Adrian Willi
Stephan Stofer

Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Information Technology
FH Zentralschweiz

Applied Numerics - Exercise 4

Prof. Dr. Josef F. Bürgler

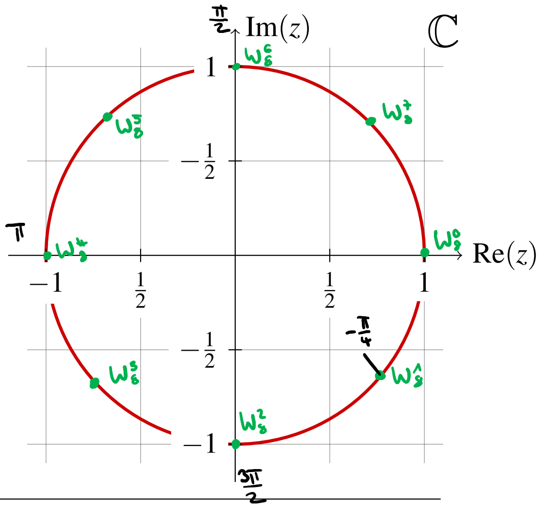
I.BA_IMATH, Semesterweek 11

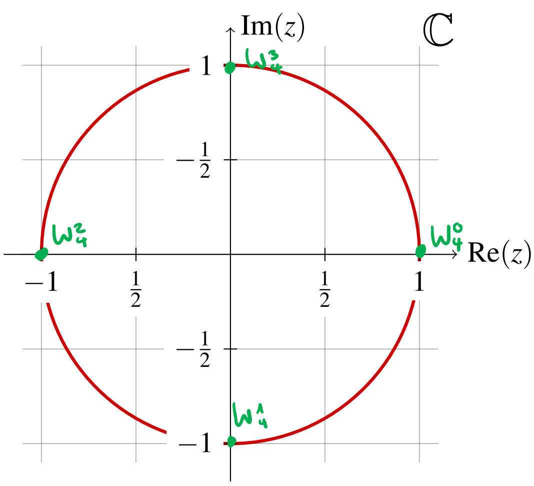
The solution of the exercises should be presented in a clear and concise manner. Numerical results should be accurate to 4 digits. The exercises are accepted if You solve 75% of the exercises adequately. Please hand in the exercises no later than at the end of the last lecture in semesterweek 12.

1 Nth root of unity

In the definition of the FFT we use the abbreviation $W_N = e^{-j\frac{2\pi}{N}}$.

1. Show that $W_N^N = 1$, i.e. the Nth power of W_N is equal to 1. Hence W_N is the Nth root of 1 (unity).
2. Show that $\sum_{k=0}^{N-1} W_N^k = 0$ by using the fact, that the terms in the sum W_N^k comprise a geometric series [1].
3. Similarely show that $\sum_{k=0}^{N-1} (W_N^n)^k = 0$ for $n = 2, 3, \ldots, N-1$.
4. Likewise show that $\sum_{k=0}^{N-1} (W_N^n)^k = N$ for $n = 0$.
5. Draw the 8 numbers $W_8^k = e^{-j\frac{2\pi}{8}}$ $^{\frac{-j\pi}{4}}$ $(k = 0, 1, \ldots, 7)$ in the left figure below. Verify the results from (2) to (4) using $N = 4$ in the right figure below!





[1] Remember: the sum of finite geometric series is given by $\sum_{k=0}^{N-1} a_0 q^k = a_0 \frac{1-q^N}{1-q}$.

