ECE 3640 - Discrete-Time Signals and Systems

Sample Rate Conversion

Jake Gunther

Spring 2017

UtahStateUniversity

Department of Electrical & Computer Engineering

outline

- mathematics of sample rate reduction
- filtering to avoid aliasing
- decimation - downsampling
- downsampling convolution
- mathematics of sample rate expansion
- filtering to avoid imaging
- interpolation - upsampling
- upsampling convolution
- sample rate conversion
- multirate convolution
- automatically designed filters

application: downsampling

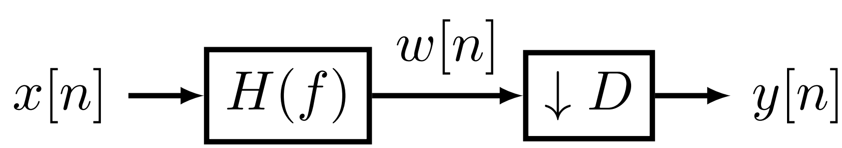
- let $y[n]$ be a D-fold down-sampled version of $x[n]$, then

$$W(f) = \frac{1}{D} \sum_{k=0}^{D-1} X\left(f - \frac{k}{D}\right)$$

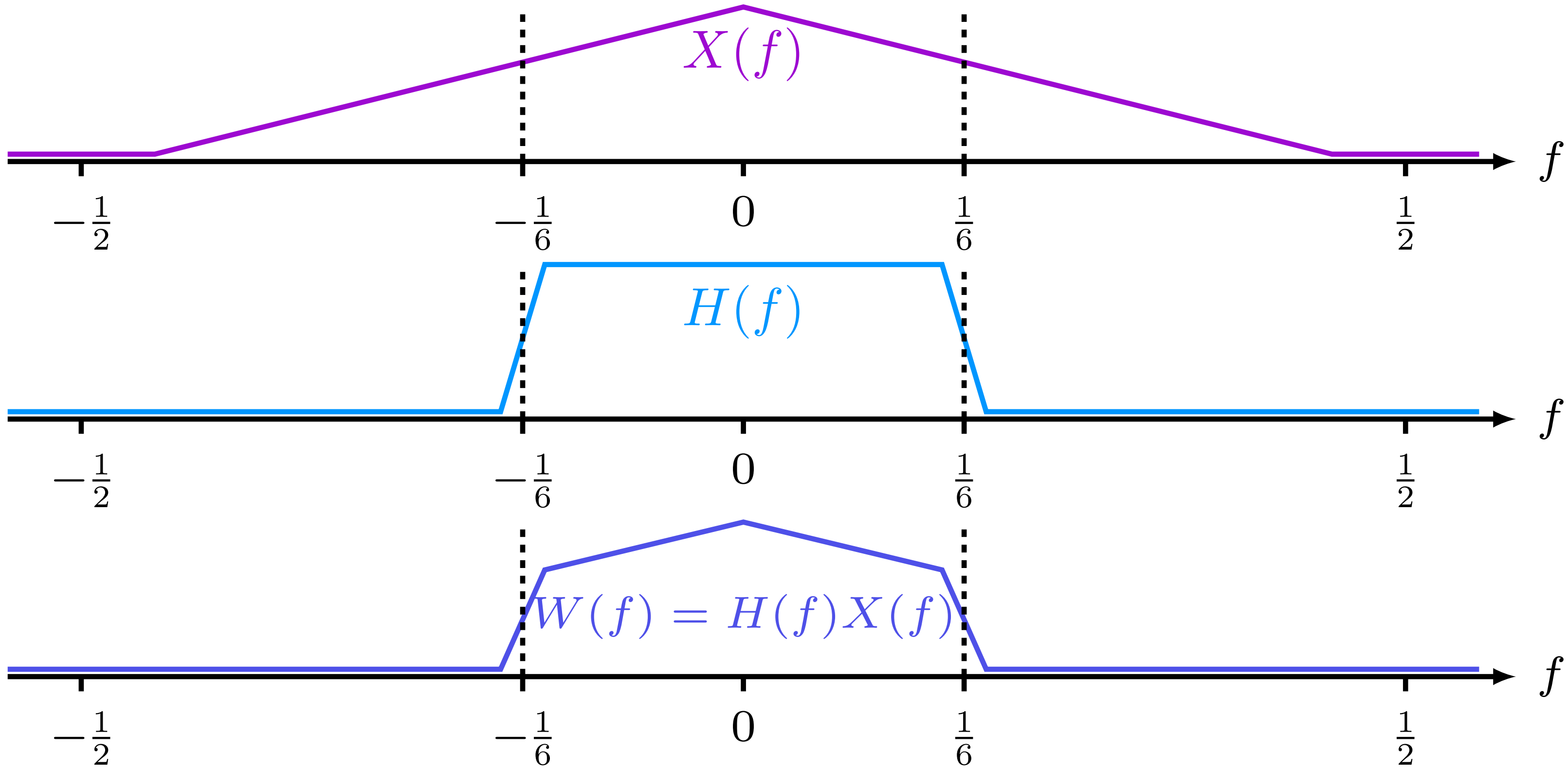
$$Y(f) = \frac{1}{D} \sum_{k=0}^{D-1} X\left(\frac{f-k}{D}\right)$$

