# Online Simultaneous Semi-Parametric Dynamics Model Learning

# Online Simultaneous Semi-Parametric Dynamics Model Learning

Joshua Smith[1] and Michael Mistry[1]

***Abstract*—Accurate models of robots' dynamics are critical for control, stability, motion optimization, and interaction. Semi-Parametric approaches to dynamics learning combine physics-based Parametric models with unstructured Non-Parametric regression with the hope to achieve both accuracy and generalizability. In this paper, we highlight the non-stationary problem created when attempting to adapt both Parametric and Non-Parametric components simultaneously. We present a consistency transform designed to compensate for this non-stationary effect, such that the contributions of both models can adapt simultaneously without adversely affecting the performance of the platform. Thus, we are able to apply the Semi-Parametric learning approach for continuous iterative online adaptation, without relying on batch or offline updates. We validate the transform via a perfect virtual model as well as by applying the overall system on a Kuka LWR IV manipulator. We demonstrate improved tracking performance during online learning and show a clear transference of contribution between the two components with a learning bias towards the Parametric component.**

***Index Terms*—Dynamics, Calibration and Identification, Model Learning for Control, Robust/Adaptive Control of Robotic Systems**

## I. Introduction

ROBOT platforms are becoming more capable with major advancements in actuator/robot mechanical design. These advancements are evident in the slew of new robotic platforms. However, one of the limitations present is the inaccuracy of their dynamics models. This inaccuracy may affect many aspects of robots such as; control, stability, motion optimization, and interaction. Any task which is dependent on accurate force control or prediction is subject to issues such as; falling over, crashing into the environment, instability, or other dangerous behaviours. As such, there has been a drive to improve the dynamics models through better measurements and data-driven learning.

Current dynamics models can be split into three major model types: Parametric [1], [2], [3], Non-Parametric [4], [5], [6], and Semi-Parametric [7], [8], [9], [10], [11]. Whether the models are Parametric or not depends on the use, or not, of parameters based on known physics. In particular, Parametric models typically use the classical rigid body dynamics equation to model the forces of the robot and heavily rely on reformulations, known as regressors [1], [12], to isolate the inertial dynamic parameters. Parametric models tend to demonstrate strong ability in generalizing over the state space of the robot, with a small enough parameter error and state feedback control. Some examples also benefit from rigorous mathematical proofs of stability [2]. The Parametric models do suffer when non-modelled forces are present when implemented on real platforms which may negatively affect performance and learning.

Non-Parametric models fall under the context of machine learning, viewing the problem as a mapping function from the input state to the output torque. These models typically do not enforce any strict structure based on physics, allowing the model to learn any non-linear function which can encapsulate all and any forces present in the dynamics of the robot. Non-Parametric methods are normally data-driven as in the case of Gaussian Process Regression [5], Neural Networks and Gaussian Mixture Models [6]. The data-driven nature of the algorithms may cause issues if the data is of poor quality or limited in quantity which also affects the generalizability of the model.

Semi-Parametric models are one of the most recent categories proposed to model dynamics and are a combination of both the previous models, Parametric and Non-Parametric. The implementations vary in structure, but generally the Parametric component is designed to handle the traditional rigid body dynamics, whilst the Non-Parametric component designed for a non-linear error which represents all forces not defined in the Parametric component. As an example in [8], [9] the authors use similar techniques using Random Feature mapping and Recursive Regularized Least Squares to learn the Non-Parametric component whilst using standard least squares for the Parametric component for the Semi-Parametric with rigid body dynamics (RBD) mean with data in small batches. We use the Semi-Parametric model with RBD mean in [8], [9] as our baseline in section IV-A as it allows independent changes to both components.

The main focus of this paper will be on the issue of online simultaneous model updates. We focus on an online algorithm for faster and continuous learning of the dynamics model. The online aspect will also allow a faster reaction to any physical changes in the dynamics model. Whilst batch-based methods can update with respect to long term physical changes in the dynamics model, they are restricted by the time it takes to collect a batch before applying an update to the model causing higher frequency changes to be missed.

Manuscript received: September, 10th, 2019; Revised December, 16th, 2019; Accepted January, 14th, 2020.

This paper was recommended for publication by Editor Paolo Rocco upon evaluation of the Associate Editor and Reviewers' comments. This work has been supported by the following grants: EPSRC UK RAI Hub ORCA (EP/R026173/1) and NCNR (EPR02572X/1), CogIMon project in the EU Horizon 2020 (ICT-23-2014), THING project in the EU Horizon 2020 (ICT-2017-1), and by grant EP/L016834/1 for the University of Edinburgh RAS CDT from EPSRC.

[1]Authors are with the School of Informatics, University of Edinburgh, UK Joshua.Smith@ed.ac.uk, mmistry@inf.ed.ac.uk

Digital Object Identifier (DOI): see top of this page.

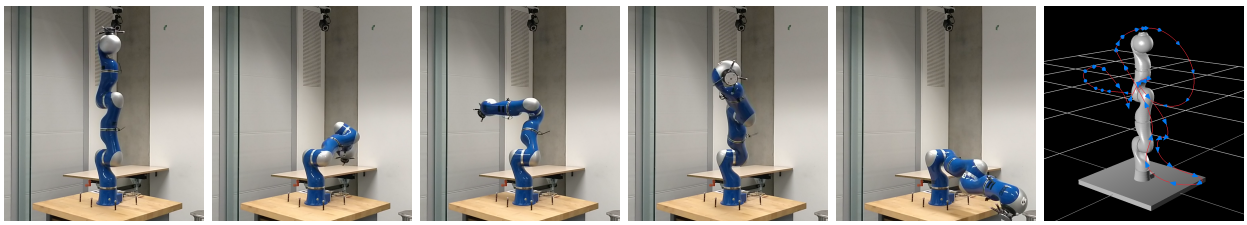


Figure 1: Snapshots of exemplar Modified Fourier Trajectory with Kuka LWR IV – The real arm was snapshot during the experiment in section IV-B, where it is initially controlled only through feedback control as shown in the first 5 figures. These photos provide an idea of the variety of configurations the robot explores. The last figure shows a visualization of the whole trajectory of the end-effector that the arm performs during the experiment.