Aufgabe 1

1) $W_N^N = 1 \Rightarrow W_N^N = \left(e^{-j\frac{2\pi}{N}}\right)^N = e^{-j\frac{2\pi}{N}\cdot N} = e^{-j\cdot 2\pi} = \underline{\underline{1}}$

$2\pi \rightarrow$ gerade 360 Grad

2) $\sum_{k=0}^{N-1} W_N^k = 0 \Rightarrow$

$$\boxed{\text{Sum of geometric series } \sum_{k=0}^{N-1} a_0 q^k = a_0 \frac{1-q^N}{1-q}}$$

siehe oben 1)

$$\overset{!}{=} \sum_{k=0}^{N-1} \left(e^{-j\frac{2\pi}{N}}\right)^k = \frac{1-\left(e^{-j\frac{2\pi}{N}}\right)^N}{1-\left(e^{-j\frac{2\pi}{N}}\right)} = \frac{1-W_N^N}{1-\left(e^{-j\frac{2\pi}{N}}\right)} = \frac{1-1}{1-\left(e^{-j\frac{2\pi}{N}}\right)} = \underline{\underline{0}}$$

3) $\sum_{k=0}^{N-1} (W_N^n)^k = 0$ for $n = 2, 3, \dots, N-1$

$$\sum_{k=0}^{N-1} (W_N^n)^k = \sum_{k=0}^{N-1} \left(e^{-j\frac{2n\pi}{N}}\right)^k = \frac{1-\left(e^{-j\frac{2n\pi}{N}}\right)^N}{1-\left(e^{-j\frac{2n\pi}{N}}\right)} = \frac{1-\left(W_N^N\right)^n}{1-\left(e^{-j\frac{2n\pi}{N}}\right)} = \frac{1-1^n}{1-\left(e^{-j\frac{2n\pi}{N}}\right)} = \underline{\underline{0}}$$

4) $\sum_{k=0}^{N-1} (W_N^n)^k = N$ for $n=0$

siehe Theorie: gibt immer 1

$$\overset{!}{=} \sum_{k=0}^{N-1} (W_N^0)^k = \sum_{k=0}^{N} 1 = N$$

2 DFT of a delta-impulse

Suppose we have the following sample $f = [1, 0, 0, 0]$. Using the **Kronecker delta** [2] we can write $f[n] = \delta_{0n}$. This signal corresponds to a delta-impuls at $n = 0$.

$\delta_{ij} = \begin{cases} 1, & \text{falls } i=j \\ 0, & \text{falls } i \neq j \end{cases}$

1. Compute the DFT using the formula (for $N = 4$)

$$F[k] = \sum_{n=0}^{\overset{(4-1)=3}{N-1}} f[n]\, W_N^{kn} \quad k = 0, 1, \ldots, (N-1)$$

Depict the $F[0]$, $F[1]$, $F[2]$, and $F[3]$ in the figure below.

$\sum_{n=0}^{3} \delta_{0n} W_4^{kn} = W_4^{k0} = 1$, wobei $k = 0,1,2,3$

Somit ist $F = [1, 1, 1, 1]$

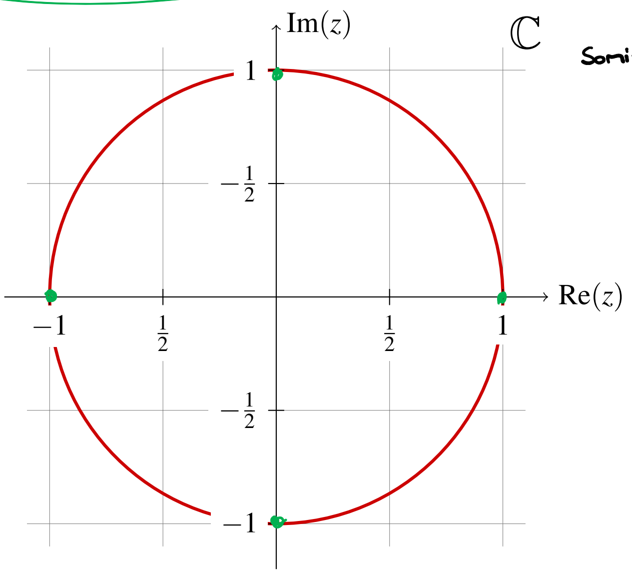
2. Using these results compute the inverse DFT using the formula (for $N = 4$)

$$f[n] = \frac{1}{N} \sum_{n=0}^{\overset{3}{N-1}} F[k]\, W_N^{-kn} \quad n = 0, 1, \ldots, (N-1).$$

