Filtering and Convolutions

Jack Xin (Lecture) and J. Ernie Esser (Lab) *

Abstract

Class notes on filtering, convolutions, eigenvalue/eigenvector, diagonalization, and z-transform.

1 Filtering

Filtering refers to linear transforms that change the frequency contents of signals. Depending on whether high (low) frequencies are attenuated, filtering process is called low (high) pass.

1.1 Low Pass Filtering

Example: two point moving average, recall the linear time invariant system:

\[ y(n) = \frac{[s(n) + s(n-1)]}{2}, \tag{1.1} \]

where \( s(n) \) is a periodic signal, \( s(-1) = s(N), N \geq 2 \). The output \( y \) is a smoother signal with higher frequency component damped and low frequency component maintained. To see this, let \( s(n) = \sin(2\pi f n/N) \), then:

\[
y(n) = \frac{1}{2} s(n) + \frac{1}{2} s(n-1) \\
= \frac{1}{2} \sin(2\pi f n/N) + \frac{1}{2} \sin(2\pi f (n-1)/N) \\
= \frac{1}{2} (1 + \cos(2\pi f/N)) \sin(2\pi f n/N) - \frac{1}{2} \sin(2\pi f/N) \cos(2\pi f n/N), \\
= A_1 \sin(2\pi f n/N) - A_2 \cos(2\pi f n/N), \tag{1.2}
\]

where:

\[ A_1 = \frac{1}{2} (1 + \cos(2\pi f/N)), \quad A_2 = \frac{1}{2} \sin(2\pi f/N). \]

*Department of Mathematics, UCI, Irvine, CA 92617.
At low frequency, \( f \sim 0 \), \( A_1 \sim 1 \), \( A_2 \sim 0 \), so
\[
y(n) \sim \sin(2\pi f n/N) = s(n).
\]
Low frequency input is almost preserved.

At high frequency, \( f \sim N/2 \) (near Nyquist frequency), \( A_1 \sim 0 \), \( A_2 \sim 0 \), so \( y(n) \sim 0 \). High frequency input is almost zeroed out.

![fft magnitude of h=[1/2,1/2,zeros(1,126)]](image.png)

Figure 1: Illustration of frequency domain response of two point moving average.

### 1.2 Convolution

A useful way to view filtering is by convolution. The first type of convolution is:

**Definition 1.1** (Circular Convolution). Let \( x \) and \( y \in C^N \). Circular convolution vector \( w = (w_1, w_2, \cdots, w_{N-1}) \in C^N \) is:

\[
w_m = \sum_{k=0}^{N-1} x_k y_{(m-k) \mod N},
\]

for \( 0 \leq m \leq N - 1 \), \( \mod \) is the remainder of \( m - k \) divided by \( N \). Notation is \( w = x \odot y \).
Figure 2: Illustration of frequency domain response of two point difference.

In Matlab, the command

$$\text{mod}(m - k, N),$$

computes $m - k \mod N$. For example, $\text{mod}(7,2)=1$, $\text{mod}(9,4)=2$, $\text{mod}(11,4)=3$.

Componentwise, $w_0$ is the inner product:

$$w_0 = [x_0 \ x_1 \ x_2 \ \cdots \ x_{N-1}] \cdot [y_0 \ y_{N-1} \ y_{N-2} \ \cdots \ y_1],$$

similarly

$$w_1 = [x_0 \ x_1 \ x_2 \ \cdots \ x_{N-1}] \cdot [y_1 \ y_0 \ y_{N-1} \ \cdots \ y_2],$$

and

$$w_2 = [x_0 \ x_1 \ x_2 \ \cdots \ x_{N-1}] \cdot [y_2 \ y_1 \ y_0 \ \cdots \ y_3],$$

and so on, until:

$$w_{N-1} = [x_0 \ x_1 \ x_2 \ \cdots \ x_{N-1}] \cdot [y_{N-1} \ y_{N-2} \ y_{N-3} \ \cdots \ y_0].$$

Then $w_m$ repeats these values, $w_m = w_{m \mod N}$.

