# Implementation Aspects of Embedded MPC with Fast Gradient Method

J. Novak, P. Chalupa

***Abstract*—** In this paper we investigate the use of the Model Predictive Control (MPC) technique on a low power embedded computing platform. The control approach uses a quadratic optimization problem to compute the optimal control signal. The problem is solved subject to a linear model of the system and the physical limitations of the system. The optimization problem is solved online using the Fast Gradient method. The proposed controller has been implemented on a Stellaris Launchpad board with ARM Cortex processor. By means of two simulation studies we detail the software and the hardware aspects concerning a fast real-time MPC implementation. In the first example linear MPC is used for stabilization of a quadrotor model. In the second example nonlinear pH neutralization plant is controlled using fuzzy MPC algorithm.

***Keywords*—** Embedded Systems, Fast Gradient Method, Model Predictive Control, Multiple Models

## I. Introduction

MODEL predictive control (MPC) has gained a lot of interest of both academia and industry in the recent years. The main reason for the wide-scale adoption of MPC is its ability to handle constraints on inputs and states that arise in most applications. Moreover, MPC problem formulation enables direct inclusion of predictive information, allowing the controller to react to future changes in reference signal. MPC naturally handles processes with multiple inputs or outputs and its concept can be used with dynamic models of any dimension. MPC technology can now be found in a wide variety of application areas including chemicals, food processing, automotive, and aerospace applications [1]. With each new measurement the input to plant is determined by solving a finite horizon optimal control problem [2]. The problem is often in the form of a quadratic cost criterion with input constraints. Since the solution of the optimization problem is required every sample time the MPC was initially restricted to slow dynamics processes.

To avoid online optimization the solution of the control problem for different states can be pre-computed off-line. This explicit solution represent a piece-wise affine map over a partition of the state-space and can be stored efficiently in the form of a look-up table [3]. The explicit MPC offers reduction in online evaluation time but the primary limitation is that the complexity can grow quickly with the problem size, thus limiting the applicability of explicit MPC to small and medium-sized control problems.

The increase in computational power such as ARM Cortex processors and advances in optimization algorithms has opened new trend which brings MPC capabilities also to complex and fast systems.

Interior point method (IPM) and active set method (ASM) appear to be the most efficient approaches for online solving of quadratic programming problem. A fast implementation of Interior point method is reported in [4] and its applicability is demonstrated in simulation studies. The method exploits the particular structure of the MPC problem and considerably reduces the computation time of control action. The comparison of both methods for implementation of MPC is presented in [5]. Richter et al. [6] reported an online-optimization for systems with input constraints using Fast Gradient Method (FGM) developed by Nesterov [7]. The strategy to compute an upper bound for the maximum number of iterations needed to ensure a predefined accuracy is provided in [8]. Kogel and Findeisen [9] developed a method for computation of the gradient in Fast Gradient method exploiting the problem structure, which requires less memory and is faster than the standard method for large horizons. In [10] the problem of input quantization and how it can be exploited in order to determine a suboptimality level is shown. The authors also present a real application with Segway-like robot controlled using a hard real-time operating system and a low-cost microcontroller. There are several reports of implementations of the MPC on a chip with reduced computational power and memory. Bleris and Kothare present a real-time implementation of the MPC on a microcontroller for Glucose regulation in [11] and [12], where they used logarithmic number system and Newton method for optimization of the objective function. Due *et al*. demonstrated in [12] applications of Multi-parametric model based control on a chip for a slow industrial system but also for fast sampled active valve train engine. MATLAB framework for generating fast model predictive controllers for embedded targets such as ARM processors has been developed and tested on inverted pendulum in [13]. The optimization algorithm is based on the work by Stephen Wright [14] . With the development of cheap multi-core CPU in microcontrollers, the parallel computation might be the promising way for further decrease of

Jakub Novak is with Faculty of Applied Informatics, Tomas Bata University in Zlin, nam. T.G. Masaryka 5555, 76001 Zlin, Czech Republic (email: jnovak@fai.utb.cz)

Petr Chalupa is with Faculty of Applied Informatics Tomas Bata University in Zlin, nam. T.G. Masaryka 5555, 76001 Zlin, Czech Republic (email: chalupa@fai.utb.cz).

The research was supported by the European Regional Development Fund under the project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089. This assistance is very gratefully acknowledged.

computation time. In this work we focus on the implementation aspects of MPC on embedded systems where the quadratic programming problem is solved with FGM. Criterions for stability and robustness guarantees are beyond the scope of this paper.

The paper is structured as follows: Section 2 briefly repeats the MPC formulation. Section 3 describes the fast gradient algorithm. A description of the experimental setup can be found in Section 4. Section 5 and 6 contains the results for implementation of MPC on embedded system for two simulation studies. Finally, the main conclusions are summarized in Section 7.

## II. Problem Statement

In the MPC the control actions which optimize the forecasted process behavior are recalculated each sampling interval. The forecasted process behavior is based on a dynamic model of the process to be controlled. Thus, at each sampling instant an optimal control problem must be solved. Afterwards, the optimized control action is applied to the process until the next sampling instant when new states are available. Hence, MPC is sometimes also referred to as receding horizon control. The optimization problem is based on a time-invariant discrete process model, linear constraints and a convex quadratic objective function:

$$\min_{\Delta u(k),\Delta u(k+1),\dots,\Delta u(k+N_c-1)} J(k) \tag{1}$$

$$J(k) = \sum_{i=1}^{N_p}\left(\hat{y}(k+i) - y^r(k+i)\right)^T Q\left(\hat{y}(k+i) - y^r(k+i)\right) + \sum_{i=1}^{N_c} \Delta u(k+i-1)^T R \Delta u(k+i-1) \tag{2}$$