$f[n] = \delta_{0n} = \begin{cases} 1 & \text{falls } n=0 \\ 0 & \text{falls } n \neq 0 \end{cases}$

$\mathbb{C}$

Somit ist $f = [1, 0, 0, 0]$



3 FFT with $N = 4$

Compute the discrete fourier transform of the sampled sequence $f = [f_0, f_1, f_2, f_3]$ using the FFT. First compute the normal DFT and then try to optimize using the FFT. Count the number of operations (multiplications) in each case.

Compare the result with Matlab/Octave for the real values $f = [2, 4, 6, 8]$. What do You notice?

4 FFT of a sin-wave

To generate a sample of sin-wave of frequency $f = 440$ Hz of length $T = 3$ seconds at a sampling rate of $f_s = 8192$ Hz the following Matlab/Octave commands can be used:

[2] The **Kronecker delta** is defined by $\delta_{ij} = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$, i.e. it is only 1 (one) if $i = j$, otherwise it is 0 (zero).

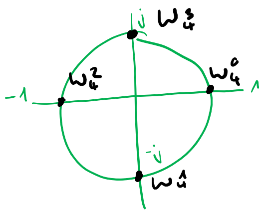
Aufgabe 2

direkt in Aufgabenstellung gelöst

Aufgabe 3

DFT for $f = [f_0, f_1, f_2, f_3]$, $W_4 = e^{-j\frac{2\pi}{N}}$, $f = [2, 4, 6, 8]$

$$\begin{bmatrix} F[0] \\ F[1] \\ F[2] \\ F[3] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} f_0 & + f_1 & + f_2 & + f_3 \\ f_0 & -jf_1 & -f_2 & +jf_3 \\ f_0 & -f_1 & +f_2 & -f_3 \\ f_0 & +jf_1 & -f_2 & -jf_3 \end{bmatrix}$$



→ reale f-Abtastwerte einsetzen

$$= \begin{bmatrix} 2 & +4 & +6 & +8 \\ 2 & -j4 & -6 & +j8 \\ 2 & -4 & +6 & -8 \\ 2 & +j4 & -6 & -j8 \end{bmatrix} = \begin{bmatrix} 20 \\ -4+4j \\ -4 \\ -4-4j \end{bmatrix}$$

FFT mit Octave

```
f = [2 4 6 8]';
fft(f)
```

```
ans =

   20 +  0i
   -4 +  4i
   -4 +  0i
   -4 -  4i
```

— Resultate sind identisch

```
fsHz = 8192;               % sampling frequency in Hz
dur  = 1;                  % duration of sample (3 seconds)
fHz  = 440;                % frequency of sin-wave in Hz
A    = 1;                  % amplitude of sin-wave
dt   = 1/fsHz;             % sample periode
ts   = 0:dt:dur-dt;        % sample times
f    = A*sin(2*pi*fHz*ts); % compute sound signal at sample times
wavwrite(f, "sound.wav")
system('play␣sound.wav')
```

The last two commands play the sound on the system.

To visualize the spectrum of the signal the following commands can be used:

```
N = fsHz * dur;
F = fft(f,N)/N;
absF = abs(F);
Faxis = linspace(-fsHz/2,fsHz/2,N);
figure(1)
clf
subplot(1,2,1)
plot(linspace(0,fsHz,N)/1000,absF);
box off;
axis([0 10 0 0.6])
subplot(1,2,2)
plot(Faxis/1000,fftshift(absF));
axis([-5 5 0 0.6])
xlabel('Frequency␣(KHz)')
```

The right figure depict the magnitude of the Fourier transform in the range from -$f_s/2$ to $f_s/2$ (f_s: sampling frequency).

