Learning based Model Predictive Control (LBMPC)

SUBMITTED IN PARTIAL FULFILLMENT FOR THE REQUIREMENTS
OF THE DEGREE OF

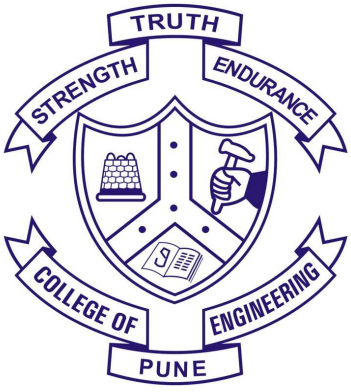
Master of Technology

BY

Saket Adhau
MIS - 121717001

UNDER THE GUIDANCE OF

Dr. D. N. Sonawane

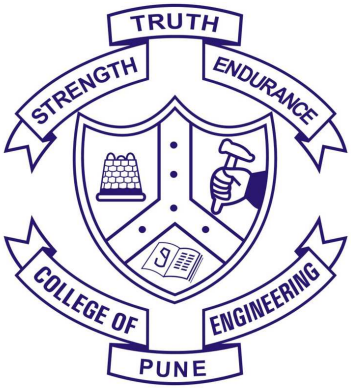


Department of Instrumentation and Control

COLLEGE OF ENGINEERING, PUNE

INDIA, JUNE 2019

CERTIFICATE



This is to certify that the dissertation entitled **Learning based Model Predictive Control** submitted by **Saket Adhau (MIS : 121717001)** in the partial fulfillment of the requirement for the award of degree of Master of Technology (Instrumentation and Control) with specialization in Biomedical Engineering at College of Engineering Pune, affiliated to Savitribai Phule Pune University is a record of his own work.

Dr. Dayaram Sonawane
Guide
Instrumentation and Control
College of Engineering, Pune.

Dr. Dayaram Sonawane
Head of the Department
Instrumentation and Control
College of Engineering, Pune.

Date :

Place : Pune, India.

Approval Sheet

This dissertation entitled

Learning Based Model Predictive Control (LBMPC)

by

Mr. Saket Adhau

(121717001)

is approved for degree of

Master of Technology with specialization in Biomedical Instrumentation

of

Department of Instrumentation and Control

College of Engineering, Pune

(An autonomous institute of Govt. of Maharashtra)

Examiners	**Name**	**Signature**
1. External Examiner		
2. Internal Examiner		
3. Supervisor		

Date:

Place:

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Saket Adhau
(121717001)

Date:
Place:

Acknowledgement

First of all, I would like to express my deepest and sincerest thanks to my guide, Dr. D. N Sonawane for his constant motivation, inspiration and teachings throughout the period of my M. Tech. I thank him for his constant suggestions and insights about the research he provided during this period. I am thankful to him for introducing me to the theory of Model Predictive Control and Optimization.

Apart from this, my sincere gratitude goes to my academic co-supervisor, Dr. Deepak Ingole, Post-Doctoral Researcher from IFFSTAR and University of Lyon. He was the person who convinced me and directed my theis work towards machine learning based methods for MPC. His in-depth knowledge about control systems made wonders. I sincerely hope everyone gets a mentor like him.

I would like to specially thank my lab-mate Ms. Sayli Patil for her help throughout the period of my masters thesis. She was always ready and kind enough to help me in her busy schedule. I would also like to thank all my lab-mates including Mr. Pramod Ubare, Mr. Chaitanya Jugade and all the others. I would like to thank all my classmates and faculty members for their valuable suggestions and motivation.

Finally, I would like express my deepest thanks to my mom and my dad, for all kind of support. They always supported me in all my decisions and encouraged me throughout these two years. This thesis is wholeheartedly dedicated to all of you.

Abstract

Model predictive control (MPC) has emerged as an excellent control strategy owing to its ability to include constraints in the control optimization and robustness to linear as well as highly non-linear systems. There are many challenges in real-time implementation of MPC on embedded devices, including computational complexity, numerical instability, and memory constraints. Advances in machine learning-based approaches have widened the scope to replace the traditional and intractable optimization algorithms with advanced algorithms. This drawback is eliminated by introducing a versatile framework for synthesis of simple, yet well-performing control strategies that mimic the behaviour of optimization-based controllers. This thesis work devises a novel deep learning-based model predictive control (DNN-MPC). The proposed MPC uses recurrent neural network (RNN) to accurately predict the future output states based on the previous training data. Using deep neural networks for the real-time embedded implementation of MPC, on-line optimization is completely eliminated leaving only the evaluation of some linear equations. The main advantage of the proposed methods stems from their easy implementation even on low-level hardware without the need for advanced software libraries. Closed-loop performance evaluation of the DNN-MPC is verified through hardware-in-loop (HIL) co-simulation on ARM microcontroller and a 4x speed-up in computational time for a single iteration is achieved over the conventional MPC. Detailed analysis of DNN-MPC complexity (speed and memory requirement) is presented and compared with traditional MPC. Results show that the proposed DNN-MPC performs faster and with less memory footprints while retaining the controller performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Present Scenarios and Challenges

Recent advances in machine learning-based approaches for optimization and control systems is leading to new paradigms to design and implement real-time embedded MPC for industrial use. Leveraging these machine learning-based methods for control systems, control laws could be extracted using improvised optimization algorithms and feature extraction methods, (Baumeister et al., 2018).

Model predictive control (MPC) has emerged as an excellent controller for complex systems with both, slow and fast dynamics. MPC is much popular as it the solves numerical optimization problem at every sampling instant to calculate a sequence of optimal control inputs over a prediction horizon subjected to system dynamics and constraints, (Yang et al., 2015). This optimization loop is repeated over the complete time period giving optimized outputs at every instant. The above discussed capabilities and robustness of MPC has made its way in industries and academics. However, solving the optimization problem is indeed a computationally expensive task restricting the usage of MPC with systems having slow dynamics. Furthermore, MPC requires an accurate description of the plant model and is susceptible to uncertainties in plant parameter,s (Xie et al., 2011).

