[2] B. Armstrong-Hélouvry, P. Dupont, and C. Canudas de Wit. "A survey of models, analysis tools and compensation methods for the control of machines with friction". *Automatica*, Vol. 30:1083–1138, 1994.

[3] P. R. Bélanger. "Estimation of angular velocity and acceleration from shaft encoder measurements". In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 585–592, Nice, France, May 1992.

[4] M. Corless and G. Leitmann. "Continuous state feedback guaranteeing uniform ultimate boundedness for uncertain dynamic systems". *IEEE Trans. Automat. Control.*, Vol. AC-26:1139–1144, 1981.

[5] Jaritz, André , "An experimental comparison of passivity based robust controllerson a direct drive manipulator' ", CSL Report UILU-ENG-95-2220, University of Illinois, June 1995.

[6] F.L. Lewis, C.T. Abdallah, and D.M. Dawson. *Control of Robot Manipulators.* MacMillan, New York, 1993.

[7] G. Liu and A. Goldenberg. "On robust saturation control of robot manipulators". *Proceedings of the 32nd Conference on Decision and Control*, pages 2115–2120, December 1993.

[8] Nethery, J., and Spong, M.W., "Robotica: A Mathematica Package for Robot Analysis," *IEEE Robotics and Automation Magazine*, Vol. 1, No. 1, Mar., 1994.

[9] R. Ortega and M. W. Spong. "Adaptive motion control of rigid robots: a tutorial". *Automatica*, Vol. 25:877–888, 1989.

[10] Parker Hannifin Corporation. *Compumotor Digiplan Positioning Control Systems and Drives*, 1992.

[11] M. W. Spong. "On the robust control of robot manipulators". *IEEE Transaction on Automatic Control*, Vol. 37:1782–1786, 1992.

[12] SSPA Systems, Göteborg, Sweden. *Simnon, User's Guide for MS-DOS Computers*, January 1990.

[13] M. W. Spong and M. Vidyasagar. *Robot Dynamics and Control.* John Wiley, New York, 1989.

[14] P. J. Walsh. "Feedback linearization of a robotic manipulator". Master's thesis, University of Illinois at Urbana-Champaign, 1994.

[15] S. Zenieh and M. Corless. "Simple robust tracking controllers for robotic manipulators". In *Proc. SYROCI'94*, Capri, Italy, September 1994.

Table 11: Nominal parameter vector

θ_{10}	θ_{20}	θ_{30}
0.731	0.108	0.082

12.2 Parameters for the Algorithm ZC of Zenieh and Corless

We calculate the bounds β_i using the parameters of the unloaded and the maximum loaded D2R2 as

$$\begin{aligned}\beta_0 &= 0.0276\\ \beta_1 &= 1.3254\\ \beta_2 &= 0.2571\end{aligned}$$

We do not consider any friction in this calculation, because we are compensating the robot for the friction terms. Note that β_0 is not used in the actual controller, which means that in this design only upper bounds for the Euler-Lagrange dynamics are used in the control law.

12.3 Parameters for the Algorithm LG+ of Liu and Goldenberg

The values of θ_{M_0} and θ_{C_0} are calculated as

$$\boldsymbol{\theta}_{M_0} = \begin{pmatrix} 0.8386 + 0.1638\cos(q_2) \\ 80.1076 + 0.08190\cos(q_2) \\ 0.1076 \end{pmatrix}$$

$$\theta_{C_0} = -0.8190\sin(q_2)$$

This yields

$$\begin{aligned}\Delta D_{11} &= 0.1281 + 0.1065|\cos(q_2)|\\ \Delta D_{12} &= 0.07502 + 0.05327|\cos(q_2)|\\ \Delta D_{22} &= 0.07502\\ \rho_M &= \sqrt{\Delta D_{11}^2 + \Delta D_{12}^2 + \Delta D_{22}^2}\\ \Delta h &= \rho_C = 0.050327|\sin(q_2)|\end{aligned}$$

References

[1] Y. Aoustin and B. Cherki. “Computed torque of robots with estimated velocities”. In *Proc. SYROCI'94*, Capri, Italy, September 1994.

Table 8: θ_i for the unloaded arm

θ_1	θ_2	θ_3
0.678	0.033	0.029

Regarding the velocity estimates we found an acceptable solution using our averages of section 6. Using an observer design was also found to work, although its design is more complicated.

12 Appendix

12.1 Parameters for the Algorithms SP1 and SP2 of Spong

Identification of the robot D2R2 was performed in [14] and hence nominal values of the parameters were available. For the parametrization given by (14) the calculated values of these parameters for the unloaded arm are shown in (table 8).

With the load attached to the end of the second link, we designed a controller to provide robustness in the intervals

$$\begin{aligned} 0\text{ kg} &\leq \Delta m_2 \leq 1.1841\text{ kg} \\ 0\text{ m} &\leq \Delta l_{c2} \leq 0.0694\text{ m} \\ 0\text{ kgm}^2 &\leq \Delta I_2 \leq 0.109\text{ kgm}^2 \end{aligned} \tag{93}$$

Then we calculated $\boldsymbol{\theta}$ for the maximum loaded arm (table 9). The nominal parameter vector $\boldsymbol{\theta}_\text{o}$ is

Table 9: θ_{il} for the loaded arm (both weights including bolts)

θ_{1l}	θ_{2l}	θ_{3l}
0.784	0.183	0.135

chosen as the mean value of the unloaded and loaded arm (table 11). For the algorithm SP2, the elements of $\tilde{\boldsymbol{\theta}}$ are bounded separately which yields the vector $\boldsymbol{\rho}$ shown in table 10. The uncertainty

Table 10: Uncertainty bounds $\rho_i = \theta_{i0} - \theta_i$

ρ_1	ρ_2	ρ_3
0.053	0.075	0.053

bound ρ used in the algorithm SP1 is the Euclidean norm of $\boldsymbol{\theta}_0 - \boldsymbol{\theta}_\text{unloaded}$ equals $\rho = 0.1062$

were the simplest to design and showed the best performance. The algorithm LG+ is more difficult to design and and to implement and showed inferior performance to SP1 and SP2. For this reason, the added complexity of generating separate uncertainty measures for the inertia, Coriolis (and gravity) terms does not seem justified. The tracking performance of the algorithm ZC, although clearly inferior to the other algorithms, is still remarkably good given the simplicity of its on–line computational requirements.

It is natural at this point to conjecture how these algorithms might compare for robots with higher degree of freedom. The algorithm ZC is attractive because the on–line computational requirements do not increase significantly as the number of links increases. The computation of the uncertainty bounds becomes more difficult, but these are performed off–line. However, since the algorithm basically functions as a nonlinear, high–gain PD controller, it is very likely, in order not to excite unmodeled dynamics and reject disturbances, that the controller gains (especially ϵ) would have to be detuned to such an extent that tracking performance would suffer. In other words, the algorithm is simple but conservative as a result.

The algorithm of Liu/Goldenberg cannot be recommended at this point, since the performance was not better than the simpler algorithms of Spong. There is no reason to expect this situation would change as the number of links increases.

The algorithms SP1 and SP2 differ only in the number of uncertainty bounds that must be calculated and the number of parameters ϵ_i that must be tuned. These differences are only important if the values for the uncertainty bounds ρ_i design vary too widely so that using a single bound leads to a conservative design. For multi–DOF systems, both the off–line calculation of the regressor matrix and its on–line computation become an issue. The off–line computation problem is made easier by the increasing availability of symbolic software, such as the package Robotica [8]. On–line computation is also helped greatly by the ever increasing speed and decreasing cost of microprocessor based controllers. Thus, ease of design is increasing the most important factor in control system design. In light of this, the algorithms SP1 and SP2 are recommended unless its on–line computation cannot be achieved in which case the algorithm ZC is recommended.

We note that, although we were able to achieve the high sample rate of 2500Hz, good performance is still expected at lower rates which, of course, increases the amount of on–line computation possible. In other words, we would expect performance to degrade gracefully with decreasing sample rate (up to a point). It would be interesting, therefore, to produce a second comparision between the algorithms SP1, SP2, and ZC where the sample rate is varied to reflect the computational advantages of ZC. Such a study is currently under consideration.

Friction compensation as proposed in section 7.1 has proven to improve performance quite well, although the approximation of the friction model was rather inaccurate. Further improvement of the friction model used will improve tracking accuracy, however the improvement will be only marginal, since the largest fraction of the friction model is already incorporated.

