Model Predictive Control in the Legendre Domain

Graeme Damberger[1] and Chris Eliasmith[2]

***Abstract*— We present a reformulation of the model predictive control problem using a Legendre basis. To do so, we use a Legendre representation both for prediction and optimization. For prediction, we use a neural network to approximate the dynamics by mapping a compressed Legendre representation of the control trajectory and initial conditions to the corresponding compressed state trajectory. We then reformulate the optimization problem in the Legendre domain and demonstrate methods for including optimization constraints. We present simulation results demonstrating that our implementation provides a speedup of 31-40 times for comparable or lower tracking errors with or without constraints on a benchmark task.**

I. INTRODUCTION

Model predictive control (MPC) is a control algorithm that recurrently solves the optimal control problem for state and control trajectories over a prediction window given a set of constraints. The algorithm assumes an accurate model of the plant is available and uses it to predict trajectories. The optimal predicted trajectory is chosen based on a predefined cost function and applied to the plant. The strength of MPC results from this process being rapidly repeated, so the most recent control signal considers a fixed horizon of future points while obeying a set of constraints. Consequently, MPC optimizes over future time, as do other optimal control methods, but MPC uses a receding horizon over a fixed prediction window rather than an infinite horizon.

MPC has been applied extensively in complex non-linear applications with strict constraints, from autonomous vehicles [1] to power electronics [2]. However, like all control algorithms, MPC is not without its challenges. For one, the quality of the MPC is highly dependent on predicting future dynamics with respect to an initial condition and a control trajectory. Modelling errors in such predictions can result in significant performance degradation through the prediction window [3]. As well, MPC is computationally intensive, so the optimization process can severely limit the achievable quality of the prediction. The computational costs can be alleviated through linearization to find a closed-form solution. However, given highly non-linear dynamics found in many applications, performance will severely degrade as linear estimates may not be relevant throughout the whole prediction window.

The base MPC algorithm optimizes the control signal u at each sample point by solving the following optimization problem over the prediction window:

$$\underset{u}{\operatorname{argmin}} \sum_{k=0}^{N_p-1} V(x_k, u_k) \tag{1}$$

$$\begin{aligned} \text{Subject to: } x_{k+1} &= f(x_k, u_k) \\ x_{k=0} &= x_0 \\ S(x_k, u_k) &\leq 0, \end{aligned}$$

where V is a cost function typically in the form:

$$V(x_k, u_k) = (x_k - x_{r,k})^T Q (x_k - x_{r,k}) + u_k^T R u_k, \tag{2}$$

with weight matrices Q and R. Additionally, k indexes the sample time, x_0 is the initial condition of the state at the given sample time, $S(x_k, u_k)$ is the set of constraints acting on the control and state trajectories, and $x_{r,k}$ is the reference trajectory. The dynamics of the system are assumed to be non-linear and as such are denoted by $x_{k+1} = f(x_k, u_k)$. Here model predictive control is formulated in the context of a discrete time implementation of sample points k. Upon solving (2), the optimal control trajectory u is applied to the system until the next sample point in time. As the sampling step size decreases, the formulation approaches a continuous closed loop where the initial conditions x_0 are frequently updated and consequently a new optimal control trajectory u is applied often. This formulation highlights the challenges discussed previously, where: 1) an accurate model of the dynamics $x_{k+1} = f(x_k, u_k)$ is necessary; and 2) the sampling time is limited by the compute time of the optimization.

Towards efficiency in MPC, [4] introduced pseudo-spectral methods in MPC, discretizing the problem according to weighted quadrature points [5] in the time domain before applying a polynomial expansion. [6] enhanced the accuracy of these methods while retaining computational efficiency by discretizing over Chebyshev-Gauss-Lobatto points. To account for uncertainties, [7] utilized pseudo-spectral methods in stochastic optimal control problems. Furthermore, towards robustness and reliability, [8] proposed an approach to generate safety envelopes of the generated trajectory. Despite their efficiency, pseudo-spectral techniques remain entirely model-dependent and can suffer in scaling, as higher-order approximations can lead to dense differentiation matrices with high computational complexity. However, despite these limitations, pseudo-spectral methods provide a computationally efficient alternative to traditional MPC and remain relevant in modern applications [9], [10].

