In: Biological Cybernetics vol. 75 nr. 5, pp. 433-440

Analysis and control of a rubbertuator arm

P. van der Smagt[123], F. Groen[2], and K. Schulten[3]

July 2, 1996

Abstract

The control of light-weight compliant robot arms is cumbersome due to the fact that its Coriolis forces are large, and the forces exerted by the relatively weak actuators may change in time due to external (e.g., temperature) influences.

We describe and analyse the behaviour of a light-weight robot arm, the *SoftArm* robot. It is found that the hysteretic force-position relationship of the arm can be explained from its structure. This knowledge is used in the construction of a neural-network based controller. Experiments show that the network is able to accurately control the robot arm after a training session of only a few minutes.

When a robot system is designed, the focus generally is a design such that friction, gravity, and payloads can be practically neglected. Therefore, robots are built extremely stiff (i.e., non-compliant) and are equipped with joint actuators which are strong enough to overcome threshold friction, position-dependent gravity, and payloads. The merit of such an approach is that relatively simple control algorithms can be used to position the end-effector with high accuracy. However, apart from the high cost of such robot systems and their high energy consumption because of their heavy construction, their large strength makes their use in environments where humans operate, such as hospitals and household environments, too dangerous.

Therefore the search for simpler, more compliant robot systems is of importance. One of such systems is the *SoftArm* robot. The pneumatically driven actuators of this robot consist of *rubbertuators*, which are modelled after skeletal muscle systems. The rubbertuators have a high force-to-weight

[1]German Aerospace Research Establishment, Dept. of Robotics and System Dynamics, P. O. Box 1116, D–82230 Wessling, GERMANY, phone: +49 8153 281152, fax: +49 8153 281134, email: `smagt@dlr.de`

[2]Department of Computer Systems, University of Amsterdam, Kruislaan 403, NL–1098 SJ Amsterdam, NETHERLANDS

[3]Department of Theoretical Biophysics, Beckman Institute, University of Illinois, Urbana, IL 61801, USA

ratio and are very compliant, such that the robot is safe for operation in direct contact with human operators.

Yet control of such a system is a difficult problem. Naturally, coarse positional control can be obtained with simple feedback algorithms. As is shown in this paper, it is possible to use a standard PID controller in a feedback loop to control the joint values of the robot towards their desired values. The resulting precision, however, is rather bad; the desired trajectory is only coarsely followed (lagging and hysteresis problems), and the error in joint position is up to $10°$. The algorithms that can be used for controlling industrial non-compliant robots are not usable to control compliant robots due to their complex, highly nonlinear dynamics.

It has been shown that neural networks can be well applied to robot control. But how do these algorithms behave when applied to robots where self-imposed problems such as changing kinematics are no academic assumption anymore, but are reality? In this paper we want to demonstrate the usability of neural-network based adaptive algorithms in those cases where conventional algorithms cannot be used anymore.

Neural networks have been applied previously to the control of a SoftArm robot. In (Hesselroth, Sarkar, van der Smagt, & Schulten, 1994), a visual observation of the real and desired end-effector position is directly translated to rubbertuator pressure by using a Kohonen-type neural network. This network learns to position the end-effector within 1cm of its desired position after learning. However, learning sessions are very time-consuming (in the order of hours), and each positioning trial takes in the order of 30s, such that the system has limited practical applicability. Furthermore, the trajectory of the robot arm which connects one endpoint to another is uncontrolled and oscillatory. In (Sarkar & Schulten, 1996) a hierarchical self-organising network is used to not only position the end-effector of the robot arm, but also control the orientation of the gripper. Only 300 learning trials are required to accurately train the system, yet again the motion of the robot arm is slow and oscillatory, resulting in a training time in the order of hours. In (Katayama & Kawato, 1992), a single joint of a similar robot arm is dynamically controlled to follow a trajectory in joint space, resulting in an error in the order of about $1°$ for a fast movement. Again, these good results are only obtained after long training sessions: in this case, the authors report to have followed a trajectory 2,000 times before this acceptably high accuracy obtained.

In this paper a feed-forward network based dynamic control system for the SoftArm robot as produced by Bridgestone Corp. is proposed. It is required that the system learns correct behaviour quickly. In section 1 the various parts of the robot system are described. In section 2 the measured dynamical behaviour of the robot arm is described, and theoretically explained. Section 3 describes a simple neural system for stably controlling a single joint of the robot, the results of which are given in section 4. A

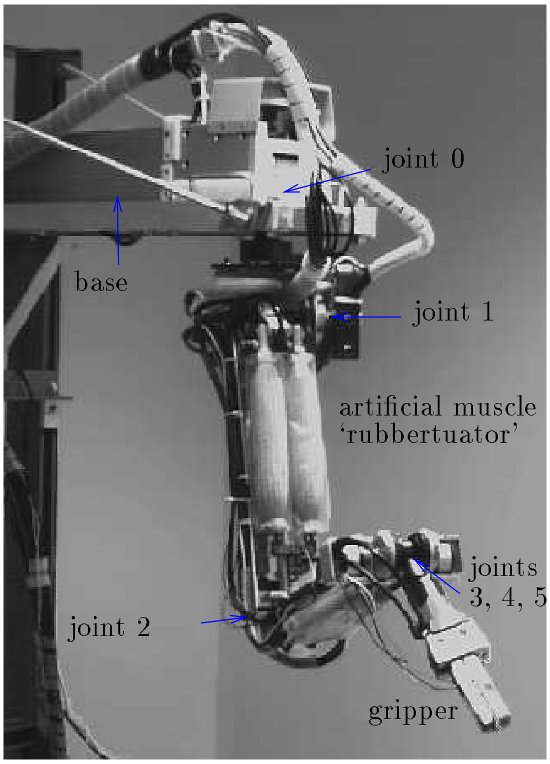

Figure 1: The structure of the SoftArm robot.

summary is provided in section 5.

1 The robot system

The robot arm was built mainly from components manufactured by Bridgestone Corporation of Tokyo, Japan. The whole robot system consists of a robot arm, an air compressor, servo-drive and servo-valve units, and a gripper.

1.1 Kinematic system

The robot is a four-link manipulator with five degrees of freedom. It is mounted by suspending it from its top joint. A labeled picture of the SoftArm is reproduced in figure 1. The arrangement of the joints and their range of movement is basically modelled after the human arm. Because its pneumatic actuators, each consisting of two or four inflatable rubber tubes named *rubbertuators*, are relatively light, the arm weighs only 12 kg yet can lift 3 kg. Because of its weight and compliant characteristics, this arm can be employed for application around human operators or fragile equipment.

Table 1: Dimensions of the links and motion range of the joints.

