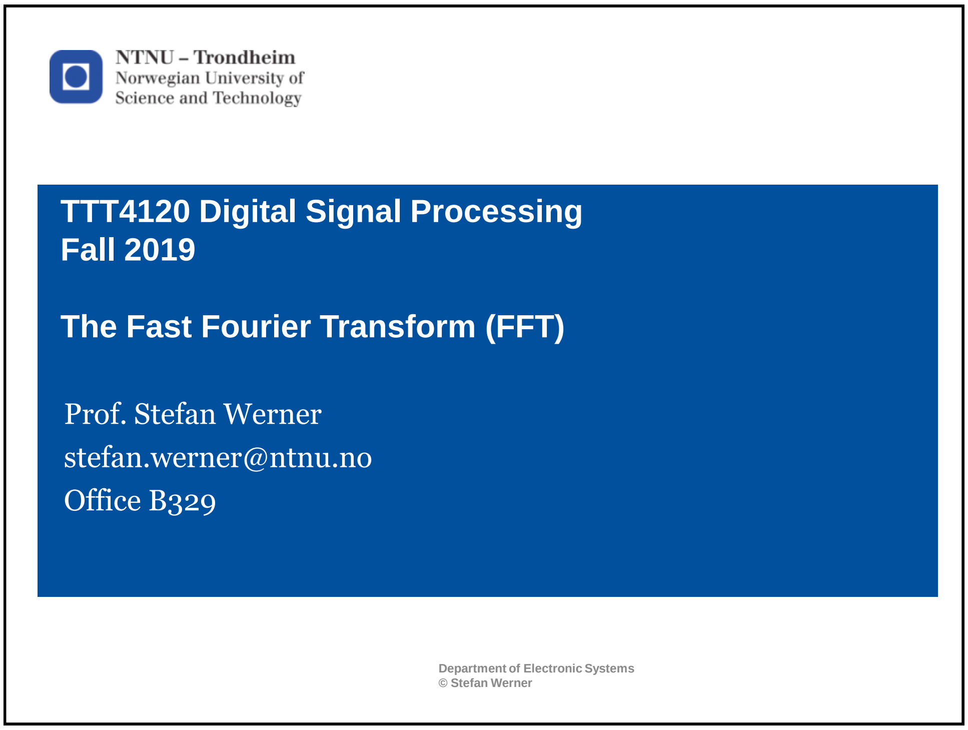
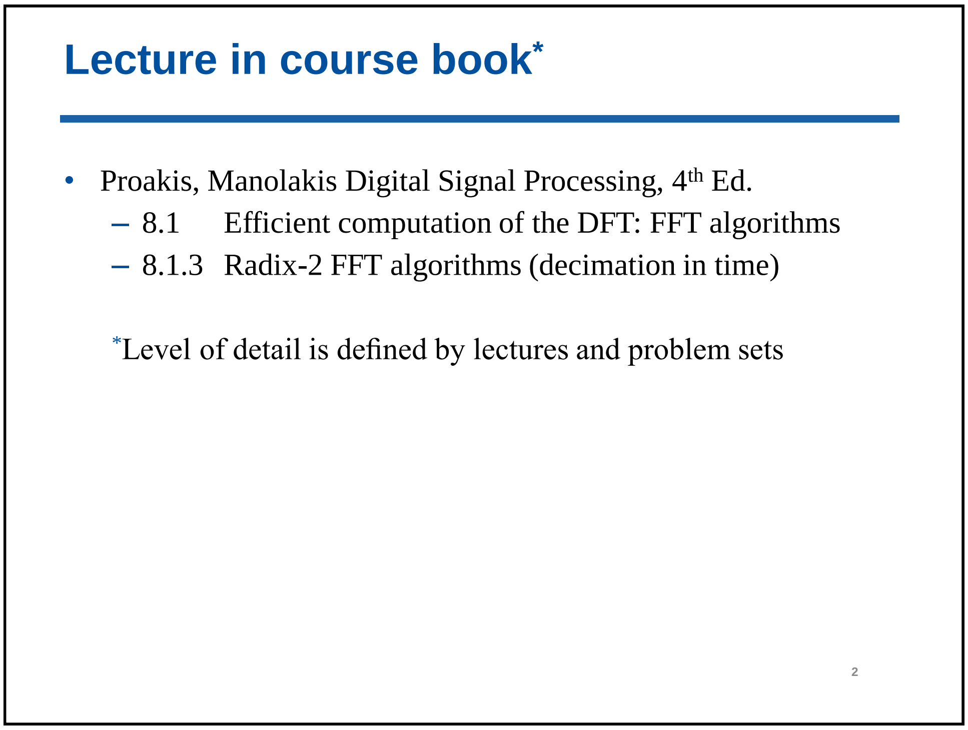


