An Algorithm for RLS Identification of Parameters that Vary Quickly with Time

James E. Bobrow and Walter Murray

Abstract—**An alternative to the standard recursive least-squares algorithm for fixed-order systems with exponential data weighting is presented. The approach uses Givens orthogonal transformations to update the Cholesky factor of the information matrix without ever needing to form it. The resulting algorithm gives a higher precision solution and is less sensitive to ill-conditioning when compared to other reported approaches. It is demonstrated by an example that ill-conditioned problems with parameters that vary quickly can be modified to stabilize erratic parameter fluctuations.**

I. Introduction

This work was motivated by the need in some robotics applications for on-line identification of parameters that vary quickly with time. In many instances, the identification problem is extremely ill-conditioned. Consequently, when designing algorithms for such problems, it is essential to exercise care, otherwise there may be no precision in a computed solution.

The problem of interest may be couched as that of finding the solution of a sequence of least-squares problems. Namely

$$\underset{\theta_k}{\text{minimize}} \|D_k A_k \theta_k - D_k y_k\| \tag{1.1}$$

for $k = 1, 2, \cdots$, where $A_{k+1}^T = (A_k^T, \theta_{k+1})$, $D_k = \text{diag}\ (\lambda^{k-1}, \lambda^{k-2}, \cdots, 1)$, and $0 < \lambda < 1$. The scalar λ is known as the "forgetting factor," and is used to place less weight on past data.

The use of orthogonal transformations for solving least-squares problems is well established, as is the inadvisability of utilizing the normal equations [4], [3]. However, in the class of problems considered here, the use of normal equations is commonplace. This may be due, in part, to the fact that for large k, the orthogonal factor, which is a $k \times k$ matrix, would be too expensive to recur. Recently, orthogonal transformations have been used for solving variable-order lattice problems [6].

Typically, identification algorithms take new data available at each time step and update the error covariance matrix P (P^{-1} is termed the information matrix). They then solve for the corresponding change in parameter estimates based on this new data. The most straightforward update for P is equivalent to using the matrix-inversion lemma to incorporate new data into P at each iteration. The recursive equations obtained are also the conventional Kalman filter update [7]. A more numerically stable method is obtained from a U^TDU factorization of P, where U is an upper triangular matrix and D is a diagonal matrix. A rank-one change is made to this factorization, when new data is available [2].

In this note, we describe an approach that does not use the P matrix. The numerical properties of the algorithm are similar to solving (1.1) directly by a QR factorization. We show how R may be updated using orthogonal matrices without the need to store Q. The number of operations required is only slightly more than alternative approaches.

Manuscript received April 12, 1991. This work was supported in part by the National Science Foundation under Grant ECS-8715153 and by the Office of Naval Research under Contract N00014-87-K-0142.

J. E. Bobrow is with the Department of Mechanical and Aerospace Engineering, University of California, Irvine, CA 92717.

W. Murray is with the Department of Operations Research, Systems Optimization Laboratory, Stanford University, Stanford, CA 94305-4022.

IEEE Log Number 9203114.

II. Review of Batch Least-Squares Problems

Consider a set of equations of the form $y_i = \phi_i^T\theta + r_i$, $i = 1, 2, \cdots, k$, where y_i is a scalar output, $\phi_i \in \Re^n$, r_i is a residual error, and $\theta \in \Re^n$. We seek the vector θ^* that solves

$$\underset{\theta \in \Re^n}{\text{minimize}} \|A\theta - y\|_2^2 \tag{2.1}$$

where $A^T = (\phi_1, \phi_2, \cdots, \phi_n)$. We assume that $k \geq n$. If A has full-column rank, then the unique minimizer θ^* of (2.1) is defined by the well-known *normal equations*

$$A^TA\theta^* = A^Ty. \tag{2.2}$$

An obvious way to solve (2.2) is by computing the Cholesky factorization of A^TA. However, the condition number of A^TA is the *square* of the condition number of A. Hence, the computed solution of (2.2) may be severely degraded even if A is only moderately ill-conditioned.

