A New Model of Computation for Reduction of Execution Time of LTI Controllers

Raktim Bhattacharya * Gary J. Balas †

Department of Aerospace Engineering & Mechanics,

University of Minnesota,

Minneapolis, MN-55455

November 11, 2002

Abstract

In this paper we present an algorithm for computationally efficient digital implementation of linear time invariant controllers. The algorithm transforms a linear time invariant controller into a periodically time varying system, which can be digitally implemented in a computationally efficient manner. This is achieved by first decomposing the controller into a dual rate system. A scheduling policy is adopted that spreads the computation, required to update the states of the slower system, over a time horizon. This reduces the total number of states updated at a given time step and hence the computational overhead. A theoretical framework is also developed to analyse the effect of this transformation on the closed-loop stability performance. The theoretical analysis framework relies on multi-rate filter bank theory and lifting techinique used in the analysis of multi-rate control systems.

*Graduate Assistant, raktim@aem.umn.edu

†Professor, balas@aem.umn.edu

1 Introduction

Most control algorithms today are implemented in digital computers. The popularity is due to the versatility of implementing control algorithms in software and the drop in the cost of computing. Increasingly complex control systems are now being designed because implementation can be entirely software based. Therefore control systems today, are essentially a composite of computational tasks. Since most control systems interact with the real world, the constituting computational elements of the control system need to execute in real-time. Therefore control algorithms are realised as real-time computational systems during implementation.

For economic reasons, the hardware used to realise a control system, usually executes several computational tasks, each with its own deadline. Several scheduling algorithms that guarantee on-time completion of these computational tasks have been proposed and studied in the past three decades. In 1973, Liu and Layland [1] published a seminal paper that addressed scheduling algorithms for multiprogramming in a hard real-time system. Since then, a vast amount of work has been done by both the operating research and computer science communities. Ramamritham and Stankovic in [2] summarises the current state of the real-time scheduling algorithms.

The execution time or run-time of the computational tasks, that constitute a real-time control system, plays a vital role in any real-time scheduling algorithm. Depending on the task set and their run-times, a scheduling algorithm may or may not be feasible. A scheduling algorithm is said to be feasible for a task set, if it guarantees on time completion of all the tasks. Therefore, if the run-time of the computational tasks can be reduced, then more tasks could be feasibly scheduled in a hardware with a given clock speed or dually, a task set can be feasibly realised in a hardware with slower clock speed. In addition to this, reduction in the run-times of the scheduled tasks increase the guaranteeability of tasks with timing constraints. The motivation for the research work presented in this paper stems from these facts.

In this paper we present an algorithm that transforms linear time invariant controllers to periodically time varying systems, which can be digitally implemented in a computationally efficient manner. Controllers designed in continuous time are discretised when implemented in digital computers. The sampling frequency of the discrete-time controller is typically chosen to be ten times faster than the cutoff frequency of the closed-loop system. If the natural frequencies of the controller are sparsely spaced, then the states corresponding to the slow modes are updated at a rate more than necessary. This results in unneccesary computation.

The computational overhead or execution time can be reduced if the states corresponding to the slow modes of the controller are updated at a slower rate. The simplest transformation would be to decompose the controller into two subsystems (or computational tasks), one containing the fast modes and the other the slow modes. Reduction in computational overhead can then be achieved by simply operating the two subsystems at different rates. With such a scheme however, there will be a periodic increase in computational requirement at time instants when both the subsystems need to be updated, which is not desirable. One of the salient features of the proposed algorithm is the distribution of the computation, required to update the slowly varying states, over time to achieve a uniform reduction in the computational overhead. From the point of view of real-time tasks, this algorithm decomposes the original task into two tasks, each with reduced run-time but different periodicity.

The paper is organised as follows. Initially some fundamental techniques used in multi-rate systems and digital signal processing are presented. This is followed by details of the transformation of a linear time invariant (LTI) controller to a multi-rate system and the scheduling policy used to update the states to reduce the computational overhead. A framework to analyse of the effect of such a transformation on the closed-loop performance and stability is developed. The theoretical framework for analysis is built on multi-rate filter bank theory and lifting techinique used in the analysis of multi-rate control systems. In the end, this approach is applied to a B737-100 TSRV (Transport System Research Vehicle) linear longitudinal motion model and simulation results are presented. A summary section concludes the paper.

2 Fundamentals of Digital Multi-Rate Systems

This section presents some fundamentals used in analysing digital multi-rate systems. The content of this section is based on references [3, 4]

2.1 Sampler S

Let us denote S as the sampler that transforms continuous-time signal to a discrete-time signal. The block diagram shown in Fig.1, takes in continuous-time signal $v(t)$ and transforms it to discrete-time signal $w(k)$.

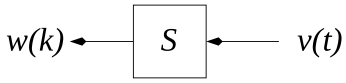


Figure 1: A sampler.

Denoting $V(j\omega)$ and $W(e^{-j\omega h})$ as the Fourier transform of $v(t)$ and $w(k)$, the following relationship holds (pg. 50 of [4])

$$W(e^{-j\omega h}) = \frac{1}{h} V_e(j\omega) \tag{1}$$

where h is the sampling-time and

$$V_e(j\omega) := \sum_{k=-\infty}^{\infty} V(j\omega + jk\omega_s); \omega_s = 2\pi/h$$

If $v(t)$ is a band-limited signal, i.e. $V(j\omega) = 0$ for $|\omega| > \frac{\omega_s}{2}$, then

$$W(e^{-j\omega h}) = \frac{1}{h} V(j\omega); \text{for} |\omega| < \frac{\omega_s}{2} \tag{2}$$

2.2 Zero-Order Hold H

A zero-order hold is a device H, that transforms a discrete-time signal to a continuous-time signal. The transformation is achieved by a simple causal reconstruction given by

$$y(t) = y(t_k) \; ; \; t_k \leq t < t_{k+1}$$

This means that the reconstructed signal is piecewise constant, continuous from the right, and equal to the sampled signal at the sampling times. The reconstructed value is thus held constant until the next sampling instant. The system H has the impulse response

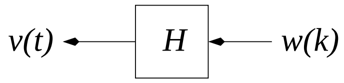


Figure 2: A zero-order hold.

$$H(t) = \frac{1}{h}1(t) - \frac{1}{h}1(t-h)$$

The transfer function is therefore (pg 52 in [4])

$$H(j\omega) = \frac{1 - e^{-j\omega h}}{j\omega h} \tag{3}$$

The Fourier transforms of $v(t)$ and $w(k)$ in Fig.2, are related by the following equation

$$v(j\omega) = hH(j\omega)W(e^{-j\omega h}) \tag{4}$$

2.3 Lifting Operator

Lifting is a commonly used technique in the analysis of multi-rate systems [5, 6]. An interconnection of multi-rate systems can be analysed, using tools developed for single-rate systems, by lifting the faster systems to match the rate of the slowest system. The lifting operator is defined both in continuous time (Ch.10 in [4]) and in discrete time (Ch.8 in [4]). Discrete-time lifting techniques are used in this paper, the basics of which are described in the following sections.

2.3.1 Lifting of Discrete-Time Signals

Suppose $v(k) = \{v(0), v(1), \cdots\}$ is a discrete-time signal. If we rewrite the signal as

$$\left\{ \begin{pmatrix} v(0) \\ v(1) \\ \vdots \\ v(M-1) \end{pmatrix}, \begin{pmatrix} v(M) \\ v(M+1) \\ \vdots \\ v(2M-1) \end{pmatrix}, \dots \right\} \equiv \{\underline{v}(0), \underline{v}(1), \dots .\} \tag{5}$$

then the mapping of $v(k) \mapsto \underline{v}(k)$ is denoted by $\underline{v}(k) = L_M v(k)$. The operator L_M is called the *lifting operator* and the signal $\underline{v}(k)$ is called the *lifted signal.* The subscript in L_M denotes the factor by which the dimension of the signal has been increased or lifted. The inverse operation, i.e the mapping $\underline{v}(k) \mapsto v(k)$, is the reconstruction of $v(k)$ from $\underline{v}(k)$. This is denoted by $v(k) = L_M^{-1}\underline{v}(k)$. In general we will drop the subscript from L. The factor by which the dimension of the signal is lifted will be clear from context.

The operator L, as a system is non-causal and time-varying, however it is norm-preserving (pg.204 in [4]). Therefore

$$||Lv||_2 = ||v||_2$$

The inverse operator L^{-1} is causal but time-varying.

2.3.2 Lifting Discrete-Time Systems

Consider the system shown in Fig.3, where $\hat{G}$ is a discrete-time, finite-dimensional LTI system

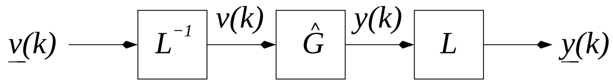


Figure 3: Lifting of discrete-time LTI system.

with underlying period h/M. Lifting the input and output signals so that the lifted signals correspond to the base period h results in the lifted system $\underline{\hat{G}}$, defined as $\underline{\hat{G}} \equiv L_M \hat{G} L_M^{-1}$.
If $\hat{G} = \left[\begin{array}{c|c} \hat{A} & \hat{B} \\ \hline \hat{C} & \hat{D} \end{array}\right]$, then the lifted system can be written as

$$\underline{\hat{G}} = \left[\begin{array}{c|cccc} \hat{A}^M & \hat{A}^{M-1}\hat{B} & \hat{A}^{M-2}\hat{B} & \cdots & \hat{B} \\ \hline \hat{C} & \hat{D} & 0 & \cdots & 0 \\ \hat{C}\hat{A} & \hat{C}\hat{B} & \hat{D} & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ \hat{C}\hat{A}^{M-1} & \hat{C}\hat{A}^{M-2}\hat{B} & \hat{C}\hat{A}^{M-3}\hat{B} & \cdots & \hat{D} \end{array}\right] \tag{6}$$

If $\hat{A}$ is stable then $\hat{A}^M$ is also stable. Since lifting preserves norms, it follows that the norms of the two transfer functions $\hat{G}$ and $\underline{\hat{G}}$ satisfy (pg.206 in [4]):

$$||\hat{G}||_2 = ||\underline{\hat{G}}||_2/M \tag{7}$$

$$||\hat{G}||_\infty = ||\underline{\hat{G}}||_\infty \tag{8}$$

2.4 M-Fold Decimator

A M-fold decimator, represented by $\downarrow M$ in Fig.4, is a device that takes an input sequence $v(k)$ and returns

$$v_D(k) = v(kM) \tag{9}$$

where M is an integer. Only those samples of $v(k)$ which occur at time equal to multiples of M are retained by the decimator. In terms of z transforms, v and v_D are related as

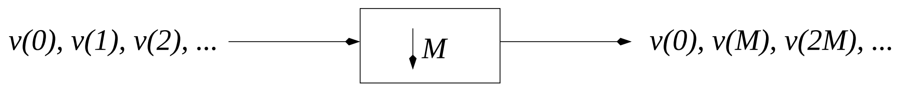


Figure 4: A M-fold decimator.

$$V_D(z) = \frac{1}{M}\sum_{n=0}^{M-1} V(z^{1/M} \times e^{-j2\pi n/M}) \tag{10}$$

Representing $e^{-j2\pi/M}$ as W, the M^{th} root of unity, we can rewrite eqn.(10) as

$$V_D(z) = \frac{1}{M}\sum_{n=0}^{M-1} V(z^{1/M} W^n) \tag{11}$$

M-fold decimators are linear time-varying systems that down-sample the signal at input. From eqn.(11) we see that decimation causes aliasing by adding multiple copies of the compressed spectrum. In general, to avoid aliasing in down-sampling by a factor of M, the bandwidth of the admitted signal should be limited to ω_N/M, where $\omega_N = \omega_s/2$ is the Nyquist frequency. This is achieved by passing the signal through a low-pass filter prior to down-sampling.

