Towards Efficient Learning-Based Model Predictive Control via Feedback Linearization and Gaussian Process Regression

Jack Caldwell[1] and Joshua A. Marshall[1]

Abstract— **This paper presents a learning-based Model Predictive Control (MPC) methodology incorporating nonlinear predictions with robotics applications in mind. In particular, MPC is combined with feedback linearization for computational efficiency and Gaussian Process Regression (GPR) is used to model unknown system dynamics and nonlinearities. In this method, MPC predicts future states by leveraging a GPR model and optimizes a sequence of inputs over feedback linearized states. The controller was tested in simulation by using a two-link planar robot in the presence of model uncertainty. With respect to trajectory-tracking error, the proposed controller outperformed a conventional Proportional-Derivative Inverse Dynamics controller and a GPR-augmented version. Although a fully nonlinear MPC formulation achieved slightly better performance, the proposed controller had an average control calculation time that was $82\times$ faster.**

I. INTRODUCTION

Tracking control is a critical functionality for robotic applications. The overall performance and safety of such systems rely upon high-bandwidth low-error tracking control. One common approach in robot control is to feedback linearize the plant dynamics such that linear control techniques can be employed. The control parameters are then fed through an inverse dynamics model prior to actuation. However, this approach is especially susceptible to inaccuracies in model parameters.

Adaptive controllers, which attempt to also track a system's dynamic parameters, have been studied in detail but can fail to converge. More recent work attempts to augment robot dynamic models with learning-based models. These approaches can ensure more accurate system models without the need for extensive system modelling [1]. Gaussian Process Regression (GPR) is a promising machine learning technique that has been employed for learning-based controllers with positive results. This approach also enables the use of GPR covariance for robustness and uncertainty propagation; e.g., see [2] and references therein.

The goal of a robot control system is to track desired state trajectories. Future desired trajectories are often known in advance. However, popular linear control techniques (e.g., Proportional-Derivative using inverse dynamics models) consider only the current tracking errors. This approach fails to leverage information about future tracking objectives. More recent work has explored Model Predictive Control (MPC) techniques for robotic control. MPCs leverage future trajectories to encode foresight into the control law. This approach tends to reduce overall tracking error, but typically at the expense of computational effort.

This paper compares a set of learning-based controllers. System models are augmented by using GPR and linearization techniques are employed for computational efficiency. The resulting learning-based linearized MPC is more computationally efficient than a fully nonlinear one that requires an iterative solution, but continues to leverage the nonlinear model and GPR for state predictions. The developed controllers were simulated on a two degree of freedom (DOF) planar manipulator in the presence of model uncertainty.

This paper is organized as follows. Section II provides a review of relevant robot control research. Section III provides definitions and a classical inverse dynamics controller. Section IV outlines nonlinear and linear MPC methodologies, and describes the proposed hybrid linearized MPC with nonlinear predictions. Section V augments the controller nominal process models with GPR. Section VI applies the proposed controllers on a two DOF planar manipulator in simulation. Finally, Section VII summarizes the paper with a look at planned experiments on actual robot devices.

II. RELATED WORK

Model predictive control (MPC) optimizes a sequence of control inputs over a finite horizon, leveraging a model of the process to predict and minimize future errors. MPC was originally developed for power plants and petroleum refineries but has since expanded into many industries (e.g., chemical plants, automotive, food processing, aerospace, forestry) [3]. More recently, MPC algorithms have been tailored for robotic manipulator control. In [4], the dynamics of a SCARA manipulator were feedback linearized and MPC was employed. Results were compared to an inverse dynamics/computed torque controller and they were found to have similar tracking performance. Notably, the MPC was able to better reject disturbances acting on the system. However, the MPC algorithms used employed iterative optimization techniques, which imply a large computational burden.

In [5], a three-link planar manipulator mounted on a free-flying space robot was controlled via a nonlinear Model Predictive Controller (NMPC) in simulation. The prediction horizon was a single time-step and thus computational analysis is meaningless. The NMPC was found to outperform a sliding mode controller in simulation. However, model uncertainty is not considered and it follows that the NMPC would perform ideally. In [6], NMPC was applied to a

*This project was funded in part by the NSERC Canadian Robotics Network (NCRN) under grant NETGP 508451-17.

[1]Jack Caldwell and Joshua Marshall are with Department of Electrical & Computer Engineering and the Ingenuity Labs Research Institute, Queen's University, Kingston, ON K7L 3N6, Canada {jack.caldwell, joshua.marshall}@queensu.ca

hydraulic forestry crane with positive results, but a computational analysis is missing. In [7], NMPC was applied to a hydraulic bulldozer blade, but only to the first cylinder, thus reducing the dynamic complexity. Researchers in [8] proposed a feedback linearized MPC scheme as applied to an underactuated manipulator. The controller employed linear quadratic optimization on a linearized system, with nonlinear variable input constraints. These authors successfully showed MPC outperforming a PD inverse dynamics (PD-ID) controller. Notably, for their computing setup, they found the control updates computed at an average of 4.1 ms intervals.

