

# Algorithms of Scientific Computing

## Discrete Fourier Transform (DFT)

Michael Bader

Summer Term 2015



# Fast Fourier Transform – Outline

- Discrete Fourier transform
- Fast Fourier transform
- Special Fourier transform:
  - real-valued FFT
  - sine/cosine transform
- Applications:
  - Fast Poisson solver (FST)
  - Computergraphics (FCT)
- Efficient Implementation

# Discrete Fourier Transform (DFT)

## Definition:

For a vector of  $N$  complex numbers  $(f_0, \dots, f_{N-1})^T$ , the **discrete Fourier transform** (DFT) is given by the vector  $(F_0, \dots, F_{N-1})^T$ , where

$$F_k = \frac{1}{N} \sum_{n=0}^{N-1} f_n e^{-i2\pi nk/N}.$$

## Interpretation:

The DFT can be derived as

- trigonometric interpolation/approximation
- approximation of the coefficients of the Fourier series

# DFT as Interpolation (1)

## Interpolation problem:

- $N$  ansatz functions:  $g_k(x) := e^{ikx}$  in the interval  $[0, 2\pi]$ ,  $k = 0, \dots, N-1$
- $N$  supporting points:  $x_n := 2\pi n/N$ ,  $n = 0, \dots, N-1$
- $N$  interpolation value  $f_n$ ,  $n = 0, \dots, N-1$
- find  $N$  weights  $F_k$  such that at all supporting points

$$f_n = \sum_{k=0}^{N-1} F_k g_k(x_n) \quad \Leftrightarrow \quad f_n = \sum_{k=0}^{N-1} F_k e^{i2\pi nk/N}.$$

“trigonometric interpolation”

# DFT as Interpolation (2)

## Interpolation problem:

- $N$  ansatz functions:  $\tilde{g}_k(z) := z^k$  (complex unit polynomials),  $k = 0, \dots, N - 1$
- $N$  supporting points:  $z_n := e^{j2\pi n/N} = \omega_N^n$ , where  $\omega_N := e^{j2\pi/N}$
- $N$  interpolation values  $f_n$ ,  $n = 0, \dots, N - 1$ , respectively.
- find the  $N$  weights  $F_k$  such that at all supporting points

$$f_n = \sum_{k=0}^{N-1} F_k \tilde{g}_k(z_n) \quad \Leftrightarrow \quad f_n = \sum_{k=0}^{N-1} F_k e^{j2\pi nk/N}.$$

Polynomial interpolation at the “**complex unit roots**”  $\omega_N^n$

# Interpretation of the Interpolation Problem

Starting from the first formulation,

$$f_n = \sum_{k=0}^{N-1} F_k g_k(x_n), \quad g_k(x_n) = e^{i2\pi nk/N},$$

we look for a representation of the signal  $f_n$  – or of a function  $f(x)$  – of the form

$$f(x) = \sum_{k=0}^{N-1} F_k g_k(x), \quad g_k(x) = e^{i2\pi kx}.$$

The ansatz functions are sine or cosine oscillations:

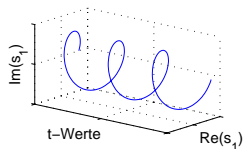
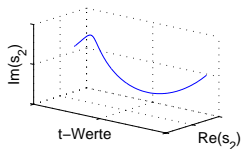
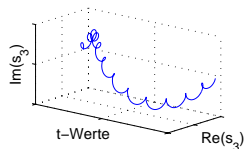
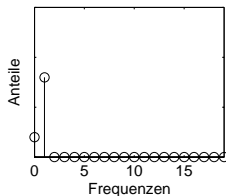
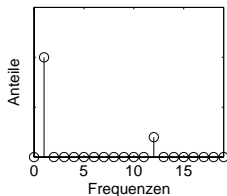
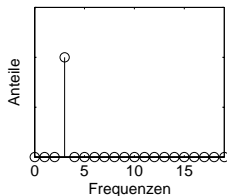
$$e^{ikx} = \cos(kx) + i \sin(kx)$$

## Interpretation of the Interpolation Problem (2)

Conclusions:

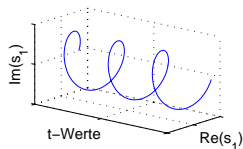
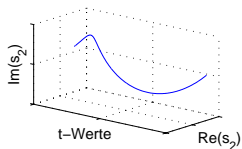
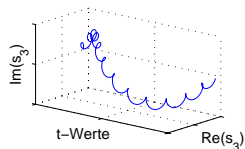
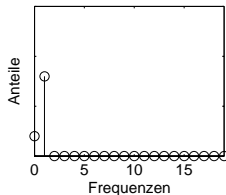
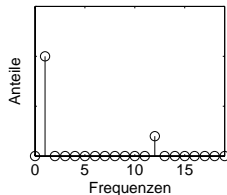
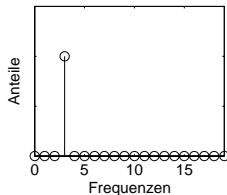
- we look for the **representation of a periodic function** as a sum of sines and cosines
- the  $F_k$  are, thus, called **Fourier coefficients**:
  - $k$  represents the wave number
  - the value of  $F_k$  represents the amplitude of the corresponding frequency
- the Fourier transform leads to a **frequency spectrum**
- useful when a problem is easier to solve in the **frequency domain** than in the **spatial domain**.

