Sliding Window Recursive Quadratic Optimization with Variable Regularization

Jesse B. Hoagg, Asad A. Ali, Magnus Mossberg, and Dennis S. Bernstein

Abstract—In this paper, we present a sliding-window variable-regularization recursive least squares algorithm. In contrast to standard recursive least squares, the algorithm presented in this paper operates on a finite window of data, where old data are discarded as new data become available. This property can be beneficial for estimating time-varying parameters. Furthermore, standard recursive least squares uses time-invariant regularization. More specifically, the inverse of the initial covariance matrix in standard recursive least squares can be viewed as a regularization term, which weights the difference between the next estimate and the initial estimate. This regularization is fixed for all steps of the recursion. The algorithm derived in this paper allows for time-varying regularization. In particular, the present paper allows for time-varying regularization in the weighting as well as what is being weighted. Specifically, the regularization term can weight the difference between the next estimate and a time-varying vector of parameters rather than the initial estimate.

I. INTRODUCTION

Within signal processing, identification, estimation, and control, recursive least squares (RLS) and gradient-based optimization techniques are among the most fundamental and widely used algorithms [1]–[8]. The standard RLS algorithm operates on a growing window of data, that is, new data are added to the RLS cost function as they become available and old data are not directly discarded but rather progressively discounted through the use of a forgetting factor. In contrast, a sliding-window RLS algorithm operates on a finite window of data with fixed length; new data replace old data in the sliding-window RLS cost function. Sliding-window least-squares techniques are available in both batch and recursive formulations [9]–[13].

A sliding-window RLS algorithm with time-varying regularization is developed in the present paper. A growing-window RLS algorithm with time-varying regularization is presented in [14]. In standard RLS, the positive-definite initialization of the covariance matrix can be interpreted as the weighting on a regularization term within the context of a quadratic optimization. Until at least \( n \) measurements are available, this regularization term compensates for the lack of persistency in order to obtain a unique solution from the RLS algorithm. Traditionally, the regularization term is fixed for all steps of the recursion. In the present work, we derive a sliding-window variable-regularization RLS (SW-VR-RLS) algorithm, where the weighting in the regularization term may change at each step. As a special case, the regularization can be decreased in magnitude or rank as the rank of the covariance matrix increases, and can be removed entirely when no longer needed. This ability is not available in standard RLS where the regularization term is weighted by the inverse of the initial covariance.

A second extension presented in this paper also involves the regularization term. Specifically, the regularization term in traditional RLS weights the difference between the next estimate and the initial estimate. In the present paper, the regularization term weights the difference between the next estimate and an arbitrarily chosen time-varying vector. As a special case, the time-varying vector can be the current estimate, and thus the regularization term weights the difference between the next estimate and the current estimate. This formulation allows us to modulate the rate at which the current estimate changes from step to step. For these extensions, we derive SW-VR-RLS update equations.

In the next section, we derive the update equations for SW-VR-RLS. In the remaining sections of the paper, we investigate the performance of SW-VR-RLS under various conditions of noise and persistency.

II. SW-VR-RLS ALGORITHM

For all \( i \geq 0 \), let \( A_i \in \mathbb{R}^{n \times n} \), \( b_i, \alpha_i \in \mathbb{R}^n \), and \( R_i \in \mathbb{R}^{n \times n} \); where \( A_i \) and \( R_i \) are positive semidefinite, define \( A_0 = 0 \), \( b_0 = 0 \), let \( r \) be a nonnegative integer, and assume that, for all \( k \geq r \), \( \sum_{i=k-r}^{k} A_i + R_k \) and \( \sum_{i=k-r+1}^{k} A_i + R_{k+1} \) are positive definite. Define the sliding-window regularized quadratic cost

\[
J_k(x) = \sum_{i=k-r}^{k} (x^T A_i x + b_i^T x) + (x - \alpha_k)^T R_k (x - \alpha_k),
\]

where \( x \in \mathbb{R}^n \) and \( x_0 = \alpha_0 \) is the minimizer of \( J_0(x) \). The minimizer \( x_k \) of (1) is given by

\[
x_k = -\frac{1}{2} \left( \sum_{i=k-r}^{k} A_i + R_k \right)^{-1} \left( \sum_{i=k-r}^{k} b_i - 2 R_k \alpha_k \right).
\]

