Switching Predictive Control Using Reconfigurable State-Based Model

MARAL AMIR, University of California, Irvine
FRANK VAHID, University of California, Riverside
TONY GIVARGIS, University of California, Irvine

Advanced control methodologies have helped the development of modern vehicles that are capable of path planning and path following. For instance, Model Predictive Control (MPC) employs a predictive model to predict the behavior of the physical system for a specific time horizon in the future. An optimization problem is solved to compute optimal control actions while handling model uncertainties and nonlinearities. However, these prediction routines are computationally intensive and the computational overhead grows with the complexity of the model. *Switching MPC* addresses this issue by combining multiple predictive models, each with a different precision granularity. In this artcle, we proposed a novel switching predictive control method based on a model reduction scheme to achieve various model granularities for path following in autonomous vehicles. A state-based model with tunable parameters is proposed to operate as a reconfigurable predictive model of the vehicle. A runtime switching algorithm is presented that selects the best model using machine learning. We employed a metric that formulates the tradeoff between the error and computational savings due to model reduction. Our simulation results show that the use of the predictive model in the switching scheme as opposed to single granularity scheme, yields a 45% decrease in execution time in tradeoff for a small 12% loss in accuracy in prediction of future outputs and no loss of accuracy in tracking the reference trajectory.

CCS Concepts: • **Computing methodologies** → **Modeling methodologies**; *Machine learning*; *Modeling and simulation*; • **Computer systems organization** → *Embedded and cyber-physical systems*;

Additional Key Words and Phrases: Modeling, switching predictive control, neural networks, linear regression, reconfigurable

ACM Reference format:
Maral Amir, Frank Vahid, and Tony Givargis. 2018. Switching Predictive Control Using Reconfigurable State-Based Model. *ACM Trans. Des. Autom. Electron. Syst.* 24, 1, Article 2 (November 2018), 21 pages.
https://doi.org/10.1145/3267126

1 INTRODUCTION AND BACKGROUND

With the recent developments in autonomous driving and the futuristic vision offered by automated vehicles, it has been acknowledged that it is just a matter of time before this technology continues to take over humans in driving autonomous and semi-autonomous vehicles [34].

Authors' addresses: M. Amir and T. Givargis, School of Information and Computer Sciences, University of California, Irvine, 6210 Donald Bren Hall, Irvine CA 92697; emails: {mamir, givargis}@uci.edu; F. Vahid, Department of Computer Science and Engineering, University of California, Riverside, 900 University Avenue, Riverside CA 92501; email: vahid@cs.ucr.edu.

1084-4309/2018/11-ART2 $15.00
https://doi.org/10.1145/3267126

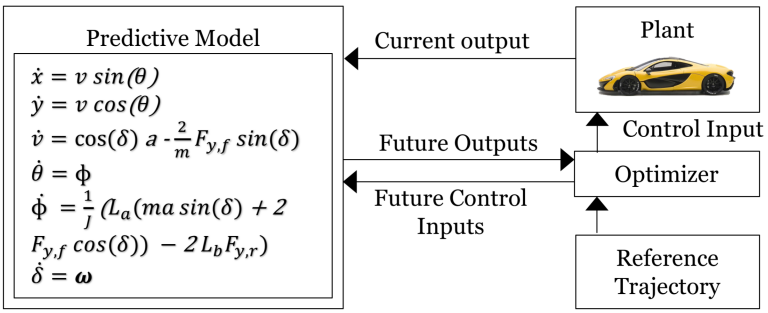


Fig. 1. MPC loop.

Advanced control methodologies have emerged to empower the development of modern vehicles for path planning and path following applications. Nonlinear Model Predictive Control (MPC) is leveraged to develop path following control systems while handling model uncertainties, constraints and nonlinearities. A predictive model of the physical plant is used to estimate the future outputs for a prediction horizon within a window of time and with respect to known input and output values as shown in Figure 1. In general, mathematical descriptions in the form of Ordinary Differential Equations (ODE) are used to mimic the linear/nonlinear behavior of the physical system [30]. An example ODE model of the vehicle dynamics [34] is shown in the figure. ODE solvers are applied to approximate solutions that converge to the exact solution of an equation or system of equations [18]. A runtime optimization routine is evaluated as a parametric quadratic function to calculate the set of future control inputs subject to constraints enforced by the environment and system dynamics. These routines are computationally intensive and for nonlinear physical models, the computational overhead grows with complexity of the model [21]. With respect to structured PID controller, MPC has the ability to handle constraints and changes in system parameters for robust control. Moreover, MPC can be applied to large, multivariable processes. The disadvantage of MPC arise from its strong dependence on the model. However, improvements in data-driven modeling and collection of massive amount of sensor data, this may not be as much of a difficulty.

Complex models of physical systems may be composed of thousands of non-linear ODEs, requiring considerable computing power to execute. These ODE models introduce challenges in terms of scalability, performance, power consumption, and accuracy [12, 25]. Discretization methods (Euler, zero-order hold, etc.) are applied to transform the continuous-time differential equations into discrete-time equivalents, appropriate for numerical computing. The discretized differential equations are solved using numerical algorithms. Iterative solutions are used to solve the non-linear ODE models of physical systems, where a series of linear equations are solved iteratively to converge to the solution for the non-linear ODEs [17]. Therefore, the computation complexity of solving N samples of ordinary differential equations may grow with respect to the type of the discretization algorithm, numerical ODE solver, and the number and order of the ordinary differential equations in the physical model. Moreover, the demand for higher accuracy and more mathematically sound control solutions causes an increase in resource and energy consumption that must be taken into account during the design cycle [1, 23, 24, 32].

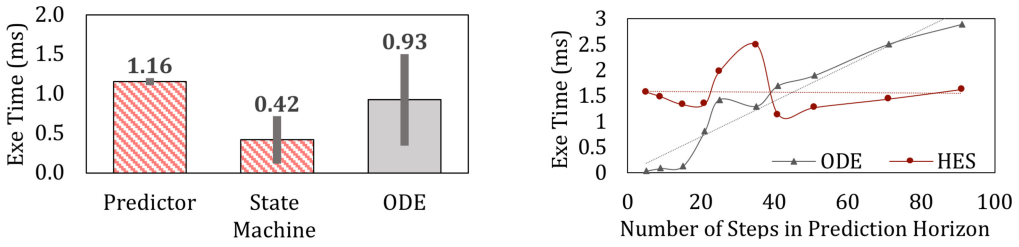
Autonomous behavior in advanced control systems is desirable so they perform well under changing conditions in the physical plant and the environment [2, 3]. Therefore, we proposed a novel switching control methodology to augment the control system to adapt to changes affecting the operating region of the system. In switching predictive control schemes, the controller switches between predictive models of different granularities based on a metric that computes the current dynamic state of the system. An optimal switching control problem consists of (1) a sequence of

switching events, (2) a sequence of modes, and (3) a sequence of control inputs associated with each mode [5]. Switched systems are used to model classes of systems with multi-mode features and switching control schemes may be applied as a solution to address the online computational complexity [34, 36]. We reviewed the state-of-the-art strategies to address the computational overhead specifically in MPC systems in the following section.

2 RELATED WORK

Advanced techniques have been proposed to resolve the MPC computational burden. A common approach to reduce the computational complexity of traditional MPC is the switching MPC [21, 34] methodology. Here, the controller combines the use of predictive models of different granularities in a switching control scheme. The controller switches between the predictive models based on a metric that computes the tradeoff between the error and computational savings due to model reduction. Zhang et al. [34] proposed a binary switching controller based on two coarse-grained and fine-grained predictive models of the vehicle for path planning and path following application. The proposed method considers only two levels of complexity to be included in the MPC application. Gao et al. [11] designed a hierarchical MPC scheme for path planning and path following application to overcome the computational complexity. The high-level controller is formulated to plan an obstacle free path using a simplified-point mass model of the vehicle. Moreover, a low-level controller is designed based on a nonlinear dynamic model of the vehicle to follow the planned path as the reference. More levels of complexities for the physical model enables the MPC to adapt its performance to a wider range of environmental constraints and uncertainties [36]. Jadbabaie et al. [16] suggested that a stable MPC controller requires a sufficiently large prediction horizon. However, short prediction horizons are preferred for improved prediction accuracy of predictive models. This is because harmful effects of the poor estimates are amplified over a long prediction horizon time. Here, the problem is addressed by proposing an MPC approach that uses an adaptive prediction horizon with respect to quality measures [7]. However, the numerical effort needed to solve the optimal control problem for a long prediction horizon still remains significant.