Simultaneously updating both components of the model introduces the issue of a non-stationary target for the Non-Parametric component, which shall be described in the next section. We aim to solve this using a consistency transform that will avoid the need to retrain the Non-Parametric component, or the need to ignore previous data.

The contribution of this work is as follows:

- Introduction of the non-stationary issue during simultaneous learning for Semi-Parametric models.
- Proposal of an approximate consistency transform.
- Defining and analyzing an online simultaneous Semi-Parametric algorithm.

## II. Problem Statement

The true dynamics of a fixed robotic platform, with no contact, can be represented as:

$$\mathbf{M}(q)\ddot{q} + \mathbf{C}(q,\dot{q})\dot{q} + G(q) + F(q,\dot{q}) + \epsilon = \tau \tag{1}$$

Where $\mathbf{M}$ represents the inertia matrix, $\mathbf{C}$ the Coriolis and centrifugal matrix, $G$ the gravity vector and $F$ the Coulomb friction vector, which combined form the Parametric contribution of the model. The $\epsilon$ vector represents any other forces present, such as friction effects not in the coulomb model. $\tau$ is the joint torques due to the sum of these forces which are the full torques experienced/measured by the platform.

As mentioned the two components of the Semi-Parametric model learns the full torque model, $\tau$, where $\epsilon$ is learned by the Non-Parametric component which has the following form:

$$\epsilon = f(q,\dot{q},\ddot{q}) = \tau - \tau_c \tag{2}$$

$$\tau_c = \mathbf{M}(q)\ddot{q} + \mathbf{C}(q,\dot{q})\dot{q} + G(q) + F(q,\dot{q}) \tag{3}$$

Thus $\epsilon$ represents the torque error between the current state's ($q$, $\dot{q}$, $\ddot{q}$) parametric modelled torque, $\tau_c$, and the full measured torque output, $\tau$.

With the formulation of the model in (2), when we update the Parametric component the target function of the Non-Parametric component will change. Due to the change in the Non-Parametric target function, the problem is classed as non-stationary with respect to change in the inertial parameters. This non-stationary function means that the previously learned component may be inconsistent with new errors, when the Parametric component is updated, leading to incorrect predictions. The inaccurate predictions can increase errors in trajectory tracking relying on state feedback to control these incorrect torques or lead to instability.

The non-stationary nature of the problem is an important issue if we wish to maximize the contribution of the Parametric component such that the overall model becomes more state generalizable. The same issue can occur if we want to also improve estimations of the inertial parameters through the Parametric component, which can be used by other model predictive controllers.

To solve the issue of the non-stationary target function we propose an online Semi-Parametric algorithm with a consistency transform for the Non-Parametric component with respect to the known inertial parameter change. The consistency transform will allow the previously learned component to approximately remain consistent with the new target function without further learning or retraining.

## III. Methodology

In the implementation of this paper, we have used Composite Adaptive Control [2] and Gaussian Mixture Models [13] (GMMs) for the Parametric and Non-Parametric components respectively. The Parametric component can be freely chosen, however, [2] was chosen due to the proven stable adaptation. The Composite Adaptive Control algorithm also defines two sources of learning, the state (position and velocity) and torque errors. The two errors can be beneficial for Semi-Parametric controllers, as often the state errors are driven towards zero quickly, due to both components learning the correct torque combined. When the state errors approach zero the learning using this error stalls, meaning that the Parametric component will stop learning even if the inertial parameters are incorrect. By using the torque errors as well we can still drive the Parametric learning even if the state errors tend towards zero, allowing us to maximize the Parametric component's contribution and to minimize the error on the inertial parameters.

GMMs were chosen for the Non-Parametric component for several reasons. Firstly, efficient implementations of the algorithm exist as evidenced in [13], which compared against other methods such as Gaussian Process Regression, have a significantly smaller complexity. Secondly, the representation is statistically significant, which can reduce the amount of memory required due to previous data being described by the statistics. The last reason is due to the GMM's statistical nature, the space is easier to understand allowing a linear transformation to be defined and applied.

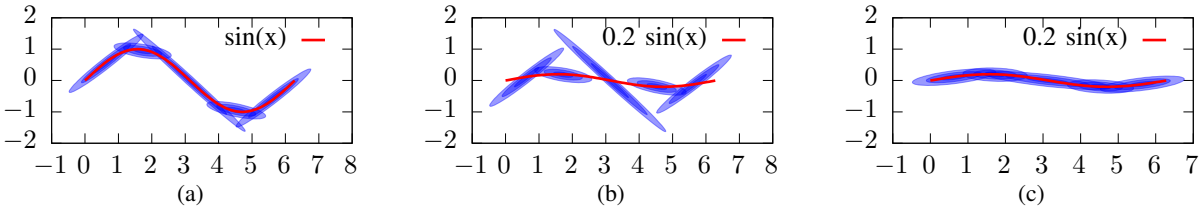



Figure 2: Consistency Example - Starting from the original Sine Wave (a), we demonstrate the idea of transforming the GMM with a known change in the local function. In this example, we take a standard Sine Wave and multiply it by 0.2 and apply the transformations from section III-D. (b) shows the mean transformation of the GMM sub-models which put the centre of each Gaussian onto the new updated function. (c) shows the result when the covariance transformation is applied to the sub-models in (b). The covariances adjust to the new slopes of the function and the new GMM is now consistent with the change in the function. It is important to note that no learning is done between (a) - (c), only the consistency transform applied.

