EMBEDDED MODEL PREDICTIVE VIBRATION CONTROL ON LOW-END 8-BIT MICROCONTROLLERS VIA AUTOMATIC CODE GENERATION

Gergely Takács
Slovak University of Technology, Faculty of Mechanical Engineering, Institute of Automation, Measurement and Applied Informatics, Nám. slobody 17, 812 31 Bratislava, Slovakia, email: gergely.takacs@stuba.sk

Pablo Zometa, Rolf Findeisen
Otto-von-Guericke-University Magdeburg, Laboratory for Systems Theory and Automatic Control, Magdeburg, Germany: email: {pablo.zometa, rolf.findeisen}@ovgu.de

Boris Rohal'-Ilkiv
Slovak University of Technology, Faculty of Mechanical Engineering, Institute of Automation, Measurement and Applied Informatics, Nám. slobody 17, 812 31 Bratislava, Slovakia, email: boris.rohal-ilkiv@stuba.sk

Model predictive control is a promising approach for vibration attenuation, as it allows the direct consideration of constraints and the use of system models for predicting future behavior. However, the straightforward application of model predictive control is in general challenging, as the efficient implementation requires expert insight, especially for fast systems on low-end embedded platforms. This paper investigates the practical feasibility and deployment of model predictive control for the active vibration attenuation of a cantilever beam using a low-end, low-cost 8-bit microcontroller. The control aim is to suppress vibrations while respecting the input constraints due to the depolarization limits of the piezoceramic actuators. The embedded model predictive control code is automatically generated using the $\mu$AO-MPC code generation tool. The resulting implementation is real-time feasible even on the utilized low-cost, low-end Arduino UNO electronics prototyping platform, which is based on the widely used Atmel ATmega328, a 8-bit AVR architecture reduced instruction set processor. The memory demand, computation time and performance are examined under various settings. As shown, real-time execution of the $\mu$AO-MPC generated code for horizon lengths of up to 7 steps for a second order model and a 20 ms sample time are computationally feasible even on the low cost hardware used here.

1. Introduction

Until recently, model predictive control (MPC)—a control strategy that is based on the on-line repeated solution of an optimal control problem—was limited to slow dynamics and the use of powerful computing hardware [1]. Virtually every application field—including active vibration control—may benefit from the increased control performance and the constraint handling features offered by MPC. However, the computational requirements and implementation burden are often prohibitive, especially when compared to traditional control methods such as positive position feedback (PPF) or linear quadratic control (LQ). This makes the implementation of MPC for fast and cost sensitive applications, such as many active vibration control problems, difficult. This is especially true if, instead of the customary industrial computers, the target platform is only a low-cost microcontroller.
unit (MCU) with restrictive speed and memory. While this can be partially overcome by the use of powerful modern MCUs, it is important to realize that the 8-bit reduced instruction set MCU of yesteryear are making a comeback due to their low unit cost and size [2]. This comeback is especially driven by the desire to have computation and communication in every device, ranging from power tools, smart heating devices to toys. This desire raises the question, if it is possible employ MPC on such limited platforms, e.g. for vibration control, in a cost efficient way and without the demand of expert hand-coding.

Significant advancements with respect to the computational efficient and ease of implementation of MPC have been made over the last decade. Exploiting the problem structure of the predictive control problem, computational speeds have drastically improved by many orders of magnitude [3, 4].

By now, constrained MPC has been deployed on numerous platforms, from general purpose processors [5, 6], to field-programmable gate arrays (FPGAs) [7, 8], and application-specific integrated circuits (ASICs) [9, 10]. Efficient MPC for active vibration control has been considered specifically on a digital signal processor (DSP) using direct machine code [11], powerful MCU [12] and even via analog electronics by approximation [13].

