walls to detect multiple echo pulses. The TOF determination as well as the detection of the echo pulses is accomplished by searching the correlation peaks of the matched filter output. The invariant spatial relationship between the ultrasonic sensor and the reflectors in our method allows simple but robust matching between the echoes obtained at different sensor orientations. The multiple echo pulses are first collected as groups on the basis of their TOF's, then the groups of echo pulses are tested for their physical validity. The distances to the reflectors are estimated from the mean times-of-flight of the groups of echo pulses, and the directions are estimated by fitting the modeled echo amplitude pattern to the magnitude patterns of the groups of echo pulses in least-square sense.

In our method, the specular effect and the wide beam width are exploited positively. The former reduces the number of reflectors to a manageable size in typical indoor environments, and the latter allows large sampling step size in scanning.

The validity as well as the performance of the proposed method was shown through the experiments to localize a mobile robot in real environment. We first extracted multiple acoustic landmarks whose positions were known with respect to the world coordinate frame. Assuming that the spatial relationship between the mobile robot and its ultrasonic sensors was given, the mobile robot was localized to within 2 cm in distance and 0.1 degrees in direction from the true positions.

REFERENCES

[1] A. Elfes, "Sonar-based real-world mapping and navigation," *IEEE J. Robot. Automat.*, vol. RA-3, no. 3, pp. 249–265, June 1987.

[2] D. W. Cho, "Certainty grid representation for robot navigation by a bayesian method," *ROBOTICA*, vol. 8, pp. 159–165, 1990.

[3] M. Beckerman and E. M. Oblow, "Treatment of systemetic errors in the processing of wide-angle sonar sensor data for robotic navigation," *IEEE Trans. Robot. Automat.*, vol. 6, no. 2, pp. 137–145, Apr. 1990.

[4] M. Drumhèller, "Mobile robot localization using sonar," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, no. 2, pp. 325–332, Mar. 1987.

[5] R. Kuc and M. W. Siegel, "Physically based simulation model for acoustic sensor robot navigation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, no. 6, pp. 766–778, Nov. 1987.

[6] B. Barshan and R. Kuc, "Differentiating sonar reflections from corners and planes by employing an intelligent sensor," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, no. 6, pp. 560–569, June 1990.

[7] Ö. Bozma and R. Kuc, "Building a sonar map in a specular environment using a single mobile sensor," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, no. 12, pp. 1260–1269, Dec. 1991.

[8] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Trans. Robot. Automat.*, vol. 7, no. 3, pp. 376–382, June 1991.

[9] H. Peremans, K. Audenaert, and J. M. Van Campenhout, "A high-resolution sensor based on tri-aural perception," *IEEE Trans. Robot. Automat.*, vol. 9, no. 1, pp. 36–48, Feb. 1993.

[10] L. Kleeman and R. Kuc, "An optimal sonar array for target localization and classification," in *Proc. 1994 IEEE Int. Conf. Robotics and Automaion*, San Diego, CA, May 1994, pp. 3130–3135.

[11] L. E. Kinsler, A. R. Frey, A.n B. Coppens, and J. V. Sanders, *Fundamentals of Acoustics,* 3rd ed. New York: Wiley, 1982, pp. 178–187.

[12] Polaroid Corp., *Ultrasonic Ranging System* (manual and handbook), Cambridge, MA, 1984.

Pattern Reconfiguration in Swarms—Convergence of a Distributed Asynchronous and Bounded Iterative Algorithm

Gerardo Beni and Ping Liang

Abstract— **Swarms are physical realizations of self-organizing distributed robotic systems (DRS). This paper provides a rigorous analysis of swarm behavior and introduces a new methodology for using swarms to solve DRS pattern reconfiguration problems. We introduce the linear swarm model and show that it is an iterative method for asynchronously solving linear systems of equations under physically relevant constraints. The main result of the paper is a proof of a sufficient condition for the asynchronous convergence of a linear swarm to a synchronously achievable configuration. This is important since a large class of DRS self-organizing tasks can be mapped into reconfigurations of patterns in swarms.**

I. INTRODUCTION

Advanced robotics applications require: 1) flexibility; 2) fault tolerance; and 3) intelligence. The traditional robot design strategy is to develop a single sophisticated robot with the above properties. In contrast, the distributed robotic systems (DRS) strategy is to develop a robotic system with the above properties via self-organization of multiple simpler autonomous agents. DRS research has accelerated recently because of: 1) advances in computer and communication technologies; and 2) its relation to other fields such as decentralized autonomous systems, multiagent systems, self-organizing systems, distributed artificial intelligence, and artificial life.

Swarms have been developed as models for the self-organization of distributed robotic systems [1]–[3]. DRS [4] have promising applications in many fields, for example, in: 1) flexible manufacturing and modular/reconfigurable robotics [5]; 2) the operation of groups of autonomous vehicles [6], [7]; 3) distributed intelligence and distributed sensors [8]; and 4) distributed intelligent structures [9], [10]. The objectives of a DRS include carrying out: 1) tasks impossible to single robots; and/or 2) tasks possible to single robots with the advantage of being more reliable, self-repairable and lower in cost due to their simplicity of construction.

An advantage of the swarm models is the increased privacy of the process, since information from a subset of units does not directly reveal the final configuration, even when the size of the swarm is known. Since swarms can model manufacturing, as well as defense/law-enforcement operation processes, the increased privacy results in increased security since the "capture" of a subset of the swarm would not reveal the goal of the process.

