7th German Conf. on Robotics ROBOTIK 2012, Munich, Germany, May 2012

Control with a Compliant Force-Torque Sensor

Friedrich Lange, Michael Suppa, and Gerd Hirzinger
Robotics and Mechatronics Center, German Aerospace Center (DLR), Weßling, Germany

Abstract

There are assembly tasks which require a compliant device at the end-effector since possible disturbances are beyond the bandwidth of robot control. This paper discusses a compliant force-torque sensor for assembly. Two aspects are explained in detail: Force control considering a significant force dependent displacement, and control of an end-effector with an elastic mounting during fast unconstrained motion. The latter uses an adaptive scheme which serves as a further level in a hierarchical position-based control. Experimental results are given which show the limits of industrial robots.

1 Introduction

For fast handling and assembly of heavy objects it is advantageous to have a built-in compliance at the robot end-effector. Otherwise big forces or torques can be generated, that possibly cause a damage. Such problems may arise

- if an object that is not expected is close to the robot path and is therefore hit unintentionally,
- with an inaccurate model of the assembly scenario, such that contact is found before the expected contact point is reached or at a slightly different pose,
- with a stochastically moving heavy object to which a smaller part has to be assembled by a stiff robot.

The first two problems can be solved by a more accurate modeling, e.g. by using a vision system that online surveys all objects in the vicinity. Assembly to moving parts, however, requires a compliance between the robot and its end-effector. Only in this way it can be ensured that contact forces and torques are small, since usually both, industrial robots and fixtures for assembly are quite stiff.
So a typical application for a compliance is moving belt assembly [1], e.g. the mounting of wheels to a car body that is transported by a conveyor as in [2]. A full-scale setup has been prototyped at the Institute for Machine Tools and Industrial Management (*iwb*) in Augsburg (Fig. 1). In this arrangement, the compliance is integrated within a scalable force sensing device [3, 4] which connects the end-effector with the robot flange (Fig. 3).
Two problems have to be solved when using compliance or a compliant sensor in control:

- Force control has to be revised to account for significant displacements. This has been presented in [5]. The key ideas will be summarized in section 3.
- Compliance does not only comply to forces but also to accelerations. This effect has to be compensated during free motion. That is the main issue of the present paper.

Concerning force control, though compliant sensors usually are not as accurate as stiff ones, they are better suited for industrial robots, as the latter are position controlled. This means that, in the outer loop, it is not possible to increase the joint torque without any motion. Instead, a minimum pose difference is commanded, that may cause overshooting of the force. Theoretically, force control via a position interface is impossible, if there is no compliance present at all. So force control via a position interface uses the fact that there is always a compliance. Without a designed (significant) compliance it is hard to tune a controller since the resulting (small) compliance may vary depending on the position and the environment. Thus a setup with a compliant force-torque sensor is the simpler case, if the displacements are considered correctly.

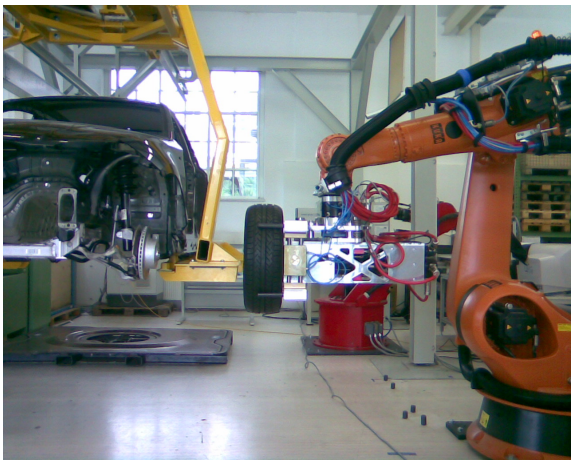

Figure 1: Setup at *iwb* with heavy end-effector for the assembly of a wheel to a conveyed car body.

