Simultaneous Decoupling of Inertia and Input via Coordinate Change: Investigation of the 2-DOF Case and Learning Based Solution

Thesis: Master Robotics

Kian Heus - 4876768

July 14, 2025

Main supervisor: Dr. C. (Cosimo) Della Santina
Daily supervisor: Ir. J. (Jingyue) Liu

TU Delft

Simultaneous Decoupling of Inertia and Input via Coordinate Change: Investigation of the 2-DOF Case and Learning Based Solution

Kian T.J. Heus
Department of Cognitive Robotics (CoR-ME)
Delft University of Technology
Delft, The Netherlands

***Abstract*—Fully actuated robots may be controlled using well-understood techniques, such as computed torque control, which override natural system dynamics. For underactuated robots these dynamics cannot be fully cancelled out, and must instead be leveraged, complicating the control problem. We present a classification of underactuated robotic systems based on the degree to which their dynamics can be decoupled. Finding coordinates that decouple the system dynamics simplifies control for two classes of robots, which we identify as partially- and fully decouplable robotic systems. In these decoupling coordinates, the Euler-Lagrange system representation has a block-diagonal inertia matrix and decoupled input matrix. After delving into the fundamentals of this proposed classification, this work implements an autoencoder as a first ML-based framework to learn these decoupling coordinates for the 2 degrees of freedom (DOF) case. Furthermore, we demonstrate how such representations simplify control. Input decoupling allows for collocated control using a straightforward PD + gravity compensation controller. Inertial decoupling enables non-collocated control through feedback linearization within a small set of states. To demonstrate the theory, decoupling coordinates are learned for a 2-DOF toy system. Performance of the learned coordinate transform is analysed, and controllers on learned and analytic decoupling coordinates are compared.**

***Index Terms*—underactuated robots; classification; control; machine learning coordinates; decoupling**

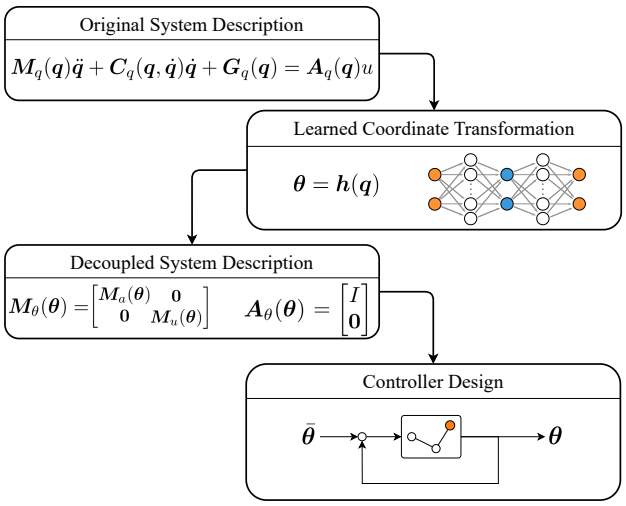

Fig. 1: An intelligent choice of coordinates simplifies control of underactuated robots. An autoencoder framework is presented which learns a coordinate transformation, leading to an input decoupled and inertially decoupled system. This facilitates design of collocated and non-collocated controllers.

I. Introduction

Fully actuated robots are mechanical systems with as many independent actuators as degrees of freedom. This allows for the use of standard control techniques (e.g. computed torque control) which effectively cancel the natural system dynamics [1]. Fully actuated robots have been the industry standard for decades. Their stiffness and strength mean they are reliable, and comparatively easy to model and control [2]–[4]. Common examples are articulated robots for tasks like pick-and-place, assembly and welding [5], [6].

In contrast, underactuated robots possess fewer independent actuators than degrees of freedom. Underactuation may be the result of intentional inclusion of soft or flexible components, or an unactuated base [7]. It may also be caused by actuator failure, lightweight components or the omission of actuators to save cost. There are many advantages of underactuated designs. Soft components, for example, are beneficial when handling delicate objects [8], and for safety in human-robot interaction [9]. Flexibility may improve robustness to unmodelled environmental effects [10], and greatly improve energy efficiency in repetitive motion [11]. Omission of actuators can also reduce robot cost and mass, which is critical for example in space applications [12], [13].

Underactuation, however, also greatly complicates the control problem. The lack of full control authority means that the natural system dynamics cannot be fully overwritten, and must instead be leveraged for control performance [14]. Arbitrary accelerations cannot be imposed on every degree of freedom, which limits the reachable states of such systems [15], [16]. Moreover, many underactuated systems are non-minimum phase, so that non-collocated input-output inversion introduces unstable zero dynamics [17], [18].

Despite the difficulty of the underactuated control challenge, controllers have been designed for a wide range of underactuated systems. One commonly used strategy for underactuated robot control is to rely on the property of strong inertial coupling for feedback linearization [19]–[22]. However, inertial coupling is configuration dependent, and many robots (such as series elastic actuators) are not strongly inertially coupled [23], [24]. If such coupling does not exist, it is favourable to have no inertial coupling at all.

A cornerstone work in the field of underactuated robotics control is the paper by M. Spong on modelling and control of elastic joint robots [25]. There, Spong presents controllers based on an inertially decoupled and input decoupled system description. Inertial decoupling has since been used for controller design of floating (and flexible) base robots [26]–[28], elastic manipulators [29] and soft robots [30], [31], among others. This decoupled representation is coordinate-dependent, and is often obtained through some coordinate transformation. Finding such a transformation requires expert knowledge, and the methodology is rarely applicable to other types of robots. This means that every control engineer has to effectively "reinvent the wheel" when writing a model-based controller for a new kind of robot.

Machine learning may be able to offer a solution, approximating decoupling coordinates where an analytic solution cannot be found. Autoencoders are commonly used for dimensionality reduction, but are also suited for this problem, since they are able to approximate a deterministic change of coordinates and its inverse [32]–[34].

Our work presents a first step towards systematically defining and then discovering these decoupling coordinates of underactuated robots, which simplifies and unifies controller design. Concretely, the contributions of this thesis are as follows:

- A novel classification of underactuated robots based on the extent to which their inertial- and input matrices may be decoupled. The three identified classes are "generic inertially coupled", "partially decouplable" and "fully decouplable".
- A new 2-DOF "toy" system which serves as an example of the partially decouplable systems class.
- A machine learning framework which can learn decoupling coordinates for decouplable underactuated mechanical systems.
- Design and performance analysis of collocated and non-collocated controllers on the toy system.

Section II describes the kinds of systems considered in this thesis. Section III explains how inertial and input decoupling affect controller design, and presents a classification based on decouplability. A machine learning approach to find such decoupling coordinates is presented in Section IV. Controllers based on these coordinates are explained in Section V, and simulation results are presented in Section VI. Section VII provides a discussion of the methodology, results and potential future work, and Section VIII concludes this report.

II. Preliminaries

The scope of this thesis is limited to underactuated mechanical systems whose trajectories follow the Euler-Lagrange equations of motion. For a system with generalized coordinates $\boldsymbol{q} \in \mathbb{R}^n$ and Lagrangian $L(\boldsymbol{q}, \dot{\boldsymbol{q}})$, the multibody dynamics may be written as:

$$\frac{d}{dt}\Big(\frac{\partial L(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial \dot{\boldsymbol{q}}}\Big) - \frac{\partial L(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial \boldsymbol{q}} = 0, \tag{1}$$

where ∂ indicates a partial derivative and $\frac{d}{dt}$ indicates the time derivative. The symbol $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ is a shorthand for the time derivative of $\boldsymbol{q}$. Besides the internal dynamics, we consider three more contributions to the dynamics. These are the damping forces, spring forces and actuator input forces. Damping is modelled by a Rayleigh dissipation function as $\boldsymbol{D}_q(\boldsymbol{q})\dot{\boldsymbol{q}} = \frac{\partial R(\dot{\boldsymbol{q}})}{\partial \dot{\boldsymbol{q}}} \in \mathbb{R}^n$. Elastic forces are captured in the vector $\boldsymbol{K}_q(\boldsymbol{q}) \in \mathbb{R}^n$, and actuation is modelled by $\boldsymbol{A}_q(\boldsymbol{q}) \in \mathbb{R}^{n\times m}$. Including these terms, the dynamics can be written in matrix form as:

$$\boldsymbol{M}_q(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}_q(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}_q(\boldsymbol{q}) + \boldsymbol{D}_q(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{K}_q(\boldsymbol{q}) = \boldsymbol{A}_q(\boldsymbol{q})\boldsymbol{u}. \tag{2}$$

Here, the internal dynamics contributions are split into the inertia matrix $\boldsymbol{M}_q(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$, Coriolis- & centrifugal matrix $\boldsymbol{C}_q(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{n\times n}$ and gravity force vector $\boldsymbol{G}_q(\boldsymbol{q}) \in \mathbb{R}^n$. Control inputs $\boldsymbol{u} \in \mathbb{R}^m$ are mapped to generalized forces by the input matrix $\boldsymbol{A}_q(\boldsymbol{q})$.

III. Proposed Classification

A. Inertial and input decoupling

A new set of coordinates $\boldsymbol{\theta} = \boldsymbol{h}(\boldsymbol{q})$ may be defined. With an appropriate choice of transformation function $\boldsymbol{h}(\boldsymbol{q}) : \mathbb{R}^n \to \mathbb{R}^n$, this results in properties of the dynamical matrices that can be exploited for controller design. The two properties that are central to this thesis are input decoupling and inertial decoupling. For an input decoupled system, the control input $\boldsymbol{u}$ must only affect m degrees of freedom $\boldsymbol{\theta}_a \in \mathbb{R}^m$, leaving the other $l = n - m$ degrees of freedom $\boldsymbol{\theta}_u \in \mathbb{R}^l$ unactuated [35]. In matrix form, such an input matrix can be written as:

$$\boldsymbol{A}_\theta = \begin{bmatrix} \boldsymbol{I}_m \\ \boldsymbol{0}_{l\times m} \end{bmatrix}. \tag{3}$$

The second important property is inertial decoupling. This requires that there are no inertial couplings between actuated (or active, collocated) and unactuated (or passive, non-collocated) degrees of freedom. In matrix form, the requirement is for the inertia matrix to be block-diagonal:

$$\boldsymbol{M}_\theta(\boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{M}_a(\boldsymbol{\theta}) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{M}_u(\boldsymbol{\theta}) \end{bmatrix}. \tag{4}$$

In the above formulation, the active and passive inertia blocks may still be a function of the full state $\boldsymbol{\theta}$, and this will therefore be referred to as partial inertial decoupling. This cross-dependency results in coupling terms in the Coriolis- & centrifugal matrix. A stricter requirement may be imposed,

namely full inertial decoupling. In this case, each block in the inertia matrix must only be a function of its corresponding degrees of freedom, i.e:

$$\boldsymbol{M}_\theta(\boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{M}_{a,a}(\boldsymbol{\theta}_a) & \mathbf{0} \\ \mathbf{0} & \boldsymbol{M}_{u,u}(\boldsymbol{\theta}_u) \end{bmatrix}. \tag{5}$$

Such an inertia matrix does not result in cross-couplings from the Coriolis- & centrifugal matrix.

