Biomedical Signal Processing

- Filters: Digital Filters -

2021. 5. 20

Prof: Boreom Lee

Introduction

- Filtering in digital domain
  - Currently available fast processors & dedicated digital signal processing (DSP) hardware
  - Most biomedical instruments – at least some digital filter operations
  - More flexible filtering:
    - A different frequency response – obtainable with a simple change of parameters instead of an alteration of the hardware
  - Allowing arbitrary attenuation of undesired frequencies in the frequency domain representation of a signal
    - Limitation: manipulations in the frequency domain → serious oscillations in the filter's response as well as unwanted transients in the time domain (Appendix 11.1)

IIR and FIR Digital Filters

- Continuous time LTI systems:
  - A rational function – used to describe the input/output relationship in the time domain, frequency domain, and the s- (Laplace) domain
- Discrete time:
  - The same approach – used for the sampled function using the z-domain (instead of s-domain)
    - Time-delayed values (instead of derivatives) – used to characterize the system evolution
  - A system with input x(n) and output y(n) (n=0,1,2,…):

$$\sum_{k=0}^{M} a_k y(n-k) = \sum_{k=0}^{N} b_k x(n-k)$$

    - a_k & b_k: the parameters determining the filter's characteristic

IIR and FIR Digital Filters

- Discrete time:
  - A system with input x(n) and output y(n) (n=0,1,2,…):
    - Transfer function by ZT:

$$(1)\quad Y(z)\sum_{k=0}^{M} a_k z^{-k} = X(z)\sum_{k=0}^{N} b_k z^{-k} \rightarrow H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N} b_k z^{-k}}{\sum_{k=0}^{M} a_k z^{-k}}$$

      - Dividing the numerator and denominator by a_0(the coefficient of y(n)):

$$(2)\quad H(z) = \frac{\sum_{k=0}^{N} b_k z^{-k}}{1 + \sum_{k=1}^{M} a_k z^{-k}}$$

  - Infinite impulse response (IIR) filter
    - If a filter output depends on the previous output (i.e., $a_k \neq 0$ for $k \geq 1$)
      - The response to an impulse at n=0 never completely disappears
      - Continues to reverberate through the system indefinitely
    - The impulse response continues forever ($n \rightarrow \infty$)
  - Finite impulse response (FIR) filter
    - $a_k = 0$ for $k \geq 1$ → the output depends on a finite set of input terms
    - The impulse response - finite

IIR and FIR Digital Filters

- Factorization of the polynomials in the numerator and denominator in equation (1)
  - The rational function – fully characterized by a constant gain factor (K) plus the zeros of the numerator (z_k) and the zeros of the denominator, poles (p_k):

$$H(z) = \frac{Y(z)}{X(z)} = K\frac{(z^{-1} - z_1)(z^{-1} - z_2)\ldots(z^{-1} - z_n)}{(z^{-1} - p_1)(z^{-1} - p_2)\ldots(z^{-1} - p_m)}$$

    - H(z) – undefined at the poles ($\because$ an output/input ratio explodes toward ∞)
    - A stable system – must not have any poles in the region of convergence (ROC) (Appendix 9.2)
  - IIR filter equation – includes poles → potentially unstable
  - FIR filter equation – no poles → always stable

AR, MA, and ARMA Filters

- Alternative classification of digital filters – based on the type of algorithm associated with the filter

1. Autoregressive (AR) filter
   - Dependence on previous output → characterized by an infinite impulse response (IIR)
   - Example: $y(n) = Ay(n-1) + By(n-2)$
     - A and B: constants
     - Potentially unstable depending on the coefficients
2. Moving average (MA) filters
   - Only dependent on the input → finite impulse response (FIR)
   - Example: $y(n) = \frac{x(n) + x(n-1) + x(n-2)}{A}$ (A: a constant)
3. The combination of AR and MA = ARMA filter
   - Dependent on both previous output and input → IIR (∵ previous output – involved)
   - Example: $y(n) = Ay(n-1) + Bx(n) + Cx(n-1)$ (A, C, and B: constants)

❖ The AR, MA, and ARMA classifications – overlap with the IIR and FIR terminology