2.5 M-Fold Expander

A M-fold expander, represented by $\uparrow M$ in Fig. 5 , is a device that takes in input sequence $v(k)$ and produces output

$$v_E(k) = \begin{cases} v(k/M) & ; \ k = nM \ ; \ n \in \{0, 1, \cdots\} \\ 0 & ; \ k \neq nM \ ; \ n \in \{0, 1, \cdots\} \end{cases}$$

where M is a positive integer.

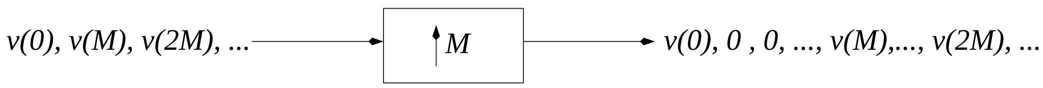


Figure 5: M-fold expander.

In terms of z-transforms, v_E and v are related as

$$V_E(z) = V(z^M) \tag{12}$$

Thus the Fourier transform of the output of the expander is a frequency-scaled version of the Fourier transform of the input. Since Fourier transforms repeat itself every 2π, the Fourier transform of the output will contain M copies of the compressed spectrum of the input. These multiple copies in the spectrum of $V_E(z)$ are called images. Hence the expander causes an imaging effect.

The zero-valued samples at $k \neq nM$ and the imaging effect created by the expander can be eliminated by passing the output of the expander through a lowpass filter with gain M and cutoff ω_N/M. The function of the lowpass filter, called the interpolation/expansion filter, in frequency domain is to remove the images created by the expander. While in the time domain, the convolution of v_E with the impulse response of the filter results in the filling up of the zero-valued samples with interpolated values. Depending on the nature of the recontruction of v_E, the interpolation filter can be designed to linearly interpolate the intermediate values or hold constant the values at $k = nM$ for M time-steps in the future.

2.6 Type 1 Polyphase Decomposition of Digital Filters

To explain the basic idea behind the Type 1 polyphase decomposition of a digital filter, as defined in page 121 of [3], consider a digital filter

$$\hat{H}(z) = \sum_{n=-\infty}^{\infty} h(n)z^{-n}$$

By separating the odd and even terms we can write

$$\hat{H}(z) = \sum_{n=-\infty}^{\infty} h(2n)z^{-2n} + z^{-1} \sum_{n=-\infty}^{\infty} h(2n+1)z^{-2n}$$

Extending this idea further to M terms we can decompose $\hat{H}(z)$ as

$$\hat{H}(z) = \sum_{n=-\infty}^{\infty} h(nM)z^{-nM} + z^{-1} \sum_{n=-\infty}^{\infty} h(nM+1)z^{-nM} + \cdots + z^{-(M-1)} \sum_{n=-\infty}^{\infty} h(nM+1-M)z^{-nM}$$

This can be compactly written as

$$\hat{H}(z) = \sum_{l=0}^{M-1} z^{-l}\hat{H}_l(z^M) \tag{13}$$

where

$$\hat{H}_l(z) = \sum_{n=-\infty}^{\infty} h_l(n)z^{-n} \tag{14}$$

with

$$h_l(n) \equiv h(Mn+l), \quad 0 \leq l \leq M-1$$

The filters $\hat{H}_l(z)$ are called the polyphase components of the filter $\hat{H}(z)$.

2.7 Lifting Operator and Blocking Mechanism in Digital Signal Processing

Figure 6(a) represents a digital filter and Fig.6(b) represents the same implementation as a block digital filter. The signals $y_B(k)$ and $u_B(k)$ are the blocked versions of the signals $y(k)$ and $u(k)$

and the block filter $\hat{\mathbf{H}}(z)$ is the blocked version of the filter $\hat{H}(z)$ defined by (pg.431, [3]),

$$\hat{\mathbf{H}}(z) = \begin{bmatrix} \hat{H}_0(z) & \hat{H}_1(z) & \cdots & \hat{H}_{M-1}(z) \\ z^{-1}\hat{H}_{M-1}(z) & \hat{H}_0(z) & \cdots & \hat{H}_{M-2}(z) \\ \vdots & \vdots & & \vdots \\ z^{-1}\hat{H}_1(z) & z^{-1}\hat{H}_2(z) & \cdots & \hat{H}_0(z) \end{bmatrix} \tag{15}$$

where $\hat{H}_l(z)$ are the polyphase components of $\hat{H}(z)$ as defined in eqn.(14).

The blocking mechanism in digital signal processing is identical to the lifting operator used in digital control theory. Therefore the mapping of $y(k)$ to $y_B(k)$ in Fig.6(b) is identical to the mapping of $v(k)$ to $\underline{v}(k)$ defined in eqn.(5). Similarly, the mapping of $\underline{v}(k)$ to $v(k)$ is identical to the mapping of $u_B(k)$ to $u(k)$.

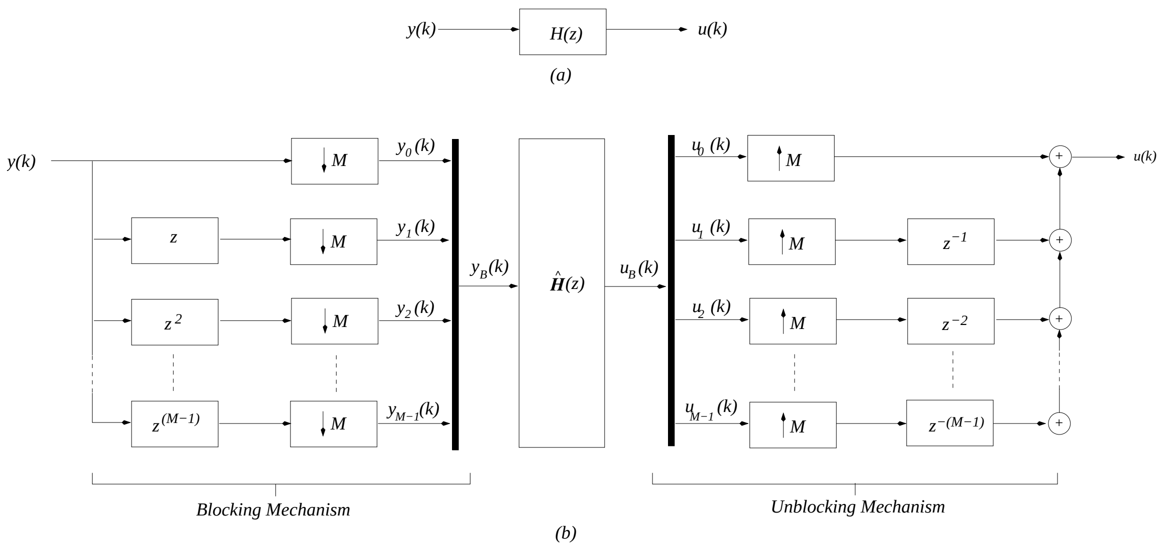


Figure 6: Representation of digital block filtering.

The block-filtering setup in Fig.6(b) can be represented using L and L^{-1} as shown in Fig.7. Figure 7(a) represents a linear discrete-time system $\hat{G}$, with input $y(k)$ and output $u(k)$. Introducing $LL^{-1} = I$ in both input and output channels results in Fig.7(b). By absorbing L^{-1} at the input and L at the output of $\hat{G}$, we transform it to $\underline{\hat{G}} = L\hat{G}L^{-1}$. Figure 7(c) is identical to Fig.6(b) if we replace $\underline{\hat{G}}$ by $\hat{\mathbf{H}(\mathbf{z})}$, L by the blocking mechanism and L^{-1} by the unblocking mechanism. Thus we see that the blocking mechanism in digital signal processing is equivalent to the lifting operator L in control theory, the L^{-1} operator is equivalent to the unblocking mechanism and the system $\underline{\hat{G}}(z)$ represents the blocked version of the system $\hat{G}(z)$.

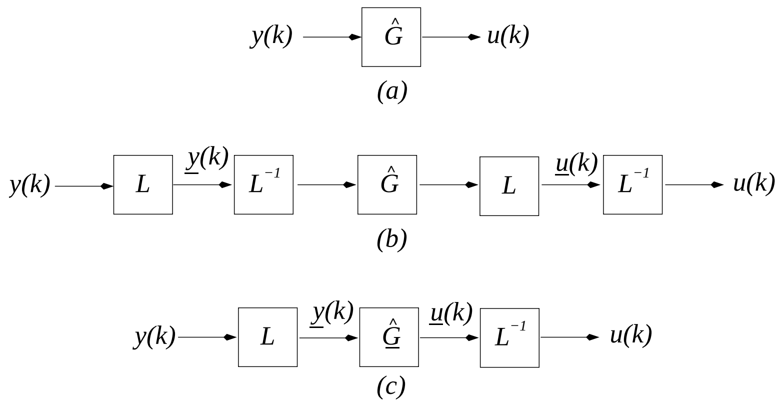


Figure 7: Lifting of discrete-time LTI system.

