APPLICATION OF NEURAL NETWORKS FOR DYNAMIC MODELING OF ROBOTIC MECHANISMS

Tadej Petrič, Leon Žlajpah

Jožef Stefan Institute
1000 Ljubljana, Jamova 39, Slovenia

tadej.petric@ijs.si(Tadej Petrič)

Abstract

Nowadays the robots must be able perform more and more complex tasks at higher velocities and they must be able to interact with the environment. This can be achieved by using advanced control algorithms, which require exact cancellation of nonlinearities and coupling. For that, exact robot dynamic models are needed. Hence, accurate modeling of the robot manipulator dynamics has remained one of the important issues in robotic. Robot manipulators are highly coupled nonlinear systems and in practice we have to deal also with model parametric uncertainties and obtaining. So, accurate dynamic model is a challenging task. Direct measurements of the robot characteristics are usually impractical or even impossible in many cases. This fact complicates the modeling and identification. The idea is to employ neural networks (NN) for model based control, i.e. to use the ability of neural networks (NN) to represent the non-linear relationship for modeling the robot and to include NN in the control strategy. The proposed model is evaluated in a simulation as well as in real world experiment.

Keywords: Dynamic modeling, neural network, friction effects, gravity compensation.

Presenting Author's Biography

Tadej Petrič attended the Faculty of Electrical Engineering and Computer Science at the University of Maribor, Slovenia, where he obtained the B.Sc degree in Electrical Engineering in 2008. His B.Sc. covered modeling and robotic control of underactuated dynamic system. For his work, he received the prof. dr. Vratislav Bedjanič award in 2008. He is currently a researcher at the Department for Automation, Biocybernetics and Robotics at the Jožef Stefan Institute and a PhD student at the Faculty of Electrical Engineering at the University of Ljubljana.

1 Introduction

In recent years many control algorithms that include NN [1], which can compensate the nonlinearities in the system, have been developed. The general mapping and learning properties are the main advantage of NN [2], compared to the existing general methods, since they can learn the system characteristic without knowing the model structure in advance [3]. This learning capability is used to learn a certain function [4], e.g. highly nonlinear function, direct dynamic, inverse dynamic or. any other characteristic of the system.

The learning process can be done in two ways: the off-line and the on-line method. Using the off-line approach, the NN learns through some learning data sets without interacting with the robot. Using the second method, the NN tries to adapt on-line during the learning phase. This is usually preformed during a normally long training period, when the robot system is controlled by some independent supervised control [5]. If the learning process of NN is not switched off after the learning phase, the NN-based controller works as an adaptive controller [6].

In this paper, we present the application of a NN for modeling of the robot dynamics and incorporating it in the control strategy. This is possible due to the ability of training NN to learn both a system input-output relationship or its corresponding inverse relationship. Because, the torque depends only on the current state of the robot, i.e. acceleration, velocity and position of each joints, a static NN can been used.

Due to the analytically complex determination of the dynamic model, and initially referred properties of neural networks, we propose to replace the dynamic model with a NN, where the inputs into the network are joint positions $\boldsymbol{q}$ and velocities $\dot{\boldsymbol{q}}$ and outputs are compensating joint torques $\boldsymbol{\tau}$. In this approach, the inertia term has not been included into the NN because acceleration are usually not available on real robots. Hence, we have identified only the position and velocity dependent terms, i.e. the joint friction and gravity forces.

For friction identification we propose a method, where the system has to learn not only the friction parameters but also the shape and properties of friction torque. Considering only the torques which appear due to the friction, we can separate the problem into the identification of friction in individual axes.

When the robot kinematics is known, gravity compensation is usually analytically and computationally undemanding. However, the model parameters like masses and mass center points are sometimes very hard to obtain. In such a case, the gravity torque compensation requires advance identification algorithms for identifying the gravity influence. Gravity compensation is a multidimensional problem, which depends on the number of DOFs. The interaction between separate axes is significant and can not be avoided by separating the problem into individual axes, as in case of friction compensation. Therefore, it is necessary to use multiple dimension inputs and outputs for the NN.

To verify the proposed approach we tested it first using simulation, where the robot model and its parameters are known. Simulation results showed that NN based model for friction and gravity compensation can effectively compensate nonlinearities in the robotic system. Next we have implemented the algorithm on a real system. We have compared the behavior of the robot with and without the NN compensator.

