[6] E. D. Engeberg and S. G. Meek, "Adaptive object slip prevention for prosthetic hands through proportional-derivative shear force feedback," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2008, pp. 1940–1945.
[7] E. D. Engeberg and S. G. Meek, "Adaptive sliding mode control for prosthetic hands to simultaneously prevent slip and minimize deformation of grasped objects," *IEEE/ASME Trans. Mechatronics*, vol. 18, no. 1, pp. 376–384, 2013.
[8] Z. Ding, N. Paperno, K. Prakash, and A. Behal, "An adaptive control-based approach for 1-click gripping of novel objects using a robotic manipulator," *IEEE Trans. Control Syst. Technol.*, 2018. doi: 10.1109/TCST.2018.2821651.
[9] S. Andersson, A. Söderberg, and S. Björklund, "Friction models for sliding dry, boundary and mixed lubricated contacts," *Tribology Int.*, vol. 40, no. 4, pp. 580–587, 2007.
[10] A. K. Padthe, B. Drincic, J. Oh, D. D. Rizos, S. D. Fassois, and D. S. Bernstein, "Duhem modeling of friction-induced hysteresis," *IEEE Control Syst. Mag.*, vol. 28, no. 5, pp. 90–107, 2008.
[11] K. Astrom and C. Canudas De Wit, "Revisiting the LuGre friction model," *IEEE Control Syst. Mag.*, vol. 28, no. 6, pp. 101–114, 2008.
[12] F. Al-Bender, V. Lampaert, and J. Swevers, "Modeling of dry sliding friction dynamics: From heuristic models to physically motivated models and back," *Chaos: Interdiscip. J. Nonlinear Sci.*, vol. 14, no. 2, pp. 446–460, 2004.
[13] Y. F. Liu, J. Li, Z. M. Zhang, X. H. Hu, and W. J. Zhang, "Experimental comparison of five friction models on the same test-bed of the micro stick-slip motion system," *Mech. Sci.*, vol. 6, pp. 15–28, Mar. 2015.
[14] A. Prach, J.-J. Cabibihan, N. V. Thakor, and D. S. Bernstein, "Pareto-front analysis of a monotonic PI control law for slip suppression in a robotic manipulator," in *Proc. Int. Conf. Robotics Biomimetics*, 2017, pp. 2728–2733.
[15] J. K. Hale, *Ordinary Differential Equations*. New York: Wiley, 1969.

Recursive Least Squares for Real-Time Implementation

SYED ASEEM UL ISLAM and DENNIS S. BERNSTEIN

Many estimation and control problems involve a process of the form

$$y_k = \phi_k \theta, \tag{1}$$

where $k = 0, 1, 2, \ldots$ is the discrete-time step corresponding to the continuous-time step size T_s, the scalar or vector $y_k \in \mathbb{R}^p$ is the measurement at step k, the matrix $\phi_k \in \mathbb{R}^{p \times n}$ is the regressor at step k whose entries consist of current and past data, and $\theta \in \mathbb{R}^n$ is a column vector of n unknown parameters. The objective is to use y_k and ϕ_k to estimate the components of θ. In applications, y_k and ϕ_k are corrupted by noise, and thus (1) does not hold exactly. This motivates the need for the least squares estimates of θ given below.

The measurements y_k and the data in ϕ_k are typically obtained from a continuous-time process and, as such, are available at the sample times kT_s, where T_s is the sample interval. The batch approach to this problem is to collect a large amount of data and then apply least squares optimization to the collected data to compute an estimate of θ. In particular, collecting data over the time window $i = 0, \ldots, k$, it follows from (1) that

$$Y = \Phi \theta, \tag{2}$$

where

$$Y \triangleq \begin{bmatrix} y_0 \\ \vdots \\ y_k \end{bmatrix}, \quad \Phi \triangleq \begin{bmatrix} \phi_0 \\ \vdots \\ \phi_k \end{bmatrix}. \tag{3}$$

Digital Object Identifier 10.1109/MCS.2019.2900788
Date of publication: 17 May 2019

Note that (2) has the form $Ax = b$, where A denotes Φ, x denotes θ, and b denotes Y.

