Distributed Algorithms with Random Processor Failures

G. P. Papavassilopoulos

Abstract—We examine a distributed algorithm where the processors may fail in a random fashion. This results in a model with random communication delays. Convergence conditions are derived. Extensions of the analysis and results to cases where the random processor failures are perceived and corrected within random time intervals are possible. For the sake of simplicity, the analysis is presented for a two processor model for solving a system of linear equations.

I. Introduction

The area of asychronous parallel and distributed algorithms has been recognized as being of great importance in achieving the solution of very large problems (see [1]–[11]). One of the issues that has attracted the attention of researchers is the study of convergence under several types of delays by which the messages arrive to the processors from other ones. In this correspondence, we examine a generic case where the message delays are due to processor failures which can happen in a random fashion. We show that random failures of processors result in a model which is generically covered by the Markovian delay model of [13], and that the conditions for convergence assume a particular and quite meaningful form for the problem at hand. For the sake of simplicity, we consider in detail the case of two processors which work in parallel toward solving a system of linear equations.

In Section II we state the problem, and in Section III we provide the solution. Finally, in Section IV, we discuss the results, delineate important issues to be studied further pertaining to the problem considered here, and point out other classes of interesting problems that can be cast and analyzed in the same framework.

II. Problem Description

Let us consider the system of linear equations

$$x = Ax + b \tag{1}$$

where A is an $n \times n$ real matrix and b is a vector in R^n. A way of solving this system for x is given by the algorithm

$$x_{k+1} = Ax_k + b. \qquad k = 0, 1, 2, \cdots \tag{2}$$

which converges if all eigenvalues of A are within the unit disk. Consider now that we have two processors—P1 and P2—implementing algorithm (2) in parallel. P1 updates x_k^1, and P2 updates x_k^2, where x_k^1 is composed of the first n_1 components of x_k, and x_k^2 of the remaining $n_2 = n - n_1$ components. Thus, splitting A and b appropriately, (2) can be written as

$$x_{k+1}^1 = A_{11}x_k^1 + A_{12}x_k^2 + b_1 \tag{3a}$$

$$x_{k+1}^2 = A_{21}x_k^1 + A_{22}x_k^2 + b_2. \tag{3b}$$

Equations (3a) and (3b) assume implicitly that x_k^2 comes to P1 immediately so that P1 uses x_k^2 in updating x_k^1 according to (3a). Similarly, x_k^1 becomes immediately available to P2. If x_k^2 suffers at time k, a delay $d(2, 1, k) = 1, 2, 3, \cdots$, in arriving from P2 to P1, and x_k^1 suffers a delay $d(1, 2, k) = 1, 2, 3, \cdots$, in arriving from P1 to P2, then (3a and 3b) must be substituted by

$$x_{k+1}^1 = A_{11}x_k^1 + A_{12}x_{k-d(2,1,k)}^2 + b_1 \tag{4a}$$

$$x_{k+1}^2 = A_{21}x_{k-d(1,2,k)}^1 + A_{22}x_k^2 + b_2. \tag{4b}$$

For the cases where the delays $d(i, j, k)$ are bounded, see the analysis in [1]–[5], and [11]; and for the case where they are stochastic, see the analysis in [13], and [14]. The case that we wish to examine in this correspondence is the following one. We assume that P1 never fails, whereas P2 may fail to function as may happen with probability $1 - p, 0 \le p \le 1$. If P2 fails, it takes a certain amount of time to have it fixed, namely, m instants of time. During this period, P1 keeps on working using the last reported value of x^2, whereas P2 remains idle during this period. when it restarts, it uses the x^1 value communicated to it by P_1 at this current time, and the x^2 value that it had when it stopped functioning. A formal description of this case is as follows. Let

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \qquad \overline{A} = \begin{bmatrix} A_{11} & A_{12} \\ 0 & I \end{bmatrix}$$

$$b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \qquad \overline{b} = \begin{bmatrix} b_1 \\ 0 \end{bmatrix}. \tag{5}$$

If

$$x_{k+1} = Ax_k + b \tag{6}$$

then

$$x_{k-2} = \begin{cases} Ax_{k-1} + b, & \text{with probability } p \\ \overline{A}x_{k-1} + b, & \text{with probability } 1 - p. \end{cases} \tag{7}$$

If

$$x_{k-1} = \overline{A}x_k + \overline{b}$$

$$x_k = \overline{A}x_{k-1} + \overline{b}$$

$$\vdots$$

$$x_{k-l+1} = \overline{A}x_{k-l} + \overline{b}$$

$$x_{k-l} = \overline{A}x_{k-l-1} + \overline{b}$$

$$x_{k-l-1} = Ax_{k-l-2} + b \tag{8}$$

then

$$x_{k+2} = \begin{cases} \overline{A}x_{k+1} + \overline{b} & \text{if: } l + 1 < m \\ Ax_{k+1} + b, & \text{if: } l + 1 = m. \end{cases} \tag{9}$$

Our objective is to study the convergence of the sequence x_k, generated by (6)–(9).

