Lecture 10, Multirate Signal Processing Transforms as Filter Banks

Equivalent Analysis Filters of a DFT

From the definitions in lecture 2 we know that a DFT of a block of signal x is defined as

$$X(k)=\sum_{n=0}^{N-1} x(mN+n)e^{-j2\frac{\pi}{N}\cdot k\cdot n}$$

Hence we can rewrite the DFT as a matrix multiplication with the matrix $\boldsymbol{T}$ with the elements,

$$\boldsymbol{T}_{n,k}=e^{-j\frac{2\pi}{N}\cdot n\cdot k}$$

n is the time and row index, and k is the frequency and column index. Using the signal block vector

$$\boldsymbol{x}(m)=[x(mN),x(mN+1),\ldots,x(mN+N-1)] \quad (2)$$

we can rewrite the DFT of block m as a matrix multiplication,

$$\boldsymbol{y}(m)=\boldsymbol{x}(m)\cdot\boldsymbol{T} \qquad (3)$$

This is the matrix formulation of the DFT, where

the matrix multiplication (3) is

$$y_k(m)=\sum_{n=0}^{N-1} x(mN+n)\cdot e^{\frac{-j\cdot 2\pi}{N}\cdot k\cdot n} \quad (4)$$

Here we can now extract the equivalent impulse responses.
Now we can **compare** above transform eq. (4) with the **convolution and downsampling** eq. (2 of last time),

$$y_{n_0}^{\downarrow N}(m)=\sum_{n=0}^{L-1} x(mN+n_0-n)\,h_k(n)$$

(eq. 2 of lecture 10, see also lecture 1)
We see that here the index n for our signal x has the **reverse order** /reverse sign as in eq. 4. But since we are interested in the equivalent impulse response, we simply **reverse the index order** in the transform equation (4) (we use the index substitution $n\rightarrow N-1-n'$).
We still obtain the same sum, because the sum ordering doesn't change the result.

$$y_{N-1}^{\downarrow N}(m)=y_k(m)=\sum_{n'=0}^{N-1} x(mN+N-1-n')\cdot e^{-j\frac{2\pi}{N}\cdot(N-1-n')\cdot k}$$

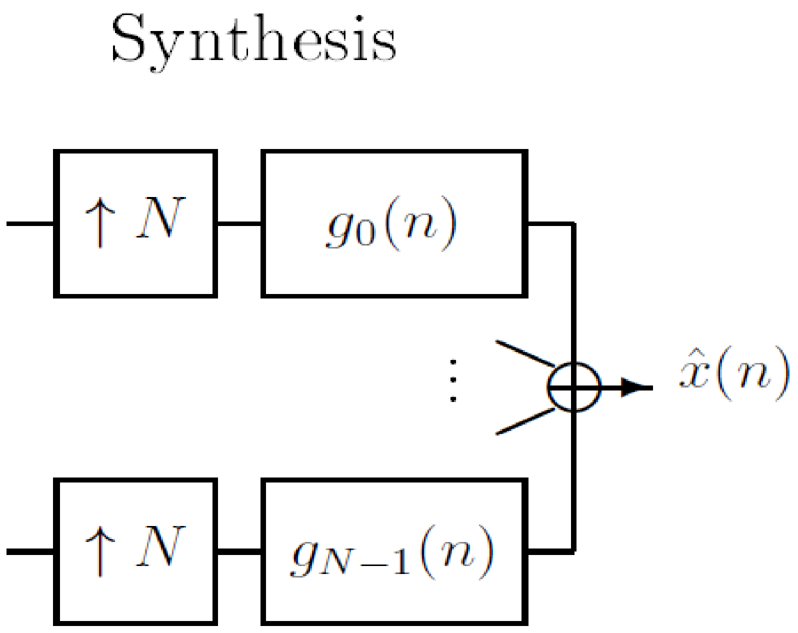
Compare this with our convolution sum in (2 of lecture 10), it looks like the convolution sum, with the phase index of $n_0=N-1$. This indicates that $y_k(m)$ contains **phase N-1** of the **downsampled subband signal**.
Through this comparison we obtain the **equivalent impulse response** of our DFT as

$$h_k(n) = e^{-j\frac{2\pi}{N}\cdot(N-1-n)\cdot k} = \boldsymbol{T}_{N-1-n,k}$$

for $k, n = 0, \ldots, N-1.$
This is now the equivalent analysis impulse response of our DFT, interpreted as analysis filter bank! Observe that it can also be interpreted as a **rectangular window** of length N **with modulation** of this exponential term!

