NEURAL NETWORK PREDICTIVE TRAJECTORY TRACKING OF AN AUTONOMOUS TWO-WHEELED MOBILE ROBOT [1]

Martin Seyr * **Stefan Jakubek** * **Gregor Novak** **

* *Institute of Mechanics and Mechatronics, Div. of Control and Process Automation, TU Vienna*
** *Institute for Computer Technology, TU Vienna*

Abstract: As a prerequisite for precise trajectory tracking of a two-wheeled mobile robot, accurate control of the velocity and the curvature along a predefined trajectory is vital. After offline training, a neural network is used for nonlinear predictive control. To make the system more robust against modeling inaccuracies and other disturbance influences, the control error is integrated and used to adjust the control variables calculated by the predictor.

Keywords: neural networks, predictive control, autonomous mobile robot, trajectory tracking

1. INTRODUCTION

Extensive research has been done on the subject of trajectory tracking of two-wheeled mobile robots. Most authors concentrate on the non-holonomic problem that arises when side slip and tangential slip of the wheels are neglected.
(Yang and Kim, 1998) use sliding mode control to stabilize the nonholonomic system. (Kim and Tsiotras, 2002) give an overview over different controller structures, most of which are discontinuous, i. e. sliding mode controllers.

The concept of neural network predictive control has been applied to a number of different problems, but not to the problem of trajectory tracking of mobile robots.
(Tan and Van Cauwenberghe, 1996) use a neural network to control several benchmark problems, but only for single-input-single-output-systems (SISO) and only one-step-ahead prediction. (Gil *et al.*, 1999) use a recurrent *Elman*-network with an on-line learning algorithm for nonlinear systems. (Kambhampati *et al.*, 2000) concentrate on stability analysis of a one-step-ahead predictor applied to SISO-systems. (Haley *et al.*, 1999) apply neural network predictive control using a fast *Newton-Raphson* optimization algorithm and a cost function that explicitly accounts for control variable bounds.

This paper investigates the possibility of applying neural network predictive control to mobile robot trajectory tracking. The approach presented can be considered as the inner control loop of a cascading control scheme.
The general principle for neural network predictive control is taken from (Norgaard *et al.*, 1999), modified and adapted for multiple-input-multiple-output-systems (MIMO).

[1] This work is supported by the Austrian Science Fund (FWF), http://www.fwf.ac.at

2. THE PLANT: AN AUTONOMOUS TWO-WHEELED MOBILE ROBOT

The robot, Fig. 1, has two wheels with rubber tires and two felt shoes, one at the front and one at the rear to stabilize it around the pitch axis. The two wheels are supported by ball bearings and powered by two individual DC-motors. A microcontroller produces two pulse-width modulated (PWM) constant voltage signals, which are amplified by a dual full bridge driver. The amplified signals drive the two DC-motors.

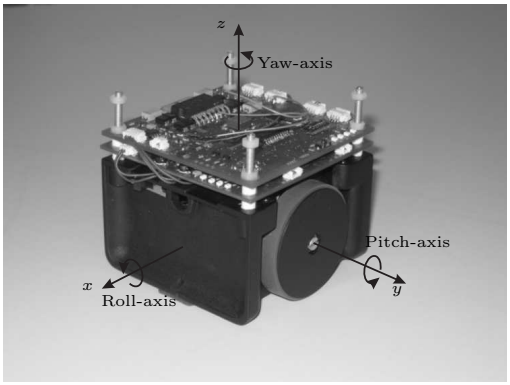

Fig. 1. Autonomous mini-robot Tinyphoon, http://www.tinyphoon.com

Nonlinearities originate from variable switching times and dead zones in the amplifier circuit, friction characteristics of bearings and gearboxes, wheel slip dynamics and the kinematics of a three-degree-of-freedom object moving in a plane.
So the robot can be considered as a nonlinear MIMO-system with two inputs (the PWM-duty cycles) and two respectively three output variables (see section 4.1).