In contrast, oscillations of the end-effector due to accelerations of the tool center point (tcp) are a disadvantage. They may be solved by a pneumatic locking mechanism as of-

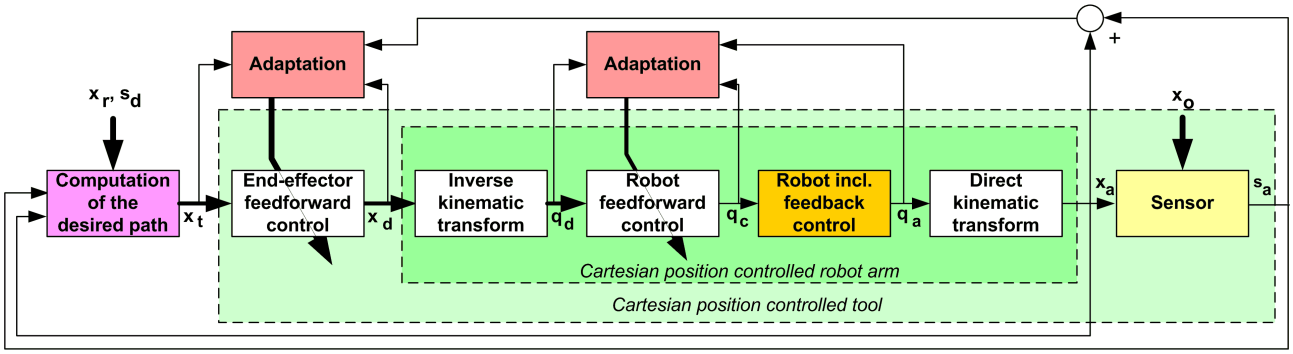

Figure 2: Control architecture for position-based control using a compliant force-torque sensor.

fered in [3]. The idea is to lock the end-effector with the current displacement, then to move it with possibly high accelerations, before unlocking it in order to comply during the assembly process. But this may be not satisfactory if the end-effector is tilted between blocking and releasing. A second drawback is that accelerations just before the expected contact cannot be masked in this way. Therefore a control solution is superior.

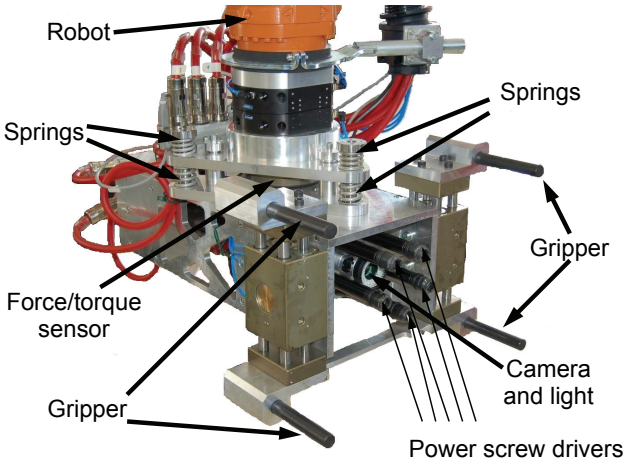

Figure 3: End-effector at *iwb*. Between the springs a sensor is integrated that perceives displacements and thus forces and torques that are exerted by the robot.

Not only oscillations are unfavorable. In addition, quasi static displacements during acceleration phases have to be compensated since they worsen the positional accuracy. Both problems are treated in section 4.

To the knowledge of the authors there is no previous work on the control of oscillations caused by a mechanical compliance at a robot wrist. At DLR the problem has been investigated with different methods. Kamel [6] applied Input Shaping [7, 8, 9] to damp the oscillation of a compliantly suspended end-effector. Here, motion commands are shaped (filtered), such that they do not excite possible oscillations. However, this method is sensitive to the load parameters and does not consider the mentioned offset during the deceleration.

Reconsidered, it is not required to damp or control all displacements to zero. Displacements that are due to desired accelerations can hardly be reduced, but they can be considered in the pose of the robot. This can be realized by a method with an inverse dynamical model of the robot, e.g. the approach of model-predictive control (MPC) [10, 11]. In this way Jehle [12] identified an inverse model of the robot and the sensor (as a mechanical device) and applied it to design a feed-forward controller. Its output is the desired pose for the robot arm, which yields the desired trajectory for the displaced pose of the tcp. Unfortunately, this approach is sensitive to uncertainties in the model of the robot and the underlying position control (see section 2). Therefore an adaptive approach is proposed in this paper, which accounts for the special structure of the used position control.

