Task Scheduling for Control Oriented Requirements for Cyber-Physical Systems

Fumin Zhang, Klementyna Szwaykowska, Wayne Wolf, and Vincent Mooney
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA, 30332
Email: {fumin, klimka, wolf, mooney}@gatech.edu

Abstract

The wide applications of cyber-physical systems (CPS) call for effective design strategies that optimize the performance of both computing units and physical plants. We study the task scheduling problem for a class of CPS whose behaviors are regulated by feedback control laws. We co-design the control law and the task scheduling algorithm for predictable performance and power consumption for both the computing and the physical systems. We use a typical example, multiple inverted pendulums controlled by one processor, to illustrate our method.

1. Introduction

Cyber-physical systems range from relatively small systems, such as aircrafts and automobiles, to large systems like the national power grid. The physical characteristics of such systems can be well regulated by controllers by putting computers and algorithms into the loop, but to achieve the full benefit of integrating cyber and physical systems, control, computing, and power consumption can no longer be optimized separately. The dynamics of computers and the dynamics of physical systems interact in ways that require fundamentally new design approaches.

In this paper, we demonstrate our results towards developing a theoretical framework aiming to provide guidelines for computing-control-power co-design. More specifically, considering the power of the entire system is constrained, we address the interactions between control algorithms and the scheduling algorithms to implement multiple control algorithms on one single processor. The key question is that how can we ensure robustness of controller performance in the case of variations in the scheduling software under low power conditions. This question is often encountered in applications where cyber-physical systems are employed.

Development of theoretical results often requires that we simplify real world systems to achieve idealized models that capture the most significant properties. In this paper, we study an idealized example where multiple inverted pendulums are controlled by a single processor. The inverted pendulum model is chosen since the related control problems have been intensively studied and well understood by the control community. In addition, a large class of real world control problems, for example, stabilizing robotic arms and legs, can be simplified to studying the inverted pendulum problems c.f. [2, 11].

We develop both off-line and on-line algorithms for determining the period of multiple control tasks to balance robustness of the system and power consumption under the constraints of schedulability. In particular, a simple proportional feedback controller is used to stabilize each pendulum under sampling effect and delay. Due to differences in the physical parameters, the control gain and the period of the control tasks for each controller is different. The rate monotonic scheduling (RMS) algorithm of Liu and Layland [10] is applied. Hence the periods of control tasks are constrained by the RMS schedulability condition.

Noise enters the pendulum control systems through imperfect sensors and actuators. The sensitivity function is a gain function that associates the fluctuations of the inverted pendulum around its upright position with the strength of noise. The H_∞ norm measures the magnitude of the sensitivity function for noise with all possible frequencies. Hence, if the sensitivity function of a system has a lower H_∞ norm, the system is more robust to noise.

To find the optimal time period for each control task, we define the objective function to be optimized as the sum of the H_∞ norm of the sensitivity functions of all the pendulums. We first develop an off-line algorithm to find the task periods that minimize the objective function. Then we design an online feedback scheduling algorithm that yields sub-optimal solutions but with short computing times. Next, we incorporate power consumption of the system into the objective function and derive algorithms to optimize both system robustness and power consumption. Such algorithms balance the robustness, schedulability, and

power consumption of the cyber physical system.

Some of the early approaches on period selection and adaptive load adjustment are presented in [9] and [14]. There are a number of recent papers addressing the period selection problem for control-scheduling co-design. The popular approach is to formulate the problem as a constrained optimization problem [5]. Different methods are distinguished by the cyber-physical models, the selection of objective function for system performance, the constraints, and the optimization algorithms. In [12], the robustness of the system with respect to parameter uncertainty is optimized and a "branch and bound" algorithm is proposed. However, performance degradation caused by sensor noise is not considered. In [17], the objective function measures error between a planned path and the actual path for a robot manipulator. An online feedback scheduling algorithm is developed to reduce such error. This objective function excluded effects of time delay. The system performance in [15] is based on a quadratic cost function for which an optimal controller is designed. Simulation tools that help to determine system performance with changing time periods under time delay, i.e. "Jitterbug" and "TrueTime", are introduced by the authors of [4]. Reviews of the current state of the field are presented in [18] and [16].

A significant number of results have been obtained in real-time scheduling for dynamic voltage scaling (DVS). Yao et al. [19] propose one of the first DVS-aware scheduling algorithms. Ishihara and Yasuura [7] analyze DVS scheduling with discrete voltages. Azevedo et al. [1] develop heuristic schedulers that take advantage of program profile information. Jejurikar et al. [8] introduce procrastination scheduling, which maximizes idle periods. Some recent results on power management in sensor network applications are presented in [13].

This paper distinguishes from existing published work in that we measure the system performance by its robustness subject to sensor noise, sampling effect, and time delay. In addition, we add a measurement of power consumption to the objective function. We introduce the concept of non-critical system and non-critical operation points which are optimal solutions that are robust to variations in computing time and physical parameters. We show that although computing such a non-critical operation point usually requires complex algorithms, simple heuristic algorithms can be designed to find sub-optimal solutions. These heuristic algorithms can then be implemented as online scheduling algorithms. Shifting between operation points will create transients for the physical system and a transient time needs to be guaranteed for stability.

The paper is organized as follows. Section 2 introduces the problem of using one processor to control N inverted pendulums. Control laws are designed to stabilize the pendulums under sampling effects and time delay. Section 3 presents online and offline task scheduling algorithms that decide the operation points of the cyber-physical system. Section 4 analyzes the transient performance of the system. Section 6 presents simulation results. Conclusions and future work are discussed in Section **??**.

2. Control of inverted pendulums with delay

In this section, we introduce an idealized model of a single processor controlling N inverted pendulums. We give an introduction and overview of the procedure to design a feedback controller that stabilizes each single inverted pendulum.

