for k—a sufficiently large negative integer. Thus, by choosing the negative integer k that is arbitrarily large, we see from (51) and (52) that $h > 0$ can be made sufficiently small and $R > 0$ can be made arbitrarily large, which proves the theorem.

Remark 3: If K in (42) is less than one, Theorem 2 does not hold since if $\operatorname{Re}(|\lambda|) \to \infty$ and $h > 0$, (43) is asymptotic to

$$1 < \frac{1}{K} = |e^{-\lambda h}| \tag{54}$$

which implies $\operatorname{Re}(\lambda) < 0$.

References

[1] G. Chen, S. G. Krantz, D. W. Ma, and C. E. Wayne, "The Euler–Bernoulli beam equation with boundary energy dissipation," in *Operator Methods for Optimal Control Problems*, Sung J. Lee, Ed. New York: Macel Dekker, 1987.

[2] G. Chen and J. Zhou, "The wave propagation method for the analysis of boundary stabilization in vibrating structures," *SIAM J. Appl. Math.*, vol. 50, pp. 1254–1283, 1990.

[3] R. Datko, J. Lagnese, and M. P. Polis, "An example of the effect of time delays in boundary feedback stabilization of wave equations," *SIAM J. Contr. Optimiz.*, vol. 24, pp. 152–156, 1986.

[4] R. Datko, "Not all feedback stabilized hyperbolic systems are robust with respect to small time delays in their feedbacks," *SIAM J. Contr. Optimiz.*, vol. 26, pp. 697–713, 1988.

[5] ——, "Two questions concerning the boundary control of certain elastic systems," *J. Diff. Eq.*, vol. 92, pp. 27–44, 1991.

[6] F. L. Huang, "Characteristic conditions for exponential stability of linear dynamical systems in Hilbert spaces," *Ann. Diff. Eqs.*, vol. 1, pp. 43–53, 1985.

[7] J.-L. Lions, "Exact controllability, stabilization and perturbations for distributed parameter systems," *SIAM Rev.*, vol. 30, pp. 1–68, 1988.

On a Conjecture About Assigning Jobs to Processors of Differing Speeds

Richard Weber

***Abstract*—A difficult queueing control problem concerns jobs that arrive to a buffer in a Poisson process and are to be assigned to m processors of different speeds. These processors operate in parallel, and processing times are independent and exponentially distributed. Once a job is assigned to a free processor, it occupies that processor until completed and may not be reassigned to a faster processor if one becomes free. A reasonable conjecture, that remains unproven for more than two processors, is that the policy that minimizes the mean waiting time is of threshold type—meaning that if it assigns a job to the fastest-available free processor when the buffer has k jobs, then it also does so if the processors are identically occupied and there are $k + 1$ jobs in the buffer. This note discusses the conjecture, and shows that whether or not a job should be assigned to a processor can depend not only on the number in the queue and the speed of the fastest-available processor. but also on whether slower processors are busy or idle. A strengthened form of the conjecture is proposed.**

I. Introduction

A well-known, and still unsolved, queueing control problem concerns the optimal assignment of job to processors of differing speeds. Identical jobs enter a single buffer according to a Poisson process of rate λ. There are m processors of different speeds that are available to process the jobs. These operate in parallel, and all processing times are independent and exponentially distributed. Processor i is the ith fastest, it processes jobs in times that are exponentially distributed with parameter μ_i, where $\mu_1 \geq \cdots \geq \mu_m$. The objective is to minimize the steady-state mean time in the system, or mean delay of a job. Once a job is assigned to a given free processor, it occupies that processor until its processing is complete and may not be reassigned to a faster processor if one later becomes free. This means that when there are a small number of jobs in the buffer, it can be sensible to leave processors idle if they are of relatively slow speed, and make no assignment until a faster processor becomes available or further jobs arrive. Of course if an assignment is to be made, then it should be made to the fastest available processor. This fact is proved in Section II. A much more difficult, but also reasonable, conjecture is that the optimal assignment rule is of threshold type. The idea is formalized as Conjecture 1.

