INTERNATIONAL JOURNAL OF ROBUST AND NONLINEAR CONTROL
Int. J. Robust. Nonlinear Control 2012; **22**:1691–1702
Published online 25 July 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/rnc.1778

Designing linear distributed algorithms with memory for fast convergence

Sandip Roy[1,*,†], Yan Wan[2], Ali Saberi[1] and Mengran Xue[1]

[1]*Department of Electrical Engineering, Washington State University, Pullman, WA 99164-2752, USA*
[2]*Department of Electrical Engineering, University of North Texas, Denton, TX 76203-5017, USA*

SUMMARY

Motivated by both distributed computation and decentralized control applications, we studied the distributed linear iterative algorithms with memory. Specifically, we showed that the system of linear equations $G\mathbf{x} = \mathbf{b}$ can be solved through a distributed linear iteration for arbitrary invertible G using only a single memory element at each processor. Further, we demonstrated that the memoried distributed algorithm can be designed to achieve much faster convergence than a memoryless distributed algorithm. Two small simulation examples were included to illustrate the results. Copyright © 2011 John Wiley & Sons, Ltd.

Received 3 December 2009; Revised 28 May 2011; Accepted 5 June 2011

KEY WORDS: distributed algorithms; large-scale networks; decentralized control theory; numerical algorithms

1. INTRODUCTION AND PROBLEM FORMULATION

Stationary linear iterative strategies have long been used to solve systems of linear equations [1–4]. In this classical literature, it is recognized that using multiple past estimates of the solution (or in other words using an iteration with longer memory) can (i) broaden the class of linear equations amenable to solution and (ii) speed up the algorithm. In the last few years, linear iterations have been used in numerous distributed algorithms, including in agreement/consensus and sensor-fusion tasks, with the motivation that such methods are efficient in exploiting the network's topological structure and have verifiable convergence rate [5–9]. By drawing on the numerical methods literature, Cao and coworkers have recently illustrated that faster linear distributed algorithms (in particular, faster *distributed consensus algorithms* used in, for example, sensor-fusion applications) can be constructed using extended memories at the processors [7].

Meanwhile, the problem of shaping network dynamics has, in parallel, been studied in the control engineering community, under the heading of *decentralized control* [10]. Unfortunately, the wide historical literature in decentralized control is not apt for distributed algorithm construction, because these works view the network as a disturbance to be dominated rather than a means for coordination. However, very recently, a few researchers in the decentralized control community including our group have pursued the design of memoried (and memoryless) distributed controls that shape network dynamics by exploiting the network's topological structure [11–16]. These efforts, which are motivated by both autonomous-agent applications (such as vehicle coordination) and infrastructure networking problems (such as air traffic flow management), are largely focused on stabilizing and shaping (e.g. through eigenvalue placement) a *continuous-time* linear time-invariant dynamics rather than a discrete-time iteration. The control theoretic studies also differ from the efforts on

*Correspondence to: Sandip Roy, Department of Electrical Engineering, Washington State University, Pullman, WA 99164-2752, USA.
†E-mail: sroy@eecs.wsu.edu

numerical computing and distributed algorithms, in that a more general and varied class of memory updates are used, and often a broader class of dynamics need to be shaped.

Here, we demonstrate the design of iterative linear distributed algorithms through a canonical problem—namely, the distributed solution of a general system of linear equations—by cohering and building on the existing approaches in both the numerical methods and decentralized controls communities. Specifically, we prove that an algorithm with a single extra memory element can be designed to (i) achieve the solution task for an *arbitrary* system of linear equations, and (ii) significantly outperform the algorithm without extra memory in convergence rate for a broad class (in particular, roughly allowing a *spectral radius*, that is, a small distance μ away from 1 to be moved to a larger distance on the order of $\sqrt{\mu}$ away from 1). We also note the applicability of the iterative methods for numerous discrete-time decentralized control problems (in, e.g., macroeconomics or virus spread-control applications) and argue that our results hold promise to foster advances in the classical numerical methods literature.

A variety of distributed algorithms that can be modeled as discrete-time linear dynamics (linear iterations) have been proposed in recent years in such diverse application areas as Web search and sensor fusion. In parallel, numerous discrete-time distributed control tasks have arisen in application areas ranging from virus-spreading control to macroeconomics and clock synchronization. The memoried iteration concept that we are introducing here can be adapted to various of these algorithmic and control tasks, but our focus here is on convergence rate design rather than the particular algorithmic task pursued. Thus, we focus on one canonical algorithmic task, namely, the distributed solution of the system of linear equations

$$G\mathbf{x} = \mathbf{b}, \tag{1}$$

where $G \in R^{n\times n}$ is nonsingular but otherwise arbitrary, $\mathbf{b} \in R^n$ is an arbitrary real vector, and the goal is to compute $\mathbf{x} \in R^n$ in a distributed, iterative fashion. That is, we seek an iterative solution using n processors that each only use some statistics specified by the **topology matrix** G. We note that this problem arises in purely computational domains (for instance, in sensor networking or parallel processing applications) and in discrete-time decentralized control applications wherein the iteration represents a costly regulation of local states (rather than simply a numerical computation). We describe a model for the iterations that encompasses both cases.

Formally, we consider a system with n **processors** or **agents**, labeled $1, \ldots, n$. Each processor i has a scalar **internal status** $x_i[k]$ that evolves at discrete time instances $k = 0, 1, 2, \ldots$. These internal statuses represent stored numerical values in computational applications and physical local quantities in control applications (e.g., numbers of infectives in virus-spreading control problems or cash and product reserves in a macroeconomic system). In addition, we assume that each processor has an available scalar **storage variable** $z_i[k]$ that also can be updated in discrete time. We find it convenient to assemble the internal statuses and storage variables into vectors, specifically defining an **internal state vector** $\mathbf{x}[k] = \begin{bmatrix} x_1[k] & x_2[k] & \ldots & x_n[k] \end{bmatrix}^T$ and a **storage vector** $\mathbf{z}[k] = \begin{bmatrix} z_1[k] & z_2[k] & \ldots & z_n[k] \end{bmatrix}^T$.

