Worst-Case Optimal Average Consensus Estimators for Robot Swarms

Matthew L. Elwin Randy A. Freeman Kevin M. Lynch

Abstract— **Average consensus estimators enable robots in a communication network to calculate the mean of their local inputs in a distributed manner. Many distributed control methods for robot swarms rely on these estimators. The performance of such estimators depends on their design and the network topology. For mobile sensor networks, this topology may be unknown, making it difficult to design average consensus estimators for optimal performance. We introduce a design method for proportional-integral (PI) average consensus estimators that decouples estimator synthesis from network topology. This method also applies to the more general internal model (IM) estimator, yielding extended PI estimators that improve convergence rates without increasing communication costs. In simulations over many geometric random graphs, the extended PI estimator consistently reduces the estimation error settling time by a factor of five.**

I. INTRODUCTION

Average consensus estimators allow groups of robots to agree on the mean of their individual inputs using local communication in a network. These estimators form the basis for many multi-robot coordination techniques because they allow robots to share global information without requiring a central computer or all-to-all communication [1]–[4]. One such estimator, the proportional-integral (PI) estimator, allows the robots to compute the average of constant signals with zero steady-state error. This estimator's robustness to initialization error makes it particularly useful: events such as robot failure, network switching, or step-wise input changes cause re-initialization disturbances that the PI estimator asymptotically rejects [5], [6]. Due to these properties, PI estimators and similar robust average consensus estimators have been successfully used for tasks requiring multi-robot cooperation such as environmental modeling, formation control, and decentralized simultaneous localization and mapping (SLAM) [1], [6], [7]. For example, in the multi-robot SLAM algorithm of [7], average consensus estimators allow individual robots to merge their local maps into a global map. Although these estimators have been studied in both continuous and discrete time, this paper exclusively discusses the discrete-time case.

One potential problem with consensus estimators, including the PI estimator, is convergence speed. Until the estimator converges, the robots operate with inaccurate information. Precise estimator convergence rates depend on the estimator and the network structure, particularly the network's algebraic connectivity [5], [8]–[10]. This connectivity measure depends on the network topology and the weights of individual communication links. For known network topologies, weights can be chosen optimally to maximize the estimator's convergence rate [9], [10]. For robot swarms, however, the network is often unknown and may vary during deployment.

We present a method for designing discrete-time PI estimators that minimize the worst-case settling time when the network topology is unknown. Our method decouples estimator design from specific network structures by using the separation principle introduced in [11]. This principle applies not only to PI estimators, but also to a more general class of robust average consensus estimators known as internal model (IM) estimators [12]. These estimators can be designed to provide robust averaging of time-varying signals such as ramps and sinusoids. Viewing the PI estimator from the internal model perspective allows us to create optimal extended PI estimators that greatly improve performance without additional communication. The resulting estimators can be implemented on robots as linear state equations.

Our design approach formulates estimator synthesis as a robust control problem, distilling the effect of the network into a single uncertain parameter [11]. We automate the synthesis process by using generic global optimization routines to minimize the estimator's settling time [13]. Although these numerical algorithms may not necessarily reach the global minimum, they result in estimators that have consistently good performance across a wide range of networks. Our simulation results indicate that our design method results in optimal extended PI estimators that converge five times faster than the optimal PI estimator over a large variety of geometric random graphs.

We organize the paper as follows. Section II reviews the PI estimator and shows how it fits into the IM estimator framework. Section III discusses the separation principle and our estimator synthesis method. Section IV presents our simulations and shows how adding additional states to the PI estimator can improve its performance. We conclude with a discussion of future work.

II. THE ESTIMATORS

A. Average Consensus

Consider N robots in a communication network represented by a weighted undirected graph. Robots i and j can communicate if an edge exists between vertex i and j in the graph. The edges have weights $a_{ij} = a_{ji} > 0$, with $a_{ij} = 0$ if i and j cannot communicate. We represent the graph by

This work is supported by the Office of Naval Research, Grant N00014-13-1-0331. The authors are with the Department of Mechanical Engineering (Elwin and Lynch), the Northwestern Institute on Complex Systems (Freeman and Lynch), and the Department of Electrical Engineering and Computer Science (Freeman), Northwestern University, Evanston, IL 60208 USA. Emails: elwin@u.northwestern.edu, freeman@eecs.northwestern.edu, kmlynch@northwestern.edu.

its Laplacian matrix $L \in \mathbb{R}^{N\times N}$, with

$$[L]_{ij} \triangleq \begin{cases} \sum_{j=1}^{N} a_{ij}, & i = j \\ -a_{ij}, & \text{otherwise} \end{cases}. \quad (1)$$

The Laplacian is positive-semidefinite and always singular. Let λ_2 and λ_N be the second smallest and largest Laplacian eigenvalues. The communication graph is connected if and only if $\lambda_2 > 0$; we assume that this connectivity condition holds.