Besides tailored solution approaches and the availability of powerful microcontrollers, the easy implementation of MPC on embedded hardware is by now supported by a series of code generation tools. These either provide tailored quadratic programming solvers for MPC or automatically generate an easily deployable C language source from a high-level problem definition. Examples are the Multi-Parametric Toolbox [14], the ACADO Toolkit [15], the Matlab MPC toolbox, FORCES [16], or µAO-MPC [6], which is considered in this study.

In this paper, we show that it is possible to implement MPC for flexible structural vibration problems even on low-cost 8-bit microcontroller units using auto-generated code. To the best of our knowledge this paper presents the first use of linear model predictive control via auto-code generation on a low-cost 8-bit microcontroller.

2. Experimental setup

We consider the active vibration control of a cantilever beam as shown in Figure 1(a). The aluminum cantilever beam is equipped with a pair of MIDÉ QP16n piezoceramic transducers as actuators. The transducers are mounted close to the fixed part of the beam to maximize the bending moment for the first resonant mode. They work as actuators connected counter-phase to the same input signal via a MIDÉ EL-1225 amplifier. The feedback signal is measured using a Keyence LK-G82 laser triangulation head mounted at the free end of the beam. The measurement head is connected to a Keyence LK-G3001V central processing unit. Furthermore, the beam assembly includes a linear motor, which acts as an instantaneous disturbance source to the beam. Upon receiving a digital signal via an amplifier and control circuit, this stinger mechanism pushes the beam away from its equilibrium, then retracts to its original position after the signal disappears. The role of the stinger mechanism is to provide repeatable transient disturbances to the structure. For feedback control and data processing an Atmel ATmega 328p 8-bit RISC-based microcontroller unit with 32 kB flash memory and 2 kB static random access memory (SRAM), clocked to 16 MHz (16 MIPS) is used. It is part of the Arduino UNO R3 electronics prototyping platform, see Figure 1(b). For actuation and sensing the microcontroller is connected to an NXP PCF8591 8-bit digital to analog and analog to digital converter (DAC, ADC) via the I2C-bus. The DAC/ADC chip is part of a prototyping board and its input is protected by a voltage clamp (Fig. 1(c)). The prototyping board is powered and programmed via USB using a personal computer.

The 0–5 V output of the controller is connected to an Advantech ADAM 3014 signal processor. The signal processor amplifies the received signal by a factor of two and shifts it by -5 V down. The output is amplified to the maximal ±100 V working voltage of the piezoceramics via a capacitive amplifier. The output of the laser triangulation system is scaled to 0–5 V and is connected to the ADC
The 23rd International Congress of Sound and Vibration

(a) Mechanical and electrical setup of the active cantilever beam.

(b) Microcontroller prototyping board.

c) ADC/DAC prototype shield.

Figure 1: Active cantilever beam and the microcontroller running the MPC algorithm via the voltage clamp.

Data logging and excitation system Experiment management and data logging work independently from the controller implemented on the MCU. This portion of the experimental hardware is only responsible to record the input and output of the controller along with execution timing; and to govern the activity of the stinger mechanism. The software for this subsystem is created in Matlab/Simulink 8.3 (v. R2014a) on the so-called host PC then, after automatic code compilation, is transferred from to the target PC via TCP/IP. The target PC running the Simulink Real-Time suite contains a National Instruments PCI-6030E laboratory measurement card that gathers input, output data and total execution timing (TET) from the MCU. The TET signal is measured for the overall step and for the MPC algorithm using two digital outputs from the MCU, which are connected to the gate input of the counter peripherals of the measurement card. The stinger mechanism is also connected to a digital output of the measurement card (Fig. 2).