3. NEURAL NETWORK PREDICTIVE CONTROL ALGORITHM

3.1 Cost function

Neural network predictive control is based on the iterative minimization of a quadratic cost-function containing predicted future control errors and future control increments, here denoted for a system with n outputs and m control variables.

$$V(t) = \frac{1}{2h_\mathrm{p}} \hat{\boldsymbol{e}}^T \boldsymbol{L} \hat{\boldsymbol{e}} + \frac{1}{2h_\mathrm{u}} \boldsymbol{\Delta u}^T \boldsymbol{R} \boldsymbol{\Delta u}, \quad (1)$$

$$\hat{\boldsymbol{e}} = \begin{bmatrix} w_1(t+2) - \hat{y}_1(t+2) \\ w_1(t+3) - \hat{y}_1(t+3) \\ \vdots \\ w_n(t+2) - \hat{y}_n(t+2) \\ \vdots \\ w_n(t+h_\mathrm{p}+1) - \hat{y}_n(t+h_\mathrm{p}+1) \end{bmatrix} \quad (2)$$

and

$$\boldsymbol{\Delta u} = \begin{bmatrix} u_1(t+1) - u_1(t) \\ u_1(t+2) - u_1(t+1) \\ \vdots \\ u_m(t+1) - u_m(t) \\ \vdots \\ u_m(t+h_\mathrm{u}) - u_m(t+h_\mathrm{u}-1) \end{bmatrix}, \quad (3)$$

where t denotes the integer sampling instants, $i \in [1; h_\mathrm{p}]$ and $j \in [1; h_\mathrm{u}]$.

- The cost function $V(t)$ has to be evaluated and minimized during each sampling interval.
- Future reference values $w_l(t+i+1)$ up to the prediction horizon h_p must be known.
- Future control variable values $u_k(t+j)$ are optimized up to the control horizon h_u, control variable values beyond the control horizon are assumed to be constant.
- The weight matrices $\boldsymbol{L} \in \mathbb{R}^{nh_\mathrm{p} \times nh_\mathrm{p}}$ and $\boldsymbol{R} \in \mathbb{R}^{mh_\mathrm{u} \times mh_\mathrm{u}}$ determine to what extent the future control variable increments and the future control errors are considered.

The cost function (1) does not require a specific process model. In this paper the application of parallel first order external recurrent feedforward networks as process model is presented, Fig. 2.

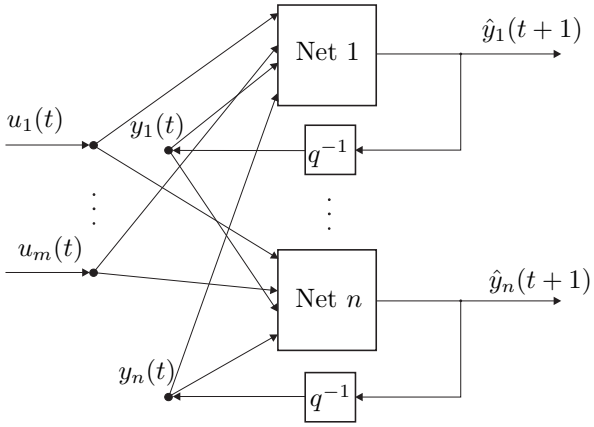

Fig. 2. Parallel networks as process model for a MIMO-system

The general form of such a model is

$$\hat{y}_l(t+1) = f_l(y_1(t), \ldots, y_n(t), u_1(t), \ldots, u_m(t)) \quad (4)$$

where $l \in [1; n]$. Each $f_l, \mathbb{R}^{n+m} \rightarrow \mathbb{R}$ is a static nonlinear map realised by a single layer perceptron network. The networks are trained using a *Levenberg-Marquardt*-Algorithm, with the mean squared error (MSE) as performance criterion.
From $y_l(t)$ and the intended future control inputs $u_k(t)$, $k \in [1; m]$, estimates of the future outputs $\hat{y}_l(t+1+i)$ can be calculated recursively,

$$\hat{y}_l(t+2) = f_l(\hat{y}_1(t+1), \ldots, \hat{y}_n(t+1), u_1(t+1), \ldots, u_m(t+1)). \quad (5)$$

For the initial cycle of the iteration the control variable sequence is chosen to be constant and equal to $u_k(t)$.