Erlien et al. [8] adopted variable length time-steps in the prediction horizon as a solution to the computational complexity and cost of nonlinear MPCs. This method can associate different prediction horizons targeted for stabilization and collision avoidance tasks. The approach adjusts the sampling time to allow longer prediction horizon in obstacle avoidance steering task as well as short time steps in the prediction horizon for more detailed dynamic behavior prediction. In Reference [33] the prediction horizon is a function of the vehicle speed and the sample time in path following applications. Mahadevan et al. [26] suggested flattening the non-linear differential equation model of the physical system to reduce the computational overhead. For nonlinear ODE systems, flatness is achieved if all the states and input variables can be written in terms of a set of variables—flat outputs and their derivatives. For the dynamic ODE systems that can be recast as a differentially flat system, the runtime optimization problem is reformulated with simpler constraints, and hence smaller computational complexity. Linear Time-Varying (LTV) MPC is a method that employs a model of the physical system linearized along the simulated path at each time step [9]. Even though successful applications of this approach have been presented in the literature, the overhead raised from the linearization over successive time steps is not resolved. Ferreira et al. [10] proposed a methodology to organize libraries of models for electro-hydraulic elements with variations in terms of model complexity. The purpose of the work is to use the appropriate model with regards to the kind of physical domain and the timing requirements for platform. Specifically, different levels of complexities for the target physical system under test shall be provided by the user for a specific application. Here, the sudden changes to the physical system caused by the environment may not be considered at runtime.



(a) Mean and variance of execution time.

(b) Performance of ODE vs. HES [3].

Fig. 2. Comparison of execution time for ODE and HES models.

In general, the main burden in managing the computational complexity of nonlinear MPC applications is the concurrent solving of a large number of nonlinear ordinary differential equations. To overcome this computational overhead, we proposed a more general approach that integrates all the above outlined techniques while eliminating the limitations associated with ODE models. The following motivational case study will discuss this further.

2.1 Motivational Case Study

The work in Reference [3] presented a framework to generate Harmonic Equivalent State (HES) machine, a state-based model of the physical system. HES is applied as the predictive model in the MPC loop to estimate the behavior of the physical system at future time instants based on the calculated future control inputs. The proposed framework uses the Fast Fourier Transform (FFT) decomposition and synthesis functions to generate a reconfigurable model of various granularities. The granularity is adjusted based on the tradeoff between model accuracy and computation time. A machine-learning model is trained to estimate the dynamic behavior of the target physical system. MPC simulation is conducted with ODE model of the physical system to collect the training dataset. A Neural Network (NN) model fits the relation between the future control inputs and harmonic frequency information of the predicted outputs in prediction horizon of size T. Then, a state machine generation algorithm uses the harmonic frequency information to produce a reconfigurable representation of the model in the form of concurrent state machines. Each concurrent state machine is executing at the rate of one of the harmonic frequencies to generate a square-wave output. Tuning parameters are provided to reconfigure the model and tune it for the desired execution time and granularity level. A band-pass filter is used to translate the generated square-waves to sinusoidal equivalents. Finally, the sine-wave harmonics are integrated into the final output signal. Figure 2 compares the execution time of HES model of a vehicle in comparison with an ODE-based predictive model for MPC in a path following application. The performance of the models is evaluated over different prediction horizon sizes for a constant time step. To further analyze the performance of the proposed HES in comparison with the ODE model, we computed the execution time of the HES model as the sum of its two main components: the Harmonic Predictor block and State Machine Generator block as shown in Figure 2(a). The wide bars represent the mean of execution time for each component with respect to changes in the prediction horizon size. The results in the figure indicate that the mean of execution time for the ODE model is 2 times more than the State Machine Generator block, and the Harmonic Predictor block has the highest mean value for the execution time.

To evaluate the performance sensitivity to the size of the prediction horizon, we also computed the variance of execution time for each component shown as in narrow bars. The interesting observation here is that even though the Harmonic Predictor block has the highest mean of execution time, it has a very small variance as opposed to other components and ODE has the highest

variance. To be exact, the variance of ODE is 4 times higher than the variance of the Harmonic Predictor block and 2.5 times higher than the state machine component. That is, the small variations in execution time of the Harmonic Predictor block occurs for different values of prediction horizon size. However, the performance of the ODE model varies more drastically for different prediction horizon sizes. Therefore, it can be concluded that for larger values of prediction horizon, known as long prediction horizon problems, the HES model can outperform the ODE model in terms of execution time. The execution time for both ODE and HES models are compared in Figure 2(b) with respect to the common parameter, the prediction horizon size. The dotted trendlines represent linear changes in the execution time for different values of prediction horizon size. The results show that the HES model outperforms the ODE equivalent with 32% improvement in performance for large prediction horizon. The improvement of performance is in a tradeoff for a minor loss in accuracy for applications that are error tolerant [3].

Conclusion from the observations: The HES model holds promising properties to be employed as the predictive model in novel control methodologies. The model can be reconfigured into various levels of granularities at runtime to be employed as the predictive model in switching MPC approaches. New features may be added to the model to enable runtime reconfiguration subject to current state of the system. Our observations indicate that the error is mostly caused by the filter and the challenges associated with automatic tuning of the filter per harmonic component. Therefore, an alternative solution to replace the filter in the proposed framework is preferred.

2.2 Main Contributions

Based on existing literature and the motivational example described in Section 2.1, we proposed a computationally efficient MPC methodology. Our contributions in this work can be summarized as follows:

(1) The performance of the HES model is improved in terms of execution time and model accuracy. The Neural Network model is modified to better estimate the dynamic behavior of the physical system. Furthermore, the band-pass filter is eliminated and a Look-Up Table (LUT) is included to generate sinusoidal signals.
(2) A novel switching model predictive control methodology is proposed based on the reconfigurable HES model as the predictive model.
(3) Machine-learning techniques are employed to design a runtime switching algorithm that determines the optimal granularity level of the current predictive model in use.
(4) Simulation experiments are conducted to evaluate the switching controller in a path following application containing curved and straight routes.

The rest of the artcle is organized as follows. In Section 3, the high-level architecture of the proposed switching predictive controller is described. The HES model is defined in Section 4. The proposed switching algorithm is described in Section 5. We demonstrated the workings and effectiveness of our framework for path following application in Section 6. Finally, we stated our conclusions in Section 8.

3 SWITCHING MODEL PREDICTIVE CONTROL

In general, a switching MPC system can be defined as a family of sub systems and a rule that orchestrates the switching among these subsystems as shown in Figure 3. The switching function can be classified into state dependent or time dependent based on the function that governs the switching rule [35]. In state-dependent switching, the state space is partitioned into several operating regions and the switching occurs when the system state reaches a certain switching surface.

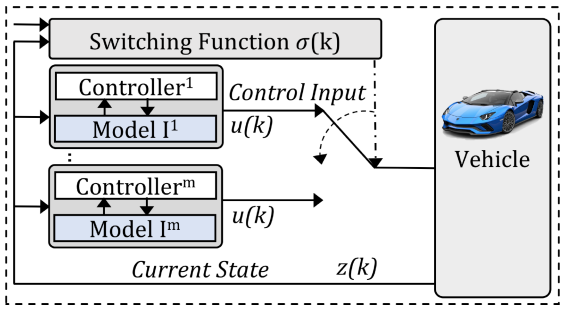


Fig. 3. General switching model predictive control architecture.

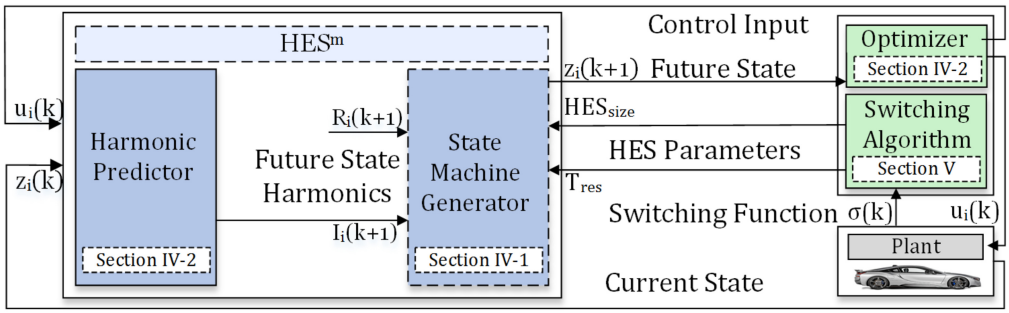


Fig. 4. Switching model predictive control loop with HES as the predictive model.

