A NEURAL APPROXIMATION TO THE EXPLICIT SOLUTION OF CONSTRAINED LINEAR MPC

H. Haimovich*, M.M. Seron*, G.C. Goodwin* and J.C. Agüero*

* Centre for Integrated Dynamics and Control, The University of Newcastle, Callaghan, NSW 2308, Australia,
Ph: +61 2 4921 7072, Fax: +61 2 4960 1712
{hhaimo,seron,eegcg,jaguero}@ecemail.newcastle.edu.au

Keywords: Neural networks, model predictive control, constrained linear control, explicit solution, approximation.

Abstract

The solution to constrained linear model predictive control (MPC) problems can be pre-computed off-line in an explicit form as a piecewise affine (PWA) state feedback law defined on polyhedral regions of the state space. Even though real-time optimization is avoided, implementation of the PWA state-feedback law may still require a significant amount of computation due to the problem of determining which polyhedral region the state lies in. In this paper, a neural network approach to this problem is investigated.

1 Introduction

Model predictive control (MPC), or receding horizon control (RHC), is a state feedback strategy that employs the first of a sequence of control actions corresponding to the solution of an open-loop optimal control problem. For constrained open-loop optimisation problems, MPC typically computes the control action on-line, given the current state vector (see e.g., [10]).

Recently, different research teams ([2, 15]) have explored the option of expressing the control action that solves the constrained linear MPC problem as an explicit function of the state. Given this explicit state-feedback law, on-line optimisation may be avoided, rendering MPC suitable for applications where a fast sampling rate is needed.

Even though the control action can now be calculated using the explicit state-feedback law, this may still require a significant amount of computation due to the fact that this law is expressed as a collection of affine control laws, each one valid inside a polyhedral region of the state-space, and that the number of polyhedral regions may be large. This issue has motivated research aimed at increasing the efficiency of the evaluation of the state-feedback law as well as at finding approximate explicit solutions.

In [12], efficiency of the evaluation of the exact state-feedback law is addressed by constructing a binary search tree which achieves a computation time logarithmic in the number of polyhedral regions. A method for generating an approximate state-feedback law based on a binary search tree is also proposed. In [9], a suboptimal strategy is considered, where an approximation to the optimal cost function is utilized, imposing restrictions on the allowed switching between the active constraint sets during the prediction horizon. Input trajectory parameterization is studied in [11], as well as the development of a search tree for efficient evaluation of the PWA solution. In [1], approximate (suboptimal) solutions to the multiparametric quadratic programming problem are found by relaxing the Karush-Kuhn-Tucker optimality conditions. In [7], it is shown how system structure can be exploited to derive reduced dimension multiparametric quadratic programs that lead to suboptimal explicit PWA feedback solutions to the state and input constrained LQR problem. In [8], approximate explicit solutions to the MPC problem are built in correspondence with given bounds for suboptimality and constraint violation. In [4], approximate multiparametric quadratic programming is used, structuring the partition as a binary search tree. A neural network approximation to a receding horizon regulator for a nonlinear dynamic system with a nonquadratic cost function is proposed in [13].

The approach proposed here is novel in the sense that it aims to approximate the determination of the region the state lies in, i.e., given the state, determine which region it corresponds to. We do this by constructing a multilayer feedforward neural network and by developing a selection procedure for the training points. An advantage of the proposed neural network approach is that not every region in the polyhedral partition has to be considered but only those that have different control laws assigned. That is, regions with equal control laws can be joined even if the resulting partition involves nonconvex regions.

The remainder of the paper proceeds as follows. Section 2 outlines the formulation of the constrained linear MPC problem as a multiparametric quadratic program. Section 3 details the construction of a neural network for the selection of a control law corresponding to a given state vector. Section 3.1 develops a procedure for selecting the training points for the network. The structure of the neural network is adopted in Section 3.2 and an interpretation of the consequences of the suggested approach is given in Section 3.3. Section 4 applies the suggested approach to an example and shows simulation results. Conclusions are drawn in Section 5.

2 Explicit Constrained Linear MPC

The system model is given by

$$x(k+1) = Ax(k) + Bu(k)\,, \qquad k = 0, 1, 2, \ldots \quad (1)$$
$$y(k) = Cx(k)\,, \quad (2)$$

where $x(k) \in \mathbb{R}^n$ is the state vector, $u(k) \in \mathbb{R}^m$ is the input vector, and $y(k) \in \mathbb{R}^p$, is the "output" vector or combinations of states of interest. It is assumed that (A, B) is stabilisable and that the matrices B and C have full column and row ranks, respectively.

