

TDK-DOLGOZAT
GYÖRÖK BENDEGÚZ MÁTÉ



BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
GÉPÉSZMÉRNÖKI KAR
MECHATRONIKA, OPTIKA ÉS GÉPÉSZETI INFORMATIKA TANSZÉK



HUN-REN SZÁMÍTÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓINTÉZET

Fizikai modellek hatékony adatalapú kiegészítése

Györök Bendegúz Máté
MSc IV. félév

Konzulensek:

Dr. Tóth Roland

Tudományos főmunkatárs / Full professor

HUN-REN SZTAKI

Eindhoven University of Technology

Dr. Péni Tamás

Tudományos főmunkatárs

HUN-REN SZTAKI

Témavezető:

Dr. Budai Csaba

Adjunktus

BME-GPK MOGI

BUDAPEST, 2024. november 9.

Contents

Abstract	V
1 Introduction	1
1.1 Overview of the literature	2
1.2 Main contributions	3
1.3 Structure of the paper	3
2 Model augmentation framework	5
2.1 Problem statement	5
2.2 Introducing augmented states	6
2.3 Additive model augmentation structure	6
2.4 Multiplicative model augmentation structure	8
2.5 LFR-based model augmentation structure	9
3 Model training and physics-guided regularization	13
3.1 Truncated prediction loss and subspace encoder	13
3.2 Tuning of physical parameters	14
3.3 Orthogonalization-based regularization	16
3.4 Generalizing the orthogonalization-based approach for nonlinear first-principle models	18
3.5 Further options for improving training stability	18
4 Encoder initialization method	20
4.1 Generalized encoder network	20
4.2 Generalization for NL FP models	21
4.3 State reconstruction for LTI models	22
4.4 Simulation example	23
5 Modeling and simulation of the F1Tenth vehicle	25
5.1 Modeling considerations	25
5.2 Drivetrain model	27
5.3 Tire model	27

5.4	Physical parameters of the model	28
5.5	High fidelity simulation environment	28
5.6	Advantages of the digital twin model	30
6	Identifying the dynamics of the F1Tenth vehicle	33
6.1	Additive augmentation results	34
6.1.1	Finding the optimal physical parameters	35
6.1.2	Results with orthogonalization-based cost function	37
6.2	Multiplicative augmentation results	38
6.3	LFR-based augmentation results	38
6.4	Comparison with black-box methods	39
6.4.1	Model selection and black-box results	40
6.4.2	Convergence speed	40
7	Mass-Spring-Damper identification example	42
8	Summary and conclusions	45
8.1	Results	45
8.2	Conclusions, outlook	46
	Reference	52

Abstract

In recent years, accurate dynamic models have become essential for reliable motion control and trajectory planning of autonomous vehicles. While simplified mechanical models based on physical principles can be provided, the performance of these models tends to diminish during aggressive maneuvers, because complex nonlinear effects, such as aerodynamic forces, friction characteristics, tire dynamics, etc., are often neglected. Alternatively, data-driven identification techniques can generate highly accurate models without prior physical insight; however, these black-box models lack physical interpretability, making them difficult to utilize for control and path-planning purposes. Therefore, to marry the advantages of each of these modeling paradigms, the first-principle dynamics can be augmented with deep-learning-based techniques, aiming to achieve highly accurate, yet physically interpretable models.

Model augmentation can be accomplished using various structures, such as additive or multiplicative combinations. The selection of the best-performing structure usually depends on the baseline model and what is needed to be learned from data, however, this property is only known after training each structure and comparing them. The main contribution of the paper is developing a general augmentation structure based on a Linear Fractional Representation (LFR), as the modular framework of LFRs makes flexible augmentation of the physics-based models possible. Moreover, the uncertainty of the physical parameters can also result in poor performance of physics-based models, thus it is reasonable to tune them jointly with the neural network parameters, however, the model becomes overparametrized with this modification. To address this, novel physics-based regularization methods are introduced.

Finally, we compare the presented augmentation methods with state-of-the-art black-box approaches by identifying the dynamics of a small-scale electric vehicle (F1Tenth).

1 Introduction

Continuously rising performance requirements and the growing complexity of control tasks resulted in model-based methods gaining a dominant role in the vehicle industry. For example, reliable path planning and motion control of autonomous vehicles require accurate, personalized dynamic models. Simplified baseline models can usually be obtained from physical knowledge and practical engineering insight, but these models can only provide an approximated system description. Furthermore, the state-of-the-art new generation of vehicles is often utilized in extreme operational conditions where the nonlinear effects become dominant. During these aggressive maneuvering situations, simplified models tend to perform inaccurately. Two examples are shown in Figure 1.1 for vehicles that can not be described using traditional, first-principles-based models, as highly nonlinear effects govern their motion.

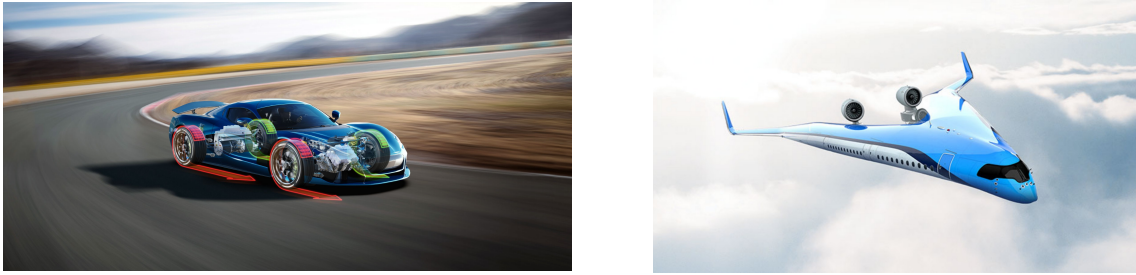


Figure 1.1: Rimac Nevara with illustrations of the All Wheel Torque Vectoring (AWTV) system¹ (left), and the Flying-V aeroplane² (right).

As an alternative approach, recent advances in deep-learning-based nonlinear system identification have resulted in highly accurate and reliable methods, e.g. the SUBNET method [1]. These so-called black-box models can capture highly nonlinear physical phenomena, and can even accurately describe vehicle dynamics [2], but it is hard to use them for trajectory planning and control purposes since they can not be physically interpreted.

This work aims to propose different model augmentation methods, which can allow the efficient combination of first-principles-based mechanical models (or shortly *first-principle* (FP)) models) and deep-learning-based identification methods. The goal is to produce models with similar accuracies as state-of-the-art data-driven identification methods while preserving (at least some of) the physical meaning of FP models. Furthermore, utilizing physical knowledge in the model structure can reduce optimization time and improve model accuracy on unseen data. One of the most important open questions in data-driven system identification is the frugal learning of system dynamics, which practically means that model optimization should require much smaller amounts of data than traditional machine learning algorithms. If implemented efficiently, model augmentation is a promising direction in that regard as well.

¹Source: <https://www.rimac-technology.com/en/product-categories/torque-vectoring/>

²Source: <https://www.travelwires.com/the-flying-v-plane-s-design-aims-to-reduce-fuel-consumption>

1.1 Overview of the literature

Combining first-principle models with deep-learning-based terms is a relatively new, but well-researched field, mainly because of the known limitations of data-driven methods. One of the most important drawbacks of black-box models is that physical knowledge about the given system is disregarded. When sufficiently large training and test data are available, this property usually does not cause problems, but models typically extrapolate inaccurately from sparse training data, furthermore, neural networks are more prone to overfitting when trained on small datasets.

For *Linear Time Invariant* (LTI) system representations, the grey-box modeling approach has been applied successfully in system identification for years now [3]. Grey-box models utilize physics-based system descriptions (usually in LTI form) and estimate the model parameters directly from data [4]. This method often applies statistical learning algorithms [5], but deep-learning-based techniques are also used [6]. Although conventional grey-box approaches can enhance the accuracy of physics-based models by (partly) eliminating parameter uncertainties, a more powerful combination of FP models and learning-based techniques could potentially further improve modeling accuracy. However, an important question emerges: how to marry the advantages of physics-based modeling and deep-learning-based approaches in an efficient manner?

An alternative approach is to utilize prior physical knowledge during model training because, with black-box models, there is no guarantee of producing models that adhere to the known laws of physics. To address this challenge, *physics-informed neural networks* (PINNs) and *physics-guided neural networks* (PGNNs) were introduced in [7], and [8], respectively. Both of these methods utilize an additional term in the cost function, which penalizes deviations from the apriori-selected physical laws. While physics-based loss functions can be efficiently used for producing physically consistent models, this work aims to incorporate physical knowledge directly into the model structure.

However, the combination of the FP part and the data-driven part is not intuitive, as different model augmentation structures can be defined, e.g. [8] also proposes various formulations, such as additive, multiplicative, etc. Moreover [9] provides a comprehensive description of different combinations. Many more augmentation frameworks can be found in the literature, e.g. [10, 11, 12, 13], but none of these methods present a generalized description of the model augmentation problem. The selection of the best-performing structure is usually system-dependent, which is a hard task for complex, nonlinear systems. Hence there is a growing need for a generalized structure, which can efficiently compensate the model uncertainties and learn the unmodeled dynamics simultaneously. A promising direction is proposed in this work based on *Linear Fractional Representations* (LFRs) since an LFR-like modular framework allows the flexible augmentation of first-principle models in a generalized form [14]. Furthermore, the proposed LFR-based model augmentation framework can be interpreted as a generalization of the previously mentioned existing structures. Another great advantage of LFR-based techniques is that

several fundamental control algorithms can be found in the literature for classical LFRs [15, 16], hence a huge potential lies in that aspect.

Several other recent works have been focusing on identifying models directly for control purposes, e.g. physics-based and neural network layers are combined to form a feedforward control filter. This way, the physical parameters are identified simultaneously with the neural network parameters, which causes over-parametrization, physical parameters can be tuned to unrealistic values, ultimately resulting in poor extrapolation capabilities. To prevent competition among the physical and neural network layers, and to ensure interpretable physical parameters, [17] proposed a novel regularization framework, which we aim to implement into the general model augmentation setup. Another regularization based on orthogonal projection was presented in [18], which is applicable for parallel connected FP models and neural networks. The original method can be only applied to FP models, that are linear in the parameters, but we aim to extend the approach for general nonlinear models.

1.2 Main contributions

The materials presented in this paper enhance current state-of-the-art model augmentation techniques significantly in terms of generalization capabilities, convergence properties, and model accuracy. The main contributions of this work can be summarized as:

- C1 Presenting an efficient implementation of simple augmentation structures (additive, multiplicative formulations) with derived initialization and normalization schemes. The subspace encoder method is also utilized for dynamic augmentation purposes, then all mentioned augmentation structures are generalized for the dynamic form.
- C2 Proposing a general model augmentation framework based on LFRs.
- C3 Presenting a regularization method for the physical parameters of the FP models, to improve convergence properties. An orthogonal projection-based regularization is also presented for the additive structure.
- C4 Deriving an efficient encoder initialization method for systems where full-state measurements are not possible.

1.3 Structure of the paper

The rest of this paper is organized as follows: Section 2 starts with the problem formulation of the identification task and then discusses various structures for the model augmentation framework. Starting from the simple, intuitive formations to a generalized, LFR-based method. For each structure, a comprehensive parameter initialization is also presented, moreover, the differences between static and dynamic model augmentation methods are discussed.

In Section 3, different training options are detailed. A general, multiple shooting-based loss function is derived for the model augmentation setup, and then the tuning of the physical parameters is also discussed. The challenge of identifiability arises when the physical parameters are tuned simultaneously with the neural network parameters. To address this problem, different physics-guided regularization methods are proposed.

When full-state measurements of the investigated system are not available, an encoder network is utilized to approximate the initial states. However black-box identification of the so-called reconstructability map is quite inefficient. To enhance the applicability of the approach, a novel initialization scheme is derived in Section 4, and then the performance of the method is analyzed with a short simulation example.

Section 5 presents the first-principle model of the F1Tenth electric car with the modeling considerations and identified physical parameters, while also discussing the data acquisition in a high-fidelity simulation environment. The advantages of gathering data with a digital twin model are also thoroughly analyzed.

In Section 6, the capabilities of the proposed approach are demonstrated by identifying the dynamics of the F1Tenth vehicle. Each developed model augmentation method is extensively studied by investigating the test results, furthermore, the presented parameter regularization methods are also examined. The model augmentation methods are compared to state-of-the-art black-box identification methods.

As all model states can be measured directly for the F1Tenth vehicle, to test the proposed encoder initialization scheme, in Section 7 the method is applied to identify the nonlinear dynamics of a mass-spring-damper system. The data-generating system can be described by using 6 states, however, only one of them is measured, which is why the encoder network is necessary for reconstructing the initial states. After the identification task, the results are analyzed and compared to a widely used random initialization scheme.

Then finally, Section 8 presents a brief summary of the proposed methodology, and conclusions are drawn based on the achieved results. Moreover, possible future applications and research ideas are also detailed.

2 Model augmentation framework

2.1 Problem statement

Consider the dynamics of the data generating system represented by the following *continuous-time* (CT) nonlinear *state-space* (SS) representation:

$$\dot{x}(t) = f_c(x(t), u(t)), \quad (2.1a)$$

$$\tilde{y}(t) = h(x(t)), \quad (2.1b)$$

where $t \in \mathbb{R}$ is the continuous time, $x(t) \in \mathbb{R}^{n_x}$ is the collection of states, $u(t) \in \mathbb{R}^{n_u}$ is the control input, and $\tilde{y}(t) \in \mathbb{R}^{n_y}$ is the output of the system, while $f_c : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x}$ and $h : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_y}$ are possibly nonlinear bounded functions. Based on digital actuation and data acquisition devices, we assume that the input satisfies $u(t) = u[k]$ for all $kT_s \leq t < (k+1)T_s$, where $u[k]$ is a *discrete-time* (DT) signal with $k \in \mathbb{Z}$ and $T_s \in \mathbb{R}_+$ denotes the sampling time, corresponding to a *zero-order-hold* (ZOH) setting. Under these assumptions, the dynamics represented by (2.1) can be reformulated as

$$x[k+1] = f(x[k], u[k]), \quad (2.2a)$$

$$y[k] = h(x[k]) + e[k], \quad (2.2b)$$

where $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x}$ is the DT state transition function. In (2.2b), $y[k]$ is the measured output, where $e[k]$ represents the measurement noise.

We assume that based on prior knowledge and physical insight, a baseline model is available that describes some of the dominant dynamic relations of the system as

$$x_b[k+1] = f_\theta(x_b[k], u[k]), \quad (2.3a)$$

$$\hat{y}[k] = h_\theta(x_b[k], u[k]), \quad (2.3b)$$

where $x_b[k] \in \mathbb{R}^{n_x}$ is the model state (or baseline state), $\hat{y}[k] \in \mathbb{R}^{n_y}$ is the model output, $f_\theta : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x}$ is the FP state transition function, and $h_\theta : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_y}$ is the output function, that both depend on $\theta \in \mathbb{R}^{n_\theta}$ physical parameters, for which we assume θ_0 nominal values are known.

Physical models are often subject to large amounts of uncertainty and unmodeled dynamics, which can be the main causes of model errors. To compensate for these effects, deep-learning-based model augmentation is introduced to enhance the model accuracy while still producing interpretable models. The exact augmentation framework will be detailed throughout this section.

2.2 Introducing augmented states

To consider the memory effect of the unmodeled dynamics, the so-called *augmented states* are introduced in the following structure:

$$\hat{x}[k] = \begin{bmatrix} x_b[k] \\ x_a[k] \end{bmatrix}, \quad (2.4)$$

where $x_b[k] \in \mathbb{R}^{n_x}$ stands for the baseline- and $x_a[k] \in \mathbb{R}^{n_a}$ for the augmented states. Now $\hat{x}[k] \in \mathbb{R}^{n_x+n_a}$ denotes the combined model state of the augmentation structure.

Of course for systems, where the unmodeled terms only affect the baseline system as static nonlinearities, utilizing augmented states causes over-parametrization. Thus, we introduce static augmentation when $n_a = \dim x_a = 0$, and dynamic augmentation, where $n_a > 0$. Note that static augmentation is just a special case of the dynamic one, thus in the detailed description of specific structures, we do not differentiate between these two approaches. Furthermore, for notational convenience, during some parts of the paper, the DT state transition will be noted as $x^+ = f(x, u)$, while the output map as $y = h(x, u)$.