The switching system is defined as time dependent, when a constant function of time decides the switch among models. The discrete-time linear switched system can be formulated as [35]:

$$\mathbf{z(k+1)} = A_\sigma \mathbf{z(k)} + B_\sigma \mathbf{u(k)}, \tag{1a}$$

$$\mathbf{y(k)} = C_\sigma \mathbf{z(k)}, \tag{1b}$$

where $\mathbf{z}$ is the state vector, $\mathbf{u}$ is the input vector, and $\mathbf{y}$ is the output vector. At any time instant k, the switching function σ formulated in Equation (2) may be dependent on time, its past values, the state/output vectors, and external signal and takes its value from $I^m = 1, \ldots, M$, where M is the number of subsystems.

$$\sigma(k_i) = \Phi([k_0, k_N), \sigma([k_0, k_N)), z([k_0, k_N))/y([k_0, k_N)) \qquad i \in 0, \ldots, N. \tag{2}$$

We proposed a state-dependent switching predictive control system based on HES models as the predictive model. The switching algorithm monitors the current dynamic state of the system and changes the configuration of the HES as a reconfigurable predictive model at runtime. The high-level architecture of the proposed switching predictive control methodology is illustrated in Figure 4. MPC employs a predictive model to compute, at each sampling step, an optimal control problem over a finite prediction horizon. In switching predictive control, the controller selects among a library of predictive models with different levels of granularity based on a switching function σ that considers the control performance tradeoff [34]. The predictive model in a discrete time domain can be expressed as:

$$\mathbf{z_i(k+n|k)} = f^m(\mathbf{z_i(k|k)}, \mathbf{u_i(k+n|k)}), \tag{3}$$

where n is the number of time steps in the prediction horizon T and i is the index for number of variables. The notation $\mathbf{z_i(k+n|k)}$ refers to the value of the state variable z_i in time instant $k+n$,

estimated at time k. The index $m \in 1, \ldots, M$ denotes the level of granularity for the predictive model currently in use. As depicted in Figure 4, we employed the reconfigurable predictive model HES^m to estimate the state vector variables of the prediction horizon T. The HES model is described in Section 4. A switching algorithm is designed as a part of the controller to compute the optimal tuning parameters of a HES model based on the switching function σ, which is a function of state variables as described in Reference [34]. This switching approach is elaborated in Section 5.

4 HES MODEL AS RECONFIGURABLE PREDICTIVE MODEL

The three main merits of the HES model that makes it a valid candidate for a predictive model in switching predictive control are as follows:

- **Adaptive:** The adaptive term means that it can adapt to the behavior of real physical systems for different inputs. The model incorporates machine-learning blocks. This empowers the HES model to be adaptive in runtime control applications in that it can fit the relation between any features and targets with proper training. The neural network model implemented in Section 4.2 and the experiments illustrated in Section 6 validate the working of machine-learning models to estimate dynamic behavior of physical systems.
- **Reconfigurable:** Rather than designing a library of models, one HES model can be reconfigured for different levels of granularities at runtime. The tuning parameters introduced in Section 4.1 adopt this reconfigurability feature to the model.
- **Computationally Efficient and Handling Model Uncertainty:** HES model can generate multiple outputs as time-series data. MPC employs a dynamic model of the physical system to predict the future outputs in a determined prediction horizon. The HES model can advantage MPC application in that the future outputs in the specified prediction horizon are generated all at once. This is as opposed to the ODE predictive models, which are generally required to be solved iteratively to estimate future outputs in a certain prediction horizon. Moreover, the uncertainty is also handled by HES model as poor estimates are not accumulated over a long prediction horizon. This is as opposed to iterative methods that suffer from amplified noise in long prediction horizons.

The high-level architecture of the reconfigurable HES model in the MPC loop is shown in Figure 4. The model is composed of two main blocks: State Machine Generator and Harmonic Predictor. The architecture of HES model is based on the concept of signal decomposition and synthesis to generate a reconfigurable state machine model of a target physical system. The process of calculating the frequency domain information of the signal from time domain representation is called decomposition and the inverse process is signal synthesis. The **State Machine Generator** block captures the harmonic components of the output signal in a prediction horizon T provided by the **Harmonic Predictor**. These harmonic components are integrated into the synthesis function and the future state vector variables are computed as time-series data. The granularity level of the final output is determined by the switching algorithm during the integration process.

4.1 State Machine Generator Block

This block captures the harmonic components from the Harmonic Predictor block and the tuning parameters from the switching algorithm to synthesize the future state vector variables $\mathbf{z_i(k+n|k)}$ in the form of $(N/2+1)$ concurrent state machines. These state machines are updated at the rate of frequency harmonics $\mathbf{Fr^z}$. The synthesis equation of FFT for signal $\mathbf{z_i(n)}$ of size N is employed as presented in Equation (4). In this equation, n stands for the index of samples running from 0 to N-1. The vectors $\overline{\mathbf{R^z}}\mathbf{[i]}$ and $\overline{\mathbf{I^z}}\mathbf{[i]}$ are the normalized frequency spectrum coefficients for the sine

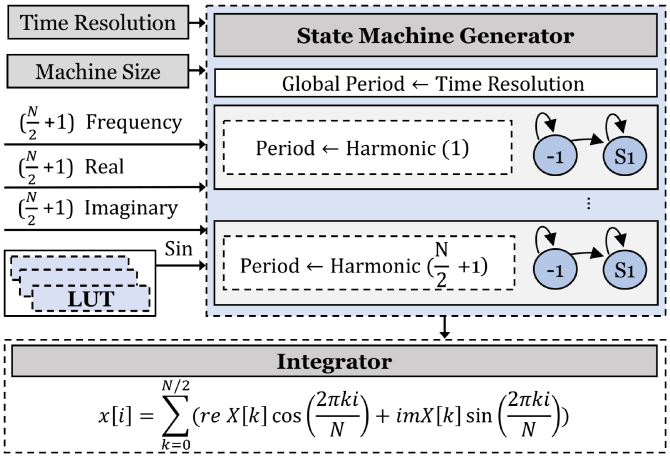


Fig. 5. Concurrent state machine architecture.

and cosine waves with index i running from 0 to $N/2$ for the respective harmonic frequencies [29],

$$z[n] = \sum_{i=0}^{N/2} \overline{\mathbf{R}^{\mathbf{z}}}[\mathbf{i}] cos(2\pi in/N) + \sum_{i=0}^{N/2} \overline{\mathbf{I}^{\mathbf{z}}}[\mathbf{i}] sin(2\pi in/N). \quad (4)$$

The State Machine Generator block is established based on this synthesis Equation (4) to generate the reconfigurable representation of the output signal for models with various levels of granularity. A lookup table (LUT) is employed to collect the sinusoidal values for this equation. The use of LUT has improved the performance of the block drastically. The level of the granularity for the generated signal can be adjusted with respect to following model parameters [2].

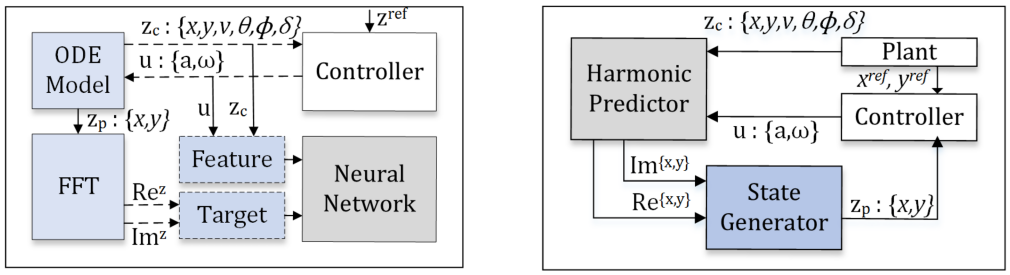
Machine Size (HES_{size}): defines the number of harmonic concurrent state machines to be integrated in the synthesis Equation (4) ranging from 1 to ($N/2$+1).

Time Resolution (T_{res}): is the global period at which rate the generated state machine will be executed.

Figure 5 illustrates the concurrent state machine architecture for the proposed methodology. HES_{size} concurrent state machines are executed at a global rate T_{res}, which is configured by the switching algorithm. This global clock represents the time resolution of the state machine. The outputs of these state machines are integrated into the synthesis equation (4) to compute the future state vector variables $\mathbf{z(k+n|k)}$ in the form of time-series data. The HES model as the predictive model of the physical system expressed in Equation (3) should compute future state variables $\mathbf{z(k+n|k)}$ as a function of current state variables $\mathbf{z(k|k)}$ and future control inputs $\mathbf{u(k+n|k)}$. Therefore, the Harmonic Predictor block is designed to fit these variables to a function of machine-learning model as described in the following section.

4.2 Harmonic Predictor Block

Neural Networks (NN) are capable of solving complex nonlinear relations between the input features and target outputs [4, 27, 31]. Classic NNs have a three layer structure, namely, input, hidden, and output layers. Each layer contains a set of nodes with edges to pass forward the information. Each node carries an activation function, e.g., sigmoid, that limits the variation to output values with respect to changes in NN parameters. The edges entering the nodes are associated with weights that are factors to inputs of the nodes—these weights are selected in the neural network



(a) Training. (b) Prediction.