### *A. Parametric Model*

The basic approach for the Parametric learning component is to rearrange the rigid body dynamics equations to be linear with respect to the dynamic parameters, creating a non-linear matrix based on the robot state and the kinematic parameters of the robot known as a regressor. The regressor is computable in closed form given kinematic knowledge of the robot platform. The regressor can then be inverted or used in an iterative least-squares algorithm to find the inertial parameters given input data. In this work we use a Composite Adaptive Control scheme [2], this combines both dynamics learning and control in a way that is provably stable with convergence in position, velocity and inertial parameters.

Composite Adaptive Control [2] learns using two sources of error from two different regressors. These regressors are defined as direct and indirect in [2]. The direct regressor formulation uses (4), known as resolved accelerations and velocities, to eliminate the need for acceleration measurements by using the knowledge of the desired trajectory. This allows the model learning to be driven by the position and velocity errors during the task, which can be used to show stable convergence in the tracking errors. We can then reformulate the rigid body dynamics equation into a matrix which is independent of the inertial parameters (5).

$$\dot{q}_r = \dot{q}_d - \boldsymbol{\Lambda}(q - q_d) \qquad \ddot{q}_r = \ddot{q}_d - \boldsymbol{\Lambda}(\dot{q} - \dot{q}_d) \tag{4}$$

$$\mathbf{M}(q)\ddot{q}_r + \mathbf{C}(q,\dot{q})\dot{q}_r + G(q) + F(q,\dot{q}) = \mathbf{Y}(q,\dot{q},\dot{q}_r,\ddot{q}_r)\pi \tag{5}$$

Where $\bullet_d$ is the desired value of $\bullet$, $\boldsymbol{\Lambda}$ is a positive definite gain matrix, $\mathbf{Y}$ is the direct regressor, and $\pi$ is the inertial parameter vector. Typically for each link $i$ the inertial parameters are: $\pi_i = [m_i, m_i c_i{}^T, I_{xxi}, I_{xyi}, I_{xzi}, I_{yyi}, I_{yzi}, I_{zzi}]^T$, where $m_i$, $c_i$ and $I_i$ are the mass, centre of mass and link inertia respectively.

The direct regressor suffers with parameter convergence of the model as the learning is not dependent on the torque predictions. The indirect formulation solves this issue by driving the update law on the predicted filtered torque error at the given state of the robot. The indirect regressor uses a filtering approach to reformulate the dynamics equation without the need for acceleration measurements.

$$\int_0^t w(r)\tau(r)dr = \mathbf{W}(\dot{q},q)\pi \tag{6}$$

Where $w(r)$ is the impulse response from a first-order filter and $\mathbf{W}$ is the filtered regressor.

By combining the two regressors into the single update law (7), the learning process is driven by both tracking and prediction errors. When combined with the control law in [2], the controller can be shown to be stable in the Lyapunov sense with convergence in the parameters, velocity and position [2].

$$\Delta\hat{\pi} = -\mathbf{P}(\mathbf{Y}(q,\dot{q},\dot{q}_r,\ddot{q}_r)^T s + \mathbf{W}(q,\dot{q})^T \mathbf{R}e) \tag{7}$$

Where $\Delta\bullet$ is the change in $\bullet$, $\hat{\bullet}$ is the estimated value of $\bullet$, $s = \dot{q} - \dot{q}_r$, $e$ is the measured torque error, and $\mathbf{P}$ and $\mathbf{R}$ are positive definite weight matrices.

### *B. Non-Parametric Model*

The general concept of the Gaussian Mixture Model is to represent a non-linear function through a sum of multidimensional Gaussian sub-models. This represents the full joint probability space for the inputs and outputs, as each Gaussian has the dimension of the input plus output dimensions. To define these Gaussian sub-models, to learn a function, they need to be placed throughout the state space following the training data. An example of a trained GMM on a Sine Wave is shown in figure 2a.

The GMM learning was implemented using the Iterative Gaussian Mixture Model (IGMM) algorithm defined in [13] which can be learned iteratively at a low computational cost. The algorithm also defines a metric for adding and removing components.

Using the GMM we apply a regression algorithm, known as Gaussian Mixture Regression, to predict an output based on a given input. The regression uses the fact that each Gaussian is defined as a prior, a mean vector ($\mu$), and a covariance matrix ($\Sigma$). As each Gaussian also represents the joint probability of the inputs and outputs, we can decompose the parameters representing the mean and covariance as (8). Using this structure we condition each Gaussian on the input dimensions, with a given input, to create a conditional mean for the output dimensions (9).

$$\mu^j = \begin{bmatrix} \mu_i^j \\ \mu_o^j \end{bmatrix} \qquad \Sigma^j = \begin{bmatrix} \Sigma_{i,i}^j & \Sigma_{i,o}^j \\ \Sigma_{o,i}^j & \Sigma_{o,o}^j \end{bmatrix} \tag{8}$$

$$\hat{x}_o = \sum_{j=1}^{M} p(j|x_i)\left(\mu_o^j + \Sigma_{o,i}^j {\Sigma_{i,i}^j}^{-1}\left(x_i - \mu_i^j\right)\right) \tag{9}$$

Where the $\bullet^j$ indicates the Gaussian $\bullet$ belongs to, $p(j|x_i)$ is the posterior probability of the input given the $j$th Gaussian, $i$ and $o$ represent the input ($q$, $\dot{q}$, $\ddot{q}$) and output ($\tau$) dimensions respectively, $\mu_i$ and $\mu_o$ represent the input and output dimension means, $\Sigma_{k,l}$ represents the covariance between the dimensions $k$ and $l$, and $\hat{x}_o$ represents the conditioned prediction output of the GMR algorithm.

### C. Semi-Parametric Model

The Semi-Parametric model takes both of these components and combines them such that the Parametric component describes the forces related to the dynamic parameters ($\mathbf{M}(q)$, $\mathbf{C}(q,\dot{q})$, $G(q)$, $F(q,\dot{q})$), whilst the Non-Parametric target is the error between the actual torque and the Parametric component. As mentioned in section II this can create an inconsistency between the components if both are updated, which could potentially cause erratic or dangerous behaviour through poor predictions.