Manuscript received July 21, 1992; revised January 14, 1993 and March 25, 1993. This work was supported in part by the National Science Foundation under Grant CCR-9222734

The author is with the Department of Electrical Engineering–Systems, University of Southern California, Los Angeles, CA 90089-2563 USA.

IEEE Log Number 9216467.

III. Solution

For reasons of simplicity, and without loss of generality, we will consider that b is the zero vector. Also, for reasons of simplicity, we will consider $m = 3$, i.e., when P2 fails, it remains idle for 3 instants of time before recuperating. The generalization of the results for any $m \geq 1$ will be obvious from our analysis.

It is clear from (6)–(9) that the evolution of the algorithm depends on the updating formula used during the last three instants of time. In particular,

$$\begin{aligned}
&\text{if } x_k = \overline{AAA}x_{k-3} \quad \text{then } x_{k+1} = A\overline{AAA}x_{k-3} = A\overline{AA}x_{k-2}\\
&\text{if } x_k = \overline{AA}Ax_{k-3}, \quad \text{then } x_{k+1} = \overline{AAA}Ax_{k-3} = \overline{AAA}x_{k-2}\\
&\text{if } x_k = \overline{A}AAx_{k-3}, \quad \text{then } x_{k+1}\\
&\quad = \overline{AA}AAx_{k-3} = \overline{AA}Ax_{k-2}\\
&\text{if } x_k = \overline{A}A\overline{A}x_{k-3}, \quad \text{then } x_{k+1}\\
&\quad = \overline{AA}A\overline{A}x_{k-3} = \overline{AA}Ax_{k-2}\\
&\text{if } x_k = A\overline{AA}x_{k-3}, \quad \text{then } x_{k+1}\\
&\quad = \begin{cases} AAA\overline{A}x_{k-3} = AAAx_{k-2} & \text{with probability } p\\ \overline{A}AA\overline{A}x_{k-3} = \overline{A}AAx_{k-2} & \text{with probability } 1-p \end{cases}\\
&\text{if } x_k = A\overline{AA}x_{k-3}, \quad \text{then } x_{k+1}\\
&\quad = \begin{cases} AA\overline{AA}x_{k-3} = AA\overline{A}x_{k-2} & \text{with probability } p\\ \overline{A}A\overline{AA}x_{k-3} = \overline{A}A\overline{A}x_{k-2} & \text{with probability } 1-p \end{cases}\\
&\text{if } x_k = AAAx_{k-3}, \quad \text{then } x_{k+1}\\
&\quad = \begin{cases} AAAAx_{k-3} = AAAx_{k-2} & \text{with probability } p\\ \overline{A}AAAx_{k-3} = \overline{A}AAx_{k-2} & \text{with probability } 1-p. \end{cases}
\end{aligned} \tag{10}$$

It is clear now that we can cast the evolution of the algorithms (6)–(9) in the following framework. Let

$$\Gamma_1 = \overline{AAA}, \quad \Gamma_2 = \overline{AA}A, \quad \Gamma_3 = \overline{A}AA,$$

$$\Gamma_4 = \overline{A}A\overline{A}, \quad \Gamma_5 = A\overline{AA},$$

$$\Gamma_6 = A\overline{AA}, \ \Gamma_7 = AAA. \tag{11}$$

Then (6)–(9) can be described equivalently as

$$x_{k+3} = \Gamma(k) \cdot x_k \tag{12}$$

where $\Gamma(k)$ is a Markovian process of matrices, which can take the take values $\Gamma_1, \Gamma_2, \cdots, \Gamma_7$ for each k and the Markovian transition matrix is P:

$$\begin{aligned}
P &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0\\ 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1-p & 0 & 0 & 0 & p\\ 0 & 0 & 0 & 1-p & p & 0 & 0\\ 0 & 0 & 1-p & 0 & 0 & 0 & p \end{bmatrix}^3\\
&= \begin{bmatrix} 0 & 1-p & p(1-p) & 0 & 0 & 0 & p^2\\ 0 & 0 & 0 & 1-p & p & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 0\\ 1-p & p(1-p) & p^2(1-p) & 0 & 0 & 0 & p^3\\ 1-p & p(1-p) & p^2(1-p) & 0 & 0 & 0 & p^3\\ 1-p & p(1-p) & p^2(1-p) & 0 & 0 & 0 & p^3 \end{bmatrix}.
\end{aligned} \tag{13}$$