In a similar fashion to pseudo-spectral techniques, [11] projected the prediction window over a set of Laguerre

polynomials, such that the optimization of Equation: 2 is conducted over the basis coefficients. The usage of the Laguerre polynomials provides a computationally efficient MPC implementation with stability guarantees with exponentially decaying behaviour. The numerical stability of these methods was demonstrated in [12], however, in practice, these methods are limited to their reliance on an accurate model and linear dynamics. However, Laguerre functions remain a modern and relevant approach in MPC, as demonstrated in their usage in robotics and power robotics [13], [14].

Our work demonstrates a data driven method that provides a reformulation of the MPC problem in the Legendre basis to dramatically improve optimization time performance, even for nonlinear dynamics. We utilize the Legendre basis for its optimal compression properties of time series data [15]. We efficiently approximate the dynamics through a neural network that maps control trajectories and initial conditions to the corresponding state trajectories using the coefficients of the Legendre basis. With this technique, we demonstrate a speedup in computation time of between 23 and 47 times, while retaining comparable tracking performance.

In the remainder of the paper, we begin by formulating the unconstrained MPC problem in the Legendre domain and then incorporate constraints in Section II. In Section III we use our method on the cart pole problem, demonstrating the improvements in runtime, and the preservation of those improvements when introducing constraints.

II. Methodology

In this section, we first present the process of system identification using the Legendre basis by predicting trajectories with a neural network. We then reformulate the MPC problem in the Legendre domain, which uses the trajectory prediction. Finally, we discuss the benefits and challenges associated with working in the Legendre domain for MPC.

A. Dynamics Modelling

In this section, we introduce our method of dynamics modelling using a Legendre representation. For this application we apply the specific shifted and scaled Legendre polynomials proposed in [16], as they have been shown to be optimal for streaming time series representations [15]. These polynomials are orthogonal over the interval $[0, 1]$ and take on values over $[-1, 1]$ as seen in Fig. 1. For the remainder of the paper, we refer to this as the Legendre basis. We use the Legendre basis to approximate the dynamics of the system by mapping between the Legendre coefficients of the state and control signals with a neural network. This mapping is learned, and hence the method is a form of data-driven modelling. As well, the initial conditions are provided to the neural network as input as shown in (3). Consequently, the neural network predicts the entire compressed state trajectory in a single inference step.

$$M_{X_{1:N_p}} = \hat{F}_{NN,L}(x_0, M_{U_{0:N_p-1}}), \tag{3}$$

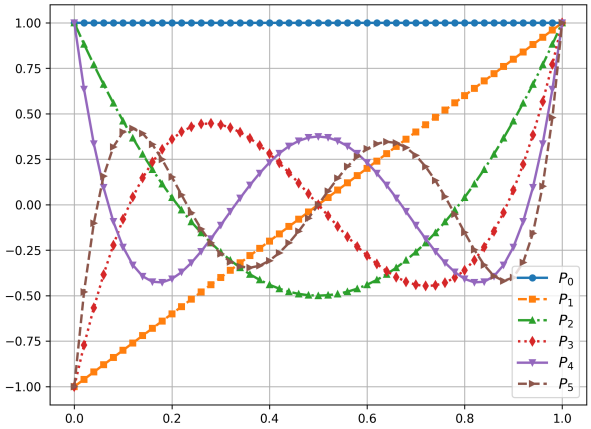

Fig. 1. Shifted and scaled Legendre polynomials up to and including order 5

where $F_{NN,L}$ is a neural network function mapping the initial state x_0 and the coefficients M_u of the Legendre representation of the control signal u to M_x, the coefficients of the Legendre representation of the state x trajectory across the prediction window N_p. The size of the network's inputs and outputs is determined by the state dimension and the order of the Legendre representation. Critically, our implementation has far fewer trainable parameters than a trajectory mapping in the time domain, which has an input and output size equal to the prediction window length.