  - replicate $X(f)$ D times at k/D for $k = 0, 1, \cdots, D-1$
  - scale the frequency axis by $1/D$
  - scale the amplitude by $1/D$

- to avoid aliasing when downsampling, pre-filter with cutoff $1/(2D)$

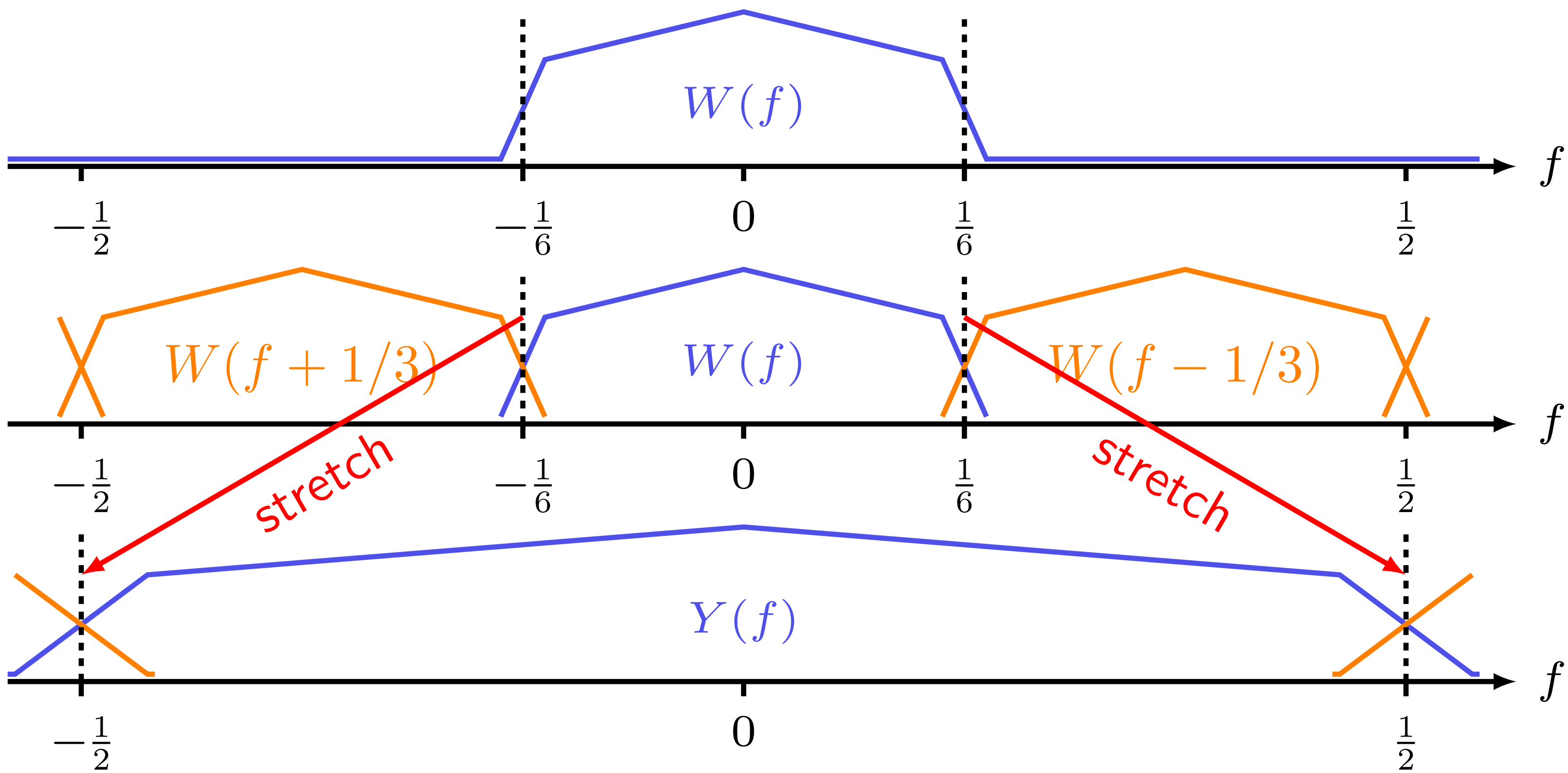


downsampling spectra, pre-filtering: $D = 3$



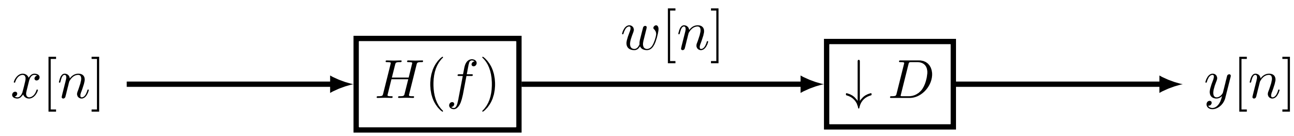
- pre-filter transition band symmetric about $1/(2D)$ leads to aliasing at high frequencies
- aliasing avoided when stop band edge $= 1/(2D)$

downsampling spectra, downsampling: $D = 3$



- transition band symmetric about $1/(2D)$ leads to aliasing at high frequencies
- aliasing avoided when stop band edge $= 1/(2D)$

