A RECURSIVE TOTAL LEAST SQUARES ALGORITHM FOR DECONVOLUTION PROBLEMS

Piet Vandaele, Marc Moonen

ESAT - Katholieke Universiteit Leuven
K. Mercierlaan 94, 3001 Heverlee -Belgium
tel 32/16/32 18 00 fax 32/16/32 19 70
piet.vandaele@esat.kuleuven.ac.be marc.moonen@esat.kuleuven.ac.be

ABSTRACT

Deconvolution problems are encountered in signal processing applications where an unknown input signal can only be observed after propagation through one or more noise corrupted FIR channels. The first step in recovering the input usually entails an estimation of the FIR channels through training based or blind algorithms. The 'standard' procedure then uses least squares estimation to recover the input. A recursive implementation with constant computational cost is based on the Kalman filter. In this paper we focus on a total least squares based approach, which is more appropriate if errors are expected both on the output samples and the estimates of the FIR channels. We will develop a recursive total least squares algorithm (RTLS) which closely approximates the performance of the non-recursive TLS algorithm and this at a much lower computational cost.

1. INTRODUCTION

Deconvolution problems arise in digital signal processing applications as diverse as high speed data transmission, reverberation cancelation, reflection seismology and image restoration [4, 2]. In these situations an unknown input signal can only be observed after propagation through one or more noise corrupted FIR channels. The FIR channels can be estimated through training based or blind algorithms. Once these estimates are available, the input signal can be retrieved in several ways. In the case of digital communications the input belongs to a finite alphabet and a Viterbi algorithm can be used. However, the complexity of the Viterbi algorithm grows exponentially with the length of the FIR channels. Therefore cheaper solutions are desirable. One could alternatively estimate the transmitted sequence as the least squares solution of the problem. If we assume that errors are present in both the output samples and the estimates of the channel parameters, a total least squares (TLS) solution [1],[5] seems to be the appropriate way to recover the input [1]. For reasons of computational complexity it is desirable to turn the classical TLS algorithm into an adaptive form. In this paper we will present such a recursive solution which has a constant computational complexity.
In section 2 the data model in presented. Section 3 develops the new algorithm. Section 4 presents simulation results and finally in section 5 some conclusions are drawn.

Piet Vandaele is a Research Assistant supported by the Flemish I.W.T. (the Flemish Institute for the Advancement of Scientific-Technological Research in Industry). Marc Moonen is a Research Associate with the Fund for Scientific Research - Flanders (Belgium) (F.W.O.). This research work was carried out at the ESAT laboratory of the Katholieke Universiteit Leuven, in the framework of a Concerted Action Project of the Flemish Community, entitled *Model-based Information Processing Systems* (GOA/MIPS/95/99/3) as well as the IT-program of the I.W.T., *Integrating Signal Processing Systems* (ITA/GBO/T23) and was partially sponsored by IMEC (Flemish Interuniversity Microelectronics Center) and IUAP P4-02 (1997-2001): Modeling, Identification, Simulation and Control of Complex Systems. The scientific responsibility is assumed by its authors.

2. DATA MODEL

Assuming a single input-multiple output (SIMO) system, we define $\mathbf{y}[k] = [y_1[k] \dots y_M[k]]^T$, $\mathbf{x}[k] = [x[k-L] \dots x[k]]^T$ and $\mathbf{n}[k] = [n_1[k] \dots n_M[k]]^T$ as respectively the output, input and noise vector at time k. This results in the following input/output formula:

$$\mathbf{y}[k] = \underbrace{\begin{bmatrix} h_1^{(k)}[L] & \dots & h_1^{(k)}[1] & h_1^{(k)}[0] \\ \vdots & & & \vdots \\ h_M^{(k)}[L] & \dots & h_M^{(k)}[1] & h_M^{(k)}[0] \end{bmatrix}}_{\boxed{H^{(k)}}} \mathbf{x}[k] + \mathbf{n}[k]$$

Note that there are M (possibly time varying) FIR channels $h_1^{(k)}[\cdot]$,...,$h_M^{(k)}[\cdot]$ of length L. If we then stack data vectors over $N-L$ symbol periods and define $Y_{L+1|N} = [\mathbf{y}[L+1]^T \cdots \mathbf{y}[N]^T]^T$, $X_{1|N} = [x[1] \cdots x[N]]^T$ and $N_{L+1|N} =$

[1] Note that a structured total least squares solution would even perform better, but is computationally very expensive [3].