In this work we use a neural network consisting of a single hidden layer with rectified linear neurons, to obtain fast inference times. For a given hidden layer, the total number of trainable parameters is a function of the number of neuron biases and connection weights, which is equal to

$$\begin{aligned} N_w &= (D_i + D_o) * N_n \\ N_b &= N_n + D_o \\ N_t &= N_w + N_b, \end{aligned} \tag{4}$$

where N_w, N_b, and N_t are the number of weights, biases, and total parameters, N_n is the number of neurons in the hidden layer, and D_i and D_o are the number of dimensions in the input and output, respectively. For the Legendre domain network, we let D_i be the order of the representation L plus the number of states, and D_o is only L. Since we are representing a trajectory over a window with the Legendre coefficients, to do the equivalent computation in the time domain, we would have to map each point in the time domain. Therefore, in the time domain, D_i would be equal to the length of the prediction window N_p plus the number of states, and D_o is simply N_p. We assume that $N_p > L$ to justify the usage of a Legendre representation as a form of compression.

To train the neural network, we first constructed our training datasets through simulation of the plant. We randomly sampled initial conditions and control trajectory coefficients in the Legendre domain from their respective feasible sets. We then simulated the plant's response to the sampled input and compressed the corresponding state trajectory from the

time domain into the Legendre domain as the output data. We trained our model with PyTorch [17] using the Adam Optimizer which employs a gradient descent technique over the training data.

B. MPC in the Legendre Domain

We can now use a neural network to predict the system dynamics using the Legendre basis, which suggests formulating the remaining elements of the MPC problem (2) in the Legendre domain as well. We do this with the goal of decreasing runtime by optimizing directly within the compressed Legendre space. By conducting all steps of the optimization process in the Legendre domain, our method is an efficient alternative to time domain implementations.

We start by considering a trajectory X across a time fixed window. We can represent this trajectory by a linear combination of the Legendre basis polynomials and corresponding coefficients in the following form:

$$x(t) = \sum_{n=0}^{\infty} P_n(t) M_{n,x}, \tag{5}$$

where $P_n(t)$ is the n^{th} Legendre polynomial and $M_{n,x}$ is the n^{th} scalar coefficient corresponding to the representation of $x(t)$. Since it is impractical to use an infinite order representation to capture $x(t)$, we can use a reduced order representation, that may compress the original $x(t)$:

$$\hat{x}(t) = \sum_{n=0}^{L} P_n(t) M_{n,x} \tag{6}$$

where $\hat{x}(t)$ is represented to the L^{th} order. The error, $\varepsilon(t)$, resulting from this representation is:

$$x(t) - \hat{x}(t) = \varepsilon(t) = \sum_{n=L+1}^{\infty} P_n(t) M_{n,x}. \tag{7}$$

Since $\varepsilon(t)$ only contains higher frequency (i.e., polynomial) components, we assume it to be negligible by choosing L such that $\hat{x}(t)$ is a good representation of $x(t)$.

Let us now consider the cost function defined in (2). Since x, x_r, u are trajectories over the window $[0, N_p]$, we can represent them in the Legendre basis:

$$\begin{aligned} V(M_u) &= \sum_{k=0}^{N_p-1} \left[\left(\sum_{n=0}^{L} P_n(k) M_{n,x} - \sum_{n=0}^{L} P_n(k) M_{n,x_r} \right)^T \right. \\ &* Q \left(\sum_{n=0}^{L} P_n(k) M_{n,x} - \sum_{n=0}^{L} P_n(k) M_{n,x_r} \right) \\ &+ \left(\sum_{n=0}^{L} P_n(k) M_{n,u} \right)^T \\ &* \left. R \left(\sum_{n=0}^{L} P_n(k) M_{n,u} \right) \right] \end{aligned} \tag{8}$$

In vector notation, we can rewrite (8) as:

$$V(M_u) = (M_x - M_{x_r})^T \hat{Q} (M_x - M_{x_r}) + M_u^T \hat{R} M_u, \tag{9}$$

where we remove all M terms from the time dependant summation, and define $\hat{Q}$ and $\hat{R}$ as the following augmented weighting matrices, which can be computed offline:

$$\hat{Q} = \sum_{k=0}^{N_p-1} P(k)^T Q P(k) \tag{10}$$

$$\hat{R} = \sum_{k=0}^{N_p-1} P(k)^T R P(k). \tag{11}$$

Note that this cost function has no dependence on time, and can provide a continuous time representation which may provide benefits for evaluating the cost function compared to a discrete representation in the time domain.