downsampling convolution



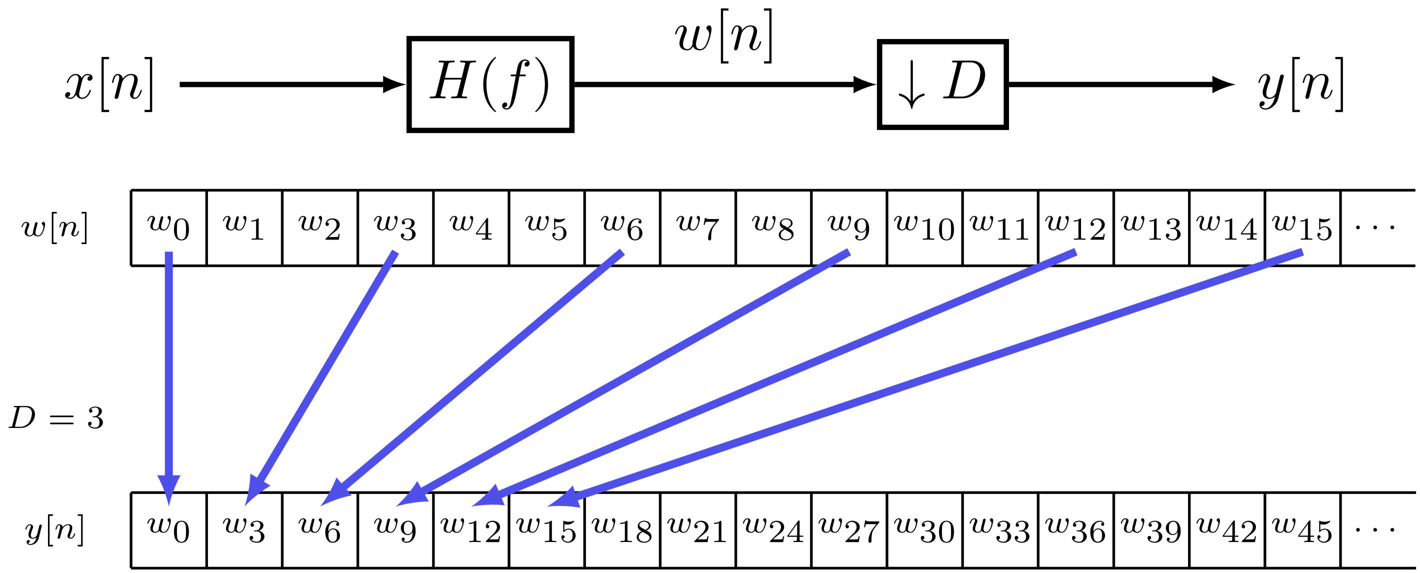
$$w[n] = \sum_k h[k]x[n-k]$$

$$y[n] = w[Dn] = \sum_k h[k]x[Dn-k]$$

normal filtering	1 sample in	1 sample out
down sampling filtering	D samples in	1 sample out

- computing all the outputs and then throwing away $D-1$ out of D is inefficient
- compute only the needed samples (i.e. compute every Dth output)

down sampling



- computing values of $w[n]$ that are not needed is inefficient
- only compute every Dth value of $w[n]$ because $y[n] = w[Dn]$

downsampling convolution using circular time-reversed buffering

```
#define D 4
#define L 7
float h[L] = {0.08, 0.25, 0.64, 0.95, 0.95, 0.64, 0.25, 0.08};
float x[L] = {0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00};
float y;
int k, i=L-1, d=0;
FILE *fx=fopen( "inputfile","rb");
FILE *fy=fopen("outputfile","wb");
fread(x+i,sizeof(float),1,fx); // read in first sample
while(!feof(fx)) {
  if(d==0) {
    for(y=0.0, k=0; k<L; k++) {
      y += h[k]*x[(k+i) % L]; // MAC with circular indexing
    }
    fwrite(&y,sizeof(float),1,fy); // save output
    d=D-1;
  } else {
    d--;
  }
  i = (i+L-1) % L; // update circular index
  fread(x+i,sizeof(float),1,fx); // read in next sample
}
fclose(fx);
fclose(fy);
```

