Local Polynomial Filters

3.1 Introduction

We mentioned in Sec. 2.4 that there are limits to the applicability of the plain FIR averager filter—in order to achieve a high degree of noise reduction, its length $N$ may be required to be so large that the filter’s passband becomes smaller than the signal bandwidth, causing the removal of useful high frequencies from the desired signal.

In other words, in its attempt to smooth out the noise $v_n$, the filter begins to smooth out the desired signal $x_n$ to an unacceptable degree. For example, if $x_n$ contains some short-duration peaks, corresponding to the higher frequencies present in $x_n$, and the filter’s length $N$ is longer than the duration of the peaks, the filter will tend to smooth the peaks too much, broadening them and reducing their height.

Local polynomial smoothing filters [36–99] are generalizations of the FIR averager filter that can preserve better the higher frequency content of the desired signal, at the expense of not removing as much noise as the averager. They can be characterized in three equivalent ways:

1. They are the optimal lowpass filters that minimize the NRR, subject to additional constraints than the DC unity-gain condition (2.4.1)—the constraints being equivalent to the requirement that polynomial input signals go through the filter unchanged.
2. They are the optimal filters that minimize the NRR whose frequency response $H(e^{\omega})$ satisfies certain flatness constraints at DC.
3. They are the filters that optimally fit, in a least-squares sense, a set of data points to polynomials of different degrees.

Local polynomial smoothing (LPSM) filters have a long history and have been rediscovered repeatedly in different contexts. They were originally derived in 1866 by the Italian astronomer Schiaparelli [36] who formulated the problem as the minimization of the NRR subject to polynomial-preserving constraints and derived the filters in complete generality, discussing also the case of even-length filters. They were rederived in 1871 by De Forest [65] who generalized them further to include the case of “minimum-roughness” or minimum-$R_s$ filters. Subsequently, they were rediscovered many times and used extensively in actuarial applications, for example, by Gram, Hardy, Sheppard, Henderson, and others. See Refs. [68–75] for the development and history of these filters. In the actuarial context, smoothing is referred to as the process of “graduation.” They were revived again in the 1960s by Savitzky and Golay [42] and have been applied widely in chemistry and spectroscopy [42–53] known in that context as Savitzky-Golay filters. They, and their minimum-$R_s$ versions [65–99] known typically as Henderson filters, are used routinely for trend extraction in financial, business, and census applications.

Some recent incarnations also include predictive FIR interpolation, differentiation, fractional-delay, and maximally-flat filters [152–187], and applications to the representation of speech and images in terms of orthogonal-polynomial moments [137–150].

[104] derived in 1864 the discrete Chebyshev orthogonal polynomials, also known as Gram polynomials, which provide convenient and computationally efficient bases for the solution of the least-squares optimization problem. Several applications and reviews of the discrete Chebyshev orthogonal polynomials may be found in [104–151]. The minimum-$R_s$ Henderson filters also admit similar efficient representations in terms of the Hahn orthogonal polynomials, a special case of which are the discrete Chebyshev polynomials. We discuss Henderson filters in Sec. 4.2 and orthogonal polynomial bases in Sec. 4.3.

3.2 Local Polynomial Fitting

We begin with the least-squares polynomial fitting approach. We assume that the signal model for the observations is:

$$y_n = x_n + v_n$$

where $v_n$ is white noise and $x_n$ is a smooth signal to be estimated. Fig. 3.2.1 shows five noisy signal samples [$y_{−2}, y_{−1}, y_0, y_1, y_2$] positioned symmetrically about the origin. Later on, we will shift them to an arbitrary position along the time axis. Polynomial smoothing of the five samples is equivalent to replacing them by the values that lie on smooth polynomial curves drawn between the noisy samples. In Fig. 3.2.1, we consider fitting the five data to a constant signal, a linear signal, and a quadratic signal.

The corresponding smoothed values are given by the 0th, 1st, and 2nd degree polynomials defined for $m = −2, −1, 0, 1, 2$:

\[
\hat{y}_m = c_0 \quad \text{(constant)}
\]

\[
\hat{y}_m = c_0 + c_1 m \quad \text{(linear)}
\]

\[
\hat{y}_m = c_0 + c_1 m + c_2 m^2 \quad \text{(quadratic)}
\]

For each choice of the polynomial order, the coefficients $c_i$ must be determined optimally such that the corresponding polynomial curve best fits the given data. This can be accomplished by a least-squares fit, which chooses the $c_i$ that minimize the total mean-square error. For example, in the quadratic case, we have the performance index:

$$J = \sum_{m=-2}^{2} e_m^2 = \sum_{m=-2}^{2} (y_m - (c_0 + c_1 m + c_2 m^2))^2 = \min$$

\[\text{not to be confused with the ordinary Chebyshev polynomials.}\]
3. Local Polynomial Filters

3.2. Local Polynomial Fitting

Therefore,

\[ \hat{y} = c_0 s_0 + c_1 s_1 + c_2 s_2 = [s_0, s_1, s_2] \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} \equiv Sc \quad (3.2.4) \]

The 5×3 basis matrix S has as columns the three basis vectors \( s_0, s_1, s_2 \). It is given explicitly as follows:

\[
S = [s_0, s_1, s_2] = \begin{bmatrix}
1 & -2 & 4 \\
1 & -1 & 1 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
1 & 2 & 4
\end{bmatrix} \quad (3.2.5)
\]

Writing \( e = y - \hat{y} = y - Sc \), we can express the performance index (3.2.2) as the dot product:

\[
J = e^T e = (y - Sc)^T (y - Sc) = \min \quad (3.2.6)
\]

To minimize this expression with respect to \( c \), we set the gradient \( \partial J / \partial c \) to zero:

\[
\frac{\partial J}{\partial c} = -2St e = -2St (y - Sc) = -2 (St y - St Sc) = 0 \quad (3.2.7)
\]

Therefore, the minimization condition gives the so-called orthogonality equations and the equivalent normal equations:

\[
\frac{\partial J}{\partial c} = 0 \iff St e = 0 \iff St Sc = St y \quad (3.2.8)
\]

with optimal solution:

\[
c = (St S)^{-1} St y \equiv G^T y \quad (3.2.9)
\]

where we defined the 5×3 matrix \( G \) by

\[
G = S (St S)^{-1} \quad (3.2.10)
\]

We note that the solution (3.2.9) is none other than the unique least-squares solution of the full-rank overdetermined linear system \( Sc = y \), as given for example by Eq. (15.4.10), \( c = S^+ y \), where \( S^+ = (St S)^{-1} St \) is the corresponding pseudoinverse. Inserting the optimal coefficients \( c \) into Eq. (3.2.4), we find the smoothed values:

\[
\hat{y} = Sc = SG^T y = S (St S)^{-1} St y = By \quad (3.2.11)
\]

where we defined the 5×5 matrix \( B \) by

\[
B = B^T = SG^T = GS^T = S (St S)^{-1} St \quad (3.2.12)
\]

The symmetric 3×3 matrix \( F = St S \), which appears in the expressions for \( G \) and \( B \), has matrix elements that are the dot products of the basis vectors, that is, the \( ij \)th matrix element is \( F_{ij} = (St S)_{ij} = s_i^T s_j \). Indeed, using Eq. (3.2.5), we find:

\[
F = St S = \begin{bmatrix}
s_0^T \\
s_1^T \\
s_2^T
\end{bmatrix} [s_0, s_1, s_2] = \begin{bmatrix}
s_0^T s_0 & s_0^T s_1 & s_0^T s_2 \\
s_1^T s_0 & s_1^T s_1 & s_1^T s_2 \\
s_2^T s_0 & s_2^T s_1 & s_2^T s_2
\end{bmatrix} \quad (3.2.13)
\]

\footnote{Although \( B \) is symmetric, we prefer to write \( \hat{y} = By \), which generalizes to the non-symmetric case of minimum-roughness filters of Sec. 4.2.}
3. Local Polynomial Filters

Using Eq. (3.2.5), we calculate $F$ and its inverse $F^{-1}$:

$$ F = \begin{bmatrix} 5 & 0 & 10 \\ 0 & 10 & 0 \\ 10 & 0 & 34 \end{bmatrix}, \quad F^{-1} = \frac{1}{35} \begin{bmatrix} 17 & 0 & -5 \\ 0 & 3.5 & 0 \\ -5 & 0 & 2.5 \end{bmatrix} \quad (3.2.14) $$

Then, we calculate the $5\times3$ matrix $G = S(S^T S)^{-1} = SF^{-1}$:

$$ G = \frac{1}{35} \begin{bmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix} \begin{bmatrix} 17 & 0 & -5 \\ 0 & 3.5 & 0 \\ -5 & 0 & 2.5 \end{bmatrix} \quad \text{or,} \\
G = \frac{1}{35} \begin{bmatrix} -3 & -7 & 5 \\ 0 & -3.5 & -2.5 \\ 17 & 0 & -5 \\ 12 & 3.5 & -2.5 \\ -3 & 7 & 5 \end{bmatrix} = [g_0, g_1, g_2] \quad (3.2.15) $$

As we see below, the three columns of $G$ have useful interpretations as differentiation filters. Next, using Eq. (3.2.12), we calculate the $5\times5$ matrix $B$:

$$ B = GS^T = \frac{1}{35} \begin{bmatrix} -3 & -7 & 5 \\ 12 & -3.5 & -2.5 \\ -3 & 7 & 5 \\ 31 & 9 & -3 \\ 9 & 13 & 12 \\ -3 & 12 & 17 \\ -5 & 6 & 12 \\ 3 & -5 & -3 \\ 3 & 5 & 9 \end{bmatrix} = [b_{-2}, b_{-1}, b_0, b_1, b_2] \quad (3.2.16) $$

Because $B$ is symmetric, its rows are the same as its columns. Thus, we can write it either in column-wise or row-wise form:

$$ B = [b_{-2}, b_{-1}, b_0, b_1, b_2] = [b_{-2}^T, b_{-1}^T, b_0^T, b_1^T, b_2^T] = B^T $$

The five columns or rows of $B$ are the LPSM filters of length 5 and polynomial order 2. The corresponding smoothed values $\tilde{y}$ can be expressed component-wise in terms of these filters, as follows:

$$ \tilde{y}_{-2} = \hat{y}_{-2} = b_{-2}^T y, \quad \tilde{y}_{-1} = \hat{y}_{-1} = b_{-1}^T y, \quad \tilde{y}_0 = \hat{y}_0 = b_0^T y, \quad \tilde{y}_1 = \hat{y}_1 = b_1^T y, \quad \tilde{y}_2 = \hat{y}_2 = b_2^T y $$

Thus, the $m$th filter $b_m$ dotted into the data vector $y$ generates the $m$th smoothed data sample. In a similar fashion, we can express the polynomial coefficients $c_i$ as dot products. Using the solution Eq. (3.2.9), we have

$$ \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = c = G^T y = [g_0^T, g_1^T, g_2^T] y = [g_0^T y, g_1^T y, g_2^T y] $$

or, explicitly,

$$ \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \frac{1}{35} \begin{bmatrix} -3 & 12 & 17 \\ -7 & -3.5 & 0 \\ 5 & -2.5 & -5 \end{bmatrix} \begin{bmatrix} y_{-2} \\ y_{-1} \\ y_0 \end{bmatrix} \quad (3.2.19) $$

Thus, the coefficients $c_i$ can be expressed as the dot products of the columns of $G$ with the data vector $y$:

$$ c_i = g_i^T y, \quad i = 0, 1, 2 $$

(3.2.20)

Of the five columns of $B$, the middle one, $b_0$, is the most important because it smooths the value $y_0$, which is symmetrically placed with respect to the other samples in $y$, as shown in Fig. 3.2.1.

In smoothing a long block of data, the filter $b_0$ is used during the steady-state period, whereas the other columns of $B$ are used only during the input-on and input-off transients. We will refer to $b_0$ and the other columns of $B$ as the steady-state and transient LPSM filters.

Setting $m = 0$ into Eq. (3.2.1), we note that the middle smoothed value $\tilde{y}_0$ is equal to the polynomial coefficient $c_0$. Using Eqs. (3.2.17) and (3.2.20), we find: $\tilde{y}_0 = c_0 = b_0^T y = g_0^T y$ (the middle column of $B$ and the first column of $G$ are always the same, $b_0 = g_0$).
3. Local Polynomial Filters

To express (3.2.18) as a true filtering operation acting on an input sequence \( y_n \), we shift the group of five samples to be centered around the \( n \)th time instant, that is, we make the substitution:

\[
[y_{n-2}, y_{n-1}, y_n, y_{n+1}, y_{n+2}] \rightarrow [y_{n-2}, y_{n-1}, y_n, y_{n+1}, y_{n+2}]
\]

The corresponding five smoothed values will be then:

\[
\begin{bmatrix}
\hat{y}_{n-2} \\
\hat{y}_{n-1} \\
\hat{y}_{n} \\
\hat{y}_{n+1} \\
\hat{y}_{n+2}
\end{bmatrix} = \frac{1}{35}
\begin{bmatrix}
31 & 9 & -3 & -5 & 3 \\
9 & 13 & 12 & 6 & -5 \\
-3 & 12 & 17 & 12 & -3 \\
-5 & 6 & 12 & 13 & 9 \\
3 & -5 & -3 & 9 & 31
\end{bmatrix}
\begin{bmatrix}
y_{n-2} \\
y_{n-1} \\
y_{n} \\
y_{n+1} \\
y_{n+2}
\end{bmatrix}
\] (3.2.21)

In particular, the middle sample \( y_n \) is smoothed by the filter \( b_0 \):

\[
\hat{x}_n = \frac{1}{35}(-3y_{n-2} + 12y_{n-1} + 17y_n + 12y_{n+1} - 3y_{n+2})
\] (3.2.22)

where, in accordance with our assumed model of noisy observations \( y_n = x_n + v_n \), we denoted \( y_n \) by \( x_n \), i.e., the estimated value of \( x_n \).

