# Distributed and Recursive Parameter Estimation in Parametrized Linear State-Space Models

S. Sundhar Ram, Venugopal V. Veeravalli, and Angelia Nedić

***Abstract*—We consider a network of sensors deployed to sense a spatio-temporal field and infer parameters of interest about the field. We are interested in the case where each sensor's observation sequence is modeled as a state-space process that is perturbed by random noise, and the models across sensors are parametrized by the *same* parameter vector. The sensors collaborate to estimate this parameter from their measurements, and to this end we propose a distributed and recursive estimation algorithm, which we refer to as the incremental recursive prediction error algorithm. This algorithm has the distributed property of incremental gradient algorithms and the on-line property of recursive prediction error algorithms.**

***Index Terms*—Incremental recursive prediction error (IRPE), recursive prediction error (RPE).**

## I. Introduction

We consider a network of sensors deployed to sense a spatio-temporal field and infer parameters of interest about the field. We are interested in the case where each sensor's observation sequence is modeled as a state-space process that is perturbed by random noise, and the models across sensors are parametrized by the *same* unknown parameter vector. The network goal is to estimate this unknown parameter using the sensor observations. State-space models arise in many applications, directly, or as linear approximations to non-linear models [1].

We propose a *distributed and recursive* estimation procedure, which is suitable for in-network processing. Each sensor locally processes its own data and shares only a summary of this data with other sensors in each time slot. The sensors form a cycle and update incrementally, whereby each sensor updates the estimate using its local information and the received estimate from its upstream neighbor, and passes the updated estimate to its downstream neighbor. Such an incremental computational model is a recognized technique to reduce the total in-network communication and we refer the reader to [2], [3] for implementation issues. Furthermore, the sensor updates are generated recursively from every new measurement using only a summary statistic of the past measurements. This enables the network to have an estimate at all times and also allows each sensor to purge its old measurements reducing the memory requirements.

The estimation criterion that we use is a direct extension of the recursive prediction error (RPE) criterion of [1] to the multi-sensor case. We call it the *incremental recursive prediction error* (IRPE) criterion. We propose an algorithm that builds on the incremental gradient algorithm in the same way the RPE algorithm of [1] builds on the standard gradient algorithm. We call the algorithm the IRPE algorithm.

A survey of centralized methods for estimation in linear systems is available in [1]. A related algorithm is the *parallel recursive prediction error* algorithm proposed in [4] that updates the components of the parameter vector in parallel. This technical note extends our earlier work [5], where we considered the problem of recursive and incremental estimation for non auto-regressive stationary models. Also related is the incremental LMS algorithm discussed in [3] for fitting linear regression models.

The rest of the technical note is organized as follows. We formulate the problem, and introduce our notation in Section II. We then discuss the standard recursive prediction error algorithm [1] and the incremental gradient algorithm of [6] in Section III. These are then used to develop the IRPE algorithm in Section IV, where we also state our main convergence result. We discuss the proof for the convergence of the algorithm in Appendix A. We conclude in Section V.

## II. Problem Formulation

We consider a network of $m$ sensors, indexed $1, \ldots, m$, deployed to sense a spatio-temporal diverse field to determine the value of some quantity of interest, denoted by $x$, $x \in \Re^d$. We sometimes find it convenient to use $\mathcal{I}$ to denote the set of sensors, i.e., $\mathcal{I} := \{1, \ldots, m\}$. We assume that time is slotted and each sensor sequentially senses the field once in every time slot. We denote by $r_i(k)$ the actual measurement collected by sensor $i$ at time slot $k$, and we assume that $r_i(k) \in \Re^p$. The goal is to use the sensor measurements to estimate $x$.

To aid in the estimation process each sensor has an *approximate* model for the dependence between its measurements and the unknown parameter $x$. We will use $R_i(k;x)$ to denote the model for $r_i(k)$ and consider stochastic models in which $\{R_i(k;x)\}$ has the following dynamics:

$$\begin{aligned} \Theta_i(k+1;x) &= D_i(x)\Theta_i(k;x) + W_i(k;x) \\ R_i(k+1;x) &= H_i\Theta_i(k+1;x) + V_i(k+1). \end{aligned} \tag{1}$$

The state vector $\Theta_i(k+1;x)$ is a vector of dimension $q$. We impose the following assumptions on the system and observation models.

A.1. The processes $\{W_i(k;x)\}$ and $\{V_i(k)\}$ are zero mean i.i.d. random sequences and the matrix function $D_i(x)$ is twice differentiable.

A.2. The quantities $D_i(x)$, $H_i$, $\mathsf{E}[\Theta_i(0;x)]$, $\mathsf{Cov}(W_i(0;x))$, $\mathsf{Cov}(W_i(k;x))$ and $\mathsf{Cov}(V_i(k))$ are available at sensor $i$.

A.3. At all the sensors a closed and convex set $X$ is available such that the system in (1) is stable, observable and controllable for all $x \in X$.

A.4. The sequence $\{r_i(k)\}$ is asymptotically mean stationary and exponentially stable.

A.5. The joint statistics of $\Theta_i(k+1;x)$ and $\Theta_j(k+1;x)$ are not known.