# Example: Spatial vs. Frequency Domain

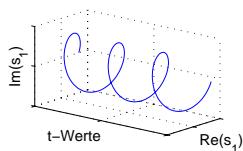
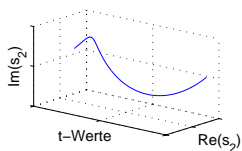
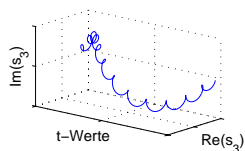
3D-Kurve des Signals  $s_1$ 3D-Kurve des Signals  $s_2$ 3D-Kurve des Signals  $s_3$ Frequenzspektrum  $f_1$ Frequenzspektrum  $f_2$ Frequenzspektrum  $f_3$ 



# Example: Spatial vs. Frequency Domain

3D-Kurve des Signals  $s_1$ 3D-Kurve des Signals  $s_2$ 3D-Kurve des Signals  $s_3$ Frequenzspektrum  $f_1$ Frequenzspektrum  $f_2$ Frequenzspektrum  $f_3$ 

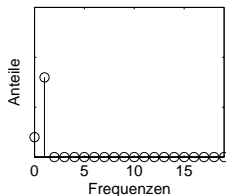
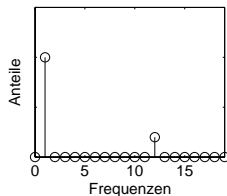
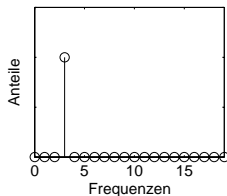
# Example: Spatial vs. Frequency Domain

3D-Kurve des Signals  $s_1$ 3D-Kurve des Signals  $s_2$ 3D-Kurve des Signals  $s_3$ 

$$s_1 = e^{3it},$$

$$s_2 = 0.2i + 0.8e^{it},$$

$$s_3 = e^{it} + 0.2e^{12it}$$

Frequenzspektrum  $f_1$ Frequenzspektrum  $f_2$ Frequenzspektrum  $f_3$ 

# Solution of the Interpolation Problem

Both interpolation problems lead to the identical linear systems of equations:

$$f_n = \sum_{k=0}^{N-1} F_k \omega_N^{nk}, \quad \text{for all } n = 0, \dots, N-1;$$

where  $\omega_N := e^{i2\pi/N}$ , i.e.  $\omega_N^{nk} := e^{i2\pi nk/N}$ .

If we write the vectors of the  $f_n$  and  $F_k$  as  $\mathbf{f} := (f_0, \dots, f_{N-1})$  and  $\mathbf{F} := (F_0, \dots, F_{N-1})$ , the linear system of equations can be formulated in matrix-vector notation

$$\mathbf{W}\mathbf{F} = \mathbf{f},$$

where the entries of the **Fourier matrix**  $\mathbf{W}$  are given by  $W_{nk} := \omega_N^{nk}$ .

# Properties of the Fourier Matrix $W$

- $W$  is symmetric:  $W = W^T$ , and has the form

$$W = \begin{pmatrix} \omega_N^0 & \omega_N^0 & \omega_N^0 & \dots & \omega_N^0 \\ \omega_N^0 & \omega_N^1 & \omega_N^2 & \dots & \omega_N^{(N-1)} \\ \omega_N^0 & \omega_N^2 & \omega_N^4 & \dots & \omega_N^{2(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ \omega_N^0 & \omega_N^{(N-1)} & \omega_N^{2(N-1)} & \dots & \omega_N^{(N-1)(N-1)} \end{pmatrix}$$

- $W (W^T)^* = WW^H = NI$ , since

$$[WW^H]_{kl} = \sum_{j=0}^{N-1} \omega_N^{kj} (\omega_N^{lj})^* = \sum_{j=0}^{N-1} \omega_N^{(k-l)j} = \begin{cases} N & \text{if } k = l \\ 0 & \text{if } k \neq l. \end{cases}$$

# Computation of the Fourier Coefficients $F_k$

- Since  $\mathbf{W}\mathbf{W}^H = N\mathbf{I}$ , the inverse of  $\mathbf{W}$  is  $\mathbf{W}^{-1} = \frac{1}{N}\mathbf{W}^H$ :

$$\mathbf{W}^{-1} = \frac{1}{N} \begin{pmatrix} \omega_N^0 & \omega_N^0 & \omega_N^0 & \dots & \omega_N^0 \\ \omega_N^0 & \omega_N^{-1} & \omega_N^{-2} & \dots & \omega_N^{-(N-1)} \\ \omega_N^0 & \omega_N^{-2} & \omega_N^{-4} & \dots & \omega_N^{-2(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ \omega_N^0 & \omega_N^{-(N-1)} & \omega_N^{-2(N-1)} & \dots & \omega_N^{-(N-1)(N-1)} \end{pmatrix}$$