C. Constraints

There are a variety of constraints we may wish to impose on our controller, including control bounds, control or state energy bounds, system state constraints, and terminal constraints. A potential limitation of our method is that by projecting the problem to the Legendre domain, it becomes unclear how we can enforce constraints specified in the time domain.

For control bounds, constraints are often expressed as an inequality. For instance, bounding the control signal $u(t)$ below a maximum value $\bar{U}$:

$$u(k) \leq \bar{U}, \forall k \in [0, N_p - 1].$$

Writing this in the Legendre domain results in the following:

$$\sum_{n=0}^{L} P_n(k) M_{n,u} \leq \sum_{n=o}^{L} P(k) M_{n,\bar{U}}, \forall k \in [0, N_p - 1].$$

However, this result is computationally equivalent to decoding the Legendre representation into the time domain and checking constraints for all points k in $[0, N_p - 1]$. While effective, decoding at every time step will be computationally costly, sacrificing the efficiency gains from moving to the Legendre domain.

A naive approach is to simply bound the coefficients represented by M directly. However, this would lead to bounds with different meanings in the two domains. For instance, with a constant control bound $\bar{U}$, only the first Legendre polynomial is needed to represent a constant, so direct bounds on the coefficients M_u give the following:

$$M_{n,u} \leq M_{n,\bar{U}}, \forall n \in [0, L],$$

where:

$$M_{n,\bar{U}} = \begin{cases} \bar{U} & \text{if } n = 0, \\ 0 & \text{otherwise.} \end{cases}$$

This forces all higher order terms to be equal to 0, meaning only constant control signals can be selected, which is not consistent with the time domain constraint.

We use an alternative approach that compromises between reliability and runtime by evaluating constraints in the time domain, but only at specific sample points across the prediction window. Consequently, we decode the Legendre representation into the time domain at N sample points which we

can then directly evaluate against time domain constraints. Our choice of the number of samples is driven by a trade-off between the efficiency of optimization and satisfaction of constraints. A larger N increases the optimization time while providing more confidence that the constraints are satisfied. However, we can carefully select sample points to leverage the known properties of the Legendre polynomials, such as examining the peaks of the polynomial components for the trajectory extrema rather than the entire prediction window. In practice, we find very few samples are required to effectively enforce constraints (see Section III).

Turning now to energy bounds, the solution is more straightforward. We consider an energy bound $\bar{E}_u$ on the control signal U over the window $[0, Np-1]$.

$$\sum_{k=0}^{N_p-1} u(k)^2 \leq \bar{E}_u.$$

Recall that:

$$\begin{aligned} u(k) &= \sum_{n=0}^{\infty} P_n(k) M_{n,u} \\ &= \sum_{n=0}^{L} P_n(k) M_{n,u} + \sum_{n=L}^{\infty} P_n(k) M_{n,u} \\ &= \hat{u}(k) + \varepsilon(k). \end{aligned}$$

We can now rewrite the constraint as: $\sum_{k=0}^{N_p-1} \hat{u}(k)^2 \leq \bar{E}_u - \sum_{k=0}^{N_p-1} \varepsilon(k)^2$, where cross terms are equal to zero because the basis is orthogonal. Rewriting this in vector form gives: $M_u^T \hat{P} M_u \leq \bar{E}_u - \sum_{k=0}^{N_p-1} \varepsilon(k)^2$, where $\hat{P}$ can be computed offline.With an appropriate choice of order, which is also required for a good representation of control and state trajectories, the energy of the error signal is small, meaning that the bound will be well-respected.

Finally, we consider terminal constraints, as these are specifically helpful for ensuring stability through Lyapunov techniques [18]. By noticing that the Legendre polynomial's terminal points are all valued at one and have a positive derivative, we can simplify the constraint to apply directly to the sum of the trajectory coefficients:

$$\sum_{n=0}^{L} M_{n,x} \leq \bar{M}_x.$$

However, this sum may overestimate the true terminal value, as signals represented by the Legendre polynomials often contain end effects from this feature of the basis. Therefore, we note that to ensure good performance, the value of $\bar{M}_x$ must be carefully selected by increasing the terminal bounds or applying the terminal constraint slightly before the end point.

III. Results

In this section, we first demonstrate the ability to effectively predict dynamics using the Legendre basis with a neural network while using fewer parameters than doing the same in the time domain. We then use our proposed MPC formulation alongside a ground truth time domain implementation with and without constraints.