The mean square convergence of the sequence $\{x_{3l}\}(l = 0, 1, 2, \cdots)$ can be studied by applying directly the results of [12] and [13], in particular Theorem 1 of [13] or Theorem 1 in Chapter 4 of [12]. References [12] and [13] consider the general iteration

$$y_{k+1} = \Gamma(k)y_k$$

where $\Gamma(k)$ is a stationary Markovian process with state space the $n \times n$ matrices $\Gamma_1, \cdots, \Gamma_m$ and with associated transition matrix P of dimension $m \times m$. Thus, applying the results of [12] and [13] and using the easily verifiable fact that for the sequence generated by (6)–(9) with $m = 3$, mean square convergence of $\{x_{3l}\}(l = 0, 1, 2, 3, \cdots)$ implies mean square convergence of $\{x_k\}(k = 0, 1, 2, 3 \cdots)$ yields the following.

Theorem 1:[1,2] Let

$$\begin{aligned}
&\Gamma = (\Gamma_1 \otimes \Gamma_1) \oplus (\Gamma_2 \otimes \Gamma_2) \oplus \cdots \oplus (\Gamma_7 \otimes \Gamma_7)\\
&\bar{I} = \text{unit matrix with the same dimension as } \Gamma.
\end{aligned} \tag{14}$$

The sequence $\{x_k\}$ generated by (6)–(9) converges in the mean square sense to the solution of (1),[3] for every initial condition, if and only if all the eigenvalues of the matrix $\Gamma(P' \otimes \bar{I})$ are strictly inside the unit circle.

We will now proceed by examining the particular form that the condition of Theorem 1 assumes when the Γ_i's are as in (11) and P is as in (13). Let λ be an eigenvalue of $\Gamma(P' \otimes \bar{I})$ with associated eigenvector $x' = (x_1', x_2', x_3', x_4', x_5', x_6', x_7')$, where each $x_i \in R^n$. It holds

$$\begin{aligned}
(1-p)\bar{\Gamma}_1(x_5 + x_6 + x_7) &= \lambda x_1\\
(1-p)\bar{\Gamma}_2 x_1 + p(1-p)\bar{\Gamma}_2(x_5 + x_6 + x_7) &= \lambda x_2\\
p(1-p)\bar{\Gamma}_3 x_1 + p^2(1-p)\bar{\Gamma}_3(x_5 + x_6 + x_7) &= \lambda x_3\\
(1-p)\bar{\Gamma}_4 x_2 &= \lambda x_4\\
p\bar{\Gamma}_5 x_2 &= \lambda x_5\\
\bar{\Gamma}_6(x_3 + x_4) &= \lambda x_6\\
p^2\bar{\Gamma}_7 x_1 + p^3\bar{\Gamma}_7(x_5 + x_6 + x_7) &= \lambda x_7
\end{aligned} \tag{15}$$

where

$$\bar{\Gamma}_i = \Gamma_i \odot \Gamma_i.$$

Let

$$y = x_5 + x_6 + x_7.$$

If the eigenvalue λ is zero, then the condition of Theorem 1 is met. Thus, consider $\lambda \neq 0$. We then have

$$\begin{aligned}
x_1 &= \frac{1-p}{\lambda}\bar{\Gamma}_1 y\\
x_2 &= \frac{p(1-p)}{\lambda}\bar{\Gamma}_2 y + \frac{1-p}{\lambda}\bar{\Gamma}_2\frac{1-p}{\lambda}\bar{\Gamma}_1 y\\
x_3 &= \frac{p^2(1-p)}{\lambda}\bar{\Gamma}_3 y + \frac{p(1-p)}{\lambda}\bar{\Gamma}_3\frac{1-p}{\lambda}\bar{\Gamma}_1 y\\
x_4 &= \frac{1-p}{\lambda}\bar{\Gamma}_4\left[\frac{p(1-p)}{\lambda}\bar{\Gamma}_2 y + \frac{1-p}{\lambda}\bar{\Gamma}_2\frac{1-p}{\lambda}\bar{\Gamma}_1 y\right]
\end{aligned}$$

[1] The symbols $\otimes$ and $\oplus$ denote, respectively, the Kronecker product and sum of two matrices, and $'$ denotes the transpose of a matrix.

[2] It should be pointed out that Theorem 1 of [13] provides a sufficient condition for convergence in the mean square sense of the iteration $y_{k+1} = \Gamma(k)y_k + w_k$, where w_k is noise. Nonetheless, it is clear, from equations (18) and (20) of [13], that if the noise is identically zero, then the conditions provided are also sufficient. See also [12].