We assume that each processor must update its internal status (and storage variable) using only certain local statistics of the current internal state vector and storage vector. Precisely, we assume that at time k, each processor i has available statistics $y_i[k] = \mathbf{g}_i^T\mathbf{x}[k]$ (where $\mathbf{g}_i^T$ is the ith row of G) and $z_i[k]$. We also assume that the processor i has knowledge of b_i, the ith entry of the vector $\mathbf{b}$. Each processor has the capability to *augment* its current internal status using these statistics and to update its storage variable using the statistics. We, here, consider a stationary linear augmentation of the internal status, and a linear update of the storage variable, as follows:

$$\begin{aligned} x_i[k+1] &= x_i[k] + q_i y_i[k] + r_i z_i[k] + s_i b_i, \\ z_i[k+1] &= t_i z_i[k] + v_i y_i[k]. \end{aligned} \tag{2}$$

We comment on a couple of critical aspects of the aforementioned iteration. First, the reader should note, we enforce, that the update of the internal status is an augmentation (i.e., $x_i[k+1] = x_i[k] + \ldots$). We enforce this structure because (i) the application domain often enforces this structure (for instance, in virus-spreading or economics applications where statuses remain unchanged

DOI: 10.1002/rnc

except when acted on by expensive controls or in clock-synchronization applications where the augmentation may be hardwired) and (ii) this update is desirable in many numerical procedures (see the classical work on iterative solutions of linear systems [1, 4] for justification). Second, we again stress that, in general, the processor may only have access to the statistic $y_i[k]$ and *not* to the particular internal statuses of any particular processor (including itself). Thus, our formulation permits us to pursue computation when the statistic available to processor i is *any* linear combination of internal statuses. We note that this generalizes the case typically considered in, for example, the linear consensus algorithms literature, where only certain averages of status differences are permitted (and hence a 'diffusive' topology codified by a Laplacian or doubly stochastic topology matrix G is obtained). Third, we note that the iteration earlier differs significantly from those in the numerical methods literature and recent distributed computing literature in that the storage variable is updated based on its own previous value, as well as on the observation of the internal state. This update structure fundamentally allows the incorporation of the whole history of the internal statuses, rather than only a number of recent status values, in the iterative update. We will see that this update structure is crucial both in permitting distributed computation for general G and in allowing fast convergence.

Our goal is to design the aforementioned distributed iteration so that the solution $\mathbf{x} = G^{-1}\mathbf{b}$ to the system of linear equations (1) is distributively computed. That is, we aim to design the aforementioned iteration so that, asymptotically, processor i's internal status converges to the ith entry of the vector $G^{-1}\mathbf{b}$. We also seek for *fast convergence*, or in other words, seek to make the error between each processor i's internal status $x_i[k]$ and the desired status value $(G^{-1}\mathbf{b})_i$, or $e_i[k] = |(G^{-1}\mathbf{b})_i - x_i[k]|$, small within a few time steps.

It is also worthwhile for us to make precise how the the aforementioned iteration can be implemented in a distributed fashion, in both computational and control applications. In the iteration, each processor i in the network only stores a scalar (not the whole vector $\mathbf{x}[k]$) that we can view as a local state. To compute the next local state value, the processor must have available its own stored value $x_i[k]$ (or at least must be able to increment this value) and the statistic $\mathbf{g}_i^T\mathbf{x}[k]$. We note that, in order for processor i to have the statistic $\mathbf{g}_i^T\mathbf{x}[k]$, it will require current state information (or, precisely, a particular linear combination of this state information) from processors j such that g_{ij} is nil; we view such processors j as being graphical neighbors of i, according to the topology or graph matrix G. This information can be viewed as being communicated to the processor in computational applications and as being either communicated or sensed in physical control applications; thus, the algorithm is by no means completely decentralized (it requires communication/sensing), but certainly the processors are working in parallel and interacting through communication or sensing (which will be sparser when the graph is sparser). We note that the algorithm does require *a priori* distribution of local information to each processor and, in general, may also require global design of the weights (gains) in the linear algorithm. In cases where global design of the weights is not feasible, we note that a fully distributed solution may still sometimes be possible: the high-performance design that we present for the case of symmetric G allows use of identical weights at each processor, and the algorithm weights obtained through the design process may be identical for *classes* of topology matrices rather than individual ones.

We conclude our formulation by briefly connecting our results with associated studies in the linear distributed algorithms and numerical methods literature. With regard to the linear distributed algorithms literature, our study is most closely connected to that of Cao (which builds on the earlier work of Boyd), in that memoried iterations are being used to achieve convergence and to improve the convergence rate [5, 7]. However, here, we consider explicit *design* of the algorithm weights for verifiable convergence and performance shaping, allow for a much more general interaction topology, and use a more general class of memoried iterations (as discussed earlier). Our efforts are also fundamentally connected with classical iterative numerical methods for solving large linear systems given in, for example, [2–4]. Our iterative method may be viewed as a generalization of the classical two-term stationary iteration introduced by Young [3], with the generalization necessitated by the need for a distributed algorithm in our context. We note that another classical iterative method, developed by Fisher and Fuller, implicitly enforces decentralization in analogy with our work but only considers a memoryless (one-term) iteration [2]. It is worth noting that we have focused here on the problem of solving linear systems, but the methodology can be adapted to other distributed

DOI: 10.1002/rnc

computation/control problems such as consensus and formation problems (e.g., [5, 7]). Because the control gains/weights need to be distributed to the processors, our methodology does require *a priori* use of a centralized processor, in general, but fully decentralized solutions are possible for special topology classes as discussed earlier.