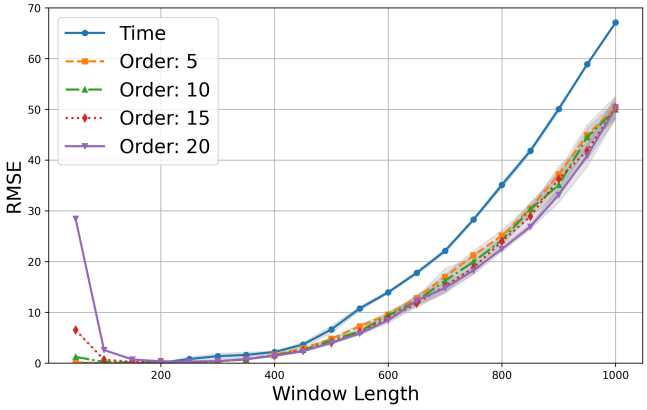

Fig. 2. RMSE of the neural network model with a constant hidden layer size for Legendre and time domain mappings. RMSE is always calculated in the time domain.

A. Dynamics Modelling in the Legendre Domain

We use a cart pole benchmark to conduct our simulations with the Legendre MPC. A cart pole has four states corresponding to the angular and lateral position and their velocity components. Typically the goal of the controller is to reach a desired angular position through a lateral forcing control signal u. We define the state space model as:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{u + m_p l \, sin(x_3) x_4^2 - m_p g \, cos(x_3) sin(x_3)}{m_c + m_p sin^2(x_3)} \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= \frac{g \, sin(x^3) - cos(x_3)(\frac{u + m_p l \, sin(x_3) x_4^2}{m_c + M_p sin^2(x_3)})}{l(\frac{4}{3} - \frac{m_p cos^2(x_3)}{m_c + m_p sin^2(x_3)})}, \end{aligned}$$

where x_1 is the lateral position of the cart and x_3 is the angle of the pole, which is the output state to be controlled.

We apply our modelling technique introduced in Section II-A to approximate the dynamics of the cart. Specifically, we map u, X_0 to x_3 trajectories over the prediction window. In Fig. 2 we vary the prediction window length from 50 to 1000 time steps, while the hidden layer of the neural network is fixed at a size of 128 neurons. We evaluate the root mean square error (RMSE) between the ground truth and predicted state trajectory decoded into the time domain at each window length. For each trial, the model was trained over 20 epochs with a learning rate of 10^{-4}. The improved performance of the Legendre basis network is evident across window sizes, and not particularly sensitive to the choice of the order of the representation, although high-order representations can cause aliasing at short window lengths.

In Fig. 3, we repeat the experiment but fix the total number of parameters to approximately 10,000 and vary the middle layer size. Our results are similar in that using neural networks in the Legendre domain is an effective way of mapping from control trajectories to state trajectories

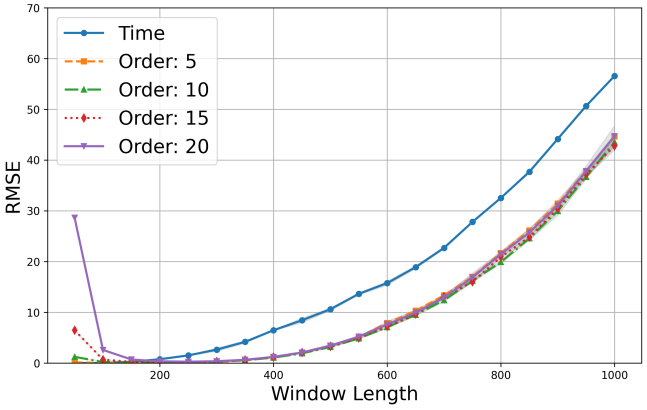

Fig. 3. RMSE of the neural network model with a constant number of parameters for Legendre and time domain mappings. RMSE is always calculated in the time domain.

in one inference step. Specifically, these graphs show that the Legendre representation is consistently better for fixed network resources.