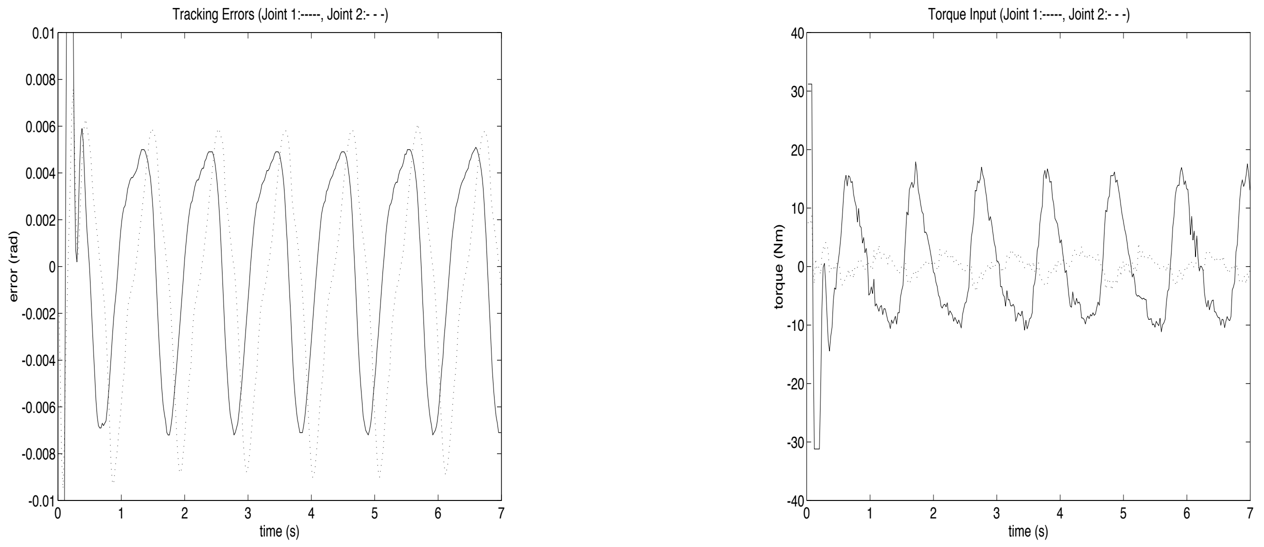


Figure 10: Algorithm ZC, Circle Trajectory, Friction Compensation, Maximum Load

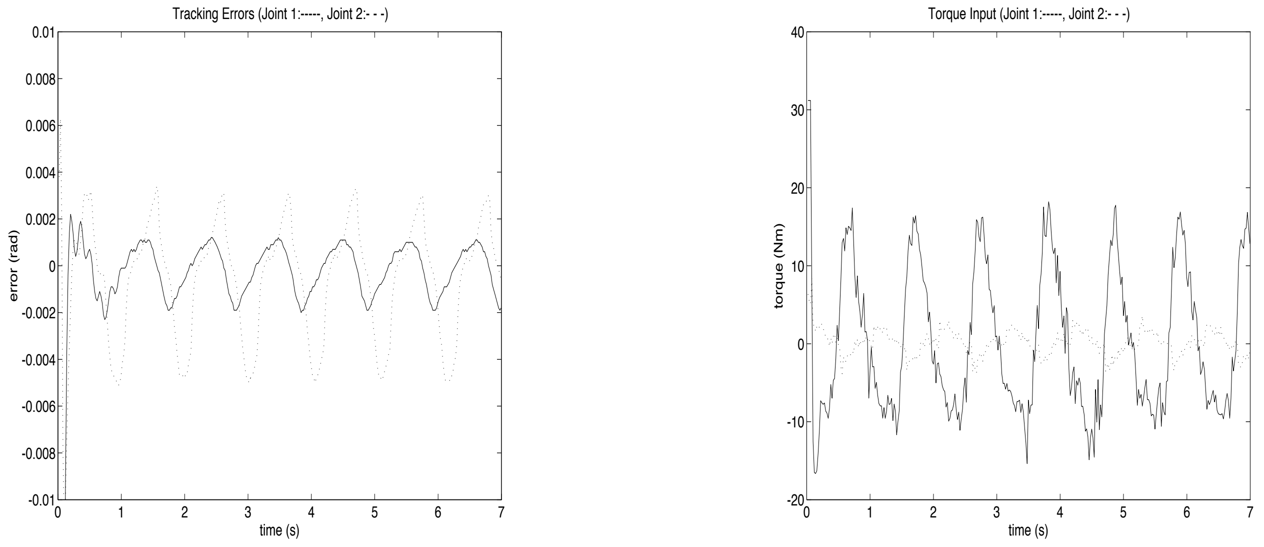
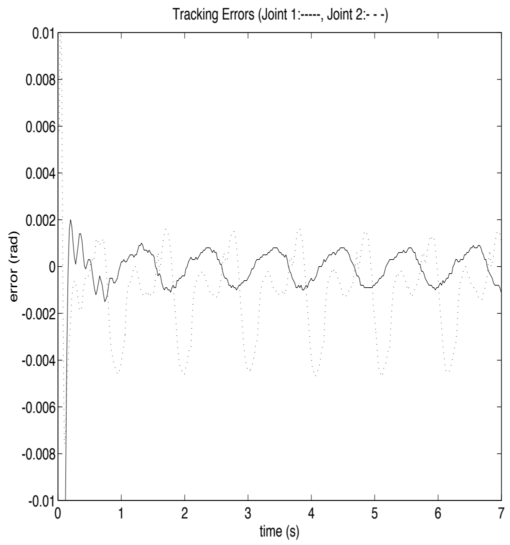


Figure 11: Algorithm LG+, Circle Trajectory, Friction Compensation, Maximum Load



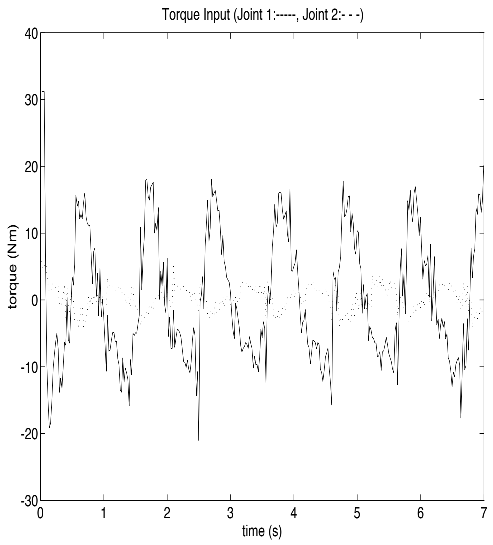
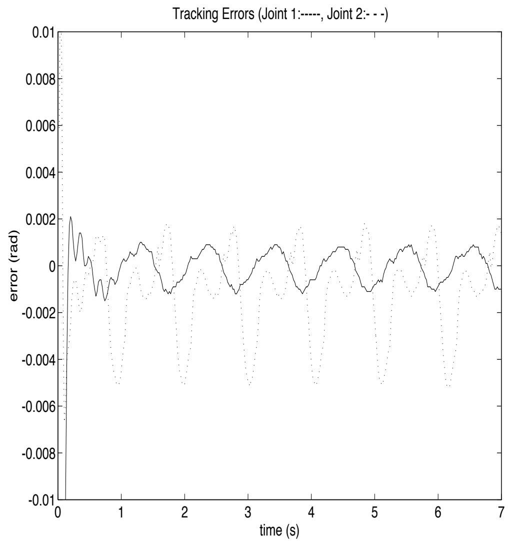


Figure 8: Algorithm SP1, Circle Trajectory, Friction Compensation, Maximum Load



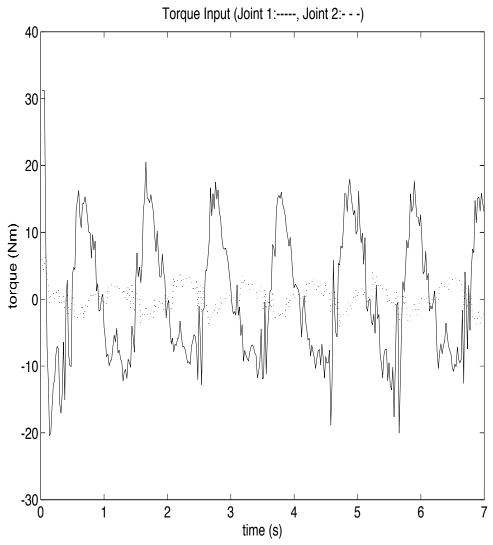


Figure 9: Algorithm SP2, Circle Trajectory, Friction Compensation, Maximum Load

circle and not the tracking from a reposing position to the circle.

Principally we wanted to hold our values for ϵ constant, but during this experiment we had to increase some of the ϵ to reduce occurring vibrations in order to improve tracking. Specifically we had to increase ϵ to 45 for the algorithm ZC and $\epsilon_M = 20$ and $\epsilon_C = 2.5$ for the algorithm of LG+. With these values the algorithm of ZC still showed vibrations for the robot carrying no load. The results are presented in table 6 and 7. As before, the algorithms SP1 and SP2 gave the best results.

Table 6: Circle trajectory with friction compensation, no load

Algorithm	in rad				in m	
	$\|error_1\|$	$maxerr_1$	$\|error_2\|$	$maxerr_2$	$error_{\text{tasksp.}}$	$maxerr_{\text{tasksp.}}$
SP1	0.00422	0.0007	0.0349	0.0041	0.0104	0.00119
SP2	0.00424	0.0005	0.0331	0.0037	0.0100	0.00113
ZC	0.0681	0.009	0.0602	0.0109	0.0375	0.00527
LG+	0.00755	0.0012	0.0453	0.0041	0.0151	0.00146

At 6radians/sec angular speed the maximum deviation of the end effector from the circle was just over 1mm with no load and still less than 2mm with load, which is excellent performance. The algorithm LG+ did not come as close to the performance of SP1 and SP2 as it did for the CPT trajectory in terms of tracking errors. The algorithm ZC had errors for the first joint about ten times higher than the other controllers, with a task space error of nearly 4 times higher.

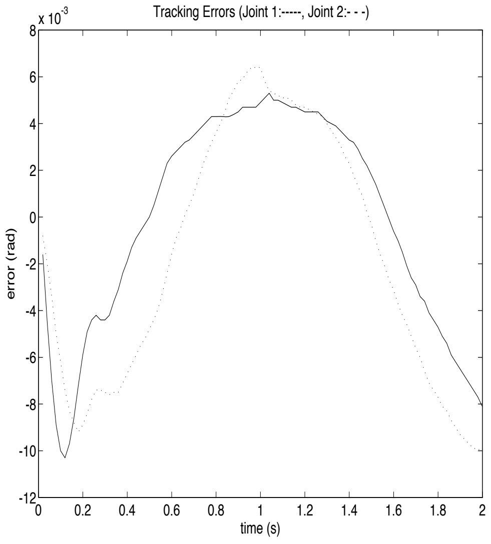
Table 7: Circle trajectory with friction compensation, load

Algorithm	in rad				in m	
	$\|error_1\|$	$maxerr_1$	$\|error_2\|$	$maxerr_2$	$error_{\text{tasksp.}}$	$maxerr_{\text{tasksp.}}$
SP1	0.0106	0.0011	0.0352	0.0047	0.0114	0.00161
SP2	0.0115	0.0012	0.0381	0.0052	0.0124	0.00181
ZC	0.0741	0.0072	0.0766	0.0090	0.0398	0.00383
LG+	0.0170	0.002	0.0426	0.005	0.0163	0.00182

Figures 8 through 11 show the tracking errors and the torque inputs for the various algorithms with maximum load for the circle trajectory. The tracking errors during the first second are much larger than during the remaining time, because of the initial velocity error.