It is interesting to note that even though the decimators and the expanders in Fig.6(b) produce aliasing components $U(zW^k)$, $Y(z)$ is free from aliasing. A filter-bank shown in Fig.6(b) is alias free if, and only if, $\hat{\mathbf{H}}(z)$ is pseudo-circulant [7].

A matrix is said to be circulant (pg.249, [3]), if every row is obtained using a single right-shift of the previous row with the rightmost element, which spills over in the process, be circulated back to become the left most element. A pseudo-circulant matrix is a circulant matrix with the additional feature that the elements below the main diagonal are multiplied by z^{-1}. An example of a 3×3 pseudo-circulant matrix is

$$P(z) = \begin{bmatrix} P_0(z) & P_1(z) & P_2(z) \\ z^{-1}P_2(z) & P_0(z) & P_1(z) \\ z^{-1}P_1(z) & z^{-1}P_2(z) & P_0(z) \end{bmatrix}$$

From the definition of $\hat{\mathbf{H}}(z)$ in eqn.(15), we see that $\hat{\mathbf{H}}(z)$ is pseudo-circulant. Hence the input to output mapping in Fig.6(b) is alias free.

To determine the relationship between the Fourier transforms of $u(k)$ and $y(k)$ let us redraw Fig.6(b) with delay chains as shown in Fig.8. The relationship between the Fourier transforms

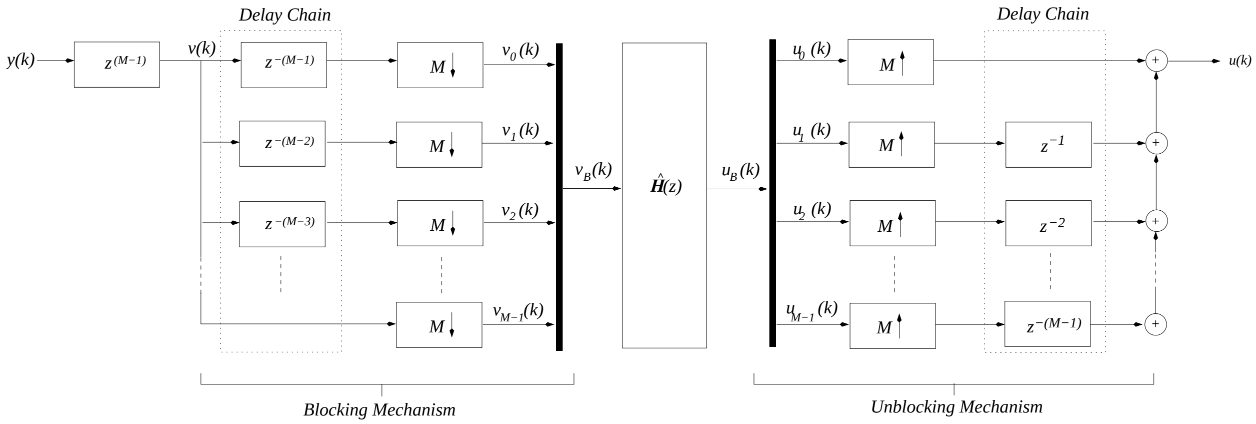


Figure 8: Representation of digital block filtering with delay chain.

of $v(k)$ and $u(k)$ in Fig.8 is given by (pg.253, [3]),

$$U(z) = \frac{1}{M} \sum_{s=0}^{M-1} \left(z^{-(M-1-s)} \sum_{l=0}^{M-1} z^{-l} \hat{H}_{s,l}(z^M) \right) V(z) \tag{16}$$

Substituting $V(z) = z^{M-1}Y(z)$ in eqn.(16), we get

$$U(z) = \frac{1}{M} \sum_{s=0}^{M-1} \left(z^{s} \sum_{l=0}^{M-1} z^{-l} \hat{H}_{s,l}(z^M) \right) Y(z) \tag{17}$$

where $\hat{H}_{s,l}(z)$ are the elements of $\hat{\mathbf{H}}(z)$. Equation 17 can be written in a more compact manner as

$$U(z) = \frac{1}{M} \begin{bmatrix} 1 & z^{-1} & z^{-2} & \cdots & z^{-(M-1)} \end{bmatrix} \hat{\mathbf{H}}(z^M) \begin{bmatrix} 1 \\ z \\ z^2 \\ \vdots \\ z^{(M-1)} \end{bmatrix} Y(z) \tag{18}$$

For the lifted system $\underline{\hat{G}}(z)$, the relationship between the Fourier transforms of $u(k)$ and $y(k)$ in Fig.7(c) is also given by eqn.(18) with $\hat{\mathbf{H}}(z^M)$ replaced by $\underline{\hat{G}}(z^M)$ where

$$\underline{\hat{G}}(z^M) = \underline{\hat{D}} + \underline{\hat{C}}(z^M I - \underline{\hat{A}})^{-1}\underline{\hat{B}}$$

with $(\underline{\hat{A}}, \underline{\hat{B}}, \underline{\hat{C}}, \underline{\hat{D}})$ defined by the matrix partitions in eqn.(6).

3 Transformation of LTI Controllers to Multi-Rate Systems

In this section we present a computationally efficient way of implementing discrete-time LTI controller. The computational overhead is measured in terms of the number of states being updated at a given time step. The proposed algorithm updates the states of the controller at a rate based on their natural frequencies. This results in a multi-rate system. Since all the states of the controller are not updated at the same time, the computational requirements for digital implementation is reduced. The effect of such a transformation on system behaviour is analysed using tools from multi-rate filter banks and lifting techniques.

Let us assume that the linear controller, denoted by K, has the following form,

$$K := \begin{cases} \dot{x}_c(t) &= A_c x_c(t) + B_c y(t) \\ u(t) &= C_c x_c(t) \end{cases}$$

where $u(t)$ denotes the controller output, $y(t)$ the plant output fed into the controller and $x_c(t)$ the controller states. Let us also decompose controller K into two sub-systems K_f and K_s, such that $K(s) = K_f(s) + K_s(s)$, where K_f and K_s contains the fast and slow modes, respectively. The system decomposition is shown in Fig.9.

In a typical digital implementation of K, the sampling rate ω_s is chosen to be ten times faster than the cutoff frequency of the closed-loop system. All the states of the controller are updated

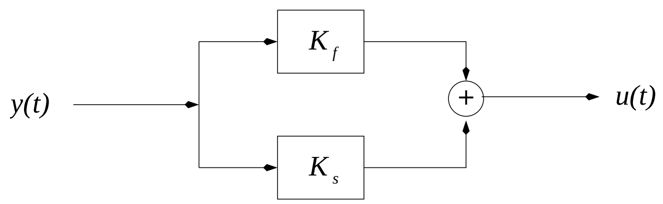


Figure 9: Transformation of LTI controllers to multi-rate systems.

at this rate. If the modes of K are sparsely distributed in frequency, then this update rate is more than sufficient for the states of K_s. We identify this as unnecessary computation, which can be avoided if the states of K_s are updated at a slower rate.

Let us assume that $h = 1/\omega_s$ is the time step used to discretise the original controller K. Let us also assume that K_s is such that is suffices to update its states every M time steps, where M is a positive integer. For digital implementation, K_f and K_s are transformed to discrete-time systems $\hat{K}_f$ and $\hat{K}_s$, with time-step h and hM, respectively. Therefore the states of $\hat{K}_f$ are updated at every time step, denoted by k, and the states of $\hat{K}_s$ are updated every kM time step.

From table(1), we see that the computational requirement at time steps $k \neq nM, \ n \in \mathcal{Z}^*$ is lower than that at time steps $k = nM$. This is so because at $k \neq nM$ only the states of K_f are being updated. At time steps $k = nM$, all the states of the controller are being updated and hence the computational requirement at these time steps is equal to the overhead encountered in conventional digital implementation. Conventional digital implementation implies that all states of the controller are updated at every time step.

Time Step	0	1	2	$\cdots$	M-1	M	M+1	$\cdots$
System Updated	$\hat{K}_f, \hat{K}_s$	$\hat{K}_f$	$\hat{K}_f$	$\cdots$	$\hat{K}_f$	$\hat{K}_f, \hat{K}_s$	$\hat{K}_f$	$\cdots$

Table 1: State updation pattern of dual-rate linear Controllers

Therefore, by simply implementing K as a multi-rate system, we can reduce the computational overhead at time-steps $k \neq nM$. However the periodic increase in the CPU requirement is undesirable and we wish to reduce the computational requirement uniformly at all time steps.

The key idea presented in this paper is the scheduling algorithm for updating the states of $\hat{K}_s$, so that the computational overhead is reduced at all time-steps.

3.1 Scheduling Algorithm for Uniform Reduction in CPU Overhead

Uniform reduction in computational overhead is achieved by distributing the computation, required to update the states of $\hat{K}_s$, over time. If the state space of $\hat{K}_s$ can be partitioned into M subsets, then the distribution of computation can be achieved by updating these subsets one after another. With such a update policy, all the states of $\hat{K}_s$ are updated every M time steps as required.

Since the subsets will contain fewer states than $\hat{K}_s$, updating them along with those of $\hat{K}_f$ will reduce the spikes in computational overhead at times $k = nM$ but at the same time will increase the computational requirements at times $k \neq nM$. Clearly, the uniformity of CPU overhead depends on the uniformity of the number of states in each partition.

The process of updating partial states of $\hat{K}_s$ can be achieved quite easily, if $\hat{K}_s$ is decomposed into modal form. Let us assume that $\hat{K}_s$ has M distinct eigen-values. Therefore modal decomposition of $\hat{K}_s$ will yield M sub-systems, denoted by $\hat{K}_s = \{\hat{\mathcal{K}}_{s_0}, \hat{\mathcal{K}}_{s_1}, \ldots ., \hat{\mathcal{K}}_{s_{M-1}}\}$. The computation required to update the states of K_s is spread over time by updating these modal sub-systems one after another. In this manner, all the states of $\hat{K}_s$ are updated in M time-steps.

From table(2), we see that at any given time-step k, systems $\hat{K}_f$ and $\hat{\mathcal{K}}_{s_i}$ are being updated, where i is the remainder of the integer division k/M. Note that this round-robin scheduling of $\hat{\mathcal{K}}_{s_i}$ has transformed the controller into a periodically time-varying system. Let us denote the controller with dynamics shown in table(2) as $\hat{\Psi}$ and the transformation of K to $\hat{\Psi}$ as $\hat{\Psi} = \mathcal{T}(K)$.