where $\hat{y}(k+i)$ is the $i$th step output prediction, $y^r(k+i)$ is the $i$th step reference trajectory, $\Delta u(k)$ is difference between $u(k)$ and $u(k-1)$, $R, Q$ are positive definite matrices and $N_p$ and $N_c$ are the prediction and control horizons, respectively. Only the first element of the optimal predicted input sequence $\Delta u(k)$ is applied to the plant:

$$u(k) = u(k-1) + \Delta u(k) \tag{3}$$

The criterion (1) can be rewritten into a condensed quadratic problem

$$\min_{\substack{\Delta u(k),\Delta u(k+1),\dots,\Delta u(k+N_c-1)\\ u(k)\epsilon U}} \left(\frac{1}{2}\Delta \boldsymbol{u}^T \boldsymbol{H} \Delta \boldsymbol{u} + \boldsymbol{f}(x)\Delta \boldsymbol{u}\right) \tag{4}$$

with input *u(k)* constrained in each step to a closed set $U$. The Hessian matrix $\boldsymbol{H}$ and vector $\boldsymbol{f(x)}$ depend on the cost criterion and system dynamics. Only input constraints are considered so the condensed problem has $2N_c$ inequality constraints, but only $N_c$ optimization variables. As usual in receding control strategy only the first input obtained by minimization of the quadratic criterion and satisfying the constraints is applied to system. The discrete linearized model is assumed in the form:

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) \end{aligned} \tag{5}$$

where $\boldsymbol{u(k)}$ is the vector of manipulated variables or input variables; $\boldsymbol{y(k)}$ is the vector of the process outputs and $\boldsymbol{x(k)}$ is the state variable vector. Using the linear model the model predictive controller would exhibit steady – state offset in the presence of plant/model mismatch or unmeasured disturbance due to lack of integral action. In order to introduce integral behavior, a new state variable vector is chosen to be:

$$\boldsymbol{x} = [\Delta \boldsymbol{x}(k)^T \;\; \boldsymbol{y}(k)]^T \tag{6}$$

Combining (6) and (7) leads to the following state-space model:

$$\begin{aligned} \begin{bmatrix}\Delta \boldsymbol{x}(k+1)\\ \boldsymbol{y}(k+1)\end{bmatrix} &= \begin{bmatrix}\boldsymbol{A} & \boldsymbol{0}\\ \boldsymbol{CA} & \boldsymbol{I}\end{bmatrix}\begin{bmatrix}\Delta \boldsymbol{x}(k)\\ \boldsymbol{y}(k)\end{bmatrix} + \begin{bmatrix}\boldsymbol{B}\\ \boldsymbol{CB}\end{bmatrix}\Delta \boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= [0 \quad 1]\begin{bmatrix}\Delta \boldsymbol{x}(k)\\ \boldsymbol{y}(k)\end{bmatrix} \end{aligned} \tag{7}$$

The predictor for new state vector $\boldsymbol{x}$ and control increment sequence $\Delta U = [\Delta \boldsymbol{u}(k), \Delta \boldsymbol{u}(k+1), \Delta \boldsymbol{u}(k+Nc-1)]^T$ for given horizons can be formulated in terms of vectors as:

$$\boldsymbol{Y} = \boldsymbol{K}\boldsymbol{x} + \boldsymbol{L}\Delta \boldsymbol{U} \tag{8}$$

where

$$\boldsymbol{Y} = \begin{bmatrix}\boldsymbol{y}(k+1)\\ \vdots \\ \boldsymbol{y}(k+N_p)\end{bmatrix} \Delta \boldsymbol{U} = \begin{bmatrix}\Delta \boldsymbol{u}(k)\\ \vdots \\ \Delta \boldsymbol{u}(k+N_c-1)\end{bmatrix} \tag{9}$$

and the relations for matrices $\boldsymbol{K}$ and $\boldsymbol{L}$ are:

$$\boldsymbol{K} = \begin{bmatrix}\boldsymbol{CA}\\ \boldsymbol{CA^2}\\ \vdots \\ \boldsymbol{CA}^{N_p-1}\end{bmatrix} \tag{10}$$

$$\boldsymbol{L} = \begin{bmatrix}\boldsymbol{L_1}\\ \boldsymbol{L_2}\end{bmatrix}, \boldsymbol{L}_1 = \begin{bmatrix}\boldsymbol{CB} & 0 & \dots & 0\\ \boldsymbol{CAB} & \boldsymbol{CB} & \dots & 0\\ \vdots & \vdots & \ddots & 0\\ \boldsymbol{CA}^{N_c-1}\boldsymbol{B} & \boldsymbol{CA}^{N_c-2}\boldsymbol{B} & \dots & \boldsymbol{CB}\end{bmatrix} \tag{11}$$

$$\boldsymbol{L}_2 = \begin{bmatrix}\boldsymbol{CA}^{N_c}\boldsymbol{B} & \dots & \boldsymbol{CA^2B} & \boldsymbol{C}(\sum_{j=0}^{1}\boldsymbol{A}^j)\boldsymbol{B}\\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{CA}^{N_p-1}\boldsymbol{B} & \dots & \boldsymbol{CA}^{N_p-N_c+1}\boldsymbol{B} & \boldsymbol{C}(\sum_{j=0}^{N_p-N_c}\boldsymbol{A}^j)\boldsymbol{B}\end{bmatrix} \tag{12}$$

The Hessian matrix $\boldsymbol{H}$ and vector *f(x)* from criterion can then be formulated as:

$$\boldsymbol{H} = \boldsymbol{L}^T\boldsymbol{Q}\boldsymbol{L} + \boldsymbol{R}, f(x) = -\boldsymbol{R}(\boldsymbol{Y_r} - \boldsymbol{K}x(k))^T\boldsymbol{L} \tag{13}$$

where $\boldsymbol{Y_r}$ represents the vector of reference signals on prediction horizon.

