MATLAB Exercise 2 – Discrete Fourier Transform and Block Processing

Digital Signal Processing
Sommersemester 2022
S. Doclo, D. Fejgin

04.07.2022

The objective in this exercise is to develop an understanding of digital filter implementation through DFT–based block processing. Section 1 focuses on the circular convolution property of the DFT and the relationship between the linear convolution and the circular convolution. Section 2 focuses on how to employ DFT to implement linear time–invariant filtering.

1 Circular convolution

The DFT domain filtering uses the fact that the circular convolution of $u[n]$ and $h[n]$ corresponds to the multiplication of their DFTs, i.e., $U[k]\,H[k]$. However, some samples of the circular convolution output may be not same as the linear convolution output, because of the time–aliasing inherent in circular convolution. It is therefore necessary to identify those samples, particularity for the block processing (overlap-save algorithm).

1. Write a MATLAB function `myCircConv(u,h,M)` to implement a circular convolution in DFT domain for an input signal. The function inputs should include an input signal $u[n]$, a filter $h[n]$ of length N and a DFT size of length $M > N$. *Hints*: i) Using the MATALB function `fft` instead of DFT matrix $\mathbf{W}$ in the below stated notation is more computationally efficient, ii) Proper consideration of the dimensions of the employed matrices and vectors is crucial!

   To remind you, the input signal $u[n]$ is split into blocks of length M with no overlap between blocks so that the l^{th} block of time samples can be expressed as

   $$\mathbf{u}_l = [u[lM], \ldots, u[lM + M - 1]]^T$$

   The block circular convolution output in the DFT domain is computed as

   $$\mathbf{y}_l[k] = \mathbf{U}_l[k]\,\mathbf{h}_{\text{zp}}[k]$$

   where $\mathbf{y}_l[k] = [Y_l[0], \ldots, Y_l[M-1]]^T$ is the vector of stacked frequency bins $Y_l[k]$. $\mathbf{U}_l[k]$ denotes a $M \times M$ frequency representation diagonal matrix of $\mathbf{u}_l[k]$, i.e.,

   $$\mathbf{U}_l[k] = \text{diag}\{\mathbf{u}_l[k]\} = \text{diag}\{\mathbf{W}\mathbf{u}_l\}$$

   with the $M \times M$ dimensional DFT matrix $\mathbf{W}$, and $\mathbf{h}_{\text{zp}}[k]$ denotes the frequency representation vector of the zero-padded filter (after appending zeroes, filter has length M), i.e.

   $$\mathbf{h}_{\text{zp}}[k] = \mathbf{W}\left[\mathbf{h}^T[n], \mathbf{0}^T\right]^T$$

   with $\mathbf{h}[n] = [h[0], \ldots, h[N-1]]$, and $\mathbf{0}$ the vector of length $M - N$ containing only zeroes. The block circular convolution output in the time domain is then computed as

   $$\mathbf{y}_l = \mathbf{W}^{-1}\mathbf{y}_l[k]\,.$$

2. Consider the following input signal and filter,

$$u[n] = \begin{cases} (0.9)^n & 0 \le n \le 12 \\ 0 & \text{otherwise}, \end{cases} \tag{1}$$

$$h[n] = \begin{cases} 1 & 0 \le n \le 11 \\ 0 & \text{otherwise}, \end{cases} \tag{2}$$

   (a) Use your implemented code to compute the circular convolution output using a 16–point DFT

   (b) Try a linear convolution in the time domain using MATLAB function `conv`. Determine the output indices where the circular convolution output derived in (a) are the same as the linear convolution output.

   (c) Try a a 16–point circular convolution in the time domain using MATLAB function `cconv`. Compare this output with the output in (a)

   (d) How to obtain less samples corrupted by the time–aliasing in the circular convolution output?

3. Consider 'speech.wav' as the input signal. This signal was recorded with a sampling frequency of $f_s = 16000$ Hz. Generate a random time-domain filter $h[n]$ with a length of 16 ms via MATLAB function `randn`. Normalize $h[n]$ such that its energy would be 1. Save the filter since it will be required for the next implementation (overlap– save algorithm).

   - Use your implemented code to compute the circular convolution output using a DFT size of 16 ms.

2 Overlap–save Implementation

In this part, implementation of the overlap–save algorithm is aimed.

1. Write a MATLAB function to implement the overlap–save algorithm. The function inputs should include an input signal $u[n]$, a filter $h[n]$ of length N and a DFT size $M = 2N$. To remind you, unlike the previous implementation, $\mathbf{u}_l$ includes now two consecutive blocks (old and new blocks), each block of length N, i.e.

$$\mathbf{u}_l = [u[lN - N], \ldots, u[lN + N - 1]]^T.$$

   In the DFT domain, the circular convolution output is computed as

$$\mathbf{y}_l[k] = \mathbf{U}_l[k]\,\mathbf{h}_{\text{zp}}[k]$$

   where $\mathbf{y}_l[k] = [Y_l[0], \ldots, Y_l[M-1]]^T$ is the vector of stacked frequency bins $Y_l[k]$. $\mathbf{U}_l[k]$ is a $M \times M$ frequency representation diagonal matrix of $\mathbf{u}_l$, i.e.,

$$\mathbf{U}_l[k] = \text{diag}\{\mathbf{u}_l[k]\} = \text{diag}\{\mathbf{W}\mathbf{u}_l\}$$

   with the $M \times M$ dimensional DFT matrix $\mathbf{W}$, and $\mathbf{h}_{\text{zp}}[k]$ denotes the frequency representation vector of the zero-padded filter (after appending zeroes, filter has length M), i.e.

$$\mathbf{h}_{\text{zp}}[k] = \mathbf{W}\left[\mathbf{h}^T[n], \mathbf{0}^T\right]^T$$

   with $\mathbf{h}[n] = [h[0], \ldots, h[N-1]]$, and $\mathbf{0}$ the vector of length N containing only zeroes. In the time domain, the filtered output is then computed as

$$\mathbf{y}_l = \mathbf{K}\mathbf{W}^{-1}\mathbf{y}_l[k]\,; \quad \mathbf{K} = [\mathbf{0}_N, \mathbf{I}_N]$$

with $\mathbf{0}_N$ the $N \times N$-dimensional matrix containing only zeroes and $\mathbf{I}_N$ the $N \times N$-dimensional identity matrix. The matrix $\mathbf{K}$ contributes to the time-domain output signal by skipping the output first half (which has been corrupted by the time-aliasing) and retaining the output last half (which equals to the linear filtering output).

2. Use the input signal from Eq. (1) and the filter from Eq. (2) to compute the filter output using a 32–point DFT. Compare the output with the output of task 2 from section 1. Is there a difference between the outputs? Explain why?

3. Use 'speech.wav' as the input signal and the filter you generated in task 3 from the previous section. Compute the filter output using a DFT size of 32 ms. Compare this output with the output of task 3 from the previous section. Is there a difference between the outputs? Explain why?