NTNU – Trondheim
Norwegian University of
Science and Technology

TTT4120 Digital Signal Processing
Fall 2019

The Fast Fourier Transform (FFT)

Prof. Stefan Werner
stefan.werner@ntnu.no
Office B329



Lecture in course book*

- Proakis, Manolakis Digital Signal Processing, 4th Ed.
  - 8.1 Efficient computation of the DFT: FFT algorithms
  - 8.1.3 Radix-2 FFT algorithms (decimation in time)

*Level of detail is defined by lectures and problem sets

2

Contents and learning outcomes

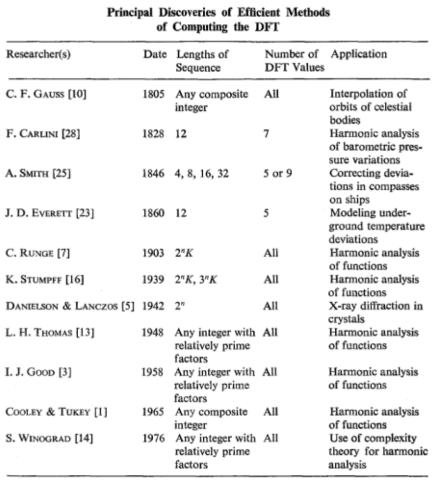
- Complexity of the DFT
- Divide and conquer
- Properties of $e^{j2\pi nk/N}$
- Radix-2 FFT

Motivation

- *Fast Fourier transforms* (FFTs) have revolutionized DSP
- What is the FFT?
  - DFT calculation made much faster
  - Speedup increases with DFT size
- Focus on the simplest formulation: the radix-2 decimation-in-time FFT algorithm

Motivation

- Some dates:
  - Gauss 1805
  - Cooley and Tukey 1965



Principal Discoveries of Efficient Methods of Computing the DFT

Researcher(s)	Date	Lengths of Sequence	Number of DFT Values	Application
C. F. Gauss [10]	1805	Any composite integer	All	Interpolation of orbits of celestial bodies
F. Carlini [28]	1828	12	7	Harmonic analysis of barometric pressure variations
A. Smith [25]	1846	4, 8, 16, 32	5 or 9	Correcting deviations in compasses on ships
J. D. Everett [23]	1860	12	5	Modeling underground temperature deviations
C. Runge [7]	1903	2^nK	All	Harmonic analysis of functions
K. Stumpff [16]	1939	2^nK, 3^nK	All	Harmonic analysis of functions
Danielson & Lanczos [5]	1942	2^n	All	X-ray diffraction in crystals
L. H. Thomas [13]	1948	Any integer with relatively prime factors	All	Harmonic analysis of functions
I. J. Good [3]	1958	Any integer with relatively prime factors	All	Harmonic analysis of functions
Cooley & Tukey [1]	1965	Any composite integer	All	Harmonic analysis of functions
S. Winograd [14]	1976	Any integer with relatively prime factors	All	Use of complexity theory for harmonic analysis

Michael T. Heideman, Don H. Johnson, and C. Sideny Burrus. Gauss and the History of the Fast Fourier Transform. Archive for History of Exact Sciences (Springer), 34(3):265-277, September 1985.

Complexity of the DFT

- DFT involves computing the sequence:

$$X(k) = \sum_{n=0}^{N-1} x[n]e^{-j2\pi nk/N}$$
$$= \sum_{n=0}^{N-1} x[n]W_N^{nk}$$

- IDFT involves computing the sequence

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X(k)W_N^{-nk}$$

- Similar calculations (same phasors but different direction)

  ⇒ concentrate on the DFT

Complexity of the DFT…

- What do we mean by computational efficiency?
  - Number of additions
  - Number of multiplications
  - Memory requirements
  - Scalability