B. Decoupled dynamics for 2-DOF systems

Analysis is simplified when the 2-DOF case is considered. The decoupled input matrix straightforwardly becomes:

$$\boldsymbol{A}_\theta = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{6}$$

Partial inertial decoupling can be formulated as:

$$\boldsymbol{M}_\theta(\boldsymbol{\theta}) = \begin{bmatrix} M_a(\boldsymbol{\theta}) & 0 \\ 0 & M_u(\boldsymbol{\theta}) \end{bmatrix}, \tag{7}$$

with the corresponding Coriolis- & centrifugal matrix:

$$\boldsymbol{C}_\theta(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \begin{bmatrix} C_a(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) & C_{au}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \\ C_{ua}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) & C_u(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \end{bmatrix}, \tag{8}$$

where individual terms may be computed using standard methods (e.g. Christoffel symbol derivation). When damping is neglected, the equations of motion may be written as follows:

$$M_a(\boldsymbol{\theta})\ddot{\theta}_a + C_a(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\theta}_a + C_{au}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\theta}_u + G_a(\boldsymbol{\theta}) + K_a(\boldsymbol{\theta}) = u, \tag{9a}$$

$$M_u(\boldsymbol{\theta})\ddot{\theta}_u + C_{ua}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\theta}_a + C_u(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\theta}_u + G_u(\boldsymbol{\theta}) + K_u(\boldsymbol{\theta}) = 0. \tag{9b}$$

The absence of control input u in Equation (9b) clearly shows that arbitrary accelerations of the unactuated DOF cannot be imposed through control inputs. Moreover, it can be seen that although the two degrees of freedom do not affect each other at the acceleration level, there are coupling effects through velocity. The ignored damping forces may introduce similar velocity cross-couplings. If instead, the 2-DOF system is fully inertially decoupled, the inertia matrix is as follows:

$$\boldsymbol{M}_\theta(\boldsymbol{\theta}) = \begin{bmatrix} M_{a,a}(\theta_a) & 0 \\ 0 & M_{u,u}(\theta_u) \end{bmatrix}. \tag{10}$$

The Coriolis- & centrifugal matrix simplifies to:

$$\boldsymbol{C}_q(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \begin{bmatrix} C_{a,a}(\theta_a, \dot{\theta}_a) & 0 \\ 0 & C_{u,u}(\theta_u, \dot{\theta}_u) \end{bmatrix}, \tag{11}$$

resulting in the following equations of motion:

$$M_a(\boldsymbol{\theta})\ddot{\theta}_a + C_{a,a}(\theta_a, \dot{\theta}_a)\dot{\theta}_a + G_a(\boldsymbol{\theta}) + K_a(\boldsymbol{\theta}) = u, \tag{12a}$$

$$M_u(\boldsymbol{\theta})\ddot{\theta}_u + C_{u,u}(\theta_u, \dot{\theta}_u)\dot{\theta}_u + G_u(\boldsymbol{\theta}) + K_u(\boldsymbol{\theta}) = 0. \tag{12b}$$

In this case, neither accelerations nor velocities in one DOF affect the other. Since we can again not affect $\ddot{\theta}_u$ through the control input u, we must rely on potential couplings from gravity or spring forces to affect the unactuated degree of freedom. These properties may be leveraged for controller design.

C. Proposed classification

Current classifications of underactuated robots often focus on the cause of underactuation, but neglect the effect of underactuation on system dynamics [37]–[39]. This makes it difficult to generalise control methods between systems. We propose three classes of systems, which are separated based on "decouplability", i.e. the degree to which their inertia- and input matrix can be decoupled. The classes are "Generic underactuated", "Partially decouplable" and "Fully decouplable" (see Figure 2).

1) Generic underactuated systems: Generic robotics systems which possess fewer actuators than degrees of freedom are called generic underactuated systems. Depending on the type of actuation, input decoupling coordinates may be obtained. An example system is an adapted version of the Pendubot [36]. Namely, a double pendulum with point masses on the elbow and end-effector, and an actuator at the base (see Figure 2, system 1). A standard choice of coordinates, namely the absolute joint angles, reveals an input decoupled form:

$$\boldsymbol{A}_q = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{13}$$

The mass matrix, however, contains off-diagonal elements:

$$\boldsymbol{M}_q(\boldsymbol{q}) = \begin{bmatrix} l_0^2(m_0 + m_1) & l_0 l_1 m_1 \cos(q_0 - q_1) \\ l_0 l_1 m_1 \cos(q_0 - q_1) & l_1^2 m_1 \end{bmatrix}. \tag{14}$$

As is shown in Appendix A, no choice of coordinates leads to inertial decoupling for this system.

2) Partially decouplable systems: Systems are called partially decouplable when they can be described in input decoupled and partially inertially decoupled form. To provide an example, we introduce a toy system, which is an adaptation of the well-understood double pendulum.

This new system is a double pendulum with no inertial components apart from a point mass at the end-effector. For this reason, we will call it by the shorthand of single-mass double pendulum (SMDP). Its only actuator is a rod-like actuator which is attached from a set location (x_a, y_a) to the end-effector of the robot (see Figure 2, system 2). This rod is able to push and pull on the point mass.

An analytic coordinate transformation has been determined which describes the system in input decoupled and partially inertially decoupled form. The collocated DOF, namely the rod length, has been determined for input decoupling following methodology described in [35]. The resulting input matrix of the system is:

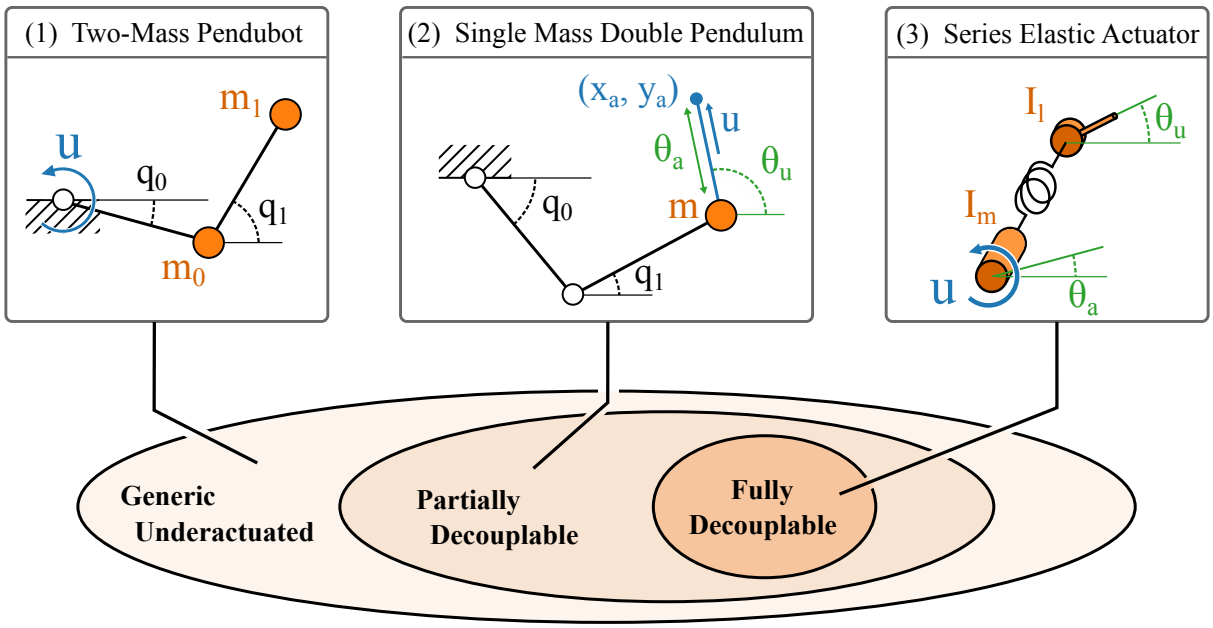

Fig. 2: The proposed classification of underactuated robots based on their degree of decouplability. An example system for each class is provided. System (1) is a two-mass version of the Pendubot [36], which cannot be simultaneously input decoupled and inertially decoupled. System (2) is the newly presented SMDP which may be input- and partially inertially decoupled. System (3) is a series elastic actuator, which may be input- and fully inertially decoupled.

$$\boldsymbol{A}_\theta = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{15}$$

This leaves the non-collocated coordinate free. When the rod angle (i.e. the angle of the vector from end-effector to rod attachment point) is chosen as the second degree of freedom, the inertia matrix is as follows:

$$\boldsymbol{M}_\theta(\theta_a) = \begin{bmatrix} m & 0 \\ 0 & m\theta_a^2 \end{bmatrix}. \tag{16}$$

This is exactly the partial inertial decoupling as described above. A proof that there is no possible choice of coordinates to achieve full decoupling is provided in Appendix B.

Despite its similarity to a fully actuated generic double pendulum, control is significantly more difficult, as will be demonstrated in following sections. This in-between class presents an interesting control challenge. Section V shows that for a controller applied to the SMDP, cross-dependency in the inertia matrix can be neglected, and the system can be treated as fully decouplable.

3) Fully decouplable systems: When a choice of coordinates exists for which the system is both input and fully inertially decoupled, the system is fully decouplable. An example system is the Series Elastic Actuator, which is fully decoupled for the standard choice of coordinates, namely the motor and link angle (see Figure 2, system 3). This results in the input matrix of (15). The mass matrix is as follows:

$$\boldsymbol{M}_\theta = \begin{bmatrix} I_m & 0 \\ 0 & I_1 \end{bmatrix} \tag{17}$$

This clearly shows that the system is input decoupled and fully inertially decoupled.

Controllers based on this decoupling can then be applied. For example, a non-collocated controller based on feedback linearization of the normal form representation may be used. Such a controller is described in Section V. Alternatively, existing controllers such as those discussed in Section I may be applied.

To the authors' best knowledge, a classification of underactuated robots based on the degree to which their dynamics may be decoupled has not been described in the existing literature. We believe that this provides a step towards unifying the control problem of fully- and partially decouplable robots.

IV. Machine Learning Decoupling Coordinates

Certain systems cannot be decoupled fully, or finding such coordinates is prohibitively difficult. In this case, machine learning solutions may be able to find decoupling coordinates.

To this end, an autoencoder framework is proposed for the task of learning decoupling coordinates (see Figure 3). Autoencoders naturally learn both a forward- and inverse mapping, and physically meaningful transformations can be obtained through an appropriate selection of loss terms. This makes them particularly suitable for learning of decoupling coordinates $\boldsymbol{\theta}_L$.

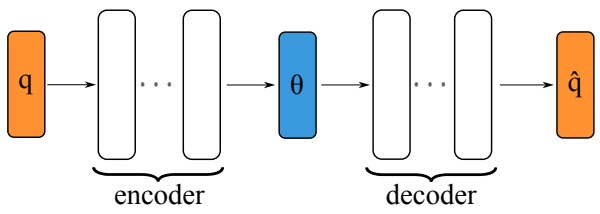

Fig. 3: The autoencoder framework, consisting of an encoder and decoder. Decoupling properties in the latent space are obtained through a combination of loss terms.

A. Loss Function

To ensure that the autoencoder learns a meaningful transformation, loss terms have been added which impose physical constraints on the encoder, latent space and decoder. The loss function is composed of seven terms in three categories. These categories are: reconstruction, input decoupling and inertial decoupling. All loss terms are mean squared error (MSE) across each training batch, to ensure batch size independence. In the following definitions, the loss calculation for a single sample is shown. In the case of matrix results, the loss is then computed using the Frobenius norm.

1) Reconstruction: The first category of loss term ensures that the autoencoder learns a proper coordinate transformation. To this end, a standard reconstruction loss is imposed. Reconstruction loss can be calculated as:

$$L_{\texttt{recon}} = \boldsymbol{q} - \hat{\boldsymbol{q}}, \tag{18}$$

where $\hat{\boldsymbol{q}}$ is the reconstructed state. A second loss term requires that the encoder and decoder Jacobian are each other's inverse. This is important, because a valid coordinate transformation must be invertible, with a well-defined derivative. Any non-singular square matrix multiplied by its inverse results in the identity matrix. We leverage this property to calculate the Jacobian inverse loss:

$$L_{J\texttt{inv}} = \boldsymbol{J}_{\texttt{enc}}(\boldsymbol{q})\boldsymbol{J}_{\texttt{dec}}(\boldsymbol{q}) - \boldsymbol{I}, \tag{19}$$

where $\boldsymbol{J}_{\texttt{enc}}(\boldsymbol{q})$ and $\boldsymbol{J}_{\texttt{dec}}(\boldsymbol{q})$ are the encoder and decoder Jacobian, respectively.