Figure 2: Overall experimental setup.
3. Control objective, modeling and basic MPC formulation

The control objective is to actively damp the vibration of the beam via piezoceramic actuators. As the first mode of vibration dominates the dynamic response of this structure, the beam is approximated by a one degree of freedom linearly driven mass-spring-damper. The resulting mathematical model takes the standard form \( \ddot{q}(t) + 2\zeta \omega \dot{q}(t) + \omega^2 q(t) = cu(t) \), where \( q(t) \) (m) is position, \( \omega \) (rads\(^{-1}\)) is the first natural frequency, \( \zeta \) (-) is the damping ratio, \( c \) is the voltage-force conversion constant and \( u(t) \) (V) is the input signal. Choosing position and velocity as state variables \( x(t) \) leads to the state space equations \( \dot{x}(t) = \begin{bmatrix} 0 & -\omega \zeta & -\omega^2 \\ -\omega \zeta & 1 & 0 \\ -\omega^2 & 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(t) \), and \( y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \).

**MPC formulation:** The control objective is to damp the vibrations of the beam using model predictive control. We do not provide an in depth introduction to model predictive control, we rather refer to [1][7][18]. Basically, an optimal control problem is solved at the sampling times, that is formulated in terms of a cost function \( J \), a model of the system and the constraints. The first part of the resulting optimal input sequence is applied and the procedure is repeated if new measurements become available. We use a discrete time formulation of MPC, considering a sampling time of \( T_s \), i.e. all \( t = T_s k \) time instances an optimal control problem, looking \( n_p \) steps into the future is solved:

\[
J(u_k, x_k) = \sum_{i=0}^{n_p-1} (x_{k+i}^T Q x_{k+i} + u_{k+i}^T R u_{k+i}) + x_{k+n_p}^T P x_{k+n_p},
\]

subject to:

\[
x_{k+i+1} = Ax_{k+i} + Bu_{k+i}, \quad i \geq 0, \quad x_k = x(T_s k)
\]

Here \( x_k = x(T_s k) \) is the measured or estimated state of the plant at time \( t_k = T_s k \). The matrices \( A, B \) and \( C \) are the discretized state-space model. The states, inputs and the last state are penalized by \( Q, R \) and \( P \) respectively, while \( \bar{u} \) and \( \underline{u} \) define the upper and lower input constraints. Penalizing the deviation of the states from the origin allows to formulate the goal to surpise oscillations of the beam. The MPC problem in Eq. (1) can be reformulated in form of a quadratic program (QP) \( J^*(x_k) = \min_{u_k} \left\{ \frac{1}{2} \bar{u}_k^T H \bar{u}_k + \bar{u}_k^T G x_k \right\} \) subject to \( \underline{u} \leq u_k \leq \bar{u}, \) where \( H \) and \( G \) are uniquely determined for a nominal system. Solving this convex QP, if the matrices are chosen suitably, leads to the optimal input sequence \( \bar{u}_k \) with the corresponding minimum cost \( J^*(x_k) \). The first element of the input is applied to the system and the optimization is repeated once new state measurements are available.

The solution of the QP must obtained in real-time for each time sample. This poses implementation challenges, especially, if a system with a relatively fast dynamics is to be controlled using a low-cost microcontroller, as is the case here.

**MPC code generation:** The embedded code for the MPC algorithm was automatically generated using \( \mu \text{AO-MPC} \) from a high-level definition file specifying the prediction horizon, numeric type and precision, linear model and penalty matrices. This definition file was then compiled using \( \mu \text{AO-MPC} \), Python 2.7.6, and numpy 1.8.2. The resulting, efficient target C is tailored for the given problem. \( \mu \text{AO-MPC} \) uses for a variant of Nesterov’s fast gradient method (FGM) for the numerical solution of the QP, exploiting the specific problem structure.

4. Hardware experiments - detailed setup

**Output measurements and state estimation:** The auto-generated MPC code forms the heart of the embedded control system. To obtain the required state information, the analog input from the laser triangulation system was re-scaled linearly to the position measurement in meters. An 8-th order finite impulse response (FIR) filter was used to remove the static component of the position signal. The filter was implemented with a circular buffer to increase computational efficiency. The state of the system \( \hat{x}_k \) was estimated using a linear Kalman filter, which was then fed to \( \mu \text{AO-MPC} \).
Overall feedback and fixed point numerics: The non specific MPC control algorithm was also coded in C. The complete control algorithm was compiled and flashed onto the MCU using Arduino IDE 1.6.5. Note, that the MCU used in this study does not contain a floating point unit (FPU), therefore, this numeric type must be emulated by software using the compiler. Furthermore, the Arduino IDE does not emulate a double precision floating point number for ATmega microcontroller family, so the entire algorithm depends on a single precision (32-bit) floating point number representation, that is likely to reduce the performance of the MPC controller.