For this model, at time k, the RHC technique poses the following *finite-horizon open-loop optimal control problem*: given the current state measurement $x(k) = x$, find the N-move control sequence (Nm control vector)

$$\mathbf{u} = \mathrm{col}\,(\mathrm{u(k)}, \mathrm{u(k+1)}, \ldots, \mathrm{u(k+N-1)})$$

that minimises the performance index:

$$V_N(x, \mathbf{u}) = \sum_{t=k}^{k+N-1} x^{\mathrm{T}}(t)Qx(t) + u^{\mathrm{T}}(t)Ru(t) + x^{\mathrm{T}}(k+N)Px(k+N)\,, \quad (3)$$

subject to the constraints

$$G\mathbf{u} \leq W + Ex. \quad (4)$$

Once this N-move control sequence has been found, only the first one of the moves $(u(k))$ is applied and the same calculations are repeated at the next sampling instant, when a new (measurement of the) state vector becomes available.

In (3), N is the prediction horizon; $Q \geq 0$ and $R > 0$ are the state and control weighting matrices, respectively, and $x^{\mathrm{T}}Px$, $P > 0$, is the terminal cost function.

By some algebraic manipulations, the problem can be reformulated as

$$V_z^*(x) = \min_z \frac{1}{2} z^{\mathrm{T}} H z \quad (5)$$
$$\text{subject to} \qquad Gz \leq W + Sx\,, \quad (6)$$

where $z = \mathbf{u} + H^{-1}F^{\mathrm{T}}x$ and $H > 0$. The solution of the optimization problem (5)-(6) can be found in explicit form $z^* = z^*(x)$. The solution, $z^*(x)$, is a continuous piecewise-affine (PWA) function of x defined on a polyhedral partition of the state space. For further details see [2]. For a derivation of the explicit solution based on the geometrical structure of MPC, see [15, 14].

Given the function $z^*(x)$, the corresponding MPC control law, $u = \kappa(x) : \mathbb{R}^n \to \mathbb{R}^m$, can be easily found. The control law $\kappa(x)$ is also a PWA function defined on a polyhedral partition of the state space.

3 Neural Network Approach

As the number of polyhedral regions in the partition may be very large, the evaluation of the exact explicit solution may still require a considerable amount of computation effort. This fact motivates the search for efficient and/or approximate explicit solutions. The evaluation of the explicit solution $u = \kappa(x)$ may be separated into two stages: determination of the current polyhedral region and evaluation of the associated affine feedback law.

In [13], an approximation to the map $u = \kappa(x)$ was implemented by means of a multilayer feedforward neural network. Here, we investigate the case where the neural network is used only to approximate the region where the current state x lies i.e., only for the first one of the stages outlined above. As our aim in determining the region is to select the correct control law, different regions with equal control laws (e.g., saturated u) can be joined. This generates regions which are not necessarily convex but this can be easily handled by the neural network approach. On the other hand, the number of regions to consider is noticeably reduced, as can be seen in Figure 1.

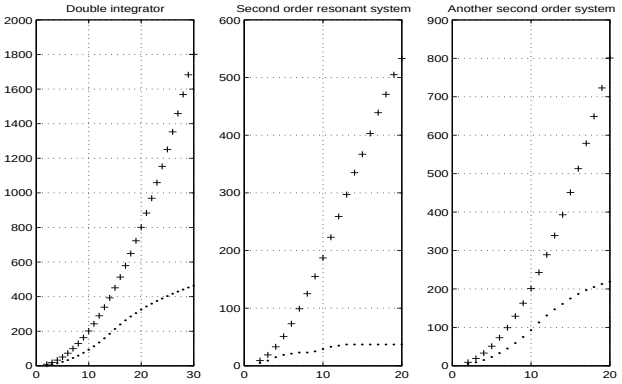

Figure 1: Number of different convex regions (+) and of different control laws (·) for 3 different second order systems as a function of the prediction horizon.

Figure 2 shows the structure of the proposed control system. The neural network receives the current state as an input and calculates a number that identifies the corresponding control law. Once this number has been calculated, the block named "Control Laws" is able to select the corresponding (affine) control law and produce the desired control action $u(k)$. The control laws stored in the "Control Laws" block can be calculated, for example, by one of the methods described in [2, 14, 15]. We emphasize that the approach proposed here is to approximate the determination of which one of the stored control laws (previously obtained by some other method) corresponds to a given state vector.