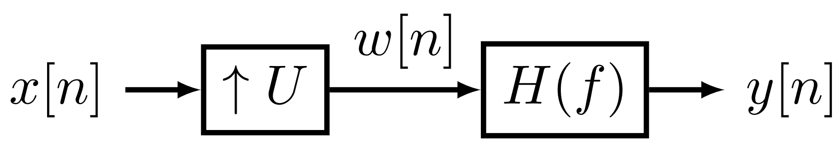
application: up-sampling

- let $y[n]$ be a U-fold up-sampled version of $x[n]$, then

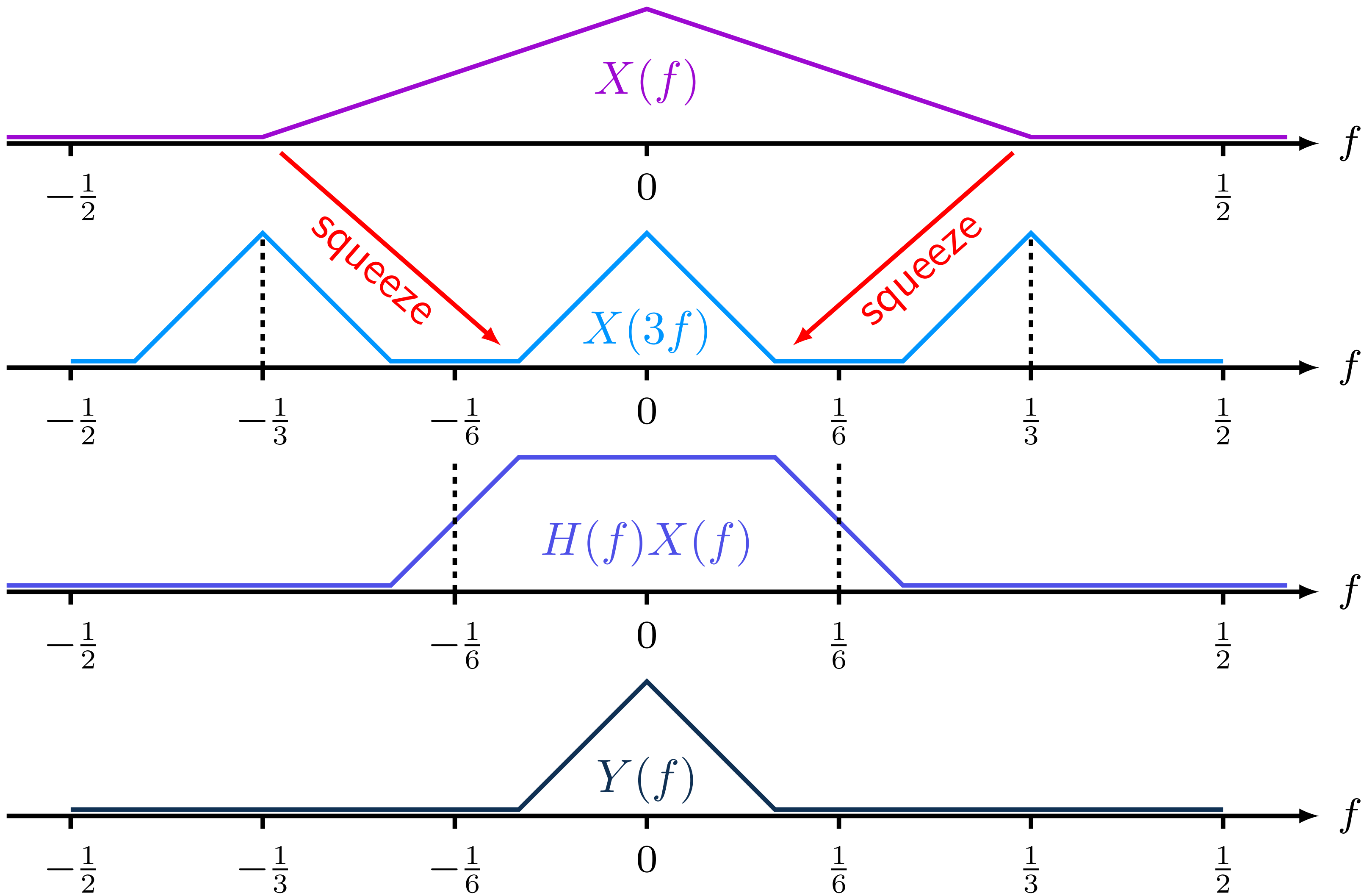
$$Y(f) = X(Uf)$$

  – scale the frequency axis by D

- to remove images that appear when up-sampling, postfilter with cutoff $1/(2U)$

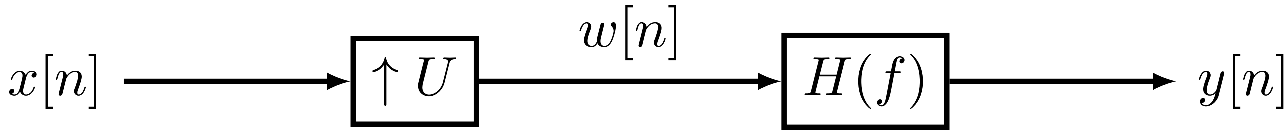


up-sampling spectra: $U = 3$



- pass band should cover low pass replica; stop band should cut off other replicas
- pass and stop band edges symmetric about $1/(2U)$

upsampling convolution



$$w[n] = \begin{cases} x[n/U], & \text{if n/U = integer,} \\ 0, & \text{otherwise} \end{cases} = \sum_{k=-\infty}^{\infty} x[k]\delta[n-kU]$$

$$w[kU+l] = \begin{cases} x[k], & \text{if } l = 0, \\ 0, & \text{if } l = 1, \cdots, U-1 \end{cases}$$

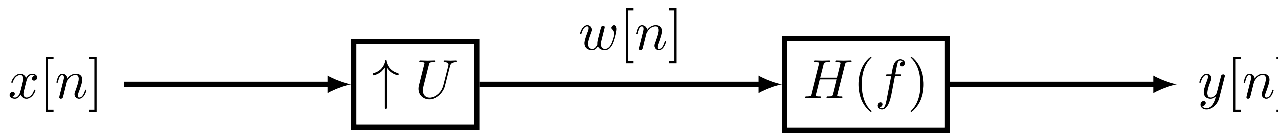
$$y[n] = \sum_{k} h[k]w[n-k]$$

$$y[iU+j] = \sum_{l=0}^{K-1} \sum_{m=0}^{U-1} h[lU+m]w[(i-l)U+(j-m)] = \sum_{l=0}^{K-1} h[lU+j]x[i-l]$$