In matrix form,

$$w = M_y x,$$
where \( x = [x_0 \ x_1 \ x_2 \ \cdots \ x_{N-1}] \), and

\[
M_y = \begin{bmatrix}
y_0 & y_{N-1} & y_{N-2} & \cdots & y_1 \\
y_1 & y_0 & y_{N-1} & \cdots & y_2 \\
y_2 & y_1 & y_0 & \cdots & y_3 \\
& \ddots & \ddots & \ddots & \ddots \\
y_{N-1} & y_{N-2} & y_{N-3} & \cdots & y_0
\end{bmatrix}
\]

The matrix \( M_y \) is called circulant matrix, notice that its row entries rotate around. Alternatively, each diagonal is a vector with identical entries.

In Matlab, the command

\[
\text{diag}(y_0 * \text{ones}(N,1), 0),
\]

produces the \( N \times N \) diagonal matrix with \( y_0 \) on the diagonal. The command

\[
\text{diag}(y_1 * \text{ones}(N-1,1), -1),
\]

gives the \( N \times N \) matrix with \( y_1 \) on the first subdiagonal. Similarly,

\[
\text{diag}(y_{N-1} * \text{ones}(N-1,1), 1),
\]

gives the \( N \times N \) matrix with \( y_{N-1} \) on the first super-diagonal. Next do

\[
\text{diag}(y_2 * \text{ones}(N-2,1), -2) + \text{diag}(y_{N-2} * \text{ones}(N-2,1), 2),
\]

to fill in the next sub(super)-diagonals, and so on. Adding these matrices up generates \( M_y \).

The two point moving average filter can be written (strictly speaking approximated) as circular convolution \( w = x \circ h \), \( h_0 = 1/2 \), \( h_1 = 1/2 \), \( h_k = 0 \), \( k = 2 : N - 1 \). In Matlab, do:

\[
N = 8;
\]

\[
M_h = \text{diag}(1/2 * \text{ones}(N,1), 0) + \text{diag}(1/2 * \text{ones}(N - 1,1), -1);
\]

\[
M_h(1,N) = 1/2,
\]

display shows that \( M_h \) has 1/2 on the main diagonal, first subdiagonal, and upper right corner. Now enter:

\[
x = [1, -1, 1, -1, 1, -1, 1, -1]'; w = M_h * x,
\]
we see \( w = \text{zeros}(8, 1) \). Oscillation in \( x \) is averaged out to zero.

Now re-define \( x \) and calculate \( w \):

\[
x = [1, 1, -1, 1, -1, 1, -1, -1]^\prime; \quad w = M_h * x
\]
gives:

\[
w = [0, 1, 0, 0, 1, 0, 0, 0],
\]
notice that averaging reduces sign changes in \( x \).

Another way to compute circular convolution is using the convolution-multiplication theorem. Thanks to periodicity of mod \( N \),

\[
DFT(w) = DFT(x) \cdot DFT(h).
\]
So we take \texttt{fft} to do regular multiplication in Fourier domain, then \texttt{ifft} back to recover \( w \).

For the above example, enter one line in Matlab:

\[
\text{real(}\text{ifft}((\text{fft}(x) \cdot \text{fft}(h))))\text{)};
\]
to get the same results on \( w \).

The effect of convolution with \( h \) in Fourier (frequency) domain is componentwise multiplication by vector \( \text{DFT}(h) \), which is easier to visualize by plotting. 
In Matlab,
define \( h = [1/2, 1/2, \text{zeros}(1, 126)] \),
plot(abs(fft(h))),
we get Figure 1 which shows a low pass filter (attenuating high frequencies).