One concern with MPC (and feedback linearization) is the reliance on an accurate system model. GPR is a widely used non-parametric Bayesian regression technique, well suited for nonlinear modeling. Moreover, GPR has been leveraged to estimate hydraulic manipulator inverse dynamics, and has been shown to outperform the rigid body mechanics (Euler-Lagrange) method on a SARCOS manipulator [9]. However, GPR is computationally expensive, with a time complexity of $\mathcal{O}(n^3)$, where n is the size of the training set. In [9], a variety of solutions are proposed to reduce the computational burden. In this paper, a simple approach was taken for GPR and such enhancements were left for future work.

GPR provides a variance estimate, which can be used for uncertainty evaluation [2], [8], [10]. In [2], GPR is used with feedback linearization to augment a linear quadratic regulator. Furthermore, these researchers managed to probabilistically guarantee an ultimate bound on the tracking error via a robustness term derived from the GPR variance. The work presented in this paper leverages the predictive mean provided via GPR for state estimation, but leaves robustness and uncertainty propagation as future work. The emphasis of this paper is a novel hybrid linearization of MPC and the use of GPR for predictive modelling.

Research has demonstrated that NMPC algorithms can be enhanced with GPR and other non-parametric models. In [1], NMPC is augmented with a GPR disturbance model in order to predict both environmental disturbances and vehicle nonlinearities. The resulting controller was applied to a mobile robot for path following. The learnt model was updated online at each trial in order to adapt to a more accurate estimate. The controller performed path-following well but required the use of iterative optimization for the implementation of NMPC. Others have also proposed the combination of MPC with feedback linearzation for improved computational efficiency [11], although we have not seen this method applied in robotics or combined with GPR.

III. PROBLEM BACKGROUND

The problem outlined in this section follows similar methods presented in [2], [12] and others. Consider a robot model given by a nonlinear control-affine system defined by

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{x}_2 \\ \mathbf{f}(\mathbf{x}) + \boldsymbol{\Gamma}(\mathbf{x})\mathbf{u} \end{bmatrix} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state, where n is assumed to be an even number, and $\mathbf{u} \in \mathbb{R}^m$ is the input of dimension $m = n/2$. These dynamics may be unknown, thus a nominal robot model's dynamics are given by

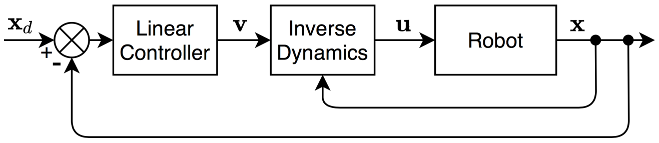


Fig. 1: Feedback linearization controller block diagram.

$$\dot{\mathbf{x}}_2 = \hat{\mathbf{f}}(\mathbf{x}) + \hat{\boldsymbol{\Gamma}}(\mathbf{x})\mathbf{u}. \tag{2}$$

A. Feedback Linearization

Assume the system is input-output feedback linearizable with respect to output $\mathbf{y} = \mathbf{h}(\mathbf{x}) = \mathbf{x}_1$. Thus, we may choose a new state $\mathbf{z}$ such that

$$\dot{\mathbf{z}} = \begin{bmatrix} \dot{\mathbf{y}} \\ \ddot{\mathbf{y}} \end{bmatrix} = \mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{v} = \begin{bmatrix} \mathbf{z}_2 \\ \mathbf{v} \end{bmatrix}, \tag{3}$$

which is linear when the transformed system's input $\mathbf{v}$ is defined via the appropriate nonlinear change of inputs [12]

$$\mathbf{u} = \hat{\boldsymbol{\Gamma}}^{-1}(\mathbf{x}) \left(\mathbf{v} - \hat{\mathbf{f}}(\mathbf{x}) \right). \tag{4}$$

See Fig. 1 for a block diagram.

B. Inverse Dynamics Control

One approach is to apply PD feedback to the linearized system (3) [10], [12]. In this case, the controller is

$$\mathbf{v} = \mathbf{K}_p\mathbf{e} + \mathbf{K}_d\dot{\mathbf{e}} + \ddot{\mathbf{y}}_d, \tag{5}$$

where $\mathbf{e} = \mathbf{z}_{1,d} - \mathbf{z}_1$ and $\dot{\mathbf{e}} = \dot{\mathbf{z}}_{1,d} - \dot{\mathbf{z}}_1$ are the state errors, and $\mathbf{K}_p \in \mathbb{R}^{m\times m}$ and $\mathbf{K}_d \in \mathbb{R}^{m\times m}$ are appropriate gain matrices. This leads to the error dynamics

$$\ddot{\mathbf{e}} = \ddot{\mathbf{z}}_{1,d} - \ddot{\mathbf{z}}_1 = -\mathbf{K}_p\mathbf{e} - \mathbf{K}_d\dot{\mathbf{e}} \tag{6}$$

and the gains can be designed to achieve desired behaviour.

