Determining the initial states in forward-backward filtering

Fredrik Gustafsson*

Submitted to IEEE Trans. on Signal Processing

Abstract

Forward-backward filtering is a common tool in off-line filtering. Filtering first forwards and then backwards and the other way around do not give the same result generally. Here we propose a method to choose the initial state in the filter to eliminate this discrepancy. The objectives are to obtain uniqueness and to remove transients in both ends.

1 Introduction

In batchwise signal processing one has the possibility not only to filter a signal forwards in time as usual, but also apply a filter backwards on the filtered signal. Forward-backward filtering is needed in the following applications for example:

- Zero-phase filtering using IIR filters.
- Implementation of non-causal Wiener filters or other filters with poles both inside and outside the unit circle.

The first trick to obtain zero-phase filters is used in many software packages, see for instance the function `filtfilt` in Matlab [2], although it is not much discussed in

*Department of Electrical Engineering, Linköping University, S-58183 Linköping, Sweden

standard text books. The design of non-causal Wiener filters leads to a filter with one stable part and one unstable one as discussed in for instance [1]. The unstable part must be implemented using backward filtering. In both cases, there might be a problem with transients when applied to finite signals.

A filter $G(q)$ becomes all-pass, or zero phase, when applied first forwards, $y_f(t) = G(q)u(t)$, and then backwards on the reversed filtered signal, $y_{fb}^R(t) = G(q)y_f^R(t)$, and finally the output is reversed again. The superindex R stands for reversed sequences, the indexes f for forward and b for backward and q denotes the shift operator $qu(t) = u(t+1)$. We can also first filter backwards and then forwards and obtain $y_{bf}(t)$. In both cases, the total effect is a zero phase filter with transfer function $|G(e^{i\omega})|^2$. However, we have $y_{fb}(t) \neq y_{bf}(t)$ generally due to unknown initial conditions, and this is not satisfactory from the logical point of view. Besides, the FB (forward-backward) filtered sequence has visible transients in the end and the BF filtered sequence at the beginning.

One way to obtain symmetry is to implement a filter with poles both inside and outside the unit circle by using the partial fractions of the stable poles and unstable poles respectively. This is the standard way in non-causal Wiener filtering. The stable filter is applied forwards on the input and the unstable filter backwards and the filter output is the sum of these terms. Since both filters are applied to the input, the result is of course independent of the order of computations. However, there still is a problem with transients, this time in both ends, due to unknown initial conditions.

We will here determine the initial conditions that makes $y_{bf}(t) = y_{fb}(t)$ and at the same time removes the transients. That is, the initial condition of one filter is chosen to match the end of the other filter. Instead of solving the system of equations arising from the condition $y_{bf}(t) = y_{fb}(t)$, we will work in a least squares framework, which immediately gives a feasible implementation. The method is illustrated on filtered white noise. The improvement for short data records or long filter impulse responses is obvious.

2 Determining the initial state

A linear filter can always be expressed as

$$Y = \mathcal{H}U + \mathcal{O}x_0,$$

where $Y = (y_0, y_1, .., y_{N-1})^T$ is the vector of outputs, $U = (u_0, u_1, .., u_{N-1})^T$ is the vector of inputs to the filter and x_0 its initial state. $\mathcal{H}$ is a Toeplitz matrix of impulse response

coefficients

$$\mathcal{H} = \begin{pmatrix} h_0 & 0 & \cdots & 0 \\ h_1 & h_0 & 0 & \vdots \\ \vdots & & \ddots & \\ h_{N-1} & \cdots & h_1 & h_0 \end{pmatrix} \tag{1}$$

where N is the number of inputs, and $\mathcal{O}$ is an "observability" matrix. If the filter is written in state space form

$$\begin{aligned} x(t+1) &= Fx(t) + Gu(t) \\ y(t) &= Hx(t) + Du(t) \end{aligned}$$

then

$$\mathcal{H} = \begin{pmatrix} D & 0 & \cdots & 0 \\ HG & D & 0 & \vdots \\ \vdots & & \ddots & \\ HF^{N-2}G & \cdots & HG & D \end{pmatrix} \tag{2}$$

$$\mathcal{O} = \begin{pmatrix} H \\ HF \\ \vdots \\ HF^{N-1} \end{pmatrix} \tag{3}$$

