# Homework for Digital Signal Processing
*Sheet 7*

**Exercise 1.** Implement a function which takes a vector $\vec{f} \in \mathbb{R}^n$ as input and returns $B^*\vec{f}$. As discussed in the lecture the function should not execute $n^2$ multiplications but only $(n^2+n)/2$. Take the algorithm from the lecture as a guideline. Compare the results with a "normal" matrix multiplication using some random test vectors $\vec{f}$. Due to rounding errors small deviations may occur.

**Exercise 2.** Prove that

$$\sum_{\ell=0}^{n-1} f_\ell e^{-2\pi jk\ell/n}$$

$$= \sum_{\ell=0}^{n/2-1} f_\ell^{(e)} e^{-2\pi jk\ell/(n/2)} + e^{-2\pi jk/n} \sum_{\ell=0}^{n/2-1} f_\ell^{(o)} e^{-2\pi jk\ell/(n/2)}$$

for $k = 0, 1, \ldots, n/2 - 1$ where

$$f_\ell^{(e)} = f_{2\ell}$$
$$f_\ell^{(o)} = f_{2\ell+1}, \qquad \ell = 0, 1, \ldots, n/2 - 1.$$

Note that vector $\vec{f}^{(e)}$ contains the even indexed components of $\vec{f}$ and $\vec{f}^{(o)}$ the odd ones.

Hint: Begin by splitting the sum into two sums where the first runs over even $\ell$ and the second over odd $\ell$. Next, transform the summation indices such that both sums run from 0 to $n/2 - 1$. Further, use

$$\frac{2\ell}{n} = \frac{\ell}{n/2}.$$

**Exercise 3.** Prove that

$$a_{k+n/2} = a_k$$
$$b_{k+n/2} = b_k.$$

where

$$a_k = \sum_{\ell=0}^{n/2-1} f_\ell^{(e)} e^{-2\pi jk\ell/(n/2)}$$

$$b_k = \sum_{\ell=0}^{n/2-1} f_\ell^{(o)} e^{-2\pi jk\ell/(n/2)}.$$

Hint: $a_{k+n/2}$ is obtained from $a_k$ if each $k$ is replaced by $(k + n/2)$. All you have to do is apply the properties of the exponential function and

$$e^{-2\pi j\ell} = 1$$

for all integers $\ell$.

**Exercise 4.** Implement the $n$-FFT as a recursive function where $n$ is a power of two. The FFT has to give the same results as the DFT up to rounding errors. Test your results with some random vectors $\vec{f}$.

Count in a global variable the number of complex multiplications and verify that it is in fact ony $\frac{1}{2}n\,\mathrm{ld}(n)$.

Make sure to compute a vector with entries

$$e^{2\pi jk/n}, \quad k = 0, 1, \ldots, n/2 - 1$$

*in advance* and store it either as a global variabel or pass it to the FFT function. As the evaluation of a comlex $e$-function is expensive and your system will compute many FFTs, it is essential that those exponentials are not recomputed in each call.

**Exercise 5.** Implement a function for the computation of $B^* \vec{f}$ with the FFT.

- Implement the function first in a straight forward way with recursive function calls.
- Next implement the iterative version of the FFT. You have to re-arrange the components of the input vector $\vec{f}$ according to the bit reverse scheme.
- **Important:** The computation of the coefficients $e^{-2\pi jk/n}$ has to be done in a separate function in advance. Pass those values to the FFT function as a parameter. In later applications the FFT is executed many times consecutively with different input vectors $\vec{f}$ and it would be inefficient to recompute those coefficients each time.

The results of recursive and iterative FFT have to be exactly identical even in the presence of rounding errors as the same operations are executed.

**Exercise 6.** Modify your implementation of the FFT such that the inverse FFT $\vec{f} = B\vec{z}$ is computed. Essentially you merely have to change one sign. You will figure this out by studying the derivation of the FFT as discussed in the lecture.

**Exercise 7.** In the lecture we computed the components $z_k$ of the FFT using formulas

$$
\begin{aligned}
z_k &= a_k + b_k r_k \\
z_{k+n/2} &= a_k - b_k r_k, \qquad k = 0, \ldots, n/2 - 1.
\end{aligned}
$$

This means that the computation of $z_k$ for $k = n/2, \ldots, n-1$ costs no more multiplications, but some additions. However, we could have computed those $z_k$ also using

$$
\begin{aligned}
z_{n-k} &= \overline{z_k}, \qquad k = 1, \ldots, n/2 - 1 \\
z_{n/2} &= 0
\end{aligned}
$$

which would save these additions. What is the reason why we did not do this?