The other estimated values \( \{y_{m+n}, m = \pm 1, \pm 2\} \), are not used for smoothing, except, as we see later, at the beginning and end of the signal block \( y_n \). They may be used, however, for prediction and interpolation.

The filter (3.2.22) corresponds to fitting every group of five samples \( \{y_{n-2}, y_{n-1}, y_n, y_{n+1}, y_{n+2}\} \) to a quadratic polynomial and replacing the middle sample \( y_n \) by its smoothed value \( \hat{x}_n \). It is a lowpass filter and is normalized to unity gain at DC, because its coefficients add up to one.

Its NRR is the sum of the squared filter coefficients. It can be proved in general that the NRR of any steady-state filter \( b_0 \) is equal to the middle value of its impulse response, that is, the coefficient \( b_0(0) \). Therefore,

\[
\mathcal{R} = b_0^T b_0 = \sum_{m=-2}^{2} b_0(m)^2 = b_0(0)^2 = \frac{17}{35} = \frac{17/7}{5} = \frac{2.43}{5} = 0.49
\]

By comparison, the length-5 FIR averager operating on the same five samples is:

\[
\hat{x}_n = \frac{1}{5}(y_{n-2} + y_{n-1} + y_n + y_{n+1} + y_{n+2})
\] (3.2.23)

with \( \mathcal{R} = 1/N = 1/5 \). Thus, the length-5 quadratic-polynomial filter performs 2.43 times worse in reducing noise than the FIR averager. However, the higher-order polynomial filters have other advantages to be discussed later.

We saw that the coefficient \( c_0 \) represents the smoothed value of \( y_0 \) at \( m = 0 \). Similarly, the coefficient \( c_1 \) represents the slope, the derivative, of \( y_0 \) at \( m = 0 \). Indeed, we have from Eq. (3.2.1) by differentiating and setting \( m = 0 \):

\[
\hat{y}_0 = \left. \frac{dy_0}{dm} \right|_0 = c_1, \quad \hat{y}'_0 = \left. \frac{d^2y_0}{dm^2} \right|_0 = 2c_2
\]

3.2. Local Polynomial Fitting

Thus, \( c_1 \) and \( 2c_2 \) represent the polynomial estimates of the first and second derivatives at \( m = 0 \). Using Eq. (3.2.20) we can express them in terms of the second and third columns of the matrix \( G \):

\[
\hat{y}_0 = c_1 = b_1^T y \quad \hat{y}'_0 = 2c_2 = 2b_2^T y
\] (3.2.24)

Shifting these to the \( n \)th time sample, and denoting them by \( \hat{x}_n \) and \( \hat{\hat{x}}_n \), we find the length-5 local polynomial filters for estimating the first and second derivatives of \( x_n \):

\[
\hat{x}_n = \frac{1}{35}(-7y_{n-2} - 3.5y_{n-1} + 3.5y_{n+1} + 7y_{n+2})
\]

\[
\hat{\hat{x}}_n = \frac{2}{35}(5y_{n-2} - 2.5y_{n-1} - 5y_n - 2.5y_{n+1} + 5y_{n+2})
\] (3.2.25)

The above designs can be generalized in a straightforward manner to an arbitrary degree \( d \) of the fitted polynomial and to an arbitrary length \( N \) of the data vector \( y \). We require only that \( d \leq N - 1 \), a restriction to be clarified later. Assuming that \( N \) is odd, say, \( N = 2M + 1 \), the five-dimensional data vector \( y = [y_{-2}, y_{-1}, y_0, y_1, y_2]^T \) is replaced by an \( N \)-dimensional one, having \( M \) points on either side of \( y_0 \):

\[
y = [y_{-M}, \ldots, y_{-1}, y_0, y_1, \ldots, y_M]^T
\] (3.2.26)

The \( N \) data samples in \( y \) are then fitted by a polynomial of degree \( d \):

\[
\hat{y}_m = c_0 + c_1m + \cdots + c_dm^d, \quad -M \leq m \leq M
\] (3.2.27)

In this case, there are \( d+1 \) polynomial basis vectors \( s_i, i = 0, 1, \ldots, d \), defined to have components:

\[
s_i(m) = m^i, \quad -M \leq m \leq M
\] (3.2.28)

The corresponding \( N \times (d+1) \) basis matrix \( S \) is defined to have the \( s_i \) as columns:

\[
S = [s_0, s_1, \ldots, s_d]
\] (3.2.29)

The smoothed values (3.2.27) can be written in the vector form:

\[
\hat{y} = \sum_{i=0}^{d} c_i s_i = [s_0, s_1, \ldots, s_d]
\]

\[
\begin{bmatrix}
c_0 \\
c_1 \\
\vdots \\
c_d
\end{bmatrix}
= Sc
\] (3.2.30)

The design steps for the LPSM filters can be summarized then as follows:

\[
F = S^T S \quad F_{ij} = s_i^T s_j, \quad i, j = 0, 1, \ldots, d
\]

\[
G = SF^{-1} = [g_0, g_1, \ldots, g_d]
\]

\[
B = GS^T = SF^{-1} S^T = [b_{-M}, \ldots, b_0, \ldots, b_M]
\] (3.2.31)
3. Local Polynomial Filters

The corresponding coefficient vector \( c \) and smoothed data vector \( \hat{y} \) will be:

\[
c = G^T y \quad \Leftrightarrow \quad c_i = g_i^T y, \quad i = 0, 1, \ldots, d
\]

\[
\hat{y} = B^T y \quad \Leftrightarrow \quad \hat{y}_m = b_m^T y, \quad -M \leq m \leq M
\]  

(3.2.32)

The middle smoothed value \( \hat{y}_0 \) is given in terms of the middle LPSM filter \( b_0 \):

\[
\hat{y}_0 = b_0^T y = \sum_{k=-M}^{M} b_0(k) y_k
\]

The \( N \)-dimensional vector \( y = [y_{-M}, \ldots, y_{-1}, y_{0}, y_{1}, \ldots, y_{M}]^T \) can be shifted to the \( n \)th time instant by the replacement:

\[
[y_{-M}, \ldots, y_{-1}, y_{0}, y_{1}, \ldots, y_{M}] \rightarrow [y_{-M-n}, \ldots, y_{-1-n}, y_{0-n}, y_{1-n}, \ldots, y_{M-n}]
\]

The resulting length-\( N \), order-\( d \), LPSM filter for smoothing a noisy sequence \( y_n \) will be, in its steady-state form (denoting again \( \hat{s}_n = \hat{y}_n \)):

\[
\hat{s}_n = \hat{y}_n = \sum_{k=-M}^{M} b_0(k) y_{n+k} = \sum_{k=-M}^{M} b_0(-k) y_{n-k}
\]  

(3.2.33)

The second equation expresses the output in convolutional form.\(^1\) Because the filter \( b_0 \) is symmetric about its middle, we can replace \( b_0(-k) = b_0(k) \). The non-central estimated values are obtained from the \( b_m \) filters:

\[
\hat{y}_{n+m} = \sum_{k=-M}^{M} b_m(k) y_{n+k} = \sum_{k=-M}^{M} b_m^T(k) y_{n-k}, \quad -M \leq m \leq M
\]  

(3.2.34)

These filters satisfy the symmetry property \( b_m^T(k) = b_m(-k) = b_{-m}(k) \) and can be used for prediction, as we discuss later.

The \( d+1 \) columns of the \( N \times (d+1) \)-dimensional matrix \( G \) give the LPSM differentiation filters, for derivatives of orders \( i = 0, 1, \ldots, d \). It follows by differentiating Eq. (3.2.27) \( i \) times and setting \( m = 0 \):

\[
\hat{y}^{(i)}_0 = \left. \frac{d^i \hat{y}_m}{dm^i} \right|_{m=0} = b_i c_i = b_i g_i^T y
\]

Shifting these to time \( n \), gives the differentiation convolutional filtering equations:

\[
\hat{x}^{(i)}_n = b_i \sum_{m=-M}^{M} g_i^T(m) y_{n-m}, \quad i = 0, 1, \ldots, d
\]  

(3.2.35)

where, \( g_i^T(m) = g_i(-m) \) and as in Eq. (3.2.33), we reversed the order of writing the terms, but here the filters \( g_i \) are not necessarily symmetric (actually, they are symmetric for even \( i \), and antisymmetric for odd \( i \)).

\(^1\) We use the notation \( b^T \) to denote the reverse of a double-sided filter \( b \), that is, \( b^T(k) = b(-k) \).

3.2. Local Polynomial Fitting

Example 3.2.1: We construct the length-5 LPSM filters for the cases \( d = 0 \) and \( d = 1 \). For \( d = 0 \), corresponding to the constant \( \hat{y}_m = c_0 \) in Eq. (3.2.1), there is only one basis vector \( s_0 \) defined in Eq. (3.2.3). The basis matrix \( S = [s_0] \) will have just one column, and the matrix \( F \) will be the scalar

\[
F = S^T S = s_0^T s_0 = [1, 1, 1, 1, 1] \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = 5
\]

The matrix \( G \) will then be

\[
G = SF^{-1} = \frac{1}{5} s_0 = \frac{1}{5} [1, 1, 1, 1, 1]^T
\]

resulting in the LPSM matrix \( B \):

\[
B = GS^T = \frac{1}{5} s_0 s_0^T = \frac{1}{5} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}
\]

Thus, the steady-state LPSM filter is the length-5 averager:

\[
b_0 = \frac{1}{5} [1, 1, 1, 1, 1]^T
\]

For the case \( d = 1 \), corresponding to the linear fit \( \hat{y}_m = c_0 + c_1 m \), we have the two basis vectors \( s_0 \) and \( s_1 \), given in Eq. (3.2.3). We calculate the matrices \( S, F \), and \( F^{-1} \):

\[
S = [s_0, s_1] = \begin{bmatrix} 1 & -2 \\ 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}, \quad F = S^T S = \begin{bmatrix} 5 & 0 \\ 0 & 10 \end{bmatrix}, \quad F^{-1} = \frac{1}{5} \begin{bmatrix} 1 & 0 \\ 0 & 0.5 \end{bmatrix}
\]

This gives for \( G \) and \( B \):

\[
G = SF^{-1} = \frac{1}{5} \begin{bmatrix} 1 & -1 \\ 1 & -0.5 \\ 1 & 0 \\ 1 & 0.5 \\ 1 & 1 \end{bmatrix}, \quad B = GS^T = \frac{1}{5} \begin{bmatrix} 3 & 2 & 1 & 0 & -1 \\ 2 & 1.5 & 1 & 0.5 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0.5 & 1.5 & 2 & 0 \\ -1 & 0 & 1 & 2 & 3 \end{bmatrix}
\]

Thus, the steady-state LPSM filter \( b_0 \) is still equal to the length-5 FIR averager. It is a general property of LPSM filters, that the filter \( b_0 \) is the same for successive polynomial orders, that is, for \( d = 0, 1, d = 2, 3, d = 4, 5 \), and so on. However, the transient LPSM filters are different. □
3. Local Polynomial Filters

**Example 3.2.2:** Here, we construct the LPSM filters of length \( N = 5 \) and order \( d = 3 \). The smoothed estimates are given by the cubic polynomial:

\[
\hat{y}_m = c_0 + c_1 m + c_2 m^2 + c_3 m^3
\]

There is an additional basis vector \( s_3 \) with components \( s_3(m) = m^4 \). Therefore, the basis matrix \( S \) is:

\[
S = \{s_0, s_1, s_2, s_3\} = \begin{bmatrix}
1 & -2 & 4 & -1 \\
1 & -1 & 1 & -1 \\
1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 \\
1 & 2 & 4 & 8
\end{bmatrix}
\]

Because of the checkerboard pattern of this matrix, its inverse can be obtained from the inverses of the two \( 2 \times 2 \) interlaced submatrices:

\[
\begin{bmatrix}
5 & 10 \\
10 & 34
\end{bmatrix}^{-1} = \frac{1}{70} \begin{bmatrix}
34 & -10 \\
-10 & 5
\end{bmatrix}, \quad \begin{bmatrix}
10 & 34 \\
34 & 130
\end{bmatrix}^{-1} = \frac{1}{144} \begin{bmatrix}
130 & -34 \\
-34 & 10
\end{bmatrix}
\]

Interlacing these inverses, we obtain:

\[
F^{-1} = \begin{bmatrix}
34/70 & 0 & -10/70 & 0 \\
0 & 130/144 & 0 & -34/144 \\
-10/70 & 0 & 5/70 & 0 \\
0 & -34/144 & 0 & 10/144
\end{bmatrix}
\]