2 Hierarchical Control

Control is designed in a hierarchical way (Fig. 2). The uppermost level concerns force control, processing the output of the force-torque sensor. This will be explained in section 3. The output of the force controller is a desired pose[1] of the tool $\mathbf{x}_t$ which is the input to a 3-level position control.

Its two lower levels (dark green region) are almost unchanged with respect to [13]. They act on the level of the robot axes. The lowermost (servo) level is represented by the standard robot controller (orange block) which is provided by the robot manufacturer. Its input $\mathbf{q}_c$, the motion command, is the desired position for the position servo loop. $\mathbf{q}_c$ is computed by a feedforward controller (second level) in such a way that a predefined trajectory of the robot axes is exactly tracked, i.e. the *desired* axis positions $\mathbf{q}_d(k)$ are reached by the *actual* axis positions $\mathbf{q}_a$ without time-delay. Transformed to the Cartesian space this means that the actual pose $\mathbf{x}_a$ of the arm is identical to its desired

[1] Transformation matrices of homogeneous coordinates are used and multiplied, since this is the general case. For the sake of clarity, here we use the notation with pose vectors and their summation.

value $\mathbf{x}_d$. This is called an *ideal* robot.
Only a sampled value of the desired position $\mathbf{q}_d(k)$ is not sufficient as control input for the tracking of a given trajectory without delay. In addition, either derivatives of this current position or future sampled values are required. If available in advance, these future values $\mathbf{q}_d(k+i)$ can be processed by a feedforward controller and may result in an ideal position control. Such an inclusion of future timesteps is denoted by a bold face line in Fig. 2.
The robot feedforward controller is designed using a finite impulse response (FIR) filter as the model of the robot. Inversion[2] of this model yields a linear feedforward filter of high order, e.g. 20 (see [14] for details).
The uppermost (third) level of position control (light green region) concerns the difference between the motion of the robot arm[3] and the motion of the tool, i.e. the displacements of the sensor. So it computes the desired trajectory $\mathbf{x}_d$ of the robot arm, considering the displacements $\mathbf{s}_a$ within the suspension of the end-effector, in order to track a given trajectory $\mathbf{x}_t$ of the tcp. This level acts in Cartesian space. The controller is designed in section 4. Its input is provided by the force control algorithm which feeds back the sensor data.
Without contact the complete position control yields

$$\mathbf{x}_a + \mathbf{s}_a = \mathbf{x}_t, \tag{1}$$

i.e. the real (displaced) tool pose and its desired value[4] are identical.

3 Force Control

In contrast to other force control schemes the authors propose to compute the desired pose of the tcp $\mathbf{x}_t(k)$ at the current time step k by

$$\mathbf{x}_t(k) = \mathbf{x}_a(k) + (\mathbf{s}_a(k) - \mathbf{s}_d(k)) \tag{2}$$

from the currently sensed control error and the current actual pose $\mathbf{x}_a(k)$. This assumes that the sensor does not output forces and torques but directly the sensed displacement which, for convenience, is transformed to a displacement $\mathbf{s}_a$ at the tcp. In the same manner, instead of desired forces and torques, a vector $\mathbf{s}_d$ of desired displacements is given. It is computed from the desired forces and torques $\mathbf{f}_d$ by

$$\mathbf{s}_d(k) = \mathbf{C}_t \cdot \mathbf{f}_d(k), \tag{3}$$

where $\mathbf{C}_t$ is the 6×6 compliance matrix. It can be computed by the (diagonal) compliance matrix of the sensor, i.e. its inverse stiffness matrix, and by the compliance of the environment at the contact point.[5] [6]