IV. MODEL PREDICTIVE CONTROLLERS

In model predictive control (MPC), a sequence of control inputs is optimized with respect to a cost function over a finite prediction horizon of p time steps. The cost function can be tailored to the application and desired metrics. For all MPC formulations in this paper, the states are defined such that they track the control error, that is $\mathbf{z}_{1,k} = \mathbf{e}_k$ and $\mathbf{z}_{2,k} = \dot{\mathbf{e}}_k$. Furthermore, the state is extended to include an integral term, provided via $\boldsymbol{\zeta}_{k+1} = \boldsymbol{\zeta}_k + \mathbf{z}_{1,k}$.

A. Nonlinear Model Predictive Control (NMPC)

Consider MPC applied to a nonlinear robot where the cost function is defined as

$$J(\mathbf{U}) = \mathbf{Z}^T\mathbf{Q}\mathbf{Z} + \mathbf{U}^T\mathbf{R}\mathbf{U}, \tag{7}$$

where $\mathbf{Z}$ is a vector of predicted state errors

$$\mathbf{Z} = (\mathbf{z}_{k+1}, \mathbf{z}_{k+2}, \ldots, \mathbf{z}_{k+p}), \tag{8}$$

$\mathbf{U}$ is the sequence of controlled inputs

$$\mathbf{U} = (\mathbf{u}_k, \mathbf{u}_{k+1}, \ldots, \mathbf{u}_{k+p-1}), \tag{9}$$

$\mathbf{Q} \in \mathbb{R}^{np \times np}$ is a positive semi-definite matrix of weights on the state error, and $\mathbf{R} \in \mathbb{R}^{mp \times mp}$ is a positive definite matrix of weights on the system inputs.

The predicted states over the horizon are nonlinear functions of the inputs. A nonlinear discrete-time model describing the system dynamics is given by

$$\mathbf{z}_{k+1} = \mathbf{\Phi}(\mathbf{z}_k, \mathbf{u}_k). \tag{10}$$

Similarly, a nominal nonlinear discrete-time model leveraged for MPC state predictions is assumed to be

$$\mathbf{z}_{k+1} = \hat{\mathbf{\Phi}}(\mathbf{z}_k, \mathbf{u}_k). \tag{11}$$

Because the robot dynamics are nonlinear, solving for the optimal control inputs subject to the cost function (7) is a nonlinear optimization problem. One approach is to solve this iteratively. In this case, we use a method similar to [1] where the system is locally linearized about an operating point as $\mathbf{z} = \bar{\mathbf{z}} + \delta\mathbf{z}$ and $\mathbf{u} = \bar{\mathbf{u}} + \delta\mathbf{u}$, where

$$\bar{\mathbf{z}}_{k+b+1} = \hat{\mathbf{\Phi}}(\bar{\mathbf{z}}_{k+b}, \bar{\mathbf{u}}_{k+b}) \tag{12}$$
$$\delta\mathbf{z}_{k+b+1} \approx \mathbf{H}_{\mathbf{z},k+b}\delta\mathbf{z}_{z+b} + \mathbf{H}_{\mathbf{u},k+b}\delta\mathbf{u}_{k+b}, \tag{13}$$

where $\mathbf{H}_{\mathbf{z},k+b}$ and $\mathbf{H}_{\mathbf{u},k+b}$ are the system Jacobians.

Substituting the local linearization into (7) and minimizing with respect to the change in input yields

$$\delta\mathbf{U}^T = (\mathbf{H}^{'T}\mathbf{Q}\mathbf{H}^{'} + \mathbf{R})^{-1}(-\mathbf{Z}^T\mathbf{Q}\mathbf{H}^{'} - \bar{\mathbf{U}}^T\mathbf{R}), \tag{14}$$

where $\mathbf{H}' = (\mathbf{1} - \mathbf{H}_\mathbf{z})^{-1}\mathbf{H}_\mathbf{u}$. This is repeated until convergence of the control sequence, for each timestep k.

B. Linear Model Predictive Control

We now consider the application of MPC to the feedback linearized system. First, the continuous-time linear state space model (3) must be discretized as

$$\mathbf{z}_{k+1} = \mathbf{F}\mathbf{z}_k + \mathbf{G}\mathbf{v}_k. \tag{15}$$

The new input is $\mathbf{v}_k$, so the cost function is re-defined as

$$J(\mathbf{V}) = \mathbf{Z}^T\mathbf{Q}\mathbf{Z} + \mathbf{V}^T\mathbf{R}\mathbf{V}, \tag{16}$$

where $\mathbf{V}$ is the sequence of controlled inputs

$$\mathbf{V} = (\mathbf{v}_k, \mathbf{v}_{k+1}, \ldots, \mathbf{v}_{k+p-1}). \tag{17}$$