Note that $X$ may even be the entire space $\Re^d$. Asymptotic stationarity means that if we view the sequence $\{r_i(k)\}$ as the realization of a random process then that process must in the limit exhibit stationarity. Exponential stability essentially implies that what happens at time slot $s$ has very little influence on what happens at time slot $t$, when $t \gg s$. See [1, p. 170] for the precise mathematical definitions of asymptotically mean stationary and exponentially stable. A sufficient condition for (A.4) is the existence of a $x^* \in X$ such that $\{r_i(k)\}$ can be viewed as a sample path of $\{R_i(k;x^*)\}$, which additionally also has finite fourth moments ([1, p. 172]). Assumption (A.5.) implies that even if information about the dependencies between the random processes is available we do not use it, with the understanding that this is the loss in efficiency that we suffer in order to obtain a distributed algorithm. The *problem* is to estimate the parameter $x$ from the collection of sensor measurements $\{r_i(k)\}$ with an algorithm that is: (a) distributed, i.e., sensor $i$ does not share its raw measurements $\{r_i(k)\}$

Manuscript received May 11, 2008; revised October 28, 2008. First published January 08, 2010; current version published February 10, 2010. This work was supported by research grants from Vodafone and Motorola. Recommended by Associate Editor T. Zhou.

S. S. Ram and V. V. Veeravalli are with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Champaign, IL 61820-5711 USA (e-mail: ssriniv5@illinois.edu; vvv@illinois.edu).

A. Nedić is with the Department of Industrial and Enterprise Systems Engineering., University of Illinois at Urbana-Champaign, Champaign, IL 61820-5711 USA (e-mail:{angelia}@illinois.edu).

Digital Object Identifier 10.1109/TAC.2009.2037460

with any other sensor, and (b) recursive, i.e., at all times, sensor $i$ stores only a summary statistic of a constant size, i.e., size does increase with the number of measurements collected by the sensor.

## III. Preliminaries

To make the technical note self contained we briefly discuss the incremental gradient algorithm of [6] and the RPE algorithm of [1].

### A. Incremental Gradient Descent Algorithm

The incremental gradient algorithm can be used to solve optimization problems of the form

$$\min_{x\in X}\sum_{i=1}^{m} f_i(x)$$

when the function $f_i$ is known only to sensor $i$. In this algorithm, the iterates are generated according to

$$\begin{aligned} x_k &= z_{m,k} = z_{0,k+1} \\ z_{i,k+1} &= \mathcal{P}_X\left[z_{i-1,k} - \alpha_{k+1}\nabla f_i(z_{i-1,k})\right]. \end{aligned} \tag{2}$$

Here, the scalar $\alpha_{k+1} > 0$ is the step-size, $\mathcal{P}_X$ denotes the projection onto the set $X$ and $\nabla f_i$ denotes the gradient of the function $f_i$. In the $k$-th iteration sensor $i$ receives the iterate $z_{i-1,k+1}$ from sensor $i-1$, incrementally updates it using the gradient of the locally available function $f_i$ and passes the updated iterate to the sensor $i+1$.

### B. Kalman Predictor

We write $R_i^k(x)$ to denote the collection of random variables $\{R_i(1;x),\ldots,R_i(k;x)\}$, which should be viewed as a collection of random variables parametrized by $x$ and *not as a function of* $x$. Furthermore, in line with our notation, $r_i^k$ denotes the collection $\{r_i(1),\ldots,r_i(k)\}$, and $r^k$ denotes the collection $\{r_1^k,\ldots,r_m^k\}$.

For $x\in X$, we assumed that the system in (1) is stable, observable and controllable. The Kalman gain for the system therefore converges to a finite time-invariant value [7]. Let $G_i(x)$ be the Kalman gain for the state-space system in (1), which is determined from $D_i(x), H_i$, $\mathsf{Cov}\,(W_i(k;x))$, and $\mathsf{Cov}\,(V_i(k))$ as the solution to the Riccati equation [1]. Define $F_i(x) = D_i(x) - G_i(x)H_i$. The Kalman predictor, $g_{i,k+1}\left(x; r_i^k\right)$, is defined as

$$\begin{aligned} \phi_{i,k+1}\left(x; r_i^k\right) &= F_i(x)\phi_{i,k}\left(x; r_i^{k-1}\right) + G_i(x) r_i(k) \\ g_{i,k+1}\left(x; r_i^k\right) &= H_i\phi_{i,k+1}\left(x; r_i^k\right) \end{aligned} \tag{3}$$

with $\phi_{i,0}\left(x; r_i^0\right) = \mathsf{E}\left[\Theta_i(0;x)\right]$. For later reference we next determine the gradient of $g_{i,k+1}\left(x; r_i^k\right)$. Let $x^{(\ell)}$ denote the $\ell$-th component of $x$, and define

$$\begin{aligned} \zeta_{i,k}^{(\ell)}\left(x; r_i^{k-1}\right) &= \frac{\partial \phi_{i,k}\left(x; r_i^{k-1}\right)}{\partial x^{(\ell)}}, \quad \nabla^{(\ell)}F_i(x) = \frac{\partial F_i(x)}{\partial x^{(\ell)}} \\ \eta_{i,k}^{(\ell)}\left(x; r_i^{k-1}\right) &= \frac{\partial g_{i,k}\left(x; r_i^{k-1}\right)}{\partial x^{(\ell)}}, \quad \nabla^{(\ell)}G_i(x) = \frac{\partial G_i(x)}{\partial x^{(\ell)}}. \end{aligned}$$

Thus the gradient $\nabla g_{i,k}\left(x; r_i^{k-1}\right)$ is the $p\times d$ matrix

$$\nabla g_{i,k}\left(x; r_i^{k-1}\right) = \left[\eta_{i,k}^{(1)}\left(x; r_i^{k-1}\right)\ldots\eta_{i,k}^{(d)}\left(x; r_i^{k-1}\right)\right]. \tag{4}$$

