Agreement on Stochastic Multi-Robot Systems with Communication Failures

Fayette W. Shaw, Albert Chiu, and James D. McLurkin

Abstract— **Agreement algorithms allow individual agents in a population to estimate a global quantity by sharing information. A common example is computing the global mean of a sensor measurement from each agent. We present a practical agreement algorithm, *input-based consensus* (IBC), that produces bounded error and recovery in the face of significant communications failures in a stochastic distributed system. We compare our algorithm to *linear average consensus* (LAC), which produces an exact result under ideal conditions, but is not robust to message loss. For both algorithms, we measure performance with respect to a varying percentage of dropped messages. The algorithms are examined analytically, simulated using the Stochastic Simulation Algorithm, and demonstrated experimentally on a testbed of 20 robots. In all cases, the IBC algorithm produced reasonable values, even when tested with up to 90% message loss.**

I. Introduction

Agreement algorithms are widely used on multi-agent systems to diffuse information and combine local measurements of a global quality. An example of this type of algorithm is *linear average consensus* (LAC) [1], in which agents compute the global mean of a locally measured quantity by repeatedly computing pairwise averages between agents. Assuming no communications messages are lost and no part of the system is disconnected indefinitely, each agent's local estimate will converge to the global mean. At this point, it can be said that the agents are "in agreement", as the variance of the estimates on each agent can be made arbitrarily small. However, in the presence of intermittent communications failures, information about the global mean can be lost permanently, making agreement to the correct global value impossible to reach. In this paper, we apply the *input-based consensus* (IBC) algorithm [2], which weights both current states and initial states, to address this problem.

The contributions of this paper are as follows: First, we extend previous results [2] by examining the algorithm in the event of dropped messages; we demonstrate that the agents converge with bounded error. Second, we use LAC as a benchmark and compare its performance via root mean square error. We compare the two algorithms analytically, via simulation, and experimentally on a system of 20 mobile robots.

II. Related Work

There is a great deal of literature on agreement in multi-agent systems and on stochastic systems in general. We group this work into two categories: agreement algorithms, including graph theory and faulty agreement, and stochastic processes.

A. Agreement Algorithms and Graph Theory

Consensus in multi-agent systems is used to describe multi-agent systems from flocking birds to algorithms for controlling robots [1], [3], [4]. A common task is to compute the average of the agents' initial states. To reach agreement in a distributed manner, one approach is for each agent, k, to maintain a state variable, x_k, which is updated when interacting with neighboring agents:

$$\dot{x}_k = \sum_{j \in \mathcal{N}_k(t)} f_k(x_j, x_k). \tag{1}$$

Here, $j \in \mathcal{N}_k$ signifies that agent k has access to x_j, the state variable of agent j. Note that the neighbor set $\mathcal{N}_k$ may vary with time. When the output of the update rule, f_k, is a linear combination of the input, the resulting protocol is linear average consensus.

Graphs are used to model the communications topology between agents. A graph $\mathcal{G}$ consists of vertices and edges. We represent an undirected graph as an adjacency matrix, $\boldsymbol{A}$, where each entry (i, j) is 1 if vertices i and j are connected and 0 otherwise. The degree matrix $\boldsymbol{D}$ is a diagonal matrix each entry (i, i) is the sum of incident edges of vertex i. We define the Laplacian matrix $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$. Much of the consensus literature considers controllers of the form $\dot{\boldsymbol{x}} = -\boldsymbol{L}\boldsymbol{x}$, where systems of agents are governed by *Laplacian dynamics* [1].

Linear average consensus (LAC) is a specific case of Laplacian dynamics where agents are computing the average of the initial global state. In this paper, we examine faulty consensus in the presence of dropped messages, using LAC as a baseline to compare to our algorithm. LAC has the constraint that the sum of initial conditions, $\sum_i y_i^0$, must be constant for all time, where y_i^0 is the initial value for robot i. If messages are lost, the global sum changes, so linear average consensus will not converge to the correct value. The major flaw of linear average consensus is that the system cannot recover from any disturbance that changes the global sum.

Although LAC is not robust to message loss, we use it as a baseline because it is representative of a large class of consensus algorithms; those in which agents converge to the convex combination of initial states

and a globally invariant quantity is maintained through local interactions. Olfati-Saber et al. [1] refer to the class of algorithms with this invariance property as *average-consensus* algorithms. To our knowledge, computing global averages is the simplest such consensus application, thus, we found it suitable to use LAC to compute the global average as a baseline for comparison.