Any recursive identification method that updates the error covariance matrix $(A^TA)^{-1} \equiv P$ will suffer from the above-mentioned malady. This includes the standard recursive least-squares algorithm [5] or the more numerically stable update based on a U^TDU factorization of P [2]. The following approach retains the conditioning of the original problem [4].

Suppose R is known such that

$$Q^TA = \begin{pmatrix} R \\ 0 \end{pmatrix} \tag{2.3}$$

where Q is an orthogonal $k \times k$ matrix; and R is an upper triangular $n \times n$ matrix. The term *square-root filter* has also been used for R in estimation literature [1].

Since Euclidean length is preserved by an orthogonal transformation, (2.1) is equivalent to

$$\underset{\theta \in \Re^n}{\text{minimize}} \|Q^TA\theta - Q^Ty\|^2 \tag{2.4}$$

or

$$\underset{\theta \in \Re^n}{\text{minimize}} \|R\theta - \bar{y}\|^2 + \|\tilde{y}\|^2 \tag{2.5}$$

where $Q^Ty = \begin{pmatrix} \bar{y} \\ \tilde{y} \end{pmatrix}$. If $\bar{y}$ is also known, the minimizer of (2.1) may be found by solving $R\theta^* = \bar{y}$, which requires $n(n+1)/2$ flops. The value of the minimum residual is $\|\tilde{y}\|^2$.

For the batch least-squares problem, the matrix R is found from a sequence of Householder transformations applied to A [4]. We may apply these transformation to y simultaneously, hence, Q need not be stored explicitly. It will be seen that the same is true for sequential least-squares parameter identification. The required triangular factor is found recursively using a sweep of Givens rotations, as described in the next section.

III. Sequential Identification

It follows from (1.1) that at the $(k+1)$ stage, we wish to solve the following problem:

$$\underset{\theta \in \Re^n}{\text{minimize}} \left\| \begin{pmatrix} \lambda A_k \\ \phi_{k+1}^T \end{pmatrix} \theta - \begin{pmatrix} \lambda y \\ y_{k+1} \end{pmatrix} \right\|^2. \tag{3.1}$$

At first sight, this does not look promising, since A_{k+1} differs from A_k by a rank n matrix. However, it will be seen from the following lemma that R_k may be recurred quite simply.

Lemma 3.1: If $A_k = Q\begin{pmatrix} R_k \\ 0 \end{pmatrix}$ and $Q^T y = \begin{pmatrix} \bar{y}_k \\ \tilde{y} \end{pmatrix}$, where Q is an orthogonal matrix and R_k is an upper triangular matrix, then the minimizer of (3.1) is also the minimizer of

$$\underset{\theta \in \Re^n}{\text{minimize}} \left\| \begin{pmatrix} \lambda R_k \\ \phi_{k+1}^T \end{pmatrix} \theta - \begin{pmatrix} \lambda \bar{y} \\ y_{k+1} \end{pmatrix} \right\|^2. \tag{3.2}$$

Proof: Form the normal equations for both systems and compare them. Let $\bar{A} = \begin{pmatrix} \lambda A_k \\ \phi_{k+1}^T \end{pmatrix}$, then

$$\bar{A}^T \bar{A} = \lambda^2 A_k^T A_k + \phi_{k+1}\phi_{k+1}^T = \lambda^2 R_k^T R_k + \phi_{k+1}\phi_{k+1}^T$$

and

$$\begin{aligned} \bar{A}^T \begin{pmatrix} \lambda y \\ y_{k+1} \end{pmatrix} &= \lambda^2 A_k^T y + \phi_{k+1} y_{k+1} \\ &= \lambda^2 (R_k^T 0) Q^T y + \phi_{k+1} y_{k+1} \\ &= \lambda^2 R_k^T \bar{y}_k + \phi_{k+1} y_{k+1}. \end{aligned}$$

■

To obtain the minimizer of (3.2), which we refer to as θ_{k+1}, one only needs to update the factor $\begin{pmatrix} \lambda R_k \\ \phi_{k+1}^T \end{pmatrix}$ to form R_{k+1} and solve the triangular system $R_{k+1}\theta_{k+1} = \bar{y}_{k+1}$. However, because it is preferable to solve for the *change* in θ_k rather than θ_k itself, we write

$$\theta_{k+1} = \theta_k + \delta_k \tag{3.3}$$

and solve (3.2) for δ_k.