11 Conclusion

For the 2-link D2R2 robot none of the algorithms was difficult to design and to implement and all showed good performance. A few general comments are in order. The algorithms SP1 and SP2



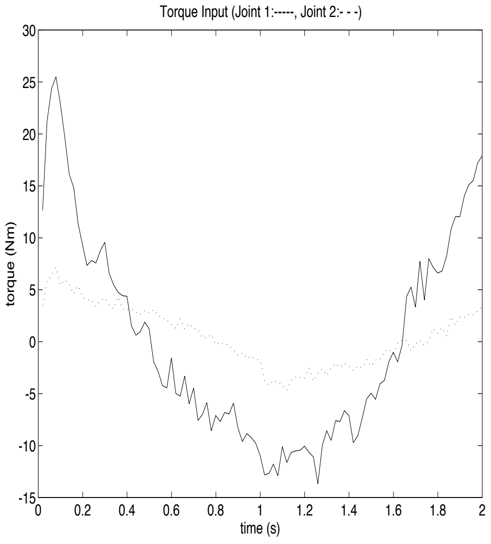
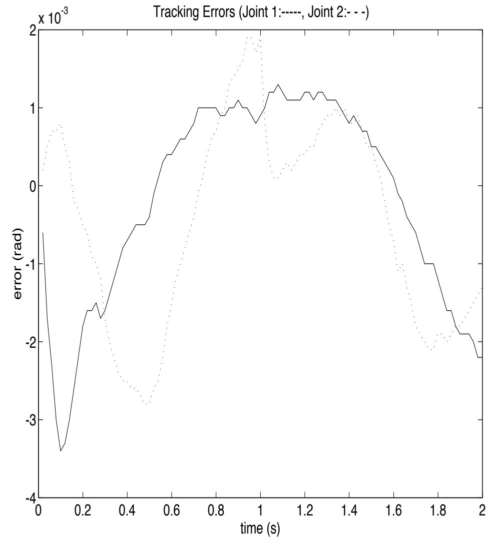


Figure 6: Algorithm ZC with CPT Trajectory, Friction Compensation, Maximum Load



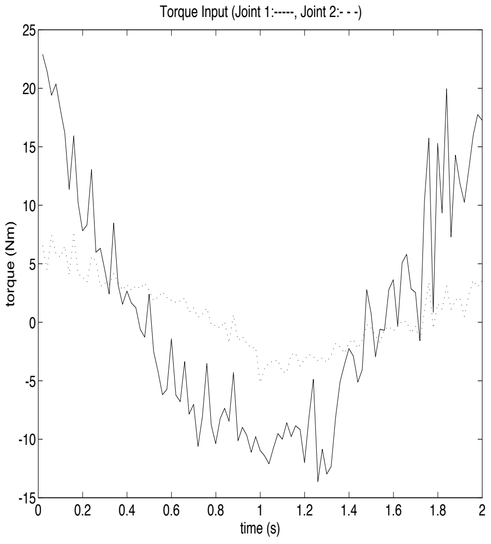


Figure 7: Algorithm LG+ with CPT Trajectory, Friction Compensation, Maximum Load

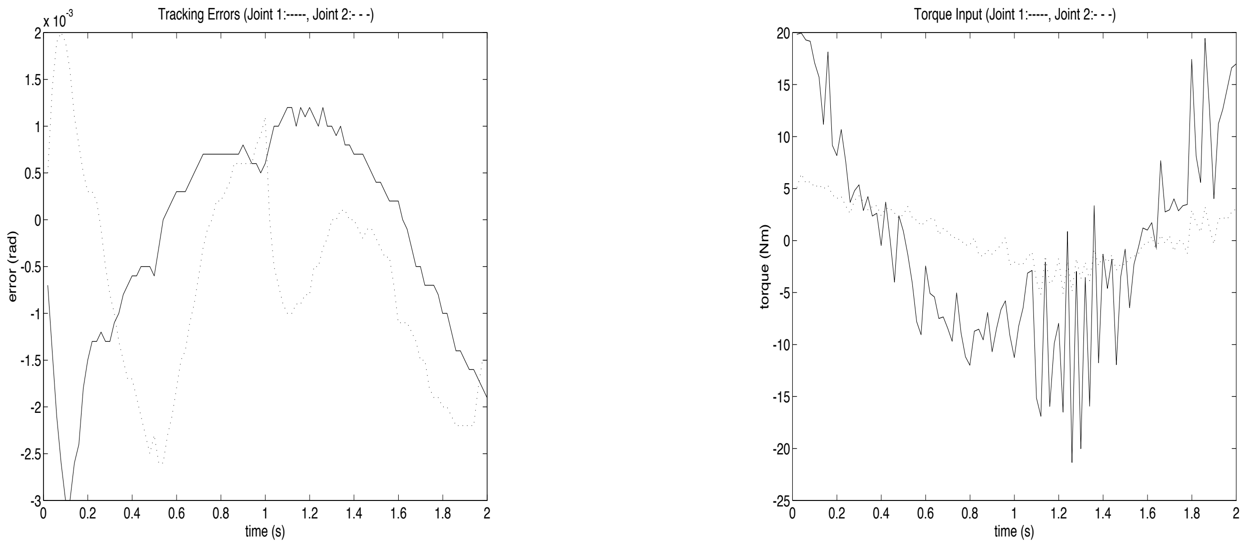


Figure 4: Algorithm SP1 with CPT Trajectory, Friction Compensation, Maximum Load

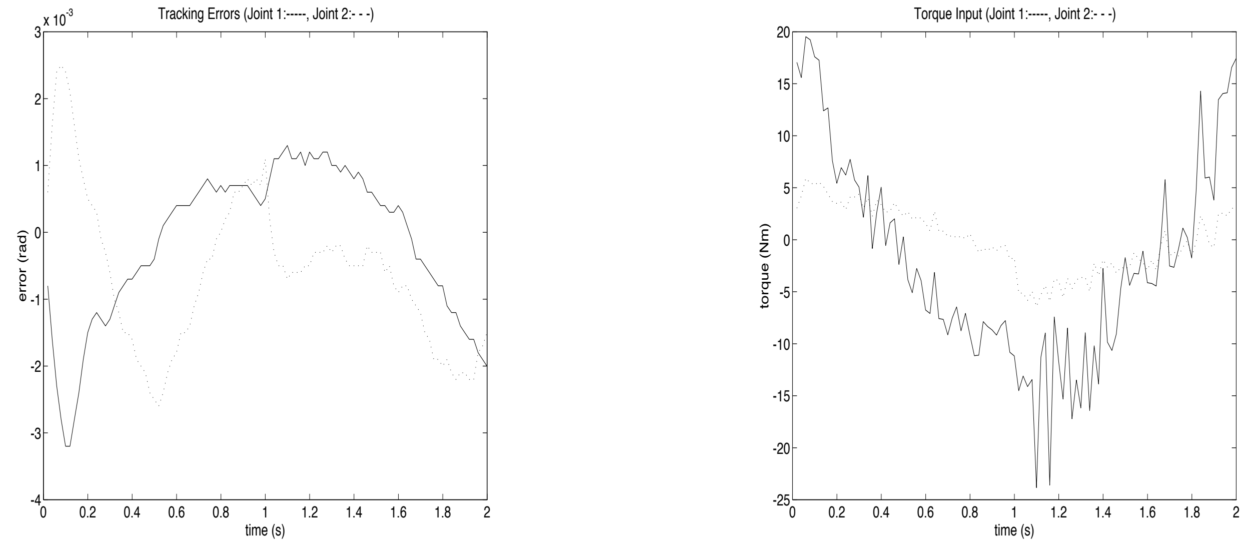


Figure 5: Algorithm SP2 with CPT Trajectory, Friction Compensation, Maximum Load

Table 4: CPT with friction compensation, no load

Algorithm	in rad				in m	
	$\|error_1\|$	$maxerr_1$	$\|error_2\|$	$maxerr_2$	$error_{\text{tasksp.}}$	$maxerr_{\text{tasksp.}}$
SP1	0.00442	0.0009	0.0224	0.0047	0.00746	0.00135
SP2	0.00472	0.0010	0.0223	0.0052	0.00737	0.00144
ZC	0.0240	0.0042	0.113	0.0028	0.0156	0.00300
LG+	0.00478	0.0008	0.0251	0.0053	0.00933	0.00195

Table 5: CPT with friction compensation, load

Algorithm	in rad				in m	
	$\|error_1\|$	$maxerr_1$	$\|error_2\|$	$maxerr_2$	$error_{\text{tasksp.}}$	$maxerr_{\text{tasksp.}}$
SP1	0.0110	0.0030	0.0129	0.0026	0.00774	0.00161
SP2	0.0114	0.0032	0.0130	0.0026	0.00754	0.00165
ZC	0.0459	0.0103	0.0568	0.0101	0.0405	0.00839
LG+	0.0130	0.0034	0.0139	0.0028	0.0099	0.00183

accuracy is improved by up to 50 % and more depending on the error criteria. Still the algorithms SP1, SP2, and LG+ perform best. However the algorithms SP1 and SP2 seem to be slightly superior, which is caused by the lower value for ϵ.

