A Systematic Design Process for Internal Model Average Consensus Estimators

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

***Abstract*— In the dynamic average consensus problem, agents in a communication network use information from their immediate neighbors to track the average of the group's time-varying inputs. Estimators based on the internal model principle solve this decentralized averaging problem with zero steady-state tracking error while providing robustness to network topology changes, agent failures, and communication faults. We develop a systematic process for designing these estimators. By formulating estimator synthesis as a robust control problem, we decouple the design process from specific networks. This formulation allows us to use an existing robust pole placement method to design estimators that meet performance specifications for a set of networks.**

I. Introduction

The average consensus problem, where agents in a network use local communication to track the mean of their individual inputs, applies to several distributed tasks, including sensor fusion [1]–[4], formation control [5], and map merging [6]. Non-robust dynamic average consensus estimators solve this problem for time-varying inputs, but only when initialized correctly [7]–[9]. This lack of robustness to initialization becomes problematic under network topology changes, the addition or removal of agents, and communication faults because these events introduce nonzero initial conditions into the estimator dynamics.

Robust dynamic average consensus estimators, unlike their non-robust counterparts, converge correctly regardless of the initial conditions [10]. One such estimator, based on the internal model principle [11], achieves robust dynamic average consensus by incorporating a linear time-invariant (LTI) model of its inputs into its dynamics [12]. In addition to its robustness, this internal model estimator's memory, computation, and communication requirements remain constant regardless of the number of agents. Additionally, implementing the estimator is relatively simple: each agent executes state equations and communicates using a one-hop protocol. The robustness, scalability, and simplicity of the internal model estimator make it a promising method for averaging time-varying inputs in a decentralized manner.

Designing an internal model estimator requires choosing two transfer functions that, together with the network, determine the estimator's performance. One method for choosing these transfer functions is given in [12]. This process uses root locus techniques and is not automated. To improve upon this manual approach, we introduce a separation principle that decouples the performance of the estimator from specific networks. This new formulation distills the network into a single uncertain parameter, allowing us to apply robust control theory [13] to the design problem. Specifically, we use the robust pole placement method of [14] to partially automate estimator synthesis and guarantee performance over a range of networks.

This paper proceeds as follows. In Section II we define the dynamic average consensus problem and the internal model estimator. We formulate the design problem, review the robust pole placement technique of [14], and explain the systematic design process in Section III. The examples and simulations of Section IV demonstrate the usefulness of this process by comparing the systematically and manually designed estimators. Section V discusses conclusions and future work.

A. Notation

The symbol $\star$ is s or z. A transfer function's poles are $\mathrm{poles}(\cdot)$. The column vector 1_N has N entries, all equal to one. The degree of a polynomial or a graph vertex is $\deg(\cdot)$. Graphs are undirected. A diagonal matrix $\mathrm{diag}(x_1, \ldots, x_n)$ has the elements $x_1, \ldots, x_n$ along its diagonal.

II. Problem Formulation

A. Dynamic Average Consensus

Consider a group of N agents communicating over a network modeled by an undirected graph whose vertices represent agents and whose edges represent communication links. If agents i and j communicate directly, the edge between them has weight $a_{ij} = a_{ji} > 0$ and they are neighbors. Otherwise, $a_{ij} = a_{ji} = 0$.

Every graph has an associated weighted Laplacian matrix $L \in \mathbb{R}^{N \times N}$ with entries

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

The Laplacian matrix is symmetric and positive semidefinite, with eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$. Our analysis assumes that the communication graph always remains connected, which implies that the algebraic connectivity of the graph λ_2 will always be greater than zero. We omit the time dependence of L, however it may switch at discrete points.

This work is supported in part by the Office of Naval Research. The authors are with the Department of Mechanical Engineering (Elwin and Lynch), the Northwestern Institute on Complex Systems (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.

Every agent has three local scalars: a time-varying input $\phi_i(t)$, a global average estimate $v_i(t)$, and an auxiliary output $\eta_i(t)$. The agents transmit $v_i(t)$ and $\eta_i(t)$ to their neighbors. The vectors $\phi(t)$, $v(t)$, and $\eta(t)$ contain N entries corresponding to each agent's $\phi_i(t)$, $v_i(t)$, and $\eta_i(t)$ values.