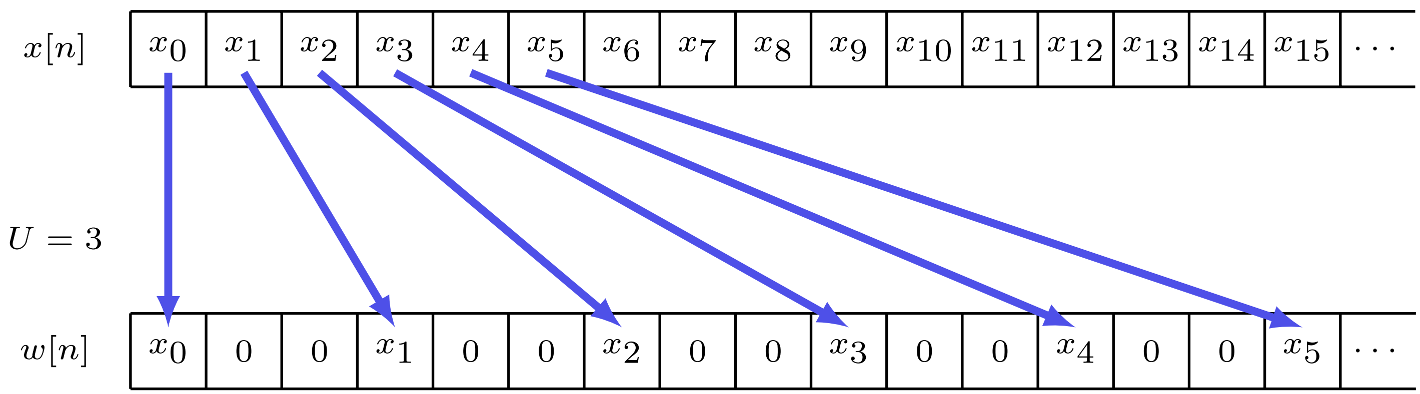
- up sampling filtering achieved by inserting $U-1$ zeros into the data buffer between each input sample, or ...

- convolve with a subset of the filter coefficients $h[lU+j], \;\; j = 0, 1, \cdots, U-1$

upsampling

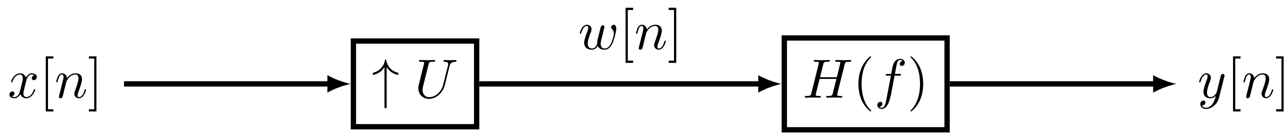


$$w[kU+l] = \begin{cases} x[k], & \text{if } l = 0, \\ 0, & \text{if } l = 1, \cdots, U-1 \end{cases}, \qquad y[iU+j] = \sum_{l=0}^{K-1} h[lU+j]x[i-l]$$



- it is wasteful to multiply by zero and to accumulate zero
- do only the multiplications necessary
- convolve with a subset of the filter coefficients $h[lU+j], \;\; j = 0, 1, \cdots, U-1$

upsampling



$$w[kU+l] = \begin{cases} x[k], & \text{if } l = 0, \\ 0, & \text{if } l = 1, \cdots, U-1 \end{cases}, \qquad y[iU+j] = \sum_{l=0}^{K-1} h[lU+j]x[i-l]$$

$h[n]$	h_0	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}	h_{11}	h_{12}	h_{13}	h_{14}
$w[15]=x[5]$	x_5	0	0	x_4	0	0	x_3	0	0	x_2	0	0	x_1	0	0
$w[16]=0$	0	x_5	0	0	x_4	0	0	x_3	0	0	x_2	0	0	x_1	0
$w[17]=0$	0	0	x_5	0	0	x_4	0	0	x_3	0	0	x_2	0	0	x_1

$h[3l]$	h_0	h_3	h_6	h_9	h_{12}
$h[3l+1]$	h_1	h_4	h_7	h_{10}	h_{13}
$h[3l+2]$	h_2	h_5	h_8	h_{11}	h_{14}
$x[5]$	x_5	x_4	x_3	x_2	x_1

- it is wasteful to multiply by zero and to accumulate zero
- do only necessary multiplications
- convolve with subsets of the filter coefficients, $h[lU+j], j = 0, 1, \cdots, U-1$

