Digital Signal Processing

Jiří Málek

Part I

Discrete Fourier Transform (DFT)

Comparison of DFT and DTFT

- **DTFT:** *infinite, aperiodic, discrete signals* $x[n]$
- DTFT output: *continuous, complex, periodic spectrum* $X(e^{j\omega})$ (period 2π)
- **DFT:** *infinite, periodic, discrete signals* $\tilde{x}[n]$
- In practice: the computation is performed on a single period $x[n]$ (of duration N samples) of the periodic signal $\tilde{x}[n]$
- DFT output: *discete, complex, periodic* spectrum $\tilde{X}[k]$ (period N)
- In practice: only single period $X[k]$ (N-point DFT) of the periodic spectrum $\tilde{X}[k]$ is computed
- The values of the *DFT spectrum* can be obtained by sampling of DTFT spectrum at frequencies $\omega = \frac{2\pi k}{N}, k = 0, 1, \ldots N-1$
- The distance between frequency samples $\Delta\omega = \frac{2\pi}{N}$ is called *frequency resolution*

DFT properties I

- **Linearity:** Let signals $x_1[n]$ and $x_2[n]$ have spectrum $X_1[k]$ and $X_2[k]$. Then it holds

$$ax_1[n] + bx_2[n] \overset{DFT}{\Longleftrightarrow} aX_1[k] + bX_2[k] \tag{1}$$

- Sequences must have equal duration, if not, the shorter one is zero-padded
- **Symmetry:** If $x[n] \in \mathcal{R}$, then $X[k]$ is conjugate symmetric:

$$X[k] = X^*[-k] = X^*[N-k]_N \tag{2}$$

- If $x[n]$ is imaginary, then $X[k]$ is conjugate antisymmetric:

$$X[k] = -X^*[-k] = -X^*[N-k]_N \tag{3}$$

- EXAMPLE: DFT symmetry

DFT properties II

- **Simplified notation:**

$$W_N \overset{def.}{=} e^{-j2\pi/N} \tag{4}$$

$$W_N^{nk} \overset{def.}{=} e^{-j2\pi kn/N} \tag{5}$$

- **Circular shift:** Circular shift by n_0 samples is defined by

$$(x[n-n_0])_N R_N[n] = \tilde{x}[n-n_0]R_N[n] \tag{6}$$

where $R_N[n]$ is rectangular window of length N

- Example: Circular shift
- Circular shift in the time domain causes in the frequency domain a change of the phase spectrum

$$(x[n-n_0])_N R_N[n] \overset{DFT}{\Longleftrightarrow} W_N^{n_0 k} X[k] \tag{7}$$

DFT properties III

- **Circular convolution:** Let $x[n]$ and $h[n]$ be two finite sequences with N-point DFTs $X[k]$ and $H[k]$, then sequence with DFT equal to $Y[k] = H[k]X[k]$ is given by a formula

$$y[n] = x[n] \circledN h[n] = \left[\sum_{k=0}^{N-1} h[k]\tilde{x}[n-k] \right] R_N[n] \qquad (8)$$

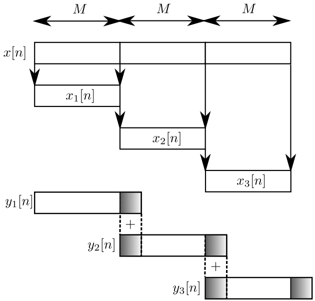
- This is a convolution of $h[n]$ with periodical $\tilde{x}[n]$, evaluated using only single period of $\tilde{x}[n]$
- *Circular convolution* is generally not equal to *linear convolution* (filtering), even lengths of these two operations differ.
- Using suitable zero-padding, both operations coincide (give the same result).
- Then, the circular convolution can be used for fact computation of linear convolution (using Fast Fourier Transform - FFT)

Computation of linear convolution via the circular one

- DFT and circular convolution can be used for effective computation of linear convolution
- Having two finite sequences $h[n]$ and $x[n]$ with lengths N_1 and N_2, respectively, the linear convolution can be computed as follows:
  1. Zero-padding of $h[n]$ and $x[n]$ to length $N \geq N_1 + N_2 - 1$
  2. Computation of N-point DFT of sequences $h[n]$ and $x[n]$
  3. Multiplication $Y[k] = H[k]X[k]$
  4. Inverse DFT of $Y[k]$
- This procedure becomes *much less computationally demanding* then the definition formula of convolution, when FFT is used to compute the DFT.
- Unlike the definition formula, this procedure is not suitable for very long sequences $x[n]$
  - It requires the knowledge of the whole $x[n]$ (which disables real-time processing)
  - $h[n]$ is usually much shorter than $x[n]$ (a lot of zero-padding)
- These negatives are mitigated using *block-wise computation of convolution*