In Conclusion: The **DFT** can also be seen as a **special kind of filter bank**. Each DFT coefficent can be seen as a sample of one of the downsampled subbands.
Each column of our transform matrix $\boldsymbol{T}$ represents the impulse response of one subband filter, but in reversed order (we can read the impulse response out of the transform matrix starting from the bottom and going up).

Equivalent Synthesis Filter Bank



A synthesis filter bank.

We still need the find the equivalent synthesis filters, if we apply the inverse transform for the perfect reconstruction of our signal (we get perfect reconstruction, i.e. the original signal, if the transform $\boldsymbol{T}$ is invertible).
For the mathematical formulation of the inverse transform, we can again write the reconstructed signal $\hat{x}$ as a sequence of blocks,
$\hat{\boldsymbol{x}}(m) := [\hat{x}(mN), \hat{x}(mN+1), \ldots, \hat{x}(mN+N-1)]$
We obtain the reconstructed signal by simply multiplying the blocks of subband samples $\boldsymbol{y}(m)$ with the inverse transform matrix,

$$\hat{\boldsymbol{x}}(m) = \boldsymbol{y}(m) \cdot \boldsymbol{T}^{-1} \qquad (5)$$

The elements of this inverse DFT matrix are

$$(\boldsymbol{T}^{-1})_{k,n} = \frac{1}{N} e^{j \cdot \frac{2\pi}{N} \cdot n \cdot k}$$

Eq. (5) can now also be written as (again using row times column)

$$\hat{x}(mN+n) = \sum_{k=0}^{N-1} y_k(m) \frac{1}{N} e^{j \cdot \frac{2\pi}{N} \cdot n \cdot k} \qquad (6)$$

with $n=0,\ldots,N-1$.
To see the resulting impulse response of the equivalent synthesis filter bank, we need the equations for upsampling and filtering in the

synthesis filter bank.

In the synthesis filter bank, the subband signals $y_k(m)$ are **upsampled** first, and we obtain the upsampled signal $y_{0,k}^{\uparrow N}(n)$, with phase 0 and time index n because we are at the higher sampling rate. The following filtering is the **convolution** with the synthesis subband impulse responses $g_k(n)$, hence we get the output of the k'th filter as

$$x_k(n) := y_{0,k}^{\uparrow N}(n) * g_k(n) = \sum_{n'=0}^{L-1} y_{0,k}^{\uparrow N}(n-n') \cdot g_k(n') \quad (7)$$

where L is the length of the synthesis filter (see also lecture 1). We assume that L is a multiple of N, which we can always obtain by appending zeros if necessary. Observe that most samples of $y_{0,k}^{\uparrow N}(n)$ are zero. To rewrite (7) such that we only process the non-zero samples, we again use the block wise processing, by substituting $n \rightarrow mN+n$ and $n' \rightarrow m'N+n'$ (using block indices m and m' and in block or phase indices n and n') to obtain

$$\hat{x}_k(mN+n) = \sum_{m'=0}^{L/N-1} \sum_{n'=0}^{N-1} y_{0,k}^{\uparrow N}(mN+n-m'N-n') \cdot g_k(m'N+n')$$

with $n=0,\ldots,N-1$.
Here we can see that we get the non-zero elements of $y_{0,k}^{\uparrow N}(n)$ if its argument is integer multiples of N, and we get this if we have

$n=n'$. Because of this, the inner sum only consists of one summand (because the sum over n' disappears),

$$\hat{x}_k(mN+n)=\sum_{m'=0}^{L/N-1} y_{0,k}^{\uparrow N}((m-m')N)\cdot g_k(m'N+n)$$

For the reconstructed signal, we simply **add up** all the N **subbands**,

$$\hat{x}(mN+n)=\sum_{k=0}^{N-1}\sum_{m'=0}^{L/N-1} y_{0,k}^{\uparrow N}((m-m')N)\cdot g_k(m'N+n)$$

Here we can now replace $y_{0,k}^{\uparrow N}(mN)$ by $y_k(m)$, since we now address only the non-zero elements,

$$\hat{x}(mN+n)=\sum_{k=0}^{N-1}\sum_{m'=0}^{L/N-1} y_k(m-m')\cdot g_k(m'N+n) \qquad (8)$$