<table>
<tr><th colspan="3">Item</th><th>Specification</th></tr>
<tr><td colspan="3">model</td><td>FAS–501</td></tr>
<tr><td colspan="3">degree of freedom</td><td>5</td></tr>
<tr><td rowspan="10">rotation angle and arm length</td><td rowspan="2">first (shoulder)</td><td>angle</td><td>$\pm 60°$</td></tr>
<tr><td>length</td><td>–</td></tr>
<tr><td rowspan="2">second (upper arm)</td><td>angle</td><td>$\pm 50°$</td></tr>
<tr><td>length</td><td>410 mm</td></tr>
<tr><td rowspan="2">third (lower arm)</td><td>angle</td><td>$\pm 50°$</td></tr>
<tr><td>length</td><td>370 mm</td></tr>
<tr><td rowspan="2">fourth (wrist pitch)</td><td>angle</td><td>$\pm 45°$</td></tr>
<tr><td>length</td><td>270 mm</td></tr>
<tr><td rowspan="2">fifth (wrist roll)</td><td>angle</td><td>$\pm 90°$</td></tr>
<tr><td>length</td><td>–</td></tr>
<tr><td colspan="3">lifting capability</td><td>max. 3 kg</td></tr>
</table>

Intended uses are in hospitals, around the handicapped, for household tasks and in areas where electrical circuits cannot be introduced. The dimensions and range of movement of the joints are given in table 1.

The torque applied to each joint can be controlled by setting the pressures ρ_1 of the agonist and ρ_2 of the antagonist rubbertuator pairs. The rubbertuators to drive link i are fixed parallel to each other in link $i-1$. The free ends are connected to each other by a chain. The chain goes around a sprocket fixed in link $i-1$ and connected to link i. The angular position of joint i thus depends on the relative lengths of the tubes as shown in fig. 2. This relationship can be expressed as

$$\theta = \frac{l_1 - l_2}{2\pi r} \tag{1}$$

where r is the radius of the sprocket and l_1 and l_2 are the respective lengths of the rubbertuators.

One of the greatest advantages of a rubbertuator is its very high force-to-weight ratio, about 240, compared to a value of about 16 for DC servo motors. This is especially good for robotics applications in which the actuators for the extreme joints are in motion as part of the arm.

The *stiffness* of any joint is defined as the total pressure $\rho_0 = \rho_1 + \rho_2$ of the rubbertuators that drive it. When this total pressure is high, the joint exhibits a stiff behaviour, whereas a low ρ_0 results in a compliant behaviour.

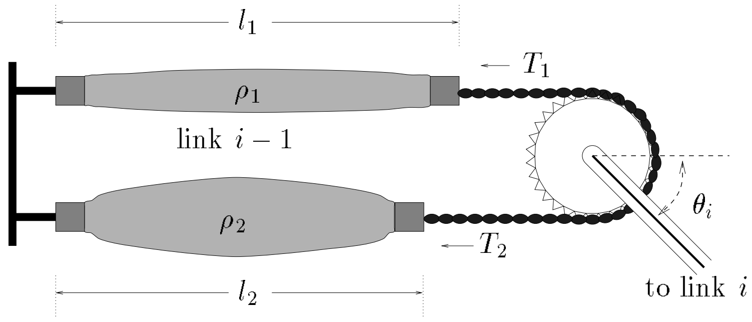

Figure 2: An agonist and an antagonist rubbertuator are connected via a chain across a sprocket; their relative lengths determine the joint position θ_i.

1.2 The rubbertuator drive system

The robot is supplied with compressed air of constant pressure. Five servo-drive units (SDU's) provide the internal control circuitry for the robot. Each unit receives 11-bit precision pressure signals from the host computer, converts them to analogue signals, and sends them to a servo-valve unit (SVU). The servo-valve unit senses the pressure of each of the two rubbertuators it controls, and regulates this pressure by opening or closing electric valves.

1.3 The gripper and its controlling valves

A gripper of about 1kg is installed at the end of the arm. It has a simple two-fingered clamping action and is powered by air pressure. The fingers are approximately 10 cm long. Two inlets are required, one for opening and the other for closing. The air pressure is supplied through electric valves which can be controlled by the computer.

2 Dynamics

The dynamics of any d degree of freedom robot with rotational joints can be described by the equation (Craig, 1986)

$$T\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}\right) = F_1(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + F_2(\boldsymbol{\theta})\left[\dot{\boldsymbol{\theta}}\dot{\boldsymbol{\theta}}\right] + F_3(\boldsymbol{\theta})\left[\dot{\boldsymbol{\theta}}^2\right] + F_4(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + F_5(\boldsymbol{\theta}) \quad (2)$$

where T is a d-vector of torques exerted by the links, and $\boldsymbol{\theta}$, $\dot{\boldsymbol{\theta}}$, and $\ddot{\boldsymbol{\theta}}$ are d-vectors denoting the positions, velocities, and accelerations of the d joints. $[\dot{\boldsymbol{\theta}}\dot{\boldsymbol{\theta}}]$ and $[\dot{\boldsymbol{\theta}}^2]$ are vectors

$$\left[\dot{\boldsymbol{\theta}}\dot{\boldsymbol{\theta}}\right] = \left[\dot{\theta}_1\dot{\theta}_2, \dot{\theta}_1\dot{\theta}_3, \ldots, \dot{\theta}_{d-1}\dot{\theta}_d\right]^T, \qquad \left[\dot{\boldsymbol{\theta}}^2\right] = \left[\dot{\theta}_1^2, \dot{\theta}_2^2, \ldots, \dot{\theta}_d^2\right], \quad (3)$$

$F_1(\theta)$ is the matrix of inertia, $F_2(\theta)$ is the matrix of Coriolis coefficients, $F_3(\theta)$ is the matrix of centrifugal coefficients, $F_4(\theta, \dot{\theta})$ is a friction term, and $F_5(\theta)$ is the gravity working on the joints.

When the robot has to move from one joint position to another, a torque must be applied which generates T. The problem of calculating the correct torques (forces) to have the robot arm follow a specified trajectory is known as **inverse dynamics**. Industrial robots are generally designed to eliminate the interdependence between the joints, such that the robot arm can be approximated by d independent moving bodies. In that case, F_1 and F_3 are diagonal matrices and F_2 is zero. This reduces the $3d$-values vector field (2) to d independent functions of three variables for which the coefficients have to be found. Also, the link actuators are usually made so powerful that F_1, F_3, F_4, and F_5 can be considered independent of θ. For this simplified (and common) case, various standard methods exist to compute the inverse dynamics (Fu, Gonzalez, & Lee, 1987). This controller eliminates the requirement of knowledge of the robot arm in order to control it.

2.1 The dynamics of the SoftArm

For the SoftArm, however, the above simplifications cannot be made. Due to the use of compliant material in the actuators, the Coriolis forces of F_2 cannot be neglected, and the various joints affect each other heavily, leading to non-diagonal matrices F_1 and F_3. Furthermore, some of these matrices change in time due to external influences.