We introduce the row and column reversing operators R and C with the following easily checked properties:

$$\begin{aligned} A &= BC \Rightarrow \\ A^R &= B^R C \\ A^C &= BC^C \\ BC^R &= B^C C \\ (BC^{RC})^{RC} &= B^{RC} C \\ A^{RC} &= A^T \text{ if } A \text{ Toeplitz} \end{aligned}$$

The forward and backward filters are characterized by the pairs $(\mathcal{H}_f, \mathcal{O}_f)$ and $(\mathcal{H}_b, \mathcal{O}_b)$. Using the notation x_0 and x_{N-1} for the initial conditions for forward and backward filters, the result of forward-backward and backward-forward filtering can be written as

$$\begin{aligned} Y_{fb} &= (\mathcal{H}_b Y_f^R + \mathcal{O}_b x_{N-1})^R \\ &= \mathcal{H}_b^R (\mathcal{H}_f U + \mathcal{O}_f x_0)^R + \mathcal{O}_b^R x_{N-1} \\ &= \mathcal{H}_b^R \mathcal{H}_f^R U + \mathcal{H}_b^R \mathcal{O}_f^R x_0 + \mathcal{O}_b^R x_{N-1} \end{aligned}$$

and

$$\begin{aligned} Y_{bf} &= \mathcal{H}_f Y_b^R + \mathcal{O}_f x_0 \\ &= \mathcal{H}_f (\mathcal{H}_b U^R + \mathcal{O}_b x_{N-1})^R + \mathcal{O}_f x_0 \\ &= \mathcal{H}_f^C \mathcal{H}_b^C U + \mathcal{H}_f \mathcal{O}_b^R x_{N-1} + \mathcal{O}_f x_0 \end{aligned}$$

respectively. Note that $\mathcal{H}_b^R \mathcal{H}_f^R \neq \mathcal{H}_f^C \mathcal{H}_b^C$.

It turns out that there always is a solution to $Y_{fb} - Y_{bf} = 0$. We will not prove this statement. Instead we formulate the problem as minimizing $\|Y_{fb} - Y_{bf}\|_2^2$, which turns out to be easier to solve. We have

$$Y_{fb} - Y_{bf} = (\mathcal{H}_b^R \mathcal{H}_f^R - \mathcal{H}_f^C \mathcal{H}_b^C) U + (\mathcal{H}_b^R \mathcal{O}_f^R - \mathcal{O}_f) x_0 + (I - \mathcal{H}_f) \mathcal{O}_b^R x_{N-1}$$

Since this is a linear expression in x_0 and x_{N-1}, the minimizing arguments are given by the least squares estimate

$$\begin{pmatrix} \widehat{x_0} \\ x_{N-1} \end{pmatrix} = \left[(\mathcal{H}_b^R \mathcal{O}_f^R - \mathcal{O}_f) \ , \ (I - \mathcal{H}_f) \mathcal{O}_b^R \right]^{\#} \left(\mathcal{H}_b^R \mathcal{H}_f^R - \mathcal{H}_f^C \mathcal{H}_b^C \right) U$$

Here # denotes pseudo-inverse. By substituting these estimated initial states into the FB and BF filters we get

$$\begin{aligned} \hat{Y}_{fb} &= \mathcal{H}_b^R \mathcal{H}_f^R U + \left[\mathcal{H}_b^R \mathcal{O}_f^R \ , \ \mathcal{O}_b^R \right] \left[(\mathcal{H}_b^R \mathcal{O}_f^R - \mathcal{O}_f) \ , \ (I - \mathcal{H}_f) \mathcal{O}_b^R \right]^{\#} \left(\mathcal{H}_b^R \mathcal{H}_f^R - \mathcal{H}_f^C \mathcal{H}_b^C \right) U \\ \hat{Y}_{bf} &= \mathcal{H}_f^C \mathcal{H}_b^C U + \left[\mathcal{O}_f \ , \ \mathcal{H}_f \mathcal{O}_b^R \right] \left[(\mathcal{H}_b^R \mathcal{O}_f^R - \mathcal{O}_f) \ , \ (I - \mathcal{H}_f) \mathcal{O}_b^R \right]^{\#} \left(\mathcal{H}_b^R \mathcal{H}_f^R - \mathcal{H}_f^C \mathcal{H}_b^C \right) U \end{aligned}$$

The expressions given above are not useful for implementation, since $N \times N$ matrices are involved. Next we seek a feasible implementation without computing $\mathcal{H}$ explicitely.