The determination of the region to which a given state vector belongs can be thought of as the evaluation of a function with finite range:

$$\mathrm{Reg}(x) : \mathbb{R}^n \to \{1, 2, \ldots, N_l\}, \quad (7)$$

where N_l denotes the number of regions in the partition, which in our case is the same as the number of different control laws.

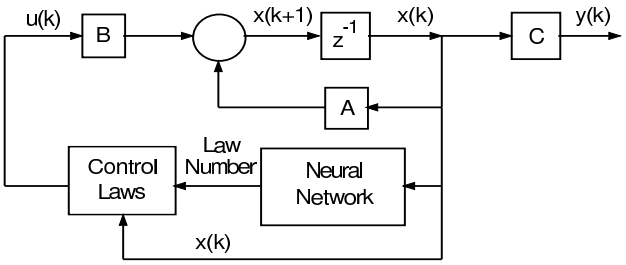

Figure 2: Control system structure.

3.1 Selection of the Training Points

We consider here the approximation of the function $\mathrm{Reg}(x)$ within a hypercube defined by

$$\begin{bmatrix} x_{1min} \\ x_{2min} \\ \vdots \\ x_{nmin} \end{bmatrix} \leq \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} x_{1max} \\ x_{2max} \\ \vdots \\ x_{nmax} \end{bmatrix}. \qquad (8)$$

Within this hypercube, some points have to be chosen as training points for the network. There exists a tradeoff in the selection of the training points: the more training points used, the more accurate the approximation but the greater the amount of calculation effort. Note that this calculation effort represents off-line computation and is only needed for the training of the network. Once the network is trained, the amount of on-line calculation does not depend on the number of training points previously chosen.

Although this calculation is made off-line, it may be impractical to perform without careful selection of the training points. For example, random selection of the training points from a unique uniform distribution within the hypercube (8) would result in a very high number of training points needed to obtain a sensible approximation. One good choice would be to draw the same number of samples from N_l different uniform distributions, each one supported on one of the polyhedral regions. The complexity of this selection scheme motivated the use of a simplified selection procedure.

Using an algorithm similar to the one described in [8], a partition of the hypercube (8) is found imposing an orthogonal search tree structure on it. A slight modification of this algorithm is made to avoid partitioning a hypercube if the same control law applies for all of its vertices. Once this orthogonal search tree partition is available, the set of points for training the network is constructed by drawing the same number of samples from uniform distributions supported on each one of the hypercubes in the partition.

3.2 Neural Network Structure

It is well-known that multilayer feedforward neural networks are able to approximate, within a hypercube, any continuous nonlinear mapping from a finite-dimensional space to another, to any desired degree of accuracy (see e.g., [6, 3, 5]), provided the activation functions meet some requirements and the number of neurons in the network is large enough. One of the requirements imposed on the activation functions for pointwise approximation is that they be continuous. Here, we intend to approximate the piecewise-constant finite-range function $\mathrm{Reg}(x)$ (7), which is not continuous. Our approach is to obtain $\mathrm{Reg}(x)$ by quantizing a continuous function. This continuous function will be provided by the neural network and the quantizing will be performed as a postprocessing step. The adoption of this approach imposes some conditions on the selection of the structure of the output layer of the network.

For example, if we consider a structure consisting of only one neuron in the output layer with a linear activation function, whose output value has to be quantized to an integer between 1 and N_l, an important drawback will arise. In Figure 3 the borders of 4 arbitrary regions are shown. Points P_1 and P_4 belong to regions 1 and 4, respectively. As we move from point P_1 to point P_4, following the line joining them (shown as a dotted line in Figure 3), we expect the output of the network to change from a value close to 1 to a value close to 4. If the activation functions of all the neurons in the network are continuous (which is a requirement, as explained above), then the output of the network will take on values 2 and 3 as well for some intermediate points. Hence, in this scheme the control law corresponding to region 3 will be used for a set of points which lie far away from region 3, producing an undesired control action.

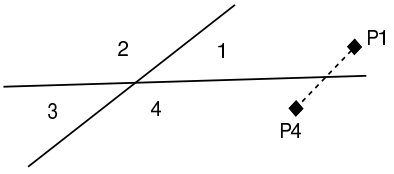

Figure 3: Borders of 4 arbitrary regions.