The basic DRS problem is to design a system that is capable of carrying out a useful task as a group, and only as such; i.e., its component units are assumed incapable of the task if they do not form a group above a critical size. A special case is the problem of swarm intelligence [11], [12]. The main difficulty of such a problem, and of DRS in general (and, even more generally, of distributed computers systems), is the asynchronicity, of task execution. The *asynchronicity*

Manuscript received August 8, 1994; revised June 15, 1995. This paper was recommended for publication by Associate Editor A. De Luca and Editor S. E. Salcudean upon evaluation of reviewers' comments.

The authors are with the College of Engineering, University of California, Riverside, CA 92508 USA.

Publisher Item Identifier S 1042-296X(96)01147-0.

constraint distinguishes models of DRS behavior from e.g., the cellular automata models which were used originally for the self-organization (self-reproduction) of automata [13]. An example is the cellular robotic system (CRS) model [14]. In the CRS model, besides the asynchronicity requirement, the physical constraint of "mass" conservation was introduced to distinguish models of groups of robots from models of groups of computers. The conservation requirement can apply to either the number of robotic units themselves or to the amount of "material" exchanged between them [14]. The former case models the behavior of robots as self-organizing biomorphic entities, the latter case models the behavior of robots reorganizing patterns of material, e.g. on a manufacturing line. Both cases have been included in a more general formulation than the CRS, i.e., the swarm model [1].

Swarms are defined as DRS capable of self-organization. Although the term "self-organization" has many different, and often vague, meanings in the technical literature, for the swarm problem it was unambiguously defined in [1] via the concept of a synextask. A synextask is a "synchronously executable task," i.e., a task subdivided into subtasks which can, but do not have to be necessarily, executed synchronously by the units of the DRS. A self-organization task is a synextask which during execution conserves some property of the DRS, e.g. the number of its units, or the total amount of "material" belonging to the DRS.

Thus, we have seen that *asynchronous and conservative execution of synchronously executable tasks is the fundamental and defining characteristic of the swarm*. Mathematically, the task execution by the swarm can be modeled as set of N entities redistributing asynchronously among M states. In other words, task execution is the asynchronous "distortion" (since N is conserved) of a pattern over the M states of the swarm.

This fundamental characteristic of the swarm (conservative and asynchronous execution of synextasks) has been used in [2] to solve a large class of swarm problems, i.e., the formation of patterns which are solutions to difference equations. One practical advantage of executing tasks according to this type of pattern formation in swarms, is the privacy of the process, since the final global pattern and even the final local state is unknown to the units executing the task. The method developed in [2] for forming patterns in swarms was based on the asynchronous execution of a system of equations generated by a low order difference equation with a conservation condition. The low order choice reflected the modeling of local relations, in the spirit of DRS problems. In fact, DRS's are typically modeled as evolving via local relations. If the relationships were global, the requirement of decentralization would be weakened. On the other hand, the locality of the relation is not as essential as the locality of interaction (e.g., sensing can be long range, but actuation must be local) which is a defining property of a DRS. This will be discussed more clearly after the model of a linear swarm is introduced below.

The defining property of the swarm requires that the task be synchronously executable; hence the system of equations solved asynchronously must be solvable synchronously as well. For this requirement, in the uniform swarm considered in [2], all units act identically, except one. This "symmetry breaking unit" is necessary because each unit corresponds to a linear equation and the conservation requirement provides another linear equation. Thus, together there are $N+1$ equations and only N variables where N is the number of units. Therefore, one unit must be made *inactive* to eliminate one equation so that the system of equations can be solved simultaneously with the conservation condition. The main objective of this paper is to provide a rigorous proof of the conditions for convergence of the swarm models. A second objective is to generalize the notion of cyclic swarm model given in the next section.

II. Swarm Models

Cyclic boundary conditions were chosen in [2] because of the uniformity of the swarm. Hence, from [2], the "Cyclic Swarm," i.e., a swarm that generates patterns as solutions to difference equations with cyclic boundary conditions, is defined as follows.

Definition 1—Cyclic Swarm: A set of M ordered entities ($k = 1, 2, \cdots, M$) such that:

1) they share one resource y allocated to the units with some *conservation* constraint; (e.g., $\Sigma_k y_k = N$);
2) they are connected via an identical local function G_k of the resource allocation, $G_k(y_k, y_{k+1}, y_{k+2}, \cdots, y_{k+m};\ m \ll M)$, with cyclic conditions: $y_{M+i} = y_i$;
3) all entities, except for a symmetry breaking one (i.e., for $k = 1, 2, \cdots M-1$), update their, and their neighbor's, resource allocation by *asynchronously* exchanging a quantum of resource according to an *identical, local* rule, as follows: Calculate G_k for all k
   - if $G_k > \varepsilon$ then $y_k \to y_k - 1;\ y_{k+1} \to y_{k+1} + 1$;
   - if $G_k < -\varepsilon$ then $y_k \to y_k + 1;\ y_{k+1} \to y_{k+1} - 1$;
   - else do nothing; (ε is a constant: $0 \le \varepsilon \le 1$).

It is clear that the "asynchronicity plus conservation" characteristics of a swarm pattern reconfiguration is contained in part 3 of the definition of a cyclic swarm, which describes the elementary act of an asynchronous/conservative mechanism of reconfiguring a pattern. Cyclic swarms as defined above have also been investigated in [16], where an external input was considered.