The paper is organised as follows. In section 2, we give a brief description of the dynamics of rigid body. In section 3 the proposed neural network structure and learning algorithm are given. In section 4 the identification, simulation and real world results are given. And finally the conclusions are given in section 5

2 Robot dynamics

Robotic control based on a computed torque algorithm is a challenging problem, since the robotic system is non-linear and highly coupled [7]. To cope with this problems, a general theory of linear systems can not be directly applied [8]. However, to solve these problems, two methods have been developed: Lagrange-Euler and Newton-Euler formulation. Block diagram of the robot dynamic is shown in Fig. 1.

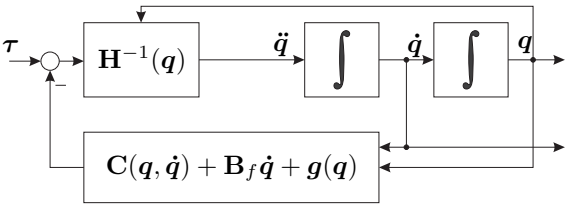

Fig. 1 Block diagram of the robot model

The robot dynamics, which describes the relationship between the joint torques $\boldsymbol{\tau}$ and positions $\boldsymbol{q}$, is given by the Lagrangian equation

$$\boldsymbol{\tau} = \mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \mathbf{B}_f\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}), \qquad (1)$$

where $\mathbf{H}(\boldsymbol{q})$ is the inertial matrix given by

$$h_{j,k} = \sum_{i=max(j,k)}^{I} \text{Trace}[(\frac{\partial}{\partial q_k}\mathbf{T}_i^0)\mathbf{J}_i(\frac{\partial}{\partial q_j}\mathbf{T}_i^0)^T],$$

where I is the number of DOFs. Next, $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is the matrix of Coriolis and centrifugal contributions, defined using Christoffes symbols as

$$c_j = \sum_{k=1}^{I}\sum_{l=1}^{I} c_{j,k,l}\dot{q}_k\dot{q}_l,$$

$$c_{j,k,l} = \sum_{i=max(j,k,l)}^{I} \text{Trace}[(\frac{\partial^2}{\partial q_k \partial q_l}\mathbf{T}_i^0)\mathbf{J}_i(\frac{\partial}{\partial q_k}\mathbf{T}_i^0)^T],$$

where $j, k, l = 1, ..., I$, $\mathbf{J}_i$ is the pseudo inertia tensor and $\mathbf{T}_j^i$ is the homogenous transformation matrix between frame i and frame j. Next, $\mathbf{B}_f$ is the friction

and finally $\boldsymbol{g}(\boldsymbol{q})$ is the vector of the gravity contribution given by

$$g_j = \sum_{i=j}^{I} (-m_i \boldsymbol{g} (\frac{\partial}{\partial q_j} \mathbf{T}_i^0) r_i).$$

Here m_i is the mass of the i-th link and r_i contains the homogeneous coordinates of the center of mass of link i expressed in the i-th coordinate frame

$$r_i = [x_i \; y_i \; z_i \; 1]^T.$$

Element $\boldsymbol{g}$ represents the gravity vector $\boldsymbol{g} = [0\,0\,-g\,0]$, where $g = 9.8\; m/s^2$.

To compensate the nonlinearities and the coupling, the control law is given in the form

$$\boldsymbol{\tau}_u = \hat{\mathbf{H}}(\boldsymbol{q})\boldsymbol{u} + \hat{\mathbf{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \hat{\mathbf{B}}_f \dot{\boldsymbol{q}} + \hat{\boldsymbol{g}}(\boldsymbol{q}) \qquad (2)$$

where $\hat{}$ represents the computed model, which is in general not equal to the exact model. The outer loop control law $\boldsymbol{u}$ is equal to

$$\boldsymbol{u} = \ddot{\boldsymbol{q}}_d + \mathbf{K}_d \dot{\boldsymbol{e}} + \mathbf{K}_p \boldsymbol{e}. \qquad (3)$$

Here $\boldsymbol{e}$ is the difference between the desired and the actual value joint positions, ($\boldsymbol{e} = \boldsymbol{q}_d - \boldsymbol{q}$), $\mathbf{K}_p$ and $\mathbf{K}_d$ are gain matrices. From Eq. (2) it follows that exact feedback compensation is possible only if the model equals to the real system. In this case, the close-loop error is

$$\ddot{\boldsymbol{e}} + \mathbf{K}_d \dot{\boldsymbol{e}} + \mathbf{K}_p \boldsymbol{e} = 0 \qquad (4)$$

and with properly selected gains $\mathbf{K}_d$ and $\mathbf{K}_p$ the asymptotic stability of the system can be assured.