An N-inverted pendulum system controlled by a single processor is illustrated in Figure 1. We let m be the total mass of a pendulum and plot the mass as a solid ball on top of a weightless strut in Figure 1. Let l be the length of the pendulum and let g be the gravity constant. We define $J = ml^2$ as the moment of inertia. The position of a pendulum is described by θ, that is, the angle that the pendulum makes with its vertical axis. The angular speed of the pendulum is $\dot{\theta}$. Since there is usually friction at the base, we use a symbol γ to represent the friction constant and assume that the friction force applied to the ball works against the direction of motion with strength $\gamma\dot{\theta}$.

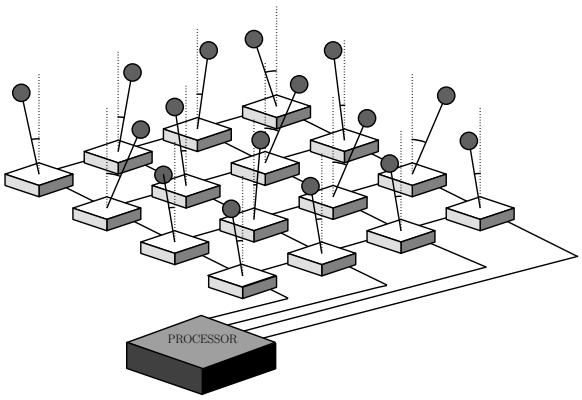


Figure 1. A single processor controlling N pendulums with different physical parameters.

An inverted pendulum needs to be actively controlled in order to stay in the upright state corresponding to $\theta = 0$. Such control can be achieved by installing a motor at its base providing a torque that is equivalent to a force u applied at the center of mass. Stability will be achieved if we design u as a function of the position θ that can be measured by sensors such as a shaft encoder in the motor. Such a function is called a *feedback control law*. Since the physical pa-

rameters l, J and γ of the pendulums are usually different, the control laws are different for different pendulums.

2.1. Modeling of an inverted pendulum

The dynamics of the single inverted pendulum system can be described by a set of differential equations, called *the state-space model*. The differential equations are then linearized to simplify the model. Defining $x(t) = [\theta(t), \dot{\theta}(t)]'$ and

$$A = \begin{bmatrix} 0 & 1 \\ \frac{g}{l} & -\frac{\gamma}{J} \end{bmatrix} \qquad B_c = \begin{bmatrix} 0 \\ \frac{l}{J} \end{bmatrix}$$
$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

we can write the linearized state-space equations for the inverted pendulum as:

$$\dot{x} = Ax(t) + B_c u(t-T) \quad (1)$$
$$y(t) = Cx(t) \quad (2)$$

Since the controller is implemented on a computer which is based on discrete time signals, we measure the state of the system at discrete time intervals, with sampling interval T. The corresponding control effort must typically be computed by the processor in some time $Q < T$. Assuming that the result of each computation is released at the same time as the start of the next computation, there will be a time delay T associated with the control effort u. This explains the use of $u(t-T)$ in the above equation. We assume that u is constant on each time interval $kT \leq t < (k+1)T$, and T is constant and known. The sampling and control delay can be captured more concisely by discretizing the above system equations to get the following:

$$x_{k+1} = e^{AT} x_k + B u_{k-1} \quad (3)$$

where x_k and u_{k-1} are shorthand notations for $x(kT)$ and $u((k-1)T)$, e^{AT} denotes the matrix exponential of AT, and $B = \int_0^T e^{A(T-\sigma)} B_c d\sigma$.

The inverted pendulum model and its variations play an important role in control theory and applications such as control of robotic manipulators or legged robots. In this paper, we will only generalize results obtained for this model to a class of systems called *single input, single output (SISO)* linear systems. For the inverted pendulum, the single input is the control torque u and the single output is the measurement of angle θ.

A typical SISO linear system with computing delay can be modeled using a block diagram, as shown in Figure 2. The diagram is generated by applying the Z-transform to the state space model to convert the state space recursive difference equations into algebraic equations. The variable z here is a complex variable that represents frequencies. To

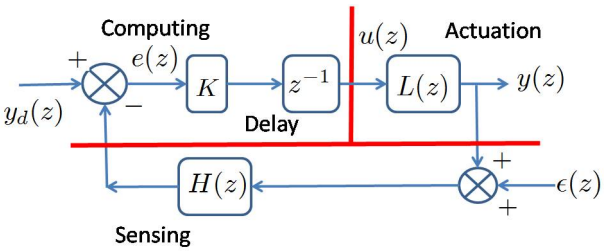


Figure 2. The block diagram of a single input, single output system. The tasks for computing, sensing and actuation are illustrated.

understand the diagram, let us start with the signals. The symbol $y_d(z)$ is the "desired" output value, which the system should track, and $y(z)$ is the measured output. The input $\epsilon(z)$ represents random noise in the measured output signal which is fed back to the controller. Next, the block K represents the implementation of the controller by a processor, the block z^{-1} represents the time delay in computing, the block $L(z)$ represents the physical plant (e.g., motor and pendulum), and the block $H(z)$ represents any possible sensing and noise reduction algorithm. For the inverted pendulum we have

$$L(z) = C\left(Iz - e^{AT}\right)^{-1} B \quad (4)$$

and $H(z) = 1$.

A simple feedback control law to stabilize the inverted pendulum at its upright position is $u_{k-1} = -K\theta_{k-1}$ with K a non-negative constant called the *gain of the controller*. Then the transfer function, which describes the relationship between the input $y_d(z)$ and the output $y(z)$, is

$$G(z) = \frac{KC\left(Iz - e^{AT}\right)^{-1} Bz^{-1}}{1 + KC\left(Iz - e^{AT}\right)^{-1} Bz^{-1}} \quad (5)$$

which results in $y(z) = G(z) y_d(z)$.