We also observe improvements in training and inference time. For example, for our system with 4 states and a prediction window of 100 time steps being represented with a 5^{th} order Legendre representation, we can use (4) to find the total number of trainable parameters. With a middle layer fixed at 128, the Legendre domain neural network contains 1925 parameters, while the corresponding time domain neural network contains 26,340 parameters. Such a parameter reduction provides a speed up in both training and inference time.

B. MPC Simulation Results

Our simulations are performed in Python with the Sequential Least Squares Quadratic Programming (SLSQP) algorithm from SCIPY [19], which uses a gradient descent technique to solve the optimization with specified constraints. We use a single shooting technique in which we sample and evaluate the entirety of a single trajectory before updating the optimization control trajectory. We use a simulation time of 1s with a sampling time of 1 ms, corresponding to 1000 time steps. We use a prediction window N_p of 0.1s with a 10^{th} order Legendre representation. Finally, we use a step size of 0.002 for the numerical approximation of the gradient in the optimization. The remaining parameters vary by simulation.

The final model used in the following closed loop control task was constructed with a hidden layer of 500 neurons. We trained the model on 10^6 training samples uniformly sampled across the control and state space in the Legendre domain. The model was trained through 20 epochs with a learning rate of 10^-4.

We first show a simple simulation with no constraints and a cost function that sets Q at 10 and R at $2 * 10^{-5}$. We set the reference as both a step function (see Fig. 4a) and a shifted sine wave (see Fig. 4b). Furthermore, we simulate a time domain implementation as formulated in (2) that uses standard numerical modelling methods for

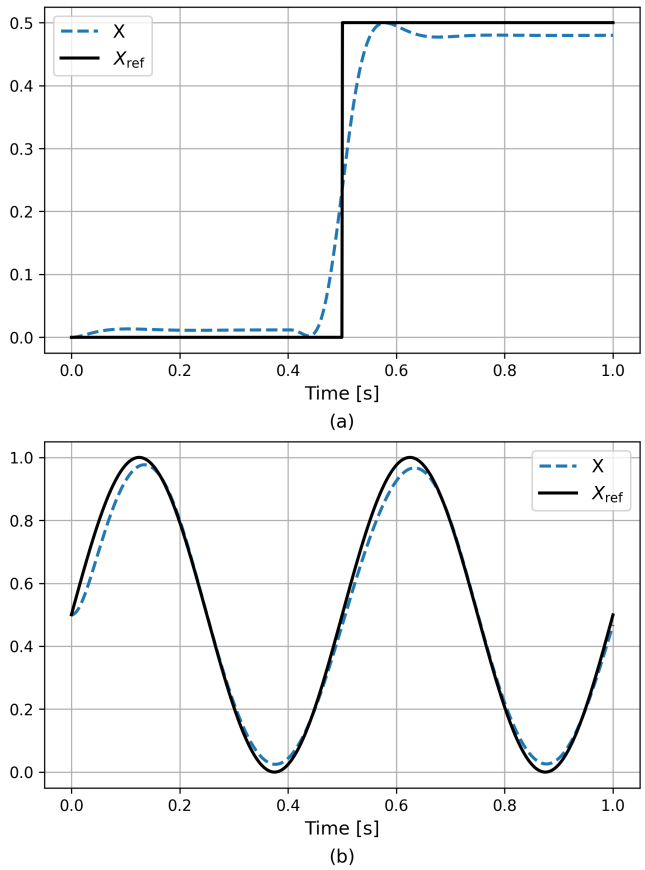

Fig. 4. Legendre domain MPC applied to the cart pole problem with a) step function, and b) shifted sine wave reference signals

TABLE I

SIMULATION RESULTS OF TIME AND LEGENDRE MPC FOR REFERENCE TRAJECTORIES OF A SINE WAVE AND A STEP FUNCTION

	Sine Wave		Step Function	
	Time	Legendre	Time	Legendre
Time [s]	894.50	21.96	425.62	11.67
RMSE	0.03532	0.0338	0.0428	0.0436
Speedup	40.73		36.47	

predicting dynamics and with identical parameters to provide baseline results for comparison. We summarize the results of these simulations, including runtime and error between the reference and state trajectory in Table I. We observe similar tracking performance between the standard time domain and our Legendre domain implementations of MPC. However, our Legendre domain MPC achieves a speedup between 31-40x that of the time domain. While the time domain implementation can achieve significantly higher tracking performance by tuning step size and state error weighting, this consequently also deteriorates the runtime performance.

