Real-Time Infinite Horizon Adaptive/Predictive Control for Smart Home HVAC Applications

Michael Short
Electronics & Control Group
Teesside University
Middlesbrough, UK
m.short@tees.ac.uk

Abstract

The potential benefits that adaptive and predictive control schemes can bring to building environment control applications such as HVAC have been well documented in recent years. One significant drawback of these schemes is that in many cases they require large computational burdens, especially in adaptive situations and when constraints are present. A secondary drawback is the large number of parameters that have to be potentially tuned. As high-bandwidth real-time embedded computing platforms and advanced levels of control knowledge may not be realistic expectations for many users, the applicability of advanced schemes is currently limited for most Smart home applications. In this paper, a prototype infinite-horizon 'plug-and-play' adaptive MPC scheme is presented for input-constrained HVAC applications. The scheme has been optimized for real-time implementation on small, low-cost embedded processors and requires little user preconfiguration. Preliminary hardware-in-the-loop based experimental results indicate good performance of the technique coupled with extremely low overheads.

1. Introduction

The potential benefits that adaptive and predictive control schemes can bring to building environment control applications such as Heating, Ventilation and Air Conditioning (HVAC) have been well documented in recent years [1-5]. In addition to 'classical' adaptive/predictive control schemes, Model Predictive Control (MPC) has also seen successes. MPC schemes employ dynamic models of a process in conjunction with the use of on-line optimization to compute an optimal sequence of input moves to minimize the predicted future values of an objective function [6][7]. When used with on-line parameter estimation schemes, adaptive MPC becomes a powerful (albeit computationally demanding) control solution which is well suited to optimal control of HVAC equipment [7][8]. The principal components of a typical adaptive MPC implementation for HVAC plant are as shown in Fig. 1, and consist of an online parameter estimator and adaption mechanism, a predictor of the future process behavior given the current (known) state, and an optimizer to determine the optimal constrained control sequence. MPC used a receding-horizon approach, in that the first control in this sequence is applied, and the estimation, adaptation and optimizations are again carried out at the next time step with knowledge of the actual process evolution [6][7].

Drawbacks of such a scheme include the large computational burden in constrained and/or adaptive situations, and the large number of parameters that potentially have to be tuned [6][7]. As high-bandwidth real-time computing facilities and advanced levels of control knowledge may not be realistic expectations for many enthusiasts, the applicability of such advanced schemes is currently severely limited for most 'smart' home and office applications. In this paper, a prototype of an infinite-horizon adaptive 'plug-and-play' MPC scheme is presented for input-constrained HVAC applications. The scheme has been optimized for real-time implementation on small, low-cost embedded processors and requires minimal user preconfiguration. As will be shown, the technique performs well in the presence of both parameter variations and unknown disturbances, whilst exhibiting extremely low computational and memory overheads.

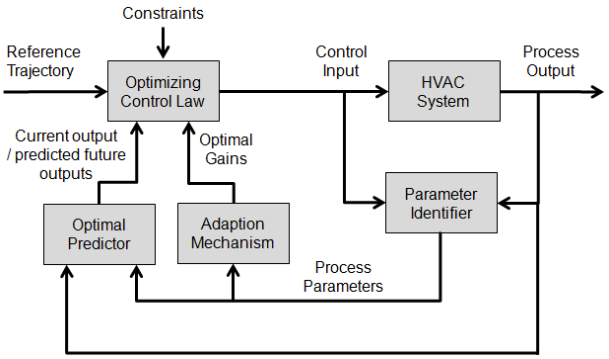

Figure 1. Structure of the Adaptive/Predictive Controller for HVAC Systems.

978-1-4673-4737-2/12/$31.00 ©2012 IEEE

The remainder of this paper is structured as follows. Section 2 presents the assumed model of the process, and shown how predictions of the future process behavior can obtained. Section 3 presents the optimal predictive control law. Section 4 presents the online identification mechanism that has been developed to enable the adaptation of the controller. Section 5 presents preliminary experimental results using a microcontroller-based implementation of the proposed control scheme, and the paper is concluded with area for future work in Section 6.