Due to the analytically complex determination of the dynamic model, and initially referred properties of neural networks, we propose to replace the dynamic model described by Eq.2 with a NN. The proposed controller based on NN dynamic compensation is given as

$$\boldsymbol{\tau}_u = \hat{\mathbf{H}}(\boldsymbol{q})\boldsymbol{u} + \mathrm{FFN}(\boldsymbol{q}, \dot{\boldsymbol{q}}), \qquad (5)$$

where the $\mathrm{FFN}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is a static neural network. Inputs into the network are joint positions $\boldsymbol{q}$ and velocities $\dot{\boldsymbol{q}}$ and outputs are compensating joint torques as shown in Fig. 2.

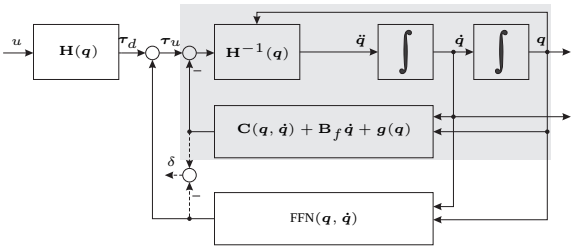

Fig. 2 Block diagram of the robot model (gray) with the compensation part based on NN

As we can see in Fig.2 the inertia term has not been included into the NN because accelerations are usually not available on real robots. Hence we have identified only the position and velocity dependent terms, i.e. the joint friction and gravity forces.

3 Neural network architecture and learning algorithm

In this section we described two-layered structure of a NN, which is used for modeling of the robot dynamics and can be incorporated into the control strategy. We propose to use a static two-layered feedforward NN because in theory the torque depends only on the current state of the robot, i.e. the acceleration, velocity and position of each joints. This NN are usually trained in the off-line mode.

This kind of a NN architecture is most commonly used with the backpropagation algorithm - the multilayer feedforward network. The backpropagation is the generalization of the Widrow-Hoff learning rule to multiple-layer networks and nonlinear differentiable transfer functions [2]. Input vectors and the corresponding target vectors are used to train a network until it can approximate a function.

In [3] authors have shown that any nonlinear function can be approximated with an arbitrary precision using a two-layer NN with biases, a sigmoid layer, and a linear output layer. In this case the approximation error depends only on the size of the number of hidden neurons. The standard backpropagation is a gradient descent algorithm in which the network weights are moved along the negative of the gradient of the performance function [2].

Properly trained backpropagation (feedforward) networks tend to give reasonable results when presented with a new inputs that they have never seen. Typically, such input leads to an output similar to the correct output for input vectors used in training that are similar to the new input being presented [2]. Therefore, it is possible to train a network on a representative set of input and corresponding outputs pairs and to get good results without training the network on all possible pairs.

The structure of a feedforward NN that was used for modeling of the robot dynamics is given in Fig. 3. According to [2], this NN can be used as a general function approximator. Since it can, with sufficient neurons in the hidden layer, approximate any function with a finite number of discontinuities.

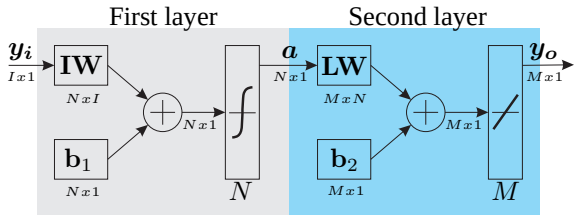

Fig. 3 Structure of the proposed NN for modeling the robot dynamics. The description is given in the text

The basic structure of the NN presented in Fig. 3 has I inputs, N neurons in the hidden layer and M neurons in the output layer. According to [2], the neurons in the

hidden layer are given with

$$\boldsymbol{a} = \frac{1}{1 + e^{(\mathbf{IW}\boldsymbol{y_i} + \boldsymbol{b}_1)}} - 1, \tag{6}$$

where the $\boldsymbol{y_i}$ is input vector, $\mathbf{IW}$ is the weight matrix and $\boldsymbol{b}_1$ is the weight vector. Next, the output layer is given with

$$\boldsymbol{y_o} = \mathbf{LW}\boldsymbol{a} + \boldsymbol{b}_2, \tag{7}$$

where the $\boldsymbol{y_o}$ is the system output, $\mathbf{LW}$ is the output layer gain matrix and $\boldsymbol{b}_2$ is the output layer gain vector.

Once the weights and biases are determined for the feedforward NN, the network is ready for the training. The training process requires a set of example inputs and corresponding targets pairs. During the training the weights and biases of the network are iteratively adjusted to minimize the NN performance function [2]. Usually the default performance function for feedforward networks is the mean square error defined as

$$E_{mse} = \frac{1}{N} \sum_{i=M}^{M} (e_i)^2. \tag{8}$$

where N is the number of inputs and corresponding targets pair and e_i is the difference between the targets and NN outputs and is not refereed to the e form Eq. 4.

