Feedforward Control for a Manipulator with Flexure Joints Using a Lagrangian Neural Network

Eline Heerze, Bojana Rosic, and Ronald Aarts(✉)

Applied Mechanics and Data Analysis, University of Twente, Enschede, The Netherlands
elineheerze@hotmail.com, {B.Rosic,R.G.K.M.Aarts}@utwente.nl

Abstract. Feedforward control of a manipulator can be generated with a sufficiently accurate stable inverse model of the manipulator. A Feedforward Neural Network (FNN) can be trained with experimental data to generate feedforward control without knowledge about the system at hand. However, the FNN output can show unphysical behaviour especially in operational regimes where the training data is sparse. Instead, the output of a Lagrangian Neural Network (LNN) is limited by physical constraints and hence is expected to predict the (inverse) multibody system behaviour more robustly. We propose to generate the feedforward control by first training a LNN that captures already most features in experimental data and next add a FFN to account for a relatively small residual. Experimental results from a fully actuated 2-DOF manipulator with flexure joints show that the accuracy of the controlled motion using this approach is comparable to using an identified inverse plant model built from the system's equations of motion.

1 Introduction

Feedforward control can greatly improve the accuracy of a manipulator. In a typical implementation an inverse dynamic model of the multibody system at hand is used to compute the required actuator input to follow the reference trajectory. The achievable performance gained from this feedforward control depends heavily on the correctness and completeness of the model. In a *model-based* approach a *white-box* model with the equations of motion of the multibody system is derived and its parameters are estimated [6,10]. Assuming these parameters have a clear physical meaning, it is expected that the model can be used for a wide variety of trajectories. However, the "richness" of the model is obviously restricted to the features included in the model structure.

Alternatively, in a *data-driven* approach a *black-box* model is identified purely from data e.g. using machine learning techniques. Abdul-hadi [1] presents a Feedforward Neural Network (FNN) to learn the dynamic behaviour of a robotic system without any knowledge about the system dynamics and its parameters. It proved to be feasible to solve this problem with reasonable accuracy. However,

K. Nachbagauer and A. Held (Eds.): IUTAM 2022, IUTAM Bookseries 42, pp. 130–141, 2024.
https://doi.org/10.1007/978-3-031-50000-8_12

care has to be taken to avoid overfitting and likely the model outputs are incorrect for operating conditions that were not sufficiently excited in the training data.

To mitigate this risk physics informed neural networks (PINN) are being researched, where the training is constrained to a predefined physical law. Lutter *et al.* [8,9] propose a Lagrangian Neural Network (LNN), or Deep Lagrangian Networks (DeLaN), to incorporate the Lagrangian dynamics into a neural network. The authors could obtain an inverse dynamic model of a robotic system of which the accuracy is similar to the performance of an analytical model. Furthermore, the trained model can handle extrapolations to new trajectories, e.g. with increased velocities. One drawback of this LNN is the exclusion of non-conservative forces like damping and friction in the Lagrangian formulation which can result in significant errors when applied to real physical systems. Liu *et al.* [7] extend an LNN with a parallel FNN to approximate non-conservative physics. Both neural networks are trained simultaneously, where the unconstrained FNN is penalised to discourage it from learning conservative dynamics. However, it was found that the results strongly depend on the penalisation factor.

In this paper we research the use of combining a LNN and FNN to learn an inverse dynamic model of a manipulator with flexure joints [3]. For such manipulator it is expected that the conservative contribution dominates. Hence it is proposed to train the neural networks sequentially: First the LNN should capture the dominant conservative dynamic behaviour and next the FNN is trained with the relatively small residue. The outputs of both networks are added to estimate the actuator forces needed to perform a specified motion. This estimation is applied as feedforward control and the improvement of the motion accuracy is compared to results obtained with feedforward control computed with a white-box manipulator model.