When the underlying network topology is known, the edge weights a_{ij} may be chosen to maximize the convergence rate [9], [10]. In the absence of such knowledge, we can choose a systematic weight assignment scheme. Although numerous weighting methods exist for average consensus estimators, we will use inverse-sum-degree weighting because it provides a known bound for λ_N, in this case $\lambda_N \leq 1$ [14]. With inverse-sum-degree weighting, $a_{ij} = \frac{1}{\deg(i)+\deg(j)}$, where $\deg(i)$ denotes the degree of robot i. This weighting scheme can be implemented in a distributed manner: each robot transmits its degree to its neighbors in the network.

Every robot has an estimate of the average $y_i(k)$ and an input $\phi_i(k)$. All robots' estimates and inputs are stacked into the vectors $y(k)$ and $\phi(k)$. The goal of an average consensus estimator is for every robot's steady-state output to equal the average of every robot's input; that is

$$\lim_{k\to\infty} y(k) = \frac{1}{N} 1_N 1_N^T \phi(k), \quad (2)$$

where 1_N is a column vector of N ones. Robust average consensus estimators, like the PI estimator discussed in the next section, achieve this goal even when initialized improperly. This robustness property allows the estimator to converge with zero-steady state error for constant inputs, even if events such as step input changes or robot failure occur. For slowly changing inputs, the estimator will have a small tracking error [1].

B. Proportional-Integral Estimator

To implement the PI estimator each robot executes the following estimation equations

$$\begin{aligned} y_i(k+1) &= y_i(k)(1-\gamma) + \gamma\phi_i(k) \quad (3) \\ &\quad - k_p[L]_i y(k) + k_i[L]_i \eta(k) \\ \eta_i(k+1) &= \eta_i(k) - k_i[L]_i y(k), \quad (4) \end{aligned}$$

where $[L]_i$ is the i-th row of the Laplacian, $\eta_i(k)$ is an auxiliary variable, k_p is the proportional gain, k_i is the integral gain, and γ is the forgetting factor [5]. Each robot transmits $y_i(k)$ and $\eta_i(k)$ to its neighbors in the communication network. Intuitively, the proportional term $k_p[L]_i y(k)$ drives the robot's estimate closer to that of its neighbors, while the integral term $k_i[L]_i\eta(k)$ ensures zero steady-state error. The forgetting factor γ causes the initial state of the estimator to decay, decoupling the steady-state response from the initial conditions. If the communication graph is connected, the PI estimator's dynamics will converge with zero steady-state error [5], [6]. Note that the estimator's dynamics depend on both the parameters (k_p, k_i, and γ) and the Laplacian L.

C. Internal Model Estimator

The IM estimator is a robust average consensus estimator that converges with zero steady-state error for certain time-varying inputs [12]. This estimator requires every robot to have a common input model $d(z)$, derived from the z-transform of the anticipated input. All inputs follow

$$\phi_i(z) = \frac{c_i(z)}{d(z)}, \quad (5)$$

where $c_i(z)$ can vary between robots and is unknown. Ramp inputs with slope m_i, for example, have $\phi_i(k) = m_i k$, $\phi_i(z) = \frac{m_i z}{(z-1)^2}$, and $d(z) = (z-1)^2$. For the PI estimator, which converges with zero steady-state error for step inputs, $d(z) = z - 1$ and $c_i(z) = q_i z$, where q_i is constant.

To derive the IM estimator, we introduce the transfer functions $h(z)$, $g(z)$, and $k(z)$ and rewrite the PI estimator:

$$y_i(z) = h(z)(\phi_i(z) - k(z)[L]_i y(z) - [L]_i\eta(z)) \quad (6)$$

$$\eta_i(z) = g(z)[L]_i y(z). \quad (7)$$

For the PI estimator,

$$h(z) = \frac{\gamma}{z-1+\gamma}, \quad (8)$$

$$g(z) = \frac{k_i^2}{\gamma(z-1)}, \quad (9)$$

and

$$k(z) = \frac{k_p}{\gamma}, \quad (10)$$

whereas for the IM estimator $h(z)$ and $g(z)$ must satisfy

$$h(z) = \frac{\frac{k_h(z)}{d(z)}}{1 + \frac{k_h(z)}{d(z)}} \quad (11)$$

and

$$g(z) = \frac{k_g(z)}{d(z)}, \quad (12)$$

where $k_h(z)$ and $k_g(z)$ are transfer functions that the designer chooses [11], [12]. The designer also must choose $k(z)$, but unlike $g(z)$ and $h(z)$, its structure is not prescribed. Note that the PI estimator is an IM estimator.