## III. FAST GRADIENT ALGORITHM

Efficient solution of the introduced quadratic programming problems with discrete-time linear model is a key feature for fast MPC control scheme. The optimization method utilized here to solve (1) is based on Nesterov's method also known as the Fast Gradient method, see [7]. The main benefit of the classical gradient schemes is that they do not rely on second order derivative information and take a damped steepest descent step in each iteration. Fast gradient method modifies this idea to yield faster convergence. Fast gradient method is easy to implement as it requires only to compute gradient and to perform projection operation into feasible set in each step. The optimization is started with an initial guess $\Delta u^0$ and stops after $i_{max}$ iterations, such that

$$J(\Delta u^{i_{max}}) - J(\Delta u^*) \leq \epsilon \tag{14}$$

where $J(\Delta u^*)$is the value of the optimal solution, $\epsilon > 0$ is the suboptimality level, and $\Delta u^{i_{max}}$ is called a suboptimal point. Theoretical bound for $i_{max}$ can be found in [6]. We can describe the iterative scheme of fast gradient method by algorithm.

**Fast Gradient method algorithm**

Requirements: Initial guess $\Delta u^0$, number of iterations $i_{max}$ , maximum and minimum eigenvalues of *H:* L, μ

1. Set $u_{old} = \Delta u^0$, w = $\Delta u^0$
2. For i=1 to $i_{max}$
   Compute $u = P_U(\mathrm{w}, 1/\mathrm{L})$
   Compute $w = u + c(u - u_{old})$
3. Return *u*

The constant *c* is defined as:

$$c > 0, c = \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}} \tag{15}$$

Numbers *L* and $\mu$ are Lipschitz constant of the gradient and convexity parameter respectively. Both are computed from the eigenvalues of the Hessian matrix. *L* is the maximal eigenvalue and $\mu$ is the minimal eigenvalue. Note that, the projected gradient step $P_U(w, 1/L)$ is an Euclidean projection of *w* resulting from the gradient step

$$u = w - \frac{1}{L}\nabla J(w) \tag{16}$$

into the feasible set *U.* This projection is very easy for box constraints but for general constraints it gets more computationally demanding. The gradient of the cost function (5) is simply:

$$\nabla J(u) = \boldsymbol{H}\Delta\boldsymbol{u} + \boldsymbol{f}(x) \tag{17}$$

The zero vector policy $\Delta u(k+i) = 0, i = 1,2,..,N_c - 1$ can be used as an initial feasible guess (*cold-starting*). However, using the solution of the previous optimal control problem, called *warm-starting* usually decreases computational effort.

## IV. SELECTED HARDWARE PLATFORM

The proposed fuzzy logic predictive controller was implemented on The Stellaris® LM4F120 board which is a low-cost evaluation platform for 32-bit ARM® Cortex™-M4F-based microcontrollers from Texas Instruments. The microcontroller runs at 80 MHz. The board has 32KB of SRAM memory, 256KB of flash memory and 2KB EEPROM. The Fig. 1 shows the device. For implementation of MPC controller the requirements for memory and evaluation speed must be considered. The board has only 32KB of RAM however model parameters can be pre-computed offline.

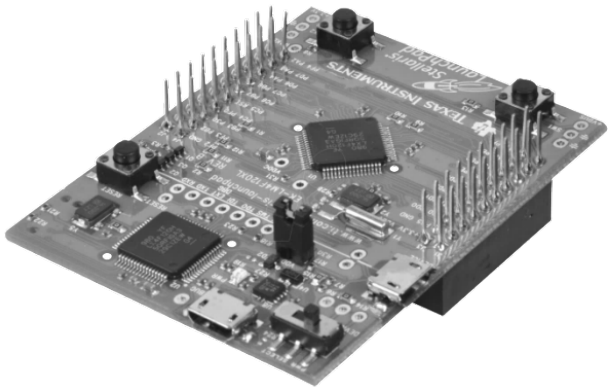

**Fig. 1** Stellaris LM4F120 board

## V. EXAMPLE I – STABILIZATION OF QUADCOPTER

In the first example the stabilization of the simulated quadrotor model is considered. A considerable amount of effort has been invested in controlling quadrotor helicopters and several control strategies have been tested. Classical PID approach, which assumed simplified dynamics of quadrotor, was developed in [15]. In [16] the authors proposed a backstepping controller using a quadrotor's simplified model. In [17], a method for controlling the quadrotor using a combination of MPC and H -infinity control was described and tested in simulations. In [16] controllers for altitude and attitude are developed based on PID and model predictive control and compared through flight tests.

The quadroter is an underactuated mechanical system with 6 degrees of freedom and only with 4 control inputs, hence it is described with 12 states (see Fig. 2). The first six states represent the position to Earth and speeds. The other six states are attitude and its change. In particular, *x* and *y* are the coordinates in the horizontal plane, *z* is the vertical position, $\phi$ is the yaw angle (rotation around the z-axis), $\theta$ is the pitch angle (rotation around the x-axis), and $\psi$ is the roll angle

(rotation around the y-axis). As described in Fig. 2, each of the four motors generate, respectively, four thrust forces and four torques. The quadcopter rotates on three axes by simply adjusting the angular velocity of each rotor in relation to the other three. If two rotors rotate at the same speed and other two rotate counterclockwise at the same speed the net yaw is zero. A difference in speeds between the two pairs of motors creates a net yaw. The roll or pitch rotation is caused by the difference in speeds between the two pairs of motors. For example, to pitch forward the back rotor is rotated at a greater velocity than the front motor.