3.2 Optimization

The optimization is done using a *Gauss-Newton* algorithm.
After the calculation of an initial estimate of the future outputs the error $\hat{e}_l(t+1+i)$ can be calculated. Next, it is expanded in a first order Taylor series expansion,

$$\hat{\boldsymbol{e}}_l(\boldsymbol{\delta u}_k) \doteq \hat{\boldsymbol{e}}_{l,0} + \frac{\partial \hat{\boldsymbol{e}}_l}{\partial \boldsymbol{u}_k}\boldsymbol{\delta u}_k = \hat{\boldsymbol{e}}_{l,0} - \frac{\partial \hat{\boldsymbol{y}}_l}{\partial \boldsymbol{u}_k}\boldsymbol{\delta u}_k. \quad (6)$$

Its argument $\delta u_k(t+j)$ is a sequence of (small) control variable variations.
For the evaluation of (6) the matrices $\boldsymbol{D}_{lk} \in \mathbb{R}^{h_\mathrm{p}-1 \times h_\mathrm{u}}$ are defined,

$$\boldsymbol{D}_{lk} = \frac{\partial \hat{y}_l(t+1+i)}{\partial u_k(t+j)} \quad (7)$$

The recursive calculation of the derivatives is carried out as follows:

- $i = j$, direct calculation,

$$\frac{\partial \hat{y}_l(t+1+i)}{\partial u_k(t+j)} = \frac{\partial f_l(\hat{y}_1(t+i), \ldots, u_m(t+i))}{\partial u_k(t+j)}. \quad (8)$$

- $i > j$, chain rule,

$$\frac{\partial \hat{y}_l(t+1+i)}{\partial u_k(t+j)} = \sum_{r=1}^{n} \frac{\partial \hat{y}_l(t+1+i)}{\partial \hat{y}_r(t+i)} \frac{\partial \hat{y}_r(t+i)}{u_k(t+j)}, \quad (9)$$

  where the second term is taken from the previous step.
- $i < j$, no influence possible,

$$\frac{\partial y_l(t+1+i)}{\partial u_k(t+j)} = 0. \quad (10)$$

The derivatives in (8) and the first term in (9) are calculated from the structure of the single-layer perceptron network, Fig. 3.

$$\frac{\partial \hat{y}_l(t+1+i)}{\partial u_k(t+j)} = \sum_{s=1}^{p_l} W_s N_s'(o_s(t+i)) w_{\mathrm{u},sk},$$
$$o_s(t+i) = \sum_{k=1}^{m} w_{\mathrm{u},sk} u_k(t+i) + \sum_{l=1}^{n} w_{\mathrm{y},sl} y_l(t+i), \quad (11)$$

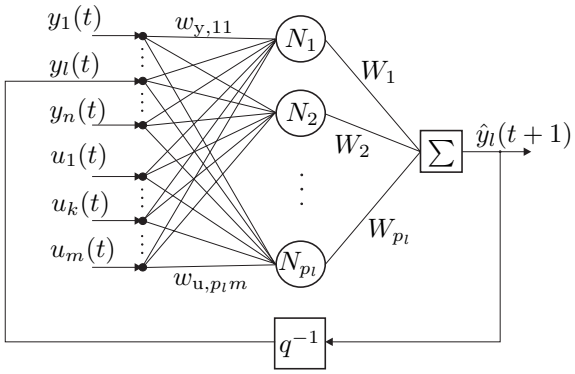

Fig. 3. Structure and nomenclature of an external recurrent single layer feedforward perceptron network for dynamic identification

N_s' is the derivative of the activation function of the s-th neuron with respect to its input o_s and p_l is the number of neurons of the l-th network.
The derivatives with respect to $\hat{y}_r(t+i)$ are calculated in the same way.
Now the Taylor series expansion of the control error (6) and the modified control input are inserted into (1). The expression for the control increment reads

$$\begin{aligned}
&\boldsymbol{\Delta u}_k = \\
&= [u_k(t+1) - u_k(t) + \delta u_k(t+1) \\
&\quad u_k(t+2) - u_k(t+1) + \delta u_k(t+2) - \delta u_k(t+1) \\
&\quad \vdots \\
&\quad u_k(t+h_\mathrm{u}) - u_k(t+h_\mathrm{u}-1) + \\
&\qquad + \delta u_k(t+h_\mathrm{u}) - \delta u_k(t+h_\mathrm{u}-1)]. \quad (12)
\end{aligned}$$

This can be rewritten into

$$\boldsymbol{\Delta u}_k = \boldsymbol{T_1 u}_k + \boldsymbol{T_2 \delta u}_k, \quad (13)$$

where

$$\boldsymbol{T_1} = \begin{bmatrix} -1 & 1 & 0 & & \ldots & 0 \\ 0 & -1 & 1 & 0 & \ldots & 0 \\ & & \ddots & \ddots & & \\ 0 & \ldots & 0 & -1 & 1\ 0 & \ldots\ 0 \end{bmatrix}, \quad (14)$$

$$\boldsymbol{T_2} = \begin{bmatrix} 1 & 0 & & \ldots & 0 \\ -1 & 1 & 0 & \ldots & 0 \\ & \ddots & \ddots & & \\ 0 & \ldots & 0 & -1 & 1 \end{bmatrix}, \quad (15)$$