Time Step	0	1	2	$\cdots$	M-1	M	M+1	$\cdots$
System Updated	$\hat{K}_f, \hat{\mathcal{K}}_{s_0}$	$\hat{K}_f, \hat{\mathcal{K}}_{s_1}$	$\hat{K}_f, \hat{\mathcal{K}}_{s_2}$	$\cdots$	$\hat{K}_f, \hat{\mathcal{K}}_{s_{M-1}}$	$\hat{K}_f, \hat{\mathcal{K}}_{s_0}$	$\hat{K}_f, \hat{\mathcal{K}}_{s_1}$	$\cdots$

Table 2: Scheduling algorithm for uniform reduction in CPU overhead.

3.2 Formal Definition of the Transformation

In this section we formally define the transformation $\mathcal{T}(K)$ that achieves uniform reduction of computational overhead. Before we formally define $\mathcal{T}(K)$ some definitions are necessary.

Definition 1. *Let $\pi(K)$ denote the maximum number of distinct eigen-values of K, where K is a LTI system.*

Definition 2. *Let $\phi(K, h)$ denote the continuous to discrete time transformation of LTI system K, with discretisation time-step h. The discrete time system is denoted by $\hat{K} = \phi(K, h)$. If $K \equiv (A, B, C)$ then $\hat{K} \equiv (\hat{A}, \hat{B}, \hat{C})$.*

We assume that ϕ is such that for $\hat{K} = \phi(K, h)$, $\pi(\hat{K}) = \pi(K)$.

Definition 3. *Define $\mu(i, j)$ as*

$$\mu(i, j) = i - j \times \lfloor i/j \rfloor \, ; i, j \in \mathcal{Z}, j \neq 0$$

This function simply returns the remainder of the integer division (i/j).

Definition 4. *Define $\mathcal{M}(K)$ such that, $\mathcal{M}(K)$ decomposes a LTI system K, into modal form and generates a set of modal sub-systems $\{\mathcal{K}_i\}$ where $i \in \mathcal{Z}^*$, $i < \pi(K)$. Each of the modal systems $\mathcal{K}_i$ has associated*

- *state vector x_i*
- *output vector u_i*

- *input vector y, which is common to all the modal sub-systems*
- *and dynamics defined by $(\mathcal{A}_i, \mathcal{B}_i, \mathcal{C}_i)$.*

It is assumed that K has no direct feedthrough, therefore $\mathcal{K}_i$ also does not have any direct feedthrough.

At this point we are ready to formally define the transformation of a given continuous-time linear time invariant controller K to a multi-rate, linear periodically time varying, discrete-time system $\hat{\Psi}$ as follows,

Definition 5. *$\mathcal{T}(K)$ is the transformation from $K \mapsto \hat{\Psi}$, defined by the following steps*

1. *Decompose K into K_s and K_f, where K_f and K_s contain the fast and the slow modes of K, respectively.*
2. *Obtain $\hat{K}_f = \phi(K_f, h)$, where h is the time-period of the base clock.*
3. *Obtain $\hat{K}_s = \phi(K_s, h\pi(K_s))$.*
4. *Obtain $\{\hat{\mathcal{K}}_{s_i}\} = \mathcal{M}(\hat{K}_s), i = \{0, 1, \ldots, \pi(\hat{K}_s) - 1\}$.*
5. *Define dynamics of $\hat{\Psi}$ as*

$$\begin{aligned} x_f^{k+1} &= \hat{A}_f x_f^k + \hat{B}_f y^k \\ x_{s_i}^{k+1} &= \begin{cases} \hat{\mathcal{A}}_{s_i} x_{s_i}^k + \hat{\mathcal{B}}_{s_i} y_s^k & ;\, if\ i = \mu(k, \pi(\hat{K}_s)) \\ x_{s_i}^k & ;\, if\ i \neq \mu(k, \pi(\hat{K}_s)) \end{cases} \end{aligned} \tag{19}$$

In eqn.(19), the controller input y_s^k can be one of the following three

$$y_s^k = y^{\acute{k}};\ \ \acute{k} = \left\lfloor k/\pi(\hat{K}_s) \right\rfloor \tag{20}$$

$$y_s^k = y^k \tag{21}$$

$$y_s^k = \left(y^k + y^{k-1} + \cdots + y^{k-\pi(\hat{K}_s)+1}\right)/\pi(\hat{K}_s);\ \ y^k = 0\ for\ k < 0 \tag{22}$$

The first definition of y_s^k means that the states of $\hat{\mathcal{K}}_{s_i}$ are updated using the same value of the plant output. The second definition uses the most recent plant output to update x_{s_i}, and *the last definition uses the running average of the plant output over $\pi(\hat{K}_s)$ time-steps, to update the states of $\hat{\mathcal{K}}_{s_i}$.*

6. *Define output of $\hat{\Psi}$ as*

$$u^k = u_f^k + u_s^k \tag{23}$$

where $u_f^k = \hat{C}_f x_f^k$ and u_s^k is one of the following

$$u_s^k = \sum_{i=0}^{\pi(\hat{K}_s)-1} \hat{\mathcal{C}}_{s_i} x_{s_i}^{\acute{k}}; \quad \acute{k} = \left\lfloor k/\pi(\hat{K}_s) \right\rfloor \tag{24}$$

$$u_s^k = \sum_{i=0}^{\pi(\hat{K}_s)-1} \hat{\mathcal{C}}_{s_i} x_{s_i}^k \tag{25}$$

The difference between the two definitions of u_s^k is that in eqn.(24), the updated states of the modal systems do not affect the controller output until all the modal systems have been updated. Whereas in eqn.(25), the updated states of $\hat{\mathcal{K}}_{s_i}$ immediately affect u_s^k.

Since the input and output of $\hat{\Psi}$ has been defined in more than one way, the definition of $\hat{\Psi}$ as a dynamical system is dependent on the combination of the input and output definitions. In this paper, we only analyse the frequency domain characteristics of $\hat{\Psi}$ as defined by the following four combinations. We define,

- $\hat{\Psi}_1$ to be the dynamics of $\hat{\Psi}$ with y_s^k given by eqn.(20) and u_s^k given by eqn.(24).
- $\hat{\Psi}_2$ to be the dynamics of $\hat{\Psi}$ with y_s^k given by eqn.(21) and u_s^k given by eqn.(24).
- $\hat{\Psi}_3$ to be the dynamics of $\hat{\Psi}$ with y_s^k given by eqn.(21) and u_s^k given by eqn.(25).
- $\hat{\Psi}_4$ to be the dynamics of $\hat{\Psi}$ with y_s^k given by eqn.(22) and u_s^k given by eqn.(24).

3.3 Frequency Response of $\hat{\Psi} = \mathcal{T}(K)$

From control system point of view, it is important to analyse the dynamics of $\hat{\Psi} = \mathcal{T}(K)$ in the frequency domain. To determine the relationship between the norms $||y(k)||_2$ and $||u(k)||_2$, it is necessary to represent the mapping $y(k) \mapsto u(k)$ in terms of LTI systems. In the following subsection, we transform the multi-rate systems $\hat{\Psi}_i$, defined in section 3.2, to LTI systems [8] and define transfer function for $y(k) \mapsto u(k)$.

3.3.1 Transfer Function for $\hat{\Psi}_1$

From the definition of $\hat{\Psi}_1$, we see that all the modal systems are updated by the same value of the plant output, which is obtained at every kM time-steps, where $M = \pi(\hat{K}_s)$. Also, the states of $\hat{\mathcal{K}}_{s_i}$ do not affect u_s^k until all the modal systems have been updated. Therefore, from system point of view $\hat{\Psi}_1$ is identical to the dual-rate system shown in table(1). It only differs from the point of view of computation. Therefore the dynamics of $\hat{\Psi}_1$ can be represented by Fig.10.

Figure 10 represents the mapping $y(k) \mapsto u(k)$ in terms of samplers and holds. The subscripts s and f in the samplers and holds denote slow and fast sampling and hold respectively. The signal $y(k)$ is sampled at the rate required by $\hat{K}_f$ and the filter F is the anti-aliasing filter for the slow sampler S_s.

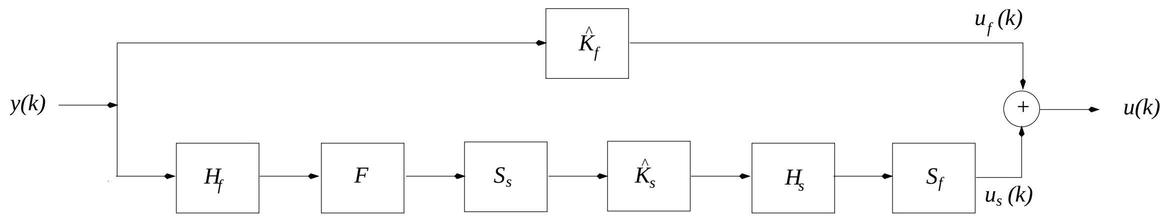


Figure 10: $\hat{\Psi}_1$ as a linear periodically time varying system.

The multi-rate system in Fig.10 can be transformed into a single rate system with the help of lifting operators as shown in Fig.11.

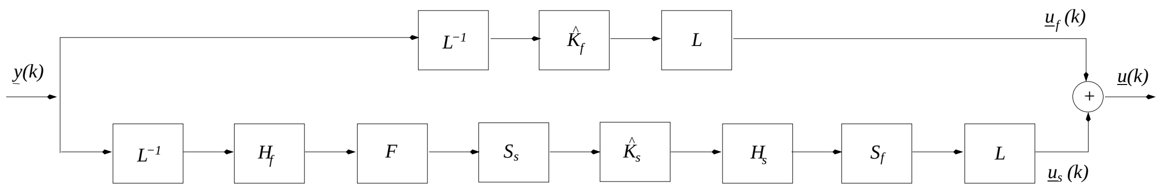


Figure 11: $\hat{\Psi}_1$ as a linear time invariant system.