2. RESULTS: DESIGN FOR CONVERGENCE AND FAST CONVERGENCE

We report two key results regarding design of the iterative algorithm (2). First, we show that the distributed iteration can be designed to solve the system of equations (1) for arbitrary full-rank G. Second, we demonstrate how the iterative algorithm can be designed for fast convergence, in particular, showing that for a broad class of matrices G, the *spectral radius* of the algorithm's *state matrix* can be made much smaller than for an iteration without an extra memory element.

We begin with some preliminary observations regarding the iterative algorithm (2) and then present the two main results.

2.1. Preliminaries

We find it convenient to assemble the update equations for each processor into a single linear dynamics. Specifically, defining the **full state vector** as

$$\mathbf{a}[k] = \begin{bmatrix} \mathbf{x}[k] \\ \mathbf{z}[k] \end{bmatrix},$$

we obtain

$$\mathbf{a}[k+1] = \begin{bmatrix} I_n + QG & R \\ VG & T \end{bmatrix} \mathbf{a}[k] + \begin{bmatrix} S\mathbf{b} \\ \mathbf{0} \end{bmatrix}, \tag{3}$$

where Q, R, S, T, and V are all diagonal matrices with ith diagonal entries given by q_i, r_i, s_i, t_i, and v_i, respectively. For convenience, we refer to the matrix

$$A = \begin{bmatrix} I_n + QG & R \\ VG & T \end{bmatrix}$$

as the **full state matrix** of the iteration.

Our goal is to design the diagonal **weight matrices** Q, R, S, T, and V so that the first n entries in the vector $\mathbf{a}[k]$ (i.e., the vector $\mathbf{x}[k]$) converge quickly to the vector $G^{-1}\mathbf{b}$; hence, the system of linear equations is solved. From classical results on linear system stability [17] along with characterization of the fixed point of the dynamics (3), we immediately obtain the following implicit condition under which the iteration finds the solution to the system of equations.

Theorem 1
The distributed algorithm (2) solves the system of linear equations (1), if (i) Q, R, V, and T are designed so that the eigenvalues of the full state matrix

$$A = \begin{bmatrix} I_n + QG & R \\ VG & T \end{bmatrix}$$

are strictly within the unit circle in the complex plane and T is full rank, and if (ii) S is chosen as $S = -(Q + R(I - T)^{-1}V)$.

Proof
It is automatic from classical linear systems theory that the stationary iteration converges to a fixed point if all eigenvalues of A are strictly within the unit circle. The fixed point of this iteration is the solution $\bar{\mathbf{x}}$ and $\bar{\mathbf{z}}$ to the system of equations

$$\begin{aligned} \bar{\mathbf{x}} &= (I + QG)\bar{\mathbf{x}} + R\bar{\mathbf{z}} + Sb, \\ \bar{\mathbf{z}} &= VG\bar{\mathbf{x}} + T\bar{\mathbf{z}}, \end{aligned} \tag{4}$$

where $S = -(Q + R(I - T)^{-1}V)$. With just a little algebra, we obtain $\bar{\mathbf{x}} = G^{-1}\mathbf{b}$. □

DOI: 10.1002/rnc

Our design methodology, presented in the following two subsections, yields weight matrices that actually meet the conditions of Theorem 1 for general (full-rank) G, and hence give distributed algorithms that solve the system of linear equations. We also note that the **spectral radius** of the full state matrix A (the maximum magnitude among the eigenvalues of A, a number less than 1 for any convergent iteration) specifies the convergence rate for the iteration; in particular, a smaller spectral radius guarantees faster convergence. Thus, we will also seek to design the weight matrices to make the spectral radius of A small.

Before we pursue the designs for convergence and high performance, let us quote an important linear algebra result of Fisher and Fuller [2] that we will apply in proving both our convergence and performance results. This early result was also motivated by the need for iterative solutions for systems of linear equations. In particular, the authors were seeking an algorithm *without extra memory* for solving the system (1) that would work for a broad class of nonsymmetric matrices G. Their result, which incidentally has been occasionally revisited in the decentralized controls and numerical methods communities [18, 19], can be phrased in purely linear algebraic terms as follows.

Lemma 1 (Fisher and Fuller [2]*)*
If an $n \times n$ matrix G has a sequence of n nested principal minors (with dimensions 1×1, 2×2, $\ldots$, $n \times n$) all of full rank, then an $n \times n$ diagonal matrix K can be constructed such that the eigenvalues of KG are all real and positive.

A couple of notes about the aforementioned lemma are worthwhile.

(i) The lemma immediately yields that a distributed linear iterative method without extra memory (i.e., one in which an estimate at the next time step is computed purely as an affine function of the current estimate) can solve the system of equations (1) whenever G has a nested sequence of n principal minors of full rank. It is also easy to check that a distributed linear iteration without memory cannot always be used to solve the system of equations when the sequential full rank condition is not satisfied.

(ii) The class of matrices with a nested sequence of n principal minors of full rank includes all M matrices, positive-definite matrices, and D-stable matrices. Thus, it includes, as a very special case, the family of *grounded Laplacian matrices* that have been of wide interest in the linear distributed algorithms literature.

2.2. A convergent iteration for arbitrary G

We show that, for arbitrary full-rank G, the weight matrices in the iteration ((2) and (3)) can be designed so that the it is convergent and the system of linear equations (1) is solved. We progress in two steps. First, we show how the eigenvalues of a matrix related to the state matrix A can be placed in the open left-half plane (OLHP) for general G. Next, through a scaling argument, we demonstrate that the discrete-time iteration (3) can be made convergent and hence can generally solve the equation (1).

