Intermittent Predictive Control of an Inverted Pendulum

Peter J Gawthrop [a,1] Liuping Wang [b]

[a]*Centre for Systems and Control and Department of Mechanical Engineering, University of Glasgow, GLASGOW. G12 8QQ Scotland.*
P.Gawthrop@eng.gla.ac.uk
FAX: +44 141 330 4343

[b]*Discipline of Electrical Energy and Control Systems, School of Electrical and Computer Engineering, RMIT University, Melbourne, Victoria 3000, Australia*

Abstract

Intermittent predictive pole-placement control is successfully applied to the constrained-state control of a prestabilised experimental inverted pendulum.

Key words: Inverted pendulum; control systems; real-time control; model-based predictive control; basis functions.

[1] Corresponding author

1 Introduction

Predictive control is a popular and well-established control method in the process industries– see the tutorial of Rawlings (2000) and the book of Maciejowski (2002) for overviews. In particular, it has the advantage that input, state and output constraints are explicitly incorporated. Most of these methods are found in a *discrete-time* setting; however, a *continuous-time* approach has the advantage that it can make direct use of (possibly nonlinear) *physical system models*. Unlike process systems, it is often straightforward to obtain physical models of *mechanical systems* and thus continuous-time methods are particularly appropriate to this application area. However, unlike applications in the process industry which typically have long time constants and correspondingly long sample intervals of a few seconds, mechanical systems typically have fast dynamics and sample intervals less than 10ms. Hence the optimisation involved in continuous-time predictive control of mechanical systems gives rise to two interrelated implementation issues: making time for the optimisation without sacrificing controller performance and making the optimisation algorithm fast and reliable. In this paper, four approaches to achieve practical predictive control of mechanical systems are combined: the use of *basis-functions* to reduce the number of optimisation parameters, *intermittent* control to allow time for optimisation, a *fast* quadratic programming (QP) algorithm and a *hierarchical* approach to control system design.

The *predictive pole-placement* (PPP) algorithm of Gawthrop and Ronco (2002) is a continuous-time basis-function based approach. As discussed elsewhere, the number of basis functions is the order of the controlled system leading to a correspondingly small number of optimisation parameters. In Section 2 of this paper it is explained how this algorithm can be used within an *intermittent* formulation – whereby a continuous-time *open-loop* control signal is readjusted intermittently (with interval Δ_{ol}) (Ronco, Arsan, and Gawthrop, 1999) – which lies between the extremes of *discrete-time* and *continuous-time* control approaches. The combination of intermittent and predictive pole placement is called *intermittent predictive pole-placement* (IPPP) control. Section 3 describes a practical way of solving QP. Practical experience teaches that a hierarchical approach to control is often more effective than using a single loop approach. For this reason, Section 2.2 discusses how the predictive controller can be used as the upper layer of an hierarchical control systems using fast, simple control loops at lower levels.

The aim of this paper is to verify that this approach can be used for the control of mechanical systems by the experimental state-constrained control of the laboratory inverted pendulum described in Section 4. The results of this study are presented and discussed in Section 4.5. Theoretical stability results are available (Gawthrop, Chen, and Wang, 2005) but not discussed further here.

There are three research areas related to this paper which are now briefly discussed:

physiological control systems, the command governor approach and control of limited bandwidth channels. The intermittent approach has a long history within the physiological literature dating back to at least 1947. In particular, Craik (1947) suggests that: “The human operator behaves basically as an intermittent correction servo.” and further that “The intermittent corrections consist of ‘ballistic’ movements.” Interpreting “ballistic” as “open-loop” this description of the “human operator” also applies to the intermittent algorithm of Section 2. The *command governor* (CG) approach of Bemporad (1998); Casavola, Mosca, and Papini (2004) is a closely-related approach to handling input and state constraints. Briefly, the CG approach has two parts: the design of a “primal” controller which provides high-performance control whilst ignoring constraints and an optimisation-based CG design which chooses a piecewise-constant setpoint (parametrised by the scalar parameter $\beta(t)$), to the primal controller such as to avoid constraints whilst remaining close to the original setpoint. $\beta(t)$ is recomputed at each sample interval. This is similar to the IPPP approach of this paper insofar as it is the setpoint to an inner loop that is involved in a moving-horizon optimisation to avoid input and state constraints; but it differs in a number of respects: the IPPP uses a set of basis function in place of the constant, a *vector* parameter $U(t)$ is computed, and the computed setpoint is applied over a time interval Δ_{ol} longer than the basic sample interval Δ. The effect of the QP processing bottleneck has a similar effect to that of communicating between the controller and plant using a limited-bandwidth communication channel. It is known (for example as discussed by Nair and Evans (2003)) that the latter imposes fundamental restrictions on the stabilisation of the closed-loop system. It appears that these ideas would also apply to IPPP.

Section 4 describes the experimental equipment, computer hardware and software, system modelling, controller design and some experiments. Section 5 concludes the paper.

2 Intermittent PPP Control (IPPP)

This section briefly summarises the basic theoretical ideas of PPP followed by more details on the practical implementation as IPPP.