A linear swarm defined below is a generalization of Definition 1 (variations from the "cyclic" swarm definition are underlined for clarity):

Definition 2—Linear Swarm: A set of M ordered entities ($k = 1, 2, \cdots, M$) such that:

1) they share one resource y allocated to the units with some conservation constraint; (e.g., $\Sigma_k y_k = N$);
2) they are connected via linear functions (not necessarily identical, and not necessarily local) G_k of the resource allocation, i.e. $G_k = y_k - \Sigma_j b_{kj} y_j$ where b_{kj} are constant ($j = 1, 2, \cdots M;\ j \neq k$);
3) all entities, except for a symmetry breaking one (i.e., for $k = 1, 2, \cdots, M-1$), update their, and their neighbor's, resource allocation by asynchronously exchanging a quantum of resource according to an identical, local rule, as follows: Calculate G_k for all k
   - if $G_k > \varepsilon$ then $y_k \to y_k - 1;\ y_{k+1} \to y_{k+1} + 1$;
   - if $G_k < -\varepsilon$ then $y_k \to y_k + 1;\ y_{k+1} \to y_{k+1} - 1$;
   - else do nothing; (ε is a constant: $0 \le \varepsilon \le 1$).

The generalizations introduced in part 2 pertain to the locality of the relation, and, as noted, to the uniformity. Locality and uniformity of interaction are still contained in part 3.

The condition for convergence of the linear swarm can be found by realizing that what it does is to solve *asynchronously* and iteratively a system of linear equations.[1] It is important to recall that although we are seeking an asynchronous solution, the self-organization of a swarm (i.e., its reconfiguration) must be a synextask, i.e., a task that can (but not necessarily must) be subdivided into synchronously executable subtasks. Hence, any asynchronous solution to the reconfiguration problem must have a synchronous counterpart. Therefore, the linear relation G_k and the conservation condition form a system of

[1] This scheme may not be as efficient as the standard relaxation algorithms for solving systems of linear equations. However, this is not the objective of the linear swarm which is a model of pattern formation in DRS.

linear equations that must have a synchronous solution. The algorithm for the asynchronous solution is discussed next.

III. Distributed Asynchronous Bounded (DAB) Iterative Algorithm

Mathematically, a swarm implements an iterative scheme for solving a linear system of equations under asynchronous, distributed, and quantized—hence "bounded"—updating. In most physically relevant cases, the additional constraint of conservation is also included in the updating. Since rule 3 is a local, asynchronous and "quantized" exchange of resources, we call the algorithm Distributed Asynchronous Bounded iteration algorithm, or briefly DAB. The DAB technique and the convergence analysis of the swarm are presented in the next two sections.

Standard iterative schemes [17] are either synchronous (Jacobian and JOR) or sequential with a fixed order (Gauss–Seidel and SOR). It can be shown that the Jacobian (simultaneous) and Gauss–Seidel (sequential) iteration schemes converge under the same condition that the matrix norm is less than one, even when the updating is done asynchronously (without time delay) and randomly in each iteration. Difficulty in convergence arises when time delay is present.

In the following, we formulate the DAB algorithm. Given a linear system of equations

$$\mathbf{A}\mathbf{y} = \mathbf{s} \tag{1}$$

where $\mathbf{A}$ is an $M \times M$ nonsingular matrix and $\mathbf{y}$ and $\mathbf{s}$ are $M \times 1$ vectors. Without loss of generality, assuming nonvanishing diagonal elements a_{kk}, (1) can be written into

$$\mathbf{y} = \mathbf{B}\mathbf{y} + \mathbf{c} \tag{2}$$

where

$$b_{kj} = \begin{cases} -a_{kj}/a_{kk}, & k \neq j \\ 0, & k = j \end{cases} \quad \text{and} \quad c_k = s_{kj}/a_{kk}.$$

The constraints on the swarms defined in Section II translate into the following constraints on the iterative method.

1) *Distributed Asynchronicity*: The self-activation of a unit occurs according to its internal probability distribution, and it is totally independent of the probability distribution of self-activation of other units.

   Consequence: Since each sites self-activates for updating at each iteration according to an internal probability distribution $0 < p_k^{(i)} \leq 1$, any subset of sites may be updated at an iteration.

   Significance: Unlike Gauss–Seidel, SOR, Jacobian, or JOR iterative schemes, where one unit must wait for all the others to be activated before being allowed to self-activate again, the DAB scheme allows for independent (possibly parallel), asynchronous updating of each unit. Physically, the DAB scheme is an iterative scheme that does not requires centralized supervision of the asynchronous updating process throughout the system. Hence the DAB scheme can model a "distributed asynchronous" process.

2) *Synchronicity*: No time delay between interacting sites.

   Consequence: The resource values on the units participating in the updating, for sensing, communicating and/or transferring resources, must be available synchronously to those units. This limits the asynchronicity of the swarm.

   Significance: Mathematically this contrasts with models of asynchronous distributed computing, where the asynchronicity is in fact due to delays in interactions between the units [18]; physically, this restriction on asynchronicity is very plausible for swarms, which are intended to model systems interacting locally. On a local scale, synchrony is easily realized; the implausibility of the "no-time delay" restriction scales directly with the range of the interaction.