Conjecture 1: An optimal policy has the property that whenever it assigns a job to an available processor it makes the assignment to the fastest available processor. Moreover, there exist thresholds, one for each state of the processors, such that an optimal policy is to assign a job to an available processor if, and only if, the number of jobs in the buffer exceeds the relevant threshold for the present state of the processors.

Agrawala *et al.* [1] showed that Conjecture 1 is true when the arrival rate is 0. The problem is to minimize the expected total waiting time of n jobs that are initially in the buffer. In this case, it is optimal to assign a job to processor i if, and only if, all faster processors are busy and the number of jobs in the buffer k satisfies

$$\frac{k + i - 1}{\mu_1 + \cdots + \mu_{i-1}} > \frac{1}{\mu_i}. \tag{1}$$

Notice that the decision can be made independently of the states of the slower processors. Perhaps this is not surprising, since once processor i is allowed to become idle, it will not be used at any later time. Interestingly, the optimal policy can also be implemented by ordering the jobs in the buffer and offering the fastest-available processor to each job in turn. Suppose processor i is the fastest available, and the job who is at the front of the buffer chooses to be processed by processor i if, and only if, $i/(\mu_1 + \cdots + \mu_{i-1})$ exceeds $1/\mu_i$. If it is declined, then processor i is offered to the second job in the buffer, who chooses it only if $(i + 1)/(\mu_1 + \cdots + \mu_{i-1})$ exceeds $1/\mu_i$. This continues until processor i has been chosen by some job, or declined by every job. This implements the policy described by [1]. However, Kumar and Walrand [3] have also shown that these decisions are optimal for each job individually: in that, each job minimizes its own expected waiting time, subject to the fact that jobs in front of it are given first refusal of any available processor. This contrasts with the fact that in most queueing control problems, socially and individually optimal policies do not coincide.

The problem with arrivals is much more difficult. Lin and Kumar [4] have proved Conjecture 1 for two processors. Walrand [10] has also given a proof based on stochastic coupling. However, the case of two processors is very special, and the generalization to more than two processors seems very difficult. Luh and Viniotis [5] have studied the problem using a novel approach. The approach of forward induction, so commonly used for this sort of problem, is not helpful and runs into difficulty with just three processors.

Manuscript received May 17, 1991; revised September 25, 1991. Paper recommended by Associate Editor, A. Shwartz.

The author is with the University Engineering Department, Management Studies Group, University of Cambridge, Cambridge CB2 1RX, United Kingdom.

IEEE Log Number 9203142.

Since the problem with arrivals is so difficult, researchers have looked at other variations of the problem without arrivals. Righter [6] and Righter and Xu [7] have considered cost functions such as expected weighted flow time, weighted discounted flow time, and weighted number of tardy jobs. They have also [8] considered a problem in which the distribution of processing times at processor i has a nondecreasing hazard rate in the range $[\underline{h}_i, \overline{h}_i]$, where $\overline{h}_{i+1} \leq \underline{h}$. The objective is to minimize $E[\Sigma c_i g(C_i)]$ for some increasing concave g. The main conclusion is that if the jobs are ordered in the queue by their weights, then the individually optimal policy is also globally optimal. This establishes a conjecture of Kumar and Walrand.

Mirchandani and Xu have considered a problem in which jobs belong to a number of priority classes and are to be processed by two processors. Jobs within class C_j have processing times that are exponentially distributed with rates $\mu_{1j} \geq \mu_{2j}$ for processors 1 and 2. They showed that if an optimal threshold policy is applied to every higher priority class, then the expected flow time of class C_i jobs is also minimized by a threshold policy. Xu *et al.* [11] have considered the same problem, but under the more general assumption that processing times have distributions with decreasing expected remaining processing times. The problem of minimizing expected makespan when there are no arrivals is also difficult. It is reasonable that Conjecture 1 holds, but this has been shown for just two and three processors [2].