2) Input decoupling: The second category of loss terms concerns input decoupling of the system. This is based on an input decoupling formulation on the actuated coordinate and forward Jacobian [35], and is imposed using three loss terms. The first straightforwardly requires the active coordinates $\boldsymbol{\theta}_a$ to equal their analytic counterparts.

$$L_{\theta_a} = \boldsymbol{\theta}_{a,L} - \boldsymbol{\theta}_{a,A}. \tag{20}$$

Here, $\boldsymbol{\theta}_{a,L}$ and $\boldsymbol{\theta}_{a,A}$ are the learned and analytic actuated coordinates, respectively. To compute the second term derived from [35] it is useful to first partition the forward Jacobian as:

$$\boldsymbol{J}_{\texttt{enc}}(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{J}_{\texttt{a}}(\boldsymbol{q}) \\ \boldsymbol{J}_{\texttt{u}}(\boldsymbol{q}) \end{bmatrix}. \tag{21}$$

Here, $\boldsymbol{J}_{\texttt{a}}(\boldsymbol{q}) \in \mathbb{R}^{m\times n}$ is the Jacobian component related to active DOFs, and $\boldsymbol{J}_{\texttt{u}}(\boldsymbol{q}) \in \mathbb{R}^{l\times n}$ is the component related to passive DOFs. The second input decoupling term then requires $\boldsymbol{J}_{\texttt{a}}(\boldsymbol{q})$ to equal the transpose of the input matrix in $\boldsymbol{q}$-space, $\boldsymbol{A}_q(\boldsymbol{q})$. This can be computed as:

$$L_{J\texttt{act}} = \boldsymbol{J}_{\texttt{a}}(\boldsymbol{q}) - \boldsymbol{A}_q(\boldsymbol{q})^T. \tag{22}$$

The control input u should have no effect on the passive DOFs. To this end, the third term penalises non-zero values on the unactuated coordinates of the input matrix in $\boldsymbol{\theta}$-space, $\boldsymbol{A}_{\theta,u}(\boldsymbol{\theta}_L)$:

$$L_{A_{\theta,u}} = \boldsymbol{A}_{\theta,u}(\boldsymbol{\theta}_L). \tag{23}$$

One could alternatively bypass all these loss terms and use the analytically computed input decoupling coordinate $\boldsymbol{\theta}_{a,A}$ directly, while only learning the unactuated coordinate $\boldsymbol{\theta}_{u,L}$. This would reduce computational complexity and provide an exact solution. During testing, however, it was found that providing the autoencoder with freedom of both coordinates increased the convergence rate and improved final performance significantly.

3) Inertial decoupling: The last category of loss terms enforces inertial decoupling, which requires the off-diagonal entries of the mass matrix to be significantly smaller than the block-diagonal ones. A simple ratio between these two was considered, but this often caused poor learning and numerical instability. Instead, inertial decoupling loss through two terms is proposed. The first term requires that the off-diagonal value of the mass matrix in $\boldsymbol{\theta}$-space, $\boldsymbol{M}_{au}(\boldsymbol{\theta}_L)$, be zero:

$$L_{M,\texttt{off-dia}} = \begin{bmatrix} 0 & \boldsymbol{M}_{au}(\boldsymbol{\theta}_L) \\ \boldsymbol{M}_{au}(\boldsymbol{\theta}_L) & 0 \end{bmatrix}. \tag{24}$$

The second term forces the diagonal elements, $\boldsymbol{M}_a(\boldsymbol{\theta})$ and $\boldsymbol{M}_u(\boldsymbol{\theta})$, to be non-zero. To compute this in the 2-DOF case, first the block-diagonal matrix entries are truncated to some value σ. The value of σ should be chosen to match system masses. The matrix terms are truncated because we do not want to reward large inertias, so long as they are sufficiently greater than zero. The truncation can be computed as:

$$\tilde{M}_a(\boldsymbol{\theta}) = min(M_a(\boldsymbol{\theta}), \sigma), \tag{25}$$

and

$$\tilde{M}_u(\boldsymbol{\theta}) = min(M_u(\boldsymbol{\theta}), \sigma), \tag{26}$$

where $min(a, b)$ indicates the minimum between a and b. $\tilde{M}_a(\boldsymbol{\theta})$ and $\tilde{M}_u(\boldsymbol{\theta})$ are the truncated inertia matrix components. These terms are then compared to the parameter σ:

$$L_{M,\texttt{dia}} = \begin{bmatrix} \tilde{M}_a(\theta_a) & 0 \\ 0 & \tilde{M}_u(\theta_u) \end{bmatrix} - \sigma\boldsymbol{I}. \tag{27}$$

In the higher-dimensional case, the minimum values of the block-diagonal components $\boldsymbol{M}_a(\boldsymbol{\theta})$ and $\boldsymbol{M}_u(\boldsymbol{\theta})$ should be considered carefully. Multiple parameters σ_i may be required to set lower bounds for inertias and inertial couplings.

TABLE I: Loss terms and their weights as used for training the autoencoder

Category	Reconstruction		Input Decoupling			Inertial Decoupling	
Loss term	`recon`	J`inv`	θ_a	J`act`	$A_{\theta,u}$	M, `dia`	M, `off` $-$ `dia`
Weight	4	1	1	0.8	1	1.25	1.25

The loss components and their weights are summarized in Table I. Besides the loss terms, a number of hyperparameters needed to be selected. More information about the hyperparameter tuning process is given in Appendix C.

B. Simulated System

In order to test the performance of the designed machine learning framework, the autoencoder has been used to learn the partial decoupling coordinates of the SMDP. For this system, a set of partial decoupling coordinates are known, which serves as a baseline for analysis of learned coordinates, and related controller performance.

The analytic decoupling coordinates $\boldsymbol{\theta}_A$ are the length of the rod and its angle (see Figure 2, system 2). For every end effector position, there are two valid inverse kinematic solutions to $\boldsymbol{q}$. To ensure that the coordinate transform between joint angles and decoupling coordinates is bijective, the configuration space of the robot must be limited to contain either only "arm-up" or "arm-down" configurations, such as $q_0 \in [-\pi, \pi)$ and $q_1 \in [q_0, q_0 + \pi)$ for the "arm-down" case.

In practice, the configuration space is limited further to improve performance of the autoencoder. Two parameters, $\kappa = 0$ and $\lambda = 0.2$ were chosen to limit the configuration space to $q_0 \in [-\pi, \kappa)$ and $q_1 \in [q_0 + \lambda, q_0 + \pi - \lambda)$.

The remaining parameters used for numerical simulations are indicated in Table II, including link lengths l_i, end-effector mass m and gravitational acceleration g.

TABLE II: Simulation parameters

$l_0\,[m]$	$l_1\,[m]$	$m\,[kg]$	$g\,[m/s^2]$	$x_a\,[m]$	$y_a\,[m]$
2.5	2.5	3	9.81	2	5

V. Controller Design

Describing a robot system in decoupling coordinates allows the use of controllers that depend on this decoupling. To illustrate this, two kinds of controllers have been designed for the SMDP. The first is a collocated controller, which aims to drive the collocated degree of freedom, θ_a, to a desired value. The second controller is a non-collocated controller, which instead has its system output as the unactuated degree of freedom θ_u.

A. Collocated control

For the case of collocated control, the SMDP without springs, and with varying degrees of joint damping was considered. By choosing as system output the actuated degree of freedom, the SMDP can be controlled using a simple proportional-derivative controller with gravity compensation (PD+). This results in the following control law:

$$u = k_p(\bar{\theta}_a - \theta_a) + k_d(\bar{\dot{\theta}}_a - \dot{\theta}_a) + G_{\theta,a}(\boldsymbol{\theta}). \tag{28}$$

Here, k_p and k_d are gains that may be chosen for example through pole placement or LQR optimization. $\bar{\theta}_a$ and $\bar{\dot{\theta}}_a$ indicate the desired position and velocity of the active DOF, and $G_{\theta,a}(\boldsymbol{\theta})$ is the active component of the potential force vector. The obtained control input u is then applied to the real system in $\boldsymbol{q}$-space, where the system dynamics are simulated. The control loop, including coordinate transformations is shown in Figure 4. There, ∇h indicates the computation of $\dot{\boldsymbol{\theta}}$ from $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$. The proportional and derivative control term are indicated with P and D, respectively, and the simulated system is represented by the SMDP icon. Note that this controller relies on the property of input decoupling, but does not leverage the inertially decoupled description.

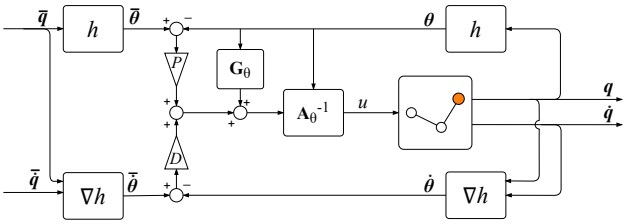

Fig. 4: Block diagram of the PD + gravity compensation controller in $\boldsymbol{\theta}$-space.

In order to analyse controller performance, a baseline controller in the "naive" joint space coordinates was developed for comparison. For this controller in $\boldsymbol{q}$-space, there is no direct mapping between input and output. Because of this, the system cannot be controlled by treating it as an output regulation problem. Instead, a combination of feedforward regulation to the desired state and feedback control of the linearized state may be used in the neighbourhood of the linearization point.

For the SMDP system without springs, the feedforward term only needs to compensate gravity. For more complex systems, numerical solvers may be used to determine the required feedforward control term. The set of desired states is limited to those directly below the actuator, so that the feedforward term may be written as:

$$u_{ff} = -mg. \tag{29}$$

The feedback control term assumes small deviation from the linearization point. This deviation is defined as:

$$\boldsymbol{e}_q = \begin{bmatrix} \boldsymbol{q} - \bar{\boldsymbol{q}} \\ \dot{\boldsymbol{q}} - \bar{\dot{\boldsymbol{q}}} \end{bmatrix}. \tag{30}$$

A feedback control input, u_{fb}, proportional to the error steers the system to the desired location. A set of gains $\boldsymbol{K}_{naive} \in \mathbb{R}^{1\times 4}$ can be chosen based on performance requirements. The complete control law is then:

$$u = u_{ff} + \boldsymbol{K}_{naive}\boldsymbol{e}_q. \tag{31}$$

B. Non-collocated control

The abovementioned collocated controller does not consider the unactuated DOF, which leaves oscillatory zero dynamics. Without damping, a non-collocated controller may be preferred. A feedback linearizing controller on the non-collocated DOF is able to regulate the full state to a steady value, as will be described below.

For non-collocated control of the SMDP, the unactuated DOF is chosen as system output:

$$y = \theta_u. \tag{32}$$

A linear system can be obtained through a coordinate change from the state $\boldsymbol{\theta}$ and velocity $\dot{\boldsymbol{\theta}}$, to the system output y and its derivatives. For the partially decoupled SMDP, this leads to the following formulation:

$$y = \theta_u, \tag{33}$$
$$\dot{y} = \dot{\theta}_u, \tag{34}$$
$$\ddot{y} = \ddot{\theta}_u = \frac{1}{M_{uu}(\boldsymbol{\theta})}\big(-\boldsymbol{C}_u(\boldsymbol{\theta},\dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} - G_u(\boldsymbol{\theta})\big), \tag{35}$$
$$\dddot{y} = f(\boldsymbol{\theta},\dot{\boldsymbol{\theta}},u). \tag{36}$$

Here, $f(\boldsymbol{\theta},\dot{\boldsymbol{\theta}},u)$ is some non-linear function that describes the jerk of the unactuated degree of freedom. The dependency of the jerk on control input u shows that this choice of output leads to a system of relative degree 3. If instead, the system is assumed to be fully inertially decoupled, the lack of inertial cross-couplings simplifies the Coriolis- & centrifugal matrix, resulting in a system of relative degree 4. The neglected terms are a function of $\dot{\theta}_a\dot{\theta}_u$, which may destabilize the system for large velocities. In practice however, this effect is limited, as is shown in Section VI-C.

Using the fully decoupled representation, we can describe the passive DOF and its derivatives as a linear system. The full state $\boldsymbol{Y} \in \mathbb{R}^4$ can be written as a column vector as follows:

$$\boldsymbol{Y} = \begin{bmatrix} y & \dot{y} & \ddot{y} & \dddot{y} \end{bmatrix}^T. \tag{37}$$

The dynamics in this $\boldsymbol{Y}$-space can be represented in companion form as:

$$\begin{bmatrix} \dot{y} \\ \ddot{y} \\ \dddot{y} \\ y^{iv} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \\ \dddot{y} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v, \tag{38}$$

where $v = y^{iv}$ is a virtual control input used to control the system. This is a linear time-invariant (LTI) system, namely a chain of integrators. As a result, standard controllers for LTI systems such as pole placement or LQR may be applied. In order to relate this virtual control v input to the real system input u we need a description of y^{iv}. This term can be written in normal form as:

$$y^{iv} = \alpha(\boldsymbol{Y}) + \beta(\boldsymbol{Y})u, \tag{39}$$

where $\alpha(\boldsymbol{Y})$ indicates the internal dynamics contribution, and $\beta(\boldsymbol{Y})$ is the control effectiveness of control input u. Any virtual control input determined based on the dynamics in $\boldsymbol{Y}$-space, can then be linked to a corresponding input u based on (39).