Assuming (pg. 211-213, [4]) $S_f H_s = L^{-1}\begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix}$, $S_s = S_s H_f S_f$, $S_s H_f = \begin{bmatrix} I & 0 & \cdots & 0 \end{bmatrix} L$ and $\hat{G} = SGH$, the transfer function between $\underline{y}(k)$ and $\underline{u}(k)$, denoted by $\underline{\hat{\Psi}}$, can therefore be written as

$$\begin{aligned}
\underline{\hat{\Psi}}_1 &= L\hat{K}_f L^{-1} + L S_f H_s \hat{K}_s S_s F H_f L^{-1} \\
&= L\hat{K}_f L^{-1} + L(S_f H_s)\hat{K}_s (S_s H_f)(S_f F H_f) L^{-1} \\
&= L\hat{K}_f L^{-1} + L L^{-1}\begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix} \hat{K}_s \begin{bmatrix} I & 0 & \cdots & 0 \end{bmatrix} L(S_f F H_f) L^{-1}
\end{aligned}$$

or

$$\underline{\hat{\Psi}}_1 = \underline{\hat{K}}_f + \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix} \hat{K}_s \begin{bmatrix} I & 0 & \cdots & 0 \end{bmatrix} \underline{\hat{F}}_f \tag{26}$$

3.3.2 Transfer Function for $\hat{\Psi}_2$

If the most recent value of y^k is used to update x_{s_i}, and x_{s_i} does not affect u_s^k until all $\hat{\mathcal{K}}_{s_i}$ have been updated, then $\underline{\hat{\Psi}}_2$ is defined by the interconnection shown in Fig.12. Therefore,

$$\underline{\hat{\Psi}}_2 = L\hat{K}_f L^{-1} + \hat{\mathbf{K}}_s L(S_f F H_f) L^{-1}$$

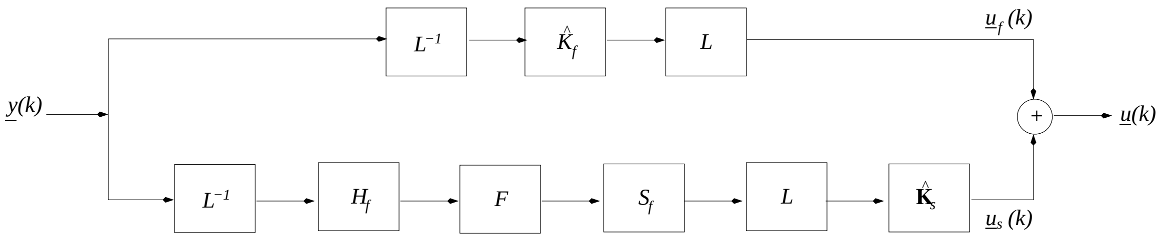


Figure 12: $\hat{\Psi}_2$ as a LTI system.

or,

$$\underline{\hat{\Psi}}_2 = \underline{\hat{K}}_f + \hat{\mathbf{K}}_s \underline{\hat{F}}_f \tag{27}$$

where $\hat{\mathbf{K}}_s$ is given by

$$\hat{\mathbf{K}}_s = \left[\begin{array}{cccc|cccc} \hat{\mathcal{A}}_{s_0} & 0 & \cdots & 0 & \hat{\mathcal{B}}_{s_0} & 0 & \cdots & 0 \\ 0 & \hat{\mathcal{A}}_{s_1} & \cdots & 0 & 0 & \hat{\mathcal{B}}_{s_1} & \cdots & \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \hat{\mathcal{A}}_{s_{M-1}} & 0 & 0 & \cdots & \hat{\mathcal{B}}_{s_{M-1}} \\ \hline \hat{\mathcal{C}}_{s_0} & \hat{\mathcal{C}}_{s_1} & \cdots & \hat{\mathcal{C}}_{s_{M-1}} & 0 & 0 & \cdots & 0 \\ \hat{\mathcal{C}}_{s_0} & \hat{\mathcal{C}}_{s_1} & \cdots & \hat{\mathcal{C}}_{s_{M-1}} & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \hat{\mathcal{C}}_{s_0} & \hat{\mathcal{C}}_{s_1} & \cdots & \hat{\mathcal{C}}_{s_{M-1}} & 0 & 0 & \cdots & 0 \end{array}\right] \tag{28}$$

3.3.3 Transfer Function for $\hat{\Psi}_3$

If the most recent value of y^k is used to update x_{s_i}, and x_{s_i} affects u_s^k immediately, then $\underline{\hat{\Psi}}_2$ is also defined by the interconnection shown in Fig.12, but with $\hat{\mathbf{K}}_s$ defined as

$$\hat{\mathbf{K}}_s = \left[\begin{array}{ccccc|ccccc} \hat{\mathcal{A}}_{s_0} & 0 & 0 & \cdots & 0 & \hat{\mathcal{B}}_{s_0} & 0 & 0 & \cdots & 0 \\ 0 & \hat{\mathcal{A}}_{s_1} & 0 & \cdots & 0 & 0 & \hat{\mathcal{B}}_{s_1} & 0 & \cdots & 0 \\ 0 & 0 & \hat{\mathcal{A}}_{s_2} & \cdots & 0 & 0 & 0 & \hat{\mathcal{B}}_{s_2} & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & \hat{\mathcal{A}}_{s_{M-1}} & 0 & 0 & 0 & \cdots & \hat{\mathcal{B}}_{s_{M-1}} \\ \hline \hat{\mathcal{C}}_{s_0} & \hat{\mathcal{C}}_{s_1} & \hat{\mathcal{C}}_{s_2} & \cdots & \hat{\mathcal{C}}_{s_{M-1}} & 0 & 0 & 0 & \cdots & 0 \\ \hat{\mathcal{C}}_{s_0}\hat{\mathcal{A}}_{s_0} & \hat{\mathcal{C}}_{s_1} & \hat{\mathcal{C}}_{s_2} & \cdots & \hat{\mathcal{C}}_{s_{M-1}} & \hat{\mathcal{C}}_{s_0}\hat{\mathcal{B}}_{s_0} & 0 & 0 & \cdots & 0 \\ \hat{\mathcal{C}}_{s_0}\hat{\mathcal{A}}_{s_0} & \hat{\mathcal{C}}_{s_1}\hat{\mathcal{A}}_{s_1} & \hat{\mathcal{C}}_{s_2} & \cdots & \hat{\mathcal{C}}_{s_{M-1}} & \hat{\mathcal{C}}_{s_0}\hat{\mathcal{B}}_{s_0} & \hat{\mathcal{C}}_{s_1}\hat{\mathcal{B}}_{s_1} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots \\ \hat{\mathcal{C}}_{s_0}\hat{\mathcal{A}}_{s_0} & \hat{\mathcal{C}}_{s_1}\hat{\mathcal{A}}_{s_1} & \hat{\mathcal{C}}_{s_2}\hat{\mathcal{A}}_{s_2} & \cdots & \hat{\mathcal{C}}_{s_{M-1}} & \hat{\mathcal{C}}_{s_0}\hat{\mathcal{B}}_{s_0} & \hat{\mathcal{C}}_{s_1}\hat{\mathcal{B}}_{s_1} & \hat{\mathcal{C}}_{s_2}\hat{\mathcal{B}}_{s_2} & \cdots & 0 \end{array}\right] \tag{29}$$

Therefore,

$$\underline{\hat{\Psi}}_3 = \underline{\hat{K}}_f + \hat{\mathbf{K}}_s \underline{\hat{F}}_f \tag{30}$$

3.3.4 Transfer Function for $\hat{\Psi}_4$

The definition of $\hat{\Psi}_4$ requires the most recent running average of the plant output over, $M = \pi(\hat{K}_s)$, time-steps. The computation of the running average is assumed to be a separate process, as is often the case, and hence its overhead is not accounted for in the controller implementation.

The sequence y_s^k as defined by eqn.(22) can be written as

$$y_s(k) = \left(\frac{y(k) + y(k-1) + \cdots + y(k-M+1)}{M}\right)$$

Therefore, in terms of z transforms,

$$\begin{aligned} Y_s(z) &= y_s(0) + y_s(1)/z + \cdots + y_s(k)/z^k + \cdots \\ &= \left(\tfrac{y(0)+0+\cdots+0}{M}\right) + \left(\tfrac{y(1)+y(0)+0+\cdots+0}{Mz}\right) + \left(\tfrac{y(2)+y(1)+y(0)+0+\cdots+0}{Mz^2}\right) + \cdots \\ &= \left(\tfrac{y(0)+y(1)/z+\cdots}{M}\right) + \left(\tfrac{y(0)+y(1)/z+\cdots}{Mz}\right) + \cdots + \left(\tfrac{y(0)+y(1)/z+\cdots}{Mz^{M-1}}\right) \\ &= \left(\tfrac{1+1/z+\cdots+1/z^{M-1}}{M}\right) Y(z) \end{aligned}$$

Therefore the running average can be computed by a digital filter defined by,

$$\hat{F}_{avg} = \frac{1}{M}\left(I_{n_y} + z^{-1}I_{n_y} + z^{-2}I_{n_y} + \cdots + z^{-(M-1)}I_{n_y}\right) \tag{31}$$

where I_{n_y} is a $n_y \times n_y$ identity matrix and n_y is the dimension of the signal $y(k)$. Figure 13 shows the frequency response of $\hat{F}_{avg}$ with sampling interval $h = 0.01$. The dashed line in Fig.13 is a first order LTI system with cut-off $2\pi/(Mh)$. Comparing the two plots, we see that $\hat{F}_{avg}$ acts

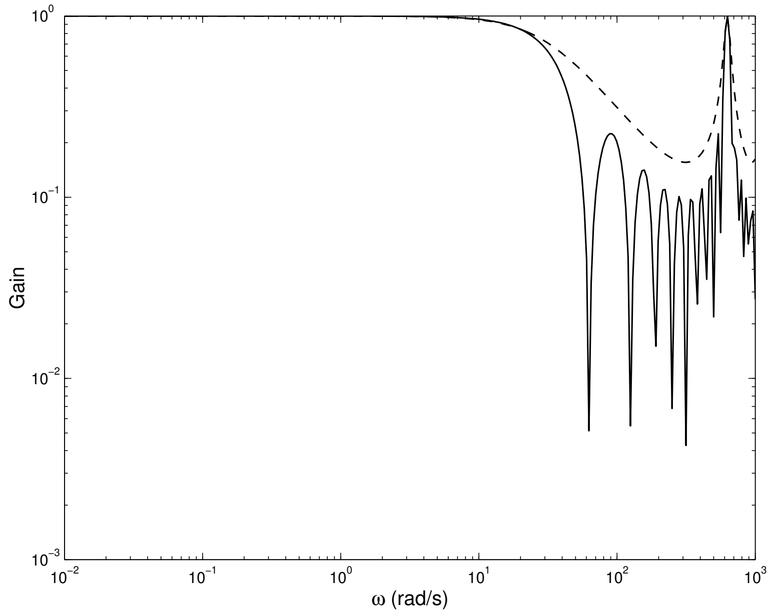


Figure 13: Frequency response of $\hat{F}_{avg}$(solid), first-order LTI system with cut-off $\frac{2\pi}{Mh}$(dashed).

similar to a lowpass filter with the desired cutoff of $2\pi/(Mh)$.