Additionally, we observe in Fig. 4 that a steady state error persists even with state feedback. This error is the result of a modelling discrepancy between the neural network from Equation (3) and the ground truth dynamics defined in Equation (2). As a result, our model is predicting an optimal trajectory with respect to the formulation in Equation (9).

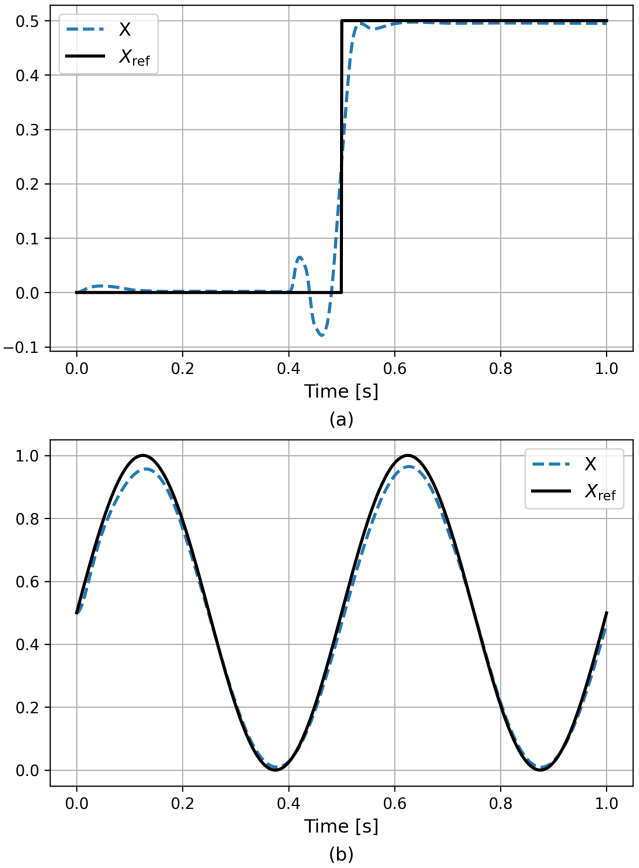

Fig. 5. Legendre domain MPC applied to the cart pole problem with terminal and control constraints for a) step function, and b) shifted sine wave reference signals

TABLE II
SIMULATION RESULTS OF TIME AND LEGENDRE MPC UNDER VARIOUS CONSTRAINTS

	Sine Wave		Step Function	
	Time	Legendre	Time	Legendre
Time [s]	996.60	32.05	1312.35	33.42
RMSE	0.0078	0.0263	0.0271	0.0331
Speedup	31.10		39.27	

However, the actual plant follows a trajectory which can incur a steady state error relative to the reference trajectory. This suggests that enforcing constraints may be helpful for improving control performance.

We remove the control signal weighting on R and enforce both control signal and terminal constraints for a step function and sine wave in Fig. 5a and b. We choose terminal constraints of ± 0.1 and control signal bounds of ± 500 as to match the control bounds from the training data. In simulation the control trajectory obeys the control bounds for both reference trajectories. We summarize the results of each simulation in Table II and show that the speedup of our method persists with an improvement of performance.

C. Discussion

Our reformulation of the MPC problem with a Legendre basis provides significant speedups compared to standard solutions in the time domain. We tested the method with other bases (e.g., Leguerre) and found that they were not as performant as the Legendre basis (results not shown).

However, the formulation in the Legendre domain requires the selection of hyperparameters that are not as familiar to researchers. For instance, careful choice of the order of the representation used is important, as it must balance between efficiency and accuracy of representing the space of possible trajectories. The best choice will be task dependent. This is similar to choosing a discretization time step.

As well, specifying constraints in the Legendre formulation is less familiar, although we have provided several examples here. Notably, we did not demonstrate hidden state constraints, as they provide additional challenges due to our neural network formulation of Equation (3). Specifically, because our model only maps the control trajectory and initial conditions to the controlled output state, we do not predict hidden state values. While we can infer the state derivative directly from the state trajectory, if constraints were to be applied to the hidden state, we must include the corresponding state trajectory as an output of our neural network. This would increase the complexity of the neural network as our model would need to map the control input and initial condition to both the output and hidden state. We leave such considerations for future work, alongside further experiments to verify the results presented hold in more complex control scenarios.