Then, we compute the derivative filter matrix \( G \):

\[
G = SF^{-1} = \frac{1}{35} \begin{bmatrix}
-3 & 35/12 & 5 & -35/12 \\
12 & -70/3 & -2.5 & 35/6 \\
17 & 0 & -5 & 0 \\
12 & 70/3 & -2.5 & -35/6 \\
-3 & -35/12 & 5 & 35/12
\end{bmatrix}
\]

and the LPSM matrix \( B \):

\[
B = SG^T = \frac{1}{35} \begin{bmatrix}
34.5 & 2 & -3 & -0.5 \\
2 & 27 & 12 & -8 \\
2 & -8 & 12 & 27 \\
-0.5 & 2 & -3 & 2 & 34.5
\end{bmatrix}
\]

As mentioned above, the steady-state LPSM filter \( b_0 \) is the same as that of case \( d = 2 \). But, the transient and differentiation filters are different. \( \Box \)

### 3.3 Exact Design Equations

In practice, the most common values of \( d \) are 0, 1, 2, 3, 4. For these \( d \)s and arbitrary filter lengths \( N \), the LPSM matrix \( B \) can be constructed in closed form; see references [36-99], as well as the extensive tables in [54]. Denoting the inverse of the \((d+1) \times (d+1)\) matrix \( F = S^T S \) by \( \Phi = F^{-1} \), we can write

\[
B = SF^{-1}S^T = S\Phi S^T = \sum_{i=0}^{d} \sum_{j=0}^{d} s_i s_j^T \Phi_{ij}
\]

which gives for the \( mk \)th matrix element

\[
B_{mk} = \sum_{i=0}^{d} \sum_{j=0}^{d} s_i(m) s_j(k) \Phi_{ij} = \sum_{i=0}^{d} \sum_{j=0}^{d} m^i k^j \Phi_{ij}, \quad -M \leq m, k \leq M
\]

Because of symmetry, \( B_{mk} = B_{km} \), these matrix elements represent the \( k \)th component of the LPSM filter \( b_m \) or the \( m \)th component of the filter \( b_k \), that is,

\[
B_{mk} = B_{km} = b_m(k) = b_k(m) = \sum_{i=0}^{d} \sum_{j=0}^{d} m^i k^j \Phi_{ij}
\]

The matrix \( \Phi \) can be determined easily for the cases \( 0 \leq d \leq 4 \). The matrix \( F \) is a Hankel matrix, that is, having the same entries along each antidiagonal line. Therefore, its matrix elements \( F_{ij} \) depend only on the sum \( i + j \) of the indices. To see this, we write \( F_{ij} \) as the inner product:

\[
F_{ij} = (S^T S)_{ij} = s_i^T s_j = \sum_{m=-M}^{M} s_i(m) s_j(m) = \sum_{m=-M}^{M} m^{i+j}, \quad \text{or,}
\]

\[
F_{ij} = \sum_{m=-M}^{M} m^{i+j} \equiv F_{i+j}, \quad 0 \leq i, j \leq d
\]

Note that because of the symmetric limits of summation, \( F_{i+j} \) will be zero whenever \( i + j \) is odd. This leads to the checkerboard pattern of alternating zeros in \( F \) that we saw in the above examples. Also, because \( d \leq 4 \), the only values of \( i + j \) that we need are: \( i + j = 0, 2, 4, 6, 8 \). For those, the summations over \( m \) can be done in closed form:

\[
F_0 = \sum_{m=-M}^{M} m^0 = N = 2M + 1
\]

\[
F_2 = \sum_{m=-M}^{M} m^2 = \frac{1}{3} M(M+1)(2M+1)
\]

\[
F_4 = \sum_{m=-M}^{M} m^4 = \frac{1}{5} (3M^2 + 3M - 1) F_2
\]

\[
F_6 = \sum_{m=-M}^{M} m^6 = \frac{1}{7} (3M^4 + 6M^2 + 5M - 1) F_2
\]

\[
F_8 = \sum_{m=-M}^{M} m^8 = \frac{1}{15} (5M^6 + 15M^4 + 15M^2 + 9M - 3) F_2
\]
We can express $F$ in terms of these definitions for various values of $d$. For example, for $d = 0, 1, 2, 3$, the $F$ matrices are:

$$
\begin{bmatrix}
F_0, & \begin{bmatrix}
 F_0 & 0 & F_2 \\
 0 & F_2 & 0 \\
 F_2 & 0 & F_4 \\
 F_0 & 0 & F_4 
\end{bmatrix}
\end{bmatrix},
\begin{bmatrix}
 F_0 & 0 & F_2 \\
 0 & F_2 & 0 \\
 F_2 & 0 & F_4 \\
 F_0 & 0 & F_4 
\end{bmatrix}
\begin{bmatrix}
 F_0 & 0 & F_2 \\
 0 & F_2 & 0 \\
 F_2 & 0 & F_4 \\
 F_0 & 0 & F_4 
\end{bmatrix}
$$

The corresponding inverse matrices $\Phi = F^{-1}$ are obtained by interlacing the inverses of the checkerboard submatrices, as in Example 3.2.2. For $d = 0, 1, 2$, we have for $\Phi$:

$$
\begin{bmatrix}
1/F_0, & \begin{bmatrix}
 1/F_0 & 0 & -F_2/F_4 \\
 0 & 1/F_2 & 0 \\
 -F_2/F_4 & 0 & F_0/F_4 
\end{bmatrix}
\end{bmatrix},
\begin{bmatrix}
 1/F_0 & 0 & -F_2/F_4 \\
 0 & 1/F_2 & 0 \\
 -F_2/F_4 & 0 & F_0/F_4 
\end{bmatrix}
\begin{bmatrix}
 1/F_0 & 0 & -F_2/F_4 \\
 0 & 1/F_2 & 0 \\
 -F_2/F_4 & 0 & F_0/F_4 
\end{bmatrix}
$$

and for $d = 3$:

$$
\Phi = F^{-1} = \begin{bmatrix}
 F_4/D_4 & 0 & -F_2/D_4 & 0 \\
 0 & F_6/D_8 & 0 & -F_4/D_8 \\
 -F_2/D_4 & 0 & F_0/D_4 & 0 \\
 0 & -F_4/D_8 & 0 & F_2/D_8 
\end{bmatrix}
$$

where the $D_4$ and $D_8$ are determinants of the interlaced submatrices:

$$
D_4 = F_0 F_4 - F_2^2 = \frac{1}{45}M(M + 1)(2M + 1)(2M + 3)(4M^2 - 1)
$$

(3.3.6)

$$
D_8 = F_2 F_6 - F_4^2 = \frac{3}{35}M(M + 2)(M^2 - 1)D_4
$$

Inserting the above expressions for $\Phi$ into Eq. (3.3.3), we determine the corresponding LPSM filters. For $d = 0$, we find for $-M \leq m, k \leq M$:

$$
b_m(k) = B_{mk} = \frac{1}{F_0} = \frac{1}{N}
$$

(3.3.7)

For $d = 1$:

$$
b_m(k) = B_{mk} = 1 + \frac{mk}{F_0} = \frac{1}{F_0} + \frac{mk}{F_2}
$$

(3.3.8)

For $d = 2$:

$$
b_m(k) = B_{mk} = \frac{F_4}{D_4} + \frac{1}{F_2}mk - \frac{F_2}{D_4}(m^2 + k^2) + \frac{F_0}{D_4}m^2k^2
$$

(3.3.9)

For $d = 3$:

$$
b_m(k) = B_{mk} = \frac{F_4}{D_4} + \frac{F_6}{D_8}mk - \frac{F_2}{D_4}(m^2 + k^2) + \frac{F_0}{D_4}m^2k^2
$$

(3.3.10)

$$
-\frac{F_4}{D_8}(km^3 + mk^3) + \frac{F_2}{D_8}m^3k^3
$$

3.3. Exact Design Equations

The required ratios are given explicitly as follows:

$$
\begin{align*}
F_4 &= \frac{3}{(2M^2 + 3M - 1)(2M^2 + 3M + 4M^2 - 1)} \\
D_4 &= \frac{2M + 3}{(2M^2 + 3M + 4M^2 - 1)} \\
F_2 &= \frac{15}{(2M + 3)(4M^2 - 1)} \\
D_2 &= \frac{2M + 3}{(4M^2 - 1)} \\
F_0 &= \frac{45}{M(M + 1)(2M + 3)(4M^2 - 1)} \\
D_0 &= \frac{25}{M(M + 2)(2M^2 + 3M + 4M^2 - 1)} \\
F_6 &= \frac{35}{3M^2 - 1}
\end{align*}
$$

(3.3.11)

$$
D_6 = \frac{M(M + 2)(M^2 - 1)(2M + 3)(4M^2 - 1)}{175}
$$

In a similar fashion, we also find for the case $d = 4$:

$$
b_m(k) = B_{mk} = D_{12}D_8 mk - \frac{D_{10}}{D_8}m^4k^2 + \frac{D_8}{D_0}m^2k^2
$$

(3.3.12)

$$
-\frac{F_4}{D_8}(km^3 + mk^3) + \frac{F_2}{D_8}m^3k^3 + \frac{F_0}{D_8}m^2k^2 - \frac{D_0}{D_8}(m^2k^4 + m^4k^2) + \frac{D_4}{D_0}m^4k^4
$$

(3.3.13)

where

$$
D_6 = F_0 F_6 - F_2 F_4 \\
D_{10} = F_2 F_6 - F_4 F_8 \\
D_{12} = F_4 F_8 - F_6^2
$$

These are given explicitly as follows:

$$
D_6 = \frac{1}{7}(6M^2 + 6M - 5)D_4
$$

(3.3.14)

$$
D_{10} = \frac{1}{21}M(M + 2)(M^2 - 1)(2M^2 + 2M - 3)D_4
$$

$$
D_{12} = \frac{1}{735}M(M + 2)(15M^2 + 30M^3 - 35M^2 - 50M + 12)D_4
$$

$$
D = \frac{4}{11025}M(M + 2)(M^2 - 1)(2M + 5)(4M^2 - 9)(4M^2 - 1)D_4
$$
3. Local Polynomial Filters

and the required ratios are:

\[
\begin{align*}
D_{12}/D &= \frac{15(15M^4 + 30M^3 - 35M^2 - 50M + 12)}{4(2M + 5)(4M^2 - 1)(4M^2 - 9)} \\
D_{10}/D &= \frac{525(2M^2 + 2M - 3)}{4(2M + 5)(4M^2 - 1)(4M^2 - 9)} \\
E_8/D &= \frac{2205(4M^4 + 8M^3 - 4M^2 - 8M + 5)}{4M(M + 2)(M^2 - 1)(2M + 5)(4M^2 - 1)(4M^2 - 9)} \\
D_8/D &= \frac{945}{4(2M + 5)(4M^2 - 1)(4M^2 - 9)} \\
D_6/D &= \frac{1575(6M^2 + 6M - 5)}{4M(M + 2)(M^2 - 1)(2M + 5)(4M^2 - 1)(4M^2 - 9)} \\
D_4/D &= \frac{11025}{4M(M + 2)(M^2 - 1)(2M + 5)(4M^2 - 1)(4M^2 - 9)}
\end{align*}
\]

(3.3.15)

In this case, the matrix \( F \) and its two interlaced submatrices are:

\[
F = \begin{bmatrix}
F_0 & 0 & F_2 & 0 & F_4 \\
0 & F_2 & 0 & F_4 & 0 \\
F_2 & 0 & F_4 & 0 & F_6 \\
0 & F_4 & 0 & F_6 & 0 \\
F_4 & 0 & F_6 & 0 & F_8
\end{bmatrix},
\begin{bmatrix}
F_0 & F_2 & F_4 \\
F_2 & F_4 & F_6 \\
F_4 & F_6 & F_8
\end{bmatrix},
\begin{bmatrix}
F_2 & F_4 \\
F_4 & F_6
\end{bmatrix},
\begin{bmatrix}
F_0 & F_2 & F_4 \\
F_2 & F_4 & F_6 \\
F_4 & F_6 & F_8
\end{bmatrix}
\]

Its inverse—obtained by interlacing the inverses of these two submatrices—can be expressed in terms of the determinant quantities of Eq. (3.3.13):

\[
\phi = F^{-1} = \begin{bmatrix}
D_{12}/D & 0 & -D_{10}/D & 0 & D_8/D \\
0 & F_0/D & 0 & -F_2/D & 0 \\
-D_{10}/D & 0 & E_8/D & 0 & -D_6/D \\
0 & -F_4/D & 0 & F_2/D & 0 \\
D_8/D & 0 & -D_6/D & 0 & D_4/D
\end{bmatrix}
\]

Eqs. (3.3.5)-(3.3.15) provide closed-form expressions for the LPSM filters \( b_m(k) \) of orders \( d = 0, 1, 2, 3, 4 \). Setting \( m = 0 \), we obtain the explicit forms of the steady-state filters \( b_0(k) \), \(-M \leq k \leq M\). For \( d = 0, 1 \):

\[
b_0(k) = \frac{1}{2M + 1}
\]

for \( d = 2, 3 \):

\[
b_0(k) = 3(3M^2 + 3M - 1 - 5k^2) / (2M + 3)(4M^2 - 1)
\]

(3.3.17)

and for \( d = 4, 5 \):

\[
b_0(k) = 15(15M^4 + 30M^3 - 35M^2 - 50M + 12 - 35(2M^2 + 2M - 3)k^2 + 63k^4) / 4(2M + 5)(4M^2 - 1)(4M^2 - 9)
\]

(3.3.18)

3.4. Geometric Interpretation

Example 3.3.1: Determine the quadratic/cubic LPSM filters of lengths \( N = 5, 7, 9 \), using (3.3.17) with \( M = 2, 3, 4 \), we find for \(-M \leq k \leq M\):

\[
b_0(k) = \frac{17 - 5k^2}{35} = \frac{1}{35}[-3, 12, 17, -12, -3]
\]

\[
b_0(k) = \frac{7 - k^2}{21} = \frac{1}{21}[-2, 3, 6, 7, 6, 3, -2]
\]

\[
b_0(k) = \frac{59 - 5k^2}{231} = \frac{1}{231}[-21, 14, 39, 54, 59, 39, 14, -21]
\]

where the coefficients have been reduced to integers as much as possible.