By differentiating in (3), we can immediately see that

$$\begin{aligned} \begin{bmatrix} \phi_{i,k+1}\left(x; r_i^k\right) \\ \zeta_{i,k+1}^{(\ell)}\left(x; r_i^k\right) \end{bmatrix} &= \begin{bmatrix} F_i(x) & 0 \\ \nabla^{(\ell)}F_i(x) & F_i(x) \end{bmatrix} \begin{bmatrix} \phi_{i,k}\left(x; r_i^{k-1}\right) \\ \zeta_{i,k}^{(\ell)}\left(x; r_i^{k-1}\right) \end{bmatrix} \\ &\quad + \begin{bmatrix} G_i(x) \\ \nabla^{(\ell)}G_i(x) \end{bmatrix} r_i(k) \end{aligned}$$

$$\begin{bmatrix} g_{i,k+1}\left(x; r_i^k\right) \\ \eta_{i,k+1}^{(\ell)}\left(x; r_i^k\right) \end{bmatrix} = \begin{bmatrix} H_i(x) & 0 \\ 0 & H_i(x) \end{bmatrix} \begin{bmatrix} \phi_{i,k+1}\left(x; r_i^k\right) \\ \zeta_{i,k+1}^{(\ell)}\left(x; r_i^k\right) \end{bmatrix}. \tag{5}$$

### C. RPE Criterion and Algorithm

We illustrate the RPE criterion and algorithm of [1] by using it to estimate $x$ *only* using sensor $i$'s measurements $\{r_i(k)\}$. Thus, there is no collaboration with the other agents. For this system, the recursive prediction error criterion is

$$f_i(x) = \lim_{N\to\infty}\frac{1}{N}\sum_{k=1}^{N}\left\|r_i(k) - g_{i,k}\left(x; r_i^{k-1}\right)\right\|^2. \tag{6}$$

Note that under assumption (A.4) on the observation sequence $\{r_i(k)\}$, the limit on the RHS of (6) depends only on $x$ and not on $\{r_i(k)\}$. The RPE algorithm generates a sequence of iterates $\{x_k\}$ that converges to a local minimum of the function $f_i(x)$. The RPE algorithm is essentially a gradient projection algorithm with stochastic errors. Suppose the standard gradient projection algorithm is used to minimize $f_i(x)$, then the iterates are generated according to

$$x_{k+1} = \mathcal{P}_X\left[x_k - \alpha_{k+1}\nabla f_i(x_k)\right].$$

The iterates of the RPE algorithm are obtained by approximating $\nabla f_i(x_k)$ to make the algorithm recursive. The approximation involves: (a) an LMS-like approximation for the gradient, and (b) an approximation to make the LMS approximations recursive. If the model for the measurements is a simple regression model then the LMS approximation itself is recursive and approximation (b) is not required. Thus, the RPE generalizes the LMS algorithm to state-space systems. We refer the reader to [1] for the details of the algorithm. The final algorithm can be stated as follows:

$$\begin{aligned} \begin{bmatrix} h_{k+1} \\ \xi_{k+1}^{(\ell)} \end{bmatrix} &= \begin{bmatrix} H_i & 0 \\ 0 & H_i \end{bmatrix} \begin{bmatrix} \psi_{k+1} \\ \chi_{k+1}^{(\ell)} \end{bmatrix} \\ \epsilon_{k+1} &= r(k+1) - h_{k+1} \\ \underline{x}_{k+1}^{(\ell)} &= x_k^{(\ell)} - \alpha_{k+1}\left(\xi_{k+1}^{(\ell)}\right)^T\epsilon_{k+1} \\ \underline{x}_{k+1} &= \left[\underline{x}_{k+1}^{(1)}\cdots\underline{x}_{k+1}^{(d)}\right]^T \\ x_{k+1} &= \mathcal{P}_X\left[\underline{x}_{k+1}\right] \\ \begin{bmatrix} \psi_{k+2} \\ \chi_{k+2}^{(\ell)} \end{bmatrix} &= \begin{bmatrix} F_i(x_{k+1}) & 0 \\ \nabla^{(\ell)}F_i(x_{k+1}) & F_i(x_{k+1}) \end{bmatrix} \begin{bmatrix} \psi_{k+1} \\ \chi_{k+1}^{(\ell)} \end{bmatrix} \\ &\quad + \begin{bmatrix} G_i(x_{k+1}) \\ \nabla^{(\ell)}G_i(x_{k+1}) \end{bmatrix} r(k+1). \end{aligned} \tag{7}$$

Here $l = 1,\ldots,d$. The algorithm is initialized with values for $\psi_1, \chi_1^{(\ell)}$ and $x_0$. Observe that to update $x_k$, the algorithm requires only $r(k+1)$, $\chi_{k+1}^{(1)},\ldots,\chi_{k+1}^{(d)}$ and $\psi_{k+1}$, and therefore, it is recursive. Under (A.1)–(A.4) and convergent $k\alpha_k$, it is shown in [1, Theorem 4.3] that $\{x_k\}$ converges to a local minimum of $f_i(x)$ in (6) over the set $X$, with probability 1.

## IV. IRPE Algorithm

The direct extension of the RPE criterion in (6) to the case when all the $m$ sensors cooperate to estimate $x$ is

$$f(x) = \sum_{i=1}^{m} f_i(x)$$

$$= \sum_{i=1}^{m} \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} \left\| r_i(k) - g_{i,k}\left(x; r_i^{k-1}\right)\right\|^2. \quad (8)$$

We refer to this criterion as the *incremental recursive prediction error criterion.* The function $f_i$ is potentially available only to sensor $i$. If the incremental gradient descent algorithm is used to minimize the function $f(x)$, then the iterates are generated according to

$$\begin{aligned} x_k &= z_{m,k} = z_{0,k+1} \\ z_{i,k+1} &= \mathcal{P}_X\left[z_{i-1,k} - \alpha_{k+1}\nabla f_i\left(z_{i-1,k}\right)\right]. \end{aligned} \quad (9)$$

The IRPE can be viewed as incremental gradient descent with stochastic errors that are generated when the term $\nabla f_i(z_{i-1,k})$ is approximated using the same two approximations that were used in the RPE algorithm. The first is the LMS like approximation, and the second is the recursive approximation to make the LMS approximation recursive. If only the LMS approximation is made, which would be the case in a simple linear regression problem, the IRPE algorithm simplifies to the incremental LMS algorithm of [3].