We begin with the key lemma regarding placement of eigenvalues of a particular matrix associated with the state matrix in the OLHP. Here is the result.

Lemma 2
Consider the matrix

$$\overline{A} = \begin{bmatrix} \overline{Q}G & \overline{R} \\ \overline{V}G & \overline{T} \end{bmatrix},$$

where $G \in R^{n \times n}$, and $\overline{Q}$, $\overline{R}$, $\overline{T}$, and $\overline{V}$ are diagonal $n \times n$ matrices that are amenable to design. For arbitrary full rank G, the matrices $\overline{Q}$, $\overline{R}$, $\overline{T}$, and $\overline{V}$ can be chosen so that the eigenvalues of $\overline{A}$ are in the OLHP.

Proof
To begin, we consider the $n \times n$ matrix $Z(I_n - fG)$, where f is a positive scalar, Z is an $n \times n$ diagonal matrix, and I_n is the $n \times n$ identity matrix. We claim the following: for all f sufficiently

DOI: 10.1002/rnc

large, there exists a diagonal Z such that the eigenvalues of $Z(I_n - fG)$ are all in the open right-half plane (ORHP), that is, all have real parts strictly greater than 0. To see this, we note that all principal minors of $I_n - fG$ are of full rank, for all except a finite set of values f. Thus, for all sufficiently large f, $I_n - fG$ has a nested sequence of n principal minors of full rank. Thus, from the important classical result of Fisher and Fuller (Lemma 1), we obtain that for all sufficiently large f, there exists a diagonal matrix Z such that the eigenvalues of $Z(I_n - fG)$ are in the ORHP. Incidentally, we note that such a diagonal matrix Z has full rank and so does ZG.

Now, we choose $\overline{Q}$, $\overline{R}$, $\overline{T}$, and $\overline{V}$ as follows: $\overline{Q} = -f(I - \epsilon Z)$, $\overline{R} = -f\epsilon^2 Z^2$, $\overline{V} = I_n$, and $\overline{T} = -\epsilon Z$, where f is a large positive constant, Z is a diagonal matrix chosen as specified previously (i.e., so that $Z(I_n - fG)$ has eigenvalues in the ORHP), and ϵ is another positive constant that is chosen sufficiently large (compared with f). We claim that for sufficiently large f and ϵ, and Z chosen in this way, the eigenvalues of

$$\overline{A} = \begin{bmatrix} \overline{Q}G & \overline{R} \\ \overline{V}G & \overline{T} \end{bmatrix}$$

will be in the OLHP; hence, the lemma will be proved.

To verify the claim, note that the eigenvalues are the $2n$ solutions of the following equation:

$$\det \begin{bmatrix} sI_n + f(I_n - \epsilon Z)G & f\epsilon^2 Z^2 \\ -G & sI_n + \epsilon Z \end{bmatrix} = 0.$$

Using the formula for determinants of partitioned matrices and doing some algebra, we obtain the eigenvalues as the $2n$ solutions to the equation

$$\det[s^2 I_n + (\epsilon Z + f(I_n - \epsilon Z)G)s + f\epsilon ZG] = 0.$$

It turns out that of the $2n$ solutions to the aforementioned equation, n solutions come arbitrarily close to $s = -1$ as f and ϵ are made large, whereas the remaining solutions become large in magnitude and have negative real parts. To see this, we will consider a couple of substitutions for s. First, we use the substitution $\bar{s} = 1/s$. In terms of $\bar{s}$ (and after doing some algebra), we have

$$\det[\bar{s}^2 fZG + (Z - fZG)\bar{s} + \frac{f}{\epsilon}G\bar{s} + \frac{1}{\epsilon}I_n] = 0.$$

Thus, as long as we choose ϵ sufficiently large compared with f, from Rouche's theorem of complex variables, the solutions (with respect to $\bar{s}$) are arbitrarily near the solutions of $\det[\bar{s}^2 fZG + (Z - fZG)\bar{s}] = 0$. Thus, n solutions are arbitrarily near $\bar{s} = 0$, whereas the other n solutions are arbitrarily near the solutions of $\det[\bar{s}fZG + (Z - fZG)] = 0$, or in other words near the eigenvalues of the matrix $((fZG)^{-1})(Z - fZG) = (1/f)G^{-1} - I$. Thus, we automatically recover that by choosing f sufficiently large, the eigenvalues can be placed arbitrarily close to $\bar{s} = -1$. Because the solutions in terms of s are the inverses of the roots in terms of $\bar{s}$, we immediately recover that n solutions are arbitrarily near $s = -1$, whereas the remaining are large in magnitude.

It remains to show that the n eigenvalues of large magnitude are in the OLHP. To do so, consider $\tilde{s} = s/\epsilon$. In terms of $\tilde{s}$, we have $\det[\tilde{s}^2 I_n + (Z + f((1/\epsilon)I_n - Z)G)\tilde{s} + \frac{1}{\epsilon}fZG] = 0$. Thus, for ϵ large, we see that n roots $\tilde{s}$ are near the origin[‡], whereas the remaining n solutions are arbitrarily near the solutions to $\det[\tilde{s}I_n + Z - fZG] = 0$. We thus find that n solutions $\tilde{s}$ are arbitrarily near the eigenvalues of $-Z(I - fG)$. From our choice of f and Z, we see that these eigenvalues are in the OLHP (i.e., have real parts strictly less than 0). From the relationship between $\tilde{s}$ and s, we thus obtain that $\overline{A}$ has n eigenvalues with magnitude ϵ in the OLHP. These eigenvalues are clearly distinct from the eigenvalues at $s = -1$, and so we have shown that all $2n$ eigenvalues of $\overline{A}$ are in the OLHP. □

[‡]These correspond to the roots $s = -1$.