For a successful real-time embedded implementation of MPC, the microcontrollers or microprocessors should be able to solve the optimization problem at each sampling instant. This would require an efficient C/C++ code to be deployed on embedded devices, (Adhau et al., 2019a). Methods such as fast MPC (Wang and Boyd, 2009) and alternating directions method of multipliers (ADMM) (Boyd et al., 2011) have been used towards efficient implementation of MPC and optimization on embedded platforms.

Other significant tools such as FORCES PRO (Domahidi and Perez, 2014) and ODYS (Cimini et al., 2017), μAO-MPC (Zometa et al., 2013), CVXGEN (Mattingley and Boyd, 2012), OSQP (Stellato et al., 2018) and jMPC toolbox (Currie, 2011) also generate efficient embedded C/C++ codes.

Apart from efficient embedded optimization solvers, another solution is to use explicit MPC (EMPC), which uses pre-computed control law to search for optimal solutions (Bemporad et al., 2000). EMPC has been used for systems having fast dynamics with sampling times in milliseconds. This control law is stored in the form of a look-up table (LUT) for all possible initial states and point location algorithm is used to efficiently search through LUT's to get an optimized control value for current state. The major challenges in the implementation of EMPC is the number of regions increase exponentially with increase in the prediction horizon and the problem size as shown in (Adhau et al., 2019b). Toolboxes such as multi-parametric toolbox (MPT) (Herceg et al., 2013) and hybrid toolbox (Bemporad, 2004) are used for generating explicit MPC solutions.

Reinforcement learning, deep neural networks, and other statistical learning methods have been consistently used for learning-based MPC (LBMPC). In this paper, we present a machine learning-based approach to train MPC controllers as one of the alternative solution for on-line optimization. Authors in (Negenborn et al., 2004), used reinforcement learning in MPC for markov decision processes which can be used in areas like vehicle automation, traffic control, and logistics. Authors also validated the stability of proposed controller on models such as quadrotor (see, (Bouffard et al., 2012)), for HVAC models (see, (Aswani et al., 2011)), etc. In addition to this, a robust learning model predictive control (RLMPC) for iterative tasks such as automatic racing cars is presented (see, (Rosolia and Borrelli, 2017)). Authors in (Drgoňa et al., 2018), used deep time delay neural networks (TDNN) and regression trees (RT) for MPC. They have significantly reduced the implementation cost and complexity while retaining the controller performance. Learning-based explicit nonlinear MPC has also been proposed in (Trinh et al., 2016).

In past years, efforts have been made to design and implement anesthesia closed-loop control system using numerous control schemes which includes fixed-gain (proportional-integral-derivative controller PID (Padula et al., 2017)), knowledge-based (fuzzy logic (Mendez et al., 2018)), and model-based (MPC (Ingole and Kvasnica, 2015)), etc. which performed well in case of inter-variability patient model, disturbance rejection, and constraints handling. Recently, there has been some works on reinforcement learning (RL)-based anesthesia control see, (Padmanabhan et al., 2015; Moore et al., 2014). Their results show that the RL-based control strategy is promising and robust to system uncertainties.

1.2 Goals and Contribution of the Thesis

The academic goals of the thesis were summarized as follows:

- Comprehensive research in the field of machine learning, and relevance evaluation of the integration of the MPC strategies in modern advanced control.
- Development of efficient MPC strategies, tier algorithmic formulations, analysis and simulation studies on various control problems.
- Experimental validation of developed algorithms using HIL.

The objective of this thesis is to achieve comparable performance as of linear MPC on embedded platforms while replacing the optimization part using deep neural network (DNN) trained controller. The trained controller is supposed to mimic the control law of MPC without any significant performance loss Previously, deep neural networks have been employed for linear, nonlinear, and hybrid MPC (Kumar et al., 2018; Lucia et al., 2018). The first contribution is the investigation of the influence of the controller model accuracy on the performance evaluation. We have rigorously trained the DNN model using large data sets and verified the performance and robustness of the designed model. We consider ARM Cortex M3 for embedded implementation. Features like high-performance architecture, flexible peripheral connectivity, analog support, and low power consumption makes ARM based processor a suitable choice in various industrial applications. It is mainly used in automotive production and automotive control where fast, precise and real-time control is required. As a case study, a single-input, single-output intravenous anesthesia drug delivery model (Ingole and Kvasnica, 2015) has been chosen. We show how this controller, drastically reduces memory footprint and computational complexity and is easily deployable on embedded platforms. To the best of our knowledge, this is the first work that has investigated the performance of deep learning-based MPC for anesthesia control.

Chapter 2

Model Predictive Control

Model predictive control (MPC) belongs to a class of computer control algorithms, more specifically to the optimal control methods which are using mathematical model of the process to predict the future response of process on a sequence of control variable manipulations. Once the predictions aremade, the control algorithmwith usage of the optimization techniques computes appropriate control actions to provide desired output behavior of the process in optimal fashion.

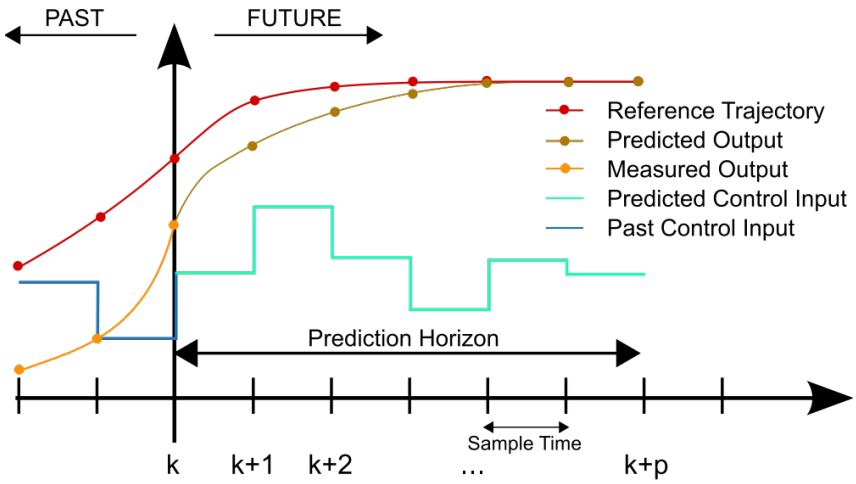


Figure 2.1: MPC Scheme

Colloquially we can describe this method as a ”look ahead” strategy, when the controller is able to foresee a future behavior of the process with usage of given knowledge about that particular process and consequently evaluate the optimal control strategy to achieve the best possible outcome, which are satysfying long term goals and criteria. This strategy stands in contrast with classical control theory techniques e.g. PID controllers, which are able to achieve only short term goals set in actual time, resulting in more costly and of then unsatisfactory long term

performance.