In previous work, we show the convergence and stability of the estimator and tracking of a changing value with finite variance [2]. LAC can also perform tracking with the inclusion of an exogenous input, or relative state change [5]. There have also been consensus-like algorithms developed to perform distributed Kalman filtering in noisy systems [6], [7].

Consensus problems have attracted the interest of many authors particularly under the condition of faulty communication. Often, consensus is considered with additive noise representing sensor error [8], [9], [10] and also as packet losses [11], [12]. Huang assumes an observer under the event of a known packet loss [11]. Our work is similar to the latter, in that we can also model packet loss as a Markov process. However, we assume that packet losses are not observable.

B. Stochastic Processes

Passive parts [13], non-locomoting but actuated robots [14], [15], and mobile robots [16] have been described using Markov process formalism. In these testbeds, robots move randomly and generate random communication graphs. There has been much work on random graphs in this context [17], [18], [19]. Agreement is reached often when the union of the graph induced by the communication events is connected [20]. However, this work does not directly address communication failure.

Our experimental multi-robot system generates random graphs through the mobility of the robots. We assume the system is well-mixed; that is, every agent is equally likely to interact with any other agent in the system. Thus, we can associate a rate k for interactions, where $k\,dt$ is the probability in the next dt seconds of an event occurring.

We model our system as a Stochastic Hybrid Systems (SHS) [21]. A SHS consists of a set of discrete and continuous states governed by a state-dependent stochastic differential equation. Since our system evolves via a deterministic update at random times for random pairs of robots, we find the SHS formalism fitting. To reason about the expected behavior over many runs, we examine the first and second moment dynamics for the estimate, $\langle \hat{\boldsymbol{x}} \rangle$ and $\langle \hat{\boldsymbol{x}} \hat{\boldsymbol{x}}^T \rangle$ respectively.

III. Preliminaries

Consider a set of n robots where each robot i has a constant discrete *internal state*, $q_i \in \{0, 1\}$. Define the *population fraction* to be

$$x \triangleq \frac{1}{n} \sum_{i=1}^{n} q_i, \tag{2}$$

or the discrete value averaged over all the robots. Each robot also maintains an *estimate* $\hat{x}_i(t) \in \mathbb{R}$ of x. It is assumed that each robot knows the value of n. The vector $\boldsymbol{q} = (q_1 \dots q_n)^T$ is defined to be the vector of internal states, $\hat{\boldsymbol{x}}(t) = (\hat{x}_1(t) \dots \hat{x}_n(t))^T$ to be the vector of estimates, and $\mathbb{1}$ to be the vector of 1s.

In our system, the robots move randomly through the environment, and randomly select a partner within its communication range to exchange information. We assume that the agents do not know when a message is lost, that they have limited computational and communication resources, and we design a a protocol to be as simple as possible.

IV. Input-Based Consensus

We introduced IBC in our previous work [2] and here, we apply the algorithm to a mobile multi-robot system. We consider the case in which the estimator update function is defined by a convex combination of the estimates and states of the interacting robots. In particular, if robot i interacts with robot j at time t then the robots update their estimates according to

$$\begin{aligned} \hat{x}_i(t+1) &= f(\hat{x}_i(t), q_i, \hat{x}_j(t), q_j) \\ \hat{x}_j(t+1) &= f(\hat{x}_j(t), q_j, \hat{x}_i(t), q_i) \\ \hat{x}_l(t+1) &= \hat{x}_l(t) \text{ for all } l \neq i, j, \end{aligned}$$

where $f(\hat{x}_i(t), q_i, \hat{x}_j(t), q_j)$ is defined by

$$\zeta \left(a\hat{x}_i(t) + (1-a)\hat{x}_j(t) \right) \; + \; (1-\zeta) \left(\frac{1}{n} q_i + \left(\frac{n-1}{n} \right) q_j \right) \tag{3}$$

at a particular time t. Here $\zeta \in [0, 1]$ is the *consensus parameter*, which is the weighting of the relative importance of the estimates and discrete states in the update rule; $\frac{1}{n}$ is the weighting of a robot's own discrete state; and $a \in [0, 1]$ is the weighting on a robot's own estimate. The last line of the above update rule represents the fact that robots not participating in the interaction do not update their estimates. We call this algorithm *input-based consensus* (IBC) because an agent's state appears as an input in the update equation. LAC is equivalent to IBC when $\zeta = 1$, we call ζ the consensus parameter. Here we demonstrate that our algorithm is robust to dropped communication between agents where the message loss is not detected.