Complexity of the DFT…

$$X(k) = \sum_{n=0}^{N-1} x[n]e^{-j2\pi nk/N}\,, k = 0, 1, \dots, N-1$$

- For each $k, n = 0, \dots, N-1$
  - Evaluate $e^{-j2\pi nk/N} = \cos\frac{2\pi kn}{N} - j\sin\frac{2\pi kn}{N}$
  - Multiply two complex numbers $x[n]e^{-j2\pi nk/N}$
- Total complexity for direct computation
  - $2N^2$ evaluations of trigonometric functions
  - $4N^2$ real multiplication
  - $2N(2N-1)$ real additions
- In addition indexing and addressing operations

Complexity of the DFT…

- Direct computation of the DFT is highly inefficient
- Complexity grows with the square of the signal length
- Severely limits the practical use for long signals
- How to reduce the complexity of the DFT?
  - Divide-and conquer approach
  - Exploit symmetry and periodicity properties
- Resulting algorithm will have complexity $N\log_2 N$!

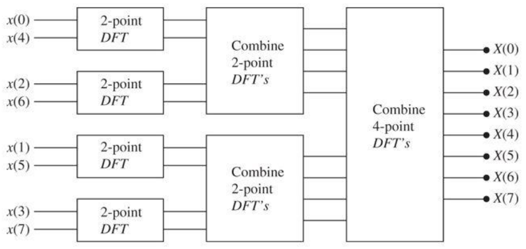
9

Divide-and-conquer

- FFT algorithms are based on a divide-and-conquer approach:
  - Divide a problem instance into subproblems
  - Conquer the subproblems by solving them recursively
  - Combine the solutions for the subproblems to a solution for the original problem
- Example 1: You are only capable of adding two numbers. How to efficiently compute the sum $S_N = \sum_{n=0}^{N-1} x[n]$?
  Consider the computation time for cases:
  1) Single person is assigned the task
  2) A group of persons is assigned the task

10

Divide-and-conquer…



- The approach taken in deriving the FFT is to recursively divide the N-point DFT into successively smaller DFTs
- Exploit symmetry and periodicity of $W_N^{kn} = e^{-j2\pi kn/N}$

Properties of $e^{j2\pi nk/N}$

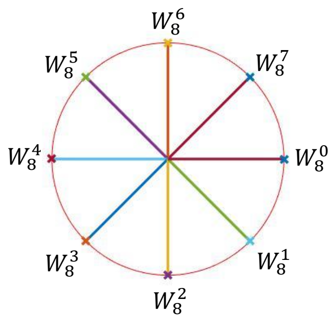
- The key to reduce the complexity of the DFT is to exploit properties of $W_N = e^{-j2\pi/N}$

$$W_N^{k+N} = ?$$
$$W_N^{k+N/2} = ?$$
$$W_N^{2k} = ?$$

Properties of $e^{j2\pi nk/N}$…

- Example 2: $W_8 = e^{-j2\pi/8}$

$$W_N^{k+N} = ?$$
$$W_N^{k+N/2} = ?$$
$$W_N^{2k} = ?$$

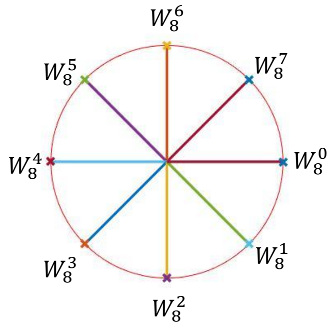


13

Properties of $e^{j2\pi nk/N}$…

- The key to reduce the complexity of the DFT is to exploit properties of $W_N = e^{-j2\pi/N}$

$$W_N^{k+N} = e^{-\frac{j2\pi(k+N)}{N}} = W_N^k$$
$$W_N^{k+N/2} = e^{-\frac{j2\pi\left(k+\frac{N}{2}\right)}{N}} = -W_N^k$$
$$W_N^{2k} = e^{-\frac{j2\pi k}{N/2}} = W_{N/2}^k$$



14

Radix-2 FFT

- Assume $N = 2^{\nu}$
- Split sequence $x[n]$ into two subsequences

$$f_1[n] = x[2n] \text{ and } f_2[n] = x[2n+1]$$

- We can write the DFT in terms of even and odd values of n

$$X(k) = \sum_{n=0}^{N-1} x[n] W_N^{nk}, k = 0, \dots N-1$$

$$= \sum_{n \text{ even}} x[n] W_N^{nk} + \sum_{n \text{ odd}} x[n] W_N^{nk}$$

$$= \sum_{m=0}^{N/2-1} x[2m] W_N^{2mk} + \sum_{m=0}^{N/2-1} x[2m+1] W_N^{(2m+1)k}$$

15

Radix-2 FFT…

- Rewrite in terms of decimated sequences…

$$X(k) = \sum_{m=0}^{N/2-1} f_1[m] W_{N/2}^{mk} + W_N^k \sum_{m=0}^{N/2-1} f_2[m] W_{N/2}^{mk}$$

- Have we divided the problem into subproblems?