Increase the frequency of the signal from $f = 440$ Hz to 1 kHz, 3 KHz, and 5.

In each case compute the Fourier transform and then the inverse Fourier transform; then listen to this reconstruction. What do You notice?

→ Ton wird höher und die Werte bewegen sich immer weiter von der 0 weg

5 2-D Fourier transform

First we generate a black image of 256 × 256 pixels

```
M = 256;
N = 256;
f = zeros(M, N);
```

and then place a white rectangle of 8 × 16 pixels in the middle.

Next we visualize the image, together with its Fouriertransform. First we visualize the amplitude (top right), then the logarithm of the amplitude (bottom left) and finally the same but this time the origin is shifted in the middle, i.e. the origin of the coordinate system of the frequency domain is in the middle, as opposed to the left upper corner. The DC component is in the middle, and high frequencies are at the boundary of the frequency domain.

```
M1 = 8;
N1 = 16;
f(M/2-M1/2+1:M/2+M1/2, N/2-N1/2+1:N/2+N1/2) = 1;
figure(1);
```

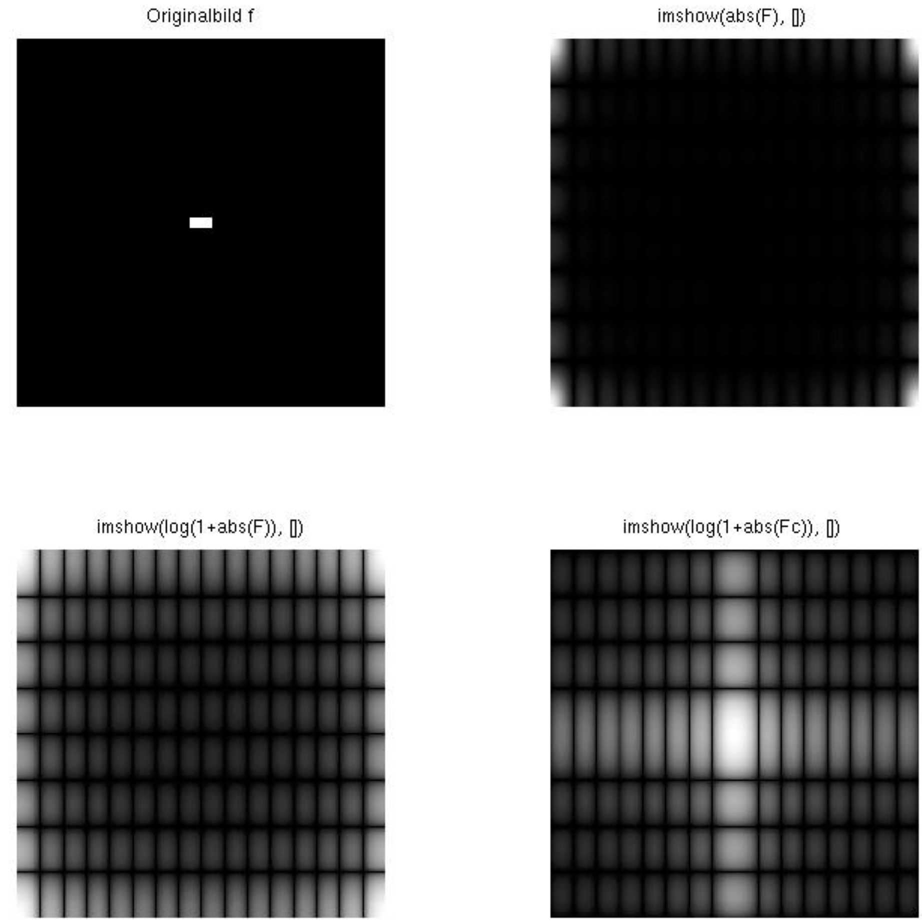


Figure 1: Image and its Fouriertransforms visualized in three different ways.

```
subplot(2,2,1)
imshow(f);
title('Originalbild f')
F = fft2(f);
subplot(2,2,2)
imshow(abs(F), [])
title('imshow(abs(F), [])')
subplot(2,2,3)
imshow(log(1+abs(F)), [])
title('imshow(log(1+abs(F)), [])')
Fc = fftshift(F);
subplot(2,2,4)
imshow(log(1+abs(Fc)), [])
title('imshow(log(1+abs(Fc)), [])')
```

Note that with the command

```
Fn = ifftshift(F);
```

the shift-operation can be reversed.