Formally, the iterates are generated by the following relations for $i \in \mathcal{I}$ and $\ell = 1, \ldots, d$

$$x_k = z_{m,k} = z_{0,k+1}$$

$$\begin{bmatrix} h_{i,k+1} \\ \xi_{i,k+1}^{(\ell)} \end{bmatrix} = \begin{bmatrix} H_i & 0 \\ 0 & H_i \end{bmatrix} \begin{bmatrix} \psi_{i,k+1} \\ \chi_{i,k+1}^{(\ell)} \end{bmatrix} \quad (10)$$

$$\epsilon_{i,k+1} = r_i(k+1) - h_{i,k+1} \quad (11)$$

$$\underline{z}_{i,k+1}^{(\ell)} = z_{i-1,k+1}^{(\ell)} - \alpha_{k+1}\left(\xi_{i,k+1}^{(\ell)}\right)^T \epsilon_{i,k+1} \quad (12)$$

$$\underline{z}_{i,k+1} = \left[\underline{z}_{i,k+1}^{(1)} \cdots \underline{z}_{i,k+1}^{(d)}\right]^T \quad (13)$$

$$z_{i,k+1} = \mathcal{P}_X[\underline{z}_{i,k+1}] \quad (14)$$

$$\begin{bmatrix} \psi_{i,k+2} \\ \chi_{i,k+2}^{(\ell)} \end{bmatrix} = \begin{bmatrix} F_i(z_{i,k+1}) & 0 \\ \nabla^{(\ell)}F_i(z_{i,k+1}) & F_i(z_{i,k+1}) \end{bmatrix} \begin{bmatrix} \psi_{i,k+1} \\ \chi_{i,k+1}^{(\ell)} \end{bmatrix} + \begin{bmatrix} G_i(z_{i,k+1}) \\ \nabla^{(\ell)}G_i(z_{i,k+1}) \end{bmatrix} r_i(k+1). \quad (15)$$

The initial values for the recursion are fixed at $x_0 = x_s$, $\psi_{i,1} = \psi_{i,s}$ and $\chi_{i,1}^{(\ell)} = \chi_{i,s}^{(\ell)}$. To see that the algorithm has a distributed and recursive implementation assume sensor $i-1$ communicates $z_{i-1,k+1}$ to sensor $i$ in slot $k+1$. Sensor $i$ then uses[1] $r_i(k+1)$ to updates the iterate $z_{i-1,k+1}$ to generate $z_{i,k+1}$. This is then passed to the next sensor in the cycle. Observe that in updating $z_{i-1,k+1}$, sensor $i$ requires only $\chi_{i,k+1}^{(1)}, \ldots \chi_{i,k+1}^{(d)}$ and $\psi_{i,k+1}$, which were calculated by sensor $i$ in the previous time slot. Thus, the algorithm is recursive and distributed. Furthermore, note that sensor $i$ only needs to know its own system matrices $H_i$, $F_i(x)$ and $G_i(x)$.

To establish convergence we will consider a hypothetical centralized system and prove that the iterates generated by the IRPE are identical to the iterates generated by the RPE algorithm when used on the hypothetical centralized system. We only state the final result here and discuss the proof in Appendix A.

*Theorem 1:* Let (A.1)–(A.5) hold. Moreover, let the step-size $\alpha_k$ be such that $k\alpha_k$ converges. Then, the iterates $x_k$ generated by the IRPE algorithm in (10)–(15) converge to a local minimum of $f(x)$ in (8) over the set $X$, with probability 1.

We have not included an explicit input in modeling the system. The results immediately follow when there is a deterministic open-loop input $\{u_i(k)\}$ that drives the system in (1). Of course, $\{u_i(k)\}$ should be known to sensor $i$. Another immediate extension is to the case when the matrix $H_i$ and noise $V_i(k)$ are also parametrized by $x$.

[1]We are assuming that sensor $i$ obtains its measurement before it receives the iterate. From an implementation perspective, each time slot can be divided into two parts. In the first part, the sensors make measurements and in the second part they process.

## V. Discussion

The IRPE algorithm ignores any information about the parameter available in the joint statistics of the random process $\{\Theta_i(k;x)\}$ and $\{\Theta_j(k;x)\}$. A centralized system, on the other hand, can use the joint density information to generate better estimates. Thus, there is a trade-off between the quality of the estimates and the 'distributedness' of the estimation scheme. For numerical simulations that capture this trade-off and an application of the IRPE algorithm to localizing a diffusing source, we refer the reader to [8].

To truly understand the performance of the algorithm in practical settings, we need to obtain convergence results when there are communication errors. Further, we have considered a simple class of networks where the topology is fixed. It is important to obtain an algorithm that is similar to the IRPE for networks with a random and time-varying topologies.