To evaluate the performance of the group, we define the global tracking error as

$$e(t) \triangleq v(t) - \frac{1}{N} 1_N 1_N^T \phi(t). \tag{2}$$

Estimators achieve robust dynamic average consensus when they, regardless of initialization, converge with zero steady-state tracking error: $\lim_{t\to\infty}|e(t)| = 0$. Internal model estimators, discussed in the next section, have this property: therefore, a change in the network will cause a transient error response that exponentially decays to zero.

B. Internal Model Estimator

For reference, we describe the internal model estimator, as presented in [12]. This robust dynamic average consensus estimator allows a group of agents, communicating over a network with Laplacian L, to track the average of their individual time-varying inputs $\phi_i(\star)$ with zero steady-state error, even when initialized incorrectly.

The internal model estimator generalizes the more common proportional-integral (PI) estimator, used and discussed in [1], [2], [4]–[6], [10], [15]. The PI estimator is a robust average consensus estimator for constant inputs; that is inputs of the form $\phi_i(s) = \frac{c_i}{s}$ in continuous time or $\phi_i(z) = \frac{c_i}{z-1}$ in discrete time. The internal model estimator works on a more general class of inputs, modeled by

$$\phi_i(\star) = \frac{c_i(\star)}{d(\star)}, \tag{3}$$

where $c_i(\star)$ is a numerator polynomial that can be different for each agent and $d(\star)$ is the input model, a monic polynomial common to all agents. Such a model captures a wide range of signals including sinusoids $\frac{c_i}{s^2+\omega^2}$, ramps $\frac{c_i}{s^2}$, exponential growth $\frac{c_i}{s-a^2}$, and linear combinations thereof. Additionally, when using the estimator for decentralized Kalman filtering, the model needed by the Kalman filter directly provides $d(\star)$ [16].

When implementing the internal model estimator consensus protocol, just as with the PI estimator, the agents compute their estimate $v_i(\star)$ and auxiliary output $\eta_i(\star)$ according to

$$v_i(\star) = h(\star)(\phi_i(\star) - k_p(\star)[L]_i v(\star) - [L]_i \eta(\star)) \tag{4}$$

$$\eta_i(\star) = g(\star)[L]_i v(\star), \tag{5}$$

where $[L]_i$ is the i-th row of the Laplacian matrix, and $h(\star)$, $g(\star)$, and $k_p(\star)$ are transfer functions the designer chooses. In the continuous time PI estimator, $h(s) = \frac{\gamma}{s+\gamma}$, $g(s) = \frac{\gamma}{s}$, and $k_p(\star) = k_p$, where γ is the rate at which new information replaces old information and k_p is a proportional gain.

Fig. 1 depicts the system when every agent implements equations(4) and (5). The resulting error dynamics are

$$e(\star) = P(\star)\phi(\star), \tag{6}$$

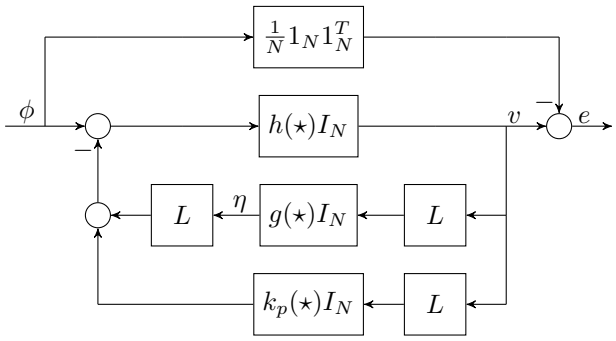

Fig. 1. Block diagram of the global error dynamics. The $N \times N$ identity matrix is I_N and is included to emphasize that every agent implements $h(\star)$, $g(\star)$, and $k_p(\star)$.

where the input-to-error transfer function $P(\star)$ is

$$P(\star) = (I + h(\star)g(\star)L^2 + k_p(\star)h(\star)L)^{-1}h(\star) - \frac{1}{N}1_N 1_N^T. \tag{7}$$

Equations(4) and (5) contain quantities that the designer must choose: the transfer functions $k_p(\star)$, $h(\star)$, and $g(\star)$. To simplify synthesis we set $k_p(\star)$ to zero, reducing complication at the expense of potentially limiting performance.