Several different training algorithms for the feedforward NN exist for minimizing the preformance function. All these algorithms use the gradient of the performance function to determine how to adjust the weights [2]. The gradient is determined with a technique called the backpropagation.

In our case, we use an algorithm called Levenberg-Marquardt [9] to optimise the weights. This method was developed in order to learn the weights with the second order convergence, without computing the Hessian matrix [10] directly. A detailed description of the above algorithm is given in [9].

Structures of NN and methods presented in this section are included in Neural Network Toolbox for Matlab/Simulink and can therefore be used out-of-the box.

4 Identification and control examples

The main part of our experimental system is a laboratory manipulator developed specially for testing different control algorithms. The manipulator has four revolute DOFs acting in a plane (Fig 4). As the task space is two-dimensional (x-y) the manipulator has two redundant DOFs. In [11], authors presented the exact dynamic model of this robot suitable for simulation. This enables us to test proposed NN structure in both simulation and real-world experiment. Hence, we can compare the behavior of the control system if ideal models for friction or gravity compensation are used to the models based on NN.

As we already mentioned, the inertia term has not been included into the NN because acceleration is not available on this robot. Based on our experience the acceleration contribution can be neglected if the experiments

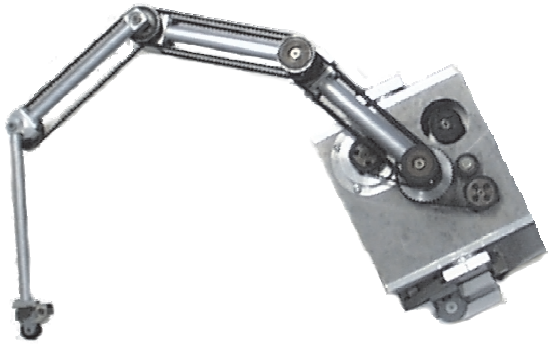

Fig. 4 Experimental planar manipulator

are done with low and constant velocities. Consequently, only the position and velocity dependent terms are included in the model, i.e. the joint friction and gravity forces.

4.1 Friction identification - ($\mathbf{B}_f\dot{\boldsymbol{q}}$)

In recent years many models for friction in robotic joints were developed, e.g. [7, 12]. Usually their common feature is that the friction model is predefined and only the parameters are identified. However, if NNs are used to model the friction, the system has to learn the properties and the shape as well. Such an approach is called a black box principle. Because the friction in a specific joint is usually not cupped with other joints, we can separate the problem into individual axes.

To determine the friction-velocity relationship for the joints of a planar manipulator, each joint of the robot was commanded to move at a constant torque and the velocity at that torque was measured. The Simulink model for collecting the data is presented on Fig. 8. We assumed that the command torque corresponds to the friction for the measured velocity. To characterise the friction behavior, the velocity data was divided between -5 rad/s and 5 rad/s with 0.05 rad/s increments. For each increment the mean torque value was calculated. The acquired data, a total of 241 torque-velocity pairs per joint, was used for learning the NN.

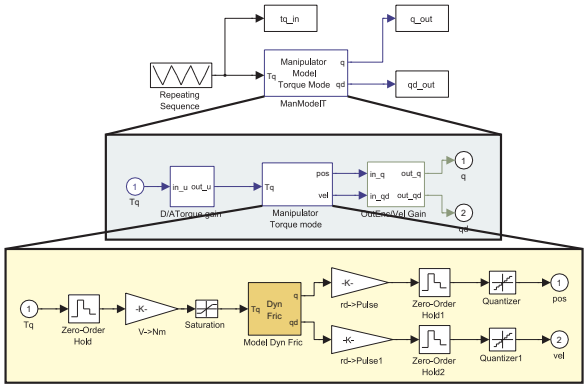

Fig. 5 Simulink block diagram of the planar manipulator

The best fit approximation of experimental data of the

NN is shown in Fig. 6, for both negative and positive velocities and for all 4 joint. As shown in Fig. 6 all 4 joints have a linear relationship between the friction torques and the velocity. It is a typical combination of Coulomb and viscous friction (the model used in the planar manipulator dynamic model).