To analyze PI and IM estimators we define the global error

$$e(z) = y(z) - \frac{1}{N} 1_N 1_N^T \phi(z), \quad (13)$$

so that when the estimator tracks the average of the inputs, the error is zero. When all the robots implement an IM estimator according to (6) and (7) the error dynamics become

$$e(z) = P(z, L)\phi(z), \quad (14)$$

where

$$P(z,L) = (I + h(z)g(z)L^2 + k(z)h(z)L)^{-1}h(z) - \frac{1}{N}1_N 1_N^T. \quad (15)$$

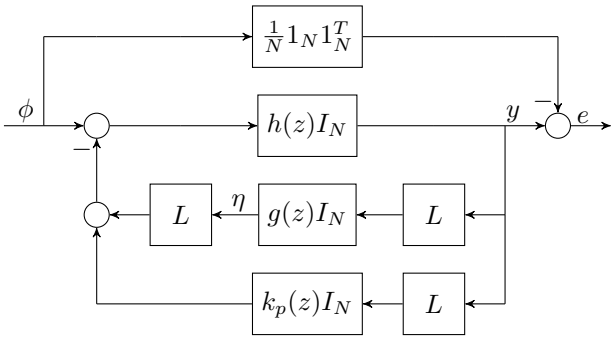

Fig. 1. Block diagram of the global error dynamics. The $N \times N$ identity matrix I_N emphasizes that every robot implements $h(z)$, $g(z)$, and $k_p(z)$.

Fig. 1 provides a block diagram of the global error dynamics. We explicitly show that $P(z, L)$ depends on L to emphasize that its value depends on a specific graph.

For the global error to reach zero in steady-state, the IM estimator transfer functions, and therefore the PI estimator gains, must satisfy the following stability theorem [12].

Theorem 1: A group of robots executing an IM estimator over a network with Laplacian L will achieve zero steady-state tracking error for inputs of the form of (5) if and only if the following conditions hold:

1) $h(z)$ and $k(z)$ are stable, and the unstable zeros of $h(z)$ and the unstable poles of $g(z)$ are disjoint.
2) $h(z) = \frac{\frac{k_h(z)}{d(z)}}{1+\frac{k_h(z)}{d(z)}}$
3) $g(z) = \frac{k_g(z)}{d(z)}$
4) $\frac{h(z)}{1+h(z)g(z)\lambda^2+h(z)k(z)\lambda}$ is stable for $\lambda \in \lambda_2 \dots \lambda_N$.

Additionally, to implement the estimator using a one-hop protocol, $h(z)$ and $g(z)$ must be strictly proper.

Proof: See [12] for a proof. ∎

We assume that $h(z)$ and $g(z)$ are strictly proper to avoid needing a two-hop protocol, which would require the robots to send information to the neighbors of their neighbors.

III. Synthesis

A. Separation Principle

Theorem 1 provides a guide for designing both PI and IM estimators. Note that the fourth condition of Theorem 1 relates global stability to the Laplacian's eigenvalues; this forms the basis of the separation principle and will allow us to decouple estimator synthesis from specific networks. We use Theorem 1 condition 4 to define the separated system as

$$T(z, \lambda) = \frac{h(z)}{1 + h(z)g(z)\lambda^2 + h(z)k(z)\lambda}. \tag{16}$$

The following proposition describes the separation principle, which was introduced in [11].

Proposition 1: The poles of $P(z, L)$ are the poles of $h(z)$ and the poles of $T(z, \lambda)$ for $\lambda = \lambda_2 \dots \lambda_N$.