2.1 Outline of PPP

Following Gawthrop and Ronco (2002), the linear systems considered in this paper are described by:

$$\begin{cases} \frac{d}{dt}x(t) & = Ax(t) + Bu(t) \\ y(t) & = Cx(t) \end{cases} \tag{1}$$

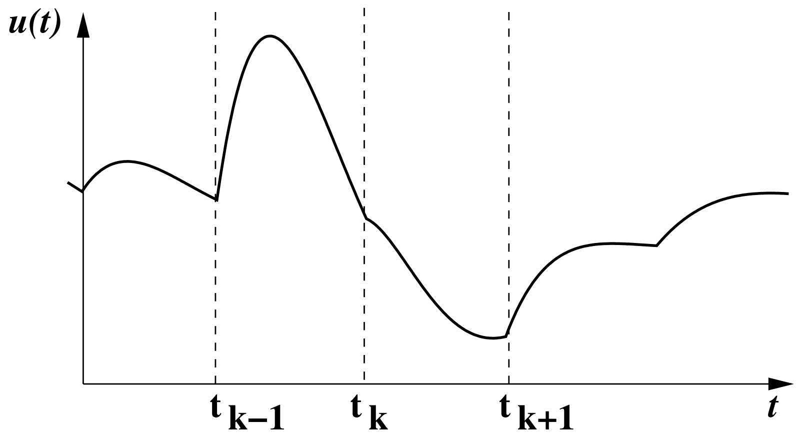


Fig. 1. Intermittent control

and the moving-horizon open-loop controller is given by:

$$u^{\star}(t,\tau) = U^{\star}(\tau)U(t) \tag{2}$$

where t is “real time” as in (1), τ is the moving-horizon time variable, $U^{\star}(\tau)$ is a n_U row vector of functions of τ. The components of $U^{\star}(\tau)$ can be regarded as a set of *basis functions* for the control signal $u^{\star}(t,\tau)$ and the components of $U(t)$ the corresponding weights the values of which are obtained by QP. In the continuous-time formulation, the control signal is obtained as: $u(t) = u^{\star}(t,0)$. In this paper, $U^{\star}(\tau)$ is chosen as the free response of an LTI system with state transition matrix A_u and initial condition $\tilde{U}_0^{\star}$:

$$\tilde{U}^{\star}(\tau) = e^{A_u\tau}\tilde{U}_0^{\star} \tag{3}$$

The use of exponential weighting functions had been previously suggested by a number of authors (Cannon and Kouvaritakis, 2000; Wang, 2001). However, those papers *approximate* the optimisation solution using a sum of such functions whereas in this paper precisely n_x functions are used in such a way as to give an *exact* solution to the unconstrained problem. In this paper, the use of a small number of basis functions (and corresponding optimisation variables), speeds up real time application. Moreover, following (Gawthrop and Ronco, 2002), the basis functions are explicitly linked to closed-loop system poles.

Given a time-varying setpoint $w(t)$, the corresponding moving horizon setpoint $w^{\star}(t,\tau)$ is assumed to be constant and of the form $w^{\star}(t,\tau) = w(t)$; more complicated versions appear in (Gawthrop and Ronco, 2002). The vector $U(t)$ is to be chosen to minimise (at a given time t) the quadratic cost function

$$\begin{aligned}
&J(U(t), x(t), w(t)) \\
&= \frac{1}{2}\int_0^{\tau_2} (y^\star(t,\tau) - w(t))^T Q (y^\star(t,\tau) - w(t)) d\tau \\
&+ \frac{1}{2}\int_0^{\tau_2} u^\star(t,\tau)^T R u^\star(t,\tau) d\tau \\
&+ (x^\star(t,\tau_2) - x_w)^T P (x^\star(t,\tau_2) - x_w)
\end{aligned} \tag{4}$$

subject to constraints on the system input and state which can be mapped on to input constraints of the form (Gawthrop et al., 2005):

$$\Gamma U(t) \leq \gamma \tag{5}$$

where $Q \in \Re^{n_y \times n_y}$, $R \in \Re$, $P \in \Re^{n_x \times n_x}$ are positive definite matrices weighting the system output, input and terminal state respectively. $x_w \in \Re^{n_x}$ is the state corresponding to the the steady-state solution.

The following aspects of PPP are emphasised:

(1) Following Rawlings and Muske (1993), the terminal weighing matrix P is chosen to correspond to the infinite horizon steady-state solution of the appropriate Ricatti equation and the corresponding closed-loop eigenvalues are computed.
(2) As discussed by Gawthrop et al. (2005), the basis function generation matrix A_u of (3) is chosen to have the *same* eigenvalues as those arising from 1.

These two design choices have the following consequences(Gawthrop et al., 2005; Gawthrop and Ronco, 2002):

(1) The n_x basis functions provide an exact solution to the unconstrained problem
(2) The open and closed loop solutions are identical.

2.2 *Implementation issues*

This subsection introduces the practical issue of optimisation in real time by moving the optimisation horizon in an *intermittent*, as opposed to continuous, fashion. The concept of intermittent control was explicitly introduced by Craik (1947), and has been implicitly used by Cannon and Kouvaritakis (2000) in the predictive control context. More details are given by Ronco et al. (1999). The QP algorithm takes a finite amount of time. In the continuous-time context, it is necessary to take account of this using an *intermittent* feedback approach whereby the open-loop trajectory $u^\star(t,\tau)$ is implemented between intermittent closing of the loop by taking a state-measurement and recomputing the open-loop control. In particular with reference to Figure 1:

$$u(t) = u^\star(t_k, t - t_k) = U^\star(t - t_k)U(t_k) \quad t_k \le t < t_{k+1} \tag{6}$$

were the times t_k are an ordered set of time instants. In particular the interval $\Delta_k = t_{k+1} - t_k \ge 0$ must be large enough to accommodate the optimisation time. In this paper, it is assumed that $\Delta_k = \Delta_{ol}$, a constant. Practically, this is achieved by adding an appropriate wait after the optimisation code [2]. Under certain circumstances Gawthrop and Ronco (2002, Lemma 2), the open and closed-loop trajectories are the same in the continuous-time case. Similarly, in the absence of constraints the open-loop, intermittent and closed-loop trajectories are identical.

In many cases, the system state is not available. In such cases, a *state observer* (Kwakernaak and Sivan, 1972) is used to provide a state estimate $\hat{x}$. In this linear case, such observers are easy to design. However, the performance of the observer is crucial as both optimisation and constraints are applied to the *estimated*, not actual, state.