### D. Semi-Parametric Consistency

When the Parametric component updates are known with respect to the inertial parameters, we can determine the change in torque in any given state given the change in parameters. In the following, we demonstrate the linear transformation in the Non-Parametric function with respect to the inertial parameters.

$$\mathbf{M}(q)\ddot{q}+\mathbf{C}(q,\dot{q})\dot{q}+G(q)+F(q,\dot{q})+f(q,\dot{q},\ddot{q})=\tau \quad (10)$$

$$f(q,\dot{q},\ddot{q})=\tau-\mathbf{Y}(q,\dot{q},\ddot{q})\hat{\pi} \quad (11)$$

$$\frac{\partial f(q,\dot{q},\ddot{q})}{\partial \pi}=-\mathbf{Y}(q,\dot{q},\ddot{q}) \quad (12)$$

$$\Delta f(q,\dot{q},\ddot{q})=-\mathbf{Y}(q,\dot{q},\ddot{q})\Delta\hat{\pi} \quad (13)$$

Equation (11) is obtained through substituting (5) in (10). Then taking the derivative with respect to the inertial parameters ($\pi$) results in (12). When multiplying this by the change in inertial parameters ($\Delta\pi$) leads to (13). Using (13) we can approximately update each Gaussian in the GMM in their local space by linearly approximating the change in torque with respect to state and parameter update. In particular noting that each of the Gaussian sub-models is described through two main variables; the mean, and the covariance, which we can update with linearly approximated space transformations. Figure 2 gives a visual example of the transformation.

The mean is particularly straightforward to update using:

$$\tau_{\mu'}=\tau_{\mu}-\mathbf{Y}(q_\mu,\dot{q}_\mu,\ddot{q}_\mu)\Delta\hat{\pi} \quad (14)$$

Where $\bullet_\mu$ is the metric $\bullet$ obtained from the relevant dimensions of $\mu$ of the Gaussian and $\bullet'$ is the updated metric.

The covariance is not as straightforward as it lies across the relevant space that changes non-linearly. We should be able to make a rough assumption that the covariance matrix in this situation roughly is equivalent to (15) for the case of dynamics.

$$\Sigma=\begin{bmatrix} I_{n\times n} & 0 & 0 & \frac{\partial\tau}{\partial q} \\ 0 & I_{n\times n} & 0 & \frac{\partial\tau}{\partial\dot{q}} \\ 0 & 0 & I_{n\times n} & \frac{\partial\tau}{\partial\ddot{q}} \\ \frac{\partial\tau}{\partial q} & \frac{\partial\tau}{\partial\dot{q}} & \frac{\partial\tau}{\partial\ddot{q}} & I_{n\times n} \end{bmatrix} \quad (15)$$

Where $n$ is the number of degrees of freedom of the robot.

Using the fact the covariance should describe how those variables should change with respect to the other dimensions, the off-diagonal terms should be equal to the partial derivatives of those dimensions. In the context of this paper, we can omit the effect of the diagonal and off-diagonal terms on $q$, $\dot{q}$, $\ddot{q}$ as they should not change with respect to the inertial parameters. The important terms to consider are those on the off-diagonal terms involving the $\tau$.

Using (15) as an assumption of the covariance layout, the update to the covariance matrix is as follows:

$$\mathbf{T}=\begin{bmatrix} I_{n\times n} & 0 & 0 & -\frac{\partial\mathbf{Y}(q_\mu,\dot{q}_\mu,\ddot{q}_\mu)}{\partial q}\Delta\hat{\pi} \\ 0 & I_{n\times n} & 0 & -\frac{\partial\mathbf{Y}(q_\mu,\dot{q}_\mu,\ddot{q}_\mu)}{\partial\dot{q}}\Delta\hat{\pi} \\ 0 & 0 & I_{n\times n} & -\frac{\partial\mathbf{Y}(q_\mu,\dot{q}_\mu,\ddot{q}_\mu)}{\partial\ddot{q}}\Delta\hat{\pi} \\ 0 & 0 & 0 & I_{n\times n} \end{bmatrix}^T \quad (16)$$

The covariance matrix can be updated by this matrix by:

$$\Sigma'=\mathbf{T}\Sigma\mathbf{T}^T \quad (17)$$

Using these equations with a trained GMM model you can maintain the consistency between the two Semi-Parametric components, and freely update both online and simultaneously with a minimal conflict between the components or invalidating the previously learned GMM model.

It is important to note at this point that the mean update is exact, however, the covariance update is only approximate in nature due to the Gaussian sub-models being a first-order approximation to the data. This can mean that if the Gaussian approximates an area where the inertial parameters have a large affect the approximation may deteriorate with large changes of these parameters.

### E. Implementation

The implementation of this system was done using the OROCOS framework [14] which specializes in real-time execution. The dynamics matrices, regressors, and state derivatives of the regressors have been calculated using the Adaptive Robotics Dynamics Library (ARDL)[1], specializing in support for adaptive dynamics algorithms, using algorithms inspired by [12]. The velocity and acceleration of the joints were estimated through the use of a single-dimensional Kalman filter and PLL filter respectively [15].

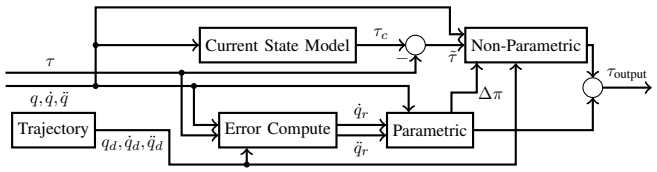



Figure 3: System Diagram

Figure 3 shows the control system in the entirety with each component implemented as described in section III with only the key input and outputs shown. It is key to note that the current model component is using the same model as the Parametric component and replicates the change in the inertial parameters.