Proof: See [11]. ∎

Note that if the poles of $T(z, \lambda)$ satisfy some condition for $\lambda_{\min} \leq \lambda_2 \leq \lambda \leq \lambda_N \leq \lambda_{\max}$, then, by Proposition 1, the poles of $P(z, L)$ also satisfy that condition for all $L \in L_{\text{set}}$, where L_{set} is the set of all Laplacians L that satisfy $\lambda_{\min} \leq \lambda_2(L) \leq \lambda_N(L) \leq \lambda_{\max}$. Additionally, the poles of $h(z)$ are independent of the network. We can therefore eliminate the dependence of Proposition 1 on specific Laplacian eigenvalues by choosing appropriate bounds $\lambda_{\min}$ and $\lambda_{\max}$. Such bounds may be known even when the specific graph is unknown. For instance, $\lambda_{\min}$ can be viewed as a minimum connectivity specification, and with inverse-sum-degree weighting $\lambda_{\max} \leq 1$ [14]. We discuss how to determine $\lambda_{\min}$ and $\lambda_{\max}$ further in Section IV.

B. The Problem

Our goal is to produce estimators that converge quickly. A good proxy for settling time in discrete systems is the spectral radius of the system matrix; in our case, $P(z, \lambda)$. For unknown graphs, we will use the separation principle of Proposition 1, which implies that

$$\rho(P(z, L)) = \max(\rho(h(z)), \max_{\lambda \in \lambda_2 \dots \lambda_N} \rho(T(z, \lambda))), \tag{17}$$

where $\rho(\cdot) = \max(|\text{poles}(\cdot)|)$ is the spectral radius.

Given a set of graphs L_{set} whose nontrivial Laplacian eigenvalues lie between $\lambda_{\min}$ and $\lambda_{\max}$, we define the worst-case spectral radius as

$$R_{\max} = \max_{\lambda_{\min} \leq \lambda \leq \lambda_{\max}} R(\lambda), \tag{18}$$

where

$$R(\lambda) = \max(\rho(h(z)), \rho(T(z, \lambda))). \tag{19}$$

Note that $\rho(P(z, L)) \leq R_{\max}$ for all graphs whose eigenvalues fall between $\lambda_{\min}$ and $\lambda_{\max}$; thus $R_{\max}$ provides a bound on the worst-case estimator performance.

Overall, we would like to minimize the worst-case estimator performance, leading to the following robust control problem, where we find an IM estimator that minimizes the maximum of the spectral radius over all of the eigenvalues:

$$\min_{k_h(z), k_g(z), k(z)} R_{\max}. \tag{20}$$

For the PI estimator, this problem simplifies to minimizing over the PI estimator parameters:

$$\min_{k_p, k_i, \gamma} R_{\max}. \tag{21}$$

C. Parameters

For IM estimator synthesis, the parameter space of (20) is infinite because $k_h(z)$, $k_g(z)$, and $k(z)$ can have any number of poles and zeros. We convert this problem into a finite optimization by fixing the order of these design transfer functions. This fixed order becomes an input into the synthesis method, allowing the designer to control the order of the resulting estimator.

The minimum number of parameters used for estimator synthesis depends on r_d, the degree of the internal model $d(z)$. A minimal order IM estimator is the internal model estimator whose dynamics have the lowest order that can still satisfy Theorem 1. In a minimal order IM estimator,

$$\text{order}(h(z)) = \text{order}(g(z)) = r_d \tag{22}$$

and $k(z)$ is a static gain; therefore, the minimal estimator order is $2r_d$. To maximize design freedoms, the relative degree of $h(z)$ and the relative degree of $g(z)$ should be one. Therefore, for the minimal order IM estimator,

$$\deg(\mathrm{num}(k(z))) = 0 \tag{23}$$

$$\deg(\mathrm{num}(k_h(z))) = \deg(\mathrm{num}(k_g(z))) = r_d - 1, \tag{24}$$

and

$$\deg(k_h(z)) = \deg(k_g(z)) = \deg(k(z)) = 1. \tag{25}$$

When $r_d > 1$, $k_h(z)$ and $k_g(z)$ will be improper; however, in the estimator implementation, these transfer functions are combined with $d(z)$ to yield strictly proper $h(z)$ and $g(z)$.

Minimal order estimators can be expanded by adding pole-zero pairs to $k_h(z)$, $k_g(z)$, and $k(z)$. Let r_h, r_g, r_k be the number of pole-zero pairs added to the minimal $k_h(z)$, $k_g(z)$, and $k(z)$ respectively. Assuming that the denominators of $k_h(z)$, $k_g(z)$, and $k(z)$ are monic, the estimator parameters can be expressed as the vector $x \in \mathbb{R}^{2(r_d+r_h+r_g+r_k)+1}$, where each element of x corresponds to a coefficient in one of the synthesized transfer functions.