Each complete experimental run lasted a total of $T = 50$ s, with the stinger mechanism pushing the beam to the initial position of $\approx 10$ mm in 5 s periods, then retracting and leaving the beam to settle back to its equilibrium in open- or closed-loop. All experiments used the same system model and penalty matrices. We investigated the effect of changing the horizon, solver iteration number and the use of warm starting on the execution time and vibration attenuation performance. Data featured here is related to the entire duration of the experiment.

Control parameters: A common sampling time of $T_s = 20$ ms was used in all experiments. The coefficients of the transfer function of the FIR filter were set to $H(z) = 0.875 - 0.125z^{-1} - \ldots - 0.125z^{-7}$. Both the Kalman filter and the MPC algorithms employ the discretized version of the model with $\omega = 50.9$ rads$^{-1}$, $\zeta = 0.005$ - and $c = 5.9$ NkV$^{-1}$kg$^{-1}$. The Kalman filter was initialized with a zero initial state estimate $\hat{x}_0 = [0 \ 0]^T$ and a zero state estimate covariance $P_f = \text{diag}[0 \ 0]$. Process noise in the Kalman filter was set to $Q_f = \text{diag}[0.01 \ 0.1]$ and measurement noise to $R_f = 0.01$. The inputs of the MPC controller were penalized by $R = 1$, the states by $Q = \text{diag}[1 \ 1]$ and as final state penalty the solution of the discrete-time Lyapunov equation was used. The input constraints represent the polarization limits of the piezoceramic material: $-100 \leq \overline{u}_k \leq 100$ V.

5. Results and discussion

The results of a typical experiment are shown in Fig. 3. As can be seen, MPC allows to suppress the vibration efficiently. However, as this is not the main focus of the paper, no comparison to other control approaches are provided. Instead we stress and examine computational demand and efficiency of the generated code on the limited computing power of the MCU. The bottom of Fig. 3 shows the

![Figure 3: Experimental results. Top: Open-loop and MPC controlled deflection of the beam after an excitation. Middle: Resulting input. Grey bars denote the input constraints, in red the “exact” solution of the QP using quadprog is depicted. Bottom: Total TET and TET of the MPC controller.](image-url)
TET for a horizon of $n_p = 7$ steps for 9 FGM iterations. This, corresponds to the maximum possible prediction horizon combined with the maximal iteration number for a sampling time of $T_s = 20$ ms sample. The horizon length covers an entire oscillation period for the first resonance frequency. Notice that the non MPC related computations, including the FIR and Kalman filters required only 2 ms computation time.