$[\mathbf{n}[L+1]^T \cdots \mathbf{n}[N]^T]^T$ we obtain:

$$Y_{L+1|N} = \underbrace{\begin{bmatrix} \boxed{H^{(L+1)}} & 0 & 0 & \dots \\ 0 & \boxed{H^{(L+2)}} & 0 & \dots \\ & & \ddots & \\ 0 & 0 & \dots & \boxed{H^{(N)}} \end{bmatrix}}_{\mathcal{H}_N} \cdot X_{1|N} + N_{L+1|N} \quad (1)$$

In this paper we will assume that $\mathcal{H}_N$ has been estimated through training based or blind algorithms and focus on the recovery of the input vector $X_{1|N}$ using a TLS type approach.

The TLS solution [1],[5] of equation 1 is proportional to the right singular vector corresponding to the smallest singular value of the matrix $A_N = [\mathcal{H}_N Y_{L+1|N}]$. A scaling is applied such that the last entry in the singular vector equals -1.

Notice that the dimensions of the matrix A_N (and hence the complexity in computing the right singular vector) grow in each iteration step so that a straightforward application of the TLS procedure results in a growing computational complexity per iteration. The aim of this paper is to exploit the sparsity of $\mathcal{H}_N$ to obtain a recursive total least squares deconvolution algorithm with constant computational requirements. In a first step we will derive an algorithm with growing matrix dimensions, which we will then transform into a solution with constant computational requirements.

3. RTLS ALGORITHM

3.1. Version 1

At time $k-1$, a new $H^{(k-1)}$ has been estimated, and a new data vector $\mathbf{y}[k-1]$ becomes available. At that moment, all the previous equations in $H^{(L+1)}, \dots, H^{(k-2)}$ together with the new equations in $H^{(k-1)}$ may be assembled in a matrix to form one large overdetermined set of equations in $X_{1|k-1}$. The total least squares solution $X_{1|k-1}$ then equals the right singular vector (up to a scaling factor) associated with the smallest singular value of A_{k-1}, with $A_{k-1} = [\mathcal{H}_{k-1} Y_{L+1|k-1}]$.

This right singular vector may now be computed in an iterative way, *i.e.* for the computation of the right singular vector at time k one may use the results obtained at time step $k-1$. The singular vector is updated in two steps.

The first step is a QR-updating operation in which the triangular R-factor of the QR-decomposition of the matrix A_{k-1} is tracked [1]. In the second step the singular vector is computed through inverse iteration [1] using the R-factor computed in the first step.

3.1.1. QR Updating

Denote the QR-decomposition of A_{k-1} as $A_{k-1} = Q_{k-1} \cdot R_{k-1}$ and assume that the $k \times k$ matrix R_{k-1} is known at time step $k-1$.

To go from time step $k-1$ to k, three operations have to be performed on the matrix A_{k-1}. First we have to add a column of zeros to A_{k-1}. Secondly we change the last and one but last column of this updated A_{k-1}. Finally we append the rows of

$$\alpha_k = \left[\; 0_{M \times (k-(L+1))} \quad \boxed{H^{(k)}} \quad \Big| \quad \mathbf{y}[k] \; \right] \quad (2)$$

to the updated A_{k-1}.

The first and second operation imply respectively that R_{k-1} is extended with an extra row and column of zeros and that this updated R_{k-1} is multiplied on the rhs with P:

$$P = \begin{bmatrix} I_{k-1} & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (3)$$

This modified R_{k-1} matrix remains upper triangular and can be updated to R_k using QR-updating:

- $E \Leftarrow \begin{bmatrix} R_{k-1} & 0 \\ 0 & 0 \end{bmatrix} \cdot P$
- for $r = 1, \dots, M$

$$\left[\frac{E}{0}\right] \Leftarrow Q_r^H \cdot \left[\frac{E}{\alpha_k(r,:)}\right] \quad (4)$$

  end
- $R_k \Leftarrow E$

The Q_r^H matrices are composed of Givens rotations which cancel out the bottom part in the rhs matrices.

3.1.2. Inverse Iteration

Since the right singular vectors of R_k equal the eigenvectors of $(R_k^H \cdot R_k)$, the right singular vector v_k corresponding to the smallest singular value of R_k, can be computed through inverse iteration [1]:

- initialize v_k
- iteration:
  for $r = 1, \dots, P$

$$R_k^H \cdot R_k \cdot s_k = v_k \quad (5)$$

$$v_k = -\frac{s_k}{s_k(k+1)} \quad (6)$$

  end

The first equation in the iteration is solved for s_k. The last equation ensures that the last component of the vector v_k is -1 and hence forms the TLS solution. Equation 5 can be refined as:

$$R_k^H \cdot w_k = v_k \quad (7)$$

$$R_k \cdot s_k = w_k \quad (8)$$

The first equation is solved for w_k (using forward substitution) and the second one for s_k (using back substitution). Clearly the computations required in this procedure grow in time since the dimensions of R_k and R_k^H grow. The next section explains how to transform the above algorithm in an algorithm with constant computational requirements.