The PI estimator is a minimal order IM estimator because $\deg(z-1) = 1$ and it has two states. We will show that adding extra poles and zeros can greatly increase the PI estimator's performance. Although every pole-zero pair slightly increases memory and computation requirements, communication costs remain constant because robots transmit their outputs, not their internal states.

D. Optimization

Given the parameterization above, the inputs to the minimax problem (20) are an initial parameter vector, the internal model $d(z)$, an eigenvalue range $[\lambda_{\min}, \lambda_{\max}]$, and the numbers of additional pole-zero pairs. Solving this problem, however, is difficult because it is non-smooth and non-convex. Nevertheless, generic global optimization routines yield estimators that settle quickly over a range of networks.

We divide the problem into two parts: the inner maximization (18) and the outer minimization (20). The inner problem is a one-dimensional optimization over λ. We use the deterministic DIRECT-L algorithm from NLopt [15] limited to 38 objective function evaluations, which provides a good compromise between computation time and accuracy. This procedure needs to terminate quickly because the inner maximization must be solved to evaluate the objective function of the outer minimization. As the maximum over the range of λ may occur at the endpoints, we explicitly compare the output of the algorithm with $\rho(T(z, \lambda_{\min})$ and $\rho(T(z, \lambda_{\max}))$.

To solve the outer minimization problem (20) we use CRS2, the NLopt-provided implementation of a controlled random search with local mutation [16]. This algorithm is stochastic; therefore we run the optimization multiple times and take the best result. We chose this method after testing a broad array of global optimization algorithms.

The algorithms we employ cannot guarantee that they return global extrema. Let $R'_{\min}$ and $R'_{\max}$ be the results of the outer minimization and inner maximization algorithms, and $R_{\min}$ and $R_{\max}$ be the corresponding global extrema. For now, assume that the inner maximization finds the global maximum such that $R'_{\max} = R_{\max}$. If the outer minimization returns a sub-optimal point such that $R'_{\min} > R_{\min}$, the resulting estimator will be sub-optimal. This sub-optimal estimator, however, will provide the performance guarantee $\rho(P(z, L)) < R'_{\min}$ for all $L \in L_{\text{set}}$ and will likely be useful in practice. The effects of the inner maximization returning a sub-maximal value, however, are more deleterious. If the inner maximization returns a sub-optimal value, $R'_{\max} < R_{\max}$, $R'_{\min}$ no longer provides a performance guarantee because it does not bound $\rho(P(z, L))$ for all $L \in L_{\text{set}}$. The failure of the inner maximization effectively causes the outer minimization to ignore any λ for which $R(\lambda) > R'_{\max}$.

Fortunately, in our experience, we can handle these issues. The inner maximization is one-dimensional; therefore, after synthesizing an estimator we can sample $R(\lambda)$ at numerous points to determine an upper bound R_{bnd} on $R(\lambda)$. Using this approach after synthesizing an estimator allows us to determine the actual performance bound even when the inner maximization returns a sub-optimal point. Theoretically, this gridding approach could be used for the inner optimization; however, this would significantly increase computation time. Additionally, the outer minimization seems to produce designs that make $R(\lambda)$ flat over a significant portion of the λ range; although this makes the inner optimization more difficult, it also means that $R(\lambda) > R'_{\max}$ for only a small interval, and hence a small proportion of graphs. This tendency matches our intuition about the problem: the outer minimization tries to minimize equally over all values of λ, resulting in an estimator that performs similarly over all values of λ. The examples section includes a situation where $R(\lambda) > R'_{\max}$, and shows that the resulting estimator maintains good performance across a wide range of graphs.

IV. Examples

We have applied the estimator synthesis method to a sample decentralized estimation design problem. First, we generate a large set of networks, designed to roughly model wireless sensor networks. We then synthesize several estimators, analyze them, and simulate them over these networks. The analysis and simulations demonstrate the advantages of applying IM estimator synthesis techniques to PI estimators. Although we did not perform experiments, real robots have accomplished tasks using PI estimators [7].

A. The Networks

The design technique proposed in this paper relies upon knowing an upper and lower bound for the Laplacian eigenvalues. To determine these bounds for this example, we generate several random graphs and find the maximum and minimum eigenvalues from the resulting graph set. We use geometric random graphs as a model for the networks formed during the operation of a wireless sensor network with

radius-limited communication [17]. The robots are randomly distributed over the unit square; two agents are neighbors if they are within a $\frac{1}{4}$ radius of each other. Overall, we generated 2210 networks ranging in size from 80 to 300 robots.