This is now our reconstructed signal, where the first sum is over the subbands, and the second sum is over the blocks.
At this point, we can compare this filter bank eq. (8) with the synthesis transform eq. (6).

$$\hat{x}(mN+n)=\sum_{k=0}^{N-1} y_k(m)\frac{1}{N}e^{j\cdot\frac{2\pi}{N}\cdot n\cdot k} \qquad (6)$$

First we see that the transform eq. (6) has no sum over the blocks, meaning the filters only have length <u>L=N</u>, hence the inner sum disappears with <u>m'=0</u>.
With this, (8) becomes

$$\hat{x}(mN+n)=\sum_{k=0}^{N-1} y_k(m)\cdot g_k(n)$$

Compare this with the transform equation (6), we can see now that the equivalent impulse responses are

$$g_k(n) = \frac{1}{N} e^{j \cdot \frac{2\pi}{N} \cdot n \cdot k} = (\boldsymbol{T}^{-1})_{k,n}$$

Looking at our transform matrix, we see that this impulse responses correspond to each row of the inverse transform matrix, non-reversed.

In Conclusion: the **impulse responses of our subbands are the rows of the synthesis transform**, here **not time reversed!** (We can read out the impulse responses of the synthesis filter bank from left to right in each column of the transform matrix).

Observe: Each time we got our result by comparing the convolution sum of our equivalent filter bank with the transform sum.

Python Example:

Take a DFT of size $N=4$. Its transform matrix T can be obtained with

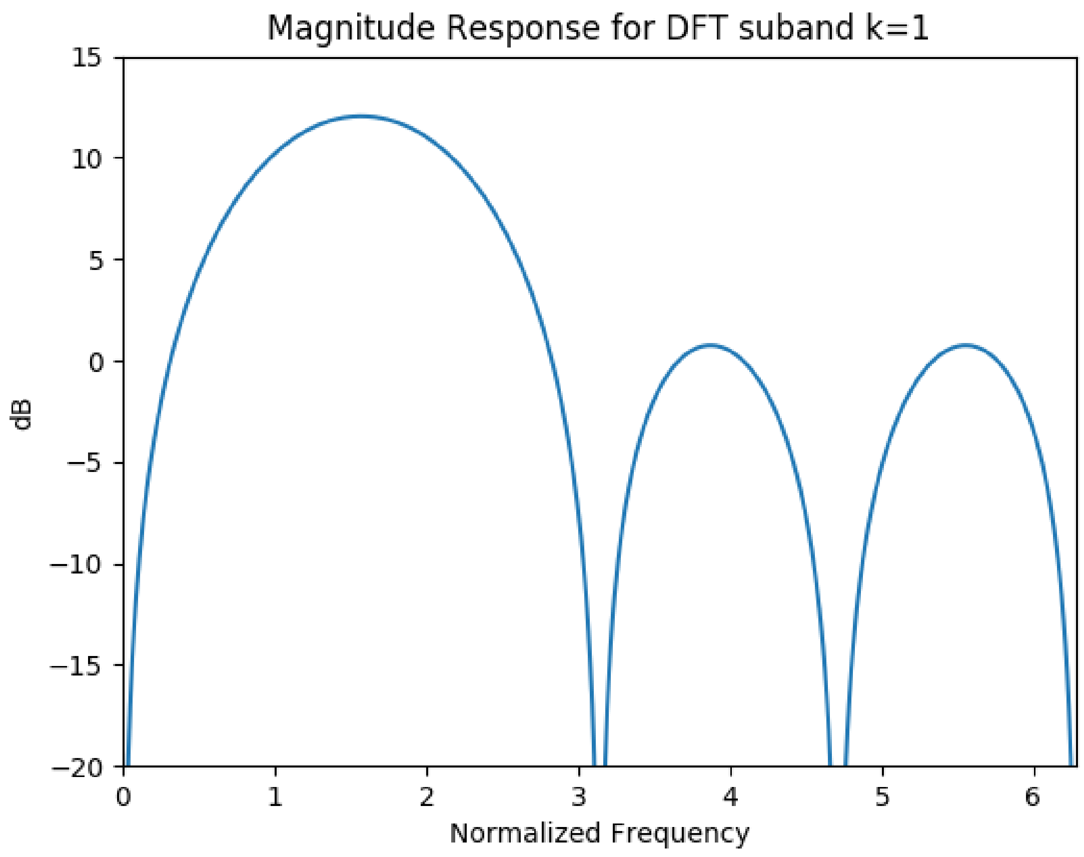
```
ipython --pylab
T=fft.fft(eye(4))
T
Out:
array([[ 1.+0.j,  1.+0.j,  1.+0.j,  1.+0.j],
       [ 1.+0.j,  0.-1.j, -1.+0.j,  0.+1.j],
       [ 1.+0.j, -1.+0.j,  1.+0.j, -1.+0.j],
       [ 1.+0.j,  0.+1.j, -1.+0.j,  0.-1.j]])
```