## Appendix

*Proof of Theorem 1:* For positive integers $a$ and $b$, let $\mathcal{M}_{a\times b}$ be the vector space of all real matrices of dimensions $a\times b$. A block vector in $\mathcal{M}_{a\times b}$ is a vector whose elements are from $\mathcal{M}_{a\times b}$. The length of a block vector is the number of block elements. In a similar manner, block matrices in $\mathcal{M}_{a\times b}$ are matrices where each element is itself a matrix from $\mathcal{M}_{a\times b}$. While writing block matrices we will allow for a slight abuse of notation and use 0 and $I$ to denote the zero and identity matrices, respectively. Their dimensions can be unambiguously fixed from the dimensions of the other blocks in the block matrix. We will use $\boldsymbol{U}_b^a$, $b \le m$, to denote the unit block vector in $\mathcal{M}_{a\times a}$ of length $m$, with the $b$th block equal to the identity matrix in $\mathcal{M}_{a\times a}$ and all the other blocks equal to the zero matrix in $\mathcal{M}_{a\times a}$. We allow $i, j$ to take values in the set $\mathcal{I} = \{1, \ldots, m\}$. We define $\delta[\cdot]$ as the Kronecker delta. Recall that the dimension of the matrices $\Theta_i(k;x)$ is $q$, the dimension of the measurement $r_i(k)$ is $p$, and the dimension of the parameter vector $x$ is $d$.

*Hypothetical Centralized System:* Without loss of generality, assume that each time slot has duration of $m$ time units. Consider a hypothetical centralized scheme where at time $mk+j$, sensor $j$ communicates $r_j(k+1)$ to the fusion center over a perfect delayless link. For $i \neq j$, sensor $i$ communicates a predetermined constant value, say 0, that does not convey any information about the value taken by the parameter $x$.

Denote the sequence communicated by a sensor $i$ by $\{\bar{r}_i(mk+j)\}$, with

$$\bar{r}_i(mk+j) = r_i(k+1)\delta[i-j]. \quad (16)$$

Next, denote the observation sequence at the fusion center by $\{\tilde{r}(mk+j)\}$, where

$$\tilde{r}(mk+j) = \left[\bar{r}_1(mk+j) \quad \ldots \quad \bar{r}_m(mk+j)\right]^T = \boldsymbol{U}_j^p r_j(k+1).$$

The model for $\{\tilde{r}(mk+j)\}$, which we denote by $\{\tilde{R}(mk+j;x)\}$, can be defined starting from $\{R_i(k;x)\}$ in an identical manner. We now consider the problem of estimating $x$ from observation sequence $\{\tilde{r}(mk+j)\}$ using the RPE algorithm. To use the RPE algorithm, the random process $\{\tilde{R}(mk+j;x)\}$ has to be represented as the output vector of a suitably defined state-space system. We do this next using the model for $r_i(k)$ in (1).

*State Space Model for* $\{\tilde{r}(mk+j;x)\}$*:* Observe that to use the RPE algorithm the state and observation matrices must be fixed and not change with time. Note that from (16), we have $\bar{R}_i(mk+j;x) =$

$R_i(k+1;x)\delta[i-j]$. Let $\bar{D}_i(x)$ be the following $m \times m$ block matrix in $\mathcal{M}_{q\times q}$:

$$\bar{D}_i(x) = \begin{bmatrix} 0 & I & 0 & \cdot & \cdot & 0 \\ 0 & 0 & I & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdot & I \\ D_i(x) & 0 & 0 & \cdot & \cdot & 0 \end{bmatrix}. \tag{17}$$

Also, define $\bar{H}_i = H_i\,(\boldsymbol{U}_1^q)^T$, and note that $\bar{H}_i\boldsymbol{U}_j^q = H_i\,(\boldsymbol{U}_1^q)^T\,\boldsymbol{U}_j^q = H_i\delta[j-1]$. Define $\bar{\Theta}_i(0;x) = \boldsymbol{U}_i^q\Theta_i(0;x)$, and

$$\bar{\Theta}_i(mk+j;x) = \begin{cases} \boldsymbol{U}_{i-j+1}^q\Theta_i(k+1;x) & \text{if } j \le i \\ \boldsymbol{U}_{m+1-(j-i)}^q\Theta_i(k+2;x) & \text{if } j > i \end{cases}$$
$$\bar{W}_i(mk+j;x) = \boldsymbol{U}_m^q W_i(k+1;x)\delta[i-j], \text{ and}$$
$$\bar{V}_i(mk+j) = V_i(k+1)\delta[i-j].$$

We next state the following result that describes the evolution of $\{\bar{R}_i(n+1;x)\}$. The result can be verified by substituting from the definitions defined above and a proof is available in [8].

*Proposition 1:* For all $n \ge 0$, we have

$$\bar{\Theta}_i(n+1;x) = \bar{D}_i(x)\bar{\Theta}_i(n;x) + \bar{W}_i(n;x), \tag{18}$$
$$\bar{R}_i(n+1;x) = \bar{H}_i\bar{\Theta}_i(n+1;x) + \bar{V}_i(n+1). \tag{19}$$

From (18) and (19), we provide evolution equations for $\{\tilde{R}(n;x)\}$. Define

$$\tilde{F}(x) = \operatorname{diag}\left(\bar{F}_1(x),\ldots,\bar{F}_m(x)\right)$$
$$\tilde{H}(x) = \operatorname{diag}\left(\bar{H}_1(x),\ldots,\bar{H}_m(x)\right)$$
$$\tilde{\Theta}(n;x) = \left[\bar{\Theta}_1(n;x)\ldots\bar{\Theta}_m(n;x)\right]^T$$
$$\tilde{W}(n;x) = \left[\bar{W}_1(n;x)\ldots\bar{W}_m(n;x)\right]^T$$
$$\tilde{V}(n;x) = \left[\bar{V}_1(n;x)\ldots\bar{V}_m(n;x)\right].$$

Using the relations in (18) and (19), we can write

$$\tilde{\Theta}(n+1;x) = \tilde{D}(x)\tilde{\Theta}(n;x) + \tilde{W}(n;x), \tag{20}$$
$$\tilde{R}(n+1;x) = \tilde{H}\tilde{\Theta}_i(n+1;x) + \tilde{V}(n+1). \tag{21}$$

Equations (20) and (21) describe the state-space model for the fusion centers observation $\{\tilde{r}(n)\}$. To use the RPE algorithm we need to evaluate the Kalman predictor for the system in (21).