Conjecture 1 remains open for the originally stated problem with arrivals, and the purpose of this note is to cast some light on why it may be difficult to resolve. In Section III, we show that, unlike the cases in which there are no arrivals or just two processors, the optimal assignment rule can depend on the states of the processors that are slower than the fastest-available processor. In Section IV, we suggest an explanation and conclude with a strengthened form of Conjecture 1 that might be easier to prove or disprove.

II. Optimality of Assignment to the Fastest Available Processor

There is, at least, one thing that can be proved for the model with arrivals. It is the first part of Conjecture 1, as reasserted in the following theorem.

Theorem 1: For heterogeneous processors, jobs arriving in a Poisson process and an objective of minimizing mean waiting time, an optimal policy has the property that whenever it assigns a job to an available processor it makes the assignment to the fastest-available processor.

Proof: The proof is by a simple coupling argument that demonstrates that, if at some decision point, a policy assigns a job to a processor that is not the fastest available, then there is another policy that assigns the job to a faster processor, with the result that, at all times, the number of jobs in the system is stochastically no greater. The argument can be expressed in a number of ways. Perhaps the simplest way is to consider a uniformization of the system, in which we assume $\lambda + \Sigma_i \mu_i = 1$, and so in which observation times occur as a Poisson process of rate 1. At each observation time, exactly one of $m + 1$ possible events occurs: namely, a job arrival or a potential job completion at one of the processors. A potential job completion is an actual job completion if the processor at which the potential completion occurs is busy.

By induction on n, we show that the number of jobs in the system at the end of a time horizon that concludes just after the nth observation time is stochastically minimized by restricting attention to policies that only assign jobs to the fastest of any available processors. The statement is clearly true for $n = 1$. If it is true for all n, an application of Little's formula completes the proof. So, suppose as an inductive hypothesis that the statement is true for a horizon of $n - 1$, and consider the problem of stochastically minimizing the number of jobs that are in the system just after n observation times. Let π be a policy that, at the start, assigns a job to processor j and leaves a faster processor i idle. Let π' be an identical policy to π, except that it assigns a job to i but not to j. By the following comparisons we see that π' is better than π. Note first that if the event that occurs at the first observation time is an arrival or a potential service completion at a processor other than i or j, then the inductive hypothesis for $n - 1$ implies that the system is left in a better state under π' than under π. If on the other hand, as happens with probability μ_j, the event is a service completion at processor j, then π leads to a state in which a job has completed on processor j, while under π' a job remains on processor i. In this case, π has turned out to be better than π'. However, this advantage for π is more than offset by the fact that with greater probability μ_i, π' leads to a state in which a job has completed on processor i, while under π a job remains on processor i. In this case, π' has turned out to be better. Note that the inductive hypothesis for $n - 1$ implies that, other things being equal, it is better to have a job remaining on processor i than on processor j. For both processors to be idle and the job to be completed is clearly even better. These facts and the inductive hypothesis imply that given that one of the above-two events occurs, the number of jobs in the system after n observation times is stochastically no greater under π' than under π. □

III. Dependence on the State of Slower Processors

Let (k, α) denote a state in which there are k jobs in the buffer. Here, α is a binary number of m bits, whose ith most significant bit is 1 or 0 as processor i is, respectively, busy or idle. Because of the memoryless nature of the exponential processing times and the Poisson arrival process, it is sufficient to restrict assignments of jobs to processors to *decision times* immediately after job arrivals or service completions. Call a state *stable* or *unstable* as it is, respectively, optimal or not optimal, to make no further assignments before proceeding to the next decision time. Of course, states for which $\alpha_1 = 0$ and $k \geq 1$ are always unstable.

We present two numerical examples. Both have the feature that $(k, 100)$ is unstable if, and only if, $k \geq 2$, whereas $(k, 101)$ is unstable if, and only if, $k \geq 1$. The point of this is to show that whether or not a job should be assigned to processor 2 can depend not only on the number of jobs in the buffer, but also on the state of processor 3. In these examples, it is optimal to make an assignment in state (1, 101), but not in state (1, 100).