Model predictive control is the robust technique for the explicitly including constraints in the control law. Ability to handle multi-input multi-output models and features like economic MPC further add to the popularity of the controller. In recent years, MPC has also been surpassed proportional integral derivative (PID) controller and linear quadratic regulator (LQR).

2.1 MPC Overwiev and Features

MPC is a control strategy that uses an optimization to calculate the optimal control inputs, with usage of mathematical model of the system and current state measurements for predicting a evolution of the system behavior, and keeping these future predictions in account during optimization. In the MPC framework the cost or also called objective function evaluates fitness of a particular predicted profile of state, output and inputs with respect to qualitative criteria. Task of the optimization is then to compute the optimal profile of predicted control actions for which the cost function is minimized. The set of admissible decisions to choose from is then represented by the constraints of the optimization problem. The MPC is based on iterative character of an optimization process executed over finite time interval also called a prediction horizon, which can be simplistically percieved as measure of how far into the future the MPC algorithm can see. At current time the plant states are being measured and a cost minimizing control strategy is computed, via a numerical algorithms over given prediction horizon. Basic building elements forming characteristic structure of standard MPC are sumarised and listed as follows.

- Model of the system
- State measurements
- Constraints
- Objective
- Prediction horizon
- Sampling time

2.2 Standard MPC Formulation

In this work, we assume that the state-update and output equations are both discrete-time and linear-time invariant (LTI) in the form as follows:

$$x(t+1) = Ax(t) + Bu(t), \tag{2.1a}$$
$$y(t) = Cx(t) + Du(t), \tag{2.1b}$$

where, $x(t) \in \mathbb{R}^{n_x \times 1}, y(t) \in \mathbb{R}^{n_y \times 1}$, and $u(t) \in \mathbb{R}^{n_u \times 1}$ are differential state vector, output vector, and control input vector of the plant, respectively. Also, $A \in \mathbb{R}^{n_x \times n_x}, B \in \mathbb{R}^{n_x \times n_u}$, $C \in \mathbb{R}^{n_y \times n_x}$, and $D \in \mathbb{R}^{n_y \times n_u}$ are the system's input and output matrices describing the plant dynamics. MPC solves constrained finite-time optimal control (CFTOC) problem for reference tracking at each iteration which is given as follows:

$$\min_{U_N} \sum_{k=0}^{N-1} (y_k - y_{\mathrm{r},k})^T Q (y_k - y_{\mathrm{r},k}) + \Delta u_k^T R \Delta u_k, \tag{2.2a}$$

s.t.

$$x_{k+1} = Ax_k + Bu_k, \qquad k = 0, \ldots, N-1, \tag{2.2b}$$
$$y_k = Cx_k + Du_k, \qquad k = 0, \ldots, N-1, \tag{2.2c}$$
$$\Delta u_k = u_k - u_{k-1}, \qquad k = 0, \ldots, N-1, \tag{2.2d}$$
$$u_{\min} \leq u_k \leq u_{\max}, \qquad k = 0, \ldots, N-1, \tag{2.2e}$$
$$\Delta u_{\min} \leq \Delta u_k \leq \Delta u_{\max}, \qquad k = 0, \ldots, N-1, \tag{2.2f}$$
$$x_{\min} \leq x_k \leq x_{\max}, \qquad k = 0, \ldots, N-1, \tag{2.2g}$$
$$y_{\min} \leq y_k \leq y_{\max}, \qquad k = 0, \ldots, N-1, \tag{2.2h}$$
$$u_{-1} = u(t-1), \tag{2.2i}$$
$$x_0 = x(t), \tag{2.2j}$$

where, x_k, u_k, and y_k are the predictions of differential states, control input, and output at instant k. The term in (2.2a) is the objective function with prediction horizon N. The matrices $Q \in \mathbb{R}^{n_x \times n_x}$ and $R \in \mathbb{R}^{n_u \times n_u}$ are the weighting matrices, with conditions $Q \succeq 0$ to be positive semi-definite, and $R \succ 0$ to be positive definite. The term y_{r}, is the reference trajectory of the output. Finally, (2.2e)-(2.2h) are polyhedral constraint sets of state, input, and output, respectively. Solving the CFTOC problem in (2.2) for minimization of cost function for initial conditions x_0, we get sequence of optimal control input $(U_N^\star = u_0^\star, \ldots, u_{N-1}^\star)$ over the prediction horizon (N). Only the first term of the control sequence $(u_0^\star)$ is fed to the plant and current states $(x(t))$ are

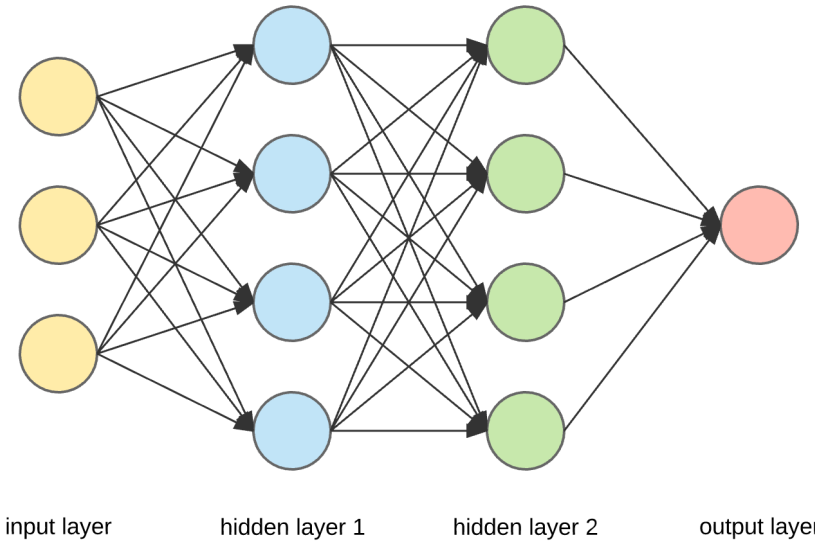


Figure 2.2: Artificial Neural Networks (ANN) are multi-layer fully-connected neural nets

measured which are then again sent back to the controller. Using updated values of the state measurement x_0 entire procedure is repeated for next time instant $t+1$. This concept is known as receding horizon control (RHC) (Borrelli et al., 2017, Chapter 12).