2.2. Controller performance

Applying control theoretic methods, we compute the relationship between system stability, gain, and computing delay for an inverted pendulum system. Results pertaining to the pendulum system can be easily generalized to SISO linear systems. Although these results are well known in the control community, we feel it is necessary in this paper to illustrate these results to reveal the connections between controller design, scheduling algorithms, and power analysis.

In classical control, there are several graphical approaches to analyze the stability of SISO linear system. We

perform an analysis based on the root-locus plot to determine how stability of the system depends on gain K and delay T. The basic idea of root-locus plot is to plot the trajectories of the complex singular values of the closed loop system (i.e., complex values that make the magnitude of the transfer function go to infinity) when the gain K changes. The singularities are also called the *poles* of the system. If all the poles are inside the unit circle in the complex plane where z belongs, then the system is stable; if any pole is outside of the unit circle, then the system is unstable [2, 11].

Using the root-locus plot approach, we determine the values of the time delay τ for which the system is stabilizable. We say the inverted-pendulum system is stabilizable only if there exists a value of gain K such that the poles of the system lie inside the unit circle. This can only be true when $T \leq \frac{\gamma l}{Jg} = \frac{\gamma}{mgl}$. This maximum time delay serves as the *hard deadline* for the computing task. In addition, we must have $K \geq \frac{Jg}{l} = mgl$ for the system to be stabilized, for any value of T that is shorter than the hard deadline $\frac{\gamma}{mgl}$.

Next, we derive a deadline for computation delay for a specific gain K. When K is fixed, we solve for the maximum corresponding delay such that the overall feedback system is stable. A plot of this maximum delay versus K is shown in Figure 3. This figure is achieved by solving for values of T such that the significant branches of the root-locus plot cross the unit circle boundary when a specific value of gain K is used.

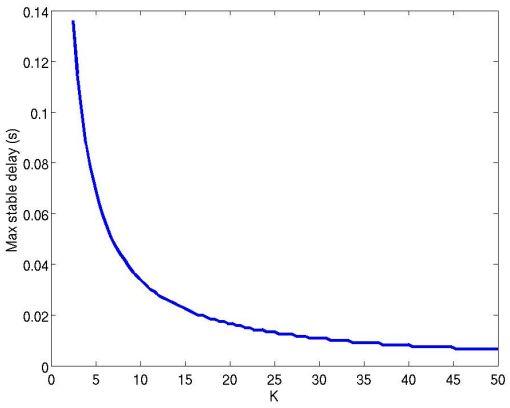


Figure 3. Maximum period T for which the system is stable as a function of gain K. The horizontal axis is for the gain and the vertical axis is for the maximum time delay. The system is always unstable for $K \leq mgl$

From Figure 3, we see that using higher gains allows significantly shorter periods for computing the control laws. This is equivalent to saying that if longer computing delays are expected, lower gain should be used to control the physical system, i.e., the pendulum.

Even if the system is stable, however, delay may result in unacceptable degradation in controlled system performance. We measure system performance by analyzing the sensitivity function $S(z)$ which is the transfer function from disturbances in the measurements (represented by $\epsilon(z)$ in the block diagram Figure 2) to the tracking error $e(z)$. For the inverted pendulum system, the sensitivity function $S(z)$ is:

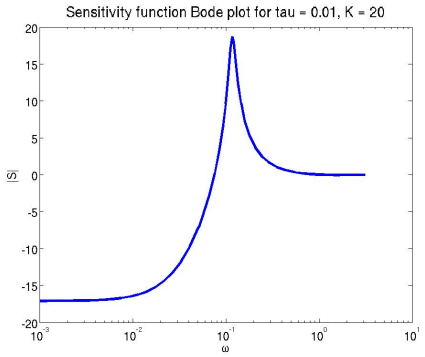
$$S(z) = \frac{1}{1 + KC\left(Iz - e^{AT}\right)^{-1} Bz^{-1}}. \qquad (6)$$

To track the input signal $y_d(z)$ and to reject noise in the measurement, the magnitude of the (complex) sensitivity function $S(z)$ i.e., $|S(z)|$, should be as close to 0 as possible across all frequency spectrum to keep $e(z)$ small for all frequencies. However this goal cannot be realized because decreasing $|S(z)|$ at one frequency range will inevitably increase $|S(z)|$ at another frequency range as illustrated by Figure 4. The best approach is to keep $|S(z)|$ very small at low frequency and the maximum value of $|S(z)|$ not too big. The controlled system will be effective in rejecting low frequency noise and allowing some amount of high frequency noise. This can be achieved for the inverted pendulum system by choosing sufficiently high values of the gain K (see Figure 4). It eventually becomes impossible to choose an appropriate gain K so that the peak value of $S(z)$ is sufficiently small, as suggested by Figure 3.

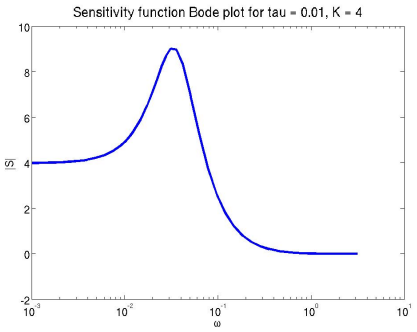
Before we finish this section, we would like to point out that the results generalize to other SISO systems. In fact the relationship between control gain and deadline shown in Figure 3 is typical for any SISO system, and the maximum magnitude of $S(z)$ is the H_∞ norm of the sensitivity function for a SISO system [20]. These performance measures can also be extended to multiple input and multiple output linear systems (MIMO), which will not be discussed in this paper.

3. Task scheduling algorithms

In this section, we develop task scheduling algorithms that schedule the computation of controllers on a single processor to simultaneously control N pendulums. Each controller is implemented by a computing task and there is no data dependency between the tasks. Therefore the rate-monotonic scheduling (RMS) method by Liu and Layland [10] is appropriate. We assume the time spent to compute each controller is fixed. Our goal is to find the optimal periods for all tasks, which we call an *optimal operation point* that guarantees both the schedulability and the controller performance of the N tasks and the N pendulums.