Fig. 6. Training and prediction for Harmonic Predictor block [3].

framework using a training algorithm that minimizes a cost function. We applied neural networks to design the proposed Harmonic Predictor block. This block contributes the most to estimate the dynamic behavior of the physical system as in Equation (3). The input features of the NN model are control input $\mathbf{u_i(k+n|k)}$ and current state $\mathbf{z_i(k|k)}$ vector variables concatenated, respectively. The target outputs are real and imaginary—$\mathbf{Re_i(k+n|k)}$ and $\mathbf{Im_i(k+n|k)}$—components of state vector variables $\mathbf{z_i(k+n|k)}$ in the next n time steps.

To better represent the behavior of the real physical system, we modified this block in [3] to accept all the current state variables in addition to control inputs as the additional features to the NN model. Moreover, we increased the number of nodes in the hidden layer to mean of input features and target outputs sizes according to an empirically derived rules-of-thumb [14]. This is to fit a more complex pattern and improve the accuracy that comes as a tradeoff for more computational overhead. However, the HES model with better execution time have space for more complex NN with better accuracy. This is due to the replacement of the filter with the lookup table that not only enhances the accuracy but also saves computation time. The results in Section 6.2 evaluate the performance of these two architectures. The Harmonic Predictor block is employed in the following training and prediction steps:

1. Training: The process of training the NN is performed in two phases. First, the architecture of NN is determined with respect to the number of hidden layers, hidden neurons and layer types (e.g., Fully-connected). This part of the design of the architecture is usually done empirically. We employed all fully-connected layers with one hidden layer for our architecture. Once the architecture is defined, a training algorithm is employed to adjust the weight values until the NN reaches the performance objective. The weight adjustment is frequently done using the back-propagation algorithm or some extension of it [22]. For our training algorithm, we used the Damped Least-Squares (DLS), which is a combination of Gradient Descent and Gauss-Newton methods [28]. This algorithm is initially designed as a numerical method to minimize computing sums of squares of nonlinear functions. It also benefits the neural network training, where the performance metric is the mean squared error. To collect the input features and target output values for the training datasets, an offline simulation of MPC application is conducted with ODE model of the physical system as shown in Figure 6(a). That is, the NN model aims to estimate the behavior of the ODE equivalent. Therefore, this ODE model determines the maximum level of granularity available in the proposed HES model. We assume that mathematical models are well designed to accurately capture the dynamic behavior of real physical systems. Here, the proposed method is described and evaluated in MPC for path following of autonomous vehicle application. It needs to be noted that the proposed methodology is generic to all MPC applications. The ODE model of the vehicle dynamics [34] as shown in Figure 7 is formulated as

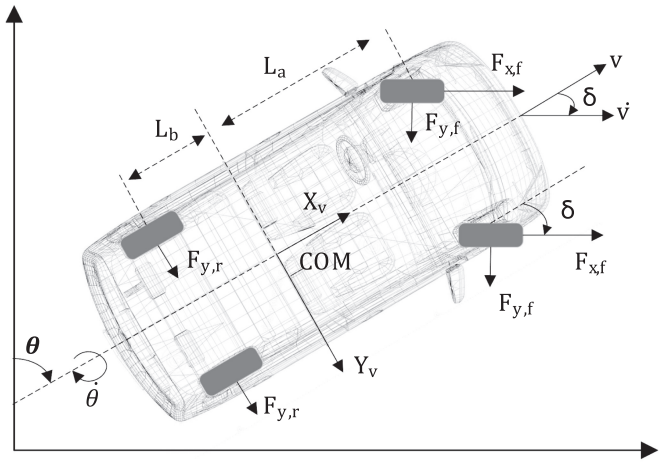


Fig. 7. Schematic view of the vehicle model [3].

$$\dot{x} = v\sin(\theta), \tag{5a}$$

$$\dot{y} = v\cos(\theta), \tag{5b}$$

$$\dot{v} = \cos(\delta)a - \frac{2}{m}F_{y,f}\sin(\delta), \tag{5c}$$

$$\dot{\theta} = \phi, \tag{5d}$$

$$\dot{\phi} = \frac{1}{J}(L_a(ma\sin(\delta) + 2F_{y,f}\cos(\delta)) - 2L_bF_{y,r}), \tag{5e}$$

$$\dot{\delta} = \omega, \tag{5f}$$

where x and y are longitudinal and lateral positions, v and a are longitudinal velocity and acceleration, θ is the yaw angle, and ϕ is the yaw rate. The variables δ and ω represent the steering angle and angular speed, respectively. The variables L_a and L_b are the distance of sprung mass center of gravity from the front and rear axles respectively, and J is the angular momentum. The variables $F_{y,f}$ and $F_{y,r}$ stand for front and rear tire lateral force. More details regarding the model may be found in References [3, 34].

These forces are computed from the following equations:

$$F_{y,f} = C_y\left(\delta - \frac{L_a\phi}{v}\right), \tag{6}$$

$$F_{y,r} = C_y\left(\frac{L_b\phi}{v}\right), \tag{7}$$

where C_y is lateral tire stiffness. We applied real-world parameters of 2011 Ford Fusion as $L_a = L_b = 1.5$m, mass $m = 1700$kg and tire stiffness data for our experiments. The MPC formulation to follow the reference trajectory x^r, y^r is the solution to the following optimization problem:

$$\min_{x,y} \sum_{t=0}^{T_p} \|\hat{x}(k+1|k) - x^r(k+1|k)\|^2_{Q_c}, \tag{8a}$$

$$+\|\hat{y}(k+1|k) - y^r(k+1|k)\|^2_{Q_c}, \tag{8b}$$

$$s.t.$$

$$-0.75 \leq \delta \leq 0.75, \tag{8c}$$

$$-3 \leq \omega \leq 3, \tag{8d}$$

$$-40 \leq a \leq 40, \tag{8e}$$

MPC is simulated to optimize the control input vector variables $\mathbf{u_i}(\mathbf{k}+\mathbf{n}|\mathbf{k})$ for the prediction horizon T with respect to the cost function and enforced constraints. These control input vector variables are fed as the feature values to the NN model. To be a better representation of Equation (3), we included the current state vector variables $\mathbf{z_c}(\mathbf{k}|\mathbf{k})$ from the actual plant as additional features to the NN model. Therefore, for the ODE example formulated in Equation (5), we considered the acceleration and steering angular speed as our control input variables $u_i \in \{a, \omega\}$ to predict future states $\mathbf{z_p}(\mathbf{k}+\mathbf{n}|\mathbf{k}) \in \{x, y\}$. For current state vector variables $\mathbf{z_c}(\mathbf{k}|\mathbf{k})$, we employed all the state variables from Equation (5) as $\mathbf{z_c} \in \{x, y, v, \theta, \phi, \delta\}$.

Next, the State Machine Generator block, accepts only frequency domain components as the input. Therefore, the target outputs of the NN model should be the $\mathbf{Re_i}(\mathbf{k}+\mathbf{n}|\mathbf{k})$ and $\mathbf{Im_i}(\mathbf{k}+\mathbf{n}|\mathbf{k})$ as the frequency information of state vector variables $\mathbf{z_i}(\mathbf{k}+\mathbf{n}|\mathbf{k})$ in the next n time steps. Recall that the FFT algorithm on a sample signal of size N decomposes the signal into real and imaginary components of size $(N/2+1)$. Therefore, here the predicted state vector variables $\mathbf{z_i}(\mathbf{k}+\mathbf{n}|\mathbf{n})$ from simulation of ODE are fed into FFT algorithm in time windows of T to derive the frequency information. The training is performed offline and adds no additional computational complexity at runtime to the application.

2. Prediction: The mapping function that is established during the training phase where the NN learns to correctly associate input patterns to output patterns is automatically retrieved during runtime prediction. Therefore, runtime control input vector variables $u_i \in \{a, \omega\}$ in the next n time steps and current state variables $\mathbf{z_i} \in \{x, y, v, \theta, \phi, \delta\}$ are fed into the NN predictor as shown in Figure 6(b) and the harmonic components of the future output vector variables—$\mathbf{Re_i}(\mathbf{k}+\mathbf{n}|\mathbf{k})$ and $\mathbf{Im_i}(\mathbf{k}+\mathbf{n}|\mathbf{k})$—are estimated. The predicted harmonic information is fed into the State Machine Generator block for output generation—that is, the output of the proposed physical model $\mathbf{z_i}(\mathbf{k}+\mathbf{n}|\mathbf{k}) \in \{x, y\}$ can adapt to variations in control inputs $\mathbf{u_i}(\mathbf{k}+\mathbf{n}|\mathbf{k})$ at runtime as in Equation (3).