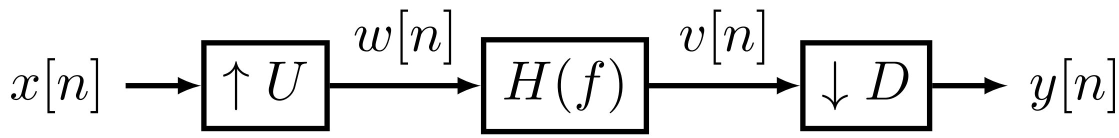
up sampling convolution using circular time-reversed buffering

```
#define U 3 /* up sampling factor */
#define L 7 /* length of filter impulse response */
float h[L] = {0.08, 0.25, 0.64, 0.95, 0.95, 0.64, 0.25, 0.08};
float x[L] = {0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00};
float y; // accumulator
int k, i=L-1, u=0;
FILE *fx=fopen( "inputfile","rb");
FILE *fy=fopen("outputfile","wb");
fread(x+i,sizeof(float),1,fx); // read in first sample
while(!feof(fx)) {
  for(y=0.0, k=0; k<L; k++) {
    y += h[k]*x[(k+i) % L]; // MAC with circular indexing
  }
  fwrite(&y,sizeof(float),1,fy); // save output
  i = (i+L-1) % L; // update circular index
  if(u==0) {
    fread(x+i,sizeof(float),1,fx); // read in next sample
    u = U-1;
  } else {
    x[i] = 0.0; // set next sample to zero
    u--;
  }
}
fclose(fx);
fclose(fy);
```

up sampling convolution using circular time-reversed buffering

```c
#define U 3 /* up sampling factor */
#define L 7 /* length of impulse response */
float filter_coefs[L] = {0.08, 0.25, 0.64, 0.95, 0.95, 0.64, 0.25, 0.08};
float y; // accmumulator
int i=L-1, j, k, m;
int M = L/U + ((L%U)>0); // same as M = ceil(L/U)
int N = M*U; // length of padded impulse response
float *x = (float*)calloc(sizeof(float),M); // circular data buffer
float *h = (float*)calloc(sizeof(float),N); // zero-padded filter coefs
memcpy(h,filter_coefs,sizeof(float)*L); // filt coefs => zero-padded buffer
FILE *fx=fopen( "inputfile","rb");
FILE *fy=fopen("outputfile","wb");
fread(x+i,sizeof(float),1,fx); // read first sample into circ data buffer
while(!feof(fx)) {
  for(j=0; j<U; j++) { // loop over subsets of filter coefficients
    for(y=0.0, k=0, m=0; k<M; k++, m+=U) {
      y += h[m+j]*x[(k+i) % M]; // MAC with circular indexing
    }
    fwrite(&y,sizeof(float),1,fy); // write U outputs for ever 1 input
  }
  i = (i+L-1) % L; // update circular index
  fread(x+i,sizeof(float),1,fx); // read in next sample
}
fclose(fx);
fclose(fy);
```