The relationship between the average execution time, different horizon lengths, and FGM iterations is shown in Figure 4. Maximal and minimal observed times are signified by the dash-dot lines. Figure 4(a) shows execution times for various prediction horizons for a fixed number of 9 FGM iterations, while 4(b) shows the influence of the FGM iterations for a fixed prediction horizon of $n_p = 7$. As is visible, the execution times do not vary greatly from the average, while the MPC algorithm takes up most of the computation cycle as expected. Because the 16 MHz clocking of the MCU is typical for the class of devices considered here (with 20 MHz maximum), horizon length or better precision achieved by increased FGM iteration number should not be sacrificed to decrease computational time for the sake of simpler hardware. There is, however, the possibility to control structures that are stiffer and require a shorter sampling times. In this case, while a horizon of 4–5 steps is feasible for a $T_s = 10$ ms sampling based on Fig. 4(a), the shorter horizon length will not cover the resonant mode of the structure, possibly leading to lower vibration suppression performance. Besides the processor speed, the required static flash (ROM) and dynamic memory (RAM) are of importance for embedded control on low-cost MCUs. Table 1 shows the memory requirements for various prediction horizons. Enabling the warm starting feature lead to no increase in the flash memory footprint, while it increases the RAM requirements by 8 bytes in all cases. As expected, changes in the FGM iteration number do not affect memory usage. It is clear, that the static memory footprint does not pose a significant challenge for most processors available on the market. The $n_p = 7$ steps horizon occupies only about 30% of the 32 kB total available static memory of the ATmega 328p. On the other hand, the available dynamic memory proves to be more restricting for the implementation of MPC on low cost micro-controllers. The $n_p = 7$ steps horizon nearly uses all (89%) of the available 2 kB dynamic memory of the ATmega 328p. In case a lower prediction horizon is deemed to be sufficient, lower class and price MCU (eg. ATmega 168, etc.) with 1 kB dynamic memory could handle an $n_p = 3$ steps horizon for the given problem.

Table 1: Memory requirement in bytes for increasing prediction horizons.

<table>
<thead>
<tr>
<th>Horizon:</th>
<th>No MPC</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static (Flash):</td>
<td>6420</td>
<td>9632</td>
<td>9806</td>
<td>9912</td>
<td>10012</td>
<td>10136</td>
<td>10284</td>
<td>10456</td>
<td>32768</td>
</tr>
<tr>
<td>Dynamic (RAM):</td>
<td>363</td>
<td>733</td>
<td>853</td>
<td>997</td>
<td>1165</td>
<td>1357</td>
<td>1573</td>
<td>1813</td>
<td>2048</td>
</tr>
</tbody>
</table>
Reducing the horizon length to fit the controller into less dynamic memory or limiting the FGM iteration number to achieve a shorter sample time shall, of course, affect the vibration suppression performance. How serious is the loss of performance, given one needs to make such concessions? Tables 4-5 compare the output and input performance of the controller, as measured by the root of sum of squares (RSS) indicator expressing the difference between the measured and zero position ($Y_{RSS}$) and voltage ($U_{RSS}$).

Table 2: Performance indicators for increasing prediction horizons, and fixed $I=9$ FGM iterations.

<table>
<thead>
<tr>
<th></th>
<th>Free</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>No WS</td>
<td>$Y_{RSS}$ (mm):</td>
<td>559</td>
<td>284</td>
<td>280</td>
<td>277</td>
<td>276</td>
<td>275</td>
<td>276</td>
</tr>
<tr>
<td></td>
<td>$U_{RSS}$ (V):</td>
<td>-</td>
<td>8138</td>
<td>8023</td>
<td>7963</td>
<td>7785</td>
<td>7842</td>
<td>7893</td>
</tr>
<tr>
<td>WS</td>
<td>$Y_{RSS}$ (mm):</td>
<td>-</td>
<td>282</td>
<td>281</td>
<td>277</td>
<td>277</td>
<td>276</td>
<td>275</td>
</tr>
<tr>
<td></td>
<td>$U_{RSS}$ (V):</td>
<td>-</td>
<td>8102</td>
<td>8038</td>
<td>7915</td>
<td>7815</td>
<td>7796</td>
<td>7894</td>
</tr>
</tbody>
</table>

Table 4 summarizes a case with increasing prediction horizons and the maximal possible number of FGM iterations achievable on the given hardware. This would indicate a scenario, where a reduction of dynamic memory is desired without the need to shorten sample times. Similarly, Table 2 explores the performance for various horizon lengths but assuming a fixed $i = 9$ FGM iterations; meaning a reduction in both memory footprint and speed (transferrable to shorter sample times). Finally, Table 3 demonstrates performance indicators for a fixed $n_p=7$ steps prediction horizon and with various FGM iteration numbers, suitable to achieve shorter sampling times with the same MCU.