3) *Resource Conservation*: The total resource is a constant, i.e., $\sum_k y_k = N$.

   Consequence: This requires that if a unit is to increase its value by one, it must decrease the value of another unit by one, and vice versa. Note that y_k may be allowed to be both positive and negative with a negative value indicating an unsatisfied request for resource. Alternatively, it may be enforced that y_k be positive at all time, e.g., if a site attempts to increase by one and the value of its immediate neighbor is zero, the increment will not occur. In this case, convergence is possible only for a system which has a solution with positive values at all sites.

   Significance: Mathematically, conservation complicates the problem. The conserved resource is a global variable that controls the overall pattern. Physically, conservation is a very common occurrence since most real systems have finite total resources.

4) *Bounded Quantization*: A swarm unit can deal with a finite number of resource quanta at a time.

   Consequence: The resource value at a site can only increase or decrease by a discrete bounded amount at each update. Therefore, an exact solution with $|g_k| = 0$ may not be achieved and convergence should be considered as achieved when $|g_k| < \varepsilon$ (in what follows we let $\varepsilon = 1$).

   Significance: Mathematically this constraint does not reduce the generality of the algorithm; but it is physically significant since it models realistically the fact that any physical system, e.g. a swarm unit, has bounds on the amount of resources that it can deal with. Standard iterative algorithms do not consider this physical restriction and allow for arbitrary update values.

The previous four constraints can be cast mathematically as the following iterative scheme. Define

$$g_k = y_k^{(i)} - \sum_j b_{kj} y_j^{(i)} - c_k. \tag{3}$$

The updating rule at two neighboring sites of the swarms is formulated as follows:

$$y_k^{(i+1)} = y_k^{(i)} - \operatorname{sgn}(g_k) S(|g_k| - 1) \tag{4}$$

$$y_{k+1}^{(i+1)} = y_{k+1}^{(i)} + \operatorname{sgn}(g_k) S(|g_k| - 1) \tag{5}$$

where

$$\operatorname{sgn}(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases} \quad S(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases}. \tag{6}$$

Equation (4) is referred to as the *active* updating rule, and (5) the *passive* updating rule. Each unit updates itself according to the active updating rule and updates the nearest neighbor according to the passive updating rule. Hence, the last unit (i.e., the symmetry breaking unit) is only *passively* updated, to implement the conservation constraint.

The above updating rule can be cast into a relaxation formulation as

$$\mathbf{y}^{(i+1)} = \mathbf{B}(i)\mathbf{y}^{(i)} + \mathbf{c}(i) \tag{7}$$

where

$$\mathbf{B}(i) = \operatorname{diag}\left(\omega^{(i)}\right)\mathbf{B} + \left(\mathbf{I} - \operatorname{diag}\left(\omega^{(i)}\right)\right)$$

and

$$\mathbf{c}(i) = \operatorname{diag}\left(\omega^{(i)}\right)\mathbf{c}. \tag{8}$$

$\mathrm{diag}(\omega^{(i)})$ in (8) consists of the variable relaxation parameters $\omega^{(i)}$. Relaxation methods with variable ω has been considered in the literature [17]. But ω is the same for all variables in a single iteration. In the formulation of our scheme, the variable relaxation parameter is different for each variable in the same iteration. Therefore, we have a vector $\omega^{(i)}$ which can be found for each iteration from the updating rules in (4) and (5).

On the surface, the DAB iteration (8) appears now as a discrete time-variant linear dynamic system. The problem of convergence of the DAB equation becomes the problem of finding the asymptotic fixed point of the time-variant linear dynamic system in (8). However, the state transition matrix $\mathbf{B}(i)$ and the vector $\mathbf{c}(i)$ are nonlinear functions of the states. A convergence proof of the swarm using updating rules in (4) and (5) is given below.

IV. Convergence of DAB

In this section, we provide a sufficient condition for the convergence of DAB. Also, methods to reduce the error of the converged solution to the true solution are discussed. The convergence condition and convergence speed for a special class of swarms are also investigated. We first find the convergence condition without the conservation constraint.

A. DAB Without Conservation

If the conservation constraint is removed, there is no *passive* updating. That is, resources must be requested or dispensed at each site, there is no *passing* between sites. We prove the following theorem:

Theorem 1: If the matrix norm $\delta = \|\mathbf{B}\|_\infty = \max_{1\le k\le M} \sum_{j=1}^{M} |b_{kj}| < 1$ and the probability of each site being selected for updating is nonzero, i.e., $0 < p_k^{(i)} \le 1$ for all k in $\mathbf{p} = [p_1^{(i)}, p_2^{(i)}, \cdots p_M^{(i)}]^T$ the DAB algorithm without conservation converges to the vicinity of the true solution of (2). After convergence, the maximum error is bounded by

$$\left|\varepsilon^{(n)}\right|_{\max} = \max_k \left|y_k^{(n)} - y_k^*\right| < \frac{1}{1-\delta} \tag{9}$$

where y_k^* is the true solution. Convergence here means (9) is satisfied for all iterations after the nth iteration.

Remark: The convergence condition in this theorem is essentially identical with the standard iteration methods [16]. The differences are in the asynchronicity of updating, and in the need of adaptation to bounded updating. The theorem can be easily adapted to any quantized updating, i.e., the updating term $-\mathrm{sgn}(g_k^{(i)})S(|g_k^{(i)}| - 1)$ in (4) and (5) can be replaced by $-\alpha_k \mathrm{sgn}(g_k^{(i)})S(|g_k^{(i)}| - \beta_k)$ where α_k and β_k are any positive real numbers.