2. Process Model

2.1. Approximate FOPDT Model

As has been demonstrated in many previous studies (e.g. [3-5]), the short-term dynamic behavior between the input (e.g. heater/cooler or VAV flow setting) and the measured output (e.g. air temperature, humidity) in HVAC zones can be very well approximated by a First-Order Plus Dead Time (FOPDT) model:

$$G(s) = \frac{Ke^{-\theta s}}{1+\tau s} \tag{1}$$

Where the static gain K, time constant τ and time delay θ are generally time-varying and are dependent upon several factors, including the longer-term operating conditions of the system. The process is also subject to numerous disturbances which can be modeled as both measured and unmeasured input perturbations. In this paper, we consider the case of unmeasured disturbances; the extension to measured disturbances is relatively straightforward. As the model parameters are time-varying (and the delay θ can also be large), adaptive and predictive control schemes are warranted [8][9]. Assuming that the delay θ is sufficient close to an inter multiple of the sampling time T, digitization of the model (1) assuming a Zero-Order Hold (DAC) results in:

$$G(z) = \frac{bz^{-d}}{1-az^{-1}} \tag{2}$$

With $a = e^{-T/\tau}$, $b = K(1-a)$ and $d = 1 + (\theta / T_s)$. If the delay contains a fractional component, then an extra numerator parameter in (2) is required; however, as has been done in previous work it is assumed that rounding to the nearest integer value gives a 'good enough' approximation of the HVAC dynamics for control purposes. Under the assumption that the process may experience additive input disturbances, the addition of an integrator into the noise sequence of (2) gives a GPC-style CARIMA model of the process:

$$y(t)(1-az^{-1}) = bz^{-d}u(t) + \frac{e(t)}{\Delta} \tag{3}$$

Where the difference operator $\Delta = 1-z^{-1}$. The inclusion of this disturbance model generally results in controllers which have in-build integral action [7][10]. The next Section will show how optimal predictions of the future process behavior in the presence of unmeasured disturbances can be obtained.

2.2. Optimal Predictions

Assuming first that the a, b and d parameters for the discrete model (2) are known, optimal predictions for the process (3) can be obtained by multiplying out the difference operator Δ, see for example [7]. However this results in more complex state-space description of the process; to keep things as simple as possible, suppose that we define the input disturbance at time t as the signal $v(t)$, giving an ARMA model equivalent of (3):

$$y(t) = ay(t-1) + b\left[u(t-d) + v(t)\right] \tag{4}$$

Once the parameters have been identified, if we define the prediction error at time t as $\varepsilon(t)$:

$$\varepsilon(t) = y(t) - \hat{y}(t) \tag{5}$$

Which is the difference between the actual and predicted output (assuming first that $v(t)$ is zero), i.e.:

$$\hat{y}(t) = ay(t-1) + bu(t-d) \tag{6}$$

Then we have the best estimate of the unknown disturbance at step t is given by:

$$\hat{v}(t) = \frac{\varepsilon(t)}{b} \tag{7}$$

As we have that $E\{e(t+k)\} = 0\ \forall k > 0$ in the model (3), we also have that $E\{v(t+k)\} = v(t)\ \forall k > 0$ in the model (4) - i.e. the disturbance remains constant over the prediction horizon. As such the optimal k-step ahead predictions can be recursively obtained as:

$$\begin{aligned}\hat{y}(t+k \mid t) &= a\hat{y}(t-1+k \mid t) \\ &\quad + b\left[u_s(t-d+k) + \hat{v}(t)\right]\end{aligned} \tag{8}$$

It is straightforward to verify that the predictions produced by (8) are identical to those produced by the CARIMA model (3). Equations (5)-(8) can easily be programmed onto a microcontroller, and require trivial CPU time and memory overheads which are both directly proportional to d for their computation.

3. Adaptive Optimal Control Law

The control law to be developed seeks to minimize the following constrained infinite-horizon cost function for the process model (4):

$$J = E\left\{\sum_{i=0}^{\infty}\left[y(t+d+i\mid t)-r(t)\right]^2 + \lambda\sum_{j=0}^{\infty}\left[u(t+j)-u_\infty\right]^2\right\} \qquad (9)$$

$$\text{Subject to:}$$

$$\forall k \geq 0; \quad u^- \leq u(t+k) \leq u^+$$

Where $E\{.\}$ denotes mathematical expectation, $y(k|t)$ is an optimal k-step ahead prediction of the process output at time t, d • 1 is the integer part of the process time delay, λ is a regularization parameter used to penalize deviations of the control signal from u_∞, which is the required steady-state control signal to maintain the output y_∞ at the current reference (setpoint) signal $r(t)$. As the assumed model of (1) is open-loop stable, u_∞ will always exist. The magnitude of the input (control signal) u is bounded to lie between the hard constraints $u^- < u^+$. In (9), it is implicitly assumed that the predicted future output $y(k|t)$ is constrained to coincide with the reference as $k \to \infty$; this may lead to infeasibility given the input constraints, and is therefore not explicitly included as a hard constraint. Note that although most MPC strategies apply a penalty to the current and future control *increments* [6][7], in HVAC systems the input $u(k)$ is usually associated with the amount of input energy delivered to the system; thus the objective function (9) has the penalty applied to the energy required to deliver the system back to the steady-state $r(t)$. Before considering the constrained version of (9), the next Section develops the unconstrained control law using Linear Quadratic Regulation (LQR) principles.