Note that our optimization is now over the feedback linearized states. The predicted future states for (15) become

$$\mathbf{Z} = \mathbf{L}\mathbf{z}_k + \mathbf{M}\mathbf{V}, \tag{18}$$

where $\mathbf{z}_k$ is the current state, $\mathbf{L} = \begin{bmatrix}\mathbf{F} & \mathbf{F}^2 & \ldots & \mathbf{F}^p\end{bmatrix}^T$ and

$$\mathbf{M} = \begin{bmatrix} \mathbf{G} & 0 & \ldots & 0 \\ \mathbf{F}\mathbf{G} & \mathbf{G} & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{F}^{p-1}\mathbf{G} & \mathbf{F}^{p-2}\mathbf{G} & \ldots & 0 \end{bmatrix}. \tag{19}$$

Substituting (18) into (16), and minimizing for the optimal sequence of inputs

$$\mathbf{V}^* = \mathbf{K}_{\text{MPC}}\mathbf{M}^T\mathbf{Q}\mathbf{L}\mathbf{z}_k, \tag{20}$$

where $\mathbf{K}_{\text{MPC}} = -(\mathbf{M}^T\mathbf{Q}\mathbf{M} + \mathbf{R})^{-1}$ is a constant that can be pre-computed. The first of these inputs is then converted to actuator space via (4) and applied to the robot.

C. Hybrid Model Predictive Control

As can be seen in the state predictions (18), the above linear MPC formulation disregards the system's nonlinear dynamics. Thus, we propose a hybrid MPC solution. First, we re-define the states as

$$\mathbf{z}_{k+1} = \Delta\mathbf{z}_{k+1} + \mathbf{z}_k. \tag{21}$$

Thus, the sequence of predicted future states can now be defined as $\mathbf{Z}_{k+1} = \mathbf{Z}_k + \Delta\mathbf{Z}_{k+1}$, where

$$\mathbf{Z}_k = (\mathbf{z}_k, \mathbf{z}_{k+1}, \ldots, \mathbf{z}_{k+p-1}) \tag{22}$$
$$\Delta\mathbf{Z}_{k+1} = (\Delta\mathbf{z}_{k+1}, \Delta\mathbf{z}_{k+2}, \ldots, \Delta\mathbf{z}_{k+p}). \tag{23}$$

The inputs are also re-defined as

$$\mathbf{v}_{k+1} = \Delta\mathbf{v}_{k+1} + \mathbf{v}_k. \tag{24}$$

Then changes in states are provided by the linear model as

$$\Delta\mathbf{z}_{k+1} = \mathbf{F}\Delta\mathbf{z}_k + \mathbf{G}\Delta\mathbf{v}_k, \tag{25}$$

which together yield

$$\Delta\mathbf{Z} = \mathbf{L}\Delta\mathbf{z}_k + \mathbf{M}\Delta\mathbf{V}, \tag{26}$$

where $\mathbf{L}$ and $\mathbf{M}$ are as defined above. The nonlinear discrete-time model (11) is used to predict $\mathbf{Z}_k$, where $\mathbf{u}_k$ is calculated via the nonlinear mapping (4).

Substituting the above definitions into (16) and minimizing, the optimal change in input is

$$\Delta\mathbf{V}_k^* = \mathbf{K}_{\text{MPC}}(\mathbf{M}^T\mathbf{Q}(\mathbf{Z}_k + \mathbf{L}\Delta\mathbf{z}_k) + \mathbf{R}\mathbf{V}_{k-1}^*), \tag{27}$$

where $\mathbf{V}_{k-1}^*$ is the optimal control sequence from the previous time step. Finally, the current optimal control sequence is given by

$$\mathbf{V}_k^* = \Delta\mathbf{V}_k^* + \mathbf{V}_{k-1}^*. \tag{28}$$

The control system block diagram is depicted in Fig. 2, and pseudo-code in Algorithm 1.

Algorithm 1 Simulation of proposed hybrid linear MPC with nonlinear predictions

```
1: $\mathbf{V}_{k-1}^* \leftarrow \mathbf{0}$
2: for $k \leftarrow 1$ to $N$ do
3:     $\Delta\mathbf{z}_k \leftarrow \mathbf{z}_k - \mathbf{z}_{k-1}$
4:     $\mathbf{Z}_k(1) \leftarrow \mathbf{z}_k$ {Populate current state}
5:     for $i \leftarrow 1$ to $p$ do
6:         $\mathbf{u} \leftarrow \frac{\mathbf{V}_{k-1}^*(i+1) - \hat{\mathbf{f}}(\mathbf{x}_{k+i})}{\hat{\mathbf{g}}(\mathbf{x}_{k+i})}$
7:         $\mathbf{Z}_k(i+1) \leftarrow \hat{\mathbf{\Phi}}(\mathbf{Z}_k(i), \mathbf{u})$ {Predict future states}
8:     end for
9:     $\Delta\mathbf{V}_k^* \leftarrow \mathbf{K}_{\text{MPC}}(\mathbf{M}^T\mathbf{Q}(\mathbf{Z}_k + \mathbf{L}\Delta\mathbf{z}_k) + \mathbf{R}\mathbf{V}_{k-1}^*)$
10:    $\mathbf{V}_k^* \leftarrow \mathbf{V}_{k-1}^* + \Delta\mathbf{V}_k^*$ {Update inputs}
11:    $\mathbf{u}_k \leftarrow \frac{\mathbf{V}_k^*(1) - \hat{\mathbf{f}}(\mathbf{x}_k)}{\hat{\mathbf{g}}(\mathbf{x}_k)}$ {Convert input}
12:    $\mathbf{z}_{k+1} \leftarrow \mathbf{\Phi}(\mathbf{z}_k, \mathbf{u}_k)$ {Simulate system}
13:    $\mathbf{V}_{k-1}^* \leftarrow \mathbf{V}_k^*$
14: end for
```