Suppose an orthogonal matrix $\tilde{Q}$ is known such that

$$\tilde{Q}\begin{pmatrix} \lambda R_k \\ \phi_{k+1}^T \end{pmatrix} = \begin{pmatrix} R_{k+1} \\ 0 \end{pmatrix}. \tag{3.4}$$

From (3.2)–(3.4), it follows that δ_k is the minimizer of

$$\underset{\delta}{\text{minimize}} \left\| \tilde{Q}\begin{pmatrix} \lambda R_k \\ \phi_{k+1}^T \end{pmatrix} \delta - \tilde{Q}\left\{ \begin{pmatrix} \lambda \bar{y}_k \\ y_{k+1} \end{pmatrix} - \begin{pmatrix} \lambda R_k \\ \phi_{k+1}^T \end{pmatrix} \theta_k \right\} \right\| \tag{3.5}$$

which reduces to

$$\underset{\delta}{\text{minimize}} \left\| \begin{pmatrix} R_{k+1} \\ 0 \end{pmatrix} \delta - \tilde{Q}\begin{pmatrix} 0 \\ u_{k+1} \end{pmatrix} \right\| \tag{3.6}$$

where $u_{k+1} = y_{k+1} - \phi_{k+1}^T \theta_k$. Hence, δ_k may be found by solving the triangular system

$$R_{k+1}\delta_k = \bar{y}_{k+1} \tag{3.7}$$

where $\bar{y}_{k+1}$ satisfies

$$\begin{pmatrix} \bar{y}_{k+1} \\ r_{k+1} \end{pmatrix} = \tilde{Q}\begin{pmatrix} 0 \\ u_{k+1} \end{pmatrix}. \tag{3.8}$$

To find the $\tilde{Q}$ that satisfies (3.4), we use a sweep of Givens plane rotations to annihilate the ϕ_{k+1}^T term in (3.4). Because $\tilde{Q}$ must also premultiply the right-hand side of (3.8), we perform the required sweep on the augmented matrix

$$\begin{pmatrix} \lambda R_k & 0 \\ \phi_{k+1}^T & u_{k+1} \end{pmatrix}. \tag{3.9}$$

IV. A MATLAB Program

A listing of a MATLAB program that computes one update is given in the Appendix. It may be seen by examining the listing of routine "sweep," that the multiplications of R_k by λ can be avoided by modifying the Givens transformations. This modification requires only $3n$ multiplications rather than the $n(n+1)/2$ required otherwise. The number of operations for one sweep of plane rotations is $2n^2 + 7n$ multiplications, $2n$ divisions, and n square roots.

The steps needed for each recursive update are as follows.

1) Compute the prediction error $y_{k+1} - \phi_{k+1}^T \theta_k$; routine "lspr" in the Appendix.
2) Form the $(n+1) \times (n+1)$ matrix in (3.9); routine "newqr" in the Appendix.
3) Sweep the bottom n left-hand elements of the above matrix to zero using a sequence of Givens rotations; routines "sweep" and "givens" in the Appendix.
4) Solve the upper-diagonal system (3.7) for δ_k; routine "solve" in the Appendix.

The number of multiplications for the standard Kalman update (taking into account symmetry) is $1.5n^2 + 4.5n$ [2]. The total for our algorithm $2.5n^2 + 7.5n$ multiplications, $2n$ divisions, and n square roots. The Givens computations can be reduced by approximately 50% with the divisions, and square roots eliminated using "fast Givens" transformations [4]. However, we did not implement this method because: 1) the potential exists for numerical instability; and 2) a moderate amount of extra logic is required.