16

Radix-2 FFT…

- Problem reduced to the sum of two DFTs of size $N/2$

$$X(k) = \sum_{m=0}^{N/2-1} f_1[m] W_{N/2}^{mk} + W_N^k \sum_{m=0}^{N/2-1} f_2[m] W_{N/2}^{mk}$$

$$= F_1(k) + W_N^k F_2(k),\ \ k = 0, \dots N-1$$

- Since $F_1(k)$ and $F_2(k)$ are $N/2$-point DFTs:

$$F_1\left(k + \frac{N}{2}\right) = F_1(k)$$

$$F_2\left(k + \frac{N}{2}\right) = F_2(k)$$

17

Radix-2 FFT…

- Exploiting periodicity of $F_1(k)$ and $F_2(k)$

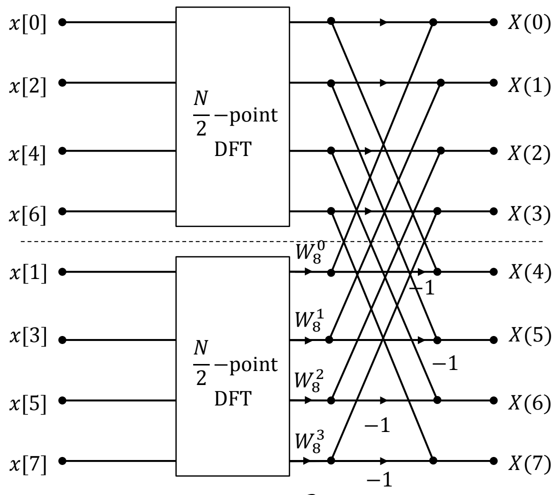
$$X(k) = F_1(k) + W_N^k F_2(k),\ \ k = 0, \dots N/2-1$$

$$X(k + N/2) = F_1(k) - W_N^k F_2(k),\ \ k = 0, \dots N/2-1$$

- Have we gained anything?

18

Radix-2 FFT…



- Number of multiplications: $2\left(\frac{N}{2}\right)^2 + \frac{N}{2}$

19

Radix-2 FFT…

- By splitting original problem into two we reduced the number of multiplications by a factor of two
- Why stop there?
- Repeat decimation for sequence $f_1[n]$

$$F_1(k) = V_{11}(k) + W_{\frac{N}{2}}^{k} V_{12}(k),\ k = 0, \dots N/4 - 1$$

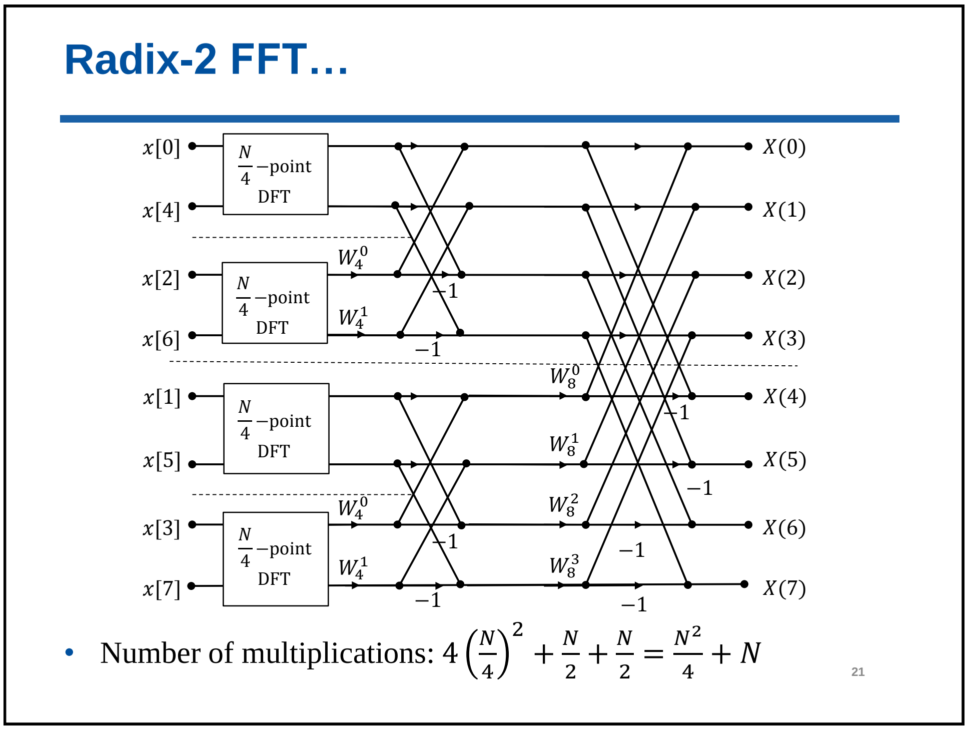
$$F_1(k + N/4) = V_{11}(k) - W_{\frac{N}{2}}^{k} V_{12}(k),\ k = 0, \dots N/4 - 1$$