(a) $K = 20$



(b) $K = 4$

Figure 4. Magnitude of sensitivity functions for inverted pendulum system with $T = 0.01$ and gain $K = 20$ (a) and $K = 4$ (b). Note that too high a gain produces an undesirable resonance peak, while too low a gain causes undue sensitivity to low-frequency disturbances.

3.1. Problem setup

We represent each controlling task using a three tuple (P_i, Q_i, T_i) where P_i is the priority of the task, Q_i is the computing time and T_i is the period for task i. RMS claims that the sufficient condition for schedulability of the N tasks is $\sum_i Q_i/T_i \leq a$ where $a < 1$ is known. The priority of the tasks are assigned such that $P_i > P_k$ if $T_i < T_k$.

Consider the control systems design results shown in Figure 3. The computing period T_i should be less than the control deadline $\tau_i(K_i)$, i.e., $T_i \leq \tau_i(K_i)$. This requirement is to guarantee that any pendulum is still stable in the worst case scenario. Letting $T_i = \tau_i$ may guarantee the schedulability, but this sacrifices the physical system performance and may even result in instability under perturbations. Therefore, a critical question is how to determine T_i to ensure **both** schedulability and robustness of the physical systems. We call the collection $(T_1, T_2, ..., T_N)$ an *operation point* of the CPS.

As an example, let us consider the case $N = 2$. We have two independent pendulum systems with different physical parameters, controlled by a single processor. Since we are considering a simple proportional-gain feedback controller, the computation time for both pendulum systems are equal ($Q_1 = Q_2$). Without loss of generality, assume that $task_1$ is the higher-priority task. Suppose now that Q_1 is preempted by an interrupt or an unexpected task running on the same processor or Q_1 becomes longer due to faults in the computing systems. The task set ($task_1$, $task_2$) is no longer feasible under RMS. To maintain feasibility, we can increase the task deadlines T_1 or T_2. Which deadline can be extended depends on the physical performances of the systems being controlled. Using the conclusions from root-locus methods, we know that each pendulum system has a maximum delay $\frac{\gamma_i}{mgl_i}$. If this maximum delay is exceeded, the system becomes unstable regardless of the control effort. If, however, we can extend either T_1 or T_2 without danger of destabilizing either pendulum system, our choice for which deadline to extend should depend on the expected performance of each system under additional delay.

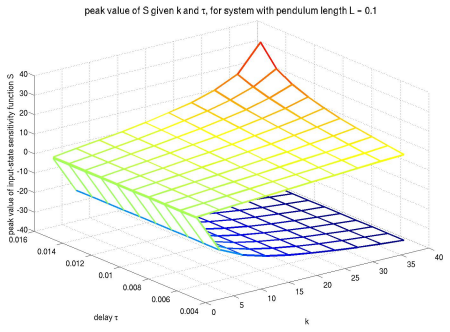
3.2. Optimal operation points

We measure the performance of the inverted-pendulum system by the H_∞ norm of the sensitivity function, represented by $\|S(z)\|_\infty$. Let $z = e^{j\omega T}$, we obtain $S(\omega, T)$, the Fourier transform of the sensitivity function. Then $\|S(z)\|_\infty$ is the maximum magnitude $|S(\omega, T)|$ for all frequencies ω. The H_∞ norm tells us the worst-case noise amplification factor that needs to be minimized. In addition, the value of $|S(\omega, T)|$ when $\omega = 0$, i.e., $|S(0)|$, measures rejection of low frequency noise. We must maintain sufficient low-frequency attenuation that can be represented by smaller $|S(0)|$. Figure 5 shows a plot of $\|S(z)\|_\infty$ as a function of controller gain K and delay T (upper branch), as well as the low frequency sensitivity value $|S(0)|$ as a function of K and T (lower branch). We can observe that lower gain K decreases the H_∞ norm but increases $|S(0)|$.

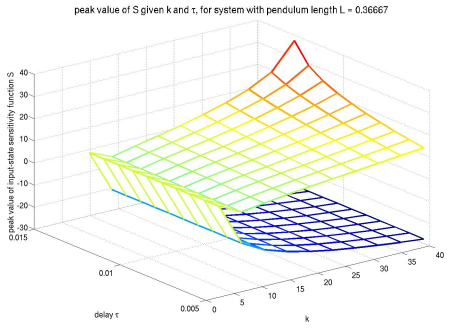
For continuous systems, the value $|S(0)|$ is independent of time delay. Therefore, the control gain K can be selected to achieve a small low frequency sensitivity value that is not significantly affected by sampling time and delay. After the gain is fixed for each pendulum, the overall system performance as a function of time delay $(T_1, T_2, ...T_N)$ for all pendulums can be measured by summing all the H_∞ norms:

$$\mathcal{S}_a(T_1, T_2, ...T_N) = \sum_{i=1}^{N} \|S_i(\omega, T_i)\|_\infty. \qquad (7)$$

Therefore, to determine T_i, we need to solve the following



(a) $l = 0.1$



(b) $l = 0.37$

Figure 5. Peak and low-frequency values of $\|S(z)\|_\infty$ as a function of controller gain K and delay T, for different values of pendulum length l.

minimization problem:

$$\begin{aligned} &\text{Minimize } \mathcal{S}_a(T_1, T_2, ..., T_N) \\ &\text{under the following constraints:} \\ &0 \le \sum_i Q_i/T_i \le a \\ &Q_i \le T_i \le \tau_i \ . \end{aligned} \tag{8}$$

where $a < 1$ is known.