The aim of the identification is to provide a system description in the form of

$$\begin{bmatrix} x_b^+ \\ x_a^+ \\ \hat{y} \end{bmatrix} = \Phi(x_b, x_a, u), \quad (2.5)$$

where the augmentation structure $\Phi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_a} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x} \times \mathbb{R}^{n_a} \times \mathbb{R}^{n_y}$ determines both the state transition and the output map of the augmented model. A generalized description can be also expressed using the extended state as

$$\begin{bmatrix} \hat{x}^+ \\ \hat{y} \end{bmatrix} = \Phi(\hat{x}, u). \quad (2.6)$$

2.3 Additive model augmentation structure

One of the simplest ways to augment the baseline first-principle dynamics is to introduce an additive term into the state transition. This formulation is one of the most well-known augmentation approaches (mainly in the static sense), e.g. similar structures are applied for augmenting ordinary differential equations [19] or augmenting physics-based feedforward control filters [17]. The FP output function h_θ is assumed to contain simple relations (can even be an identity map), thus the aim is to identify the system dynamics in the form of

$$\hat{x}^+ = \begin{bmatrix} x_b^+ \\ x_a^+ \end{bmatrix} = \begin{bmatrix} f_\theta(x_b, u) \\ 0_{n_a} \end{bmatrix} + \mathcal{F}_\phi(x_b, x_a, u), \quad (2.7a)$$

$$\hat{y} = h_\theta(x_b, u), \quad (2.7b)$$

where $\mathcal{F}$ is a multi-layer, feedforward, fully connected neural network with parameter $\phi \in \mathbb{R}^{n_\phi}$, containing the weights and biases. The structure of the state transition is illustrated in Figure 2.1.

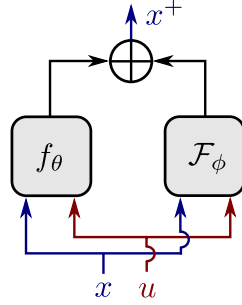


Figure 2.1: Schematic representation of the state transition in case of static additive augmentation structure.

To efficiently train artificial neural networks, *input-output* (IO) **normalization** is essential to ensure that the training data is zero-mean and has a standard deviation of 1. This step is essential since most initialization approaches - e.g. [20] - assume normalized data and the "active" range of the most popular activation functions is in the range of -1 to 1. For black-box ANN-SS models, the data normalization process is usually implemented before the optimization, and after each model evaluation, back-scaling is applied to compare the results with the true system outputs [1]. For the grey-box approach, a different normalization process needs to be applied, as the original, physically true data is needed for the FP model. A transformation is defined similarly as in [14], resulting in a state transition that can be expressed as

$$\hat{x}^+ = \begin{bmatrix} f_\theta(x_b, u) \\ 0_{n_a} \end{bmatrix} + \begin{bmatrix} T_x^{-1} & 0_{n_x \times n_a} \\ 0_{n_a \times n_x} & I_{n_a} \end{bmatrix} \mathcal{F}_\phi(T_x x_b, x_a, T_u u), \quad (2.8)$$

where the transformation matrix $T_u \in \mathbb{R}^{n_u \times n_u}$ is a diagonal matrix containing the inverse of the standard deviation of the input signal calculated based on the training data, as $T_u = \text{diag}(1/\sigma_{u,i})$ for $i = 1, \dots, n_u$, and σ notes the standard deviation. The transformation matrix T_x can be defined similarly, but we may not have information about the state values if full-state measurements are not available. In that case, these states can be obtained by e.g., forward simulating the FP model on the training data. This normalization approach works well if the data is zero-mean, otherwise, a hybrid scheme is proposed as $T_u = \text{diag}(1/\sqrt{\mu_{u,i}^2 + \sigma_{u,i}^2})$, where μ notes the mean, moreover T_x can be also defined similarly. This alternative scaling method considers both the central tendency and spread of the data simultaneously, while normalization and denormalization are still realized by linear operators (as opposed to traditional min-max scaling or standardization, which can be only realized by affine terms).

Besides IO normalization, a reliable **parameter initialization scheme** is also important for efficient model training to improve accuracy and potentially decrease the convergence time. A similar initialization method can be applied as in [21], which guarantees

that the augmented state transition function behaves like f_{θ_0} at the beginning of training because the FP model is assumed to provide already quite accurate results. Furthermore, fully random initialization can cause stability losses at the beginning of the optimization. The presented method also prevents this, by setting the weight and bias values of the last layer in the ANN to zero. Consider a simple feedforward neural network as

$$\mathcal{F}_\phi(\hat{x}, u) = W_{L+1}\sigma(W_L \dots \sigma(W_0 [\hat{x}^\top u^\top]^\top + b_0) + b_L) + b_{L+1}, \quad (2.9)$$

where $\sigma(\cdot)$ is an element-wise activation function, typically hyperbolic tangent, rectified linear, etc. Technically, each layer (or even each neuron) can have different activation functions, but usually (2.9) gives an appropriate description. Furthermore, W_i represents the weight values, while b_i notes the biases, hence the network parameters are $\phi = \{W_i, b_i\}_{i=0}^{L+1}$, and L notes the number of hidden layers.

The proposed method suggests that $\{W_i, b_i\}_{i=0}^L$ should be initialized randomly, e.g. by using the Xavier initialization approach [20], then the parameters belonging to the last layer should be initialized as zeros, $W_{L+1} = 0$, $b_{L+1} = 0$.

2.4 Multiplicative model augmentation structure

As we have discussed in Section 2.3, additive model augmentation is mostly favored because of its simplicity, which leads to easy implementation, but the main problems of the additive framework originate from its structure. With the neural network being parallel to the FP model, especially when the physical parameters are also tuned, instead of learning the unmodeled dynamics, the neural network and the baseline model start to "compete" with each other and produce suboptimal results. A possible solution to this problem is to apply different physics-based regularization schemes, for example Sections 3.2 and 3.3 discuss two different implementations. An alternative approach is to connect the neural network in series with the FP model, achieving the so-called multiplicative augmentation structure. This configuration allows the neural network to be trained more freely, avoiding "competition" among the two layers and enabling the representation of more complex, unmodeled nonlinearities without structural constraints. However, we lose some of the model interpretability, as the state transition is determined by a black-box model, although one that incorporates the FP model. As a result, the multiplicative structure falls between fully black-box ANN-SS models and additive model augmentation in terms of physical interpretability. The schematics of the method are illustrated in Figure 2.2, and mathematically can be expressed as

$$\xi = f_\theta(x_b, u), \quad (2.10a)$$

$$\hat{x}^+ = \mathcal{F}_\phi(x_b, x_a, u, \xi), \quad (2.10b)$$

$$\hat{y} = h_\theta(x_b, u), \quad (2.10c)$$

where $\xi \in \mathbb{R}^{n_x}$ is a latent variable corresponding to the FP state transition equation, and the machine learning part is now $\mathcal{F}_\phi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_a} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_x+n_a}$, implemented as a multi-layer neural network.

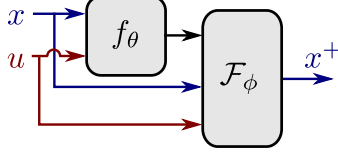


Figure 2.2: Schematics of the static multiplicative augmentation structure.

As previously mentioned, the multiplicative structure can be interpreted as a generalization of the additive case, which can be easily proved, as shown in Appendix A.

A similar **normalization approach** can be defined as for the additive structure, like

$$\xi = f_\theta(x_b, u), \quad (2.11a)$$

$$\hat{x}^+ = \begin{bmatrix} T_x^{-1} & 0_{n_x \times n_a} \\ 0_{n_a \times n_x} & I_{n_a} \end{bmatrix} \mathcal{F}_\phi(T_x x_b, x_a, T_u u, T_x \xi). \quad (2.11b)$$

As shown, the multiplicative structure is a more general approach, hence the importance of a reliable **initialization method** is higher compared to the additive case. The main idea is the same as in Section 2.3: the model parameters should be initialized so that at the beginning of the optimization, the output of the augmented state transition should match the output of the FP state equation. For the multiplicative case, this can be reached by applying a simple residual network, as

$$\mathcal{F}_\phi(\hat{x}, u, \xi) = \underbrace{W_{L+1} \sigma(W_L \dots \sigma(W_0 [\hat{x}^\top \ u^\top \ \xi^\top]^\top + b_0) + b_L) + b_{L+1}}_{\text{nonlinear part}} + \underbrace{W_r [\hat{x}^\top \ u^\top \ \xi^\top]^\top + b_r}_{\text{linear part}}, \quad (2.12)$$

where W_r , and b_r are the weight and bias values for the linear part of the network. It can be seen that including both b_{L+1} and b_r in the equation causes over-parametrization, thus $b_r \equiv 0$ is applied. Thus the parameters of the neural network are extended to $\phi = \text{vec}(\{W_i, b_i\}_{i=0}^{L+1}, W_r)$. The same initialization is kept as in the additive case, namely $\{W_i, b_i\}_{i=0}^L$ are initialized randomly, and W_{L+1} , b_{L+1} are initialized as zeros. The linear part of the residual network should be considered as $W_r [\hat{x}^\top \ u^\top \ \xi^\top]^\top = \xi$, which can be achieved by setting

$$W_r = \begin{bmatrix} 0_{n_x \times n_x} & 0_{n_x \times n_a} & 0_{n_x \times n_u} & I_{n_x} \\ 0_{n_a \times n_x} & 0_{n_a \times n_a} & 0_{n_a \times n_u} & 0_{n_a \times n_x} \end{bmatrix}, \quad (2.13)$$

where I_{n_x} notes the $n_x \times n_x$ identity matrix.

2.5 LFR-based model augmentation structure

As detailed in Section 2.3, and Section 2.4, a few different model augmentation structures are used in the literature, and it is not intuitive which structure could provide better results because it can be system-dependent. Hence there is a need for a generalized structure that can be efficiently tuned to capture model uncertainties and unmodeled

effects at the same time. We propose a promising method based on *Linear Fractional Representations* (LFRs), because the modular framework of LFRs makes it possible to augment the FP dynamics in a flexible, generalized form. Another advantage of LFR-based augmentation is that there are various established control techniques available for classical LFRs [15, 16], thus a huge advantage of the proposed methodology is that the resulting models can be used for control design directly. A schematic illustration of classical LFRs is visible in Figure 2.3.

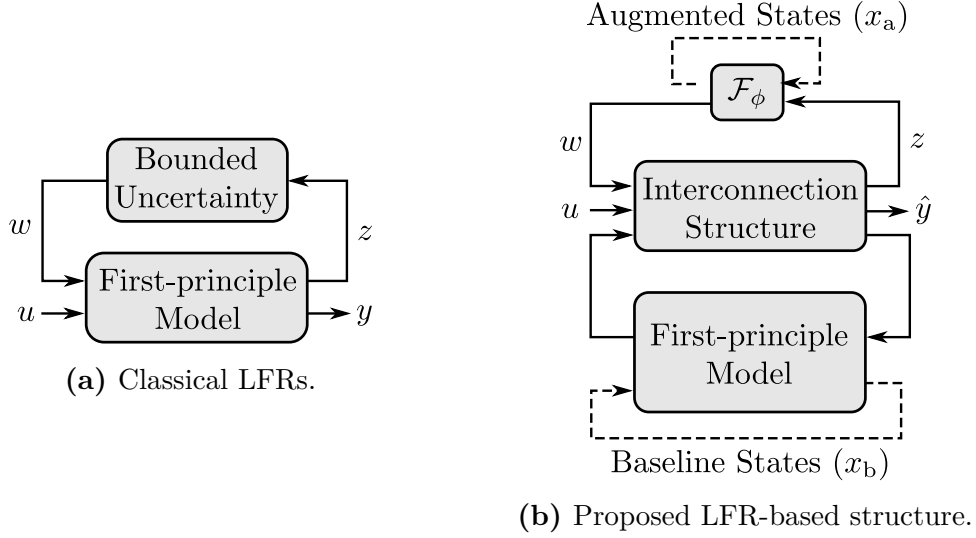


Figure 2.3: Schematic illustration of classical LFRs (a), and the proposed LFR-based model augmentation concept (b).

To express the input-output relations of the LFR-based augmented model, first, we start from the baseline model in (2.3). Then, latent variables are introduced as $z[k] \in \mathbb{R}^{n_z}$, and $w[k] \in \mathbb{R}^{n_w}$. The augmented state transition and the interconnection of the latent variables can be expressed with

$$\begin{bmatrix} x_a^+ \\ w \end{bmatrix} = \mathcal{F}_\phi(x_a, z), \quad (2.14)$$

where $\mathcal{F}_\phi$ is implemented as a multi-layer, fully connected feedforward neural network with $\phi \in \mathbb{R}^{n_\phi}$ containing its parameters. Then, the baseline state transition equation can be expressed as

$$x_b^+ = f_\theta(x_b, u) + B_w w. \quad (2.15)$$

Now the LFR outputs are formalised as

$$z = C_z x_b + D_{zu} u + D_{zw} w, \quad (2.16)$$

$$\hat{y} = h_\theta(x_b, u) + D_{yw} w. \quad (2.17)$$

In equations (2.15), (2.16), and (2.17) the matrices $B_w \in \mathbb{R}^{n_x \times n_w}$, $C_z \in \mathbb{R}^{n_z \times n_x}$, $D_{zu} \in \mathbb{R}^{n_z \times n_u}$, $D_{zw} \in \mathbb{R}^{n_z \times n_w}$, and $D_{yw} \in \mathbb{R}^{n_y \times n_w}$ contain parameters that are co-estimated with

the ANN weight and bias values in ϕ during the optimization. The LFR-based model augmentation concept is illustrated in Figure 2.3.

If $D_{zw} = 0$, then this is a hybrid additive structure where augmentation can happen both at the state and output level. It can be easily demonstrated that the proposed LFR-based approach is a generalization of the additive model augmentation as shown in Appendix B.1. Currently, the LFR-based formulation is not a direct generalization of the multiplicative approach, however, it offers significant advantages over the multiplicative structure. For instance, the resulting model structure is more interpretable physically and can be directly applied to control design. Moreover, with the introduction of tunable LFR model matrices, improved convergence properties and model accuracy can be achieved (as will be demonstrated empirically in Section 6). Appendix B.1 shows a possible direction for generalizing the LFR-based approach to also include serial model augmentation connections.

The LFR-based structure can be also given in a more compact form, if the FP model is in a *linear time invariant* (LTI) SS form and only static nonlinearities are assumed (thus $n_a = 0$). Then,

$$f_\theta(\hat{x}, u) = A_\theta \hat{x} + B_\theta u, \quad (2.18a)$$

$$h_\theta(\hat{x}, u) = C_\theta \hat{x} + D_\theta u, \quad (2.18b)$$

where $A_\theta \in \mathbb{R}^{n_x \times n_x}$, $B_\theta \in \mathbb{R}^{n_x \times n_u}$, $C_\theta \in \mathbb{R}^{n_y \times n_x}$, and $D_\theta \in \mathbb{R}^{n_y \times n_u}$ are the model matrices and all assumed to depend on θ physical parameter set. Using (2.18), we arrive to the well-known NL-LFR form [22], as

$$\begin{bmatrix} \hat{x}^+ \\ z \\ \hat{y} \end{bmatrix} = \begin{bmatrix} A_\theta & B_\theta & B_w \\ C_z & D_{zu} & D_{zw} \\ C_\theta & D_\theta & D_{yw} \end{bmatrix} \begin{bmatrix} x \\ u \\ w \end{bmatrix}, \quad (2.19a)$$

$$w = \mathcal{F}_\phi(z). \quad (2.19b)$$

Please note that, in practice $D_{zw} \equiv 0$ is applied to prevent algebraic loops in the system, thus avoiding well-posedness problems. Also, by increasing the dimensions of z and w , the capabilities of the model to capture nonlinear effects can be enhanced, thus n_z , and n_w should be selected with thoughtful consideration based on prior physical insight.

Similarly as before, **IO normalization** is essential for improved training convergence. Applying the previously defined transformation matrices, the following structure can be expressed:

$$\begin{bmatrix} x_a^+ \\ w \end{bmatrix} = \mathcal{F}_\phi(x_a, z), \quad (2.20a)$$

$$x_b^+ = f_\theta(x_b, u) + T_x^{-1} B_w w, \quad (2.20b)$$

$$z = C_z T_x x_b + D_{zu} T_u u, \quad (2.20c)$$

$$\hat{y} = h_\theta(x_b, u) + T_y^{-1} D_{yw} w, \quad (2.20d)$$

where the term with D_{zw} is now intentionally left out to prevent algebraic loops. To initialize the model parameters, the same principle is applied as for the additive augmentation structure, hence the weights and bias values of the last layer are set to zero, and the other parameters are initialized randomly in $\mathcal{F}_\phi$ (e.g. according to [20]). For the model matrices, a similar method is proposed as in [22], thus

$$C_z \sim \mathcal{U}(-1, 1), \quad (2.21)$$

$$D_{zu} \sim \mathcal{U}(-1, 1), \quad (2.22)$$

$$B_w = 0_{n_x \times n_w}, \quad (2.23)$$

$$D_{yw} = 0_{n_y \times n_w}, \quad (2.24)$$

where $\mathcal{U}(a, b) \in [a, b]$ denotes a randomly generated number from a uniform distribution with support from a to b .

3 Model training and physics-guided regularization

3.1 Truncated prediction loss and subspace encoder

To train the augmented models, it is important to provide sufficiently exciting input signals that can bring out all essential system features, and cover a reasonably large amount of the state- and input space. To gather such a data set, a thorough experiment design is needed, which is a unique task for each system, as there is no universal guideline. Section 5 presents suitable trajectories for generating training data in case of cars.