[1] https://github.com/smithjoshua001/ARDL

## IV. Experimental Validation

To demonstrate the use of the consistency transformation we define two main experiments. The first will use a pre-trained baseline model [8] and a pre-trained GMM model trained solely on artificial dynamics model data. The second shall train the proposed model with the transformation online on a real platform.

### A. Virtual Model Experiment

To demonstrate the non-stationary effect, we shall use a virtual dynamics model of a Kuka LWR 4 manipulator, with simulated inertial parameters ($\pi$). This model will generate the torque data required to train both components and to provide the updated torque when the inertial parameters are changed. State data will be generated from seven pre-defined exciting trajectories appended to the end of each other. This state will contain $q$, $\dot{q}$, $\ddot{q}$, and $\tau$.

Initially, the Parametric component uses $0\pi$ for the inertial parameters, such that it produces zero torque. The baseline [8] and GMM model are then trained on the error as stated in (2) using every third data point. Effectively the models are trained to learn the entire robot dynamics.

The Parametric component is then updated to use $0.5\pi$ as inertial parameters. We then compare the output of the Parametric component plus the baseline/GMM model with the virtual dynamics model to get the normalized Mean Squared Error (nMSE) measure (18). The consistency update is then applied to the GMM model and the outputs compared again.

$$\text{nMSE}_\bullet = \frac{\frac{\sum(\bullet - \hat{\bullet})^2}{N}}{(\bullet_{\max} - \bullet_{\min})} \tag{18}$$

Where $\bullet_{\min}$ and $\bullet_{\max}$ are obtained as the maximum and minimum values of the desired position and velocity when $\bullet$ is $q$ or $\dot{q}$. When $\bullet$ is $\tau$ however, $\bullet_{\min}$ and $\bullet_{\max}$ are obtained as the maximum and minimum values of the recorded measured joint torque of the robot.

The metrics that will then be analysed will be the initial baseline and GMM torque prediction error on the full dynamics, and the Semi-Parametric torque prediction error with and without the transformed GMM and the baseline model with the new inputs. The trajectories that will be used will be several different Modified Fourier trajectories [16], a cyclic trajectory that guarantees a particular starting state.

### B. Real Platform Experiment

To demonstrate the online simultaneous Semi-Parametric controller we aim to show three key features. First, the Non-Parametric component can learn the desired function, online, starting with an incorrect Parametric component. Second, applying the transform without updating the Non-Parametric component does not significantly increase the tracking or torque errors. Third, to show that both components can be learned simultaneously with the same effect.

To do this we use a Kuka LWR IV to execute five repeating trajectories for 10 minutes. The following phases are designed to show the initial errors due to the initial incorrect model, followed by the previously defined features.

1) Starting from an incorrect Parametric component and an empty Non-Parametric component, the trajectory will be executed. The Non-Parametric component will be updated iteratively but not output to the control. The Parametric component does not update. (0-90 seconds)
2) The Non-Parametric component will then continue to be updated iteratively and will output the learned torque error. The Parametric component does not update. (90-180 seconds)
3) The Parametric component will then be updated iteratively with the Non-Parametric component being transformed accordingly but not updated. (180-360 seconds)
4) The Parametric component will continue to be updated. The Non-Parametric component will both be updated iteratively and transformed continuously. (360-600 seconds)

The phases will also be modified to show the effects of not applying the consistency transform. In particular, phase 3 will become identical to phase 4.

With the real robot, due to the lack of ground truth of the dynamics at the desired state, $(q_d, \dot{q}_d, \ddot{q}_d)$, we cannot directly use the torque as an performance metric. We use other metrics as indirect indicators of performance. In particular, we shall look at the state errors for position and velocity, which provide the feedback signal for the controller, and the error between the measured torque and the estimate of the current torque from the Semi-Parametric model at the robot's current state.

The exemplar trajectory, shown in figure 1, will be shown in a graph with and without the transform. The main evidence will be provided as nMSE of the performance metrics, calculated from the concatenated data from each phase of each trajectory. The desired outcome of this experiment should show minimal change to the normalized mean squared error (nMSE) or a reduced nMSE.

The parameters for the Parametric component are as follows: $\mathbf{R} = I_{n\times n}$, $\mathbf{P}$ is obtained through equations (16) and (17) in [2], with $\lambda_0 = 1.5$ and $k_0 = 0.1$. The error gains $\boldsymbol{\Lambda} = \text{diag}([20, 20, 20, 20, 10, 10, 10])$ and $\mathbf{K}_D = \text{diag}([5, 5, 5, 5, 2, 2, 2])$ from (6) and (8) in [2]. The initial $\Sigma$ for the IGMM algorithm was obtained through a subset of pre-collected data from the trajectory shown in figure 1.

## V. Results

### A. Transformation Validation

The results from the simulated experiment are shown in figure 4. The chart shows three main results; the errors for the baseline model [8], with 400 random features, and the GMM model initially trained on the full torque, the recall errors for both models when the inertial parameter update is applied, and the recall error for the GMM model transformed with the consistency transform.

The results show that the baseline method [8] and the GMM model without the consistency transform demonstrate the non-stationary effect increasing the nMSE in all joints. The GMM model with the consistency transform does show an increase in error from the original model. This is reduced compared to the baseline and GMM without the transform models. This shows that the transformation, whilst not perfect due to the