Similarly, we can also define another performance measure to be

$$\mathcal{S}_b(T_1, T_2, ...T_N) = \max_i [\|S_i(\omega, T_i)\|_\infty], \tag{9}$$

and a constrained minimization problem using performance measure $\mathcal{S}_b$:

$$\begin{aligned} &\text{Minimize } \mathcal{S}_b(T_1, T_2, ..., T_N) \\ &\text{under the following constraints:} \\ &0 \le \sum_i Q_i/T_i \le a \\ &Q_i \le T_i \le \tau_i \ . \end{aligned} \tag{10}$$

The following lemma claims that a solution to each of these minimization problem always exists as long as the feasible set defined by

$$\begin{aligned} F = &\{(T_1, T_2, ..., T_N) | 0 \le \sum_i Q_i/T_i \le a \\ &\text{and } Q_i \le T_i \le \tau_i)\}. \end{aligned} \tag{11}$$

is not empty. This is based on the fact that F is a compact set.

Lemma 3.1 *If the feasible set F is not empty, then a solution to each of the minimization problems exists.*

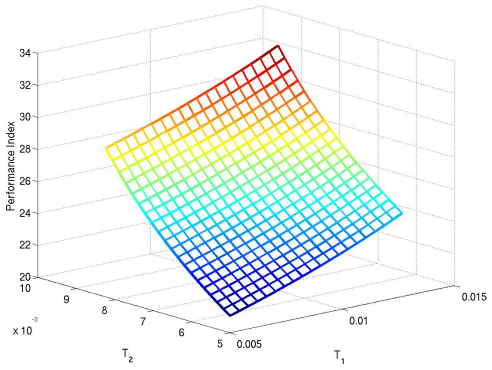


Figure 6. The objective function $\mathcal{S}_a$ of two controlled pendulums. The pendulums have lengths $l_1 = 0.1$ and $l_2 = 0.37$.

For pendulum control, numerical analysis shows that for each i, the $\|S_i(\omega, T_i)\|_\infty$ is a monotone function of T_i. This can be observed from Figure 6 where the objective function for two pendulums are illustrated. Therefore, we have the following result.

Theorem 3.2 *If $\|S_i(\omega, T_i)\|_\infty$ is monotone in T_i for each i, then the optimal operation point lies on the Liu-Layland hyper-surface L defined by*

$$L = \{(T_1, T_2, ..., T_N) | \sum_i Q_i/T_i = a \text{ and } Q_i \le T_i \le \tau_i\}. \tag{12}$$

Proof We prove the claim by contradiction. Suppose we can find an operation point $(T_1^0, T_2^0, ..., T_N^0)$ that belongs to F, but does not lie on L, resulting in a minimum of the cost function $\mathcal{S}_a$ or $\mathcal{S}_b$. Then $\sum_i Q_i/T_i^0 < a$. Without lost of generality, assume that $T_1^0 > T_1 > Q_1$. We can then decrease T_1^0 to T_1 but hold other T_i^0 unchanged such that $Q_1/T_1 + \sum_{i=2}^N Q_i/T_i^0 = a$. Since $\|S_1\|_\infty$ is monotone for T_1, the point $(T_1, T_2^0, ..., T_N^0)$ yields a smaller value for $\mathcal{S}_a$. This results in a contradiction. For $\mathcal{S}_b$, without loss of generality, we can assume that $\|S_1\|_\infty$ is greater than all other $\|S_i\|_\infty$ in the neighborhood of the minimum operation point. Then changing from T_1^0 to T_1 decreases $\|S_1\|_\infty$ and also decreases $\mathcal{S}_b$, a contradiction. ■

The monotonicity of the H_∞ norm in T_i results in very simple algorithms to solve the optimization problems. We can compute the function $\|S_i\|_\infty$ and its derivative $\frac{\partial\|S_i\|_\infty}{\partial T_i}$ off-line. Using this information and the knowledge that the optimal operation point lies on the Liu-Layland hyper-surface, we can find the optimal operation point using a gradient based optimization algorithm [3] that searches for the minimum point on the hyper-surface. This algorithm can be performed online. But it still requires searching in the $N-1$ dimensional hyper-surface, and this may occupy significant amount of CPU time.

The optimization problem can also be solved by using the Lagrange multiplier method. Define the Lagrangian $\mathcal{L}$ as follows:

$$\mathcal{L} = \mathcal{S}_a + \lambda(\sum_{i=1}^N \frac{Q_i}{T_i} - a) \tag{13}$$

where λ is the Lagrange multiplier. Taking derivatives with respect to T_i and λ, we get, for $i = 1, 2, ..., N$, the following set of equations:

$$\frac{\partial\|S_i\|_\infty}{\partial T_i} - \lambda\frac{Q_i}{T_i^2} = 0 \tag{14}$$

$$\sum_{i=1}^N \frac{Q_i}{T_i} - a = 0. \tag{15}$$

Solving the N equations in (14) for λ, we obtain $\lambda = \frac{T_i^2}{Q_i}\frac{\partial\|S_i\|_\infty}{\partial T_i}$ for $i = 1, 2, ..., N$. Then equation (15) can be solved for T_i. We use the short hand notation $\partial_i\|S_i\|_\infty$ to represent $\frac{\partial\|S_i\|_\infty}{\partial T_i}$. Then for all $i, j = 1, 2, ..., N$, we have

$$T_j = \sqrt{\frac{Q_j}{Q_i}\frac{\partial_i\|S_i\|_\infty}{\partial_j\|S_j\|_\infty}}T_i. \tag{16}$$

Next we solve (15) to get the following results:

$$T_i = \frac{1}{a}\sum_{j=1}^N \sqrt{Q_iQ_j\frac{\partial_j\|S_j\|_\infty}{\partial_i\|S_i\|_\infty}}. \tag{17}$$

This is the solution for the operation point $(T_1, T_2, ..., T_N)$ on the Liu-Layland hyper-surface L.