DOI: 10.1002/rnc

Remark
Although we have proved this lemma from the first principles here, it is deeply connected with the philosophy for decentralized controller design introduced in our recent studies [12, 13]. In these studies, we have put forth the idea that *derivatives* of local observations used in feedback can permit stabilization and pole placement in decentralized systems. It turns out that these derivative controllers are not physically implementable, and, in practice, approximation of the derivative feedbacks using memoried controllers is needed. In [13], we have proved that a derivative feedback paradigm with memoried controller implementation can achieve stabilization (placement of eigenvalues in the OLHP) and a certain group eigenvalue placement. The eigenvalue placement result that we have obtained here can alternately be obtained through our decentralized controller design philosophy (see [13] for further connection).

Starting from the crucial lemma earlier, we can apply a simple scaling argument to design weights for which the iterative algorithm (3) converges; hence, the system of linear equations (1) is solved. In particular, we obtain the following.

Theorem 2
The weight matrices in the iterative algorithm (3) can be designed to solve the linear system (1). In particular, we first choose $\overline{Q}$, $\overline{R}$, $\overline{T}$, and $\overline{V}$ be according to Lemma 2, so that the eigenvalues of

$$\overline{A} = \begin{bmatrix} \overline{Q}G & \overline{R} \\ \overline{V}G & \overline{T} \end{bmatrix}$$

are in the OLHP. Then, we choose the weight matrices as $Q = \beta\overline{Q}$, $R = \beta\overline{R}$, $V = \beta\overline{V}$, $T = I_n + \beta\overline{T}$, and $S = -(Q + R(I - T)^{-1}V)$, where β is a positive scalar. For all sufficiently small β (i.e., for $0 < \beta < \widehat{\beta}$), the system of equations (1) is solved by the iteration (3).

Proof
We show that the conditions of Theorem 1 are met for all sufficiently small β and hence that the system of equations (1) is solved. We notice that S has been selected according to Theorem 1, and so condition (ii) of the theorem is met assuming T is full rank (invertible). Next, we wish to prove that the eigenvalues of

$$A = \begin{bmatrix} I_n + QG & R \\ VG & T \end{bmatrix}$$

are strictly within the unit circle and further that T is full rank. However, this is the same as proving the eigenvalues of

$$A - I_{2n} = \begin{bmatrix} QG & R \\ VG & T - I_n \end{bmatrix} = \beta \begin{bmatrix} \overline{Q}G & \overline{R} \\ \overline{V}G & \overline{T} \end{bmatrix}$$

are within a circle of radius 1 centered at point $(-1, 0)$ in the complex plane. Because the eigenvalues of

$$\begin{bmatrix} \overline{Q}G & \overline{R} \\ \overline{V}G & \overline{T} \end{bmatrix}$$

are in the OLHP, clearly the eigenvalues of $A - I_{2n}$ are in the desired circle for all sufficiently small β. Further, because $T = I_n + \beta\overline{T}$, it is full rank for sufficiently small β, and the result is proved. □

Thus, we have obtained one key result of our development, namely, that the system of linear equations (1) can be solved using a distributed iteration of the form (2), for general G. To the best of our knowledge, this is the first result in either the distributed computation or the numerical methods literature showing that a distributed iteration with a single extra storage variable (or, equivalently, a three-term recursion with diagonal preconditioning) is capable of solving the system of linear

DOI: 10.1002/rnc

equations (1) in the general case. We note that our result extends the early result of Fisher and Fuller, in that it shows how one further memory element can broaden the class of linear equations amenable to solution from those with sequential full rank G to those with arbitrary full rank G.

We also briefly conceptualize the design given in Theorem 2. Fundamentally, the design constitutes a time scaling of the design for continuous-time dynamics implicitly codified in Lemma 2. More precisely, the continuous-time design in Lemma 2 is based on approximating a derivative feedback using a memoried control. This approximation is made to approach the derivative-based control dynamics very rapidly, and, in consequence, we are able to guarantee convergence. In contrast, the discrete-time iteration considered here, in some sense, only admits a coarser approximation of a derivative feedback, and so we are forced to slow down both the approximation and the desired derivative-feedback-based dynamics to achieve convergence; the scaling argument earlier does this. The contrast between the two cases exposes that a continuous-time decentralized control could be achieved arbitrarily quickly with sufficient actuation and precision, although the settling rate of a discrete-time iteration is fundamentally limited and should be properly designed. In the next section, we will demonstrate that use of extra memory also permits acceleration of this settling process compared with an iteration without such memory.

We again connect our results with the classical numerical methods literature. We recall that the work [3] provided a *stationary second-degree method* that also uses extra memory elements to iteratively solve the linear system problem (1). The primary difference between [3] and our work is that our algorithm numerically solves (1) in a distributed way, that is, in each iteration, an agent updates its value only based on the local information including itself and the neighbors, as specified by the topology matrix G. Because of the structural limitation forced by the distributed nature of the problem, we have been motivated to obtain a result for a general G. In addition, of relevance to our work, the article [7] also provides a distributed computing algorithm that uses extra memory, with the purpose of solving another type of distributed computing task—the consensus problem. However, in our work, we consider designing a distributed algorithm using a general topology matrix G, rather than a topology matrix that is a doubly-stochastic matrix, as is considered in [7]. Moreover, the consensus problem addressed in [7] requires the dynamics of the system to be in an **invariant manifold** (specifically, the sum of the processors' states must remain the same at all times), whereas the solution of our algorithm does not have this property.

2.3. Accelerating convergence using the memoried iteration

Fast settling is desirable for many distributed-algorithms problems. Here, we will show how the memoried iteration ((2) and (3)) can be designed for fast solution of (1) for the broad class of G that have a nested sequence of n principal minors of full rank. Precisely, we will show that the memoried iteration permits fast settling compared with the basic iteration where each processor updates its state solely based on its current observation (equivalently, a two-term recurrence with diagonal preconditioning).