In spite of this tremendous success of MPC, there are still many restrictions to the use of MPC for real-time implementation. As previously mentioned, MPC has a huge burden of solving complex optimization problem within every sampling instant. Further, linear MPC has huge matrices which take up the major chunks of available memory on the embedded devices which restricts the implementation of MPC for small systems. However, deep learning methods for MPC are able to accurately approximate the linear MPC control law reducing the computational complexity, numerical instability and memory footprints of the controller (Karg and Lucia, 2018).

2.3 Deep Neural Network

Deep learning has gained immense recognition due to the wide and easy applicability to many applications in almost every domain. Deep learning architecture includes reinforcement learning, convolutional neural networks, and variational encoders (VAEs) which provide a robust mathematical framework for supervised as well as unsupervised learning. Using the power of multi-layer deep architecture, huge labeled data sets can be trained and analyzed.

Mathematical validation to DNN can be found in (Hornik et al., 1989; Hornik et al., 1990),

wherein they proved that with enough hidden layers and a linear output layer can be used to learn any arbitrary function using DNN. Training should be done meticulously as over-fitting or under-fitting might compromise the accuracy of the trained model. Also, there are no fixed rules to determine the required number of hidden layers to locate the global optimal.

In this thesis, we explicitly investigate the use of deep neural networks for MPC. Deep neural networks are a combination of multiple non-linear processing layers which are made up of simple elements in parallel operation and are inspired by biological nervous systems. These networks consist of multiple hidden layers, an input layer and an output layer which are connected using nodes and neurons. Recurrent neural networks (RNN) prove to be excellent for sequential data which throughput state of the art performance for the tasks including speech recognition, machine translation, language modeling, and time-series data (Zaremba et al., 2014; Mikolov and Zweig, 2012; Graves et al., 2013).

2.4 Structure of RNN

The model which is to be trained is approximated using RNN. The recurrent neural networks basically consist of two components: encoder and decoder, see Fig. 2.3. Actual future states prediction is done by the decoder module which consists of N cells, one for every single time step. The encoder takes care of long term dynamics and predicts the latent states (Bieker et al., 2019). Latent states or latent variables are simply the hidden variables or states in a system. Only a small portion of the decoder cell is contained in the encoder cell to predict the latent variables.

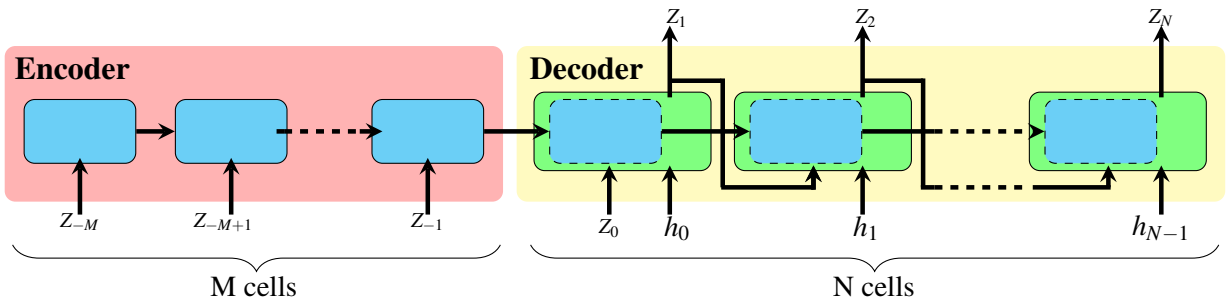


Figure 2.3: DNN encoder-decoder network. Here, green cells depict decoder cell whereas encoder cells are shown in blue. Encoder cells are just a part of the decoder cells.

A sequence of past observations and current control input is fed as input to each RNN cell. These RNN cells are further divided into three parts which capture long term dynamics of the system, the effect of the control input and current dynamics. The input of every single input

cell, i is divided into an observable time series $(z,h)_{i-2d-1,\ldots,i}$ and the corresponding sequence of control inputs, d being the delay is $h_{i-d,\ldots,i}$. This combined system is used to capture the system dynamics and perform the predictions.

Problems including MPC coupled with RNN can be solved using Levenberg Marquardt, which interpolates between gradient descent method and gauss-newton algorithm (GNA). Generalized back-propagation through time is used in the process. In the next section, we will see how system dynamics are captured using previous data, the training of the controller and its implementation.

Basic RNNs are a network of neuron-like nodes organized into successive ”layers”, each node in a given layer is connected with a directed (one-way) connection to every other node in the next successive layer. Each node (neuron) has a time-varying real-valued activation. Each connection (synapse) has a modifiable real-valued weight. Nodes are either input nodes (receiving data from outside the network), output nodes (yielding results), or hidden nodes (that modify the data en route from input to output).

For supervised learning in discrete time settings, sequences of real-valued input vectors arrive at the input nodes, one vector at a time. At any given time step, each non-input unit computes its current activation (result) as a nonlinear function of the weighted sum of the activations of all units that connect to it. Supervisor-given target activations can be supplied for some output units at certain time steps. For example, if the input sequence is a speech signal corresponding to a spoken digit, the final target output at the end of the sequence may be a label classifying the digit.

In reinforcement learning settings, no teacher provides target signals. Instead a fitness function or reward function is occasionally used to evaluate the RNN’s performance, which influences its input stream through output units connected to actuators that affect the environment. This might be used to play a game in which progress is measured with the number of points won.

Each sequence produces an error as the sum of the deviations of all target signals from the corresponding activations computed by the network. For a training set of numerous sequences, the total error is the sum of the errors of all individual sequences.