Example 3.3.2: Determine the quartic and quintic LPSM filters of length \( N = 7, 9 \), using Eq. (3.3.18) with \( M = 3, 4 \), we find:

\[
b_0(k) = \frac{131 - 61.25k^2 + 5.25k^4}{231} = \frac{1}{231}[-30, 75, 131, 75, -30, 5]
\]

\[
b_0(k) = \frac{179 - 46.25k^2 + 2.25k^4}{429} = \frac{1}{429}[-15, 55, 30, 135, 179, 135, 30, -55, 15]
\]

3.4 Geometric Interpretation

The LPSM filters admit a nice geometric interpretation, which is standard in least-squares problems. Let \( \mathcal{Y} \) be the vector space of the \( N \)-dimensional real-valued vectors \( y \), that is, the space \( \mathbb{R}^N \), and let \( S \) be the \((d+1)\)-dimensional subspace spanned by all linear combinations of the basis vectors \( s_i, i = 0, 1, \ldots, d \).

Thus, the matrix \( S = [s_0, s_1, \ldots, s_d] \) is a (non-orthogonal) basis of the subspace \( \mathcal{S} \). The smoothed vector \( \hat{y} \) being a linear combination of the \( s_i \) belongs to the subspace \( \mathcal{S} \). Moreover, because of the orthogonality equations (3.2.8), \( \hat{y} \) is orthogonal to the error vector \( e \):

\[
\hat{y}^T e = (Sc)^T e = c^T \mathcal{S} e = 0
\]

Then, the equation \( e = y - \hat{y} \) can be rewritten as the orthogonal decomposition:

\[
y = \hat{y} + e
\]

(4.3.1)

which expresses \( y \) as a sum of a part that belongs to the subspace \( \mathcal{S} \) and a part that belongs to the orthogonol complement subspace \( \mathcal{S}^\perp \). The decomposition is unique and represents the direct sum decomposition of the full vector space \( \mathcal{Y} \):

\[
\mathcal{Y} = \mathcal{S} \oplus \mathcal{S}^\perp
\]

This geometric interpretation requires that the dimension of the subspace \( \mathcal{S} \) not exceed the dimension of the full space \( \mathcal{Y} \), that is, \( d + 1 \leq N \). The component \( \hat{y} \) that lies in \( \mathcal{S} \) is the projection of \( y \) onto \( \mathcal{S} \). The matrix \( B \) in Eq. (3.2.11) is the corresponding projection matrix. As such, it will be symmetric, \( B^T = B \), and idempotent:

\[
B^2 = B
\]

(3.4.2)
3. Local Polynomial Filters

The proof is straightforward:

\[ B^2 = (S^{-1}S^T)^2 = S^{-1}(S^T S)S^{-1}S^T = S^{-1}S^T = B \]

The matrix \((I - B)\), where \( I \) is the \( N \)-dimensional identity matrix, is also a projection matrix, projecting onto the orthogonal subspace \( S^\perp \). Thus, the error vector \( e \) belonging to \( S^\perp \) can be obtained from \( y \) by the projection:

\[ e = y - \hat{y} = (I - B)y \]

Because \((I - B)\) is also idempotent and symmetric, \((I - B)^2 = (I - B)\), we obtain for the minimized value of the performance index \( J \) of Eq. (3.2.6):

\[ J_{\text{min}} = e^T e = y^T (I - B)^2 y = y^T (I - B) y = y^T y - y^T B y \quad (3.4.3) \]

3.5 Orthogonal Polynomial Bases

Computationally, the non-orthogonal basis \( S = [s_0, s_1, \ldots, s_d] \) is not the most convenient one. The Gram-Schmidt orthogonalization process may be applied to the columns of \( S \) to obtain an orthogonal basis. This procedure amounts to performing the QR-factorization\(^1\) on \( S \), that is,

\[ S = QR \quad (3.5.1) \]

where \( Q \) is an \( N \times (d+1) \) matrix with orthonormal columns, that is, \( Q^T Q = I \), and \( R \) is a \((d+1) \times (d+1)\) non-singular upper-triangular matrix.

The columns of \( Q = [q_0, q_1, \ldots, q_d] \), correspond to the (orthonormalized) discrete Chebyshev or Gram polynomials \( q_i(n) \), \( i = 0, 1, \ldots, d \), constructed from the monomial basis \( s_i(n) = n^i \) by the Gram-Schmidt process. Noting that \( STS = R^T (Q^TQ)R = R^T R \), the design of the filter matrices \( B, G \) can be formulated more efficiently as follows:

\[ F = S^T S = R^T R \]
\[ G = S F^{-1} = Q R^{-T} \]
\[ B = S F^{-1} S^T = Q Q^T \]

which lead to the explicit construction of the differentiation and LPSM filters in terms of the Chebyshev polynomials \( q_i(n) \):

\[ g_i(n) = \sum_{j=0}^{i} q_j (R^{-1})_{ij} \Rightarrow g_i(n) = \sum_{j=0}^{i} q_j(n) (R^{-1})_{ij} \]
\[ B_m(k) = \sum_{i=0}^{d} q_i(k) q_i(m) \]

The expression for \( B_m(k) \) can be simplified further using the Christoffel-Darboux identity for orthogonal polynomials. We discuss these matters further in Sec. 4.3. The MATLAB function \( \text{lpsm} \) implements (3.5.2). Its inputs are \( N, d \) and its outputs \( B, G \):

3.6 Polynomial Predictive and Interpolation Filters

The function constructs the basis matrix \( S \) with the help of the function \( \text{lpbasis} \) and carries out its QR-factorization with the help of the built-in function \( \text{qr} \). The following code fragment illustrates the computational steps:

\[ [B, G] = \text{lpsm}(N, d); \]

\[ [Q, R] = \text{qr}(S, 0); \]

\[ G = Q R^T; \]

\[ B = Q^T Q^T; \]

\[ \text{B, G; } \]

\[ \text{local polynomial smoothing and differentiation filter design} \]

The case \( d + 1 = N \) or \( d = N - 1 \) is of special interest, corresponding to ordinary polynomial Lagrange interpolation. Indeed, in this case, the basis matrix \( S \) becomes a square non-singular \( N \times N \) matrix with an ordinary inverse \( S^{-1} \), which implies that \( B \) becomes the identity matrix,

\[ B = S (S^T S)^{-1} S^T = S (S^{-1} S^T) S = I \]

or, equivalently, the subspace \( S \) becomes the full space \( \mathbb{Y} \). The optimal polynomial of degree \( d = N - 1 \) fits through all the sample points of the \( N \)-dimensional vector \( y \), that is, \( e = 0 \) or \( y = \hat{y} = S c \), with solution \( c = S^{-1} y \), and interpolates between those samples. This polynomial is defined for any independent variable \( t \) by:

\[ \hat{y}_t = \sum_{i=0}^{N-1} c_i t^i = c^T u_t = y^T S^{-T} u_t = y^T b_t = \sum_{k=-M}^{M} b_t(k) y_k \quad (3.6.1) \]

where we set,

\[ u_t = \begin{bmatrix} 1 & t & \vdots & t^{N-1} \end{bmatrix}, \quad b_t(k) = \sum_{i=0}^{N-1} (S^{-1})_{ik} t^i \quad (3.6.2) \]

The polynomials \( b_t(k) \) of degree \((N-1)\) in \( t \) are the ordinary Lagrange interpolation polynomials, interpolating through the points \( y_k \). To see this, we note that at each discrete value of \( t \), say \( t = m \) with \(-M \leq m \leq M\), we have:

\[ b_m(k) = \sum_{i=0}^{M} (S^{-1})_{ik} m^i = \sum_{i=0}^{N-1} (S^{-1})_{ik} S_{mi} = (S S^{-1})_{mk} = \delta(m - k) \quad (3.6.3) \]

so that the polynomial passes through the signal values at the sampling instants:

\[ \hat{y}_t |_{t=m} = \sum_{k=-M}^{M} b_m(k) y_k = \sum_{k=-M}^{M} \delta(m - k) y_k = y_m \]
3. Local Polynomial Filters

It is straightforward to show using the property (3.6.3) that $b_t(k)$ is given by the usual Lagrange interpolation formula:

$$b_t(k) = \sum_{m=-M}^{M} \frac{(t - m)}{k - m} , \quad -M \leq k \leq M \tag{3.6.4}$$

Indeed, Eq. (3.6.4) states that the $(2M)$ roots of $b_t(k)$ are the points $t = m$, for $-M \leq m \leq M$ and $m \neq k$, which fixes the polynomial up to a constant. That constant is determined by the condition $b_t(k) = 1$.

**Example 3.6.1:** For $N = 5$ and $d = N - 1 = 4$, the fourth degree Lagrange polynomials, constructed from Eq. (3.6.4), can be expanded in powers of $t$:

$$
\begin{bmatrix}
    b_t(-2) \\
    b_t(-1) \\
    b_t(0) \\
    b_t(1) \\
    b_t(2)
\end{bmatrix} = \frac{1}{24}
\begin{bmatrix}
    0 & 2 & -1 & -2 & 1 \\
    0 & -16 & 16 & 4 & -4 \\
    24 & 0 & -30 & 0 & 6 \\
    0 & 16 & 16 & -4 & -4 \\
    0 & -2 & -1 & 2 & 1
\end{bmatrix}
t^0 \begin{bmatrix}
    1 \\
    t \\
    t^2 \\
    t^3 \\
    t^4
\end{bmatrix}
$$

The coefficient matrix is recognized as the inverse transposed of the basis matrix $S$:

$$S = \begin{bmatrix}
    1 & -2 & 4 & -8 & 16 \\
    1 & -1 & 1 & -1 & 1 \\
    1 & 0 & 0 & 0 & 0 \\
    1 & 1 & 1 & 1 & 1 \\
    1 & 2 & 4 & 8 & 16
\end{bmatrix} \quad \Rightarrow \quad S^{-T} = \frac{1}{24}
\begin{bmatrix}
    0 & 2 & -1 & -2 & 1 \\
    0 & -16 & 16 & 4 & -4 \\
    24 & 0 & -30 & 0 & 6 \\
    0 & 16 & 16 & -4 & -4 \\
    0 & -2 & -1 & 2 & 1
\end{bmatrix}
t^0 \begin{bmatrix}
    1 \\
    t \\
    t^2 \\
    t^3 \\
    t^4
\end{bmatrix}
$$

which verifies Eq. (3.6.2).

We note that $b_t(k)$ can be written in the following analytical form, which shows the relation of the Lagrange interpolation filter to the ideal sinc-interpolation filter:

$$b_t(k) = \frac{\Gamma(M + 1 + t) \Gamma(M + 1 - t)}{\Gamma(M + 1 + k) \Gamma(M + 1 - k)} \cdot \sin\left(\pi (t - k)\right) / \pi (t - k) \tag{3.6.5}$$

Some alternative expressions are as follows:

$$b_t(k) = (-1)^{M+k} \sum_{m=M+k}^{2M} m \binom{M + t}{m} \binom{m}{M} (-1)^m \tag{3.6.6}$$

and since the $b_t(k)$ sum up to one, we also have [156]:

$$b_t(k) = \left[ \sum_{n=-M}^{M} b_t(n) b_t(k) \right]^{-1} = \left[ \sum_{n=-M}^{M} (-1)^{k-n} \frac{(M + k)! (M - k)!}{(M + n)! (M - n)!} \right]^{-1} \tag{3.6.8}$$

3.6. Polynomial Predictive and Interpolation Filters

For polynomial orders $d < N - 1$, one can still interpolate approximately and smoothly between the samples $y_m$. In this case, using $c = G^T y = (S^T S)^{-1} S^T y$, we have:

$$\hat{y}_t = \sum_{i=0}^{d} c_i t^i = c^T u_t = y^T G u_t = y^T b_t = \sum_{k=-M}^{M} b_t(k) y_k \tag{3.6.9}$$

where now

$$u_t = \begin{bmatrix}
    t^0 \\
    t^1 \\
    \vdots \\
    t^d
\end{bmatrix}, \quad b_t = G u_t = S(S^T S)^{-1} u_t \quad \Rightarrow \quad b_t(k) = \sum_{i=0}^{d} G_{ki} t^i \tag{3.6.10}$$

and shifting the origin $k = 0$ to the arbitrary time instant $n$, we obtain the interpolation formula for a shift $t$ relative to the time instant $n$:

$$y_n \xrightarrow{b_t^k} \hat{y}_{n+t} = \sum_{k=-M}^{M} b_t(k) y_{n+k} = \sum_{k=-M}^{M} b_t^k(k) y_{n-k} \tag{3.6.11}$$

where $b_t^k(k) = b_t(-k)$. Such formulas can also be used for prediction by choosing $t > M$ so that $n + t > n + M$, that is, it lies beyond the end of the filter range.