The servo-drive units allow the robot to be controlled in two modes: *position control mode* (closed loop control) and *pressure control mode* (open loop control). When the SoftArm is controlled in position control mode, an internal PID controller (see, e.g., (Craig, 1986)) is used in a feedback loop. This PID controller uses joint position feedback from the optical shaft encoders mounted on each joint to determine the pressure of the joints in a closed loop. Figure 3 shows a representative move of one joint of the robot arm. The feedback mechanism should generate a smooth motion, but due to non-optimal feedback control the move is oscillatory.

In pressure control mode, the pressure values sent by the host computer are directly translated to currents for the valves and the rubbertuator pressures are set correspondingly. The pressure generates a force in the rubbertuators which makes the joint rotate to assume a new equilibrium position.

2.2 Behaviour of a rubbertuator driven joint

In order to further understand the dynamics of the SoftArm robot, we will first have to investigate the behaviour of a single rubbertuator. Figure 4 shows the structure of a rubbertuator.

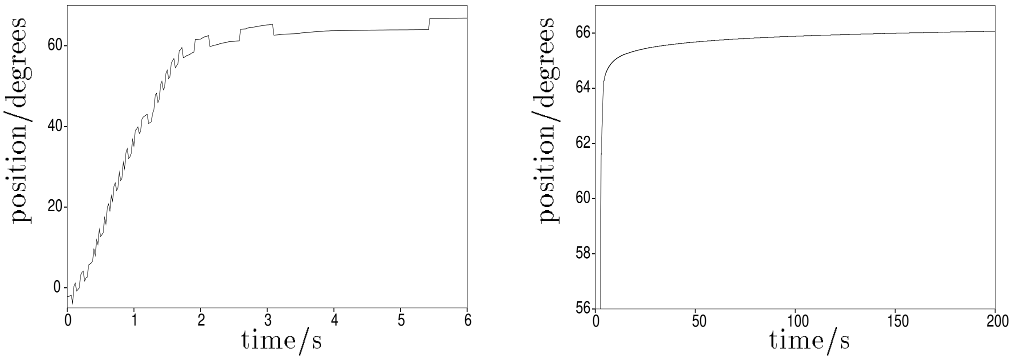

Figure 3: Joint 2 of the rubbertuator robot moving in position and pressure control mode. The left figure shows the position control mode, i.e., closed loop control. Notice the jagged curve due to the feedback in the internal PID controller; this is probably caused by an incorrect parameter setting for the Integrator part of the PID controller. In the right figure, pressure control mode (open loop control) is used. The figure clearly shows that, due to the behaviour of the rubber, it takes a long time before the joint settles to its steady state due to the elastic behaviour of the rubber.

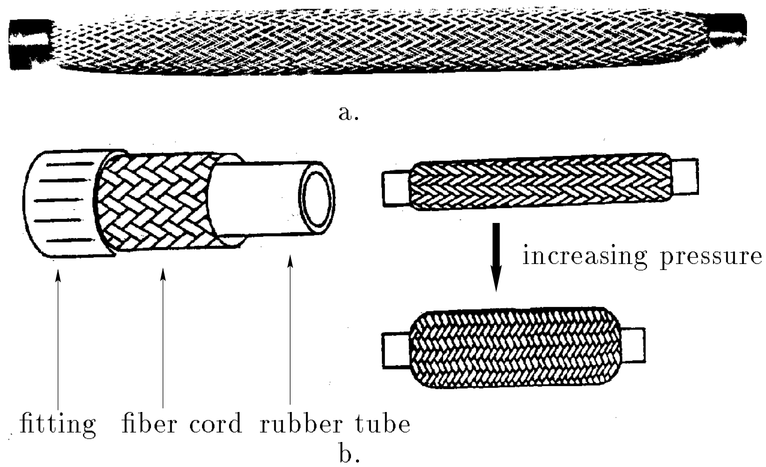

Figure 4: The structure of a single rubbertuator. In a., a photograph of a rubbertuator is shown. In b., a schematic representation of the structure of the rubbertuator tube is given.

Each actuator consists of a rubber tube sealed on one end and with an air inlet on the other end. The contraction force T_j exerted by rubbertuator $j \in \{1, 2\}$ for each joint is specified by the manufacturer as

$$T_j = \rho_j D_j^2 \left(a(1 - \varpi_j)^2 - b \right) \tag{4}$$

where ρ_j is the supply pressure, a and b are constants depending on the particular tube, $0 \leq \varpi_j < 0.2$ is the contraction ratio which is directly (approximately linearly) related to the rubbertuator length l_j, and D_j is the effective diameter of the tube before displacement. Although (4) is not a precise model of the rubbertuators, it suffices to qualitatively describe their behaviour.

The driving force of a rubbertuator varies with pressure and the contraction ratio. For instance, under constant pressure it has such a spring characteristic that the contracting force becomes stronger as the degree of contraction becomes smaller. When the contraction ratio is constant, the force increases with increasing pressure. Thus the rubbertuator has spring-like characteristics, while allowing a varying spring constant.

Pressure–position relation. From (4) it can be seen that for any specific choice of ρ_j there exists an infinite number of values ϖ_j and D_j which realise a specific exerted force T_j. Therefore, when a joint is in equilibrium, i.e., the external forces (gravity) are equal to $T_1 - T_2$, the joint angle is not only dependent on the pressure but also on the diameter of the tube before displacement. Since the diameter depends on the pressure and the elongation (before the displacement), the new joint position depends on the new pressure as well as on the previous position. Figure 5a demonstrates this hysteresis effect for joint 1.

This hysteresis can be shown by moving a joint along a pressure trajectory from $\rho_1 = 0, \rho_2 = \rho_{\max}$ to $\rho_1 = \rho_{\max}, \rho_2 = 0$ and back again by incrementing and decrementing the pressures by a constant value $\Delta\rho$. This results in the behaviour shown in figure 5b. The width of the gap between the two curves depends on how fast the pressures are changed; the slower the change of the pressures, the narrower the gap.

The trajectory and velocity in joint space followed for a constant pressure increase and decrease is depicted in figures 5c and d. The velocity is numerically computed from the position. Near the extreme values the joint velocity decreases since the increase in exerted force for a constant change in pressure is less.