- … and for sequence $f_2[n]$

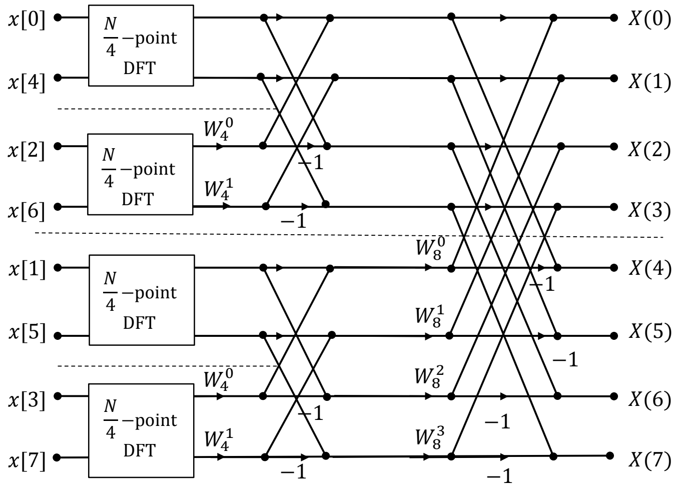
$$F_2(k) = V_{21}(k) + W_{\frac{N}{2}}^{k} V_{22}(k), \qquad k = 0, \dots N/4 - 1$$

$$F_2(k + N/4) = V_{21}(k) - W_{\frac{N}{2}}^{k} V_{22}(k),\ k = 0, \dots N/4 - 1$$