We can obtain closed-form expressions for the interpolation filters $b_t(k)$ for $d = 0, 1, 2, 3, 4$ and arbitrary $M$, by replacing in Eqs. (3.3.7)-(3.3.12) the variable $m$ in $b_m(k)$ by the variable $t$. For example, for $d = 1, 2, 3, 4$, we have, respectively:

$$b_t(k) = \frac{1}{F_0} + \frac{t k}{F_2}$$

$$b_t(k) = \frac{F_4}{D_8} + \frac{1}{F_2} t k - \frac{F_2}{D_8} (t^2 + k^2) + \frac{F_0}{D_8} t^2 k^2$$

$$b_t(k) = \frac{F_4}{D_8} + \frac{F_6}{D_8} t k - \frac{F_2}{D_8} (t^2 + k^2) + \frac{F_0}{D_8} t^2 k^2 - \frac{F_6}{D_8} (k^3 + tk^2) + \frac{F_2}{D_8} t k^3$$

$$b_t(k) = \frac{D_{12}}{D} + \frac{F_6}{D} t k - \frac{D_{10}}{D} (t^2 + k^2) + \frac{E_8}{D} t k^2 - \frac{F_6}{D} (k^3 + tk^2) + \frac{F_2}{D} t k^3$$

$$\frac{F_2}{D} t^2 k^3 + \frac{D_8}{D} (t^4 + k^4) - \frac{D_8}{D} (t^2 k^4 + k^4) + \frac{D_8}{D} t k^4$$

where the required coefficient matrices are given by Eqs. (3.3.11) and (3.3.15). The interpolation filter (3.6.10) may be written in terms of the columns of $G = [g_0, g_1, \ldots, g_d]$:  

$$b_t(k) = \sum_{i=0}^{d} G_{ki} t^i = \sum_{i=0}^{d} g_i(k) t^i \quad \Rightarrow \quad b_t = \sum_{i=0}^{d} g_i t^i \tag{3.6.13}$$

This representation admits a convenient realization, known as a Farlow structure, which allows the changing of the parameter $t$ on the fly without having to redesign the
3. Local Polynomial Filters

3.6. Polynomial Predictive and Interpolation Filters

The filters \( b_i(k) \) and \( h_i(k) \) satisfy the following polynomial-preserving moment constraints (being equivalent to \( S^j b_i = u_i \)), where \( j = 0, 1, \ldots, d \):

\[
\sum_{k=-M}^{M} k^i b_i(k) = t^i \quad \Rightarrow \quad \sum_{k=0}^{N-1} k^i h_i(k) = (-\tau)^i, \quad \sum_{k=0}^{N-1} k^i h_{-\tau}(k) = \tau^i \quad (3.6.18)
\]

These constraints imply that Eqs. (3.6.16) and (3.6.17) are exact for polynomials of degree \( r \leq d \). For any such polynomial \( P(n) \), we have:

\[
\sum_{k=0}^{N-1} h_{-\tau}(k) P(n-k) = P(n-\tau) \quad (3.6.19)
\]

For example, we have for the monomial \( n^r \) with \( r \leq d \):

\[
\sum_{k=0}^{N-1} h_{-\tau}(k) (n-k)^r = \sum_{k=0}^{N-1} h_{-\tau}(k) \sum_{i=0}^{r} \binom{r}{i} n^{-i} (-1)^i k^i
\]

\[
= \sum_{i=0}^{r} \binom{r}{i} n^{-i} (-1)^i \sum_{k=0}^{N-1} k^i h_{-\tau}(k) = \sum_{i=0}^{r} \binom{r}{i} n^{-i} (-1)^i \tau^i = (n-\tau)^r
\]

It is in the sense of Eq. (3.6.19) that we may think of the transfer function of the filter \( h_{-\tau}(k) \) as approximating the ideal fractional delay \( z^{-\tau} \):

\[
\sum_{k=0}^{N-1} h_{-\tau}(k) z^{-k} \approx z^{-\tau} \quad (3.6.20)
\]

Further insight into the nature of the approximation (3.6.20) can be gained by considering the Lagrange interpolation case, \( d = N-1 \). From the definition of \( h_{-\tau}(k) = b_{M-\tau}(M-k) \) and Eqs. (3.6.4) and (3.6.6), we obtain, for \( k = 0, 1, \ldots, N-1 \):

\[
h_{-\tau}(k) = \sum_{i=0}^{N-1} \binom{N-1-\tau}{i} \binom{i}{N-1-k} (-1)^i (N-1-k)^{(N-1-k)} \quad (3.6.21)
\]

The \( z \)-transform of \( h_{-\tau}(k) \) is then:

\[
\sum_{k=0}^{N-1} h_{-\tau}(k) z^{-k} = \sum_{k=0}^{N-1} \sum_{i=N-1-k}^{N-1} \binom{N-1-\tau}{i} \binom{i}{N-1-k} (-1)^i (N-1-k)^{(N-1-k)} z^{-k}
\]

\[
= z^{-(N-1)} \sum_{k=0}^{N-1} \sum_{i=N-1-k}^{N-1} \binom{N-1-\tau}{i} \binom{i}{N-1-k} (-1)^i (N-1-k)^{(N-1-k)} z^{N-1-k}
\]

Changing summation variables and using the binomial expansion of \( (z-1)^j \), we obtain:

\[
\sum_{k=0}^{N-1} h_{-\tau}(k) z^{-k} = z^{-(N-1)} \sum_{i=0}^{N-1} \sum_{j=0}^{m} \binom{N-1-\tau}{i} \binom{i}{j} (-1)^j z^j
\]

\[
= z^{-(N-1)} \sum_{i=0}^{N-1} \binom{N-1-\tau}{i} (z-1)^j \quad (3.6.22)
\]
Applying the binomial identity,

\[(1 + x)^\alpha = \sum_{m=0}^{\infty} \binom{\alpha}{m} x^m\]

(3.6.23)

with \(x = z - 1\) and \(\alpha = N - 1 - \tau\), we have,

\[z^{N-1-\tau} = (1 + z - 1)^{N-1-\tau} = \sum_{i=0}^{\infty} \binom{N-1-\tau}{i} (z-1)^i\]

(3.6.24)

We recognize the sum in Eq. (3.6.22) to be the first \(N\) terms of (3.6.24). Thus, taking that sum to approximately represent \(z^{N-1-\tau}\), we have,

\[\sum_{k=0}^{N-1} h_{-\tau}(k) z^{-k} = z^{-(N-1)} z^{N-1-\tau} = z^{-\tau}\]

(3.6.25)

This approximation becomes exact whenever \(\tau\) is an integer, say \(\tau = m\), with \(m = 0, 1, \ldots, N-1\). Indeed in this case, the summation range 0 \(\leq i \leq N-1\) in Eq. (3.6.22) can be restricted to 0 \(\leq i \leq N-1-m\) because the binomial coefficient vanishes whenever its (integer) arguments satisfy \(N-1-m < i \leq N-1\). We then have an ordinary binomial expansion for an integer power:

\[\sum_{k=0}^{N-1} h_{-\tau}(k) z^{-k} = z^{-(N-1)} \sum_{i=0}^{N-1-m} \binom{N-1-m}{i}(z-1)^i = z^{-(N-1)} (1 + z - 1)^{N-1-m} = z^{-m}\]

which implies the expected result \(h_{-\tau}(k) = \delta(k-m)\).

This interpolates between the points \([y_{N-1}, y_{N-2}, \ldots, y_0]\) with \(\tau\) measured backwards from the end-point \(y_N\). We may measure the interpolation distance forward from the first point \(y_{N-1}\) by defining \(x = N-1-\tau\). Then, Eq. (3.6.26) reads,

\[\hat{y}_{N-1-\tau} = \sum_{k=0}^{N-1} h_{-\tau}(k) y_{N-k} = \sum_{i=0}^{N-1} \binom{N-1-\tau}{i} \Delta y_{N-1}\]

(3.6.26)

This interpolates between the points \([y_{N-1}, y_{N-2}, \ldots, y_0]\) with \(\tau\) measured backwards from the end-point \(y_N\). We may measure the interpolation distance forward from the first point \(y_{N-1}\) by defining \(x = N-1-\tau\). Then, Eq. (3.6.26) reads,

\[\hat{y}_{N-1-\tau} = \sum_{k=0}^{N-1} h_{-\tau}(k) y_{N-k} = \sum_{i=0}^{N-1} \binom{N-1-\tau}{i} \Delta y_{N-1}\]

(3.6.26)

and setting \(n = N - 1\) so that the data range is \([y_0, y_1, \ldots, y_{N-1}]\), we obtain the usual way of writing Newton's polynomial interpolation formula:

\[\hat{y}_x = \sum_{i=0}^{N-1} \binom{x}{i} \Delta^i y_0 = \sum_{i=0}^{N-1} \frac{x(x-1) \cdots (x-i+1)}{i!} \Delta^i y_0\]

(3.6.28)

We note also that Eq. (3.6.21) is valid for either even or odd values of \(N\). For \(N = 2, 3, 4\), we obtain for the corresponding filter coefficients:

\[
\begin{bmatrix}
\hat{h}_{-\tau}(0) \\
\hat{h}_{-\tau}(1) \\
\hat{h}_{-\tau}(2) \\
\hat{h}_{-\tau}(3)
\end{bmatrix} = \begin{bmatrix}
1 & -\tau & \tau^2 & -\tau^3 \\
0 & 1 & 2\tau & -3\tau^2 + \tau^3 \\
0 & 0 & 1 & -2\tau^2 + \tau^3 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

(3.6.29)

and the corresponding interpolation formulas:

\[
\hat{y}_{N-\tau} = (1 - \tau)y_N + \tau y_{N-1}
\]

\[
\hat{y}_{N-\tau} = \frac{1}{2}(1 - \tau)(1 - 2\tau)y_N - (\tau - 2)y_{N-1} + \frac{1}{2}(\tau - 1)y_{N-2}
\]

(3.6.30)

Example 3.6.2: Fig. 3.6.3 shows in the top row an example of a Lagrange fractional-delay filter with \(N = 3\) and polynomial order \(d = N - 1 = 2\) for the delay values \(\tau = m/10\), \(m = 1, 2, \ldots, 10\). The bottom row is the case \(N = 5\) and \(d = N - 1 = 4\) with delays \(\tau\) extending over the interval 0 \(\leq \tau \leq 2\). This filter interpolates between the samples \([y_{N-4}, y_{N-3}, y_{N-2}, y_{N-1}, y_N]\). The chosen range of \(\tau\)’s spans the gaps between \([y_{N-2}, y_{N-1}, y_N]\). For the subrange 0 \(\leq \tau \leq 1\) which spans \([y_{N-1}, y_N]\), the magnitude response is greater than one, while it is less than one for the more central range 1 \(\leq \tau \leq 2\) which spans \([y_{N-2}, y_{N-1}]\). The following MATLAB code segment illustrates the generation of the upper two graphs:

```matlab
% frequencies 0 \leq \omega \leq \pi
N=3; d=N-1; M = floor(N/2);
Hmag = []; Hdel = []; M = floor(N/2);
Hmag = []; Hdel = [];
for m=1:10,
    tau = m/10; % desired delays
    h = interp1(N, d, tau); % interp1 is discussed in Sec. 3.8
    H = freqz(N, 1, h); % magnitude responses in dB
    Delay = angle(H)/w; Delay(1) = tau; % phase delays
    Hmag = [Hmag; 10*log10(abs(H))];
    Hdel = [Hdel; Delay];
    figure; plot(f,Hmag); figure; plot(f,Hdel);
end
```

The filters were calculated with the function \(\text{interp1}\) (from Sec. 3.8) with arguments \(d = N - 1 - M\) and \(\tau = M - K\), with reversed output to account for the definition \(h_{-\tau}(k) = b_{M-k}(M-K)\). In both cases, we observe that the useful bandwidth of operation, within which both the phase delays have the correct values and the magnitude response is near unity, is fairly narrow extending to about \(\omega = 0.2\pi\), or \(f = f_s/10\) in units of the sampling rate \(f_s\).
3. Local Polynomial Filters

3.7 Minimum Variance Filters

Taking matrix elements, we have \( B_{km} = (B^T)^{mk} = (B^T B)^{mk} \). But, \( B_{km} \) is the kth component of the \( m \)th column \( b_m \). Using a similar argument as in Eq. (3.2.13), we also have \( (B^T B)^{mk} = b_m^T b_k \). Therefore,

\[
B_m^T b_k = b_m(k)
\]

For \( k = m \), we have the diagonal elements of \( B^T B = B \):

\[
R = b_m^T b_m = b_m(m)
\]