To implement the estimator using one-hop communication, both $h(\star)$ and $g(\star)$ must be strictly proper. Otherwise, the current output of $h(\star)$ or $g(\star)$ will depend directly on the current input: since components of these inputs and outputs must be communicated, a two-hop protocol would be required. Another requirement comes from the internal model principle [11], which imposes structural requirements on $h(\star)$ and $g(\star)$. We use these requirements to partition $h(\star)$ and $g(\star)$ into two parts: the internal model $d(\star)$ and the controllers $k_h(\star)$ and $k'_g(\star)$. Given these transfer functions, $h(\star)$ takes the form of the unity feedback system

$$h(\star) = \frac{\frac{k_h(\star)}{d(\star)}}{1 + \frac{k_h(\star)}{d(\star)}}, \tag{8}$$

whereas $g(\star)$ is the series interconnection

$$g(\star) = \frac{k'_g(\star)}{d(\star)}. \tag{9}$$

With the internal model $d(\star)$ given, the designer must choose the controllers $k_h(\star)$ and $k'_g(\star)$ to specify the estimator.

The final condition on $h(\star)$ and $g(\star)$ relates to the stability of the input-to-error transfer function $P(\star)$. To ensure internal stability, the numerator of $h(\star)$ and the denominator of $g(\star)$ must have no common unstable roots. Additionally, $P(\star)$ must be stable. The following theorem, adapted from [12], relates the stability of $P(\star)$ to the aforementioned conditions and the eigenvalues of the Laplacian.

Theorem 1: Assume that $h(\star)$ and $g(\star)$ are strictly proper, $k_p(\star)$ is zero, $h(\star)$ is stable, and the numerator of $h(\star)$ and the denominator of $g(\star)$ have no common unstable roots. Additionally, assume that the input $\phi(\star)$ and transfer functions $h(\star)$ and $g(\star)$ follow equations (3), (8), and (9) respectively. Then, when implemented with a one-hop protocol over a connected communication network, the internal

model estimator will track the average of the agents' inputs with zero steady-state error if and only if

$$T(\star) = \frac{h(\star)}{1 + g(\star)h(\star)\lambda_i^2} \tag{10}$$

is stable for $i = 2 \dots N$.

Proof: See [12] for proof. ∎

If the conditions for Theorem 1 hold, the internal model estimator will converge with zero steady-state tracking error, regardless of the estimator's initial state. Events that cause incorrect state initializations such as network topology changes, dropped communication packets, agent failures, and agent additions will result in a transient response that decays to zero. Although useful for analysis, this theorem does not address the synthesis problem, which we discuss in the next section.

III. The Design Method

A. Separation Principle

Every stable internal model estimator consists of transfer functions $h(\star)$ and $g(\star)$ that satisfy Theorem 1. Note that condition (10) depends on the communication topology because it must be satisfied for all nonzero Laplacian eigenvalues. Not only does this condition depend on a specific network, but it effectively corresponds to $N-1$ design constraints if the Laplacian has $N-1$ unique eigenvalues. This dependence on the network complicates estimator design because the Laplacian may be unknown. Without knowledge of the Laplacian, an engineer must choose $h(\star)$ and $g(\star)$ with particular attention to the gain margin to obtain an estimator that remains stable over a range of eigenvalues. Gain margin techniques, however, do not account for performance; therefore, we introduce a separation principle that relates the poles of the multi-input multi-output (MIMO) input-to-error transfer function $P(\star)$ to the poles of a single-input single output (SISO) separated system $T(\star)$.

Proposition 1: The poles of $P(\star)$, with $k_p(\star) = 0$, are the poles of $h(\star)$ and the poles of $T(\star)$ for λ_i, $i = 2, \dots, N$.

Proof: As in [12], Equation (7) can be written as

$$P(\star) = Q^T \begin{pmatrix} h(\star) & 0 \\ 0 & \bar{P}(\star) \end{pmatrix} Q - \frac{1}{N} 1_N 1_N^T, \tag{11}$$

where Q is orthonormal, $L = Q^T \operatorname{diag}(\lambda_1, \dots, \lambda_N) Q$, and

$$\bar{P} = \operatorname{diag}\left(\frac{h(\star)}{1 + g(\star)h(\star)\lambda_2^2}, \cdots, \frac{h(\star)}{1 + g(\star)h(\star)\lambda_N^2} \right). \tag{12}$$

The result is true by inspection. ∎

Proposition 1 means that if the poles of $h(\star)$ and $T(\star)$ lie in a given region for all possible nonzero Laplacian eigenvalues λ_i, the poles of the error transfer function $P(\star)$ will also be in that region. Therefore, we can apply robust control techniques to ensure that the poles of $T(\star)$ lie in a region $\mathcal{D}$ for all $\lambda_{\min} < \lambda_i < \lambda_{\max}$, allowing us to design estimators that meet a pole placement constraint without specific knowledge of the network or the number of agents.