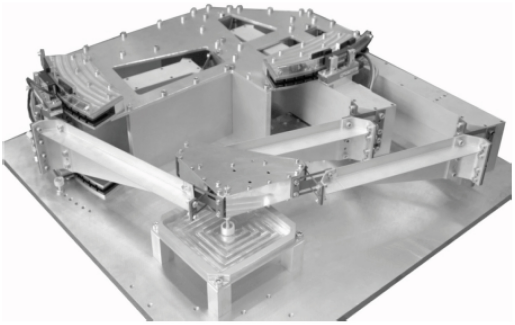
2 Method

2.1 2-DOF Manipulator with Flexure Joints

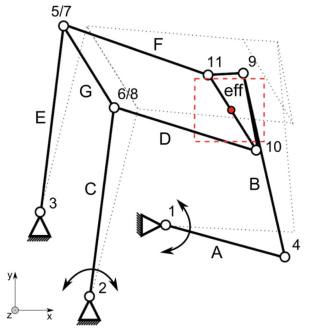
We consider the fully actuated manipulator with two degrees of freedom (DOF) shown in Fig. 1(a) [3]. The schematic drawing of Fig. 1(b) illustrates that two actuators drive the rotation of two arms ("A" and "C") resulting in a translational end-effector ("eff") motion in a horizontal plane. All joints are flexure joints allowing rather smooth operation with low friction and hysteresis. Consequently, contributions from the link mass and joint stiffness dominate in the non-linear equation of motion written in independent generalised coordinates $\boldsymbol{q}$ as

$$\boldsymbol{F} = \boldsymbol{F}_{\mathrm{c}} + \boldsymbol{F}_{\mathrm{nc}} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \frac{1}{2}\dot{\boldsymbol{q}}^T \frac{\partial \boldsymbol{M}(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \frac{\partial V(\boldsymbol{q})}{\partial \boldsymbol{q}} + \boldsymbol{F}_{\mathrm{nc}}, \tag{1}$$

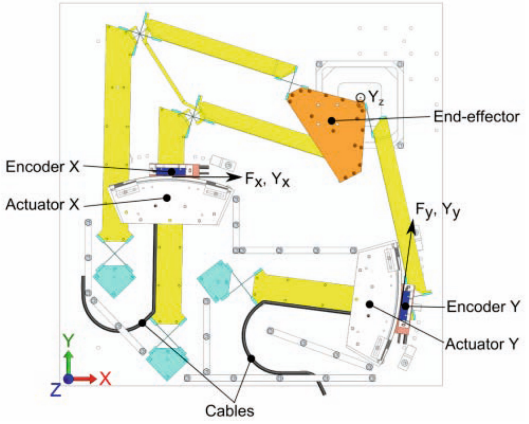
where $\boldsymbol{F}_{\mathrm{c}}$ represents the conservative part in the total actuator force vector $\boldsymbol{F}$. It is expressed in the symmetric and positive definite mass matrix $\boldsymbol{M}$ and potential energy V, the latter due to the stiffness in all joints. The non-conservative



(a) Photo (by Ger Folkersma).



(b) Schematic view.



(c) Top view.

Fig. 1. The 2 DOF manipulator with flexure joints [3]. The view of the manipulator in the photo is rotated relative to the other views.

forces $\boldsymbol{F}_{\mathrm{nc}}$ describe all remaining, possibly non-linear effects like cogging or friction caused by the cables connected to the moving parts of the actuators and sensors. Both rotation angles of the actuated arms are chosen as independent coordinates $\boldsymbol{q}$. These are computed from the displacements Y_x and Y_y measured with "Encoder X" and "Encoder Y" in Fig. 1(c). Both actuator forces F_x and F_y shown in this figure are calculated from the applied motor currents assuming a constant and known ratio between force and current for each motor.

2.2 White-Box Model

For the white-box model the kinematic relations must be derived that express the motion of all links and the joint rotations in the independent coordinates $\boldsymbol{q}$. The kinematic expressions are simplified by ignoring pivot shifts that are inherently introduced by the cross flexure joints used for the joints in this setup.

In the dynamic parameters 5 independent mass and inertia contributions from the links and end-effector can be defined. To represent the joint stiffnesses

6 independent stiffness parameters are needed. The damping and friction contribution $\boldsymbol{F}_{\mathrm{nc}}$ is assumed to be dominated by 2 damping parameters that capture the (viscous) damping arising from the cables of which the deformations are directly linked to the joint rotations and hence the independent coordinates $\boldsymbol{q}$. Finally it was observed in experimental data that the measured forces can exhibit a constant offset which results in 2 more parameters.