1.3 High Pass Filtering

\textit{Example: two point difference}, consider the linear time invariant system:

\[
y(n) = [s(n) - s(n - 1)]/2,
\]
where \( s(n) \) is a periodic signal, \( s(-1) = s(N) \), \( N \geq 2 \). The output \( y \) is a less smooth signal with low frequency component damped and frequency component maintained.
In Matlab, define \( h = [1/2, -1/2, \text{zeros}(1, 6)] \), \( x = [1, 1, 1, 1, -1, -1, -1, -1] \); then:

\[
w = \text{real(}\text{ifft}((\text{fft}(x) \cdot \text{fft}(h))))\text{)} = [1, 0, 0, 0, -1, 0, 0, 0],
\]
notice that two point differencing zeros out identical entries, and maintains entries with sign change. The first 1 is due to the last -1 in $x$, the middle -1 stays because the entry in front of it equals 1.

To visualize the filtering effect in Fourier domain, define $h = [1/2, -1/2, \text{zeros}(1, 126)]$, plot(abs(fft(h))),

we get Figure 2 which shows a high pass filter.

### 1.4 Eigenvalues of Square Matrix

Eigenvector $v \neq 0$ is a special vector of a square matrix $A$ such that $A v$ lies in the same direction of $v$:

$$A v = \lambda v, \ v \neq 0,$$

where $\lambda$ is a complex number in general, so called eigenvalue. Rewriting (1.5) as

$$(A - \lambda I)v = 0,$$ (1.6)

shows that $A - \lambda I$ is degenerate (or its rows are not linearly independent), hence:

$$\text{determinant}|A - \lambda I| = 0.$$ (1.7)

Equation (1.7) is a polynomial in $\lambda$.

Example 1:

$$A = \begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix}$$

Equation (1.7) is:

$$\text{det} \begin{vmatrix} 1 - \lambda & 2 \\ 1 & -\lambda \end{vmatrix} = \lambda^2 - \lambda - 2 = 0.$$

Two roots (eigenvalues) are:

$$\lambda_1 = -1, \ \lambda_2 = 2.$$ 

When $\lambda = \lambda_1$, $v_1 - \lambda_1 v_2 = 0$, so the eigenvector is $v = (1, -1)$ (or any multiple of it). When $\lambda = \lambda_2$, the corresponding eigenvector is $(2, 1)$.

In Matlab, define:

$$A = [1 \ 2; \ 1 \ 0],$$

command: $[V, D] = \text{eig}(A)$ produces a diagonal matrix $D$ of eigenvalues and a full matrix $V$ whose columns are the corresponding eigenvectors.

$$[V, D] = \text{eig}(A)$$

6
shows that:

\[ D = \begin{bmatrix} 2 & 0 \\ 0 & -1 \end{bmatrix} \]

\[ V = \begin{bmatrix} 0.8944 & -0.7071 \\ 0.4472 & 0.7071 \end{bmatrix} \]

The first column vector of \( V \) is \( 0.4472*[2,1]' \), an eigenvector corresponding to eigenvalue 2. The second column vector of \( V \) is \(-0.7071*[1, -1]'\), an eigenvector corresponding to eigenvalue -1, in the same order as 2 and -1 appear on the diagonal of \( D \). Notice that eigenvectors are determined only up to a constant multiple.

**Example 2:**

\[ A = \begin{bmatrix} 1 & -1 \\ 1 & 0 \end{bmatrix} \]

The eigenvalue equation is:

\[ \lambda^2 - \lambda + 1 = 0, \]

giving two complex eigenvalues:

\[ \lambda_{1,2} = (1 \pm \sqrt{3}i)/2, \]

and eigenvectors \((\lambda_{1,2}, 1)\).

