Digital Signal Processing Laboratory
Laboratory handout

Multirate Signal Processing

V 1.3.3, November 30, 2005
Christian Feldbauer, feldbauer@tugraz.at, Marián Képesi, kepesi@tugraz.at
Klaus Witrisal, witrisal@tugraz.at, Erhard Rank, erank@tugraz.at
Signal Processing and Speech Communication Laboratory, www.spsc.tugraz.at
Graz University of Technology
Inffeldgasse 16c/II

Abstract

In single-rate systems, only one sampling rate is used throughout a digital signal processing system, whereas in multirate systems the sampling rate is changed at least once. This laboratory deals with the realization and analysis of multirate systems. In the first experiment we will examine the effects of decimation and interpolation. Next, a tree structure filter bank with quadrature mirror filters (QMFs) will be built and the quality of the reconstructed output signal will be discussed. Finally, we will analyze an example where the short-time Fourier transform (STFT) is used as a filter bank.

1 Theoretical Overview

1.1 Decimation and Interpolation

Decimation and interpolation are processes that transform a discrete-time signal with sampling rate f_s to another discrete-time signal with a new sampling rate f_s'. A decimator realizes a sampling rate conversion (down-sampling) by an integer factor M: $f_s' = f_s/M$, by using only every M^{th} input signal sample in the output signal. The output signal is at a lower sampling rate, and thus has a lower bandwidth ($f_s'/2$) than the input signal ($f_s/2$). To avoid aliasing and thus ensure correct reproduction of the signal spectrum in $f_s'/2$, the input signal has to be low-pass filtered before sampling rate conversion. The block diagram of a decimator and example spectra of input, intermediate, and output signal for a decimation factor $M = 3$ are shown in figure 1. Note the different scaling of the normalized frequency θ' axis in the output signal spectrum.

For interpolation a sampling rate conversion (up-sampling) by an integer factor L is achieved by inserting $L - 1$ zero samples between adjacent input signal samples. The resulting signal has a sampling rate $f_s' = Lf_s$, and a spectrum that includes periodic images of the original signals spectrum. To correctly reproduce the spectrum of the input signal (in the frequency range up to $f_s/2$) subsequent low-pass filtering is necessary to avoid imaging [1]. The block diagram of an interpolator, example signals and spectra for an interpolation factor $L = 3$ are shown in figure 2.

Fractional changes $\frac{L}{M}$ of the sampling rate can be achieved by combining a decimator with factor M with an interpolator with factor L. For block processing the realization of