3.1. Unconstrained Control Law

To put the model (4) in a form more amenable to the application of a predictive control law derived from LQR, it may be equivalently re-written as follows:

$$y(t+d+1) = ay(t+d) + b\left[u_c(t)+v(t+d)\right] \qquad (10)$$

And since a constant input disturbance is assumed, letting $u_s(t) = u_c(t) + v(t)$ gives:

$$y(t+1+d) = ay(t+d) + bu_s(t) \qquad (11)$$

For an infinite-horizon discrete LQR design, we must solve the Discrete Algebraic Ricatti Equation to obtain the steady-state cost-to-go term P [6][11]:

$$P = A^T PA - \left(A^T PB\right)\left(R + B^T PB\right)^{-1} B^T PA + Q \qquad (12)$$

Setting $Q = 1$ and $R = \lambda$ corresponds to the objective function given in (9), and in this case the expression for P (which will be a scalar given the simplicity of the model) simplifies to:

$$P = 1 + \frac{a^2 P\lambda}{b^2 P + \lambda} \qquad (13)$$

Although it is straightforward to iterate the simplified Ricatti equation of (13) until convergence, starting with $P = Q$, some further algebra leads to a scalar quadratic equation in P:

$$b^2P^2 + \left(\lambda - \lambda a^2 - b^2\right)P - \lambda = 0 \qquad (14)$$

Which can be solved analytically (taking the positive root to ensure $P > 0$) using:

$$P = \frac{-\alpha + \sqrt{\alpha^2 + 4b^2\lambda}}{2b^2} \qquad (15)$$

$$\text{with } \alpha = (\lambda - \lambda a - b^2)$$

Once P has been obtained from (15), computation of the optimal LQR gain K simplifies to [6][11]:

$$K = -\frac{abP}{bP^2 + \lambda} \qquad (16)$$

In the case of regulation to the origin with zero disturbances, the control law would simply become $u_c(t) = Ky(t+d)$. However, as we wish to track the current setpoint $r(t)$, the origin of the LQR problem must be shifted for both input u and output y [11]. Shifting the origin of y requires only subtracting the current reference $r(t)$ from the future output $y(t+d)$. Shifting the origin of the control signal, on the other hand, requires finding the steady-state control value u_∞ that will maintain the output $y(t)$ at $r(t)$, and can be obtained by dividing $r(t)$ by the process static gain (which for the model (11) is easily computed as $b/(1-a)$). Remembering that $u_s(t)$ also contains the input disturbance term, this must also be taken into account through subtraction of (7). After some simple algebra the required steady-state control signal u_∞ can be obtained as:

$$u_\infty = \frac{r(t) - ar(t) - \varepsilon(t)}{b} \tag{17}$$

Thus the resulting control law will drive the output to the reference signal, and will provide a weighted penalty on the sum of squared deviations of the control signal from its required steady-state level. Finally, replacing the output $y(t+d)$ with its prediction obtained by iterating (8) allows the optimal unconstrained predictive control law to be written:

$$u(t) = u_\infty + K\left[\hat{y}(t+d \mid t) - r(t)\right] \tag{18}$$

Note that there is no need to specify minimum and maximum costing or control horizons in this control law; λ is the only adjustable parameter. As large deviations of the process static gain can be expected due to the nature of the process and the assumption of 'plug-and-play' control, then it is suggested that to achieve robust performance λ can be made a time-varying parameter which is updated according to the square of the process static gain:

$$\lambda(t) = \frac{b^2}{(1-a)^2} \tag{19}$$

This relationship has previously been suggested for use in tuning predictive controllers [6], and in practice has been found to give excellent performance regulation in the current context.