Table 3: Performance indicators for increasing prediction horizons, for fixed $N=7$.

<table>
<thead>
<tr>
<th></th>
<th>Free</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>No WS</td>
<td>$Y_{RSS}$ (mm):</td>
<td>559</td>
<td>274</td>
<td>274</td>
<td>270</td>
<td>269</td>
<td>266</td>
<td>265</td>
<td>265</td>
<td>264</td>
</tr>
<tr>
<td></td>
<td>$U_{RSS}$ (V):</td>
<td>-</td>
<td>7408</td>
<td>7527</td>
<td>7487</td>
<td>7494</td>
<td>7540</td>
<td>7491</td>
<td>7569</td>
<td>7635</td>
</tr>
<tr>
<td>WS</td>
<td>$Y_{RSS}$ (mm):</td>
<td>-</td>
<td>279</td>
<td>279</td>
<td>277</td>
<td>278</td>
<td>277</td>
<td>274</td>
<td>274</td>
<td>272</td>
</tr>
<tr>
<td></td>
<td>$U_{RSS}$ (V):</td>
<td>-</td>
<td>7728</td>
<td>7703</td>
<td>7599</td>
<td>7640</td>
<td>7646</td>
<td>7658</td>
<td>7704</td>
<td>7756</td>
</tr>
</tbody>
</table>

It is somewhat surprising that, except for very low horizon lengths, the vibration suppression performance remains relatively constant despite of the reduced horizon lengths or FGM iterations. This, of course, can be easily accounted for by the quantization errors of the ADC/DAC device and the limited single floating point numeric precision, along with the mechanical and electric noise present in an average laboratory setting. On the upside, this also means that, if concessions regarding the horizon length or FGM iterations need to be made for the previously listed reasons, they may not seriously affect the performance in a realistic implementation. With the exception of the single step horizon which proved to be (numerically) unstable, the rest has shown a good performance. The effect of warm starting is negligible on the performance, thus combined with the slightly higher execution times its use is unjustified for such and similar applications.

Table 4: Performance for increasing prediction horizons, and maximal possible FGM iterations.

<table>
<thead>
<tr>
<th></th>
<th>Free</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>No WS</td>
<td>$Y_{RSS}$ (mm):</td>
<td>559</td>
<td>286</td>
<td>281</td>
<td>273</td>
<td>273</td>
<td>275</td>
<td>275</td>
</tr>
<tr>
<td></td>
<td>$U_{RSS}$ (V):</td>
<td>-</td>
<td>13452</td>
<td>8363</td>
<td>8187</td>
<td>8031</td>
<td>8045</td>
<td>7983</td>
</tr>
<tr>
<td>WS</td>
<td>$Y_{RSS}$ (mm):</td>
<td>-</td>
<td>288</td>
<td>281</td>
<td>274</td>
<td>271</td>
<td>274</td>
<td>274</td>
</tr>
<tr>
<td></td>
<td>$U_{RSS}$ (V):</td>
<td>-</td>
<td>12063</td>
<td>8402</td>
<td>8197</td>
<td>7993</td>
<td>8028</td>
<td>7972</td>
</tr>
</tbody>
</table>

6. Conclusion

The implementation of MPC on computationally limited platforms, as for example often used for vibration control, is in general challenging. It typically requires expert knowledge and hand coding.
As shown in this work, by exploiting code generation tools for MPC, for example µAO-MPC it is possible to use MPC even on low-cost 8-bit microcontrollers.

Acknowledgements

G. Takács and B. Rohal’-Ilkiv gratefully acknowledge financial support by the Slovak Research and Development Agency (APVV), contracts APVV-14-0399 and APVV-14-0244; and the Scientific Grant Agency (VEGA) of the Ministry of Education, Science, Research and Sport of the Slovak Republic, contract 1/0144/15. The authors would like to thank for the financial assistance from the STU Grant scheme for the Support of Excellent Teams of Young Researchers.

REFERENCES