Online Frequency Adjustment of IIR Filters

Pierre Vacher
ONERA, Department of Systems Control and Flight Dynamics, Toulouse, France

ABSTRACT: Some applications in signal processing require an online adjustment of the frequency specifications of digital filters in order to adapt to the evolution of the environment in which the filters are operating. For instance we might need to modify in real-time the cut-off frequency of a low-pass filter or the centre frequency of a pass-band filter. This paper describes an efficient algorithm to perform this operation for the four types of frequency selective infinite impulse response (IIR) filters. A benchmark evaluation reveals that they can process data with an acceptable extra workload as compared to fixed filters.

1 INTRODUCTION

1.1 General framework of the article

The type of filter adjustment addressed in this paper is depicted in Fig. 1. For low-pass and high-pass filters designated by “one-sided filters”, the adjustment concerns the modification of a characteristic frequency $f$ of the filter frequency response such as its cut-off frequency. For band-stop or band-pass filters called “band filters”, this article deals with the adjustment of the band specifications which are specified by the centre frequency $f$ and the relative bandwidth $b$.

As shown on the figure, the values of the adjusted frequency parameters are supposed to be available at each sampling time. As the frequency characteristics of a digital filter is based on its steady-state response, the filter will implement its designed frequency response only after the transient of its impulse response has died out. Therefore, we have to make the assumption that the variations of the adjusted parameters are slow in comparison with the filter time response.

![Diagram](https://via.placeholder.com/150)

Figure 1: The two types of filter adjustment considered in the paper.

1.2 An illustrative example

This example is extracted from the paper (Vacher and Bucharles 2006) describing a tool for detecting the turbulence that might occur in the course of a flight test. In order to detect efficiently the occurrence of a gust, we must remove from the measurements the contribution of the aircraft response to sine-sweep excitations. This is performed by running an adjusted band-stop filter which attenuates the frequencies in the vicinity of the instantaneous frequency of the excitation.
This operation is depicted by the spectrograms of Fig. 2. The left spectrogram corresponds to the raw measurement. We can notice the occurrence of a gust in the second part of the sweep. The second spectrogram shows that the contribution of the excitation signal can be efficiently attenuated by an adjusted filter.

![Figure 2: Spectrograms of the raw and filtered signal.](image)

1.3 Main objective of the paper

In many instances, the adjusted filter is a part of a real-time application which must cope with the sampling rate of the data. So the main objective of this study is to develop efficient algorithms in order to reduce the computational load of filter adjustment. As some applications require the processing of several signals concurrently, we shall also concentrate on implementations that can efficiently handle many signals in parallel.

In other words, this article tries to provide an answer to the simple question: what is the cost in terms of computational load of the online adjustment of digital filters in comparison with conventional fixed filters?

1.4 Overview of the solution developed

Filter adjustment can be viewed as a partial and online re-design of digital filters. The method developed in this article is equivalent to the classical analog prototyping approach for the design of Infinite Impulse Response (IIR) filters. This method is depicted in Fig. 3.

As indicated in this figure, the adjustment concerns only the last two phases, the analog prototype remaining unchanged. The other two steps of the design are systematic and non-iterative operations and therefore well-suited for an online adjustment. The bilinear transformation is used for the discretization phase.

The algorithms developed in this paper stems from a numerical interpretation and reformulation of the bilinear transformation. This formulation is integrated in the design procedure to produce efficient online processing procedures.

![Figure 3: Analog prototyping approach for the design of digital filters.](image)
1.5 Overview of the solution developed

The next section describes this reformulation of the bilinear transformation. The following one deals with its application to the online adjustment of the four standard frequency selective filters. The last section concerns the implementation and the evaluation on a benchmark of the adjustment algorithms. More details about these methods can be found in the report (Vacher 2008).

2 BILINEAR TRANSFORMATION

Referring to Fig. 1, a simple approach of online adjustment would consist in performing two operations at each sampling time:

- the computation of the filter coefficients for the current values of the adjusted parameters $f_k$ and, possibly, $b_k$;
- the processing of the current signal sample $u_k$ with this updated filter coefficients.