3 Fast implementation

In this section we describe how Y_{fb}, without proof claimed to be the same as Y_{bf}, can be computed efficiently. First an exact method is detailed, where the initial states are computed with linear in time complexity, and then a constant complexity approximation is proposed.

We first compute the nominal values of Y_{fb} and Y_{bf} for zero initial conditions using standard filter routines. That is, we have

$$\begin{aligned} Y_{fb}^0 &= \mathcal{H}_b^R \mathcal{H}_f^R U \\ Y_{bf}^0 &= \mathcal{H}_f^C \mathcal{H}_b^C U \end{aligned}$$

We also need arbitrary state space realizations of the filters, (F_f, G_f, H_f, D_f) and (F_b, G_b, H_b, D_b). The "observability" matrixes $\mathcal{O}_f$ and $\mathcal{O}_f$ are computed according to (3). Now, we just have to compute the quantities $\mathcal{S}_{bf} \stackrel{\triangle}{=} \mathcal{H}_b \mathcal{O}_f^R$ and $\mathcal{S}_{fb} \stackrel{\triangle}{=} \mathcal{H}_f \mathcal{O}_b^R$ and we can express the least squares estimate as

$$\begin{pmatrix} \widehat{x_0} \\ x_{N-1} \end{pmatrix} = \left[(\mathcal{S}_{fb}^R - \mathcal{O}_f) \;\; \mathcal{O}_b^R - \mathcal{S}_{bf} \right]^{\#} (Y_{bf}^0 - Y_{fb}^0).$$

Thus, we have

$$Y_{fb} \;=\; Y_{fb}^0 + \left[\mathcal{S}_{bf}^R \;,\; \mathcal{O}_b^R \right] \left[(\mathcal{S}_{fb}^R - \mathcal{O}_f) \;\; \mathcal{O}_b^R - \mathcal{S}_{bf} \right]^{\#} (Y_{bf}^0 - Y_{fb}^0). \tag{4}$$

Next is shown that $\mathcal{S}_{fb}$ can be computed without forming the $N \times N$ matrix $\mathcal{H}_f$. The vectors of impulse response coefficients can be expressed in h_f as

$$h_f = \begin{bmatrix} D_f \\ \mathcal{O}_f(1:N-1,:) G_f \end{bmatrix}.$$

Here $\mathcal{O}_f(1:N-1,:)$ means rows 1 to $N-1$ of $\mathcal{O}_f$. Note that h_f is the first column of $\mathcal{H}_f$. Let $\mathcal{S}_{fb}(t.\,:)$ denote the t'th row of $\mathcal{S}_{fb}$ and $h_f(t)$ the t'th component of h_f. From (1), (2) and (3) we have

$$\begin{aligned} \mathcal{S}_{fb}(N,:) &= (h_f^R)^T \mathcal{O}_b^R = (h_f^T)^C \mathcal{O}_b^R = h_f^T \mathcal{O}_b \\ \mathcal{S}_{fb}(N-1,:) &= \mathcal{S}_{fb}(N,:) F_b - h_f(N) H_b F_b^{N-1} \\ \mathcal{S}_{fb}(t-1,:) &= \mathcal{S}_{fb}(t,:) F_b - h_f(t) H_b F_b^{N-1} \end{aligned}$$

Thus, $\mathcal{S}_{fb}$ can be computed by the recursion

$$\begin{aligned} \mathcal{S}_{fb}(N,:) &= h_f^T \mathcal{O}_b \\ \mathcal{S}_{fb}(t-1,:) &= \mathcal{S}_{fb}(t,:) F_b - h_f(t) H_b F_b^{N-1}, \quad t = N, N-1, ..., 2. \end{aligned}$$

and $\mathcal{S}_{bf}$ is computed similarly. In this way, we do not have to compute the $N \times N$ matrix $\mathcal{H}$ explicitely, but only the $N \times 2n$ matrices $\mathcal{O}$ and $\mathcal{S}$.