In Matlab, we get by the same command \([V,D]=\text{eig}(A)\) that:

\[ D = \begin{bmatrix} 0.5 + 0.8660i & 0 \\ 0 & 0.5 - 0.8660i \end{bmatrix} \]

\[ V = \begin{bmatrix} 0.3536 + 0.6124i & 0.3536 - 0.6124i \\ 0.7071 & 0.7071 \end{bmatrix} \]

### 1.5 Eigenvectors and Diagonalization

If eigenvalues are distinct, the corresponding eigenvectors are linearly independent. Consider \( \lambda_1 \neq \lambda_2 \), suppose there are two constants \( c_1 \) and \( c_2 \) such that:

\[ c_1 V_1 + c_2 V_2 = 0, \]  \hspace{1cm} (1.8)

where \( V_1 \) and \( V_2 \) are corresponding eigenvectors. Then multiplying \( A \) on both sides of (1.8) gives:

\[ c_1 \lambda_1 V_1 + c_2 \lambda_2 V_2 = 0. \]  \hspace{1cm} (1.9)
Equations (1.8)-(1.9) can be combined into:

$$\begin{bmatrix} 1 & 1 \\ \lambda_1 & \lambda_2 \end{bmatrix} \begin{bmatrix} c_1V_1 \\ c_2V_2 \end{bmatrix} = 0,$$

where $V_1$ and $V_2$ are row vectors. Because $\lambda_1 \neq \lambda_2$, the left square matrix in invertible (or has nonzero determinant), hence $c_1V_1 = 0$, $c_2V_2 = 0$, implying that $c_1 = c_2 = 0$ since $V_1 \neq 0$ and $V_2 \neq 0$. The general case follows in similar manner.

If matrix $A$ is order $N \times N$ with $N$ distinct eigenvalues, then putting the eigenvectors together form an invertible matrix $V = [V_1 \ V_2 \ \cdots \ V_N]$, here $V_j'$ are column vectors. It follows that

$$AV = VD,$$

where $D$ is a diagonal matrix with $\lambda_1, \cdots, \lambda_N$ on its diagonal. Taking inverse of $V$ gives:

$$V^{-1}AV = D,$$

which says that $A$ is similar to a diagonal matrix (diagonalizable). Diagonalization holds if and only if $A$ has $N$ linearly independent eigenvectors. Having $N$ distinct eigenvalues is a sufficient condition.

### 1.6 Diagonalization of Circulant Matrix

Define so called basic waveform $E_{N,m} \in C^N$ as:

$$E_{N,m} = [e^{2\pi im0/N}, e^{2\pi im1/N}, \cdots, e^{2\pi im(N-1)/N}],$$

the complex conjugate of the $m$-th row vector of DFT matrix, $m = 0, 1, \cdots, N-1$. Then $\text{DFT}(E_{N,m}) = N\delta(k - m)$. Let $w = h \odot E_{N,m}$, then

$$W = \text{DFT}(w) = \text{DFT}(h)(k) \ast \text{N} \delta(k - m) = \text{N} \text{DFT}(h)(m) \delta(k - m),$$

ifft back to get:

$$w = h \odot E_{N,m} = M_h E_{N,m} = \text{DFT}(h)(m) E_{N,m}.$$  \hspace{1cm} (1.13)

So $E_{N,m}$ is an eigenvector of circulant matrix $M_h$ with eigenvalue $\text{DFT}(h)(m)$. Because vectors $E_{N,m}$, $m = 0, 1, \cdots, N-1$, are orthogonal to each other, the circulant matrix can be diagonalized by an orthogonal matrix.

In Matlab, define $vm = \delta(k - m), k = 0, \cdots, N - 1$; or $vm = \text{zeros}(1, N), \text{vm}(m) = 1$.

$$E_{N,m} = (vm \ast \text{fft(eye(N))))'}.$$
2 Linear Convolution and z-Transform

Given \( x = x(n), n = 0, \cdots, N_1 - 1; y = (y(n)), n = 0, \cdots, N_2 - 1, \)

**Definition 2.1** (Linear Convolution).

\[
    w(n) = \sum_{m=m_1}^{m_2} x(m) y(n - m),
\]

where \( m_1 = \max\{0, n + 1 - N_2\}, \ m_2 = \min\{N_1 - 1, n\}. \) The linear convolution is denoted by \( w = x * y. \)

