Only the compensation of couplings between the joints is limited to trajectories which are similar to the training path. In the moment a prestructured net approach is under developement which is able to represent the whole nonlinear robot behaviour after training of some selected training paths.

For repetitive paths as found in industrial applications, learning can be continued by direct modification of commands according to chapt. 2. (trajectory learning). The first run will be executed using feedforward control. Therefore the huge errors of Figure 5 (solid and dashed line) during the first learning iterations do not appear. After training manual modifications by an operator are not necessary anymore.

Then it is either possible to follow off-line programmed trajectories which are statically modified because of sensed deviations. Or it is possible to track on-line sensed paths. In that case a predictive sensor is recommended allowing feedforward control as is the case for the off-line defined paths.

The experiments were executed with a Manutec R2 industrial robot and ist standard controller. Measurements were taken using the internal motor encoders. Similar learning results can be expected for other robots and other robot controllers, as the techniques developed here are fairly general.

5. REFERENCES

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

[2] K. Arbter, J. Heindl, G. Hirzinger, K. Landzettel, F. Lange: New Techniques for Teach-In, Acceleration and Learning in Sensor-Controlled Robots, 9th IFAC World Congress, Budapest, Hungary, July 1984

[3] S. Arimoto, T. Naniwa, and H. Suzuki: Selective Learning with a Forgetting Factor for Robotic, Motion Control 1991 IEEE Int. Conference on Robotics and Automation, Sacramento, California

[4] C. G. Atkeson, E. W. Aboaf, J. McIntyre, and D. J. Reinkensmeyer: Model-Based Robot Learning, 4th Int. Symp. of Robotic Research, Santa Cruz, Aug. 1987

[5] J. J. Craig: Adaptive Control of Mechanical Manipulators, Addison-Wesley Publishing Company, Reading, Massachusetts, 1988

[6] G. Hirzinger: ROTEX - The First Robot in Space, 6th Int. Conf. on Advanced Robotics, Tokyo, Japan, Nov. 1993

[7] T. Kavli: Frequency Domain Synthesis of Trajectory Learning Controllers for Robot Manipulators, Journal of Robotic Systems, Vol. 9, No. 5, pp. 663-680, 1992

[8] F. Lange and G. Hirzinger: Iterative Self-Improvement of Force Feedback Control in Contour Tracking, 1992 IEEE Int. Conference on Robotics and Automation, Nice, France

[9] F. Lange und G. Hirzinger: Erhöhung der Bahngenauigkeit von positionsgeregelten Robotern, VDI-Bericht Nr. 1094, 1993 (in German)

[10] F. Lange and G. Hirzinger: Learning to Improve the Path Accuracy of Position Controlled Robots, IEEE/RSJ/GI Int. Conference on Intelligent Robots and Systems, München, Germany, Sept. 1994

[11] F. Lange: Fast and Accurate Training of Multilayer Perceptrons Using an Extended Kalman Filter, Internal paper, available by http://www.op.dlr.de/FF-DR-RS/STAFF/friedrich_lange/, Sept. 1995

[12] E. Lunde and J. G. Balchen: Practical Trajectory Learning Algorithms for Robot Manipulators, 1990 IEEE Int. Conf. on Robotics and Automation, Cincinnati, May 1990

[13] P. S. Maybeck: Stochastic Models, Estimation and Control, Vol. 2, Mathematics in Science and Engineering, Vol 141-2, Academic Press, 1982

[14] W. T. Miller III, F. H. Glanz, L. G. Kraft III: Application of a General Learning Algorithm to the Control of Robotic Manipulators, The International Journal of Robotics Research, Vol. 6, No. 2, pp. 84-98, 1987

[15] H. Miyamoto, M. Kawato, T. Setoyama, R. Suzuki: Feedback-Error-Learning Neural Network for Trajectory Control of a Robotic Manipulator, Neural Networks Vol. 1, pp. 251-265, 1988

[16] W. Neubauer, M. Möller, S. Bocionek, and W. Rencken: Learning Systems Behavior for the Automatic Correction and Optimization of Off-line Robot Programs, 1992 IEEE Int. Conference on Intelligent Robots and Systems, Raleigh, NC