Here, we must assume that the predictive model given as in Equation (5) is suitable for the plant under control. Our approach is neither intended to stabilize systems with a large model mismatch or guarantee if switching is sufficiently slow. Here, we assume that selecting a model will not drive the system to a point of instability since otherwise that model would not be selected a suitable predictive model for our MPC controller.

5 SWITCHING ALGORITHM

We designed a runtime switching algorithm based on the HES model mentioned above. The purpose here, is to choose values for the tuning parameters of HES model that *reconfigures* the model for the desired optimal granularity-level *in runtime*. Research shows that, the optimal granularity level for the predictive model in MPC applications varies based on a metric that formulates the tradeoff between the error and computational savings due to model reduction [34]. This metric can be associated with state variables of the physical system as in Equation (2). For instance, the work in Reference [6] proposed a multi-model switching predictive control strategy that employs the speed variable to schedule the switching rules of the controller. Accordingly, in switching predictive control application, the dynamic state of the system may be monitored to select the optimal granularity level for the predictive model. For that, we formulated two switching functions: σ_{state} and σ_{opt}. The former that we call the state metric is formulated as a function of dynamic state of

the system. The latter, optimal granularity metric, is to coordinate the tradeoff between the error and computation time for optimal configuration of HES model.

We used the following function of *steering angle* δ and *velocity* v from Reference [34] as our switching function σ_{state} to compute the dynamic state of autonomous vehicles for $\mathbf{z_i(k|k)} \in \{v, \delta\}$:

$$\sigma_{state}(k|k) = \Phi\left(\mathbf{z_i(k|k)}\right) = v(k) - c\left|\frac{1}{\delta}\right|. \tag{9}$$

The optimal granularity metric σ_{opt} is defined as the ratio of execution time e to model divergence d. Model divergence captures the error between the current model and the model with highest level of granularity,

$$\sigma_{opt}(k|k) = \Phi\left(e, d\right) = \frac{e}{d}. \tag{10}$$

The current dynamic state of the system $z_i(k|k)$ at time instant k defines the range for σ_{state} switching function. Moreover, the range for σ_{opt} switching function is defined by the available granularity levels $m \in 1, \ldots, M$ for the predictive models HES^m that are determined by the its tuning parameters. The switching algorithm computes the current dynamic state of the physical system from σ_{state} and associates this value to a range for σ_{opt} as in Equation (11). That is, the switching algorithm maps the current dynamic state of the system to an optimal granularity level for efficient performance throughout the reference trajectory. The switching algorithm determines the parameter values for HES model with respect to the computed optimal granularity level,

$$\sigma_{opt}(k|k) = \alpha \times \sigma_{\mathbf{state}}(\mathbf{k|k}) + \beta. \tag{11}$$

The parameters α and β are adjusted to map the values of state-dependent switching function σ_{state} to the range for optimal granularity metric σ_{opt}. This mapping enables HES model configuration based on the current dynamic state of the system for efficient performance. The value for parameters α and β varies based on the form of the reference path and the number of desired granularity levels. To compute the values for α and β, Equation (9) is employed to approximate the range for current dynamic state throughout the reference path $r = [\ x^r\ ,\ y^r\ ]$. The value for σ_{state} defines higher optimal granularity levels for large steering angles and small velocity values for the vehicle. However, the optimal granularity level, decreases with smaller values for steering angle and larger velocities. This relation can associate the optimal granularity level of the predictive model with the reference path's *degree of curvature*. Research shows that the optimal granularity level needed for curved path where the vehicle is driving with slower speed and larger steering angle value is higher than in straight routes [34].

We used the following equations to estimate the values of velocity v and steering angle δ to travel the target reference path in distance intervals of Δs meters:

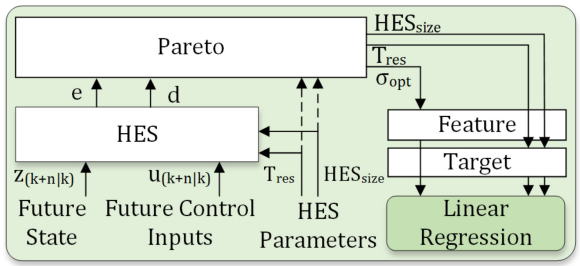
$$\Delta s^r = \sqrt{\Delta x_r^2 + \Delta y_r^2}, \tag{12}$$

$$v^r = \frac{\Delta s^r}{\Delta t^r}, \tag{13}$$

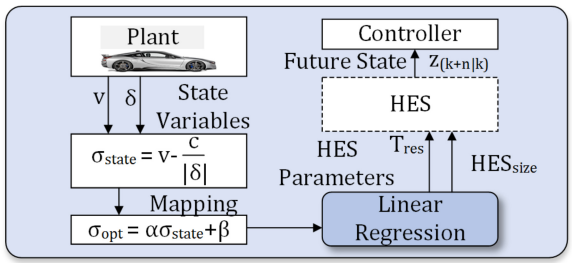
$$\theta^r = \arctan\frac{\Delta x^r}{\Delta y^r}, \tag{14}$$

$$\delta^r = \Delta\theta^r. \tag{15}$$

These values are employed in Equation (9) to approximate the vector σ_{state} for the reference path r. To collect values for optimal granularity level as in σ_{opt} , the MPC simulation for path following application is conducted for t seconds with ODE model of a vehicle as the predictive model. The predicted output values $\mathbf{z_i(k+n|k)} \in \{x, y\}$ are recorded for prediction horizon of size T each representing a vector of size n for n is the number of steps in the prediction horizon.



(a) Training.



(b) Prediction.

Fig. 8. Training and prediction for the switching algorithm.

Then, these vector values are fed to FFT function to generate their harmonic components that are used as inputs to HES model. The HES model is executed for configurations of tuning parameters HES_{size}=[HES_{size}^{min}, HES_{size}^{max}] and T_{res}=[T_{res}^{min}, T_{res}^{max}] dynamically and generates vectors of predicted outputs $\mathbf{z_i(k+n|k)} \in \{x, y\}$. The performance metrics—execution time and model divergence—are recorded for each configuration over different prediction horizons in the simulation time. We computed the mean of these performance metrics throughout the simulation time. Design space exploration is performed to select the Pareto optimal points from the possible pairs of model parameters (HES_{size}, T_{res}) considering the tradeoff between execution time and model error. The ratio of execution time to model divergence for these Pareto optimal points defines the optimal granularity metric σ_{opt}. Now that ample values for σ_{state} and σ_{opt} are collected, the Equation (11) and its respective parameters α and β from can be computed through data-fitting for later usage. The optimal granularity values σ_{opt} and respective pairs of model parameters (HES_{size}^{opt}, T_{res}^{opt}) are later employed as the training data in the proposed machine-learning model.

Machine-learning techniques are applied to predict the tuning parameters of the HES model for the desired optimal granularity level σ_{opt}. We employed linear regression machine-learning models that use $i \in [1, n]$ number of feature values g_i and their respective weights b_i to predict target outputs s_i as in Equation (16). The relation can be fitted on a line using least squares method (LS) that minimizes the sum of the squares of the vertical distance from each data point on the line [13]. The model is implemented in two training and prediction steps as illustrated in Figure 8,

$$s_i = b_0 + b_i g_i. \tag{16}$$

1. Training: The linear regression model is trained to fit the relation between the optimal granularity level σ_{opt} as the input feature and respective HES model parameters—HES_{size} and T_{res}—as the target outputs. The training dataset is collected from the above-mentioned design space exploration experiment that collects corresponding optimal granularity values σ_{opt} and respective pairs of model parameters (HES_{size}^{opt}, T_{res}^{opt}).

2. Prediction: As shown in the switching Algorithm 1, the values for current state variables—velocity v and steering angle δ—are used to calculate the metric σ_{state} from Equation (9) at runtime. These values are captured from the simulation of predictive controller for path following application with current HES model as the predictive model. The value of σ_{state} is inserted in Equation (11) to fit in the range of optimal metric σ_{opt}. Then, the σ_{opt} value is fed to the linear regression model as the input feature to predict the corresponding HES model parameter pair (HES_{size}, T_{res})—that is, the switching algorithm estimates the values for HES model's tuning parameters in that the granularity level of the predictive model is optimal with respect to performance metrics.