$\boldsymbol{u}_k = [u_k(t)\ u_k(t+1)\ \ldots\ u_k(t+h_\mathrm{p})]^T$, $\boldsymbol{T}_1 \in \mathbb{R}^{h_\mathrm{u} \times h_\mathrm{p}+1}$ and $\boldsymbol{T}_2 \in \mathbb{R}^{h_\mathrm{u} \times h_\mathrm{u}}$.
The cost function (1) now reads

$$V(t, \boldsymbol{\delta u}) = \frac{1}{2h_\mathrm{p}}(\hat{\boldsymbol{e}}_0 + \boldsymbol{D\delta u})^T \boldsymbol{L}(\hat{\boldsymbol{e}}_0 + \boldsymbol{D\delta u}) + \frac{1}{2h_\mathrm{u}}(\tilde{\boldsymbol{T}}_1 \boldsymbol{u} + \tilde{\boldsymbol{T}}_2 \boldsymbol{\delta u})^T \boldsymbol{R}(\tilde{\boldsymbol{T}}_1 \boldsymbol{u} + \tilde{\boldsymbol{T}}_2 \boldsymbol{\delta u}), \quad (16)$$

where $\boldsymbol{D} \in \mathbb{R}^{n(h_\mathrm{p}-1)\times mh_\mathrm{u}}$ is assembled from the matrices $-\boldsymbol{D}_{lk}$ and $\tilde{\boldsymbol{T}}_1 \in \mathbb{R}^{mh_\mathrm{u}\times m(h_\mathrm{p}+1)}$ and $\tilde{\boldsymbol{T}}_2 \in \mathbb{R}^{mh_\mathrm{u}\times mh_\mathrm{u}}$ are block diagonal matrices containing the sub-matrices $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ respectively in their main diagonals.
The expression (16) is expanded and differentiated with respect to $\boldsymbol{\delta u}$. To minimize (16), the derivative is set equal to zero and the resulting equation is solved for $\boldsymbol{\delta u}$,

$$\boldsymbol{\delta u} = (\frac{1}{h_\mathrm{p}}\boldsymbol{D}^T\boldsymbol{L}\boldsymbol{D} + \frac{1}{h_\mathrm{u}}\tilde{\boldsymbol{T}}_2^T\boldsymbol{R}\tilde{\boldsymbol{T}}_2)^{-1} \cdot$$
$$\cdot(\frac{1}{h_\mathrm{p}}\boldsymbol{D}^T\boldsymbol{L}\hat{\boldsymbol{e}}_0 + \frac{1}{h_\mathrm{u}}\tilde{\boldsymbol{T}}_2^T\boldsymbol{R}\tilde{\boldsymbol{T}}_1\boldsymbol{u}). \quad (17)$$

The control variable variations $\boldsymbol{\delta u}$ are now added to the original control variable sequence $\boldsymbol{u}$. To stabilize the iteration, a stepsize $\mu < 1$ is chosen.

$$\boldsymbol{u}_\mathrm{new} = \boldsymbol{u} + \mu\boldsymbol{\delta u}. \quad (18)$$

With this new control variable sequence $\boldsymbol{u}_\mathrm{new}$ the next cycle of the iteration is carried out. Usually it is sufficient to run through the iteration only a few times.

Remark I Since the optimization is performed during every sampling interval, only the first control variable values are ever actually applied to the system.
Remark II The design parameters are the weight matrices, the number of iterations, the prediction and the control horizon and the stepsize.