Figures 4 through 7 show the tracking errors and the torque inputs of the algorithms SP1, SP2, ZC, and LG+ with maximum load and with friction compensation for the cubic polynomial reference trajectory. Plots of all responses are available in [5].

10 Circle Trajectory

In this section we present experimental results for a reference trajectory consisting of a circle in task space. Due to the limit of the motor torque a maximum angular velocity of $\omega = 6\frac{\text{rad}}{\text{s}}$ was achievable(radius $r = 0.1$ m, center=(0.4 m, 0 m)). Higher angular velocities resulted in saturated input torques when the maximum load was attached to the D2R2. Friction compensation was included in all experiments.

The motion was performed for 7 seconds after first moving the robot to a position on the circle. Since the trajectory included only the continuous tracking of the circle the initial velocity error is large, which results also in a relatively large initial position tracking error. To compensate the error criteria for this large initial tracking error, we used only the last 6 seconds of the trajectory data in the calculation of the error measures. By this method we compare only the tracking of the

Table 1: The values for ϵ

SP1	$\epsilon = 1$
SP2	$\epsilon_i = 0.5\ \forall i$
LG+	$\epsilon_M = 1.5$, $\epsilon_C = 1$
ZC	$\epsilon = 25$

The first series of experiments was performed without any compensation for friction. The results are shown in table 2 for the D2R2 without load and in table 3 with maximum load. As one can

Table 2: CPT, no load

Algorithm	in rad				in m	
	$\lvert error_1 \rvert$	$maxerr_1$	$\lvert error_2 \rvert$	$maxerr_2$	$error_{\text{tasksp.}}$	$maxerr_{\text{tasksp.}}$
SP1	0.00624	0.0014	0.0644	0.0097	0.0216	0.00339
SP2	0.00646	0.0014	0.0691	0.0099	0.0231	0.00342
ZC	0.0265	0.0052	0.0672	0.0098	0.0299	0.00471
LG+	0.00722	0.0016	0.0594	0.0099	0.0207	0.00332

Table 3: CPT, load

Algorithm	in rad				in m	
	$\lvert error_1 \rvert$	$maxerr_1$	$\lvert error_2 \rvert$	$maxerr_2$	$error_{\text{tasksp.}}$	$maxerr_{\text{tasksp.}}$
SP1	0.0126	0.0030	0.0636	0.0108	0.0235	0.00375
SP2	0.0129	0.0032	0.0666	0.0108	0.0246	0.00391
ZC	0.0494	0.0121	0.0934	0.0166	0.0503	0.0113
LG+	0.0143	0.0036	0.0600	0.0108	0.0231	0.00383

see, the algorithms SP1 and SP2 and LG+ result in the lowest tracking errors, while the errors for the ZC algorithm are considerably larger. Since the ZC design does not uses any lower bounds for the uncertainty interval (remember, only β_0 is a lower boundary for the inertia matrix, but it is not used in the control law) its design is very conservative. For this reason the corresponding value for ϵ had to be chosen relatively high in order to avoid resonance which detunes the system. The results are higher tracking errors.

A second series of experiments was performed with the same reference trajectory with friction compensation as proposed in section 7.1. For the parameters of the Coulomb friction we chose $\hat{k}_1 = 1$ and $\hat{k}_2 = 2$. The other parameters remained unchanged. The results are shown in table 4 for the D2R2 without load and in table 5 with maximum load.

Comparing the tracking errors to the same trajectory without friction compensation we see that parameter vector into the uncertainty parameter vector $\boldsymbol{\theta}$ as

$$\boldsymbol{\theta} = \begin{pmatrix} m_1 l_{c1}^2 + m_2 l_1^2 + I_1 \\ m_2 l_{c2}^2 + I_2 \\ m_2 l_1 l_{c2} \\ k_1 \\ k_2 \end{pmatrix} \tag{91}$$

which leads to the redefined regressor

$$\begin{bmatrix} a_1 & a_1 + a_2 & y_{13} & \operatorname{sgn}(\dot{q}_1) & 0 \\ 0 & a_1 + a_2 & y_{23} & 0 & \operatorname{sgn}(\dot{q}_2) \end{bmatrix} \tag{92}$$

However, when this controller was implemented, the tracking performance was not better than the friction compensation of section 7.1, probably due to the fact that our estimates for the friction parameters are sufficiently good. When the friction terms were incorporated into the other algorithms, ZC and LG, the performance was, in fact, much worse. Furthermore the friction does not change significantly, which means that the controllers do not have to be robust to a change in the friction parameters. For this reason we did not include the friction compensation in the robust controller, but simply added the fixed compensation term to the total control input torque for all algorithms.

8 Experimental Results

For the comparison of the different algorithms we ran every controller with the same fixed gains K, Λ on several trajectories to analyze the strength and weakness of each design. As mentioned previously we wanted to chose the gains as high as possible; for that purpose we chose $K = \text{diag}(30, 10)$ and $\Lambda = \text{diag}(50, 20)$. For the velocity estimation we used (64) and (66) as described in section 5. For reasons of space we present here only the results for two trajectories, namely, the joint space cubic polynomial trajectory, and the circle in task space. See [5] for the complete data on the other trajectories.

9 Cubic Polynomial Trajectory (CPT)

In this section we present results for a cubic polynomial reference trajectory. The reference consists of two consecutive trajectories, each lasting for 1 second. In order to pick the lowest possible value for the ϵ_i in the various algorithms we first ran the experiment without load, since for this mechanical configuration vibrations are most likely. The resulting ϵ values are listed in table 1. We denote by LG+ our modification of the original algorithm LG that places the nominal parameter vector in the middle of the uncertainty interval. For further design parameters see the Appendix.

$\hat{k}_1$ and $\hat{k}_2$ are not necessarily equal. For instance for the D2R2 we found that $\hat{k}_2$ is about twice as large as $\hat{k}_1$, because the motors are designed to carry heavy loads. This results in relatively higher friction for a load less than the load for which they are designed. Generally friction compensation using our finite differences method for the velocity filter achieved the best improvements. The Kalman filter and the nonlinear observer used for the control without friction compensation performed quite inferior compared to the finite difference filters when used with friction compensation. The main reason for that is probably that these filters have to be tuned again for the friction compensated system, which is an disadvantage of the observer filter design.

Compensating for the viscous friction did not improve performance significantly, which is due to the fact that the coefficients of this friction are small enough to be neglected, since our joint velocities are not too high. Figure 3 shows the friction force as a function of velocity (solid line) and a friction approximation (dotted line) using only the Coulomb friction. As one can see, for small velocities this approximation comes very close to the real friction for low velocities.

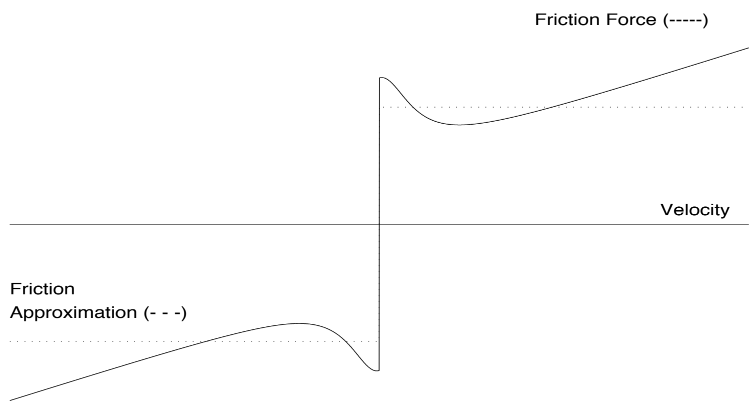


Figure 3: Friction force and friction approximation

7.2 Friction Compensation using the Robust Control Law

Another method to compensate for the friction, especially if we are uncertain about the magnitude of the friction, is to include it into the robust control law. For example, to compensate for the Coulomb friction using the algorithms (SP1) or (SP2) we would include the Coulomb friction with T the sampling period. We chose $L_1 = 3000, L_2 = 2.25 \cdot 10^6$ for the observer gains which achieves a performance as good as the Kalman filter in section (6.1). Including the nonlinearity $M^{-1}(\mathrm{x}_1)\tau$ in the observer improves tracking compared to critically damped linear observer, furthermore we could also add the nonlinear Coriolis and centrifugal terms. However this nonlinear observer performs equivalent to a linear Kalman design in terms of noise reduction and tracking accuracy.

7 Friction and Friction Compensation

Even though the manipulator is direct–drive and so does not suffer from gear friction, there is nevertheless considerable bearing friction. For this reason we investigated various friction compensation schemes to compensate for friction of the form

$$F(\dot{\mathrm{q}}) = (F_v|\dot{\mathrm{q}}| + F_C))\mathrm{sgn}(\dot{\mathrm{q}}) \tag{86}$$

F_v represents the coefficient matrix of viscous friction, and F_C of Coulomb friction. Since friction is a local effect, $F(\dot{\mathrm{q}})$ will be uncoupled among the joints. Then viscous friction is of the form

$$F_v|\dot{\mathrm{q}}| = \begin{bmatrix} v_1|\dot{q}_1| & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & v_n|\dot{q}_n| \end{bmatrix} \tag{87}$$

with v_i known constant coefficients. Then the Coulomb friction is of the form

$$F_C = \begin{bmatrix} k_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & k_n \end{bmatrix} \tag{88}$$

with known constant coefficients. For more discussion, see [2]

7.1 Friction Compensation by Adding Friction Terms

To compensate the robot for the friction a very simple method is to increase the input torque by the estimated amount of friction. Adding the Coulomb friction to the model (1) results in

$$M(\mathrm{q})\ddot{\mathrm{q}} + C(\mathrm{q}, \dot{\mathrm{q}})\dot{\mathrm{q}} + F(\dot{\mathrm{q}}) = \tau + \hat{F}_C\mathrm{sgn}(\dot{\mathrm{q}}) \tag{89}$$

where $\hat{F}_C$ is our estimate for F_C.