In the presence of noise corrupting the data Y and Φ, (2) may not have a solution. In this case, it is useful to replace (2) by a least squares optimization problem of the form

$$\begin{aligned} J_k(\hat{\theta}) &\triangleq \sum_{i=0}^{k} (y_i - \phi_i \hat{\theta})^{\mathrm{T}} (y_i - \phi_i \hat{\theta}) + (\hat{\theta} - \theta_0)^{\mathrm{T}} R (\hat{\theta} - \theta_0) \\ &= (Y - \Phi \hat{\theta})^{\mathrm{T}} (Y - \Phi \hat{\theta}) + (\hat{\theta} - \theta_0)^{\mathrm{T}} R (\hat{\theta} - \theta_0), \end{aligned} \tag{4}$$

where R is a positive semidefinite (and thus, by definition, symmetric) matrix, and θ_0 is an initial estimate of θ. Assuming that R is chosen such that the inverse in (5) exists, the regularization term $(\hat{\theta} - \theta_0)^{\mathrm{T}} R (\hat{\theta} - \theta_0)$ weights the initial estimate and ensures that J_k has a unique global minimizer. In particular, the *batch least squares* (*BLS*) minimizer of (4) is given by

$$\theta_{\mathrm{opt},R} = (\Phi^{\mathrm{T}} \Phi + R)^{-1} (\Phi^{\mathrm{T}} Y + R \theta_0). \tag{5}$$

Note that the inverse required to compute (5) is of size $n \times n$, and thus the computational requirement of the inverse is of order n^3. In addition to the inverse, three matrix multiplications are needed. Note also that the memory needed to store Φ grows with k. Furthermore, if Φ has full column rank, then R can be set to zero, and thus (5) becomes

$$\theta_{\mathrm{opt},0} = (\Phi^{\mathrm{T}} \Phi)^{-1} \Phi^{\mathrm{T}} Y. \tag{6}$$

In the case where (2) has a solution and Φ has full column rank, (6) is the unique solution of (2).

In many applications, computational speed and memory are limited. One way to alleviate these requirements is

1066-033X/19©2019IEEE

to recursively update an estimate of $\theta_{\text{opt},R}$ using each additional measurement y_k. A recursive algorithm of this type is especially convenient for real-time applications.

Recursive least squares (*RLS*) is an iterative implementation of BLS that significantly reduces the computational and storage requirements of BLS. The purpose of this article is to provide a statement of RLS that highlights its real-time implementation along with a self-contained derivation (see "Summary").

Variations of RLS have been studied for more than half a century. An early exposition is given in [1], which emphasizes the real-time utility of RLS relative to BLS. Applications of RLS to adaptive control are discussed in [2, pp. 41, 103]. Numerous extensions of RLS have been developed to address initialization, forgetting, and numerical stability, for example, [3]–[6]. The development of RLS that is closest to the present article is given in [7, pp. 26–28].

Summary

Recursive least squares (RLS) is a technique used for minimizing a quadratic cost function, where the minimizer is updated at each step as new data become available. RLS is more computationally efficient than batch least squares, and it is extensively used for system identification and adaptive control. This article derives RLS and emphasizes its real-time implementation in terms of the availability of the data as well as the time needed for the computation.

RECURSIVE LEAST SQUARES

This section provides a statement and proof of the RLS algorithm. This result involves a recursive algorithm for optimizing J_k at each step k. The optimization of J_k updates the estimate θ_k of θ as measurements and data become available. As an extension of (4), the cost function (7) includes a forgetting factor λ, which provides higher weighting to more recent measurements and data.

Theorem 1

For all $k \geq 0$, let $\phi_k \in \mathbb{R}^{p\times n}$ and $y_k \in \mathbb{R}^p$. Furthermore, let $\theta_0 \in \mathbb{R}^n$, $P_0 \in \mathbb{R}^{n\times n}$ be positive definite, and $\lambda \in (0,1]$. For all $k \geq 0$, denote the minimizer of the function

$$J_k(\hat{\theta}) \triangleq \sum_{i=0}^{k} \lambda^{k-i}(y_i - \phi_i\hat{\theta})^{\mathrm{T}}(y_i - \phi_i\hat{\theta}) + \lambda^{k+1}(\hat{\theta} - \theta_0)^{\mathrm{T}}P_0^{-1}(\hat{\theta} - \theta_0) \tag{7}$$

by

$$\theta_{k+1} \triangleq \operatorname*{argmin}_{\hat{\theta}\in\mathbb{R}^n} J_k(\hat{\theta}). \tag{8}$$