Let the true solution vector be $\mathbf{y}^*$

$$\mathbf{y}^* = \mathbf{B}\mathbf{y}^* + \mathbf{c}. \tag{10}$$

Rewrite (10) as

$$\mathbf{c} = \mathbf{y}^* - \mathbf{B}\mathbf{y}^*. \tag{11}$$

Define the error vector

$$\varepsilon^{(i)} = \mathbf{y}^{(i)} - \mathbf{y}^* \tag{12}$$

and substitute (11) into the expression of $\mathbf{g}^{(i)}$, we have

$$\mathbf{g}^{(i)} = \mathbf{y}^{(i)} - \mathbf{B}\mathbf{y}^{(i)} - \mathbf{y}^* + \mathbf{B}\mathbf{y}^* = \varepsilon^{(i)} - \mathbf{B}\varepsilon^{(i)}. \tag{13}$$

Therefore,

$$\begin{aligned}\varepsilon^{(i+1)} &= \mathbf{y}^{(i+1)} - \mathbf{y}^* = \mathbf{y}^{(i)} - \mathbf{y}^* - \mathrm{sgn}\left(\mathbf{g}^{(i)}\right)\mathbf{S}\left(\left|\mathbf{g}^{(i)}\right| - 1\right) \\ &= \varepsilon^{(i)} - \mathrm{sgn}\left(\varepsilon^{(i)} - \mathbf{B}\varepsilon^{(i)}\right)\mathbf{S}\left(\left|\mathbf{g}^{(i)}\right| - 1\right)\end{aligned} \tag{14}$$

where

$$\mathrm{sgn}\left(\mathbf{g}^{(i)}\right) = \left[\mathrm{sgn}\left(g_1^{(i)}\right), \mathrm{sgn}\left(g_2^{(i)}\right), \cdots, \mathrm{sgn}\left(g_n^{(i)}\right)\right]^T$$

and

$$\begin{aligned}S\left(\left|\mathbf{g}^{(i)}\right| - 1\right) = \big[&S\left(\left|g_1^{(i)}\right| - 1\right), \\ &S\left(\left|g_2^{(i)}\right| - 1\right), \cdots, S\left(\left|g_n^{(i)}\right| - 1\right)\big]^T.\end{aligned}$$

By the definition of swarms, if $|g_k^{(i)}| < 1$, there is no updating at site k in iteration i. Therefore, we ignore these sites in an iteration where no updating takes place, and assume $|g_k^{(i)}| \ge 1$ at each site picked for updating in an iteration.

Assume $\varepsilon_k^{(i)} > 0$, and the maximum error is out of the bound, i.e., $|\varepsilon^{(i)}|_{\max} \ge \frac{1}{1-\delta}$, at the ith iteration. Since

$$g_k^{(i)} \ge \varepsilon_k^{(i)} - \Sigma_j |b_{kj}| \left|\varepsilon_j^{(i)}\right| \ge \varepsilon_k^{(i)} - \delta \left|\varepsilon^{(i)}\right|_{\max} \tag{15}$$

therefore, if

$$\varepsilon_k^{(i)} - \delta \left|\varepsilon^{(i)}\right|_{\max} \ge 1 \tag{16}$$

then $g_k^{(i)} \ge 1$. From (16) and $|\varepsilon^{(i)}|_{\max} \ge \frac{1}{1-\delta}$, we have

$$\varepsilon_k^{(i)} \ge 1 + \delta \left|\varepsilon^{(i)}\right|_{\max} \ge 1 + \frac{\delta}{1-\delta} = \frac{1}{1-\delta} > 1. \tag{17}$$

Therefore, if (16) is true

$$\left|\varepsilon_k^{(i+1)}\right| = \left|\varepsilon_k^{(i)}\right| - 1 < \left|\varepsilon_k^{(i)}\right|. \tag{18}$$

Obviously, (16) is a sufficient condition for (18). At the site with the maximum error, i.e., $\varepsilon_k^{(i)} = |\varepsilon^{(i)}|_{\max}$, (16) is satisfied. Hence, if the site with the maximum error is selected for updating and $|\varepsilon^{(i)}|_{\max} \ge \frac{1}{1-\delta}$, its magnitude will be reduced by 1.

From (15), we know that if

$$\varepsilon_k^{(i)} - \delta \left|\varepsilon^{(i)}\right|_{\max} > -1 \tag{19}$$

$g_k^{(i)} > -1$. Then, either (18) is true or

$$\left|\varepsilon_k^{(i+1)}\right| = \left|\varepsilon_k^{(i)}\right|. \tag{20}$$

Therefore, the only case where the magnitude of $\varepsilon_k^{(i)}$ may be increased by 1 is when

$$\varepsilon_k^{(i)} - \delta \left|\varepsilon^{(i)}\right|_{\max} \le -1, \quad \text{i.e.,} \quad \left|\varepsilon_k^{(i)}\right| \le \delta \left|\varepsilon^{(i)}\right|_{\max} - 1. \tag{21}$$

The error at a site where (21) is true satisfies the following inequality even with an increase of 1

$$\left|\varepsilon_k^{(i+1)}\right| \le \delta \left|\varepsilon^{(i)}\right|_{\max} < \left|\varepsilon^{(i)}\right|_{\max}. \tag{22}$$

Next, consider $\varepsilon_k^{(i)} < 0$. Similar to (15), we have

$$-g_k^{(i)} \ge -\varepsilon_k^{(i)} - \sum_j |b_{kj}| \left|\varepsilon_j^{(i)}\right| \ge \left|\varepsilon_k^{(i)}\right| - \delta \left|\varepsilon^{(i)}\right|_{\max}. \tag{23}$$

Similar analysis shows that the same conclusions as in the $\varepsilon_k^{(i)} > 0$ case hold for the $\varepsilon_k^{(i)} < 0$ case.