sample rate conversion by rational factor U/D

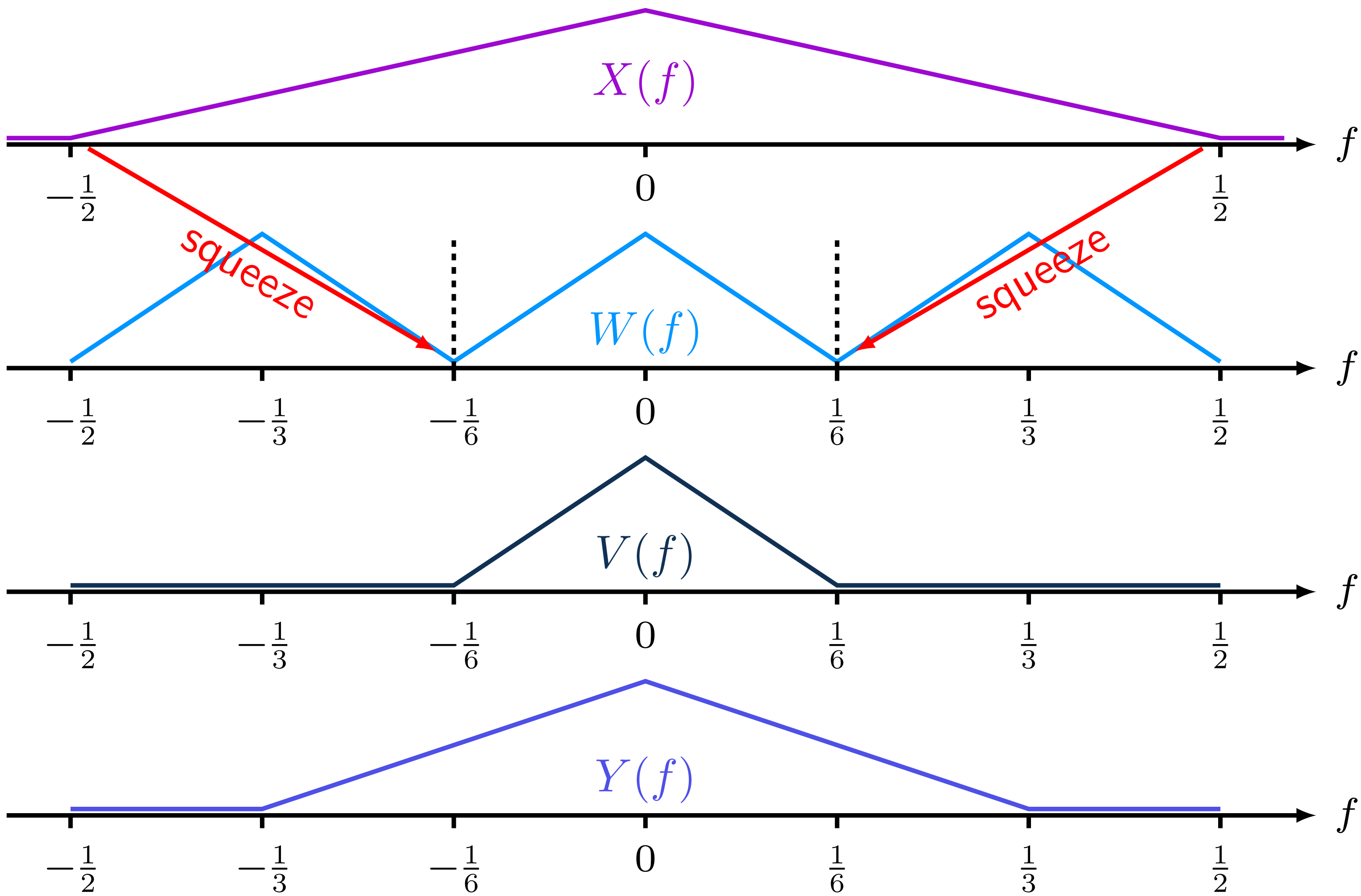


- if $U > D$ then the output sample rate is higher than the input sample rate
- if $U < D$ then the output sample rate is lower than the input sample rate
- cut off frequency for $H(f)$ is $\min\{1/(2D), 1/(2U)\} = 1/(2\max\{D, U\})$