V. Two Examples

As a simple example of the benefits of the improved precision obtained with this algorithm, consider the system

$$y_k = \sin(0.1k) u_k \tag{5.1}$$

where we assume the unknown time-varying parameter is $\theta_k = \sin(0.1k)$. We do not assume that an analytic description of the parameter variation with time is known. Therefore, we are forced to identify a nonstationary quantity. If the input is $u_k = \sin(k)$ for $k = 1, \cdots, 100$ and $\lambda = 0.9$, the plot shown in Fig. 1 results. The solid plot is the desired parameter estimate $\sin(0.1k)$, the dashed plot is the estimate from our algorithm, and the dotted line is the estimate from a *UD* update. It should be noted that both algorithms could be made to track more closely by decreasing λ.

To demonstrate what happens as the A matrix begins to lose rank when an input signal is barely exciting, consider the system

$$y_k = u_k + u_{k-1} + \sin(0.1k) u_{k-2}$$

where we assume the unknown time-varying parameter is $\theta_k = [1, 1, \sin(0.1k)]^T$. If the input is

$$u_k = \sin(0.2k) + \alpha \sin(k)$$

then as α decreases, the rank of the A matrix will shift from 3 to 2. The top plot in Fig. 2 shows the third component of θ_k when $\alpha = 1$, and the bottom plot is for $\alpha = 0.01$. In both cases $\lambda = 0.5$. Larger values of λ caused a considerable lag in the parameter estimates, and did not decrease the fluctuations in the estimates until λ was very close to 1. Clearly, there is a need to smooth these estimates. Our approach is described in the next section.

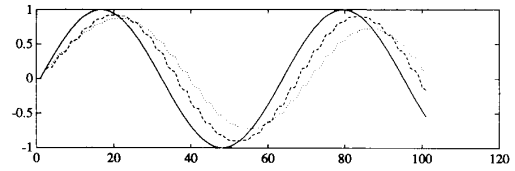

Fig. 1. Estimating a sin wave ($\lambda = 0.9$).

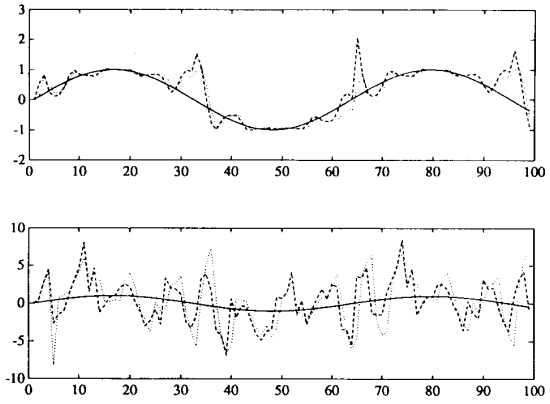

Fig. 2. The third component of θ_k for $\lambda = 0.5$. The top plot is due to a richer input ($\alpha = 1$) than the bottom ($\alpha = 0.01$).

VI. Internal Filtering of the Estimates

As the forgetting factor λ is decreased for faster parameter tracking, and as the number of parameters is increased, the matrix R_k becomes more ill-conditioned. It may be observed from the bottom plot of Fig. 2 that this may cause substantial fluctuations in the parameter estimates. It is probably why many adaptive control algorithms use large values for λ (usually $\lambda \geq 0.95$). Unfortunately, this yields a system with very poor estimates when the parameters are changing rapidly.

One way to decrease parameter fluctuations, without sacrificing tracking, is by adding to the cost function the desire to minimize parameter fluctuation at each instant. For example, one might want the parameter estimates to decay according to

$$E(q)\theta_k = \left(q^p + a_{p-1}q^{p-1} + a_{p-2}q^{p-2} + \cdots + a_0\right)\theta_k = 0 \tag{6.1}$$

where $E(q)$ is a stable polynomial in the forward shift operator q ($qx_k = x_{k+1}$). A cost associated with this can be added to the least-squares update by solving (6.1) for the desired change in parameter estimate in terms of the previous estimates

$$\delta_k^f = -[(1 + a_{p-1})\theta_{k-1} + a_{p-2}\theta_{k-2} + \cdots + a_0\theta_{k-p}] \equiv g(\theta_k) \tag{6.2}$$

where $g(\theta_k)$ is updated at each k. With the desired value δ_k^f known, the augmented cost function is

$$\left\| \begin{pmatrix} R_{k+1} \\ wI \end{pmatrix} \delta - \begin{pmatrix} \bar{y}_{k+1} \\ wg(\theta_k) \end{pmatrix} \right\|^2 \tag{6.3}$$