Written as matrices, the update in which robots 1 and 2 happen to interact at time t can be written as

$$\text{or } \hat{\boldsymbol{x}}(t+1) = \zeta \boldsymbol{A}_{12} \hat{\boldsymbol{x}}(t) + (1-\zeta) \boldsymbol{B}_{12} \boldsymbol{q}. \tag{4}$$

In general, matrices $\boldsymbol{A}_{ij}$ and $\boldsymbol{B}_{ij}$ are defined as follows:

$$\begin{array}{ll} \boldsymbol{A}_{ij}(i,i) = \boldsymbol{A}_{ij}(j,j) = a & \boldsymbol{B}_{ij}(i,i) = \boldsymbol{B}_{ij}(j,j) = \frac{1}{n} \\ \boldsymbol{A}_{ij}(i,j) = \boldsymbol{A}_{ij}(j,i) = 1-a & \boldsymbol{B}_{ij}(i,j) = \boldsymbol{B}_{ij}(j,i) = \frac{n-1}{n} \\ \boldsymbol{A}_{ij}(l,l) = \frac{1}{\zeta} & \text{for all } l \neq i,j \end{array} \tag{5}$$

and all remaining matrix entries are 0. Matrices $\boldsymbol{A}_{12}$ and $\boldsymbol{B}_{12}$ are derived in this manner.

We prove in the condition without dropped messages that the estimate converges to the actual population fraction of the system [2].

A. Dropped Messages

In the case of dropped messages, the update function if robot i drops a message is

$$\begin{aligned} \hat{x}_i(t+1) &= \hat{x}_i(t) \\ \hat{x}_j(t+1) &= f(\hat{x}_j(t), q_j, \hat{x}_i(t), q_i) \\ \hat{x}_l(t+1) &= \hat{x}_l(t) \text{ for all } l \neq i,j, \end{aligned}$$

where $f(\hat{x}_i(t), q_i, \hat{x}_j(t), q_j)$ is defined as above (3). This results in the following update matrices where

$$\begin{array}{ll} \boldsymbol{C}_{iji}(j,j) = a & \boldsymbol{D}_{iji}(j,j) = \frac{1}{n} \\ \boldsymbol{C}_{iji}(j,i) = (1-a) & \boldsymbol{D}_{iji}(j,i) = \frac{n-1}{n} \\ \boldsymbol{C}_{iji}(i,i) = \boldsymbol{C}_{iji}(l,l) = \frac{1}{\zeta} & \end{array} \tag{6}$$

for a message dropped by robot i, with all the remaining matrix entries 0. There are corresponding matrices $\boldsymbol{C}_{ijj}$ and $\boldsymbol{D}_{ijj}$ for message dropped by robot j.

B. Moment Dynamics

Because the system is a stochastic process, we reason about the moments of the system, particularly, the mean for the estimate $\hat{x}$. The first moment of the estimator process is examined using the Stochastic Hybrid Systems (SHS) formalism introduced in the related work section. In the present case, the extended generator $\mathcal{L}$ is defined by

$$\mathcal{L}\psi(\hat{\boldsymbol{x}}, \boldsymbol{q}) = \sum_{i<j} \lambda_i \left(\psi\left(\phi_i\left(\hat{\boldsymbol{x}}, \boldsymbol{q}\right)\right) - \psi\left(\hat{\boldsymbol{x}}, \boldsymbol{q}\right)\right), \tag{7}$$

where ψ is the test function, ϕ_i is the update function for interaction i, and λ_i is the associated rate at which this process occurs. Since we are interested in the behavior of the mean of the estimate, which evolves at rate k, so we choose the test function to be $\psi(\hat{\boldsymbol{x}}) = \hat{\boldsymbol{x}}$ and substitute into (7), which gives

$$\begin{aligned} \frac{d}{dt}\langle \hat{\boldsymbol{x}} \rangle &= \gamma k \left\langle \left(\sum_{i<j} \left(\zeta \boldsymbol{A}_{ij} - \boldsymbol{I} \right) \right) \hat{\boldsymbol{x}} + (1-\zeta) \sum_{i<j} \boldsymbol{B}_{ij} \boldsymbol{q} \right\rangle \\ &+ (1-\gamma) k \left\langle \left(\sum_{i<j} \sum_{m \in i,j} \left(\zeta \boldsymbol{C}_{ijm} - \boldsymbol{I} \right) \right) \hat{\boldsymbol{x}} \right. \\ &\left. + (1-\zeta) \sum_{i<j} \sum_{m \in i,j} \boldsymbol{D}_{ijm} \boldsymbol{q} \right\rangle \end{aligned} \tag{8}$$

where the indices $i < j$ refer to the possible interactions between robots i and j. The second sum where $m \in i, j$ indicates that either robot i or j can drop a message.