The dynamics of $\hat{\Psi}_4$ in the lifted input-output space is shown in Fig.14, and the transfer function is given by

$$\underline{\hat{\Psi}}_4 = \underline{\hat{K}}_f + \hat{\mathbf{K}}_s\underline{\hat{F}}_{avg} \tag{32}$$

where $\hat{\mathbf{K}}_s$ is given by eqn.(28) and $\underline{\hat{F}}_{avg}$ is given by,

$$\underline{\hat{F}}_{avg} = \begin{bmatrix} I_{n_y} & z^{-M}I_{n_y} & z^{-M}I_{n_y} & \cdots & z^{-M}I_{n_y} \\ I_{n_y} & I_{n_y} & z^{-M}I_{n_y} & \cdots & z^{-M}I_{n_y} \\ I_{n_y} & I_{n_y} & I_{n_y} & \cdots & z^{-M}I_{n_y} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ I_{n_y} & I_{n_y} & I_{n_y} & \cdots & I_{n_y} \end{bmatrix}$$

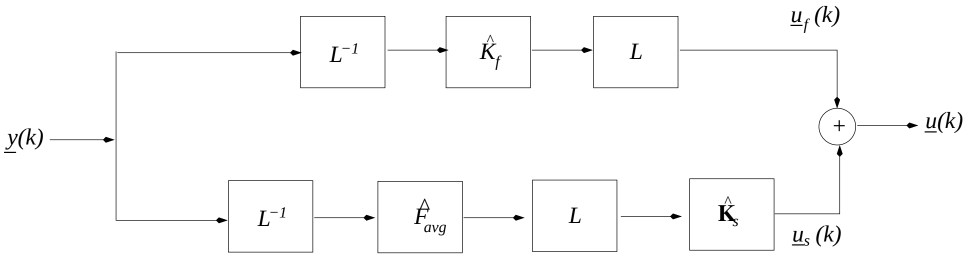


Figure 14: $\hat{\Psi}_4$ as a LTI system.

3.4 Reduction in Computational Overhead

The reduction in computational overhead achieved by the transformation $\mathcal{T}(K)$ can be determined as follows. Let us denote $\eta(K)$ as the number of states of a LTI system K and $T_{cpu}(n), n \in \mathcal{Z}^*$, as the computational time or overhead required to update n states of a discrete-time LTI system. Let us also assume that $T_{cpu}(n_1) \geq T_{cpu}(n_2) \geq 0$ if $n_1 \geq n_2 \geq 0$.

Therefore the number of states being updated at any time step k is

$$\eta(\hat{K}_f) + \eta(\hat{\mathcal{K}}_{s_i})$$

where $i = \lfloor k/\pi(\hat{K}_s) \rfloor$. The reduction in computational overhead at any time step k can therefore be defined as

$$\Delta T_{cpu}(k) = T_{cpu}(\eta(\hat{K})) - T_{cpu}(\eta(\hat{K}_f) + \eta(\hat{\mathcal{K}}_{s_i})); \quad i = \lfloor k/\pi(\hat{K}_s) \rfloor \tag{33}$$

where $T_{cpu}(\eta(\hat{K}))$ denotes the computational time required if all the states of the controller K are updated at every time step. From the definition of ΔT_{cpu} in eqn.(33), we see that it depends on the number of states of $\hat{\mathcal{K}}_{s_i}$. Therefore ΔT_{cpu} is a periodic function of time, the minimum value of which determines the savings in terms of CPU overhead. The lower bound on ΔT_{cpu} can be determined as follows.

If the maximum multiplicity of the eigen-values of $\hat{K}_s$ is p, then

$$\eta(\hat{\mathcal{K}}_{s_i}) \leq p, \ \forall i \in \{0, 1, \ldots, \pi(\hat{K}_s) - 1\}$$

therefore the minimum value of $\Delta T_{cpu}(k)$ is given by

$$\Delta T_{cpu,min} = \min_{k} \ \ \Delta T_{cpu}(k) = T_{cpu}(\eta(\hat{K})) - T_{cpu}(\eta(\hat{K}_f) + p)$$

Since $\Delta T_{cpu,min}$ satisfies

$$\Delta T_{cpu,min} > 0 \ ; \ \text{if } p < \eta(\hat{K}_s)$$

we can conclude that the transformation $\mathcal{T}(K)$ achieves a uniform reduction in the computational overhead by a nonzero amount quantified by $\Delta T_{cpu,min}$.

4 Example

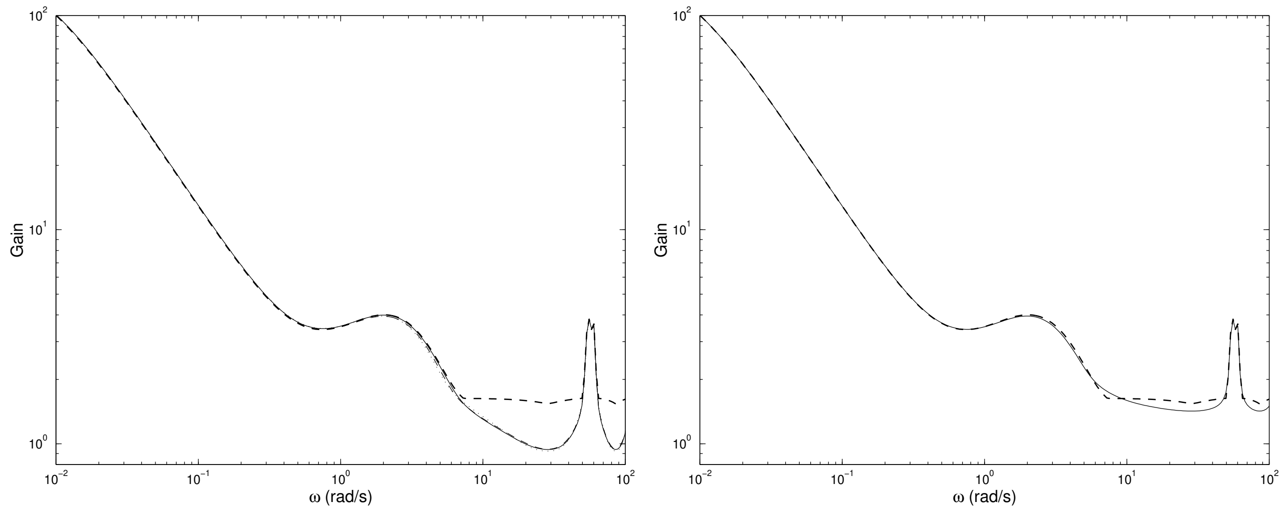
In this section we study the effect of the transformation $\mathcal{T}(K)$ on a controller designed for a B737-100 TSRV(Transport System Research Vehicle) linear longitudinal motion model. The aircraft model has four states: longitudinal velocity V, angle-of-attack α, pitch rate q and pitch angle θ; two control inputs: thrust T(lb) and elevator deflection δ_e. The elevator actuator and the engine are modelled as $16/(s+16)$ and $20/(s^2+12s+20)$ respectively. The control objective is to achieve decoupled response of V and flight-path-angle γ reference signals. The controller was designed using $\mathcal{H}_\infty$ theory and has 18 states. Details of the controller design can be obtained from reference [9].

The sampling rate of 100 Hz is sufficiently fast to implement this controller in a digital computer. From table(4), we see that the natural frequencies of the controller are vary from 4.4039×10^{-05} rad/s to 169.87 rad/s. Clearly, updating all the states of this system at 100 Hz will result in unnecessary computation.

No.	Natural Frequency	No.	Natural Frequency
1	169.87	10	2.0644
2	169.87	11	0.9937
3	80.226	12	0.8000
4	80.226	13	0.8000
5	16.790	14	0.1600
6	5.0055	15	0.1600
7	3.1566	16	7.7971×10^{-02}
8	3.1566	17	7.8772×10^{-03}
9	2.0644	18	4.4039×10^{-05}

Table 3: Natural frequencies of the controller designed in reference [9].

For this example, we decompose the controller K by assigning the fastest two modes to K_f and the rest to K_s. Modal decomposition of K_s will yield eleven modal sub-systems, therefore $M = \pi(K_s) = 11$. Note that the maximum multiplicity of the eigen-values is two, hence we can expect a substantial reduction in the computational overhead. From section 3.2, we can generate systems $\hat{\Psi}_1$, $\hat{\Psi}_2$, $\hat{\Psi}_3$ and $\hat{\Psi}_4$ for this system. The largest singular value plots of their corresponding transfer functions, in the lifted input-output space, are shown in Fig.15(a) and 15(b). The figures reveal that $\underline{\hat{\Psi}}_i$, $i = 1, 2, 3, 4$ are close to the original system K at low frequencies, in terms of the largest singular value plots. There is, however, distortion at higher frequencies. Systems $\underline{\hat{\Psi}}_i, i = 1, 2, 3$, cause almost identical distortions at these frequencies. It is interesting to note that $\underline{\hat{\Psi}}_4$ causes the least distortion.



(a) $\bar{\sigma}(\underline{\hat{\Psi}}_i)$, $i = 1, 2, 3$ (solid), $\bar{\sigma}(\underline{\hat{K}})$ (dashed)

(b) $\bar{\sigma}(\underline{\hat{\Psi}}_4)$ (solid), $\bar{\sigma}(\underline{\hat{K}})$ (dashed)

Figure 15: Maximum singular value plots of $\underline{\hat{\Psi}}_i$, $i = 1, 2, 3, 4$.