Frequency Characteristic of Digital Filters

- The ZT of the output/input ratio (the transfer function) – closely related to the system's frequency response
- Transfer function of a digital filter
  - Variable z – represent $e^{s\tau}$ ($s=\sigma+j\omega$, complex variable)
- Frequency response
  - Interested in the imaginary, frequency-related part of the transfer function
  - Substituting $e^{j\omega\tau}$ for z in the transfer function → the frequency response of a digital filter

Frequency Characteristic of Digital Filters

- Example: 3-point smoothing (MA, FIR) filter with A=3
  - ZT of the time domain equation: $$Y(z) = \frac{X(z)(1+z^{-1}+z^{-2})}{3}$$
  - Transfer function: $$\frac{Y(z)}{X(z)} = H(z) = \frac{(1+z^{-1}+z^{-2})}{3}$$
  - Multiplying the numerator and denominator by z, substituting $e^{j\omega\tau}$ for z, and using Euler's relation for $\cos(\omega\tau)$:

$$H(z) = \frac{(z+1+z^{-1})}{3z} \rightarrow H(j\omega) = \frac{(e^{j\omega\tau}+e^{-j\omega\tau}+1)}{3e^{j\omega\tau}} = \frac{e^{-j\omega\tau}}{3}[2\cos(\omega\tau)+1]$$

    - τ = sample interval → $1/\tau$ = sample rate, $1/(2\tau)$ = Nyquist frequency for the filter in Hz, and π/τ = Nyquist frequency for the filter in rad/s
    - Bode plot – constructed for the values of ω ranging btw 0 and π/τ rad/s

Frequency Characteristic of Digital Filters

- Example: 3-point smoothing (MA, FIR) filter with A=3
  - Behaves as a low-pass filter
  - Stable FIR filter → but amplitude ratio of the frequency characteristic – far from ideal d/t a large side lobe above $2\pi/3 \approx 2.1$ rad/s