3.2. Constrained Control Law

In practice, there will always exist hard constraints on the amplitude of the input control signal due to saturation, and that $u_c(t)$ is required to satisfy: $u^- \le u_c(t) \le u^+$, with u^- and u^+ being non-zero time-invariant upper and lower bounds. Considering the constraints in the objective function (9), although it seems a constrained optimization problem of infinite size must be solved at each time step, recent results (e.g. refer to [12] and the references therein) have shown that the constraints may be relaxed for some upper bound N, under the assumption that the process enters a terminal state upon which the constraints are no longer active. Although this results in a finite dimension optimization problem, the bound N can be large enough to rule out on-line optimization in many cases [12]. In the non-adaptive case, there are several ways to overcome this problem; for example offline multi-parametric QP [7], dynamic programming [13] and partially precomputed active set methods [14] can produce control laws which essentially result in state look-up tables. However, these methods require exponential time (normally $O(3^N)$) to determine the control law and are not suitable for on-line implementation. In other related works, however, it has been shown that the *saturated* LQR control law is optimal in the presence of input constraints in many practical situations [12][15]. Importantly, LQR is optimal for the class of first-order sampled data systems without delays [12]. The saturation function is defined as:

$$sat(u) = \begin{cases} u^+ & \text{if} \quad u \ge u^+, \\ u & \text{if} \quad u^- \le u \le u^+, \\ u^- & \text{if} \quad u \le u^-. \end{cases} \tag{20}$$

Although the model (4) clearly contains a delay, the restructuring by (10) does not contain an input delay, and the predictive control law (18) uses only optimal predictions obtained from (8). Thus the minimization of the expectation (9) reduces to finding an optimal constrained control sequence for a delay-free first-order process, and the result from [12] holds (note that a similar result can be derived using [15]). Therefore the optimal solution to the infinite horizon constrained optimization problem defined by (9) is obtained by applying the saturation function (20) to the control law (18) to give:

$$u_{opt}(t) = sat\left(u_\infty + K\left[\hat{y}(t+d \mid t) - r(t)\right]\right) \tag{21}$$

3.3. Summary of the Control Law

A summary of the steps required for the adaptive predictive controller is given in pseudocode in Fig. 2 below. Note that equations (15)-(21) can easily be programmed onto a microcontroller and require only constant CPU time and memory overheads for their computation. Identification of the process parameters and delay is the topic of the next Section.

```
At every sample index k:

(i)   Obtain an estimate of the process
      parameters a, b and d;
(ii)  Obtain an estimate of v(t) using
      (4),(5) and (6);
(iii) Obtain a d-step ahead prediction of
      the output y(t+d) using (7);
(iv)  Calculate Lambda using (18);
(v)   Compute the optimal cost P and LQR
      gain K using (14) and (15);
(vi)  Obtain and apply the optimal
      control action u using (16) and
      (20).
```

Figure 2. Pseudocode for the control algorithm.

4. Online Model Parameter Identification

A key requirement for success of any adaptive/predictive control schemes is the simultaneous and accurate online estimation of the process parameters and time delay (assuming that this is also unknown or varying) [8][16][17]. Assuming that the integer delay d is known and fixed for the model (2), in an adaptive/predictive scheme the online estimation of parameters b and a can be trivially accomplished by using an algorithm such as Recursive Least Squares (RLS) [8]. When d is unknown and/or time varying and also requires identification, the complexity of the problem is significantly increased. Several approaches to overcome this problem rely on variations of the Extended B Polynomial (EBP) method, see e.g. [4][7][8][16], in which the numerator of the transfer function in (1) is expanded such that coefficients to cover all expected values of delay are included. If $d_{min} \bullet 1$ and d_{max} represent known lower and upper bounds on the integer time delay, defining $r = d_{max} - d_{min}$ gives the estimation model given by (22):

$$\frac{y(t)}{u(t)} = z^{-d_{\min}} \frac{b_0 + b_1 z^{-1} + \ldots + b_r z^{-r}}{1 - az^{-1}} \qquad (22)$$

Although this results in a more complicated estimation model with $r + 2$ parameters, all possible values of integer delays can be estimated. After updating the parameters in (22) using RLS, usually some additional on-line technique is employed to fit the expanded model parameters to the simpler model of (2). The 'Low Frequency Polynomial Fitting' (LFPF) technique has proved to be useful for this purpose in the context of adaptive/predictive control [4][16]. When r is small, e.g. $\bullet$ 10, the EBP method works well, however as the needed range of r increases the efficiency of the EBP method starts to decrease dramatically. Both the computational and memory requirements of the RLS algorithm increase quadratically in the number of estimated parameters [8][17], and additionally the convergence speed of the estimator can be significantly affected by the increased number of numerator parameters. Although simplified schemes such as LMS or approximate RLS (e.g. see [8][18]) may be used to reduce the time and memory requirements of the on-line estimator, this is generally at the expense of even slower convergence. A slower sampling rate can be employed to reduce r, but this is at the expense of closed-loop bandwidth and disturbance rejection properties. An alternative technique for estimating model parameters and time delay was recently suggested by the author in [16], and due to its computational efficiency and accuracy is selected for use in the current work.