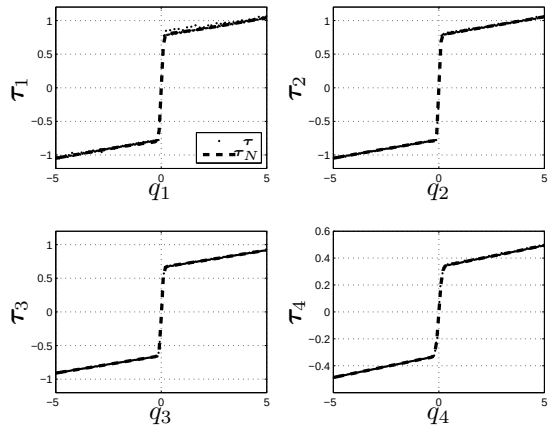

Fig. 6 Velocity-dependent friction for all four joints approximated with NNs in simulation. The friction data along with the NN curves are shown for positive and negative velocities

However, as it can be seen form real experiments in Fig. 7 joints 2 and 3 exhibited varying degrees of decreasing viscous with increasing velocity. Therefore, it is hard to determine the basic shape for friction model, which can be used in all four joints. Hence, we propose a NN to model the friction, since it can easily cope with non-symmetrical curve as shown in the bottom right-hand plot on Fig.7.

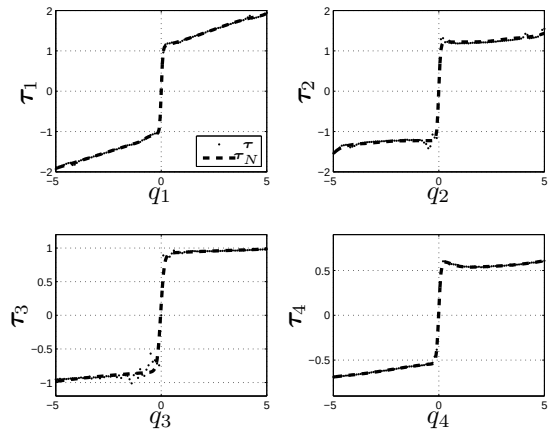

Fig. 7 Velocity-dependent friction for all four joints approximated with NNs in real experiment. The friction data along with the NN curves are shown for positive and negative velocities

To verify our friction model for the planar manipulator, we fed-forward the torques computed by our friction models. Fig. 8 shows the Simulink block structure of

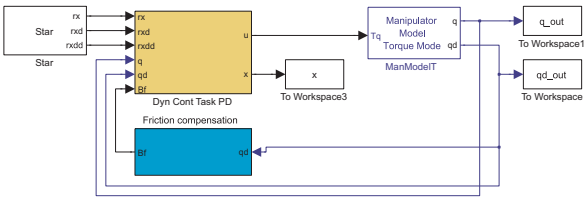

Fig. 8 Simulink block diagram of the task space control for a planar manipulator with friction compensation

the system with the control algorithm defined as

$$\boldsymbol{u} = \mathbf{J}^{\#}(\ddot{\boldsymbol{x}}_d + \boldsymbol{K}_d(\dot{\boldsymbol{x}}_d - \dot{\boldsymbol{x}}) + \boldsymbol{K}_p(\boldsymbol{x}_d - \boldsymbol{x})) - \dot{\mathbf{J}}\boldsymbol{q},$$

where $\ddot{\boldsymbol{x}}$, $\dot{\boldsymbol{x}}$, $\boldsymbol{x}$ are the acceleration, velocity and positions in task space, $\mathbf{J}^{\#}$ is the Moor-Penrose generalized inverse, K_p, K_d are gains with empirically determine values 1000 and 160. The reference trajectory $\boldsymbol{x}_d$ has been selected as

$$\begin{aligned} x_{d1} &= 0.2(\cos(t) + \sin(2t)), \\ x_{d2} &= 0.15(\sin(t) + \cos(2t)). \end{aligned}$$

In the first three steps, we have made a simulations using the complete model of the planar manipulator including Columbic and viscous friction. The model was tested with an ideal friction compensation (Perfect comp.), with a friction compensation using NN (Comp.) and without any friction compensation (No comp.) as shown in Fig. 9. For the NN friction model the results presented in Fig. 6 were used. We can see that the mean square error (MSE) between the desired and the actual value ($(\boldsymbol{x}_d - \boldsymbol{x})^2$) is in the case of perfect model and of the NN based friction model close to zero. However, the in case, where the friction was not compensated the MSE was significant (see the bottom plot on Fig. 9).