These are recognized as the NRRs of the filters \( b_m \). In particular, for \( m = 0 \), we have \( R = b_0^T b_0 = b_0(0) \). Setting \( k = 0 \) in Eqs. (3.3.16)-(3.3.18), we find that the NRRs of the cases \( d = 0, 1 \), \( d = 2, 3 \), and \( d = 4, 5 \) are given by the coefficient ratios \( 1/F_0, F_4/D_4 \), and \( D_{12}/D \). Therefore:

\[
\begin{align*}
\text{for } d = 0, 1 & : R = \frac{1}{N} \\
\text{for } d = 2, 3 & : R = \frac{3(3M^2 + 3M - 1)}{(2M + 3)(4M^2 - 1)} \\
\text{for } d = 4, 5 & : R = \frac{15(15M^4 + 30M^3 - 35M^2 - 50M + 12)}{4(2M + 5)(4M^2 - 1)(4M^2 - 9)}
\end{align*}
\]

In the limit of large \( N \) or \( M \), we have the approximate asymptotic expressions:

\[
\begin{align*}
\text{for } d = 0, 1 & : R \approx \frac{1}{N} \\
\text{for } d = 2, 3 & : R \approx \frac{9}{4N} = 2.25 \\
\text{for } d = 4, 5 & : R \approx \frac{225/64}{N} = 3.52/N
\end{align*}
\]

Thus, the noise reductions achieved by the quadratic/cubic and quartic/quintic cases are 2.25 and 3.52 times worse than that of the plain FIR average of the same length \( N \). Another consequence of the projection nature of \( B \) is:

\[
B^T S = S, \quad S^T B = S^T \quad (3.7.4)
\]

Indeed, \( B^T S = BS = S(S^T S)^{-1}S^T S = S \). Column-wise the first equation states that:

\[
B^T [s_0, s_1, \ldots, s_d] = [s_0, s_1, \ldots, s_d] \Rightarrow B^T s_i = s_i, \quad i = 0, 1, \ldots, d
\]

Thus, the basis vectors \( s_i \) remain invariant under projection, but that is to be expected because they already lie in \( S \). In fact, any other linear combination of them, such as Eq. (3.2.30), remains invariant under \( B \), that is, \( B^T \hat{y} = \hat{y} \).

This property answers the question: When are the smoothed values equal to the original ones, \( \hat{y} = y \), or, equivalently, when is the error zero, \( e = 0 \)? Because \( e = y - B^T y \), the error will be zero if and only if \( B^T y = y \), which means that \( y \) already lies in \( S \), that is, it is a linear combination of \( s_i \). This implies that the samples \( y_m \) are already \( d \)th order polynomial functions of \( m \), as in Eq. (3.2.27).
3. Local Polynomial Filters

The second equation in (3.7.4) implies certain constraints on the filters \( b_m \), which can be used to develop an alternative approach to the LPSM filter design problem in terms of minimizing the NRR subject to constraints. To see this, we write the \((d+1) \times N\) transposed matrix \( S^T \) column-wise:

\[
S^T = [u_{-M}, u_{-1}, u_0, u_1, \ldots, u_M] \quad (3.7.5)
\]

For example, in the \( N = 5, d = 2 \) case, we have:

\[
S^T = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
-2 & -1 & 0 & 1 & 2 \\
4 & 1 & 0 & 1 & 4
\end{bmatrix}
\]

It is easily verified that the \( m \)th column \( u_m \) is simply

\[
u_m = \begin{bmatrix}
1 \\
m \\
m^2 \\
\vdots \\
m^d
\end{bmatrix}, \quad -M \leq m \leq M \quad (3.7.6)
\]

which is the same as \( u_t \) at \( t = m \), in terms of the definition (3.6.10). Using \( B = GST \), we can express the LPSM filters \( b_m \) in terms of \( u_m \), as follows:

\[
[b_{-M}, \ldots, b_{-1}, b_0, b_1, \ldots, b_M] = B = GST = G[u_{-M}, \ldots, u_{-1}, u_0, u_1, \ldots, u_M]
\]

which implies:

\[
b_m = Gu_m = SF^{-1}u_m \quad (3.7.7)
\]

Multiplying by \( S^T \), we find \( S^TB = S^TF^{-1}u_m = u_m \), or,

\[
S^TB_m = u_m = \begin{bmatrix}
s_1 b_m \\
s_2 b_m \\
\vdots \\
s_d b_m
\end{bmatrix}
\]

These relationships are the column-wise equivalent of \( S^TB = S^T \). Thus, each LPSM filter \( b_m \) satisfies \((d+1)\) linear constraints:

\[
s_i^Tb_m = m^i, \quad i = 0, 1, \ldots, d \quad (3.7.9)
\]

Writing the dot products explicitly, we have equivalently:

\[
\sum_{n=-M}^{M} n^i b_m(n) = m^i, \quad i = 0, 1, \ldots, d \quad (3.7.10)
\]

3.7. Minimum Variance Filters

In particular, for the steady-state LPSM filter \( b_0 \), we have \( u_0 = [1, 0, 0, \ldots, 0]^T \), with \( i \)th component \( \delta(i) \). Therefore, the constraint \( S^Tb_0 = u_0 \) reads component-wise:

\[
\sum_{n=-M}^{M} n^i b_0(n) = \delta(i), \quad i = 0, 1, \ldots, d \quad (3.7.11)
\]

For \( i = 0 \), this is the usual DC constraint:

\[
\sum_{n=-M}^{M} b_0(n) = 1 \quad (3.7.12)
\]

and for \( i = 1, 2, \ldots, d \):

\[
\sum_{n=-M}^{M} n^i b_0(n) = 0 \quad (3.7.13)
\]

The quantity in the left-hand side of Eq. (3.7.11) is called the \( i \)th moment of the impulse response \( b_0(n) \). Because of the symmetric limits of summation over \( n \) and the symmetry of \( b_0(n) \) about its middle, the moments (3.7.13) will be zero for odd \( i \) and therefore are not extra constraints. However, for even \( i \), they are nontrivial constraints.

These moments are related to the derivatives of the frequency response at \( \omega = 0 \). Indeed, defining,

\[
B_0(\omega) = \sum_{n=-M}^{M} b_0(n)e^{-j\omega n}
\]

and differentiating it \( i \) times, we have:

\[
j^iB_0^{(i)}(\omega) = j^i \frac{d^iB_0(\omega)}{d\omega^i} = \sum_{n=-M}^{M} n^i b_0(n)e^{-j\omega n}
\]

Setting \( \omega = 0 \), we obtain:

\[
j^iB_0^{(i)}(0) = \sum_{n=-M}^{M} n^i b_0(n) \quad (3.7.14)
\]

Thus, the moment constraints (3.7.12) and (3.7.13) are equivalent to the DC constraint and the flatness constraints on the frequency response at \( \omega = 0 \):

\[
B_0(0) = 1, \quad B_0^{(i)}(0) = 0, \quad i = 1, 2, \ldots, d \quad (3.7.15)
\]

The larger the \( d \), the more derivatives vanish at \( \omega = 0 \), and the flatter the response \( B_0(\omega) \) becomes. This effectively increases the cutoff frequency of the lowpass filter—letting through more noise, but at the same time preserving more of the higher frequencies in the desired signal.

Figure 3.7.1 shows the magnitude response \( |B_0(\omega)| \) for the cases \( N = 7, 15 \) and \( d = 2, 4 \). The quadratic filters are flatter at DC than the plain FIR averager because of the extra constraint \( B_0^{(2)}(0) = 0 \). Similarly, the quartic filters are even flatter because
they satisfy two flatness conditions: $B_0''(0) = B_d''(0) = 0$. The cutoff frequencies are approximately doubled and tripled in the cases $d = 2$ and $d = 4$, as compared to $d = 0$.

A direct consequence of the moment constraints (3.7.11) is that the moments of the input signal $y(n)$ are preserved by the filtering operation (3.2.33), that is,

$$
\sum_{i} n^i \hat{X}(n) = \sum_{i} n^i y(n), \quad i = 0, 1, \ldots, d
$$

(3.7.16)

This can be proved easily working in the frequency domain. Differentiating the filtering equation $\hat{X}(\omega) = B_0(\omega)Y(\omega)$ $l$ times, and using the product rules of differentiation, we obtain:

$$
\hat{X}^{(i)}(\omega) = \sum_{j=0}^{l} \binom{l}{j} B_0^{(j)}(\omega)Y^{(l-j)}(\omega)
$$

Setting $\omega = 0$ and using the moment constraints satisfied by the filter, $B_0''(0) = \delta(j)$, we observe that only the $j = 0$ term will contribute to the above sum, giving:

$$
\hat{X}(0) = B_0(0)Y(0) = Y(0), \quad i = 0, 1, \ldots, d
$$

which implies Eq. (3.7.16), by virtue of Eq. (3.7.14) as applied to $x(n)$ and $y(n)$.

The preservation of moments is a useful property in applications, such as spectroscopic analysis or ECG processing, in which the desired signal has one or more sharp peaks, whose widths must be preserved by the smoothing operation. In particular, the second moment corresponding to $i = 2$ in Eq. (3.7.16) is a measure of the square of the width [42–52, 56, 58, 178].

The above moment constraints can be used in a direct way to design the LPSM filters. We consider first the more general problem of designing an optimum length-$N$ filter that minimizes the NRR subject to $d + 1$ arbitrary moment constraints. That is, minimize

$$
\mathcal{R} = b^T b = \sum_{n=-M}^{M} b(n)^2 = \min
$$

(3.7.17)

subject to the $d + 1$ constraints, with a given $u = [u_0, u_1, \ldots, u_d]^T$:

$$
s_i^T b = \sum_{n=-M}^{M} n^i b(n) = u_i, \quad i = 0, 1, \ldots, d \quad \Rightarrow \quad S^T b = u
$$

(3.7.18)

The minimization of Eq. (3.7.17) subject to (3.7.18) can be carried out with the help of Lagrange multipliers, that is, adding the constraint terms to the performance index:

$$
J = b^T b + 2 \sum_{i=0}^{d} \lambda_i (u_i - s_i^T b) = b^T b + 2 \lambda^T (u - S^T b)
$$

(3.7.19)

The gradient of $J$ with respect to the unknown filter $b$ is:

$$
\frac{\partial J}{\partial b} = 2b - 2S \lambda
$$

Setting the gradient to zero, and solving for $b$ gives:

$$
b = S \lambda = [s_0, s_1, \ldots, s_d]
$$

$$
\lambda_0 \quad \lambda_1 \quad \ldots \quad \lambda_d = \sum_{i=0}^{d} \lambda_i s_i
$$

Component-wise this means that $b(n)$ has the polynomial form:

$$
b(n) = \sum_{i=0}^{d} \lambda_i n^i
$$

The Lagrange multiplier vector $\lambda$ is determined by imposing the desired constraint:

$$
u = S^T b = S^T S \lambda = F \lambda \quad \Rightarrow \quad \lambda = F^{-1} u
$$

resulting in the optimum $b$:

$$
b = S \lambda = SF^{-1} u = S (S^T S)^{-1} u = Gu
$$

(3.7.20)

Since the solution minimizes the norm $b^T b$, it is recognized to be the minimum-norm solution of the $(d+1) \times N$ full-rank under-determined linear system $S^T b = u$, which can be obtained by the pseudoinverse of $S^T$, that is, $b = (S^T)^+ u$, where according to Eq. (15.4.10), $(S^T)^+ = S (S^T S)^{-1}$. In MATLAB, we can simply write $b = pinv(S^T) u$.

Comparing this solution with Eqs. (3.7.7) and (3.7.8), we conclude that the LPSM filters $b_m$ can be thought of as the optimum filters that have minimum NRR with constraint vectors $u = u_m$, that is, the minimization problems,

$$
\mathcal{R} = b_m^T b_m = \min, \quad \text{subject to} \quad S^T b_m = u_m
$$

(3.7.21)

have solutions,

$$
b_m = SF^{-1} u_m = Gu_m, \quad -M \leq m \leq M
$$

(3.7.22)
and putting these together as the columns of $B$, we obtain Eq. (3.2.31):

$$B = [..., b_m, ...] = G [..., u_m, ...] = GST = SF^{-1}S^T \quad (3.7.23)$$

In particular, the steady-state LPSM filter $b_0$ minimizes the NRR with the constraint vector $u = u_0 = [1, 0, \ldots, 0]^T$. This was precisely the problem first formulated and solved using Lagrange multipliers by Schiaparelli [36].