The equation of motion (1) can be written in a parameter linear expression as

$$\boldsymbol{\Phi}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\ \boldsymbol{\theta} = \boldsymbol{F}, \tag{2}$$

where parameter vector $\boldsymbol{\theta}$ stores the 15 parameters and regression matrix $\boldsymbol{\Phi}$ depends only on $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$. From a mathematical point of view it is expected that the regression matrix is non-singular for the proposed parameter set and all parameters can be found with the Moore–Penrose pseudoinverse $\boldsymbol{\Phi}^{\dagger}$. However, using experimental data it may appear that with a feasible excitation not all parameters can be identified accurately. Common linear regression techniques can be used to determine a base parameter vector, e.g. from a singular value decomposition of $\boldsymbol{\Phi}$ [6,10].

2.3 Lagrangian Neural Network (LNN)

To obtain the black-box model it would be possible to learn the total forces $\boldsymbol{F}$ from trajectory data $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ with a FNN, but then the physical structure of $\boldsymbol{F}_{\mathrm{c}}$ is not taken into account. Hence a LNN [8,9] is used for this part of the feedforward control that can be written as

$$\boldsymbol{F}_{\mathrm{c}} = \boldsymbol{L}(\boldsymbol{q})\boldsymbol{L}^{T}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \frac{1}{2}\dot{\boldsymbol{q}}^{T}\frac{\partial(\boldsymbol{L}(\boldsymbol{q})\boldsymbol{L}^{T}(\boldsymbol{q}))}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \frac{\partial V(\boldsymbol{q})}{\partial \boldsymbol{q}}, \tag{3}$$

where the lower triangular matrix $\boldsymbol{L}$ is the Cholesky decomposition of the mass matrix $\boldsymbol{M}$. For a 2-DOF system it has only one non-zero off-diagonal term l_{o1}

$$\boldsymbol{L} = \begin{bmatrix} l_{d1} & 0 \\ l_{o1} & l_{d2} \end{bmatrix}, \tag{4}$$

and both diagonal terms l_{d1} and l_{d2} are positive to ensure a positive definite mass matrix $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^{T}$. The rightmost term in Eq. (3) includes the non-negative potential energy V.

Figure 2 shows the structure of the LNN implementation in which a "Physics layer" is added to a FNN. This combination and the implementation of the FNN assure that the LNN represents Eq. (3). The outputs of the FNN are estimates of the potential energy $\hat{V}$ and matrix $\hat{\boldsymbol{L}}$ where the latter is split into the off-diagonal $\hat{\boldsymbol{l}}_o$ and positive diagonal $\hat{\boldsymbol{l}}_d$ terms. All estimates are functions of the positions $\boldsymbol{q}$ only. The blue neuron ($\hat{\boldsymbol{l}}_o$) in the output layer represents a linear activation function and the orange neurons ($\hat{\boldsymbol{l}}_d$, $\hat{V}$) have a ReLu or rectifier activation, which means that their output equals the neuron's input for non-negative values and zero otherwise. Automatic differentiation [2] is used to compute the derivatives

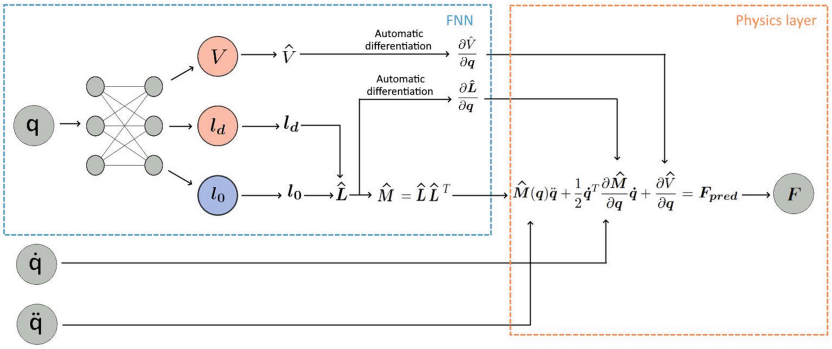


Fig. 2. Structure of the LNN, including the physics transformations (adapted from [5]).

of $\hat{V}$ and $\hat{\boldsymbol{L}}$ with respect to $\boldsymbol{q}$ that appear in Eq. (3). The network is trained to minimise the mean squared error between the estimated forces denoted $\boldsymbol{F}_{pred}$ in Fig. 2 and measured forces.