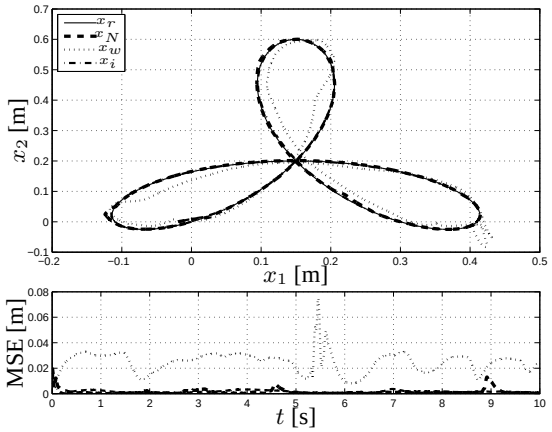

Fig. 9 Tracking results from simulation in the top plot, where x_r is the reference trajectory (-), x_N are results using NN friction model (–), x_w are results without compensating friction (...) and x_i are the results with ideal friction compensation (-.-). The same is valid for the bottom plot where the MSE is shown

In the fourth and the fifth step we have used the same approach, excluding the approach with the prefect friction compensation, which is on a real system impossible to obtain. For the NN based friction model the results presented in Fig. 7 were used. The positions and tracking MSE are shown in Fig. 10. Again we can see, that in the case, where friction model was not used, the tracking results are poor.

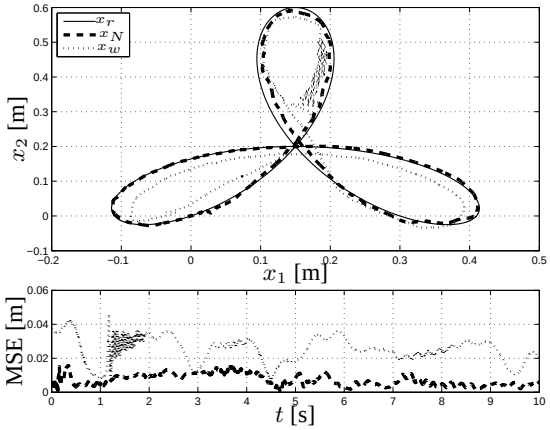

Fig. 10 Tracking results from real experiment in the top plot, where x_r is the reference trajectory (-), x_N are results using NN friction model (–) and x_w are results without compensating friction (...). The same is valid for the bottom plot where the MSE is shown

By comparing the simulation and real experiment average MSE we can conclude that the results are similar in both cases for included compensations as well as without the compensation (see Tab. 1). Furthermore, the average MSE between perfect friction model and NN based friction model, are almost identical. Hence, the model based on NN can reproduce the exact friction behavior.

Tab. 1 Average MSE error for tracking experiments with different friction models for friction compensation

	Simulation			Real robot	
	Perfect comp.	Comp.	No comp.	Comp.	No comp.
MSE	0.0014	0.0016	0.0248	0.0071	0.0291

4.2 Gravity identification - ($\mathbf{g}(q)$)

When the robot kinematics is known in advance, the calculation of the gravity compensation torque is usually analytically and computationally undemanding. The parameters used for the gravity compensation are usually taken from technical documentation or determined through least-square based techniques. These different techniques are usually difficult to implement due to the limitations, e.g. the kinematic error and the compliance in the harmonic drive system could not be directly measured without an output axis resolver measurement. Therefore, such techniques always include some error due to the lack of sufficient information to full characterise the state of the system.

The gravity compensation is a multi-dimensional problem, which depends on the number of DOFs. The interaction between separate axes is significant and can not be avoided by separating the problem into individual axes as in the case of the friction compensation. Therefore, it is necessary to use multiple dimension inputs and outputs for the NN. To demonstrate the applicability of the proposed NN structure we a use planar manipulator (Fig.4) where the base was rotated for 90 degrees in such a way that the gravity effect was significant for all four joints.

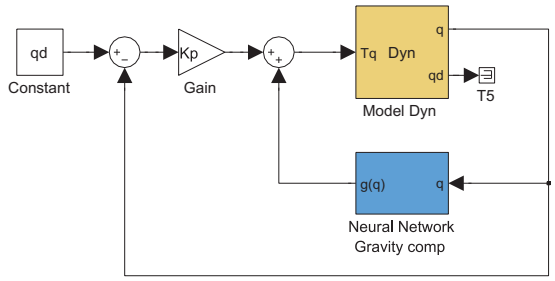

Fig. 12 Simulink block diagram of the joint control of a planar manipulator with gravity compensation

To verify our NN base gravity model for the planar manipulator, we fed-forward the compensation torques. Fig. 12 shows the Simulink block structure of the system with control algorithm defined as

$$\boldsymbol{\tau}_u = \hat{\boldsymbol{g}}(\boldsymbol{q}) + K_p(\boldsymbol{q}_d - \boldsymbol{q}),$$

where $\boldsymbol{q}_d$ is the desired joint angle, $\hat{\boldsymbol{g}}(\boldsymbol{q})$ is the gravity model based on NN and K_p is a gain for position controller. The value of $K_p = 0.025$ was determined empirically.