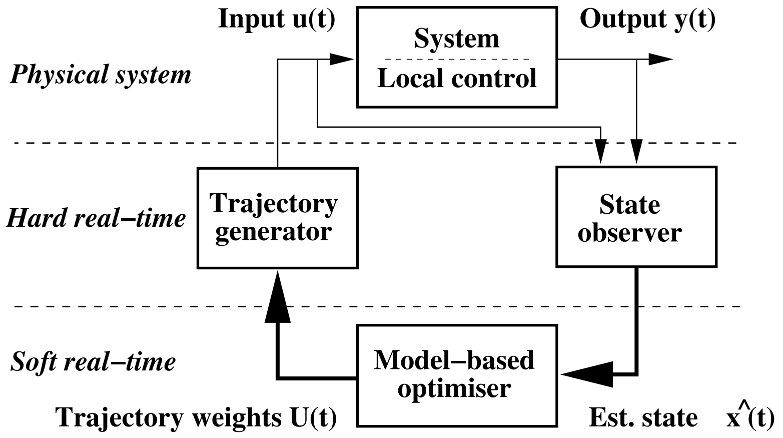


Fig. 2. Architecture

Figure 2 outlines the architecture of the controller and controlled system. There are three layers:

Physical The actual system together local control loops and signal conditioning. Such local control loops are an important part of the design; two examples appear in Section 4.2.

Hard real-time Kernel space code running at a fixed sample interval $\Delta \ll \Delta_{ol}$. At each sample instant

(1) Check if new U received; if so restart trajectory generation:

  (a) set $\tau_i = 0$

[2] The case of time-varying Δ_k is of practical importance as it would allow for varying optimisation times whilst making full use of processor power. The difficulty is that step 2 of the soft real-time part of the algorithm requires the value of Δ_k at the *beginning* of the interval in question.

(b) send current $\hat{x}$

(2) Update the state estimate $\hat{x}$ using the observer equations

Soft real-time User space code running within Octave(Eaton, 2002) at a fixed sample interval Δ_{ol}

(1) Receive the estimated state $\hat{x}(t_k)$ from observer.
(2) Predict $\hat{x}(t_{k+1})$ from $\hat{x}(t_k)$ and U_k.
(3) Using $\hat{x}(t_{k+1})$, perform the QP to generate U_{k+1}.
(4) Wait until time $t = t_{k+1}$
(5) Send U_{k+1} to the trajectory generator.

There are two interfaces:

Physical/hard real-time. Analogue-digital converters, encoders and digital-analogue converters convert the system output y and control signal u between the two domains every Δs.

Hard real-time/soft real-time. Two real vectors: the trajectory weights $U(t)$ (2) and state-estimate $\hat{x}(t)$ each with n_x elements are passed between these layers every Δ_{ol}s. The communication channel is application dependent; in Section 4 the GNU/Linux *pipe* mechanism within a single memory space is used.

The trajectory generation and observer code within the hard real-time layer is fast and deterministic; in contrast, the optimisation code within the soft real-time layer is slow and non-deterministic. Correspondingly, the interface between the physical and hard real-time layers is high bandwidth and that between the hard real-time and soft real-time layers is low bandwidth. This two-time-scale division is the essence of the intermittent approach.

3 Practical Solution of the QP

Equations (4) and (5) define a Quadratic Program (QP). Numerical solutions to this type of problems have been studied extensively in the literature (see for example, Luenberger (1984)). In this practically-oriented paper, it is appropriate to use a practically-orientated solution this is the subject of this section.

There are four types of numerical methods devised for solving this constrained control problem. They are primal methods, dual methods, penalty and barrier methods and Lagrange methods. They, respectively, work in the spaces of dimension $n_U - n_c$, n_c, n_U and $n_U + n_c$ where n_U is the number of optimisation variables and n_c the number of constraints. In the literature of model predictive control, the primal methods have dominated the numerical solutions (see for example, Muske and Rawlings (1993)) until recent years specially tailored interior-point methods applicable to MPC have appeared (see, for example (Rao, Wright, and Rawlings, 1998)). These algorithms solve the constrained control problem by utilising the special structure of the control system.

For simplicity, we will use the notation $J_{Ux} = \frac{\partial^2 J(U(t),x(t),w(t))}{\partial U(t)\partial x(t)}$ (etc) and define

$$F = J_{Ux}x(t) - J_{Uw}w(t) \tag{7}$$

The dual problem to the original quadratic problem Luenberger (1969) is described as follows. Assuming feasibility (i.e. there is an $U(t)$ such that $\Gamma U(t) < \gamma$), the QP of (4) and (5) is equivalent to

$$\max_{\lambda\geq 0}\min_{U(t)}[\frac{1}{2}U(t)^T J_{UU}U(t) + U^T(t)F + \lambda^T(\Gamma U(t) - \gamma)] \tag{8}$$

The minimisation over $U(t)$ is unconstrained and is attained by

$$U(t) = -J_{UU}^{-1}(F + \Gamma^T\lambda) \tag{9}$$

Substituting Equation (9) into (8), the dual problem becomes

$$\max_{\lambda\geq 0}(-\frac{1}{2}\lambda^T H\lambda - \lambda^T K - \frac{1}{2}F^T J_{UU}^{-1}F) \tag{10}$$

where $H = \Gamma J_{UU}^{-1}\Gamma^T$ and $K = \gamma + \Gamma J_{UU}^{-1}F$. Thus the dual is also a quadratic programming problem. Equation (10) is equivalent to

$$\min_{\lambda\geq 0}(\frac{1}{2}\lambda^T H\lambda + \lambda^T K + \frac{1}{2}\gamma^T E^{-1}\gamma) \tag{11}$$

Note that the dual problem may be much easier to solve than the primal problem because the constraints are simpler. A simple algorithm, called Hildreth's Quadratic Programming Procedure (Luenberger, 1969), was proposed for solving this dual problem. In this algorithm, the direction vectors were selected to be equal to the basis vectors $e_i = (0, 0, \ldots, 1, \ldots, 0, 0)$. Then the λ vector can be varied one component at a time. At a given step in the process, having obtained a vector $\lambda \geq 0$, we fix our attention on a single component λ_i. The objective function may be regarded as a quadratic function in this single component. We adjust λ_i to minimise the objective function. If that requires $\lambda_i < 0$, we set $\lambda_i = 0$. In any case, the objective function is decreased. Then we consider the next component λ_{i+1}.