4 Fixed complexity approximation

For long signals, it seems to be overkill to work with the complete impulse response h and observability matrix $\mathcal{O}$. If we assume that the impulse response is approximately

zero for $t > M$ a logical approximation in (4) is to replace F^t by 0 if $t > M$. This leads to an $\hat{\mathcal{O}}$ having non-zero elements in its first M rows only, and $\hat{\mathcal{S}}$ having non-zero elements in its last M rows. Thus, we are faced with a least squares problem with $2M$ equations and $2n$ unknowns, so all that is needed is to compute the pseudo-inverse of a $2M \times 2n$ matrix in (4).

5 Simulations

In this section, a band-pass Butterworth filter applied to a white noise sequence[1] will be examined. The filter is of tenth order with (normalized) cut-off frequencies 0.2 and 0.25 and the noise sequence is of length 200. Figure 1 shows first Y_{fb} and Y_{bf} using the initial states in Matlab's `filtfilt`[2] function in the signal processing toolbox and second the result using zero inital states x_0 and x_{N-1}, *i.e.* $Y_{fb} = \mathcal{H}^R\mathcal{H}^R U$ and $Y_{bf} = \mathcal{H}^C\mathcal{H}^C U$. Then follows the signals (4) using optimized initial states. Finally, the result using partial fractions,

$$|H(q)|^2 = \frac{B(q)}{A(q)}\frac{B(q^{-1})}{A(q^{-1})} = \frac{C(q)}{A(q)} + \frac{D(q)}{A(q^{-1})},$$

is shown. As seen, the last two curves are identical showing the symmetry of the proposed initialization method. The 2-norms are as follows:

$$\begin{aligned} \|Y_{fb}^{filtfilt} - Y_{fb}^{filtfilt}\|_2 &= 1.6 \\ \|Y_{fb}^{0} - Y_{fb}^{0}\|_2 &= 1.0 \\ \|\hat{Y}_{fb} - \hat{Y}_{fb}\|_2 &= 8 \cdot 10^{-13} \end{aligned}$$

Thus, there is a significant difference depending on the order of forward and backward filtering for the first two choices of initial conditions. The transient from the filter is clearly visible either in the beginning or the end. The proposed method and the partial fraction method are perfectly symmetric.

To highlight the transients due to unknown initial conditions in the partial fraction method, a very simple zero-phase high-pass filter is used,

$$y(t) = \left|\frac{1}{q+0.8}\right|^2 u(t) = \frac{125/36}{q+1.25} - \frac{20/9}{q+0.8}$$

[1] The result can be reproduced using Matlab's Gaussian distribution with seed 0

[2] Version 4.1 and earlier of Matlab has a bug. With the following definition of the initial state `zi`, the function works as intended: `zi = (eye(nfilt-1) + [aa(2:nfilt)' [-eye(nfilt-2); zeros(1,nfilt-2)]])\ (bb(2:nfilt)-bb(1)*aa(2:nfilt))';`