To get the required data for learning the NN the following assumptions ware made: the velocity is constant, acceleration is zero, and the term ($\mathbf{B}(q)\ddot{q}$) in Eq. 2 can be neglected. To decrease the friction effects we have averaged the torques for up and down movements.

To determine the gravity relationship for the joints of a planar manipulator, each joint of the robot was commanded to move at a low constant velocity up and down at different angles (0 to 2π with a step of $\frac{2\pi}{7}$). For each movement the mean torque value was calculated using the above assumptions. The acquired data, a total of 7^4 torque-velocity pairs per joint, was used for learning the NN, with 4 inputs and 4 outputs. For this particular case, a 25 hidden neurons in NN was used. The number of hidden neurons was determine empirically.

The simulation results of the ideal gravity compensation and gravity compensation based on NN are given in Fig. 14. We can see that the NN based gravity model behaves almost exactly as the ideal model. This shows that model based on NN is able to successfully identify the exact gravity torques. If we compare this results with the results, where the gravity compensation was excluded we can see that the P-control-loop is not able to compensate the gravity effects, and therefore the error between the desired and the actual joint angle was

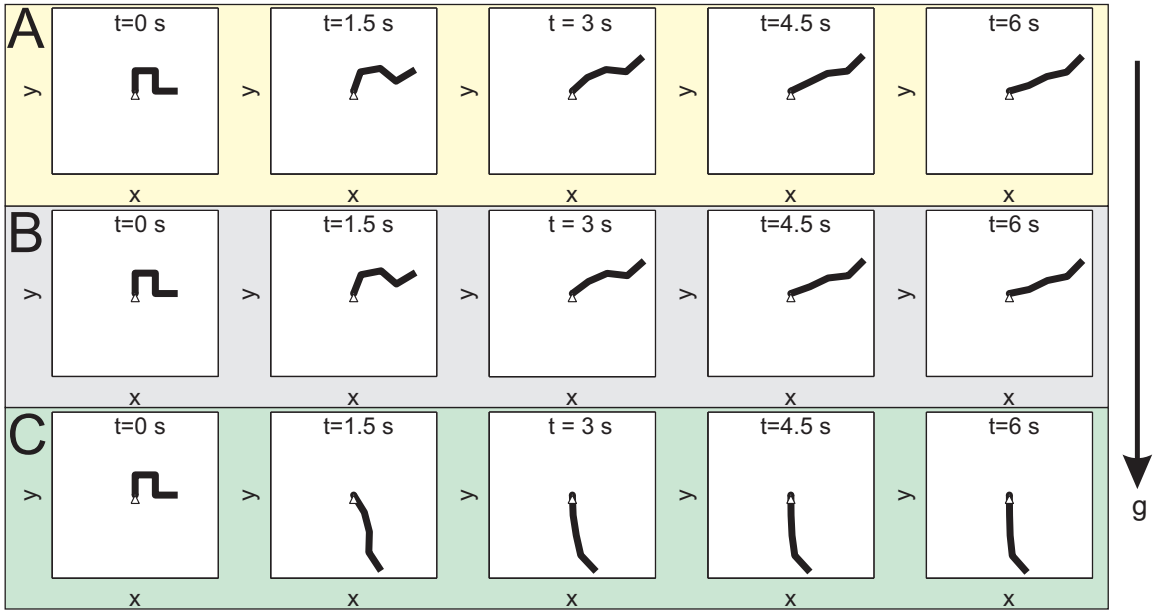

Fig. 11 Time sequence of simulated joint tracking results for all four joints with: A) ideal gravity compensation, B) gravity compensation based on NN and C) no gravity compensation

significant (see Fig. 13). In Fig. 11 the results are given for three cases: A) ideal gravity compensation, B) gravity compensation based on NN and C) without gravity compensation.