In order to control the system towards the desired state, the virtual control input is defined as

$$v = \boldsymbol{K}_Y(\bar{\boldsymbol{Y}} - \boldsymbol{Y}), \tag{40}$$

where $\bar{\boldsymbol{Y}}$ is the desired state in linearized coordinates. Corresponding control gains $\boldsymbol{K}_Y \in \mathbb{R}^{1\times 4}$ are determined using Ackermann's formula for pole placement [40]. Figure 5 shows a block diagram of the used control scheme. There, r is a shorthand for the mapping from $\boldsymbol{\theta}$ to the linearized state $\boldsymbol{Y}$, K indicates the calculation of the virtual control input v and s indicates the calculation of the final control input u. This input is applied to the real system, after which the dynamics are simulated, as indicated by the SMDP icon.

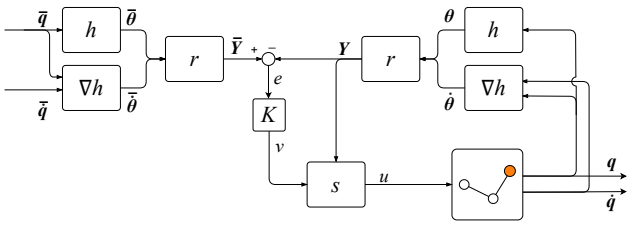

Fig. 5: Block diagram of the non-collocated controller in $\boldsymbol{Y}$-space.

VI. Results

A. Learning of coordinates

The autoencoder has been trained to learn the partial decoupling coordinates of the non-collocated single mass double pendulum (SMDP). The following section analyses the autoencoder performance based on reconstruction, the degree of inertial- and input decoupling, and dynamics prediction of the system.

A comparison between the learned and analytic $\boldsymbol{\theta}$-coordinates, as a function of initial coordinates $\boldsymbol{q}$, is shown in Figure 6. The learned collocated coordinate $\theta_{a,L}$ closely matches its analytical counterpart $\theta_{a,A}$. The second coordinate $\theta_{u,L}$ does not exactly match $\theta_{u,A}$, but the two are related through an affine transformation. This discrepancy is expected, since the loss function does not directly impose any constraints on $\theta_{u,L}$. Instead, requirements on the Jacobian ensure that the system is inertially decoupled in $\boldsymbol{\theta}_L$. An affine scaling of the

unactuated coordinate does not affect this decoupling, which explains the discrepancy.

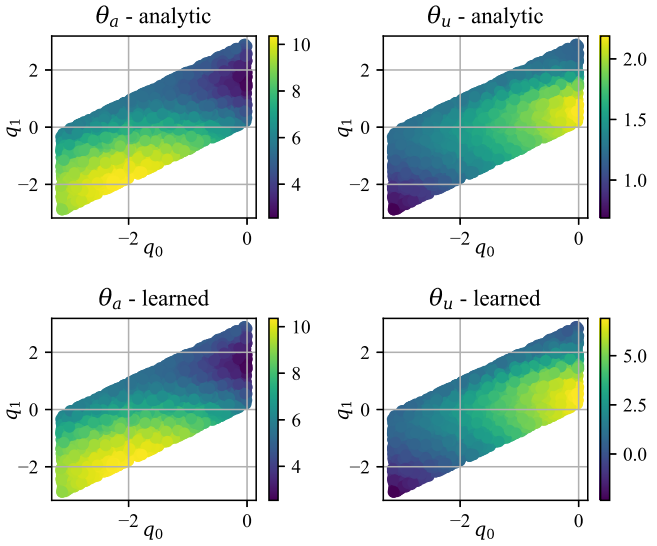

Fig. 6: Comparison between learned and analytic $\boldsymbol{\theta}$. Note that the θ_u-range is different between the analytic and learned transformation.

Jacobian invertibility

Another important measure of learned coordinate performance is whether the encoder and decoder Jacobian are each other's inverse, to ensure that the learned transformation is invertible. Figure 7 shows the Jacobian identity error, calculated equally to the loss term of Equation (19). The Frobenius norm is indicated as $||\dots||_F$. This clearly shows that although there are some points of poor invertibility along the edge of the training set, the encoder and decoder Jacobians generally match well.

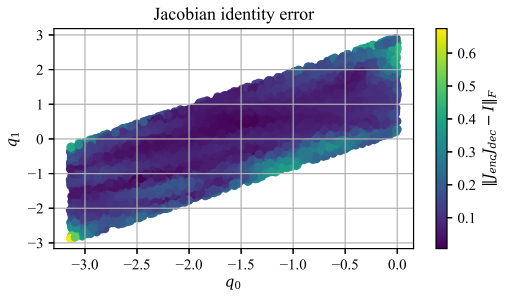

Fig. 7: Error between the encoder and decoder Jacobian product ($J_{enc}J_{dec}$) and identity.

Dynamical matrices

The goal of the autoencoder is to learn a set of coordinates for which the system is both input decoupled and inertially decoupled. This can directly be illustrated through the values of $\boldsymbol{A}_\theta$ and $\boldsymbol{M}_\theta$ across the configuration space of $\boldsymbol{q}$.

Input decoupling was achieved, with the first term of $\boldsymbol{A}_\theta(\boldsymbol{\theta}_L)$ being between 0.8 and 1.2 for 94% of samples. The second term of $\boldsymbol{A}_\theta(\boldsymbol{\theta}_L)$ was between -0.2 and 0.2 for 96% of samples. However, deviations do occur especially for the actuated DOF, mostly along the edge of the domain. The input decoupling error is shown for different $\boldsymbol{q}$ in Figure 8. This shows a similar pattern to that of the Jacobian invertibility.

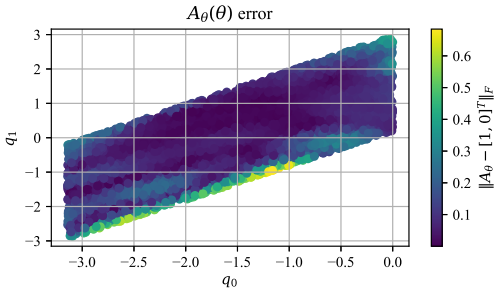

Fig. 8: Input decoupling error of $\boldsymbol{A}_\theta(\boldsymbol{\theta}_L)$ as a function of $\boldsymbol{q}$.

The second goal is inertial decoupling, which requires maximising diagonal elements of $\boldsymbol{M}_\theta(\boldsymbol{\theta}_L)$ while minimising its off-diagonal elements. This was achieved with the diagonal terms of $\boldsymbol{M}_\theta(\boldsymbol{\theta}_L)$ being greater than 1 in 98% of cases, and 95% of the off-diagonal terms being smaller than 0.1. Figure 9 shows the inertial decoupling error. For this computation, the inertia matrix entries were first truncated to be no greater than one, similarly to Equation (25). This truncated inertia matrix, $\tilde{\boldsymbol{M}}_\theta(\boldsymbol{\theta}_L)$, was then compared to the desired identity matrix.

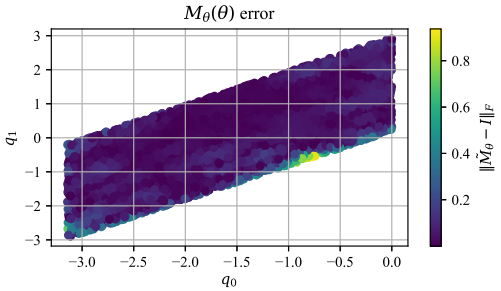

Fig. 9: Inertial decoupling error of $\boldsymbol{M}_\theta(\boldsymbol{\theta}_L)$ as a function of $\boldsymbol{q}$.

Dynamics prediction

Although the previously described metrics indicate good performance of the autoencoder, the model's physical accuracy can be understood better by simulating the system dynamics in the learned coordinates. As a measure of physical accuracy, trajectories with varying starting coordinates and control input were simulated. For each starting condition, a trajectory was simulated both using the real dynamics in $\boldsymbol{q}$-space, as well as using the learned dynamics in $\boldsymbol{\theta}_L$-space. The trajectory in learned coordinates was then transformed to $\hat{\boldsymbol{q}}$-space for comparison with the real trajectory. For more information on trajectory generation, see Appendix F.

The root-mean-square error (RMSE) of the learned joint angles compared to their analytic counterpart is shown in Figure 10. This shows how the approximated trajectories start at a small deviation, which increases with time. However, the error is very small, as after three seconds, the RMSE is less than 0.05 radians, or less than 3 degrees, for both joints. The standard deviation is quite large compared to the RMSE. This shows that the autoencoder performance varies across its trained domain. The coordinate prediction error is larger at the edge of the training set, which explains why simulation error increases as simulations drift from the centre.

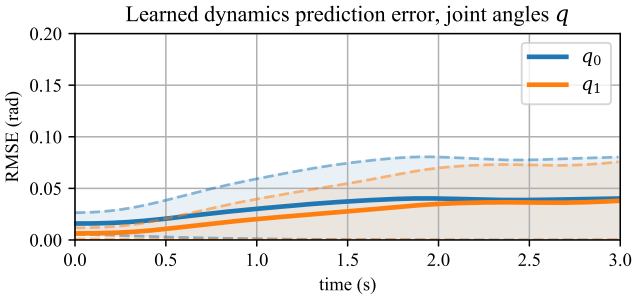

Fig. 10: Root mean square error of learned coordinates versus time. One standard deviation shown in shaded colour. n = 207

The combination of good performance in reconstruction, decoupling and dynamics simulation shows that the autoencoder framework is able to successfully learn a decoupling coordinate transformation.

B. Collocated controller

Three collocated controllers were compared on the SMDP system: the PD+ controllers in analytic coordinates $\boldsymbol{\theta}_A$ and learned coordinates $\boldsymbol{\theta}_L$, and the controller on the linearized state in "naive" coordinates $\boldsymbol{q}$. With properly tuned gains, all three controllers locally stabilise the system.

Controller performance was compared based on four performance metrics: configuration error, control effort, settling time and overshoot (see Appendix E for details). Control gains were optimised for each controller type using a grid search weighing these four metrics. Dynamics were then simulated with varying joint damping. The first set of simulations has no joint damping in $\boldsymbol{q}$-space, the second has equal joint damping of 10 Nms/rad on each joint, and the last similarly has joint damping of 50 Nms/rad.

An example 15 second trajectory of the undamped case is shown in Figure 11. This clearly shows that all controllers converge to oscillatory behaviour. However, the controllers on decoupled coordinates very closely reach the desired tendon length θ_a, whereas the controller on naive coordinates oscillates for all degrees of freedom, both in $\boldsymbol{q}$-space as well as in $\boldsymbol{\theta}$-space.

Since the undamped system does not converge to a steady state, the rod length θ_a is chosen for performance measures. Figure 12 shows controller performance of the undamped case for 46 trajectories. This clearly shows that controller performance is very similar between the learned and analytic decoupling coordinates. Furthermore, the controller on naive coordinates has significantly larger average positional error, often does not settle within bounds and overshoots more. Control effort is similar between controllers, with the $\boldsymbol{q}$-space controller showing slightly more conservative control input.

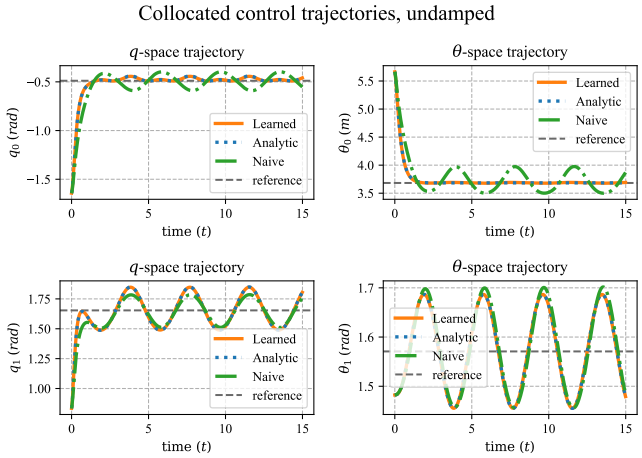

Fig. 11: Example simulation using collocated controllers (orange and blue), as well $\boldsymbol{q}$-space controller (green) on undamped system. The $\boldsymbol{\theta}$-space trajectory shows the true (i.e. analytic) values of the rod length and angle for each simulation.