We now derive the update equations for the SW-VR-RLS algorithm. To rewrite (2) recursively, define

\[
P_k = \left( \sum_{i=k-r}^{k} A_i + R_k \right)^{-1}.
\]
which means that
\[ x_k = -\frac{1}{2} P_k \left( \sum_{i=k-r}^k b_i - 2R_k \alpha_k \right). \]

and
\[
P_k^{-1} = \sum_{i=k+1-r}^{k+1} A_i + R_{k+1}
= \sum_{i=k-r}^k A_i + A_{k+1} - A_{k-r} + R_{k+1} - R_k
= P_k^{-1} + A_{k+1} - A_{k-r} + R_{k+1} - R_k.
\]

Now,
\[
x_{k+1} = -\frac{1}{2} P_{k+1} \left( \sum_{i=k+1-r}^{k+1} b_i - 2R_{k+1} \alpha_{k+1} \right)
= -\frac{1}{2} P_{k+1} \left( \sum_{i=k-r}^k b_i + b_{k+1} - b_{k-r} - 2R_{k+1} \alpha_{k+1} \right).
\]

and it follows from (3) that
\[
x_{k+1} = -\frac{1}{2} P_{k+1} \left( -2P_{k+1}^{-1} x_k + 2R_k \alpha_k + b_{k+1} - b_{k-r} - 2R_k \alpha_{k+1} \right)
= -\frac{1}{2} P_{k+1} \left( -2P_{k+1}^{-1} x_k + 2R_k \alpha_k + b_{k+1} - b_{k-r} - 2R_k \alpha_{k+1} \right)
= x_k - P_{k+1} \left( (A_{k+1} - A_{k-r})x_k + (R_{k+1} - R_k)x_k + R_k \alpha_k + \frac{1}{2} (b_{k+1} - b_{k-r}) - R_{k+1} \alpha_{k+1} \right).
\]

To rewrite \( P_{k+1} \) recursively, consider the decomposition
\[ A_{k+1} = \psi_{k+1} \psi_{k+1}^T, \]

where \( \psi_{k+1} \in \mathbb{R}^{n \times n_{k+1}} \) and \( n_{k+1} \triangleq \text{rank}(A_{k+1}) \). Consequently,
\[ P_{k+1} = (P_k^{-1} + A_{k+1} - A_{k-r} + R_{k+1} - R_k)^{-1}, \]

where the inverse exists since \( \sum_{i=k-r}^k A_i + R_k \) is positive definite. Next, define
\[ M_{k+1} \triangleq (P_k^{-1} + R_{k+1} - R_k - A_{k-r})^{-1}, \]

where the inverse exists since
\[ P_k^{-1} + R_{k+1} - R_k - A_{k-r} = \sum_{i=k-r+1}^k A_i + R_{k+1}, \]

which is assumed to be positive definite. It follows from (4)-(6) that
\[ P_{k+1} = (M_{k+1}^{-1} + A_{k+1})^{-1} = (M_{k+1}^{-1} + \psi_{k+1} \psi_{k+1}^T)^{-1}. \]

Using the matrix inversion lemma
\[ (X + UCV)^{-1} = X^{-1} - X^{-1}U \times (C^{-1} + VX^{-1}U)^{-1} VX^{-1}, \]

with \( X \triangleq M_{k+1}^{-1}, \ U \triangleq \psi_{k+1}, \ C \triangleq I, \) and \( V \triangleq \psi_{k+1}^T, \) it follows that
\[ P_{k+1} = M_{k+1} \left( I_n - \psi_{k+1} \left( I_{n_{k+1}} + \psi_{k+1}^T M_{k+1} \psi_{k+1} \right)^{-1} \times \psi_{k+1}^T M_{k+1} \right). \]