[3] The condition of Theorem 1 that $\Gamma(P' \otimes \bar{I})$ has all its eigenvalues strictly inside the unit circle implies that A has no eigenvalue equal to 1, and thus (1) has a unique solution. For if A has the eigenvalue 1 with associated eigenvector $\bar{x}$, then $(\bar{x}', \bar{x}')$ is an eigenvector of both $A \otimes A$ and $\bar{A} \otimes \bar{A}$ with associated eigenvalue 1, and thus 1 is also an eigenvalue of $\Gamma(P' \otimes \bar{I})$.

$$x_5 = \frac{p}{\lambda}\overline{\Gamma}_5\left[\frac{p(1-p)}{\lambda}\overline{\Gamma}_2 + \frac{1-p}{\lambda}\overline{\Gamma}_2\frac{1-p}{\lambda}\overline{\Gamma}_1\right]y$$

$$x_6 = \frac{1}{\lambda}\overline{\Gamma}_6\left[\frac{p^2(1-p)}{\lambda}\overline{\Gamma}_3 + \frac{p(1-p)}{\lambda^2}\overline{\Gamma}_3\overline{\Gamma}_1\right]y + \frac{1}{\lambda}\overline{\Gamma}_6\frac{1-p}{\lambda}\overline{\Gamma}_4\left[\frac{p(1-p)}{\lambda}\overline{\Gamma}_2 + \frac{(1-p)^2}{\lambda^2}\overline{\Gamma}_2\overline{\Gamma}_1\right]y$$

$$x_7 = \frac{p^3}{\lambda}\overline{\Gamma}_7 y + \frac{p^2}{\lambda}\overline{\Gamma}_7\frac{1-p}{\lambda}\overline{\Gamma}_1 y. \tag{16}$$

Adding now the last three equations, we have

$$y = \left\{\frac{p^2(1-p)}{\lambda}\overline{\Gamma}_5\overline{\Gamma}_2 + \frac{p(1-p)^2}{\lambda^3}\overline{\Gamma}_5\overline{\Gamma}_2\overline{\Gamma}_1 + \frac{p^2(1-p)}{\lambda^2}\overline{\Gamma}_6\overline{\Gamma}_3 + \frac{p(1-p)^2}{\lambda^3}\overline{\Gamma}_6\overline{\Gamma}_3\overline{\Gamma}_1 + \frac{p(1-p)^2}{\lambda^3}\overline{\Gamma}_6\overline{\Gamma}_4\overline{\Gamma}_2 + \frac{(1-p)^3}{\lambda^4}\overline{\Gamma}_6\overline{\Gamma}_4\overline{\Gamma}_2\overline{\Gamma}_1 + \frac{p^3}{\lambda}\overline{\Gamma}_7 + \frac{p^2(1-p)}{\lambda^2}\overline{\Gamma}_7\overline{\Gamma}_1\right\}y \tag{17}$$

which implies that λ satisfies

$$\det|-\lambda^4 I + \lambda^2 p^2(1-p)\overline{\Gamma}_5\overline{\Gamma}_2 + \lambda p(1-p)^2\overline{\Gamma}_5\overline{\Gamma}_2\overline{\Gamma}_1 + \lambda^2 p^2(1-p)\overline{\Gamma}_6\overline{\Gamma}_3 + \lambda p(1-p)^2\overline{\Gamma}_6\overline{\Gamma}_3\overline{\Gamma}_1 + \lambda p(1-p)^2\overline{\Gamma}_6\overline{\Gamma}_4\overline{\Gamma}_2 + (1-p)^3\overline{\Gamma}_6\overline{\Gamma}_4\overline{\Gamma}_2\overline{\Gamma}_1 + \lambda^3 p^3\overline{\Gamma}_7 + \lambda^2 p^2(1-p)\overline{\Gamma}_7\overline{\Gamma}_1| = 0. \tag{18}$$

Let

$$L = A \odot A. \qquad M = \overline{A} \odot \overline{A}. \tag{19}$$

Using the property that $(A \odot B)(C \otimes D)(E \odot F) = (ACE) \otimes (BDF)$ holds for any matrices A, B, C, D, E, F with combatible dimensions, we obtain