For a two-link robot like the D2R2 $\hat{F}_C$ will be defined as

$$\hat{F}_C = \begin{bmatrix} \hat{k}_1 & 0 \\ 0 & \hat{k}_2 \end{bmatrix} \tag{90}$$

not restricted in terms of memory a full-dimensional order design was used. Starting from the Euler-Lagrange dynamics (1) for the D2R2 without gravity

$$M(\mathrm{q})\ddot{\mathrm{q}} + C(\dot{\mathrm{q}}, \ddot{\mathrm{q}})\dot{\mathrm{q}} = \tau \tag{74}$$

we derive

$$\begin{aligned} \dot{\mathrm{x}}_1 &= \mathrm{x}_2 \qquad (75) \\ \dot{\mathrm{x}}_2 &= -M^{-1}(\mathrm{x}_1)C(\mathrm{x}_1, \mathrm{x}_2)\mathrm{x}_2 + M^{-1}(\mathrm{x}_1)\tau \qquad (76) \end{aligned}$$

in state space. x_1 and x_2 represent the joint angle and joint velocity respectively. Then

$$\dot{\mathrm{x}} = A\mathrm{x} + M^{-1}(\mathrm{x}_1) + f(\mathrm{x}) \tag{77}$$

where A represents the linear part and $M^{-1}(\mathrm{x}_1)\tau + f(\mathrm{x})$ the nonlinear part. Further

$$\mathrm{y} = G\mathrm{x} = \mathrm{x}_1 \tag{78}$$

which postulates that only the joint positions are observable. A and G are then

$$A = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \tag{79}$$

and

$$G = \begin{bmatrix} I & 0 \end{bmatrix} \tag{80}$$

with I the two-by-two identity matrix. Assuming exact knowledge of the uncertainty in M we can define the observer as

$$\dot{\hat{\mathrm{x}}} = A\hat{\mathrm{x}} + L(\mathrm{y} - G\hat{\mathrm{x}}) + M^{-1}(\mathrm{y})\tau \tag{81}$$

and

$$\mathbf{e} = \mathrm{x} - \hat{\mathrm{x}}. \tag{82}$$

(For the actual design we will chose the robot parameters for M lying in the middle of the uncertainty range.) Note that

$$\dot{\mathbf{e}} = (A - LG)\mathbf{e} + \begin{bmatrix} 0 \\ M^{-1}(\mathrm{x}_1)C(\mathrm{x}_1, \mathrm{x}_2)\mathrm{x}_2 \end{bmatrix} \tag{83}$$

and therefore $e \not\to 0$ as desired. For the pole placement of the linear part we determine the eigenvalues of $(A - LG)$. They are

$$\lambda_{1/2} = -\tfrac{L_1}{2} \pm \sqrt{\tfrac{L_1^2}{4} - L_2}. \tag{84}$$

for both joints. Using the Euler approximation to convert our observer system into discrete time since it is not linear anymore, leads to

$$\hat{\mathrm{x}}_{k+1} = \mathrm{x}_k + (A\hat{\mathrm{x}}_k + L(\mathrm{y}_k - G\hat{\mathrm{x}_\mathrm{k}}) + M^{-1}(\mathrm{x}_1)\tau)T \tag{85}$$

6.1 Kalman Filter

A more sophisticated method to estimate the velocities is by using an optimized Kalman filter for the following system model as in [3]

$$\dot{\mathrm{x}}(t) = A\mathrm{x}(t) + \Gamma w(t) \tag{68}$$

$$q(t) = C\mathrm{x}(t) + e(t) \tag{69}$$

with x_1 and x_2 representing joint angle and joint velocity, respectively. w represent white Gaussian noise with covariance c. The white noise $w(t)$ is a surrogate for $\ddot{q}(t)$; the more wide-band $\ddot{q}$ is, the better is the approximation of the deterministic robotics model by the stochastic model. Since robot motion is not well characterized by such a stationary random process, c may be regarded as a filter parameter to be adjusted. $e(t)$ represents the quantization error, assumed white with zero mean, and a variance R defined as:

$$R = \frac{\theta_m^2}{3} \tag{70}$$

where $\theta_m = \frac{1}{\text{encoder resolution / rad}} = 9.58740 \cdot 10^{-6}$rad is the interpulse angle for the D2R2. The matrices A, C, and Γ using simply the linear part of the D2R2 are given as:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{71}$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{72}$$

$$\Gamma = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{73}$$

The matrices were transferred into a discrete time system with Matlab's `c2d` function. The Kalman gain matrix was estimated with the Matlab's `dlqe` function, and the discrete Kalman filter was designed using Matlab's `destim` function. For a sampling rate of 2500 Hz a covariance c of about 1000 achieved the best results for the filter. The results were as good as using (64), depending on trajectory and controller. For the velocity error estimation the same filter design was used.

6.2 Nonlinear Velocity Observer

The Kalman filter is a linear state observer. Its gains are optimized for the assumed noise. Another possible solution to our problem would be to design a state observer, which includes the nonlinearities of the robot. Linear state observer using inverse dynamics control to avoid the nonlinearities have already been designed [1]. The problem with our robust controllers is that we cannot linearize the system exactly due to the uncertainty of the robot parameters. For the state observer we can chose between full-dimensional order and reduced-dimensional order design; since we are is commonly used to compute joint velocity estimates v_k [6]. The filter parameter ν is a design parameter. If ν is small, it corresponds to a fast pole near $z = 0$, which provides some low-pass filtering to reject unwanted sensor noise. The velocity estimation filter design can be optimized for the given encoder noise statistics to reconstruct v_k. For the estimation of the velocity error $\dot{e}_k = q_k - q_k^d$ one may use the filter [6]

$$\dot{e}_k = \nu \dot{e}_{k-1} + \frac{\tilde{q}_k - \tilde{q}_{k-1}}{T} \tag{62}$$

with $\tilde{q}_k = q_k - q_k^d$. This equals

$$\dot{e}_k = \nu \dot{e}_{k-1} + \frac{q_k - q_{k-1}}{T} - \frac{q_k^d - q_{k-1}^d}{T}. \tag{63}$$

In other words the desired velocity for the velocity error is calculated using the same filter to avoid a phase shift between the desired and the actual velocity which would lead to oscillation in the control. However with the high sampling rate that we are using in our controllers, the above filters were found to be inadequate and we encountered severe resonance problems from all of the control algorithms (and even with a simple PD controller). For this reason, more sophisticated filters had to be designed to reconstruct the velocity from the encoder measurements.

The trade-off lies between noise reduction and using the most recent values for the control. One solution for the D2R2 is to simply average the last n velocity estimates with $\dot{q}_k$ from (60):

$$v_k = \frac{1}{nT} \sum_{i=0}^{n-1} \dot{q}_{k-i} = \frac{1}{n}(q_k - q_{k-n}). \tag{64}$$

For the D2R2 $n = 3$ was found to remove the noise best (at a sampling rate of 2500 Hz). Equation (64) is comparable to using a lower sample rate for the velocity estimation to avoid the velocity degradation which comes with such a high sampling frequency. For the velocity errors it turned out that

$$\dot{e}_k = \frac{1}{n} \sum_{i=0}^{n-1} \frac{\tilde{q}_k - \tilde{q}_{k-1}}{nT} = \frac{1}{nT}(q_k - q_{k-n}) - \frac{1}{nT}(q_k^d - q_{k-n}^d) \tag{65}$$

which only compensated for the phase shift between desired an actual velocity was not working sufficiently. Therefore the filter

$$\dot{e}_k = \frac{1}{n} \sum_{i=1}^{n-1} \dot{e}_{k-i} + \frac{q_k - q_{k-1}}{T} - \frac{q_k^d - q_{k-1}^d}{T} \tag{66}$$

had to be used. This combination eliminates high frequency noise in a very simple way. Of course, for the velocity estimation also

$$v_k = \frac{1}{n} \sum_{i=1}^{n-1} v_{k-i} + \frac{q_k - q_{k-1}}{T} \tag{67}$$

can be used. However the tracking errors seem to be slightly larger with this velocity filter than with (64). The choice $n = 3$ achieved the best results in this case also.

5 Sampling Rate and Velocity Estimates

The controllers are designed in continuous time and the theoretical proofs of convergence are valid only in continuous time. In continuous time the tracking errors generally decrease as gains are increased. This is, in fact, another consequence of the passivity property of rigid robots. Therefore it is desirable to chose the gains as high as possible in order to improve accuracy. On the other hand high gains require a high sampling rate to guarantee that the approximation of the continuous time controller by the discrete time controller stays feasible. Choosing the sampling rate too low for the selected gains results in chattering of the input torques. Since the input torques are limited due to the characteristics of the motors the chattering might become visible in the joint positions, which increases the chattering in the velocity to an even higher extend. Since the velocity is used in the control law this feedback will cause the robot to vibrate. Furthermore these vibrations might resonate the system, and the performance will deteriorate if gains are chosen too high. Because of the speed of our DSP development system, we were able to run all control experiments using a sample rate of 2500Hz. The gains were empirically designed accordingly for the given sample rate.

Note that all of the algorithms depend on one or more parameters ϵ_i. It can be shown that increasing these ϵ–values will also decrease the chattering. The control laws are discontinuous at $\epsilon_i = 0$, so for any choice of gains and sampling period the control will chatter for small ϵ. However increasing ϵ too high will reduce accuracy, since these parameters determine the size of the ultimate boundedness region in state space. In other words, increasing ϵ has the effect of detuning the system to reject, unmodeled dynamics, noise and disturbances. This suggest that there exists an optimal value for ϵ. Below this value the chattering will resonate the system and decrease accuracy, above this value we are detuning the system more than necessary which results in larger tracking errors. Our approach was to empirically determine the smallest ϵ–values and largest gains, K, Λ, possible for the given trajectories.