Similarly, the interpolating filter $b_1 = Gu_i$ of Eq. (3.6.10) can be thought of as the solution of the constrained minimization problem:

$$\mathcal{R} = b^T b = \min, \quad \text{subject to} \quad S^T b = u_i, \quad \text{where} \quad u_i = [1, t, t^2, \ldots, t^d]^T$$

### 3.8 Predictive Differentiation Filters

Going back to the polynomial fit of Eq. (3.6.9), that is,

$$\hat{y}_t = \sum_{i=0}^{d} c_i t^i = c^T u_t = y^T Gu_t = y^T b_t, \quad \text{where} \quad b_t = Gu_t, \quad (3.8.1)$$

we recall that the differentiation filters (3.2.24) were derived by differentiating (3.8.1) at $t = 0$, and therefore, they correspond to the center of the data vector $y$:

$$\hat{y}_t |_{t=0} = c_0 = b^T_0 y = g^T_0 y$$

$$\hat{y}_t |_{t=0} = c_1 = g^T_1 y$$

$$\hat{y}_t |_{t=0} = 2c_2 = g^T_2 y, \quad \text{etc.}$$

The first derivative at an arbitrary value of $t$ is given by:

$$\hat{y}_t = y^T b_t, \quad b_t = Gu_t$$

where the differentiation operation can be expressed as matrix multiplication:

$$u_t = \begin{bmatrix} 1 \\ t \\ t^2 \\ \vdots \\ t^d \end{bmatrix} \Rightarrow \dot{u}_t = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & d \end{bmatrix} \equiv D u_t \quad (3.8.2)$$

where $D$ is the $(d+1) \times (d+1)$ matrix with the sequence of numbers $\{1, 2, \ldots, d\}$ along its first subdiagonal and zeros everywhere else. Such a matrix can be constructed trivially in MATLAB, for example, by:

$$D = \text{diag}(1:d, -1);$$

### 3.8. Predictive Differentiation Filters

It follows that the first-order differentiation filter is $b_1 = GDu_t$. In particular, the differentiation filter at the sample point $t = m$ is $b_m = GDu_m$ and the corresponding estimated derivative:

$$\hat{y}_m = b^T_m y = u_m^T D^T G^T y, \quad -M \leq m \leq M \quad (3.8.3)$$

Stacking these together into a column vector, we obtain:

$$\hat{y} = SD^T G^T y = B^T y, \quad \text{where} \quad B = GDST = SF^{-1}DST \quad (3.8.4)$$

so that $B$ has the $b_m$ as columns. Higher-order derivatives correspond to higher powers of the matrix $D$, for example, $u_t = D^t u_t$, and so on, with the highest non-trivial power being $D^d$, because $D^{d+1} = 0$, or equivalently, because the elements of $u_t$ are monomials up to $t^d$. Therefore, the order-$i$ differentiation matrix will be:

$$B^{(i)} = SF^{-1}D^iT^T, \quad i = 0, 1, \ldots, d \quad (3.8.5)$$

Centering the data vector $y$ at time $n$ and denoting the $m$-th column of $B^{(i)}$ by $b^{(i)}_m$, we obtain the filtering equation for the $i$-th estimated/interpolated derivative:

$$\hat{y}^{(i)}_{n+m} = \sum_{k=-M}^{M} b^{(i)}_m(k) y_{n+k} = \sum_{k=-M}^{M} b^{(i)}_m(-k) y_{n-k} \quad (3.8.6)$$

We note that at the data-vector center $m = 0$, we have $b^{(i)}_0 = g$. For arbitrary $t$, we have $b^{(i)}_t = GD^t u_t$ and we obtain the estimated/interpolated derivative:

$$\hat{y}^{(i)}_{n+t} = \sum_{k=-M}^{M} b^{(i)}_t(k) y_{n+k} = \sum_{k=-M}^{M} b^{(i)}_t(-k) y_{n-k} \quad (3.8.7)$$

As in Eq. (3.6.15), the redefinition $h^{(i)}_t(k) = b^{(i)}_{M+t}(M-k)$ will result into a causal version of the predictive differentiator filter, with Eq. (3.8.7) transforming into:

$$\hat{y}^{(i)}_{n+t} = \sum_{k=0}^{N-1} h^{(i)}_{t}(k) y_{n-k} \quad \text{(causal predictive differentiator)} \quad (3.8.8)$$

One can easily obtain closed-form expressions for the differentiation filters $b^{(i)}_t(k)$ for $d = 0, 1, 2, 3, 4$ and arbitrary $M$, by replacing the variable $m$ in Eqs. (3.3.7)-(3.3.12) by the variable $t$ and differentiating $i$-times with respect to $t$. For example, for $d = 1, 2, 3, 4$, we differentiate Eqs. (3.6.12) once to get the first derivative:
3. Local Polynomial Filters

### 3.8. Predictive Differentiation Filters

The corresponding first- and second-order differentiation matrices will be:

$$
B = GD^1S^T = \frac{1}{35} \begin{bmatrix}
-27 & -17 & -7 & 3 & 13 \\
-6.5 & 1.5 & -3.5 & -8.5 & -13.5 \\
10 & 0 & -10 & -20 \\
13.5 & 8.5 & 3.5 & -1.5 & -6.5 \\
13 & -3 & 7 & 17 & 27
\end{bmatrix}, \quad D^1 = \begin{bmatrix}
0 & 0 & 0 \\
1 & 0 & 0 \\
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 2 & 0
\end{bmatrix}
$$

$$
B = GD^2S^T = \frac{1}{35} \begin{bmatrix}
10 & 10 & 10 & 10 & 10 \\
-5 & -5 & -5 & -5 & -5 \\
-10 & -10 & -10 & -10 & -10 \\
-5 & -5 & -5 & -5 & -5 \\
10 & 10 & 10 & 10 & 10
\end{bmatrix}, \quad D^2 = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
2 & 0 & 0 \\
2 & 0 & 0 \\
2 & 0 & 0
\end{bmatrix}
$$

The central columns agree with Eq. (3.2.25). The interpolating smoothing and first-order differentiation filters are given by:

$$
b_t = Gu_t = \frac{1}{35} \begin{bmatrix}
-3 & -7 & 5 \\
12 & -3.5 & -2.5 \\
17 & 0 & -5 \\
12 & 3.5 & -2.5 \\
-3 & 7 & 5
\end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \end{bmatrix} = \frac{1}{35} \begin{bmatrix}
-3 - 7t + 5t^2 \\
12 - 3.5t - 2.5t^2 \\
17 - 5t \\
12 + 3.5t - 2.5t^2 \\
-3 + 7t + 5t^2
\end{bmatrix}
$$

$$
b_t = GDu_t = \frac{1}{35} \begin{bmatrix}
-3 & -7 & 5 \\
12 & -3.5 & -2.5 \\
17 & 0 & -5 \\
12 & 3.5 & -2.5 \\
-3 & 7 & 5
\end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \end{bmatrix} = \frac{1}{35} \begin{bmatrix}
-7 + 10t \\
-3 - 5t \\
-3.5 - 5t \\
3.5 - 5t \\
7 + 10t
\end{bmatrix}
$$

where $b_t$ can be obtained either by the indicated matrix multiplication or by simply differentiating $b_t$ with respect to $t$.

The MATLAB function `1pdiff` implements the design of the differentiation matrices:

$$\hat{B} = 1pdiff(N, d, i);$$

Like `1psm`, it carries out a Gram-Schmidt QR-transformation on the monomial basis $S$ and constructs the $B^{(i)}$ by:

$$S = QR, \quad Q^TQ = I, \quad R = \text{upper triangular}$$

$$G = S(S^T S)^{-1} = QR^{-T}$$

$$B^{(i)} = GD^iS^T = Q(R^{-T} D^i R^T) Q^T$$

The `predictive` and `interpolating` differentiation filters $b^{(i)}$ are the minimum-norm solution of the under-determined linear system $S^T b = D^i u_t$, or, equivalently, the solution of the constrained minimization problem:

$$\mathcal{R} = b^T b = \text{min}, \quad \text{subject to} \quad S^T b = D^i u_t$$

The MATLAB function `1pinterp` implements the design of predictive and interpolating differentiation filters, essentially carrying out the operation $b = \text{pinv}(S^T) D^i u_t$:
3. Local Polynomial Filters

The bottom-left graph shows the noisy signal \( y_n \). In the bottom-right graph, the output (solid line) of the same differentiation filter applied to the noisy signal \( y_n \) is compared with the true noiseless differentiated signal \( \hat{s}_n \), as well as to the differenced signal \( \text{diff}(y)/\Delta t \).

The following MATLAB code illustrates the generation of the bottom-right graph:

```matlab
T = 10; L = 50; Dt = T/L; w = 2*pi/T; sigma = 0.1;

% noiseless signal
s = 0.5 - 0.5*cos(w*t);
noiseless signal
differentiated signal
true derivative
estimated derivative of s(t)

% derivative from the noisy signal
sd = 0.5*w*sin(w*t);

% first-order differentiation filter
N=3; d=3; B1 = lpdiff(N,d,1);

% estimated derivative from the noisy signal
x1 = lpfilt(B1,y)/Dt;

% differenced signal estimates the derivative
td = t(2:end);

plot(t,sd,'--', td,x1,'-');
```

3.9 Filtering Implementations

In smoothing a length-\( L \) signal block \( y_n, n = 0, 1, \ldots, L - 1 \), with a double-sided filter \( h_m, -M \leq m \leq M \), the output signal \( \hat{x}_n \) is given by the convolutional form:

\[
\hat{x}_n = \frac{1}{M+1} \sum_{m=0}^{M} h_m y_{n-m}, \quad -M \leq n \leq L + M - 1
\]  

The range of the output index \( n \) and the limits of summation in (3.9.1) are determined from the inequalities \(-M \leq m \leq M\) and \(0 \leq n - m \leq L - 1\) that must be satisfied by the indices of \( h_m \) and \( y_{n-m} \). However, only the subrange \( [\hat{x}_n, 0 \leq n \leq L - 1] \) is of interest since these output samples represent the smoothed values of the corresponding input samples \( [y_n, n = 0, 1, \ldots, L - 1] \). This is illustrated in Fig. 3.9.1.

The first and last \( M \) samples in the subrange \( 0 \leq n \leq L - 1 \) are still parts of the input-on and input-off transients. To clarify these remarks, we consider the case \( L = 8, M = 2 \). The full output (3.9.1) may be represented by the usual convolution matrix of the filter acting on the input signal block.
3. Local Polynomial Filters

3.9. Filtering Implementations

Dropping the first and last two outputs, we obtain the outputs in the subrange $0 \leq n \leq 7$:

\[
\begin{bmatrix}
\hat{x}_{-2} \\
\hat{x}_{-1} \\
\vdots \\
\hat{x}_0 \\
\hat{x}_1 \\
\vdots \\
\hat{x}_6 \\
\hat{x}_7 \\
\vdots \\
\hat{x}_9
\end{bmatrix} = \begin{bmatrix}
h_{-2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
h_{-1} & h_{-2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\vdots \\
h_0 & h_{-1} & h_{-2} & 0 & 0 & 0 & 0 & 0 & 0 \\
h_1 & h_0 & h_{-1} & h_{-2} & 0 & 0 & 0 & 0 & 0 \\
\vdots \\
h_5 & 0 & 0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} & 0 \\
h_6 & 0 & 0 & 0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} \\
\vdots \\
0 & 0 & 0 & 0 & 0 & h_2 & h_1 & h_0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & h_2 & h_1 & h_0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & h_2 & h_1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_2
\end{bmatrix}
\begin{bmatrix}
y_0 \\
y_1 \\
\vdots \\
y_5 \\
y_6 \\
\vdots 
\end{bmatrix} = H \begin{bmatrix}
\hat{x}_0 \\
\hat{x}_1 \\
\vdots \\
\hat{x}_6 \\
\hat{x}_7 \\
\vdots 
\end{bmatrix}
\tag{3.9.3}
\]

The first two and last two of these outputs are still transient and are being computed with only a subset of the filter coefficients, and therefore, may not adequately represent the corresponding smoothed values. This so-called "end-point problem" has been addressed repeatedly with a number of solutions.

One method that is widely used by the government to process census and business-data (e.g., the X12-ARIMA method) is to backcast and forecast $M$ estimated values at the beginning and end of the length-$L$ input block, so that $y_n$ is now defined over $-M \leq n \leq L - 1 + M$, and the desired output samples over the subrange $0 \leq n \leq L - 1$ will be steady-state outputs being computed with the full filter.

