Discovering interpretable piecewise nonlinear model predictive control laws via symbolic decision trees*

Ilias Mitrai¹

Abstract—In this paper, we propose symbolic decision trees as surrogate models for approximating model predictive control laws. The proposed approach learns simultaneously the partition of the input domain (splitting logic) as well as local nonlinear expressions for predicting the control action leading to interpretable piecewise nonlinear control laws. The local nonlinear expressions are determined by the learning problem and are modeled using a set of basis functions. The learning task is posed as a mixed integer optimization, which is solved to global optimality with state-of-the-art global optimization solvers. We apply the proposed approach to a case study regarding the control of an isothermal reactor. The results show that the proposed approach can learn the control law accurately, leading to closed-loop performance comparable to that of a standard model predictive controller. Finally, comparison with existing interpretable models shows that the symbolic trees achieve both lower prediction error and superior closed-loop performance.

I. INTRODUCTION

Model predictive control (MPC) is a commonly used control strategy for constrained systems [1]. MPC relies on the iterative solution of an open-loop optimal control problem for a given state of the system. Although the theoretical aspects of MPC have been established for a wide range of dynamical systems, the online implementation is challenging due to high computational cost.

Several acceleration techniques have been proposed to reduce the computational time. In general, these techniques focus either on the problem formulation or the solution method. For the former, the standard approach is to approximate the dynamic behavior of the system with linear models, leading to the so-called linear MPC. Although this modeling approach leads to convex optimization problems, the control performance can be limited for highly nonlinear systems or cases where the set point can change significantly. The second acceleration approach focuses on developing faster optimization algorithms that exploit the structure of the optimal control problem [2], [3], [4], [5]. Despite the significant improvement in the computational performance of linear and nonlinear optimization solvers, the solution time can still exceed the sampling time for high-dimensional nonlinear systems.

The aforementioned limitations have motivated the application of machine learning (ML) for reducing the computational effort required for computing the control action at each sampling time. For example, ML surrogate models can

be used to develop surrogate models [6], [7], [8], [9]. An alternative approach, and the one we focus on in this paper, is to use machine learning to approximate the solution of a MPC problem.

A model predictive controller is a map $\pi(\cdot)$ between Euclidean spaces, i.e., the current state of the system x_0 and the optimal control action $u = \pi(x_0)$. We will refer to $\pi(\cdot)$ as the control law. Although for linear and certain nonlinear systems the control law can be computed exactly [10], [11], for general nonlinear systems, the computation is intractable. This has motivated the application of machine learning, specifically deep neural networks [12], [13], [14], [15], [16], for learning the control law. In this approach, the training data are generated from closed-loop simulations, i.e., for each data point, the features capture the state of the system and the label is the control action. Although deep neural networks can approximate general nonlinear functions due to their universal approximation properties, they are inherently blackbox. Thus, neural network verification techniques, which are computationally expensive, must be used to analyze the stability of the learned controller [17].

An alternative to using deep learning is the use of an interpretable machine learning model for learning the control policy. Decision trees, both the ones with piecewise constant [18] and piecewise linear [19] expressions on the leaves, have been used recently to approximate the control law [20], [21], [22], [23]. Decision trees are inherently interpretable models, since one can understand the prediction of the model by examining the splitting logic and the expression used for prediction at each leaf. Moreover, the splitting logic can naturally identify regions where active constraints do not change, i.e., critical regions. Despite these advantages, the application to generic nonlinear systems is challenging, since the control law can be highly nonlinear. Although one can use the existing decision trees with linear and polynomial [24] expressions, a very fine partition of the input domain might be required to achieve high accuracy, i.e., the depth of the decision tree increases significantly [25].

In this paper, we propose symbolic decision trees (see Fig. 1) as a surrogate model for learning interpretable piecewise continuous and nonlinear symbolic control laws. The proposed approach combines standard decision trees and basis functions, leading to a mixed integer optimization learning problem. Specifically, the splitting logic is modeled using binary variables, similar to the optimal classification trees presented in [26], [27]. The main difference of the proposed model from other decision tree models is the presence of general nonlinear expression on the leaves, which

^{*}This work was supported by the McKetta Department of Chemical Engineering.