If we consider one complete cycle through the components to be one iteration taking the vector λ^m to λ^{m+1}, the method can be expressed explicitly as

$$\lambda_i^{m+1} = \max(0, w_i^{m+1}) \tag{12}$$

$$w_i^{m+1} = -\frac{1}{h_{ii}}[k_i + \sum_{j=1}^{i-1} h_{ij}\lambda_j^{m+1} + \sum_{j=i+1}^{n} h_{ij}\lambda_j^m] \tag{13}$$

$$\lambda_{act} = -(\Gamma_{act}J_{UU}^{-1}\Gamma_{act}^T)^{-1}(\gamma_{act} + \Gamma_{act}J_{UU}^{-1}F) \tag{14}$$

$$U(t) = -J_{UU}^{-1}(F + \Gamma_{act}^T\lambda_{act}) \tag{15}$$

Equations (12) and (13) form the basis for the iterative solution for λ. Noting that there is no matrix inversion involved in the iterative solutions, this simple procedure could be used to deal with conflicting constraints, in which case an approximation is reached by truncating the iteration to get positive λ. Thereafter, the active λ is used in Equation (15) to obtain the sub-optimal solution for $U(t)$. Our experience is that this approach relaxes one or more constraints numerically when conflict occurs. Of course, in the cases of no conflict of constraints, the exact solution of positive λ can be found using Equation (14), which leads to optimal solution of U(t) using Equation (15).

These considerations give rise to the following algorithm

Algorithm 1 (Practical QP) *Using the current value of the state and setpoint, use (7) to compute the current value of F. Then:*

(1) Find the optimal unconstrained solution using

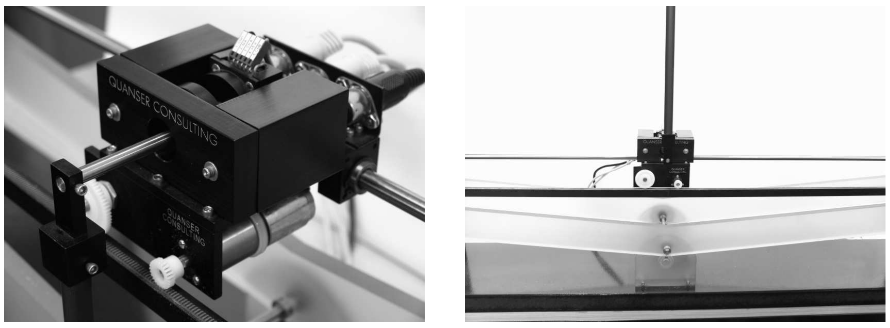
$$U_0(t) = -J_{UU}^{-1}F \tag{16}$$

(2) Check if all constraints are satisfied with the unconstrained solution: i.e. if $\Gamma U_0(t) - \gamma \leq 0$. If so, then $U(t) = U_0(t)$, and the optimal solution is found. Otherwise go to step 3.

(3) Set $H = \Gamma J_{UU}^{-1}\Gamma^T$ and $K = \gamma + \Gamma J_{UU}^{-1}F$ and calculate the active constraints using equations(12) and (13).

(4) find the closed-form solutions with the active constraints and prespecified equality constraints (if any) based on equations (14) and (15) In practice, a standard linear equation solver, not *matrix inversion is used to solve these equations.*

4 The experimental inverted pendulum system



(a) Cart (b) Cart and inverted pendulum

Fig. 3. Experimental equipment

The experimental equipment is based on the Quanser IP-02 “Self-erecting, Linear Motion, Inverted pendulum” experiment. Figure 3(a) shows a cart running on a horizontal track driven by a DC motor and the smaller gear wheel; the linear position y is measured by an encoder attached to the larger gear wheel. The pendulum is pivoted on a frictionless shaft and freely swings in the vertical plane. Figure 3(b) shows the cart and pendulum with the pendulum controlled in the up position.

4.1 Computer hardware and software

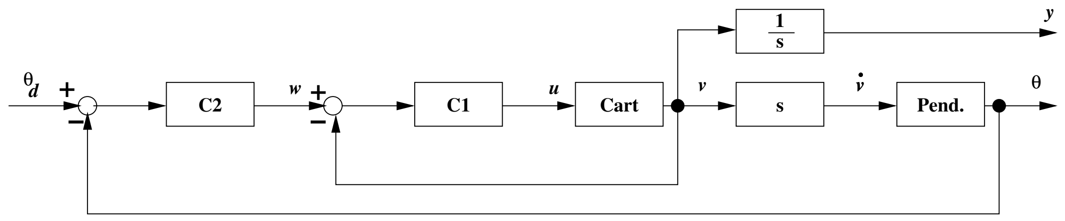
The controller was implemented on a 2.66GHz Pentium P4 based processor on a AICMB800 motherboard with 512MB DRAM. The encoder signals and analogue output were handled by a Quanser MultiQ PCI-based data acquisition card. The software was built upon the Linux 2.4.20 kernel patched to support the Real-time Applications Interface (RTAI) version 24.1.11. This provides a hard real-time platform which in term supports the Linux Control and Measurement Device Interface (COMEDI) version 0.7.66 and the corresponding library (comedilib) version 0.7.20 providing access to the data acquisition card [3] and the Real Time Laboratory RT-Lab (Christini and Culianu, 2003) providing a high-level programming interface together with archiving of experimental data.