The remaining forces $\boldsymbol{F}_{\mathrm{nc}}$ are estimated with a general FNN, where it is assumed the forces only depend on positions $\boldsymbol{q}$ and velocities $\dot{\boldsymbol{q}}$. In the training procedure of both networks we prefer to avoid the dependency on a penalisation factor that is used in simultaneous training [7]. We concluded from simulations that training the LNN first on the full $\boldsymbol{F}$ results in a sufficiently accurate estimate of $\boldsymbol{F}_{\mathrm{c}}$. Apparently the relatively small contribution of $\boldsymbol{F}_{\mathrm{nc}}$ does not result in a significantly biased estimate of the $\boldsymbol{F}_{\mathrm{c}}$ part. Hence the LNN is trained first and next the FNN is trained on the residue $\boldsymbol{F} - \boldsymbol{F}_{\mathrm{c}}$. Tensorflow is used for the training of both LNN and FNN.

3 Results

3.1 Training Data

To identify the parameters in the white-box model and to train the neural networks, 9 datasets have been used from as many controlled motion experiments. For each experiment a two-dimensional trajectory has been generated which specifies the desired end-effector x and y positions as functions of time. Each of these desired positions is described with a non-periodic smooth random function [4] defined by Fourier series with random coefficients. For a given wavelength parameter $\lambda > 0$ and interval $[0, L]$, the function is given by

$$f(t) = \sqrt{2} \sum_{j=1}^{m} \left[a_j \cos\left(\frac{2\pi j t}{L}\right) + b_j \sin\left(\frac{2\pi j t}{L}\right) \right], \tag{5}$$

where $m = L/\lambda$ and coefficients a_j, b_j are independent samples from a normal distribution with zero mean and variance $\frac{1}{2m+1}$. The datasets are generated with $\lambda \in [10, 8.0, 6.0, 4.0, 0.9, 0.8, 0.6, 0.5, 0.4]$, a total duration of $L = 120$ s and sample time $t_s = 0.0001$ s. All paths are scaled to a maximum amplitude

of ±30 mm and converted to actuator displacements using inverse kinematic relations assuming rigid links and ideal rotational joints. These paths are the reference positions for controlled actuator motion where a PD feedback controller is used that should result in reasonably accurate tracking. The actual encoder reading and applied motor currents are recorded. The independent coordinates $\boldsymbol{q}$ and actuator forces $\boldsymbol{F}$ are computed as described in Sect. 2.1. Velocities $\dot{\boldsymbol{q}}$ and accelerations $\ddot{\boldsymbol{q}}$ are derived off-line with numerical differentiation of the position data and low-pass filtering.

The collected data in each dataset originally represents 1,200,000 time samples. The first 200,000 samples are discarded to eliminate any initial transient response. The last 200,000 samples are discarded as well, such that 800,000 samples remain in each dataset. Datasets with larger values of λ have more low frequent content and excite in particular the stiffness properties, i.e. the potential energy $V(\boldsymbol{q})$, whereas datasets with smaller λ reveal more mass dominated dynamics. Contributions from the non-conservative forces can appear at various frequencies, depending on the cause of these forces. E.g. viscous damping reduces the significance of the resonance peaks which appear near 1 Hz and 2 Hz. Hence this contribution can be identified from data sets with smaller λ that include these resonance frequencies.