Equation (8) can be simplified as follows. Define

$$\begin{aligned} \boldsymbol{A} &\triangleq \sum_{i<j} \boldsymbol{A}_{ij}, & \boldsymbol{B} &\triangleq \sum_{i<j} \boldsymbol{B}_{ij}, \\ \boldsymbol{C} &\triangleq \sum_{i<j} \boldsymbol{C}_{iji} + \sum_{i<j} \boldsymbol{C}_{ijj} & \boldsymbol{D} &\triangleq \sum_{i<j} \boldsymbol{D}_{iji} + \sum_{i<j} \boldsymbol{D}_{ijj}. \end{aligned}$$

For now, we consider the estimate independent from any changing discrete values and thus, assume that the discrete state $\boldsymbol{q}$ is constant. It can be shown that

$$\begin{aligned} \boldsymbol{A} &= \left(\frac{1}{\zeta} \left(\binom{n}{2} - n + 1 \right) + na - 1 \right) \boldsymbol{I} + (1-a) \mathbb{1}\mathbb{1}^T \\ \boldsymbol{B} &= \boldsymbol{D} = \frac{n-1}{n} \mathbb{1}\mathbb{1}^T \\ \boldsymbol{C} &= \boldsymbol{A} + \frac{1}{\zeta} \binom{n}{2} \boldsymbol{I}. \end{aligned} \tag{9}$$

We can show that equation (8) becomes

$$\frac{d}{dt}\langle \hat{\boldsymbol{x}} \rangle = k \left(\left(\zeta \boldsymbol{A} - \binom{n}{2} \boldsymbol{I} \right) \langle \boldsymbol{x} \rangle + (1-\zeta) \boldsymbol{B} \boldsymbol{q} \right), \tag{10}$$

which yields the same desired equilibrium solution

$$\langle \hat{\boldsymbol{x}} \rangle^* = x \mathbb{1}$$

and the same results for the dynamics of the second moment as in our previous work [2]. Note that as the consensus parameter ζ approaches 1, the variance decreases. This is observable in our data.

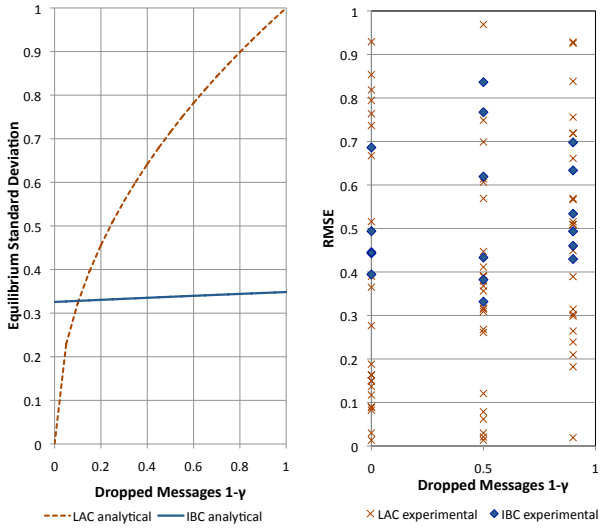
We numerically solve the differential equation (8) for the mean $\langle \hat{\boldsymbol{x}} \rangle$ and second moment $\langle \hat{\boldsymbol{x}} \hat{\boldsymbol{x}}^T \rangle$ for LAC and IBC for $\gamma = 0.5$ and plot them below. Figure 3(a) and 4(a) show the analytical solutions for the expected value of the estimate for success fraction $\gamma = 0.25$. An example robot trajectory for one experimental run is plotted in orange. Note that for LAC, Figure 3(a), the single trajectory converges to an incorrect steady state value, while the single robot for IBC, Figure 4(a), oscillates about the correct value.

C. Analysis

We compare algorithms by calculating the standard deviations. Figure 1 shows the standard deviation at equilibrium of LAC to IBC as the function of communications failures, $1 - \gamma$. The numerical solutions are found at the steady-state of the mean estimate dynamics (10) and the second moment dynamics. Effectively, this evaluates the standard deviation across multiple executions of each algorithm, not within a single run. Note that the standard deviation of IBC is almost unaffected by communications loss, but the standard deviation of LAC approaches the maximum value – the initial conditions.