With reference to Figure 2, the local control loops, state observer and the trajectory generator were implemented in C and compiled as a kernel module running in hard real time at 500Hz ($\Delta = 2$ ms) communicating with the data acquisition card via RTLab and Comedi and to a user interface module (programmed in C++ using the QT library and running in user space) via shared memory. The observer code was automatically generated as discussed in Gawthrop and McGookin (2004); the nonlinear controller code was automatically generated using MTT (2002). The model-based optimiser of Figure 2 was implemented in Octave(Eaton, 2002) using the QP algorithm of Section 3. It receives the state estimate $\hat{x}$ from the kernel module via the user interface and a GNU/Linux pipe and returns the computed U in a similar way. This soft real-time process was timed using the Octave `usleep` command.

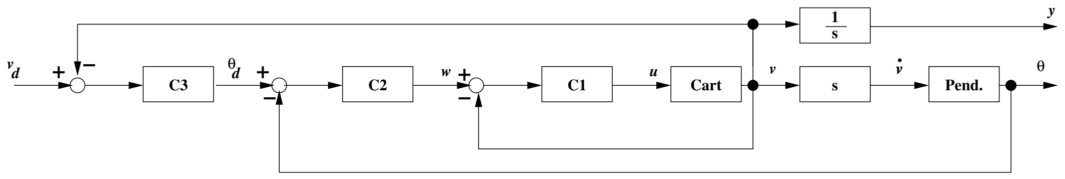
4.2 Hierarchical control

Hierarchical control, in the sense that a practical control system is built up out of a number of nested loops, each designed to achieve a specific local goal, is commonplace in control design Skogestad and Postlethwaite (1996). In particular, inner loops have been used to simplify the design of predictive control Cannon and Kouvaritakis (2000); Kouvaritakis, Rossiter, and Chang (1992).

[3] The driver for this card (multiqpci.c) was written by Linh Vu at UNSW and modified by the first author



(a) System 1: Using basic cascade control



(b) System 2: Using additional cart velocity controller C3

Fig. 4. Cascade Control Structure. C1 controls the inner-loop (cart velocity $v = \dot{y}$), C2 controls the pendulum loop (pendulum angle θ) and C3 controls the cart outer-loop (cart velocity $v = \dot{y}$). The predictive controller (not shown) controls cart position y to a desired setpoint (with a constraint on θ) using θ_d (System 1) or v_d (System 2) as control input.

As discussed elsewhere (Gawthrop and Ballance, 2005), the pendulum can be stabilised in the up position using either of the cascade control configurations of Figure 4. In each case, the block labelled C1 is a high-gain proportional controller giving tight velocity control of the cart position and C2 is a non-linear swing-up-and-hold controller based on the virtual actuator approach of Gawthrop (2004, 2005). For the purposes of this paper, the cascade controllers are used merely to stabilise the controller in the upright position and thus the closed-loop systems corresponding to Figures 4(a) and 4(b) can be represented by linear state-space equations; the closed loop system corresponding to 4(a) is referred to as System 1 and that corresponding to 4(b) is referred to as System 2. The basic cascade controller of Figure 4(a) stabilises the pendulum as a second-order system with prespecified natural frequency ω_1 and damping ratio ζ_1 whilst leaving the cart effectively open loop giving a double integrator effect. The purpose of C3 in Figure 4(b) is to give velocity control of the cart leading to a single integrator in the corresponding closed loop system.

System 1: basic cascade control

With reference to Figure 4(a) it can be shown that the transfer function $\frac{y}{\theta_d} = G_1(s)$ in Figure 4(a)

$$G_1(s) = g_1 \frac{s^2 - \omega_0^2}{s^2(s^2 + 2\zeta\omega_1 + \omega_1^2)} \tag{17}$$
$$= -49.05 \frac{(s+6.67)(s-6.67)}{s^2(s+13.36)^2} \tag{18}$$

In state-space terms:

$$\begin{cases} \dot{x} = A_1 x + B_1 \theta_d \\ Y = C_1 x \end{cases} \tag{19}$$

where $Y = \begin{pmatrix} y & \theta \end{pmatrix}^T$ and

$$A_1 = \begin{pmatrix} 0 & 262.36 & 0 & 0 \\ -0.68 & -26.71 & 0 & 0 \\ 49.05 & 1541.74 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad B_1 = \begin{pmatrix} 0 \\ 0.85 \\ -49.05 \\ 0 \end{pmatrix} \quad C_1^T = \begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{pmatrix} \tag{20}$$

System 2: cascade control with velocity feedback

In Figure 4(b), the outer loop is partially closed using a velocity feedback controller:

$$\theta_d = k_v(v_d - v) \tag{21}$$

where v is the cart velocity as deduced from an observer and v_d is the corresponding desired value. This gives the transfer function $\frac{y}{v_d} = G_2(s)$

$$G_2(s) = g_2 \frac{s^2 - \omega_0^2}{g_2 s(s^2 - \omega_0^2) + s^2(s^2 + 2\zeta\omega_1 + \omega_1^2)} \tag{22}$$
$$= -14.715 \frac{(s+6.67)(s-6.67)}{s(s+4.54)(s+3.73 \pm j11.43)} \tag{23}$$

where $g_2 = k_v g_1$.

$$A_2 = \begin{pmatrix} 0 & 262.36 & 0 & 0 \\ -0.68 & -26.71 & -0.25 & 0 \\ 49.05 & 1541.74 & 14.72 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad B_2 = \begin{pmatrix} 0 \\ 0.25 \\ -14.72 \\ 0 \end{pmatrix} \quad C_2^T = \begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{pmatrix} \tag{24}$$

4.4 IPPP design

A standard observer was designed to be of the form:

$$\dot{\hat{x}} = A_1 \hat{x} + B_1 \theta_d - L(y - C_1 \hat{x}) \tag{25}$$

The observer gain was adjusted to give a reasonable trade off between measurement and state noise as:

$$L^T = \begin{pmatrix} 2.18 & -0.04 & 48.98 & 9.71 \\ 4.64 & 0.05 & 42.64 & 2.18 \end{pmatrix} \tag{26}$$

which gives observer poles at $s = -5.13 \pm j4.03$ and $s = -15.40 \pm j8.24$. The same observer is appropriate to both models when using θ_d as the observer input.