3.2 White-Box Identification

To identify the 13 physical parameters in the white-box model it is not needed to use the large amount of 7,200,000 samples from all 9 datasets. Only 1/1000 of these time steps are used in the following identification. Regarding the offsets mentioned in Sect. 2.2 it was found that datasets 8 and 9 showed a slightly different offset compared to datasets 1–7. Hence 2×2 offset parameters are included in the parameter vector giving rise to 17 parameters in total. However, it was found from a singular value decomposition of the regression matrix $\boldsymbol{\Phi}$ that no more than 12 independent parameters can be identified. This is confirmed from an analysis of the residual error which hardly decreases when more than 12 (linear combinations of) parameters are identified.

Figure 3 shows the measured forces F and the misfit of the forces ΔF for all 9 datasets. The datasets are concatenated in the plot, but as explained in Sect. 3.1 these datasets are collected independently and hence discontinuities can be seen in the plot. The force misfit is less than about 10% compared to the actual force. This error is somewhat larger than expected beforehand, which could be caused by experimental imperfections e.g. from the cables (see Fig. 1(c)) or by (white-box) model errors e.g. due to an incorrect kinematic model in which pivot shift is neglected.

3.3 LNN Training

Initially only the LNN is trained for which also only a relatively small subset of all data is used with 20,000 samples, i.e. every 400^{th} sample. These samples are shuffled after which 70% is used for training and 30% for validation. The

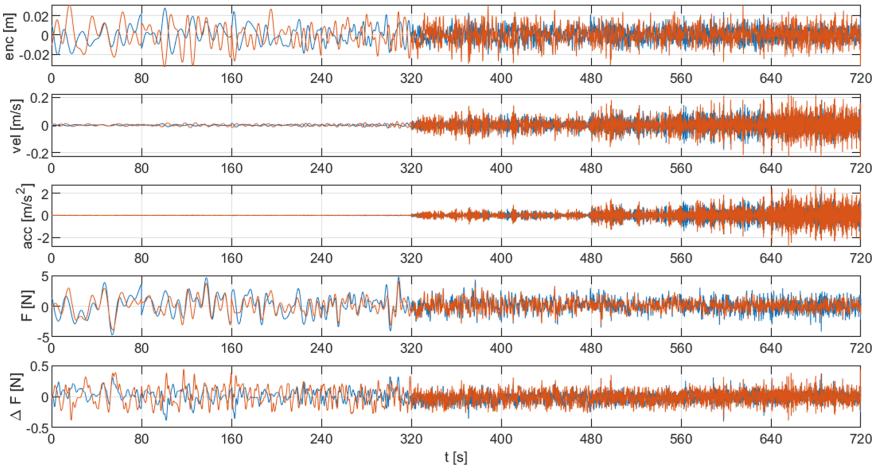


Fig. 3. Manipulator positions, velocities, accelerations, forces F and force misfit ΔF from an identification with 12 independent parameters. The different colours represent x and y data respectively.

Adam solver is used and the batch size is 32. The training is run for 40 epochs with a learning rate $l_r = 0.001$, next 10 epochs with $l_r = 0.0001$ and finally 10 epochs with $l_r = 0.00001$. Furthermore training is initialised 10 times with random seeds. The performance is measured with the Mean Absolute Error (MAE) between target and approximation. The MAE is evaluated for the best training result from all random seeds, as well as for its standard deviation to measure the robustness of the training.

To select the activation function, a network with 6 hidden layers with 32 neurons each is used. The activation function of the output layer is fixed as outlined in Sect. 2.3. Softplus, sigmoid, ReLu and tanh activation functions are evaluated [5]. The latter two give similar results in terms of best MAE, where tanh shows the smallest standard deviation and therefore is the preferred activation function.

For this activation function the effect of varying network size is evaluated next for different numbers of hidden layers and neurons per layer [5]. A network with 32 neurons per layer gives better result where a minimal MAE is found for 8 hidden layers. Hence 8 hidden layers with 32 neurons will be used for the LNN.

For this network two additional check have been done. The number of time samples has been varied. It appeared that the MAE increases when less than 14,000 samples were used, but hardly changes when using 14,000 or 35,000 samples. The trainings presented in this section are well within the latter range. Finally, the batch size and optimizer have been evaluated. It was found that training with a batch sizes of 16, 32 and 64 does not yield a significant change in outcome. Similarly, using Adagrad or SGD instead of the Adam optimiser didn't give significantly different results either.