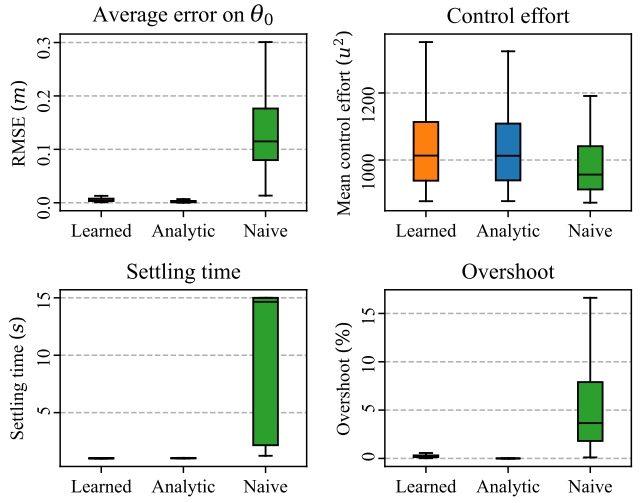

Fig. 12: Controller performance results for system without damping, average of 46 simulations. Performance measures are defined on the tendon length $\theta_{a,A}$.

Similar analysis was performed in the damped case. The trajectories in Figure 13 show how oscillations for each controller type gradually die out with time. This causes performance of the three controllers to become more similar.

Performance metrics on the joint angles paint a different picture from the undamped case, as can be seen in Figure 14. Error on the joint angles is slightly larger for the controller on naive coordinates, but overall very small. Control effort remains similar to the undamped case. Settling time and overshoot increase significantly for the controllers on decoupling coordinates, because their control objective is not defined on the joint angles. However, they are still comparable to the results achieved by the controller on naive coordinates.

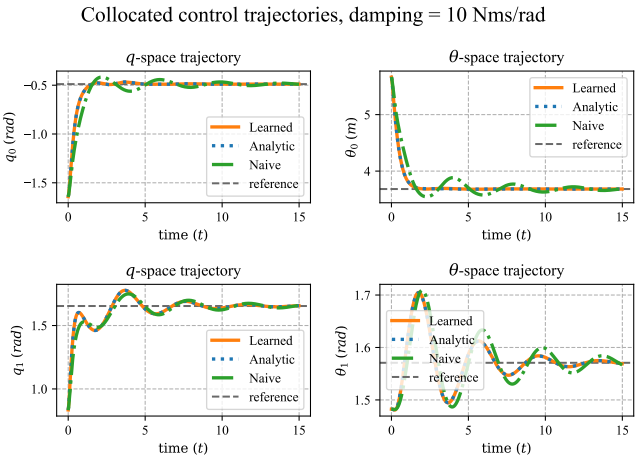

Fig. 13: Example simulation using collocated controllers on lightly damped system. The oscillations of all three trajectories converge over time.

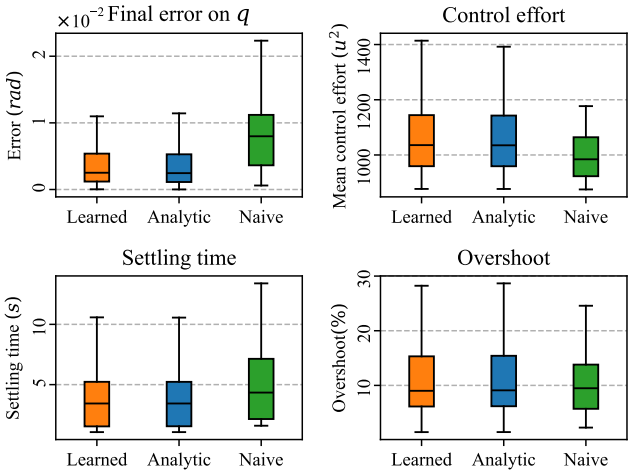

Fig. 14: Controller performance results for system with joint damping of 10 Nms/rad, average of 46 simulations. Performance measures are defined on robot configuration $\boldsymbol{q}$.

As damping is increased, different controller trajectories converge and become nearly indistinguishable. This is shown clearly in the example of Figure 15.

The similarity between trajectories is also visible in the performance metrics presented in Figure 16. Joint error angle decreases to less than 10^{-4} radians, revealing error in the learned coordinate mapping. Control effort, settling time and overshoot do not vary with any significance between controllers.

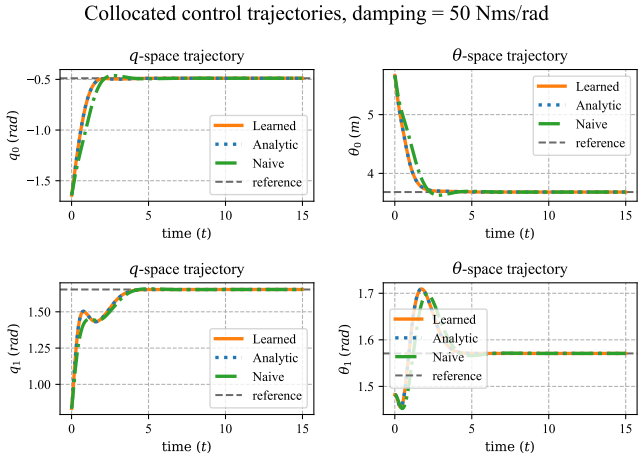

Fig. 15: Example simulation using collocated controllers on strongly damped system. The trajectories converge quickly.

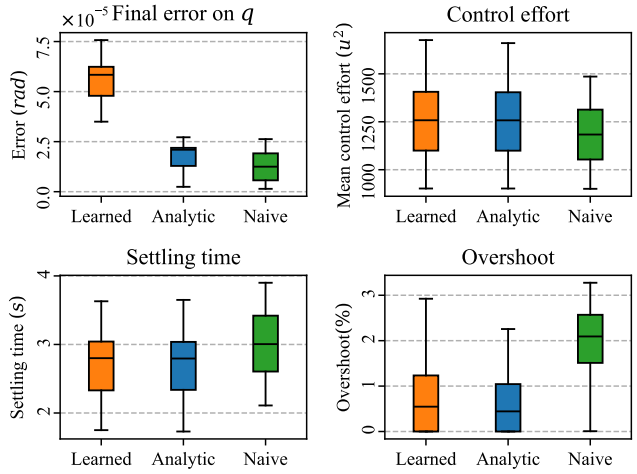

Fig. 16: Controller performance results for system with joint damping of 50 Nms/rad, average of 44 simulations. Performance measures are defined on robot configuration $\boldsymbol{q}$.

C. Non-collocated controller

For non-collocated simulation, no joint damping was applied. Instead, two torsional springs with a constant stiffness of 4 Nm/rad were included in the system. These springs shape the potential function of the system, which allows the use of the previously described non-collocated controller.

The non-collocated controller is able to drive the system to a stable desired state, for a specific set of starting and desired

conditions. An example trajectory is shown in Figure 17, for the controller applied to the analytic coordinates. This shows that although the controller is able to drive the passive DOF to its desired value, there is steady-state error in the active DOF. This error highlights a limitation of the applied methodology, specifically with the linearizing transform $\boldsymbol{Y} = \boldsymbol{n}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$. This mapping is only a true coordinate transformation locally, and for a sufficiently large domain the transform is not one-to-one. As a result, the state in $\boldsymbol{Y}$-space may be equal to the desired state, while the same is not true in $\boldsymbol{\theta}$-space. For application of this controller it is therefore imperative to limit the set of permissible states accordingly.

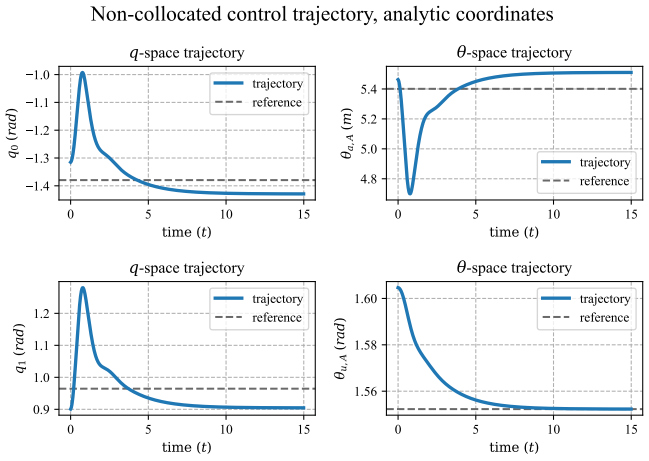

Fig. 17: Example trajectory of non-collocated controller applied to analytic coordinates $\boldsymbol{\theta}_A$.

For the same starting and desired state, the trajectory following from control on learned coordinates is shown in Figure 18. Although the general trend is similar to the analytic trajectory, there are two key differences. Firstly, the final state differs, and secondly, the trajectory oscillates. Both properties are a result of the inaccuracy of the learned transform, which influences the feedback linearizing parameters $\alpha(\boldsymbol{Y})$ and $\beta(\boldsymbol{Y})$. Error in these parameters causes a mismatched between the real dynamics and those cancelled by the controller. This results in the observed oscillations, as well as in steady state offset.

These results show that for a small set of states, a non-collocated controller can drive the SMDP system exactly to a desired state without relying on damping. However, implementation is significantly limited by loss of controllability and the shape of the potential function. A discussion of non-collocated controller limitations is provided in the following section.

VII. Discussion

The results presented in this report show that the control problem for certain classes of underactuated robots may be simplified using an appropriate choice of coordinates. Furthermore, the feasibility of obtaining a coordinate transformation using machine learning is demonstrated. In this section we highlight a number of interesting findings, limitations of the current methodology, and avenues for future work.

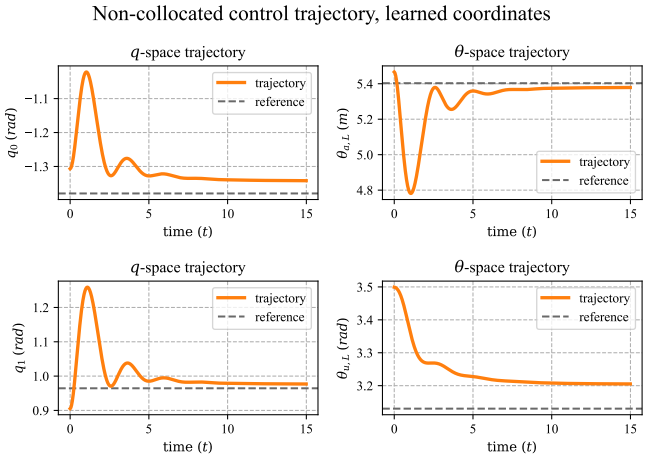

Fig. 18: Example trajectory of non-collocated controller applied to learned coordinates $\boldsymbol{\theta}_L$.

A. Autoencoder

1) Higher-dimensional systems: The autoencoder framework has been developed for 2-DOF systems, and was tuned for the SMDP. Performance on the series elastic actuator decoupling coordinates, without retuning hyperparameters, shows promising results for generalisability of the framework (see Appendix D). However, further testing must be performed to verify if the methodology can be extended to higher dimensions. A more expressive hidden architecture, or even a different architecture, may be required to handle more complex problems.

2) Alternative ML frameworks: In this work, a simple autoencoder architecture was used, and required physical properties were imposed through loss terms. The choice of autoencoder was based on their straightforward implementation and ease of loss function definition. A regular autoencoder was selected rather than a variational autoencoder (VAE), since there was no need to generate new data from the latent space. A VAE could potentially improve smoothness of the learned transformation, but this was not found to be a problem for the considered systems. It would be interesting to experiment with alternative machine learning frameworks, such as invertible neural networks, symbolic regression or VAEs, to compare their results.