After generating the underlying network topology, we applied the inverse-sum-degree weighting scheme to the edges. Overall, the graphs in this set have eigenvalues ranging from 0.0084 to 0.7626. Using these bounds as a guide, we set the minimum connectivity as $\lambda_{\min} = 0.0080$. Inverse sum degree weighting guarantees that $\lambda_N \leq 1$; therefore, for half of the designs we set $\lambda_{\max} = 1$. Based on the bounds of the randomly generated graph set, we also design estimators with $\lambda_{\max} = 0.763$. The estimators designed with $\lambda_{\max} = 1$ will be stable for any inverse-sum-degree weighted graph, whereas those designed with $\lambda_{\max} = 0.763$ may become unstable for graphs with $\lambda > \lambda_{\max}$; however, these estimators should perform better within the eigenvalue range.

B. The Estimators

We synthesize six estimators; three for $\lambda_{\max} = 1$, which guarantees robust performance over any graph with inverse-sum-degree weighting, and three for $\lambda_{\max} = 0.763$, which guarantees performance for all graphs in the set we generated. Each estimator comes from the best result after running the global optimization algorithm 30 times. We also compute the true worst-case performance bound R_{bnd} for each estimator by evaluating $R(\lambda)$ over a grid of of λ values spaced at intervals of 0.0001. For every optimization, we set all initial design variables to one and bound the parameters by -1000 and 1000. Tables I and II present a summary of the synthesis results for the standard PI estimator, the PI estimator with an extra state in $g(z)$ (PI-g), and the PI estimator with extra states in $g(z)$ and $k(z)$ (PI-gk). We chose these particular combinations of extra pole-zero pairs because they offered the best performance.

First, we discuss the standard PI estimators. For $\lambda_{\max} = 1$, the PI estimator gains are $k_p = 3.976$, $k_i = 1.992$, and $\gamma = 0.008$, whereas for $\lambda_{\max} = 0.763$ the gains are $k_p = 5.20$, $k_i = 2.61$, and $\gamma = 0.0105$. The $\lambda_{\max} = 1$ estimator performs slightly worse than the $\lambda_{\max} = 0.763$ estimator, but remains stable for all inverse-sum-degree weighted graphs.[1] This extra stability is important if the graph becomes disconnected: although the eigenvalues will be less than one, they may increase beyond the specification, destabilizing the $\lambda_{\max} = 0.763$ estimators. The estimators designed with $\lambda_{\max} = 1$, however, remain stable and each subgraph will converge to the average of its connected component.

The PI estimator with $\lambda_{\max} = 0.763$ displays the largest disparity between the spectral radius returned by the optimization $R'_{\max}$ and the actual spectral radius R_{bnd}. This is a result of the inner maximization problem returning a sub-optimal value. Fig. 2 shows that the potential upper bound of $R'_{\max}$ will only be violated for a narrow range

[1] A rough estimate of the ratio between settling times for systems with spectral radii of ρ_1 and ρ_2 is $\frac{\log(\rho_1)}{\log(\rho_2)}$; therefore small differences in the spectral radius can amount to large differences in settling time.

	PI	PI-g	PI-gk
$k_h(s)$	0.00798	0.0471	0.0634
$k_g(s)$	497	$\frac{154(z-0.346)}{z-0.797}$	$\frac{148(z+0.261)}{z-0.801}$
$k(s)$	498	97.6	$\frac{93.1(z+0.145)}{z-0.306}$
R_{bnd}	0.9940	0.9620	0.9424
Best $R'_{\min}$	0.9920	0.9620	0.9424
Mean $R'_{\min}$	0.9920	0.9768	0.9863
Std. $R'_{\min}$	$4e^{-15}$	0.0144	0.188

TABLE I

SYNTHESIS RESULTS FOR $\lambda_{\min} = 0.008$ AND $\lambda_{\max} = 1$.

	PI	PI-g	PI-gk
$k_h(s)$	0.0105	0.0587	0.0747
$k_g(s)$	650	$\frac{210(z-0.330)}{z-0.777}$	$\frac{188(z+0.277)}{z-0.787}$
$k(s)$	497	101	$\frac{97.1(z+0.142)}{z-0.327}$
R_{bnd}	0.9923	0.9527	0.9314
Best $R'_{\min}$	0.9895	0.9526	0.9314
Mean $R'_{\min}$	0.9895	0.9648	0.9667
Std. $R'_{\min}$	$5e^{-15}$	0.0166	0.0218

TABLE II

SYNTHESIS RESULTS FOR $\lambda_{\min} = 0.008$ AND $\lambda_{\max} = 0.763$.

of eigenvalues; therefore, for most graphs in the range, $R'_{\max}$ will be a more appropriate measure of performance than R_{bnd}. Thus, in this instance, while the failure of the maximization procedure does produce worse performance, its practical effect is limited: only one out of the 2210 graphs we test causes the spectral radius to violate the $R'_{\max}$ bound.