Suppose, without loss of generality, that $\lambda + \mu_1 + \mu_2 + \mu_3 = 1$. Using the technique of uniformization, we can restrict attention to times at which there is a potential change of state. These *observation times* include the decision times above, and occur as a Poisson process of rate 1. At each such time, exactly one of four possible events can take place: an arrival, or a potential service completion at processor 1, 2, or 3. By a potential service completion at processor i, we mean that a job completes processing at that processor if the processor is busy. The probabilities with which these events occur are λ, μ_1, μ_2, and μ_3, respectively.

Consider a finite horizon problem, in which the objective is to minimize the expected holding cost accrued by the nth observation time. Clearly, this is the same problem as minimizing the

TABLE I
FINITE HORIZON EXAMPLE

k, α	$V_1 = \bar{V}_1$	$\bar{V}_2$	V_2	$\bar{V}_3$	V_3	$\bar{V}_4$	V_4
0, 000	0	0.270	0.270	0.6737	0.6737	1.1697	1.1697
0, 100	1	1.765	1.765	2.5109	2.5109	3.2860	3.2860
0, 010	1	2.125	2.125	3.2597	3.2597	4.3855	4.3855
0, 001	1	2.190	2.190	3.4400	3.4400	4.7148	4.7148
1, 100	2	3.765	3.620	5.2016	5.1145	$\underline{6.5334}$	$\underline{6.5334}$
0, 110	2	3.620	3.620	5.1145	5.1145	$\underline{6.5340}$	$\underline{6.5340}$
0, 101	2	3.685	3.685	5.2773	5.2773	6.8370	6.8370
0, 011	2	4.045	4.045	6.0261	6.0261	7.9364	7.9364
2, 100	3	5.765	5.540	8.1104	7.8320	10.2124	9.8184
1, 110	3	5.620	5.540	7.8320	7.8320	9.8184	9.8184
1, 101	3	5.685	5.540	7.9896	7.9025	$\underline{10.0874}$	$\underline{10.0849}$
0, 111	3	5.540	5.540	7.9024	7.9025	$\underline{10.0849}$	$\underline{10.0849}$
3, 100	4	7.765	7.540	11.0700	10.6200		
2, 110	4	7.620	7.540	10.7800	10.6200		
2, 101	4	7.685	7.540	10.9100	10.6200		
1, 111	4	7.540	7.540	10.6200	10.6200		
4, 100	5	9.765	9.540				
3, 110	5	9.620	9.540				
3, 101	5	9.685	9.540				
2, 111	5	9.540	9.540				
5, 100	6						
4, 110	6						
4, 101	6						
3, 111	6						

expected total delay accrued by the nth decision time. For a starting state (k, α), denote this cost as $V_n(k, \alpha)$. Let $|\alpha|$ be the number of bits of α that are set, i.e., the number of busy processors. Let $A_j\alpha$ be the number obtained from α by setting to 1 those j most significant bits of α that are equal to 0. Thus, $(k, \alpha) \to (k - j, A_j\alpha)$ corresponds to assigning j jobs from the buffer to the j fastest available processors. Similarly, let $D_i\alpha$ be obtained from α by setting $\alpha_i = 0$; so $(k, \alpha) \to (k, D_i\alpha)$ corresponds to a potential service completion at processor i.