V. Simulated Results

We simulated the system using the *Stochastic Simulation Algorithm* [22]. At every time step in the sim-



(a) Numerical analysis. (b) Experimental results.

Fig. 1. **(a)** LAC compared to IBC numerically. We plot the standard deviation as a function of communication failures $(1-\gamma)$ for 20 robots. The red (dashed) line is the numerical standard deviation for LAC and the blue (solid) line is the standard deviation for IBC. **(b)** LAC compared to IBC experimentally. We conducted a total of 96 runs. The experimental data shows a much larger variance for the LAC RMSE than the IBC RMSE.

ulation, a pair of robots is chosen to interact and update their estimates. This produces a well-mixed system, *i.e.* interactions between any pair is uniform. Two representative trajectories for the simulated results are shown in Figure 3(b) and Figure 4(b), for LAC and IBC respectively. The communications success rate is $\gamma = 0.25$. LAC converged with variance 0 to a value that is not the correct value while IBC oscillates about the correct value with a finite variance. The results from simulation are confirmed by experimental results.

VI. Experimental Results

The SwarmBot robot platform [23] was used to validate algorithm performance. The robots are autonomous, using only local computation and communication to run the algorithms. Each robot has a infra-red communication system that allows robots to communicate to neighbors within approximately a 1.0 meter radius. This produces multi-hop networks within the confines of our experimental workspace, which is a 2.43 m $\times$ 2.43 m (8′ $\times$ 8′) square.

We generated dynamic network topologies by moving the robots randomly throughout the workspace. Since their communication range is smaller that the workspace dimension, this will force frequent changes in neighbors, producing random interactions. The robots drive straight until they contact an obstacle, then use their bump sensors to estimate the angle of incidence and rotate to "reflect" themselves back into the environment. Each trial was 5 minutes long and initialized with 20 robots. Robots would occasionally run out of batteries during the experiment and were not replaced.

We can calculate the effect of losing a robot on the global estimate. As shown in previous work [2], the equilibrium estimate is

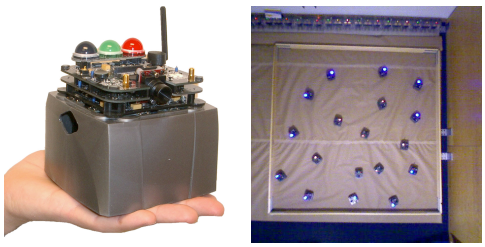
$$\langle \hat{\boldsymbol{x}} \rangle^* = (1-\zeta)\left(\binom{n}{2}\boldsymbol{I} - \zeta \boldsymbol{A}\right)^{-1} \boldsymbol{B}\boldsymbol{q}.$$

Suppose that one robot has a dead battery and the rest of the robots still believe that there are n robots in the system, but instead there are actually $n-1$ robots. The error of the equilibrium is $\langle \hat{\boldsymbol{x}} \rangle_{actual} - \langle \hat{\boldsymbol{x}} \rangle_{measured} = e\mathbb{1}$. We can solve this error as

$$e = (1-\zeta)\left(\binom{n}{2}\boldsymbol{I} - \zeta \boldsymbol{A}\right)^{-1} (\boldsymbol{B}_{actual} - \boldsymbol{B}_{measured})\boldsymbol{q}$$

where $\boldsymbol{B}_{actual} = \frac{n-2}{n-1}\mathbb{1}\mathbb{1}^T$ and $\boldsymbol{B}_{measured}\frac{n-1}{n}\mathbb{1}\mathbb{1}^T$. As the number of robots increases, the effect of an incorrect coefficient decreases. For example, parameters for a representative experiment are $n = 20$ robots, two robots with state 1, $a = 0.5$ and $\zeta = 0.95$. If there are 19 active robots at the end of an experiment, the error is 0.27%. For 18 active robots, the error is 0.58% and so on.

Both algorithms require pairwise updates between neighbors, which required a communications handshaking protocol. First, if a robot is not updating, it selects a neighbor at random and transmits a "partner request" message. Then, if that neighbor is not updating, it replies with a "partner acknowledge" message and runs the update rule. Upon receipt of the acknowledgement, the original robot runs its update rule with probability γ. This success probability parameter γ allows us to simulate communication losses much larger than the actual loss of the system, which is less than 1% [23].



(a) A SwarmBot. (b) An experiment.