Next, define
\[ Q_{k+1} \triangleq (P_k^{-1} + R_{k+1} - R_k)^{-1}, \]

where this inverse exists since \( P_k^{-1} + R_{k+1} - R_k \geq M_{k+1}^{-1}, \) and thus \( Q_{k+1} \leq M_{k+1}. \) It follows from (4), (6), and (9) that
\[ M_{k+1} = (Q_{k+1}^{-1} - A_{k-r})^{-1} = (Q_{k+1}^{-1} - \psi_{k-r} \psi_{k-r}^T)^{-1}. \]

Using (7) with \( X \triangleq Q_{k+1}^{-1}, \ U \triangleq \psi_{k-r}, \ C \triangleq -I, \) and \( V \triangleq \psi_{k-r}^T, \) it follows that
\[ M_{k+1} = Q_{k+1} \left( I_n - \psi_{k-r} \left( -I_{n_{k+1}} + \psi_{k-r}^T Q_{k+1} \psi_{k-r} \right)^{-1} \times \psi_{k-r}^T Q_{k+1} \right). \]

Next, consider the decomposition
\[ R_{k+1} - R_k = \phi_{k+1} S_{k+1} \phi_{k+1}^T, \]

where \( \phi_{k+1} \in \mathbb{R}^{n \times m_{k+1}}, \ m_{k+1} \triangleq \text{rank}(R_{k+1} - R_k), \) and \( S_{k+1} \in \mathbb{R}^{m_{k+1} \times m_{k+1}} \) is a matrix of the form
\[
S_{k+1} \triangleq \begin{bmatrix}
\pm 1 & 0 & \cdots \\
0 & \pm 1 & \ddots \\
\vdots & \ddots & \ddots & \pm 1
\end{bmatrix}.
\]

It follows from (9) and (10) that
\[ Q_{k+1} \triangleq \left( P_k^{-1} + \phi_{k+1} S_{k+1} \phi_{k+1}^T \right)^{-1}. \]

Using (7) with \( X \triangleq P_k^{-1}, \ U \triangleq \phi_{k+1}, \ C \triangleq S_{k+1}, \) and \( V \triangleq \phi_{k+1}^T, \) it follows that
\[ Q_{k+1} = P_k \left( I_n - \phi_{k+1} \left( S_{k+1} + \phi_{k+1}^T P_k \phi_{k+1} \right)^{-1} \times \phi_{k+1}^T P_k \right). \]

In summary, for \( k \geq 0, \) the recursive minimizer of (1) is
given by
\[ Q_{k+1} = P_k \left( I_n - \phi_{k+1} (S_{k+1} + \phi_{k+1}^T P_k \phi_{k+1})^{-1} \right. \]
\[ \times \left. \phi_{k+1}^T P_k \right), \]
(11)
\[ M_{k+1} = Q_{k+1} \left( I_n - \psi_{k} \left( -I_{n_{k+1}} + \psi_{k}^T Q_{k+1} \psi_{k} \right)^{-1} \times \psi_{k}^T Q_{k+1} \right), \]
(12)
\[ P_{k+1} = M_{k+1} \left( I_n - \psi_{k+1} \left( I_{n_{k+1}} + \psi_{k+1}^T M_{k+1} \psi_{k+1} \right)^{-1} \times \psi_{k+1}^T M_{k+1} \right), \]
(13)
\[ x_{k+1} = x_k - P_{k+1} \left( (A_{k+1} - A_k)x_k + (R_{k+1} - R_k)x_k \right. \]
\[ + R_k \alpha_k + \frac{1}{2} (b_{k+1} - b_k - R_{k+1} \alpha_{k+1}), \]
(14)

where \( x_0 = \alpha_0, P_0 = R_0^{-1}, \psi_{k+1} \) is given by (4), and \( \phi_{k+1} \) is given by (10). In the case where the regularization weighting is constant, that is, for all \( k \geq 0, R_k = R_0 > 0 \), (11) simplifies to \( Q_{k+1} = P_k \), and thus propagation of \( Q_k \) is not required.