Ilias Mitrai is with the McKetta Department of Chemical Engineering, University of Texas at Austin, 78712, Austin, TX, USA imitrai@che.utexas.edu

are represented by a set of basis functions. The resulting learning task is a mixed integer optimization problem. We note that a similar approach has been proposed in [28], where the function at each leaf is represented symbolically. However, the training is done heuristically using genetic programming. In this paper, we use basis functions and pose the learning problem as a mixed integer optimization problem and solve it to global optimality.

We use the proposed symbolic decision trees to learn the control law of a model predictive controller used to control the concentration in an isothermal continuously stirred tank reactor. We compare the proposed approach with existing interpretable surrogate models. The results show that symbolic decision trees can approximate the control law accuracy, leading to better closed-loop control performance compared to other decision tree models with constant and linear predictors.

The rest of the document is organized as follows: In Section II, we present the symbolic decision tree learning problem and the application to learning model predictive control laws. In Section III we apply the proposed approach to the case study and compare the predictive and closed-loop control performance of the proposed approach with existing baselines.

II. DISCOVERING CONTROL LAWS VIA SYMBOLIC DECISION TREES

A. Learning symbolic decision trees via mixed integer optimization

We assume that we are given a data set $\mathcal{D} = \{x_i, y_i\}_{i=1}^{N_d}$ with N_d input output samples generated by a function f(x). We denote as $\mathcal{I} = \{1, \dots, N_d\}$ the data set index. Each sample has N_f features with x_{if} being the value of the feature f for data point i.

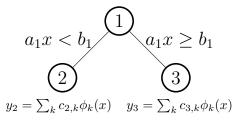


Fig. 1: Decision trees with expressions in the leaves

We model the function f(x) as a decision tree of fixed depth D, where each node n has two children 2n and 2n+1. We define set $\mathcal{N}=\{1,...,2^{D+1}-1\}$ as the set of nodes in the tree, $\mathcal{T}=\{2^D,...,2^{D+1}-1\}$ as the set of terminal nodes, and $\mathcal{N}_{int}=\mathcal{N}\setminus\mathcal{T}$ as the set of internal nodes. Finally, we define as $\mathcal{K}=\{1,\ldots,N_K\}$ the set of basis functions used at the nodes. Under this setting, each node in the tree is either a branching node, i.e., one where the validity of a linear inequality $a^{\top}x < b$ and $a^{\top}x \geq b$ is checked, a terminal node, i.e., one where the data points are assigned, or an inactive node. The constraints used to capture the structure of the tree and the assignments of the data to nodes are adapted from [26].

Given these sets, the learning problem has two sets of variables: ones that determine the structure and splitting logic of the tree, and ones used to compute the label of a data point based on the splitting logic. We define a binary variable d_n which is equal to one if node n is a branching node and 0 otherwise. The first set of constraints guarantees that the structure of the decision tree is correct, i.e., a node can be a branching node only if its parent is also a branching node. This is enforced by the following constraints

$$d_{2n} \leq d_n \ \forall n \in \mathcal{N}_{int}$$

$$d_{2n+1} \leq d_n \forall n \in \mathcal{N}_{int}$$

$$d_1 = 1$$

$$d_n = 0 \ \forall n \in \mathcal{T}.$$
(1)

The third and fourth constraints enforce that the root node is a branching node, whereas the leaf nodes are not.

We define a binary variable z_{in} which is equal to one if data point i is assigned to node n and zero otherwise. The assignment of a data point to a node is determined by the nature of the node and the splitting logic. First, if node n is a branching node, i.e., $d_n=1$, then data can not be assigned to that node, i.e., $z_{in}=0 \ \forall i\in\mathcal{I}$. This is enforced via the following constraint

$$z_{in} \le 1 - d_n \ \forall i \in \mathcal{I}, n \in \mathcal{N}.$$
 (2)

For a given branching logic, each data point must be assigned to a node, i.e., the tree must generate a label for all data. This is enforced by the following constraint

$$\sum_{n \in \mathcal{N}} z_{in} = 1 \ \forall \ i \in \mathcal{I}. \tag{3}$$

If a data point is assigned to a node n, then all the ancestors of node n are branching nodes. This requirement enforces that data are not assigned to inactive nodes and is enforced via the following constraint

$$z_{in} \le d_{n'} \ \forall \ i \in \mathcal{I}, n' \in A(n), \tag{4}$$

where A(n) are the ancestors for node n, i.e., all the nodes from the root until the parent of node n.