Combining (18), (20), and (22), if the site with the maximum error $|\varepsilon^{(i)}|_{\max}$ is selected for updating, we have

$$\left|\varepsilon^{(i+1)}\right|_{\max} < \left|\varepsilon^{(i)}\right|_{\max} \quad \text{when} \quad \left|\varepsilon^{(i)}\right|_{\max} \ge \frac{1}{1-\delta}. \tag{24}$$

Since updating is asynchronous, at an iteration, the site with the maximum error may not be selected for updating. At these iterations, the maximum error will stay the same, i.e., $|\varepsilon^{(i+1)}|_{\max} = |\varepsilon^{(i)}|_{\max}$.

Therefore, if all sites have a nonzero probability of being selected for updating and $|\varepsilon^{(i)}|_{\max} \geq \frac{1}{1-\delta}$, the maximum error will be reduced to below the bound after a sufficient number of iterations. Thus, the convergence is proved.

That $1/(1-\delta)$ is actually a bound for the maximum error after convergence can also be shown in another way. Consider the site with the maximum error $|\varepsilon^{(n)}|_{\max}$ after convergence, i.e., after $|g_k^{(n)}| < 1$ is reached for all sites. Assume $|\varepsilon^{(n)}|_{\max}$ occurs at site k, then

$$\left|\varepsilon^{(i)}\right|_{\max} \leq \left|g_k^{(i)}\right| + \sum_{j=1}^{M} |b_{kj}| \left|\varepsilon_k^{(i)}\right| < 1 + \delta \left|\varepsilon^{(i)}\right|_{\max}. \tag{25}$$

Therefore, we have

$$\left|\varepsilon^{(n)}\right|_{\max} < \frac{1}{1-\delta}. \tag{26}$$

□

This shows that the smaller the δ, the smaller the error bound. This is observed in the simulation. After the maximum error gets within the bound $1/(1-\delta)$, the maximum error may or may not further decrease, it may also oscillate within the bound. If the errors are reduced to $|\varepsilon_k^{(i)}| \leq |\varepsilon^{(i)}|_{\max} \leq \frac{1}{2}$, all updating will definitely stop since necessarily $|g_k^{(i)}| < 1$ for all sites as shown below:

$$\left|g_k^{(i)}\right| \leq \left|\varepsilon_k^{(i)}\right| + \sum_{j=1}^{M} |b_{kj}| \left|\varepsilon_k^{(i)}\right| \leq \left|\varepsilon^{(i)}\right|_{\max} + \delta \left|\varepsilon^{(i)}\right|_{\max} \leq (1+\delta)\frac{1}{2} < 1. \tag{27}$$

Note that the condition for convergence in the theorem is only a sufficient condition. Also, the error bound is loose due to the amplification used in deriving the inequalities in (15) and (25). In actual implementation, we observed that the error is often much smaller than the bound. In cases where the accuracy is insufficient since no updating takes place after $|g_k^{(i)}| < 1$, the deficiency can be overcome by a scaling method shown below.

Observe that if $\mathbf{y}^*$ is a solution to (2), then $\bar{\mathbf{y}}^* = \alpha \mathbf{y}^*$ is a solution to

$$\mathbf{y} = \mathbf{B}\mathbf{y} + \alpha \mathbf{c}. \tag{28}$$

If we apply our iteration scheme to (28), by Theorem 1, the error bound is

$$\left|\bar{y}_k^{(n)} - \bar{y}_k^*\right| = \alpha \max_k \left|y_k^{(n)} - y_k^*\right| < \frac{1}{1-\delta}. \tag{29}$$

Therefore

$$\max_k \left|y_k^{(n)} - y_k^*\right| < \frac{1}{\alpha(1-\delta)}. \tag{30}$$

By choosing $\alpha \gg 1$, an arbitrarily accurate solution can be found for the original equation. The accuracy can also be improved by replacing the updating term $-\mathrm{sgn}(g_k^{(i)})S(|g_k^{(i)}| - 1)$ in (4) and (5) with $-\alpha_k \mathrm{sgn}(g_k^{(i)})S(|(g_k^{(i)})| - \beta_k)$ using small positive α_k and β_k. These two methods can also be applied to the updating scheme with conservation constraint discussed in the next section.

B. DAB with Conservation

The theorem proved above assumes no conservation constraint, i.e., each site increases or decreases its value independently. If the conservation constraint is imposed, then passive updating will be present. Also, the last site can only be passively updated to implement the conservation constraint, or in other words, to break the symmetry.

Once the conservation constraint is imposed by passive updating, general conditions of convergence with asynchronous updating are difficult to prove. Convergence conditions may be found if the updatings are not totally asynchronous. For asynchronous operation of the swarm, we prove a convergence theorem under restrictive conditions for the DAB algorithm with the conservation constraint.