Using data sequence $\mathcal{D}_N = \{(y_i, u_i)\}_{i=1}^N$ acquired from (2.2), the aim is to minimize a T -step ahead prediction cost on multiple overlapping subsections, as

$$V_{\mathcal{D}_N}^{(\text{sec})}(\phi) = \frac{1}{N_{\text{sec}}} \sum_{k \in \mathcal{I}_{N_{\text{sec}}}} \frac{1}{T} \sum_{l=0}^{T-1} \|y[k+l] - \hat{y}[k+l|k]\|_2^2, \quad (3.1a)$$

$$\begin{bmatrix} \hat{x}[k+l+1|k] \\ \hat{y}[k+l|k] \end{bmatrix} = \Phi(\hat{x}[k+l|k], u[k+l]), \quad (3.1b)$$

$$\hat{x}[k|k] = \psi_\eta(u_{k-n}^{k-1}, y_{k-n}^k) \quad (3.1c)$$

where $\phi \in \mathbb{R}^{n_\phi}$ contains the weights and biases of the applied neural network in the augmentation structure, $\hat{x}[k+l+1|k]$, and $\hat{y}[k+l|k]$ are determined according to the state transition function and output map of the given structure Φ , respectively (see Section 2 for the description of the structures and their parametrization). Furthermore $\mathcal{I}_{N_{\text{sec}}} \subseteq \{n+1, \dots, N-T+1\}$ are the starting times of the sections s.t. $|\mathcal{I}_{N_{\text{sec}}}| = N_{\text{sec}}$ and the pipe notation $(|)$ is used to denote time indexing inside of the section in terms of (current index|start index).

By using cost-function (3.1), we develop an extension of the SUBNET method [1] under the considered model parameterizations. The aim of the optimization is to minimize $V_{\mathcal{D}_N}^{(\text{sec})}$ by tuning the model parameters with a gradient-based optimization algorithm, e.g. the Adam optimizer [23]. Using the T -step ahead prediction cost brings two main benefits compared to the traditional one-step-ahead prediction loss: (i) forward simulation on the subsections can be calculated parallel, therefore providing less computational demand, and allowing the use of batch optimization algorithms; (ii) using overlapping sections increases cost function smoothness [24], which further enhances the efficiency of gradient-based optimization methods.

Cost-function (3.1) uses the initial state of each subsection for calculating the loss value, however, generally, the state values are not known. To address this issue an encoder network is introduced for estimating $\hat{x}$ at the k^{th} time step from n number of past input and output values as shown in (3.1c), where ψ is implemented as a feedforward neural

network with $\eta \in \mathbb{R}^{n_\eta}$ containing its parameters. Given input and output signals

$$u_{k-n}^{k-1} = \begin{bmatrix} u[k-1] & u[k-2] & \dots & u[k-n] \end{bmatrix} \in \mathbb{R}^{n_u \times n}, \quad (3.2)$$

$$y_{k-n}^k = \begin{bmatrix} y[k] & y[k-1] & \dots & y[k-n] \end{bmatrix} \in \mathbb{R}^{n_y \times (n+1)}, \quad (3.3)$$

the encoder-based state estimation can be expressed as

$$\hat{x}[k|k] = \psi_\eta(u_{k-n}^{k-1}, y_{k-n}^k), \quad (3.4)$$

where $n \in \mathbb{R}$ stands for the number of past input and output pairs used for the state estimation, also called encoder lag. When implemented this way, ψ_η realizes the reconstructibility map of the system. However, a black-box reconstruction of the physical states is generally inefficient. To enhance the method's applicability, we introduce a **physics-based encoder initialization method** in Section 4, which provides an initial guess for the parametrization of ψ_η , and then fine-tunes it during model training. This initialization approach accelerates convergence, as will be demonstrated in Section 7.

IO normalization is also essential for improved training of ψ_η . With the previously defined transformation matrices:

$$\begin{bmatrix} x_b[k|k] \\ x_a[k|k] \end{bmatrix} = \begin{bmatrix} T_x^{-1} & 0_{n_x \times n_a} \\ 0_{n_a \times n_x} & I_{n_a} \end{bmatrix} \psi_\eta(T_u u_{k-n}^{k-1}, T_y y_{k-n}^k). \quad (3.5)$$

One can notice that the expression is not multiplied by $T_{x_a}^{-1}$ from the left, as opposed to previous normalization schemes, because only the ANN part uses the augmented states, thus, back-scaling is unnecessary.

Furthermore, when full-state measurements of the system are available (e.g. for the F1Tenth vehicle, as detailed in Section 5), the initial baseline states are known, therefore the encoder network only estimates the augmented states. This version of the augmented state-space structure with the applied encoder network is illustrated in Figure 3.1.

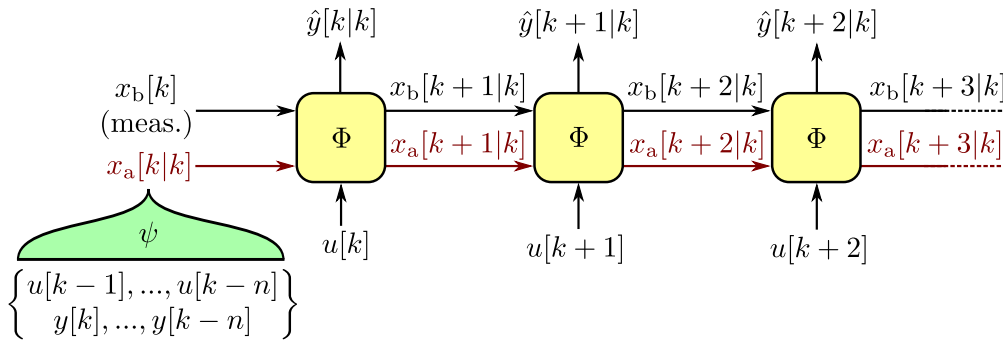


Figure 3.1: The resulting SS model in case of dynamic augmentation, when full-state measurements are available.

3.2 Tuning of physical parameters

Modern mechatronic systems contain a wide range of components for which manufacturers do not always provide sufficient data to model the dynamic relations of the given

subcomponent. As a result, first-principle models often require prior coefficient estimation, which can be inaccurate in the first place or even change over time. To solve this problem, it is reasonable to fit the first-principle model parameters simultaneously with the ANN structure. As we have previously discussed, the baseline model depends on parameters θ for which we assume to know nominal values θ_0 from FP modeling. During the optimization, the goal is to find the optimal parameter values θ^* that accurately describe the system.

The problem with this idea is that the augmented model only optimizes the data fit of the whole system, thus the ANN part can compensate for the effects of parameter uncertainty of θ_0 instead of only compensating for the unmodeled dynamics. This problem can generate not only unrealistic parameters in the FP model, but the increased over-parametrization can result in convergence problems. A possible solution is an addition to the cost function that penalizes large deviations in the physical parameters of the FP model from the nominal values.

Based on [17, 25], a new term is added to the cost function as

$$\|\theta - \theta_0\|_{\Lambda_0} = (\theta - \theta_0)^\top \Lambda_0 (\theta - \theta_0), \quad (3.6)$$

where $\Lambda_0 \in \mathbb{R}^{n_\theta \times n_\theta}$ contains the relative weights for penalizing the deviations from θ_0 . The modified cost function can be expressed as

$$V_{\mathcal{D}_N}^{(\text{reg})}(\phi, \theta) = V_{\mathcal{D}_N}^{(\text{sec})}(\phi, \theta) + \alpha \|\theta - \theta_0\|_{\Lambda_0}, \quad (3.7)$$

where $\alpha \in \mathbb{R}$ is a regularization coefficient. According to [17], a possible selection for Λ_0 is an identity matrix, but a more meaningful selection is a normalization weight for each parameter based on its nominal value, similarly as in [25]. Thus the weight matrix is composed as

$$\Lambda_0 = \text{diag} \left(1/\theta_{0,i}^2 \right), \quad (3.8)$$

for $i = 1, \dots, n_\theta$. This way the regularization cost term results in the following values for specific parameters:

$$\alpha \|\theta_0 - \theta_0\|_{\Lambda_0} = 0, \quad (3.9)$$

$$\alpha \|2\theta_0 - \theta_0\|_{\Lambda_0} = \alpha. \quad (3.10)$$

It is also visible from (3.6), that the value of α determines how much the network can deviate from the nominal parameter set, relative to the change in T -step-ahead prediction cost (3.1).

This method of regularization-based estimation is advantageous in all the following cases:

- If the ANN structure can not identify the unmodeled dynamics, the accuracy of the first-principle model can still be improved by finding the accurate physical parameters.

- If the ANN structure can not identify the unmodeled dynamics, the accuracy of the first-principle model can still be improved by finding the accurate physical parameters.

3.3 Orthogonalization-based regularization

The main problem with the additive model augmentation structure (detailed in Section 2.3) is that the ANN part can compensate for the effects of parameter uncertainty instead of focusing on identifying the unmodeled dynamics. Since $\mathcal{F}_\phi$ and f_θ are connected parallel, as a result, the optimal ϕ^* , θ^* parameters are not unique as shown by [18]. This practically means that different ϕ , θ parameters can result in the same input-output behavior. It does not always decrease the performance of the model, but having a non-identifiable parametrization means that there are gradient directions that are indistinguishable from others. This leads to solver convergence and larger chances of getting stuck in a local minimum. During empirical evaluations, we have experienced that this problem mainly influences the static additive structure, as the introduced augmented states in the dynamic structure can compensate for these effects. Therefore, in this section, only the static additive model augmentation structure is considered.

The main problem is that f_θ and $\mathcal{F}_\phi$ can both capture the known dynamics, instead of the ANN part only identifying the unmodeled part of the governing equations. By adding a new term to the cost function based on [18], f_θ can be forced to prioritize the fitting of the known dynamics. It is achieved by regularizing the output of $\mathcal{F}_\phi$ in the subspace of f_θ , thus it can be interpreted as a targeted ℓ_2 regularization of the ANN that only penalizes the directions of ϕ which generates output in the subspace of the model f_θ .

In [18], the approach is derived for feedforward control, but it can be easily modified for identification problems if the FP model is *linear in the parameters* (LIP):

$$\hat{x}^+ = f_\theta(\hat{x}, u) = \tilde{f}(\hat{x}, u)\theta. \quad (3.11)$$

If the training data sequence includes information about the state, thus $\mathcal{D}_N = \{y_i, x_i, u_i\}_{i=0}^N$, the following term can be expressed:

$$\underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}}_{X^+} = \underbrace{\begin{bmatrix} \tilde{f}(x_0, u_0) \\ \tilde{f}(x_1, u_1) \\ \vdots \\ \tilde{f}(x_{N-1}, u_{N-1}) \end{bmatrix}}_{\tilde{F}(X, U)} \theta + \underbrace{\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{bmatrix}}_E, \quad (3.12)$$

where $X = [x_0^\top \ x_1^\top \ \cdots \ x_{T-1}^\top]^\top \in \mathbb{R}^{Nn_x}$, and U can be defined similarly. E matrix collects the error values of the FP model state transition, which is minimized during the optimization process, but we do not currently deal with that. An explicit base for the output space of f_θ for the states and inputs in (X, U) can be calculated by taking the

reduced *singular value decomposition* (SVD) of $\tilde{F}(X, U) \in \mathbb{R}^{Nn_x \times n_\theta}$ ($Nn_x > n_\theta$ is assumed to be true) as

$$\tilde{F}(X, U) = U_{X,U} \Sigma_{X,U} V_{X,U}^\top, \quad (3.13)$$

where $U_{X,U} \in \mathbb{R}^{Nn_x \times n_\theta}$, $\Sigma_{X,U} \in \mathbb{R}^{n_\theta \times n_\theta}$, and $V_{X,U} \in \mathbb{R}^{n_\theta \times n_\theta}$. Thus (3.12) can be reformulated as

$$X^+ = U_{X,U} \Sigma_{X,U} V_{X,U}^\top \theta + E. \quad (3.14)$$

The columns of $U_{X,U}$ form a basis for the output space of f_θ for the particular state and input values in X, U . The projection matrix onto the subspace spanned by these basis vectors can be expressed as

$$\Pi_{X,U} = U_{X,U} U_{X,U}^\top. \quad (3.15)$$

Thus the component of $\mathcal{F}_\phi$ that lies in the subspace spanned by $U_{X,U}$ can be given by $\Pi_{X,U} \mathcal{F}_\phi(X, U)$, where $\mathcal{F}_\phi(X, U) = [\mathcal{F}_\phi(x_0, u_0)^\top \cdots \mathcal{F}_\phi(x_{N-1}, u_{N-1})^\top]^\top \in \mathbb{R}^{Nn_x}$. This component is then used for constructing a new regularization term which is added to the cost function as

$$V_{\mathcal{D}_N}^{(\text{orth})}(\phi, \theta) = V_{\mathcal{D}_N}^{(\text{sec})}(\phi, \theta) + \beta \|\Pi_{X,U} \mathcal{F}_\phi(X, U)\|_2^2, \quad (3.16)$$

where $\beta \in \mathbb{R}$ is the orthogonalization coefficient. This addition to the cost function promotes orthogonality between f_θ and $\mathcal{F}_\phi$, as a result, hence the ANN structure prioritizes identifying the unmodeled effects. However, the value of β needs to be carefully selected, because too high or too low values can cause unwanted behavior. With too large β the optimizer focuses more on promoting orthogonality between the baseline model and the ANN part, rather than producing an accurate model. On the other hand, too small β values can result in not promoting the orthogonality between f_θ and $\mathcal{F}_\phi$ enough, thus, the competition between the two layers remains, while still resulting in an enlarged computational demand.

Another advantage of the decomposition shown in (3.14) is that the projection matrix $\Pi_{X,U}$ is independent of the physical parameters since $\tilde{F}$ depends on the state and input values. This practically means that the SVD can be calculated before the optimization process begins, thus lowering the computational demand of the method. This simplification can only be applied when full-state measurements are available (hence X is known), otherwise an approximate $\tilde{X}$ data set can be constructed by forward simulating the FP model on the training data.

For large training sets, estimation of the gradient with respect to the model parameters on $\|\Pi_{X,U} \mathcal{F}_\phi(X, U)\|_2^2$, can still take undesirably much time. In this case a small segment of $\mathcal{D}_N$ can be selected to promote orthogonality only on that specific part. In practice, this section should still be chosen to cover a relatively large domain of the state- and input space.

3.4 Generalizing the orthogonalization-based approach for non-linear first-principle models

A big limitation to the orthogonalization-based method derived in Section 3.3 is that it only considers first-principle models that are linear in the parameters. To generalize the method, the Taylor series expansion of the FP state transition function is taken w.r.t. the physical parameters, as

$$f_\theta(\hat{x}, u) \approx f_\theta(\hat{x}, u)\big|_{\theta=\theta_0} + \underbrace{\frac{\partial f_\theta(\hat{x}, u)}{\partial \theta}\big|_{\theta=\theta_0}}_{J_f(\hat{x}, u)} (\theta - \theta_0), \quad (3.17)$$

where $\partial f_\theta(\hat{x}, u)/\partial \theta \in \mathbb{R}^{n_x \times n_\theta}$ is defined as the Jacobian matrix of $f_\theta(\hat{x}, u)$ w.r.t. θ parameters. This approximation is valid if the nominal parameters can be considered accurate enough, hence the optimal θ^* is close to the nominal values of θ . Now if the terms in (3.17) are unfolded, we arrive to the expression of

$$f_\theta(\hat{x}, u) \approx J_f(\hat{x}, u)\theta + f_{\theta_0}(\hat{x}, u) - J_f(\hat{x}, u)\theta_0, \quad (3.18)$$

where the first term is linear in θ , while the second and third terms can be considered affine. Thus a similar simulation-like term can be expressed on the training data sequence, as in (3.12):

$$\underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}}_{X^+} = \underbrace{\begin{bmatrix} J_f(x_0, u_0) \\ J_f(x_1, u_1) \\ \vdots \\ J_f(x_{N-1}, u_{N-1}) \end{bmatrix}}_{J_f(X, U)} \theta + \underbrace{\begin{bmatrix} f_{\theta_0}(x_0, u_0) - J_f(x_0, u_0)\theta_0 \\ f_{\theta_0}(x_1, u_1) - J_f(x_1, u_1)\theta_0 \\ \vdots \\ f_{\theta_0}(x_{N-1}, u_{N-1}) - J_f(x_{N-1}, u_{N-1})\theta_0 \end{bmatrix}}_{\varphi(X, U)} + \underbrace{\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{bmatrix}}_E, \quad (3.19)$$

where $J_f(X, U)$, and $\varphi(X, U)$ notations are introduced for simplicity. With an extended parameter vector (noted with $\tilde{\theta} \in \mathbb{R}^{n_\theta+1}$), the two terms in (3.19) can be combined as

$$X^+ = \underbrace{\begin{bmatrix} J_f(X, U) & \varphi(X, U) \end{bmatrix}}_{\tilde{J}_f(X, U)} \underbrace{\begin{bmatrix} \theta \\ 1 \end{bmatrix}}_{\tilde{\theta}} + E. \quad (3.20)$$

It is visible that equation (3.20) represents a similar structure as equation (3.12), hence the same orthogonalization-based cost function can be derived for it, but instead of taking the reduced SVD of $\tilde{F}(X, U)$, now it should be calculated for $\tilde{J}_f(X, U)$. Apart from that, the other aspects of the method remain the same for general nonlinear FP models.