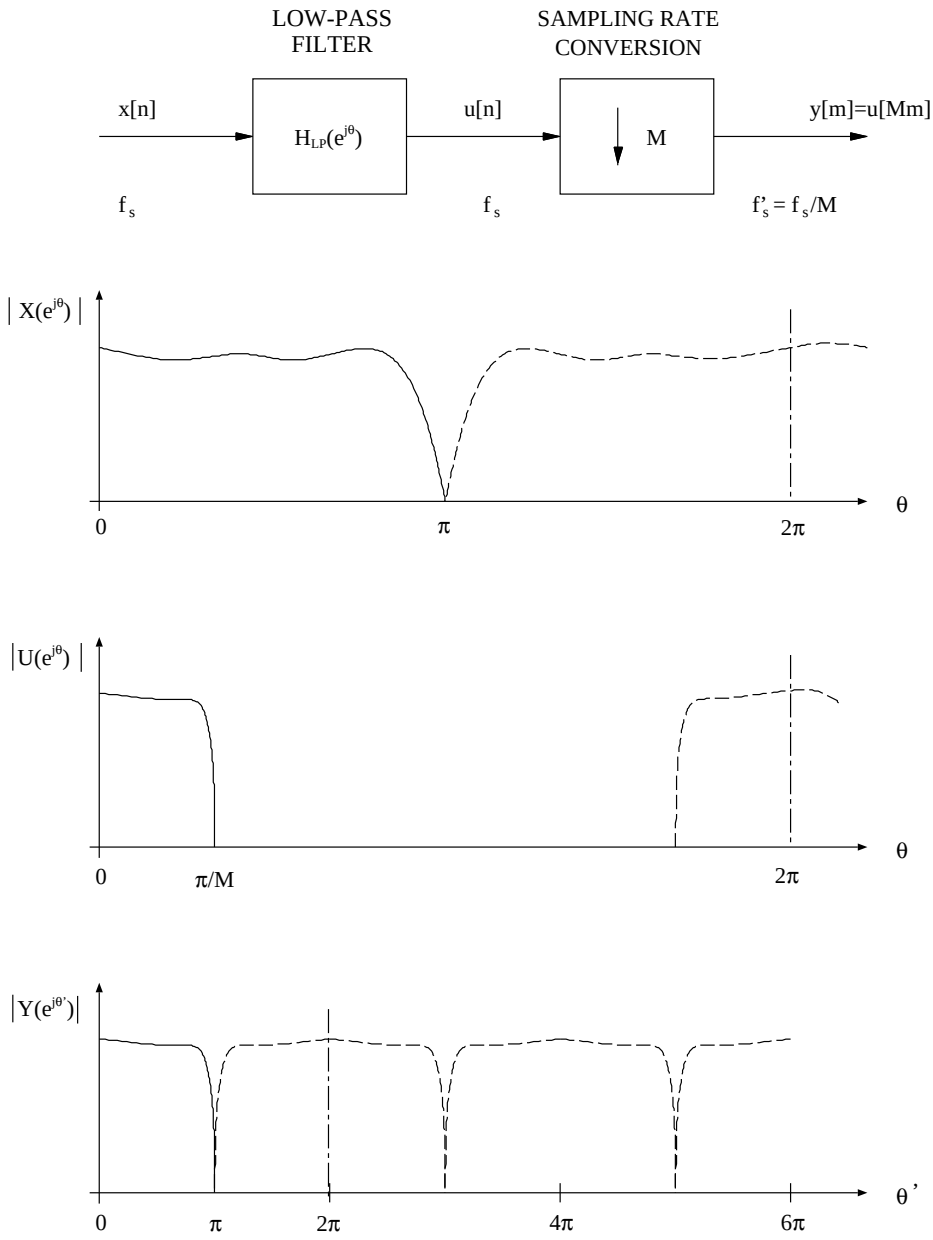

Figure 1: Block diagram of a decimator and example signal spectra for M=3.

decimation/interpolation is trivial, as long as the block size is an integer multiple of the decimation/interpolation factor. Otherwise, further considerations are necessary (scheduling, block redistribution, delay, etc.).

1.2 Tree Structure Filter Banks with Quadrature Mirror Filters (QMFs)

The basic element of tree structure filter banks is a pair of filters which consists of a low-pass and a high-pass branch. The filters are constructed in a way that the frequency band $\theta \in [0, \pi]$ which corresponds to $f \in [0, \frac{f_s}{2}]$ is split up in two equal parts mirrored at $\theta = \frac{\pi}{2}$ ($f = \frac{f_s}{4}$). The

