Globally Stable Neural Robot Control Capable of Payload Adaptation *

M. Jansen and R. Eckmiller
Department of Computer Science VI, Neuroinformatics
University of Bonn, Römerstr. 164, FR Germany
Tel.: ++49-228-550-349 FAX: ++49-228-550-425 e-mail: jan@nero.uni-bonn.de

Abstract A set of four separate Three-Layer-Perceptrons 3LP learns matrix components representing mass-coupling, coriolis, viscose, and static friction forces in an inverse robot model as a function of the robot’s current position and payload. Based on training with PTP-trajectories between random start- and goal-points that are executed with various load masses, the inverse model gradually acquires high precision over the entire robot working range. A controller using the 3LP-networks inside the feedback loop is shown to be globally $L_\infty$-stable. The stability criterion is based on guaranteed model error bounds for the complete continuous working range and for all load masses in a certain range. Results of the stability analysis and of load-adaptive control are demonstrated for a realistically simulated planar 4-joint-machine.

Introduction Model-based dynamic robot control uses an inverse model of the rigid body dynamics in series with the robot to obtain a decoupled unity-gain system [1]. For a simulated, planar (kinematically redundant) 4-joint-machine (4JM), we trained a given set of neural networks to learn such an inverse model in an identification phase and to act as adaptive controller in the subsequent control phase.

The neural networks are able to identify the robot dynamics by extracting the model information from simple Point-to-Point (PTP) training movements, whereas classical approaches of robot model identification need laborious measurement procedures to determine the parameters of an analytically represented model (i.e. Newton-Euler dynamics). Furthermore the possibility of effective, fast, parallel hardware implementation arises for the given, sufficiently small, networks. The presented approach is distinguished from other neural robot control approaches (e.g. [3], [4]) by providing, for the first time, model precision and stable control over the complete continuous working range of the robot (except for a small border zone) and for a certain continuous payload range (rather than for one single repeated trajectory and for only one possible load). Based on this feature stable control could be guaranteed using a criterion for $L_\infty$-stability, established for general model-based robot control [5].

This paper describes controller structure, training procedure, a method to rigorously guarantee stability for all positions and loads following the training phase, as well as the application of the networks as load-adaptive controller for a realistically simulated 4JM.

A non-adaptive version of the proposed controller and a brief derivation of the stability criterion were recently published [2].

With regard to non-modeled dynamic effects it should be mentioned that the robustness of the controller to joint elasticities could be demonstrated in the following examples. Further realistic effects, like controller delay, were not treated here, but their incorporation for the application to the real physical robot is currently investigated.

The Controller Structure Three-Layer-Perceptrons are used. The controller is structured into four separate networks, to represent different components of the torque vector $\tau$ due to different mechanical forces. According to the robot dynamics (eqn.1), the networks learn the matrix components representing inertia and coupling forces $M$, coriolis and centrifugal forces $C$, and viscose $F$ and static $f_s$ friction forces, respectively. The angular positions $\tilde{\theta}$ are the inputs to the network approximations $\tilde{M}$ and $\tilde{C}$. The current load mass is an additional input.

$$\tau = M(\tilde{\theta})\ddot{\theta} + C(\tilde{\theta})\dot{\theta} + F\dot{\theta} + f_s.$$  

*) Supported by Ministry for Research and Technology (BMFT-Project SENROB), grant 01 IN 105 A/O
$M$ is an $[n \times n]$ matrix, $C$ is a formal $[n \times n^2]$ matrix, and $\Theta$ is an $[n^2 \times n]$ matrix with the vectors $\dot{\theta}$ in the diagonal and zeros elsewhere, $F$ is an $[n \times n]$ diagonal matrix, and $n$ is the number of joints.

In the present example the friction terms $F$ and $\dot{f}_r$ were idealized in that they are assumed to be independent of position and load. (Simulation and compensation of $f'_r$ are not in the scope of this paper.) Gravity forces are not considered. They are not present in the planar 4JM, but could easily be included in the following considerations.

Fig. 1 depicts the location of the neural controller in the system. (The networks for compensation of friction are omitted for clarity.) Note that the inverse model is employed inside the closed loop, such that the error dynamics can be freely chosen as dynamics of decoupled linear subsystems by setting their PD-parameters. Furthermore, this structure does not necessarily need desired angular accelerations $\theta_d$ at its input, which in many real digital systems are difficult to obtain.

Direct Inverse Training and On-line Adaptation Using 3LP with about 2000 weight parameters for the whole network set, the neural model was first trained as direct inverse [3]. While the robot was PD-controlled the inverse model was trained with the continuously sampled input/output pairs $(\dot{\theta}, \ddot{\theta}, \dot{\theta}, \tau)$. The backpropagation (BP) learning rule was used based on single-step errors. A particular “pruning” method [2] for the network connections was also implemented. 