ALGORITHM 1: Switching Algorithm for MPC

Input: Current State Variables z
Output: Estimated (HES_{size}, T_{res})

1 **define** α, β ▷ Equation (11)
2 **define** b_0, b_1 ▷ linear regression training function
3 $v = z[0]$ ▷ extract velocity and steering angle
4 $\delta = z[1]$
5 $\sigma_{state} = v - \frac{c}{\delta}$ ▷ calculate current dynamic state
6 $\sigma_{opt} = \alpha\ \sigma_{state} + \beta$ ▷ find optimal granularity
7 $g \leftarrow \sigma_{opt}$
8 $s_1 = b_0 + b_1\ g_1$ ▷ predict using the regression
9 $T_{\text{res}} \leftarrow s[0]$ ▷ extract HES parameters
10 MachineSize $\leftarrow s[1]$
11 **return** $[T_{\text{res}}, \text{MachineSize}]$

6 EXPERIMENTAL RESULTS

6.1 Experimental Setup

Our experiments are performed on a PC with a quad-core Intel Core i7 and 16 GB of DDR3 RAM. The MPC formulation is implemented in software using a framework based on the ACADO Toolkit [15], which is an open source software written in C++ for automatic control and dynamic optimization. It provides a self contained environment to implement control algorithms including MPC as well as state and parameter estimation. The existence of Lyapunov function ensures the stability of autonomous dynamical systems [19]. Therefore, here the so-called LYAPINT integrator in ACADO Toolkit as an explicit Runge-Kutta45 integrator with an appropriate step size control is applied. The State Machine Generator block is implemented using the C++ programming language to enable it to be highly portable and compatible with various platforms for compilation and execution. The Neural Network model is trained by using MATLAB's neural networks module (nftool). The regression model in the switching algorithm is also implemented in MATLAB. Figure 9 illustrates the values for test error of the NN model described in Section 4.2 for 13 simulation experiments. The Neural Network described in Section 4.2 is trained using 266 training batches for 70 features in the input layer and prediction horizon of size $T = 1.55$s. The number of neurons in the hidden and output layers are 60 and 68, respectively. As shown in the figure, the value for the error is in micro range that validates the the performance of the NN described in Section 4.2 to predict the future dynamic behavior of the physical system.

6.2 Comparison to State of the Art

We compared the performance of the HES model described in Section 4 with respect to the ODE model of a vehicle formulated in Equation (5) in a runtime MPC application in path following.

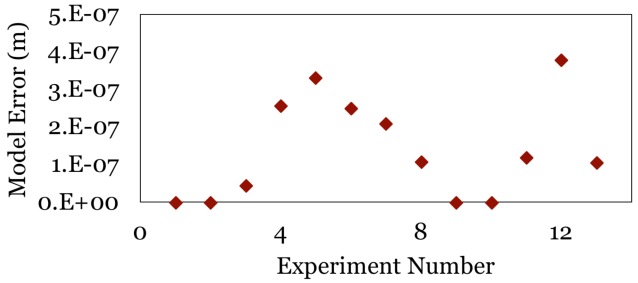


Fig. 9. Test error for neural network model in Harmonic Predictor block.

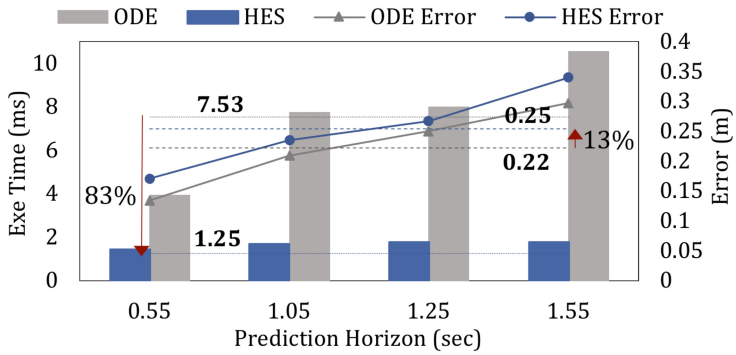


Fig. 10. Performance comparison of ODE and HES models.

Figure 10 illustrates the error and execution time values of MPC using HES and ODE models, simulated for different prediction horizon sizes. As shown in the figure, the mean of execution time for ODE model and HES model are 7.63ms and 1.25ms, respectively. This improvement in performance is gained at the expense of a minor increase in model error from average of 0.22m to 0.25m. The results indicate average of 83% reduction in MPC return time using HES model for negligible 13% loss in model accuracy. The improvement in accuracy and execution time compared with the values reported in Reference [3], is due to the removal of the filter and the use of LUT and a more complex NN model in the new design.

Figure 11 illustrates the simulation results of MPC application in path following using the HES model as the predictive model. Here, the HES model estimates the dynamic behavior of the vehicle for 1.55 (seconds) in the future. The initial velocity of the vehicle is taken as 24 (meters per second) to follow the reference trajectory as shown in the top left. As shown in the figure, the steering control inputs $\{a, \omega\}$ demanded by the controller enables the HES model to track the reference trajectory as the degree of curvature varies.

We conducted simulation experiments to evaluate the performance of the proposed switching predictive control methodology in a runtime path following application of an autonomous vehicle. As described in Section 5, we used a linear regression model in the switching algorithm to predict the parameters of the HES model based on the optimal level of model granularity in need. This optimal granularity level is aligned with the current dynamic state of the system. To compute the α and β coefficient in Equation 11, Equations (12)–(15) based on the the reference trajectory $r = [\, x^r\,, y^r\,]$ are used. Figure 12(a) shows the estimated values of σ_{state} throughout the reference trajectory. Figure 12(b) shows the Pareto optimal points of HES model parameters that are computed from the design space to collect the training data for the regression model. These Pareto optimal points are

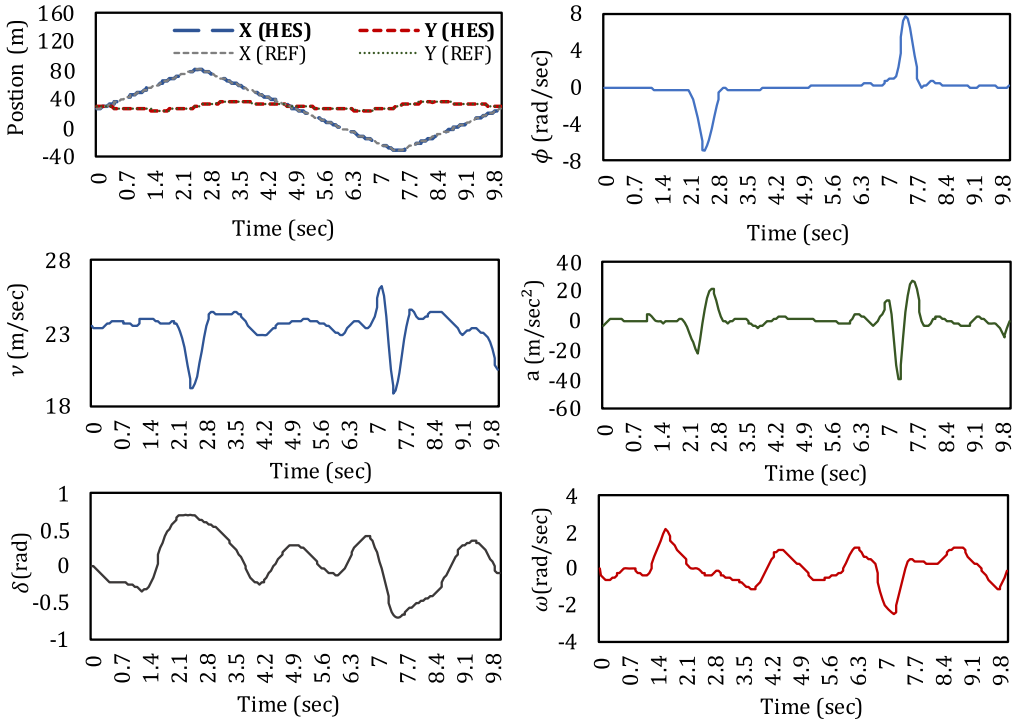
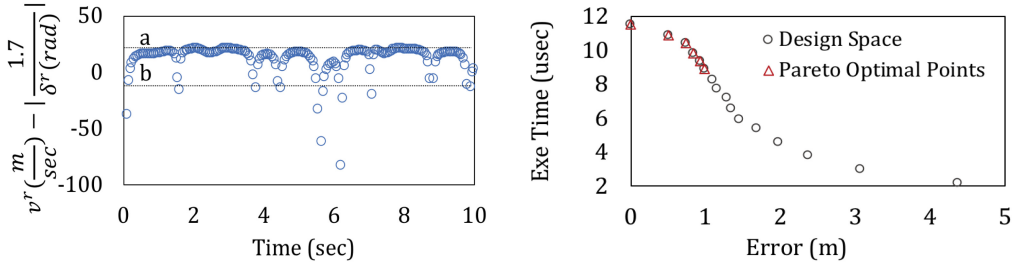


Fig. 11. The simulation results of the MPC using the HES model as the predictive model.