3.5 Further options for improving training stability

In the case of both multiplicative structures and the dynamic LFR-based structure, one can notice that $\mathcal{F}_\phi$ represents a neural state-space model on its own. For example, in

Equation (2.20a) x_a can be interpreted as the model state, w as the model output, and z as the input of $\mathcal{F}_\phi$. During the optimization process, *stochastic gradient descent* (SGD) can result in such ϕ parameters that cause instabilities when forward simulating the model. This effect naturally harms the training progress, hence, we propose $\mathcal{F}_\phi$ to be implemented as a *contracting recurrent equilibrium network* (c-REN). C-RENs are a special type of network, that guarantees contracting, which is a strong form of stability [26]. A direct parametrization can be given for c-RENs based on [27], to enable the utilization of unconstrained optimization algorithms (e.g. SGD).

Also, it is important to emphasize that not every system is prone to produce instabilities during training. When possible, it is recommended to use simple feedforward neural networks, as it is much faster to train them. But when instabilities occur, it is recommended to utilize the aforementioned contracting RENs.

4 Encoder initialization method

4.1 Generalized encoder network

As discussed in Section 3.1, the initial states of each subsection in (3.1) should be known to utilize the multiple shooting-based cost function. If full-state measurements are available, the encoder network is only utilized to estimate the initial augmented states. For the general case, when full-state measurements can not be guaranteed, the encoder network is applied to estimate both the initial baseline and augmented states based on past input and output values. In general, we aim to identify the reconstructability map in the following form:

$$\hat{x}[k] = \psi_{\eta}(u_{k-n}^k, y_{k-n}^k) = \begin{bmatrix} P_{b,u} \\ P_{a,u} \end{bmatrix} u_{k-n}^k + \begin{bmatrix} P_{b,y} \\ P_{a,y} \end{bmatrix} y_{k-n}^k + \tilde{\psi}_{\tilde{\eta}}(u_{k-n}^k, y_{k-n}^k), \quad (4.1)$$

where formally $u_{k-n}^k \in \mathbb{R}^{n_u(n+1)}$ can be interpreted as a vector, thus $P_{b,u} \in \mathbb{R}^{n_x \times n_u(n+1)}$, $P_{a,u} \in \mathbb{R}^{n_a \times n_u(n+1)}$ are optimizable parameters of the encoder, furthermore y_{k-n}^k , $P_{b,y}$, and $P_{a,y}$ can be defined similarly. Also, $\tilde{\psi}_{\tilde{\eta}} : \mathbb{R}^{n_u(n+1)} \times \mathbb{R}^{n_y(n+1)} \rightarrow \mathbb{R}^{n_x+n_a}$ is implemented as a fully-connected, feedforward neural network with $\tilde{\eta}$ parameters. Please note that the value of $u[k]$ is unnecessary for estimating $\hat{x}[k]$, thus it is not used in the real implementation of the method, however, past input values are still noted as u_{k-n}^k for convenience. The modified encoder structure is illustrated in Figure 4.1.

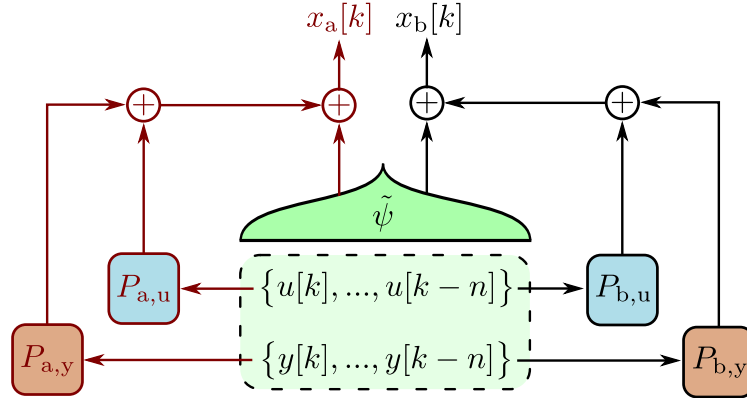


Figure 4.1: The proposed encoder structure. The network can be interpreted as a residual network with $P_{b,u}$ and $P_{b,y}$ weights corresponding to the baseline state estimation, while $P_{a,u}$ and $P_{a,y}$ to the augmented state reconstruction.

As one might see ψ_{η} in (4.1) can be interpreted as a residual network under these considerations with $\eta = \text{vec}(P_{b,u}, P_{b,y}, P_{a,u}, P_{a,y}, \tilde{\eta})$ optimizable parameters. However, black-box identification of the reconstructability map is rather inefficient, thus we aim to provide an initial state estimation based on the available first principle model, then augment that map with a deep-learning-based part. To achieve this structure, $P_{b,u}$ and $P_{b,y}$ are utilized to implement the physics-based state estimation of the baseline state x_b , while $\tilde{\psi}_{\tilde{\eta}}$ is applied to compensate for possible model mismatches and to estimate the

augmented states together with $P_{a,u}$ and $P_{a,y}$. Similarly, for the augmentation structures, the weights and bias values of the last layer of $\tilde{\psi}$ are initialized to zeros in the neural network part, while the other ANN parameters are initialized randomly, $P_{a,u}$ and $P_{a,y}$ are initialized as small random values, e.g. $P_{a,u} \sim \mathcal{U}(-10^{-3}, 10^{-3})$, and similarly for $P_{a,y}$. This way, the initialized encoder network only estimates the baseline states, while initially, the augmented states are approximated to be near zero.

An LTI state reconstruction is considered, detailed in Sections 4.3 to initialize $P_{b,u}$ and $P_{b,y}$ for estimating the physics-based part. The method is also extended for considering process noise in Appendix C. Using LTI models for state reconstruction is convenient because they have well-understood mathematical properties, which simplify both analysis and implementation. LTI models provide a balance between simplicity and accuracy for state reconstruction, with reduced computational demand compared to the nonlinear case. Furthermore, approximations are acceptable since it is only used for initializing the encoder structure, and then parameters are tuned during model training.

4.2 Generalization for NL FP models

As discussed previously, FP models are considered in LTI-SS form for the physics-based state reconstruction. The baseline model is either in LTI form in the first place, or an approximated model can be provided based on the local linearization of the NL model.

Let us assume we have a DT NL first-principle model in the form of (2.3), and an equilibrium point $(x^*, u^*) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_u}$ is chosen, corresponding to

$$f_\theta(x^*, u^*) = 0, \quad (4.2a)$$

$$h_\theta(x^*, u^*) = y^*, \forall k \in \mathbb{N}. \quad (4.2b)$$

Then local linearization around that equilibrium point can be provided by calculating the following Jacobian matrices

$$A^* := \frac{\partial f_\theta(x^*, u^*)}{\partial x}, \quad B^* := \frac{\partial f_\theta(x^*, u^*)}{\partial u}, \quad (4.3a)$$

$$C^* := \frac{\partial h_\theta(x^*, u^*)}{\partial x}, \quad D^* := \frac{\partial h_\theta(x^*, u^*)}{\partial u}. \quad (4.3b)$$

To get an LTI-SS model, the following variables are introduced: $\tilde{x} = x - x^*$, $\tilde{u} = u - u^*$, and $\tilde{y} = y - y^*$. Then

$$\tilde{x}[k+1] = A^* \tilde{x}[k] + B^* \tilde{u}[k], \quad (4.4a)$$

$$\tilde{y}[k] = C^* \tilde{x}[k] + D^* \tilde{u}[k]. \quad (4.4b)$$

After obtaining the system matrices of the linearized system, $P_{b,u}$ and $P_{b,y}$ can be calculated to reconstruct the initial state $\tilde{x}[i]$ based on $\tilde{u}_{i-n}^i$ and $\tilde{y}_{i-n}^i$. And then $x[i]$ can be calculated as $x[i] = \tilde{x}[i] + x^*$. Alternatively, the bias terms in (4.1) can be initialized

with x^* . Keep in mind that the physics-based state reconstruction will only be accurate around the equilibrium point (x^*, u^*) , but the neural network can compensate for these inaccuracies.

4.3 State reconstruction for LTI models

Let us first consider the first-principle dynamics in discrete-time *linear time-invariant* (LTI) state-space form as

$$x[k+1] = f(x, u) = Ax[k] + Bu[k], \quad (4.5a)$$

$$y[k] = h(x, u) = Cx[k] + Du[k], \quad (4.5b)$$

where $x[k] \in \mathbb{R}^{n_x}$ is the state vector, $u[k] \in \mathbb{R}^{n_u}$ is the control input, while $y[k] \in \mathbb{R}^{n_y}$ is the output. The FP model is either originally in LTI form or the nonlinear dynamics are linearized according to Section 4.2.

From (4.5a) and (4.5b), the following expression can be derived:

$$y[k+1] = C(Ax[k] + Bu[k]) + Du[k+1] = CAx[k] + CBu[k] + Du[k+1], \quad (4.6)$$

$$y[k+2] = CA^2x[k] + CABu[k] + CBu[k+1] + Du[k+2], \quad (4.7)$$

$\vdots$

$$y[k+n] = CA^n x[k] + \begin{bmatrix} D & CB & CAB & \dots & CA^{n-1}B \end{bmatrix} \underbrace{\begin{bmatrix} u[k+n] \\ u[k+n-1] \\ u[k+n-2] \\ \vdots \\ u[k] \end{bmatrix}}_{u_k^{k+n}}. \quad (4.8)$$

Now using (4.8), y_k^{k+n} can be expressed as a function of $x[k]$ and u_k^{k+n} :

$$y_k^{k+n} = \begin{bmatrix} y[k+n] \\ y[k+n-1] \\ \vdots \\ y[k+1] \\ y[k] \end{bmatrix} = \underbrace{\begin{bmatrix} CA^n \\ CA^{n-1} \\ \vdots \\ CA \\ C \end{bmatrix}}_{O_n} x[k] + \underbrace{\begin{bmatrix} D & CB & CAB & \dots & CA^{n-1}B \\ 0 & D & CB & \dots & CA^{n-2}B \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & CB \\ 0 & 0 & 0 & 0 & D \end{bmatrix}}_{\Gamma_n} u_k^{k+n}. \quad (4.9)$$

Since the system is time-invariant, shifting back the indices n -times can be used for expressing y_{k-n}^k , hence

$$y_{k-n}^k = O_n x[k-n] + \Gamma_n u_{k-n}^k. \quad (4.10)$$

From (4.10), the state vector can be expressed as

$$x[k-n] = O_n^{-1} \left(y_{k-n}^k - \Gamma_n u_{k-n}^k \right), \quad (4.11)$$

where we assume that O_n^{-1} exists, but since $O_n \in \mathbb{R}^{(n+1)n_y \times n_x}$, this is not the standard matrix inverse. When the SS representation of the system is observable, $\text{rank } O_n = n_x$ when n is greater or equal to the lag of the system, thus in that case O_n always has a left inverse. One approach to compute it is to take the Moore-Penrose pseudoinverse of O_n .

Similarly to (4.8), $x[k+n]$ can be also expressed as

$$x[k+n] = A^n x[k] + \underbrace{\begin{bmatrix} 0 & B & AB & \cdots & A^{n-1}B \end{bmatrix}}_{\gamma_n} u_k^{k+n} \quad (4.12)$$

and then shifting back time, x_k can be given by

$$x[k] = A^n x[k-n] + \gamma_n u_{k-n}^k. \quad (4.13)$$

Using (4.13), $x[k]$ can be expressed in terms of lagged instances of u and y only, if $x[k-n]$ is substituted by (4.11):

$$x[k] = A^n O_n^{-1} \left(y_{k-n}^k - \Gamma_n u_{k-n}^k \right) + \gamma_n u_{k-n}^k. \quad (4.14)$$

Note that by rearranging (4.14), a direct parametrization of $P_{b,u}$ and $P_{b,y}$ can be given by

$$P_{b,u} = -A^n O_n^{-1} \Gamma_n + \gamma_n, \quad (4.15)$$

$$P_{b,y} = A^n O_n^{-1}. \quad (4.16)$$

4.4 Simulation example

The physics-based part of the proposed method is now shown through a simple simulation example, hence

$$x_b[k] = P_{b,u} u_{k-n}^{k-1} + P_{b,y} y_{k-n}^n \quad (4.17)$$

state reconstruction is tested. In real implementations, this function will be augmented with an ANN, and the augmented states will be also dealt with.

Mechanical systems often contain the interconnection of springs with different dampings, thus a simple mass-spring system has been chosen. A more complex and detailed identification problem is detailed in Section 7. The position of the mass is denoted with $p(t)$, the velocity with $\dot{p}(t)$. The generalized state vector is $x(t) = \begin{bmatrix} p(t) & \dot{p}(t) \end{bmatrix}^\top$, while the output is the current position of the system. The equation of motion can be expressed in an LTI SS form:

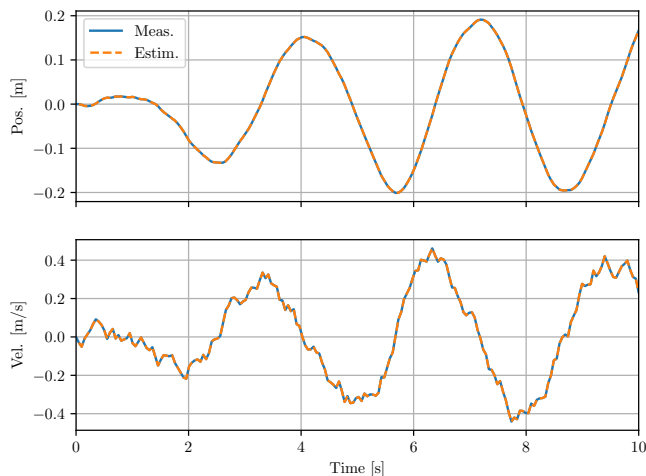
$$\dot{x}(t) = \underbrace{\begin{bmatrix} 0 & 1 \\ -k/m & -b/m \end{bmatrix}}_{A_c} x(t) + \underbrace{\begin{bmatrix} 0 \\ 1/m \end{bmatrix}}_{B_c} u(t), \quad (4.18)$$

$$y(t) = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_C x(t) + \underbrace{0}_D u(t), \quad (4.19)$$

where $m = 5$ kg is the mass of the system, $k = 20$ N/m is the stiffness of the spring and $b = 0.75$ Ns/m is the damping coefficient. The continuous time system is discretized using the Euler method with a sampling time of $T_s = 0.05$ s (the discrete-time matrices are denoted with A and B). The input is randomly generated at each time step from a uniform distribution: $u[k] \sim \mathcal{U}(-5, 5)$, $u(t) = u[k]$ for all $kT_s \leq t < (k+1)T_s$, $k \in \mathbb{Z}$.

The method is tested with 10-second-long simulation data. $P_{b,u}$, and $P_{b,y}$ are assembled using the state-space matrices of the discretized model. As the system is LTI and SISO, $n = n_x - 1 = 1$ is a sufficient choice for the previous matrices. The results of the estimation is visible in Figure 4.2. The estimated states are always constructed from n number of past input and output values, hence there is no estimation for the first time step.

The formula in (4.14) can be easily reformulated for a *linear least squares* (LLS) problem, as proposed by [28]. This way, instead of using the model matrices to construct $P_{b,u}$ and $P_{b,y}$, they are approximated from the training data $\mathcal{D}_N$. The two methods are compared on the same data set, however, Figure 4.2 does not show the LLS estimation of the states as there is no visible difference between the two methods. Quantitative results are given in Table 4.1.



	e_p	e_v
Physics-based	0	10^{-16}
LLS	10^{-15}	10^{-14}

Table 4.1: Results of the state reconstruction with different methods. e_p and e_v denote the RMS error of the position and velocity signals respectively.

Figure 4.2: Reconstructing the state from past input and output data with noiseless system dynamics.

The results have shown that both methods can accurately reconstruct the states from past IO data. However, a problem occurred with the LLS estimation when it was evaluated on different data sets, it did not always produce the same accuracy. In contrast, the model-based reconstruction always reproduced the same results. Considering the ability to reproduce the results using different simulation data, this result supports our methodology to initialize the encoder network with physics-based estimation.

5 Modeling and simulation of the F1Tenth vehicle

To test the proposed theoretical identification algorithms, we aim to identify the dynamics of the F1Tenth vehicle, by augmenting a simplified mechanical model that describes the main characteristics of the vehicle. The mechanical model contains large amounts of physical parameters, which are all estimated based on [29], however, parameter uncertainties can be present. Therefore, this example is ideal for testing the augmentation framework, when the parameters of the first-principle model are tuned simultaneously with the neural network parameters, furthermore to test the proposed physics-based regularization terms.