We define a binary variable a_{fn} which is equal to one if node n branches on feature f at node n and zero otherwise. Also, we define the continuous variable b_n , which is the split threshold at node n. The value of these coefficients depends on the branching logic of the tree as follows

$$\sum_{f \in \mathcal{F}} a_{fn} = d_n \quad \forall \ n \in \mathcal{N}_{int}. \tag{5}$$

This constraint enforces that at a branching node n, i.e., $d_n = 1$, one feature must be used for branching; otherwise, if the node is not a branching one, branching should not occur.

The constraints presented so far guarantee that the structure of the tree is correct and the data can be assigned to leaf nodes. Next, we define the routing constraints, which determine the logic conditions that must hold at each node. If a data point i is assigned to a node n, i.e., $z_{in} = 1$, then the data point must satisfy all the logic conditions that hold

at the ancestors of node n. We define as R(n) and L(n) the nodes that are in the path from the root to node n which are branched right and left, respectively. The routing constraints for each data point $i \in \mathcal{I}$ are

$$\sum_{f \in \mathcal{F}} a_{fm} x_i \le b_m - \epsilon + M(1 - z_{in}) \ \forall \ n \in \mathcal{N}, m \in L(n)$$

$$\sum_{f \in \mathcal{F}} a_{fm} x_i \ge b_m - M(1 - z_{in}) \ \forall \ n \in \mathcal{N}, m \in R(n).$$

The last set of constraints and variables is related to the computation of the model output. We define as $c_{kn} \in [c^{\mathrm{lb}}, c^{\mathrm{ub}}]$ the value of the constant multiplying the k^{th} basis function at node n. We also define as $\hat{y}_{in} \in [y^{\mathrm{lb}}, y^{\mathrm{ub}}]$ the prediction of the expression at node n for data point i and $y_i^{pred} \in [y^{\mathrm{lb}}, y^{\mathrm{ub}}]$ as the output of the tree for data point i. The variables c_{kn} are defined for each node since we do not know a priori which are the terminal nodes. The prediction at each node and data point is computed by the following constraints

$$\hat{y}_{in} = \sum_{k \in \mathcal{K}} c_{kn} \phi_k(x_i) \ \forall \ i \in \mathcal{I}, n \in \mathcal{N}, \tag{7}$$

where $\phi_k(x_i)$ is the value of the k^{th} basis function for data point x_i . If node n is a branching node, then the associated constants should be zero. This enforced via the following constraint

$$c_{kn} \le c^{\text{ub}}(1 - d_n) \ \forall \ k \in \mathcal{K}, n \in \mathcal{N}$$

$$c_{kn} \ge c^{\text{lb}}(1 - d_n) \ \forall \ k \in \mathcal{K}, n \in \mathcal{N}.$$
 (8)

The prediction for data point i is

$$y_i^{pred} = \sum_{n \in \mathcal{N}} \hat{y}_{in} z_{in}.$$

This constraint can be linearized since it contains the summation of products between a binary z_{in} and a continuous \hat{y}_{in} variable. We define $\delta_{in} \in [y^{\mathrm{lb}}, y^{\mathrm{ub}}]$ and $\delta_{in} = \hat{y}_{in}z_{in}$. Each bilinear term is linearized as follows

$$\delta_{in} \leq y^{\text{ub}} z_{in}
\delta_{in} \geq y^{\text{lb}} z_{in}
\delta_{in} \leq \hat{y}_{in} + M(1 - z_{in})
\delta_{in} \geq \hat{y}_{in} - M(1 - z_{in}),$$
(9)

where M is a large constant. Under this linearization, we have

$$y_i^{pred} = \sum_{n \in \mathcal{N}} \delta_{in} \ \forall \ i \in \mathcal{I}$$
 (10)