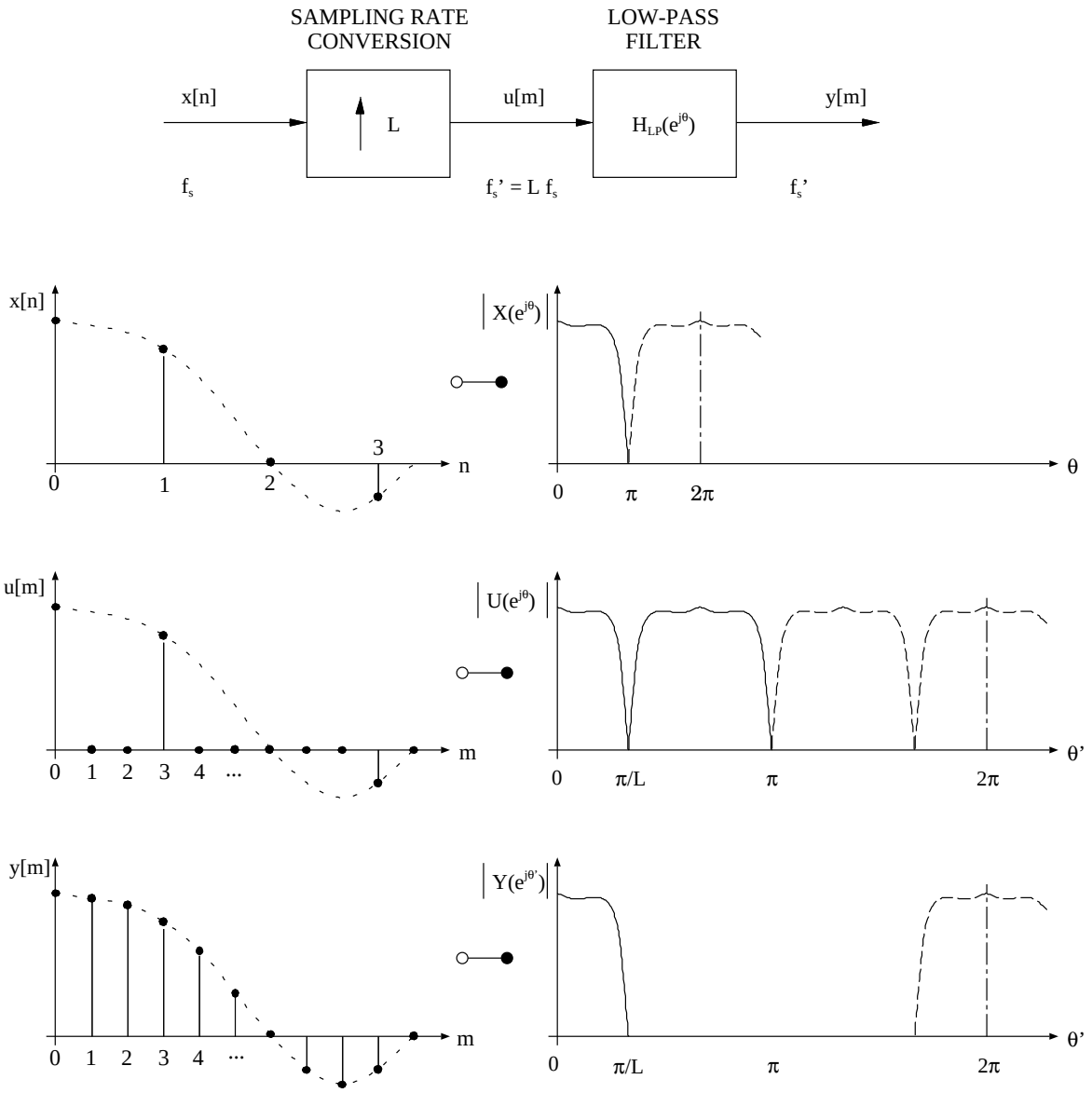

Figure 2: Block diagram of an interpolator, example signals and spectra for L=3.

frequency responses of the low-pass filter $H_L(e^{j\theta})$ and the high-pass filter $H_H(e^{j\theta})$ satisfy the following symmetry equation

$$H_H(e^{j\theta}) = H_L(e^{j(\pi+\theta)}).$$

For the corresponding transfer functions we get

$$H_H(z) = H_L(-z)$$

In the time domain, this leads to the following equation for the impulse responses

$$h_H[n] = (-1)^n h_L[n].$$

From this equation we can see that the impulse responses of the two filters differ only in the sign of every other value (for n is odd). We can exploit this to implement a joint lowpass/highpass FIR structure as depicted in figure 3 to reduce computational cost.

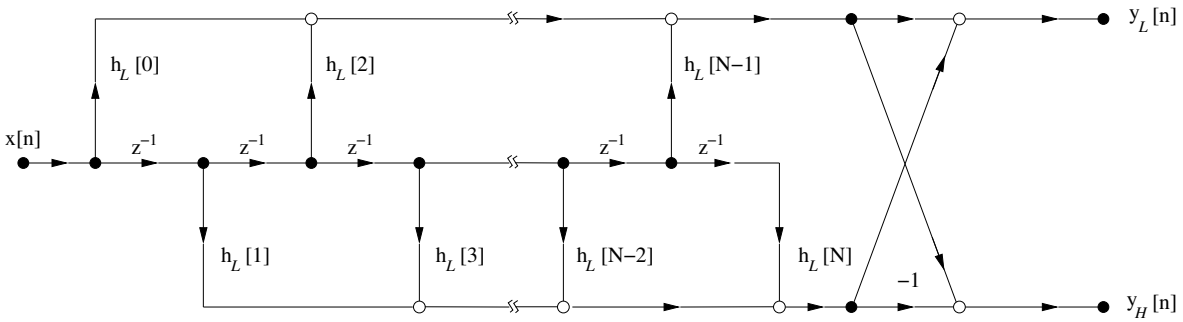

Figure 3: Signal flow graph representation of an FIR QMF filter.