4. APPLICATION OF THE ALGORITHM FOR THE GIVEN PROBLEM

4.1 Effective output variables

For trajectory tracking, the translational velocity and the curvature along the path, both as functions of time, are decisive. As a measure for the curvature the acceleration perpendicular to the instantaneous tangent to the path is chosen, because it is bounded as opposed to both the radius and the curvature. The radius becomes infinite for a straight line, the curvature for a sharp turn on the spot. The boundedness of the processed signals is vital for numerical reasons.
From the three output variables normal velocity v_n, tangential velocity v_t and yaw-rate ω two effective output variables, the velocity along the actual path v and the cross-acceleration a are calculated analytically. Fig. 4 illustrates the relationships.

$$v = \sqrt{v_\mathrm{t}^2 + v_\mathrm{n}^2}\,\mathrm{sign}(v_\mathrm{t}), \quad (19)$$

where it is assumed that the side slip angle α,

$$\alpha = \mathrm{atan}(\frac{v_\mathrm{n}}{v_\mathrm{t}}), \quad (20)$$

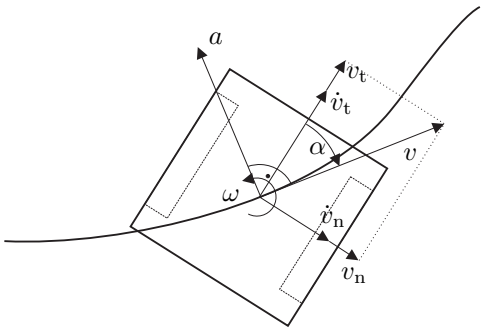

Fig. 4. Kinematics of a two-wheeled mobile robot under consideration of side slip and tangential wheel slip

is bounded in the interval $[-\frac{\pi}{2};\frac{\pi}{2}]$. The cross-acceleration is calculated by

$$a = -(\dot{v}_\mathrm{t} - v_\mathrm{n}\omega)\sin\alpha + (\dot{v}_\mathrm{n} + v_\mathrm{t}\omega)\cos\alpha. \quad (21)$$

To have a simple first order network structure without algebraic loops (as generated by interdependencies of the outputs at the same sampling instant), the chosen network outputs are the three velocities v_n, v_t and ω. The optimization, however, has to be done with respect to v and a as output variables. This means, that additional derivatives are needed.
The derivatives of v and a with respect to the inputs are calculated via the chain rule, e. g.

$$\frac{\partial v(t+1+i)}{\partial u_1(t+j)} =$$
$$= \frac{\partial v(t+1+i)}{\partial v_\mathrm{n}t+1+i}\frac{\partial v_\mathrm{n}(t+1+i)}{\partial u_1(t+j)} +$$
$$+\frac{\partial v(t+1+i)}{\partial v_\mathrm{t}t+1+i}\frac{\partial v_\mathrm{t}(t+1+i)}{\partial u_1(t+j)}. \quad (22)$$

The second part of each term is calculated as described in section 3. The derivatives of the effective outputs v and a with respect to the network outputs are calculated by partially differentiating (19) and (21).
In the expressions for a and its derivatives, $\dot{v}_\mathrm{n}$ and $\dot{v}_\mathrm{t}$ are approximated by the backwards difference quotient,

$$\dot{v}_{\mathrm{n,t}}(t+i+1) \cong$$
$$\cong \frac{1}{T_\mathrm{S}}(v_{\mathrm{n,t}}(t+1+i) - v_{\mathrm{n,t}}(t+i)), \quad (23)$$

where T_S is the sampling time.
The derivatives of $\dot{v}_\mathrm{n}$ and $\dot{v}_\mathrm{t}$ with respect to the control variables can then be written as

$$\frac{\partial \dot{v}_{\mathrm{n,t}}(t+1+i)}{\partial u_{1,2}(t+j)} =$$
$$= \frac{1}{T_\mathrm{S}}\frac{\partial v_{\mathrm{n,t}}(t+1+i)}{\partial u_{1,2}(t+j)} - \frac{1}{T_\mathrm{S}}\frac{\partial v_{\mathrm{n,t}}(t+i)}{\partial u_{1,2}(t+j)}, \quad (24)$$

where $i > j$. When (24) is evaluated for $i = j$ the second term vanishes. In practice, however, it can be seen, that the partial derivatives of $\dot{v}_\mathrm{n}$ and $\dot{v}_\mathrm{t}$ are only meaningful if *both* terms of the approximation (23) are already influenced by the control variable in question, i. e. their derivatives are non-zero.