IV. CONCLUSIONS

In this paper, we derive a Legendre domain implementation of MPC with a data driven dynamic model implemented by a neural network. We use the neural network to predict system dynamics over an entire window in a single step and demonstrate improvement over similar trajectory mapping in the time domain. We then reformulate the MPC problem in the Legendre domain to enable optimization over a compressed representation space. We show a runtime speedup of 31-40x with our technique compared to an equivalent time domain implementation with comparable tracking performance and demonstrate that this advantage remains with the introduction of various constraints.

REFERENCES

[1] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[2] S. Kouro, P. Cortes, R. Vargas, U. Ammann, and J. Rodriguez, "Model predictive control—a simple and powerful method to control power converters," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 1826–1838, 2009.

[3] L. D. Tufa and C. Z. Ka, "Effect of model plant mismatch on mpc performance and mismatch threshold determination," *Procedia Engineering*, vol. 148, pp. 1008–1014, 2016. Proceeding of 4th International Conference on Process Engineering and Advanced Materials (ICPEAM 2016).

[4] G. Elnagar, M. Kazemi, and M. Razzaghi, "The pseudospectral legendre method for discretizing optimal control problems," *IEEE Transactions on Automatic Control*, vol. 40, no. 10, pp. 1793–1796, 1995.

[5] G. H. Golub and J. H. Welsch, "Calculation of gauss quadrature rules," in *Milestones in Matrix Computation*, 1967.

[6] F. Fahroo and I. Ross, "Direct trajectory optimization by a chebyshev pseudospectral method," in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*, vol. 6, pp. 3860–3864 vol.6, 2000.

[7] F. G. Harmon, "Hybrid solution of nonlinear stochastic optimal control problems using legendre pseudospectral and generalized polynomial chaos algorithms," in *2017 American Control Conference (ACC)*, pp. 2642–2647, 2017.

[8] J. P. Allamaa, P. Patrinos, H. Van Der Auweraer, and T. D. Son, "Safety envelope for orthogonal collocation methods in embedded optimal control," in *2023 European Control Conference (ECC)*, pp. 1–7, 2023.

[9] M. N. Reddy, M. Miyatake, and J. V. P. P. Dias, "Dynamic programming application for pseudospectral optimal train control problem," in *2024 IEEE 18th International Conference on Advanced Motion Control (AMC)*, pp. 1–6, 2024.

[10] G. V. Haman and A. V. Rao, "An error estimation and mesh refinement method applied to optimal libration point orbit transfers," in *2024 American Control Conference (ACC)*, pp. 2325–2330, 2024.

[11] L. Wang, "Discrete time model predictive control design using laguerre functions," in *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)*, pp. 2430–2435 vol.3, 2001.

[12] L. Wang, "Discrete model predictive control using laguerre functions: numerical sensitivity analysis," in *Proceedings of the 2003 American Control Conference, 2003.*, vol. 2, pp. 1488–1493, 2003.

[13] J. Saeed, L. Wang, and N. Fernando, "Model predictive control of phase shift full-bridge dc–dc converter using laguerre functions," *IEEE Transactions on Control Systems Technology*, vol. 30, no. 2, pp. 819–826, 2022.

[14] S. Khoshkam, M. A. Khosravi, and R. FesharakiFard, "Model predictive control for a 3-dof suspended cable robot based on laguerre functions," in *2022 30th International Conference on Electrical Engineering (ICEE)*, pp. 827–832, 2022.

[15] A. Voelker, *Dynamical Systems in Spiking Neuromorphic Hardware*. PhD thesis, University of Waterloo, 2019.

[16] A. R. Voelker, I. Kajić, and C. Eliasmith, "Legendre memory units: Continuous-time representation in recurrent neural networks," in *Advances in Neural Information Processing Systems*, pp. 15544–15553, 2019.

[17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[18] P. Mhaskar, N. H. El-Farra, and P. D. Christofides, "Stabilization of nonlinear systems with state and control constraints using lyapunov-based predictive control," *Systems & Control Letters*, vol. 55, no. 8, pp. 650–659, 2006. New Trends in Nonlinear Control.

[19] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Wright, *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in python," 2020.