Then, for all $k \geq 0$, θ_{k+1} is given by

$$P_{k+1} = \frac{1}{\lambda}P_k - \frac{1}{\lambda}P_k\phi_k^{\mathrm{T}}(\lambda I + \phi_k P_k \phi_k^{\mathrm{T}})^{-1}\phi_k P_k, \tag{9}$$

$$\theta_{k+1} = \theta_k + P_{k+1}\phi_k^{\mathrm{T}}(y_k - \phi_k\theta_k). \tag{10}$$

Proof

Note that $J_0(\hat{\theta})$ can be written as

$$J_0(\hat{\theta}) = \hat{\theta}^{\mathrm{T}}A_0\hat{\theta} + 2b_0^{\mathrm{T}}\hat{\theta} + c_0,$$

where

$$\begin{aligned} A_0 &\triangleq \phi_0^{\mathrm{T}}\phi_0 + \lambda P_0^{-1}, \\ b_0 &\triangleq -\phi_0^{\mathrm{T}}y_0 - \lambda P_0^{-1}\theta_0, \\ c_0 &\triangleq y_0^{\mathrm{T}}y_0 + \lambda\theta_0 P_0^{-1}\theta_0. \end{aligned}$$

Defining

$$P_1 \triangleq A_0^{-1},$$

it follows from Lemma 2 (see "Three Useful Lemmas") with $A = P_0^{-1}$, $C = (1/\lambda)I$, $U = \phi_0^{\mathrm{T}}$, and $V = \phi_0$ that

$$\begin{aligned} P_1 &= \frac{1}{\lambda}(P_0^{-1} + \frac{1}{\lambda}\phi_0^{\mathrm{T}}\phi_0)^{-1} \\ &= \frac{1}{\lambda}P_0 - \frac{1}{\lambda}P_0\phi_0^{\mathrm{T}}(\lambda I + \phi_0 P_0 \phi_0^{\mathrm{T}})^{-1}\phi_0 P_0. \end{aligned}$$

Hence, (9) is satisfied for $k = 0$. Additionally, because A_0 is positive definite, it follows from Lemma 1 (see "Three Useful Lemmas") that the unique minimizer θ_1 of J_0 is given by

$$\begin{aligned} \theta_1 &= -A_0^{-1}b_0 \\ &= P_1\phi_0^{\mathrm{T}}y_0 + \lambda P_1 P_0^{-1}\theta_0 \\ &= P_1\phi_0^{\mathrm{T}}y_0 + P_1(P_1^{-1} - \phi_0^{\mathrm{T}}\phi_0)\theta_0 \\ &= P_1\phi_0^{\mathrm{T}}y_0 + (I - P_1\phi_0^{\mathrm{T}}\phi_0)\theta_0 \\ &= \theta_0 + P_1\phi_0^{\mathrm{T}}(y_0 - \phi_0\theta_0). \end{aligned}$$

Hence, (10) is satisfied for $k = 0$.

Now, let $k \geq 1$. Then $J_k(\hat{\theta})$ can be written as

$$J_k(\hat{\theta}) = \hat{\theta}^{\mathrm{T}}A_k\hat{\theta} + 2b_k^{\mathrm{T}}\hat{\theta} + c_k,$$

where

$$\begin{aligned} A_k &\triangleq \sum_{i=0}^{k} \lambda^{k-i}\phi_i^{\mathrm{T}}\phi_i + \lambda^{k+1}P_0^{-1}, \\ b_k &\triangleq -\sum_{i=0}^{k} \lambda^{k-i}\phi_i^{\mathrm{T}}y_i - \lambda^{k+1}P_0^{-1}\theta_0, \\ c_k &\triangleq \sum_{i=0}^{k} \lambda^{k-i}y_i^{\mathrm{T}}y_i + \lambda^{k+1}\theta_0 P_0^{-1}\theta_0. \end{aligned}$$

Furthermore, A_k and b_k can be written recursively as

$$\begin{aligned} A_k &= \lambda A_{k-1} + \phi_k^{\mathrm{T}}\phi_k, \\ b_k &= \lambda b_{k-1} - \phi_k^{\mathrm{T}}y_k. \end{aligned}$$