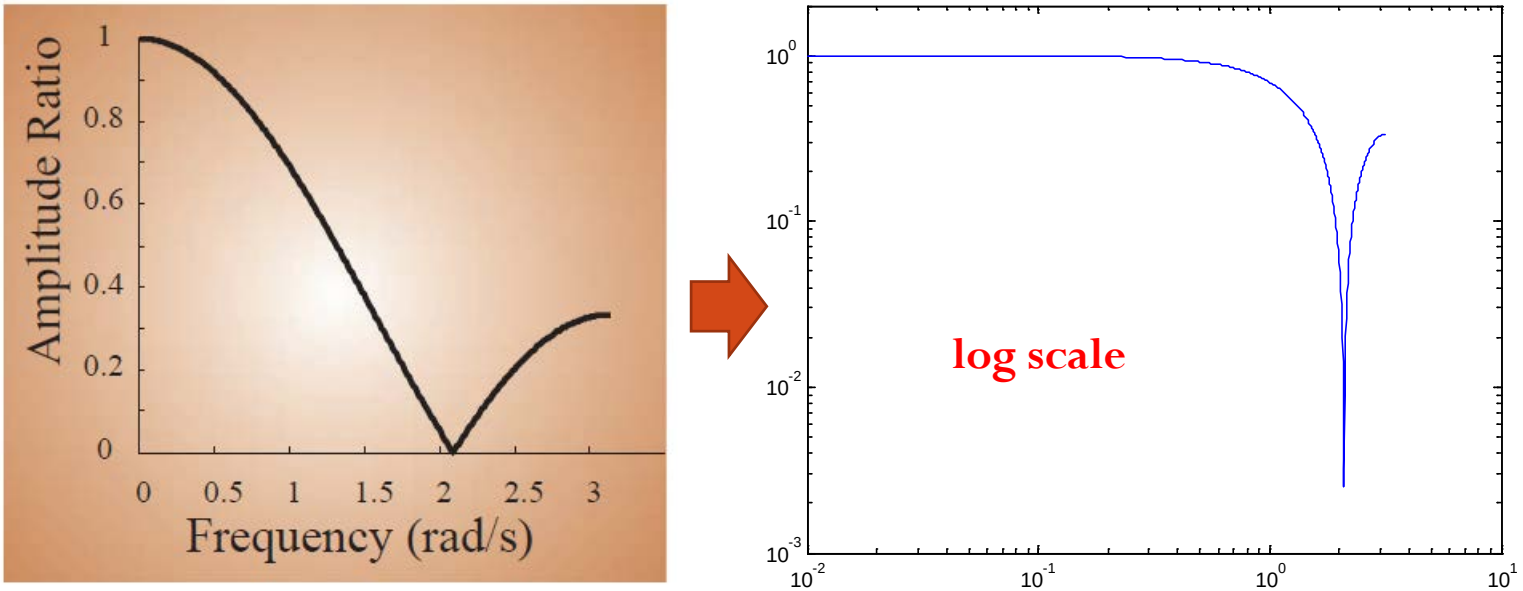


Figure 13.1 Frequency characteristic of a 3-point smoothing filter. The amplitude ratio is plotted against the frequency.

MATLAB Implementation

- MATAB "Signal Processing"Toolbox
- ***filter*** command: y=filter(B,A,x) – direct form II transposed implement
  - a_k (A) and b_k (B) coefficients for the digital filter operation on the input vector (e.g., X) → the result – placed in another vector (e.g.,Y)
  - Vector A and B – contain the a_k and b_k coefficients
  - Example: y(n) - y(n-1) + 0.8y(n-2) = x(n) → A=[1,-1,0.8] & B=[1]
    - In most cases: A & B coefficients – not known explicitly → start from a filter specification
- Example: implementation of a band-pass filter that passes btw 1~30 Hz
  - Using Butterworth filter
  - Hard way: deriving the filter's transfer function and translating this into the discrete domain
  - MATLAB: using ***butter*** command → coefficients – determined more easily

MATLAB Implementation

- Example: implementation of a band-pass filter that passes btw 1~30 Hz
  - Assumption: sample rate = 400 Hz → Nyquist frequency = 200 Hz → bandwidth = [1/200, 30/200]
  - [b,a] = butter(2, [1/200, 30/200]) → the desired coefficients for a second-order filter
    - The coefficient – used in the filter command to band-pass a signal sampled at 400 Hz btw 1 and 30 Hz
  - ❖ [b,a] = butter(6, [(60-5)/200, (60+5)/200], 'stop') → a set of coefficients for a sixth-order band-reject filter btw 55 and 65 Hz
    - Attenuate a 60-Hz noise component
- ***freqz*** command
  - Bode plot construction from the filter characteristic in the z-domain
    - On the basis of the coefficients A and B
  - freqz(b, a, 100, 400)
    - The precision of the calculation = 100 points
    - The sample frequency = 400

MATLAB Implementation

- ***impz*** command
  - Associated impulse response
  - impz(b,a,100) → impulse response of the first 100 points

- Example: an epoch of EEG sampled at 256 Hz with a large 60-Hz component → a 60-Hz band-stop filter (notch filter) (Fig. 13.2)
  - [b, a] = butter(6, [(60-5)/128, (60+5)/128], 'stop')
  - freqz(b,a,100,256), figure; impz(b,a,100)
  - figure; plot(eeg); hold, eegf=filter(b,a,eeg); plot(eegf,'r')

< Note >

- The success of a 60-Hz band reject filter – not an excuse to record poor quality data with lots of hum

  (1) No ideal filters → attenuation is never complete

  (2) 50/60-Hz notch filters – a tendency to produce oscillatory artifacts at discontinuities in the signal