F1Tenth (shown in Figure 5.1) is a 1/10 scale model of an electric car, mainly developed as a test platform for various automotive applications [30]. A *brushless direct current* (BLDC) motor drives all four wheels of the vehicle, and a servo motor controls the steering. A *Vedder Electronic Speed Controller* (VESC) unit is used to drive the actuators. The VESC unit also has a built-in IMU and speed sensor. For measuring the position and the orientation of the vehicle, an OptiTrack motion capture system is used, which is capable of high-accuracy measurement of these variables based on passive markers placed on the car. Full-state measurement of the system is available, as detailed, hence x_b is measured, and the encoder network only estimates the augmented states.

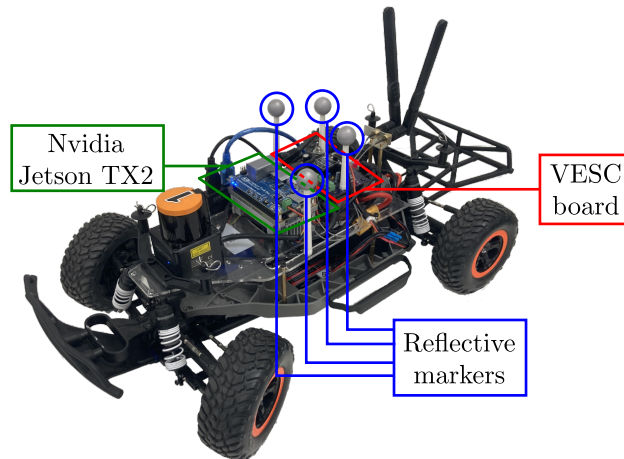


Figure 5.1: F1Tenth vehicle with the data acquisition devices shown.

5.1 Modeling considerations

One of the most commonly used simplified models of car-like vehicles is the single-track model, often referred to as the bicycle model [31]. To model the dynamics of the F1Tenth platform, the so-called dynamic single-track model has been used in the literature, because it is a compromised model between accuracy and computational cost. The model is

illustrated in Figure 5.2, and it is defined by the following equations [32]:

$$\dot{x} = v_\xi \cos \varphi - v_\eta \sin \varphi, \quad (5.1a)$$

$$\dot{y} = v_\xi \sin \varphi + v_\eta \cos \varphi, \quad (5.1b)$$

$$\dot{\varphi} = \omega, \quad (5.1c)$$

$$\dot{v}_\xi = \frac{1}{m} (F_\xi + F_\xi \cos \delta - F_{f,\eta} \sin \delta + m v_\eta \omega), \quad (5.1d)$$

$$\dot{v}_\eta = \frac{1}{m} (F_{r,\eta} + F_\xi \sin \delta - F_{f,\eta} \cos \delta - m v_\xi \omega), \quad (5.1e)$$

$$\dot{\omega} = \frac{1}{I_z} (F_{f,\eta} l_f \cos \delta + F_\xi l_f \sin \delta - F_\xi l_r), \quad (5.1f)$$

where (x, y) is the position of the *center-of-gravity* (CoG) in the (X, Y) plane, φ is the orientation of the vehicle, which is measured from the X axis. The state variables v_ξ and v_η denote the longitudinal and lateral velocities of the vehicle, respectively, while ω is the yaw rate. Furthermore, l_r and l_f are the distances of the rear and front axis from the CoG, the mass of the vehicle is denoted by m , while the inertia along the vertical axis is I_z .

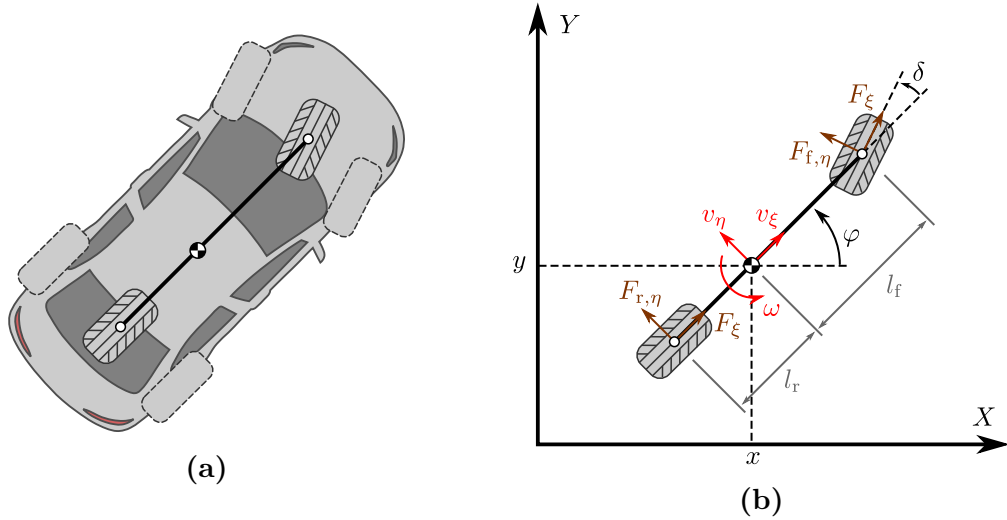


Figure 5.2: Illustrating the simplification of the single track model (a), and the dynamic model with the used coordinate frames and velocity measures (b).

The control inputs of the system are δ steering angle and d motor PWM percentage. As we have derived in Section 2, a DT model is necessary for the augmentation structures, thus (5.1) has been discretized with the forward Euler discretization scheme because no stability losses have been detected during a trial period and it is one of the most computationally effective numerical integration methods.

Instead of the bicycle model, more complex multi-body dynamic models can be also considered [33], but these higher degree-of-freedom models are often too complicated to handle. Moreover, despite their complexity, the previously mentioned modeling approaches are also highly inaccurate for extreme maneuvering situations. Thus we intend

to show that utilizing different model augmentation approaches can result in a reliable and accurate dynamical model, even in case of a simplified first-principle model.

5.2 Drivetrain model

The longitudinal tire force component in the governing equations (5.1) is denoted by F_ξ , and it is generated by the drivetrain. As F_ξ depends on the BLDC motor characteristics, the transmission between the motor shaft and wheels, and even on the friction characteristics of the whole drivetrain system, instead of a complex analytical model, an empirical formula is given according to [29], in the form of

$$F_\xi = C_{m1}d - C_{m2}v_\xi - \text{sign}(v_\xi) C_{m3}, \quad (5.2)$$

where $\text{sign}(\cdot)$ denotes the signum operand. The constants C_{m1} , C_{m2} , and C_{m3} are empirical drivetrain parameters and $d \in [-1, 1]$ is the motor input in terms of PWM percentage.

The model in equation (5.2) approximates the motor characteristics as a first-order system, which is a valid assumption as the BLDC motor dynamics are much faster than the vehicle dynamics. Furthermore, the model can capture the static Coulomb friction, as well as the viscous friction characteristics by using the above-discussed empirical parameters.

5.3 Tire model

As the drivetrain model introduced in Section 5.2 can be used to determine the longitudinal tire force component, the tire model only needs to be applied for the lateral direction. One of the largest challenges in vehicle dynamics is the modeling of tire forces and moments, as these hard nonlinear effects heavily influence the whole system, but developing a compact, yet accurate tire model is nearly impossible. Starting from the 1940s, researchers showed more and more interest in tire modeling, therefore the first mechanical models were introduced [34]. Later on, empirical formulas were developed to describe the tire forces, for example, the well-known Magic Formula, first introduced in [35]. These empirical expressions quickly became the industrial standard for tire modeling, as they can accurately represent the tire forces by only using relatively few parameters. The Magic Formula depends on the side slip angles, which can be expressed for the rear and front tires respectively as

$$\alpha_r = \arctan\left(\frac{\omega l_r - v_\eta}{v_\xi}\right), \quad (5.3)$$

$$\alpha_f = -\arctan\left(\frac{\omega l_f + v_\eta}{v_\xi}\right) + \delta. \quad (5.4)$$

As the F1Tenth vehicle is mainly used in an indoor environment, where v_ξ is usually

magnitudes larger than ωl_r , ωl_f , and v_η , the side slip angles can be approximated as

$$\alpha_r \approx \frac{\omega l_r - v_\eta}{v_\xi}, \quad (5.5)$$

$$\alpha_f \approx -\frac{\omega l_f + v_\eta}{v_\xi} + \delta. \quad (5.6)$$

Furthermore, for small α angles a simplified version of the Magic Formula can be applied, where the lateral tire forces can be expressed as

$$F_\eta \approx C\alpha. \quad (5.7)$$

Thus rear and front lateral tire forces ($F_{r,\eta}$ and $F_{f,\eta}$ respectively) can be calculated as

$$F_{r,\eta} = C_r \frac{-v_\eta + l_r \omega}{v_\xi}, \quad (5.8a)$$

$$F_{f,\eta} = C_f \left(\delta - \frac{v_\eta + l_f \omega}{v_\xi} \right), \quad (5.8b)$$

where C_r and C_f are the cornering stiffness values.

5.4 Physical parameters of the model

[29] proposed various identification experiments to determine the parameters of an F1Tenth electric car. Based on those results, the nominal parameter values with their notations and units are summarized in Table 5.1.

Parameter	Notation	Value	Unit
Weight	m	2.93	kg
Inertia along axis z	I_z	0.0796	kgm ²
Distance between rear axis and CoG	l_r	0.168	m
Distance between front axis and CoG	l_f	0.163	m
Empirical drivetrain parameter #1	C_{m1}	41.796	N
Empirical drivetrain parameter #2	C_{m2}	2.0152	Ns/m
Empirical drivetrain parameter #3	C_{m3}	0.4328	N
Rear cornering stiffness	C_r	41.7372	N/rad
Front cornering stiffness	C_f	29.4662	N/rad

Table 5.1: Identified physical parameters of the F1Tenth car.

5.5 High fidelity simulation environment

To test the proposed model augmentation methods on the F1Tenth small-scaled car, first, we need to generate sufficient training, validation, and test data. A high-fidelity simulation environment is used for data generation to test and compare the methods.

To generate data for training, a high-fidelity physics engine, namely the MuJoCo simulator¹ has been used [36]. MuJoCo is an open-source physics engine developed by Google DeepMind, and it is capable of advanced dynamical simulation with fast and accurate computing. The core functionalities are written in C++ to guarantee efficient and quick calculations, however, a high-level Python API is also available to manage the simulational environment.

The vehicle model has been constructed using the XML scheme of the MuJoCo engine. The main parts of the vehicle, such as the wheels and the chassis have been modeled as rigid bodies with joints connecting them. Furthermore, based on real-world experiments with the F1Tenth car, contact models have been defined between the wheels and the ground such that the model closely resembles the measured behavior of the real car. The digital twin model of the car can be seen in Figure 5.3.

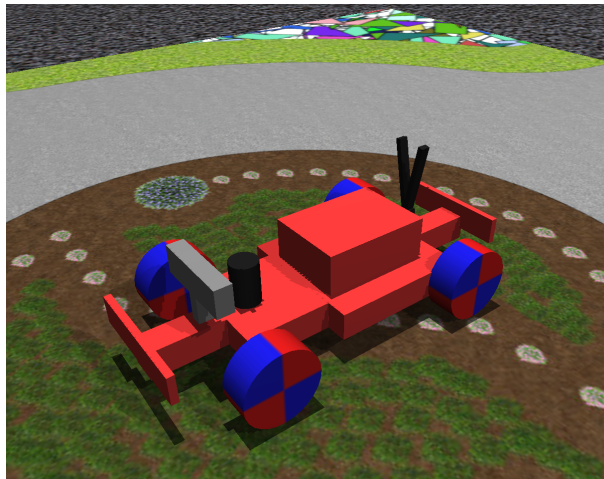


Figure 5.3: Digital twin of the F1tenth vehicle in the MuJoCo environment.

To acquire sufficient data, multiple experiments have been performed with the digital twin model. Two different trajectories (a lemniscate and a circular path, both illustrated in Figure 5.4), with 12 different velocity references have been utilized to achieve robust identification results. With a sampling frequency of $f_s = 40$ Hz, altogether 15985 data points have been recorded. Half of these measurements have been separated into a training set, while the other half into the test set, both trajectories with alternating reference velocities have been included in both. Table 5.2 presents the concatenation properties of the training and test data. Lastly, before concatenating the measurement signals for the training data set, 20% of each measurement series have been randomly selected to form a validation data set, achieving an 80%-20% ratio of the training and validation data points. The state- and input-space coverage of each data set is illustrated with histograms in Figure 5.5.

¹The virtual 3D model of AIMotionLab test arena in SZTAKI, as well as implemented MuJoCo digital twins of various vehicles can be found here: <https://github.com/AIMotionLab-SZTAKI/AIMotionLab-Virtual>

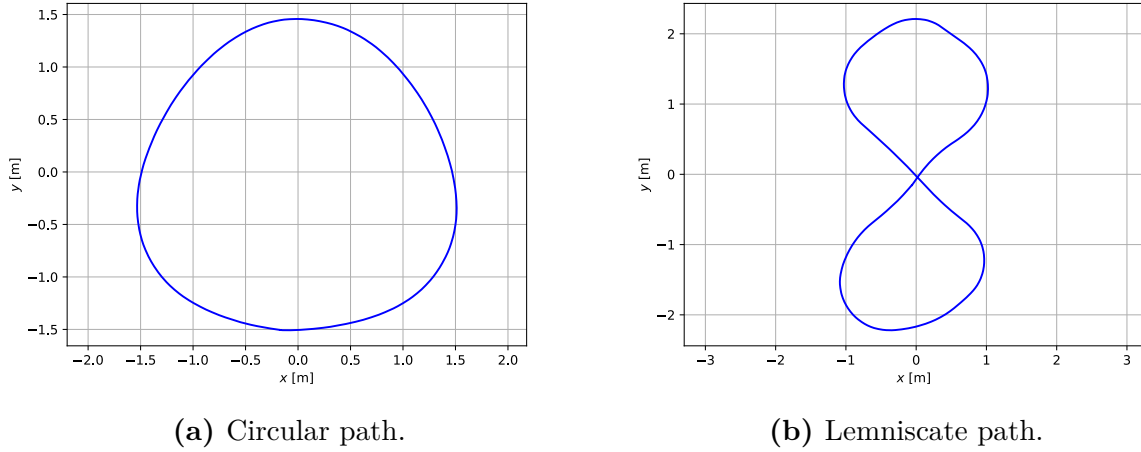


Figure 5.4: Reference trajectories for data generation.

5.6 Advantages of the digital twin model

Using data generated by the MuJoCo simulator is ideal for testing the proposed identification methods because all approaches can be evaluated and compared with noiseless training data, and then *output noise* can be manually added to the training data to achieve a desired *Signal-to-Noise Ratio* (SNR). Furthermore, data acquisition in a high-fidelity simulation environment is much more seamless than real-world experiment design. Although this work is not strictly related to control algorithm development, we can not overlook the potential applications of the derived model augmentation framework. Hence we propose the following workflow:

1. **Data acquisition in MuJoCo:** Gather sufficient training, validation, and test data for deep-learning-based identification of the F1Tenth vehicle.
2. **Identifying the dynamics of F1Tenth:** Using the acquired data sets, identify the dynamics of the simulated F1Tenth vehicle with one (or more) of the selected augmentation structures.
3. **Developing a control method:** Based on the identified model, a control algorithm needs to be formed by considering both the FP model and the augmented model. Various examples can be found in the literature, e.g. [9], [32], etc.
4. **Test the control method in MuJoCo:** After deriving the control law, first, it is reasonable to test it in the MuJoCo simulation environment.
5. **Deploy the final version of the control algorithm for the real F1Tenth:** If the developed algorithm has passed the performance requirements in the MuJoCo simulation environment, the method is ready for testing and further refinements using the real-world experimental setup.

#	Trajectory	Ref. vel. [m/s]	#	Trajectory	Ref. vel. [m/s]
1	lemniscate	0.45	1	circle	0.45
2	circle	0.5	2	lemniscate	0.5
3	lemniscate	0.55	3	circle	0.55
4	circle	0.6	4	lemniscate	0.6
5	lemniscate	0.65	5	circle	0.65
6	circle	0.7	6	lemniscate	0.7
7	lemniscate	0.75	7	circle	0.75
8	circle	0.8	8	lemniscate	0.8
9	lemniscate	0.85	9	circle	0.85
10	circle	0.9	10	lemniscate	0.9
11	lemniscate	0.95	11	circle	0.95
12	circle	1	12	lemniscate	1

Table 5.2: Parameters of the concatenated training data (left), and the test data (right) with alternating trajectory types and reference velocities.

The workflow is also depicted in Figure 5.6. The main advantages of using a high-fidelity simulation environment are *(i)* the training data can be gathered much more conveniently compared to real-world measurements, *(ii)* the developed control algorithm can be first tested in the simulation environment, thus making the process safer, *(iii)* the whole procedure is more time- and cost-effective, which are both key points of current vehicle industrial developments.