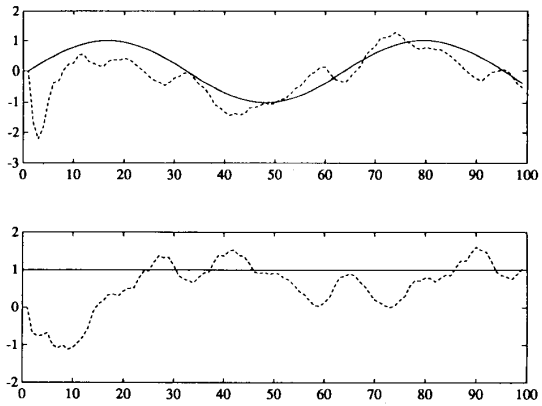

Fig. 3. The same system and input as the bottom plot in Fig. 2. The top plot is the third component of θ_k, and the bottom is the first component.

where w is a small weighting factor. Note that R_{k+1} is still the triangular factor corresponding to A_{k+1} [it is not the factor corresponding to the matrix $\begin{pmatrix} A_k \\ wI \end{pmatrix}$]. In order to solve (6.3), we need to reduce $\begin{pmatrix} R_{k+1} \\ wI \end{pmatrix}$ to upper-triangular form. Since n rows have been added to the cost function, the update would require n sweeps. In general, this is far too expensive for real-time applications, except when n is very small. An alternative is to add one row of the n-desired filtered values of δ_k^f at each iteration. The row added is being rotated with k. We now recur the triangular factor of the *augmented* matrix. In theory, we should remove the $(k + 1 - n)$th augmented row at the $k + 1$ iteration. However, by then, the forgetting factor has reduced its contribution by λ^n. In this scheme, we recur the triangular factor of the augmented matrix and not that of A_k.

The method for smoothing the estimates described above was applied to the system described in the second example for the ill-conditioned case $\alpha = 0.01$. The filter $E(q)$ was set to $(q - 0.5)^3$ and the weighting $w = 0.1$. These quantities resulted in the estimates shown in Fig. 3. The top plot is the third component of θ_k, and the bottom is the first component. Comparing the scale of the bottom plot of Fig. 2 to the scale in Fig. 3, we see that the parameter estimate is much closer to its desired value.

VII. Initialization

For the first $n - 1$ iterations, the least-squares problem is underdetermined and, hence, does not have a unique solution. We initialize $R_0 = \epsilon I$, where ϵ is suitably small. The effect of the initial choice is to make the computed solution very close to the solution of least length. The modifications to regularize the problem are only commenced after n iterations.