The objective of the learning problem has three components: prediction error \mathcal{L}_{loss} , complexity of the tree \mathcal{L}_{compl} , and magnitude of the constants \mathcal{L}_{c} . Each term is computed as follows

$$\mathcal{L}_{acc} = \frac{1}{N_d} \sum_{i \in \mathcal{I}} |y_i - y_i^{pred}|$$

$$\mathcal{L}_{compl} = \sum_{n \in \mathcal{N}} d_n$$

$$\mathcal{L}_m = \sum_{k \in \mathcal{K}} \sum_{n \in \mathcal{N}} |c_{kn}|.$$

We reformulate the absolute terms by defining the following nonnegative variables $\epsilon_i^+, \epsilon_i^-, \hat{c}_{kn}^+, \hat{c}_{kn}^-$ and writing the absolute values as follows

$$\epsilon_{i}^{+} - \epsilon_{i}^{-} = y_{i} - y_{i}^{pred}$$

$$\hat{c}_{kn}^{+} - \hat{c}_{kn}^{-} = c_{kn}.$$
(11)

The reformulated \mathcal{L}_{acc} and \mathcal{L}_m terms are

$$\begin{split} \mathcal{L}_{acc} &= \frac{1}{N_d} \sum_{i \in \mathcal{I}} \left(\epsilon_i^+ + \epsilon_i^- \right) \\ \mathcal{L}_m &= \sum_{k \in \mathcal{K}} \sum_{n \in \mathcal{N}} \left(\hat{c}_{kn}^+ + \hat{c}_{kn}^- \right), \end{split}$$

and the learning problem is

min
$$\mathcal{L}_{acc} + \lambda_c \mathcal{L}_c + \lambda_m \mathcal{L}_m$$

s.t. Eq. 1 – 11. (12)

where λ_c , λ_m are weight factors for the different terms in the objective. Overall, the learning problem is a mixed integer linear optimization problem that can be solved with existing optimization solvers.

B. Learning symbolic control laws

We consider a dynamical system with states $x(t) \in \mathbb{R}^{n_x}$ and manipulated variables $u(t) \in \mathbb{R}^{N_u}$ whose dynamic behavior is described by a set of differential equations

$$\frac{dx(t)}{dt} = f(x(t), u(t)), \tag{13}$$

where $f: \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \mapsto \mathbb{R}^{N_x}$.

We will consider the scenario where a model predictive controller is used to regulate the system at a desired set point $x^{\rm sp}$. At each sampling time, the control action is computed by solving the following optimization problem

$$\min_{x_{t}, u_{t}} \sum_{t=1}^{T} (x_{t} - x^{\text{sp}})^{2} + P(x_{T} - x^{\text{sp}})^{2}$$
s.t.
$$x_{t+1} = x_{t} + hf(x_{t}, u_{t}) \ \forall t = 1, \dots, T - 1$$

$$|u_{t+1} - u_{t}| \le h\bar{U} \ \forall t = 1, \dots, T - 1$$

$$x_{1} = x_{0}$$

$$x^{\text{lb}} \le x_{t} \le x^{\text{ub}} \ \forall t = 1, \dots, T$$

$$u^{\text{lb}} < u_{t} < u^{\text{ub}} \ \forall t = 1, \dots, T,$$

where T is the number of points used to discretize the time horizon, h is the discretization step, x_t is the value of the state variable at time t, u_t is the value of the manipulated variable at time t, x_0 is the state of the system at the sampling time, \bar{U} is the maximum rate of change of the manipulated variables, and P is a terminal cost.

It is well known that the solution of this optimization problem, i.e., the optimal control action to be implemented, depends on the initial state of the system x_0 . Therefore, a model predictive controller is a map $\pi(\cdot)$ between the state of the system x_0 and the control action $u=\pi(x_0)$. However, in classic MPC, this map is not computed explicitly; rather, the output is computed for a given state by solving the optimization problem.