The dynamic programming equations can be written as

$$V_{n+1}(k, \alpha) = \min_{j \le k \wedge (m - |\alpha|)} \left\{ \bar{V}_{n+1}(k - j, A_j\alpha) \right\} \tag{2}$$

where

$$\bar{V}_{n+1}(k, \alpha) = k + |\alpha| + \lambda V_n(k+1, \alpha) + \sum_i \mu_i V_n(k, D_i\alpha) \tag{3}$$

and $V_0(k, \alpha) = 0$. For example

$$V_{n+1}(2, 100) = \min \{\bar{V}_{n+1}(2, 100), \bar{V}_{n+1}(1, 110), \bar{V}_{n+1}(0, 111)\}$$

where

$$\begin{aligned}\bar{V}_{n+1}(2, 100) &= 3 + \lambda V_n(3, 100) + \mu_1 V_n(2, 000) \\ &\quad + \mu_2 V_n(2, 100) + \mu_3 V_n(2, 100) \\ \bar{V}_{n+1}(1, 110) &= 3 + \lambda V_n(2, 110) + \mu_1 V_n(1, 010) \\ &\quad + \mu_2 V_n(1, 100) + \mu_3 V_n(1, 110) \\ \bar{V}_{n+1}(0, 111) &= 3 + \lambda V_n(1, 111) + \mu_1 V_n(0, 011) \\ &\quad + \mu_2 V_n(0, 101) + \mu_3 V_n(0, 110).\end{aligned}$$

Here, $\bar{V}_{n+1}(k, \alpha)$ denotes the minimal cost over $n + 1$ further observation periods, given that no further jobs are assigned from the buffer until the first observation time. For the data

$$(\lambda, \mu_1, \mu_2, \mu_3) = (0.270, 0.505, 0.145, 0.080)$$

we can easily calculate the values in Table I.

These numbers were computed using a spreadsheet, and are shown to four decimal places of accuracy in this table. We have also computed these numbers using exact arithmetic; every number is a terminating decimal of no more than nine decimal places, so Table I does indeed reveal the form of the optimal policy. The underlined numbers for states (1, 100) and (1, 101) illustrate that for a time horizon of four, whether or not a job should be assigned to processor 2 depends upon whether processor 3 is busy or idle.

Is this behavior because of the small finite horizon scenario? The answer is no. A second example shows the same behavior for the criteria of minimizing the mean number in the system. The time-average optimality equation can be written as

$$f(k, \alpha) = \min_{j \le k \wedge (m - |\alpha|)} \left\{ \bar{f}(k - j, A_j\alpha) \right\} \tag{4}$$

where

$$\bar{f}(k, \alpha) = -\beta + k + |\alpha| + \lambda f(k+1, \alpha) + \sum_i \mu_i f(k, D_i\alpha). \tag{5}$$

Here β is the average number of jobs in the system under an optimal policy; and the f's are the *relative costs* (see, for example, [9]). We have solved these for the following data.

$$(\lambda, \mu_1, \mu_2, \mu_3) = (0.445, 0.376, 0.109, 0.070).$$

In Table II, the unstable states have been underlined. The entries in the last four rows apply for $k \ge 4$.

The numbers in Table II were computed by hypothesizing that the optimal policy is one whose unstable states are those underlined in the table. The relevant linear equations were solved using PC-MATLAB, and the solution then verified to satisfy the optimality equations (4) and (5). Note that the equations lead to a singular system, and one must choose the value of one of the relative costs before obtaining a solution. This always happens with time-average cost equations, since only the differences

TABLE II
INFINITE HORIZON EXAMPLE

β	5.07192	
k, α	$\bar{f}$	f
0, 000	0	0
0, 100	11.398	11.398
0, 010	16.914	16.914
0, 001	21.929	21.929
1, 100	30.178	30.178
0, 110	30.207	30.207
0, 101	34.529	34.529
0, 011	40.539	40.539
2, 100	55.960	52.950
1, 110	52.950	52.950
1, 101	55.771	55.717
0, 111	55.717	55.717
3, 100	86.330	82.400
2, 110	83.120	82.400
2, 101	84.268	82.400
1, 111	82.400	82.400
k, 100	$4.5455k^2 + 5.9321k + 28.5335$	$4.5455k^2 + 4.3049k + 28.5766$
$k - 1$, 110	$4.5455k^2 + 4.9412k + 28.5597$	$4.5455k^2 + 4.3049k + 28.5766$
$k - 1$, 101	$4.5455k^2 + 5.2958k + 28.5503$	$4.5455k^2 + 4.3049k + 28.5766$
$k - 2$, 111	$4.5455k^2 + 4.3049k + 28.5766$	$4.5455k^2 + 4.3049k + 28.5766$