4.1. Identification Algorithm:

Consider a recursive identifier for the process (2) based around the normal least squares equations, employing an estimate i for the true integer delay d:

$$\beta(k \mid i) = P(k \mid i)^{-1} H(k \mid i) \qquad (23)$$

Where the notation $(k|i)$ is here used to indicate sample index k and candidate delay value i, $\beta(k|i) = [a(k|i),\ b(k|i)]^T$ is the vector of estimated parameters, $H(k|i) = [h_1(k|i),\ h_2(k|i)]^T$ is the combined I/O vector and $P^{-1}(k|i)$ is a 2x2 symmetric covariance matrix. Defining update rules for the scalar elements comprising the upper triangular of $P(k|i)$ and vector $H(k|i)$ at each time step as follows:

$$P(k \mid i) = \begin{bmatrix} p_1(k \mid i) & p_2(k \mid i) \\ & p_3(k \mid i) \end{bmatrix}$$
$$= \begin{bmatrix} \gamma p_1(k-1 \mid i) + y(k-1)^2 & \gamma p_2(k-1 \mid i) + y(k-1)u(k-i) \\ & \gamma p_3(k-1 \mid i) + u(k-i)^2 \end{bmatrix}$$
$$H(k \mid i) = \begin{bmatrix} h_1(k \mid i) \\ h_2(k \mid i) \end{bmatrix} = \begin{bmatrix} \gamma h_1(k-1 \mid i) + y(k)y(k-1) \\ \gamma h_2(k-1 \mid i) + y(k)u(k-i) \end{bmatrix} \qquad (24)$$

Where $0 < \gamma \bullet 1$ is an exponential forgetting factor. Applying Cramer's rule to solve the matrix inversion in (22) leads to a simple solution involving only basic operations on the scalar quantities $p(k|i)$ and $h(k|i)$ updated according to (23) and (24) at each time step k (note that index $(k|i)$ is omitted for clarity):

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{p_1 p_3 - p_2^2} \cdot \begin{bmatrix} p_3 & -p_2 \\ -p_2 & p_1 \end{bmatrix} \cdot \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} \qquad (25)$$

If the total sum-of-squared outputs $Y(k|i)$ is also updated according to $Y(k|i) = \gamma Y(k\text{-}1|i) + y(k)^2$, then the residual sum-of-squares $S(k|i)$ after the application of (25) can easily be computed according to:

$$S(k \mid i) = Y(k \mid i) - a(k \mid i)h_1(k \mid i) - b(k \mid i)h_2(k \mid i) \qquad (26)$$

Now, consider updating another model in which the estimate of delay i is decreased by one. The update rules for $P(k|i\text{-}1)$, $H(k|i\text{-}1)$ and $Y(k|i\text{-}1)$ in this situation will be simplified by observing that $p_1(k|i\text{-}1) = p_1(k|i)$ and that $Y(k|i\text{-}1) = Y(k|i)$. After fitting for the new delay value, if the residual $S(k|i\text{-}1)$ is again computed according to (26) with parameter estimates from $\beta(k|i\text{-}1)$, then should this residual be smaller than $S(k|i)$, $i\text{-}1$ becomes a better estimate of the true time delay than i and the parameter vector $\beta(k|i\text{-}1)$ thus better candidates for the process

parameters. For such a procedure to be of use with the adaptive/predictive scheme described in Section 3, some further refinements are required.

4.2. Refinements

As discussed previously, for estimation purposes in the presence of sustained disturbances the model of (2) is inappropriate as bias in the parameters will be present. This may be overcome by using the model structure (3), and by replacing y and u with their first differences (i.e. Δy and Δu) in equations (23)-(26). A drawback of this approach is that the differencing operator acts as a high-pass filter, and the high-frequency measurement noise that is present in the signals may contain bias at low-frequencies [17]. This can be overcome by first low-pass filtering the signals to obtain Δy_f and Δu_f to be employed in equations (23)-(26), using the following simple relationship:

$$\begin{aligned} \Delta y_f(t) &= 0.9 \cdot \Delta y_f(t-1) + 0.1 \cdot \Delta y_f(t) \\ \Delta u_f(t) &= 0.9 \cdot \Delta u_f(t-1) + 0.1 \cdot \Delta u_f(t) \end{aligned} \quad (27)$$