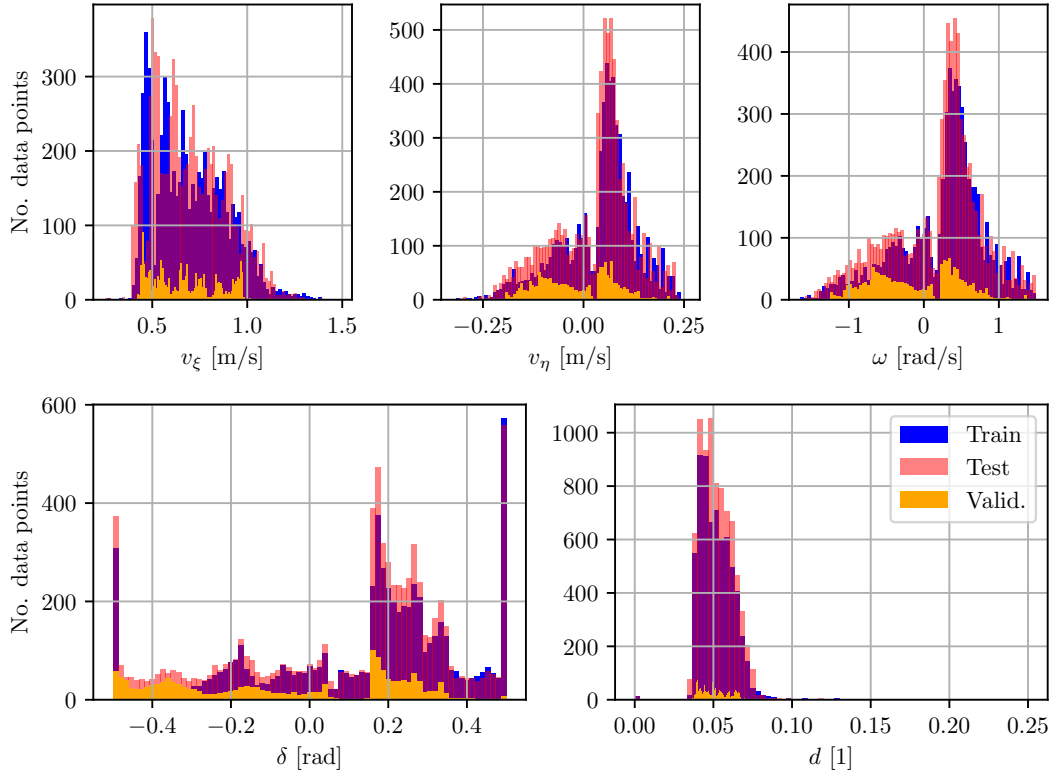


Figure 5.5: Histogram representation of the gathered data sets.

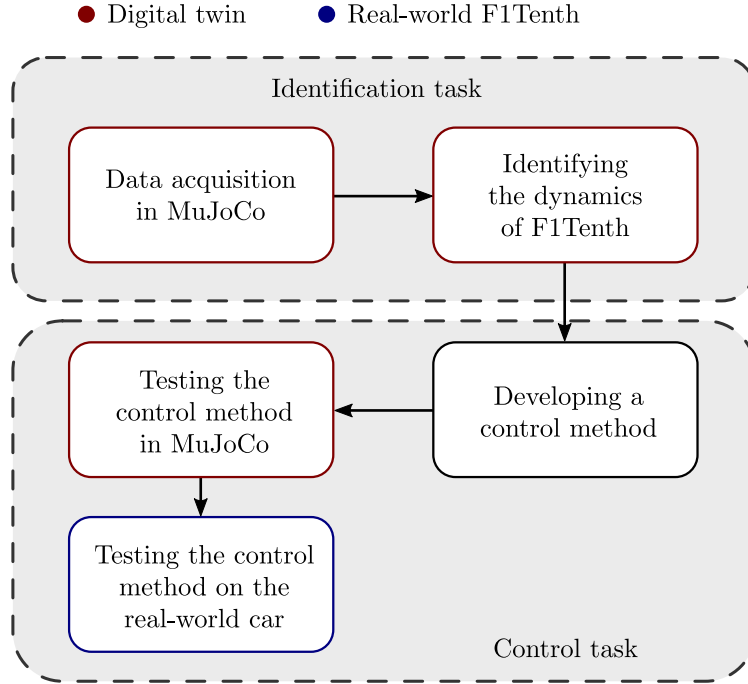


Figure 5.6: The proposed workflow for developing control algorithm for the F1Tenth vehicle based on identified dynamics of the digital twin model.

6 Identifying the dynamics of the F1Tenth vehicle

In this section, the simulated data sets will be used to showcase the capabilities of the proposed model augmentation framework. As one might see, in the equation of motion (5.1d), (5.1e) and (5.1f) contain the drivetrain and tire dynamics, which hold great amounts of uncertainty, hence can be the reason for possible model errors. In contrary (5.1a), (5.1b), and (5.1c) contain simple kinematic relations and can be separated from the rest of the equations. To simplify the neural network structure, we only augment the velocity states of the system, and then with the acquired (v_ξ, v_η, ω) values, the position and orientation can be determined by numerical integration. Thus the dimension of the baseline states are $\dim x_b = 3$, using the previously introduced notation.

Before training the models, several hyperparameters need to be selected. First of all, the length of the truncated subsections (noted with T) in the cost function (3.1). As suggested in [1], T should be a few times the largest characteristic time scale ($\lambda_{\max}$) of the data generation system, as a rule of thumb. For such a choice, the truncated prediction loss (3.1) provides a close approximation of the "regular" prediction error at a low computational cost. Increasing T naturally increases the computational demand. In [2], the time constants of an F1Tenth vehicle were approximated by exciting the system with step inputs, and then the outputs were fitted by a first-order LTI model. This way, the determined maximal characteristic time-scale is $\lambda_{\max} = 0.282$ s. With a sampling time of $t_s = 0.025$ s, the ratio of the largest time scale and the sampling time is

$$T_0 = \lambda_{\max}/t_s = 11.28 \quad (6.1)$$

In case of measured training data with a realistic noise process, the truncation length should be selected around $T \approx (4 \pm 1) T_0$, as a general guideline. For the F1Tenth vehicle, this rule of thumb practically means $T = 33.84 \dots 56.4$ is advised, e.g. $T = 40$ was applied in [2] for black-box identification of the F1Tenth dynamics. Now, as noiseless simulation data is provided, empirical evaluations showed no difference when higher values were used, hence a smaller value, $T = 15$ seemed sufficient for lowering the computational demand. All models have been trained by the Adam algorithm [23] with a batch size of 256 and a learning rate of 10^{-3} .

As detailed in Section 2.2, in case of dynamic structures, an encoder network is utilized for estimating the augmented states. For this task, a relatively small ANN structure proved to be efficient (compared to the $\mathcal{F}_\phi$ ANN structures), hence one hidden layer with 64 nodes is applied with the *hyperbolic tangent* (tanh) activation function. Based on previous black-box identification results [2], an encoder length of $n = 12$ has been selected. However, the remaining hyperparameters (mostly belonging to $\mathcal{F}_\phi$) are heavily influenced by the applied model augmentation structure, hence they will be discussed for each structure individually. The general hyperparameters are summarized in Table 6.1, the remaining hyperparameters will be detailed in the following sections.

Throughout this work, the resulting models are compared based on the *normalized*

Parameter	Notation	Value
Truncation length	T	15
Batch size	N_{batch}	256
Learning rate	γ	10^{-3}
Optimization algorithm	-	Adam
Encoder network hidden layers*	$n_{\text{hl}}^{(\text{enc})}$	1
Encoder network nodes per layer*	$n_{\text{npl}}^{(\text{enc})}$	64
Encoder network activation function*	-	tanh
Encoder length*	n	12

Table 6.1: Generally applied hyperparameters for identifying the F1Tenth dynamics based on MuJoCo simulations. *Only applicable for dynamic augmentation structures.

root-mean-squared (NRMS) error, calculated on the whole test data set, defined as

$$\text{NRMS} = \frac{1}{M} \sum_{m=1}^M \frac{\sqrt{\frac{1}{N} \sum_{k=1}^N (y_m[k] - \hat{y}_m[k])^2}}{\sigma_{y_m}}, \quad (6.2)$$

where M represents the number of channels in the model output, N is the number of simulation steps, and σ_{y_m} notes the standard deviation of y_m . Furthermore, as the test data contains multiple measurement segments, an averaged NRMS error value is presented.

The implemented identification algorithms can be found at this GitHub repository: <https://github.com/gyorokbendeguz/f1tenth-augmentation>.

6.1 Additive augmentation results

In the case of the simple static additive structure, a general ANN architecture of 2 hidden layers with 64 nodes per layer proved to be a sufficient selection for $\mathcal{F}_\phi$ with the tanh activation function. For the dynamic additive case, the network also has to consider the augmented states, thus a slightly more robust ANN has been selected with 3 hidden layers and 64 nodes per layer, also with the tanh activation function. First, the nominal parameters of the FP model (summarized in Table 5.1) were not tuned during the optimization to get a benchmark result. The accuracies of the resulting augmented models are compared to each other and also to the first-principle model in Table 6.2. Although it seems that the FP model is mildly inaccurate, the main reasons are that the drivetrain parameters do not exactly match with the MuJoCo friction model, thus the biggest simulation errors are present in the longitudinal dynamics (as Figure 6.1 shows). Also, keep in mind that the models have been evaluated on multiple 16-18 seconds long data segments and FP models tend to perform worse during these open-loop simulations.

To compare the model performances visually, the outputs of the best-performing augmentation structure and the FP model are both compared on a randomly selected lem-

niscate trajectory of the test data set in Figure 6.1.

Model	No. augmented states	Test NRMS
First-principle model	-	37.91%
Static additive augmentation	-	5.57%
Dynamic additive augmentation	1	2.84%
	2	2.49%
	3	2.88%

Table 6.2: Summary of the augmentation results.

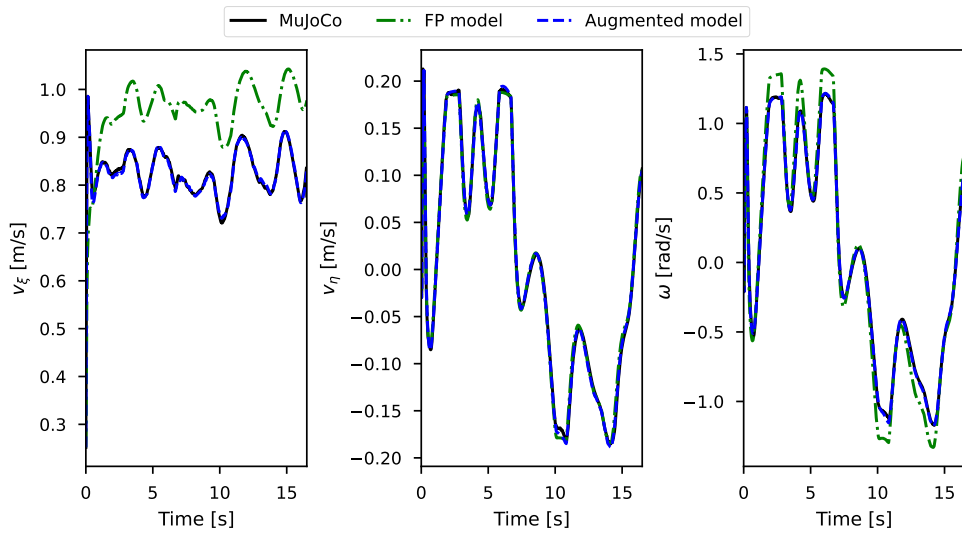


Figure 6.1: MuJoCo simulation outputs compared with the FP model and augmented model outputs. These results are achieved with the dynamic additive structure with two augmented states.

6.1.1 Finding the optimal physical parameters

With the simple additive structure, already quite high accuracy has been reached, thus the tuning of physical parameters can further decrease the model errors to achieve state-of-the-art results. For simplicity, including the FP model parameters into the optimizable parameters is only detailed in case of the additive static augmentation. For the dynamic case, only the results will be presented.

For a first try, all of the physical parameters have been optimized without any regularization to see the results. The nominal parameters and the resulting ones are presented in Table 6.3. It is visible that without regularization, the optimization process resulted in some unphysical values. The most drastic change is in the parameter l_r , which describes the distance between the rear axis and the CoG, thus $l_r < 0$ physically can not be interpreted. It is visible that even a small α value can ensure that the parameters

do not change into physically false values, but as presented in Table 6.4, α should be selected after careful consideration, because it heavily influences the results. A strange phenomenon is also visible in Table 6.4, as higher α values have resulted in such θ physical parameters that are more accurate when only the FP model is evaluated, but with lower α values higher model accuracy can be reached using the augmented structure, however, the FP model’s accuracy decreases that way. The empirical tuning process is also illustrated based on comparing the resulting test errors with different α values in Figure 6.2.

Model	m	I_z	l_r	l_f	C_{m1}	C_{m2}	C_{m3}	C_r	C_f
Nominal values	2.93	0.08	0.168	0.163	41.8	2.02	0.43	41.74	29.47
$\alpha = 1$	2.93	0.08	0.168	0.163	41.78	2.02	0.43	41.73	29.43
$\alpha = 10^{-4}$	3.19	0.08	0.017	0.147	42.13	1.81	0.44	40.76	28.35
No regularization	3.15	0.06	-0.0002	0.112	42.16	1.78	0.45	40.68	28.22

Table 6.3: Nominal and tuned physical parameters of the model. All of the parameters are presented in the appropriate SI unit, listed in Table 5.1.

Regularization type	Augmented model error	FP model error
No regularization ($\alpha = 0$)	4.74%	196.66%
$\alpha = 10^{-5}$	4.78%	192.9%
$\alpha = 10^{-4}$	4.63%	141.33%
$\alpha = 10^{-3}$	4.78%	83.31%
$\alpha = 10^{-2}$	4.80%	44.48%
$\alpha = 10^{-1}$	5.24%	37.89%
$\alpha = 10^0$	5.87%	37.43%
Nominal parameters	5.57%	37.91%

Table 6.4: The augmented and the FP model errors on the test data in case of the static additive structure. The error values are presented in the NRMS metric.

Two main takeaways can be outlined based on this experiment:

- A small value of the regularization coefficient is always beneficial in terms of training convergence, as they do not let θ parameters go completely unphysical, but the optimizer can focus more on fitting the whole system.
- If we intend to use not only the augmented system but the FP model on its own (as some augmentation-based control algorithms require), large α values are more beneficial since the FP model’s accuracy can be enhanced this way, and the complete augmented model still provides quite accurate representation. If only the whole augmented model is important, then a small α value is more beneficial.

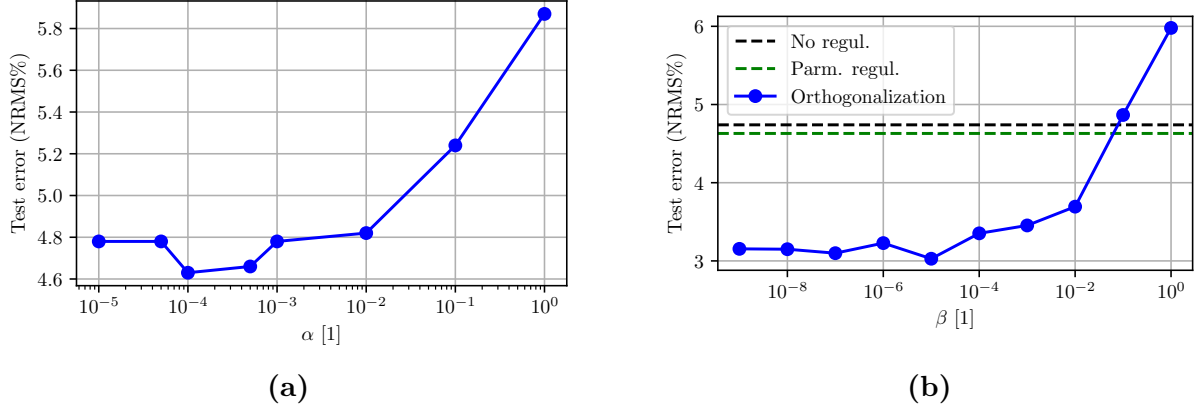


Figure 6.2: Illustrating the empirical process of finding the optimal α regularization coefficient (a), and β orthogonalization coefficient (b).

In case the dynamic additive augmentation structure 2.49% NRMS test error has been achieved with the nominal physical parameters - as shown in Table 6.2. With this high model accuracy, no drastic improvements can be expected when the physical parameters are included in the optimizable parameters, compared to the static augmentation case. With no parameter regularization, the NRMS test error was decreased to 2.21%, furthermore with $\alpha = 10^{-4}$ regularization coefficient 2.13% NRMS error has been achieved, hence optimizing the physical parameters and applying the proposed regularization method proved to be beneficial in case of both static and dynamic additive augmentation.

6.1.2 Results with orthogonalization-based cost function

Apart from the generally applicable parameter regularization method, an orthogonal projection-based method has been also derived in Section 3.3 for the static additive structure. As discussed in Section 3.3, calculating gradients w.r.t. the neural network parameters on the orthogonalization-based cost term is computationally quite intensive. To address this challenge, the orthogonal projection-based loss function component has been only evaluated on four pre-determined training trajectories: the circular path with the lowest and highest reference velocities, and similarly for the lemniscate-shaped path. In Table 5.2, these trajectories are noted with nr. #1, #2, #11, and #12 from the left side of the table.