Chapter 3

Case study

3.1 Anesthesia

To demonstrate real-time implementation of proposed DNN-MPC on embedded platforms, a biomedical application- anesthesia control system is considered. Hospitals these days have well equipped operating rooms (ORs). Advance surgical techniques bring new biomedical technologies and more instrumentation in the ORs, which often results in complex configurations and big size machinery which occupies more space. Efforts have been made to improve the quality control and replace big size machines with embedded systems (see, (Ingole, 2017)).

Anesthesia is an amount of anesthetic drug(s) given to the patient to control his state of arousal during surgery. In traditional practice, the amount of initial drug dose and it's variation during surgery is decided by an anesthesiologist depending on patient's physical factors such as age, weight, height, and gender. Manual delivery of drugs can cause over or underdosing of the anesthetic which can put patient's safety and health at risk. Due to these issues, closed-loop control of anesthesia drug delivery is a challenging research topic over the last few decades.

For automatic feedback control of anesthesia which fuses drugs based on the anesthetic level of a patient, we consider four compartments, single-input (drug rate) single-output (concentration) model of the patient with Hill's Sigmoid model to measure the depth of anesthesia (DoA) using a bi-spectral index (BIS). The output represents Propofol concentration which is controlled by Propofol infusion rate.

Table 3.1: Pharmacokinetic parameter values given as a function of patient's AGE (years) and BW (kg).

Parameter	**Value**
k_1	$0.0595^{0.75}$ L/min (age ≤ 60)
	$(0.0595\mathrm{BW}^{0.75} - 0.45\mathrm{AGE} + 2.7)$ L/min (age > 60)
k_2	$0.0969\mathrm{BW}^{0.62}$ L/min
k_3	$0.0889\mathrm{BW}^{0.55}$ L/min
k_4	0.12 L/min
V_1	$1.72\mathrm{BW}^{0.71}\mathrm{AGE}^{-0.39}$ L
V_2	$3.32\mathrm{BW}^{0.61}$ L
V_3	266 L
V_4	$0.01V_1$ L

Anesthesia model is a combination of two elements, a pharmacokinetic and pharmacodynamic (PKPD) model. Pharmacokinetic is a study of absorption, distribution, metabolism, and excretion (ADME) of bio-active compounds in a higher organism. As shown in Fig. 3.1, the pharmacokinetic model consists of four-compartment i.e., central compartment (C_1), shallow peripheral compartment (C_2), deep peripheral compartment (C_3), and effect-side compartment (C_4). Pharmacodynamics is given by Hill's sigmoid model which is a nonlinear and static relationship between concentration and BIS. For brief explanation of model see (Schüttler and Ihmsen, 2000; Sawaguchi et al., 2008).

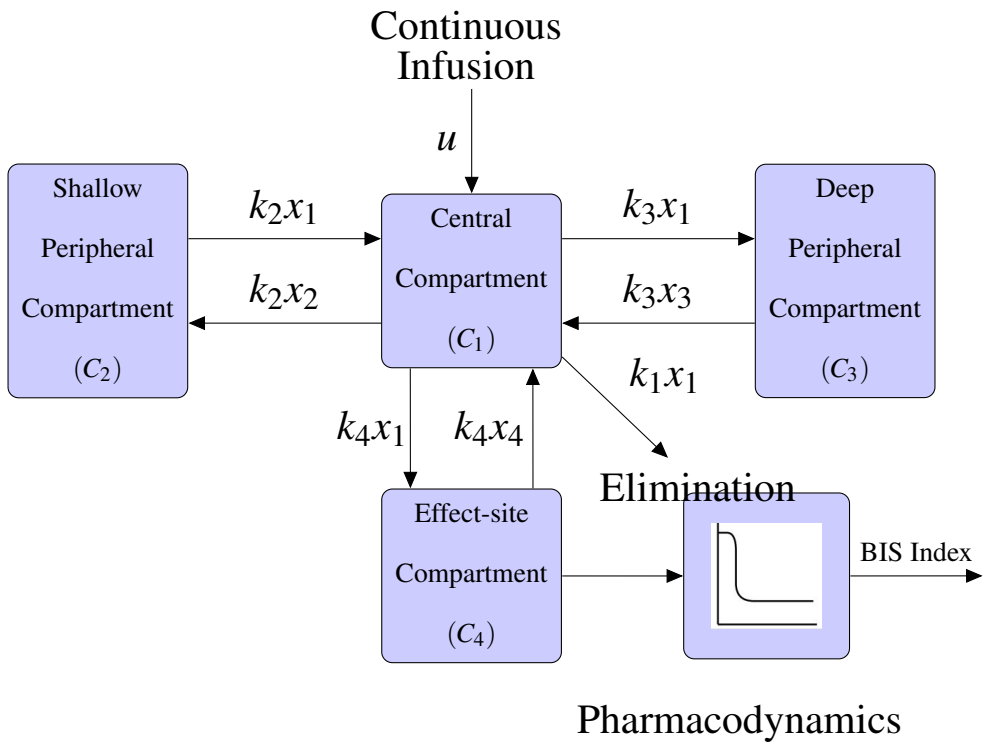


Figure 3.1: Three-compartment pharmacokinetic patient model with the effect-site compartment. (Schüttler and Ihmsen, 2000; Sawaguchi et al., 2008).

3.2 Model Formulation

The state-space representation of the pharmacokinetic model is given by,

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{3.1a}$$

$$y(t) = Cx(t) + Du(t), \tag{3.1b}$$

where, $x(t)$ is a state vector representing Propofol concentration in different compartments, $u(t)$ is the controlled vector representing Propofol infusion rate, $y(t)$ is the output vector representing effect site concentration of Propofol, and A, B, C, and D are pharmacokinetic parameters which are given by,

$$A = \begin{bmatrix} -\frac{k_1+k_2+k_3+k_4}{V_1} & \frac{k_2}{V_1} & \frac{k_3}{V_1} & \frac{k_4}{V_1} \\ \frac{k_2}{V_2} & -\frac{k_2}{V_2} & 0 & 0 \\ \frac{k_3}{V_3} & 0 & -\frac{k_3}{V_3} & 0 \\ \frac{k_4}{V_4} & 0 & 0 & -\frac{k_4}{V_4} \end{bmatrix},$$

$$B = \begin{bmatrix} \frac{1}{V_1} & 0 & 0 & 0 \end{bmatrix}^T, C = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T, \text{and } D = \begin{bmatrix} 0 \end{bmatrix}.$$

Here, the subscripts 1, 2, 3, and 4 correspond to the central, shallow peripheral, deep peripheral and effect site compartment, respectively. The parameter k and V represent clearance, and volume of different compartments which depends on the patient's age and body weight. The relation between age, weight, clearance and volume is given in Table 3.1.