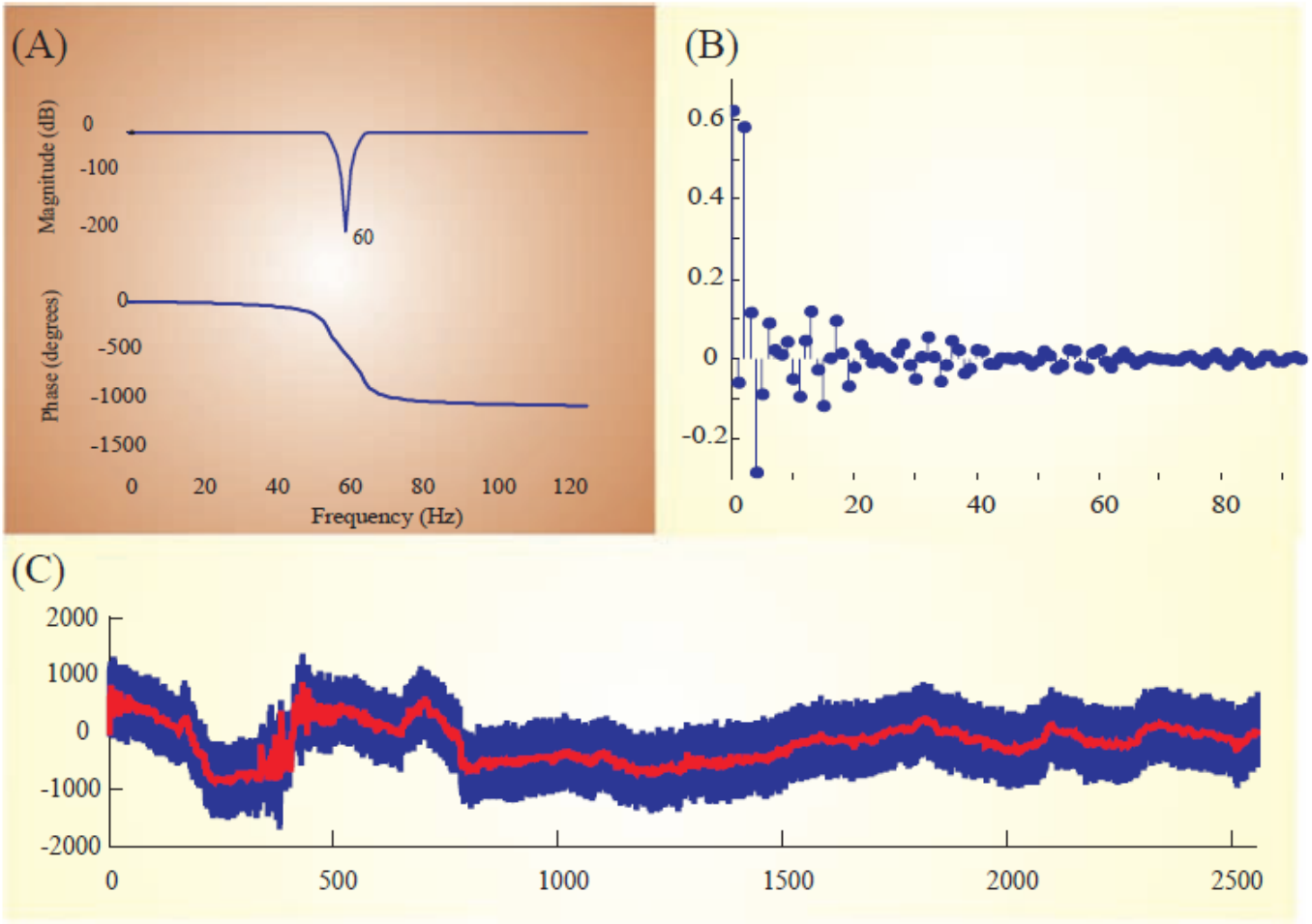


Figure 13.2 Example of a band-reject filter to remove 60-Hz hum. This type of filter is often called a notch filter. The graphs in (A) represent the Bode plot (the filter characteristic), and (B) shows the first part of the filter's impulse response. In this example, the full impulse response cannot be shown because this is an IIR-type filter. (C) EEG with 60-Hz noise (blue) and the trace that was filtered using the notch filter (red) superimposed.