*1) Time-Invariant Kalman Predictor for Centralized System:* Let us first obtain the time-invariant Kalman predictor for $\bar{R}_i(n;x)$ in (19). Fix $n = mk+j$ and define

$$\bar{\phi}_{i,n}\left(x;\bar{r}_i^{n-1}\right) = \begin{cases} \boldsymbol{U}_{i-j+1}^q\phi_{i,k+1}\left(x;r_i^k\right) & \text{if } j \le i \\ \boldsymbol{U}_{m+1-(j-i)}^q\phi_{i,k+2}\left(x;r_i^{k+1}\right) & \text{if } j > i, r \end{cases}$$
$$\bar{g}_{i,n}\left(x;\bar{r}_i^{n-1}\right) = g_{i,k+1}\left(x;r_i^k\right)\delta[j-i].$$

Define $\bar{G}_i(x) = \boldsymbol{U}_m^p G_i(x)$, and $\bar{F}_i(x) = \bar{D}_i(x) - \bar{G}_i(x)\bar{H}_i$. The matrix $\bar{F}_i(x)$ will have the same form as $\bar{D}_i(x)$ in (17) but with $D_i(x)$ replaced by $F_i(x)$. Similar to Proposition 1, we can show

$$\bar{\phi}_{i,n+1}\left(x;\bar{r}_i^n\right) = \bar{F}_i(x)\bar{\phi}_{i,n}\left(x;\bar{r}_i^{n-1}\right) + \bar{G}_i(x)\bar{r}_i(n)$$
$$\bar{g}_{i,n+1}\left(x;\bar{r}_i^n\right) = \bar{H}_i(x)\bar{\phi}_{i,n}\left(x;\bar{r}_i^n\right). \tag{22}$$

We can now obtain a predictor family for $\tilde{\Theta}(n;x)$ and $\{\tilde{R}_n(x)\}$. Define

$$\tilde{\phi}_n(x;\tilde{r}^{n-1}) = \left[\bar{\phi}_{1,n}\left(x;\bar{r}_1^{n-1}\right)\ldots\bar{\phi}_{m,n}(x;\bar{r}_m^{n-1})\right]^T$$
$$\tilde{G}(x) = \operatorname{diag}\left(\bar{G}_1(x),\ldots,\bar{G}_m(x)\right)$$
$$\tilde{g}_n\left(x;\tilde{r}^{n-1}\right) = \left[\bar{g}_{1,n}(x;\bar{r}_1^{n-1})\ldots\bar{g}_{m,n}\left(x;\bar{r}_m^{n-1}\right)\right]^T$$
$$\tilde{F}(x) = \operatorname{diag}\left(\bar{F}_1(x),\ldots,\bar{F}_m(x)\right).$$

Furthermore, from (22) one can verify that

$$\tilde{\phi}_{n+1}\left(x;\tilde{r}^n\right) = \tilde{F}(x)\tilde{\phi}_n\left(x;\tilde{r}^{n-1}\right) + \tilde{G}(x)\tilde{r}(n)$$
$$\tilde{g}_{n+1}\left(x;\tilde{r}^n\right) = \tilde{H}\phi_{n+1}\left(x;\tilde{r}^n\right). \tag{23}$$

Equation (23) is the Kalman predictor for the system in (21). For all components of $\tilde{r}(n+1)$ that are 0 the corresponding values in the predictor $\tilde{g}_{n+1}(x;\tilde{r}^n)$ is also 0. For the one component of $\tilde{r}(n+1)$ that will equal $r_i(k+1)$, for some $i \in V$, the corresponding component is $g_{i,k+1}\left(x;r_i^k\right)$.

*RPE Criterion for the Centralized System in (21):* We next evaluate the RPE criterion for the centralized system

$$\begin{aligned}\tilde{f}(x) &= \lim_{N\to\infty}\frac{1}{N}\sum_{n=1}^{N}\mathsf{E}\left[\left\|\tilde{R}(n;x^*) - \tilde{g}_n(x,\tilde{R}^n(x^*))\right\|^2\right] \\ &= \lim_{N\to\infty}\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{m}\mathsf{E}\left[\left\|\bar{R}_i(n;x^*) - \bar{g}_{i,n}\left(x,\bar{R}_i^n(x^*)\right)\right\|^2\right] \\ &= \lim_{N\to\infty}\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{m}\mathsf{E}\left[\left\|R_i(n;x^*) - g_{i,n}\left(x;R_i^n(x^*)\right)\right\|^2\right] \\ &= f(x).\end{aligned}$$

We make the following important remark. As a consequence of the assumptions on the sequences $r_i(k)$ and the models in (1) the sequence $\{\tilde{r}(n)\}$ and its model in (21) satisfy the conditions of [1, Theorem 4.3] required for the convergence of the RPE algorithm. *Thus, the sequence of iterates generated when RPE is applied to the system in (21) will converge to a local minimum of* $f(x)$. We will next show that the sequence generated by the RPE algorithm when applied to the system in (21) is identical to the sequence $\{z_{i,k}\}$ generated by the IRPE algorithm in (15).