Theorem 2: In the DAB algorithm with conservation constraint $\sum_{k=1}^{M} y_k^{(i)} = N$ imposed by passive updating, if the restricted matrix norm $\delta = \max_{1 \leq k \leq M-1} \sum_{j=1}^{M} |b_{kj}| < 1$, and the probability being selected for active updating is nonzero for sites where $|\varepsilon_k^{(i)}| > \delta |\varepsilon^{(i)}|_{\max} - 1$ and zero for sites where $|\varepsilon_k^{(i)}| \leq \delta |\varepsilon^{(i)}|_{\max} - 1$, then the total absolute error defined as $|\varepsilon^{(i)}| = \sum_{k=1}^{M} |\varepsilon_k^{(i)}|$ is nonincreasing at each iteration, and is guaranteed to decreases as the number of iterations increases.

Proof: From the proof of theorem 1 (see (19)), if the conditions in Theorem 2 are satisfied, active updatings will always reduce the magnitude of the errors. A passive updating may increase the error since it is not based on the reduction of the error at the site being passively updated. This is like transporting an error at one site to the next site. It may also decrease the error if the errors at the two neighboring sites $\varepsilon_k^{(i)}$ and $\varepsilon_{k+1}^{(i)}$ have the same sign. This latter case is not of concern. In the worst case, the passive updating increases the magnitude of the error by 1 at a neighboring site. However, since the magnitude of the error is reduced by 1 at the site of the corresponding active updating, in the worst case, we have

$$\left|\varepsilon^{(i+1)}\right| \leq \left|\varepsilon^{(i)}\right| \tag{31}$$

since

$$\sum_{k=1}^{M} \varepsilon_k^{(i)} = \sum_{k=1}^{M} y_k^* - \sum_{k=1}^{M} y_k^{(i)} = N - N = 0 \tag{32}$$

is always true for all i, at every iteration, there must be neighboring sites whose errors are of opposite signs such that both the active updating and the passive updating reduce the magnitude of error. Therefore, $|\varepsilon^{(i)}|$ will be reduced as the number of iterations increases and the theorem is proved. □

The condition that only sites with $|\varepsilon_k^{(i)}| > \delta |\varepsilon^{(i)}|_{\max} - 1$ have a nonzero probability of being selected for active updating guarantees that if a site is actively updated, the magnitude of error is reduced by 1. However, this condition is not satisfied in our implementation. Convergence is almost always observed in the simulation if $\delta < 1$. A general proof of convergence in this case is yet to be found.

The convergence time with conservation constraint is normally significantly longer than the case without the conservation constraint, depending on the initial error distribution. This is because errors at one site may need to be transported over a number of sites to be canceled by errors with an opposite sign at other sites. The relative relations of the initial errors play an important role in determining the convergence time. Recall that the two methods mentioned in the last section may be applied here as well to achieve a better accuracy.

V. Simulation Results

In this section, we present some simulation examples to illustrate the results of the previous sections. All examples are with the conservation constraint. The equation we consider is the following:

$$y_k = \frac{1}{C}(-y_{k+2} + 2y_{k+1}). \tag{33}$$

Equation (33) is the finite difference equation (with forward difference) corresponding to the following second order differential equation:

$$\frac{d^2y}{dx^2} + (C-1)y = 0. \tag{34}$$

The number of sites M used is 4 in the three examples below. Only examples with four sites are presented for easy illustration. We have tested the algorithms with up to 100 sites. Similar results are observed confirming the analysis in the previous sections. The convergence time is counted as the number of active-passive updating pairs performed. In the simulation, only one site is randomly selected for active updating at each iteration, that is, $p_k^{(i)} = 1/M$ for all sites and all iterations. In a true parallel implementation, more than one site can self-activate for updating simultaneously.

Example 1: $C = -10$. When $C = -10$, $\delta = 0.3$. Since $M = 4$, there are three equations in the form of (33) and one conservation equation. Let $N = 100$. Choose initial distribution as

$$y_1^{(0)} = 0, \quad y_2^{(0)} = 10, \quad y_3^{(0)} = -10, \quad y_4^{(0)} = 100.$$

The accurate solution is

$$y_1^* = -5.485714285714285715, \quad y_2^* = 15.88571428571428571,$$
$$y_3^* = -23.08571428571428572, \quad y_4^* = 112.6857142857142857.$$

After 37 active-passive updates, the system converges to

$$y_1 = -5, \quad y_2 = 16, \quad y_3 = -22, \quad y_4 = 111.$$

The maximum initial error is at site 3 equaling to 13.09. The convergence time is approximately $37/13.09 = 2.83$ times the maximum initial error.

The $|g_k|$'s are: $|g_1| = 0.1$, $|g_2| = 0.5$, $|g_3| = 0.4$. $|g_4|$ is not shown since it corresponds to the conservation constraint which is not used in the iteration and is always satisfied by design.