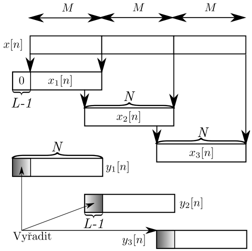
Overlap-Add (Block-wise processing)

- **Overlap-Add** (OA,OLA) is an efficient approach to compute convolution of a *long signal* $x[n]$ with impulse response of a FIR filter $h[n]$ (length L)
  - Signal $x[n]$ is split to non-overlapping sequences $x_i[n]$ of length M
  - Output signal $y[n]$ can be expressed as a sum of partial convolutions $y_i[n] = x_i[n] * h[n]$
  - The partial convolutions $y_i[n]$ have lengths $N = L + M - 1$ and are added with shift M ($L - 1$ samples of $y_i[n]$ and $y_{i-1}[n]$ thus overlap)
  - Computation of $y_i[n]$ is performed via zero-padded circular convolution using FFT



Overlap-Save (Block-wise processing)

- **Overlap-Save:** is an alternative to Overlap-Add, it efficiently computes convolution of a *long signal* $x[n]$ with impulse response of a FIR filter $h[n]$ (length L)
- $x[n]$ is split (with overlap of $L-1$ samples) into subsequences $x_i[n]$ of length N
- Overlap-Save computes the classing convolution via suitable concatenation of parts of circular convolutions involving subsequences $x_i[n]$.
- PRINCIPLE: Considering the circular convolution $x_1[n] Ⓝ h[n]$, the first $L-1$ samples differ from $x_1[n] * h[n]$, the remaining $M = N - L + 1$ samples are equal.
- These M samples constitute one interval of the output $y[n]$

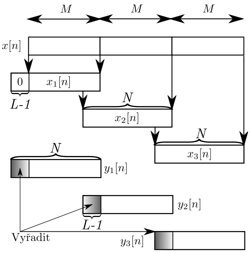


Overlap-Save II - Algorithm

1. Formation of the sequence $x_1[n]$

$$x_1[n] = \begin{cases} 0 & 0 \leq n < L-1 \\ x[n-L+1] & L-1 \leq n \leq N-1 \end{cases} \quad (9)$$

2. Computation of $x_1[n] \circledN h[n]$ using DFT.
   $L-1$ samples differ from *linear convolution*.
   Last $M = N - L + 1$ values of $x_1[n] \circledN h[n]$ are the first samples of $y[n]$



3. Formation of subsequence $x_2[n]$, where first $L-1$ samples overlap with the last samples of $x_1[n]$
4. Computation of $x_2[n] \circledN h[n]$ using DFT.
   $L-1$ samples differ from the linear convolution.
   Last $M = N - L + 1$ values form the second interval of $y[n]$
5. Steps 3. and 4. are repeated until the whole linear convolution is evaluated

Part II

Fast Fourier Transform (FFT)

Fast Fourier Transform

- **Fast Fourier Transform** is a group of algorithms allowing optimized computation of DFT and IDFT
- DFT transforms finite (or infinite periodic) sequence of time-domain samples into finite sequence of frequency components
- Computational complexity od DFT computed by definition is $O(N^2)$
- FFT is able to compute the same result in $O(N \log(N))$ operations
- The difference in computational complexity becomes apparent for growing N
- Due to FFT, the DFT algorithm is used into many scientific areas (signal processing, image processing, solution of differential equations etc.)

Fast Fourier Transform II

- Many FFT algorithms stem from factorization of the sample number N
- The N-point DFT of $x[n]$ is computed via several transforms applied to subsequences of $x[n]$
- However, even implementations suitable for prime N have been discovered
- The FFT algorithm can easily compute the IDFT (which differs by a sign in the exponent and normalization)
- **Comparison of FFT and DFT:**
- Evaluation of DFT by definition requires N^2 complex multiplications and $N(N-1)$ complex summations
- The most famous FFT version radix-2 Cooley-Tukey is suitable for N equals power of 2
- It requires $(N/2)\log_2(N)$ complex multiplications and $N\log_2(N)$ complex summations

Fast Fourier Transform III

- **Radix-2 Cooley-Tukey FFT:** Algorithm designed for sequences of length $N = 2^k,\ k \in \mathcal{Z}$
- Computational savings are achieved due to periodicity of the complex exponentials and the possibility to compute N-point DFT using two $N/2$-point DFTs
- The algorithm is recursive but can be computed non-recursively, if the input samples are suitably permuted
- DETAILS: Radix-2 Cooley-Tukey FFT

Thank you for attention!