$$\begin{aligned}
\overline{\Gamma}_5\overline{\Gamma}_2 &= [(AA\overline{A}) \odot (AA\overline{A})][(\overline{A}AA) \odot (\overline{A}AA)] \\
&= L^2MM^2L = L^2M^3L \\
\overline{\Gamma}_5\overline{\Gamma}_2\overline{\Gamma}_1 &= L^2M^3LM^3 \\
\overline{\Gamma}_6\overline{\Gamma}_3 &= LM^2ML^2 = LM^3L^2 \\
\overline{\Gamma}_6\overline{\Gamma}_3\overline{\Gamma}_1 &= LM^3L^2M^3 \\
\overline{\Gamma}_6\overline{\Gamma}_4\overline{\Gamma}_2 &= LM^2MLMM^2L = LM^3LM^3L \\
\overline{\Gamma}_6\overline{\Gamma}_4\overline{\Gamma}_2\overline{\Gamma}_1 &= LM^3LM^3LM^3 \\
\overline{\Gamma}_7 &= L^3 \\
\overline{\Gamma}_7\overline{\Gamma}_1 &= L^3M^3.
\end{aligned} \tag{20}$$

Substituting (20) in (18), and after some extensive matrix manipulations, we have the equivalent to (18)

$$\det|-\lambda^4 I + \{L[\lambda pI + (1-p)M^3]\}^3| = 0. \tag{21}$$

Setting

$$\lambda = \mu^3 \tag{22}$$

we have the equivalent

$$\det|-\mu^4 I + L[p\mu^3 I + (1-p)M^3]| = 0. \tag{23}$$

A little reflection will persuade the reader that in the general case where m is not necessarily 3, (23) should be replaced by (24). We have thus simplified the condition of Theorem 1 and we have the following.

Theorem 2: The sequence $\{x_k\}$ generated by the algorithm (6)–(9) converges in the mean square sense, for every initial condition to the solution of (1), if and only if the roots of the equation

$$\det|-\mu^{m+1}I + L[p\mu^m I + (1-p)M^m]| = 0 \tag{24}$$

are strictly within the unit disc, where $L = A \odot A$ and $M = \overline{A} \odot \overline{A}$.

In the case where Processor 2 never fails, i.e., $p = 1$, (24) becomes

$$\det|-\mu^{m+1} + L\mu^m| = 0$$

and since the eigenvalues of L are the pairwise products of the eigenvalues of A, the convergence condition says equivalently that all eigenvalues of A should be within the unit disk. Similarly, when $p = 0$, i.e., P2 fails always, after operating correctly only once, condition (24) yields

$$\det|-\mu^{m+1} + LM^m| = 0$$

which says equivalently that the eigenvalues of $(A\overline{A}) \otimes (A\overline{A})$ are within the unit disk, or equivalently that the eigenvalues of $A\overline{A}$ are within the unit disk as should be expected.

IV. Discussion of the Results and Conclusions

The rapidity of convergence for the case $m = 3$ is, of course, related to the cube of the largest magnitude root μ of (23). But since the $|\mu|^3$ yields the rate of convergence of $\{x_0, x_3, x_6, x_9, \cdots\}$, i.e., $x_{3l} \sim (|\mu|^3)^l$, we obtain that $x^k \sim (|\mu|^k)$, i.e., the whole sequence $\{x_k\}$ converges in the mean square sense with a rate like $|\mu|^k$.

It should be interesting to notice that (24) has the obvious interpretation that it considers the convex combination $p \cdot L \cdot \mu^m + (1-p)LM^m$. Notice also the symmetry between the powers of μ and M, both of which are m. Unfortunately, the dependence of the roots of (24) on p is not linear. Obviously, the roots of (24) and their interplay with p and m is an important issue. Several questions which are not trivially resolved arise. 1) If we have stability for $p = 1$ and $p = \overline{p}$, i.e., if the iteration is guaranteed to work in the best and worst cases, does that imply that it will operate successfully for any p in $(\overline{p}, 1)$? Given the nonlinear dependence of the roots of (24) on p, the answer may very well be negative. 2) If we have stability for a certain m, does that imply that we will have stability for any other m' where $1 \le m' < m$ (p is considered fixed)? Both of these questions have practical importance: for 1) the rationale is that if we substitute processor P2 with another processor which is of better quality (i.e., it has a larger p) and we were guaranteed convergence with the older, less reliable processor, should we expect things to work as well with the better quality processor? For 2), the rationale is that if we substitute P2 with another processor that may fail with the same probability as the previous one, but it takes less time to fix it, i.e., $m' < m$, should we expect things to work at least as well as with the other (more time-consuming) to repair the processor? Both of these questions essentially ask whether a better quality P2 (i.e., larger p or smaller m) is preferable. Since the answer may be negative (see Appendix A), it is worthwhile to single out classes of A matrices for which the answer to these questions is affirmative. Although it may seem bothersome that there are cases where a worse quality processor may facilitate convergence, it is not surprising. For example, classical relaxation schemes may facilitate convergence by allowing a slower pace. The fact that processors may fail is not a characteristic introduced by the algorithm but a real characteristic of any processor used. It is clearly important to single out classes of

problems (i.e., matrices A) for which it holds that a better quality processor is always beneficial.