between relative costs are germane. The value chosen can be arbitrary: we took $f(0, 000) = 0$. Terms like $f(k - 2, 111)$ were computed by considering an $M/M/1$ queue, with arrival rate λ and service rate $\mu = \mu_1 + \mu_2 + \mu_3$. The mean length of a busy period is $B = 1(\mu - \lambda)$ and the total holding cost incurred during a busy period is

$$H = \frac{1}{\mu}(1 + \lambda B) + \frac{\lambda}{\mu}H.$$

To see this, imagine the queue has a preemptive-resume service discipline. The first term on the right-hand side is the expected time in the system for the job that initiates the busy period, and the second term is the holding cost due to other sorts of busy periods that start when that job is the only one in the system and its service is preempted by the arrival of another job before it is complete. The expected number of such preemptions is λ/μ. Then

$$H = \frac{\lambda}{\mu - \lambda}\left[\frac{1}{\lambda} + \frac{1}{\mu - \lambda}\right].$$

Assuming that it is optimal to keep all processors busy whenever the number of jobs in the system is four or more, then for $k \geq 4$

$$\begin{aligned} f(k - 2, 111) &= -\beta B + kB + H + f(k - 3, 111) \\ &= (k - 3)[(k + 4)B/2 - \beta B + H] + f(1, 111). \end{aligned}$$

Also, for $k \geq 4$

$$\begin{aligned} \bar{f}(k, 100) &= -\beta + k + 1 + \lambda f(k - 1, 111) \\ &\quad + \mu_1 f(k - 3, 111) + (\mu_2 + \mu_3) f(k - 2, 111) \\ \bar{f}(k - 1, 110) &= -\beta + k + 1 + \lambda f(k - 1, 111) \\ &\quad + (\mu_1 + \mu_2) f(k - 3, 111) + \mu_3 f(k - 2, 111) \\ \bar{f}(k - 1, 101) &= -\beta + k + 1 + \lambda f(k - 1, 111) \\ &\quad + (\mu_1 + \mu_3) f(k - 3, 111) + \mu_2 f(k - 2, 111) \\ \bar{f}(k - 2, 111) &= -\beta + k + 1 + \lambda f(k - 1, 111) \\ &\quad + (\mu_1 + \mu_2 + \mu_3) f(k - 3, 111) \end{aligned}$$

IV. DISCUSSION

The examples of the previous section are by no means easy to discover. For most parameter values, the optimal policy does not depend on the states of slower processors. However, it is interesting to consider why it can sometimes occur. It is intuitive that in the problem setting with arrivals, the optimal policy assigns a job to the fastest free processor in states for which this would not be optimal if there were no arrivals. This is because there is a need to "prepare" for possible arrivals, which, if they occur, will wish to see a less congested system. Consider the states $(1, 100)$ and $(0, 110)$. Suppose the realization of the process, say ω, is such that there is one potential service completion at the second processor, followed by a large number of successive arrivals, say k in a row. Because for large k it will be optimal to have all processors busy, these states become $(k - 1, 111)$ and $(k - 2, 111)$, respectively. So $f(k - 1, 111) - f(k - 2, 111)$ measures the advantage that will be obtained in the case that ω occurs by having chosen to make the assignment to the second processor of $(1, 100) \rightarrow (0, 110)$. Intuitively, it is events like ω that provide the incentive to make an assignment to the second processor. But, if the two initial states are $(1, 101)$ and $(0, 111)$, the measure of advantage if ω occurs is $f(k, 111) - f(k - 1, 111)$. In the example given, and more generally, we expect $f(k, 111)$, and also $V_n(k, 111)$ to be convex in k. Thus, it is plausible that the incentive to make an assignment to the second processor is greater in state $(1, 101)$ than in $(1, 100)$. Numerical evidence and the above reasoning suggests the following strengthened version of Conjecture 1.