1. Change the size of the white rectangle from 8×8 to 16×32 and 32×64 and explain what You notice.
2. Set the amplitudes of high frequencies to zero, compute the inverse Fourier transform and visualize the result using Matlab/Octave commands as follows:

```
Fnew = F;
Fnew(M/4:M,:)=0;
Fnew(:,M/4:M)=0;
fnew = ifft2(Fnew);
imshow(abs(fnew), [])
```

   You should notice, that the rectangle has no longer sharp edges. Note: sharp edges can only be reconstructed using high frequencies which are now missing.

6 The aperiodic convolution

Achtung aperiodisch

Compute the aperiodic convolution using the `conv`-function from Matlab/Octave and using the FFT and iFFT.

7 The circular convolution

Program the circular convolution and check it using $x = [1,2,4,5,6]$ and $y = [7,8,9,3,0]$. You should get $[112,91,71,88,124]$. Check it also with the convolution theorem, i.e. `ifft(fft(x).*fft(y))`.

Have Fun!

Aufgabe 6

```
% Abtastwerte
f = [0 0.02 0.08 0.18 0.32 0.51 0.73 1];
g = [0 0.14 0.29 0.43 0.57 0.71 0.86 1];

% fft
F = fft([f zeros(1,length(g)-1)]);
G = fft([g zeros(1,length(f)-1)]);

% octave built in convolution function
conv(f,g)

% Piecewise multiplication of F and G and then inverse fft to get the convoluted frequency
ifft(F.*G)
```

```
ans =

 Columns 1 through 8:

   0.00000   0.00000   0.00280   0.01700   0.05700   0.14280   0.30140   0.56430

 Columns 9 through 15:

   0.97000   1.35660   1.66340   1.84690   1.84780   1.59000   1.00000

ans =

 Columns 1 through 8:

   0.00000   0.00000   0.00280   0.01700   0.05700   0.14280   0.30140   0.56430

 Columns 9 through 15:

   0.97000   1.35660   1.66340   1.84690   1.84780   1.59000   1.00000
```

=

Aufgabe 7

```
% O(n^2)
function f = circularconv(x,y)

    % get lenght of input
    len = size(x,1);

    for n = 1:len
        sum = 0;
        for k = 1:len
            % get the new position using modulo
            m = mod(n-k,len);
            % calculate the sum of every element
            sum += x(k)*y(m+1);
        end
        f(n) = sum;
    end
end
```

```
x = [1; 2; 4; 5; 6];
y = [7; 8; 9; 3; 0];



f = circularconv(x,y)
```

```
f =

   112    91    71    88   124
```

k			0	1	2	3	4
x[k]			1	2	4	5	6
y[k]			7	8	9	3	0
y[0-k]	9	8	7	0	3	9	8
y[1-k]	3	9	8	7	0	3	9
y[2-k]	0	3	9	8	7	0	3
y[3-k]	7	0	3	9	8	7	0
y[4-k]	8	7	0	3	9	8	7

$(x \circledast y)[0] = 1\cdot 7 + 2\cdot 0 + 4\cdot 3 + 5\cdot 9 + 6\cdot 8 = 112$

$(x \circledast y)[1] = 91$

$(x \circledast y)[2] = 71$

$(x \circledast y)[3] = 88$

$(x \circledast y)[4] = 124$

somit ist $(x \circledast y)[n] = [124, 91, 71, 88, 124]$