3.4 FNN Training

Once the LNN is trained as outlined in the previous section, it is fixed and the FNN is trained on the residue. A similar procedure is followed to evaluate the FNN performance and optimise its hyperparameters. Once more the Adam solver is used with a batch size of 32. As no physical structure constrains the FNN, it is expected more training data is needed and more epochs are required before the solution is converged. Therefore the FNN is trained for 100 epochs with $l_r = 0.001$ and next for 50 epochs with $l_r = 0.0001$, initially on a dataset with 50,000 time samples. The input data is scaled to values between -1 and $+1$ using the MinMaxScaler. It appeared that the probability of converging to a local minimum is less likely and hence the training is initialised with only 3 random seeds.

Different activation functions are evaluated for a FNN with 3 hidden layers and 16 neurons in each layer [5]. The tanh activation function is preferred as it results in a small MAE with a small standard deviation.

Next different numbers of hidden layers and neurons per layer are evaluated [5]. It was found that relatively large networks offer a better MAE for the training and validation data. However, when applying the results to an new test dataset it appeared that the larger networks clearly suffer from overfitting. A smaller MAE was obtained with a smaller network. The most suitable FNN structure has only 2 hidden layers with 8 neurons each.

This FNN has been trained with varying numbers of time samples. The MAE is shown in Fig. 4. Clearly a large number of samples is advantageous. Extrapolating the trend in the plots, the number of samples could even be increased beyond the maximum presented in the figure. However, this will also result in an increase of computational costs, so 200, 000 samples will be used to train the FNN.

Table 1 summarises the MAE obtained for training and validation using only the preferred LNN and combining it with the FNN as proposed above. The validation MAE is split to show the errors for both actuator forces F_x and F_y separately. The table illustrates the improvement that can be obtained by adding the FNN to the LNN. It also shows that the errors in both forces are comparable

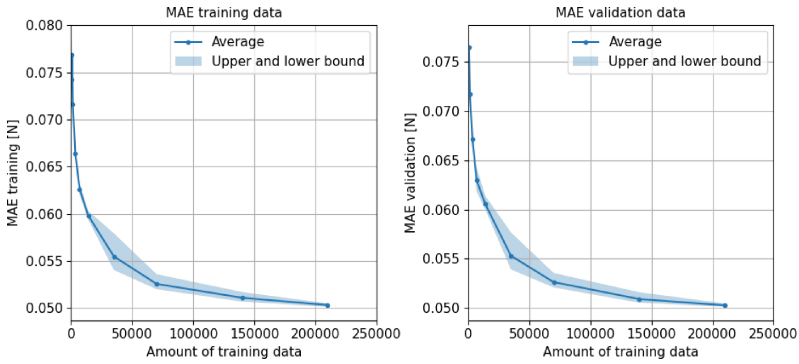


Fig. 4. Mean Absolute Error (MAE) for different numbers of time samples used for the FNN training (2 hidden layers with 8 neurons each) [5].

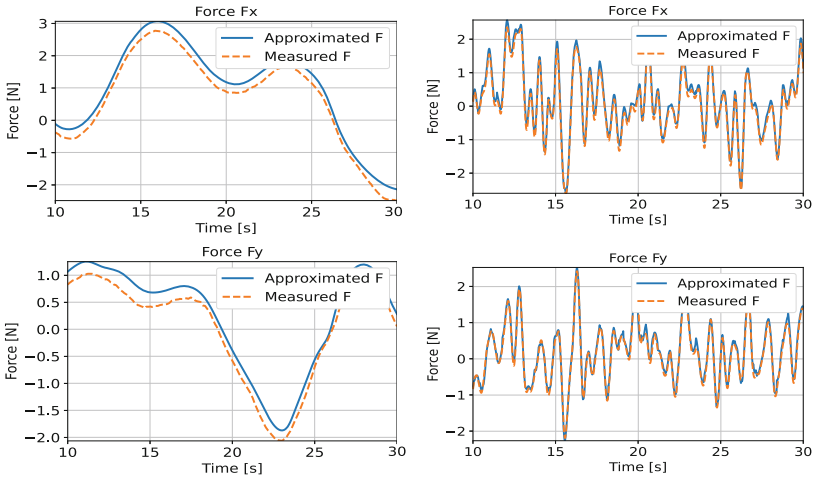
Table 1. MAE on the training and validation data using the selected LNN and FNN. The MAE of the white-box model, Sect. 3.2, is included for comparison.