Due to the symmetry every filter has a cut-off frequency of $\theta_g = \frac{\pi}{2}$. However, the cut-off frequency can be defined in several ways. One possibility is to claim that the frequency responses should be point-symmetrical to $(\theta = \frac{\pi}{2}, |H| = \frac{1}{2})$, i. e., to set $|H_{L,H}(e^{j\frac{\pi}{2}})| = \frac{1}{2}$. One advantage of such 'halfband filters' is that, if the order is even, only every second value of the impulse responses differs from zero which results in a further decrease of computational cost.

A second possibility is to claim that the sum of the power transfer functions $|H_L(e^{j\theta})|^2 + |H_H(e^{j\theta})|^2$ of the filters should be constant. This leads to point-symmetrical power transfer functions with respect to $\theta = \frac{\pi}{2}$ (actually from this property the name *quadrature mirror filters* is derived). This power complementarity can easily be achieved by implementing the QMFs as 'wave digital filters'[1]. A non-recursive (FIR) realization is possible as well, but in general it does not strictly fulfill the symmetry condition.

QMFs split the input signal in two output signals with bandwidths of half of the original bandwidth. Thus, the sampling rate of the output signals can be decimated by a factor of two. By repeated use of QMF filters and decimators a tree structure filter bank can be built. The output signal of each QMF stage is fed to another QMF and the sampling rate is decimated again. After m stages, a filter bank with 2^m outputs is built and the output signals are decimated by a factor of 2^m with respect to the input signal. It is also possible to build a non-uniformly partitioned filter bank where the number of stages is different for individual channels.

The 2^m output signals of the QMF filter bank can – possibly after being processed (encoding-decoding, individual amplification, etc.) – be re-combined to a single signal using a *synthesis filter bank* (in contrast to the *analysis filter bank* described above) also composed of QMFs. We will refer to the filters in the synthesis filter bank as inverse quadrature mirror filters (IQMF). The signal flow graph of a FIR IQMF is depicted in figure 4.

Lets denote the system functions of the synthesis IQMFs by $G_L(z)$ and $G_H(z)$. Like for the analysis filters we demand

$$G_H(z) = G_L(-z),$$

and for the impulse responses

$$g_H[n] = (-1)^n g_L[n]$$

In each stage of the synthesis filter bank the sampling rate of the signals has to be increased again. A uniformly partitioned QMF analysis/synthesis system with M channels consists of $M-1$ QMF and $M-1$ IQMF filters. For a more detailed description see [2]. The tree

[1] A wave digital filter is a structure that realizes a recursive or IIR filter.

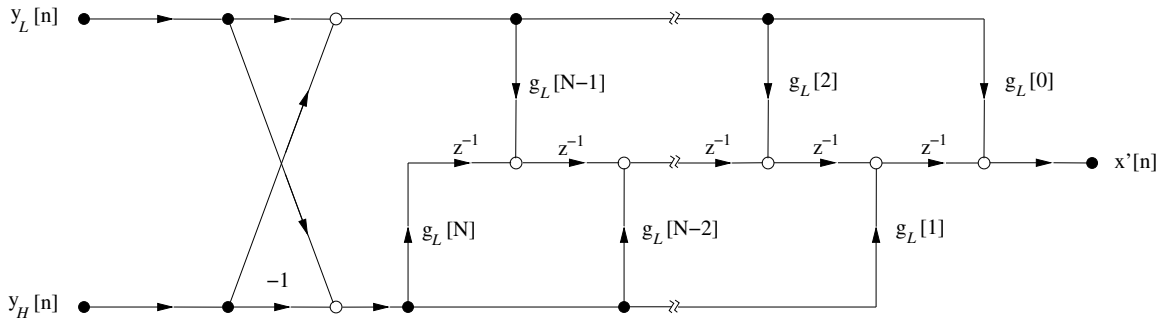

Figure 4: Signal flow graph representation of an FIR IQMF filter.

structure we are going to build in the experiment, a two channel QMF analysis/synthesis system, is depicted in figure 5.