3.2. Version 2

The algorithm with constant computational requirements is based on the following modification. At time step k, we assume that the first $k+1-j$ components of the vector v_k have been accurately estimated (the only condition on j is that $j > \lfloor \frac{L}{M-1} \rfloor + L + 1$[2]). These components can then be replaced by a fixed value in all subsequent starting vectors for the inverse iteration step, namely:

$$v_k(1:k+1-j) \approx \hat{x}(1:k+1-j) \tag{9}$$

In a digital communications context $\hat{x}(1:k+1-j)$ could e.g. represent the hard decisions on the first $k+1-j$ bits. This means that the inverse iteration step will truly iterate only over the last j components of the vector v_k, *i.e.* $v_k(k+2-j:k+1)$. In order to simplify notation, we introduce the following definitions:

$$\begin{aligned}
R^H_{11_k} &= R^H_k(1:k+1-j, 1:k+1-j) \\
R^H_{21_k} &= R^H_k(k+2-j:k+1, 1:k+1-j) \\
R^H_{22_k} &= R^H_k(k+2-j:k+1, k+2-j:k+1) \\
w_{1_k} &= w_k(1:k+1-j) \\
w_{2_k} &= w_k(k+2-j:k+1) \\
\hat{x}_k &= \hat{x}(1:k+1-j) \\
v_{2_k} &= v_k(k+2-j:k+1) \\
R_{22_k} &= R_k(k+2-j:k+1, k+2-j:k+1) \\
s_{2_k} &= s_k(k+2-j:k+1)
\end{aligned}$$

Using equations 7 and 8 the loop equations in the inverse iteration step can be rewritten as:

1. compute w_{1_k} (forward substitution):
$$R^H_{11_k} \cdot w_{1_k} = \hat{x}_k \tag{10}$$

2. compute w_{2_k} (forward substitution):
define the $j \times 1$ vector f_k as:
$$f_k = R^H_{21_k} \cdot w_{1_k} \tag{11}$$
for $k > j-1$ and $f_{j-1} = 0_{j\times 1}$.
$$R^H_{22_k} \cdot w_{2_k} = v_{2_k} - f_k \tag{12}$$

3. compute s_{2_k} (back substitution):
$$R_{22_k} \cdot s_{2_k} = w_{2_k} \tag{13}$$

4. finally, the new values for the last j entries in v_k are obtained as:
$$v_{2_k} = -\frac{s_{2_k}}{s_k(k+1)} \tag{14}$$

Equations 12, 13 and 14 are executed P times. It will now be shown how one can avoid the computation of $w_k(1:k+1-j)$, a vector with growing dimensions. To this aim, f_k is computed recursively, *i.e.* using f_{k-1}, which will then allow to set up a fully adaptive algorithm.
The crucial observation is that the first $k-L-1$ rows of R_{k-1} remain unchanged in the QR-updating step after iteration $k-1$ (except for the insertion of some zero columns), because of the initial zero block in α_k, see equation 2. Combining equations 2 and 10, it can then be shown that f_k can be computed recursively as:

$$f_k = \begin{bmatrix} f_{k-1}(2:j-1) \\ 0 \\ f_{k-1}(j) \end{bmatrix} + \frac{\hat{x}(k+1-j) - f_{k-1}(1)}{R^H_{k-1}(k+1-j, k+1-j)} \cdot \begin{bmatrix} R^H_{k-1}(k+2-j:k-1, k+1-j) \\ 0 \\ R^H_{k-1}(k, k+1-j) \end{bmatrix} \tag{15}$$

for $k > j-1$ and $f_{j-1} = 0_{j\times 1}$.
The proof is omitted here due to space limitations. In the inverse updating step, now only f_k and R_{22_k} are needed. In the QR-updating step, we can easily update R_{22_k} because of equation 2. The final algorithm is summarized below (at this point, time indices are omitted since matrices are overwritten). We adopt to the following definition for β_k:

- for $L+1 \le k < j-1$:
$\beta_k = \begin{bmatrix} 0_{M\times(k-(L+1))} & H^{(k)} & 0_{M\times(j-k-1)} & \mathbf{y}[k] \end{bmatrix}$
- for $k \ge j-1$
$\beta_k = \alpha_k(:, k+2-j:k+1)$

1. initialization:
$$\begin{aligned} v &= \begin{bmatrix} 0_{1\times(j-1)} & -1 \end{bmatrix}^T \\ R &= 0_{j\times j} \\ f &= 0_{j\times 1} \end{aligned}$$
2. iteration equations:
for $k = L+1, \ldots$
   - update R
   for $r = 1, \ldots, M$
   $$\begin{bmatrix} R \\ \hline 0 \end{bmatrix} \Leftarrow Q_r^H \cdot \begin{bmatrix} R \\ \hline \beta_k(r,:) \end{bmatrix}$$
   end
   if $k \ge j-1$

[2] This condition assures that the set of equations is overdetermined at time $j-1$, when the first input symbol is estimated: $(j-1-L)\cdot M > j-1$. Furthermore it is seen from the formula that j is at least $L+2$, which will be useful further on.