Even when the Laplacian's eigenvalues are unknown, bounds on these eigenvalues are reasonably attainable. The lower bound $\lambda_{\min}$ corresponds to a bound on how well the network is connected because this bounds λ_2, the graph's algebraic connectivity [17]. Likewise, the upper bound $\lambda_{\max}$ can be determined from the communication weighting scheme.

Two weighting methods that have been used with PI estimators and that seem appropriate for internal model estimators are constant weighting and inverse sum degree weighting [12]. With constant weighting every edge has the same weight. If the weights are set to $\tilde{N}^{-1}$, where $\tilde{N}$ is an upper bound for the number of agents, $\lambda_{\max} \leq 1$ [1]. With inverse sum degree weighting, the weights are chosen such that $a_{ij} = (\deg(i) + \deg(j))^{-1}$, ensuring that $\lambda_{\max} \leq 1$ [1]. Implementing this scheme requires neighboring agents to exchange their degrees.

B. Robust Control Formulation

The relationship between the poles of the separated system $T(\star)$ and the poles of the error transfer function $P(\star)$, as established by Proposition 1, hinges on the eigenvalues of the Laplacian. In particular, every nonzero Laplacian eigenvalue λ_i corresponds to a specific separated system $T(\star)$ with $\lambda = \lambda_i$: the poles of $P(\star)$ must be the poles of one of these systems or the poles of $h(\star)$. Rather than handling each of these separated systems individually for a specified Laplacian, we assume that we know only a range of possible eigenvalues, $\lambda_{\min} < \lambda_i < \lambda_{\max}$ for $i = 2 \dots N$. Using robust control theory, we can design controllers that meet specifications for any λ within the range.

To apply robust control theory, we put the system into the standard robust control form of Fig. 2 by partitioning it into three parts: the uncertainty Δ, the generalized plant $G(\star)$, and the controller $K(\star)$. To simplify synthesis, we set $\Delta = \delta$, where $\delta \in \mathbb{C}$ is a normalized uncertainty, with $|\delta| < 1$. We treat δ as complex because the pole placement method we use assumes complex uncertainty; however, this assumption is conservative because the Laplacian's eigenvalues are real. The uncertain Laplacian eigenvalues enter the separated system equation (10) as a quadratic term; therefore we treat λ^2 as the uncertain parameter. The following definition maps the normalized uncertainty δ to the actual uncertainty λ^2:

$$\lambda^2 \triangleq \sigma_a + \delta\sigma_d, \tag{13}$$

where $\sigma_a = \frac{\lambda_{\max}^2 + \lambda_{\min}^2}{2}$ and $\sigma_d = \frac{\lambda_{\max}^2 - \lambda_{\min}^2}{2}$. With this mapping, as δ goes from -1 to 1, λ goes from $\lambda_{\min}$ to $\lambda_{\max}$.

We also partition $k_g'(\star)$ into $k_g(\star)$ and $q(\star)$ such that

$$k_g'(\star) = k_g(\star) q(\star), \tag{14}$$

which allows us to include a fixed $q(\star)$ in the generalized plant while designing $k_g(\star)$ as part of the controller.

Fig. 3 shows the block diagram of the separated system $T(\star)$ with the uncertainty δ. To isolate this uncertainty, we define the output $y_\delta(\star)$ and input $u_\delta(\star)$ such that

$$u_\delta(\star) = \delta y_\delta(\star). \tag{15}$$

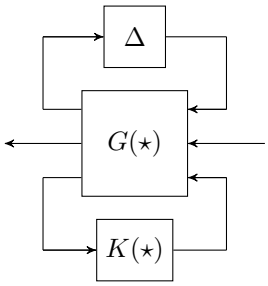

Fig. 2. Control system in standard robust control form.