Another issue of importance that can be easily incorporated into our model concerns the case where if P2 fails, it takes 0 or 1 or 2 or $\cdots$ or $l-1$ instants of time to realize that it failed with corresponding probabilities $q_1, q_2, \cdots, q_l$. As soon as it is realized that P2 failed, it takes 1 or 2 or 3 or $\cdots$ or u instants of time to fix it with corresponding probabilities $\sigma_1, \sigma_2, \cdots, \sigma_u$. This more general case can be easily cast in the generic framework of iteration (12) where the $\Gamma(k)$ matrices assume a finite number of possible values and $\{\Gamma(k)\}$ is a Markovian process. Again, the basic theorem of [12] and [13] can be applied in this case, but the notation becomes more cumbersome. See Appendix B.

Finally, the case where we do not have only two processors but many, several of which can fail, can easily be covered by extending in a straightforward manner the framework and analysis presented here, at the expense of a more involved notation.

Besides all these extensions, we believe that a deeper understanding of the conditions supplied in Theorem 2, and the study of the first two questions presented in the second paragraph of this section, is of paramount importance in understanding the generic features of parallel algorithms with random failure of processors.

A model related to the one studied here can be found in [15] where, instead of processor failures, the authors consider that a processor takes a random number of stages to complete its assigned calculation. The random failure model we consider can be recast equivalently by assuming random processing times. Reference [15] considers nonlinear iterations and provides sufficiency conditions for almost-everywhere convergence, whereas in this corresponding it is done by considering that linear iterations provide necessary and sufficient conditions for mean square convergence.

Appendix A

As an example, so that both questions 1) and 2) can be answered in the negative, let

$$A = \begin{bmatrix} -\frac{1}{2} & 1 \\ c & \frac{1}{2} \end{bmatrix}.$$

A is stable if and only if $-9/12 < c < 3/12$, and $A\overline{A}$ is stable if and only if $-5/12 < c < 27/12$. Thus, for p close to zero, the condition of Theorem 2 is met, whereas for p close to 1, it is not met if $3/12 < c < 27/12$. Thus, substituting the P2 with another one that fails with a smaller probability may be detrimental to the convergence of the algorithm. For the same range of values of c, i.e., $3/12 < c < 27/12$, the condition is met for m fixed, if $p = p(m)$ is sufficiently close to zero, whereas in the extreme case $m = 0$ (and for any p), (24) reads $\det|-\lambda I + L[p \cdot I + (1-p)\cdot I] = \det|-\lambda I + L|$ which has roots greater than one in magnitude. Thus, substituting the Processor 2 with another one that can be fixed faster (actually $m = 0$ means it never fails!) will be detrimental to the convergence of the algorithm.

Appendix B

Let us consider the same model considered in Section II with two processors—P1 and P2—where P2 can fail with probability $1-p$, and where it takes two instants of time to fix it, i.e., $m = 2$. In addition, consider that the failure is either noticed immediately with probability q or noticed one unit of time later with probability $1-q$. Clearly, if the failure is noticed right away, the algorithm will operate with $\overline{A}$ for two time periods, whereas if it is noticed with a delay of one instant of time, the algorithm will operate with $\overline{A}$ for three instants of time. A rationale similar to the one used in (10) results in the following model. Let

$$\Gamma_1 = AAAA, \quad \Gamma_2 = \overline{A}AAA, \quad \Gamma_3 = AAA\overline{A},$$

$$\Gamma_4 = \overline{AA}AA, \quad \Gamma_5 = A\overline{AA}A,$$

$$\Gamma_6 = AA\overline{AA}, \quad \Gamma_7 = \overline{A}AA\overline{A},$$

$$\Gamma_8 = \overline{AAA}A, \quad \Gamma_9 = \overline{AA}A\overline{A},$$

$$\Gamma_{10} = \overline{A}A\overline{AA}, \quad \Gamma_{11} A\overline{AAA}.$$

If $x_k = \Gamma_j x_{k-4}$, then $x_{k+1} = \Gamma_j x_{k-3}$, where the transition probabilities p_{ij} are given by the Markovian matrix

$$P = \begin{bmatrix} p & 1-p & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ p & 1-p & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & q & 0 & 0 & 1-q & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & p & 0 & 0 & 0 & 1-p & 0 \\ 0 & 0 & p & 0 & 0 & 0 & 1-p & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & q & 0 & 0 & 1-q & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & p & 0 & 0 & 0 & 1-p & 0 \end{bmatrix}.$$

Thus, considering the sequence $y_l = x_{4l}$, we have $y_{l+1} = \Gamma(l) y_l$, where $\{\Gamma(l)\}$ is a Markovian process that takes possible values $\Gamma_1, \Gamma_2, \cdots, \Gamma_{11}$ with transition matrix P^4.