6 Velocity Estimation

The robust control algorithms in this paper assume that both the joint positions q and joint velocities $\dot{q}$ are measured exactly. However, as is commonly the case, only the joint positions are available from optical encoders, and the joint velocities must be estimated from these position measurements [6]. Generally, one finds that simply computing the joint velocities using the Euler approximation

$$\dot{q}_k = \frac{q_k - q_{k-1}}{T} \tag{60}$$

does not work, especially as the sampling rate increases [3], due to the encoder measurement noise. For this reason, several different schemes to estimate the velocity were analyzed and tested. In general, a filtered derivative of the form

$$v_k = \nu v_{k-1} + \frac{q_k - q_{k-1}}{T}, \tag{61}$$

In order to quantitatively compare the performance of the various algorithms we used several error measures. We computed both the L_2 and L_∞ norms of the tracking error of each joint in joint space and the L_2 and L_∞ norm of the end–effector tracking error in task space. These errors are presented in table form in subsequent sections along with plots of the errors themselves and of the torque inputs.

Recall that the L_2 norm of the tracking error of joint n is

$$|e_n| := \sqrt{\int_{t_0}^{t_f} (q_n - q_n^d)^2 dt} \tag{52}$$

where $n \in \{1,2\}$ is the joint number. Since data is only send back at discrete time intervals, we discretize equation (52) as:

$$|e_n| \;:=\; \sqrt{\sum_{i=1}^{k} (q_n - q_n^d)^2 \Delta T} \tag{53}$$

$$= \sqrt{\Delta T}\sqrt{\sum_{i=1}^{k} (q_n - q_n^d)^2} \tag{54}$$

Because $\sqrt{\Delta T}$ is constant we will include it on the left side which yields our definition of the joint space error criteria:

$$|error_n| := \frac{|e_n|}{\sqrt{\Delta T}} = \sqrt{\sum_{i=1}^{k} (q_n - q_n^d)^2} \tag{55}$$

The maximal error in joint space is

$$maxerror_n = \max|q_n - q_n^d| \tag{56}$$

The L_2 norm of the joint error vector is given by

$$|e| := \sqrt{\int_{t_0}^{t_f} (q_1 - q_1^d)^2 + (q_2 - q_2^d)^2 \; dt} \tag{57}$$

However if we are looking at the desired “end-effector” position we should weight the error of each joint accordingly. Therefore a better single criterion is the error in task space. We define the task space error as the distance between the actual and the desired “end-effector” position. Analogous to (55) we define

$$|error_{\text{task space}}| := \sqrt{\sum_{i=1}^{k} \left(\left\| \begin{pmatrix} x - x^d \\ y - y^d \end{pmatrix} \right\|_2 \right)^2} \tag{58}$$

where $\|\cdot\|_2$ is the Euclidean norm. The maximum error in task space is

$$maxerror_{\text{task space}} := \max \left\| \begin{pmatrix} x - x^d \\ y - y^d \end{pmatrix} \right\|_2 . \tag{59}$$

For our experimental comparison we will present the measures (55), (56), (58), and (59).

where Δ represents the difference between the maximum load and no load cases. These uncertainty bounds may depend on trigonometric functions of the generalized coordinates or they can be further simplified by using bound of these trigonometric functions. We note that Liu and Goldenberg [7] define the nominal values of $\boldsymbol{\theta}_{M_0}$ and $\boldsymbol{\theta}_{C_0}$ by using the unloaded arm, i.e. at the lower end of the uncertainty interval. We found that the performance of the algorithm on the actual manipulator with load was poor with these nominal parameters. For this reason, in this paper we will instead define the nominal values in the center of the uncertainty interval as in the algorithms (SP1) and (SP2), as doing so greatly improves the tracking performance.

Comparing this algorithm with the algorithms (SP1) and (SP2) of Spong, we see that it requires more on-line computation. However the principle of the control law is very similar to SP1 or SP2 and in fact mainly just a different parameterization of the Euler-Lagrange dynamics, using an extended definition of the parameter vector $\boldsymbol{\theta}$.

4 The Reference Trajectories and Error Criteria

In order to compare the performance of the controllers on D2R2 we picked three joint space trajectories and one task space trajectory. The joint space trajectories were:

1. The output of a linear second order critically damped reference model.

$$\ddot{\mathbf{q}}^d + 2\zeta\omega\dot{\mathbf{q}}^d + \omega^2\mathbf{q}^d = \omega^2\mathbf{v} \tag{47}$$

   where $\mathbf{v}$ was a step change from zero to ninety degrees for each joint.

2. A cubic polynomial trajectory (CPT)

$$\mathbf{q}^d = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{48}$$

   where the desired motion is from zero to ninety degrees and back to zero degrees over two seconds.

3. A sinusoid

$$\mathbf{q}^d = a\sin(\omega t) \tag{49}$$

The task space trajectory was a circle of radius r located at coordinates (a, b) in the robot's workspace, i.e.,

$$x^d = a + r\cos(\omega t) \tag{50}$$

$$y^d = b + r\sin(\omega t) \tag{51}$$

The inverse kinematics of the robot were then used to translate this task space trajectory into a joint space trajectory. The reader is referred to [5] for a detailed description of these trajectories.

with the additional control inputs $\mathbf{u}_M$ and $\mathbf{u}_C$, defined as follows, using the same technique as in [11]:

$$\mathbf{u}_M = \begin{cases} -\rho_M \frac{Y_M^T(\mathbf{a})\mathbf{r}}{\|Y_M^T(\mathbf{a})\mathbf{r}\|} & \text{if} \quad \left\|Y_M^T(\mathbf{a})\mathbf{r}\right\| > \epsilon_M \\ \frac{-\rho_M}{\epsilon_M} Y_M^T(\mathbf{a})\mathbf{r} & \text{if} \quad \left\|Y_M^T(\mathbf{a})\mathbf{r}\right\| \leq \epsilon_M \end{cases} \tag{38}$$

(LG)

$$\mathbf{u}_C = \begin{cases} -\rho_C \frac{Y_C^T(\dot{\mathbf{q}},\mathbf{v})\mathbf{r}}{\|Y_C^T(\dot{\mathbf{q}},\mathbf{v})\mathbf{r}\|} & \text{if} \quad \left\|Y_C^T(\dot{\mathbf{q}},\mathbf{v})\mathbf{r}\right\| > \epsilon_C \\ \frac{-\rho_C}{\epsilon_C} Y_C^T(\dot{\mathbf{q}},\mathbf{v})\mathbf{r} & \text{if} \quad \left\|Y_C^T(\dot{\mathbf{q}},\mathbf{v})\mathbf{r}\right\| \leq \epsilon_C \end{cases} \tag{39}$$

where ϵ_M, ϵ_C, and are positive constants. It is proved in [7] that the tracking error is uniformly ultimately bounded using the same Lyapunov function (20) as in [11]. For the two-link robot, D2R2, the various quantities are given by

$$Y_M(\mathbf{a}) = \begin{bmatrix} a_1 & a_2 & 0 \\ 0 & a_1 & a_2 \end{bmatrix} \tag{40}$$

$$\boldsymbol{\theta}_M(\mathbf{q}) = \begin{pmatrix} d_{11} \\ d_{12} \\ d_{22} \end{pmatrix} \tag{41}$$

and d_{11},d_{12}, and d_{22} as in (3)

$$Y_C(\dot{\mathbf{q}},\mathbf{v}) = \begin{bmatrix} \dot{q}_2 v_1 + (\dot{q}_1 + \dot{q}_2) v_2 \\ -\dot{q}_1 v_1 \end{bmatrix} \tag{42}$$

$$\theta_C(\mathbf{q}) = h \tag{43}$$

with h as in (5). Recall that $\boldsymbol{\theta}_M$ and $\boldsymbol{\theta}_C$ are not necessarily constant vectors in this formulation. The uncertainty bounds, ρ_M and ρ_C, are obtained as:

$$\rho_M = \sqrt{\rho_{M_1}^2 + \rho_{M_2}^2 + \rho_{M_3}^2} \tag{44}$$

$$\rho_C = \rho_h \tag{45}$$

with

$$\begin{aligned} \rho_{M_1} &\geq \Delta d_{11}(q_2) \\ \rho_{M_2} &\geq \Delta d_{12}(q_2) \\ \rho_{M_3} &\geq \Delta d_{22} \\ \rho_h &\geq \Delta h(q_2) \end{aligned} \tag{46}$$

with $y := C(\mathbf{q}, \dot{\mathbf{q}})\mathbf{v}$ and $y_i = \dot{\mathbf{q}}^T L_i(\mathbf{q})\mathbf{v}$ where $L_i(\mathbf{q})$ is a square matrix.

The proof of uniform ultimate boundedness utilizes the Lyapunov function

$$V = \frac{1}{2}\mathbf{r}^T M(\mathbf{q})\mathbf{r} + \dot{\tilde{\mathbf{q}}}^T \dot{\tilde{\mathbf{q}}} \tag{29}$$

Comparing this algorithm with the controllers SP1 and SP2, we see that ZC requires less on-line computation because no regressor matrix is computed. However, the calculation of the uncertainty bounds is more complicated.