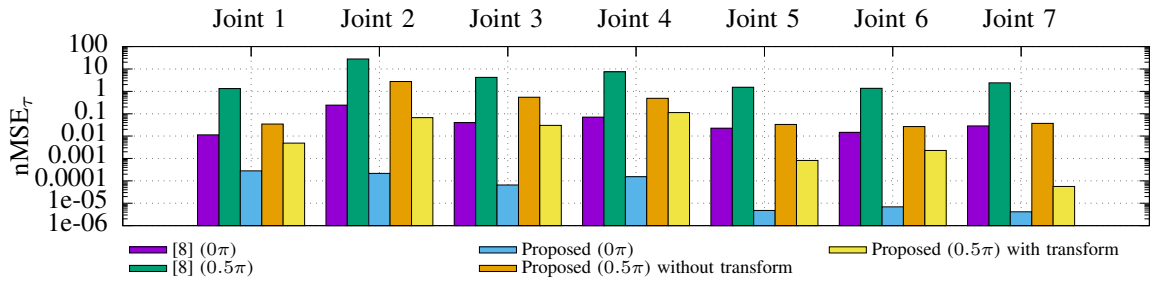



Figure 4: Virtual Model Adaptation - The random feature RRLS model [8], shows the non-stationary issue, as evident in the large increase in nMSE in all joints between the $0\pi$ and $0.5\pi$ models. The proposed GMM model demonstrates the same effect between the two models without consistency transform, however, when the consistency transform is applied the error is reduced. This is a strong indication that the model is kept more consistent with the update in inertial parameters.

approximate nature of the transform, adjusts the GMM to the change in parameters allowing the old data to remain more consistent.

### *B. Real Platform Experiment*

Figure 5 displays the nMSE results, along a log scale, when running without and with the consistency transform respectively.

In figure 5a we can see that during Phase 1, where the Non-Parametric component does not output, the errors are at the maximum they can be with respect to the high feedback gains. The torque nMSE in this phase is particularly high. During Phase 2 we can notice a drop in most of the metrics. The torque nMSE in Phase 2 again is quite high, especially in joint 2. Phases 3 and 4, which are equivalent for the no transformation scenario, show that nMSE increases almost back to the original nMSE from Phase 1.

From figure 5b, Phase 1 and 2 are very similar in scale to figure 5a. Phases 3 and 4 we can see that the nMSE metrics are of a similar or less order than in Phase 2. There is a small increase that can be noticed in some metrics from Phase 3 to 4, however, these are relatively small and are most likely the cause of the GMM having local sub-models that sub-optimally represent that space in the target function or due to numerical instability on the covariance. The lack of improvement in the nMSE measures does not mean that the models are inconsistent in this case as it can be that the components, given their initial parameters, have reached a minima.

The error behaviour is more evident with the trajectory shown in figure 1. The trajectory provides a specific run's evolution over the experiment, shown in figures 6 and 7. Figure 6, where the transform is not applied, shows that the learning appears to stall, with the sum of the two components being much greater than the output of the controller indicating high feedback torques. The deteriorating trajectory tracking also indicates the higher feedback torques needed to compensate for the incorrect model torque.

Figure 7, where the transform is applied, maintains the trajectory tracking performance whilst learning both components. Figure 7a shows clearly the transference behaviour between the two components. We can see that as the Parametric component gets updated iteratively, the torque contribution is increased. With the Non-Parametric component we can see a relative decrease in torque contribution of a similar magnitude as the increase in Parametric torque. This behaviour indicates that the components are being adapted with respect to each other so that their output is consistent, and the training biases the model towards the Parametric component.

An observation from the experiments is that the bias has the effect of simplifying the Non-Parametric target function, as evidenced by the reduced increase of sub-models in the GMM. Without the transform, we observed the GMM increased by eight sub-models on average, whereas with the transform the GMM increased by a single sub-model on average. This indicates that the GMM is kept consistent as fewer new sub-models are needed for the new target function.

## VI. Conclusion

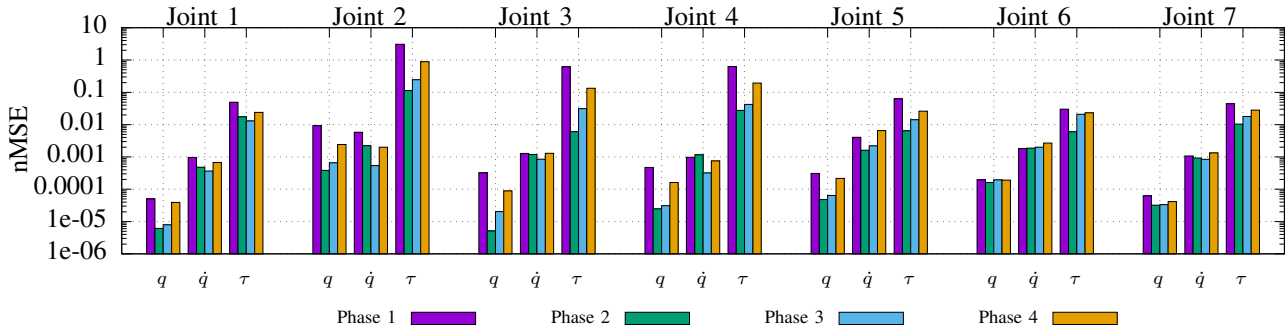
This paper addresses the non-stationary issue when updating both components of a Semi-Parametric model simultaneously and online. We have verified that a consistency transform of the Non-Parametric component with respect to changes in parameters can reduce errors caused by the non-stationary problem.

Overall, from sections V-A and V-B and figures 6 and 7, we show using the consistency transform we are able to maintain performance whilst adapting both components online without explicit retraining of the Non-Parametric component. The experiments also show that the Semi-Parametric model biases the learning towards the Parametric component, hence reducing the contribution of the Non-Parametric component. The bias in the Semi-Parametric model should allow greater ability in terms of generalization, as the Parametric component typically does a better job at generalizing over the state space. The reduced contribution of the Non-Parametric component will also reduce the effect of incorrect predictions as they should be relatively small compared to the Parametric component.