Elasticity of the rubbertuators. The long-term settling behaviour of the rubber has a large effect on the position of a joint *after* the desired pressure is reached and the joint seems to have reached its position. Figure 3 shows the position of joint 1 in time when the rubbertuators are allowed to

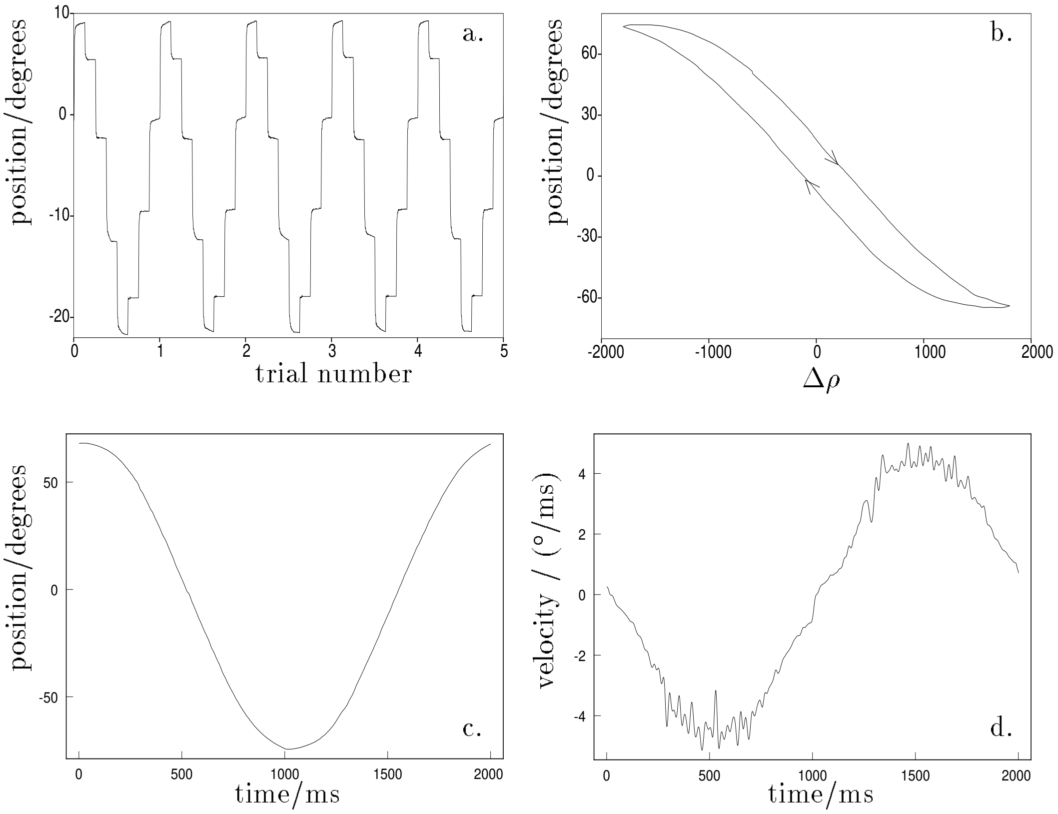

Figure 5: Hysteresis in SoftArm positioning for joint 1. In a., the joint angle reached when ρ_1/ρ_2 is changed from 600/900 to 675/825 to 750/750 to 825/675 to 900/600 to 825/675 to 750/750 to 675/825 and back again. The position reached depends both on the previous position as well as on the new pressures. In b., the joint is moved by applying a constant pressure increment $\Delta\rho$ to rubbertuator 1 and the same decrement to rubbertuator 2. When the extreme pressures are reached, the direction is reversed. Figure c shows the joint position while the trajectory of figure b is followed; d shows the velocity. An extreme joint position is reached at $t = 0$ ms and $t = 1000$ ms. From $t = 0$ to $t = 1000$ the change in pressures for the rubbertuators is constant. Since the exerted force is constant near the equilibrium point, the joint has a constant acceleration profile.

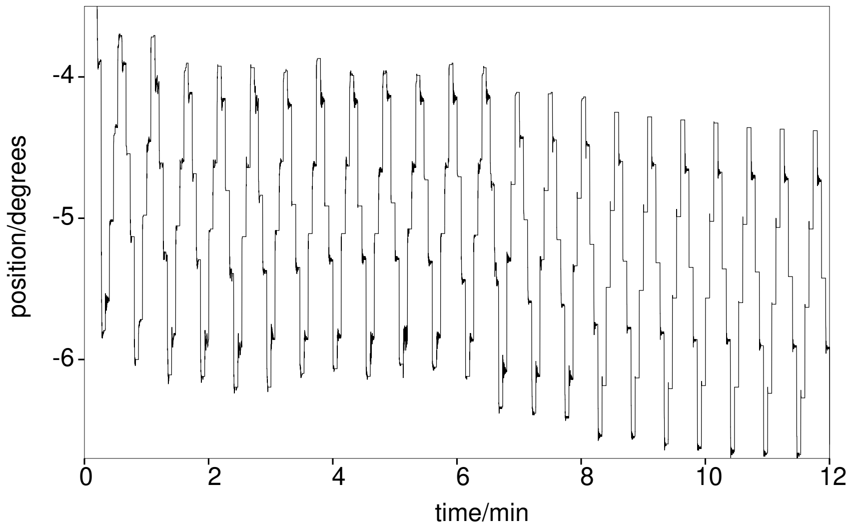

Figure 6: Drift of the rubbertuators when the robot is used for a long period of time. The pressures of the rubbertuators are repeatedly increased/decreased by 1% of the total pressure.

settle for 200 seconds in pressure control mode. During this settling time, the joint rotates for about 1°.

Influence of the temperature of the rubbertuators (which can occur due to varying climate conditions or simply by using the arm for extended periods of time) has also a large influence on the pressure–position relation. When repeatedly moving the robot to the same pressure, the system drifts gradually to different positions (see fig. 6).

2.3 Analysis of the rubbertuator behaviour

In order to explain and attempt to model the behaviour described above, we have to consider the structure of a pair of rubbertuators as shown in figure 2. The total force ΔT which the combined rubbertuators exert on the joint is, according to (4),

$$\Delta T = \rho_1(a(1-\varpi_1)^2 - b)D_1^2 - \rho_2(a(1-\varpi_2)^2 - b)D_2^2.$$

2.3.1 Pressure-force relation

If we assume that $D = D_1 = D_2$, i.e., the rubbertuators are in their 'middle' position, then

$$\Delta T = \left[\rho_1(1-\varpi_1)^2 - \rho_2(1-\varpi_2)^2)\right] aD^2 + (\rho_2 - \rho_1)bD^2 \qquad (5)$$

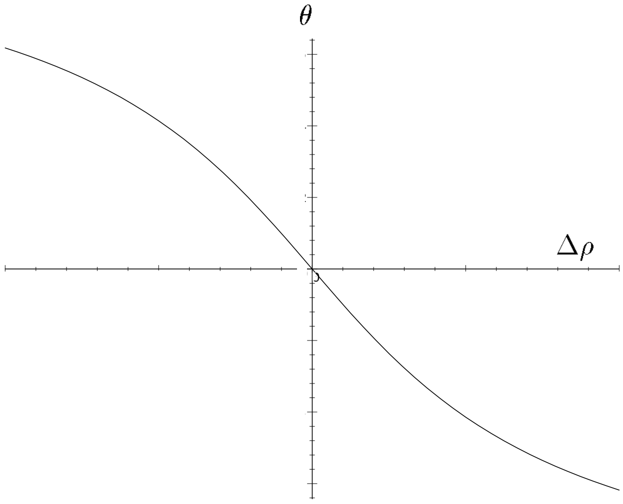

Figure 7: Equilibrium line of (6). For the figure we have taken $\mu_1 > 0$, $\mu_2 > 0$, and $\mu_3 > 0$.