	Training MAE [N]	Validation MAE [N]	Valid. MAE F_x [N]	Valid. MAE F_y [N]
LNN	0.1022	0.1022	0.0836	0.1208
LNN + FNN	0.0517	0.0514	0.0562	0.0466
white-box model	0.0864			

and are below 2%. Furthermore it appears that the MAE of the force estimates from the combined neural networks is smaller than is obtained with the white-box model of Sect. 3.2.

3.5 Offline Force Estimates

The performance of the white-box model and the trained LNN+FNN combination is evaluated using 10 test trajectories that are generated similarly to the datasets defined in Eq. (5) in Sect. 3.1 except for a shorter duration with $L = 40$ s. The first and last 10 s of the data are removed and for the remaining time samples the forces are estimated with both methods and compared to measured values. Data processing is similar to the procedure outlined in Sect. 3.1 for the training data, i.e. the trajectories are the reference positions for controlled actuator motion. The forces are estimated "offline" which means that the motion data have been captured to be processed afterwards.



(a) $\lambda = 7.0$, amplitude = 30 mm (b) $\lambda = 0.5$, amplitude = 30 mm

Fig. 5. Forces estimated by the LNN+FNN combination and measured forces for two of the test trajectories [5].

Fig. 5 illustrates the performance of the LNN+FNN combination. The force estimates for the low-frequent reference ($\lambda = 7.0$) show a clear offset. Mostly the

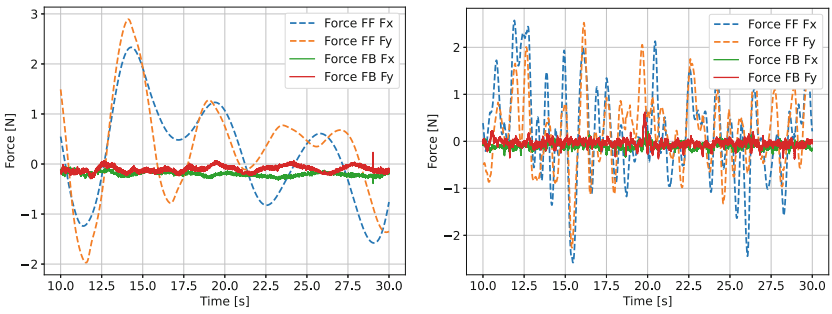
forces are estimated much better like e.g. the example with the high-frequent reference ($\lambda = 0.5$). The offset may be caused by small differences in the initial configuration of the manipulator which can arise from the cables as mentioned before. To evaluate the force estimates, the MAE is computed after accounting for these offset. Then the average MAE for all 10 test datasets is 0.0505 N for the estimate from the neural networks and 0.0794 N for the white-box estimate.

3.6 Online Feedforward Control

The ultimate goal is to improve the tracking accuracy of the manipulator by implementing the estimated forces as feedforward control inputs, i.e. "online". This performance is evaluated from a real-time experiment in which the estimated forces are added as feedforward control inputs (FF) to a standard closed-loop PD feedback controller (FB). In case of a perfect feedforward control, the estimated forces (FF) would result in perfect tracking of the reference path and the output of the feedback controller (FB) is zero due the absence of a tracking error.