Table 1
Closed-loop system poles

System	R	Poles
1	1	$-2.07 \pm j2.23$, $-13.33 \pm j1.15$
2	0.25	-1.75, -4.44, $-3.94 \pm j11.56$

The IPPP controller was designed for each system choosing output weight $Q = 1$ and control weight R (4) to give similar response in each case. Table 1 shows the weight R and corresponding closed-loop poles in each case. As discussed in Section 2, P (4) and A_u (3) were chosen using the relevant algebraic Riccati equation. In particular, and for each system, A_u was a 4×4 matrix with eigenvalues given by Table 1 and thus there were 4 basis functions in each case.

The pendulum angle θ was constrained such that:

$$|\theta(t)| < 5^o = 0.087\text{rad} \tag{27}$$

This constraint was approximated by imposing upper and lower constraints at the five time points $t_c = 0.1, 0.2, \ldots, 0.5$ giving ten constraint equations on the state $x_1 = \theta$.

Hence there were 4 weights $U(t)$ (2) to be computed on-line (using the QP algorithm of Section 3) minimising the quadratic cost function (4) subject to the state constraints (27).

4.5 Experimental Results

Table 2
Summary of Experiments

y	θ	System	Δ_{ol} sec
5(a)	5(b)	1	0.2
5(c)	5(d)	1	0.5
5(e)	5(f)	1	1.0
6(a)	6(b)	2	0.2
6(c)	6(d)	2	0.5
6(e)	6(f)	2	1.0

As summarised in Table 2, experiments were conducted corresponding to each of the cascade configurations of Figures 4(a) and 4(b) for three values of the open-loop sample interval Δ_{ol}. Following Casavola et al. (2004), a setpoint of increasing amplitude is applied to give on and off constraint behaviour in a single run. The amplitude increased in steps of 0.05m. Figures 5 and 6 each show show six graphs plotted against time for a period of 25s. The left-hand graphs show the system output ym (cart position) obtained from both experiment and simulation; the right-hand graphs shows the corresponding constrained state θrad (pendulum angle) together with the upper and lower limits of $\theta = \pm 5^o = \pm 0.087$rad implied by (27).

The results of the six experiments of Table 2 have the following features:

(1) The RHP zero at $s = 6.67$ leads to the typical reverse response appearing at the setpoint changes.
(2) The control signal first responds to the setpoint change at $t = \Delta_{ol}$.
(3) The intermittent approach performs badly on system 1; however, the approach works well for system 2.
(4) For $t > 10$, the increasing setpoint amplitude leads to the constraints on the state θ being active around setpoint changes; these constraints are approximately enforced.

The effect noted in item 3 can be explained as follows. System 1 (17) contains a double integrator (2 poles at $s = 0$). This makes the *open*-loop trajectory sensitive to disturbances and unmodelled dynamics and the effect gets worse increasing Δ_{ol} from 0.2 (Figure 5(a)) though 0.5 (Figure 5(c)) to 1.0 (Figure 5(e)). On the other hand, the single integrator (pole at $s = 0$) in System 2 (22) has much reduced sensitivity and the corresponding controller performs well for all three sample intervals of Figures 6(a)–6(e).

The effect noted in item 4 arises from two sources: the approximate nature of Hildreth's algorithm (Section 3) with a finite number of iterations and the pointwise nature of the constraints. On the other hand, the approximation allows the computations to proceed at these values of Δ_{ol} even though the QP algorithm is implemented in an interpretive language.

5 Conclusions

An implementation of intermittent linear-quadratic predictive pole-placement control is experimentally shown to achieve good performance when controlling a prestabilised inverted pendulum. Sample rates commensurate with the control of a mechanical system are achieved using a standard PC architecture, real-time Linux and a simple interpretive implementation of the QP algorithm. Clearly, even faster sampling could be achieved using special purpose hardware and a compiled version of the QP algorithm.

As noted in Section 4.2, there would be practical advantages in using a time-varying open-loop interval Δ_k; this would require significant changes to the algorithm of Section 4.2 and would be an interesting topic for further research.

The intermittent open-loop nature leads to poor performance when controlling an unstable system (system 1 containing a double integrator); however it works well on the system containing a single integrator. The paper emphasises the utility of simple inner control loops to enhance the performance of the outer, more sophisticated, control loop. Future work will consider the joint design of inner-loop controller, basis function generator and the predictive control parameters.

Although in practice it will be impossible to completely avoid constraint violation, we believe that the effect can be ameliorated by appropriate choice of basis functions and time points t_c; this will be the subject of future work.

The Introduction mentions interesting links to other research areas: physiological control systems and control over finite-bandwidth channels; these topics will be the subject of future research.

6 Acknowledgements

The experimental work described here was accomplished while the author was a Visiting Professor at the University of New South Wales within the Systems and Control Research Group of the School of Electrical Engineering & Telecommunications. He acknowledges the support and facilities provided by Tim Hesketh and the theoretical and practical help provided by David Clements and Chris Lu. The author is also grateful to Peter Neilson, Megan Neilson and Yalchin Oytam of the University of New South Wales for drawing his attention to the seminal papers of Craik in this context.

Some of the work reported here work was accomplished whilst the first author was a Visiting Professor at RMIT University, Melbourne supported by the RMIT Professorial fund

The real-time software RTLab is part of an Open Source project developed by David Christini and Calin Culianu of Cornell University funded by NSF grant DBI-0096596.

This research was partially funded by the Royal Academy of Engineering under International Travel Grant ITG 04-311.

References

A. Bemporad. Reference governor for constrained nonlinear systems. *IEEE Trans. on Automatic Control*, 43(3):415–419, March 1998.

Mark Cannon and Basil Kouvaritakis. Continuous-time predictive control of constrained nonlinear systems. In Frank Allgower and Alex Zheng, editors, *Nonlinear Model Predictive Control*, volume 26 of *Progress in Systems and Control Theory*, pages 205–215. Birkhauser, Basel, 2000.