Theorem (1) of [13] or Theorem 1 in Chapter 4 of [12] are now applicable and yield the desirable conditions for mean square convergence. It should be noticed that this model can be easily generalized to the case where the fixing period m is any integer, and where the failure is noticed right away with probability q or with one stop delay with probability $q_1, \cdots$, or with l steps delay with probability q_{l+1}, and where $p_1 + \cdots + p_{l+1} = 1$. Clearly, even for relatively small values of m and l, the possible values of the $\Gamma(l)$ stochastic matrix increase rapidly in multitude.

References

[1] D. Chazan and W. Miranaker, "Chaotic relaxation," *Linear Algebra Appl.*, vol. 2, pp. 199–222, 1969.

[2] G. M. Baudet, "Asynchronous iterative methods for multiprocessors," *J. ACM*, vol. 25, pp. 226–244, 1978.

[3] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation, Numerical Methods*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[4] P. Tseng, D. P. Bertsekas, and J. N. Tsitsiklis, "Partially asynchronous, parallel algorithms for network flow and other problems," *SIAM J. Contr. Optimiz.*, vol. 28, pp. 678–710, 1990.

[5] J. N. Tsitsiklis, D. M. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Automat. Contr.*, vol. AC-31, pp. 803–812, 1986.

[6] J. N. Tsitsiklis and G. D. Stamoulis, "On the average communication complexity of asynchronous distributed algorithms," Rep. LIDS-P-1986, Lab. Inform. Dec. Syst., M.I.T. 1990.

[7] H. J. Kushner and G. Yin, "Asymptotic properties of distributed and communicating stochastic approximation algorithms," *SIAM J. Contr. Optimiz.*, vol. 25, pp. 1266–1290, 1987.

[8] D. Mitra, "Asynchronous relaxations for the numerical solution of differential equations by parallel processors," *SIAM J. Sci. Stat. Comput.*, vol. 8, pp. 43–58, 1987.

[9] H. T. Kung, "Synchronized and asynchronous parallel algorithms for multiprocessors," in *Algorithms and Complexity: New Directions and Recent Results*, J. F. Traub *et al.*, Eds. New York: Academic, 1976.

[10] P. Spiteri, "Parallel asynchronous algorithms for solving boundary value problems," in *Parallel Algorithms and Architecture*, M. C. *et al.*, Ed. New York: North-Holland, 1986.

[11] A. Üresin and M. Dubois, "Parallel asynchronous algorithms for discrete data," *J. ACM*, vol. 37, pp. 588–606, 1990.
[12] B. H. Bharucha, "On the stability of randomly varying systems," Ph.D. dissertation, Univ. Calif., Berkeley, 1961.
[13] B. F. Beidas and G. P. Papavassilopoulos, "Convergence analysis of asynchronous linear iterations with stochastic delays," *Parallel Comput.*, vol. 19, pp. 281–302, 1993. (A shorter version appeared in *Proc. 30th IEEE CDC*, England, Dec. 1992.)
[14] ——, "Asynchronous implementation of optimization algorithms with time drift," Rep. 91-10-01, Dep. Elec. Eng.—Syst., South. Calif., 1991.
[15] A. J. Gao, Y. M. Zhu, and G. Yin, "Joint robustness of noise and Liapunov function for parallel stochastic approximation algorithms," *J. Math. Anal. Appl.*, vol. 173, pp. 229–254, 1993.

Sampled-Data Controller Design for Uncertain Systems

Richard M. Dolphus

Abstract—**This note presents a sampled-data controller design methodology for uncertain systems. A continuous system with bounded time-varying uncertainty is sampled at intervals of length T. The controller is designed using a Riccati equation approach by neglecting $O(T^2)$ uncertainty terms in the discretized system. Stability is verified for this choice of T with the $O(T^2)$ terms included. If there exists a stabilizing continuous controller, then there also exists a stabilizing sampled-data controller for a sufficiently small choice of T.**

I. Introduction

Several papers address the problem of designing a continuous controller to stabilize a linear system containing time-varying uncertainty, for example, see [2], [5] [11]–[13]. Here we consider the case where a sampled-data controller is used as shown in Fig. 1. The sampled state $x_k = x(kT)$ is found by sampling $x(t)$ at intervals of length T. The discrete control u_k is passed through a D/A device consisting of a zero-order hold to obtain $u(t)$. We work with a state-space representation of the system, as opposed to the transfer function representation used in [1]. Other previous work with uncertain sampled-data systems includes an analysis technique for robust stability [3], and design of a nonlinear (ultimate boundedness) controller for a matched system [9]. Here we are interested in linear controller design which we apply to systems with rank-1 uncertainty.