Observe that we have complex values in the transform, and hence obtain complex valued filters. To evaluate complex valued filters, we need the full circle in the frequency domain, from 0 to 2pi.

If we want to obtain the frequency response of subband $k=1$ of this DFT filter bank, we take the second column, time-reverse it, and plot the frequency response with freqz. Because Octave/Matlab starts indexing at 1, we need the index 2 for the subband k. In Octave/Matlab, we use "fft", which stands for Fast Fourier Transform, the fast implementation of the DFT, and command "freqz" uses the fft function internally,

```
import scipy.signal as sp
```

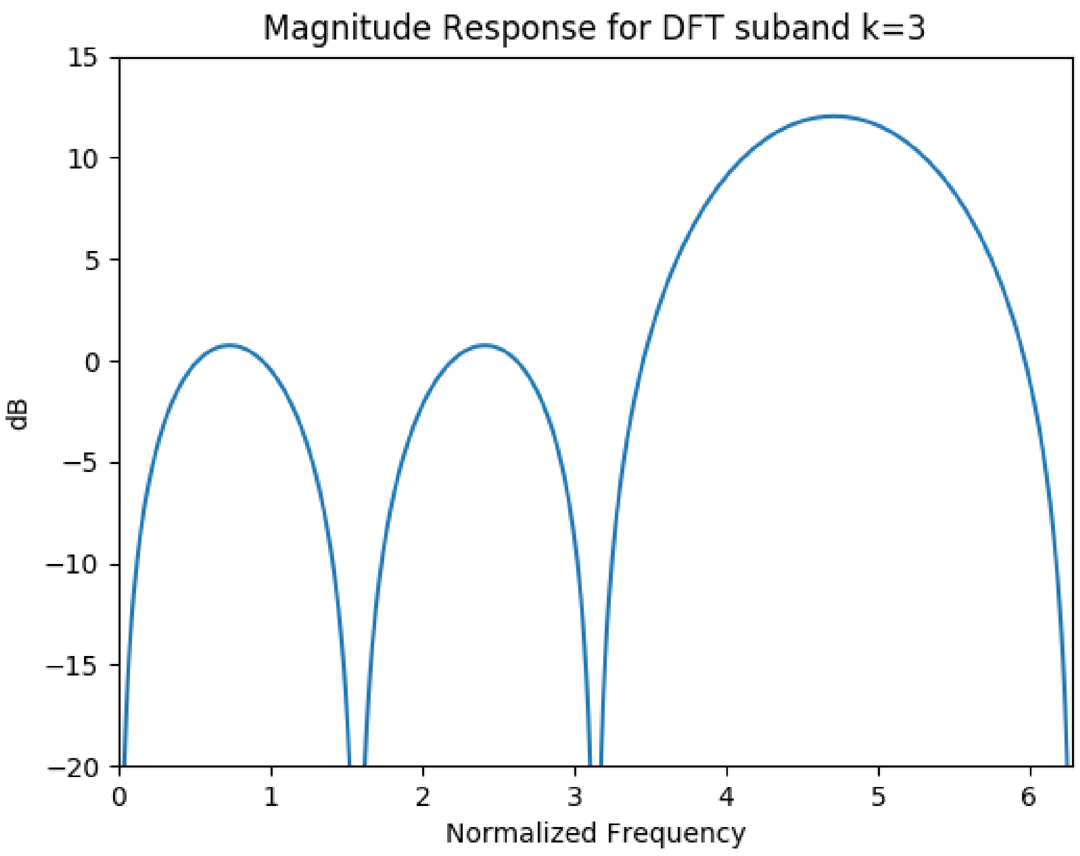
```
W,H=sp.freqz(flipud(T[:,1]),whole=True)
plot(W,20*log10(abs(H)))
axis([0,6.28, -20,15])
xlabel('Normalized Frequency')
ylabel('dB')
title('Magnitude Response for DFT suband k=1')
```