3) Effects of model uncertainty: The autoencoder training, control input calculation and dynamics simulation were all performed on the same model, without any noise. This is a very optimistic scenario, and model uncertainty and sensor error in real applications would affect the quality of the learned transformation. This would be particularly important for performance of the feedback linearizing controller, since such a controller relies on exact cancellation of system dynamics.

4) Reliance on analytic input decoupling: A number of loss terms for the machine learning setup are based on an analytic expression of the input decoupling coordinate. These terms are applicable for systems which are classified as *collocated*

in [35], and improve the autoencoder performance. It may be interesting to attempt learning decoupling coordinates without these explicit terms, to make the methodology less reliant on analytic methods. In this case, the loss function would need to be amended, to ensure that input decoupling is imposed properly.

5) Learning full inertial decoupling: The current autoencoder loss function is designed to learn a well-defined mapping that results in an input decoupled and partially inertial decoupled system. There is, however, no loss term that requires full inertial decoupling. This could be implemented with a loss term on the inertia matrix gradients, which can be computed efficiently using modern automatic differentiation tools.

B. Non-collocated control

1) Loss of control authority: The presented feedback linearizing non-collocated controller leverages potential couplings. Well-defined-ness of the potential function for a double pendulum can be guaranteed in joint space with sufficiently large springs on each joint. However, when transforming to decoupling coordinates, this property is not necessarily preserved. In parts of the SMDP state space, the potential force becomes aligned with the actuator, in which case control authority of the passive DOF θ_u is lost. This can be understood by looking at the linearizing control input $u = \frac{1}{\beta(\boldsymbol{Y})}(-\alpha(\boldsymbol{Y}) + v)$. Exactly at those points $\beta(\boldsymbol{Y})$ tends to zero. This represents a loss of control authority, and the resulting control input u will destabilise the system. (At other points in the state space, the internal dynamics term $\alpha(\boldsymbol{Y})$ may tend to infinity, causing the same issue.) This greatly reduces the number of stable trajectories, and limits the application of such a non-collocated controller. Clever tricks, such as using a constant $\alpha(\bar{\boldsymbol{Y}})$ and $\beta(\bar{\boldsymbol{Y}})$, or clamping the control input u may allow for more robust control. However, more fundamentally this problem may be alleviated by shaping the potential function.

2) Non-uniqueness of non-collocated mapping: Depending on the potential function of the system, the mapping from decoupled coordinates to linearized state $\boldsymbol{Y}$ may not be bijective. In particular, a single state in $\boldsymbol{Y}$-space may map to multiple states in $\boldsymbol{\theta}$-space. Practically, this means that the controller may steer the system towards the wrong state. To alleviate this problem, the controller can be used locally. Again, shaping of the potential function (through intelligent placement of springs) may expand the usable region of the controller.

C. Future work

The above-mentioned shortcomings present opportunities for future work in order to better judge the application potential of the presented classification and learned coordinate methodology. In particular, we think that validation of the autoencoder framework to more complex systems is vital for real-world application of this methodology.

A rigorously tested autoencoder framework could be used as a tool to better understand complex underactuated systems. Results from the learned coordinate transformation could guide researchers to derive analytic coordinate transformations, or find underlying structure between robots that at first glance seem unrelated.

VIII. Conclusion

In this thesis, we investigated how inertial and input decoupling through a change of coordinate can be used for control of underactuated robotic systems. The first contribution of this thesis is an identification of three classes of underactuated systems based on the degree to which their dynamics can be decoupled, and an analysis of how such decoupling affects controller design. Afterwards, an autoencoder framework was developed with a loss function designed to find such coordinates. Results of the learned coordinates show that the autoencoder is able to find a decoupling coordinate transformation for a 2-DOF toy system. A collocated and non-collocated controller were developed to demonstrate the benefit of decoupling coordinates on this system. Performance of different controllers was presented, which showed the benefits of input decoupling for collocated control, and full decoupling for non-collocated control. The results were discussed, and future work was identified. Such work should focus on extending the learned coordinate methodology to more complex systems, and researching the applicability of the presented non-collocated controller to systems with different potential energy functions.

References

[1] A. Isidori, *Nonlinear Control Systems*. Communications and Control Engineering Series, New York, NY: Springer, 3 ed., Aug. 2006.

[2] M. W. Spong, S. Hutchinson, M. Vidyasagar, *et al.*, *Robot modeling and control*, vol. 3. Wiley New York, 2006.

[3] M. Bottin and G. Rosati, "Comparison of under-actuated and fully actuated serial robotic arms: A case study," *Journal of Mechanisms and Robotics*, vol. 14, Dec. 2021.

[4] S. Ajwad, M. Ullah, B. Khelifa, and J. Iqbal, "A comprehensive state-of-the-art on control of industrial articulated robots," *Journal of Balkan Tribological Association*, vol. 20, no. 4, pp. 499–521, 2014.

[5] J. IQBAL, Z. H. KHAN, and A. KHALID, "Prospects of robotics in food industry," *Food Science and Technology*, vol. 37, p. 159–165, May 2017.

[6] W. Ji and L. Wang, "Industrial robotic machining: a review," *The International Journal of Advanced Manufacturing Technology*, vol. 103, p. 1239–1255, Apr. 2019.

[7] M. W. Spong, "Underactuated mechanical systems," in *Control Problems in Robotics and Automation* (B. Siciliano and K. P. Valavanis, eds.), (Berlin, Heidelberg), pp. 135–150, Springer Berlin Heidelberg, 1998.

[8] A. L. Gunderman, J. Collins, A. Myer, R. Threlfall, and Y. Chen, "Tendon-driven soft robotic gripper for berry harvesting," 2021.

[9] F. J. Ruiz-Ruiz, J. Ventura, C. Urdiales, and J. M. G. de Gabriel, "Compliant gripper with force estimation for physical human–robot interaction," *Mechanism and Machine Theory*, vol. 178, p. 105062, 2022.

[10] E. Bolívar, S. Rezazadeh, T. Summers, and R. D. Gregg, "Robust optimal design of energy efficient series elastic actuators: Application to a powered prosthetic ankle," in *2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR)*, pp. 740–747, 2019.

[11] P. A. Bhounsule, J. Cortell, A. Grewal, B. Hendriksen, J. G. D. Karssen, C. Paul, and A. Ruina, "Low-bandwidth reflex-based control for lower power walking: 65 km on a single battery charge," *The International Journal of Robotics Research*, vol. 33, no. 10, pp. 1305–1321, 2014.

[12] D. S. Carabis and J. T. Wen, "Trajectory generation for flexible-joint space manipulators," *Frontiers in Robotics and AI*, vol. 9, Mar. 2022.

[13] M. Bergerman and Y. Xu, "Robust joint and cartesian control of underactuated manipulators," *Journal of Dynamic Systems, Measurement, and Control*, vol. 118, pp. 557–565, 09 1996.

[14] R. Tedrake, *Underactuated Robotics*. 2023.

[15] G. Oriolo and Y. Nakamura, "Control of mechanical systems with second-order nonholonomic constraints: underactuated manipulators," in *[1991] Proceedings of the 30th IEEE Conference on Decision and Control*, pp. 2398–2403 vol.3, 1991.
[16] G. Turrisi, M. Capotondi, C. Gaz, V. Modugno, G. Oriolo, and A. D. Luca, "On-line learning for planning and control of underactuated robots with uncertain dynamics," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 358–365, 2022.
[17] C. Della Santina, *Flexible Manipulators*, pp. 1–15. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020.
[18] D. CHEN and B. PADEN, "Stable inversion of nonlinear non-minimum phase systems," *International Journal of Control*, vol. 64, no. 1, pp. 81–97, 1996.
[19] M. Spong, "Partial feedback linearization of underactuated mechanical systems," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*, vol. 1, pp. 314–321 vol.1, 1994.
[20] P. Liljeback, K. Y. Pettersen, O. Stavdahl, and J. T. Gravdahl, *Snake Robots: Modelling, Mechatronics, and Control*. Springer London, 2013.
[21] B. J. Emran and H. Najjaran, "A review of quadrotor: An underactuated mechanical system," *Annual Reviews in Control*, vol. 46, pp. 165–180, 2018.
[22] M. Spong, "The swing up control problem for the acrobot," *IEEE Control Systems Magazine*, vol. 15, no. 1, pp. 49–55, 1995.
[23] S. A. Tafrishi, M. Svinin, and M. Yamamoto, "Singularity-free inverse dynamics for underactuated systems with a rotating mass," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, p. 3981–3987, IEEE, May 2020.
[24] D. Wang and M. Vidyasagar, "Control of a class of manipulators with a single flexible link: Part i—feedback linearization," *Journal of Dynamic Systems, Measurement, and Control*, vol. 113, pp. 655–661, 12 1991.
[25] M. W. Spong, "Modeling and control of elastic joint robots," *Journal of Dynamic Systems, Measurement, and Control*, vol. 109, pp. 310–318, 12 1987.
[26] G. Garofalo, F. Beck, and C. Ott, "Task-space tracking control for underactuated aerial manipulators," in *2018 European Control Conference (ECC)*, pp. 628–634, 2018.
[27] F. Beck, N. Sakamoto, and C. Ott, "Control of a class of underactuated systems by successive submanifold stabilization," *IFAC-PapersOnLine*, vol. 54, no. 19, pp. 352–358, 2021. 7th IFAC Workshop on Lagrangian and Hamiltonian Methods for Nonlinear Control LHMNC 2021.
[28] G. Garofalo, B. Henze, J. Englsberger, and C. Ott, "On the inertially decoupled structure of the floating base robot dynamics," *IFAC-PapersOnLine*, vol. 48, no. 1, pp. 322–327, 2015. 8th Vienna International Conferenceon Mathematical Modelling.
[29] M. Keppler, C. Ott, and A. Albu-Schäffer, "From underactuation to quasi-full actuation: Aiming at a unifying control framework for articulated soft robots," *International Journal of Robust and Nonlinear Control*, vol. 32, p. 5453–5484, Mar. 2022.
[30] C. Della Santina, C. Duriez, and D. Rus, "Model-based control of soft robots: A survey of the state of the art and open challenges," *IEEE Control Systems Magazine*, vol. 43, no. 3, pp. 30–65, 2023.
[31] P. Pustina, C. D. Santina, and A. De Luca, "Feedback regulation of elastically decoupled underactuated soft robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4512–4519, 2022.
[32] K. Berahmand, F. Daneshfar, E. S. Salehi, Y. Li, and Y. Xu, "Autoencoders and their applications in machine learning: a survey," *Artificial Intelligence Review*, vol. 57, Feb. 2024.
[33] M. Lepri, D. Bacciu, and C. D. Santina, "Neural autoencoder-based structure-preserving model order reduction and control design for high-dimensional physical systems," 2023.
[34] S. E. Otto, G. R. Macchio, and C. W. Rowley, "Learning nonlinear projections for reduced-order modeling of dynamical systems using constrained autoencoders," 2023.
[35] P. Pustina, C. D. Santina, F. Boyer, A. De Luca, and F. Renda, "Input decoupling of lagrangian systems via coordinate transformation: General characterization and its application to soft robotics," *IEEE Transactions on Robotics*, vol. 40, p. 2098–2110, 2024.
[36] D. J. Block and M. W. Spong, "Mechanical design and control of the pendubot," *SAE transactions*, pp. 36–43, 1995.
[37] P. Liu, M. N. Huda, L. Sun, and H. Yu, "A survey on underactuated robotic systems: Bio-inspiration, trajectory planning and control," *Mechatronics*, vol. 72, p. 102443, 2020.
[38] B. He, S. Wang, and Y. Liu, "Underactuated robotics: A review," *International Journal of Advanced Robotic Systems*, vol. 16, no. 4, p. 1729881419862164, 2019.
[39] B. J. Emran and H. Najjaran, "A review of quadrotor: An underactuated mechanical system," *Annual Reviews in Control*, vol. 46, p. 165–180, 2018.
[40] J. Ackermann, "Entwurf durch polvorgabe / design by pole placement," *at - Automatisierungstechnik*, vol. 25, no. 1-12, pp. 173–179, 1977.