3.3 The Algorithm LG

Liu and Goldenberg [7] propose a robust controller that does not linearly parameterize the Lagrangian as a whole, but, instead, parameterizes the inertia matrix M and the Coriolis and centrifugal matrix C as

$$M(\mathbf{q})\ddot{\mathbf{q}} = Y_M(\ddot{\mathbf{q}})\boldsymbol{\theta}_M(\mathbf{q}) \tag{30}$$

and

$$C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = Y_C(\dot{\mathbf{q}})\boldsymbol{\theta}_C(\mathbf{q}). \tag{31}$$

Thus, the uncertainty is represented by the vectors $\boldsymbol{\theta}_M$ and $\boldsymbol{\theta}_C$, which are no longer constant, but are, in general, functions of the configuration $\mathbf{q}$. These uncertainty terms are assumed to be bounded as follows:

$$||\tilde{\boldsymbol{\theta}}_M(\mathbf{q})|| := ||\boldsymbol{\theta}_M(\mathbf{q}) - \boldsymbol{\theta}_{M_0}(\mathbf{q})|| \leq \rho_M(\mathbf{q}) \tag{32}$$

$$||\tilde{\boldsymbol{\theta}}_C(\mathbf{q})|| := ||\boldsymbol{\theta}_C(\mathbf{q}) - \boldsymbol{\theta}_{C_0}(\mathbf{q})|| \leq \rho_C(\mathbf{q}) \tag{33}$$

where $\boldsymbol{\theta}_{M_0}(\mathbf{q}$ and $\boldsymbol{\theta}_{C_0}(\mathbf{q})$ represent the nominal values. The nominal control vector is then defined as in [11]

$$\begin{aligned} \boldsymbol{\tau}_0 &= M_0(\mathbf{q})\mathbf{a} + C_0(\mathbf{q}, \dot{\mathbf{q}})\mathbf{v} - K\mathbf{r} && (34) \\ &= Y_M(\mathbf{a})\boldsymbol{\theta}_{M_0}(\mathbf{q}) + Y_C(\dot{\mathbf{q}}, \mathbf{v})\boldsymbol{\theta}_{C_0}(\mathbf{q}) - K\mathbf{r} && (35) \end{aligned}$$

with the quantities $\mathbf{r}$, $\mathbf{v}$, and $\mathbf{a}$ are defined as in (11). The control input $\boldsymbol{\tau}$ is then defined as

$$\begin{aligned} \boldsymbol{\tau} &= \boldsymbol{\tau}_0 + Y_M(\mathbf{a})\mathbf{u}_M + Y_C(\dot{\mathbf{q}}, \mathbf{v})\mathbf{u}_C && (36) \\ &= Y_M(\mathbf{a})(\boldsymbol{\theta}_{M_0}(\mathbf{q}) + \mathbf{u}_M) + Y_C(\dot{\mathbf{q}}, \mathbf{v})(\boldsymbol{\theta}_{C_0}(\mathbf{q}) + \mathbf{u}_C) - K\mathbf{r} \end{aligned}$$

where the vectors $\mathbf{u}_M$ and $\mathbf{u}_C$ are designed to achieve robustness to the parametric uncertainty. Substituting the control law (36) in (1) they obtain after some algebra

$$M(\mathbf{q})\dot{\mathbf{r}} + C(\mathbf{q}, \dot{\mathbf{q}})\mathbf{r} + K\mathbf{r} = Y_M(\mathbf{a})(\tilde{\boldsymbol{\theta}}_M + \mathbf{u}_M) + Y_C(\dot{\mathbf{q}}, \mathbf{v})(\tilde{\boldsymbol{\theta}}_C + \mathbf{u}_C) \tag{37}$$

where ξ_i denotes the i^{th} component of the vector $Y^T\mathbf{r}$ and ϵ_i are positive constants.

With either of these additional control inputs, (SP1) or (SP2), the control law (12) is continuous and it can be shown using the Lyapunov function

$$V = \frac{1}{2}\mathbf{r}^T M(\mathbf{q})\mathbf{r} + \dot{\tilde{\mathbf{q}}}^T \Lambda^T K \dot{\tilde{\mathbf{q}}} \tag{20}$$

that the closed loop systems are uniformly ultimately bounded (u.u.b.) as defined in [4]. See [11] for the proof.

3.2 The Algorithm ZC

The algorithm, ZC, of Zenieh and Corless [15] does not exploit the linear parametrization property. Uncertainty in the physical parameters is estimated by the bounds β_0, β_1, and β_2 shown below. These inequalities hold for all $\mathbf{q}$, $\dot{\mathbf{q}}$, and all uncertainty δ, which is called the *lumped uncertainty term*,

$$\lambda_{\min}[M(\mathbf{q})] \geq \beta_0 > 0 \tag{21}$$

$$\lambda_{\max}[M(\mathbf{q})] \leq \beta_1 \tag{22}$$

$$\|C(\mathbf{q}, \dot{\mathbf{q}})\| \leq \beta_2 \|\dot{\mathbf{q}}\| \tag{23}$$

where $\lambda_{min}[\cdot]$ and $\lambda_{max}[\cdot]$ denote, respectively, the minimum and maximum eigenvalues of a matrix. Zenieh and Corless propose an *s-α tracking controller* with the rate of convergence $\alpha > 0$ and a prespecified tolerance $s > 0$, which guarantees that the robot trajectory exponentially converges to any desired trajectory with the rate α and to within the tolerance s. Choosing a positive definite symmetric diagonal gain matrix Λ satisfying

$$\lambda_{\min}[\Lambda] \geq \alpha \tag{24}$$

with α the desired convergence rate and $\mathbf{r}$, $\mathbf{v}$, and $\tilde{\mathbf{q}}$ defined as in (11), the controller is given by

$$\begin{aligned} \tau &= -K\mathbf{r} - \tfrac{\rho^2 \mathbf{r}}{\|\rho \mathbf{r}\| + \epsilon} \\ \rho &:= \beta_1 \|\dot{\mathbf{v}}\| + \beta_2 \|\mathbf{v}\| \|\dot{\mathbf{q}}\| \end{aligned} \qquad \text{(ZC)} \tag{25}$$

where K is any positive-definite, symmetric, matrix which satisfies

$$\lambda_{\min}[K] > \alpha\beta_1. \tag{26}$$

ϵ is any positive scalar satisfying

$$\epsilon \leq (\alpha s)^2 \lambda_{min}[K] \frac{\beta_0}{\beta_1} \tag{27}$$

The bound β_2 may be computed using

$$\beta_2 \geq \sqrt{\sum_{i=1}^{n} \|L_i(\mathbf{q})\|^2} \quad \forall \mathbf{q}, \delta \tag{28}$$

control vector τ_0 as

$$\tau = \tau_0 + Y(\mathbf{q},\dot{\mathbf{q}},\mathbf{v},\mathbf{a})\mathbf{u} = Y(\mathbf{q},\dot{\mathbf{q}},\mathbf{v},\mathbf{a})(\boldsymbol{\theta}_0 + \mathbf{u}) - K\mathbf{r} \tag{12}$$

where the vector $\mathbf{u} \in R^p$ is an additional control input designed to provide robustness to the parameter uncertainty. Substituting (12) into (7) yields, after some algebra,

$$M(\mathbf{q})\dot{\mathbf{r}} + C(\mathbf{q},\dot{\mathbf{q}})\mathbf{r} + K\mathbf{r} = Y(\mathbf{q},\dot{\mathbf{q}},\mathbf{v},\mathbf{a})(\tilde{\boldsymbol{\theta}} + \mathbf{u}) \tag{13}$$

For the parameterization $\boldsymbol{\theta}$ of D2R2 we define

$$\boldsymbol{\theta} = \begin{pmatrix} m_1 l_{c1}^2 + m_2 l_1^2 + I_1 \\ m_2 l_{c2}^2 + I_2 \\ m_2 l_1 l_{c2} \end{pmatrix} \tag{14}$$

which leads to the regressor matrix,

$$Y(\mathbf{q},\dot{\mathbf{q}},\mathbf{v},\mathbf{a}) = \begin{bmatrix} a_1 & a_1 + a_2 & y_{13} \\ 0 & a_1 + a_2 & y_{23} \end{bmatrix} \tag{15}$$

where

$$\begin{aligned} y_{13} &= \cos(q_2)(2a_1 + a_2) - \sin(q_2)(\dot{q}_1 v_2 + \dot{q}_2(v_1 + v_2)) \\ y_{23} &= \cos(q_2)a_1 + \sin(q_2)\dot{q}_1 v_1 \end{aligned}$$

For the nominal parameter vector $\boldsymbol{\theta}_0$ we chose the mean value of $\boldsymbol{\theta}$ for the unloaded arm and the arm carrying a maximum load, i.e.

$$\boldsymbol{\theta}_0 = \frac{1}{2}(\boldsymbol{\theta}_{\text{maximum load}} - \boldsymbol{\theta}_{\text{unloaded}}) \tag{16}$$

The algorithm SP1 calculates the additional control input $\mathbf{u}$ in (12) based on a single measure of the uncertainty, ρ, where

$$\rho = (\sum_{i=1}^{p} \rho_i)^{\frac{1}{2}} \tag{17}$$

as follows:

$$\mathbf{u} = \begin{cases} -\rho \frac{Y^T\mathbf{r}}{\|Y^T\mathbf{r}\|} & \text{if} \quad \left\|Y^T\mathbf{r}\right\| > \epsilon \\ \frac{-\rho}{\epsilon} Y^T\mathbf{r} & \text{if} \quad \left\|Y^T\mathbf{r}\right\| \le \epsilon \end{cases} \qquad \text{(SP1)} \tag{18}$$

and $\epsilon > 0$.