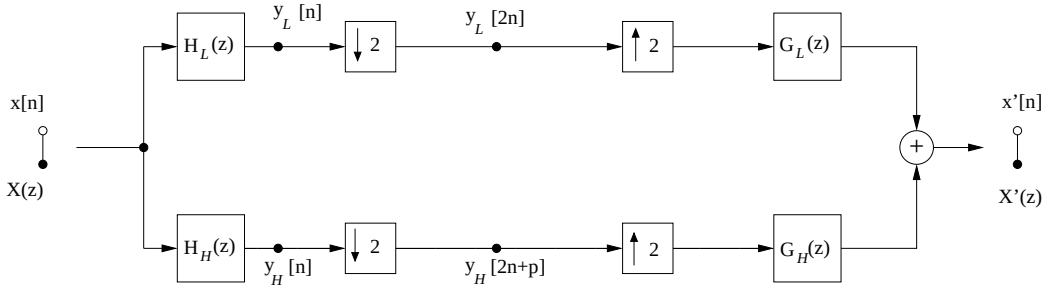

Figure 5: QMF analysis/synthesis system with 2 channels.

To ensure that all aliasing components are canceled out we have to decimate the highpass and the lowpass channel alternately (p=1) and choose the same coefficients for the synthesis filter as for the analysis filter: $G_L(z) = H_L(z)$, $G_H(z) = H_H(z)$. Another possibility is to decimate synchronously (p=0), but then we have to invert the highpass channel or modify the structure of the IQMF filters (figure 4) for $G_H(z) = -G_L(-z)$ to avoid aliasing.

1.3 Short-Time Fourier Transform (STFT)

The frequency domain is very useful for performing certain types of filtering or coding operations. Due to the convolution theorem of the Fourier transform, the time domain convolution required for filtering can often be performed more efficiently using the multiplication as the equivalent in the frequency domain.

For a discrete-time signal $x[n]$ the discrete-time Fourier transform (DTFT) is:

$$X(e^{j\theta}) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\theta n}.$$

This transform yields a continuous spectrum $X(e^{j\theta})$, with normalized frequency $\theta = 2\pi f/f_s$. $X(e^{j\theta})$ is periodic in θ with a period of 2π.

The time signal $x[n]$ can be recovered using the inverse discrete-time Fourier transform (IDTFT):

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\theta})\, e^{j\theta n}\, d\theta.$$

If $x[n]$ is time-limited to a duration of N samples: $x[n] = 0$ for $n < 0, n \geq N$, we can sample the continuous function $X(e^{j\theta})$ at N uniformly spaced frequency points in the range $\theta \in [-\pi, \pi]$ and recover the time signal $x[n]$ from the samples of $X(e^{j\theta})$ only. This corresponds to forming a periodic signal $x_p[n]$ of infinite duration and period N by concatenating the length N sequences $x[n]$ ('periodic extension'), and computing its Fourier series expansion. The corresponding transformation is called the discrete Fourier transform (DFT) of the length N discrete-time signal $x[n]$:

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\theta_k n} \qquad k = 0,\, 1,\, \ldots,\, N-1,$$

where $\theta_k = \frac{2\pi k}{N}$. If the input signal is complex-valued, all $X[k]$ carry information, but if the input signal is real-valued, only half of them do (due to the symmetry of the DFT $X[N-k] = X^*[k]$ for real-valued signals $x[n]$)[2]. The corresponding inverse discrete Fourier transform (IDFT) is given by

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]\, e^{j\theta_k n} \qquad n = 0,\, 1,\, \ldots,\, N-1,$$

which perfectly recovers the length N signal $x[n]$ from the N samples $X[k]$ of the spectrum.

DFT based analysis/synthesis methods are very common in signal processing, primarily due to the existence of the fast Fourier transform (FFT) algorithm, which allows the computation of the DFT (and IDFT) to be performed with a computational complexity of order $O(N \log N)$, rather than $O(N^2)$ if the formula above are evaluated directly.