The orthogonalization coefficient β has been tuned empirically with the L-curve, similarly as in [25]. The results are shown in Figure 6.2, as visible $\beta = 10^{-5}$ coefficient provided the best model accuracy with an NRMS test error of 3.03%.

The previous process has been implemented without using the regularization method discussed in Section 3.2, thus $\alpha = 0$ was applied. Another interesting topic is to see the combination of physical parameter regularization and the orthogonal projection-based approach. By using both methods, the physical parameters are constricted around the nominal values, while the neural network $\mathcal{F}_\phi$ is penalized when identifying behaviors that

are (partly) covered by the first-principle model. Multiple training processes have been performed with different coefficient pairs to test this setup, as shown in Table 6.5. A few coefficient pairs were able to generate better results compared to the case when only the orthogonal projection-based approach was utilized. However, the gained model accuracy is marginal, as the NRMS test error has only decreased by 0.04%. Based on these results, combining the two regularization approaches is not worth the time, which is necessary for tuning the coefficients. However, for a more concise conclusion, other systems will need to be tested in future research.

β [1] \ α [1]	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
10^{-3}	3.60%	3.37%	3.42%	3.57%	3.30%
10^{-4}	3.46%	3.39%	3.28%	3.29%	3.67%
10^{-5}	3.18%	3.16%	3.27%	3.36%	3.39%
10^{-6}	3.31%	3.18%	3.16%	3.01%	3.00%
10^{-7}	3.19%	3.22%	3.06%	3.24%	2.99%

Table 6.5: Combined effects of parameter regularization and orthogonalization. The table contains the test error values of different models, using the NRMS error metric.

6.2 Multiplicative augmentation results

For the testing of the multiplicative augmentation structure, the same principle is followed in our analysis, as in Section 6.1, thus first the physical parameters are not tuned to illustrate the basic properties of the method, and then different training options are tested. Section 6.1 presented a thorough discussion of finding the optimal parameter regularization coefficient, thus now only the final results are being presented. Regarding the hyperparameters of the ANN part, a slightly different structure proved to be more beneficial. First of all, as discussed in Section 2.4, $\mathcal{F}_\phi$ is implemented as a residual network, furthermore the multiplicative structure involves more input data for the ANN, hence a more robust structure has been selected with 3 hidden layers and 128 nodes per layer. For the dynamic multiplicative structure, 1 augmented state proved to be sufficient, and the structure of $\mathcal{F}_\phi$ changed to 4 hidden layers and 64 nodes per layer after a short trial period. The results are summarized in Table 6.6.

6.3 LFR-based augmentation results

The first task for identifying any given system with the LFR-based augmentation structure is to determine the dimensions of the latent variables z , and w . For a general guideline, $n_w = n_x$ is a great starting point, similarly for z , either $n_z = n_x$ or $n_z = n_x + n_u$ can be selected for a starting point, based on whether the input signal influences the unmodeled

Model	Test NRMS
Static multiplicative (nominal phys. parameters)	3.04%
Static multiplicative (no regularization)	2.81%
Static multiplicative ($\alpha = 10^{-5}$)	2.5%
Dynamic multiplicative (nominal phys. parameters)	2.35%
Dynamic multiplicative (no regularization)	2.28%
Dynamic multiplicative ($\alpha = 10^{-5}$)	2.21%

Table 6.6: Test results with the static and dynamic multiplicative augmentation structure using different regularization techniques.

dynamics or not, which can be determined from prior physical insight. After a short trial period, $n_w = 3$, and $n_z = 4$ seemed the optimal choice. For both static and dynamic LFR-based approaches, an ANN structure with 2 hidden layers and 128 nodes per layer has been utilized, furthermore, for the dynamic case an augmented state with dimensions of $n_a = 2$ has provided the best results. The results are summarized in Table 6.7.

Model	Test NRMS
Static LFR-based (nominal phys. parameters)	2.84%
Static LFR-based (no regularization)	2.32%
Static LFR-based ($\alpha = 10^{-4}$)	2.17%
Dynamic LFR-based (nominal phys. parameters)	1.16%
Dynamic LFR-based (no regularization)	0.97%
Dynamic LFR-based ($\alpha = 10^{-4}$)	0.93%

Table 6.7: Test results with the static and dynamic LFR-based augmentation structures using different regularization techniques.

6.4 Comparison with black-box methods

To compare the augmentation results with not only the FP model of the F1Tenth vehicle, black-box models have been trained on the same data set to provide benchmark results. The selected data-driven identification methods are the discrete-time and continuous-time SUBNET approaches, for more details please refer to [1, 37]. The building principles of the augmentation process are similar to these approaches, e.g. truncated cost function, and encoder initialization of the states, hence it provides a great possibility to compare the augmented and black-box results.

6.4.1 Model selection and black-box results

Based on available guidelines and practical examples [2], the hyperparameters have been selected uniformly for both the DT and CT cases, as summarized in Table 6.8. Furthermore, the state derivative normalization factor is chosen as $\tau = 50$ for the CT SUBNET.

Parameter name	Notation	Value
Model order	n_x	6
Encoder length	n	12
Section length	T	40
Batch size	N_{batch}	256
Learning rate	-	10^{-3}
f_ϕ, h_ϕ hidden layers	hl_{dyn}	2
f_ϕ, h_ϕ nodes per layer	n_{dyn}	64
ψ_ϕ hidden layers	hl_{enc}	2
ψ_ϕ nodes per layer	n_{enc}	64

Table 6.8: Hyperparameters of the DT and CT SUBNET structures to identify the dynamics of F1Tenth vehicle based on MuJoCo simulation data.

Model	Test error (NRMS)
DT SUBNET	2.12%
CT SUBNET	1.75%

Table 6.9: Identification results using the DT and CT SUBNET structure.

For training the SUBNET structures, the same data sets have been utilized as in the case of the augmentation structures. To evaluate the model performances, the test errors are presented in Table 6.9. It is visible that the black-box models outperformed most of the augmentation structures. As we have opted for only DT identification of the augmented models, it is only fair to compare those results with the DT SUBNET, as the CT SUBNET belongs to a different model class. The static LFR-based method, the dynamic additive, as well as the dynamic multiplicative structures have achieved similar test errors as the DT SUBNET, although the latter has a slight edge over the others. On the other hand, the dynamic LFR-based structure has resulted in a far more accurate model than all other methods, this approach even outperformed the CT SUBNET by a rather fine margin.

6.4.2 Convergence speed

Another aspect of comparing black-box identification methods with model augmentation approaches is the convergence speed. As one might expect, augmentation structures

provided more accurate initial models, because they rely on the first-principle approximations at the beginning of the training. Moreover, based on our empirical evaluations, model augmentation methods have shown much faster convergence. In Figure 6.3, both the DT and CT SUBNET model's validation loss values are shown together with the static and dynamic LFR-based augmentation structures. In this regard, the CT SUBNET performed far worse than the other methods, only reaching similar validation errors to the other three methods after around 300 epochs. The static and dynamic LFR-based approaches have performed quite similarly, although the latter has a slight edge. Then finally, the DT SUBNET model has achieved moderate results, producing comparable validation errors to the augmentation approaches after around 40 epochs. Also, keep in mind that the fine-tuning of all these models has lasted for more than 1000 epochs, but certain situations (e.g. online learning) require models that can provide close-to-optimal results in a relatively short time. For those types of applications, model augmentation methods have shown huge potential in future research topics.

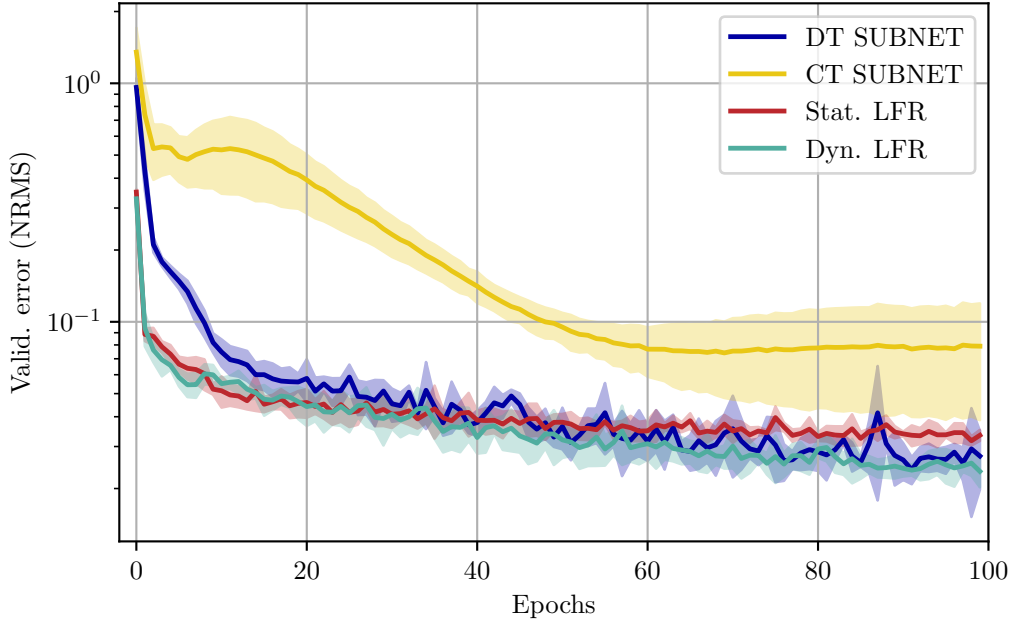


Figure 6.3: First 100 epochs of training with different identification methods. The shown results are an average of 4-4 Monte Carlo runs, the average validation error is shown, while the ± 1 standard deviation is illustrated with the shaded areas.

7 Mass-Spring-Damper identification example

Identifying the dynamics of the F1Tenth vehicle is a great example for illustrating the basic properties of each model augmentation structure and the proposed regularization methods. However, as the physical states of the vehicle are directly measured, the encoder initialization method derived in Section 4 has not been tested yet. For this purpose, a 3 DoF *Mass-Spring-Damper* (MSD) system is considered as illustrated in Figure 7.1. MSD interconnections have great importance in mechanical engineering practice, as they are used for modeling a wide range of components, e.g. the quarter-car model can represent the suspension system of a ground vehicle. These systems are usually described in LTI-SS form, as they are convenient for system analysis and control design. However, various nonlinear effects are often neglected, e.g. nonlinear spring and damping characteristics, friction effects, etc. To compensate for these unmodeled terms, model augmentation is introduced.

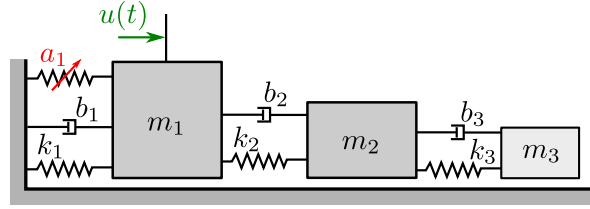


Figure 7.1: The applied mass-spring-damped system with three masses and a nonlinear force term.

The dynamics of the data-generating system can be expressed as

$$\dot{x}(t) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{-k_1 - k_2}{m_1} & \frac{k_2}{m_1} & 0 & \frac{-b_1 - b_2}{m_1} & \frac{b_2}{m_1} & 0 \\ \frac{k_2}{m_2} & \frac{-k_2 - k_3}{m_2} & \frac{k_3}{m_2} & \frac{b_2}{m_2} & \frac{-b_2 - b_3}{m_2} & \frac{b_3}{m_2} \\ 0 & \frac{k_3}{m_3} & \frac{-k_3}{m_3} & 0 & \frac{b_3}{m_3} & \frac{-b_3}{m_3} \end{bmatrix} x(t) \quad (7.1a)$$

$$+ \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{m_1} \\ 0 \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{-a_1 x_1(t)^3}{m_1} \\ 0 \\ 0 \end{bmatrix},$$

$$y(t) = [0 \ 1 \ 0 \ 0 \ 0 \ 0] x(t) + e(t), \quad (7.1b)$$

where $x = [x_1 \ x_2 \ x_3 \ \dot{x}_1 \ \dot{x}_2 \ \dot{x}_3]^\top \in \mathbb{R}^6$ is the general state, $x_i, \dot{x}_i \in \mathbb{R}$ note the position and velocity of each mass, respectively. The measured output is noted with

y , while the physical parameters are m_i masses, k_i spring constants, and b_i damping coefficients. Furthermore, a nonlinear cubic spring with coefficient a_1 is attached to m_1 . The parameter values are summarized in Table 7.1. The output is perturbed with an additive white noise $e(t) \sim \mathcal{N}(0, \sigma_e^2)$, where σ_e has been determined to achieve an SNR of 60 dB.

Index (i)	Mass (m_i)	Spring (k_i)	Damping (b_i)	Nonlin. spring (a_i)
1	0.5 kg	100 N/m	0.5 Ns/m	50 N/m ³
2	0.4 kg	100 N/m	0.5 Ns/m	-
3	0.02 kg	100 N/m	0.5 Ns/m	-

Table 7.1: Physical parameters of the MSD system.

For data generation, the system (7.1) is excited with input signal $u(t)$ generated by a multisine signal with 1666 frequency components in the range of $[0, 25]$ Hz and uniformly distributed phase in $[0, 2\pi)$. The CT system dynamics are simulated using the RK4 numerical scheme, and zero-order-hold input signal with a sampling time of $T_s = 0.02$ s. The sampled output and the input signals are collected into $\mathcal{D}_N$ data sequence. Separate data sets have been generated for training, validation, and testing with sizes $N_{\text{train}} = 2 \cdot 10^4$, $N_{\text{valid}} = 10^4$, and $N_{\text{test}} = 10^4$, respectively.

We assemble the first-principle model, completely neglecting the nonlinear spring force and the third mass. The CT dynamics of the FP model can be expressed in LTI form as

$$\dot{\hat{x}}(t) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k_1 + k_2}{m_1} & \frac{k_2}{m_1} & -\frac{b_1 + b_2}{m_1} & \frac{b_2}{m_1} \\ \frac{k_2}{m_2} & -\frac{k_2}{m_2} & \frac{b_2}{m_2} & -\frac{b_2}{m_2} \end{bmatrix} \hat{x}(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u(t), \quad (7.2)$$

$$\hat{y}(kT_s) = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \hat{x}(kT_s), \quad (7.3)$$

where $\hat{x} \in \mathbb{R}^4$ is the model state, thus $n_x = \dim x_b = 4$ in the model augmentation structure. The CT dynamics have been discretized by the ZOH method using Matlab's `c2d` function.

Section 6 thoroughly discussed the capabilities of each augmentation structure with the applied parameter regularization methods. Hence in this section, we assume that the physical parameters in the FP model are known accurately, furthermore, only the dynamic LFR-based structure is utilized, as the main objective is to test the encoder initialization method. As the FP model neglects the third mass, two states are discarded, thus $n_a = 2$ is applied for the dynamic augmentation structure. The neural network used for model augmentation ($\mathcal{F}_\phi$) has been implemented with 3 hidden layers and 64 nodes per layer, while the encoder neural network with 2 hidden layers and 64 nodes per layer, both with the hyperbolic tangent activation function. Initial evaluations have shown that

the model is prone to overfit on the training data, thus ℓ_2 regularization is applied for ϕ and η parameters in both neural networks with a coefficient of $\lambda = 10^{-6}$. Further hyperparameters are summarized in Table 7.2. All models have been trained for 5000 epochs, using the Adam algorithm with a 10^{-3} learning rate.

T	Batch size	Encoder lag	n_z	n_w
200	1024	7	5	4

Table 7.2: Hyperparameters of the MSD identification problem.

For benchmarking, models have also been trained using a well-known randomized initialization scheme, the Xavier approach [20]. The results are shown in Table 7.3. As visible, the physics-based initialization method has generated approximately the same results as the randomly initialized one. However, as visible in Figure 7.2, with the physics-based method faster convergence has been reached, moreover, there were no deviations in the initial validation errors, which ultimately results in a more consistent estimation procedure of the reconstructability map.

Initialization	RMS test error	NRMS test error
Random init.	0.000805	0.47%
Physics-based init.	0.000825	0.49%

Table 7.3: Test errors with different encoder initialization methods.

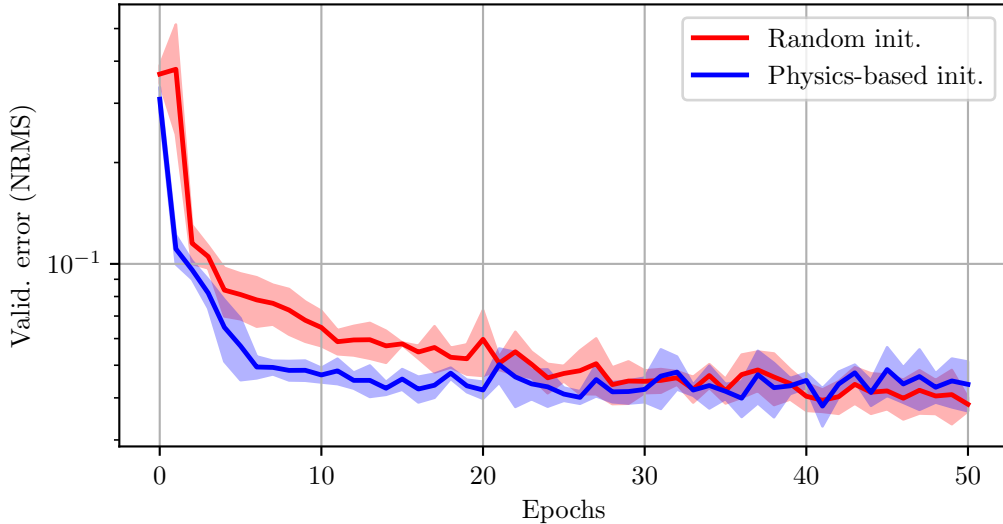


Figure 7.2: First 50 training epochs with the physics-based and random encoder initialization methods. The results are an average of 4-4 Monte-Carlo runs, and the $\pm$ standard deviation is illustrated with shaded areas.