Fig. 2. **a.** Each SwarmBot has an infra-red communication and localization system which enables neighboring robots to communicate and determine their pose relative to each other. The three lights on top are the main user interface, and let a human determine the state of the robot from a distance. The radio is used for data collection and software downloads. **b.** Each trial started with 20 robots, but often robots would run out of batteries during a trial. These robots were not replaced until the next trial.

A total of 95 experimental runs were performed with the variables $\gamma = \{0.1, 0.5, 1\}$ and $\zeta = \{0.95, 1.0\}$. Figures 3(a) and 4(a) each show one sample trajectory from the experimental data plotted with the analytical

results. We estimate $k = 0.22$ from our experimental results and see that our data compares nicely to our analytical solution. We compute the root mean square errors (RMSE) for experiments, where RMSE is $\sqrt{\langle \hat{x} - x \rangle^2}$. Figure 1(b) shows the experimental results for 96 trials where the RSME of LAC to IBC are plotted as a function of communications failures, $1 - \gamma$. The data supports the analytical results and simulations. The LAC trials have a much larger RSME across multiple runs. Although LAC produces runs with small RSME, it does not do so consistently. The IBC algorithm produces more consistent RSME, even at which is 90% message loss.

VII. Conclusions and Future Work

In this paper, we addressed agreement in the event of severe undetected message loss. We use input-based consensus (IBC) to weight initial and current agent values in order to recover the initial state. Thus, the mean estimate of the robots that is recovered is close to the actual mean in the event of message loss. We compare this algorithm to linear average consensus (LAC) by examining the resultant differential equations, simulating the robots, and performing experiments on hardware. In all cases, IBC provides robustness to message loss that LAC cannot, but at the cost of a bounded variance for minimally lossy networks.

Looking forward, we would like to find bounds on performance and derive an expression for the error due to dropped messages. In this paper, we have only demonstrated recovery of a mean initial condition, but this work can be extended to tracking a changing quantity. Our previous work leads [2], [24] us to believe that agreement upon the mean is sufficient for control. Thus, we can extend this work to consensus and control for coordination, such a task assignment, in the presence of lossy communications.

Acknowledgements

We thank Eric Klavins and Nils Napp for helpful discussions and Alvin Chou for extensive data collection. James McLurkin is supported by DARPA's ASSIST program (contract number NBCH-C-05-0137). Fayette Shaw is supported by NSF Grant #0735953: EFRI ARESCI: Controlling the Autonomously Reconfiguring Factory.

References

[1] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked Multi-Agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[2] F. Shaw and E. Klavins, "Distributed estimation and control in stochastically interacting robots," in *Proceedings of the 47th IEEE Conference on Decision and Control.*, pp. 1895–1901, Dec. 2008.

[3] W. Ren, R. Beard, and E. Atkins, "A survey of consensus problems in multi-agent coordination," in *Proceedings of the 2005 American Control Conference.*, vol. 3, pp. 1859—1864, 2005.

[4] F. Bullo, J. Cortes, and S. Martinez, *Distributed Control of Robotic Networks*. Princeton, NJ: Princeton University Press, Sept. 2008. Manuscript under contract. Electronically available at http://coordinationbook.info.

[5] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. 31, pp. 803–812, Sept. 1986.

[6] R. Olfati-Saber, "Distributed kalman filter with embedded consensus filters," in *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference*, pp. 8179–8184, 2005.

[7] R. Olfati-Saber and J. Shamma, "Consensus filters for sensor networks and distributed sensor fusion," in *Proceedings of 44th IEEE Conference on Decision and Control and European Control Conference*, pp. 6698–6703, 2005.

[8] W. Ren, R. Beard, and D. Kingston, "Multi-agent kalman consensus with relative uncertainty," in *American Control Conference, 2005. Proceedings of the 2005*, pp. 1865–1870 vol. 3, 2005.

[9] L. Xiao, S. Boyd, and S. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computing*, vol. 67, no. 1, pp. 33–46, 2007.

[10] M. Huang and J. H. Manton, "Coordination and consensus of networked agents with noisy measurements: Stochastic algorithms and asymptotic behavior," *SIAM Journal on Control and Optimization*, vol. 48, no. 1, pp. 134—161, 2009.

[11] M. Huang and S. Dey, "Stability of kalman filtering with markovian packet losses," *Automatica*, vol. 43, pp. 598–607, Apr. 2007.