We propose the application of the symbolic decision trees presented in the section above for learning the control law π . Specifically, the input to the model is the state of the system x_0 and the output is the control action u. This approach offers certain advantages over existing approaches. First, the symbolic decision tree is interpretable, since both the branching logic and the equations used to evaluate the control action in each region are symbolic in nature. Thus, one can study the convergence properties of the learned control law. Secondly, it can exploit the local structure of the control law, thus requiring simpler nonlinear expressions for achieving good predictive accuracy. Finally, compared to decision trees with linear leaves, it requires smaller trees, i.e., trees with smaller depth.

TABLE I: Features and their coefficients at each node of the decision tree

Feature	Coefficient			
	Node 4	Node 5	Node 6	Node 7
1	6.241	0.0	0.0	71.983
x	0.0	0.0	0.0	0.0
x^2	0.0	0.0	0.0	0.0
x^3	0.0	0.0	0.0	0.0
x^4	0.0	0.0	0.0	0.0
x^5	0.0	0.0	0.0	0.0
e^x	0.0	0.0	0.0	1.088
xe^x	0.0	0.0	0.0	0.0
x^2e^x	0.0	0.0	80.413	-0.407
x^3e^x	73.186	50.035	1.336	0.0
e^{-x}	53.793	0.0	0.0	0.0
xe^{-x}	0.0	0.0	0.0	0.0
x^2e^{-x}	0.0	0.0	0.0	0.0
x^3e^{-x}	0.0	0.0	0.0	0.0
$xe^{1/x}$	0.012	-1.62	-0.454	0.421
$x^2 e^{1/x}$	-0.262	20.739	0.0	0.0
$x^{3}e^{1/x}$	1.426	0.0	0.0	0.0
$e^{-1/x}$	-72.367	0.0	0.0	0.0
$xe^{-1/x}$	0.0	0.0	0.0	0.0

III. COMPUTATIONAL RESULTS

A. Case study description

We use the proposed approach to learn interpretable control laws for an isothermal continuously stirred tank reactor where an irreversible reaction $3A \to B$ occurs under constant volume. The dynamic behavior of the system is described by the following ordinary differential equation

$$\frac{dx(t)}{dt} = \frac{F(t)}{V}(x_f - x(t)) - kx(t)^3,$$
 (15)

where x(t) is the concentration of the reactant, F(t) is the inlet flowrate, V=50L is the volume, $x_f=1 \ mol/min$ is the inlet concentration, and $k=2L^2/(min \ mol^2)$ is the reaction constant.

For a given initial concentration and set-point, the control action is computed by the solution of the following optimiza-

tion problem

$$\min_{u_t, x_t} \sum_{t=1}^{T} (x_t - x^{\text{sp}})^2 + P(x_T - x^{\text{sp}})^2$$
s.t.
$$x_{t+1} = x_t + h \left(\frac{F_t}{V} (x_{feed} - x_t) - 2x_t^3 \right) \, \forall t = 1, \dots, T$$

$$x_1 = x_0$$

$$|F_{t+1} - F_t| \le h\bar{F} \quad \forall \ t = 1, \dots, T - 1$$

$$x^{\text{lb}} \le x_t \le x^{\text{ub}} \quad \forall \ t = 1, \dots, N$$

$$F^{\text{lb}} \le F_k \le F^{\text{ub}} \quad \forall \ k = 1, \dots, N - 1$$
(16)

where T=10 (h=0.5), P=100, $x^{\rm sp}=0.6$ min, $\bar{F}=50L/min$, and M=1000 (the big-M constant). This is a nonlinear optimization problem solved with IPOPT [29].