Let us zero-pad \( x \) and \( y \) to length \( N = N_1 + N_2 - 1. \) The zero padded versions are \( x_a \) and \( y_a \) respectively. Then (2.1) can be expressed as:

\[
    w(n) = \sum_{m=0}^{n} x_a(m) y_a(n - m), \quad 0 \leq n \leq N - 1.
\]

Zero-padded vectors have the same length. We calculate

\[
    x_a \odot y_a = \sum_{m=0}^{N-1} x_a(m) y_a(n - m \ mod N)
    = \sum_{m=0}^{n} x_a(m) y_a(n - m) + \sum_{m=n+1}^{N-1} x_a(m) y_a(n - m + N).
\]

For the product term in the second sum to be nonzero, the lengths \( N_1 \) and \( N_2 \) of \( x \) and \( y \) satisfy the constraints:

\[
    \begin{align*}
    n + 1 & \leq m \leq N_1 - 1, \quad (2.4) \\
    0 & \leq n - m + N, \quad (2.5) \\
    n - m & \leq -N_1. \quad (2.6)
    \end{align*}
\]

The last inequality (2.6) can be seen as follows. If \( n - m > -N_1, \) then \( n - m + N = N - N_1 = N_2 - 1, \) so \( y_a(n - m + N) = 0. \) The two inequalities (2.5)-(2.6) combine into:

\[
    N_1 + n \leq m \leq N + n,
\]

which has no overlap with (2.4). It follows that

\[
    x_a \odot y_a(n) = x * y(n), \quad 0 \leq n \leq N - 1,
\]

9
which allows one to use fft to calculate linear convolution.

The linear convolution can be computed by multiplying two polynomials with coefficients from \( x(n) \) and \( y(n) \). Define the z-polynomials:

\[
X(z) = \sum_{n=0}^{N_1-1} x_n z^{-n}, \quad (2.9)
\]
\[
Y(z) = \sum_{n=0}^{N_2-1} y_n z^{-n}, \quad (2.10)
\]

they are called z-transforms of \( x \) and \( y \) respectively. The z-transform of linear convolution \( w \) equals \( X(z)Y(z) \) which has \( N_1 + N_2 - 1 \) coefficients.

In Matlab, \( w = \text{conv}(x, y) \).

Example: compute \( w = x \ast y \), \( x = [2, 3] \), \( y = [1, -4, 5] \),

\[
w = \text{conv}(x, y) = [2, -5, -2, 15]. \quad (2.11)
\]

Here \( N_1 = 2 \), \( N_2 = 3 \), \( N = N_1 + N_2 - 1 = 4 \), so zero-pad as:

\[
x_a = [x, 0, 0], \quad y_a = [y, 0].
\]

\[
w = x_a \odot y_a = \text{real}(\text{ifft}(\text{fft}(x_a) \ast \text{fft}(y_a)))) = [2, -5, -2, 15].
\]

Finally:

\[
X(z) = 2 + 3z^{-1},
\]
\[
Y(z) = 1 - 4z^{-1} + 5z^{-2},
\]
\[
X(z)Y(z) = 2 - 5z^{-1} - 2z^{-2} + 15z^{-3},
\]
giving again (2.11).

### 2.1 Circular Convolution and z-Transform

Let \( x, y \) be \( N \)-dimensional vectors with complex components, \( w = x \odot y \). Let \( X(z) \), \( Y(z) \), \( W(z) \) be their z-transforms. Then:

\[
W(z) = X(z)Y(z) \mod z^{-N}. \quad (2.12)
\]

Here \( \mod z^{-N} \) means that \( z^n \) and \( z^{n-N} \) are regarded as the same, for any integer \( n \).
Example: $x = [1, 1, -1]$, $y = [1/2, 1/2, 0]$, $x \circ y = [0, 1, 0]$ (check it in Matlab). On the other hand, $X(z) = 1 + z^{-1} - z^{-2}$, $Y(z) = 1/2 + 1/2z^{-1}$, direct multiplication shows:

$$X(z)Y(z) = 1/2 + z^{-1} - 1/2z^{-3} = z^{-1} \mod z^{-3};$$

also producing vector $[0, 1, 0]$.