*RPE Algorithm for Centralized System:* Here, we use the RPE algorithm to estimate $x$ from $\{\tilde{r}(n)\}$. Define $\nabla^{(\ell)}\tilde{F}(x) = \partial\tilde{F}(x)/\partial x^{(\ell)}$ and $\nabla^{(\ell)}G(x) = \partial\tilde{G}(x)/\partial x^{(\ell)}$, for $\ell = 1,\ldots,d$.

The RPE algorithm applied to the system in (21) generates the iterates $\{\tilde{x}_n\}$ as follows:

$$\begin{bmatrix}\tilde{h}_{n+1}\\ \tilde{\xi}_{n+1}^{(\ell)}\end{bmatrix} = \begin{bmatrix}\tilde{H} & 0\\ 0 & \tilde{H}\end{bmatrix}\begin{bmatrix}\tilde{\psi}_{n+1}\\ \tilde{\chi}_{n+1}^{(\ell)}\end{bmatrix}$$
$$\tilde{\epsilon}_{n+1} = \tilde{r}(n+1) - \tilde{h}_{n+1}$$
$$\underline{\tilde{x}}_{n+1}^{(\ell)} = \tilde{x}_n^{(\ell)} - \tilde{\alpha}_{n+1}\left(\tilde{\xi}_{n+1}^{(\ell)}\right)^T\tilde{\epsilon}_{n+1}$$
$$\underline{\tilde{x}}_{n+1} = \left[\underline{\tilde{x}}_{n+1}^{(1)}\cdots\underline{\tilde{x}}_{n+1}^{(d)}\right]^T$$
$$\tilde{x}_{n+1} = \mathcal{P}_X\left[\underline{\tilde{x}}_{n+1}\right]$$
$$\begin{bmatrix}\tilde{\psi}_{n+2}\\ \tilde{\chi}_{n+2}^{(\ell)}\end{bmatrix} = \begin{bmatrix}\tilde{F}(\tilde{x}_{n+1}) & 0\\ \nabla^{(\ell)}\tilde{F}(\tilde{x}_{n+1}) & \tilde{F}(\tilde{x}_{n+1})\end{bmatrix}\begin{bmatrix}\tilde{\psi}_{n+1}\\ \tilde{\chi}_{n+1}^{(\ell)}\end{bmatrix} + \begin{bmatrix}\tilde{G}(\tilde{x}_{n+1})\\ \nabla^{(\ell)}\tilde{G}(\tilde{x}_{n+1})\end{bmatrix}\tilde{r}(n+1).$$

Here, $\tilde{\alpha}_n = \alpha_{k+1}$ for $n = mk+j$ for $j = 1,\ldots,m-1$. Next, we assign the initial values for the recursion. Recall that the IRPE algorithm in (15) is initialized with the values $\psi_{i,1} = \psi_{i,s}$, $\xi_{i,1}^{(\ell)} = \xi_{i,s}^{(\ell)}$ for all $i$ and $\ell$, and $x_0 = x_s$. We let $\tilde{x}_0 = x_s$, $\tilde{\psi}_0 = [\,\bar{\psi}_{1,s}\quad\ldots\quad\bar{\psi}_{m,s}\,]^T$ and $\tilde{\xi}_0^{(\ell)} = [\,\bar{\xi}_{1,s}^{(\ell)}\quad\ldots\quad\bar{\xi}_{m,s}^{(\ell)}\,]^T$. Here $\bar{\psi}_{i,s} = \boldsymbol{U}_i^q\psi_{i,s}$ and $\bar{\xi}_{i,s}^{(\ell)} = \boldsymbol{U}_i^q\xi_{i,s}^{(\ell)}$ for all $i$ and $l$.

*Proof That* $\tilde{x}_{mk+j} = Z_{j,k+1}$: Fix $n = mk+j$. Recall that $\psi_{i,k}$ and $\chi_{i,k}^{(\ell)}$ are generated in the IRPE algorithm (10)–(15). Define for $l = 1,\ldots,d$

$$\bar{\psi}_{i,n} = \begin{cases}\boldsymbol{U}_{i-j+1}^q\psi_{i,k+1} & \text{if } j \le i\\ \boldsymbol{U}_{m+1-(j-i)}^q\psi_{i,k+2} & \text{if } j > i\end{cases} \text{ and}$$
$$\bar{\chi}_{i,n}^{(\ell)} = \begin{cases}\boldsymbol{U}_{i-j+1}^q\chi_{i,k+1}^{(\ell)} & \text{if } j \le i\\ \boldsymbol{U}_{m+1-(j-i)}^q\chi_{i,k+2}^{(\ell)} & \text{if } j > i.\end{cases}$$

We next state a key lemma. We refer the reader to [8] for a detailed proof.

*Lemma 1:* Let $n = mk + j$. If $\tilde{x}_n = z_{j,k+1}$ and $\tilde{\psi}_{n+1} = [\,\bar{\psi}_{1,n+1} \;\; \cdots \;\; \bar{\psi}_{m,n+1}\,]^T$, $\tilde{\chi}^{(\ell)}_{n+1} = [\,\bar{\chi}^{(\ell)}_{1,n+1} \;\; \cdots \;\; \bar{\chi}^{(\ell)}_{m,n+1}\,]^T$ for $\ell = 1, \ldots, d$, then $\tilde{x}_n = z_{j+1,k+1}$, $\tilde{\psi}_{n+2} = [\,\bar{\psi}_{1,n+2} \;\; \cdots \;\; \bar{\psi}_{m,n+2}\,]^T$, and $\tilde{\chi}^{(\ell)}_{n+2} = [\,\bar{\chi}^{(\ell)}_{1,n+2} \;\; \cdots \;\; \bar{\chi}^{(\ell)}_{m,n+2}\,]^T$ for $\ell = 1, \ldots, d$.