8 Summary and conclusions

8.1 Results

The paper considered different model augmentation structures for efficient system identification. After discussing existing methods, such as additive and multiplicative combinations, a novel LFR-based approach has been also proposed. For more robust identification performance, augmented states have been introduced within the framework of dynamic model augmentation.

Besides tuning the neural network parameters, our approach jointly estimates the learning component and the baseline model parameters. This formulation highly enhanced the efficiency of the proposed model augmentation approach, and its effects are thoroughly discussed throughout the paper. Different regularization techniques have been considered to tackle the problems caused by the over-parametrization of the system. A general parameter regularization approach has been introduced based on [17], which can be applied to all model augmentation structures. A different regularization method has also been considered based on [18], which promotes orthogonality between the FP model dynamics and the neural network part. Originally the orthogonal projection-based approach could be only applied for baseline models that are linear in the parameters, but we have successfully extended the methodology to the NL case. The latter approach can be only utilized for the static additive structure, developing similar methods for the other augmentation structures can be a future research topic.

To enhance the applicability of the model augmentation framework, a novel encoder initialization method has also been derived for systems where full-state measurement is not possible. The efficiency of the approach has been demonstrated on simple simulation data in Section 4.4 and also on a more complex identification problem in Section 7. Based on those results, the physics-based encoder initialization proved to be efficient in terms of convergence speed, while generating the same results as a randomly initialized network. All in all, the proposed method proved to be beneficial for systems where not all states are measured directly. However, the effect of process noise has not been investigated, that is the topic of future research.

After the theoretical discussion of the methods, the capabilities of the proposed augmentation approach have been demonstrated by identifying the dynamics of a small-scale electric car (F1Tenth). Instead of the real car, a digital twin model has been used for data generation, which aligns with the recent trends in the vehicle industry. The final results are summarized in Table 8.1, and test errors of identified black-box models under similar conditions are also shown for comparison. As one might see, all structures provided exceptional accuracies, especially if considering they were simulated on multiple 16-18 seconds long input signals in open-loop format. With given regularization methods, all structures have produced similar test accuracies as the black-box models, however, our proposed dynamic LFR-based augmentation structure outperformed all other methods, even the

CT SUBNET (which belongs to the state-of-the-art black-box identification approaches).

To clearly show the high model accuracies, the static and dynamic augmentation-based models are evaluated on a lemniscate-shaped trajectory from the test set, as shown in Figure 8.1. For better comparison, the position values are calculated by numerically integrating the velocity values, hence the measured and simulated positions can be compared, while the model errors are also shown. Furthermore, it can be seen that the numerical integration errors also affect the calculation of position values, however, still extremely high model accuracy has been achieved. Although some large peaks are visible on the heading angle error sometimes, this is mainly because φ is wrapped to $[-\pi, \pi)$ and when transitions from $-\pi$ to π occur, temporary inaccuracies are present.

Model	Test error (NRMS)		
	Baseline	Tuned phys. parms.	Regularization
Static additive	5.57%	4.74%	4.63% (2.99%)*
Static multiplicative	3.04%	2.81%	2.5%
Static LFR-based	2.84%	2.32%	2.17%
Dynamic additive	2.49%	2.21%	2.13%
Dynamic multiplicative	2.35%	2.28%	2.21%
Dynamic LFR-based	1.16%	0.97%	0.93%
First-principle model	37.91%	-	-
DT SUBNET	2.12%	-	-
CT SUBNET	1.75%	-	-

Table 8.1: Final results with different model augmentation structures and regularization techniques. The table also contains results with benchmark models for comparison. *With the combination of physical regularization and orthogonalization-based regularization terms.

8.2 Conclusions, outlook

In this work, we have been able to derive a novel model augmentation approach based on flexible LFR-based representation and utilizing deep-learning models. The proposed method goes beyond the current state-of-the-art by considering flexible structures that can efficiently estimate ANN and physical parameters simultaneously, introducing augmented states via the subspace encoder method to capture dynamic unmodeled terms, and utilizing prior physical knowledge for initializing said encoder network. Furthermore, different physics-guided regularization methods are adapted from data-driven feedforward control applications, and extended for the model augmentation approach.

The introduced framework can be utilized in various application fields, all of which can be the topic of possible future research. For example, the acquired models can be used

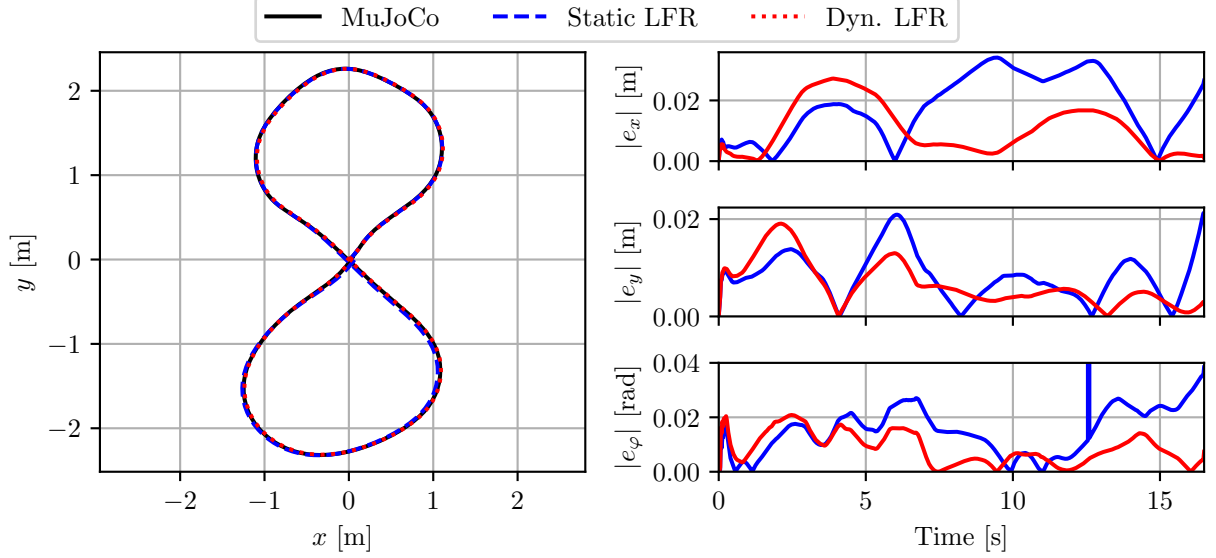


Figure 8.1: Comparing the static and dynamic LFR-based results on an arbitrary test trajectory. The position values in the (x, y) plane are shown in the left panel, while the position and orientation absolute errors are shown on the right panels.

for developing model-based control algorithms, trajectory-planning of autonomous vehicles, or constructing digital twin models. The above-mentioned applications can be all implemented using offline learning of the dynamics. This practically follows the methodology described in this work: first derive an approximate first-principle model of the data-generating system, then gather data through the planned experiments and train the model using the acquired data sets. Then finally, the trained model can be used for the previously mentioned applications.

A different approach is via online learning of the dynamics. An approximate model is required for this approach for the beginning, and then the model is tuned during operation. This approach greatly suits the model augmentation setup, as there is an approximated first-principle model at the beginning, and the machine learning part could be trained online using operational data. However, current optimization algorithms (e.g. SGD) are not time efficient, thus either a very high-performance onboard computing unit is necessary or model training could take undesirably much time. Recent research has proposed more efficient algorithms for training neural networks directly for system identification purposes [38, 39, 40]. These algorithms have showcased an extreme decrease in training time, and thus have huge potential in online learning of system dynamics. Plans include carefully combining the proposed model augmentation framework with the mentioned optimization algorithms, and creating a method capable of online learning of the dynamical relations of the system.

Acknowledgement

I would like to thank *Roland Tóth* and *Tamás Péni* at SZTAKI for their huge amount of help and professional support. I would also like to thank *Csaba Budai* at BME MOGI for his supervision.

The research was supported by the European Union within the framework of the National Laboratory for Autonomous Systems (RRF-2.3.1-21-2022-0002) and was also supported by the EKÖP-24-2-BME-123 University Research Scholarship Programme of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund.



PROGRAM
FINANCED FROM
THE NRDI FUND



Budapest, 2024

Bendegúz Máté Györök

Reference

- [1] G. I. Beintema, M. Schoukens, and R. Tóth, “Deep subspace encoders for nonlinear system identification,” *Automatica*, vol. 156, p. 111210, 2023.
- [2] M. Szécsi, B. Györök, A. Weinhardt-Kovács, G. I. Beintema, M. Schoukens, T. Péni, and R. Tóth, “Deep learning of vehicle dynamics,” *IFAC-PapersOnLine*, vol. 58, no. 15, pp. 283–288, 2024.
- [3] T. Bohlin, *Practical Grey-box Process Identification*, 1st ed., ser. Advances in Industrial Control. Springer London, 2006.
- [4] Z. Yang, D. Eddy, S. Krishnamurty, I. Grosse, P. Denno, Y. Lu, and P. Witherell, “Investigating Grey-Box Modeling for Predictive Analytics in Smart Manufacturing,” in *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2017, p. V02BT03A024.
- [5] R. Aghmasheh, V. Rashtchi, and E. Rahimpour, “Gray Box Modeling of Power Transformer Windings for Transient Studies,” *IEEE Transactions on Power Delivery*, vol. 32, no. 5, pp. 2350–2359, 2017.
- [6] N. Takeishi and A. Kalousis, “Deep Grey-Box Modeling With Adaptive Data-Driven Models Toward Trustworthy Estimation of Theory-Driven Models,” in *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, 2023, pp. 4089–4100.
- [7] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.
- [8] A. Daw, A. Karpatne, W. D. Watkins, J. S. Read, and V. Kumar, “Physics-Guided Neural Networks (PGNN): An Application in Lake Temperature Modeling,” in *Knowledge Guided Machine Learning*. Chapman and Hall/CRC, 2022.
- [9] A. Retzler, R. Tóth, M. Schoukens, G. I. Beintema, J. Weigand, J.-P. Noël, Z. Kollár, and J. Swevers, “Learning-based augmentation of physics-based models: an industrial robot use case,” *Data-Centric Engineering*, vol. 5, p. e12, 2024.
- [10] M. L. Thompson and M. A. Kramer, “Modeling chemical processes using prior knowledge and neural networks,” *AIChE Journal*, vol. 40, no. 8, pp. 1328–1340, 1994.
- [11] B. Sohlberg and E. W. Jacobsen, “Grey Box Modelling – Branches and Experiences,” *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 11 415–11 420, 2008.
- [12] B. Sun, C. Yang, Y. Wang, W. Gui, I. Craig, and L. Olivier, “A comprehensive hybrid first principles/machine learning modeling framework for complex industrial processes,” *Journal of Process Control*, vol. 86, pp. 30–43, 2020.

- [13] R.-S. Götte and J. Timmermann, “Composed Physics- and Data-driven System Identification for Non-autonomous Systems in Control Engineering,” in *Proceedings of the 3rd International Conference on Artificial Intelligence, Robotics and Control (AIRC)*, 2022, pp. 67–76.
- [14] J. H. Hoekstra, C. Verhoek, R. Tóth, and M. Schoukens, “Learning-based model augmentation with LFRs,” *arXiv preprint arXiv:2404.01901*, 2024.
- [15] K. Zhou, J. C. Doyle, and K. Glover, *Robust and Optimal Control*. Prentice Hall, 1996.
- [16] M. Schoukens and R. Tóth, “Linear Parameter Varying Representation of a class of MIMO Nonlinear Systems,” *IFAC-PapersOnLine*, vol. 51, no. 26, pp. 94–99, 2018.
- [17] M. Bolderman, M. Lazar, and H. Butler, “On feedforward control using physics-guided neural networks: Training cost regularization and optimized initialization,” in *Proceedings of the 2022 European Control Conference (ECC)*, 2022, pp. 1403–1408.
- [18] J. Kon, D. Bruijnen, J. Van De Wijdeven, M. Heertjes, and T. Oomen, “Physics-Guided Neural Networks for Feedforward Control: An Orthogonal Projection-Based Approach,” in *Proceedings of the American Control Conference (ACC)*, 2022, pp. 4377–4382.
- [19] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, “Neural Ordinary Differential Equations,” in *Advances in Neural Information Processing Systems*, vol. 31, 2018.
- [20] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.
- [21] R. Ramkannan, G. I. Beintema, R. Tóth, and M. Schoukens, “Initialization Approach for Nonlinear State-Space Identification via the Subspace Encoder Approach,” *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 5146–5151, 2023.
- [22] M. Schoukens and R. Tóth, “On the Initialization of Nonlinear LFR Model Identification with the Best Linear Approximation,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 310–315, 2020.
- [23] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *Proceedings of the 3rd International Conference on Learning Representations*, 2015, pp. 1–15.
- [24] A. H. Ribeiro, K. Tiels, J. Umenberger, T. B. Schön, and L. A. Aguirre, “On the smoothness of nonlinear system identification,” *Automatica*, vol. 121, p. 109158, 2020.

- [25] M. Bolderman, H. Butler, S. Koekebakker, E. Van Horssen, R. Kamidi, T. Spaan-Burke, N. Strijbosch, and M. Lazar, “Physics-guided neural networks for feedforward control with input-to-state-stability guarantees,” *Control Engineering Practice*, vol. 145, p. 105851, 2024.
- [26] M. Revay and I. Manchester, “Contracting Implicit Recurrent Neural Networks: Stable Models with Improved Trainability,” in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, 2020, pp. 393–403.
- [27] M. Revay, R. Wang, and I. R. Manchester, “Recurrent Equilibrium Networks: Flexible Dynamic Models With Guaranteed Stability and Robustness,” *IEEE Transactions on Automatic Control*, vol. 69, no. 5, pp. 2855–2870, 2024.
- [28] G. D. C. Cardenas and M. Schoukens, “Initialization of the SUBNET Neural Network Approach for State-Space Identification,” Master’s thesis, 2023.
- [29] K. Floch, “Model-based motion control of the F1TENTH autonomous electrical vehicle,” Bachelor’s thesis, Budapest University of Technology and Economics,, 2022. [Online]. Available: <https://eprints.sztaki.hu/10544/>
- [30] A. Agnihotri, M. O’Kelly, R. Mangharam, and H. Abbas, “Teaching Autonomous Systems at 1/10th-scale: Design of the F1/10 Racecar, Simulators and Curriculum,” in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 2020, pp. 657–663.
- [31] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, “A survey of motion planning and control techniques for self-driving urban vehicles,” *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.
- [32] K. Floch, T. Péni, and R. Tóth, “Gaussian-Process-Based Adaptive Trajectory Tracking Control for Autonomous Ground Vehicles,” in *Proceedings of the 2024 European Control Conference (ECC)*, 2024, pp. 464–471.
- [33] M. Althoff, M. Koschi, and S. Manzingier, “CommonRoad: Composable benchmarks for motion planning on roads,” in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 719–726.
- [34] J. Svendenius, “Tire Modeling and Friction Estimation,” Doctoral Thesis (monograph), Department of Automatic Control, Lund Institute of Technology, Lund University, 2007.
- [35] E. Bakker, L. Nyborg, and H. B. Pacejka, “Tyre Modelling for Use in Vehicle Dynamics Studies,” SAE Technical Paper 870421, 1987.
- [36] E. Todorov, T. Erez, and Y. Tassa, “MuJoCo: A physics engine for model-based control,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

- [37] G. I. Beintema, M. Schoukens, and R. Tóth, “Continuous-time identification of dynamic state-space models by deep subspace encoding,” in *Proceedings of the International Conference on Learning Representations*, 2023, pp. 1–15.
- [38] A. Bemporad, “An L-BFGS-B approach for linear and nonlinear system identification under ℓ_1 - and group-Lasso regularization,” *arXiv preprint arXiv:2403.03827*, 2024.
- [39] A. D. Adeoye, P. C. Petersen, and A. Bemporad, “Regularized Gauss-Newton for Optimizing Overparameterized Neural Networks,” *arXiv preprint arXiv:2404.14875*, 2024.
- [40] M. Korbit, A. D. Adeoye, A. Bemporad, and M. Zanon, “Exact Gauss-Newton Optimization for Training Deep Neural Networks,” *arXiv preprint arXiv:2405.14402*, 2024.