[12] M. Huang and S. Dey, "Kalman filtering with markovian packet losses and stability criteria," in *Decision and Control, 2006 45th IEEE Conference on*, pp. 5621–5626, 2006.

[13] K. Hosokawa, I. Shimoyama, and H. Miura, "Dynamics of self-assembling systems: analogy with chemical kinetics," *Artif. Life*, vol. 1, no. 4, p. 413427, 1994.

[14] E. Klavins, S. Burden, and N. Napp, "Optimal rules for programmed stochastic Self-Assembly," (Philadelphia, PA), 2006.

[15] E. Klavins, "Programmable Self-Assembly," *Control Systems Magazine*, vol. 24, pp. 43–56, Aug. 2007.

[16] L. Matthey, S. Berman, and V. Kumar, "Stochastic strageties for a swarm robotic assembly system," in *Proceedings of the International Conference on Robotics and Automation*, 2009.

[17] Y. Hatano and M. Mesbahi, "Agreement over random networks," *IEEE Transactions on Automatic Control*, vol. 50, pp. 1867–1872, Nov. 2005.

[18] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE ACM Transactions on Networking*, vol. 52, no. 6, pp. 2508–2530, 2006.

[19] A. Tahbaz-Salehi and A. Jadbabaie, "Necessary and sufficient conditions for consensus over random independent and identically distributed switching graphs," *Decision and Control, 2007 46th IEEE Conference on*, pp. 4209–4214, Dec. 2007.

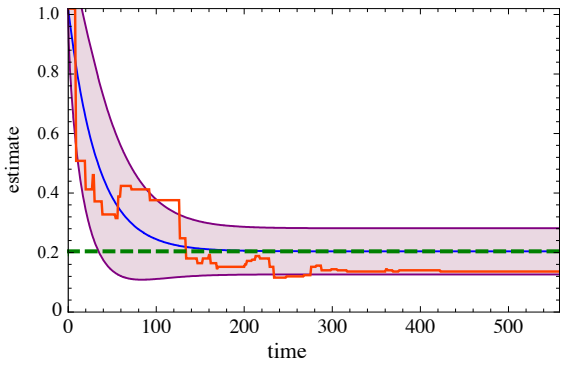
[20] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, pp. 988–1001, 2003.

[21] J. P. Hespanha, "Modeling and analysis of stochastic hybrid systems," *IEE Proceedings of Control Theory and Applications*, vol. 153, pp. 520–535, Sept. 2006.

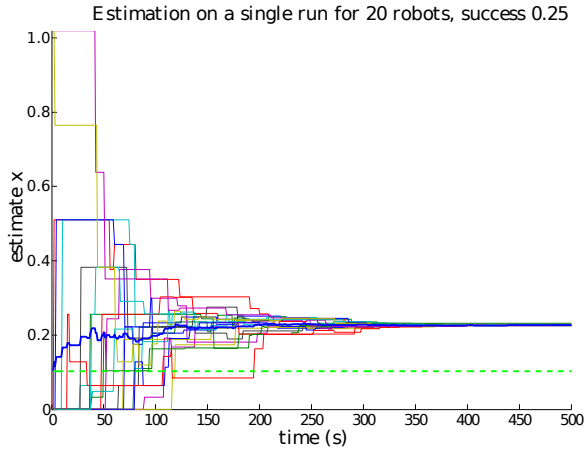
[22] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *The Journal of Physical Chemistry*, 1977.

[23] J. McLurkin, *Analysis and Implementation of Distributed Algorithms for Multi-Robot Systems*. PhD thesis, Massachusetts Institute of Technology, Cambridge, June 2008.

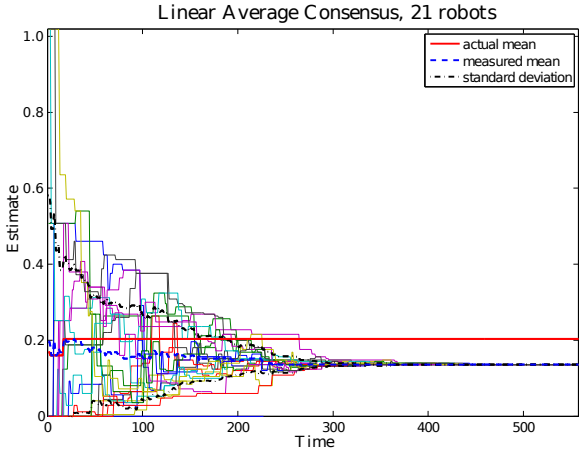
[24] F. Shaw and E. Klavins, "Distributed estimation and state assignment for stochastically interacting robots," in *IFAC Workshop on Networked Systems and Controls, Venice*, Sept. 2009.