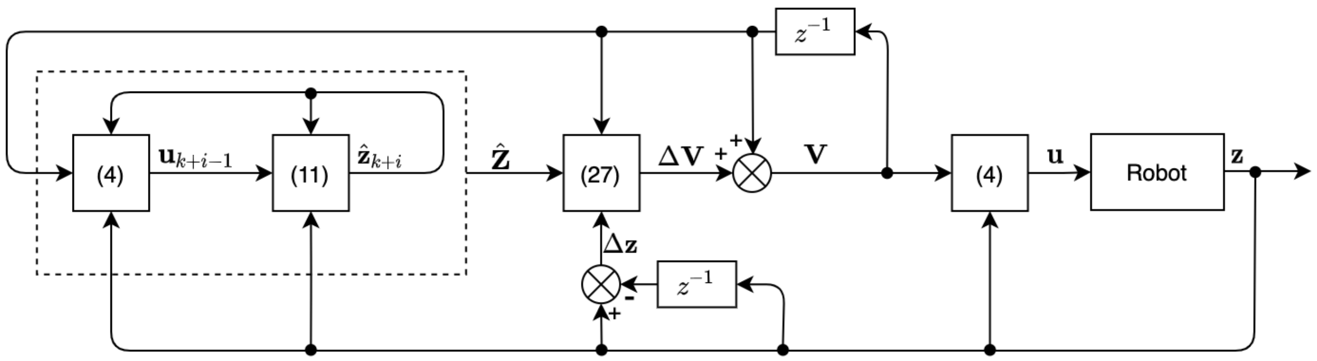


Fig. 2: Feedback linearized MPC with nonlinear predictions block diagram, with reference to equations.

V. Gaussian Process Regression

Gaussian Process Regression (GPR) is a kernel machine learning method that can be used to approximate a system by a Gaussian Process (GP) [9]. GPR finds

$$\bar{\mathbf{f}}_* \triangleq \mathbf{E}[\bar{\mathbf{f}}_* \mid \mathbf{X}, \mathbf{y}, \mathbf{X}_*] = \mathbf{K}_*^T[\mathbf{K} + \sigma_y^2\mathbf{I}]^{-1}\mathbf{y} \tag{29}$$

$$\mathrm{cov}(\mathbf{f}_*) = \mathbf{K}_{**} - \mathbf{K}_*^T[\mathbf{K} + \sigma_y^2\mathbf{I}]^{-1}\mathbf{K}_*, \tag{30}$$

where $\mathbf{X}$ is the training set, $\mathbf{X}_*$ is the testing set, $\mathbf{K}$ is the kernel function of the GPR evaluated at the training points, $\bar{\mathbf{f}}_*$ is the posterior mean, $\mathbf{K}_*$ is the covariance between the training and test points, $\mathbf{y}$ is the observed output, and σ_y^2 is the observation noise. The kernel augments the inputs into a new feature space, and is used in the GPR as the *a priori* (cov) function. One common kernel is the squared exponential kernel, or radial basis function,

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j)\right) \tag{31}$$

where σ_f^2 and l are the kernel hyperparameters. This kernel has the convenient property that it is continuously differentiable. This is important for calculations of the NMPC process Jacobians.

Training of GP models involves the tuning of the hyperparameters of the kernel function. This is done by maximizing the log of the marginal likelihood, given as

$$\begin{aligned}\log p(\mathbf{y}|X) = &-\frac{1}{2}\mathbf{y}^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y} \qquad (32)\\ &-\frac{1}{2}\log\det(\mathbf{K} + \sigma_n^2\mathbf{I}) - \frac{n}{2}\log 2\pi.\end{aligned}$$

Once the model has been trained, we simply evaluate equations (29) and (30) to acquire the predictions [9].

In practice, estimated nominal inverse dynamics (4) and forward dynamics (11) models are used. We refer to the model error between the estimated nominal model and the actual dynamics as *nonlinear mismatch*, like in [2].