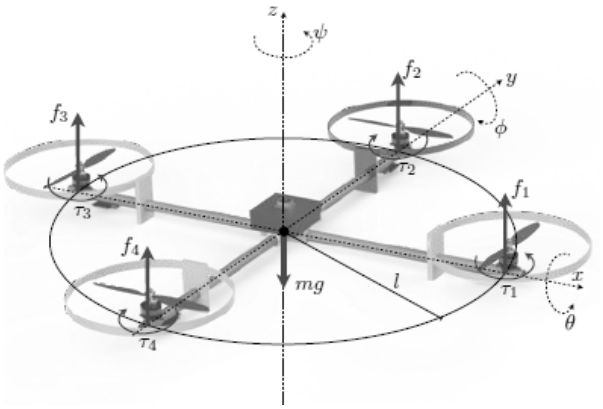


**Fig. 2** Quadcopter model

The dynamical model obtained through rotational transformation between the world frame and the quadcopter's body frame is given by:

$$\ddot{x} = (-u_1 sin\theta - \beta\dot{x})\frac{1}{m} \quad (18)$$

$$\ddot{y} = (u_1 cos\theta sin\phi - \beta\dot{y})\frac{1}{m} \quad (19)$$

$$\ddot{z} = -g + (u_1 cos\theta cos\phi - \beta\dot{z})\frac{1}{m} \quad (20)$$

$$\ddot{\theta} = \frac{u_2}{I_{xx}} \quad (21)$$

$$\ddot{\phi} = \frac{u_3}{I_{yy}} \quad (22)$$

$$\ddot{\theta} = \frac{u_4}{I_{zz}} \quad (23)$$

in which $g$ is the gravity acceleration, $m$ is the mass of the UAV, and the damping factor $\beta$ takes into account friction effects that affect the real vehicle and $I_{xx}, I_{yy}, I_{zz}$ are the components of diagonal inertia matrix of the airframe at its center of mass. The relation between the inputs and forces is given by:

$$u_1 = (f_1 + f_2 + f_3 + f_4), u_2 = (f_2 - f_4)l \quad (24)$$

$$u_3 = (f_3 - f_1)l, u_4 = (-f_1 + f_2 - f_3 + f_4)l \quad (25)$$

where $l$ is the distance between each motor and the center of gravity of the vehicle.

The aim is to control attitude and altitude through a Linear Time Invariant (LTI) Model Predictive Controller, hence the derivation of the implicit MPC solution is based on the model dynamics linearized around the equilibrium point represented by the hovering condition:

$$x, y, \theta, \phi, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\theta}, \dot{\phi}, \dot{\psi} = 0, z = 1 \quad (26)$$

$$u_1 = gm, u_2 = 0, u_3 = 0, u_4 = 0 \quad (27)$$

The model has 4 inputs $(u_1, u_2, u_3, u_4)$ and six outputs $(x, y, z, \theta, \phi, \psi)$ and 12 states.

$$x_1 = x, x_2 = \dot{x}, x_3 = y, x_4 = \dot{y}, x_5 = z, x_6 = \dot{z}, \quad (28)$$

$$x_7 = \theta, x_8 = \dot{\theta}, x_9 = \phi, x_{10} = \dot{\phi}, x_{11} = \psi, x_{12} = \dot{\psi} \quad (29)$$

While the derivation of the implicit MPC control solution is based on the linearised model dynamics, all simulations are performed taking into account the nonlinear model of the quadrotor (18–23). The extended discretized linear model was developed using (7) and (8) from linearized model. This operation adds 6 states to the original vector of states leading to the model with 4 inputs, 6 outputs and 18 states. The predictor matrices ***K***, ***L*** and hessian matrix ***H*** were stored in the flash memory. The memory requirements for given horizons are given as follows:

$$\boldsymbol{H}: \quad n_i * n_i * N_c * N_c * n_b \quad (30)$$

$$\boldsymbol{K}: \quad n_s * n_o * N_p * n_b \quad (31)$$

$$\boldsymbol{L}: \quad n_i * n_o * N_p * N_c * n_b \quad (32)$$

where $n_b$ is the number of bytes required to store a number. The 4 bytes are used in the example. The position and attitude information is assumed to be accurate in the model and the simulations. Hence, the effects of imprecise information to the flight of the quadcopter are beyond the scope of the paper. The following set of weights were applied in the optimization criterion:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (33)$$

$$R = diag([10,10,10,1,1,3]) \quad (34)$$

The prediction horizon is $N_p = 20$ steps, the control horizon is $N_c$, which, together with the choice of weights, allow obtaining a good compromise between tracking performance, robustness, and computational complexity. The sampling time of the controller is $T_s$ = 50ms. The reference trajectory is given by the hovering conditions:

$$x_r, y_r, \theta_r, \phi_r, \psi_r = 0, z = 1 \quad (35)$$

The initial conditions are set to:

$$x_0 = 0.1, y_0 = -0.2, z_0 = 1.2, \theta_0 = 0.1 \quad (36)$$

$$\phi_0 = -0.1, \psi_0 = 0.2 \quad (37)$$

with all the derivations set to 0.

The results were compared to an offline solution computed in MATLAB using the *quadprog* function and are presented in Table I. The table shows the value of cost criterion at the first

sampling point for different number of iterations and average time of execution. Cold-starting of the FGM was used in all the experiments. Warm–starting could further decrease the average evaluation times.

Table I Computational times and comparison with MATLAB solution

| Method | J(0) | *t* [ms] |
|---|---|---|
| FGM – 5 iterations | 15.5196 | 4.29 |
| FGM – 10 iterations | 13.4574 | 5.50 |
| FGM – 15 iterations | 12.8897 | 6.71 |
| FGM – 20 iterations | 12.8046 | 7.92 |
| FGM – 25 iterations | 12.6740 | 9.12 |
| FGM – 30 iterations | 12.4235 | 10.33 |
| FGM – 40 iterations | 12.2627 | 12.73 |
| FGM – 60 iterations | 12.2237 | 17.56 |
| MATLAB - quadprog | 12.2221 | - |

As the result shows solution computed by the fast gradient solver, depends on the maximum number of iterations. As the sampling time is 50ms, execution of the Fast Gradient algorithm with 60 iterations still leaves enough free time for the control loop including Kalman filter for state estimation and filtration. The system is simulated with FGM with 30 iteration and the control courses are compared with the optimal ones obtained with off-line MATLAB *quadprog* solver. The simulation results can be seen in Fig. 3. As can be seen in the simulations, the behavior of the output signals does not vary that much.