To permit comparison, we begin by reviewing the convergence rate analysis for the basic distributed iteration without extra memory. In this case, it is easy to see that the convergence rate is governed by the *spectral radius* of the matrix $I_n + KG$, where K is a diagonal matrix that is amenable to design. For the case where G has a sequence of nested principal minors of full rank, we can easily verify that a K can be designed for convergence (i.e., to make the spectral radius less than 1) by using Lemma 1 together with a simple scaling argument. Furthermore, when this design strategy is used, the best achievable spectral radius depends on the **condition number** z of KG, that is, the ratio between the largest magnitude and the smallest magnitude among the eigenvalues of KG. In particular, the best achievable spectral radius turns out to bounded by $1-(2/(z+1))$ (see, e.g., [1] for the simple argument). Several recent studies have pursued optimization of the condition number through diagonal scaling for particular matrix classes (e.g., symmetric positive-definite matrices [11, 15, 20]). However, in general, condition numbers may remain large upon diagonal scaling, and the convergence rate for an iteration without memory will be large.

Here, we will demonstrate that the memoried iteration (3) can be designed to achieve much faster convergence than the basic iteration. In particular, we will show that, when a diagonal scaling

DOI: 10.1002/rnc

according to Lemma 1 yields a condition number z, the iteration with memory can be designed to obtain a spectral radius that is approximately $1-(2/\sqrt{z})$ for large z. The following theorem formalizes the design.

Theorem 3
Consider the system of linear equations (1) in the case where G has a sequence of n nested principal minors of full rank. Let K be a diagonal matrix designed according to Lemma 1, that is, let K be a diagonal matrix such that the eigenvalues of KG are real and positive. Furthermore, assume without loss of generality that the largest eigenvalue of KG is 1 and that its condition number is $z>1$. We design the weight matrices for the iteration (3) as follows. We choose $Q=-4\left(\sqrt{z}/(\sqrt{z}+1)\right)^2 K$, $T=\left(1-(2/(\sqrt{z}+1))\right)^2 I_n$, $R=rI_n$, and $V=vQG$, where $rv=\left(1-(2/(\sqrt{z}+1))\right)^2$. Then the spectral radius of the iteration's (3) full state matrix equals $1-(2/(\sqrt{z}+1))$. Thus, by choosing $S=-(Q+R(I-T)^{-1}V)$, we can use the iteration to distributively solve the system of linear equations (1), with convergence rate governed by the spectral radius $1-(2/(\sqrt{z}+1))$.

Proof
We will prove that the $2n$ eigenvalues of A all have magnitude equal to $2/(\sqrt{z}+1)$ and hence that the spectral radius of A is that value. To prove this, we first note that the eigenvalues of QG, which we will call $u_1,\ldots,u_n$, are between $-4\left(\sqrt{z}/(\sqrt{z}+1)\right)^2$ and $-4\left(1/(\sqrt{z}+1)\right)^2$. We shall relate the eigenvalues of A with those of QG and hence prove the result.

Specifically, we note that the state matrix A for the design above is given by

$$A=\begin{bmatrix} I_n+QG & rI_n \\ vQG & tI_n \end{bmatrix},$$

where $t=\left(1-(2/(\sqrt{z}+1))\right)^2$. We consider an eigenvalue decomposition of the matrix QG, that is, we write $QG=H\Lambda H^{-1}$, where H contains the right eigenvectors and generalized eigenvectors of QG and Λ is an upper triangular matrix with diagonal entries equal to the eigenvalues $u_1,\ldots,u_n$ of KG. We now consider the matrix

$$\tilde{A}=\begin{bmatrix} H^{-1} & \\ & H^{-1} \end{bmatrix} A \begin{bmatrix} H & \\ & H \end{bmatrix}.$$

Because $\tilde{A}$ is obtained from A through a similarity transform, it has the same eigenvalues. However, we notice that

$$\tilde{A}=\begin{bmatrix} I_n+\Lambda & rI_n \\ v\Lambda & tI_n \end{bmatrix}$$

is a block upper triangular (upon appropriate permutation), with diagonal blocks given by

$$\begin{bmatrix} 1+u_i & r \\ u_i v & t \end{bmatrix},$$

for $i=1,\ldots,n$. Thus, the eigenvalues of A and hence $\tilde{A}$ are given by the eigenvalues of these diagonal blocks, or in other words the roots c of the polynomials $c^2-c(t+u_i+1)+(t+tu_i-rvu_i)$, $i=1,\ldots,n$. However, because we have chosen $rv=t$, we note that the eigenvalues of A are, in fact, the roots of $c^2-c(t+u_i+1)+t$, for $i=1,\ldots,n$. For u_i in the range specified and for $t=\left(1-(2/(\sqrt{z}+1))\right)^2$, it is easy to check that the roots c are complex, and hence each of their magnitudes is $\sqrt{t}$. Thus, we recover that the spectral radius of the matrix A is $1-(2/(\sqrt{z}+1))$. The remainder of the theorem follows automatically from Theorem 1. □

Compared with the best achievable spectral radius $1-(2/(z+1))$ in the cases without memory, the spectral radius of the matrix A is reduced to $1-(2/(\sqrt{z}+1))$ after we add a memoried iteration to the system. Hence, the converging process is accelerated . Also as stated earlier, when z is large, the spectral radius is approximately $1-(2/\sqrt{z})$.

DOI: 10.1002/rnc

We connect our convergence rate design result with the associated studies on linear distributed algorithm design and numerical solution methods. We note that the previous study on memoried linear distributed algorithms [7] showed through simulation that memoried iterations can achieve a factor of 10 improvement in settling time but did not attempt a performance analysis nor pursue design/optimization of weights. In comparison, our work provides a systematic design of memoried iterations for fast convergence. Also of interest, the scaling of performance with condition number in our design is closely matched with the performance scaling obtained for stationary two-term iterative methods in [3]. However, because decentralization is not requisite in [3], the algorithm structure and the conditions required on the interaction topology's spectrum (as well as the method of analysis) are different.