A useful structure for the output layer can be obtained by including as many neurons in the output layer as regions in the partition. Neuron N_i in the output layer will be expected to approximate the indicator function of region R_i, $I_{R_i}(x)$, defined as

$$I_{R_i}(x) \doteq \begin{cases} 1 & \text{if } x \in R_i \\ 0 & \text{otherwise} \end{cases}, \qquad i = 1, 2, \ldots, N_l. \quad (9)$$

The functions defined in (9) are not continuous but it is now evident that they can be obtained by quantizing a continuous function. To avoid classification of a given state vector as belonging to more than 1 region simultaneously, the output neuron with the highest output value will be selected as the one which will indicate the corresponding region, as il-

lustrated in Figure 4. As the output value for any output neuron needs vary only between 0 and 1, logistic sigmoid activation functions are selected for the output layer.

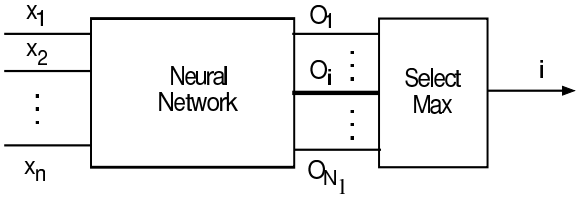

Figure 4: Neural network output layer and region selection scheme.

In [6], it is proved that a feedforward neural network with as few as one hidden layer can be a universal approximator. However, this result does not say that a single hidden layer is optimum in the sense of minimizing learning time or simplifying implementation. Results of a large number of simulations lead us to believe that the complexity in terms of on-line calculation is reduced by providing the network with 2 hidden layers as opposed to only one. Having made these considerations, it remains to select the activation functions for the hidden layers. We adopt a logistic sigmoid activation function for the first hidden layer and a linear one for the second hidden layer.

3.3 Interpretation of the Proposed Approach

A depiction of the consequences of the suggested neural approximation is sketched in Figure 5. In the top subfigure, a partition corresponding to the exact solution is shown, for an arbitrary case. Every region, distinguished by a number, has an assigned control law, valid for all the points in the same region. In the bottom subfigure, the borders of the regions are different because of the neural network approximation.

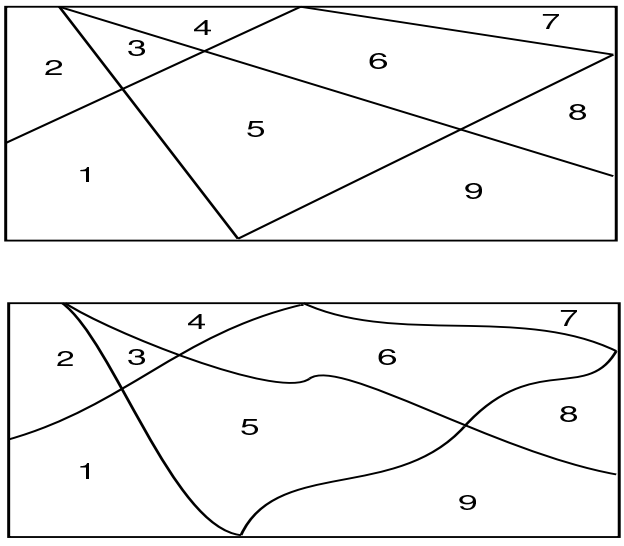

Figure 5: Regions approximated by the neural network.

4 Example and Simulation Results

We present here an example for the case of the double integrator with a prediction horizon $N = 10$, sampling time $T_s = 0.1$, $R = 1$, $Q = I_n$ (identity matrix) and P taken as the solution of the discrete algebraic Riccati equation. The control variable u is constrained such that

$$-0.5 \leq |u_t| \leq 0.5, \quad t = k,\, k+1, \ldots,\, k+N-1. \quad (10)$$

We will approximate the exact solution within the rectangle (hypercube) defined by

$$\begin{bmatrix} -10 \\ -6 \end{bmatrix} \leq \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 10 \\ 6 \end{bmatrix}. \quad (11)$$

The exact explicit solution $\kappa(x)$ consists of 93 different affine control laws, with a convex polyhedral partition having 201 polyhedral regions. Figure 6 shows the regions in the convex polyhedral partition within the selected rectangle (Equation (11)). The resulting 93 regions after joining the ones with equal control laws are shown in Figure 7.