To reduce the computing time, we can use a heuristic algorithm to find a suboptimal operation point for the objective function $\mathcal{S}_a$. We first estimate the average gradient of the objective function:

$$\bar{\nabla}\mathcal{S}_a \triangleq \left[\bar{\partial}_1\|S_1\|_\infty, \bar{\partial}_2\|S_2\|_\infty, ..., \bar{\partial}_N\|S_N\|_\infty\right]. \tag{18}$$

This estimation can be performed off-line. If we replace $\partial_i\|S_i\|_\infty$ and $\partial_j\|S_j\|_\infty$ in (17) with the components of the averaged gradient $\bar{\partial}_i\|S_i\|_\infty$ and $\bar{\partial}_j\|S_j\|_\infty$ in (18), then a heuristic operation point can be computed by (17) for $i = 1, 2, ..., N$. This heuristic algorithm serves as an *online feedback scheduling algorithm* that adjusts the task periods $(T_1, T_2, ..., T_N)$ according to the changing task execution time $(Q_1, Q_2, ..., Q_N)$.

This feedback scheduling algorithm also works for the case when the objective function $\mathcal{S}_b$ is used. In this case the optimal operation point still locates on the Liu-Layland hyper-surface. The optimal points locate either inside the hyper-surface where $T_i < \tau_i$ for all i, or locate at the boundary of the hyper-surface where $T_i = \tau_i$ for some i. If the operation point is in the interior of the hyper-surface, the normal vector to the hyper-surface must belong to the generalized gradient of the non-smooth function $\mathcal{S}_b$ [6].

The normal vector to the hyper-surface is aligned with the vector $\left[-\frac{Q_1}{T_1^2}, -\frac{Q_2}{T_2^2}, ..., -\frac{Q_N}{T_N^2}\right]$ but with unit length. The generalized gradient of $\mathcal{S}_b$ is the set that contains all the N dimensional vectors v such that the ith component v_i satisfies

$$v_i = \mu_i\frac{\bar{\partial}\|S_i\|_\infty}{\partial T_i} \tag{19}$$

for some $0 \le \mu_i \le 1$ and for $i = 1, 2, ..., N$. The rule to determine μ_i is as follows: if $\mathcal{S}_b = S_i$, then μ_i can be any number between 0 and 1; if $\mathcal{S}_b \neq S_i$, then $\mu_i = 0$.

Theorem 3.3 *At the optimal operation point in the interior of the hyper-surface L, $\|S_i\|_\infty$ are identical for all $i = 1, 2, ..., N$.*

Proof We prove by contradiction. Suppose $\|S_1\|_\infty > \|S_i\|_\infty$ for all $i = 2, ..., N$. Since the function $\|S_1\|_\infty$ is monotone, we can move the operation point along the hyper-surface to reduce T_1 and hence reduce $\|S_1\|_\infty$. The decrease in T_1 causes increases in other time periods. But since no time period is at the boundary of L, this movement of the operation point is valid. This will then reduce $\mathcal{S}_b$. This contradicts with the assumption that the operation point is optimal. ■

With the above theorem, we conclude that at the optimal operation point, $0 < \mu_i \le 1$ for all i. Let all μ_i to be identical will give us a heuristic operation point that is the same as the one when $\mathcal{S}_a$ is used.

3.3. Non-critical operation points

In the most general situation, the H_∞ norm may not be monotone in T. In this case, we may still apply the heuristic algorithms to find the optimal solutions on the Liu-Layland hyper-surface. However, we are also interested in local optimal solutions that occur in the interior of the feasible set F. Let us call solutions in the interior of F the interior minimums and solutions at the boundaries of F the boundary minimums. An interior minimum is often preferred since they are robust to boundary changes. Hence, small perturbations of Q_i and τ_i do not trigger significant changes in the scheduling algorithm and in the system performance.

Definition 3.4 *We name an interior (local) minimum for the optimization problem (8) or (10) a* non-critical operation point*. We say a cyber-physical system is* non-critical *if there exists at least one non-critical operating point.*

The non-critical operation points can be computed offline by finding the points where the gradient of the cost function vanishes. A list of these points can be stored in memory. When boundary conditions change dramatically so that the current operation point becomes invalid, and if a new non-critical operation point can be found as valid, the system can be switched to the new point without solving the optimization problem online. Of course, if a new non-critical operation point does not exist, we can use the heuristic algorithms in Section 3.2 to find operation points on the hyper-surface.

We now use a simple model for power consumption by the processor and change the objective function to include this power consumption. The resulting objective function stops being monotone and non-critical operation points can be found.

If we assume a constant clock rate and fixed voltage scheduling scheme, the processor power for N independent tasks can be modeled as

$$\mathcal{P}(T_1, T_2, ..., T_N) = U + V \sum_{i=1}^{N} \frac{1}{T_i} \tag{20}$$

where the coefficients U and V are positive real numbers that can be determined experimentally. Suppose U and V are known, we may formulate an optimization problem

$$\begin{aligned} &\text{Minimize } \mathcal{S}_a + \beta\mathcal{P} \\ &\text{under the following constraints:} \\ &0 \le \sum_i Q_i/T_i \le a \\ &Q_i \le T_i \le \tau_i \end{aligned} \tag{21}$$

where $\beta > 0$ is a scaling factor.

A candidate noncritical operation point exists where $\nabla(\mathcal{S}_a + \beta\mathcal{P}) = 0$. This implies that $\partial_i \|S_i\|_\infty = \frac{\beta V}{T_i^2}$. Therefore, the period T_i can be solved as $T_i = \sqrt{\frac{\beta V}{\partial_i \|S_i\|_\infty}}$. The term $\sqrt{\partial_i \|S_i\|_\infty}$ can replaced by the components of the averaged gradient, i.e., $\sqrt{\bar{\partial}_i \|S_i\|_\infty}$ for fast computation of operation points.

4. Transient performance

Our online feedback scheduling algorithms are capable of changing the operation points. Such changes may take a certain amount of time to happen in a processor. During this period, and in the worst case, computing of the control commands may fail. This failure in a short period of time will have effects on the physical system. After the processor shifts to a new schedule, the physical system typically needs significant time to settle to the new "steady state". This period of time, between the event that a scheduler starts shifting its operation point and the physical system reaches a new steady state, is called the *transient time*. The first portion of the transient time is the response time for the scheduler, and the second portion of the transient time is the response time for the physical system. System performance during the transient time needs to be analyzed to guarantee safety and performance.