multirate convolution using circular time-reversed buffering

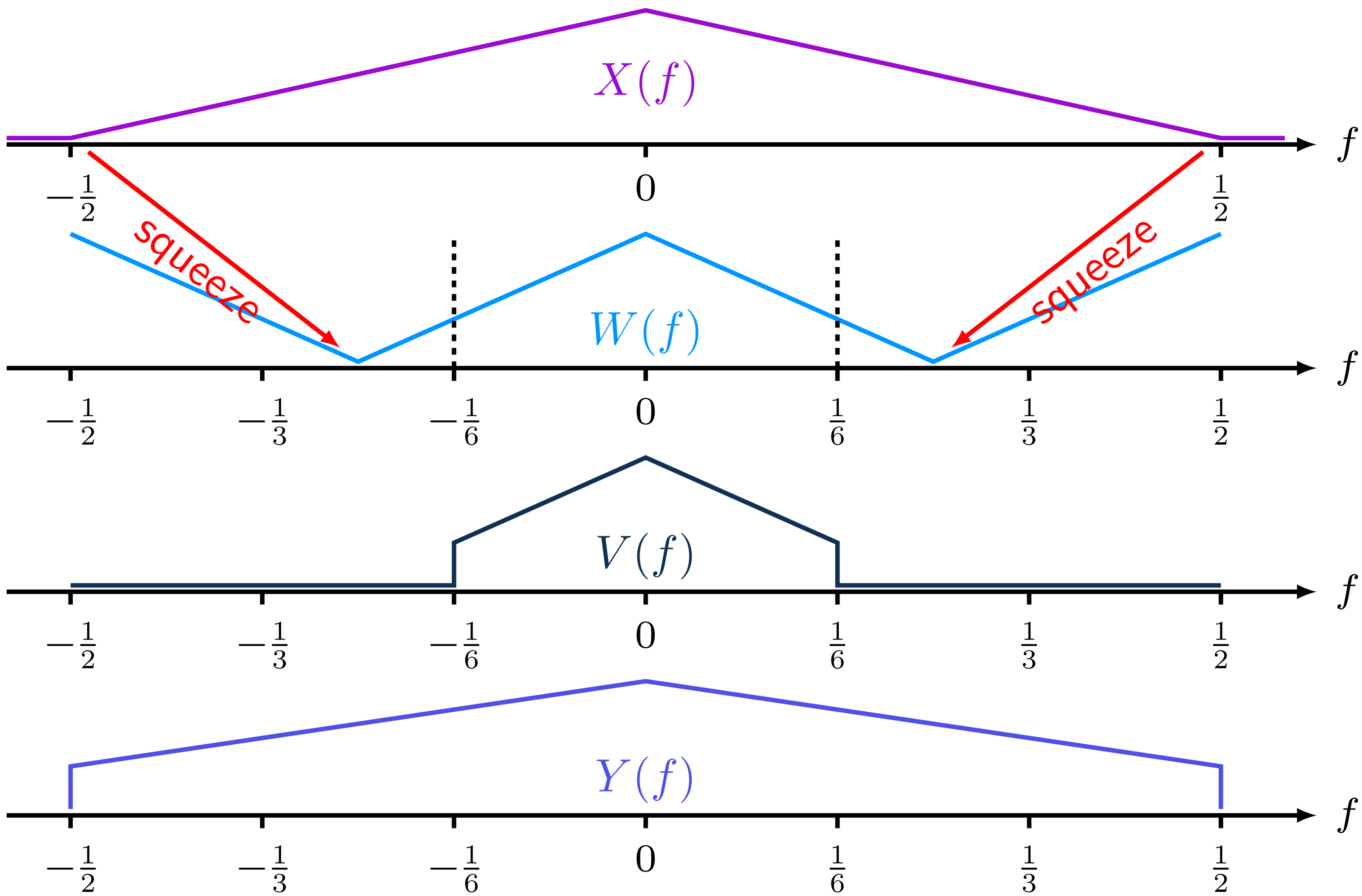
`integrate up and down sampling codes`

example: $U = 3$ and $D = 2$



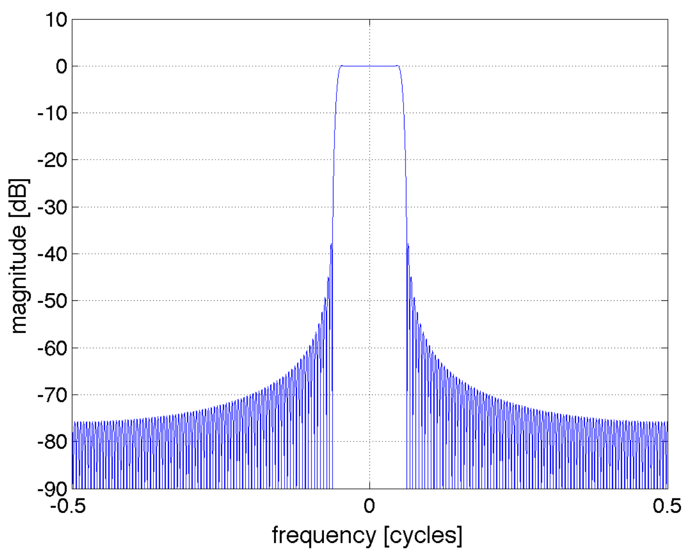
- input $f_{\text{max}} = 1/2$, output $f_{\text{max}} = (D/U) f_{\text{max}} = 1/3$

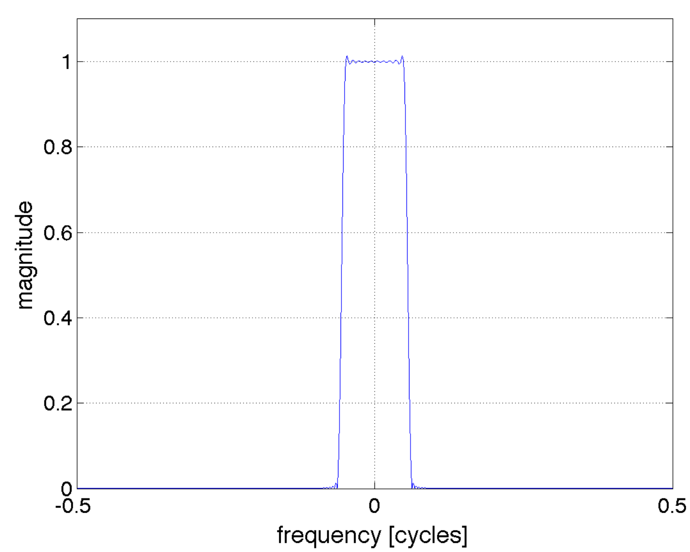
example: $U = 2$ and $D = 3$



$H(f)$ design for sample rate conversion

- given: U and D and L (filter half length)
- let: $N = \max\{U, D\}$
- let: $f_{\text{pass}} = 0.9/(2N)$, $\quad f_{\text{stop}} = 1.1/(2N)$
- compute: $f_1 = (f_s + f_p)/2$, $\quad f_2 = (f_s - f_p)/2$
- let: $h[n] = \frac{1}{N}\frac{\sin(2\pi f_1 n)}{2\pi f_1 n}\frac{\sin(2\pi f_2 n)}{2\pi f_2 n}$ for $n = -L, -L+1, \cdots, L-1, L$
- example: $L = 100$, $N = 9$





assignment

- write a C program to perform rational sample rate conversion
- the program should accept command line arguments for U, D the file to be converted, and a filter impulse response file
- design the filters in Matlab
- use the file `galway11_mono_45sec.wav` to perform the following processing steps
- up sample by $U = 2$, use $f_{\text{pass}} = 0.9/(2U)$ and $f_{\text{stop}} = 1.1/(2U)$
  - what are the input and output sample rates?
- down sample by $D = 5$, use $f_{\text{pass}} = 0.9/(2D)$ and $f_{\text{stop}} = 1.1/(2D)$
  - what are the input and output sample rates?
- perform a $U/D = 2/5$ sample rate conversion
  - what are the input and output sample rates?

- what f_{pass} and f_{stop} did you use?

- in each of these cases, choose a sufficiently long filter so that the stop band attenuation is greater than 40 dB
  - how long was the filter in each case
  - plot the magnitude response (both linear and dB scales)

- in each of these cases, plot spectrograms of the signal before and after conversion
  - comment on what you see in the output spectrogram and how it can be explained based on the sample rate conversion operation
  - compare the input and output spectrograms
  - make sure to use the correct sample rates for the spectrograms

- down sample the signal by $D = 5$ without any anti-aliasing filtering
  - listen to the input and output and compare to the case in which anti-aliasing filtering is used
  - comment on what you hear (what does aliasing sound like?)
  - compare spectrograms of the downsampled signal with and without anti-aliasing filtering
  - comment on what you see