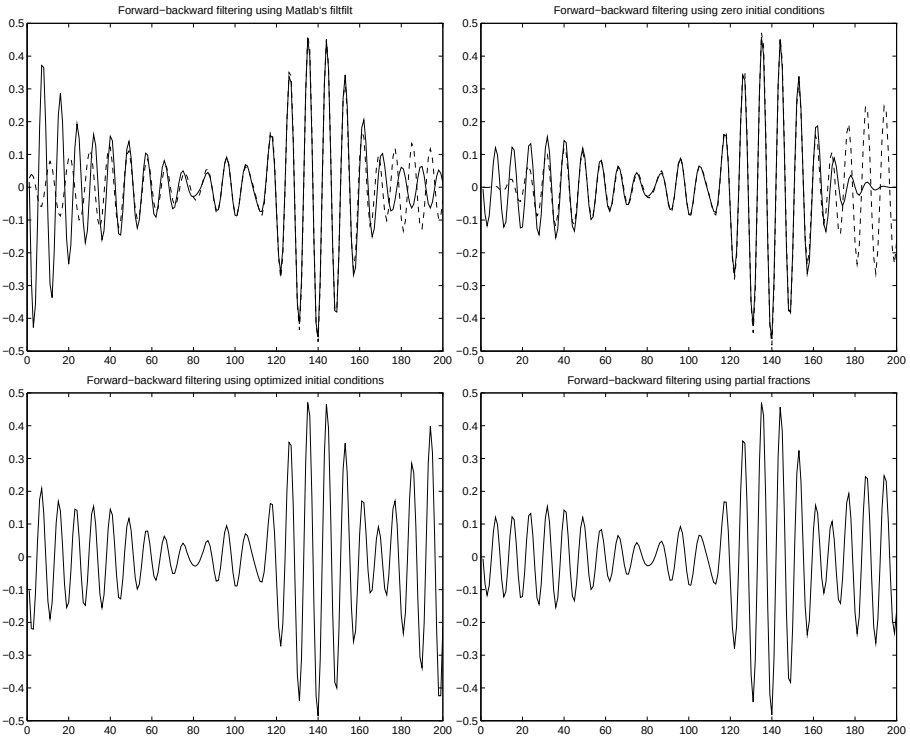

Figure 1: The forward-backward and backward-forward filtered signal using first Matlabs filtfilt, then zero initial condition, optimized initial conditions and finally using partial fractions.

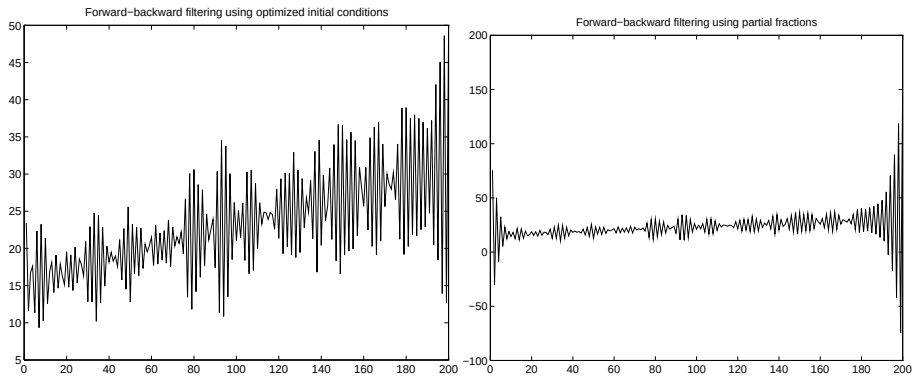

Figure 2: The filtered signal using optimized initial conditions and partial fractions.

and a DC offset of 100 is added to the $u(t)$ used in the previous example. The result is shown in Figure 2. The transients for the partial fraction method are obvious. Of course, some *ad-hoc* rules can be used to determine initial conditions to compensate for the DC offset in this example. The point is, that the proposed method computes the initial conditions for each filter to match the end of the other filter response when the transient hopefully is gone.

6 Conclusion

We have here detailed a method to choose the initial conditions for filters to be applied forwards and backwards to a signal. The objectives were to get the same result independently of the order of filtering and to minimize transients due to offsets and trends in the data. Zero initial conditions or other *ad-hoc* initializations can by examples easily be shown to give undesired transients at the beginning or end of the data sequence depending on the order of application of the forward and backward filters. Transients are also a problem when using partial fractions of the stable and unstable parts. The proposed method gives no transients and is symmetric, which was exemplified by a simulation.

References

[1] T. Kailath. *Lectures on Wiener and Kalman filtering*. Springer-Verlag, Wien, 1981.

[2] MATLAB. *Signal Processing Toolbox User's Guide*. The MathWorks, Inc, 1993.