A. Casavola, E. Mosca, and M. Papini. Control under constraints: an application of the command governor approach to an inverted pendulum. *Control Systems Technology*, 12(1):193–204, January 2004.

David Christini and Calin Culianu. RTLab. Online WWW Home Page, 2003. URL: http://www.rtlab.org.

Kenneth J Craik. Theory of human operators in control systems: Part 1, the operator as an engineering system. *British Journal of Psychology*, 38:56–61, 1947.

John W. Eaton. *GNU Octave Manual.* Network Theory Limited, 2002. ISBN 0-9541617-2-6.

Peter J Gawthrop. Bond graph based control using virtual actuators. *Proceedings of the Institution of Mechanical Engineers Pt. I: Journal of Systems and Control Engineering*, 218(4):251–268, September 2004. URL `http://dx.doi.org/10.1243/0959651041165864`.

Peter J Gawthrop. Virtual actuators with virtual sensors. *Proceedings of the Institution of Mechanical Engineers Pt. I: Journal of Systems and Control Engineering*, 219(5):371 – 377, August 2005. URL `http://dx.doi.org/10.1243/095965105X33473`.

Peter J Gawthrop, Wen-Hua Chen, and Liuping Wang. Continuous-time LQ predictive pole-placement control. In *Proceedings of the 16th IFAC World Congress*, Prague, 2005.

Peter J Gawthrop and Eric Ronco. Predictive pole-placement control with linear models. *Automatica*, 38(3):421–432, March 2002. URL `http://dx.doi.org/10.1016/S0005-1098(01)00231-X`.

P.J. Gawthrop and D.J. Ballance. Virtual actuator control of mechanical systems. In *Proceedings of the 2005 International Conference On Bond Graph Modeling and Simulation (ICBGM'05)*, Simulation Series, pages 233–238, New Orleans, U.S.A., January 2005. Society for Computer Simulation.

P.J. Gawthrop and E. McGookin. A lego-based control experiment. *IEEE Control Systems Magazine*, 24(5):43–56, October 2004. URL `http://ieeexplore.ieee.org/iel5/37/29513/01337857.pdf`.

B. Kouvaritakis, J. A. Rossiter, and A. O. T. Chang. Stable generalised predictive control: an algorithm with guaranteed stability. *IEE Proceedings-D*, 139(4):349–362, July 1992.

H. Kwakernaak and R. Sivan. *Linear Optimal Control Systems*. Wiley, 1972.

Luenberger. *Linear and Nonlinear Programming*. John Wiley and Sons, New York, 2nd edition, 1984.

D.G. Luenberger. *Optimization by Vector Space Methods*. John Wiley and Sons, New York, 1969.

J.M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, 2002.

MTT. MTT: Model transformation tools. Online WWW Home Page, 2002. URL: http://mtt.sourceforge.net.

K. R. Muske and J. B. Rawlings. Model predictive control with linear models. *Process Systems Engineering*, 39(2):262–287, February 1993.

Girish N. Nair and Robin J. Evans. Exponential stabilisability of finite-dimensional linear systems with limited data rates. *Automatica*, 39(4):585–593, April 2003.

C.V. Rao, S.J. Wright, and J.B. Rawlings. On the application of interior point methods to model predictive control. *Journal of Optimization Theory and Applications*, 99:723–757, 1998.

J. B. Rawlings and K. R. Muske. The stability of constrained receding horizon control. *IEEE Trans. on Automatic Control*, 38(10):1512–1516, 1993.

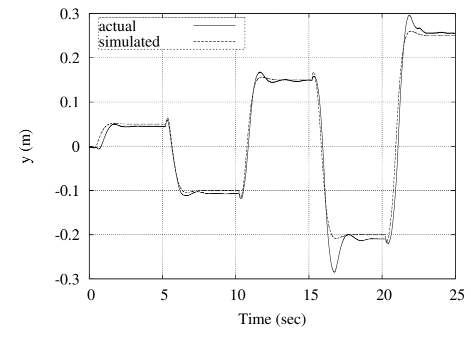
James B. Rawlings. Tutorial overview of model predictive control. *IEEE Control*, 20(3):38–52, June 2000.

E. Ronco, T. Arsan, and P. J. Gawthrop. Open-loop intermittent feedback control: Practical continuous-time GPC. *IEE Proceedings Part D: Control Theory and Applications*, 146(5):426–434, September 1999.

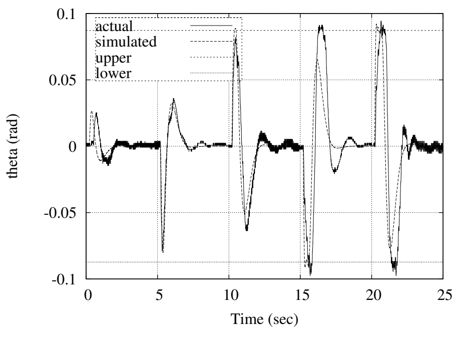
S. Skogestad and I Postlethwaite. *Multivariable Feedback Control Analysis and Design*. Wiley, 1996.

L. Wang. Continuous time model predictive control using orthonormal functions.

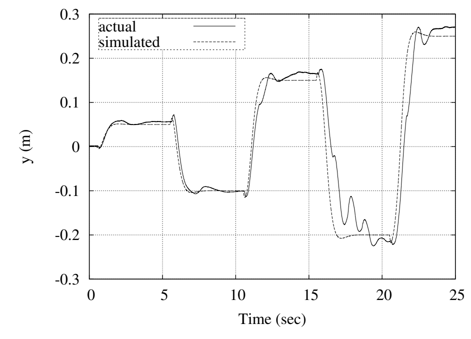
Int. J. Control, 74:1588–1600, 2001.