Consider the problem of determining a linear full state feedback sampled-data controller. One approach is to first design a continuous controller using the techniques referenced above, and then use the same feedback gain matrix in the sampled-data controller with a "sufficiently small" sampling time. It is not clear, however,what constitutes "sufficiently small" for a system with uncertainty. Instead of using a continuous controller design in a sampled-data controller setting, here we design the controller with the discretized system in mind.

The structure of the uncertainty is changed in a complex manner when the model is discretized. We show that by neglecting discretized uncertainty terms of $O(T^2)$, however, the remaining terms retain the structure of the uncertainty in the continuous model, i.e., to $O(T)$ there is no smearing of the uncertainty terms throughout the matrices. We can design based on these simpler terms, and then verify that the design is stabilizing for the true system with the $O(T^2)$ terms included. As T becomes small, the $O(T^2)$ terms become negligible, and so our ability to design a sampled-data controller is only a function of the original continuous system's uncertainty structure. The inference is that if a stabilizing continuous controller can be designed for an uncertain system, then a stabilizing sampled-data controller can also be determined.

Manuscript received October 17, 1990; revised August 17, 1992. Paper recommended by Associate Editor D. F. Delchamps. This work was supported by NASA under Grant NGT-50551.

The author is with the Department of Mechanical and Aerospace Engineering, University of California, Irvine, CA 92717 USA.

IEEE Log Number 9216468.

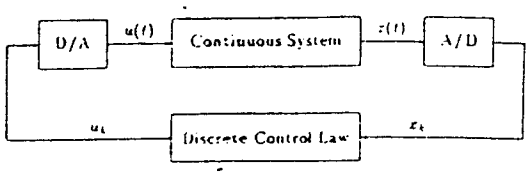

Fig. 1.

We develop a design methodology which is a sampled-data version of the continuous time design technique presented in [13]. This technique is a Riccati equation approach for systems whose uncertainty satisfies rank-1 conditions. Sampled-data versions of other continuous time design techniques can be developed using the same general framework. For matched time invariant uncertainties a sampled-data controller design methodology is given in [6]. Here we extend that work to include uncertainty that is rank-1 and time varying.

II. Problem Formulation and Notation

Consider a system described by

$$\dot{x}(t) = (\mathcal{A} + \Delta\mathcal{A}(r(t)))x(t) + (\mathcal{B} + \Delta\mathcal{B}(r(t)))u(t) \qquad (1)$$

where $x(t) \in R^n$ is the state and $u(t) \in R^m$ is the control. The uncertainty $r(t) \in R^l$ belongs to a known compact set $\mathcal{R}$ where

$$\mathcal{R} = \{r: r_i^- \le r_i \le r_i^+, \quad i = 1, 2, \cdots, l\}. \qquad (2)$$

We assume that $(\mathcal{A}, \mathcal{B})$ is a controllable pair, the matrix functions $\Delta\mathcal{A}(\cdot)$ and $\Delta\mathcal{B}(\cdot)$ are continuous, and $r(\cdot)$ is Lebesgue measurable. Using a sampled-data controller with zero-order hold, the control $u(t)$ is given by

$$u(t) = u_k \quad \text{for } kT \le t < kT + T \qquad (3)$$

where u_k is a function of x_k, the state measured at $t = kT$. Our problem is to find a linear relationship between u_k and x_k such that the closed-loop system is asymptotically stable.

Following is a discussion of notation. We use script letters $\mathcal{A}$ and $\mathcal{B}$ for the continuous system, and italic letters A and B for the discretized system. The norm $\|\cdot\|$ indicates the induced matrix 2-norm. For real symmetric matrices X and Y, we write $X > 0 (X \ge 0)$ if X is positive (semi-) definite, and $X > Y (X \ge Y)$ if $(X - Y)$ is positive (semi-) definite. We also use the ordering notation $O(\cdot)$. For scalar T and matrix $M(T)$, we write $M(T) = O(T^p)$ if there exists a constant scalar $m > 0$ such that $\|M(T)\| \le m|T|^p$ for all T in a neighborhood of zero.

III. Model Discretization

Define the nominal system as system (1) with $\Delta\mathcal{A} = 0$ and $\Delta\mathcal{B} = 0$. Let

$$x_{k+1} = Ax_k + Bu_k \qquad (4)$$