The relationship between the effect-site concentration of Propofol and the BIS index value in the pharmacodynamic model is given by the following Hill's sigmoid $E_{\max}$ model (Schüttler and Ihmsen, 2000):

$$BIS(t) = E_0 - E_{\max}\frac{y(t)^\gamma}{y(t)^\gamma + C_{50}^\gamma}, \tag{3.2a}$$

where $E_{\max} = E_0$, E_0 denotes BIS value before starting the Propofol infusion, $E_{\max}$ denote the change of the BIS index corresponding to the infinite Propofol concentration, γ denotes the Hill's coefficient, and C_{50} denote the effect site concentration corresponding to $\frac{E_{\max}}{2}$.

The objective is to control BIS index such that it will keep track of a prescribed reference by manipulating the input variable, Propofol infusion rate while strictly following the constraints.

Chapter 4

Embedded Implementation of DNN-MPC

The following sections describes the implementation of deep neural networks based MPC on a low-cost microcontroller and the method to implement it on embedded hardware. Numerical robustness, memory limitations, consideration of worst-case scenarios, tolerance against instability and safety are some of the challenges in the embedded realization of MPC (Johansen, 2017).

4.1 Training of the Controller

Training neural networks is one of the most crucial tasks. All the dynamics of the plant should be taken into account while generating the data and training the controller. Fig. **??** shows the framework in which the MPC is replaced by DNN-MPC controller. We generated data for training the controller using simulations. Linear MPC was designed and simulated in MATLAB. The designed MPC had a sampling time of 1 [s] and prediction horizon, $N = 15$. For a robust and rigorous training, we used 50000 data-points in which 70% was allocated for training while 15% each for testing and validation, respectively.

The data was trained using MATLAB-based Neural Network Time Series function, invoked using `trainNetwork` (Mathworks, 2017). The training was performed using Levenberg_Marquardt (Moré, 1978) method, as this algorithm usually takes less time but more memory. Training will automatically stop once there is no improvement in generalization as indicated by an increase in the mean square error (MSE) of the validation samples (upto 1000 epochs). The model was trained on a Windows 10 PC with 3.5 GHz Intel® Quad Core i7 with 16 GB of RAM.

Results of the training are shown in Table 4.1 with training parameters as 12 hidden neurons and a single delay. From the table, we can see that mean squared error is close to zero which shows

that the controller is properly trained with minimal errors. To further establish the robustness and accuracy of the trained model, we plot the error auto-correlation to show how the networks error at any given time correlated with itself at various tapped delays. Peak at the center and pivoting rapidly with greater lags along-with other correlations within 95% of the confidence level shows an acceptable behavior. This can be further tuned using more number of neurons or tapped delays. However, for a perfectly trained network, there should only be one nonzero value at zero time lag.

Table 4.1: Training statistics.

Scenario	**Target Values**	**MSE**	**R**
Training	35000	6.28241×10^{-9}	9.99999×10^{-1}
Validation	7500	1.14839×10^{-6}	9.94938×10^{-1}
Testing	7500	1.40538×1^{-8}	9.99999×10^{-1}

Table 4.2: Memory footprints and run-time results of proposed DNN-MPC and LMPC.

Controller	**Memory [%]**		**Run-time**
	% Data Usage	**% Program Usage**	**[ms]**
Linear MPC	3.50	4.90	11.354
DNN MPC	3.01	3.92	2.999

Chapter 5

Implementation and Results

We have demonstrated the design of deep neural network based MPC which approximates MPC control law. Complete work-flow of the mechanism used while designing and implementing DNN-MPC is shown in Fig. **??**. Particularly in this model, the safety of the controller is of prime importance. The controller should be able to maintain the desired levels without any major fluctuations in the input. The BIS level is typically maintained between $40-60$ to guarantee the effect of the drug on the patient. The HIL simulations in Fig. 5.1 show that deep neural networks have approximated the original LMPC control law precisely and follow the reference smoothly. As seen from the Fig. 5.1, it is important to note that, in spite of some persisting steady-state errors of very low magnitude, the controller does not show any kind of overshoot or undershoot, which is better for the health of the patient. These persisting steady-state error can be reduced by fine-tuning and training the model with more data. Also, the controller input stays between the prescribed infusion rate which should be between $0-20$ $[\mathrm{mg/kg/h}]$.

However, the main focus of this paper is to reduce the computational complexity and the numerical instability associated while solving the optimization problem on-line. The proposed DNN-MPC addresses this problem. The proposed controller is implemented on ARM Cortex M3 micro-controller having 512 kB flash memory and 96 kB of SRAM running at 84 MHz. From Table 4.2, it can be seen that the per iteration computational time is significantly reduced from 11.354 ms to 2.99 ms.

The total memory consumption of DNN-MPC as compared to LMPC, when deployed on ARM microcontroller, is shown in the results table. It can also be inferred from Table 4.2, that even though the reduction in memory footprint is not that pronounced for this particular model, we still believe that for larger systems having more number of states and constraints, the proposed DNN-MPC will be much more memory efficient.