Defining $\Delta\rho = \rho_1 - \rho_2$ (the 'difference pressure') and $\rho_0 = \rho_1 + \rho_2$ (the 'base pressure' or *stiffness*) we can express

$$\begin{aligned}\Delta T &= 1/2\left[\rho_0((1-\varpi_1)^2 - (1-\varpi_2)^2) + \Delta\rho((1-\varpi_1)^2 + (1-\varpi_2)^2)\right] aD^2 \\ &\quad -\Delta\rho b D^2\end{aligned}$$

By setting $c = \varpi_1 + \varpi_2$ and $K\theta = \varpi_1 - \varpi_2$, this results in

$$\begin{aligned}\Delta T &= 1/2\left[\rho_0(\varpi_1 - \varpi_2)(c-2) + 2\Delta\rho(1-c) + \frac{(\varpi_1 - \varpi_2)^2}{2}\Delta\rho + \frac{c^2}{2}\Delta\rho\right] aD^2 \\ &\quad -bD^2\Delta\rho \\ &= 1/2\left[\rho_0 K\theta(c-2) + 2\Delta\rho(1-c) + \frac{(K\theta)^2}{2}\Delta\rho + \frac{c^2}{2}\Delta\rho\right] aD^2 - bD^2\Delta\rho \\ &= \underbrace{1/2aD^2K(c-2)}_{\mu_1}\rho_0\theta + \underbrace{\frac{aD^2K^2}{4}}_{\mu_2}\theta^2\Delta\rho + \underbrace{\left[\frac{c^2aD^2}{4} - bD^2\right]}_{\mu_3}\Delta\rho.\end{aligned}$$

In conclusion, we can write

$$\Delta T = \mu_1\rho_0\theta + \mu_2\Delta\rho\theta^2 + \mu_3\Delta\rho \qquad (6)$$

To understand this result, we must look at those values of θ and $\Delta\rho$ where ΔT in (6) is 0, i.e., the system is in equilibrium. The trajectory followed in figure 7 corresponds to the measured trajectories of figure 5b.

2.3.2 Explaining hysteretic behaviour

The hysteretic behaviour of a rubbertuator driven joint can be modelled by substituting different values for D in (4). This hysteretic behaviour is, in fact, a result of the material used in the rubbertuators (Holownia, 1977).

If we assume that rubbertuator 1 has a diameter $D_1 = D + \Delta D$ before displacement, and that rubbertuator 2 has a diameter $D_2 = D - \Delta D$ before displacement, then (4) can be written as

$$\begin{aligned} \Delta T &= P_1(a(1-\varpi_1)^2 - b)(D+\Delta D)^2 + P_2(a(1-\varpi_2)^2 - b)(D-\Delta D)^2 \\ &= P_1(a(1-\varpi_1)^2 - b)(D^2 + \Delta D^2 + 2D\Delta D) \\ &\quad + P_2(a(1-\varpi_2)^2 - b)(D^2 + \Delta D^2 - 2D\Delta D). \end{aligned} \tag{7}$$

By splitting this equation in three parts for D, ΔD, and $D\Delta D$ we can apply (6) by substituting first D and then ΔD in (5), such that

$$\begin{aligned} \Delta T &= \mu_1\rho_0\theta + \mu_2\Delta\rho\theta^2 + \mu_3\Delta\rho + \\ &\quad \mu_1'\rho_0\theta + \mu_2'\Delta\rho\theta^2 + \mu_3'\Delta\rho + \\ &\quad 2D\Delta D\left[\rho_1(a(1-\varpi_1)^2 - b) + \rho_2(a(1-\varpi_2)^2 - b)\right] \\ &= (\mu_1+\mu_1')\rho_0\theta + (\mu_2+\mu_2')\Delta\rho\theta^2 + (\mu_3+\mu_3')\Delta\rho + \\ &\quad \mu_1''\Delta\rho\theta + \mu_2''\rho_0\theta^2 + \mu_3''\rho_0 \end{aligned} \tag{8}$$

by applying similar transformations as above. Thus the parameters μ_i' and μ_i'' are

$$\mu_i' = \mu_i\frac{\Delta D^2}{D^2}, \qquad \mu_i'' = \mu_i\frac{\Delta D}{D}.$$

Again we can plot the equilibrium lines of (8) by solving $\Delta T = 0$. This has to be done for $\Delta D > 0$ and $\Delta D < 0$, which results in different signs for the parameters μ_i''. The result is shown in figure 8. From this figure it is clear that the hysteresis can be explained from the model of a rubbertuator driven joint.

3 Control of the SoftArm

From the above it is obvious that a precise model for the pneumatic actuators cannot be easily constructed. When the robot arm is used for accurate positioning and orientation of the end-effector, an adaptive algorithm is preferred for controlling the robot.

In this section an adaptive system will be described which is capable of making one of the robot's joints follow a prescribed trajectory.

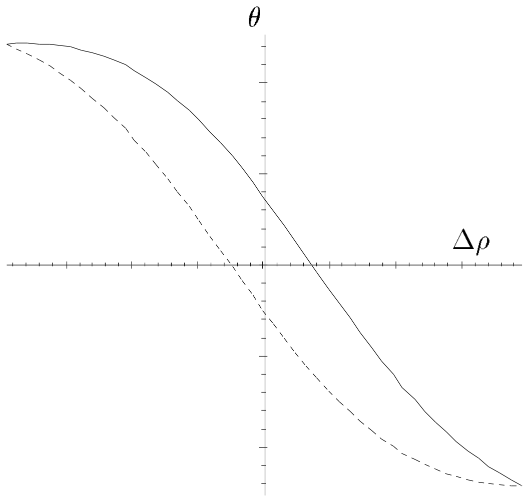

Figure 8: Equilibrium lines of (8). In this case we have taken $\mu_1 + \mu_1' > 0$, $\mu_2 + \mu_2' > 0$, $\mu_3 + \mu_3' > 0$, while we used $\mu_1'' > 0$, $\mu_2'' < 0$, and $\mu_3'' > 0$ for the dotted line, and $\mu_1'' > 0$, $\mu_2'' > 0$, and $\mu_3'' > 0$ for the solid line.