A. GP Inverse Dynamics

We define the *inverse nonlinear mismatch* as

$$\boldsymbol{\eta}_{ID}(\mathbf{z}_k, \mathbf{v}_k) = \ddot{\mathbf{y}}_{k+1} - \mathbf{v}_k, \tag{33}$$

where $\boldsymbol{\eta}_{ID}(\mathbf{z}_k, \mathbf{v}_k)$ is the dynamic error. A GPR model can be trained on sampled data to predict $\boldsymbol{\eta}_{ID}(\mathbf{z}_k, \mathbf{v}_k)$ [10]. The new control parameter is given by

$$\mathbf{a}_k = \mathbf{v}_k - \hat{\boldsymbol{\eta}}_{ID}(\mathbf{z}_k, \mathbf{v}_k). \tag{34}$$

B. GP Nonlinear Model Predictive Control

NMPC requires nonlinear models for both the state predictions as well as the cost function optimization. The state forward dynamics prediction *forward nonlinear mismatch* is

$$\boldsymbol{\eta}_{FD}(\mathbf{z}_k, \mathbf{u}_k) = \mathbf{z}_{k+1} - \hat{\boldsymbol{\Phi}}(\mathbf{z}_k, \mathbf{u}_k). \tag{35}$$

In this case, a GP is trained on sampled data to predict the state dynamic prediction error, $\boldsymbol{\eta}_{FD}(\mathbf{z}_k, \mathbf{u}_k)$. Next, the predicted state dynamics are augmented as

$$\hat{\mathbf{z}}_{k+1} = \underbrace{\hat{\boldsymbol{\Phi}}(\mathbf{z}_k, \mathbf{u}_k)}_{a\ priori} + \underbrace{\hat{\boldsymbol{\eta}}_{FD}(\mathbf{z}_k, \mathbf{u}_k)}_{learnt\ disturbance}. \tag{36}$$

The new Jacobians are

$$\mathbf{H}_{\mathbf{u},k+b} = \left.\frac{\delta\hat{\boldsymbol{\Phi}}(\cdot)}{\delta\mathbf{u}}\right|_{\bar{\mathbf{z}}_{k+b},\bar{\mathbf{u}}_{k+b}} + \left.\frac{\delta\hat{\boldsymbol{\eta}}_{FD}(\cdot)}{\delta\mathbf{u}}\right|_{\bar{\mathbf{z}}_{k+b},\bar{\mathbf{u}}_{k+b}} \tag{37}$$

$$\mathbf{H}_{\mathbf{z},k+b} = \left.\frac{\delta\hat{\boldsymbol{\Phi}}(\cdot)}{\delta\mathbf{z}}\right|_{\bar{\mathbf{z}}_{k+b},\bar{\mathbf{u}}_{k+b}} + \left.\frac{\delta\hat{\boldsymbol{\eta}}_{FD}(\cdot)}{\delta\mathbf{z}}\right|_{\bar{\mathbf{z}}_{k+b},\bar{\mathbf{u}}_{k+b}}. \tag{38}$$

C. GP Hybrid Model Predictive Control

Finally, the feedback linearized MPC with nonlinear state predictions proposed in this paper combines both forward and inverse nonlinear dynamics models in the control law. As such, the problem can be solved by two GPs: one involving the prediction (forward dynamics) and the other a conversion into actuator space (inverse dynamics).

The forward dynamics step is identical to the NMPC — i.e., the predicted states are augmented via (36). The inverse dynamics step is identical to the simpler inverse dynamic controller — i.e. the controlled input is augmented via (34).

VI. Simulation Example

The proposed controllers were verified in simulation using a two DOF planar manipulator, as illustrated in Fig. 3. The manipulator dynamics have the form

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\dot{\mathbf{q}}) \tag{39}$$

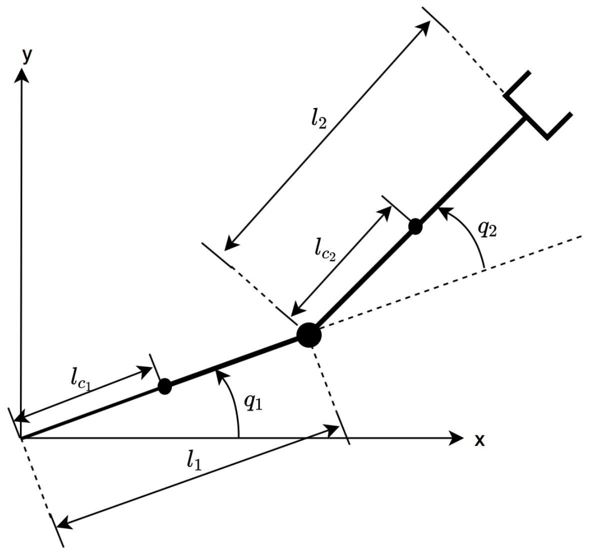


Fig. 3: Two link manipulator used in example simulations.

where $\boldsymbol{\tau} = (\tau_1, \tau_2)$ are the joint torques, $\mathbf{M}(\mathbf{q})$ is the mass matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the Coriolis and centrifugal vector, $\mathbf{G}(\mathbf{q})$ is the gravity vector, and $\mathbf{F}(\dot{\mathbf{q}})$ is the friction vector. The state was defined as $\mathbf{z} = (q_1, q_2, \dot{q}_1, \dot{q}_2)$ for the PD inverse dynamics (PD-ID) controllers and $\mathbf{z} = (q_1, q_2, \dot{q}_1, \dot{q}_2, \zeta_1, \zeta_2)$ for the MPC designs. The linear control parameter was $\mathbf{v} = (\ddot{q}_1, \ddot{q}_2)$, and the system inputs were $\mathbf{u} = \boldsymbol{\tau}$. The parameter values are provided in Table I, and were chosen to resemble the first two joints of a *Franka Emika Panda* manipulator.