PTP-trajectories with minimum-jerk profiles, moving the robot between equally distributed random goal points within its working range proved to be well suited to training. They are “classical” robot movements, which represent a good state error distribution and fast enough state changes, so as to guarantee a global function approximation without the danger of “unlearning” at some state regions. Ten different load masses between 0 and 5kg (Robot weight: 4.65kg) were periodically moved by the robot during the training phase.

With such a training the model error became very small over the complete position and load range except for a small position margin. As will be demonstrated later this was numerically checked by systematic measurements. After this identification phase, the weights of the $\hat{M}$ and $\hat{C}$ remained fixed. However, it is generally necessary to keep the robot model adaptive to physical changes that frequently occur during its operation i.e. load mass (and possibly viscous friction) variations. Therefore a new component was added: A single “synapse” $L$, which explicitly carries the value of the load estimate, was connected to the controller’s former load input according to fig.2 (thick lines). $L$ continuously adapted to various loads by “Error-Backpropagation-to-Load-Input”, i.e. $L$ was updated according to the gradient of the squared model error measure relative to $L$. As in [4], the feedback error $\ddot{v}$ was used as model error measure.

$$a_L = \frac{\partial \ddot{v}^2}{\partial \dot{M}} \frac{\partial \dot{M}}{\partial a_L} \frac{\partial a_L}{\partial L}.$$  

$a_L$ is the activation of the load-input unit to the net $\hat{M}$. $\partial \dot{M}/\partial a_L$ thereby is easily determined using BP while maintaining fixed weights of $\hat{M}$ and $\dot{\ddot{v}} = \partial \ddot{v}^2/\partial \dot{M}$ is the usual gradient rule for matrix learning as has already
been applied for the general training. A similar technique for a different purpose was used in [3].

$L_\infty$ Stability Guarantee after the Learning Phase The closed loop system dynamics are:

$$\tilde{\theta} = M^{-1}(\tilde{\theta}) \left( \dot{M}(\tilde{\theta}) \left( \tilde{\theta}' + \tilde{r} \right) + \Delta C(\tilde{\theta}) \dot{\tilde{\theta}} + \Delta F \tilde{\theta} - \Delta \tilde{f}_s \right),$$

(3)

$\Delta C$ and $\Delta F$ are network approximation errors (estimated minus real value).

Following a result for robust stability of model-based robot control [5] a stability criterion for the present controller was derived: An $L_\infty$ stabilizing PD-type controller can be found if upper bounds on $\|M^{-1}\|$, $\|\Delta C\|$, $\|\Delta F\|$ and $|\Delta \tilde{f}_s|$ are found and if:

$$\gamma := \|M^{-1}(\tilde{\theta}) \cdot \dot{M}(\tilde{\theta}) - I\| < 1 \text{ for all } \tilde{\theta}.$$  

(4)

$\| \cdot \|$ is the spectral matrix norm [6]. A brief derivation was given in [2].

The actual problem for it’s application to a neural controller is to prove that $\gamma$ is not only valid at some discrete measurement points $\tilde{\theta}_m$, but that it is valid for all $\tilde{\theta}$ in the robot’s working range, and for all loads in the trained interval, i.e. in a complete closed region. The presented method will guarantee this:

First, $\gamma$ is measured on a coarse regular grid in $\theta_2$, $\theta_3$ and $\theta_4$: Therefore $M^{-1}(\tilde{\theta}_m)$ is determined columnwise, by applying torque vectors $\tilde{r}_m - \tilde{f}_s = (1,0,0,0)^T; (0,1,0,0)^T$... etc. and measuring the accelerations $\tilde{\theta'}$. (Similar measurements are also possible for $C(\tilde{\theta}_m)$). Now, using $M(\theta_m), \gamma(\tilde{\theta}_m)$ is computed.

The whole working range is then divided into boxes $B : |\tilde{\theta} - \tilde{\theta}_m|_\infty \leq \Delta \theta$ around the points $\tilde{\theta}_m$ ($| \cdot |_\infty$ is the maximum vector component). $\gamma < 1$ must thus be shown for all positions $\tilde{\theta}_m + d\tilde{\theta} \in B$.

$$\gamma(\tilde{\theta}_m + d\tilde{\theta}) = \| \left( M^{-1}(\tilde{\theta}_m) + d M^{-1} \right) \cdot \left( \dot{M}(\tilde{\theta}_m) + d\dot{M} \right) - I \| = \gamma(\tilde{\theta}_m) + d\gamma$$

$$d\gamma \leq \| d\dot{M} \| \cdot \| M^{-1}(\tilde{\theta}_m) \| + \| dM^{-1} \| \cdot \| \dot{M}(\tilde{\theta}_m) \| + \| dM \| \cdot \| dM^{-1} \|$$