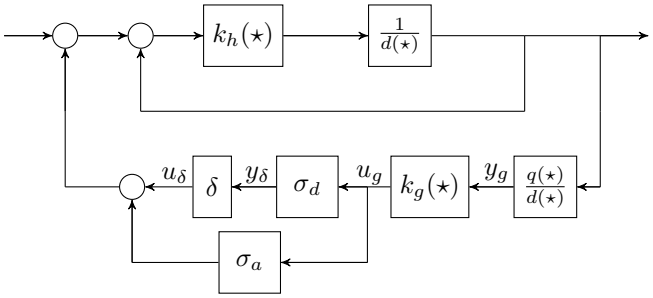

Fig. 3. Block diagram of $T(\star)$.

In this way, $y_\delta(\star)$ is an output from the plant and input to the uncertain parameter and $u_\delta(\star)$ is an input to the plant and output from the uncertain parameter.

We must also separate the controller from the plant. With $h(\star)$, $g(\star)$, and $k_g'(\star)$ in the form of equations(8), (9) and (14), the controller $K(\star)$ consists of $k_h(\star)$ and $k_g(\star)$. Incorporating both $k_h(\star)$ and $k_g(\star)$ into the controller, however, complicates the design process because $K(\star)$ will be diagonal, which requires certain control channels to be zero. To avoid this complication, we design $k_h(\star)$ prior to robust controller synthesis. Thus, $h(\star)$ is part of the plant, so the controller is $k_g(\star)$. We choose to fix $k_h(\star)$ rather than $k_g(\star)$ because the poles of $h(\star)$ are always poles of $P(\star)$.

To separate the controller from the plant, we employ the same method that we used to isolate the uncertainty. We define the plant output $y_g(\star)$ and plant input $u_g(\star)$ such that

$$u_g(\star) = k_g(\star) y_g(\star). \tag{16}$$

Thus, the controller $K(\star)$ is $k_g(\star)$.

The plant maps the uncertainty, external, and control inputs ($u_\delta(\star)$, $w(\star)$, $u_g(\star)$) into the corresponding outputs ($y_\delta(\star)$, $z(\star)$, $y_g(\star)$) according to

$$\begin{pmatrix} y_\delta(\star) \\ z(\star) \\ y_g(\star) \end{pmatrix} = G(\star) \begin{pmatrix} u_g(\star) \\ w(\star) \\ u_g(\star) \end{pmatrix}, \tag{17}$$

where

$$G(\star) = \begin{pmatrix} 0 & 0 & \sigma_d \\ -h(\star) & h(\star) & -\sigma_a h(\star) \\ -\frac{h(\star)q(\star)}{d(\star)} & \frac{h(\star)q(\star)}{d(\star)} & -\frac{\sigma_a h(\star)q(\star)}{d(\star)} \end{pmatrix}. \tag{18}$$

From this robust control formulation, we can recover the closed loop system $T(\star)$ either by making appropriate algebraic substitutions or by using linear fractional transformations [13]. We can also use robust control techniques to design $k_g(\star)$. In the next section, we discuss one such technique, robust pole placement.

C. Robust Pole Placement

To control the internal model estimator's error response, we would like to choose $h(\star)$ and $g(\star)$ such that the poles of the input-to-error transfer function $P(\star)$ lie in $\mathcal{D}$, a region of the complex plane. Rather than placing the poles of $P(\star)$ directly, which requires knowledge of the Laplacian, we will ensure that the poles of the separated system $T(\star)$ lie in $\mathcal{D}$ for a specified range of Laplacian eigenvalues. The robust pole placement algorithm of [14] allows us to accomplish this task.

The algorithm requires the region $\mathcal{D}$ to be a linear matrix inequality (LMI) region, defined by

$$\{\rho \in \mathbb{C} : Q + \rho W + \bar{\rho} W^T \prec 0\}, \tag{19}$$

where $Q = Q^T \in \mathbb{R}^{p\times p}$ and $W \in \mathbb{R}^{p\times p}$. The intersection of LMI regions is an LMI region, and any convex region that is symmetric with respect to the real axis can be approximated by an LMI region [14].