Conjecture 2: An optimal policy has the property that whenever it assigns a job to an available processor, it makes the assignment to the fastest available processor. Moreover, there exist thresholds, one for each state of the processors and decreasing in the state (seen as a boolean vector, in which busy and idle processors are indicated by 1 and 0, respectively), such that an optimal policy is to assign a job to an available processor if, and only if, the number of jobs in the buffer exceeds the relevant threshold for the present state of the processors.

The conjecture states that the thresholds depend on the state of the processors in such a manner that if it is *not* optimal to assign a job to processor i then it is *not* optimal to make an assignment to processor i until such time as there is at least one more arrival. That is, the state cannot change to one in which it becomes optimal to assign a job to processor i simply by the departure of jobs. This is the behavior we have observed numerically and argued for above.

References

[1] A. K. Agrawala, E. G. Coffman, Jr., M. R. Garey, and S. K. Tripathi, "A stochastic optimization algorithm minimizing expected flow times on uniform processors," *IEEE Trans. Comput.*, vol. C-33, pp. 351–356, 1984.
[2] E. G. Coffman, Jr., L. Flatto, M. R. Garey, and R. R. Weber, "Minimizing expected makespan on uniform processor systems," *Adv. Appl. Prob.*, vol. 19, pp. 117–201, 1987.
[3] P. R. Kumar and J. Walrand, "Individually optimal routing in parallel system," *J. Appl. Prob.*, vol. 22, pp. 989–995, 1985.
[4] W. Lin and P. R. Kumar, "Optimal control of a queueing system with two heterogeneous servers," *IEEE Trans. Automat. Contr.*, vol. AC-29, pp. 696–703, 1984.
[5] P. Luh and I. Viniotis, *Optimality of Threshold Policies for Heterogeneous Server Systems*, preprint, 1990. Raleigh, NC: North Carolina State Univ.
[6] R. Righter, "Job scheduling to minimize expected weighted flowtime on uniform processors," *Syst. Contr. Lett.*, vol. 10, pp. 211–216, 1988.
[7] R. Righter and S. Xu, "Scheduling jobs on heterogeneous processors," *Annals of OR*, vol. 28, pp. 587–602, 1991.
[8] R. Righter and S. Xu, "Scheduling jobs on nonidentical IFR processors to minimize general cost functions," *Adv. Appl. Prob.*, vol. 23, pp. 909–924, 1991.
[9] H. C. Tijms, *Stochastic Modelling and Analysis: A Computational Approach*. New York: Wiley, 1986.
[10] J. Walrand, "A note on 'Optimal control of a queueing system with two heterogeneous servers,'" *Syst. Contr. Lett.*, vol. 131, pp. 131–134, 1984.
[11] S. H. Xu, P. B. Mirchandani, S. P. R. Kumar, and R. R. Weber, "Stochastic dispatching of multipriority jobs to heterogeneous processors," *J. Appl. Prob.*, vol. 27, pp. 852–861, 1990.

On the Stability Proof of Adaptive Schemes with Static Normalizing Signals and Parameter Projection

T. Tsao and P. A. Ioannou

***Abstract*—Recently, it has been shown [8] that robust stability in model-reference adaptive control (MRAC) is guaranteed by simply using a static, instead of a dynamic, normalizing signal together with a parameter projection in the adaptive law. In this note, we show that the stability proof associated with such modification follows directly from the proof of schemes with dynamic normalization. Consequently, the proofs and stability arguments used in [8] can be simplified considerably.**