The gist of the reformulation of the bilinear transformation is to perform a direct update of the filter state and output and to skip the explicit update of the filter coefficients. This new approach is based on the interpretation of the bilinear transformation as a trapezoidal method for the numerical integration of linear systems (Stanley 1975). In this article, this interpretation is applied to the state-space expression of the bilinear transformation.

2.1 Interpretation of the bilinear transformation

The bilinear transformation (Oppenheim and Schafer 1989) converts a continuous-time filter $H_d(s)$ into a discrete-time one $H_d(z)$ according to the formula:

$$H_d(z) = H_d\left(\kappa \frac{z - 1}{z + 1}\right)$$

(1)

where $\kappa$ is the warping constant. Two strategies can be used for selecting the value of $\kappa$:

- the so-called “natural” value $\kappa = 2/\Delta t$, where $\Delta t$ is the sampling period. This choice ensures a correspondence of the frequency response of the two filters at the low frequencies.
- the computation of a value $\kappa$ that achieves an exact correspondence of the frequency response between two specific frequencies $\omega_n$ and $\omega_d$ of the analog and digital filters.

The transformation (1) defined in the frequency domain can be interpreted as a numerical integration method in the time domain. Let $u(t)$ designate a continuous signal and let $y(t)$ define the integral of $u(t)$. We suppose that these signals are discretized with a sampling period $\Delta t$. The trapezoidal method provides approximate values $y_k$ of the sampled signal $y(t)$ as a function of the samples $u_k$ of $u(t)$. These values $y_k$ are obtained by the recurrence formula:

$$y_{k+1} = y_k + \Delta t \frac{u_{k+1} + u_k}{2}$$

(2)

The continuous-time integration transfer function $1/s$ is then approximated by the above formula, which corresponds to the discrete-time transfer function $0.5 \Delta t(z+1)/(z-1)$. This is exactly the substitution performed in Eq. (1) when $\kappa = 2/\Delta t$. The formula (2) can also be recast as an approximation of the differential equation $\dot{y}(t) = u(t)$:

$$\frac{y_{k+1} - y_k}{\Delta t} = \frac{u_{k+1} + u_k}{2}$$

(3)

This form enables a better understanding of the phase of the approximated signal. The right-hand side obviously constitutes an approximation of the signal $u(t)$ at the intermediate time $t_k + \Delta t/2$. Similarly, the left-hand side is an approximation of the derivative of $y(t)$ at the middle
time of the sampling interval associated to \( y_k \) and \( y_{k+1} \). Therefore \( y_k \) is the estimate of \( y(t) \) at the sampling time \( t_k \). No time delay is then introduced by the bilinear transformation.

### 2.2 State-space form of the bilinear transformation

When \( \kappa = 2/\Delta t \), the state-space expression of the bilinear transformation is given by the equalities (Vacher 2008)

\[
A_d = \left( I - \frac{\Delta t}{2} A_a \right)^{-1} \left( I + \frac{\Delta t}{2} A_a \right) \quad B_d = \left( I - \frac{\Delta t}{2} A_a \right)^{-1} \Delta t B_a \\
C_d = C_a \left( I - \frac{\Delta t}{2} A_a \right)^{-1} \quad D_d = C_a \left( I - \frac{\Delta t}{2} A_a \right)^{-1} \Delta t B_a + D_a
\]

where the matrices of the analog system are denoted \( A_a; B_a; C_a; D_a \) and those of the transformed discrete-time system \( A_d; B_d; C_d; D_d \). The update state equation can be written as

\[
\frac{X_{k+1} - X_k}{\Delta t} = A_a \frac{X_{k+1} + X_k}{2} + B_a u_k
\]

(5)

By comparison to the continuous filter state equation

\[
\dot{X}(t) = A_a X(t) + B_a u(t)
\]