APPENDIX

A. Dynamic coupling of double mass double pendulum

Lemma 1. *The double mass Pendubot system following the dynamics of Equation* (12)*, with input matrix as Equation* (13) *and inertia matrix as Equation* (14)*, can be written in input decoupled form, but not in partially decoupled or fully decoupled form.*

Proof. Consider the dynamics when absolute joint angles are chosen as system coordinates, and spring- and damping forces are ignored, namely:

$$\boldsymbol{M}_q(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}_q(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}_q(\boldsymbol{q}) = \boldsymbol{A}_q(\boldsymbol{q})u. \tag{41}$$

The corresponding inertia matrix is:

$$\boldsymbol{M}_q(\boldsymbol{q}) = \begin{bmatrix} l_0^2(m_0+m_1) & l_0 l_1 m_1 \cos(q_0-q_1) \\ l_0 l_1 m_1 \cos(q_0-q_1) & l_1^2 m_1 \end{bmatrix}, \tag{42}$$

and the input matrix is simply:

$$\boldsymbol{A}_q(\boldsymbol{q}) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{43}$$

We look for a new set of coordinates $\boldsymbol{\phi} = \boldsymbol{f}(\boldsymbol{q})$ that result in input decoupling and (partial) inertial decoupling. From the findings of [35], we know that we must have $\phi_0 = q_0$, since the system is already input decoupled in initial coordinates. This leads to the following Jacobian:

$$\boldsymbol{J}_f(\boldsymbol{q}) = \begin{bmatrix} 1 & 0 \\ a(\boldsymbol{q}) & b(\boldsymbol{q}) \end{bmatrix}, \tag{44}$$

where the parameters $a(\boldsymbol{q})$ and $b(\boldsymbol{q})$ are free to choose. The inertia matrix in $\boldsymbol{\phi}$-space is given by:

$$\boldsymbol{M}_\phi(\boldsymbol{\phi}) = \boldsymbol{J}_f^{-T}\boldsymbol{M}_q(\boldsymbol{q})\boldsymbol{J}_f^{-1}. \tag{45}$$

From Equation (44) we know that:

$$\boldsymbol{J}_f^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{a(\boldsymbol{q})}{b(\boldsymbol{q})} & \frac{1}{b(\boldsymbol{q})} \end{bmatrix}. \tag{46}$$

For readability, we may define $c(\boldsymbol{q}) = \frac{1}{b(\boldsymbol{q})}$, such that Equation (46) can be written as:

$$\boldsymbol{J}_f^{-1} = \begin{bmatrix} 1 & 0 \\ -a(\boldsymbol{q})c(\boldsymbol{q}) & c(\boldsymbol{q}) \end{bmatrix}, \tag{47}$$

and

$$\boldsymbol{J}_f^{-T} = \begin{bmatrix} 1 & -a(\boldsymbol{q})c(\boldsymbol{q}) \\ 0 & c(\boldsymbol{q}) \end{bmatrix}. \tag{48}$$

To simplify analysis, write the inertia matrix in original coordinates as follows:

$$\boldsymbol{M}_q(\boldsymbol{q}) = \begin{bmatrix} M_{00} & M_{01} \\ M_{01} & M_{11} \end{bmatrix}. \tag{49}$$

Then, the resulting inertia matrix in proposed coordinates $\boldsymbol{\phi}$ is:

$$\boldsymbol{M}_\phi(\boldsymbol{\phi}) = \begin{bmatrix} M_{\phi_{0,0}} & c(\boldsymbol{q})M_{01} - a(\boldsymbol{q})c^2(\boldsymbol{q})M_{11} \\ c(\boldsymbol{q})M_{01} - a(\boldsymbol{q})c^2(\boldsymbol{q})M_{11} & M_{\phi_{1,1}} \end{bmatrix}. \tag{50}$$

The key requirement that follows from this equation is that the off-diagonal component must be zero. Writing out the matrix blocks we find:

$$c(\boldsymbol{q})l_0 l_1 m_1 \cos(q_0 - q_1) - a(\boldsymbol{q})c^2(\boldsymbol{q})l_1^2 m_1 = 0. \tag{51}$$

For this condition to hold for all $\boldsymbol{q}$, the two terms of this equation must each equal zero. From:

$$c(\boldsymbol{q})l_0 l_1 m_1 \cos(q_0 - q_1) = 0 \tag{52}$$

with non-zero l_0, l_1 and m_1, we find that $c(\boldsymbol{q})$ must be zero. If $c(\boldsymbol{q}) = 0$, the Jacobian devolves into:

$$\boldsymbol{J}_\phi(\boldsymbol{\phi}) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}. \tag{53}$$

This would require the second coordinate ϕ_1 to be constant, which is means the coordinate change would not be valid. Therefore, no choice of coordinates exists for which the two mass Pendubot system is input- and inertially decoupled. □

B. Inertial coupling of single mass double pendulum

Lemma 2. *The SMDP system following the dynamics of Equation* (12)*, with input matrix as Equation* (15) *and inertia matrix as Equation* (16)*, can be written in partially decoupled form, but not in fully decoupled form.*

Proof. It can be shown that no smooth and invertible mapping exists for which the SMDP system is both input decoupled and fully inertially decoupled. To show that such a mapping does do not exist, one can show that these requirements lead to conflicting terms in the transformation Jacobian.
In trying to find these new coordinates, we look for a new set of coordinates $\boldsymbol{\phi} = \boldsymbol{f}(\boldsymbol{\theta})$. The requirements on this new coordinate can be made clear through the Jacobian:

$$\boldsymbol{J}_f(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial \phi_0}{\partial \theta_0} & \frac{\partial \phi_0}{\partial \theta_1} \\ \frac{\partial \phi_1}{\partial \theta_0} & \frac{\partial \phi_1}{\partial \theta_1} \end{bmatrix} \tag{54}$$

The inertia- and input matrix in these new coordinates can be calculated as follows:

$$\boldsymbol{M}_\phi(\boldsymbol{\phi}) = \boldsymbol{J}_f^{-T}(\boldsymbol{\theta})\boldsymbol{M}_\theta(\boldsymbol{\theta})\boldsymbol{J}_f^{-1}(\boldsymbol{\theta}) \tag{55}$$

$$\boldsymbol{A}_\phi(\boldsymbol{\phi}) = \boldsymbol{J}_f^{-T}(\boldsymbol{\theta})\boldsymbol{A}_\theta(\boldsymbol{\theta}) \tag{56}$$

For an invertible mapping, the inverse Jacobian and inverse transpose Jacobian are as follows:

$$\boldsymbol{J}_f^{-1}(\boldsymbol{\phi}) = \begin{bmatrix} \frac{\partial \theta_0}{\partial \phi_0} & \frac{\partial \theta_0}{\partial \phi_1} \\ \frac{\partial \theta_1}{\partial \phi_0} & \frac{\partial \theta_1}{\partial \phi_1} \end{bmatrix}, \qquad \boldsymbol{J}_f^{-T}(\boldsymbol{\phi}) = \begin{bmatrix} \frac{\partial \theta_0}{\partial \phi_0} & \frac{\partial \theta_1}{\partial \phi_0} \\ \frac{\partial \theta_0}{\partial \phi_1} & \frac{\partial \theta_1}{\partial \phi_1} \end{bmatrix} \tag{57}$$

As such, the input matrix in $\boldsymbol{\phi}$-coordinates is as follows:

$$\boldsymbol{A}_\phi(\boldsymbol{\phi}) = \begin{bmatrix} \frac{\partial \theta_0}{\partial \phi_0} \\ \frac{\partial \theta_0}{\partial \phi_1} \end{bmatrix} \tag{58}$$

and the inertia matrix in ϕ-coordinates is:

$$\boldsymbol{M}_\phi(\boldsymbol{\phi}) = \begin{bmatrix} (\frac{\partial \theta_0}{\partial \phi_0})^2 m + (\frac{\partial \theta_1}{\partial \phi_0})^2 m\theta_0^2 & \frac{\partial \theta_0}{\partial \phi_0}\frac{\partial \theta_0}{\partial \phi_1} m + \frac{\partial \theta_1}{\partial \phi_0}\frac{\partial \theta_1}{\partial \phi_1} m\theta_0^2 \\ \frac{\partial \theta_0}{\partial \phi_0}\frac{\partial \theta_0}{\partial \phi_1} m + \frac{\partial \theta_1}{\partial \phi_0}\frac{\partial \theta_1}{\partial \phi_1} m\theta_0^2 & (\frac{\partial \theta_0}{\partial \phi_1})^2 m + (\frac{\partial \theta_1}{\partial \phi_1})^2 m\theta_0^2 \end{bmatrix}. \tag{59}$$

Following from requirement 2.(b) we must have $\frac{\partial \theta_0}{\partial \phi_1} = 0$. This results in mass matrix:

$$\boldsymbol{M}_\phi(\boldsymbol{\phi}) = \begin{bmatrix} (\frac{\partial \theta_0}{\partial \phi_0})^2 m + (\frac{\partial \theta_1}{\partial \phi_0})^2 m\theta_0^2 & \frac{\partial \theta_1}{\partial \phi_0}\frac{\partial \theta_1}{\partial \phi_1} m\theta_0^2 \\ \frac{\partial \theta_1}{\partial \phi_0}\frac{\partial \theta_1}{\partial \phi_1} m\theta_0^2 & (\frac{\partial \theta_1}{\partial \phi_1})^2 m\theta_0^2 \end{bmatrix} \tag{60}$$

Following from requirement 1.(b) we must have $\frac{\partial \theta_1}{\partial \phi_1} \neq 0$. Since requirement 1.(a) states that the off-diagonal entries must be zero, it naturally follows that $\frac{\partial \theta_1}{\partial \phi_0} = 0$.

As such, the matrix simplifies to:

$$\boldsymbol{M}_\phi(\boldsymbol{\phi}) = \begin{bmatrix} (\frac{\partial \theta_0}{\partial \phi_0})^2 m & 0 \\ 0 & (\frac{\partial \theta_1}{\partial \phi_1})^2 m\theta_0^2 \end{bmatrix} \tag{61}$$

Again, following from requirement 1.(b) we know that the bottom-right entry must not be a function of θ_0, and so we must have $(\frac{\partial \theta_1}{\partial \phi_1})^2 m\theta_0^2 = g(\theta_1)^2 m$.
This can be rearranged to $(\frac{\partial \theta_1}{\partial \phi_1})^2 = \frac{g(\theta_1)^2}{\theta_0^2}$, or $\frac{\partial \theta_1}{\partial \phi_1} = \pm\frac{g(\theta_1)}{\theta_0}$, where $g(\theta_1) \neq 0$.
We can thus construct the Jacobian as follows:

$$\boldsymbol{J}_f^{-1}(\boldsymbol{\phi}) = \begin{bmatrix} \frac{\partial \theta_0}{\partial \phi_0} & 0 \\ 0 & \pm\sqrt{g(\theta_1)}\frac{1}{\theta_0} \end{bmatrix} \tag{62}$$

$$\boldsymbol{J}_f(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial \phi_0}{\partial \theta_0} & 0 \\ 0 & \pm\theta_0\frac{1}{\sqrt{g(\theta_1)}} \end{bmatrix} \tag{63}$$

Following Schwarz's theorem on mixed partial derivatives, we know that for $\boldsymbol{\phi} = \boldsymbol{f}(\boldsymbol{\theta})$ to be a valid coordinate transformation, we must have:

$$\frac{\partial}{\partial \theta_1}(\frac{\partial \phi_1}{\partial \theta_0}) = \frac{\partial}{\partial \theta_0}(\frac{\partial \phi_1}{\partial \theta_1}) \tag{64}$$

From the Jacobian we know that:

$$\frac{\partial \phi_1}{\partial \theta_0} = 0, \qquad \frac{\partial \phi_1}{\partial \theta_1} = \pm\theta_0\frac{1}{\sqrt{g(\theta_1)}}, \tag{65}$$

$$\frac{\partial}{\theta_1}(\frac{\partial \phi_1}{\partial \theta_0}) = 0 \quad \neq \quad \frac{\partial}{\theta_0}(\frac{\partial \phi_1}{\partial \theta_1}) = \frac{1}{\sqrt{g(\theta_1)}} \tag{66}$$

Meaning that there is no selection of coordinates ϕ which leads to full inertial and input decoupling. □

C. Hyperparameter tuning

In order to achieve good learning performance, a number of hyperparameters needed to be selected. The most important of these are as follows:

The encoder and decoder equally consist of 2 hidden layers of 32 neurons. These are fully connected, and a sigmoid activation function was used (see Figure 19). The Adam optimizer was used with a learning rate of 10^{-3}. A training set of 4200 samples and validation set of 1800 samples was used to train for 5000 epochs.