Defining

$$P_{k+1} \triangleq A_k^{-1},$$

RLS is more computationally efficient than batch least squares, and it is extensively used for system identification and adaptive control.

it follows from Lemma 2 (see "Three Useful Lemmas") with $A = P_k^{-1}$, $C = (1/\lambda)I$, $U = \phi_k^{\mathrm{T}}$, and $V = \phi_k$ that

$$\begin{aligned} P_{k+1} &= [\lambda(A_{k-1} + \frac{1}{\lambda}\phi_k^{\mathrm{T}}\phi_k)]^{-1} \\ &= \frac{1}{\lambda}(P_k^{-1} + \frac{1}{\lambda}\phi_k^{\mathrm{T}}\phi_k)^{-1} \\ &= \frac{1}{\lambda}P_k - \frac{1}{\lambda}P_k\phi_k^{\mathrm{T}}(\lambda I + \phi_k P_k \phi_k^{\mathrm{T}})^{-1}\phi_k P_k. \end{aligned}$$

Hence, (9) is satisfied. Furthermore, the minimizer θ_{k+1} of J_k is given by

$$\theta_{k+1} = -A_k^{-1}b_k.$$

Because A_k is positive definite, it follows from Lemma 1 that

$$\begin{aligned} \theta_{k+1} &= -A_k^{-1}b_k \\ &= A_k^{-1}(\phi_k^{\mathrm{T}}y_k - \lambda b_{k-1}) \\ &= A_k^{-1}(\phi_k^{\mathrm{T}}y_k + \lambda A_{k-1}\theta_k) \\ &= A_k^{-1}[\phi_k^{\mathrm{T}}y_k + (A_k - \phi_k^{\mathrm{T}}\phi_k)\theta_k] \\ &= A_k^{-1}\phi_k^{\mathrm{T}}y_k + (I - A_k^{-1}\phi_k^{\mathrm{T}}\phi_k)\theta_k \\ &= \theta_k + P_{k+1}\phi_k^{\mathrm{T}}(y_k - \phi_k\theta_k). \end{aligned}$$

Hence, (10) is satisfied.

Note that, if $\lambda = 1$, R is positive definite, and $P_0 \triangleq R^{-1}$, then the RLS minimizer θ_{k+1} given by (8) is equal to the BLS minimizer $\theta_{\mathrm{opt},R}$ given by (5).

The notation θ_{k+1} for the minimizer of J_k emphasizes the fact that θ_{k+1}, which is based on data available up to step k, is not available until the update given by (9) and (10) is completed, which occurs at step $k+1$.

The derivation of RLS given by the proof of Theorem 1 is an extension of the RLS derivation given in [7]. In particular, the derivation given in [7, pp. 26–28] is based on the cost function (7) but without the regularization term involving P_0. As can be seen in the proof of Theorem 1, this term guarantees that A_0 is nonsingular in the first step and A_k is nonsingular in the inductive step. In the case of BLS, the role of P_0 is played by the matrix R.

The computational requirements of RLS are primarily determined by n. In particular, P_k is $n \times n$, and thus the computational requirement for updating P_k given by (9) is of order n^2. Furthermore, the inverse in (9) is of size $p \times p$, which, since typically $p \ll n$, is much less demanding than the inverse required by BLS. In addition, the storage requirements of RLS are of order n^2, which does not grow with k. Consequently, the computational and

Three Useful Lemmas

The following result is the quadratic minimization lemma.

LEMMA 1

Let $A \in \mathbb{R}^{n\times n}$, assume that A is positive definite, let $b \in \mathbb{R}^n$ and $c \in \mathbb{R}$, and define $f : \mathbb{R}^n \to \mathbb{R}$ by

$$f(x) \triangleq x^{\mathrm{T}}Ax + 2b^{\mathrm{T}}x + c. \tag{S1}$$

Then the unique minimizer of f is

$$x_{\mathrm{opt}} = -A^{-1}b, \tag{S2}$$

and the minimum value of f is

$$f(x_{\mathrm{opt}}) = c - b^{\mathrm{T}}A^{-1}b. \tag{S3}$$

The following result is the matrix inversion lemma [S1, p. 304].