The algorithm SP2 assigns different "weights" or gains to the component of $\mathbf{u}$ by defining the i^{th} component of the control input u as

$$u_i = \begin{cases} -\rho_i \frac{\xi_i}{|\xi_i|} & \text{if} \quad |\xi_i| > \epsilon_i \\ \frac{-\rho_i}{\epsilon_i} \xi_i & \text{if} \quad |\xi_i| \le \epsilon_i \end{cases} \qquad \text{(SP2)} \tag{19}$$

2. (**Linear Parametrizability**) There exist a vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)^T \in R^p$, of link parameters, and a matrix $Y(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$, called the regressor, such that

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = Y(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\boldsymbol{\theta} = \boldsymbol{\tau} \tag{7}$$

3 The Algorithms and Trajectories

In this section we describe the four robust control algorithms considered in this paper. The first two algorithms were derived by Spong in [11]. The third algorithm was derived by Zenieh and Corless in [15]. The fourth algorithm is due to Liu and Goldenberg and appeared in [7]. The four algorithms will be designated SP1, SP2, ZC, and LG, respectively, for conciseness. All of these algorithms exploit the skew–symmetry property of the robot dynamics and guarantee uniform ultimate boundedness of the tracking error provided bounds on the uncertainty are given a priori as described in, for example, [4]. In addition, the algorithms SP1 and SP2 exploit the linear parametrizability property. We detail the specifics of these algorithms below. The reader is referred to the original references for the proofs of stability and convergence which all use similar Lyapunov arguments.

3.1 The Algorithms SP1 and SP2

Both of the algorithms SP1 and SP2 from [11] exploit the linear parameterizability of robot dynamics given by equation (7). It is assumed only that the parameter vector $\boldsymbol{\theta}$ is "uncertain" which means that there exists a nominal parameter vector $\boldsymbol{\theta}_o = (\theta_{10}, \ldots \theta_{p0})^T \in R^p$ and nonnegative constants $\rho_i \in R_+$ such that, for $i = 1, \ldots, p$,

$$\left\|\tilde{\boldsymbol{\theta}}_i\right\| := \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_{i0}\| \leq \rho_i. \tag{8}$$

Define a nominal control vector $\boldsymbol{\tau}_0$ as

$$\begin{aligned} \boldsymbol{\tau}_0 &= M_0(\mathbf{q})\mathbf{a} + C_0(\mathbf{q}, \dot{\mathbf{q}})\mathbf{v} - K\mathbf{r} & (9) \\ &= Y(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{v}, \mathbf{a})\boldsymbol{\theta}_0 - K\mathbf{r} & (10) \end{aligned}$$

where the quantities $\mathbf{r}$, $\mathbf{v}$, and $\mathbf{a}$ are given by

$$\begin{aligned} \mathbf{r} &= \dot{\tilde{\mathbf{q}}} + \Lambda\tilde{\mathbf{q}} \\ \mathbf{v} &= \dot{\mathbf{q}}^d - \Lambda\tilde{\mathbf{q}} \\ \mathbf{a} &= \dot{\mathbf{v}} \\ \tilde{\mathbf{q}} &= \mathbf{q} - \mathbf{q}^d & (11) \end{aligned}$$

and $\mathbf{q}^d$ is a given twice continuously differentiable reference trajectory. The gain matrices K and Λ are positive definite diagonal matrices. The control input $\boldsymbol{\tau}$ is defined in terms of the nominal the vector of applied joint torques. Refering to Figure (2), the terms in the dynamic equations are given by

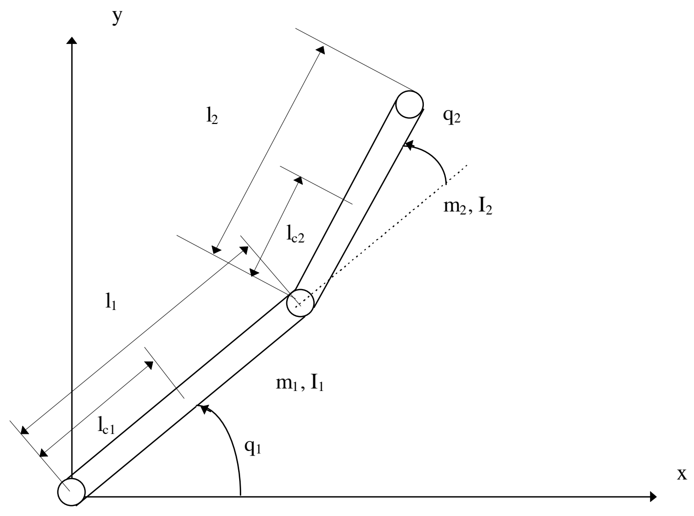


Figure 2: Schematic diagram of the two-link direct-drive D2R2

$$M(\mathbf{q}) = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix} \tag{2}$$

with

$$\begin{aligned} d_{11} &= m_1 l_{c1}^2 + m_2(l_1^2 + l_{c2}^2 + 2l_1 l_{c2}\cos q_2) + I_1 + I_2, \\ d_{12} &= d_{21} = m_2(l_{c2}^2 + l_1 l_{c2}\cos q_2) + I_2, \\ d_{22} &= m_2 l_{c2}^2 + I_2. \end{aligned} \tag{3}$$

$$C(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} h\dot{q}_2 & h(\dot{q}_1 + \dot{q}_2) \\ -h\dot{q}_1 & 0 \end{bmatrix} \tag{4}$$

with

$$h = -m_2 l_1 l_{c2} \sin q_2. \tag{5}$$

The link parameters for the D2R2 are given in the appendix.

We recall the fundamental properties of skew symmetry and linear parametrizability of these equation. See [9] for details:

1. (**Skew Symmetry**) The matrix

$$N(\mathbf{q}, \dot{\mathbf{q}}) = \dot{M}(\mathbf{q}) - 2C(\mathbf{q}, \dot{\mathbf{q}}) \tag{6}$$

is skew symmetric. This implies that the robot dynamics define a passive mapping between joint torque and joint velocity.

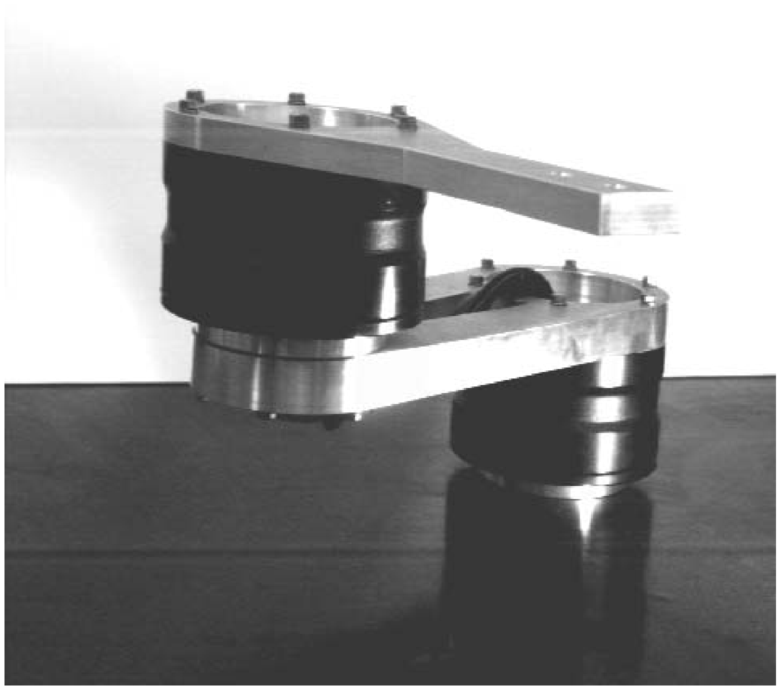


Figure 1: The two-link planar arm "D2R2"

control algorithms themselves are implemented on an MX-31 DSP development system from Integrated Motions, Inc., (IMI), which utilizes a Texas Instruments TMS320C31 32–bit floating point processor with 60 ns single–cycle instruction execution time. Control Programs are written in C, compiled on a Dell PC486 workstation, and loaded into the MX-31 through a serial communication link. IMI provides a library of support functions for tasks such as encoder reading and serial communication port service. Data are sent back to the PC during robot operation through a second serial communication link and stored in the PC for analysis in Matlab.

2 Robot Dynamics

Since the motion of the SCARA–like manipulator is in the horizontal plane, the gravitational torques about the joints are identically zero, which simplifies the derivation of both the dynamic equations of motion and the control input. The standard Euler-Lagrange dynamic equations for this system may therefore be written as [13]

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = \tau \quad \text{with} \quad \mathbf{q} \in R^n, \tau \in R^n. \tag{1}$$

where $M(\mathbf{q}) \in R^{n \times n}$ is the inertia matrix, and $\mathbf{q}, \dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$ are the joint angles, velocities, and accelerations, respectively. The vector $C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ represents centrifugal and Coriolis terms, and τ is

An Experimental Comparison of Robust Control Algorithms On a Direct Drive Manipulator [*]

André Jaritz
Mark W. Spong
Coordinated Science Laboratory
University of Illinois at Urbana–Champaign
1308 W. Main St., Urbana, Ill. 61801
m-spong@uiuc.edu

Abstract

In this paper we present an experimental comparison of recent passivity based robust control algorithms on a two link direct drive robot arm. The manipulator is actuated with high torque brushless DC motors and is controlled by a DSP development system interfaced to a PC486 workstation. Four algorithms are compared with respect to ease of design, implementation, and performance of the closed loop systems.

1 Introduction

In this paper we summarize the results of a systematic comparison of passivity based robust control algorithms on the two–link direct drive robot manipulator shown below in Figure (1). Four algorithms are compared with respect to ease of design, implementation, and performance of the closed–loop system. The reader is referred to the report [5] for the complete study, which also includes data for additional algorithms beyond those presented here.

The manipulator used in the study, known as D2R2, was constructed with Compumotor Model DM1015-B brushless DC motors, controlled via transformerless drivers operating in torque mode. Maximum torque is produced at ± 8 V from the higher-level controller which corresponds to a maximum torque command of ± 31.2 Nm [10]. Integrated optical encoders provide 655360 lines of resolution [10]. The links were designed in AutoCAD and constructed from aluminum [14]. The

[*]This research is partially supported by the National Science Foundation under grants CMS-9402229 and IRI-9216428