(a) σ_{state} estimation for reference trajectory r.

(b) σ_{opt} is computed as the ratio of execution time to model error for pareto optimal points

Fig. 12. Computing switching functions σ_{state} and σ_{opt}.

associated with pairs of $(HES_{size}, T_{res}) \in (13:17, 0.05)$ for five levels of granularity. We used these values to compute the optimal granularity metric σ_{opt} as the ratio of execution time e to model divergence d.

To compute the parameters α and β in Equation (11), we use the data collected in Figure 12(a) in the following equation.

$$\sigma_{opt} = \begin{cases} 1 & \sigma_{state} > a \\ 0 & \sigma_{state} < b \\ \alpha \times \sigma_{state} + \beta & a \leqslant \sigma_{state} \leqslant b. \end{cases} \tag{17}$$

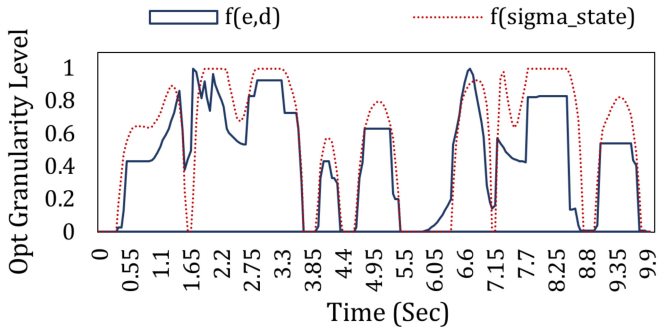


Fig. 13. Comparing σ_{opt} as a function of (e, d) and as a function of σ_{statet}.

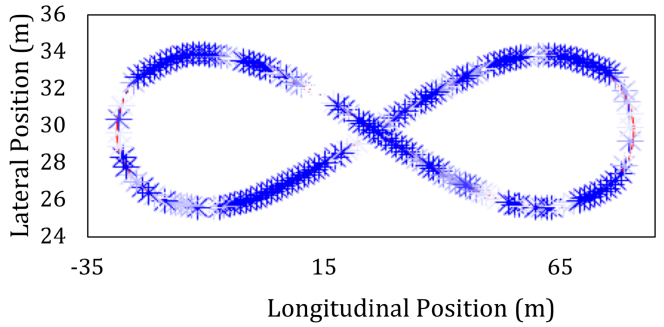


Fig. 14. Switching predictive control application for path tracking. The red line shows the computed position of HES model in the single granularity control mode and the blue star markers represent the computed position for HES model in the switching control mode.

Here, we consider $\alpha = 0.025$ and $\beta = 0.5$. As mentioned in Section 5, the α and β parameters depend on the form of the reference path and the number of desired granularity levels. Figure 13 compares the approximated values of σ_{opt} using Equation (17) and actual values computed using Equation (10). We can further tune the α and β parameters to adjust the over/under estimations shown in the figure.

The linear regression model fits the relation between the optimal granularity level σ_{opt} as the input feature and respective HES model parameters as the target outputs during the training phase.

To better evaluate the effectiveness of the switching controller, we selected the reference path to be a combination of straight and curved routes. For this purpose, we applied the Lemniscate of Bernoulli function to generate our reference path. Figure 14 shows the performance of the proposed switching predictive control methodology for tracking the Bernoulli path. The red line is representing the x, y values for the single granularity control mode and star markers are for switching control mode. The granularity level of the predictive HES model configured by the switching algorithm is associated with the RGB value of the star markers. That is, higher granularity levels are color mapped to higher RGB values, hence, lighter blues. The gradual increase in granularity level (lighter blue) as the vehicle enters the curved route validates the performance of the proposed switching algorithm in selecting the parameters and reconfiguring the HES model appropriately.

The values for the optimal granularity level σ_{opt} calculated from Equation (11) and runtime steering angle δ throughout the simulation of MPC are depicted in Figure 15. The respective HES model parameters—HES_{size} and T_{res}—are predicted by the regression model to adjust the desired

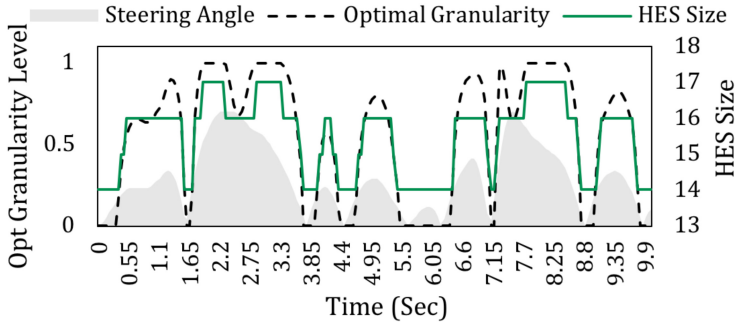
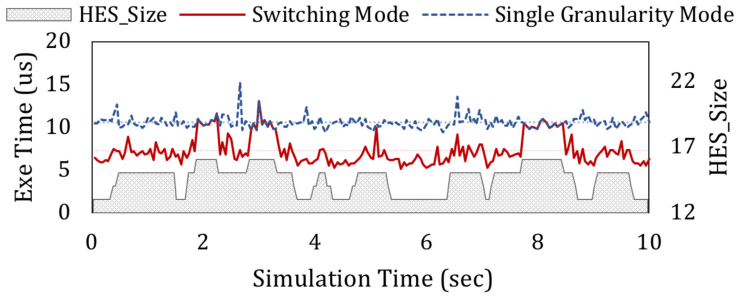
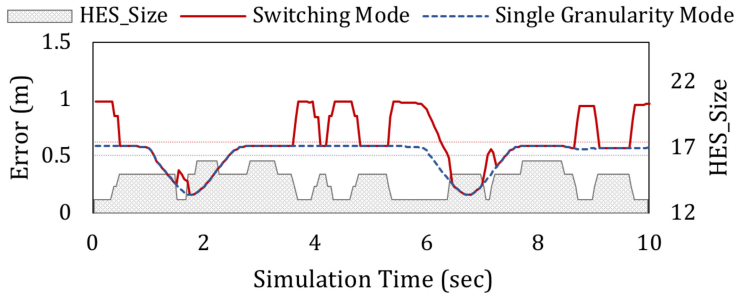


Fig. 15. HES model granularity levels.



(a) Execution time.



(b) Error.

Fig. 16. Performance analysis of HES model in two switching and single granularity modes.

optimal granularity level. The HES parameter T_{res} is set to constant value of 0.05s. Higher values of HES_{size} reconfigures the model for higher granularity levels. The results show the switching of model parameter HES_{size} with respect to the desired σ_{opt} value. As we expected, the optimal granularity level for when the vehicle is driving on a curved route with large steering angle value is higher than when on straight paths.

We compared the performance of the new improved version of the HES model in single granularity and switching modes in Figure 16. The model in the single granularity mode is configured for the highest level of granularity with static parameters (HES_{size}, T_{res})= (17, 0.05) in this example. However, the switching algorithm is employed to reconfigure the HES model's parameters in runtime. Figure 16(a) illustrates the execution time values of the HES model in the switching and single granularity control modes throughout the simulation. These values represent the

performance of the HES models in computation of output vectors $z(k+n|k)$ for the next n steps in the prediction horizon of size T. The execution time values reported for the switching mode are computed as the sum of the HES model's execution time and the overhead caused by the switching process. Since the linear regression model is formulated as a function of one input feature, the additional computational overhead in the switching mode is O(1) that is negligible. As shown in the figure, the mean execution time of MPC through the whole path using the HES model in the switching mode is 45% less than single granularity mode, dropping from 10.52 (μs) to 7.27 (μs). This is due to the presence of the switching algorithm to reconfigure the HES model for optimal performance with respect to dynamic state of the vehicle—that is, the switching algorithm selects higher values for (HES_{size} parameter and reconfigures the model to maintain high execution time and granularity level on a curved route with large steering angle value and vice versa.

Figure 16(b) compares the error values for the HES model in the switching and single granularity control modes throughout the reference path. The results show 0.5 (m) and 0.62 (m) as comparable mean of error values for the HES model, in the single granularity and switching control schemes, respectively. This is because the switching algorithm is designed to reconfigure the HES model for lower levels of accuracy *when tolerated*, as in tracking a straight curve with high velocity and low steering angle values. That is, the changes in the range of error values for HES model corresponds with the optimal granularity level. This range is directly related to curvature of the path. It needs to e noted that, the drop in accuracy is only observed in the generation of future output vectors $z(k+n|k)$. However, the error to track the reference trajectory is comparable between the switching and single granularity modes with no drop in accuracy. Our experiments indicate that the use of HES model in the proposed switching scheme acquires 45% decrease in execution time for no loss in trajectory tracking accuracy. Moreover, our proposed switching control method based on HES model is capable of choosing the optimal model for different velocity and steering angle values.