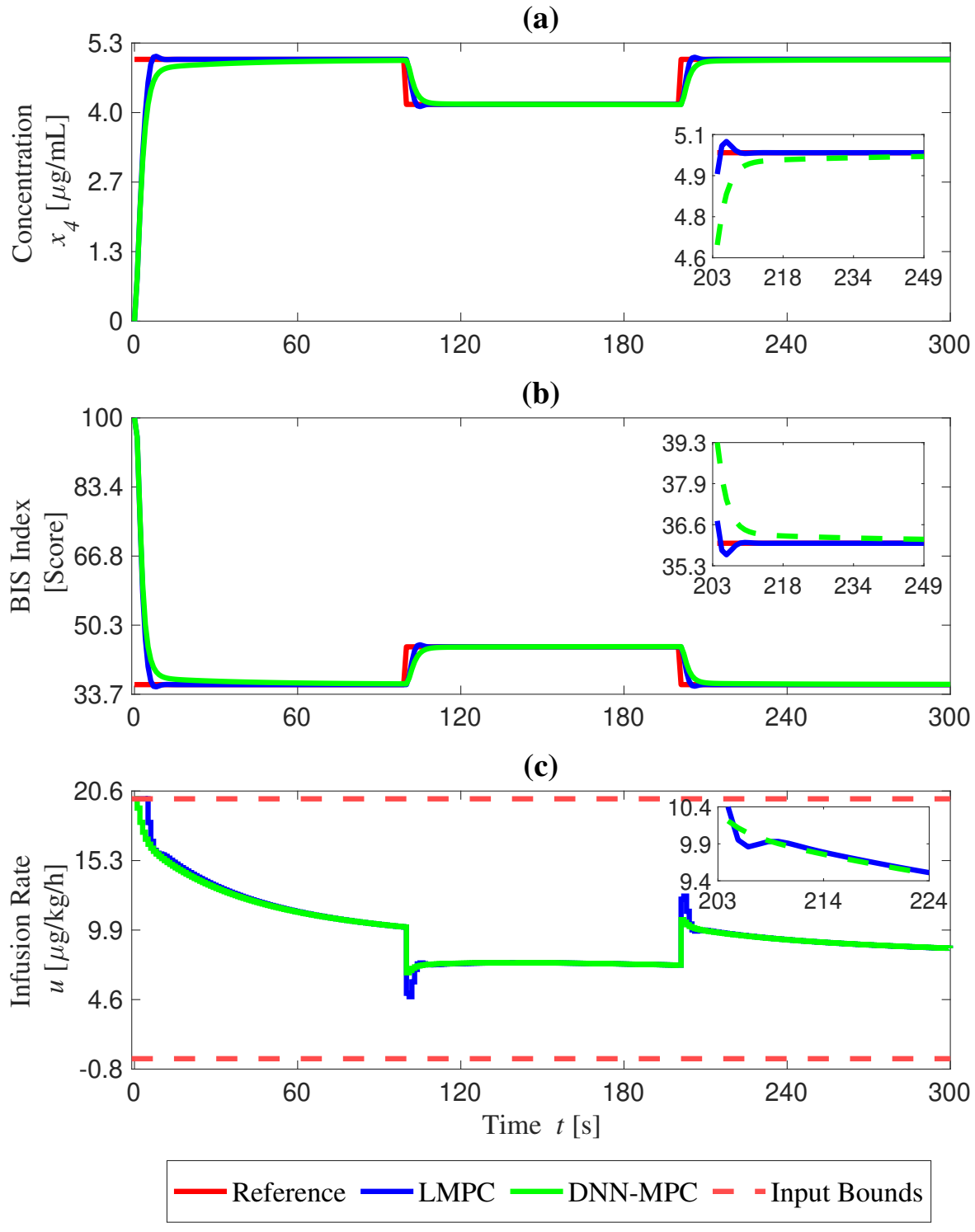


Figure 5.1: Performance comparison of proposed DNN-MPC with LMPC for anesthesia control problem.

Chapter 6

CONCLUSIONS

This work proposes the use of deep neural networks to approximate linear MPC control law, efficiently and effectively with minimal computational efforts for real-time embedded implementation. The trained DNN model was verified using HIL co-simulation with ARM Cortex M3 microcontroller. This work showcases the detailed performance comparison of linear MPC and DNN trained MPC in terms of memory utilization and time to solve one iteration. Closed-loop simulation results for anesthesia drug delivery system using DNN-MPC are presented as well. Presented results shows that the proposed approach guarantees the safety of the patient and performs faster which make it as a promising approach for safety-critical applications running on embedded systems. Future work will include to make complex non-linear MPC solutions using deep neural networks to use them on embedded hardware. This kind of studies act like the intermediate steps towards more advanced data-driven low-complexity well-performing controllers in the future. Particularly, the vision of the self-constructing and self-tuning near-optimal controllers with easy plug and play implementation is very appealing. The possible direction in minimizing the commissioning effort and cost could be combining the black- or grey-box system identification techniques with automatic MPC design and tuning, together with the automated synthesis of simplified controllers. The challenge here emanates from a complex task of hyper-parameters setting and iterative nature of the training and tuning process for the machine learning models. However, with extensive research effort nowadays in the field of hyper-parameter optimization and machine learning in general, the efficient tools for handling these challenges can soon be expected and implemented also in the building sector. Another disadvantage to being tackled in the future is the lack of the analytical closed-loop performance guarantees concerning stability and constraints satisfaction. Although, this appears to be only the minor issue for non safety-critical applications like building control.

Bibliography

Adhau, S., Patil, S., Ingole, D., and Sonawane, D. (2019a). Implementation and Analysis of Nonlinear Model Predictive Controller on Embedded Systems for Real-Time Applications. In *2019 European Control Conference (ECC)*. IEEE.

Adhau, S., Phalke, K., Nalawade, A., Ingole, D., Patil, S., and Sonawane, D. (2019b). Implementation and Analysis of Offset-Free Explicit Model Predictive Controller on FPGA. In *2019 Fifth Indian Control Conference (ICC)*, pages 231–236. IEEE.

Aswani, A., Master, N., Taneja, J., Culler, D., and Tomlin, C. (2011). Reducing transient and steady state electricity consumption in HVAC using learning-based model-predictive control. *Proceedings of the IEEE*, pages 240–253.

Baumeister, T., Brunton, S. L., and Kutz, J. N. (2018). Deep learning and model predictive control for self-tuning mode-locked lasers. *JOSA B*, pages 617–626.

Bemporad, A. (2004). Hybrid Toolbox - User's Guide. `http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox`.

Bemporad, A., Morari, M., Dua, V., and Pistikopoulos, E. N. (2000). The explicit solution of model predictive control via multiparametric quadratic programming. In *Proceedings of the 2000 American Control Conference. ACC*, pages 872–876.