3.1 System setup

From (6) it can be seen that the torque depends on the pressure difference $\Delta\rho$, the stiffness ρ_0, and the diameter before displacement $D_{1,2}$. Since the value of $D_{1,2}$ is not known and cannot be easily measured, we instead can use $\rho_{1,2}$ before displacement in combination with θ to carry the same information.

The task of the robot controller is to generate pressures $\rho_1(t)$ for the first muscle of a joint, such that a specified trajectory $\left(\theta_d(t), \dot{\theta}_d(t), \ddot{\theta}_d(t)\right)$ is followed. The 'stiffness' $\rho_1 + \rho_2$ is always kept constant, such that the pressure from the second rubbertuator can be derived from the first. Since the system is a discrete-step closed loop, we will employ an index $[i]$ instead of continuous time (t).

The robot control system, which is depicted in figure 9, receives values $\theta[i]$ from the robot at intervals of approximately 20 ms. In order to obtain noise-insensitive estimates of $\dot{\theta}$ and $\ddot{\theta}$, these values are fitted to orthonormal polynomials following an incremental algorithm derived from (Forsythe, 1957; Hayes, 1970). Thus we can, with some accuracy, find θ, $\dot{\theta}$, and $\ddot{\theta}$ at each desired time.

The measured pressure $\rho_1[i]$, $\theta[i]$, $\dot{\theta}[i]$, and $\ddot{\theta}[i]$, and the desired $\theta_d[i]$, $\dot{\theta}_d[i]$, and $\ddot{\theta}_d[i]$ are input to a feed-forward neural network $\mathcal{N}$. The network then generates a target pressure $\rho_1[i+1]$ which is sent to the robot $\mathcal{R}$. The obtained rotation, after the pressure change has been applied, is used as a

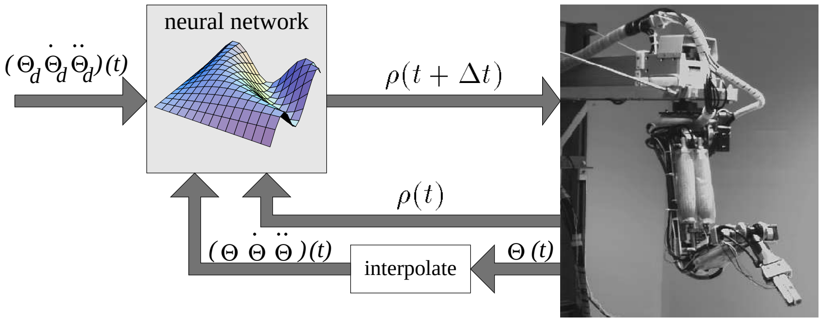

Figure 9: The neural robot control system in a feedback loop with the robot.

new learning sample. Thus, the network performs the mapping

$$\mathcal{N}\left(\rho_1[i], \theta[i], \dot{\theta}[i], \ddot{\theta}[i]\theta_d[i], \dot{\theta}_d[i], \ddot{\theta}_d[i]\right) = \rho_1[i+1].$$

The new pressure is sent to the robot to determine

$$\mathcal{R}\left(\theta[i], \dot{\theta}[i], \rho_1[i], \rho_1[i+1]\right) = \left(\theta[i+1], \dot{\theta}[i+1]\right)$$

and a new learning sample is available:

$$\mathcal{C}^0\left(\rho_1[i], \theta[i], \dot{\theta}[i], \ddot{\theta}[i]\theta[i+1], \dot{\theta}[i+1], \ddot{\theta}[i+1]\right) = \rho_1[i+1] \qquad (9)$$

where $\mathcal{C}^0$ indicates the ideal controller. This ideal controller defines the controller we would like to have to control the robot. Although we cannot analytically construct $\mathcal{C}^0$, we *can* construct learning samples which describe $\mathcal{C}^0$, and use these to approximate $\mathcal{C}^0$.

3.2 Neural network structure

The control system consists of two neural networks running on two processors in parallel. One neural network (the controlling network) gathers the data from the robot and calculates the joint velocity and acceleration, uses these data to compute the new joint pressures, and generates a learning sample. By definition of $\mathcal{N}$ and $\mathcal{R}$, the created learning sample is given by $\mathcal{C}^0$ from (9). It is sent to the second neural network (the learning network). This network maintains a set of samples to which newly generated samples are added, or are used to replace older samples (van der Smagt, 1995). Minimisation is performed on this set of samples. Thus this neural network is taught to approximate $\mathcal{C}^0$ as well as possible from the available learning samples. The resulting optimal weight matrix is sent from the learning network to the controlling network.

For the single joint problem, both networks are feed-forward networks consisting of 7 inputs, 15 hidden units, and one output. The learning network is trained using conjugate gradient optimisation with Powell restarts as

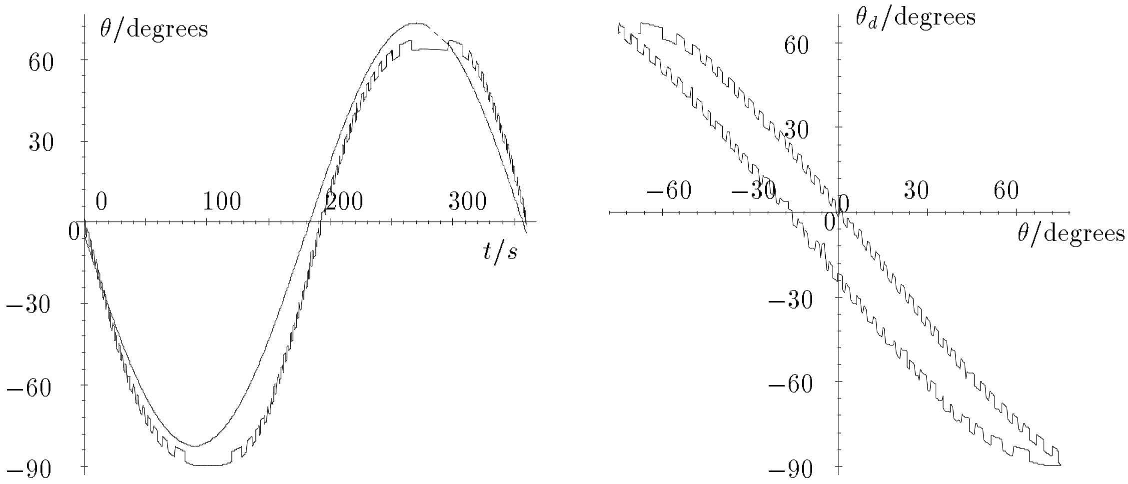

Figure 10: Using the internal PID controller to follow the trajectory $\theta_d(t) = c\sin(t)$. The left figure shows the desired and realised trajectories vs. t; the right figure depicts the desired (horizontal axis) vs. the realised (vertical axis) trajectory.

described in (van der Smagt, 1994). Conjugate gradient optimisation methods use second-order information of the function that has to be minimised. In this case, the error function is calculated in a second-order approximation using local information, which is subsequently minimised (this operation takes n steps for a system with n degrees of freedom c.q. weights). During minimisation the second-order approximation is updated using local information and the minimisation procedure is subsequently repeated. See, e.g., (Press, Flannery, Teukolsky, & Vetterling, 1986; Shewchuk, 1994) or other numerical analysis books for further information.