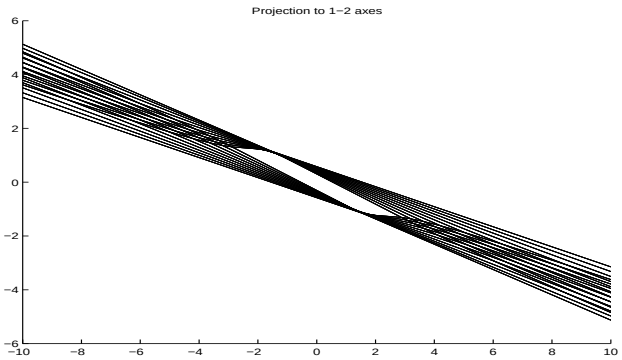

Figure 6: Polyhedral partition with convex regions.

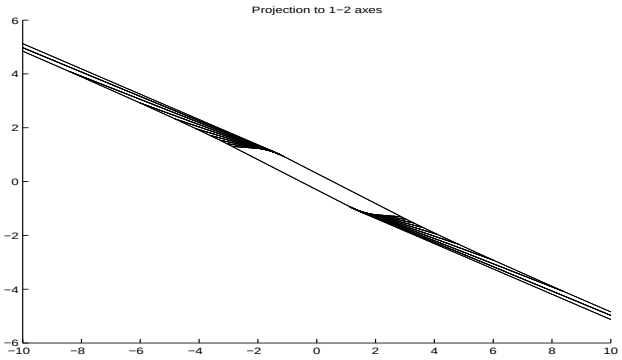

Figure 7: Regions corresponding to different control laws.

For a suboptimality bound $\bar{\epsilon} = 5$ and a constraint violation bound $\bar{\delta} = 0.1$, the orthogonal search tree partition of Figure 8 is found (see [8] for details). In this case, the partition in Figure 8 contains 1810 rectangles. Note that the bounds $\bar{\epsilon}$ and $\bar{\delta}$ are only used to find the orthogonal search tree partition and are not directly related to the accuracy of the neural network approximation to the explicit solution.

Within each one of the rectangles in the orthogonal search

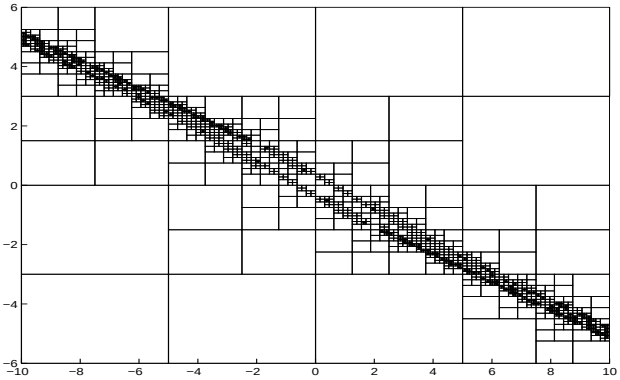

Figure 8: Orthogonal partition.

tree partition, 2 samples are drawn from a uniform distribution, generating a training set of 3620 points (Figure 9).

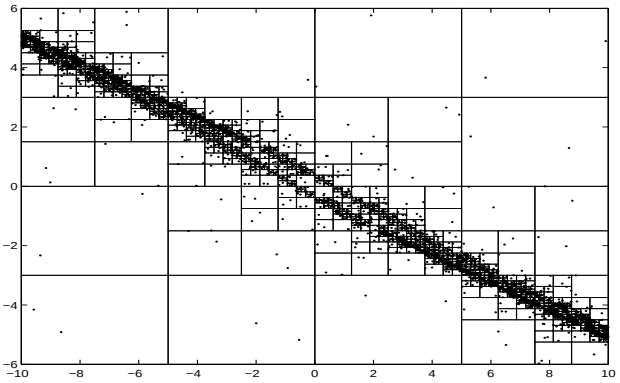

Figure 9: Training points and orthogonal search tree partition. Two points in each rectangle.

A multilayer feedforward neural network with two hidden layers of 75 and 23 neurons, respectively, was created and trained with the points in the set and their corresponding regions in the partition shown in Figure 7. As explained before, the activation functions were chosen as logistic sigmoid for the output and first hidden layers and linear for the second hidden layer. The training was performed using backpropagation training with 5000 iterations.

Finally, the state space trajectories of the system corresponding to the exact explicit solution and the neural network approximation are compared in Figures 10 and 11. It can be seen from Figure 10 that the plot corresponding to the neural network approximation is very close to the exact one. In Figure 11, a zoom has been included for distinguishing the difference between the two.