TABLE I: Simulation dynamic parameters.

Dynamic Parameter	Symbols	Value
Link Mass	m_1, m_2	2.75 kg
Link Length	l_1, l_2	0.4 m
Link Center of Mass	l_{c_1}, l_{c_2}	0.2 m
Link Inertia	I_1, I_2	0.15 kgm^2
Joint Friction	b_1, b_2	1

To generate realistic results from simulation, a level of model mismatch was introduced. It was determined that a likely real system would involve nonlinear model parameter mismatch. To accomplish this, three simulation trials were run with 10%, 20%, and 30% model mismatch applied to all dynamic parameters prior to any dynamics calculations performed via the controller nominal process models.

The desired joint positions were given by

$$q_{1,d} = \tfrac{\pi}{3} \sin(kT) \tag{40}$$

$$q_{2,d} = \tfrac{\pi}{6} \sin(kT), \tag{41}$$

where kT is the simulation time.

As is commonly done [12], the PD-ID controller gains and bandwidth were designed as $\mathbf{K}_p = \text{diag}(\omega_n^2, \omega_n^2)$ and $\mathbf{K}_d = \text{diag}(2\zeta\omega_n, 2\zeta\omega_n)$ where $\omega_n = 3$ rad/s and $\zeta = 1$. The MPC and NMPC costs were manually optimized on the simulated system. The MPC cost matrices were $\mathbf{Q} = \text{diag}(1, 1, 4, 4, 3, 3)$ and $\mathbf{R} = \text{diag}(0.002, 0.002)$. The NMPC costs were $\mathbf{Q} = \text{diag}(5000, 5000, 20, 20, 0.5, 0.5)$ and $\mathbf{R} = \text{diag}(0.1, 0.1)$. Both predictive controllers were tested with $p = 10$ for the prediction horizons.

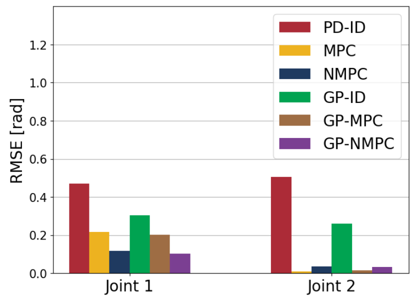
TABLE II: Average control calculation time accross all three 10-second controller simulations.

Controller	Time (ms)
PD-ID	0.049
MPC	1.3
NMPC	52
GP-ID	1.4
GP-MPC	27
GP-NMPC	2200

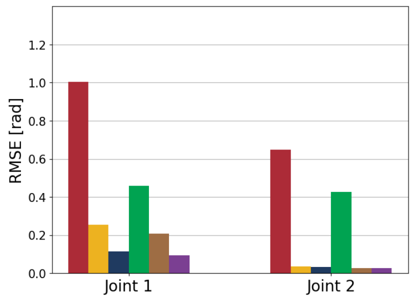
The *GPy Python* library was used for GPR. An open-loop simulation was performed wherein a series of input frequencies were excited. The output from this trial was used to train the GP models leveraged for controller testing. Four separate GPs were trained — one for the forward dynamics of each joint of the manipulator, and one for the inverse dynamics of each joint. The open-loop simulation provided 1000 data points for training. It should be noted that, in order to improve performance, it is possible to train the GPR hyperparameters offline by using a large dataset. Following this, the GPR predictions can be calculated using a smaller dataset acquired online [10]. This allows for quicker online GPR predictions and some adaptation to changing dynamic relationships (e.g., variable load).

All six controllers were tested on three 10-second simulations in *Python* 3.7. The computer used an 8x*Intel* Core i7-4720HQ CPU (2.6 GHz 16 GB DDR3). The initial joint positions were $q_1 = \frac{\pi}{4}$ rad and $q_2 = 0$ rad, thus forcing an initial error of $\mathbf{e}_1 = -\frac{\pi}{4}$ rad and $\mathbf{e}_2 = 0$ rad. The root-mean-squared errors (RMSE) were calculated for each joint, for each trial. Additionally, the average control computation time was monitored for each controller. The control calculation times are insubstantial in isolation (as they rely on the computing capacity), but allow for relational analysis because the calculations were performed on the same device. The RMSE from the 10-second trials are summarized in Fig. 4, and the calculation times in Table II.