4 Results

In order to evaluate the success of the neural controller, it is compared with the internal PID controller on two trajectories. For the first trajectory one of the joints of the robot arm has to follow a trajectory $\theta_d(t) = c\sin(t)$; for the second, the trajectory is more complicated: $\theta_d(t) = c\sin(t)\cos^2(11t)$. The success of each controller is measured by comparing $\theta_d(t)$ with the realised trajectory $\theta(t)$.

4.1 A simple trajectory

First the internal PID controller is tested on the trajectory. The result of this trial is shown in figure 10. As this figure shows, the PID controller suffers from three problems. First, the internal parameters of the controller

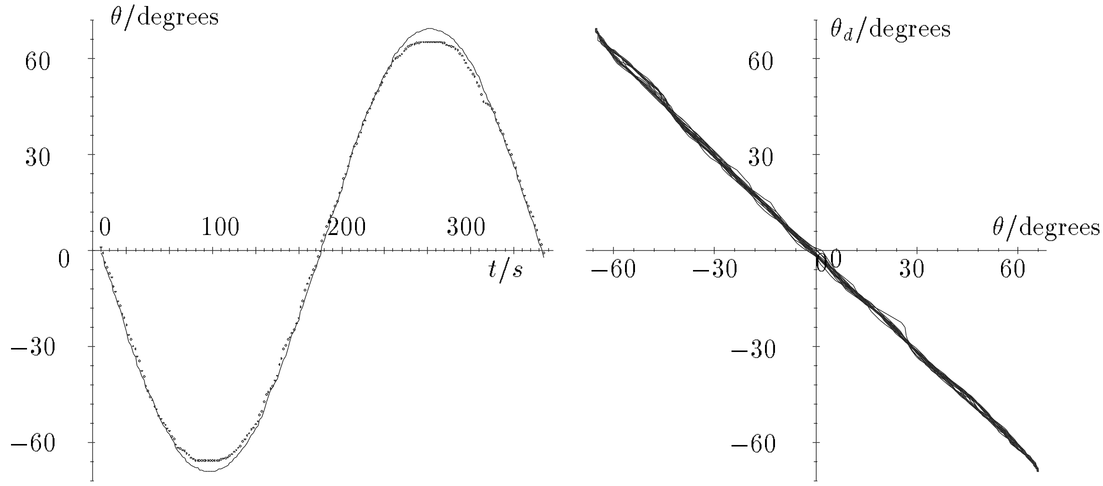

Figure 11: Using the neural network controller to follow the trajectory $\theta_d(t) = c\sin(t)$. The left figure shows the desired (solid line) and realised (dotted line) trajectories vs. t; the right figure depicts the desired (horizontal axis) vs. the realised (vertical axis) trajectory. This behaviour is recorded after five minutes of learning.

are not set correctly, leading to the jagged form of the curve. Second, the controller is lagging behind the desired trajectory. The third problem is the most serious: as the right figure in 10 shows, the PID controller does not solve the hysteresis problem. The relationship $\theta(t)$–$\theta_d(t)$ depends on the direction of motion. Furthermore, this position difference is larger than the error caused by the incorrect PID parameters (jaggedness).

Figure 11 shows the control of the same target trajectory depicted in figure 10 after a long training session. As the figure shows, the trajectory is accurately followed except in the extrema where an error of approx. $1.5°$ remains. The right figure 11 shows the desired vs. the realised trajectory. It is clear that the system does not suffer from hysteresis anymore.

The initial learning behaviour is shown in figure 12. It is clearly shown in this figure that the network learns to control the robot after only a few trials. However, the behaviour such as depicted in figure 11 is only possible after a training session of approximately 2000s.

4.2 A more complex trajectory

The system, with no a priori knowledge, has also been trained on a trajectory $\sin(t)\cos^2(11t)$. Initially, the trajectory has been followed only very coarsely. After 16 trials, however, accurate trajectory following has been obtained, with an error of $1°$ near the extrema, and less than $0.1°$ on the slope (see figure 13).

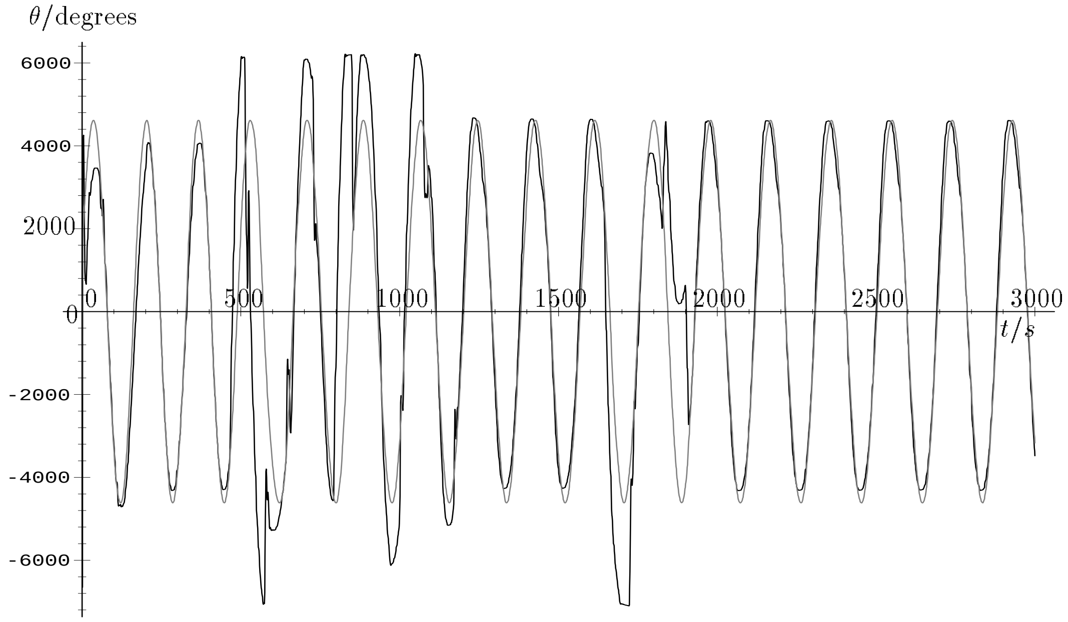

Figure 12: Learning the control. The figure shows the desired (grey line) and realised (solid line) trajectories vs. t. This figure clearly shows that the network learns to follow the trajectory very quickly.