A simple way to see (2.12) other than direct verification by multiplication is as follows. Let $\omega_k = e^{2\pi i k/N}$, the k-th root of unity. Then by definition of DFT, $W(\omega_k)$ is DFT($w$), similarly for $x$ and $y$, so:

$$W(\omega_k) = X(\omega_k)Y(\omega_k),$$

$0 \leq k \leq N - 1$. Define:

$$P(z) = X(z)Y(z) \mod z^{-N}.$$

Because $\omega_k^m = \omega_k^n$ if $m = n \mod N$,

$$P(\omega_k) = X(\omega_k)Y(\omega_k),$$

so $P(\omega_k) = W(\omega_k)$. It follows that the degree $N - 1$ polynomial $z^{N-1}(W(z) - P(z))$ vanishes at $\omega_k$, $k = 0, 1, \cdots, N - 1$, or has $N$ distinct roots, so it must be identically zero, or $W(z) \equiv P(z)$.

Formula (2.12) yields an algorithm for computing circular convolution by linear convolution. The linear convolution $u = x \ast y$ corresponds to coefficients in $X(z)Y(z)$, which contains powers $z^0, z^{-1}, \cdots, z^{-(N-1)}$. To get circular convolution, we add the coefficients of $z^{-n}$ and $z^{-n-N}$, for $n = 0, 1, \cdots, N - 2$,

$$w(n) = u(n) + u(n + N),$$

while $w(N - 1) = u(N - 1)$.

2.2 Bilateral z-Transform and Properties

The bilateral z-transform of a sequence $x(n)$ is:

$$X(z) = \sum_{n=-\infty}^{+\infty} x(n) z^{-n}, \quad (2.13)$$

for $z$ in the domain of convergence.

Example 1: $x(n) = \delta(n)$, then $X(z) = 1$. 

11
Example 2: Let \( a \) be a complex number, \( x(n) = a^n \) if \( n \geq 0 \), \( x(n) = 0 \) if \( n < 0 \). Then:

\[
X(z) = \sum_{n=0}^{+\infty} a^n z^{-n} = \sum_{n=0}^{+\infty} (az^{-1})^n = \frac{1}{1 - a z^{-1}},
\]

if \(|z| > |a|\).

Some useful properties are:

- **Linearity:** \( w(n) = a x(n) + b y(n) \implies W(z) = a X(z) + b Y(z) \), for any two complex numbers \( a \) and \( b \).
- **Time shift:** \( w(n) = x(n - m) \implies W(z) = z^{-m} X(z) \), for any integer \( m \).

\[
\sum_n x(n - m) z^{-n} = \sum_n x(n) z^{-(n+m)} = z^{-m} X(z).
\]

So \( z^{-1} \) means delay by 1.

- **Time reversal:** \( w(n) = x(-n) \implies W(z) = X(z^{-1}) \).
- **Multiplication:** \( w(n) = a^n x(n) \implies W(z) = X(z/a) \), if \( a \neq 0 \).
- **Differentiation:** \( w(n) = nx(n) \implies W(z) = -z \frac{dX(z)}{dz} \).
- **Linear convolution:** \( w(n) = (x * y)(n) \implies W(z) = X(z) Y(z) \).

Example 3: The moving average linear system:

\[
y(k) = \frac{x(k) + x(k - 1)}{2},
\]

is z-transformed into:

\[
Y(z) = \left(\frac{1}{2} + \frac{1}{2} z^{-1}\right) X(z).
\]

The function \( (1/2 + 1/2 z^{-1}) \) is called transfer function, denoted by \( H(z) \).