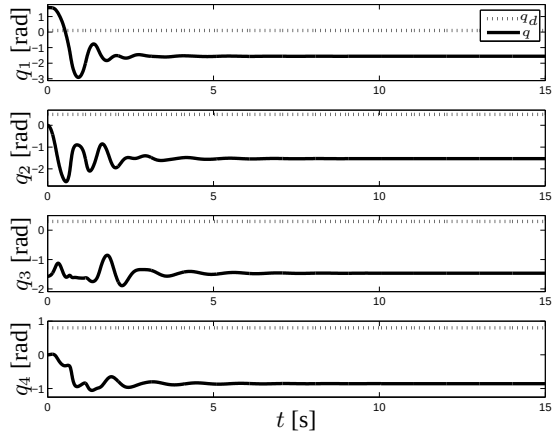

Fig. 13 Simulated joint tracking results for all four joints without gravity compensation

We have shown that, by separate approach for determining the compensation torques for friction and gravity, the NN can accurately reproduce the behavior of friction effects or gravity effects. However, if we would like to identify the complete model with a single NN, huge data set for learning would be required. The size of the leaning data set would be at least $x^{(a*DOF)}$, where x is the number of measurements for a single joint and a is the number of measured quantity for each joint, e.g. the acceleration, the velocity and the position.

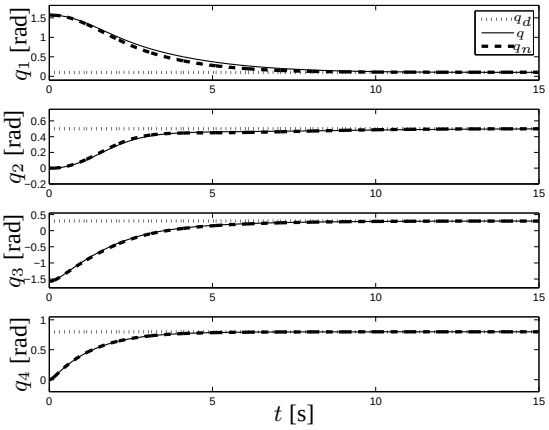

Fig. 14 Simulated joint tracking results for all four joints with ideal gravity compensation (-) and with gravity compensation based on NN (–)

5 Conclusion

In this paper we presented an approach for the modeling and the identification of the dynamic model based on neural networks. Presented experiments showed that the statical neural networks can be efficiently used for learning the properties of the robotic system.

Although, the use of the neural network was successful in all experiments, only the identification of friction shows practical usefulness. Namely, the learning data set for a robot manipulator with many degrees-of-freedom, may become very huge and therefore it is not always possible to obtain all the data required for the off-line NN learning process.

Our further work will focus primarily on the search for

suitable adaptive neural networks, which will be useful for modeling the dynamics of the robot during the operation.

6 References

[1] Daniel H. Patino, Ricardo Carelli, and Benjamin R. Kuchen. Neural networks for advanced control of robot manipulators. *IEEE Transactions on Neural networks*, 13(2):343–354, 2002.

[2] Howard Demuth, Mark Beale, and Martin Hagan. *Neural Network Toolbox 6 Users Guide*. The MathWorks, Inc., 3 Apple Hill Drive, 2009.

[3] Kumpati S. Narendra and Kannan Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural networks*, 1(1):4–26, 1990.

[4] Gu Fang and M.W.M.G. Dissanayake. Neural networks for modelling robot forward dynamicsr. In *Neural Networks, 1995. Proceedings., IEEE International Conference*, number 5, pages 2715 – 2719, 27 November - 01 December 1995.

[5] Sahin Yildirim. Adaptive robust neural controller for robots. *Robotics and Autonomous Systems*, 46(3):175–184, 2004.

[6] Christian Döschner. Modeling of robot dynamics based on a multi-dimensional rbf-like neural network. In *ICIIS '99: Proceedings of the 1999 International Conference on Information Intelligence and Systems*, page 180, Washington, DC, USA, 1999. IEEE Computer Society.

[7] Christopher W. Kennedy and Jaydev P. Desai. Model-based control of the mitsubishi pa-10 robot arm: Application to robot-assisted surgery. In *ICRA*, pages 2523–2528, 2004.

[8] Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer, 2008.

[9] M. T. Hagan and M. B. Menhaj. Training feedforward networks with the marquardt algorithm. *Neural Networks, IEEE Transactions on*, 5(6):989–993, 1994.

[10] Eric W. Weisstein. Hessian. Technical report. Available as: http://mathworld.wolfram.com/Hessian.html.

[11] L. Zlajpah. Simulation of n-r planar manipulators. *Simulation Practice and Theory*, 6:305–321, 1998.

[12] Nikolaos A. Bompos, Panagiotis K. Artemiadis, Apollon S. Oikonomopoulos, and Kostas J. Kyriakopoulos. Modeling, full identification and control of the mitsubishi pa-10 robot arm. In *Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, September 2007.