5 Discussion

In this paper we have applied a neural-network based controller to a pneumatic robot arm with complex, highly nonlinear dynamics, which change in time due to external influences. This controller has been shown to perform better than the manufacturer-specified PID controller, and learns correct trajectory following after only a few trials. The investigations have led to a better understanding of the SoftArm robot. It has been shown that the pressure-position relationship of the joints are a direct consequence of the behaviour of the rubbertuators. Furthermore, it has been shown that the time integral of the base pressure $\rho_1 + \rho_2$ is directly related to the joint rotation that this pressure change instigates.

In the current experiments, the controller is applied to only one joint at a time. This means that the system is reduced from a 35-dimensional (or 21-dimensional when the gripper is treated separately) to a 7-dimensional one. However, previous experience with scalability of neural-network driven robot controllers (van der Smagt, 1995) has shown promising results, with the availability of the required computing power.

In order to find the optimal number of hidden units needed in the neural network, as well as the optimal size of the set of learning samples over which minimisation is performed, we used the method described in (Vyšniauskas,

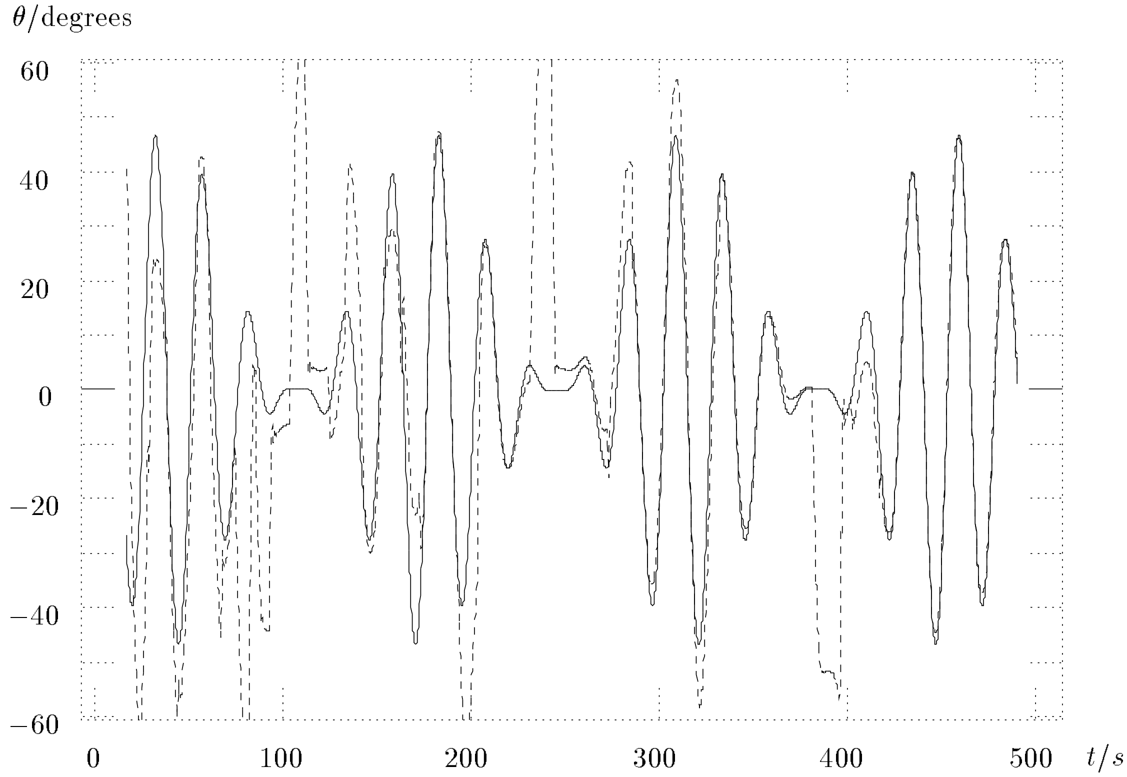

Figure 13: Training the system on a $\sin(t)\cos^2(11t)$ trajectory. The solid line is the target trajectory, the dotted line the realised trajectory. Initially, the trajectory is reasonable followed where velocity is constant; after only 16 trials, the whole trajectory is followed with an average error of $0.1°$.

Groen, & Kröse, 1992); cf. (Barron, 1991). With this method, an asymptotical model of the error function is constructed and used to find the optimal number of hidden units and learning samples to attain a certain error in the approximation.

Reference

Barron, A. R. (1991). Approximation and estimation bounds for artificial neural networks. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pp. 243–249.

Craig, J. J. (1986). *Introduction to Robotics.* Addison-Wesley Publishing Company, Reading, MA.

Forsythe, G. E. (1957). Generation and use of orthogonal polynomials for data-fitting with a digital computer. *J. Soc. Indust. Appl. Math.*, *5*(2).

Fu, K. S., Gonzalez, R. C., & Lee, C. S. G. (1987). *Robotics: Control, Sensing, Vision, and Intelligence.* McGraw-Hill Book Company, New York.

Hayes, J. G. (1970). Curve fitting by polynomials in one variable. In Hayes, J. G. (Ed.), *Numerical Approximation to Functions and Data.* University of London, The Athlone Press, London.

Hesselroth, T., Sarkar, K., van der Smagt, P., & Schulten, K. (1994). Neural network control of a pneumatic robot arm. *IEEE Transactions on Systems, Man, and Cybernetics*, *24*(1), 28–38.

Holownia, B. P. (1977). Temperature buildup in bonded rubber blocks due to hysteresis. *Rubber Chemistry and Technology*, *50*(1), 186–193.

Katayama, M., & Kawato, M. (1992). A parallel-hierarchical neural network model for motor control of a musculo-skeletal system. Tech. rep. TR–A–0145, ATR Auditory and Visual Perception Research Laboratories.

Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterling, W. T. (1986). *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, Cambridge.

Sarkar, K., & Schulten, K. (1996). Topology representing network in robotics. In van Hemmen, J. L., Domany, E., & Schulten, K. (Eds.), *Models of Neural Networks*, Vol. 3 of *Physics of Neural Networks*, pp. 281–302. Springer–Verlag, New York.

Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain. Tech. rep. CMU–CS–94–125, Carnegie Mellon University.

van der Smagt, P. (1994). Minimisation methods for training feed-forward networks. *Neural Networks*, *7*(1), 1–11.

van der Smagt, P. (1995). *Visual Robot Arm Guidance using Neural Networks*. Ph.D. thesis, Dept of Computer Systems, University of Amsterdam.

Vyšniauskas, V., Groen, F. C. A., & Kröse, B. J. A. (1992). Function approximation with a feedforward network: the optimal number of learning samples and hidden units. Tech. rep. CS–92–15, Department of Computer Systems, University of Amsterdam, Amsterdam, Netherlands.