A more general linear shift invariance filtering system is of the form:

\[
y(k) + a_2 y(k - 1) + \cdots + a_{m+1} y(k - m) = b_1 x(k) + b_2 x(k - 1) + \cdots + b_{n+1} x(k - n), \quad (2.14)
\]

assuming zero initial condition (causality). So in time:

\[
y(1) = b_1 x(1),
\]
\[ y(2) = b_1 x(2) + b_2 x(1) - a_2 y(1) \]
\[ y(3) = b_1 x(3) + b_2 x(2) + b_3 x(1) - a_2 y(2) - a_3 y(1) \]

and so on. The transfer function is:
\[
H(z) = \frac{b_1 + b_2 z^{-1} + \cdots + b_{n+1} z^{-n}}{1 + a_2 z^{-1} + \cdots + a_{m+1} z^{-m}}.
\]

When \( m = 0 \) (\( H(z) \) is a polynomial in \( z^{-1} \)), the filter is called Finite Impulse Response (FIR). When \( m > 0 \) (\( H(z) \) is a rational function in \( z^{-1} \)), the filter is called Infinite Impulse Response (IIR). When \( n = 0 \), the filter is called all-pole recursive or autoregressive (AR).

In Matlab, define:
\[
b = [b_1, b_2, \ldots, b_{n+1}], \quad a = [1, a_2, \ldots, a_{m+1}],
\]
then the output of filtering is given by:
\[
y = \text{filter}(b, a, x).
\]

For the 2-point moving average filter: \( a = 1; \ b = [1/2, 1/2] \).

## 3 Matlab Projects

1. Write a Matlab function expressing circular convolution by linear convolution command `conv`.

   function \( z = \text{cconv}(x, y) \);
   \%
   \% input vectors \( x, y \) are two vectors to be convolved
   \%
   \% output vector \( z \): result of circular convolution
   \% \text{N=length}(x);
   \% if (\text{length}(y) = \text{N});
   \% error('vectors of unequal lengths'); end
   \% \text{z=conv(reshape(x,1,N),reshape(y,1,N))};
   \% \text{z=z(1:N)+[z(N+1:2*N-1),0]};
   \%
   \% Test on some vectors \( x \) and \( y \), and compare with real(ifft((fft(x).*fft(y)))).

2. Let \( x \) be the step function defined as \( x = \text{zeros}(1, 128) \), \( x(64 : 128) = 1 \). Filter it 20 times by the moving average filter as follows:
\[ a = 1; \ b = [1/2, 1/2]; \] plot(x), hold on;
for \( i=1:20 \), \( y=\text{filter}(b,a,x) \); \( x=y \); end;
plot(y,’r’).

3. Define one second duration signal with two tones at (3, 40) Hz sampled at 100 Hz:
\[ \text{fs}=100; \]
\[ \text{t}=0:1/\text{fs}:1; \]
\[ \text{x} = \sin(2\pi \text{t} \* 3) + 0.25 \* \sin(2\pi \text{t} \* 40); \]

Next define 10-point moving average filter:
\[ \text{b} = \text{ones}(1,10)/10; \]
\[ \text{y} = \text{filter}(b,1,x); \]
plot(t,x,t,y,’r’).

Comparing output \( y \) with input \( x \), we notice that 40 Hz tone is removed, however there is a delay in \( y \) (phase distortion). Matlab double filter function “\text{filtfilt}” corrects the phase distortion, which in the z-domain proceeds as follows:
\[ X(z) \rightarrow X(z)H(z) \rightarrow X(1/z)H(1/z) \quad \text{time reverse}, \]

pass through filter again:
\[ \rightarrow X(1/z)H(1/z)H(z), \]

time reverse again:
\[ \rightarrow X(z)H(z)H(1/z). \]

If \( z = e^{i\omega} \), then \( H(z)H(1/z) = |H(e^{i\omega})|^2 \) does not alter phase.
\[ \text{yy} = \text{filtfilt}(b,1,x); \] plot(t,x,t,y,’-’,t,yy,’:’);

compare \( yy \) and \( x \).

References