(a) Analytical solution (blue) of first moment dynamics of the estimate with LAC parameters and standard deviation window (purple). Desired mean (green dashed) plotted with experimental data for one robot (orange).

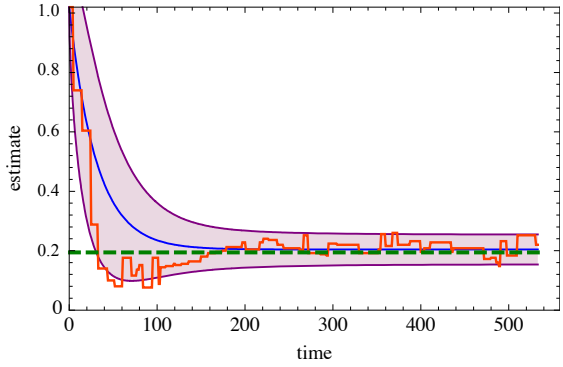


(b) Simulated Data for 20 robots using LAC. The green dashed line indicates the desired mean state. Light lines indicate individual robot trajectories and the solid blue line is the average of trajectories.

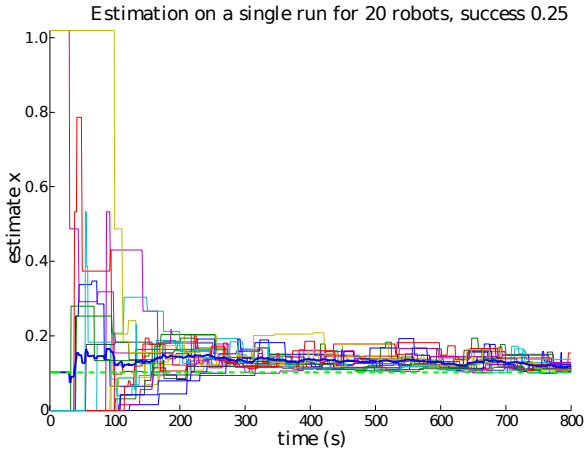


(c) Experimental Data for 21 robots using LAC. The solid red line indicates the desired mean state. Light lines indicate individual robot trajectories and the solid blue line is the average of trajectories.

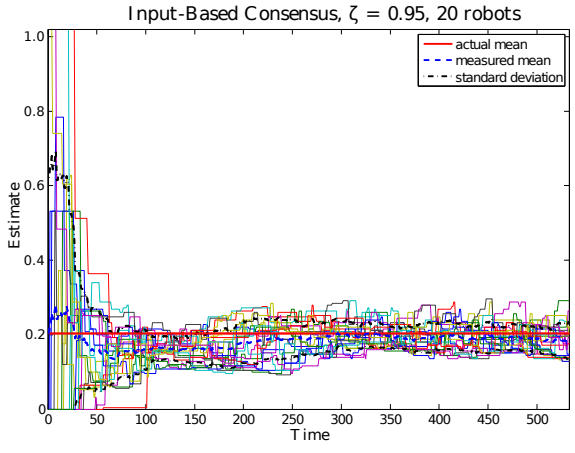
Fig. 3. Analytical, simulated, and experimental data for LAC ($\zeta = 1, \gamma = 0.25$). Figure 3(a) shows the analytical solution of the first moment dynamics of the estimate. Figure 3(b) shows robots simulated using the Stochastic Simulation Algorithm. Figure 3(c) shows data collected from the Swarmbot testbed.



(a) Analytical solution (blue) of first moment dynamics of the estimate with IBC parameters and standard deviation window (purple). Desired mean (green dashed) plotted with experimental data for one robot (orange).



(b) Simulated Data for 20 robots using IBC. The green dashed line indicates the desired mean state. Light lines indicate individual robot trajectories and the solid blue line is the average of trajectories.



(c) Experimental Data for 20 robots using IBC. The red line indicates the desired mean state. Light lines indicate individual robot trajectories and the solid blue line is the average of trajectories.

Fig. 4. Analytical, simulated, and experimental data for IBC $\zeta = 0.95, \gamma = 0.25$. Figure 4(a) shows the analytical solution of the first moment dynamics of the estimate. Figure 4(b) shows robots simulated using the Stochastic Simulation Algorithm. Figure 4(c) shows data collected from the Swarmbot testbed.