LEMMA 2

Let $A \in \mathbb{R}^{n\times n}$, $U \in \mathbb{R}^{n\times p}$, $C \in \mathbb{R}^{p\times p}$, and $V \in \mathbb{R}^{p\times n}$, and assume that A, C, and $A + UCV$ are nonsingular. Then

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}. \tag{S4}$$

LEMMA 3

Let $A \in \mathbb{R}^{n\times n}$ be positive semidefinite, let $B \in \mathbb{R}^{n\times m}$, and let $C \in \mathbb{R}^{n\times n}$ be positive definite. Then,

$$[I - AB^{\mathrm{T}}(C + BAB^{\mathrm{T}})^{-1}B]AB^{\mathrm{T}} = AB^{\mathrm{T}}(C + BAB^{\mathrm{T}})^{-1}C. \tag{S5}$$

PROOF

Note that

$$\begin{aligned} AB^{\mathrm{T}}(C + BAB^{\mathrm{T}})^{-1}C &= AB^{\mathrm{T}}(C + BAB^{\mathrm{T}})^{-1}(C + BAB^{\mathrm{T}} - BAB^{\mathrm{T}}) \\ &= AB^{\mathrm{T}}[I - (C + BAB^{\mathrm{T}})^{-1}BAB^{\mathrm{T}}] \\ &= AB^{\mathrm{T}} - AB^{\mathrm{T}}(C + BAB^{\mathrm{T}})^{-1}BAB^{\mathrm{T}} \\ &= [I - AB^{\mathrm{T}}(C + BAB^{\mathrm{T}})^{-1}B]AB^{\mathrm{T}}. \end{aligned}$$

REFERENCE

[S1] D. S. Bernstein, *Scalar, Vector, and Matrix Mathematics: Theory, Facts, and Formulas*. Princeton, NJ: Princeton Univ. Press, 2018.

memory requirements of RLS are significantly less than those of BLS.

ALTERNATIVE θ_k UPDATE WITH INVERSE

The following result is a variation of Theorem 1. In this formulation, the updates of P_k and θ_k are reversed.

Theorem 2

For all $k \geq 0$, let $\phi_k \in \mathbb{R}^{p\times n}$ and $y_k \in \mathbb{R}^p$. Furthermore, let $\theta_0 \in \mathbb{R}^n$, let $P_0 \in \mathbb{R}^{n\times n}$ be positive definite, and let $\lambda \in (0,1]$. For all $k \geq 0$, denote the minimizer of the function (7) by (8). Then, for all $k \geq 0$, θ_{k+1} is given by

$$\theta_{k+1} = \theta_k + P_k\phi_k^{\mathrm{T}}(\lambda I + \phi_k P_k \phi_k^{\mathrm{T}})^{-1}(y_k - \phi_k\theta_k), \tag{11}$$

$$P_{k+1} = \frac{1}{\lambda}P_k - \frac{1}{\lambda}P_k\phi_k^{\mathrm{T}}(\lambda I + \phi_k P_k \phi_k^{\mathrm{T}})^{-1}\phi_k P_k. \tag{12}$$

Proof

Using (9) to substitute P_{k+1} into (10) yields

$$\begin{aligned}\theta_{k+1} &= \theta_k + \Big[\frac{1}{\lambda}P_k - \frac{1}{\lambda}P_k\phi_k^{\mathrm{T}}(\lambda I + \phi_k P_k \phi_k^{\mathrm{T}})^{-1}\phi_k P_k\Big]\phi_k^{\mathrm{T}}(y_k - \phi_k\theta_k)\\ &= \theta_k + \frac{1}{\lambda}[I - P_k\phi_k^{\mathrm{T}}(\lambda I + \phi_k P_k \phi_k^{\mathrm{T}})^{-1}\phi_k]P_k\phi_k^{\mathrm{T}}(y_k - \phi_k\theta_k)\\ &= \theta_k + P_k\phi_k^{\mathrm{T}}(\lambda I + \phi_k P_k \phi_k^{\mathrm{T}})^{-1}(y_k - \phi_k\theta_k),\end{aligned}$$

where the last equality follows from Lemma 3 (see "Three Useful Lemmas"). Hence, (11) holds. Finally, (12) is identical to (9).