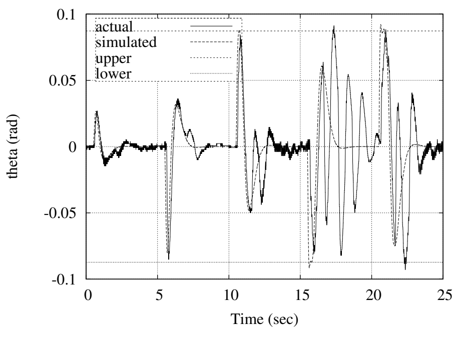
(a) $\Delta_{ol} = 0.2$: Cart position y



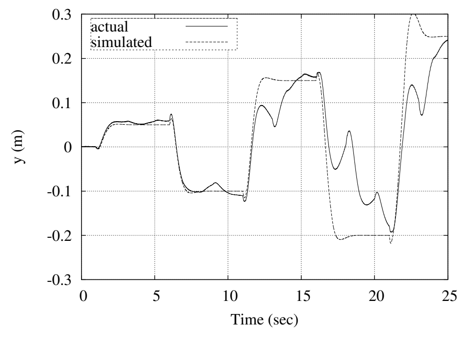
(b) $\Delta_{ol} = 0.2$: Pendulum angle θ



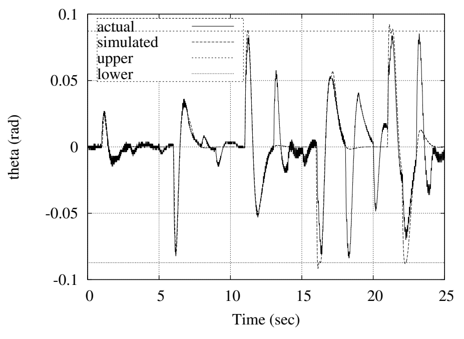
(c) $\Delta_{ol} = 0.5$: Cart position y



(d) $\Delta_{ol} = 0.5$: Pendulum angle θ

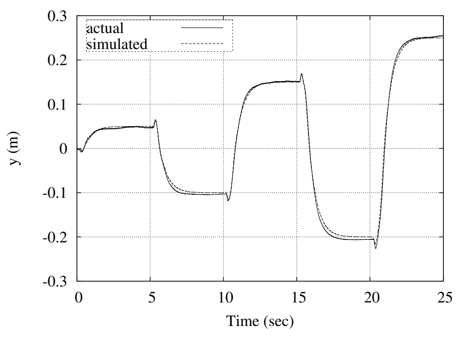


(e) $\Delta_{ol} = 1.0$: Cart position y

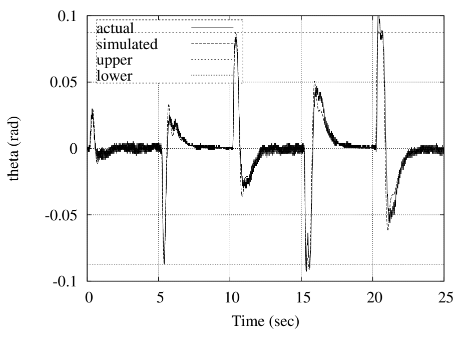


(f) $\Delta_{ol} = 1.0$: Pendulum angle θ

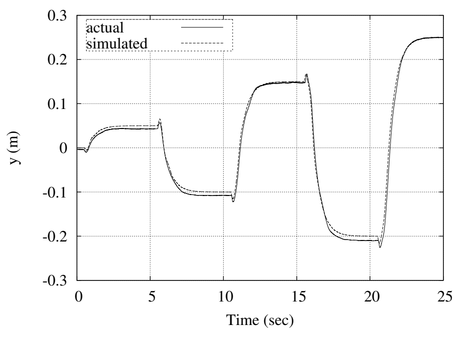
Fig. 5. IPPP Control: System 1



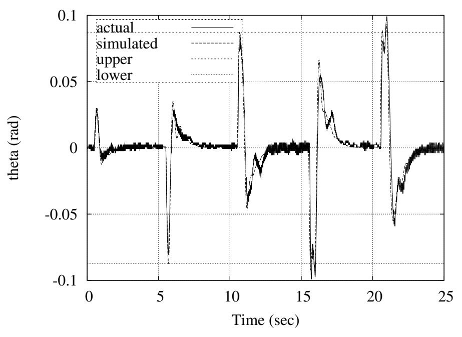
(a) $\Delta_{ol} = 0.2$: Cart position y



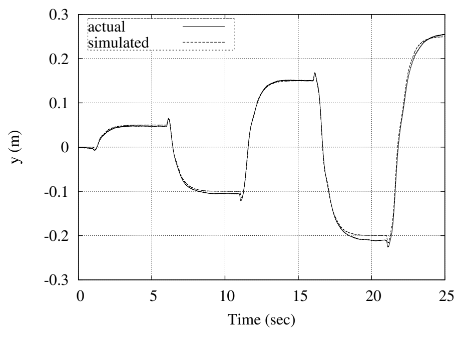
(b) $\Delta_{ol} = 0.2$: Pendulum angle θ



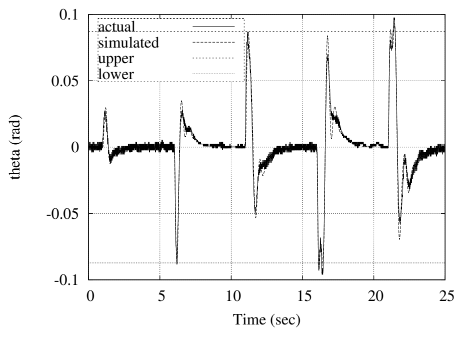
(c) $\Delta_{ol} = 0.5$: Cart position y



(d) $\Delta_{ol} = 0.5$: Pendulum angle θ



(e) $\Delta_{ol} = 1.0$: Cart position y



(f) $\Delta_{ol} = 1.0$: Pendulum angle θ

Fig. 6. IPPP Control: System 2