In the application of controlling inverted pendulums, the response time of the scheduler can be measured by how many control periods are missed. During this period, the control effort stops being a function of the state variable θ which is the first component of vector x_k. To simplify our discussion, we suppose that the control effort vanishes. The physical dynamics of one pendulum without control is then

$$x_{k+1} = e^{AT} x_k \tag{22}$$

Suppose that m periods of the control commands are missing. Then at the end of the period, the system state is

$$x_{k+m} = e^{mAT} x_k \tag{23}$$

Because one eigenvalue of matrix A has positive real parts, the corresponding eigenvalue of matrix e^{mAT} lies out of the unit circle in the complex plane. Then we will have $|\theta_{k+m}| > |\theta_k|$ and the eigenvalues of e^{mAT} determines how fast θ grows. In the worst case, the maximum $|\theta_k|$ is given by $\|S(\omega, T)\|_\infty E$ where E is the maximum noise strength. Hence the maximum error caused by missing m periods of control commands is given by

$$|\theta_{k+m}| = \rho(e^{mAT}) \|S(\omega, T)\|_\infty E \tag{24}$$

where the $\rho(e^{mAT})$ is the maximum magnitude of the eigenvalues of e^{mAT}.

After the scheduler successfully sets up the new operation point and the control commands recovered, the pendulum will be controlled to the new "steady state" where $|\theta_k| \leq \|S(\omega, \hat{T})\|_\infty E$ where $\hat{T}$ is the new control period. The time it takes for this to happen can be estimated by studying the closed-loop dynamics. To simplify the analysis, let us first assume that the effect caused by time delay and sampling can be omitted. Then the closed-loop dynamics is

$$x_{j+1} = (e^{(A-BK)\hat{T}})x_j \tag{25}$$

where K is the control gain. We then have $|\theta_{j+n}| = \rho(e^{(A-BK)n\hat{T}})|\theta_j|$. Starting from $|\theta_j| = \rho(e^{mAT})\|S(\omega, T)\|_\infty E$, the time it takes to have $|\theta_{j+n}| = \|S(\omega, \hat{T})\|_\infty E$ must be

$$n\hat{T} = \frac{1}{\rho(A-BK)}\left(\rho(A)mT - \log(\frac{\|S(\omega,\hat{T})\|_\infty}{\|S(\omega,T)\|_\infty})\right) \tag{26}$$

Therefore, for each pendulum, the transient time T_i' can be computed as

$$\begin{aligned} T_i' &= m_i T_i + n_i \hat{T}_i \\ &= \left(\frac{\rho(A-BK)+\rho(A)}{\rho(A-BK)}mT \right. \\ &\left. -\frac{1}{\rho(A-BK)}\log(\frac{\|S(\omega,\hat{T})\|_\infty}{\|S(\omega,T)\|_\infty})\right)_i \end{aligned} \tag{27}$$

where the index i indicates the ith pendulum and $n_i\hat{T}_i$ is given by (26) with index i. Therefore, for the entire N pendulum system, the minimum interval $\mathcal{T}$ between transitions of operation points must satisfy $\mathcal{T} > \max_i T_i'$ for $i = 1, 2, ..., N$. This allows the system to "absorb" the transients caused by switching operation points.

5. Simulation results

The simulations of controlling three pendulums are set up using Truetime and the control systems toolbox of MATLAB. We assume all pendulums have mass $m = 0.5$kg with friction coefficient $\gamma = 0.6$. The pendulums differ from each other by their length selected as $[l_1, l_2, l_3] = [0.2, 0.35, 0.5]$.

Following the design procedure described in Section 2, we select control gains for the three pendulums as $K_1 = 5$, $K_2 = 7$, and $K_3 = 9$. Therefore, the maximum allowed time delays are $\tau_1 = 0.0986$s, $\tau_2 = 0.0614$s, and $\tau_3 = 0.0454$s. Since a longer pendulum has lower natural frequency, the control gain has to be higher to suppress low frequency noise, resulting in a shorter allowable delay.

We first compare the online and the offline scheduling algorithms originated from equation (17). The offline algorithm requires solving (17), and the online algorithm only needs an estimate of the averaged gradient. We select the utilization factor to be $a = 0.6$ and choose the computing time for the controllers to be identically $Q = 0.004$s. Then the online algorithm produces an operation point $[T_1, T_2, T_3] = [0.0489, 0.0232, 0.0116]$s with the cost function $\mathcal{S}_a = 8.25$, but the offline algorithm produces an operation point $[0.0303, 0.0208, 0.0154]$ with the cost function $\mathcal{S}_a = 7.76$. It is easy to verify that both operation points are very close to the Liu-Layland hyper-surface. From comparing the operation points we observe that the online algorithm produces longer period for the shorter pendulums than the offline algorithm. Figure 7 illustrates the step response for pendulum 1. We see that the offline algorithm (dotted line) delivers better performance. The step responses for the other two pendulums are almost indistinguishable between the two algorithms. These results illustrate the optimality of the offline algorithm. The schedulability is demonstrated in Figure 8.

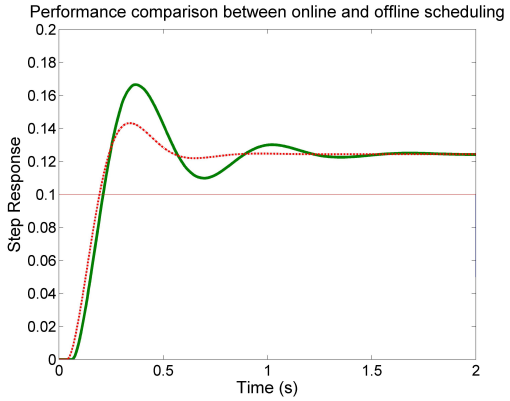


Figure 7. The step response for pendulum 1. Solid line (green) is produced by online algorithm; dotted line (red) is produced by off-line algorithm.