Examples

We include two small simulation examples to illustrate the efficacy of the design methodology.

2.4. Example 1: a 15-node example

We compare a basic iteration without extra memory with a memoried iteration that is designed for fast convergence (Theorem 3) in a small example.

We generate G and b as follows. First, we generate a graph Γ with $n = 15$ vertices, denoted 1, $\cdots$, n, as shown in Figure 1. We define G as follows: if there is not an edge between two different vertices i and j, the entry g_{ij} in the associated graph matrix G is set to zero. If there is an edge, the weight g_{ij} is set to the inverse of the distance between vertex i and j in the plane (in Figure 1). Finally, the diagonal entries of G are chosen to be larger than the absolute sum of the off-diagonal entries, in which case G has a sequence of n nested principal minors of full rank. In this example, the condition number z for G is 20.06.

As stated in Theorem 3, by choosing appropriate diagonal matrices K, Q, T, R, V and $S = -(Q + R(I - T)^{-1}V)$, we can use the memoried iteration (3) to solve the linear equation (1) and achieve a fast convergence rate. Here, we compare this memoried iteration with a basic iteration without extra memory (where the same K is used). In particular, Figure 2 below compares the norm of the difference vector $x_d[k] = x[k] - G^{-1}b$ in the two cases as a function of time. We note that the initial guesses $x[0]$ and the initial memory variables $z[0]$ were chosen randomly in this simulation.

Theoretically, we know that the full state matrix A has a spectral radius of $1 - (2/(\sqrt{z} + 1)) = 0.6349$, whereas the best spectral radius for the case without memory is $1 - (2/(z + 1)) = 0.9050$. Hence, the iteration with memory is expected to converge faster than the one without memory.

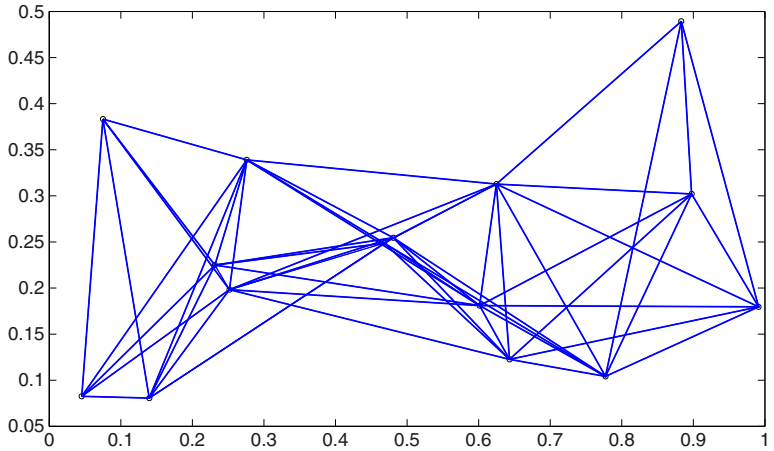

Figure 1. The graph associated with matrix G.

DOI: 10.1002/rnc

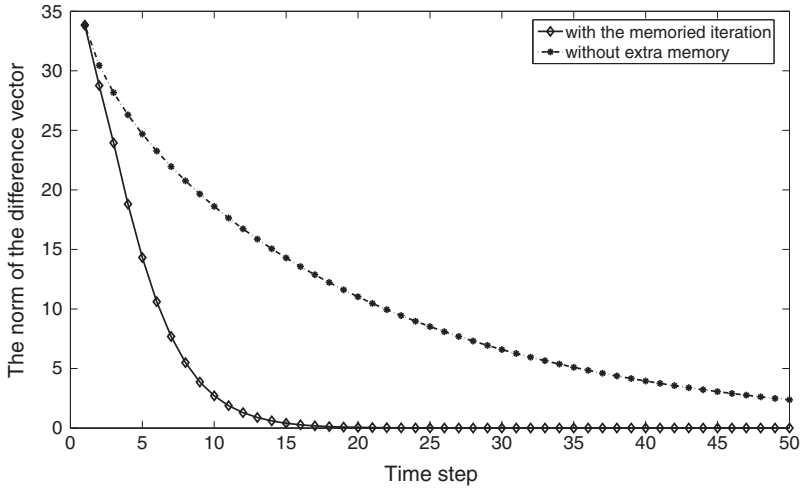

Figure 2. The two different converging rates.

2.5. *Example 2: comparing settling rates for randomly generated G*

We compare the performance of the memoried and non-memoried algorithms for randomly generated symmetric G of dimension 10×10. In particular, we consider $G = BB^T$, where B is a dense matrix whose entries are independent Gaussian random variables with zero mean and unit variance. For such G, we compare the spectral radius of the non-memoried and memoried distributed iterations that can be used to solve a system of the form $G\mathbf{x} = \mathbf{b}$ (assuming identical weights at each processor, i.e., Q is proportional to I for each algorithm). We find that the average spectral radius of the non-memoried algorithm (over $10,000$ randomly generated G) is 0.995, whereas that of the memoried iteration (over the same sample of G) is 0.928. As expected, the memoried iteration significantly outperforms the non-memoried iteration for randomly generated G.

3. CONCLUSIONS AND FUTURE WORK

We have developed a methodology for designing memoried linear iterations that (i) can solve arbitrary square algebraic linear systems and (ii) can achieve provably fast solution compared with non-memoried iterations. We would like to highlight two important conclusions that can be drawn from our case study.