For our design problem in the continuous time case, we specify $\mathcal{D}$ as the intersection of a half plane, a conic sector, and a disc, which approximately restricts the maximum and minimum natural frequency and damping ratio of the poles. The parameters for these regions are given by

$$Q_1 = 2\omega_{\min}, \quad W_1 = 1; \tag{20}$$

$$Q_2 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad W_2 = \begin{pmatrix} \sin(\theta) & \cos(\theta) \\ -\cos(\theta) & \sin(\theta) \end{pmatrix}; \tag{21}$$

and

$$Q_3 = \begin{pmatrix} -\omega_{\max} & 0 \\ 0 & -\omega_{\max} \end{pmatrix}, \quad W_3 = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}; \tag{22}$$

with $\omega_{\min}$ and $\omega_{\max}$ being the minimum and maximum natural frequency of the poles and $\theta = \arccos(\xi)$, where ξ is the minimum damping ratio. For discrete time systems we restrict the poles to lie in a disc centered at the origin. We specify a performance constraint by adjusting the disc's radius.

Given the region $\mathcal{D}$ and a range of eigenvalues, we can use the robust pole placement algorithm of [14]. The algorithm involves solving an LMI feasibility problem, which can be done using software such as CVX [18]. The resulting controller $k_g(\star)$ ensures that the poles of $T(\star)$ remain in $\mathcal{D}$ for $|\delta| < 1$; thus the poles of $P(\star)$ will remain in $\mathcal{D}$ for any graph with $\lambda_{\min} < \lambda_2 \leq \lambda_N < \lambda_{\max}$.

The algorithm is slightly conservative, so it may not find a controller even when one exists. This conservativeness is compounded because the algorithm's guarantees apply to complex δ even though the eigenvalues of the symmetric Laplacian are real. We can compensate for this conservativeness by specifying a restricted eigenvalue range because the resulting estimator often performs well and remains stable for graphs that violate the eigenvalue specification.

D. Design Process

Although the robust pole placement algorithm returns a result automatically, the inputs to this algorithm must be carefully chosen. If the algorithm fails, the area of $\mathcal{D}$ can be increased or the range of allowable λ values decreased. Additionally, choices for $k_h(\star)$ and $q(\star)$ may be adjusted.

One useful heuristic for choosing $k_h(\star)$ is to make it a polynomial of degree $\deg(d(\star) - 1)$ with minimum phase roots. Using this rule produces a strictly proper $h(\star)$ with minimal order and the maximum number of zeros. A similar rule applies to $q(\star)$: making $q(\star)$ a polynomial with $\deg(q(\star)) = \deg(d(\star)) - 1$ ensures that any $k_g(\star)$ returned by the pole placement algorithm will yield a strictly proper $g(\star)$. Conveniently, choosing $q(\star)$ equal to $k_h(\star)$ or choosing it to cancel some poles of $h(\star)$ both seem to work well. Although the exact location of these zeros appears not to matter, our experience indicates that their presence makes pole placement more likely to succeed. Using these heuristics, the design process consists of the following steps:

1) Select the LMI region $\mathcal{D}$. In continuous time this region is based on the natural frequency constraints $\omega_{\min}$, $\omega_{\max}$, and the damping constraint ξ, while in discrete time it is a disc with radius $r < 1$.
2) Select bounds on the Laplacian's eigenvalues $\lambda_{\min}$ and $\lambda_{\max}$. Any controller returned by the pole placement algorithm is guaranteed to place the poles of $T(\star)$ in $\mathcal{D}$ for any λ in this range.
3) Using a root locus diagram, choose $k_h(\star)$ such that $\text{poles}(h(\star)) \in \mathcal{D}$ and $h(\star)$ has a relative degree of one.
4) Choose $q(\star)$, perhaps by setting it to $k_h(\star)$ or to cancel some poles of $h(\star)$.
5) Use the pole placement algorithm to find $k_g(\star)$.
6) If the pole placement algorithm succeeds, tighten the constraints and iterate to meet stricter specifications on a wider range of graphs. If the pole placement algorithm fails, iterate using relaxed constraints or different $k_h(\star)$ and $q(\star)$.

IV. Examples

This section compares the settling times of manual and systematic estimator designs. We use root locus techniques and the procedure of [12] for the manual designs. In both examples $\lambda_{\min} = 0.7$ and $\lambda_{\max} = 0.9$. Typically, $\lambda_{\min}$ represents a minimum connectivity and $\lambda_{\max}$ can be determined from the weighting scheme. Although the examples use a narrow eigenvalue range, the actual robustness interval is wider due to the conservativeness of the pole placement method. We simulate the estimator over several graphs. The first seven graphs are 20 node Erdős-Rényi graphs with various link formation probabilities. These graphs use inverse sum degree weighting. The eighth graph is a 20 node geometric random graph with constant weights.