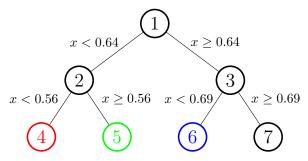


Fig. 2: Splitting logic in the learned decision tree

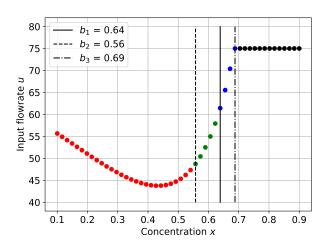


Fig. 3: Prediction of symbolic decision tree on the testing set

B. Data generation and training

We generate training data by solving the MPC problem for different initial concentrations. Specifically, we sample uniformly the initial concentration at $N_d=50$ points between $x^{\rm Ib}=0.1$ and $x^{\rm ub}=0.9$.

For the symbolic decision tree model, we use 19 basis functions presented in Table I. We fix the depth of the tree to two; thus, the decision tree has seven nodes in total and at most three branching nodes. The learning problem has

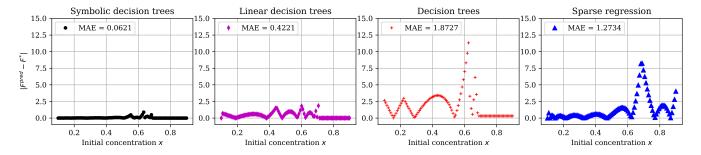


Fig. 4: Absolute error on the testing set for the proposed and baseline models

3662 constraints and 1615 variables (363 binary). We set $\lambda_c = \lambda_m = 10^{-2}$ and solve the problem using Gurobi v.11.0.3 [30]. The problem is solved in 3.4 seconds, and the value of the objective function is $3.2 ext{ } 10^{-4}$. The splitting logic of the learned tree is presented in Fig. 2. The tree uses three splits at 0.62, 0.48, and 0.69, the prediction on the testing set is presented in Fig. 3, and the coefficients of the basis function are presented in Table I. From these results, we observe that the learned model partitions the concentration domain (state of the system) into four segments with different nonlinear functional forms. Specifically, the model puts all the data with $x \ge 0.69$ in node 7, since the label, i.e., input flowrate, is constant and equal to 75 L/min. However, we note that the symbolic nonlinear expression for node 7 is not a constant. This is due to the presence of multiple symmetric optimal solutions as well as the usage of the weighting factors. For the other regions the resulting expression is a combination of the basis functions.

C. Comparison with baselines

We compare the predictive accuracy of the proposed symbolic decision trees with three baselines. The first is a model where $y=\sum_{k\in\mathcal{K}}c_k\phi_k(x)$. The parameters are determined by solving the regression problem that minimizes the absolute mean error to optimality using Gurobi. We refer to this model as the Sparse Regression model. The second model is the classic decision tree with constant predictions at the leaves. This model is trained using Skikit-learn with the maximum tree depth set equal to two. The third model is a decision tree with linear predictors at the leaf nodes, with the maximum depth set equal to two. This model is trained using the linear-tree python package.

The absolute error on the testing set for each model is presented in Fig. 4. From the results, we observe that the proposed approach has significantly lower prediction error on the testing set compared to the other methods. First, we compare symbolic decision trees with the space regression model. From the results on the testing set, the symbolic trees achieve an order of magnitude lower mean absolute error (MAE), 0.0621 vs. 1.2734. Although both models use the same basis function, the accuracy of symbolic decision trees is two orders of magnitude higher. This difference can be attributed to the fact that the symbolic decision trees can use the basis functions more effectively since the fitting of

these models occurs locally, i.e., the symbolic decision trees exploit the local nonlinear nature of the function.

Also, we compare the symbolic decision trees with the standard and linear decision trees. The major difference between these models lies in the functional form of the expression that is present in the leaves: nonlinear expressions in the proposed approach, constants in the standard decision trees, and linear expressions in the linear decision trees. From the results, we observe that the symbolic decision trees lead to lower MAE compared to both decision tree baselines, 0.0621 vs. 1.8727 compared to standard decision trees and 0.0621 vs. 0.4221 compared to linear decision trees. These results show that although all tree models had the same maximum depth, the accuracy of the symbolic decision trees is higher since they can better approximate the local nonlinearities.