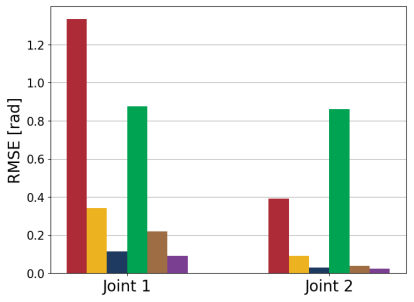
As is expected, the NMPC and GP-NMPC outperform all controllers with respect to RMSE. NMPC makes the fewest assumptions and, as such, it is expected to track with the least error. However, NMPC also performs the worst with respect to the average control computation time, by far. This work used unconstrained Gauss-Newton minimization to optimize the NMPC cost function. The associated computational burden is dependent on convergence, but due to the Jacobian calculations the NMPC requires at a minimum more computation than the hybrid MPC. It should be noted that there are alternate optimization methods to change the computational burden [13]. Shorter prediction horizons would reduce the computational burden, but would affect both controllers similarly. The proposed hybrid linearized MPC and GP-MPC track only marginally worse than the



(a) 10% nominal model mismatch



(b) 20% nominal model mismatch



(c) 30% nominal model mismatch

Fig. 4: Resulting root-mean squared error (RMSE) in rad for 10-second simulations with different nominal model mismatch under PD-ID control (proportional-derivative inverse dynamics), MPC (hybrid linearized model predictive control with nonlinear predictions), NMPC (nonlinear model predictive control), GP-ID (proportional-derivative inverse dynamics controller with GPR), GP-MPC (hybrid MPC with GPR), and GP-NMPC (NMPC with GPR).

NMPC but with significantly improved control computation time. Furthermore, adding GPR to the more accurate controllers (NMPC and MPC) lead to marginal tracking improvements, with substantial computation time penalties. These computing drawbacks might be limited by reducing the size of the GPR dataset employed for the prediction step. Future work will explore these trade-offs through physical experimentation.

VII. CONCLUSION

In summary, we have presented a generalized, efficient, learning-based, linearized MPC with the ability to reconcile unmodelled system dynamics and nonlinearities. A GPR model is trained to predict model inaccuracies. The GPR is used to augment state predictions leveraged in a feedback linearized MPC. The resulting controller successfully employs nonlinear predictions, while using only a linear optimization. The controller was verified against a PD-ID and GPR inverse dynamics controller in simulation on a two DOF manipulator in the presence of model uncertainty. Preliminary computational analysis shows dramatic improvements in average control computation time over NMPC.

In future work we plan to explore further GPR computational improvements, and the prospect of online data sampling for adaptive learning. Furthermore, we believe that robustness or uncertainty propagation can be added to this MPC scheme in a similar manner to that proposed in [2], [10], [14]. Finally, the proposed controller will be validated on an industrial-scale hydraulic manipulator.

REFERENCES

[1] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments," in *2014 IEEE International Conference on Robotics and Automation*, 2014, pp. 4029–4036.

[2] M. Greeff and A. P. Schoellig, "Exploiting differential flatness for robust learning-based tracking control using gaussian processes," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1121–1126, 2021.

[3] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, 2003.

[4] P. Poignet and M. Gautier, "Nonlinear model predictive control of a robot manipulator," *6th International Workshop on Advanced Motion Control*, no. 2, pp. 401–406, 2000.

[5] S. Kayastha, L. Shi, J. Katupitiya, and G. Pearce, "Nonlinear model predictive control of a planar three-link space manipulator," in *2017 Asian Control Conference*. IEEE, 2017, pp. 635–640.

[6] J. Kalmari, J. Backman, and A. Visala, "Nonlinear model predictive control of hydraulic forestry crane with automatic sway damping," *Computers and Electronics in Agriculture*, vol. 109, pp. 36–45, 2014.

[7] T. Mononen, M. M. Aref, and J. Mattila, "Nonlinear Model Predictive Control of a Heavy-Duty Hydraulic Bulldozer Blade," in *IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and Robotics, Automation and Mechatronics*. Bangkok, Thailand: IEEE, 2019.

[8] F. Schnelle and P. Eberhard, "Constraint mapping in a feedback linearization/MPC scheme for trajectory tracking of underactuated multibody systems," in *International Federation of Automatic Control (IFAC)*, vol. 48, no. 23. Elsevier Ltd., 2015, pp. 446–451.

[9] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: the MIT Press, 2006.

[10] M. K. Helwa, A. Heins, and A. P. Schoellig, "Provably robust learning-based approach for high-accuracy tracking control of Lagrangian systems," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1587–1594, 2019.

[11] J. Primbs and V. Nevistic, "MPC extensions to feedback linearizable systems," in *Proceedings of the 1997 American Control Conference*. IEEE, 1997, pp. 2073–2077.

[12] M. W. Spong and M. Vidyasagar, *Robot Dynamics and Control*. Wiley, 1989.

[13] M. Diehl, H. J. Ferreau, and N. Haverbeke, *Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 391–417.

[14] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *Conference on Decision and Control*. IEEE, 2018, pp. 6059–6066.