A. Continuous Sinusoid

In this example, we assume that the inputs are modeled by $d(s) = s^2 + 1$: sinusoids with unit frequency and different phases and magnitudes. Table I provides the parameters for the manual and systematic designs. For comparison purposes, both designs use the same $h(s)$ and a similar $q(s)$. To avoid having slow, lightly damped poles we specify limits on the minimum natural frequency, $\omega_{\min} = 5.5$, and the damping ratio, $\xi = 0.1$. To achieve a fair comparison, we set $\omega_{\max} = 22$ and enforce this constraint in both designs.

	Manual	Systematic
$d(s)$	$s^2 + 1$	$s^2 + 1$
$k_h(s)$	$35s + 297.5$	$35s + 297.5$
$q(s)$	$14(s + 14.72)$	$s + 14.72$
$k_g(s)$	1	$\frac{14(s+15.0)(s+4.0)(s^2+35s+316)}{(s+8.5)(s+14.7)(s^2+38.6s+400)}$
Stable λ	$(0, \infty)$	$(0, \infty)$
Robust λ	$\emptyset$	$(0.67, 0.94)$

TABLE I
CONTINUOUS TIME DESIGNS. VALUES ARE ROUNDED.

Fig. 4 depicts the region $\mathcal{D}$ and the root loci of both designs for $0 < \lambda < 1$. The poles of the manual controller violate the performance constraint for all values of λ while the poles of the robust design meet the criteria for $0.67 < \lambda < 0.94$. Both estimators have an infinite gain margin; therefore they will be stable for any graph.

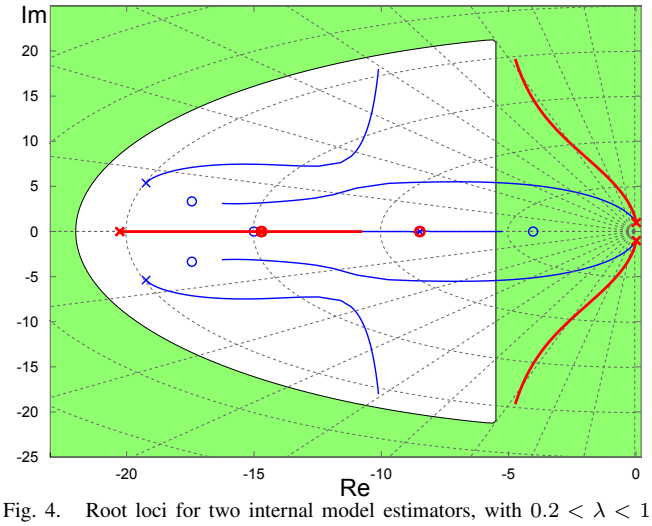

Fig. 4. Root loci for two internal model estimators, with $0.2 < \lambda < 1$. The thick red locus is the manual design and the thin blue locus is the robust pole placement design. The unshaded region is $\mathcal{D}$.

To compare the performance of the estimators, we performed simulations using unit frequency sinusoids with random magnitude and phase as inputs. The simulation results, depicted in Fig. 5, show similar settling times for both the manual and systematic designs. Many of the tested graphs have eigenvalues outside of the robustness range of the systematic design, diminishing its performance advantage. When the graph's eigenvalues fit within the robustness range, however, the systematic estimator offers a bigger performance improvement.

B. Discrete Ramp

The only difference between designing continuous and discrete time internal model estimators is the stability region.

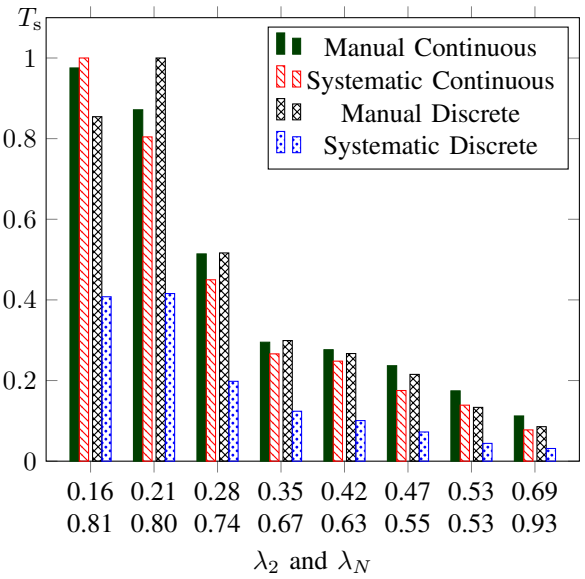

Fig. 5. Error settling time for the manual and systematic designs. The continuous and discrete examples are normalized to the largest settling time for their respective category.