(6)

this is an obvious numerical integration of this continuous-time differential equation. However, it can be seen that \( X_k \) is in fact an approximation of the state \( X(t) \) of the analog filter at the time \( t_k - \Delta t/2 \) that is to say with a delay of \( -\Delta t/2 \). The right approximation \( X_k \) of \( X(t) \) at the sampling time \( t_k \) by the trapezoidal algorithm would satisfy

\[
\frac{X_{k+1} - X_k}{\Delta t} = A_a \frac{X_{k+1} + X_k}{2} + B_a \frac{u_{k+1} + u_k}{2}
\]

(7)

It can be shown that these two approximations of \( X(t) \) are connected by the very simple relation

\[
X_k = \frac{X_{k+1} + X_k}{2}
\]

(8)

As the bilinear transformation does not introduce any phase delay, this time-lag is made up for in the measurement equation. This can be checked by computing \( X_k \) as a function of \( X_k \) and \( u_k \). By replacing \( X_{k+1} \) in Eq. (8) by its expression in the state equation, we obtain

\[
X_k = \left( I - \frac{\Delta t}{2} A_a \right)^{-1} \left( X_k + \frac{\Delta t}{2} B_a u_k \right)
\]

(9)

The expressions of the matrices \( C_a \) and \( D_a \) in (4) show that the output \( y_k \) of the filter can be computed by

\[
y_k = C_a X_k + D_a u_k
\]

(10)

The output \( y_k \) is thus an approximation of the continuous signal \( y(t) \) with no phase delay. As a conclusion, the state-space formulation of a bilinear transform can be viewed as the numerical integration of the continuous-time filter equation. But the discrete-time state \( X_k \) actually approximates the state \( X(t) \) of the analog filter with a time-lag of \( -\Delta t/2 \).

### 2.3 A direct formulation

The previous interpretation provides us with a method to integrate a continuous-time system \( A_a; B_a; C_a; D_a \). At the sampling time \( t_k \), we can compute the vector \( X_k \) using relation (9). Then
the filter output $y_k$ is readily obtained by Eq. (10). The update of the state vector $X_k$ can be computed with the relation (8).

This formulation is valid for any value of the warping constant $\kappa$. So the bilinear transformation can be reformulated into the following numerical integration method of a continuous-time state-space system.

$$
X_k = \left( I - \frac{A_d}{\kappa} \right) \left( X_k + \frac{B_d}{\kappa} u_k \right)
$$

$$
y_k = C_a X_k + D_a u_k
$$

$$
X_{k+1} = 2 X_k - X_k
$$

(11)

This formula were dubbed the Naussac formulation. In the following section, the quantity $\kappa$ will be used for the online adjustment of the frequency characteristics of the filter.

3 FILTER ADJUSTMENT

In this section, the Naussac formulation is applied to the procedure of Fig. 3 in order to derive an online adjustment procedure. To improve the efficiency of the processing, some of the computations performed in the second phase (filter type transformation) can actually be performed beforehand leading to the concept of “dedicated prototype”. Then the adjustment algorithms is based on the close integration of the remaining operations of this second phase with the Naussac formulation.

3.1 The concept of dedicated prototype

The four filters types transformations that are classically used in the analog prototype design procedure can actually be decomposed into a combination of elementary filter transformations. Some of these transformations are independent of the adjusted parameters. So they can be carried out in an initialization phase prior to the launch of the data processing. This directly saves computations in the real-time processing phase. The filter obtained by applying these preliminary transformations is called the “dedicated prototype”.

It can be shown (see Vacher 2008) that, for lowpass filters and for bandpass filters, the appropriate prototype is nothing mere than the conventional analog lowpass prototype with a design frequency $o_p$ normalized to 1 rad/s. Whereas for highpass filters and stopband filters, the dedicated prototype is a normalized highpass filter. The transformation of the lowpass prototype into this highpass filter leads to a significant alleviation of the computational load.

3.2 On-line adjustment procedures

The other consequence of computing the dedicated prototype is that the study of filter adjustment reduces to two cases:

- One-sided filters

In this case, the bilinear transformation directly applies to the dedicated prototype. The warping “constant” $\kappa$ is tuned at each sampling time to achieve the matching between the normalized frequency of the prototype $o_p = 1$ rad/s and the current desired value $f_c$ of the digital filter characteristic frequency (see Oppenheim and Schafer 1989). By applying the Naussac algorithm defined in Eqs. (11), we obtain the algorithm described in the left column of Table 1. The quantity $\tau_k$ denotes the current value of the inverse of $\kappa$.