## VI. Example II – Fuzzy MPC Control of PH Neutralization

The simulated nonlinear process of neutralization was considered as a second example. The model is used in many studies to test the nonlinear control strategies [20]. To design an MPC controller for nonlinear process, the nonlinear process is modeled by a Takagi-Sugeno fuzzy system with linear functional consequents in the fuzzy rules and local linear models [21]. Different predictive controllers are designed for different rules (local sub-systems) and the global controller output is the fuzzy weighted integration of local ones. The models to be used in the control system design are taken to be discrete state-space models. By using a state-space model, the current information required for predicting ahead is represented by the state variable at the current time.

The control policy $\Delta u(k+i), i = 0,1,2,\dots,N_u$can be developed by first generating *m* sets of local control policies, where *m* is the total number of local models. The weighted sum of the local control policies gives the overall control policy:

$$\Delta u(k+i) = \sum_{j=1}^{m} \omega^j \Delta u^j(k+i) \quad (38)$$

where $\omega^j$ is the validity of *j*th model. Apparently, the validities of local models are normalized to unity:

$$\sum_{j=1}^{m} \omega^j = 1 \quad (39)$$

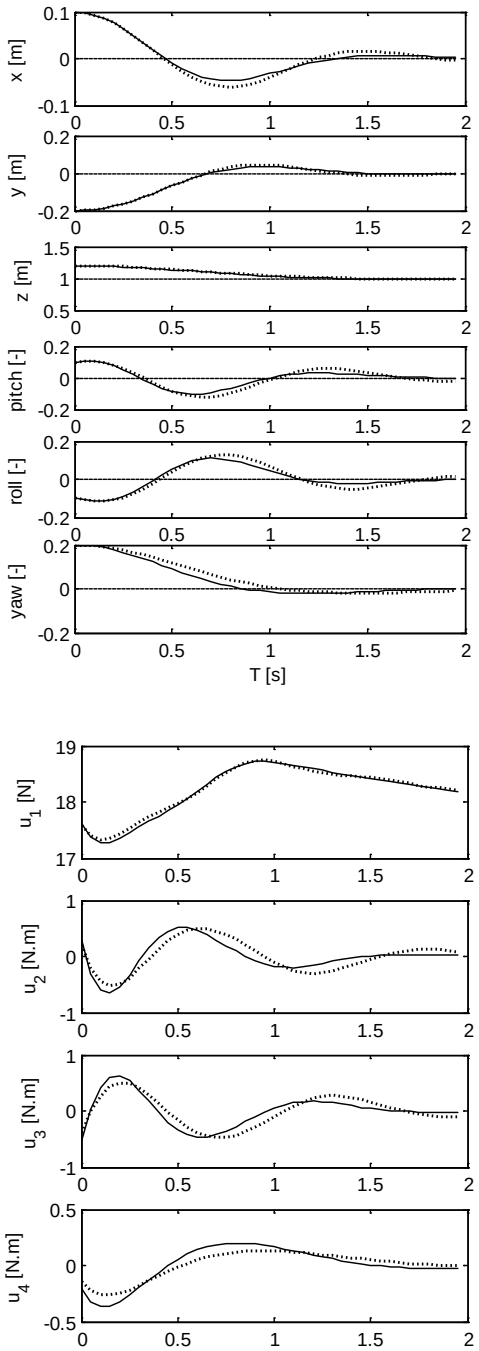


**Fig. 3** Simulation using quadprog and the fast gradient algorithm with 30 iterations (solid line– MATLAB quadprog, dotted line – Fast Gradient algorithm ).

Using the fuzzy approach the original nonlinear model is composed of *m* MIMO linear models with *j*th MIMO rule as an example,

$$R_j \quad IF\ \varphi \in Z^j THEN\ \boldsymbol{Y}_p^j(k) = K^j x(k) + \boldsymbol{L}^j \boldsymbol{U}^j(x) \quad (40)$$

where *j=1,..,m.* and $\varphi$ is the scheduling vector. Since the

consequent part of each rule is a linear equation, it is easy to design a linear controller for each rule. The global nonlinear controller is a fuzzy weighted integration of linear ones. The Fig. 4 shows the structure of multiple model predictive control.

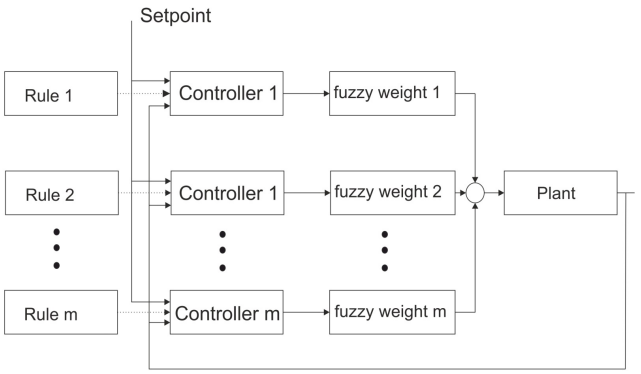


**Fig. 4** Structure of Multiple model control

The simulated system consists of a continuous stirred tank reactor (CSTR) in which neutralization reaction between a strong acid (HA) and a strong base (BOH) takes place in the presence of a buffer (BX). The system has three states, single output and single input.