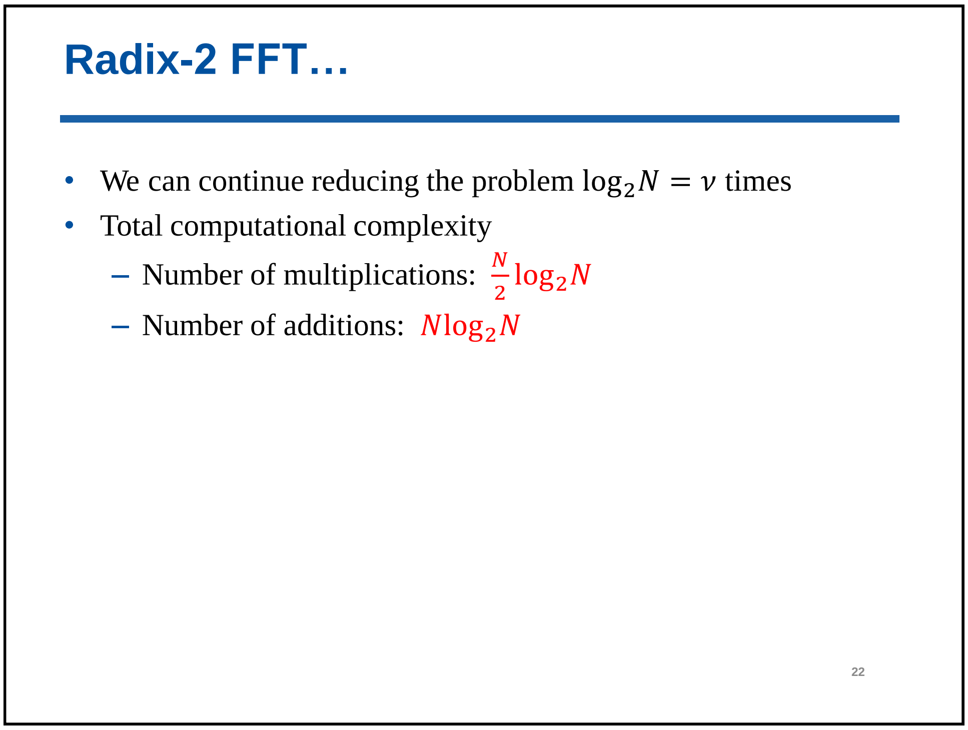
20



Radix-2 FFT…



- Number of multiplications: $4\left(\frac{N}{4}\right)^2 + \frac{N}{2} + \frac{N}{2} = \frac{N^2}{4} + N$



Radix-2 FFT…

- We can continue reducing the problem $\log_2 N = \nu$ times
- Total computational complexity
  - Number of multiplications: $\frac{N}{2}\log_2 N$
  - Number of additions: $N\log_2 N$

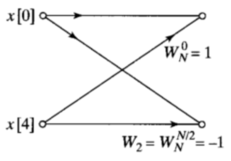
Radix-2 FFT…

- First stage is to evaluate a 2-point DFT?

$$X(k) = \sum_{n=0}^{1} x[n] e^{-\frac{j2\pi nk}{N=2}},\ k = 0, 1$$

$$\Rightarrow X(0) = x[0] + x[1]$$

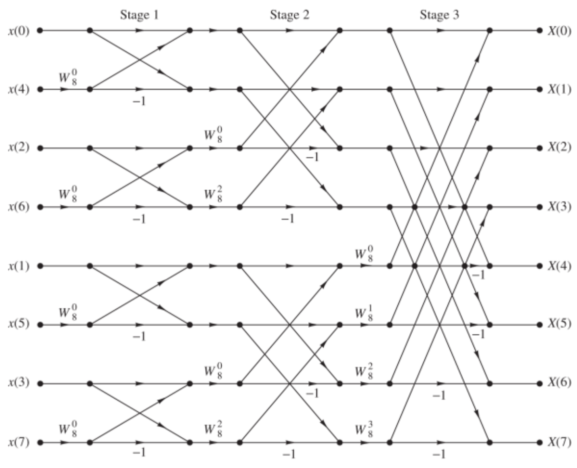
$$X(1) = x[0] - x[1]$$



23

Radix-2 FFT…

- Final flow-graph for 8-point decimation in time FFT



24

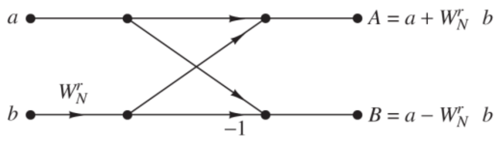
Radix-2 FFT…

- Comparing direct-computation DFT and FFT

Number of points, N	Complex multiplications in Direct computation, N^2	Complex multiplications in FFT, $\frac{N}{2}\log_2 N$
4	16	4
8	64	12
16	256	32
32	1024	80
64	4096	192
128	16384	448
256	65536	1024
512	262144	2304
1024	1048576	5120

Radix-2 FFT…

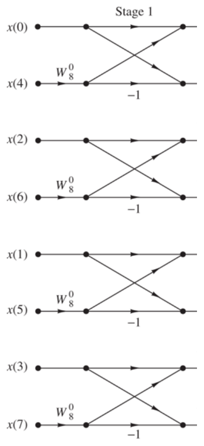
- Memory requirements?
- Basic computation performed in each stage (butterfly)



- Once butterfly operation performed, $(a, b) \longrightarrow (A, B)$, complex numbers A, B can be stored in same location as a, b
  - Computation *done in place*
  - Fixed amount of memory ($2N$ real numbers)

Radix-2 FFT…

- How to remember the order of the input to the FFT?
- Bit reversal - binary representation in reverse



27

Radix-2 IFFT

- How about the IFFT?
- Inverse DFT similar to DFT
  - Change terms W_N^k in the signal graph to W_N^{-k}
  - Divide the output of the graph by N.