- Band filters

The dedicated prototype is transformed by using a lowpass to bandpass filter transformation which has the following state-space expression
The resulting filter is subsequently discretized and the quantity $\kappa$ is used to tune the centre frequency of the final filter to the current value $f_k$. Using matrix manipulations, these two transformations can efficiently be merged (see Vacher 2008). The result appears in the right column of table 1 where $X_{1,k}$ and $X_{2,k}$ denotes the partition of the state vector $X_k$ of the digital filter into two subvectors.

<table>
<thead>
<tr>
<th>Step</th>
<th>One-sided filters</th>
<th>Band Filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Wait for samples $u_k$, $f_k$</td>
<td>Wait for samples $u_k$, $f_k$, $b_k$</td>
</tr>
<tr>
<td>2</td>
<td>Compute $\tau_k$</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$L_k = I - \tau_k A_p$</td>
<td>$L_k = \rho_k I - b_k A_p$ with $\rho_k = \frac{1 + \tau_k^2}{\tau_k}$</td>
</tr>
<tr>
<td>4</td>
<td>$V_k = X_k + \tau_k B_p u_k$</td>
<td>$V_k = \frac{X_{1,k}}{\tau_k} + b_k B_p u_k$</td>
</tr>
<tr>
<td>5</td>
<td>$X_k = L_k \backslash V_k$</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>$y_k = C_p X_k + D_p u_k$</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>$X_{k+1} = 2X_k - X_k$</td>
<td>$X_{1,k+1} = 2X_k - X_{1,k}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$X_{2,k+1} = X_{2,k} - 2\tau_k X_k$</td>
</tr>
<tr>
<td>8</td>
<td>GO TO 1</td>
<td></td>
</tr>
</tbody>
</table>

4 IMPLEMENTATION AND EVALUATION

In table 1 the adjustment procedures are expressed as functions of the state matrices $A_p$, $B_p$, $C_p$, $D_p$ of the dedicated prototype. It is obvious that the computational load highly depends on the appropriate choice of a realization for the prototype filter. This aspect is studied in the first point of this section. The following point concerns the evaluation of these procedures on a benchmark.

4.1 Analysis of prototype realizations

The most time-consuming operation in the adjustment algorithms (table 1) is the linear system inversion of the fifth step which can be recast in the following generic form

$$W = \eta L^{-1} V \quad \text{with} \quad L = \mu I - A_p$$

(13)

The structure of the matrix $A_p$ is, of course, of the utmost importance. The real diagonal form would of course be the most efficient one. However, this implies that the prototype is only composed of real modes. Unfortunately, the prototypes described in the literature (see for instance Parks and Burrus 1987) all include complex modes except for a single real mode when
their order is odd. It can also be shown that the complex diagonal form does not lead to an efficient implementation because of the redundancy between conjugate components.

Several realizations were analyzed for the prototype:

- companion (horizontal and vertical)
- block-diagonal companion (horizontal and vertical)
- block-diagonal modal

In block-diagonal representations, a $2 \times 2$ block on the diagonal of the matrix $A_p$ is associated to each complex modes $\alpha$. This block is a companion matrix for the block-diagonal companion realization. For the block-diagonal modal form, it has the following expression

$$
\begin{pmatrix}
\alpha^R & -\alpha^I \\
\alpha^I & \alpha^R
\end{pmatrix}
\quad \text{with} \quad \alpha = \alpha^R + j \alpha^I
$$

(14)

For each of these representations, a specific algorithm was developed to solve in the most efficient way the system inversion in Eq. (13) (see Vacher 2008). The count of the number of floating-point operation (flops) required for these realizations depends whether the order of the prototype $n_p$ is even or odd. When it is even, all the prototypes modes are complex. The flops count results appear in table 2 where $n_s$ is the number of signals to be processed. They should be compared to the number of operations required for fixed filters which is equal to $n_s \left(4 n_p + 1\right)$ for one-sided filters and $n_s \left(8 n_p + 1\right)$ for band filters.