$$x_1 = [A^-], x_2 = [B^+], x_3 = [X^-], y = pH, u = q_B \quad (41)$$

where $[A^-]$, $[B^+]$, $[X^-]$ are acid, base and buffer concentrations, respectively. The term $q_B$ represents the flow rate of the base. The scheme of the CSTR is depicted in Fig. 5. The process dynamics is given by the following set of differential equations:

$$\dot{x}_1 = \frac{q_A}{V}(x_{1,i} - x_1) - \frac{q_B}{V}x_1 \quad (42)$$

$$\dot{x}_2 = -\frac{q_A}{V}x_2 + \frac{q_B}{V}(x_{2,i} - x_2) \quad (43)$$

$$\dot{x}_3 = -\frac{q_A}{V}x_3 + \frac{q_B}{V}(x_{3,i} - x_3) \quad (44)$$

The pH value can be determined using the implicit equation:

$$[H^+] + x_2 + x_3 - x_1 - \frac{K_w}{[H^+]} - \frac{x_3}{1+\frac{(K_x)[H^+]}{K_w}} = 0 \quad (45)$$

where $pH = \log_{10}[H^+]$ and $K_w, K_x$ are the dissociation constants of water and buffer, respectively.

Table 1. Model Parameters

| symbol | parameter | value |
|---|---|---|
| $x_{1,i}$ | acid inlet concentration | $1.2 \times 10^{-3}$ mol/L |
| $x_{2,i}$ | base inlet concentration | $2.0 \times 10^{-3}$ mol/L |
| $x_{3,i}$ | buffer inlet concentration | $2.5 \times 10^{-3}$ mol/L |
| $K_x$ | buffer dissociation const. | $10^{-7}$mol/L |
| $K_w$ | water dissociation const. | $10^{-14}$mol$^2$/L$^2$ |
| $V$ | reactor volume | 2.5L |

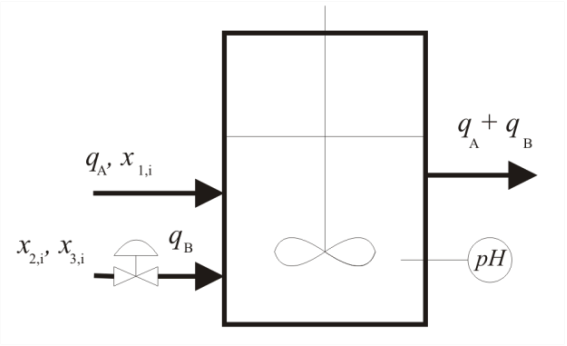


**Fig. 5** pH neutralization process

The system parameters used in this work were taken from [17] and are summarized in Table 1. The output equation is clearly strongly nonlinear. The titration curve and the gain variation that illustrate the nonlinearity of the pH neutralization process are depicted in Fig. 6.

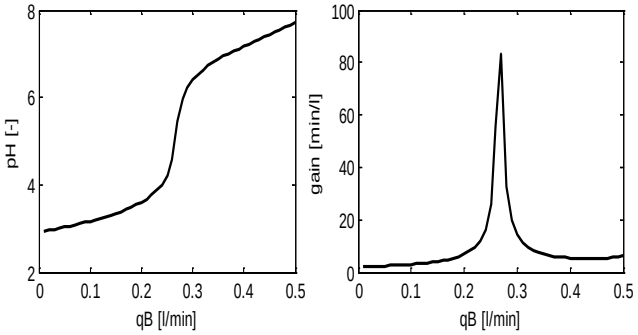


**Fig. 6** Titration curve and gain variation of pH neutralization process

The sampling of the estimation and control schemes was set to 30s due to the dynamics of the process and constraints of the process input are assumed to be *0≤u(k)≤0.5*. The concentrations (states) are assumed to be measured. Six fuzzy sets with triangular membership functions were used for approximation of the nonlinear process as shown in Fig. 4. The location of the models was obtained using C-means clustering. At these operation modes the nonlinear process was linearized to obtain parameters of local models. The fuzzy model is a good approximation of the process as presented in Fig. 7 which shows both steady-state characteristic of process and fuzzy membership functions. To account for high variation of gain of the process the weighting factor λ used in the predictive control cost function is also weighted using membership functions:

$$\lambda^j = \omega^j gain_j^2 \quad (46)$$

where the gain of the local model $M_i$ is computed:

$$gain_j = C_j(\boldsymbol{I} - \boldsymbol{A}_j)^{-1}\boldsymbol{B}_j \quad (47)$$

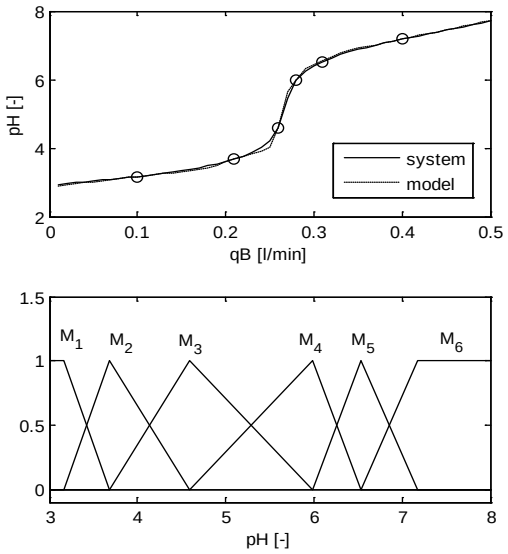


**Fig. 7** Steady-state characteristic and distribution of membership functions

The matrices from (11) $H \in R^{Np*Np*no*no}$, $K \in R^{Np*no*ns}$, $L \in R^{Np*Nc*no*ni}$ , where *no*, *ns*, *ni* are number of outputs, states and inputs, respectively are stored in the flash memory while variables such as the vectors f and the internal variables must be stored in RAM. For given number of local models *m* the memory requirements in bytes are given by

$$m * n_b * N_p * no * (N_p * n_o + n_s + N_c * n_i) \quad (48)$$

where $n_b$ is the number of bytes required to store a number. The 4 bytes are used in the example. The distribution of models in the operating space given by the centers of fuzzy sets is also stored in the flash memory.