Another trend to notice from Tables I and II is that adding an extra state will greatly improve performance, and adding two states improves performance further. For example, with λ_1, the worst-case PI estimator spectral radius is 0.992, whereas the worst-case spectral radius for the PI-g estimator is 0.962. These worst-case spectral radii translate into an approximate settling time ratio of $\frac{\log(0.992)}{\log(0.962)} \approx \frac{1}{5}$, therefore we expect the PI-g estimator to converge nearly five times faster than the standard PI.

C. Simulations

We performed simulations over all 2210 graphs and, for each run, recorded the maximum 2% settling time over all of the robots. The inputs were steps with randomly generated

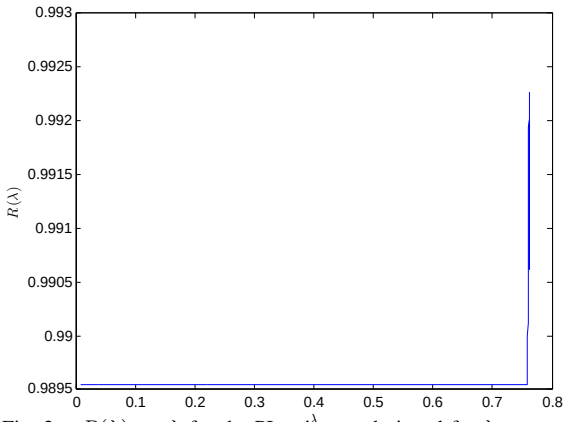

Fig. 2. $R(\lambda)$ vs. λ for the PI estimator designed for $\lambda_{\max} = 0.763$

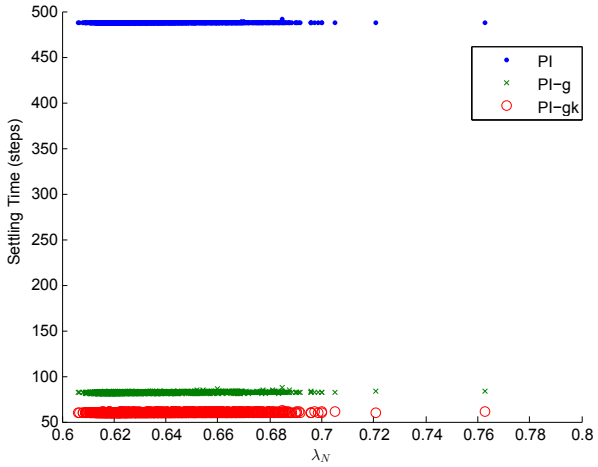

Fig. 3. Settling times for the estimators designed for $\lambda_{\max} = 1$.

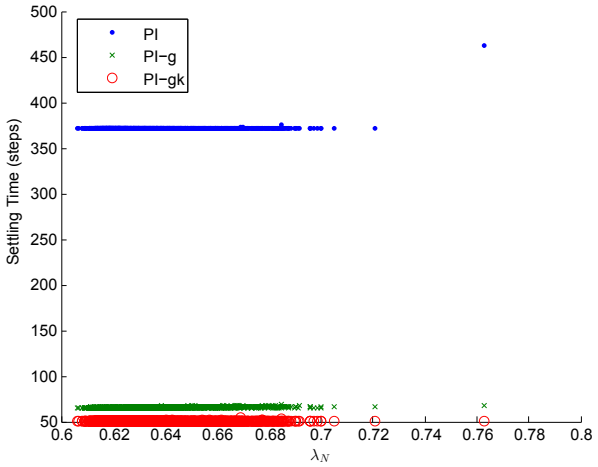

Fig. 4. Settling times for the estimators designed for $\lambda_{\max} = 0.763$.

magnitudes between 0 and 1. As depicted in Figs. 3 and 4, the settling time remains relatively flat over almost all of the graphs; thus the performance guarantees from the optimization problem generally hold. Another benefit of estimators designed using our method appears to be that they have consistent performance across a wide range of networks.