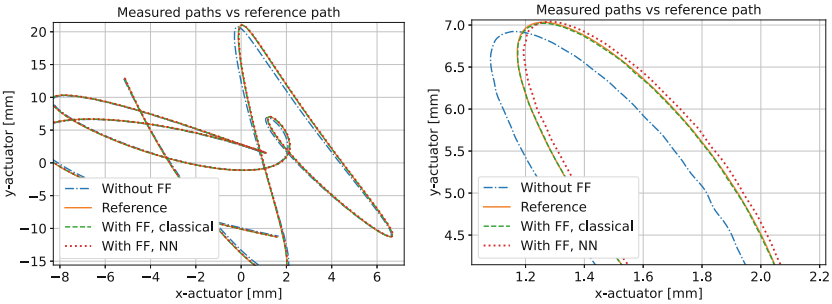
Figure 6 presents the feedforward signals (FF) and the feedback signals (FB) recorded simultaneously during two of the experiments in which the feedforward control is estimated with the trained LNN+FNN combination. It can be seen that the contribution to the actuator force from the feedback controller (FB) is not zero, but it is significantly smaller than the estimated feedforward signal (FF) which apparently accounts for by far the largest part of the total forces required for the specified motion. This proves the effectiveness of the feedforward control estimated with the combined neural networks.

In this way the tracking accuracy is also improved considerably compared to using only feedback as shown in Fig. 7. In this figure also the results are included with the feedforward control from the white-box model. At first sight the latter approach tracks the reference even better, but a large part of the tracking error is the offset that has been discussed before in Sect. 3.5. Such offset can easily be



(a) $\lambda = 4.0$, amplitude = 30 mm. (b) $\lambda = 0.5$, amplitude = 30 mm.

Fig. 6. Feedback (FB) and feedforward (FF from LNN+FNN) control inputs in closed-loop experiment with two different reference trajectories.



(a) Overview. (b) Zoomed in.

Fig. 7. Actuator motion during closed-loop experiment where the reference is generated with $\lambda = 4.0$, amplitude = 30 mm: Without FF as well as with FF from white-box model or LNN+FNN.

eliminated by using PID feedback control instead of PD control. From a more quantitative analysis it appeared that both feedforward control implementations reduce the tracking error with more than 90% compared to using only feedback control [5].

4 Conclusions

In this paper we demonstrate that the feedforward control generated by the successively trained LNN and FNN can be applied successfully for the considered 2-DOF manipulator with flexure joints. The LNN is trained first to ensure it accounts for the larger contribution of the conservative forces to the estimated actuator forces. In this way the advantages of constraining the network to a physical feasible solution are exploited. Only a rather small number of time samples is needed to train the LNN. The trained network can also handle new test trajectories which illustrates the robustness of the feedforward control. This robustness and a more general applicability of this approach will be investigated in the future.

References

1. Abdul-hadi, O.: Machine learning applications to robot control. PhD thesis, University of California, Berkeley (2018)
2. Agrawal, A., et al.: TensorFlow eager: a multi-stage, python-embedded DSL for machine learning. In: Talwalkar, A., Smith, V., Zaharia, M. (eds.) Proceedings of Machine Learning and Systems, vol. 1, 178–189 (2019)
3. Brouwer, D.M., Folkersma, K.G.P., Boer, S.E., Aarts, R.G.K.M.: Exact constraint design of a two-degree of freedom flexure-based mechanism. J. Mech. Rob. **5**(4), 041011 (2013)
4. Filip, S., Javeed, A., Trefethen, L.N.: Smooth random functions, random ODEs, and Gaussian processes. SIAM Rev. **61**(1), 185–205 (2019)

5. Heerze, E.: Data-driven feedforward control of a multi degree of freedom manipulator with flexure joints using machine learning. MSc thesis, University of Twente, Enschede (2021)
6. Khalil, W., Dombre, E.: Modeling. Identification and Control of Robots. Kogan Page Science, London (2002)
7. Liu, Z., Wang, B., Meng, Q., Chen, W., Tegmark, M., Liu, T.-Y.: Machine-learning nonconservative dynamics for new-physics detection. Phys. Rev. E **104**, 055302 (2021)
8. Lutter, M., Ritter, C., Peters, J.: Deep Lagrangian Networks: using physics as model prior for deep learning. In: 7th International Conference on Learning Representations (ICLR) (2019)
9. Lutter, M., Peters, J.: Combining physics and deep learning to learn continuous-time dynamics models. arXiv:2110.01894 (2023)
10. Wu, J., Wang, J., You, Z.: An overview of dynamic parameter identification of robots. Rob. Comput.-Integr. Manuf. **26**(5), 414–419 (2010)