The use of a *measured* pose $\mathbf{x}_a(k)$ in (2) instead of a previous *desired* pose $\mathbf{x}_t(k-1)$ is crucial for a consistent control. For a fixed object pose $\mathbf{x}_o$ and a constant desired displacement $\mathbf{s}_d$, because of $\mathbf{x}_o = \mathbf{x}_a + \mathbf{s}_a$ it yields a time-invariant desired pose for the robot tcp $\mathbf{x}_t$. This pose is regulated by the position control loop (see green block in Fig. 2) which usually is much faster than the external sensor controlled loop.
This geometric interpretation of force control as the computation of a desired pose allows some extensions, e.g. the integration of compliance in the robot joints or the transformation of displacements from the sensor to the tcp and vice versa. This is fundamental for assembly with minimum clearance. Details can be found in [5].
$\mathbf{x}_r$ is the reference pose which serves as the desired pose of the tcp for those components for which there is no contact force. Without contact $\mathbf{x}_t = \mathbf{x}_r$.
Strictly speaking, $\mathbf{s}_a$ in this section differs from $\mathbf{s}_a$ in the rest of this paper. Extending the notation, $\mathbf{s}_a$ is composed by a contact displacement $\mathbf{s}_{ac}$ and a free space displacement $\mathbf{s}_{af}$, where the latter is caused by the weight of the tool and the acceleration forces. The other sections assume $\mathbf{s}_{ac} = 0$ and thus $\mathbf{s}_a = \mathbf{s}_{af}$, while in this section, so far, $\mathbf{s}_{af}$ was ignored.
So the actual value of position control is $(\mathbf{x}_a + \mathbf{s}_{af})$ instead of $(\mathbf{x}_a + \mathbf{s}_a)$. Then the generic goal is

$$\mathbf{x}_a + \mathbf{s}_{af} = \mathbf{x}_t \tag{4}$$

instead of (1). This modification is important, since $(\mathbf{x}_a + \mathbf{s}_a)$ is constant when being in contact with a stationary object. In contrast, $\mathbf{s}_a$ in (2) is still valid since the sum $\mathbf{s}_{ac} + \mathbf{s}_{af}$ represents the real displacement of the tcp with respect to $\mathbf{x}_a$. Both equations yield $\mathbf{s}_{ac} = \mathbf{s}_d$.
A problem might be the distinction between $\mathbf{s}_{ac}$ and $\mathbf{s}_{af}$ since only $\mathbf{s}_a$ is measured. So $\mathbf{s}_{af}$ has to be predicted model-based, using the accelerations and the inertia of the end-effector that can be identified when there is no contact.

4 Unconstrained Motion

Position control in Fig. 2 is represented by the servo loop (orange block) and two feedforward blocks, one for the decoupled control of the robot which is assumed to be stiff, and the other one for the control of the robot in such a way that the compliantly suspended end-effector tracks the desired path. The latter is especially required in unconstrained motion, i.e. when there is no contact.
Both feedforward controllers are organized in a similar way. The first aims at $\mathbf{q}_a = \mathbf{q}_d$ while the other one is designed to give $\mathbf{x}_a + \mathbf{s}_a = \mathbf{x}_t$. For convenience, in the

[2] Strictly speaking, because of possible modeling errors the model is not numerically inverted but the feedforward filter is estimated using the model.
[3] For convenience, the pose of the robot arm is not defined by the pose of the robot flange but by a virtual pose of the tcp, which is computed by the forward kinematics, assuming no displacement of the sensor.
[4] Note that $\mathbf{q}_c$, $\mathbf{x}_d$ as well as $\mathbf{x}_t$ are denoted as desired trajectories, depending on the level.
[5] Within this paper it is assumed that there is only a single contact with the environment and this contact acts at the tcp.
[6] Note that the displacements $\mathbf{s}_a$ and $\mathbf{s}_d$ have the sign of a distance and not of a deflection, i.e. when pressing in positive z direction, the z displacement is negative. The sign of the force is assumed accordingly.

rest of this section, especially the latter controller and its adaptation is explained.
The controllers are designed as linear filters which weigh the future part of the desired trajectory of the tcp,

$$\mathbf{x}_d(k) = \mathbf{x}_t(k) + \sum_{i=0}^{n} \mathbf{K}_i \cdot (\mathbf{x}_t(k+i) - \mathbf{x}_t(k)). \quad (5)$$

The controller parameters $\mathbf{K}_i$ are computed by adaptation. In this way, accelerations are considered according to their real influence. It is not necessary to extract velocities or accelerations from the given or computed trajectories. Because of couplings, the $\mathbf{K}_i$ are not diagonal.
The adaptation is organized by three steps:

- Identification of a model,
- Computation of the desired controller output,
- Adaptation of the controller parameters.