7 FUTURE WORK

To further improve the accuracy of the NN in the HES model, we can increase the number of training data. Moreover, a more complex ODE model can be adopted during the simulation process to collect the training data. In our future work, we are planning to deploy Recurrent Neural Networks (RNN) as a predictive model to estimate the behavior of the physical system. RNN is a popular architecture for time-series forecasting and is distinguished from feed forward neural networks by having signals traveling in both directions and introducing loops in the network. Furthermore, we are planning to exploit a stable adaptive controller to reject controllers leading to instability [20].

8 CONCLUSION

In this artcle, a novel switching predictive control methodology is proposed that uses model reduction to achieve a desired performance granularity for autonomous vehicles in path following applications. This method is based on a state-based model of the physical system that is able to adjust its granularity level dynamically. We apply machine-learning models to design a switching algorithm. Experimental results show that our proposed switching control method decreases the overall execution time of MPC by 45% for a small 12% loss of accuracy in prediction of future output values and no loss of accuracy in tracking the reference trajectory.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation under NSF grant number 1563652.

REFERENCES

[1] Amir Aminifar, Paulo Tabuada, Petru Eles, and Zebo Peng. 2016. Self-triggered controllers and hard real-time guarantees. In *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*. EDA Consortium, 636–641.

[2] Maral Amir and Tony Givargis. 2017. HES machine: Harmonic equivalent state machine modeling for cyber-physical systems. In *Proceedings of the 2017 IEEE International High Level Design Validation and Test Workshop (HLDVT'17)*. IEEE, 31–38.

[3] Maral Amir and Tony Givargis. 2017. Hybrid state machine model for fast model predictive control: Application to path tracking. In *Proceedings of the 36th International Conference on Computer-Aided Design*. IEEE Press, 185–192.

[4] Maral Amir and Tony Givargis. 2018. Priority neuron: A resource-aware neural network for cyber-physical systems. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* (2018).

[5] Sorin C. Bengea and Raymond A. DeCarlo. 2005. Optimal control of switching systems. *Automatica* 41, 1 (2005), 11–27.

[6] Hua Chen, Li Ning, and Li Shaoyuan. 2011. Switching multi-model predictive control for hypersonic vehicle. In *Proceedings of the 2011 8th Asian Control Conference (ASCC'11)*. IEEE, 677–681.

[7] Greg Droge and Magnus Egerstedt. 2011. Adaptive time horizon optimization in model predictive control. In *Proceedings of the American Control Conference (ACC'11)*. IEEE, 1843–1848.

[8] Stephen M. Erlien, Susumu Fujita, and J. Christian Gerdes. 2013. Safe driving envelopes for shared control of ground vehicles. *IFAC Proc. Vol.* 46, 21 (2013), 831–836.

[9] Paolo Falcone, M. Tufo, Francesco Borrelli, Jahan Asgari, and H. Eric Tseng. 2007. A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems. In *Proceedings of the 2007 46th IEEE Conference on Decision and Control*. IEEE, 2980–2985.

[10] Jorge A. Ferreira, João E. De Oliveira, and Vítor A. Costa. 1999. Modeling of hydraulic systems for hardware-in-the-loop simulation: A methodology proposal. In *Proceedings of the International Mechanical Engineering Congress & Exposition*.

[11] Yiqi Gao, Theresa Lin, Francesco Borrelli, Eric Tseng, and Davor Hrovat. 2010. Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads. In *Proceedings of the ASME 2010 Dynamic Systems and Control Conference*, Vol. 1. American Society of Mechanical Engineers, 265–272.

[12] Alexander N. Gorban and Dirk Roose. 2010. *Coping with Complexity: Model Reduction and Data Analysis*. Vol. 75. Springer Science & Business Media.

[13] Herman O. Hartley. 1961. The modified Gauss-Newton method for the fitting of non-linear regression functions by least squares. *Technometrics* 3, 2 (1961), 269–280.

[14] Jeff Heaton. 2008. *Introduction to Neural Networks with Java*. Heaton Research, Inc.

[15] B. Houska, H. J. Ferreau, and M. Diehl. 2011. ACADO toolkit – An open source framework for automatic control and dynamic optimization. *Opt. Contr. Appl. Methods* 32, 3 (2011), 298–312.

[16] Ali Jadbabaie and John Hauser. 2005. On the stability of receding horizon control with a general terminal cost. *IEEE Trans. Automat. Contr.* 50, 5 (2005), 674–678.

[17] Carl T. Kelley. 1999. *Iterative Methods for Optimization*. SIAM.

[18] Carl T. Kelley. 2003. *Solving Nonlinear Equations with Newton's Method*. SIAM.

[19] S. Mohammad Khansari-Zadeh and Aude Billard. 2014. Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robot. Auton. Syst.* 62, 6 (2014), 752–765.

[20] H. J. Kim and Andrew Y. Ng. 2005. Stable adaptive control with online learning. In *Advances in Neural Information Processing Systems*. 977–984.

[21] Peter Krauthausen and Uwe D. Hanebeck. 2010. A model-predictive switching approach to efficient intention recognition. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'10)*. IEEE, 4908–4913.

[22] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. 1990. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems*. 396–404.

[23] Edward A. Lee. 2008. Cyber physical systems: Design challenges. In *Proceedings of the 2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'08)*. IEEE, 363–369.

[24] Xue Lin, Paul Bogdan, Naehyuck Chang, and Massoud Pedram. 2015. Machine learning-based energy management in a hybrid electric vehicle to minimize total operating cost. In *Proceedings of the 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'15)*. IEEE, 627–634.

[25] Yoram Louzoun, Sorin Solomon, Henri Atlan, and Irun R. Cohen. 2001. Modeling complexity in biology. *Physica A* 297, 1 (2001), 242–252.

[26] R. Mahadevan and F. J. Doyle III. 2003. Efficient optimization approaches to nonlinear model predictive control. *Int. J. Robust Nonlin. Contr.* 13, 3–4 (2003), 309–329.

[27] Hamid Mirzaei and Tony Givargis. 2017. Fine-grained acceleration control for autonomous intersection management using deep reinforcement learning. *IEEE Smart World Congress (SWC'17)*. San Francisco, 1–8.
[28] Jorge J. Moré. 1978. The Levenberg-Marquardt algorithm: Implementation and theory. In *Numerical Analysis*. Springer, 105–116.
[29] Steven W. Smith et al. 1997. The scientist and engineer's guide to digital signal processing (2nd. ed.). California Technical Publishing, San Diego, CA.
[30] Korosh Vatanparvar and Mohammad Abdullah Al Faruque. 2015. Battery lifetime-aware automotive climate control for electric vehicles. In *Proceedings of the 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC'15)*. IEEE, 1–6.
[31] Korosh Vatanparvar, Sina Faezi, Igor Burago, Marco Levorato, and Mohammad Abdullah Al Faruque. 2018. Extended range electric vehicle with driving behavior estimation in energy management. *IEEE Transactions on Smart Grid* 14, 8 (2018).
[32] Korosh Vatanparvar and Mohammad Abdullah Al Faruque. 2017. Electric vehicle optimized charge and drive management. *ACM Trans. Des. Autom. Electr. Syst.* 23, 1 (2017), 3.
[33] Muhammad Aizzat Zakaria, Hairi Zamzuri, Rosbi Mamat, Saiful Amri Mazlan, Mohd Azizi Abd Rahman, and Abdul Hadi Abd Rahman. 2014. Dynamic curvature path tracking control for autonomous vehicle: Experimental results. In *Proceedings of the 2014 International Conference on Connected Vehicles and Expo (ICCVE'14)*. IEEE, 264–269.
[34] Kun Zhang, Jonathan Sprinkle, and Ricardo G. Sanfelice. 2015. Computationally aware control of autonomous vehicles: A hybrid model predictive control approach. *Auton. Robots* 39, 4 (2015), 503–517.
[35] Lixian Zhang, Yanzheng Zhu, Peng Shi, and Qiugang Lu. 2016. *Time-dependent Switched Discrete-time Linear Systems: Control and Filtering*. Springer.
[36] Lixian Zhang, Songlin Zhuang, and Richard D. Braatz. 2016. Switched model predictive control of switched linear systems: Feasibility, stability and robustness. *Automatica* 67 (2016), 8–21.

Received March 2018; revised August 2018; accepted August 2018