D. Closed loop simulations

We also use the learned control law as a controller and compare its closed-loop control performance against the baseline models and standard MPC. We simulate the closedloop system from an initial condition equal to $0.75 \ mol/L$, and the sampling time is 0.1 minutes. For standard MPC, the prediction and control horizon are equal to 10 minutes, the nonlinear optimization problem solved at each sampling time is presented in Eq. 16. We also compare the proposed model with the other baseline models and the evolution of the concentration with time is presented in Fig. 5. From the closed-loop trajectories, we observe that the symbolic decision tree model has comparable closed-loop performance to the standard model predictive controller since the integral absolute error, defined as IAE = $\sum_{t=1}^{T} |x_t - x^{sp}|$, is similar. Moreover, we observe that the other baseline models show worst closed-loop control performance, since the integral absolute error increases by 89% for the linear decision trees, 32% for the sparse regression model, and only 1.9% for the proposed approach.

Finally, we analyze the computing effort required to compute the control action at each sampling time. The average CPU time is $6.3\ 10^{-2}$ for the standard MPC, $2.27\ 10^{-5}$ for the symbolic decision trees, $7.38\ 10^{-5}$ for the linear decision trees, and $1.58\ 10^{-5}$ for the sparse model. For this case, and for the computing hardware used for the simulations, the presence of nonlinear expressions and splitting logic when

computing the control action does not increase the computing effort significantly.

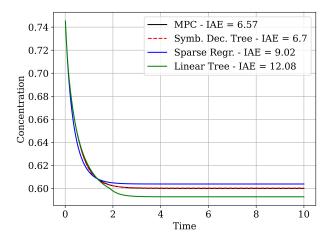


Fig. 5: Closed loop simulation with Model Predictive Control, the proposed learned model, and the baselines.

IV. CONCLUSIONS

In this paper, we proposed symbolic decision trees for learning control laws. The proposed approach automatically learns the partition of the input domain (the state of the system) and nonlinear functions that approximate the data within each domain. The proposed approach offers several advantages over existing approaches. Specifically, the learned model is interpretable since both the branching logic and the equations used to evaluate the control action in each region are symbolic in nature. Application to a reactor case study shows that the proposed method can learn the MPC policy with high accuracy which is also reflected in the closedloop performance of the system when the symbolic decision tree model is used as a controller. These results highlight the potential of learning interpretable piecewise nonlinear control laws using symbolic decision trees equipped with suitable basis functions.

REFERENCES

- [1] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," Automatica, vol. 36, no. 6, pp. 789-814, 2000.
- [2] V. M. Zavala and L. T. Biegler, "The advanced-step nmpc controller: Optimality, stability and robustness," Automatica, vol. 45, no. 1, pp. 86-93, 2009.
- [3] P. Hespanhol, R. Quirynen, and S. Di Cairano, "A structure exploiting branch-and-bound algorithm for mixed-integer model predictive control," in 2019 18th European Control Conference (ECC), pp. 2763-2768, IEEE, 2019.
- [4] L. T. Biegler, "Multi-level optimization strategies for large-scale nonlinear process systems," Comput. Chem. Eng., vol. 185, p. 108657,
- F. Pacaud, M. Schanen, S. Shin, D. A. Maldonado, and M. Anitescu, "Parallel interior-point solver for block-structured nonlinear programs on SIMD/GPU a rehitectures," Optimization Methods and Software, vol. 39, no. 4, pp. 874-897, 2024.
- S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," Proceedings of the national academy of sciences, vol. 113, no. 15, pp. 3932-3937, 2016.