The memory demands for online computation are given by the number of decision variables. The following vectors of size $N_u * n_i$ are needed for online computation: $f, y, w, u, u_{old}$ and the auxiliary vector of the same dimension to store the values of gradient.

The fuzzy controller output is given by the weighted sum of local controllers. Thus the constrained optimization problem must be solved separately for each local controller. The online computation of the fast gradient algorithm only requires the computation of gradient which is for the case of MPC control a matrix-vector multiplication ($H\Delta u$). Another two matrix-vector multiplications are needed to compute *f(x)*. The Fast Gradient algorithm is rather simple to implement as it requires only simple linear algebra such as matrix-vector multiplications, vector additions and comparisons. The initial guess $\Delta u(k+i) = 0, i = 0, \dots, N_u$ is used in the example as it represents always a feasible solution to the optimization problem. The Fig. 8 shows the time of evaluation of control input in a single sampling interval which represents call of the Fast Gradient method for all the local models. The values are averages of 20 executions with different states. The control horizon $N_c$ was set to the same value as prediction horizon. The MPC controller is implemented in plain C-code without the help of mathematical libraries BLAS/LAPACK. The control courses and weights of each model during the example simulation are depicted in Fig. 9.

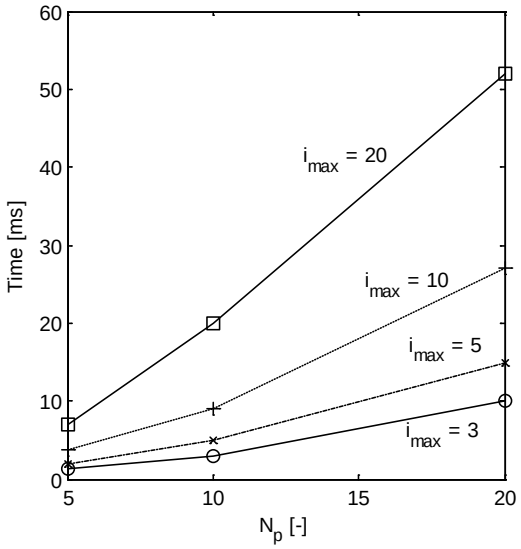


**Fig. 8** Execution times for different values of $i_{max}$

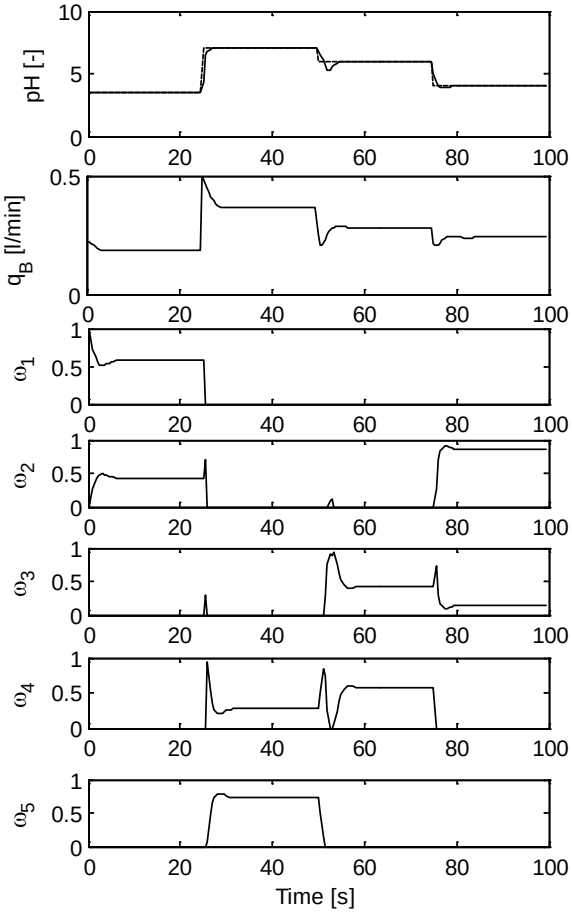


**Fig. 9** Control courses for pH neutralization example

Using multiple local models, the on-line nonlinear optimization can be avoided and simple quadratic programming problem is solved at each sampling interval. The

control performance of the proposed fuzzy control scheme with MPC based on multiple models is comparable to the performance obtained when a computationally demanding nonlinear optimization procedure is used online at each sampling instant within a nonlinear MPC controller [18].

## VII. Conclusion

In this work fast online model predictive control with input constraints for real-time implementation is considered. The implementation aspects of the Fast Gradient algorithm for finding optimal solution of the model predictive control problem are clarified with two simulation examples. In the first example a model predictive controller was applied for stabilization problem of a quadrotor helicopter. The results show that evaluation of the MPC control problem of the quadrotor system with 18 states and the prediction horizon of 20 steps and control horizon of 3 steps is manageable in less than 20ms without losing the accuracy of the solution. Enough free time remains for the control loop including Kalman filter for state estimation and filtration.

Fuzzy MPC and its performance are evaluated in the second example. Although the sampling frequency of the simulated process is in terms of second and the embedded system has much higher performance than required by the pH neutralization process, it allows applying low-power techniques that would decrease the power consumption. The example also confirms that nonlinear process modeled as set of fuzzy linear models with the algorithmic and numerical simplicity of fast gradient methods allows fast online optimization for medium length of prediction horizon.

As the work is focused on a fast MPC implementation, no frequency domain analysis to test robustness indicators such as gain, phase and delay margins is reported here.