4.2 Integration of the control error

In addition to the predictive algorithm the control error is integrated and used to further adjust the control variables. This accounts for inevitable modeling inaccuracies and disturbance influences, so as to eliminate any stationary control error.
The additional control law reads

$$u_\mathrm{applied} = u_\mathrm{pred.} + \tag{25}$$
$$+\begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix} \begin{bmatrix} \frac{\partial v(t+2)}{\partial u_1(t+1)} & \frac{\partial v(t+2)}{\partial u_2(t+1)} \\ \frac{\partial a(t+2)}{\partial u_1(t+1)} & \frac{\partial a(t+2)}{\partial u_2(t+1)} \end{bmatrix}^{-1} \begin{bmatrix} \int (v_\mathrm{r} - v)\mathrm{d}t \\ \int (a_\mathrm{r} - a)\mathrm{d}t \end{bmatrix},$$

where v_r and a_r denote the reference values. The gain values k_1 and k_2 are additional design parameters and can be adjusted to obtain the desired level of accuracy on the one hand and damping on the other.

4.3 Control scheme

The tangential and the normal acceleration $\dot{v}_\mathrm{n}$ and $\dot{v}_\mathrm{t}$ are measured by two-axis acceleration sensors, the angular velocity ω is measured by a gyro sensor. The accelerations are integrated at a sampling rate of 1kHz, the control algorithm works with a sampling rate of 50Hz.
A block diagram of the control scheme is depicted in Fig. 5.

The time shift operator q^{-1} symbolises the fact, that a calculated control variable can only be applied at the next sampling instant due to non-zero calculation time.

5. SIMULATION

5.1 Simulation Setup

The simulations are carried out using MATLAB-SIMULINK V7.0 with a physical model of the robot accounting for all beforementioned nonlinear effects.
Two different trajectories are calculated for testing purposes. One is a clothoid with constant velocity after an initial acceleration, the other one is an S-shaped curve with a sinusoidal velocity profile. The duration of both is 5s. The units in all diagrams are SI-compliant, i. e. [m] and [s]. The PWM input signals are given in the interval $[-1; 1]$.

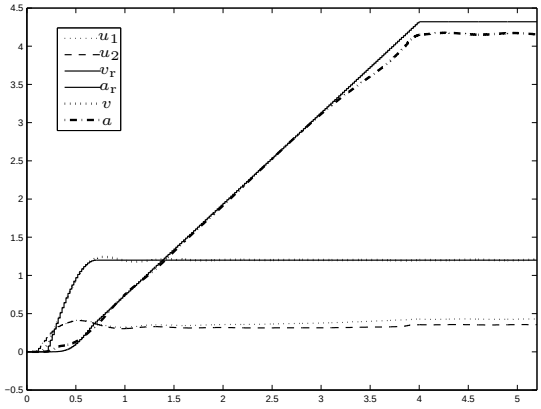

Fig. 6. Reference, control and output variables for tracking of the clothoid

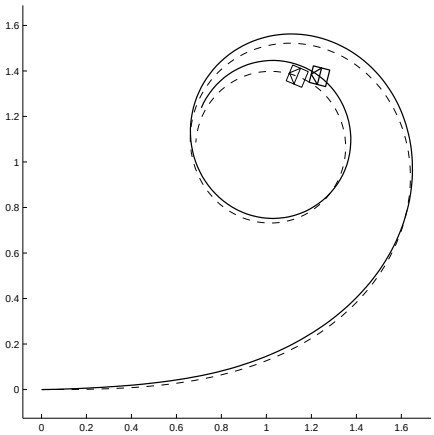

Fig. 7. Reference (dashed) and actual trajectory (solid), frame taken from an animation, clothoid

5.2 Results

First, the two trajectories are applied to the system without any disturbance influences or model perturbations. The results are depicted in Figs. 6 and 7.

Then the robustness against disturbances acting on lateral and tangential slip dynamics is investigated. This setup corresponds to changing ground conditions. In Fig. 8 the lateral slip dynamics are disturbed by a Gaussian white noise signal with a sampling time of 0.4s altering the lateral force by a mean value of 2.5% and a standard deviation of 8.0%.

In Fig. 9 the tangential slip dynamics are disturbed by a Gaussian white noise signal with a sampling time of 0.5s altering the tangential forces by a standard deviation of approximately 2%.