### III. Setup for Numerical Simulations

For all \( k \geq 0 \), let \( x_{k, \text{opt}} \in \mathbb{R}^n, \psi_{k} \in \mathbb{R}^n \), where its \( i \)th entry \( \psi_{k, i} \) is generated from a zero mean, unit variance Gaussian distribution. The entries of \( \psi_k \) are independent. Define
\[ \beta_k \triangleq \psi_{k, \text{opt}}^T x_{k, \text{opt}}. \]

Let \( l \) be the number of data points. Define
\[ \sigma_{\psi, i} \triangleq \sqrt{\frac{1}{l} \sum_{k=1}^{l} \psi_{k, i}^2} \overset{l \to \infty}{\longrightarrow} 1, \]
\[ \sigma_\beta \triangleq \sqrt{\frac{1}{l} \sum_{k=1}^{l} \beta_k^2} \overset{l \to \infty}{\longrightarrow} \sqrt{x_{k, \text{opt}}^T x_{k, \text{opt}}}. \]

Next, for \( i = 1, \ldots, n \), let \( N_{k, i} \in \mathbb{R} \) and \( M_{k} \in \mathbb{R} \) be generated from zero-mean Gaussian distributions with variances \( \sigma_{N, i}^2 \) and \( \sigma_M^2 \), respectively, where \( \sigma_{N, i} \) and \( \sigma_M \) are determined from the signal-to-noise ratio (SNR). More specifically, for \( i = 1, \ldots, n \),
\[ \text{SNR}_{\psi, i} \triangleq \frac{\sigma_{\psi, i}}{\sigma_{N, i}}, \quad \text{and} \quad \text{SNR}_\beta \triangleq \frac{\sigma_\beta}{\sigma_M}, \]
where, for \( i = 1, \ldots, n \), \( \sigma_{N, i} = \sqrt{\frac{1}{K} \sum_{k=1}^{K} N_{k, i}} \) and \( \sigma_M = \sqrt{\frac{1}{K} \sum_{k=1}^{K} M_{k}^2} \). For \( k \geq 0 \), define \( A_k \triangleq (\psi_k + N_k) (\psi_k + N_k)^T \) and \( b_k \triangleq -2(\beta_k + M_k) (\psi_k + N_k) \), where \( N_k \) is the noise in \( \psi_k \) and \( M_k \) is the noise in \( \beta_k \). Define
\[ z_1 \triangleq \begin{bmatrix} 0.08 & -1.12 & 1.6 & 1.5 & -2.2 & -2.1 & 0.32 \end{bmatrix}^T, \]
\[ z_2 \triangleq \begin{bmatrix} -1.11 & -0.2 & 1.1 & -0.2 & 0.4 & 0.23 & -2.5 \end{bmatrix}^T. \]

Unless otherwise specified, for all \( k \geq 0 \), \( x_{k, \text{opt}} = z_1, \alpha_k = x_0, \) and \( x_0 = 0_{7 \times 1} \). Define the performance
\[ \varepsilon_k \triangleq \frac{||x_{k, \text{opt}} - x_k||}{||x_{k, \text{opt}}||}. \]

### IV. Numerical Simulations of SW-VR-RLS with Noiseless Data

In this section, we investigate the effect of window size \( r, R_k \), and \( \alpha_k \) on SW-VR-RLS. The data contain no noise, specifically, for all \( k \geq 0, N_k = 0_{7 \times 1} \), and \( M_k = 0 \).

#### A. Effect of Window Size

In the following example, we test SW-VR-RLS for three different values of \( r \). Specifically, \( r = 1, r = 10, \) or \( r = 50 \). In all cases, \( \alpha_k = x_{k-1} \), for all \( k \geq 0, R_k = I_{7 \times 7}, A_k \) and \( b_k \) are the same for all cases, and
\[ x_{k, \text{opt}} = \begin{cases} z_1, & 0 \leq k \leq 115, \\ z_2, & k > 115. \end{cases} \]

For this example, Figure 1 shows that, for \( k \leq 115 \), larger values of \( r \) yield faster convergence of \( \varepsilon_k \) to zero. For \( k > 115, x_{k, \text{opt}} \neq x_{115, \text{opt}} \), and larger values of \( r \) yield faster convergence of \( \varepsilon_k \) to zero; however, larger values of \( r \) can yield worse transient performance because the larger window retains the data relating to \( z_1 \) for more time steps.