5 Conclusions

We have presented a novel approach to the approximation of solutions for MPC through the use of a multilayer feedforward neural network. The novelty of the approach lies in the use of the network for assigning a stored control law to a given state vector. Because of the structure of

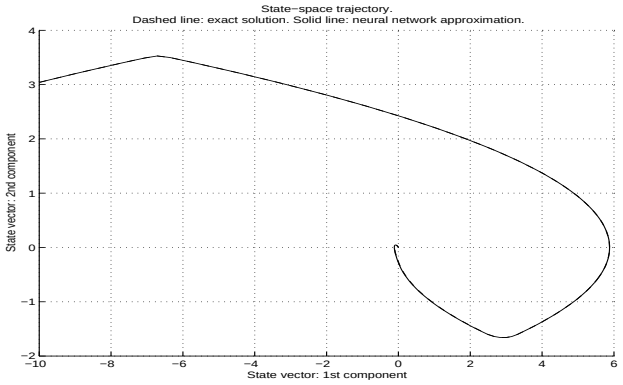

Figure 10: Complete state-space trajectory. Dashed line: Exact solution. Solid line: Neural network approximation.

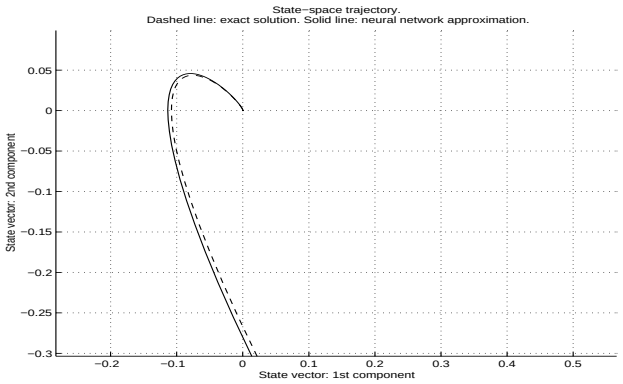

Figure 11: Zoom of state-space trajectory. Dashed line: Exact solution. Solid line: Neural network approximation.

the output layer of the neural network, the complexity of the neural network grows at least linearly with the number of different control laws. With the benefit of hindsight, it turns out that the suggested scheme can be outperformed by recent developments in other approximate methods. Nonetheless, the ideas presented here could be of independent scholarly interest and may be a worthy starting point for hybrid algorithms which combine several points of view.

References

[1] A. Bemporad and C. Filippi. Suboptimal explicit mpc via approximate multiparametric quadratic programming. In *Proc. 40th IEEE Conf. on Decision and Control, Orlando, Florida USA*, 2001.

[2] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38:3–20, 2002.

[3] Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2:183–192, 1989.

[4] A. Grancharova and T. A. Johansen. Approximate explicit model predictive control incorporating heuris-

tics. In *Proc. IEEE Conf. on Computer Aided Control Design*, pages 92–97, Glasgow, 2002.

[5] S. Haykin. *Neural Networks, A Comprehensive Foundation*. Macmillan, New York, 1994.

[6] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[7] T. A. Johansen. Structured and reduced dimension explicit linear quadratic regulators for systems with constraints. In *Proc. 41st IEEE Conf. on Decision and Control, Las Vegas*, pages 3970–3975, 2002.

[8] T. A. Johansen and A. Grancharova. Approximate explicit model predictive control implemented via orthogonal search tree partitioning. In *IFAC World Congress, Barcelona*, 2002.

[9] T. A. Johansen, I. Petersen, and O. Slupphaug. Explicit sub-optimal linear quadratic regulation with state and input constraints. *Automatica*, 38:1099–1111, 2002.

[10] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.

[11] P. Tøndel and T. A. Johansen. Complexity reduction in explicit linear model predictive control. In *IFAC World Congress*, Barcelona, 2002.

[12] P. Tøndel, T. A. Johansen, and A. Bemporad. Computation and approximation of piecewise affine control laws via binary search trees. In *Proc. 41st IEEE Conf. on Decision and Control, Las Vegas*, pages 3144–3149, 2002.

[13] T. Parisini and R. Zoppoli. A receding-horizon regulator for nonlinear systems and a neural approximation. *Automatica*, 31(10):1443–1451, 1995.

[14] M. M. Seron, J. De Doná, and G. C. Goodwin. Global analytical model predictive control with input constraints. In *Proc. 39th IEEE Conf. on Decision and Control, Sydney*, pages 154–159, 2000.

[15] M. M. Seron, G. C. Goodwin, and J. A. De Doná. Characterisation of receding horizon control for constrained linear systems. *Asian Journal of Control*, 5(2):271–286, 2003.