As stated above, the DFT operates on finite length sequences. The STFT is an extension which can represent signals of any length by fragmenting into short signal blocks, or frames, and applying the DFT to each frame. The signal $x_m[n]$ for frame m is constructed by multiplication of the signal $x[n]$ and a window function $w[n]$ (windowing), and can be expressed as

$$x_m[n] = x[n + mH]\, w[n], \qquad n, m \in \mathbf{Z},$$

where H is the number of samples advanced between frames (also called 'hop size'). The window function commonly differs from zero only for a finite number of samples, $w[n] = 0$ for $n < 0$, $n \geq N$ (length N window). Thus, the windowed signal $x_m[n]$ only differs from zero value for $0 \leq n < N$, and the DFT can be applied to it. Due to the repeated 'hopping' along the time axis the window is referred to as a 'sliding window', an illustration is given in figure 6.

The output of the STFT can either be interpreted as a series of short-time spectra, or as the (sub-sampled) output of a filter bank consisting of N complex-valued bandpass filters. As an illustration for the later we can write the STFT as

$$X_k[m] = \sum_{n=0}^{N-1} x[n + mH]\, w[n]\, e^{-j\theta_k n}$$

and (after setting $H = 1$) one can see the expression for a (non-causal) discrete convolution due to the introduction of the time hop index m. We can interpret this convolution either as

$$x[n] * \left(w[-n]\, e^{j\theta_k n} \right),$$

[2] X^* is the complex conjugate of X.