In contrast to the robot which is modeled by a finite impulse response (FIR [15]), the combination of robot and end-effector is poorly damped. Thus it has to be considered as an infinite impulse response (IIR) system. [7]
So the first step of the adaptation is solved by a least square algorithm, which fits the parameters $\mathbf{A}_i$ and $\mathbf{B}_i$ of the model equation [8]

$$\begin{aligned} &\mathbf{x}_a(k+1) + \mathbf{s}_a(k+1) \\ &= \mathbf{x}_d(k) + \sum_{i=0}^{n_b} \mathbf{B}_i \cdot (\mathbf{x}_d(k-i) - \mathbf{x}_d(k)) \\ &+ \sum_{i=0}^{n_a} \mathbf{A}_i \cdot (\mathbf{x}_a(k-i) + \mathbf{s}_a(k-i) - \mathbf{x}_d(k)). \end{aligned} \quad (6)$$

Assuming an ideal underlying control ($\mathbf{x}_a = \mathbf{x}_d$), this model turns out to represent the effect of accelerations of the tool. Because of couplings, $\mathbf{A}$ and $\mathbf{B}$ are no diagonal matrices.
The second step then computes the desired trajectory $\mathbf{x}_d^*$ of a robot which is assumed stiff, in order to get $\mathbf{x}_a + \mathbf{s}_a = \mathbf{x}_t$. The individual expressions

$$\begin{aligned} &\Delta\mathbf{x}_d(k) + \sum_{i=0}^{n_b} \mathbf{B}_i \cdot (\Delta\mathbf{x}_d(k-i) - \Delta\mathbf{x}_d(k)) \\ &- \sum_{i=0}^{n_a} \mathbf{A}_i \cdot \Delta\mathbf{x}_d(k) \\ &= \mathbf{x}_t(k+1) - (\mathbf{x}_a(k+1) + \mathbf{s}_a(k+1)) \\ &- \sum_{i=0}^{n_a} \mathbf{A}_i \cdot (\mathbf{x}_t(k-i) - (\mathbf{x}_a(k-i) + \mathbf{s}_a(k-i))) \end{aligned} \quad (7)$$

give a linear set of equations, where $\Delta\mathbf{x}_d = \mathbf{x}_d^* - \mathbf{x}_d$ represent the required changes of the desired trajectory $\mathbf{x}_d$. This means that $\mathbf{x}_d^*$ are the optimal controller outputs that yield $\mathbf{x}_t$ instead of the measured results $(\mathbf{x}_a + \mathbf{s}_a)$.
For the interpretation of (7) the expressions with $\Delta\mathbf{x}_d(k)$ can be omitted, if $\sum_{i=0}^{n_a} \mathbf{A}_i + \sum_{i=0}^{n_b} \mathbf{B}_i \approx \mathbf{I}$. This is true since the static transfer factor of the underlying robot position control is close to one. Then (7) can be abbreviated by a matrix equation [9]

$$\mathbf{B} \cdot \underline{\Delta\mathbf{x}}_d = (\mathbf{I} - \mathbf{A}) \cdot (\underline{\mathbf{x}}_t - (\underline{\mathbf{x}}_a + \underline{\mathbf{s}}_a)) = (\mathbf{I} - \mathbf{A}) \cdot \underline{\mathbf{e}}, \quad (8)$$

where underlined vectors denote vectors of poses, i.e. trajectories. In this representation the $N \times N$ matrices $\mathbf{A}$ and $\mathbf{B}$ are almost diagonal, where N is the number of sampling steps in the trajectories.