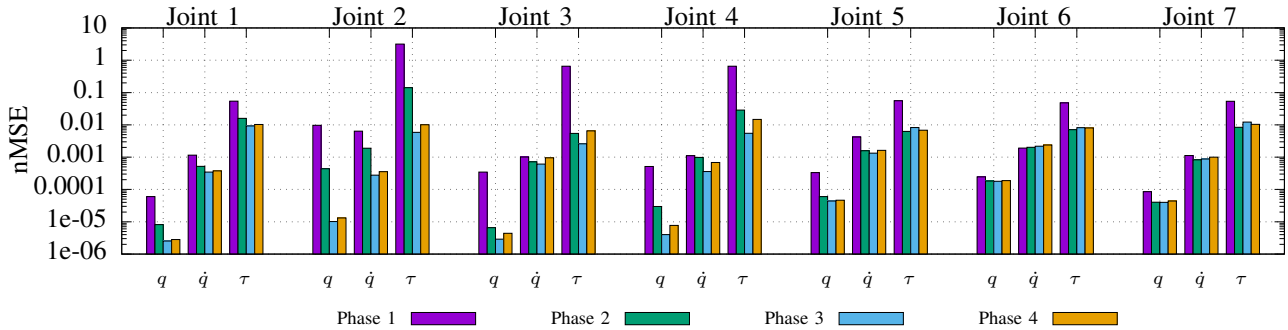
## VII. Discussion

In future we plan on expanding the analysis of the simplification of the Non-Parametric target function and the generalization capability of the model due to the bias effect towards the Parametric component. Which has been observed during the experiments in sections V-A and V-B.

The issues around the Gaussians, becoming singular or becoming numerically unstable, can be further explored as an extension. Possible solutions to explore are; adding robustness




(a) No Transform (logscale) - With no transform the nMSE appears to have a trend where Phase 2 learns the dynamics decreasing the nMSE error. Onwards though through Phase 3 and 4 the nMSE tends to increase in most joints with the final nMSE in all joints over all metrics being larger than in Phase 2. This indicates the two components conflicting.




(b) Consistency Transform (logscale) - The metrics show that the nMSE reduces from phase 2. After phase two the nMSE does not change by a large margin, especially when comparing both Phases 3 and 4 against Phase 2. This is indicative that the two components are being kept consistent with each other, maintaining better accuracy overall.

Figure 5: nMSE of metrics ($q$ (rad), $\dot{q}$ ($\mathrm{rad\,s^{-1}}$), $\tau$ ($\mathrm{N\,m}$))

to singularities, or regularization, to the GMM sub-models, adapting the idea of consistency transform to other models, such as neural networks that are better suited for pure regression problems. Another extension would be looking into the physical plausibility of the inertial parameters, as this is currently not enforced.

## REFERENCES

[1] C. G. Atkeson, C. H. An, and J. M. Hollerbach, "Estimation of Inertial Parameters of Manipulator Loads and Links," *The International Journal of Robotics Research*, vol. 5, no. 3, 1986.

[2] J.-J. E. Slotine and W. Li, "Composite adaptive control of robot manipulators," *Automatica*, vol. 25, no. 4, pp. 509–519, 7 1989.

[3] J.-A. Ting, M. Mistry, J. Peters, S. Schaal, and J. Nakanishi, "A Bayesian Approach to Nonlinear Parameter Identification for Rigid Body Dynamics," in *Robotics: Science and Systems II*, G. S. Sukhatme, S. Schaal, W. Burgard, and D. Fox, Eds. MIT Press, 2007.

[4] S. Vijayakumar and S. Schaal, "Locally Weighted Projection Regression: An O(n) Algorithm for Incremental Real Time Learning in High Dimensional Space," in *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, 2000, pp. 1079–1086.

[5] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Model Learning with Local Gaussian Process Regression," *Advanced Robotics*, vol. 23, no. 15, pp. 2015–2034, 1 2009.

[6] T. Cederborg, M. Li, A. Baranes, and P. Y. Oudeyer, "Incremental local online Gaussian Mixture Regression for imitation learning of multiple tasks," in *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*. IEEE, 10 2010, pp. 267–274.

[7] D. Nguyen-Tuong and J. Peters, "Using model knowledge for learning inverse dynamics," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 5 2010, pp. 2677–2682.

[8] R. Camoriano, S. Traversaro, L. Rosasco, G. Metta, and F. Nori, "Incremental semiparametric inverse dynamics learning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5 2016, pp. 544–550.

[9] D. Romeres, M. Zorzi, R. Camoriano, and A. Chiuso, "Online semi-parametric learning for inverse dynamics modeling," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 12 2016, pp. 2945–2950.

[10] S. Riedel and F. Stulp, "Comparing semi-parametric model learning algorithms for dynamic model estimation in robotics," *arXiv preprint arXiv:1906.11909*, 2019.

[11] D. Romeres, M. Zorzi, R. Camoriano, S. Traversaro, and A. Chiuso, "Derivative-Free Online Learning of Inverse Dynamics Models," *IEEE Transactions on Control Systems Technology*, pp. 1–15, 2019.

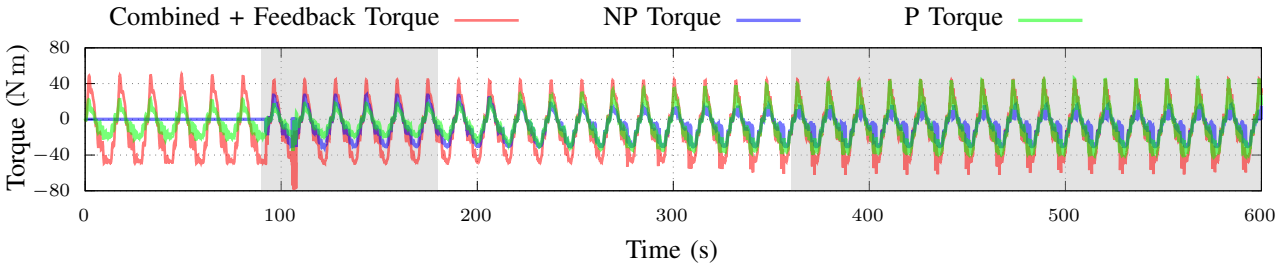
[12] G. Garofalo, C. Ott, A. Albu-Schäffer, and A. Albu-Schaffer, "On the closed form computation of the dynamic matrices and their differentiations," *IEEE International Conference on Intelligent Robots and Systems*, pp. 2364–2369, 11 2013.