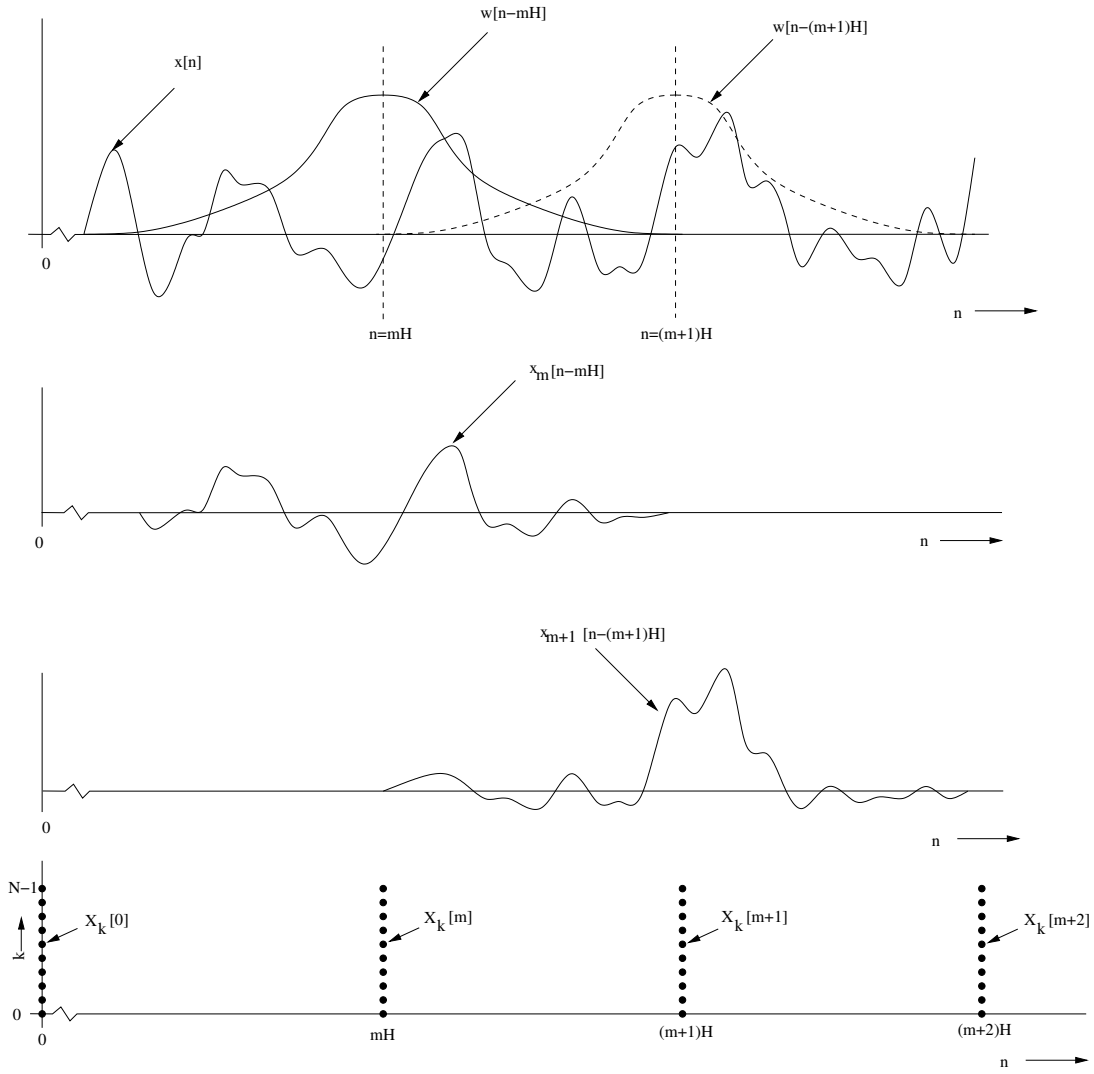

Figure 6: Frame-by-frame transform of the STFT (sliding window).

where $w[-n]e^{j\theta_k n}$ is the complex-valued impulse response of a non-causal bandpass filter, or as

$$e^{j\theta_k m}\left[\left(x[n]\; e^{-j\theta_k n}\right) * w[-n]\right],$$

i. e., as the convolution of the frequency shifted (modulated) signal $x[n]$ with the window function $w[-n]$ (and multiplication by a phase term $e^{j\theta_k m}$), which is often called 'heterodyne filter'.

Each transform of a frame provides one output value for each channel and the hop size defines the sampling rate of the bandpass signals, $f_{s,BP} = f_s/H$. For $H = 1$ the bandpass output signals have the same sampling rate as the input signal. As long as $H < N$ the individual frames have overlapping data points and for a proper choice of $w[n]$ and H a 'perfect reconstruction' of the original signal using the inverse transform is possible. The inverse STFT (ISTFT) applies an IDFT to each of the short-time spectra and adds the overlapping signal blocks. Hence, this method is often referred to as the overlap-add (OLA) method. If we set $H = N$ no overlap occurs ('maximally decimated' or 'critically sampled' filter bank), and perfect reconstruction is only possible if a rectangular window function is used.

2 Practical Part

Experiment 1:
Effects of Decimation and Interpolation

Equipment: PC + DSK, soundcard or CD player, headphones
Software: CCS, MATLAB, unzip `decint.zip` to `D:/DSP_LAB`

1. Plug the output of the PC soundcard (or a CD player) to the DSK input and headphones to the DSK output.

2. In CCS, load the project file `D:/DSP_LAB/decint/decint.pjt` .

3. Look at the file `dsp.c` . A chain of a decimator and an interpolator (without filters) is already defined there (both with a sampling rate conversion factor of 4, defined in the constants `DECFACT` and `INTFACT`). Build and run the program. Listen to the output for a music signal, can you hear distortion due to the missing low-pass filtering? How is this distortion introduced in the system!

4. Apply a sine-wave signal (at, e.g., 700 Hz) to the DSP system and observe the output spectrum on the oscilloscope. Explain the distortions! Vary the frequency of the sine-wave signal. Introduce a low-pass interpolation filter at the output. Choose a filter type and design it with MATLAB's `sptool` . Select an appropriate cut-off frequency. Add the necessary files (provided in the same directory) to the project and extend the source code in file `dsp.c` . Build and run the program. Does it reduce/eliminate the distortion for the sine-wave signal? Why/why not? What about broadband signals, like speech or music?

5. Now also put a lowpass filter at the input of the decimator before sampling rate conversion. Does this reduce/eliminate the distortion for broadband signals? Why/why not? Is the low-pass filter at the output still necessary?

6. Repeat the last step with bandpass filters (e.g., for a decimation/interpolation factor of 4, use a passband from $\frac{f_s}{8}$ to $\frac{f_s}{4}$).

   In your protocol, draw spectra (like in figure 1 and figure 2) of a fictitious signal with ramp-shaped spectrum for the different stages of the implemented multirate chain with bandpass filters.