- update the v-vector,
  for $r = 1, \ldots, P$

$$\begin{aligned} R^H \cdot w &\Leftarrow v - f \\ R \cdot v &\Leftarrow w \\ v &\Leftarrow -\frac{v}{v(j)} \end{aligned}$$

  end

- estimate the input:

$$\hat{x}[k+2-j] = v(1)$$

- append a column of zeros:

$$f = \begin{bmatrix} f(2:j-1) \\ 0 \\ f(j) \end{bmatrix} + \frac{\hat{x}[k+2-j] - f(1)}{R^H(1,1)} \cdot \begin{bmatrix} R^H(2:j-1,1) \\ 0 \\ R^H(j,1) \end{bmatrix}$$

$$R = \begin{bmatrix} R(2:j,2:j-1) & 0_{(j-1)\times 1} & R(2:j,j) \\ 0_{1\times(j-2)} & 0 & 0 \end{bmatrix}$$

$$v = \begin{bmatrix} v(2:j-1) \\ 0 \\ v(j) \end{bmatrix}$$

end
end

4. SIMULATION RESULTS

This section compares the performance of the RTLS algorithm with respect to the non-recursive TLS and least squares (LS) solutions. The simulation parameters are depicted in table 1.

M	L	j	N	P
10	5	12	100	1

Table 1: *Simulation parameters*

The input sequence x was drawn from a Gaussian constellation with unit variance. The channels were assumed to be randomly time varying and at each time instant the channel coefficients were drawn from a Gaussian distribution with unit variance. Subsequently some Gaussian noise was added to the (noise free) output samples and the channel parameters. The noise added on the channel and the outputs was of equal variance σ^2[3]. Finally the different algorithms were run using the noisy samples and the results were averaged over 100 Monte-Carlo runs. The performance index used is the mean square error, which is computed as:

$$MSE = E\{\sum_{i=1}^{N} (\hat{x}[i] - x[i])^2 / N\}$$

σ^2 / alg.	RTLS	TLS	LS
0.3^2	0.00618	0.00619	0.00704
0.5^2	0.01996	0.01963	0.02547
0.7^2	0.05847	0.05830	0.05923

Table 2: *MSE for varying noise variance*

Table 2 shows the MSE for different values of the noise variance. The TLS and RTLS algorithm outperform the LS algorithm and the RTLS algorithm closely approximates the performance of the TLS solution.
Other simulations showed that an increase of M enlarges the performance gap between the (R)TLS and LS algorithms. Increasing the number of iterations P in the inverse iteration step had very little influence. An increase in j left performance almost unaltered. On the other hand when the noise level was increased beyond a certain threshold level the LS solution provided more accurate results than the TLS/RTLS algorithms. Finally note that for the simulation parameters shown in table 1 (thus even for relatively short bursts) the number of flops required by the TLS algorithm exceeded that of the RTLS algorithm by roughly a factor 800.

5. CONCLUSIONS

In this paper we have presented a new recursive total least squares algorithm for input sequence detection in deconvolution problems. The simulation results showed that the algorithm closely approximates the performance of the classical TLS algorithm at a much lower computational complexity. Hence the algorithm might be useful in deconvolution problems whenever a TLS type approach is expected to give better results than a LS type approach.

6. REFERENCES

[1] G. H. Golub and C. F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, 1989.

[2] S. Haykin, editor. *Blind Deconvolution.* Prentice Hall, 1994.

[3] P. Lemmerling, S. Van Huffel, and B. De Moor. Structured Total Least Squares Problems: Formulations, Algorithms and Applications. In *Proceedings of 2nd International Workshop on TLS and Errors in Variables Modelling*, pages 215–223. SIAM, Philadelphia, 1997.

[4] J.M. Mendel. *Lessons in Estimation Theory for Signal Processing, Communications, and Control*. Prentice Hall Signal Processing Series, 1995.

[5] S. Van Huffel and J. Vandewalle. *The Total Least Squares Problem, Computational Aspects and Analysis.* SIAM, 1991.

[3] However, the algorithm can easily be adapted to cope with different variances by including a pre-whitening step.