- Considering the distributed solution of algebraic linear equations as a specific case study for the much broader task of design distributed algorithms/controllers, this study together with [12, 13] highlights the critical role of the control/algorithm architecture in such design. These works make clear that the architecture of a distributed algorithm/controller, not only the control weights, plays a critical role in its performance. Choosing the proper architecture for distributed control is a difficult problem, but our initial efforts begin to make clear that memoried controls are important for high performance.
- The optimal design provided for symmetric G gives an indication of the advantage of iterations having a single memory element in each processor in comparison with non-memoried ones, in this case, yielding a spectral radius whose distance from the unit circle scales inversely with the square root of the condition number rather than with the condition number itself. Some recent and classical works have shown, in limited cases, that non-memoried iterations cannot achieve this level of performance [3, 21]; thus, our work clearly indicates the benefit of memoried iterations over non-memoried ones.

DOI: 10.1002/rnc

In conclusion, we would also like to mention several directions of future work.

- It is important to extend the developed algorithm to solving non-square systems of linear algebraic equations. For the case where the system is overdetermined and further G has full column rank, it is easy to envision an extension of the presented algorithm: a subset of the processors can be used to solve a square subsystem, whereas remaining processors are used to check whether the remaining equations are satisfied asymptotically. Similar approaches based on subdivision of the processors can also be used for underdetermined systems. We leave the details of such methods to future work.
- The use of memoried controllers/algorithms, although permitting more general and faster solutions, may also incur a cost in terms of algorithm complexity, robustness, or security. These trade-offs should be made explicit in future work.
- The further advantage that can be achieved through the use of extended memory should be studied in future work.

ACKNOWLEDGEMENTS

This work was partially supported by National Science Foundation Grants ECS-0528882, ECS-0725589, and ECS-0901137 and by National Aeronautics and Space Administration Grant NNA06CN26A.

REFERENCES

1. Young DM. A historical review of iterative methods. *Proceedings of the ACM Conference on History of Scientific and Numeric Computation*, Princeton, NJ, 13–15 May 1987; 117–123.
2. Fisher ME, Fuller AT. On the stabilization of matrices and the convergence of linear iterative processes. *Mathematical Proceedings of the Cambridge Philosophical Society* 1958; **54**(4):417–425.
3. Young DM. Second-degree iterative methods for the solution of large linear systems. *Journal of Approximation Theory* 1972; **5**(2):137–148.
4. Greenbaum A. *Iterative Methods for Solving Linear Systems*. SIAM Press: Philadelphia, PA, 1997.
5. Boyd S, Ghosh A, Prabhakar B, Shah D. Randomized gossip algorithms. *IEEE Transactions on Information Theory* 2006; **52**(6):2508–2530.
6. Boyd S, Ghosh A, Prabhakar B, Shah D. Gossip algorithms: design, analysis, and applications. *Proceedings of the IEEE INFOCOM 2005*, Miami, FL, 13–17 March 2005; 1653–1664.
7. Cao M, Spielman DA, Yeh EM. Accelerated Gossip algorithms for distributed computation. *Proceedings of the 44th Annual Allerton Conference on Communication Control and Computation*, Montecelli, IL, 27–29 September 2006; 952–959.
8. Kempe D, Dobra A, Gehrke J. Gossip-based computation of aggregate information. In *44th IEEE Symposium on Foundations of Computer Science (FOCS'03)*, Cambridge, MA, 11–14 October 2003.
9. Roy S, Herlugson K, Saberi A. A control-theoretic approach on distributed discrete-valued decision-making in networks of sensing agents. *IEEE Transactions on Mobile Computing* 2006; **5**(8):945–957.
10. Siljak D. *Decentralized Control Of Complex Systems*. Academic Press: Boston, MA, 1994.
11. Roy S, Saberi A. Scaling: a canonical design problem for networks. *International Journal of Control* 2007; **80**(8):1342–1353.
12. Wan Y, Roy S, Saberi A, Stoorvogel A. A multiple-derivative and multiple-delay paradigm for decentralized controller design: uniform rank systems. *Journal of Discrete, Continuous, and Impulsive Systems,* Special Issue in Honor of Hassan Khalil's 60th Birthday 2010; **17**(6):883–907.
13. Wan Y, Roy S, Saberi A, Stoorvogel A. The design of multi-lead-compensators for stabilization and pole placement in double-integrator networks. *IEEE Transactions on Automatic Control* 2010; **55**(12):2870–2875.
14. Wan Y, Roy S. An explicit formula for difference between Laplacian-eigenvector components using coalesced graphs. *Linear Algebra and its Applications* 2010; **433**(7):1329–1335.
15. Wan Y, Roy S, Saberi A. A new focus in the science of networks towards methods for design. *Proceedings of the Royal Society* 2008; **464**(2091):513–535.
16. Boyd S. Convex optimization of graph Laplacian eigenvalues. *Proceedings of the International Congress of Mathematicians*, Madrid, Spain, 22–30 August 2006; 1311–1319.
17. Rugh WJ. *Linear System Theory*. Prentice Hall: Upper Saddle River, NJ, 1996.
18. Roy S, Saberi A, Herlugson K. Formation and alignment of distributed sensing agents with double-integrator dynamics. *IEEE Press Monograph on Sensor Network Operations* 2006.
19. Corfmat JP, Morse AS. Stabilization with decentralized feedback control. *IEEE Transactions on Automatic Control* 1976; **21**(1):679–682.
20. Wan Y, Roy S, Saberi A. Network design problems for controlling virus spread. *Proceedings of 46th IEEE Conference on Decision and Control*, New Orleans, LA, USA, 12–14 December 2007; 3925–3932.
21. Olshevsky A, Tsitsiklis JN. Convergence speed in distributed consensus and averaging. *SIAM Journal on Control and Optimization* 2009; **48**(1):33–55.

DOI: 10.1002/rnc