Digital Signal Processing SS 2019/20

Exercise Sheet 3

Due date: 24.5.2019, 10:00h
(time of the lecture)

Problem 1

When the sequences $x[n]$ and $h[n]$ are of finite duration N_x and N_h, respectively, then their linear convolution $y[n] = \sum_k x[k]h[n-k]$ can also be implemented using matrix-vector multiplication. If elements of $y[n]$ and $x[n]$ are arranged in column vectors x and y respectively, then we obtain $y = Hx$ where linear shifts in $h[n-k]$ for $n = 0, ..., N_h - 1$ are arranged as rows in the matrix H. This matrix has an interesting structure and is called a Toeplitz matrix. To investigate this matrix, consider the sequences

$$x[n] = \{\underset{\uparrow}{1}, 2, 3, 4, 5\} \quad \text{and} \quad h[n] = \{\underset{\uparrow}{6}, 7, 8, 9\}$$

1. Determine the linear convolution $y[n] = (h * x)[n]$.
2. Express $x[n]$ as a 5×1 column vector x and $y[n]$ as a 8×1 column vector y. Now determine the 8×5 matrix H so that $y = Hx$.
3. Characterize the matrix H. From this characterization can you give a definition of a Toeplitz matrix? How does this definition compare with that of time invariance?
4. What can you say about the first column and the first row of H?

Problem 2

MATLAB provides a function called `toeplitz` to generate a Toeplitz matrix, given the first row and the first column.

1. Using this function and your answer to Problem 1.4, develop another MATLAB function to implement linear convolution. The format of the function should be

```
function [y,H]=conv_tp(h,x)
% Linear Convolution using Toeplitz Matrix
% ----------------------------------------
% [y,H] = conv_tp(h,x)
% y = output sequence in column vector form
% H = Toeplitz matrix corresponding to sequence h so that y = Hx
% h = Impulse response sequence in column vector form
% x = input sequence in column vector form
```

2. Verify your function on the sequences given in Problem 1.

Problem 3

Analyse the complexity of three implementations of convolution by experiments. Implement three MATLAB functions: (1) your own version based on nested for-loops, (2) the above one, `conv_tp`, with Toeplitz matrices, and also use (3) the MATLAB function `conv`.

Compute convolutions $x * y$, where x, y are signals of equal length n, like `ones(1,n) = [1.0,1.0,...,1.0]`. Use the MATLAB function `timeit` to record the run times required by each function to convolve the signals for increasing lengths n, like

$$n = 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000.$$

Use the MATLAB function `polyfit` for polynomial curve fitting for the timings of the three functions for convolution. Plot the data and the fitting polynomials. Discuss the results: How well do the fitting functions approximate the measured data? What are the resulting complexities? What are the multiplicative constants in these? The expected result is that the first two methods have complexity $O(n^2)$. In contrast, the MATLAB function `conv` cannot be fit by a lower degree polynomial and should be $O(n \log n)$. Try to get this loglinear fit of the data using the MATLAB function `fitglm`.