Observe: We have a bad stopband attenuation (less than 20dB).
Also **observe**: The frequency axis is going from **0 to 2pi** (instead of just pi). This is because we have a **complex** impulse response. The normalized frequency 2pi is the sampling frequency . Since we have a 2π periodic frequency, this is identical to frequency 0, and

the **frequencies from pi to 2pi** can also be seen as the **negative frequencies from -pi to 0**. This shows that this filter has a **pass band at the positive frequencies**, but **not at the negative frequencies**.
The equivalent passband at the negative frequencies is obtained from subband k=3,

```
W,H=sp.freqz(flipud(T[:,3]),whole=True)
plot(W,20*log10(abs(H)))
xlabel('Normalized Frequency')
ylabel('dB')
title('Magnitude Response for DFT suband k=3')
```



Observe: This looks like the frequency mirrored version of the filter for k=1. This also shows how to separate positive and negative frequencies.

The **low pass is at k=0**, and the **high pass appears at k=2**. Observe that here the high pass appears in the middle, because of this symmetry between positive and negative frequencies.

Example Transform as Filter Bank:
Now we show in an example that the **transform is** indeed a special case of **a critically sampled filter bank** with the above computed filters.

Take the **example signal** of length 8,
`x=sin(2*pi/8*1*arange(8))`
and its decomposition into blocks of length 4:

```
xm=zeros((2,4))
xm[0,:]=x[0:4]
xm[1,:]=x[4:8]
xm
Out:
array([[  0.00000000e+00,   7.07106781e-01,   1.00000000e+00,
          7.07106781e-01],
       [  1.22464680e-16,  -7.07106781e-01,  -1.00000000e+00,
         -7.07106781e-01]])
```

Again we obtain our DFT **transform matrix** as

`T=fft.fft(eye(4))`

We obtain the **transformed blocks** with

```
yt=dot(xm,T)

yt
Out:
array([[ 2.41421356 +0.00000000e+00j, -1.00000000 +1.11022302e-16j,
        -0.41421356 +0.00000000e+00j, -1.00000000 -1.11022302e-16j],
       [-2.41421356 +0.00000000e+00j,  1.00000000 -2.22044605e-16j,
         0.41421356 +0.00000000e+00j,  1.00000000 +2.22044605e-16j]])
```

Here, **each row** contains the **spectrum** of each corresponding block.

Now we process the input signal x through a **critically sampled filter bank** with the **equivalent filter impulse responses**, the transform matrix columns, flipped up-down, and down-sampled with the last phase of the blocks, $n_0 = N-1 = 3$ (as it appeared in our derivation of the equivalent impulse responses),

```
y=zeros((2,4))*1j
for k in range(4):
   y[:,k]=sp.lfilter(flipud(T[:,k]),1,x)[3::4]

 y
Out[80]:
array([[ 2.41421356 +0.00000000e+00j, -1.00000000 +1.11022302e-16j,
        -0.41421356 +0.00000000e+00j, -1.00000000 -1.11022302e-16j],
       [-2.41421356 +0.00000000e+00j,  1.00000000 -2.22044605e-16j,
         0.41421356 +0.00000000e+00j,  1.00000000 +2.22044605e-16j]])
```

Compare with the transform output:

```
yt
Out:
array([[ 2.41421356 +0.00000000e+00j, -1.00000000 +1.11022302e-16j,
         -0.41421356 +0.00000000e+00j, -1.00000000 -1.11022302e-16j],
        [-2.41421356 +0.00000000e+00j,  1.00000000 -2.22044605e-16j,
          0.41421356 +0.00000000e+00j,  1.00000000 +2.22044605e-16j]])
```

We can see that yt from the **transform** and y from the critically sampled **filter bank** are indeed the **same**!

In conclusion: We see that a **transform** is a **special case** of a **filter bank**. The tool of reading out the impulse responses from a transform matrix allows us to **analyze the resulting filters**, and to judge if they fulfill our requirements.