To the study the frequency domain effect of the transformation $\mathcal{T}(K)$ on the closed-loop system, we need to define the closed-loop system in the lifted input-output space. This is shown in Fig.16. Note that we have used discrete-time representation of the plant for performance analysis. The discrete-time plant, sampled at 100 Hz is sufficiently close to the continuous-time plant and hence can be used for performance analysis [10]. The closed-loop performance with

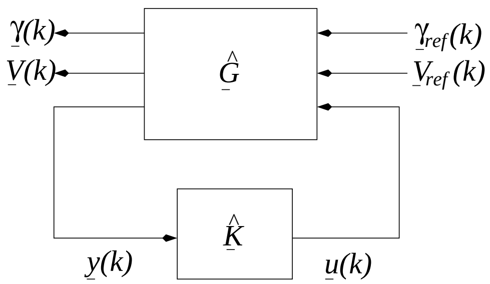


Figure 16: Closed-loop system in lifted I/O space.

controllers $\underline{\hat{\Psi}}_i$ are compared with that of the original controller $\underline{\hat{K}}$. For the purpose of discussion let us represent the closed-loop system with $\underline{\hat{K}}$ as $\underline{\hat{P}}_0$, and the closed loop system with $\underline{\hat{\Psi}}_i$ as $\underline{\hat{P}}_i$. Figure 17, plots the maximum singular values of the four closed-loop transfer functions

$\underline{\gamma}_{ref} \rightarrow \underline{\gamma}, \underline{\gamma}_{ref} \rightarrow \underline{V}, \underline{V}_{ref} \rightarrow \underline{\gamma}$ and $\underline{V}_{ref} \rightarrow \underline{V}$, for the four definitions of $\mathcal{T}(K)$. Note that we plot singular values because these transfer functions are multi-variable systems in the lifted input-output space.

From the plots in Fig.17, we see that systems $\underline{\hat{P}}_i$ do not differ from $\underline{\hat{P}}_0$ significantly, in terms of the largest singular values. At low frequencies, the tracking response of $\underline{\hat{P}}_i$ is identical to that of $\underline{\hat{P}}_0$, and the four transformation achieve satisfactory decoupling of γ and V response. There is however deviation at around 1 rad/s in the four transfer functions.

To investigate the robustness of the transformed controllers we analysed the nominal performance and robust stability of the four closed-loop systems. From our analysis we observed that none of four transformed controllers could achieve the desired robust performance. This is expected since we did not consider robustness when we decomposed K into K_f and K_s.

To study the effect of $\mathcal{T}(K)$ in time domain, the step response of the closed-loop system to velocity command are shown in Fig.18. Step responses of the systems $\hat{P}_i, i = 1, 2, 3, 4$ are quite close to that of $\hat{P}_0$, except $\hat{\Psi}_3$ generates oscillatory elevator output for a velocity step command, which is not desirable. We also observed, from plots not included in this paper, that the closed-loop respose to a step command in γ, for different $\hat{P}_i, i = 1, 2, 3, 4$, are also close to that of $\hat{P}_0$. Therefore, in time domain also, there is no significant change in the behaviour of the closed-loop due to the transformation.

Note that the oscillatory elevator output of $\hat{\Psi}_3$ is not observed in the singular plot. This is because, frequency domain analysis of discrete-time systems is restricted to the Nyquist frequency of the system. The influence of the updated states of $\hat{\mathcal{K}}_{s_i}$ on the output of $\hat{\Psi}_3$ occurs at a frequency higher than Nyquist frequency of the lifted system and hence it is not observable from the singular value plot of $\hat{\Psi}_3$.

Since the averaging filter in Ψ_4 is a FIR filter, we were interested in analysing the closed-loop response to a velocity step command in the presence of high frequency disturbances, such as a gust. The gust model used is the NASA Dryden Gust Model. Details of the gust model and the implementation is available in [9]. From the response in Fig.19, we see that the time response of $\underline{\hat{P}}_0$ and $\underline{\hat{P}}_4$ are not significantly different. Therefore, we can conclude that the FIR filter at the input of K_s doesn't degrade the high-frequency attenuation properties of the closed-loop system.

From the plots of the largest singular values of $\underline{\hat{P}}_i$ and step-responses, we observe that the transformation $\mathcal{T}(K)$ causes degradation in the respose of the closed-loop system to the reference commands. The degradation in tracking response, however, is not significant for this example.

The reduction in the computational overhead to implement the transformed controller is quite substantial. In a conventional implementation, where all the states of the controller are updated at every time step and assuming the A matrix of the controller is dense, the number of FLOPS (floating point operations) required for this example is 1000 MATLAB flops. If the controller is transformed into modal form, the A matrix of the controller is block-diagonal and the flop count in that case is 720. With the transformed controller $\mathcal{T}(K)$, a maximum of only 132 MATLAB flops are required. Therefore, for this example, it is possible to reduce the required computational overhead by 81.66% if A is block-diagonal and 86.8% if A is dense. This reduction in computational overhead is quite significant.

5 Conclusion

In this paper we have presented an algorithm for computationally efficient digital implementation of linear time invariant controllers. A theoretical framework, built on the multi-rate filter bank theory and lifting techniques, was developed to analyse the effect of the transformation on the closed-loop system performance and stability. We applied this idea on a flight control problem based on the B737-100 TSRV linear longitudinal motion model. From time and frequency domain analysis, we could conclude that, for this example, the transformation did not alter the behaviour of the closed-loop system significantly. However, the reduction in computational overhead is quite significant.

From the point of view of computation, the transformation $\mathcal{T}(K)$ can be considered to be a model reduction technique for LTI systems. Model reduction is achieved by partitioning the state space of the LTI system into two subspaces, corresponding to the slow and fast modes of the system. The states corresponding to the fast modes are updated as fast as the base clock. The states corresponding to the slow modes are updated one after another in a round-robin manner. Thus the system composed of the slower modes operates slower than the base clock. This results in a periodically time varying system. Since the number of states required to be updated at a given time instant reduces, the order of the system from the point of view of computation also reduces. Note that instead of partitioning the state space of the LTI system into just two subspaces, we could extend this idea to N partitions in general. In such a scenario, distribution of the computation required to updated the states of those N subspaces will be more complicated.

The theoretical framework developed in this paper, to analyse the effect of the round-robin state updation policy on the closed-loop system, can also be used to analyse control systems in an environment of priority based scheduling of computational resources. In such an environment, the computational tasks are alloted CPU resources based on their assigned priorities, which could be static or dynamic. Consequently, the order in which these tasks are executed depends on the assigned priorities. Therefore, if a control system is a composite of several computational tasks, the effect of the sequence of execution of these tasks, on the dynamical behaviour of the controller, can be analysed under the framework presented in this paper.

In the context of software driven distributed control systems, operating in real or simulated-time, few researchers have studied the effect of the sequence of execution on the dynamics of the overall system. Analysis of the effect of the model of computation, used to realise the control algorithm, is crucial, especially since the paradigm of control system design and implementation is shifting from a centralised, single-processor framework to a decentralised, distributed computing framework. The analysis framework developed in this paper has potential to contribute towards the development of a systematic approach to analyse these issues.

Acknowledgements

We would like to thank Prof. Bruce Francis and Prof. Tryphon Georgiou for sharing with us their insight in sampled data control systems and multi-rate systems.

This work was funded by DARPA under the Software Enabled Control program with Dr. John Bay as the program manager. The contract number is USAF/AFMC F33615-99-C-1497 and Dale Van Cleave is the technical contract monitor.

References

[1] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of Association for Computing Machinery*, 20(1):46–61, Jan 1973.

[2] K. Ramamritham and J. A. Stankovic. Scheduling algorithms and operating systems support for real-time systems. *Proceedings of IEEE*, 82(1):55–67, January 1994.

[3] P. P. Vaidyanathan. *Multirate Systems and Filter Banks*. Prentice Hall, Inc., Englewood Cliffs, N.J. 07632, 1993.

[4] T. Chen and B. Francis. *Optimal Sampled-Data Control Systems*. Springer-Verlag New York, Incorporated, 175 Fifth Avenue New York, NY 10010, 1995.

[5] K. Poolla P. P. Khargonekar and A. Tannenbaum. Robust control of linear time-invariant plant using periodic compensation. *IEEE Transactions on Automatic Control*, 30(11):1088–1096, November 1983.

[6] M. Araki and K. Yamamoto. Multivariable multirate sampled-data systems: State-space description, transfer characteristics, and nyquist criterion. *IEEE Transactions on Automatic Control*, 31(2):145–154, February 1986.

[7] P. P. Vaidyanathan and S. K. Mitra. Polyphase networks, block digital filtering, lptv systems and alias-free qmf banks: A unified approach based on pseudo-circulants. *IEEE Transactions on Acoustic, Speech, Signal Processing*, 36(3):381–391, March 1988.

[8] R. A. Meyer and C. S. Burrus. A unified analysis of multirate and periodically time-varying digital filters. *IEEE Transactions on Circuits and Systems*, 22(3):162–168, March 1975.

[9] S. Ganguli and G. Balas. A tecs alternative using robust multivariable control. *AIAA Guidance, Navigation and Control Conference*, August 2001.

[10] J. P. Keller and B. D. O. Anderson. A new approach to the discretization of continuous-time controllers. *IEEE Transactions in Automatic Control*, 37(2):214–223, February 1992.