- ⇒ the vector  $\mathbf{F}$  of the Fourier coefficients can be computed **easily** as a matrix-vector product – with computational effort  $\mathcal{O}(N^2)$ :

$$\mathbf{F} = \frac{1}{N}\mathbf{W}^H\mathbf{f} \quad \text{or} \quad F_k = \frac{1}{N} \sum_{n=0}^{N-1} f_n \omega_N^{-nk}.$$

# Inverse Discrete Fourier Transform (IDFT)

## Definition:

The inverse Discrete Fourier Transform (IDFT) of the vector  $(F_0, \dots, F_{N-1})$  is given by the vector  $(f_0, \dots, f_{N-1})$ , where

$$f_n = \sum_{k=0}^{N-1} F_k e^{i2\pi nk/N}.$$

## Observation:

DFT and IDFT are inverse operations:

$$F_k = \frac{1}{N} \sum_{n=0}^{N-1} f_n e^{-i2\pi nk/N}, \quad f_n = \sum_{k=0}^{N-1} F_k e^{i2\pi nk/N}.$$

$$\mathbf{F} = \text{DFT}(\text{IDFT}(\mathbf{F})) \quad \text{or} \quad \mathbf{f} = \text{IDFT}(\text{DFT}(\mathbf{f})).$$

# The Pair DFT/IDFT as Matrix-Vector Product

With the notation  $\omega_N := e^{i2\pi/N}$ , i.e.  $\omega_N^{-nk} := e^{-i2\pi nk/N}$ , we formulate the DFT/IDFT as

$$F_k = \frac{1}{N} \sum_{n=0}^{N-1} f_n \omega_N^{-nk} \quad f_n = \sum_{k=0}^{N-1} F_k \omega_N^{nk}$$

With the vectors  $\mathbf{f} := (f_0, \dots, f_{N-1})^T$  and  $\mathbf{F} := (F_0, \dots, F_{N-1})^T$ , we denote (and compute) the DFT und IDFT as matrix-vector products

$$\mathbf{F} = \frac{1}{N} \mathbf{W}^H \mathbf{f}, \quad \mathbf{f} = \mathbf{W} \mathbf{F},$$

where the elements of the Fourier matrix  $\mathbf{W}$  are  $W_{nk} := \omega_N^{nk}$ .

# Properties of the DFT

- DFT and IDFT are (as a matrix-vector product) **linear**:

$$\begin{aligned}\text{DFT}(\alpha f + \beta g) &= \alpha \text{DFT}(f) + \beta \text{DFT}(g) \\ \text{IDFT}(\alpha f + \beta g) &= \alpha \text{IDFT}(f) + \beta \text{IDFT}(g)\end{aligned}$$

- since  $\omega_N^{nk} = \omega_N^{n(k+N)} = \omega_N^{(n+N)k}$ , the  $f_n$  and the  $F_k$  are **periodic**:

$$f_{n+N} = f_n \quad F_{k+N} = F_k \quad \text{for all } k, n \in \mathbb{Z}$$



# Alternative Forms of the DFT

**Possible variants** (in all imaginable combinations):

- Scaling factor  $\frac{1}{N}$  in the IDFT instead of the DFT; alternatively a factor  $\frac{1}{\sqrt{N}}$  in DFT and IDFT.
- switched signs in the exponent of the exponential function in DFT and IDFT
- use  $j$  for the imaginary unit (electrical engineering)

**Shift of indices:**

- periodic data:  $F_k = F_{k+N}$
- aliasing of frequencies:  $e^{-i2\pi nk/N} = e^{-i2\pi n(k\pm N)/N}$

# DFT with Shifted Indices

Data and frequencies “symmetric”:

$$F_k = \frac{1}{N} \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_n e^{-i2\pi nk/N}, \quad f_n = \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} F_k e^{i2\pi nk/N}$$

In general:

$$F_k = \frac{1}{N} \sum_{n=P+1}^{P+N} f_n e^{-i2\pi nk/N}, \quad f_n = \sum_{k=Q+1}^{Q+N} F_k e^{i2\pi nk/N}$$

# DFT in Program Libraries

Matlab, IMSL (Int. Math. and Stat. Library):

$$F_{k+1} = \sum_{n=0}^{N-1} f_{n+1} e^{-i2\pi nk/N} \quad k = 0, \dots, N-1$$

$$f_{n+1} = \frac{1}{N} \sum_{k=0}^{N-1} F_{k+1} e^{i2\pi nk/N} \quad n = 0, \dots, N-1$$

Maple:  $\frac{1}{\sqrt{N}}$  as factor for DFT and IDFT.

**Index shift by +1, since:**

- Data/coefficients start at index 0
- Arrays to store the numbers start at index 1