$$\begin{aligned} &\begin{bmatrix} \mathbf{B}_1 & & & & \\ \cdots & \cdots & & & \\ \mathbf{B}_{n_b} & \cdots & \cdots & & \\ & \cdots & \cdots & \cdots & \\ & & \mathbf{B}_{n_b} & \cdots & \mathbf{B}_1 \end{bmatrix} \cdot \begin{bmatrix} \Delta\mathbf{x}_d(0) \\ \cdots \\ \cdots \\ \cdots \\ \Delta\mathbf{x}_d(N-1) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I} & & & & & \\ -\mathbf{A}_1 & \mathbf{I} & & & & \\ \cdots & \cdots & \cdots & & & \\ -\mathbf{A}_{n_a} & \cdots & \cdots & \cdots & & \\ & \cdots & \cdots & \cdots & \mathbf{I} & \\ & & -\mathbf{A}_{n_a} & \cdots & -\mathbf{A}_1 & \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{e}(1) \\ \cdots \\ \cdots \\ \cdots \\ \mathbf{e}(N) \end{bmatrix} \end{aligned} \quad (9)$$

The computation of $\underline{\Delta\mathbf{x}}_d$ is done by minimizing the right hand side of (8). This is done by estimation using an information filter (also called inverse Kalman filter [16]), since this can be realized numerically efficiently because of the sparsity of the matrices [14]. But, strictly speaking, the minimization of the right hand side of (8) does not minimize the control error $\underline{\mathbf{e}}$ but an error which is reduced by the system dynamics. This means that oscillations are not considered, only the disturbances which excite the oscillations.
The alternatives to estimate $\underline{\Delta\mathbf{x}}_d$ from $(\mathbf{I} - \mathbf{A})^{-1} \cdot \mathbf{B} \cdot \underline{\Delta\mathbf{x}}_d = \underline{\mathbf{e}}$ or to compute $\underline{\Delta\mathbf{x}}_d = \mathbf{B}^{-1} \cdot (\mathbf{I} - \mathbf{A}) \cdot \underline{\mathbf{e}}$ directly, both, process non sparse matrices and therefore need much more computing power. Therefore the approximation of estimating $\underline{\Delta\mathbf{x}}_d$ from the right hand side of (8) seems adequate.
In this way, after executing a test run, the controller outputs are set to $\mathbf{x}_d^*$ and thus yield a smaller control error. This can be repeated several times, since the linear model is rather simple and thus the expected result will not be reached at once. Finally, the desired trajectory of the tcp will be executed with small control errors.
The third step is not obligatory for trajectory control. When being used, new test trajectories can be executed

[7] The representation of a model as a finite impulse response function is similar to (6), but with $n_a = 0$ and n_b being the length of the impulse response. With $n_b \leq 20$ that method works well, even with higher order process dynamics. However, a poorly damped system requires $n_b \gg 20$ which yields to numeric problems. In contrast, in an IIR model $n_a \approx n_b$ correspond to the system order which is much smaller. An IIR model is usually implemented by a transfer function with matrix parameters $\mathbf{A}_i$ and $\mathbf{B}_i$.

[8] $\mathbf{x}_d(k)$ is a moving working point.

[9] Because of numerical reasons, (7) is used instead of (8). Equations (8) and (9) are shown only for the interpretation.

with small error, without further test runs for the adaptation. This step is required however, if sensor data are used, since then the trajectories always differ.
This step computes controller parameters that yield $\mathbf{x}_d^*$, given the desired trajectory $\mathbf{x}_t$. A possible equation is (5). An alternative formulation would be

$$\mathbf{x}_d^*(k) = \mathbf{x}_t(k) + \sum_{i=-n_1}^{n_2} \mathbf{K}_i \cdot (\mathbf{x}_t(k+i) - \mathbf{x}_t(k)). \quad (10)$$

Both variants only process the target trajectory, no measured values of the current test run. Therefore they are inherently stable. On the other hand this approach only compensates for systematic path errors due to the compliant setup and the desired motion. Stochastic disturbances cannot be compensated in this way. But disturbances on the desired motion, e.g. due to a sensed stochastically disturbed pose of the target object, will cause a modified target trajectory $\mathbf{x}_t$ and thus modified controller outputs $\mathbf{x}_d$. This is not a problem if the controller is adequate. Only those disturbances are not compensated, that displace the end-effector without being excited by a controller. Fortunately, these disturbances are quite small.