Bieker, K., Peitz, S., Brunton, S. L., Kutz, J. N., and Dellnitz, M. (2019). Deep Model Predictive Control with Online Learning for Complex Physical Systems. *arXiv preprint arXiv:1905.10094*.

Borrelli, F., Bemporad, A., and Morari, M. (2017). *Predictive control for linear and hybrid systems*. Cambridge University Press.

Bouffard, P., Aswani, A., and Tomlin, C. (2012). Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results. In *2012 IEEE International Conference on Robotics and Automation*, pages 279–284. IEEE.

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, pages 1–122.

Cimini, G., Bemporad, A., and Bernardini, D. (2017). ODYS QP Solver. ODYS (`https://odys.it/qp`).

Currie, J. (2011). jMPC Toolbox v3. 11 User's Guide. *User's guide, AUT University*.

Domahidi, A. and Perez, J. (2014). FORCES PRO ACADEMIC.

Drgoňa, J., Picard, D., Kvasnica, M., and Helsen, L. (2018). Approximate model predictive building control via machine learning. *Applied Energy*, 218:199–216.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. pages 6645–6649.

Herceg, M., Kvasnica, M., Jones, C. N., and Morari, M. (2013). Multi-parametric toolbox 3.0. In *2013 European Control Conference (ECC)*, pages 502–510. IEEE.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, pages 359–366.

Hornik, K., Stinchcombe, M., and White, H. (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural networks*, pages 551–560.

Ingole, D. (2017). *Embedded Implementation of Explicit Model Predictive Control*. PhD thesis, IAM FCHPT STU in Bratislava.

Ingole, D. and Kvasnica, M. (2015). FPGA implementation of explicit model predictive control for closed loop control of depth of anesthesia. *IFAC-PapersOnLine*, 48(23):483–488.

Johansen, T. A. (2017). Toward Dependable Embedded Model Predictive Control. *IEEE Systems Journal*, pages 1208–1219.

Karg, B. and Lucia, S. (2018). Deep learning-based embedded mixed-integer model predictive control. In *2018 European Control Conference (ECC)*, pages 2075–2080. IEEE.

Kumar, S. S. P., Tulsyan, A., Gopaluni, B., and Loewen, P. (2018). A Deep Learning Architecture for Predictive Control. *IFAC-PapersOnLine*, pages 512–517.

Lucia, S., Navarro, D., Karg, B., Sarnago, H., and Lucia, O. (2018). Deep Learning-Based Model Predictive Control for Resonant Power Converters. *arXiv preprint arXiv:1810.04872*.

Mathworks (2017). Train neural network for deep learning Time-Series Prediction and Modeling. `https://in.mathworks.com/help/deeplearning/ref/trainnetwork.html`.

Mattingley, J. and Boyd, S. (2012). CVXGEN: A code generator for embedded convex optimization. *Optimization and Engineering*, pages 1–27.

Mendez, J. A., Leon, A., Marrero, A., Gonzalez-Cava, J. M., Reboso, J. A., Estevez, J. I., and Gomez-Gonzalez, J. F. (2018). Improving the anesthetic process by a fuzzy rule based medical decision system. *Artificial intelligence in medicine*, 84:159–170.

Mikolov, T. and Zweig, G. (2012). Context dependent recurrent neural network language model. pages 234–239.

Moore, B. L., Pyeatt, L. D., Kulkarni, V., Panousis, P., Padrez, K., and Doufas, A. G. (2014). Reinforcement learning for closed-loop propofol anesthesia: a study in human volunteers. *The Journal of Machine Learning Research*, pages 655–696.

Moré, J. J. (1978). The Levenberg-Marquardt algorithm: Implementation and theory. In Watson, G. A., editor, *Numerical Analysis*, pages 105–116, Berlin, Heidelberg.

Negenborn, R. R., Schutter, B. D., and Wiering, M. A. (2004). Experience-based model predictive control using reinforcement learning. *Proceedings of the 8th TRAIL Congress 2004 — A World of Transport, In- frastructure and Logistics*.

Padmanabhan, R., Meskin, N., and Haddad, W. M. (2015). Closed-loop control of anesthesia and mean arterial pressure using reinforcement learning. *Biomedical Signal Processing and Control*, 22:54–64.

Padula, F., Ionescu, C., Latronico, N., Paltenghi, M., Visioli, A., and Vivacqua, G. (2017). Optimized PID control of depth of hypnosis in anesthesia. *Computer methods and programs in biomedicine*, 144:21–35.

Rosolia, U. and Borrelli, F. (2017). Learning model predictive control for iterative tasks. a data-driven control framework. *IEEE Transactions on Automatic Control*, 63(7):1883–1896.

Sawaguchi, Y., Furutani, E., Shirakami, G., Araki, M., and Fukuda, K. (2008). A model-predictive hypnosis control system under total intravenous anesthesia. *IEEE transactions on biomedical engineering*, pages 874–887.

Schüttler, J. and Ihmsen, H. (2000). Population pharmacokinetics of propofola multicenter study. *Anesthesiology: The Journal of the American Society of Anesthesiologists*, pages 727–738.

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S. (2018). OSQP: An operator splitting solver for quadratic programs. In *2018 UKACC 12th International Conference on Control (CONTROL)*, pages 339–339. IEEE.

Trinh, V.-V., Alamir, M., Bonnay, P., and Bonne, F. (2016). Explicit model predictive control via nonlinear piecewise approximations. *IFAC-PapersOnLine*, pages 259–264.

Wang, Y. and Boyd, S. (2009). Fast model predictive control using online optimization. *IEEE Transactions on control systems technology*, pages 267–278.

Xie, Y., Ghaemi, R., Sun, J., and Freudenberg, J. S. (2011). Model predictive control for a full bridge DC/DC converter. *IEEE Transactions on Control Systems Technology*, pages 164–172.

Yang, X., Griffith, D. W., and Biegler, L. T. (2015). Nonlinear programming properties for stable and robust NMPC. *IFAC-PapersOnLine*, pages 388–397.

Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Zometa, P., Kögel, M., and Findeisen, R. (2013). μAO-MPC: a free code generation tool for embedded real-time linear model predictive control. In *2013 American Control Conference*, pages 5320–5325. IEEE.