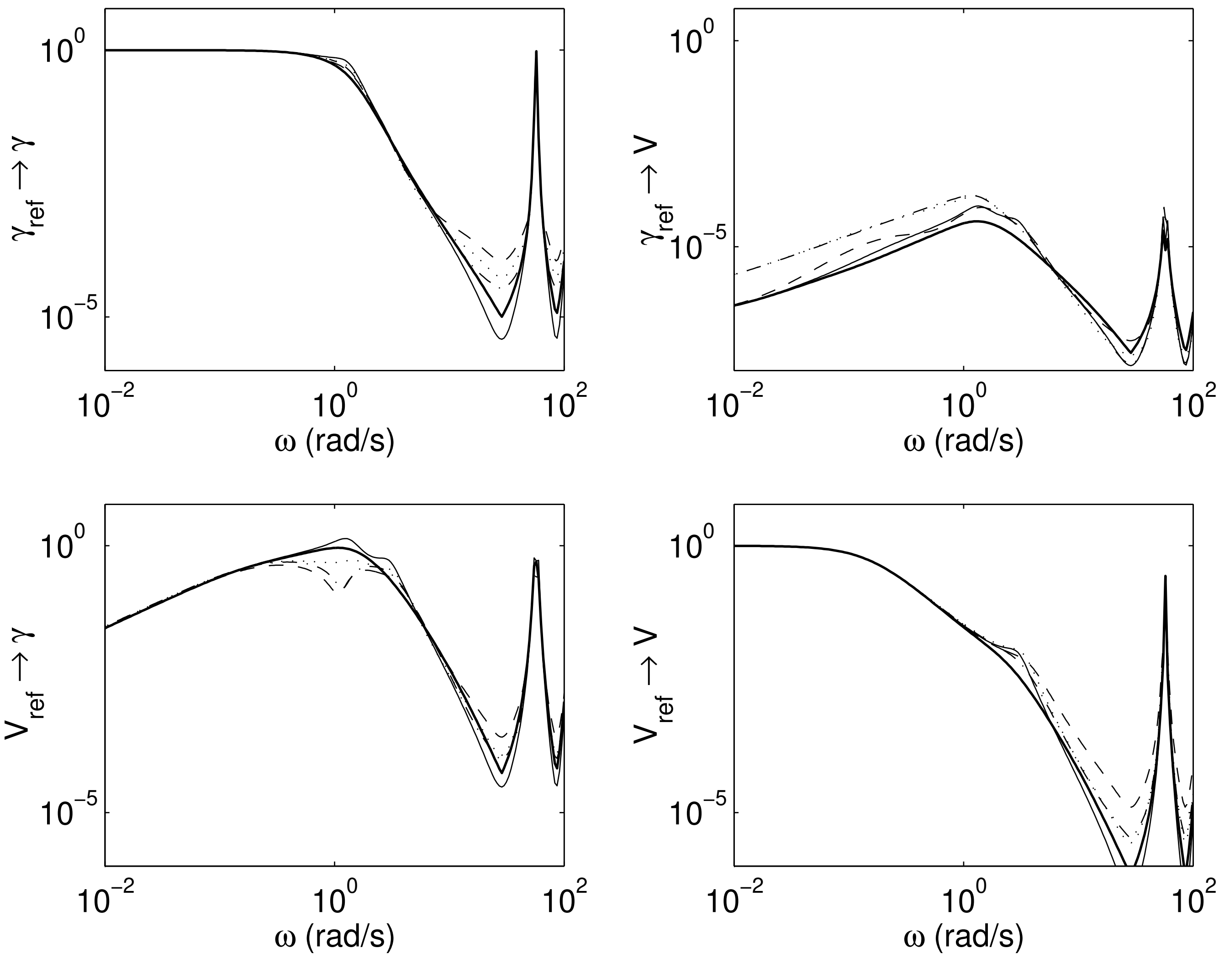


Figure 17: Frequency response of closed loop system with $\underline{\hat{\Psi}}_i$, $i = 0$ (thick solid), 1 (solid), 2 (dot), 3 (dash-dot), 4 (dash-dash)

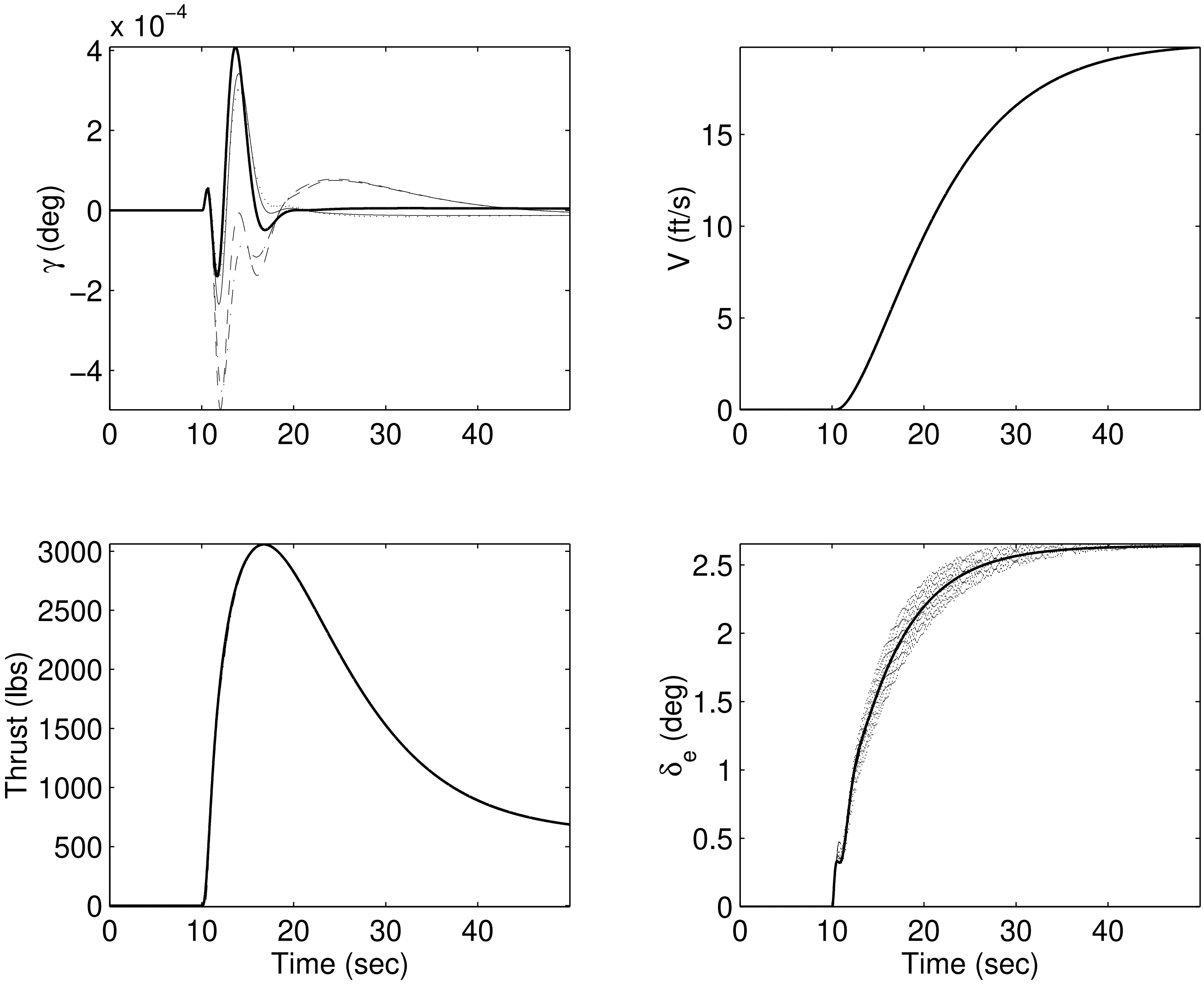


Figure 18: Response of $\hat{P}_i$, $i = 0$ (thick solid), 1 (solid), 2 (dash-dot), 3 (dot), 4 (dash-dash) to velocity step command.

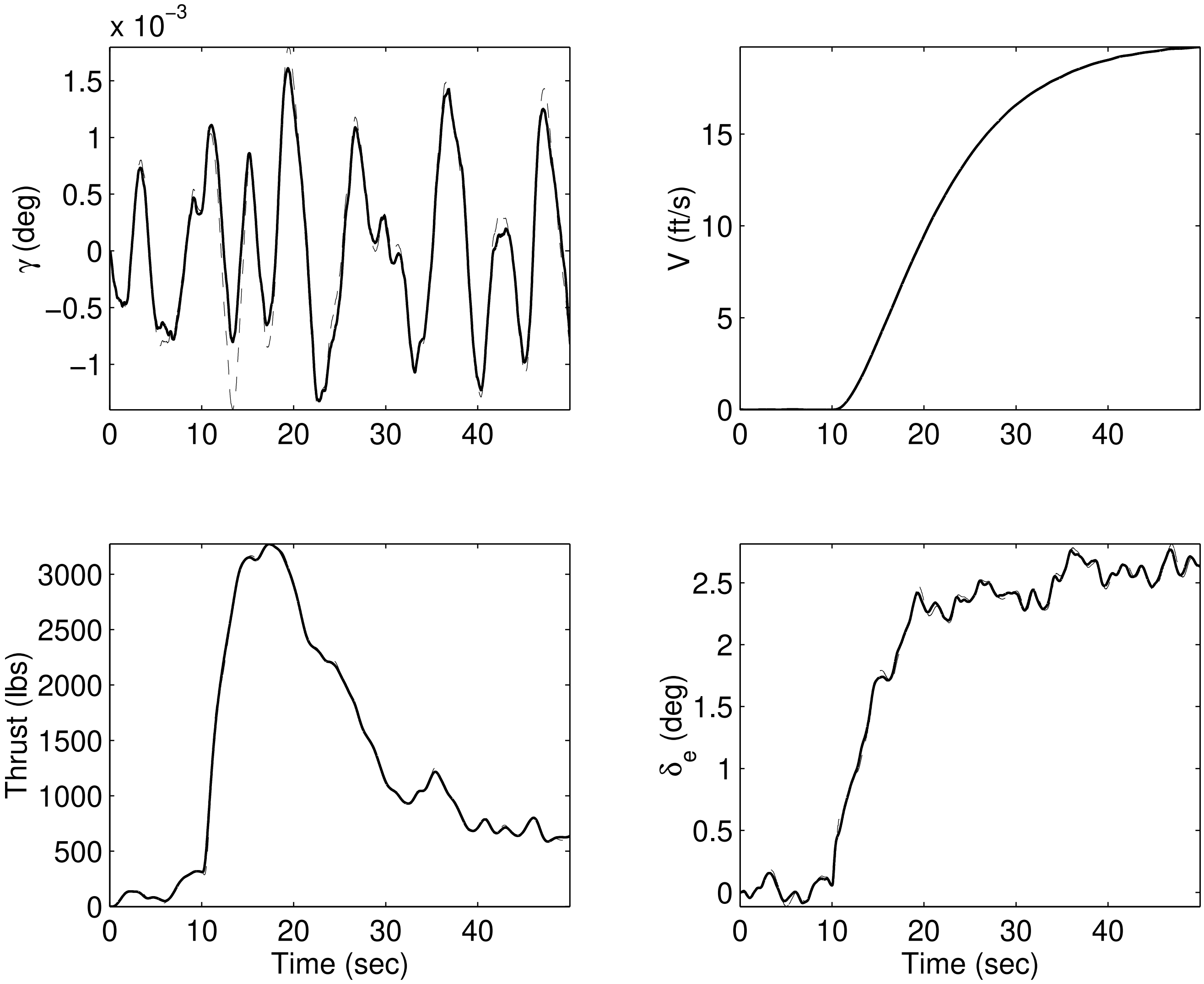


Figure 19: Response of $\hat{P}_i$, $i = 0$ (thick solid),4 (solid) to velocity step command with gust.