28

Frequency-domain filtering

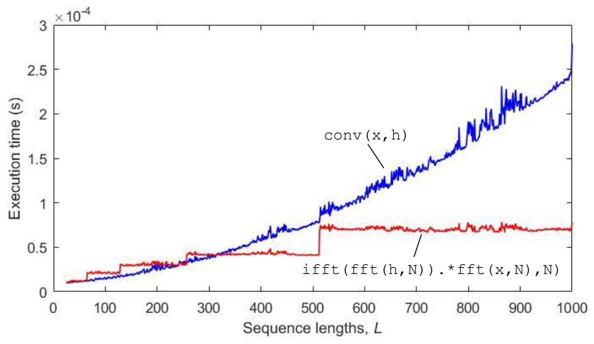
$$\begin{tikzcd}
x[n] \arrow[r] \arrow[d, leftrightarrow] & \boxed{h[n]} \arrow[r] & y[n] = \mathrm{IFFT}_N\{Y(k)\} \arrow[d, leftrightarrow] \\
X(k) = \mathrm{FFT}_N\{x[n]\} & h[n] \arrow[d, leftrightarrow] & Y(k) = X(k)H(k) \\
 & H(k) = \mathrm{FFT}_N\{h[n]\} &
\end{tikzcd}$$

- Order of complexity in terms of multiplications
  1. Transform to frequency domain FFT: $2N\log_2 N$
  2. Multiply frequency transforms: N
  3. Transform back to time domain IFFT: $N\log_2 N$

29

Frequency-domain filtering… (last week)

- Example 2: Assume that both $x[n]$ and $h[n]$ have length L



30

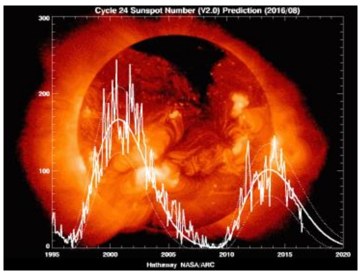
Summary

- Today we discussed:
  - Fast Fourier transform (FFT)
- Next time:
  - Stochastic processes

Example: Solar cycle

- Source: https://en.wikipedia.org/wiki/Solar_cycle

"The solar cycle or solar magnetic activity cycle is the nearly periodic 11-year change in the Sun's activity (including changes in the levels of solar radiation and ejection of solar material) and appearance (changes in the number of sunspots, flares, and other manifestations)"



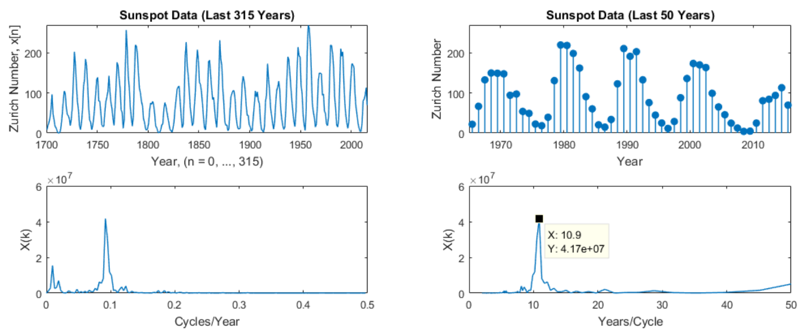
http://solarscience.msfc.nasa.gov/images/ssn_predict_l.gif

Example: Solar cycle...

- According to Wikipedia the solar cycle is “nearly periodic” with a period of approximately 11 years
- Can this be right?
- Matlab has data with 300 years of recorded sunspot numbers
- Sunspot data from 1700 until now can be downloaded at `http://www.sidc.be/silso/datafiles` and is available on BlackBoard

Example: Solar cycle...

`http://www.sidc.be/silso/datafiles`



- From figure we get a solar cycle (time between activity peaks) is approximately 11 years

Example: Solar cycle...

Matlab (Available on Blackboard)

```
load('sunspots.csv')
year = sunspots(:,1);
x = sunspots(:,2); % Relative spot number
X = fft(x);        % DFT of x
X(1) = 0; [];      % Remove DC level(sum of all numbers)
N = length(X);

% Plot some figs
figure
subplot(2,2,1);
plot(year,x) % last 316 years sunspot activity

subplot(2,2,2);
stem(year(end-50:end),x(end-50:end)) % last 51 years

freq = -0.5:1/N:(0.5-1/N);
subplot(2,2,3),
plot(freq,abs(fftshift(X)).^2) % frequency domain

period = 1./freq;
subplot(2,2,4),
plot(period,abs(fftshift(X)).^2); % show cycles/year
```

35