[13] R. Pinto and P. Engel, "Scalable and incremental learning of gaussian mixture models," *arXiv preprint arXiv:1701.03940*, 2017.

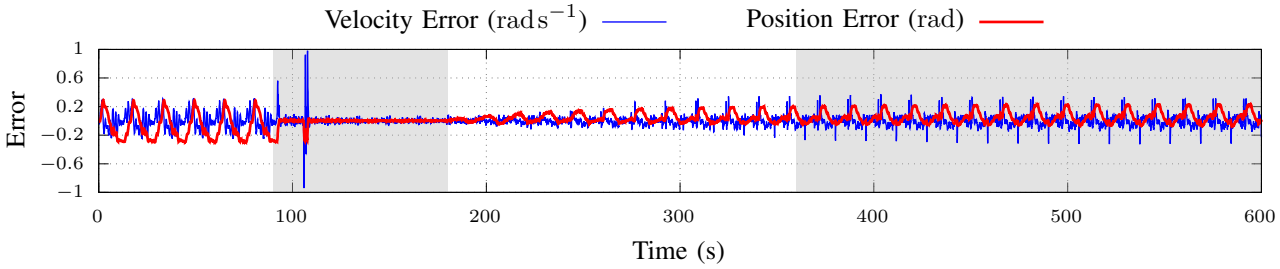
[14] H. Bruyninckx, P. Soetens, and B. Koninckx, "The real-time motion control core of the Orocos project," in *IEEE International Conference on Robotics and Automation*. IEEE, 2003, pp. 2766–2771.

[15] S. Wang and S. Wan, "Velocity and acceleration computations by single-dimensional Kalman filter with adaptive noise variance," *Przeglad Elektrotechniczny*, vol. 88, no. 2, pp. 283–287, 2012.

[16] K.-J. Park, "Fourier-based optimal excitation trajectories for the dynamic identification of robots," *Robotica*, vol. 24, no. 5, 9 2006.

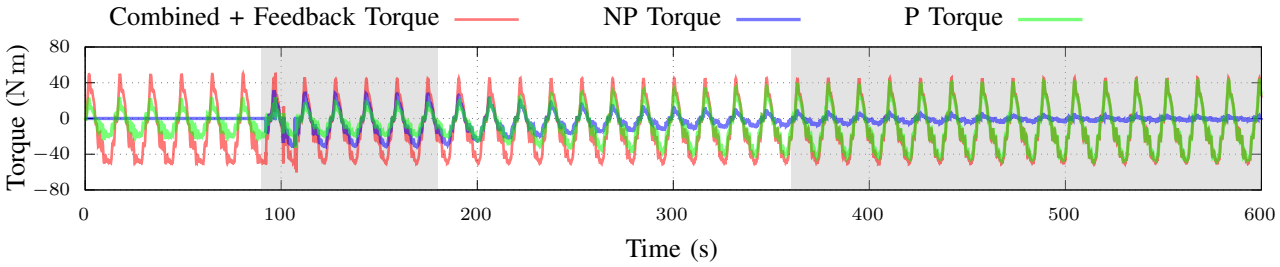



(a) Joint 2 - Torque Contributions - The contribution of both components to the combined output of the model with state feedback. With no consistency transform, we can see that the components stall in learning. The stall indicates that the change in the Parametric component is not taken into account in the Non-Parametric component. Phase changes indicated by background. Phase 1: No Parametric update, Non-Parametric updates but does not output. Phase 2: No Parametric update, Non-Parametric updates and outputs. Phase 3: Parametric updates, Non-Parametric updates but no transform is applied. Phase 4: Same as Phase 3.

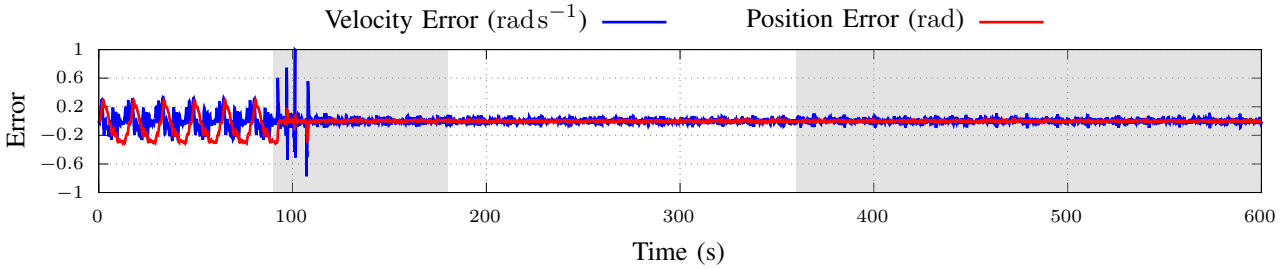



(b) Joint 2 - Position and Velocity Error - As the Parametric component continues to learn we see that the errors increase due to this inconsistency. Phase changes indicated by background.

Figure 6: Non-Consistent Transform Exemplar




(a) Joint 2 - Torque Contributions - The contribution of both components to the combined output with state feedback. We see that as the components adapt, the Parametric component increases and the Non-Parametric component equivalently decreases. Phase changes indicated by background. Phase 1: No Parametric update, Non-Parametric updates but does not output. Phase 2: No Parametric update, Non-Parametric updates and outputs. Phase 3: Parametric updates, Non-Parametric does not update but the consistency transform is applied. Phase 4: Parametric updates, Non-Parametric updates and the consistency transform are applied.




(b) Joint 2 - Position and Velocity Error - We determine that the errors during the learning do not change significantly when consistently transforming the model. Phase changes indicated by background.

Figure 7: Consistent Transform Exemplar