I. Introduction

Robustness has been an important issue in the study of adaptive control theory; and many successful modified schemes have been proposed to improve robustness with respect to unmodeled dynamics and bounded disturbances, and guarantee signal boundedness for any given initial conditions. One common feature of most robust and globally stable schemes is that the adaptive law employs a dynamic normalizing signal [1]–[7] that bounds the modeling error signal from above in addition to modifications, such as σ, switching σ, projection, and dead zones. In [8], it is shown that an adaptive law with a static normalizing signal similar to the one used in the ideal case and a parameter projection are sufficient to guarantee robust stability for a model-reference adaptive control (MRAC) scheme. The stability analysis use to establish this result is long and rather complicated and deviates from that used in the ideal case or in the case of schemes with a dynamic normalizing signal. In this note, we establish the same result by slightly modifying the proof associated with the dynamic normalizing signal. The proof is a direct application of the approach in [9], which unifies the design and analysis of various modified robust adaptive controllers. The approach of [9] is a simplification of that taken in [4], on which [8] relies to a large extent. The following notation is used in this note.

- $\|x_{t,T_0}\|_2^\delta \triangleq (\int_{T_0}^t e^{-(\delta\tau)} x^T(\tau)x(\tau)\,d\tau)^{1/2}$, where $x: [T_0, \infty) \to R^n$ has piecewise continuous elements and $\delta \geq 0$.
- $\|H(s)\|_\infty^\delta \triangleq \|H(s-(\delta/2))\|_\infty = \text{ess sup}_\omega\, \bar{\sigma}[H(j\omega-(\delta/2))]$ where $H(s)$ is a transfer function matrix analytic in $\text{Re}[s] \geq -(\delta/2)$ and $\delta \geq 0$.
- $\|H(s)\|_2^\delta \triangleq ((1/2\pi)\int_{-\infty}^\infty \text{trace}[H^T(-j\omega-(\delta/2))H(j\omega-(\delta/2)]\,d\omega)^{1/2}$ where $H(s)$ is a transfer function matrix analytic in $\text{Re}[s] \geq -(\delta/2)$ and $\delta \geq 0$.
- $x \in \mathscr{S}(y^2)$ means $\exists c_1, c_2 > 0$ such that $\int_t^{t+T} x^2(\tau)\,d\tau \leq c_1\int_t^{t+T} y^2(\tau)\,d\tau + c_2 \forall t, T \geq 0$.

II. Main Result

We consider the following plant:

$$y(t) = k_p \frac{Z_0(s)}{R_0(s)}(1+\Delta_m(s))u(t) \tag{1}$$

and reference model

$$y_m(t) = W_m(s)r(t) = k_m \frac{Z_m(s)}{R_m(s)} r(t) \tag{2}$$

where k_m is a constant; $Z_m(s)$, $R_m(s)$ are monic Hurwitz polynomials of degree m_r, n_r, respectively; and the poles and zeros of $W_m(s)$ are all in $\text{Re}[s] < -(p/2)$ for some $p > 0$. The following assumptions are made for the plant-transfer function.

1) $Z_0(s)$ and $R_0(s)$ are monic polynomials of degree m and n, respectively, $Z_0(s)$ is Hurwitz, $m \leq n-1$, and k_p is a constant gain with known sign.

2) The relative degree of $W_m(s)$ satisfies $n_r - m_r = n - m$ and $n_r \leq n$.

3) $\Delta_m(s)$ is a stable transfer function such that $\Delta_m W_m$ is strictly proper and analytic in $\text{Re}[s] \geq -(p/2)$.

Additive plant perturbations and bounded disturbances may also be included in (1) without altering the analysis and results of this note. Consequently, such perturbations are omitted without loss of generality.

The model-reference adaptive control law is given by

$$u(t) = \theta^T(t)\omega(t) = \theta_1^T(t)\frac{a(s)}{\Lambda(s)}u(t) + \theta_2^T(t)\frac{a(s)}{\Lambda(s)}y(t) + \theta_3(t)y(t) + c_0(t)r(t) \tag{3}$$

Manuscript received June 21, 1991; revised November 15, 1991.
The authors are with the Department of Electrical Engineering—Systems, University of Southern California, Los Angeles, CA 90089.
IEEE Log Number 9203143.