- [7] F. Lejarza and M. Baldea, "Data-driven discovery of the governing equations of dynamical systems via moving horizon optimization,' Scientific reports, vol. 12, no. 1, p. 11836, 2022.
- [8] Z. Wu, P. D. Christofides, W. Wu, Y. Wang, F. Abdullah, A. Alnajdi, and Y. Kadakia, "A tutorial review of machine learning-based model predictive control methods," Reviews in Chemical Engineering, vol. 41, no. 4, pp. 359-400, 2025
- [9] R. Gupta and Q. Zhang, "Data-driven decision-focused surrogate modeling," AIChE Journal, vol. 70, no. 4, p. e18338, 2024
- [10] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," Automatica, vol. 38, no. 1, pp. 3-20, 2002.
- [11] R. Oberdieck and E. N. Pistikopoulos, "Explicit hybrid modelpredictive control: The exact solution," Automatica, vol. 58, pp. 152-159, 2015.
- [12] P. Kumar, J. B. Rawlings, and S. J. Wright, "Industrial, large-scale model predictive control with structured neural networks," Comput. Chem. Eng., vol. 150, p. 107291, 2021.
- [13] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an approximate model predictive controller with guarantees," IEEE Control Syst. Lett., vol. 2, no. 3, pp. 543-548, 2018.
- [14] L. Russo, S. H. Nair, L. Glielmo, and F. Borrelli, "Learning for online mixed-integer model predictive control with parametric optimality certificates," IEEE Control Syst. Lett., vol. 7, pp. 2215 – 2220, 2023.
- B. Karg and S. Lucia, "Deep learning-based embedded mixed-integer model predictive control," in 2018 European Control Conference (ECC), pp. 2075-2080, IEEE, 2018.
- [16] A. Cauligi, A. Chakrabarty, S. Di Cairano, and R. Quirynen, "PRISM: Recurrent neural networks and presolve methods for fast mixed-integer optimal control," in Learning for Dynamics and Control Conference, pp. 34-46, PMLR, 2022.
- R. Schwan, C. N. Jones, and D. Kuhn, "Stability verification of neural network controllers using mixed-integer programming," IEEE Transactions on Automatic Control, vol. 68, no. 12, pp. 7514-7529,
- [18] L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone, Classification and regression trees. Chapman and Hall/CRC, 2017.
- [19] J. R. Quinlan et al., "Learning with continuous classes," in 5th Australian joint conference on artificial intelligence, vol. 92, pp. 343-348, World Scientific, 1992.
- [20] J. Drgoňa, D. Picard, M. Kvasnica, and L. Helsen, "Approximate model predictive building control via machine learning," Applied energy, vol. 218, pp. 199-216, 2018.
- [21] M. Klaučo, J. Drgoňa, M. Kvasnica, and S. Di Cairano, "Building temperature control by simple mpc-like feedback laws learned from closed-loop data," IFAC Proceedings Volumes, vol. 47, no. 3, pp. 581-586, 2014.
- [22] D. Masti, T. Pippia, A. Bemporad, and B. De Schutter, "Learning approximate semi-explicit hybrid mpc with an application to microgrids,' IFAC-PapersOnLine, vol. 53, no. 2, pp. 5207-5212, 2020.
- [23] A. Bemporad, "A piecewise linear regression and classification algorithm with application to learning and model predictive control of hybrid systems," IEEE Transactions on Automatic Control, vol. 68, no. 6, pp. 3194-3209, 2022
- [24] D. Bertsimas, J. Dunn, and Y. Wang, "Near-optimal nonlinear regression trees," Operations Research Letters, vol. 49, no. 2, pp. 201–206, 2021.
- E. M. Sunshine, C. C. Tedesco, S. A. Akhade, M. J. McNenly, J. R. Kitchin, and C. D. Laird, "Hyperplane decision trees as piecewise linear surrogate models for chemical process design," Computers & Chemical Engineering, p. 109204, 2025.
- [26] D. Bertsimas and J. Dunn, "Optimal classification trees," Machine Learning, vol. 106, no. 7, pp. 1039-1082, 2017.
- [27] S. Aghaei, A. Gómez, and P. Vayanos, "Strong optimal classification trees," Operations Research, vol. 73, no. 4, pp. 2223-2241, 2025.
- [28] H. Zhang, A. Zhou, H. Qian, and H. Zhang, "Ps-tree: A piecewise symbolic regression tree," Swarm and Evolutionary Computation, vol. 71, p. 101061, 2022.
- [29] A. Wächter and L. T. Biegler, "On the implementation of an interiorpoint filter line-search algorithm for large-scale nonlinear programming," *Math. Prog.*, no. 1, pp. 25–57, 2006. Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual,"
- 2021. Accessed: 2023-08-31.