Filter Types

- Butterworth filter: not ideal
  - The transition band – fairly wide
  - The phase response – frequency dependent
- No ideal filter → compromise needed in the approach to the ideal filter characteristic depending on each application
  - Some cases: strong attenuation of noise required
  - Other cases: accurately measure delays wanted → the phase response – critical

➔In the real world: a trade-off btw a small transition band (a steep roll-off) and a favorable (flat) phase response

Filter Types

- Different filter types realizing different compromises – available in MATLAB
  - Butterworth filter: a good compromise
    - Both a reasonable roll-off and phase response
    - Magnitude response $|H(j\omega)|$ – flat in the pass band & monotonic overall
  - Bessel and Elliptic filter types
    - At the extreme end of the trade-off scale
    - Either a good phase response or a steep roll-off, respectively
    - Bessel filters: an almost constant delay for the frequencies across the pass band → preserve the wave shape of the filtered signal in the time domain
    - Elliptic filter: ripple in both pass & stop bands of magnitude response (Fig. 10.1)
  - Chebyshev filter
    - Type I: ripple in the pass band & flat stop band
    - Type II: flat pass band & ripple in the stop band (the opposite)
  - ❖ Increased roll-off of the Chebyshev and Elliptic filters – at the cost of ripple in their magnitude response curves $|H(j\omega)|$

Filter Types

Table 13.1 Summary of Roll-off and Phase Characteristics of Different Filter Types That Are Available in the MATLAB Signal Processing Toolbox

Filter type	MATLAB command	Roll-off	Phase response
Bessel	besself	−	++
Butterworth	butter	±	±
Chebyshev	cheby1, cheby2	+	−
Elliptic	ellip	++	−−

Filter Bank

- Previous filters: a single input and single output
- Some applications: beneficial to look at the signal in a set of frequency bands
- A filter bank
  - A set of filters with desired frequency response → applied in parallel to the same signal
  - Preferred if one want to explore the signal's frequency content or detect features associated with certain frequency components
  - The basis for spectrogram and scalogram representation of a time series (ch. 16, Fig. 16.5)

Filter Bank

- Cochlear implant
  - An interesting biomedical application for filter bank
  - Mimics the cochlea – by separating the input (sound transduced into an electrical signal by a sensitive microphone) into separate spectral components
  - Physiology of cochlea
    - The bottom part (base) – more sensitive to high-frequency components
    - The top (apex) – more sensitive to low-frequency oscillations
  - The filter bank in the implant device
    - Mimics this normal cochlear operation & stimulates sensors connected to auditory nerve in a similar pattern to a normal cochlea
    - Only works for patients whose auditory system downstream of the cochlea is intact