#### B. Effect of \( R_k \)

In this section, we examine the effect of \( R_k \), where, for all \( k \geq 0, R_k \) is constant. In the following example, we test SW-VR-RLS for three different \( R_k \). Specifically, for all \( k \geq 0, R_k = 10I_{7 \times 7}, R_k = I_{7 \times 7}, R_k = 0.1I_{7 \times 7} \). In all cases, for all \( k \geq 0, A_k \) and \( b_k \) are the same, \( \alpha_k = x_{k-1}, \) \( r = 15, \) and
\[ x_{k, \text{opt}} = \begin{cases} z_1, & 0 \leq k \leq 115, \\ z_2, & k > 115. \end{cases} \]

For this example, Figure 2 shows that, for \( k \leq 115 \), smaller values of \( R_k \) yields faster convergence of \( \varepsilon_k \) to zero. Similarly, for \( k > 115, x_{k, \text{opt}} \neq x_{115, \text{opt}} \), and smaller values of \( R_k \) yields faster convergence of \( \varepsilon_k \) to zero.
C. Loss of Persistency

In this section, we study the effect of loss of persistency on SW-VR-RLS. More specifically, for all \( k \geq 50 \), \( A_k = A_{50} \) and \( b_k = b_{50} \). Moreover, for all \( k \geq 0 \), \( R_k = 0.1I_{7 \times 7} \), \( r = 15 \), and \( \alpha_k = x_{k-1} \). For this example, Figure 3 shows that \( \varepsilon_k \) approaches zero; however, Figure 4 shows that \( \|P_k\| \) increases after the data lose persistency, but \( \|P_k\| \) remains bounded.

V. NUMERICAL SIMULATIONS WITH NOISY DATA

In this section, we investigate the effect of window size \( r \), \( R_k \), and \( \alpha_k \) on SW-VR-RLS when the data have noise. More specifically, for all \( k \geq 0 \), \( M_k \) and \( N_{k,i} \) are generated from zero mean Gaussian distributions with variances depending on \( \text{SNR}_{\psi,i} \) and \( \text{SNR}_\beta \).

A. Effect of \( \alpha_k \)

In this section, we first compare SW-VR-RLS for different choices of \( \alpha_k \). More specifically, we let \( \alpha_k = L_\nu(k) \) where

\[
L_\nu(k) \triangleq \begin{cases} 
  x_{k-1}, & 0 < k \leq \nu, \\
  x_{k-\nu}, & k > \nu,
\end{cases}
\]

Next, we let \( \alpha_k = W_\rho(k) \) where

\[
W_\rho(k) \triangleq \begin{cases} 
  x_0, & k = 1, \\
  \frac{1}{\rho} \sum_{i=1}^{k-1} x_{k-i}, & 1 < k \leq \rho, \\
  x_{k-\rho}, & k > \rho,
\end{cases}
\]

where \( \rho \) is a positive integer. In the following example, we test SW-VR-RLS for three different \( \nu \). Specifically, \( \nu = 1 \), \( \nu = 5 \) or \( \nu = 15 \). In all cases, for all \( k \geq 0 \), \( A_k \) and \( b_k \) are the same, \( R_k = I_{7 \times 7} \), \( \text{SNR}_\beta = \text{SNR}_{\psi,i} = 5 \) and \( r = 10 \). For this example, Figure 5 shows that larger values of \( \nu \) yield smaller asymptotic values of \( \varepsilon_k \).

B. Effect of Window Size

In the following example, we test SW-VR-RLS for three different \( r \). Specifically, \( r = 50 \), \( r = 7 \), or \( r = 1 \). In all cases, \( \alpha_k = x_{k-1} \), \( \text{SNR}_\beta = \text{SNR}_{\psi,i} = 5 \), for all \( k \geq 0 \),
\( R_k = I_{7 \times 7}, \ A_k \) and \( b_k \) are the same for all cases, and
\[
x_{k, \text{opt}} = \begin{cases} 
  z_1, & 0 \leq k \leq 115, \\
  z_2, & k > 115.
\end{cases}
\]