When the number of signal $n_s$ is great, the performance of the two block-diagonal realizations are identical. As compared to fixed filters, this represents an increase of the computational load that lies in the range $62.5–67\%$ for one-sided filters and $31–35\%$ for band filters. The full companion forms proves to be less effective. When a single signal is processed ($n_s = 1$), the best performance is obtained with the block-diagonal companion representation with an extra workload that ranges between $137\%$ and $178\%$ for one-sided filters and between $68\%$ and $135\%$ for band filters. The highest workload increase is obtained for $n_p = 1$ and $n_s = 1$. In this case, all the realizations are equivalent. One-sided filters require an extra $200\%$ workload and the band filters an extra $190\%$.

<table>
<thead>
<tr>
<th>Table 2: Operation count for the complex pole case</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-sided filters</td>
</tr>
<tr>
<td>--------------------------</td>
</tr>
<tr>
<td>Companion</td>
</tr>
<tr>
<td>Block Comp.</td>
</tr>
<tr>
<td>Block Modal</td>
</tr>
</tbody>
</table>

Based on the operation count, the block-diagonal companion realization provides the best performance. The additional workload as compared to fixed filters is significant but the magnitude order of the computational load remains the same.

4.2 Benchmark evaluation

Two adjustment methods are evaluated in this subsection:

- the block-diagonal companion form
- the block-diagonal modal form

In order to evaluate the extra cost of the adjustment of filters, these algorithms are confronted with their fixed filter counterpart. Of course, this comparison is made from the sole point of view of the computational load. All these filters were implemented in FORTRAN.

The evaluation of the computational load is simply accomplished by running several time each algorithm on a sequence of data and by measuring their execution time. These results are given in table 3. They correspond to the execution durations for a set of ten million samples. At a 100 Hz sampling rate, this represents a duration of nearly 28 hours. This benchmark was executed on a SunBlade-1500 workstation with 1 GHz CPU clock rate.
Two values for the prototype order \( n_p \) were evaluated for both one-sided and band filters. When several signals are processed together, a part of the computation of the adjustment algorithms can be performed once for all signals thereby reducing the duration of the processing per each signal. This latter is indicated in parenthesis in table 3 for a value \( n_z = 100 \).

For fixed filters, the benchmark confirms the theory since the block-diagonal companion form performs better than the block-diagonal modal form which requires more computations. On the contrary, the block-diagonal modal representation is faster for adjusted filters in spite of the theoretical evaluation which was in favour of the block-diagonal companion realization. The adjusted filters are indeed slower than their fixed counterparts. But their efficiency greatly improves when processing a great number of signals together.

The orders of the one-sided filter with \( n_p = 6 \) and of the band filter with \( n_p = 3 \) are identical. We can check on table 3 that the experimental results for fixed filters are very similar for these two cases. Concerning the adjusted filters, the band filters with \( n_p = 3 \) are notably faster than their one-sided counterparts with \( n_p = 6 \). This comes from the close integration of the lowpass-to-bandpass transformation with the Naussac formulation.

<table>
<thead>
<tr>
<th>Table 3: Benchmark execution times per signal (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-Sided Filters</td>
</tr>
<tr>
<td>( n_p = 3 )</td>
</tr>
<tr>
<td>Fixed block-diag. companion</td>
</tr>
<tr>
<td>Fixed block-diag. modal</td>
</tr>
<tr>
<td>Adjusted block-diag. companion</td>
</tr>
<tr>
<td>Adjusted block-diag. modal</td>
</tr>
</tbody>
</table>

5 CONCLUSION

The benchmark evaluation revealed that when a single signal is processed, adjusted filters are 3–4 times slower than fixed filters. When many signals are processed, the extra workload is quite moderate since it lies between 25 % and 50 %. It must nevertheless be underlined that both fixed and adjusted filters are extremely fast. They are compatible with most real-time applications. The block-diagonal modal representation also proved to be the most appropriate implementation structure for the online adjustment of digital filters.

REFERENCES