VIII. Appendix

```
function X = lspr(A, y, lam)
% Sequential least-squares parameter estimation for Ax = y.
% Input
%       y - Column vector of output data.
%       A - m × n matrix.
% Output
%       X - parameter estimate vector
```

```
[m, n] = size(A); x = zeros(n, 1); dx = zeros(n, 1);
for k = 1:m
    phi = A(k, :)';
    e = y(k) - phi'*x;
    [R, yb] = newqr(R, phi, e, lam);
    dx = solve(R, yb);
    x = x + dx;
    X(k, :) = x';
end
```

```
function [R, yb] = newqr(R, phi, e, lam)
[m, n] = size(R);
BigR = [R zeros(n, 1); phi' e];
for i = 1:n
    BigR = sweep(BigR, i, lam);
end
R = BigR(1:n, 1:n);
yb = BigR(1:n, n + 1);
```

```
function A = sweep(A, i, lam)
%  Input
%        A - n × n matrix
%        i - The (i, n) rows of the A matrix are swept, with the
%            element A(n, i) being annihilated.
%      lam - The forgetting factor.
%  Output
%        A - swept matrix.
[m, n] = size(A);
[c, s, r] = givens(A(i, i)*lam, A(n, i));
A(i, i) = r;
A(n, i) = 0.0;
clam = c*lam;
slam = s*lam;
for k = i + 1:n
    a = A(i, k)*clam + A(n, k)*s;
    A(n, k) = -A(i, k)*slam + A(n, k)*c;
    A(i, k) = a;
end
```

```
function [c, s, r] = givens(a, b)
%  Input
%        a, b - Elements of a vector for the which b component
%               is to be annihilated by a plane rotation.
%  Output
%        c, s - The required transformation.
%           r - The length of the vector.
r = sqrt(a^2 + b^2);
if r < 1.0e-8,
  c = 1; s = 0; r = 1.0e-8;
else
  c = a/r;
  s = b/r;
end
```

```
function x = solve(R, b)
% Solves Rx = b for x assuming R is upper right triangular.
%  Input
%        b - Column vector of output data.
%        R - right triangular matrix.
%  Output
%        x - solution.
[m, n] = size(R);
for k = n :- 1:1,
    sum = 0.0;
    for j = k + 1:n
        sum = sum + R(k, j)*x(j, 1);
    end
    x(k, 1) = (b(k) - sum)/R(k, k);
end
```


REFERENCES

[1] A. Andrews, "A square root formulation of the Kalman covariance equations," *AIAA J.*, vol. 6, no. 6, 1968.

[2] G. J. Bierman, *Factorization Methods for Discrete Sequential Estimation.* New York: Academic, 1977.

[3] P. E. Gill, W. Murray, and M. H. Wright, *Numerical Linear Algebra and Optimization*, vol. 1. Redwood City, CA: Addison-Wesley, 1991.

[4] G. H. Golub and C. F. Van Loan, *Matrix Computations.* Baltimore, MD: The Johns Hopkins University Press, 1990.

[5] G. C. Goodwin and K. S. Sin, *Adaptive Filtering Prediction and Control.* Englewood Cliffs, NJ: Prentice-Hall, 1984.

[6] P. S. Lewis, "QR-based algorithms for multichannel adaptive least squares Lattice filters," *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. 38, no. 3, pp. 421–431, 1990.

[7] L. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification.* Cambridge, MA: M.I.T. Press, 1983.

On Constrained Optimization of the Klimov Network and Related Markov Decision Processes

Armand M. Makowski and Adam Shwartz

***Abstract*—We solve a constrained version of the server allocation problem for the Klimov network and establish that the optimal constrained schedule is obtained by randomizing between two fixed priority schemes. This generalizes the work of Nain and Ross in the context of the competing queue problem and also covers the discounted cost case. In order to establish these results, we develop a general framework for optimization under a single constraint in the presence of index-like policies. This methodology is in principle of wider applicability.**

I. INTRODUCTION

Consider the discrete-time system of K competing queues with a single Bernoulli server as described in [5] and [8]. For one-step costs which are linear in the queue sizes, it is well known [4], [5], [8] that there exists an optimal policy which is of the strict priority type, and this, under several cost criteria including the discounted and average cost criteria in which case the search for optimal policies reduces to the computation of a few parameters. Let $J_c(\pi)$ and $J_d(\pi)$ be two cost functions associated with the one-step cost functions c and d, when the system is operated under the policy π. A single constraint optimization problem can then be defined as follows:

$(\boldsymbol{P}_v)$: Minimize $J_c(\pi)$ subject to the constraint $J_d(\pi) \leq V$

Manuscript received April 12, 1992; revised February 21, 1992. This work was supported in part by the National Science Foundation under Grants ECS-83-51836 and CDR-88-03012, and partially through United States—Israel Binational Science Foundation under Grant BSF-85-00306.

A. M. Makowski is with the Department of Electrical Engineering and the Systems Research Center, University of Maryland, College Park, MD 20742.

A. Shwartz is with the Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa 32000, Israel.

IEEE Log Number 9203115.