Another method is to use different filters for the first $M$ and last $M$ outputs. For example, one can take the outputs $\hat{y}_{n-M}$ of the LPSM filters $b_m(k)$ to estimate the initial and final $M$ transients, while using the central filter $b_0(k)$ for the steady-state outputs. Indeed, the first time index when one can use the steady-state filter $b_0(k)$ is $n = M$:

\[
\hat{x}_M = \hat{y}_M = \sum_{k=-M}^{M} b_0(k) y_{M+k}
\]

Instead of calculating the previous output $\hat{x}_{M-1}$ using the transient version of $b_0(k)$,

\[
\hat{x}_{M-1} = \sum_{k=(M-1)}^{M} b_0(k) y_{M-1+k}
\]

one could estimate $\hat{x}_M$ using $\hat{y}_{M+m}$ with $m = -1$, that is, using $b_{-1}(k)$, and using $b_{-2}(k), b_{-3}(k), \ldots, b_{-M}(k)$ for the other initial $M$ outputs:

\[
\hat{x}_{M-2} = \hat{y}_{M-2} = \sum_{k=-M}^{M} b_{-2}(k) y_{M+k}
\]

\[
\vdots
\]

\[
\hat{x}_0 = \hat{y}_{M-M} = \sum_{k=-M}^{M} b_{-M}(k) y_{M+k}
\]
3. Local Polynomial Filters

Similarly, one can use the filters \( b_m(k) \) for \( m = 1, 2, \ldots, M \) to calculate the last \( M \) smoothed outputs, starting with the last steady-state output at \( n = L - 1 - M \) and proceeding to the end \( n = L - 1 \):

\[
\hat{y}_{L-M} = \hat{y}_{L-M+1} = \sum_{k=-M}^{M} b_1(k)y_{L-1-M+k}
\]

\[
\hat{y}_{L-M+1} = \hat{y}_{L-M+2} = \sum_{k=-M}^{M} b_2(k)y_{L-1-M+k}
\]

\[
\vdots
\]

\[
\hat{y}_{L-1} = \hat{y}_{L-1+M} = \sum_{k=-M}^{M} b_M(k)y_{L-1-M+k}
\]

(3.9.5)

The procedure is straightforward. First construct the ordinary full convolution submatrix \( B(L-M+1:L+M,:); \) redefine lower-right corner as \( \hat{y}_n \) and the other columns, the reversed transient filters:

\[
\hat{y} = H \hat{y}
\]

The following example illustrates the computational steps for the input-on, steady, and input-off output samples, where we denoted \( b_{m,k} = b_m(k) \) for simplicity:

\[
\hat{x} = \begin{bmatrix}
\hat{x}_0 \\
\hat{x}_1 \\
\vdots \\
\hat{x}_2 \\
\hat{x}_3 \\
\vdots \\
\hat{x}_6 \\
\hat{x}_7
\end{bmatrix} = \begin{bmatrix}
\hat{y}_0 \\
\hat{y}_1 \\
\vdots \\
\hat{y}_2 \\
\hat{y}_3 \\
\vdots \\
\hat{y}_6 \\
\hat{y}_7
\end{bmatrix} = Hy
\]

In particular, for \( N = 5 \) and \( d = 2 \), the convolutional filtering matrix will be:

\[
\hat{x} = \begin{bmatrix}
\hat{x}_0 \\
\hat{x}_1 \\
\vdots \\
\hat{x}_2 \\
\hat{x}_3 \\
\vdots \\
\hat{x}_6 \\
\hat{x}_7
\end{bmatrix} = \frac{1}{35} \begin{bmatrix}
31 & 9 & -3 & -5 & 3 & 0 & 0 & 0 \\
9 & 13 & 12 & 6 & -5 & 0 & 0 & 0 \\
\vdots & & & & & & & \\
-3 & 12 & 17 & 12 & -3 & 0 & 0 & 0 \\
0 & 0 & -3 & 12 & 17 & 12 & -3 & 0 \\
0 & 0 & 0 & -3 & 12 & 17 & -3 & 12 \\
0 & 0 & 0 & -5 & 6 & 12 & 13 & 9 \\
0 & 0 & 0 & 3 & -5 & -3 & 9 & 31
\end{bmatrix} y
\]

(3.9.6)

The following MATLAB code illustrates this approach:

\[
y = B(:,:M) \ast x(1:N); \quad \text{first } M \text{ transient outputs}
\]

\[
for \ n = M+1:L-M, \quad \text{middle } L-2M \text{ steady-state outputs}
\]

\[
y = \{ ; B(:,M+1) \ast x(n-M:n-M); \} \quad \text{filtered by central column of } B
\]

\[
y = \{ ; B(:,M+2:N) \ast x(L-N+1:L); \} \quad \text{last } M \text{ transient outputs}
\]

These steps are implemented in the MATLAB function \texttt{lpfilt2}. A faster version is the function \texttt{lpfilt}, which uses MATLAB’s built-in filtering functions. Thus, three possible ways of computing the filtered output \( \hat{x} \) given a smoothing matrix \( B \) are as follows (assuming that \( y \) is a length-\( L \) column vector):

\[
x\_\text{hat} = \texttt{lpfilt}(B,L) \ast y; \quad \text{use } L \times L \text{ convolution matrix constructed from } B
\]

\[
x\_\text{hat} = \texttt{lpfilt2}(B,y); \quad \text{use directly the filtering equations (3.9.4)-(3.9.6)}
\]

\[
x\_\text{hat} = \texttt{lpfilt}(B,y); \quad \text{fast version using the function } \texttt{filtdb1}
\]

The function \texttt{lpfilt} internally calls the function \texttt{filtdb1}, which uses the built-in function \texttt{conv} to implement the FIR filtering by the steady-state double-sided central filter. The following code segment shows the essential part of \texttt{lpfilt}:

\[
x\_\text{hat} = \texttt{filtdb1}(\texttt{filip}(B(:,M+1)), \ y); \quad \text{filter with the central column of } B
\]

\[
x\_\text{hat}(1:M) = B(:,1:M) \ast y(1:N); \quad \text{correct the first } M \text{ transient outputs}
\]

\[
x\_\text{hat}(end-M+1:end) = B(:,M+2:N) \ast y(end-N+1:end); \quad \text{correct the last } M \text{ transient outputs}
\]
where the function \( \text{filtdbl} \) has usage:

\[
y = \text{filtdbl}(h,x); \quad \text{% filtering by double-sided FIR filter}
\]

The function \( \text{filtdbl} \) is essentially the ordinary convolution of the length-\((2M+1)\) filter \( h \) and the length-\( L \) signal \( x \), with the first \( M \) and last \( M \) output points discarded. The result is equivalent to that obtained using the convolution submatrix, as for example, in Eq. (3.9.3). We note, in particular, that the \( B \) matrix that gives rise to (3.9.3) is:

\[
B = \begin{bmatrix}
h_0 & h_1 & h_2 & 0 & 0 \\
h_{-1} & h_0 & h_1 & h_2 & 0 \\
h_{-2} & h_{-1} & h_0 & h_1 & h_2 \\
0 & h_{-2} & h_{-1} & h_0 & h_1 \\
0 & 0 & h_{-2} & h_{-1} & h_0 
\end{bmatrix}
\]

and contains the reversed filter \( h \) in the central column and the transient subfilters in the other columns.

There are other methods of handling the end-point problem, most notably Musgrave’s minimum-revision method that uses end-point asymmetric filters constructed from a given central filter \( h \). We will discuss it in detail in Sec. 9.8. Here, we note that the output of this method is a \( B \) matrix, which can be passed directly into the filtering function \( \text{lpfilt} \). The MATLAB function \( \text{minrev} \) implements Musgrave’s method:

\[
B = \text{minrev}(h,R); \quad \text{% Musgrave’s minimum revision asymmetric filters}
\]

where \( R \) is a scalar parameter to be explained in Sec. 9.8. The method is widely used in the X-11 method of seasonal adjustment and trend extraction.

**Example 3.9.1:** Schiaparelli was the first one to systematically pose and solve the minimum-NRR filtering problem. He gave the solution to many specific cases, such as filter lengths \( N = 5–13 \), and polynomial orders \( d = 3, 4 \).

Here, we reproduce the example from Schiaparelli’s paper on smoothing lunar observations, the signal \( y_n \) being a measure of the moon’s influence on atmospheric effects. Fig. 3.9.2 shows 30 noisy observations (one for each lunar day) and their smoothed versions produced with an LPFSM filter of length \( N = 13 \) and polynomial order \( d = 3 \) on the left, and \( d = 4 \) on the right (Schiaparelli’s case).

The central filters for the \( d = 3 \) and \( d = 4 \) cases are:

\[
b_0 = \frac{1}{143} \begin{bmatrix} -11, 0, 9, 16, 21, 24, 25, 24, 21, 16, 9, 0, -11 \end{bmatrix}
\]

\[
b_0 = \frac{1}{2431} \begin{bmatrix} 110, -198, -135, 110, 390, 600, 677, 600, 390, 110, -135, -198, 110 \end{bmatrix}
\]

The following program segment illustrates the computations:

\[
Y = \text{loadfile('schiaparelli.dat')}; \quad \text{% data file available in the OSP toolbox}
\]

\[
n = Y(:,1); \quad y = Y(:,14); \quad \text{% extract } n \text{ and } y_n \text{ from } Y
\]

\[
N=13; \quad d=3; \quad M=\text{floor}(N/2);
\]

\[
B = \text{lpfilt}(B,y); \quad \text{% filter noisy observations}
\]

\[
x0 = \text{filtdbl}(b0,y); \quad \text{% filter with } b_0 \text{ only}
\]

\[
\text{plot}(n,y,'.', n,x,'-', n,x0,'--');
\]

In the upper-right graph, a minimum-roughness, or minimum-\( R_n \) Henderson filter was used with length \( N = 65 \), polynomial order \( d = 3 \), and smoothing order \( s = 2 \). Such filters are discussed in Sec. 4.2. The resulting trend is noticeably smoother than that of the LPFSM filter on the upper-left.

The middle-left graph uses the SVD signal enhancement method, described in Chap. 15, with embedding order \( M = 10 \) and rank \( r = 2 \), with \( K = 40 \) iterations. The middle-right graph uses the Whittaker-Henderson smoothing method, discussed in Sec. 8.1, with smoothing order \( s = 2 \) and smoothing parameter \( \lambda = 10^4 \).

The lower left and right graphs use the Whittaker-Henderson method with the \( L_1 \) criterion with differentiation orders \( s = 2 \) and \( s = 3 \) and smoothing parameter \( \lambda = 10 \), implemented with the CVX package.\(^1\) The \( s = 2 \) case represents the smoothed signal in piece-wise linear form, and the \( s = 3 \) case, in piece-wise parabolic form. This is further discussed in Sec. 8.7.

\(^1\text{http://cvxr.com/cvx/}\)
3. Local Polynomial Filters

3.9. Filtering Implementations

cvx_begin
variable x(N)
minimize( sum_square(y-x) + la * norm(D*x,1) )
cvx_end

All methods adequately handle the end-point problem. Repeating the same filtering operation several times results in even smoother trend signals. For example, Fig. 3.9.4 shows the result of repeating the filtering operation two additional times. The following MATLAB code illustrates the generation of the left graph:

```matlab
N = 65; d=3; B = lpsm(N,d); x = y;
for i=1:3, x = lpfilt(B,x); end
figure; plot(n,y,: , n,x,- );
```

Fig. 3.9.4 Filtering repeated two additional times.

For the steady-state filters $B_k(\omega)$, filtering a total of $K$ times is equivalent to an overall filter $[B_K(\omega)]^K$, an operation which makes a flat passband even flatter and a small stopband even smaller. The properties of iterated smoothing by local polynomial filters has been studied by De Forest, Schoenberg, and Greville [67,83,86]. Fig. 3.9.5 shows the estimated derivatives (solid line) of the temperature signal obtained by filtering it with the LPSM derivative filters, and compares them with the ordinary differencing operation, $\text{diff}(y)$, in MATLAB notation. Clearly, differencing is simply too noisy to give any usable results.

The upper two graphs compute the first derivative of the input by $\dot{x} = \text{lpfilt}(B_1, y)$ with the differentiator matrix obtained from $B_1 = \text{lpdiff}(N, d, 1)$ with $N = 65$ and $i = 1$, and with $d = 1$ in the upper-left, and $d = 2$ in the upper-right graph. During the two periods of almost linear growth from 1910–1940 and 1970-2005, the derivative signal becomes an almost flat positive constant (i.e., the slope). During the other periods, the temperature signal has a very slow upward or downward trend and the derivative signal is almost zero. We note the flat end-points in the the case $d = 1$, which are due to the fact that the asymmetric derivative filters are the same at the end-points ranges as shown in the first equation of (3.8.9). The case $d = 2$ estimates the end-point derivatives better and possibly indicates a faster than linear growth in recent years.
The lower-left graph uses a minimum-$R_s$ derivative filter with $N = 65$, $d = 2$, and smoothness order $s = 3$, resulting in a noticeably smoother estimated derivative than the LPSM case (the $W$ input in $lpdiff$ is discussed in the next section.) Finally, the lower-right graph shows the second derivative computed with the filter $B_2 = lpdiff(N,d,i)$ with $i = 2$, and compares it with the second difference signal, $\text{diff}(\text{diff}(y))$, which is even more noisy than the first difference. The following MATLAB code illustrates the computations:

```matlab
d=1; i=1; B1 = lpdiff(N,d,i); 
plot(n, lpfilt(B1,y), n(2:end), diff(y),':'); % LPSM differentiation filters 

plot(n, lpfilt(B1,y), n(2:end), diff(y),':'); % upper-left graph 

d=2; i=1; B1 = lpdiff(N,d,i); 
plot(n, lpfilt(B1,y), n(2:end), diff(y),':'); % upper-right graph 

s=3; W = diag(hend(N,s)); 

\[ B = \text{lpdiff}(N,d,i,W) \] % Henderson weighting matrix 

d=2; i=2; B2 = lpdiff(N,d,i); 
plot(n, lpfilt(B1,y), n(3:end), diff(y,2),':'); % second derivative filters 

plot(n, lpfilt(B1,y), n(3:end), diff(y,2),':'); % lower-right graph 
```

Fig. 3.9.5 Differentiated temperature signal.

The second derivative is essentially zero, being consistent with piecewise linear trends. Derivative signals can also be estimated for the SVD and Whittaker-Henderson methods. Since the outputs $\hat{x}_n$ of these methods are smooth signals, the corresponding derivatives can be simply computed as the difference signals, $\text{diff}(\hat{x}_n)$, with comparable results as the local polynomial methods.