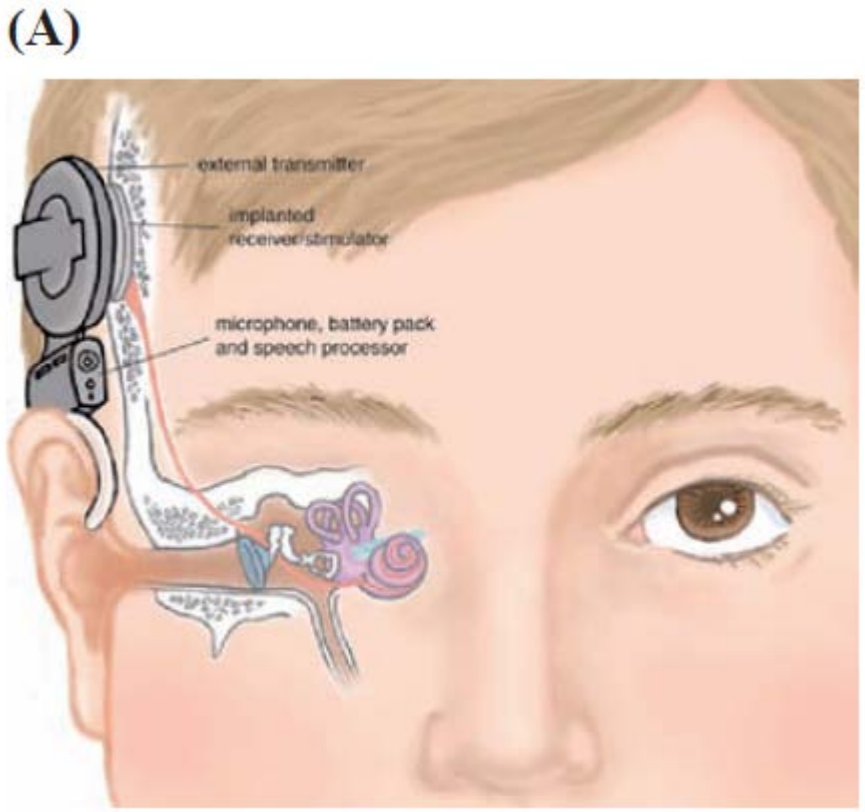
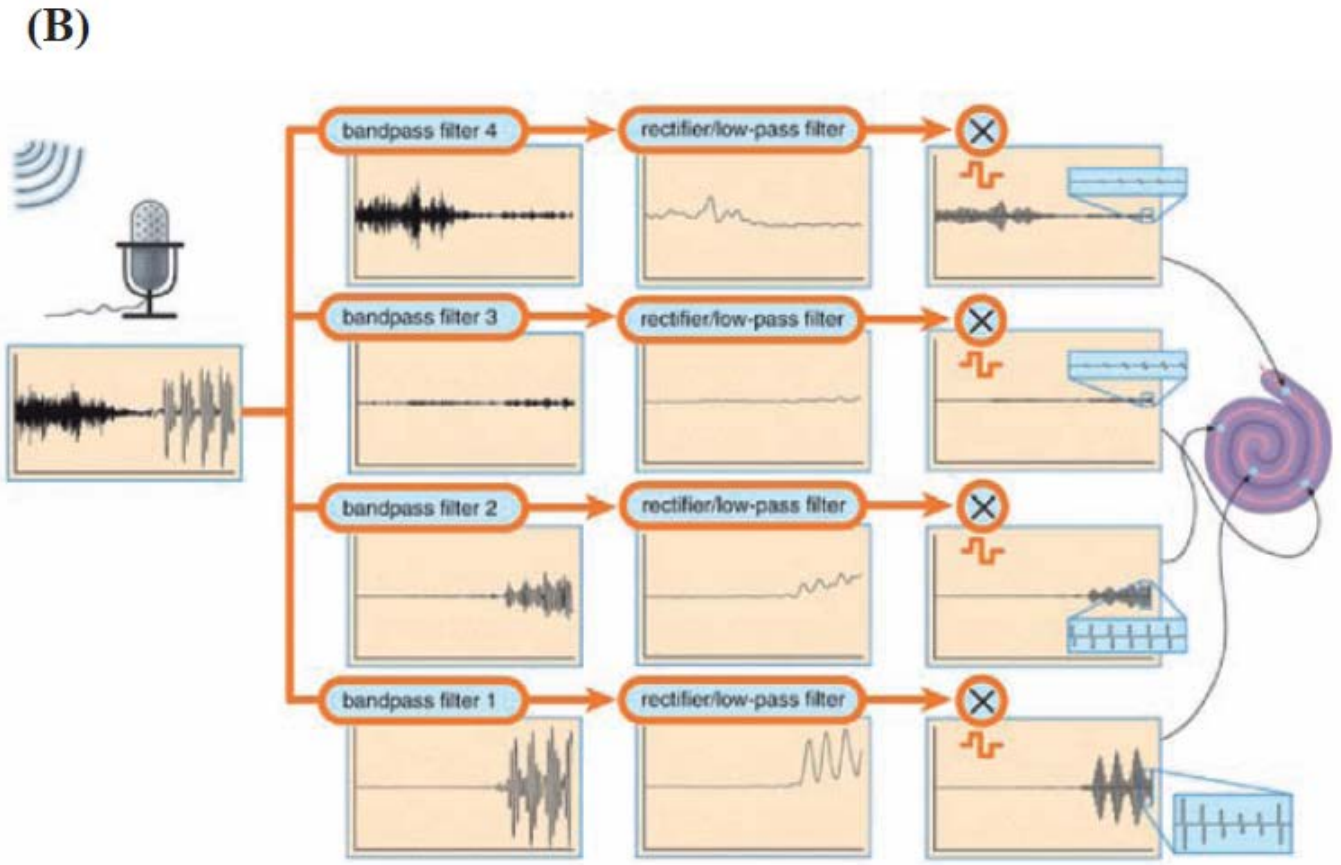


Figure 13.3 (A) Cochlear implants have five main components, only two of which are inside the body. A microphone above the ear senses sound waves, which are directed to a small computer behind the ear. The computer transforms the signals into a coded stimulus that must be delivered to a set of electrodes that are implanted in the cochlea. These stimulus signals are (via an external transmitter) transmitted through the skin to an implanted receiver, which converts them to electrical stimuli for the implanted electrodes. These stimuli excite the auditory nerve.