Although the filter time constant is an adjustable parameter, a default value of 0.9 gives robust performance assuming that the sampling rate is selected appropriately. A second drawback is that as the controller can be expected to spend much of its time in regulator mode, the input signal to the process may not be sufficiently rich enough to warrant the use of a fixed forgetting factor $\gamma < 1$. In order to prevent covariance windup and retain good disturbance rejection, a variable forgetting factor can be used. To incorporate such a forgetting factor into the algorithm described above, the squared and filtered apriori prediction error for the best model and delay obtained in the previous step is used to set the forgetting factor, using the following simple rules:

$$\begin{aligned} e_p(k) &= 0.9 e_p(k-1) + 0.1(\Delta y(k) - \Delta \hat{y}(k))^2 \\ \gamma(k) &= \max\{1 - e_p(k),\, 0.7\} \end{aligned} \quad (28)$$

Although the filer constant (0.9) and minimum forgetting factor (0.7) are essentially adjustable and user-tunable parameters, simulations using these default values give good performance and adaptability over a wide range of operating conditions. A final factor that is often neglected in real-time implementation of adaptive control schemes is the initialization behavior, and the behavior following transient disturbances in the operation of the embedded platform implementing the controller (e.g. watchdog timeout, brownout). At regular intervals of 10 seconds, the current estimated process parameters are written to EEPROM. During initialization following a reset, these parameters are retrieved from EEPROM, and clearing of the memory array corresponding to previous control values is suppressed following a watchdog/brownout reset by the reset status indicator. A simple bang-bang controller is first employed until the forgetting factor γ is ≥ 0.9 (see e.g. [8]). Once this has been achieved, the main adaptive/predictive scheme is switched on. This approach ensures that short-term interruptions do not cause a loss of control during warm-starts coupled with and effective cold-start sequence.

4.3. Summary of the Identification Algorithm

The reasoning described in this Section form the basis for the identification algorithm, which can be explained in a few simple steps as shown in Fig. 3:

```
At every sample index k:

(i)    Set i := d_max and Best := ∞;
(ii)   Recursively update H(k|i), P(k|i)
       and obtain β(k|i) and S(k|i) using
       (23) to (26);
(iii)  If S(k|i) ≤ Best, set Best := S(k|i)
       and d := i;
(iv)   Set i := i-1 and if i ≥ d_min, go to
       step (ii);
(v)    Return [β(k|d), d] as the estimated
       process parameters and time delay.
```

Figure 3. Pseudocode for the identification algorithm.

After execution at each sample index k, the algorithm returns the model parameters and delay index d that minimises the residual sum-of-squares for I/O data up to and including the current sample. As shown in [16], the update computation-time and memory requirements are both $O(r)$, where r represents the range of unknown delay, which is a vast improvement upon a standard RLS technique. In the next Section, initial results are present for the combined adaptive/predictive controller.

5. HIL-based Experimental Analysis

In order to test the proposed adaptive/predictive controller, experiments were carried out using a representative hardware-in-the-loop (HIL) test facility that has previously been developed for testing real-time control implementations [19]. Real-time hardware-in-the-loop (HIL) simulation is currently considered as an essential tool for development and testing of complex real-time and embedded control systems. Compared to software-only simulation, HIL simulation increases the realism of the simulation and provides access to hardware features of the controller platform that may help to uncover potential implementation problems that may not be evident with software-only emulation.

Deleterious effects such as oscillator drift and pipeline/cache issues (which may affect sampling period and distributions of jitter and latency) and their impact on the control system can be evaluated more realistically on the target platform.

5.1. Experimental Setup

An implementation of the described adaptive/predictive control algorithm was coded on a 60 MHz ARM7-TDMI microcontroller. This microcontroller has a 32-bit ARM7TDMI-S core, and is a typical hardware platform for the implementation of modern embedded real-time systems. The device can be operated with a CPU clock speed of up to 72 MHz, has 512 Kb of on-chip flash, 98 Kb of on-chip RAM, and a rich set of I/O peripherals. The latter includes multiple CAN Controllers and UARTs, USB and Ethernet support, Multiple Timers, PWM, SPI and Analog/Digital I/O. The RTE-SIM environment (which was originally developed for pedagogic purposes but has also proven to be extremely useful for research and development) was employed to operate with Simulink and run a real-time model of the process (1), and provide the control signal interface to the embedded target board. Application software was developed using the Keil software development tools, and the application tasks were executed using an Earliest Deadline First (EDF) real-time scheduling policy [20]. After commissioning the control software, extensive simulations were been performed. In this paper, two specific scenarios are considered to elucidate the performance of the controller. In the first scenario, the following process model was implemented:

$$G(s) = \frac{10e^{-3s}}{1+25s} \tag{29}$$

In this scenario, the controller was required to follow square-wave setpoint from a fully cold start (i.e. no process information was given). A step input disturbance (of unit magnitude) enters the process at t = 80 seconds and exits the process at t = 130 seconds. A white noise sequence of variance 0.01 was injected into the process output to emulate measurement noise. In the second scenario, the same disturbance and noise level were employed, with the following used as an initial process model:

$$G(s) = \frac{8e^{-4s}}{1+15s} \tag{30}$$

Again following a cold start, after t = 100 seconds a sudden and severe discontinuous change of operating conditions was simulated by step-changing to use the following model in the simulation:

$$G(s) = \frac{3e^{-8s}}{1+30s} \tag{31}$$

A sampling time of t_s = 500 milliseconds was employed by the controller, with the values of d_{min} and d_{max} set to 2 and 202 respectively. Hard control limits of +/- 20 were employed. The default values as suggested previously in the paper were employed for filter constants etc; if these suggested values are employed in practice, then the only configurable parameter is the sampling rate, of which many guidelines for selection exist (e.g. [8][9][21]). Note that although the employed delays and time constants are somewhat smaller than may be expected in certain real situations, and a slower sampling time than used for these experiments can often be used in practice. For a good choice of sampling rate, then it is not expected to use a range of delay values more than about 200. An on-chip timer with resolution of 0.1μs was employed to measure the Worst-Case Execution Time (WCET) of the control task (including identification/adaptation and predictive steps) during the course of the experiments.

5.2. Experimental Results

The results obtained from scenario one are as shown in Fig. 4. After power on, it can be observed that the algorithm initially outputs a step (as pre the relay initialization), and the main controller is switched on-line after approximately 7 seconds. As can be seen in the figure, both the tracking behavior and disturbance rejection properties of the controller are good, and no steady-state offset exists when disturbances are present.

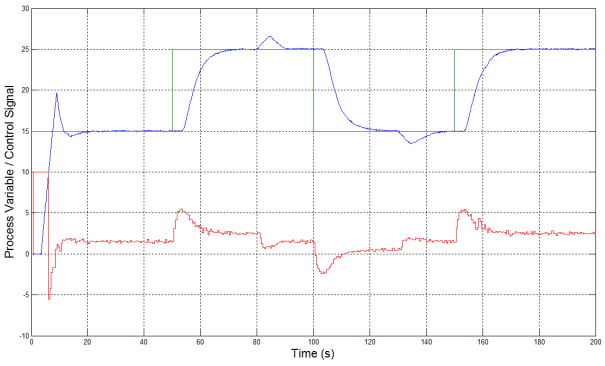

Figure 4. HIL Simulation results for scenario one: see text for details.

The results obtained from scenario two are as shown in Fig. 5. After power on, the main controller is switched on-line after approximately 8 seconds and the process output quickly driven to the setpoint. Again the tracking behavior and disturbance rejection properties of the controller are good. When the process model is step-changed at t = 100 seconds, there is a short transient period (principally due to the increase in time delay)

until the algorithm starts to re-converge. At t = 110 seconds, the step disturbance exist the system. Despite this (worst-case) series of circumstances, the process output is smoothly brought back to the setpoint by t = 150 seconds, and the final step change in reference is followed as expected.

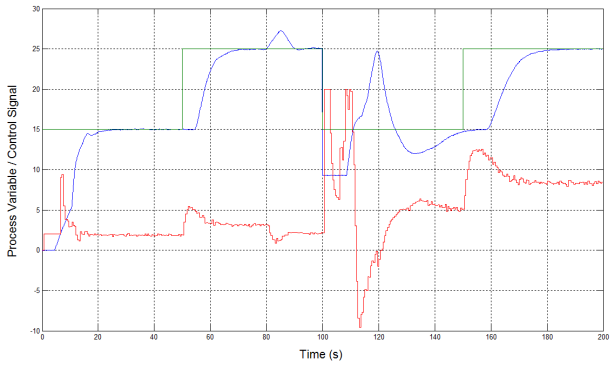

Figure 5. HIL Simulation results for scenario two: see text for details.