6. CONCLUSIONS

A number of conclusions can be drawn:

- The effect of the side-slip can be seen in Fig. 7. Even though the side-slip angle

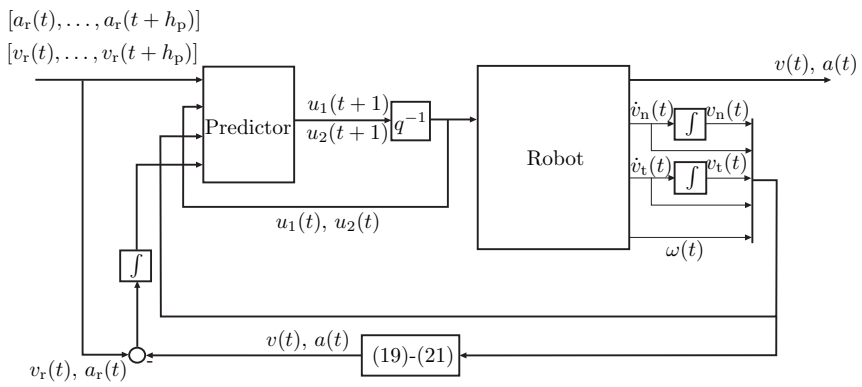

Fig. 5. Combined predictive and integral control scheme

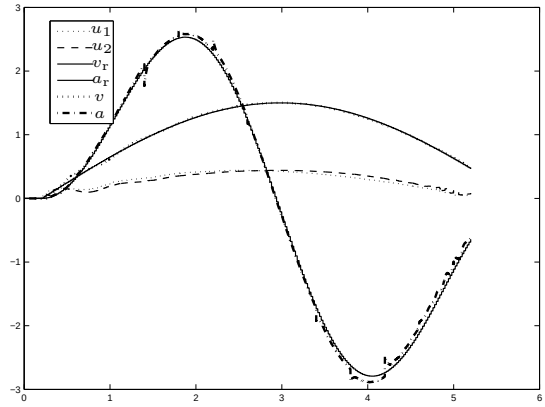

Fig. 8. Reference, control and output variables for tracking of the S-curve, disturbance acting on lateral slip dynamics

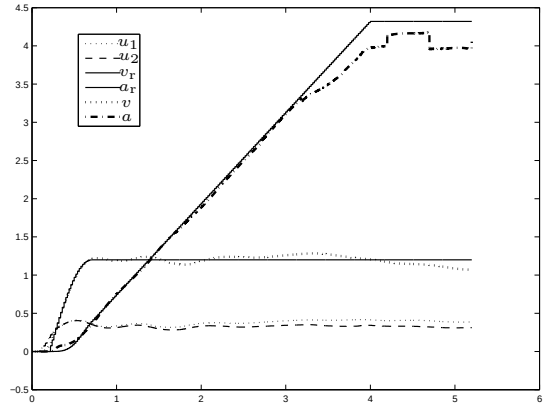

Fig. 9. Reference, control and output variables for tracking of the clothoid, disturbance acting on tangential slip dynamics

reaches values around 30°, the effective curvature and velocity can be maintained.

- The behaviour of the predictive algorithm beyond the input-domain on which the networks were trained is unpredictable. This has to be accounted for during training. Every possible situation during operation has to be included in the training data.
- The algorithm demands considerable calculation time, depending mainly on the control horizon h_u and the number of iteration steps performed.
- Due to the system's remaining sensitivity against plant changes or modeling inaccuracies, modeling has to be done very accurately, and the operating conditions cannot vary arbitrarily.

REFERENCES

Gil, P., J. Henriques, A. Dourado and H. Duarte-Ramos (1999). Non-Linear Predictive Control Based on a Recurrent Neural Network. In: *ERUDIT Conference*.

Haley, P., D. Soloway and B. Gold (1999). Real-time Adaptive Control Using Neural Generalized Predictive Control. In: *American Control Conference*.

Kambhampati, C., J. D. Mason and K. Warwick (2000). A Stable One-Step-Ahead Predictive Control of Non-Linear Systems. *Automatica* **36**, 485–495.

Kim, B. and P. Tsiotras (2002). Controllers for Unicycle-Type Wheeled Robots: Theoretical Results and Experimental Validation. In: *IEEE Transactions on Robotics and Automation.*

Norgaard, M., O. Ravn, N. K. Poulsen and L. K. Hansen (1999). *Neural Networks for Modelling and Control of Dynamic Systems.* Springer.

Tan, Y. and A. Van Cauwenberghe (1996). Nonlinear One-step-ahead Control Using Neural Networks: Control Strategy and Stability Design. *Automatica* **32**(12), 1701–1706.

Yang, J.-M. and J.-H. Kim (1998). Sliding Mode Motion Control of Nonholonomic Mobile Robots. In: *IEEE International Conference on Robotics and Automation, Leuven, Belgium.*