As predicted, the PI-g estimator dramatically outperforms the standard PI estimator, while the PI-gk estimator performs the best; however, its performance gains over the PI-g estimator are more modest. In most applications, the performance improvement from the PI-g estimator outweighs the extra memory and processing introduced by an extra state. Implementing the PI-gk estimator still provides a performance edge at low cost, but the benefits are less substantial.

The simulations show that the estimators designed for $\lambda_{\max} = 0.763$ offer a small improvement in settling time over those designed for $\lambda_{\max} = 1$. Additionally, for the standard PI estimator designed for the narrow eigenvalue range, a single graph performs worse than the others. This graph has a maximum eigenvalue of $\lambda_N = 0.7626$, a value that corresponds to the peak in Fig. 2. As discussed earlier, this peak results from the inner maximization missing the global minimum; however, as the simulations show, this affects only one of the 2210 graphs.

V. Conclusion

We have synthesized average consensus estimators that converge quickly over a range of networks. Future work will apply these enhanced PI estimators to multi-robot coordination problems such as formation control, target tracking, environmental modeling, and cooperative SLAM, investigating how faster consensus estimators improve multi-robot coordination strategies.

References

[1] K. M. Lynch, I. B. Schwartz, P. Yang, and R. A. Freeman, "Decentralized environmental modeling by mobile sensor networks," *IEEE Transactions on Robotics*, vol. 24, no. 3, pp. 710–724, June 2008.

[2] J. Cortés, "Distributed Kriged Kalman filter for spatial estimation," *IEEE Transactions on Automatic Control*, vol. 54, no. 12, pp. 2816–2827, Dec. 2009.

[3] 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*, Seville, Spain, Dec. 2005, pp. 8179–8184.

[4] C. Perterson and D. A. Paley, "Distributed estimation for motion coordination in an unknown spatiotemporal flowfield," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, no. AIAA-2011-6478, Portland, Oregon, August 2011.

[5] R. A. Freeman, P. Yang, and K. M. Lynch, "Stability and convergence properties of dynamic average consensus estimators," in *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, CA, Dec. 2006, pp. 398–403.

[6] R. Freeman, P. Yang, and K. Lynch, "Distributed estimation and control of swarm formation statistics," in *American Control Conference, 2006*, June 2006, p. 7 pp.

[7] R. Aragues, J. Cortes, and C. Sagues, "Distributed consensus on robot networks for dynamically merging feature-based maps," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 840–854, Aug. 2012.

[8] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *Automatic Control, IEEE Transactions on*, vol. 49, no. 9, pp. 1520 – 1533, Sept. 2004.

[9] P. Yang, R. Freeman, and K. Lynch, "Optimal information propagation in sensor networks," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, May 2006, pp. 3122 –3127.

[10] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems &; Control Letters*, vol. 53, no. 1, pp. 65 – 78, 2004.

[11] M. L. Elwin, R. A. Freeman, and K. M. Lynch, "A systematic design process for internal model average consensus estimators," in *Proceedings of the 52nd IEEE Conference on Decision and Control*, Florence, Italy, Dec. 2013, forthcoming.

[12] H. Bai, R. A. Freeman, and K. M. Lynch, "Robust dynamic average consensus of time-varying inputs," in *Proceedings of the 49th IEEE Conference on Decision and Control*, Dec. 2010, pp. 3104–3109.

[13] S. G. Johnson, "The NLopt nonlinear-optimization package." [Online]. Available: http://ab-initio.mit.edu/nlopt

[14] R. A. Freeman, T. R. Nelson, and K. M. Lynch, "A complete characterization of a class of robust linear average consensus protocols," in *Proceedings of the 2010 American Control Conference*, Baltimore, MD, July 2010, pp. 3198–3203.

[15] J. M. Gablonsky and C. T. Kelley, "A locally-biased form of the direct algorithm," *J. of Global Optimization*, vol. 21, no. 1, pp. 27–37, Sept. 2001.

[16] P. Kaelo and M. Ali, "Some variants of the controlled random search algorithm for global optimization," *Journal of Optimization Theory and Applications*, vol. 130, no. 2, pp. 253–264, 2006.

[17] M. Haenggi, J. Andrews, F. Baccelli, O. Dousse, and M. Franceschetti, "Stochastic geometry and random graphs for the analysis and design of wireless networks," *Selected Areas in Communications, IEEE Journal on*, vol. 27, no. 7, pp. 1029–1046, 2009.