Experiment 2:
Construction of a QMF Analysis/Synthesis System

Equipment: PC + DSK, signal generator + oscilloscope, headphones
Software: CCS, MATLAB, unzip `qmf_short.zip` to the directory `D:/DSP_LAB`

1. Connect the signal generator, the DSK, and the oscilloscope in a way to be able to measure frequency responses using sweeps.

2. In CCS, load the project file `qmf_short.pjt` .

3. Design a halfband filter (lowpass part) with MATLABs sptool (recommended parameters: Kaiser window FIR, order = 18, $\beta = 5$), choose a cut-off frequency according to the considerations in the theoretical part of this handout, and paste the coefficients into the appropriate section provided in `dsp.c` .

4. In the function `processing()` connect the lowpass output to the output signal buffer (replace `channel_low` by `sigbuf`) and build the program. Determine the frequency response of the lowpass filter.

5. Do the same for the highpass filter. Is the frequency response an exactly mirrored version of the one of the lowpass? If not, why?

6. Extend the program to realize a 2 channel analysis/synthesis filterbank using QMFs/IQMFs and decimators/interpolators. Add the missing code and include the necessary files to the project (they are provided in the same folder). Hint 1: you can use the same coefficient array for both the QMF and the IQMF. Hint 2: use `block_short_decimate_off()` and `block_short_interpolate_off()` provided in `block.h` to implement the decimation/interpolation.

7. Determine the frequency response of the overall system.

8. If there is enough time, compare the second possibility to ensure alias free reconstruction (non-alternating decimation/interpolation but negated highpass channel, see theoretical part), or build a system where aliasing occurs and examine the output signal for a sweep using the headphones. There is also a MATLAB script `test_qmf.m` provided in the directory which simulates these different cases.

Experiment 3:
Short-Time Fourier Transform

Equipment: PC + DSK, soundcard or CD player, headphones, signal generator + oscilloscope
Software: CCS, MATLAB, unzip the file `stft.zip` to `D:/DSP_LAB`

1. Plug the output of the PC soundcard (or a CD player) to the DSK input and headphones to the DSK output.

2. In CCS, load the project file `stft.pjt` .

3. Look at the function `processing()` of the program file `stft.c` . You can see a loop for muting all STFT filter bank channels but one. Choose a channel you like to let pass. Build and run the program and listen to the output signal. Repeat the last steps for a different channel, or modify the program to let several channels pass.

4. Connect the signal generator, the DSK, and the oscilloscope in a way to be able to measure frequency responses using a sweep.

5. Choose one STFT channel in the medium frequency range to let pass and measure and plot the frequency response of the overall system.

6. Until now, a rectangular window for both analysis (before the FFT) and synthesis (after the IFFT) was used. Modify the source code to use a non-rectangular window (modify the loop for the analysis window and insert a loop for the synthesis window. Note: the buffer size is `2*FFTSZ` ; input and output signals are real valued) and use the MATLAB script `win.m` to get the values of the window function. Compare the frequency responses for one pass channel and for different windows.

7. Repeat the last measurement for one window type using a different FFT length (change `HOPSZ` or/and `OVERLAP` in the headerfile `blocks.h` , and the heapsize in lnk.cmd). Do not forget to recalculate the values of the window function for the new FFT size and choose a channel which has the same center frequency as the one in the last run! Plot and compare the results.

References

[1] Crochiere, Ronald E. and Rabiner, Lawrence R., "Multirate Digital Signal Processing", Prentice-Hall, Inc., 1983.

[2] Vary, P., Heute, U. and Hess, W., "Digitale Sprachsignalverarbeitung", B.G. Teubner Stuttgart, 1998

[3] Arrowood, J., Randolph, T., Smith, M.J.T., "Filter Bank Design", In: The Digital Signal Processing Handbook, Pages: 36-1–36-17, Editors: Madisetti, V.K., Williams, D.B., CRC Press, 1998

[4] Peevers, Alan W., "A Real Time 3D Signal Analysis/Synthesis Tool Based on the Short Time Fourier Transform", MS Thesis, University of California, Berkeley, `http://www.cnmat.berkeley.edu/~alan/MS-html/MSv2_ToC.html`