(5)

To guarantee $\gamma < 1$ it must be shown that $d\gamma \leq 1 - \gamma(\tilde{\theta}_m)$. Bounds on $\| d\dot{M} \|$ can be found using the maximum partial derivatives of the $\dot{M}$ components $\dot{g}_{ij} = [\partial \dot{m}_{ij}/\partial \theta_k], \; k = 2,3,4$ for all $\tilde{\theta} \in B$. For a 3LP $\| d\dot{M} \|$ with output unit weights $w_{ijh}$ and hidden unit weights $u_{hk}$ this is:

$$\left| \frac{\partial \dot{m}_{ij}}{\partial \theta_k} \right| \leq \max \left\{ \sum_h \left( (\sigma'_{max}(\tilde{\theta}))_h \cdot (w_{ijh} u_{hk})_{pos} + (\sigma'_{min}(\tilde{\theta}))_h \cdot (w_{ijh} u_{hk})_{neg} \right) 

- \sum_h \left( (\sigma'_{max}(\tilde{\theta}))_h \cdot (w_{ijh} u_{hk})_{neg} + (\sigma'_{min}(\tilde{\theta}))_h \cdot (w_{ijh} u_{hk})_{pos} \right) \right\}, \; \tilde{\theta} \in B$$

(6)

where e.g. $(w_{ijh} u_{hk})_{neg}$ are the negative weight products, and $(\sigma'_{max}(\tilde{\theta}))_h$ is the maximum derivative of the sigmoid function $\sigma$ of the hidden unit $h$ for all $\tilde{\theta} \in B$. $\sigma'_{max}$ and $\sigma'_{min}$ can be easily computed by summing positive and negative products of $u$ and $\theta$ similarly to eqn. 6. The matrix $\| d\dot{M} \|_{\text{max}}$ with the components $\| d\dot{m}_{ij}\|_{\text{max}} = \sum_k (g_{ij} \cdot \Delta \theta_k)$

(7)

represents the required upper bound on $\| d\dot{M} \|$

Taking the nature of the robot dynamic equations into account and considering that $\dot{M} = C \dot{\theta} + (C \dot{\theta})^T$ (from Lagrange formalism), a bound on $\| d\dot{M}^{-1} \|$ can be established similarly.

The extension of the method to the additional load-input to $\dot{M} (\tilde{\theta}, L)$ is straightforward.

If in the course of the regular grid measurements the so determined deviation bound $d\gamma$ becomes too big in a certain dimension in box $B$, the method provides a grid refinement in this subregion.

Results The machine was simulated as a comparably realistic copy of a real 4JM. For example, friction and stiffness data of the harmonic drive gears was copied from technical data sheets. Gear ratios were 50, 100, 100, 50, respectively. Starting with the base joint, the 4 joint lengths and weights were: (15 cm, 1.7 kg), (15 cm, 1.3 kg), (30 cm, 1 kg) and (17 cm, 0.65 kg). Sampling time is 2 ms.

Fig. 5 shows an example desired trajectory. A reference point is drawn every 30 ms. In fig. 3 the obviously bad performance of an (optimized) PD-controller for a load mass of 3 kg is depicted. Fig. 4 shows the time courses of
the maximum and average of the stability measure $\gamma$, measured on a regular grid, for all positions in the working range (except for a small border zone) and all loads during a 30h training. Although the curve is not monotonous, it is easy to pick out a good quality network and fix its weights. For a the net after 70000 PTP-movements, we applied the above presented method and could indeed guarantee stability for the whole continuous closed working range and all weights $\in [0kg, 5kg]$. Note that $\gamma < 1$ is a sufficient condition. Nets with $\gamma > 1$ do not necessarily lead to unstable control.

Fig. 6 and 7 give an example of on-line load mass adaptation.

The development of the mean squared angular tracking error $MSE$ and the change in the load estimate $L$ during two trajectory executions are visible for a load mass step from $0kg$ to $3kg$. Obviously $L$ quickly adapts to the load. Many load transitions and many arbitrarily chosen trajectories were successfully checked to show the same behaviour.

Discussion and Conclusion A set of neural Networks applied as a controller in a closed feedback loop was rigorously proven to achieve accurate and stable robot control in the whole closed robot working range and for a certain interval of payloads. The analysis method might be too laborious to be applied to a real robot, but by using it we could show that a neural network indeed learned a stable control law under realistic conditions. Hardware implementation and consideration of controller delays are included in our current work. Based on the presented results an application to high accuracy control of real industrial robots appears within reach.

References