Example 2: $C = -5$. When $C = -5.0$, $\delta = 0.6$. Let $N = 100$. The initial distribution is the same as Example 1:

$$y_1^{(0)} = 0, \quad y_2^{(0)} = 10, \quad y_3^{(0)} = -10, \quad y_4^{(0)} = 100.$$

The accurate solution is

$$y_1^* = -35, \quad y_2^* = 55, \quad y_3^* = -65, \quad y_4^* = 145.$$

Since the probability for updating and the range of interaction remain the same with Example 1, we anticipate that the convergence time should also be about 2.83 times the maximum initial error. The maximum initial error in this case is 55. Therefore, the convergence time should be about $2.83 \times 55 = 155$ iterations. In experiment, convergence is achieved after 148 active-passive updates. The system converges to

$$y_1 = -35, \quad y_2 = 54, \quad y_3 = -63, \quad y_4 = 144.$$

The $|g_k|$'s are: $|g_1| = 0.4$, $|g_2| = 0.6$, $|g_3| = 0.6$.

Example 3: $C = -1.6$. When $C = -1.6, \delta = 1.87$. Let $N = 100$. The accurate solution is

$$y_1^* = 95.59214020180562932, \quad y_2^* = -52.84121083377588954,$$
$$y_3^* = 47.26500265533722783, \quad y_4^* = 9.984067976633032396.$$

After over 5000 active-passive updates, the system did not converge and the errors are oscillating. This shows that when $\delta > 1$, the method may not converge. However, $\delta < 1$ is not a necessary condition, and we have observed convergence for $\delta > 1$.

VI. Conclusion

As stated in the introduction, our main objective was to make a rigorous analysis of swarm behavior, motivated by the indication [16] that cyclic swarms can reconfigure asynchronously to new patterns via a process that converges under rather general conditions. The analysis carried out in this paper confirms this observation. A sufficient condition is proved for the convergence of a more general linear swarm model to its synchronous configuration (i.e., to the configuration that the swarm would achieve if it were to operate synchronously). The physical interpretation of the convergence condition $\delta < 1$ is that the swarm converges to its synchronous solution if (but not only if) the coupling of a unit to its connected neighbors is relatively "weak."

References

[1] G. Beni and S. Hackwood, "Cyclic swarms," in *Proc. 1992 Japan-USA Symp. Flexible Automation*, San Francisco, CA, July 13–15, 1992, pp. 531–536.

[2] ——, "Stationary waves in cyclic swarms," in *Proc. 1992 IEEE Int. Symp. Intelligent Control*, Glasgow, U.K., Aug. 10–13, 1992, pp. 234–242.

[3] G. Beni, "Distributed robotic systems and swarm intelligence," *J. Robot Soc. Japan*, Aug. 1992 (in Japanese).

[4] H. Asama, T. Fukuda, T. Arai, and I. Endo, "Distributed autonomous robotic systems," in *Proc. Symp. Distributed Autonomous Robotic Systems '94*, Tokyo, Japan, 1994, pp. 335–339.

[5] T. Fukuda and Y. Kawauchi, "The cellular robotic system (CEBOT), a self-organizing system," *Intelligent Robotic Systems*, S. G. Tzafestas, Ed. New York: Marcel Dekker, 1991.

[6] H. Asama, K. Ozaki, H. Itakura, A. Matsumoto, Y. Ishida, and I. Endo, "Collision avoidance among multiple mobile robots based on rules and communication," in *Proc. IEEE Int. Workshop Intelligent Robots and Systems*, Osaka, Japan, Nov. 3–5, 1991, pp. 1215–1220.

[7] M. J. Mataric, "Minimizing complexity in controlling a mobile robot population," in *Proc. 1992 IEEE Int. Conf. Robotics and Automation*, Nice, France, May 1992, pp. 830–836.

[8] S. Hackwood and G. Beni, "Self-organization of sensors for swarm intelligence," in *Proc. 1992 IEEE Int. Conf. Robotics and Automation*, Nice, France, May 1992, pp. 819–829.

[9] S. G. Ma, S. Hackwood, and G. Beni, "Control system of multi-agent supporting systems with centralized inference estimator of disturbance," College Eng., Univ. California, Riverside, Tech. Rep., 1992.

[10] P. Liang, K. Jin, and G. Beni, "Distributed control of swarm structures," in *IEEE/Nogoya University WWW Workshop Multiple/Distributed Robot Systems*, July 1993, pp. 131–136.

[11] G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems," in *Proc. NATO Advanced Workshop Robots and Biological Systems*, Il Ciocco, Tuscany, Italy, June 26–30, 1989.

[12] *Proc. 7th Annu. Meeting Robotics Society of Japan*, Shibaura, Japan, Nov. 2–4, 1989, pp. 425–428 (in Japanese).

[13] J. Von Neumann, *Theory of Self Reproducing Automata*, A. Burks, Ed. Urbana, IL: Univ. Illinois Press, 1966.

[14] G. Beni, "The concept of cellular robot," in *Proc. 3rd IEEE Symp. Intelligent Control*, Arlington, VA., Aug. 1988, pp. 57–61.

[15] J. Wang and G. Beni, "Distributed computing problems in cellular robotic systems," in *Proc. IEEE Int. Workshop Intelligent Robots and Systems (IROS '90)*, Tsuchiura, Japan, July 3–6, 1990, pp. 453–458.

[16] G. Beni and S. Hackwood, "Coherent swarm motion under distributed control," in *Proc. Int. Symp. Distributed Autonomous Robotic Systems*, Riken, Wakoshi, Japan, Sept. 21–22, 1992.

[17] D. M. Young, *Iterative Solution of Large Linear Systems*. New York: Academic, 1971.

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