REAL-TIME IMPLEMENTATION OF RECURSIVE LEAST SQUARES

In many applications, it is desirable to implement RLS so that the estimate θ_k is available in real time without latency. Note that the estimate θ_{k+1} given by (10) depends on measurements available up to and including step k, namely, y_k and ϕ_k. Since time is needed to compute θ_{k+1}, the updated estimate θ_{k+1} is not available at time k; rather, it is available at the next step, namely, $k+1$. Consequently, the minimizer of J_k is denoted by θ_{k+1}, where the subscript $k+1$ conveys the fact that the minimizer of J_k is not available until step $k+1$. In contrast, the notation used in [7, p. 27] is θ_k. Figure 1 shows how the measurements and data that are available at step k are used during the time interval $[kT_s,(k+1)T_s]$ to compute the next estimate θ_{k+1}.

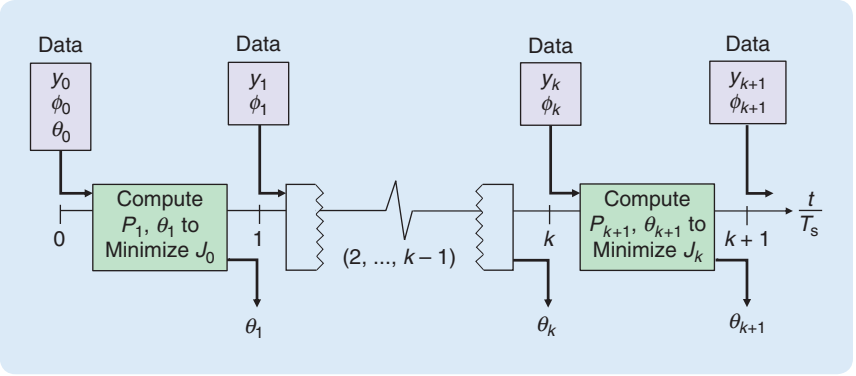

FIGURE 1 A real-time implementation of recursive least squares. Data at step k, which corresponds to time $t = kT_s$, are used to compute the minimizer θ_{k+1} of the cost J_k. Because of the time needed for the computation, the estimate θ_{k+1} of the unknown parameter θ is not available until step $k+1$.

ACKNOWLEDGMENTS

The authors thank Ankit Goel, Jonathan How, and Tam Nguyen for helpful suggestions. This research was partially supported by the Air Force Office of Scientific Research under the Dynamic Data-Driven Applications Systems grant FA9550-16-1-0071 (http://www.1ddas.org/).

AUTHOR INFORMATION

Syed Aseem Ul Islam (aseemisl@umich.edu) received the B.Sc. degree in aerospace engineering from the Institute of Space Technology, Islamabad, Pakistan, and is currently a Ph.D. student in flight dynamics and control at the University of Michigan, Ann Arbor. His research interests include data-driven adaptive control for aerospace applications.

Dennis S. Bernstein received the Sc.B. degree in applied mathematics from Brown University, Providence, Rhode Island, and the Ph.D. degree in control engineering from the University of Michigan, Ann Arbor, where he is currently a professor in the Aerospace Engineering Department. He is the author of *Scalar, Vector, and Matrix Mathematics* (Princeton University Press, 2018). His research interests include estimation and control for aerospace applications.

REFERENCES

[1] A. Albert and R. W. Sittler, "A method for computing least squares estimators that keep up with the data," *SIAM J. Control*, vol. 3, no. 3, pp. 384–417, 1965.

[2] K. Astrom and B. Wittenmark, *Adaptive Control*, 2nd ed. New York: Dover, 2013.

[3] M. E. Salgado, G. C. Goodwin, and R. H. Middleton, "Modified least squares algorithm incorporating exponential resetting and forgetting," *Int. J. Control*, vol. 47, no. 2, pp. 477–491, 1988.

[4] S. Dasgupta and Y.-F. Huang, "Asymptotically convergent modified recursive least-squares with data-dependent updating and forgetting factor for systems with bounded noise," *IEEE Trans. Inf. Theory*, vol. 33, no. 3, pp. 383–392, 1987.

[5] P. Stoica and P. Ahgren, "Exact initialization of the recursive least-squares algorithm," *Int. J. Adapt. Control Signal Process.*, vol. 16, pp. 219–230, Jan. 2002.

[6] L. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification*. Cambridge, MA: MIT Press, 1985.

[7] C. Johnson, *Lectures on Adaptive Parameter Estimation*. Englewood Cliffs, NJ: Prentice Hall, 1988.