Figure 13.3 (B) A diagram of how a filter bank is used to decompose a complex sound signal for the syllable "sa." Band-pass filters 1 to 4 (left column of panels) each pass a specific frequency component of the compound signal in the left panel. Filter 1 passes the lowest frequency components, and filter 4 passes the highest ones. A set of rectifying low-pass filters (middle column of panels) subsequently create an envelope for the activity in each of the frequency bands. This envelope signal is finally transformed into a train of biphasic pulses (right column of panels) that can be used to stimulate a specific location in the cochlea. The high-frequency components stimulate the base of the cochlea, and the low frequencies stimulate nerve fibers more toward the apex. Used with permission from Dorman MF and Wilson BS (2004), The design and function of cochlear implants. *American Scientist* 92: 436–445.



Filters in the Spatial Domain

- Processing techniques on time series x(t) – easily translated into the spatial domain
  - An intensity image as a function of two spatial dimensions I(x,y)
  - The time parameter t – replaced by a spatial variable x, y, and z
- Spatial filters
  - Used to remove or enhance spatial frequency components
  - Ex: a high-pass filter
    - Enhance sudden transitions (edges) in the spatial domain
    - While attenuating slow or gradual changes in the image
- Example: image of Lena (Fig. 13.4)
  - A simple Butterworth filter
  - High-pass filters in each horizontal line
    - Detects the vertical transitions (edges) in each row
    - MATLAB:
      - [b,a]=butter(1,100/256,'high');
      - for k=1:512;

        lenah(k,:)=filtfilt(b,a,lenah(k,:));

        end;

        Prevent phase shift
  - Another part
    - Detects abrupt horizontal transitions
  - The output of both filters – added to show the edges
    - Detected edges – superimposed on the original picture → an edge-enhanced image
    - Also usable to detect ROIs in optical imaging data sets such as microscopic images or movies

Filters in the Spatial Domain

Figure 13.4 An example of a filter application in the spatial domain using a picture of Lena (A) as input for a two-dimensional Butterworth high-pass filter. Although this spatial filter is not optimized for this application, by using the principle of high-pass filtering we can detect transitions (edges) in the spatial domain as shown in (B).

Appendix 13.1

- ***butter*** command in MATLAB
  - Filter equation:

$$\underbrace{1}_{A(1)} y_n \underbrace{- \frac{(RC/\Delta t)}{(RC/\Delta t)+1}}_{A(2)} y_{n-1} = \underbrace{\frac{1}{(RC/\Delta t)+1}}_{B} x_n$$

    - Approximated from the diagram in Fig. 11.3
    - Section 11.2.2 and Appendix 11.3
    - R=$10^4\Omega$, C=3.3μF, and Δt=1/400 → filter coefficients
      - A = [A(1) A(2)] = [1.000 -0.9296]
      - B = [0.0704]
  - More precise equation:

$$\underbrace{1}_{A(1)} y_n \underbrace{- e^{-\Delta t/RC}}_{A(2)} y_{n-1} = \underbrace{\left(1 - e^{-\Delta t/RC}\right)}_{B} x_n$$

    - Equation (A11.3-2) in Appendix 11.3 → same format as above equation
    - A = [1.000 -0.9270] and B = [0.0730]

Appendix 13.1

- ***butter*** command in MATLAB
  - A first-order butter command for f=1/(2πRC)=4.8229 Hz & a sample frequency of 400 Hz (Nyquist frequency: 200 Hz):
    - [B, A] = butter(1, 4.8229/200)
      - B = [0.0365 0.0365]
      - A = [1 -0.9270]
    - Almost identical values
      - Difference: B – two terms
        - Each term=exactly half of the single term (i.t., 0.0730/2)
        - The effect of a moving average of the input