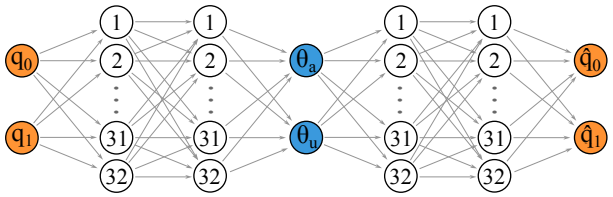

Fig. 19: The autoencoder framework, consisting of an identical encoder and decoder, which both contain two fully connected hidden layers and sigmoid activation functions.

In order to improve learning performance, the weights of the loss terms was optimized. Each loss term was initialized at a weight of 1, with the reconstruction loss weighed more heavily at a value of 4. Independently, each term was then increased or decreased by a factor 1.25, after which the model was trained. The performance of the learned and the optimal combination of these terms was selected, based on the mean squared error (MSE) of reconstruction, inertial decoupling and input decoupling. Performance varied only slightly with varying loss weights, so this crude optimization was only performed once for each term to result at the final loss weights.

D. Learning decoupling coordinates of series elastic actuator

In order to test the generalizability of the machine learning framework, the autoencoder has also been used to learn decoupling coordinates of the series elastic actuator (SEA). The standard choice of SEA coordinates, namely the motor angle θ_a and link angle θ_u, results in an input- and fully inertially decoupled system. Therefore, another set of coordinates must be chosen as "naive" coordinates.

To this end, the motor angle $q_0 = \theta_a$ and relative angle q_1 are chosen as naive coordinates. With motor and link inertias I_m and I_l, spring stiffness k and control input u, the system dynamics are:

$$\boldsymbol{M}_q \ddot{\boldsymbol{q}} + \boldsymbol{G}_q(\boldsymbol{q}) = \boldsymbol{A}_q u. \tag{67}$$

Here, the inertia matrix is:

$$\boldsymbol{M}_q = \begin{bmatrix} I_0 + I_1 & I_1 \\ I_1 & I_1 \end{bmatrix} \tag{68}$$

and the input matrix is:

$$\boldsymbol{A}_q = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{69}$$

The standard, fully decoupling coordinates can be obtained through a linear transformation. After all, the equations of motion in decoupled coordinates $\boldsymbol{\theta}$ are:

$$\boldsymbol{M}_\theta \ddot{\boldsymbol{\theta}} + \boldsymbol{G}_\theta(\boldsymbol{\theta}) = \boldsymbol{A}_\theta u \tag{70}$$

with inertia matrix:

$$\boldsymbol{M}_q = \begin{bmatrix} I_0 & 0 \\ 0 & I_1 \end{bmatrix} \tag{71}$$

and input matrix:

$$\boldsymbol{A}_\theta = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{72}$$

The analytic and learned decoupling coordinates of the series elastic actuator are shown in Figure 20. Here, a very similar pattern to the SMDP case is visible. The learned actuated coordinate agrees very with the analytic solution. For the unactuated coordinate, the learned and analytic coordinate are again related through an affine transformation.

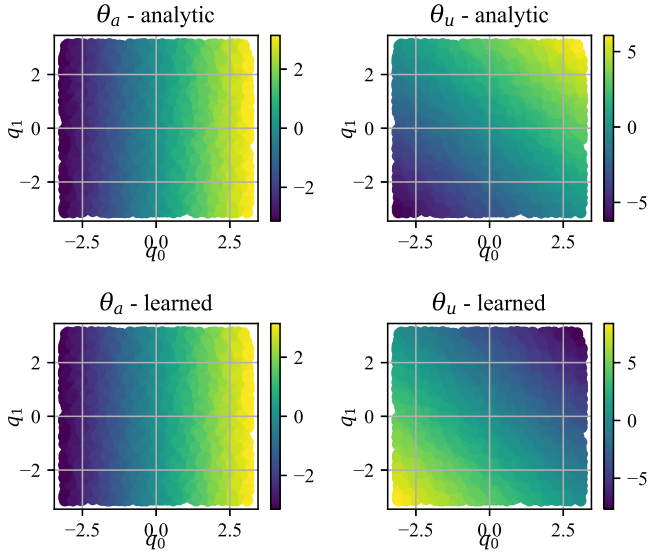

Fig. 20: Analytic and learned $\boldsymbol{\theta}$ for the series elastic actuator.

The learned coordinates provide incredibly good results for both reconstruction and decoupling. Figure 21 shows that the reconstruction error of this transform is in the order of 10^{-6} radians.

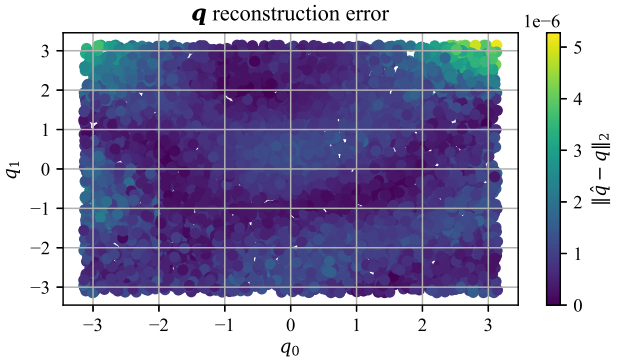

Fig. 21: Reconstruction error of SEA

Jacobian invertibility error, calculated as the difference between the product of encoder and decoder Jacobian with the identity matrix, is similarly small, as can be seen in Figure 22.

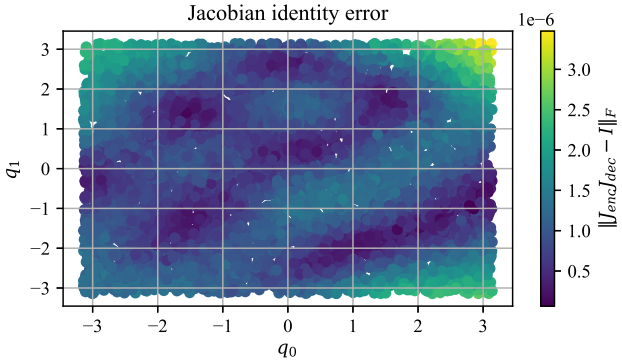

Fig. 22: Jacobian identity error of SEA

Input decoupling performance is also outstanding, with similarly negligible losses (see Figure 23).

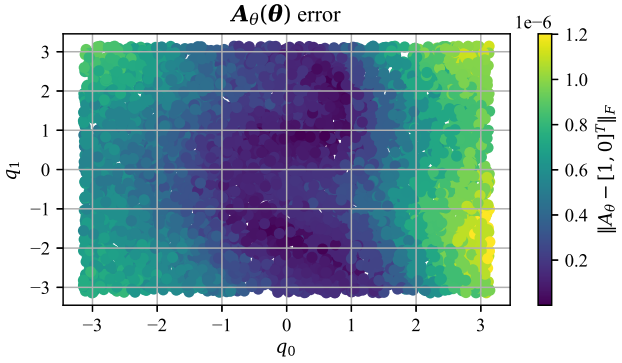

Fig. 23: Input matrix error of SEA

Lastly, inertial decoupling performance is shown in Figure 24. This paints the same picture of errors in the order of magnitude of 10^{-6} radians.

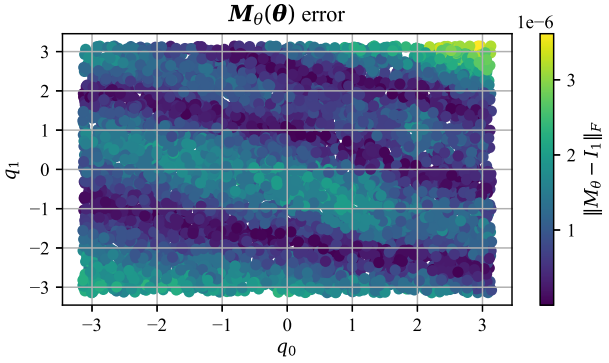

Fig. 24: Inertia matrix error of SEA

It can be concluded that at least for a simple case such as the series elastic actuator, the autoencoder framework is also successful at learning decoupling coordinates. This is a promising result, since no hyperparameters had to be altered from the SMDP learning case.

E. Collocated control performance metrics

In order to compare performance of the three collocated controllers, some metrics were selected. Their calculation is described below. Some variables are written explicitly as a function of time for clarity.

In the undamped case, the variable of interest is the rod length θ_0. Average error of the steady state oscillation can then be calculated as:

$$RMSE_\theta = \sqrt{\frac{\sum_{t=2}^{t_{end}} \left(\theta_0 - \bar{\theta}_0\right)^2}{(t_{end} - 2)/dt}} \tag{73}$$

Where t_{end} is the time at the end of the simulation, and dt is the discrete time step. This was defined as such because a single measurement at the end of the simulation may not represent the average error behaviour, and because after 2 seconds simulations had mostly settled to their steady-state behaviour. Control effort was straightforwardly defined as:

$$Control\ \ effort = \sum_{t=0}^{t_{end}} u^2 \tag{74}$$

Settling time was then measured as the time it took for a trajectory to reach and stay within a 5% of the desired rod length. This percentage is compared to the distance between desired and starting configuration. Similarly, overshoot computes the percentage that the trajectory exceeds the desired rod length, compared to this distance.

For the damped simulations, performance was measured based on the joint angles. Calculation of control effort, settling time and overshoot were identically to the undamped case. Configuration error was calculated differently, since the damped simulations reach a single steady value. This final value was compared to the desired value to determine the configuration error.

F. Learned coordinate trajectory generation

In order to test the performance of the learned coordinate mapping in practice, SMDP trajectories were simulated on the learned coordinates, and compared to ground truth trajectories. The robot starting and desired conditions were varied within the trained configuration space of the autoencoder. Every trajectory was simulated for 3 seconds. A trajectory was included in the analysis if it met three criteria: stayed within learned coordinates, arm did not fold out fully and arm did not fold in fully. In those cases, simulations destabilized and caused noisy data.

An example trajectory is shown in Figure 25. Here, the learned and ground truth trajectory are shown in orange and blue, respectively. Darker colours indicate a later moment in the simulation. As can be seen, the simulations are close for certain parts of the configuration space, and further apart for others.

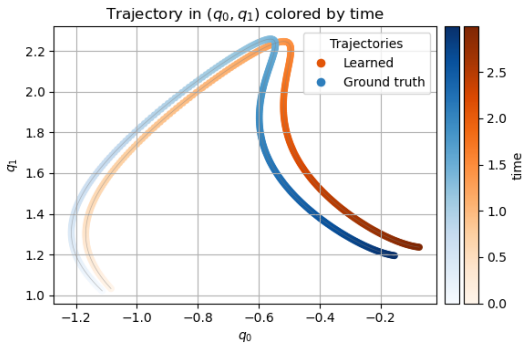

Fig. 25: Example trajectory showing the end-effector location in q-space as a function of time..

Corresponding to the trajectory in Figure 25, movement of the end-effector can also be visualized as in Figure 26. This illustrates how the simulations diverge more as the end-effector moves towards the edge of the trained domain. This is in line with the results presented in Section VI.

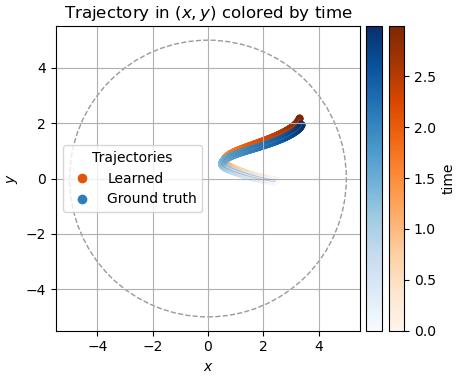

Fig. 26: Trajectory of end-effector in Cartesian space.