Heilbronn, 18.11.2025 Prof. Dr. V. Stahl WS 25/26

Homework for Digital Signal Processing with Solutions

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.

Solution for Exercise 1. Programming exercise

Exercise 2. Prove that

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

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

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

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 ℓ and the second over odd ℓ. Next, transform the summation indices such that both sums run from 0 to $n/2-1$. Further, use

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

Solution for Exercise 2.

$$\begin{aligned}
&\sum_{\ell=0}^{n-1} f_\ell e^{-2\pi jk\ell/n} \\
&\quad = \sum_{\ell=0,2,4,\ldots} f_\ell e^{-2\pi jk\ell/n} + \sum_{\ell=1,3,5,\ldots} f_\ell e^{-2\pi jk\ell/n} \\
&\quad = \sum_{\ell=0}^{n/2-1} f_{2\ell} e^{-2\pi jk2\ell/n} + \sum_{\ell=0}^{n/2-1} f_{2\ell+1} e^{-2\pi jk(2\ell+1)/n} \\
&\quad = \sum_{\ell=0}^{n/2-1} f_\ell^{(e)} e^{-2\pi jk\ell/(n/2)} + \sum_{\ell=0}^{n/2-1} f_\ell^{(o)} e^{-2\pi jk\ell/(n/2)} e^{-2\pi jk/n} \\
&\quad = \sum_{\ell=0}^{n/2-1} f_\ell^{(e)} e^{-2\pi jk\ell/(n/2)} + e^{-2\pi jk/n} \sum_{\ell=0}^{n/2-1} f_\ell^{(o)} e^{-2\pi jk\ell/(n/2)}
\end{aligned}$$

Exercise 3. Prove that

$$\begin{aligned} a_{k+{}^n\!/_2} &= a_k \\ b_{k+{}^n\!/_2} &= b_k. \end{aligned}$$

where

$$\begin{aligned} a_k &= \sum_{\ell=0}^{{}^n\!/_2-1} f_\ell^{(e)} e^{-2\pi j k\ell/(n/2)} \\ b_k &= \sum_{\ell=0}^{{}^n\!/_2-1} f_\ell^{(o)} e^{-2\pi j k\ell/(n/2)}. \end{aligned}$$

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 ℓ.

Solution for Exercise 3.

$$\begin{aligned} a_{k+{}^n\!/_2} &= \sum_{\ell=0}^{{}^n\!/_2-1} f_\ell^{(e)} e^{-2\pi j(k+n/2)\ell/(n/2)} \\ &= \sum_{\ell=0}^{{}^n\!/_2-1} f_\ell^{(e)} e^{-2\pi j(n/2)\ell/(n/2)} e^{-2\pi j k\ell/(n/2)} \\ &= \sum_{\ell=0}^{{}^n\!/_2-1} f_\ell^{(e)} \underbrace{e^{-2\pi j\ell}}_{=1} e^{-2\pi j k\ell/(n/2)} \\ &= \sum_{\ell=0}^{{}^n\!/_2-1} f_\ell^{(e)} e^{-2\pi j k\ell/(n/2)} \\ &= a_k \end{aligned}$$

$$\begin{aligned} b_{k+{}^n\!/_2} &= \sum_{\ell=0}^{{}^n\!/_2-1} f_\ell^{(o)} e^{-2\pi j(k+n/2)\ell/(n/2)} \\ &= \sum_{\ell=0}^{{}^n\!/_2-1} f_\ell^{(o)} e^{-2\pi j(n/2)\ell/(n/2)} e^{-2\pi j k\ell/(n/2)} \\ &= \sum_{\ell=0}^{{}^n\!/_2-1} f_\ell^{(o)} \underbrace{e^{-2\pi j\ell}}_{=1} e^{-2\pi j k\ell/(n/2)} \\ &= \sum_{\ell=0}^{{}^n\!/_2-1} f_\ell^{(o)} e^{-2\pi j k\ell/(n/2)} \\ &= b_k \end{aligned}$$

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.

Solution for Exercise 4. Programming exercise.

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.

Solution for Exercise 5. Programming project.

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.

Solution for Exercise 6. Programming exercise.

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?

Solution for Exercise 7. Fourier coefficients appear in conjugate pairs only if the time signal f is real. The second approach would therefore not work for the inverse FFT. Further $z_{n/2} = 0$ holds only if the time domain signal is band limited and the sampling rate is more than twice the cutoff frequency.

It is however possible to compute the FFT's of two *real* signals $\vec{f}, \vec{g}$ with only one complex FFT of $\vec{f} + j\vec{g}$ using linearity of the FFT and the conjugate complex pair property of the Fourier coefficients.

With some additional tricks it is also possible to split a signal vector $\vec{f}$ into two vectors of length $n/2$, compute the FFT's of both with one complex FFT of order $n/2$, and reconstruct the FFT of the original $\vec{f}$ from the result. This gives a speedup of a *real* FFT of (almost) factor two compared to a complex FFT.