The measured WCET for the control task was found to be just 22.01 milliseconds, giving a total CPU utilization of less than 4.5%. Since the CPU may operate in power-down mode for around 95% of its lifetime, this could significantly prolong the lifetime of a battery-powered device. From the analysis of algorithm complexity, it is not surprising that the majority of this computation time was required for the identification step; however, if a regular RLS requiring over 200 parameters were employed, then the identification step alone would take in excess of the 500 millisecond sample time on this computing platform, and parameter convergence would take significant longer [16]. Reduction of d_{max} to 102 followed by 22 gave measured WCET values of 11.19 and 2.464 milliseconds respectively. This illustrates the efficiency and also the effectiveness and scalability of the proposed algorithm.

6. Conclusions and Future Work

This paper has considered the design of a prototype 'plug and play' control system for HVAC applications in Smart homes and offices. Initial HIL simulations have shown the proposed algorithm has good adaptive and predictive capabilities combined with very low overheads, making it suitable for use even with low-cost microcontrollers. Future work will carry out more extensive testing of the prototype controller and apply it to real HVAC applications.

References

[1] J. Široky, F. Oldewurtel, Jirí Cigler & S. Prívara. Experimental analysis of model predictive control for an energy efficient building heating system. *Applied Energy*, Vol. 88, pp. 3079-3087, 2011.

[2] H. Karlsson & C.-E. Hagentoft. Application of model based predictive control for water-based floor heating in low energy residential buildings. *Building & Environment*, Vol. 46, pp. 556-569, 2011.

[3] G. Huang & S. Wang. Investigation on the Application of Robust Model Predictive Control on Air-Conditioning Systems. In: *Proc. of the 7th Asian Control Conference*, Hong Kong, China, pp. 1302-1307, 2009.

[4] J. Bai, S. Wang & X. Zhang. Development of an adaptive Smith predictor-based self-tuning PI controller for an HVAC system in a test room. *Energy and Buildings*, Vol. 40, pp. 2244–2252, 2008.

[5] K.V. Ling & A.L. Dexter. Constrained Predictive Control of Multi-Zone Air Conditioning Systems. In: *Proceedings of Control 94*, pp. 895-900, 1994.

[6] E.F. Camacho & C. Bordons. *Model Predictive Control: 2nd Edition*. Springer-Verlag, 2003.

[7] M. Morari & J. H. Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, Vol. 23, No. 4/5, pp. 667-682, 1999.

[8] K.J. Astrom & B. Wittenmark. *Adaptive Control: 2nd Edition*. Addison-Wesley Publishing, 1995.

[9] E. Ikonen & K. Najim. *Advanced Process Identification & Control*. CRC Press, 2001.

[10] P.S. Tuffs & D.W. Clarke. Self-tuning control of offset: a unified approach. *IEE Proceedings*, Vol. 132, No. 3, pp. 100-110, 1985.

[11] S.L. Lewis & V.L. Syrmos. *Optimal Control*. John Wiley & Sons, pp. 229-236, 1995.

[12] O. Marjanovik, B. Lennox, P. Goulding & D. Sandoz. Minimising conservatism in infinite-horizon LQR control. *Systems & Control Letters*, Vol. 46, pp. 271-279, 2002.

[13] J.B. Mare & J.A. De Dona. Solution of the input-constrained LQR problem using dynamic programming. *Systems & Control Letters*, Vol. 56, pp. 342-348, 2006.

[14] G. Pannocchi, N. Laachi & J.B. Rawlings. A Fast, Easily Tuned SISO Model Predictive Controller. In: Proceedings of DYCOPS 2004, Boston, USA, 2004.

[15] J.B. Mare & J.A. De Dona. Elucidation of the state-space regions wherein model predictive control and anti-windup strategies achieve identical control policies. In: *Proceedings of the American Control Conference*, Chicago, USA, pp. 1924-1928, 2000.

[16] M. Short. Fast online identification of low-order time-delayed industrial processes. *Electronics Letters*, Vol. 48, No. 3, pp. 152-153.

[17] L. Ljung. *System Identification: Theory for the User (2nd Edition)*. Prentice-Hall, USA, 1999.

[18] Farsi, M., Karam, K.Z. and Warwick, K. Simplified Recursive Identifier For ARMA Processes. *Electronics Letters*, Vol. 20, No. 22, pp. 913-915, 1984.

[19] M. Short & C.S. Cox. RTE-SIM: A simple, low cost and flexible environment to support the teaching of real-time and embedded control. *International Journal of Electrical Engineering Education*, Vol. 48, No. 4, pp. 339-358.

[20] Stankovic, J.A., Spuri, M., Ramamritham, K. and Buttazzo, G.C. *Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms*. Kluwer Academic Publishing, 1998.

[21] S. Bennett. *Real-Time Computer Control: An Introduction*. Pearson Education Limited, 1988.