We now prove Theorem 1 using the preceding lemma. Observe that the initial values for the RPE algorithm in (24) and the IRPE algorithm in (15) are chosen such that $\tilde{x}_0 = z_{0,1}$, $\tilde{\psi}_1 = [\,\bar{\psi}_{1,1} \;\; \cdots \;\; \bar{\psi}_{m,1}\,]^T$, and $\tilde{\chi}^{(\ell)}_1 = [\,\bar{\chi}^{(\ell)}_{1,1} \;\; \cdots \;\; \bar{\chi}^{(\ell)}_{m,1}\,]^T$ for all $\ell$. By using Lemma 1 and the induction on $k$, we can conclude that $\tilde{x}_{mk+i} = z_{i,k+1}$ for all $k \geq 1$ and $i \in \mathcal{I}$.

## REFERENCES


[1] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*. Cambridge, MA: The MIT press, 1983.

[2] M. G. Rabbat and R. D. Nowak, "Quantized incremental algorithms for distributed optimization," *IEEE J. Select Areas Commun.*, vol. 23, no. 4, pp. 798–808, Apr. 2005.

[3] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Trans. Signal Processing*, vol. 55, no. 8, pp. 4064–4077, Aug. 2007.

[4] S. Chen, C. Cowan, S. Billings, and P. Grant, "Parallel recursive prediction error algorithm for training layered neural networks," *Int. J. Control*, vol. 51, no. 6, pp. 1215–1228, 1990.

[5] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Incremental stochastic subgradient algorithms for convex optimization," *SIAM J. Optim.*, vol. 20, no. 2, pp. 691–717, 2009.

[6] A. Nedić and D. P. Bertsekas, "Incremental subgradient method for nondifferentiable optimization," *SIAM J. Optim.*, vol. 12, no. 1, pp. 109–138, 2001.

[7] P. Kumar and P. Varaiya, *Stochastic Systems: Estimation, Identification and Adaptive Control*. Englewood Cliffs, NJ: Prentice-Hall, 1986.

[8] S. R. Sundhar, V. V. Veeravalli, and A. Nedić, Distributed and Recursive Parameter Estimation in Parametrized Linear State-Space Models Tech. Rep., 2008 [Online]. Available: http://arxiv.org/abs/0804.1607


# A Distributed Actor-Critic Algorithm and Applications to Mobile Sensor Network Coordination Problems

Paris Pennesi and Ioannis Ch. Paschalidis

***Abstract*—We introduce and establish the convergence of a distributed actor-critic method that orchestrates the coordination of multiple agents solving a general class of a Markov decision problem. The method leverages the centralized single-agent actor-critic algorithm of [1] and uses a consensus-like algorithm for updating agents' policy parameters. As an application and to validate our approach we consider a reward collection problem as an instance of a multi-agent coordination problem in a partially known environment and subject to dynamical changes and communication constraints.**

***Index Terms*—Actor-critic methods, consensus, Markov decision processes (MDP), multi-agent coordination, sensor networks.**

## I. INTRODUCTION

We consider a setting where a *Markov Decision Problem (MDP)* problem is to be cooperatively solved by a group of agents that can simultaneously explore the state-control space. Each agent can communicate and exchange information with agents in its vicinity, thus, having the potential to modify its own policy on the basis of the information received.

The single-agent version of the problem can in principle be solved by stochastic *Dynamic programming (DP)*. To combat Bellman's curse of dimensionality, in this technical note we focus on an *Approximate Dynamic Programming (ADP)* approach: *actor-critic algorithms*[1]. In these algorithms one adopts a randomized class of policies parametrized by a (low-dimensional) parameter vector $\boldsymbol{\theta}$ and optimizes policy performance with respect to $\boldsymbol{\theta}$ by using a simulation (or a realization) of the MDP. According to its name, the algorithm interleaves two steps: (*i*) a policy improvement step at which it descends along the performance gradient with respect to $\boldsymbol{\theta}$ (the actor part), and (*ii*) a policy evaluation step at which it learns an approximate value function from a sample path that uses the current policy (the critic part).

Our main contribution is that we develop a *Distributed Multi-Agent Actor-Critic (D-AC)* algorithm. Our algorithm allows us to use multiple agents to simultaneously explore the state-control space. Each agent maintains its own $\boldsymbol{\theta}$ and updates it based on local information and information received from a subset of other agents (e.g., the ones within a certain communication range). This updating follows a consensus-like algorithm; such algorithms and their analysis go back to [2] and have garnered renewed interest [3], [4]. Under suitable conditions, we show that all agents reach consensus and converge to the optimal $\boldsymbol{\theta}$. In the algorithm we present, agents update their $\boldsymbol{\theta}$'s *asynchronously*.

The D-AC algorithm provides a useful framework for agent coordination in dynamic environments. What is particularly appealing is that we solve a dynamic problem benefiting from the parallel exploration

Manuscript received May 14, 2008; revised February 21, 2009. First published January 15, 2010; current version published February 10, 2010. This work was supported by the National Science Foundation (NSF) under Grants EFRI-0735974, DMI-0330171, ECS-0426453, and by the Department of Energy (DOE) under Grant DE-FG52-06NA27490. Recommended by Associate Editor Z. Qu.

P. Pennesi is with the RBS Global Banking & Markets, London EC2M 3UR, U.K. (e-mail: paris.pennesi@gmail.com; paris.pennesi@rbs.com).

I. Ch. Paschalidis is with the Department of Electrical and Computer Engineering, Division of Systems Engineering and the Center for Information & Systems Engineering, Boston University, Boston, MA 02215 USA (e-mail: yannisp@bu.edu).

Color versions of one or more of the figures in this technical note are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2009.2037462