For this example, Figure 7 shows that \( r = 50 \) yields a smaller asymptotic value of \( \varepsilon_k \) than \( r = 1 \) and \( r = 7 \). However, this trend is not monotonic since \( r = 7 \) yields a larger asymptotic value of \( \varepsilon_k \) than \( r = 1 \). Numerical tests suggest that the asymptotic value of \( \varepsilon_k \) increases as the window size is increased from \( r = 1 \) until it peaks at a certain value of \( r \). After this the asymptotic value of \( \varepsilon_k \) decreases as \( r \) increases.

\[ x_{k, \text{opt}} = \begin{cases} 
  z_1, & 0 \leq k \leq 115, \\
  z_2, & k > 115.
\end{cases} \]

For this example, Figure 7 shows that \( r = 50 \) yields a smaller asymptotic value of \( \varepsilon_k \) than \( r = 1 \) and \( r = 7 \). However, this trend is not monotonic since \( r = 7 \) yields a larger asymptotic value of \( \varepsilon_k \) than \( r = 1 \). Numerical tests suggest that the asymptotic value of \( \varepsilon_k \) increases as the window size is increased from \( r = 1 \) until it peaks at a certain value of \( r \). After this the asymptotic value of \( \varepsilon_k \) decreases as \( r \) increases.

\[ x_{k, \text{opt}} = \begin{cases} 
  z_1, & 0 \leq k \leq 115, \\
  z_2, & k > 115.
\end{cases} \]

For this example, Figure 7 shows that \( r = 50 \) yields a smaller asymptotic value of \( \varepsilon_k \) than \( r = 1 \) and \( r = 7 \). However, this trend is not monotonic since \( r = 7 \) yields a larger asymptotic value of \( \varepsilon_k \) than \( r = 1 \). Numerical tests suggest that the asymptotic value of \( \varepsilon_k \) increases as the window size is increased from \( r = 1 \) until it peaks at a certain value of \( r \). After this the asymptotic value of \( \varepsilon_k \) decreases as \( r \) increases.

\[ x_{k, \text{opt}} = \begin{cases} 
  z_1, & 0 \leq k \leq 115, \\
  z_2, & k > 115.
\end{cases} \]

For this example, Figure 7 shows that \( r = 50 \) yields a smaller asymptotic value of \( \varepsilon_k \) than \( r = 1 \) and \( r = 7 \). However, this trend is not monotonic since \( r = 7 \) yields a larger asymptotic value of \( \varepsilon_k \) than \( r = 1 \). Numerical tests suggest that the asymptotic value of \( \varepsilon_k \) increases as the window size is increased from \( r = 1 \) until it peaks at a certain value of \( r \). After this the asymptotic value of \( \varepsilon_k \) decreases as \( r \) increases.

\[ x_{k, \text{opt}} = \begin{cases} 
  z_1, & 0 \leq k \leq 115, \\
  z_2, & k > 115.
\end{cases} \]

For this example, Figure 7 shows that \( r = 50 \) yields a smaller asymptotic value of \( \varepsilon_k \) than \( r = 1 \) and \( r = 7 \). However, this trend is not monotonic since \( r = 7 \) yields a larger asymptotic value of \( \varepsilon_k \) than \( r = 1 \). Numerical tests suggest that the asymptotic value of \( \varepsilon_k \) increases as the window size is increased from \( r = 1 \) until it peaks at a certain value of \( r \). After this the asymptotic value of \( \varepsilon_k \) decreases as \( r \) increases.

\[ x_{k, \text{opt}} = \begin{cases} 
  z_1, & 0 \leq k \leq 115, \\
  z_2, & k > 115.
\end{cases} \]
VI. CONCLUSIONS

In this paper, we presented a sliding-window variable-regularization recursive least squares (SW-VR-RLS) algorithm. This algorithm operates on a finite window of data, where old data are discarded as new data become available. Furthermore, this algorithm allows for a time-varying regularization term in the sliding-window RLS cost function. More specifically, SW-VR-RLS allows us to vary both the weighting in the regularization as well as what is being weighted, that is, the regularization term can weight the difference between the next state estimate and a time-varying vector of parameters rather than the initial state estimate.

REFERENCES