5 Experiments

The experiments have been executed using a KUKA KR16 robot with a KRC2 controller (Fig. 4). Using test trajectories [10], first, the parameters of the *robot* feedforward controller are adapted. After a single run that identifies the 6×6 model ($n_a = 0$, $n_b = 20$), about 100 iterative test runs are executed until the *robot* controller ($n = 30$) has converged. Then, holding this controller, the *end-effector* controller is adapted. This is done in the same way, but with the shown IIR model ($n_a = n_b = 4$) instead of the FIR approach. Both is realized fully automatically, without any intervention or tuning by a human.
For the demonstration of the performance the desired trajectory is a Cartesian back and forth motion in the z direction of the tcp. Figures 5 and 6 show the influence of the end-effector control in the moved component. Without considering the compliance, the end-effector is displaced by the desired translational acceleration, overlaid by an unwanted oscillation of the tilting component. The control error is bigger than the displacement since the robot feedforward controller is not ideal. With the end-effector trajectory computed according to (7), the translational effect is compensated, though the displacements are still present. But the oscillation is not yet compensated. This is because of the interface to the robot which acts as a low-pass filter and thus inhibits high frequency actions. In addition, the effect of medium frequency control actions is not modeled properly, resulting in slightly bigger amplitudes in the first part. The oscillation damping might be superior when using a KRC4 controller ([17, 18]) which allows a higher sampling rate (250 Hz for the sensor interface) and less filtering.

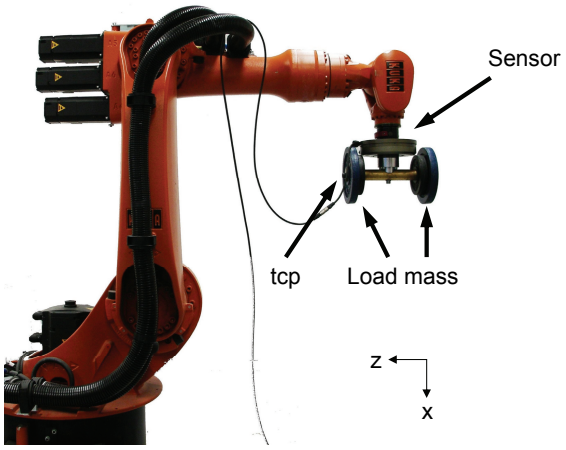

Figure 4: Setup for the experiments, using a compliant sensor with internal springs.

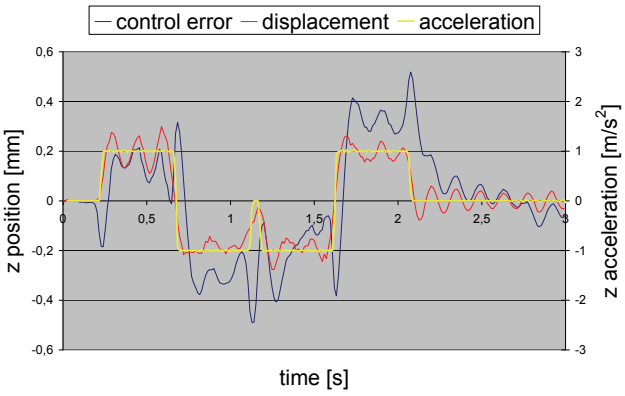

Figure 5: Displacement and control error without end-effector control.

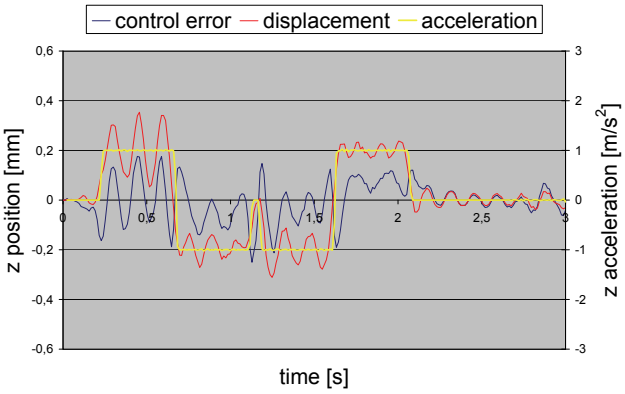

Figure 6: Displacement and control error when using an optimized trajectory for the robot arm.