We now illustrate a procedure to co-design the control gain K and the operation point. Keeping $a = 0.6$, when Q doubles to 0.008. The offline algorithm shifts the operation point to $[0.0607, 0.0396, 0.0293]$ with cost $\mathcal{S}_a = 14.27$. But the online algorithm shifted the operation point to $[0.0978, 0.0465, 0.0231]$ with high cost $\mathcal{S}_a = 35.75$. This high cost indicates that the system is about to loose stability. We can also see pendulum 1 has delay that is close to the allowable maximum delay. Simulation on Truetime confirms this observation by showing that pendulum 1 starts oscillation. In this situation, the control gain for K_1 must be decreased. Decreasing the gain K_1 by half to 2.5 will increase the allowable time delay to 0.1908. Then the online algorithm produces a new operation point at $[0.1176, 0.0453, 0.0225]$ with $\mathcal{S}_a = 13.75$. Simulation

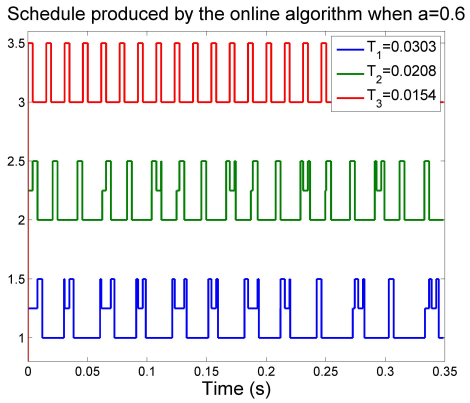


Figure 8. Schedule for control tasks.

shows that the system regains stability.

6. Conclusions

This paper develops theoretical results in designing scheduling algorithms for control applications of CPS to achieve balances among robustness, schedulability and power consumption. Our algorithms minimizes the H_∞ norm of SISO system subject to constraints posted by schedulability to produce operation points on the Liu-Layland hypersurface. We explore the existence of non-critical operation points that are robust to variations in CPU time. Approximations of these operation points can be found by heuristic algorithms that are implemented as online scheduling algorithms. Shifting of operation points typically causes performance degradation over the transient time. Such transient time limits the frequency to shift operation points by the online scheduling algorithm.

7. Acknowledgments

We would like to thank Dr. X. S. Hu, Dr. G. Quang, Dr. A. K. Mok, Dr. M. Lemmon and Dr. R. Murray for discussions and suggestions. The research is supported by ONR grant N00014–08–1–1007, NSF grant ECCS-0841195, and NSF grant 0720536.

References

[1] A. Azevedo, I. Issenin, R. Cornea, R. Gupta, N. Dutt, A. Viedenbaum, and A. Nicolau. Profile-based dynamic scheduling using program checkpoints. In *Proc. IEEE DATE 2002*, pages 168–178, 2002.

[2] P. R. Belanger. *Control Engineering: A Mordern Approach*. Saunders College Publishing, 1995.

[3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, London, 2004.

[4] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.-E. Arzen. How does control timing affect performance. *IEEE Control Systems Magazine*, (6):16–30, 2003.

[5] T. Chantem, X. S. Hu, and M. Lemmon. Generalized elastic scheduling. In *Proc. 27th RTSS*, 2006.

[6] F. H. Clarke, Y. S. Ledyaev, R. J. Stern, and P. R. Wolenski. *Nonsmooth Analysis and Control Theory*. Springer, New York, 1998.

[7] T. Ishihara and H. Yasuura. Voltage scheduling problems for dynamically variable voltage processors. In *Proc. IEEE ISLPED 1998*, pages 197–202, 1998.

[8] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proc. IEEE DATE 2004*, pages 275–281, 2004.

[9] T.-W. Kuo and A. K. Mok. Load adjustment in adaptive real-time systems. In *Proc. 12th RTSS*, pages 160–170, 1991.

[10] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a Hard-Real-Time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[11] K. Ogata. *Modern Control Engineering (4th ed.)*. Prentice Hall, NJ, 2002.

[12] L. Palopoli, C. Pinello, A. Sangiovanne-Vincentelli, L. Elghaoui, and A. Bicchi. Synthesis of robust control systems under resource constraints. In C. J. Tomlin and M. Greenstreet, editors, *HSCC 2002, LNCS 2289*, pages 337–350. Springer-Verlag, 2002.

[13] Z. Ren, B. H. Krogh, and R. Marculescu. Hierarchical adaptive dynamic power management. *IEEE Transactions on Computers*, 54(4):409–420, 2005.

[14] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin. On task schedulability in real-time control systems. In *Proc. of 17th IEEE RTSS*, pages 13–21, 1996.

[15] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin. Trade-off analysis of real-time control performance and schedulability. *Real-Time Systems*, 21(3):199–217, 2001.

[16] L. Sha, T. Abdelzaher, K.-E. Årzén, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok. Real time scheduling theory: A historical perspective. *Real-Time Systems*, 28(2-3):101–155, 1994.

[17] D. Simon, D. Robert, and O. Sename. Robust control/scheduling co-design: Application to robot control. In *Proc. of 11th IEEE RTAS*, 2005.

[18] F. Xia and Y. Sun. Control-scheduling codesign: A perspective on integrating control and computing. In *Dynamics of Continuous, Discrete and Impulsive Systems - Series B: Applications and Algorithms, Special Issue on ICSCA'06*, pages 1352–1358, Rome, Italy, 2006. Watam Press.

[19] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proc. IEEE 36th Annual FOCS*, pages 374–382, 1995.

[20] K. Zhou and J. C. Doyle. *Essentials of Robust Control*. Prentice-Hall, New Jersey, 1998.