In this example we choose the region $\mathcal{D}$ to be the disc centered at the origin with a radius of 0.84, providing some performance benefits and robust stability. We design an estimator assuming that the inputs are ramp signals modeled by $d(z) = (z-1)^2$, and choose a sampling frequency of 1 kHz. Table II shows the parameters for the manual and systematic designs. As in the continuous time example, we use the same $h(z)$ and similar $q(z)$ in both designs, primarily for comparison purposes. The exact choice of $q(z)$, however, tends to have an insignificant effect on the results. Both estimators have a wide stability range, remaining stable for any graph whose weighting scheme limits $\lambda_{\max}$ to be less than one.

	Manual	Systematic
$d(s)$	$(z-1)^2$	$(z-1)^2$
$k_h(s)$	$2z-1$	$2z-1$
$q(s)$	$0.5z-0.25$	$0.5z-0.25$
$k_g(s)$	1	$\frac{1.6(z-0.86)(z+0.14)(z+0.0074)(z-0.0081)}{(z+0.88)(z+0.15)(z^2-0.999z+0.25)}$
Stable λ	$(0, 1.2]$	$(0, 1.1]$
Robust λ	$(0.92, 1.04)$	$(0.61, 1.02)$

TABLE II
DISCRETE TIME DESIGNS. VALUES ARE ROUNDED.

Fig. 5 depicts the settling time of the error over the same graphs used in the continuous time example. The results improve upon the continuous time case, with the systematic design settling significantly faster than the manual design. Additionally, the systematic design has a wider robustness range than the manual design. This range exceeds the specified values due to the pole placement algorithm's conservativeness.

V. CONCLUSION

We have formulated internal model estimator synthesis as a robust control problem and developed a systematic design process based on robust pole placement. By using robust control theory, we were able to decouple estimator design from specific network topologies. Within the current LMI based robust pole placement framework, additional performance criteria such as H_∞ and H_2 norm bounds can be incorporated; this could help design estimators that handle noisy signals.

Our current research involves simultaneously synthesizing $h(\star)$, $g(\star)$, and $k_p(\star)$. The addition of $k_p(\star)$ can potentially improve performance; however, it adds additional structure to the uncertainty and the controller, making the pole placement problem non-convex. We are, therefore, approaching the problem with global optimization methods.

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] P. Yang, R. A. Freeman, and K. M. Lynch, "Multi-agent coordination by decentralized estimation and control," *IEEE Transactions on Automatic Control*, vol. 53, no. 11, pp. 2480–2496, Dec. 2008.

[6] 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.

[7] Y. Cao, W. Ren, and Y. Li, "Distributed discrete-time coordinated tracking with a time-varying reference state and limited communication," *Automatica*, vol. 45, no. 5, pp. 1299–1305, 2009.

[8] D. P. Spanos, R. Olfati-Saber, and R. M. Murray, "Dynamic consensus for mobile networks," in *IFAC World Congress*. Prague, Czech Republic, 2005.

[9] M. Zhu and S. Martínez, "Discrete-time dynamic average consensus," *Automatica*, vol. 46, no. 2, pp. 322–329, 2010.

[10] 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.

[11] B. Francis and W. Wonham, "The internal model principle of control theory," *Automatica*, vol. 12, no. 5, pp. 457–465, 1976.

[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] K. Zhou, J. C. Doyle, and K. Glover, *Robust and Optimal Control*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.

[14] M. Chilali, P. Gahinet, and P. Apkarian, "Robust pole placement in LMI regions," *IEEE Transactions on Automatic Control*, vol. 44, no. 12, pp. 2257–2270, 1999.

[15] 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.

[16] H. Bai, R. Freeman, and K. Lynch, "Distributed kalman filtering using the internal model average consensus estimator," in *American Control Conference (ACC), 2011*, July 2011, pp. 1500 –1505.

[17] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973.

[18] CVX Research Inc., "CVX: Matlab software for disciplined convex programming, version 1.22," http://cvxr.com/cvx, Sept. 2012.