[10] A test trajectory is used which moves the individual Cartesian components one after the other, back and forth. Instead, for the identification of the model, sine signals with increasing frequency (chirp) are used.

The control error is still inferior when an end-effector feedforward controller according to (5) is applied to compute $\mathbf{x}_d(k)$ instead of using the optimized $\mathbf{x}_d^*(k)$ itself. Nevertheless, even in this case, an improvement is obvious. These results will be the motivation for further revisions of the adaptation scheme.

6 Conclusion

The paper argues that compliant devices as a compliant force-torque sensor are useful for demanding assembly tasks. Force control as well as position control of the displaced end-effector can be designed in such a way that the compliance is explicitly considered.

Acknowledgment

This work is funded by KUKA Laboratories, Augsburg, Germany.

References

[1] G. Reinhart and J. Werner. Flexible automation for the assembly in motion. *Annals of the CIRP*, 56(1):25–28, 2007.

[2] H. Chen, W. Eakins, J. Wang, G. Zhang, and T. Fuhlbrigge. Robotic wheel loading process in automotive manufacturing automation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3814–3819, St. Louis, USA, Oct. 2009.

[3] Schunk GmbH & Co KG. *SCHUNK - Workholding Solutions, Automation Components, Toolholding Components, Gripping Systems, Linear Systems*. Lauffen, Germany. `http://www.schunk.com/schunk/schunk_websites/products/products_level_3/product_level3.html?country=INT&lngCode=EN&lngCode2=EN&product_level_1=244&product_level_2=252&product_level_3=296` last visited 2012.

[4] F. Lange, B. Willberg, and G. Hirzinger. Control of large forces and torques using an asymmetrically arranged compliant sensor. In *Proc. Joint 41th Int. Symp. on Robotics and 6th German Conf. on Robotics ISR/ROBOTIK 2010*, Munich, Germany, June 2010.

[5] F. Lange, C. Jehle, M. Suppa, and G. Hirzinger. Revised force control using a compliant sensor with a position controlled robot. In *Proc. 2012 IEEE Int. Conf. on Robotics and Automation (ICRA)*, St. Paul, MN, USA, May 2012.

[6] A. Kamel, F. Lange, and G. Hirzinger. Vibration reduction of a compliant force/torque sensor using an industrial-robots-suited command preshaping control scheme. In *The 9th Int. Conf. on Motion and Vibration Control (MOVIC)*, Munich, Germany, September 2008.

[7] S. Rhim and W. Book. Adaptive time-delay command shaping filter for flexible manipulator control. *IEEE/ASME Transactions on Mechatronics*, 9:619–626, 2004.

[8] W. Singhose. Command shaping for flexible systems: A review of the first 50 years. *International Journal of Precision Engineering and Manufacturing*, 10(4):153–168, Oct. 2009.

[9] N. C. Singer, W. E. Singhose, and W. P. Seering. Comparison of filtering methods for reducing residual vibration. *European Journal of Control*, 5:208–218, 1999.

[10] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11:733–764, 2003.

[11] E. F. Camacho and C. Bordons. *Model Predicitve Control*. Springer, 2004.

[12] C. Jehle. Entwicklung einer prädiktiven Vorsteuerung für einen nachgiebig angebrachten Endeffektor. Technical Report IB 515-10-18, DLR, Oberpfaffenhofen, 2010. (in German).

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

[14] F. Lange. *Adaptiv vorausplanende Steuerung für schnelle sensorbasierte Roboterbewegungen*. PhD-Thesis, University of Karlsruhe, 2003. `http://www.ubka.uni-karlsruhe.de/cgi-bin/psview?document=2003/informatik/1` (in German).

[15] L. R. Rabiner and B. Gold. *Theory and Application of Digital Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1975.

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

[17] KUKA industrial robots - KRC4. KUKA Roboter GmbH. `http://www.kuka-robotics.com/germany/en/products/controllers/kr_c4/start.html` last visited 2012.

[18] KUKA Roboter GmbH. *KUKA.RobotSensorInterface 3.0*, 2009. manual KST_RSI_30_en.