## References


[1] S.J. Qin and B.J. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, pp. 733–764, 2003.

[2] J.M. Maciejowski, *Predictive Control with Constraints*, Prentice Hall, 2000.

[3] A. Alessio and A. Bemporad, "A Survey on Explicit Model Predictive Control," *Nonlinear Model Predictive Control*, vol. 384, pp.345-369, 2009.

[4] Y. Wang and S. Boyd, "Fast Model Predictive Control Using Online Optimization," IEEE Transactions on Control Systems Technology, vol. 18, pp. 267-278, March 2010.

[5] M.S.K. Lau, S.P. Yue, K.V. Ling and J.M. Maciejowski, A comparison of interior point and active set methods for FPGA implementation of model predictive control, In *Proceeding of the European Control Conference*, Budapest, 2009, pp. 156-161.

[6] S. Richter, S. Mariethoz and M. Morari, "High-speed online MPC based on a fast gradient method applied to power converter control," in *Proceedings of the 2010 American Control Conference*, Baltimore, 2010, pp. 4737–4743.

[7] Y. Nesterov, "A method for solving a convex programming problem with convergence rate 1/k2," *Soviet Math. Dokl.*, vol. 27, no. 2, pp. 372–376, 1983.

[8] S. Richter, C.N. Jones and M. Morari, "Real-Time Input-Constrained MPC Using Fast Gradient Methods," in *Proceedings of the 48th IEEE Conference on Decision and Control*, Shanghai, 2009, pp. 7387-7393.

[9] M. Kogel and R. Findeisen, "A fast gradient method for embedded linear predictive control," in *Proceedings of the 18th IFAC World Congress*, Milano, 2011, pp. 1362–1367.

[10] P. Zometa, M. Kogel, T. Faulwasser and R. Findeisen, "Implementation Aspects of Model Predictive Control for Embedded Systems," in *Proceeding of the American Control Conference*, Montreal, 2012, pp. 1205-1210.

[11] L.G. Bleris and M. V. Kothare, "Implementation of Model Predictive Control for Glucose Regulation on a General Purpose Microprocessor," in *Proceedings of the 44th IEEE Conference on Decision and Control*, Seville, 2005, pp. 5162-5168.

[12] P.D. Vouzis, L.G. Bleris, M.G. Arnold and M.V. Kothare, "A System-on-a-Chip Implementation for Embedded Real-Time Model Predictive Control", *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1006-1017, Sept. 2009.

[13] P. Dua, K. Kouramas, V. Du and E.N. Pistikopoulos, "MPC on a chip—Recent advances on the application of multi-parametric model-based control," *Computers and Chemical Engineering*, vol. 32, pp. 754-765, Apr 2008.

[14] J. Currie, A. Prince-Pike and D.I. Wilson, "Auto-Code Generation for Fast Embedded Model Predictive Controllers," in *Proceedings of the International Conference on Mechatronics and Machine Vision in Practice*, Auckland, 2012, pp. 122-128.

[15] G. Gol, N.F. Bayraktar and E. Kiyak, "PID Controlling of the Quadrotor and Sensor Performance Tests," *International Journal of Circuits, Systems and Signal Processing*, vol. 8, pp. 266-275, 2014.

[16] H. Zhen, X. Qi and H. Dung, "An Adaptive Block Backstepping Controller for Attitude Stabilization of a Quadrotor Helicopter," *WSEAS Transactions on Systems and Control*, vol. 8, pp. 46-55, 2013.

[17] G.V. Raffo and M.G. Ortega. "MPC with Nonlinear H-infinity Control for Path Tracking of a Quad-Rotor Helicopter," *Proceedings of the 17th IFAC World Congress*, Seoul, 2008, pp. 8564–8569.

[18] A. S. Imam and R. Bicker, Quadrotor Model Predictive Flight Control System, *International Journal of Current Engineering and Technology*, vol. 4, pp. 355-365, February 2014.

[19] A. Bemporad and C. Rocchi, Decentralized Linear Time-Varying Model Predictive Control of a Formation of Unmanned Aerial Vehicles, *Proceedings of 50th IEEE Conference on Decision and Control*, 2011, pp. 7488-7493.

[20] J. Novak, P. Chalupa, V. Bobal, Multiple model modeling and predictive control of the pH neutralization process, *International Journal of Mathematical Models and Methods in Applied Sciences*, vol. 5, 2011, pp. 1170-1179.

[21] N. Li, S-Y. Li, Y-G. Xi, Multi-model predictive control based on the Takagi–Sugeno fuzzy models: a case study, *Information Sciences*, vol. 165, 2004, pp. 247–263.

[22] S.I. Biagiola, J.L. Figueroa, State Estimation in Nonlinear Processes-Application to pH Process Control, *Industrial & Engineering Chemical Research*, vol. 41, 2003, pp. 4777-4785.

[23] J. Novak, P. Chalupa, Nonlinear State Estimation and Predictive Control of pH Neutralization Process, *Advances in Intelligent Systems and Computing*, vol. 210, 2013, pp 285-294.



**Jakub Novak** was born in Zlín, Czech Republic in 1978. He is a Researcher at Faculty of Applied Informatics of Tomas Bata University in Zlín, Czech Republic. He graduated from Faculty of Technology of the same university with an MSc in Automation and Control Engineering in 2002 and he received a PhD in Technical Cybernetics from Faculty of Applied Informatics in 2007. He is a researcher at the CEBIA-Tech research center at Tomas Bata University in Zlin. His research interests are modeling and predictive control of the nonlinear systems.

**Petr Chalupa** graduated in 1999 from Brno University of Technology and received the Ph.D. degree in Technical cybernetics from Tomas Bata University in Zlin in 2003.

He is a researcher at the CEBIA-Tech research center at Tomas Bata University in Zlin. His professional interests are adaptive and predictive control of real-time systems.