Algorithms of Scientific Computing

Discrete Fourier Transform (DFT)

Michael Bader
Technical University of Munich

Summer 2016
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 $(f_0, \ldots, f_{N-1})^T$, the **discrete Fourier transform** (DFT) is given by the vector $(F_0, \ldots, F_{N-1})^T$, 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

**Applications:**
- JPEG, MPEG, MP3, etc.
- signal processing in general, solvers in science and engineering
DFT as Interpolation (1)

Interpolation problem:

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

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

“trigonometric interpolation”
DFT as Interpolation (2)

Interpolation problem:

- $N$ ansatz functions: $\tilde{g}_k(z) := z^k$ (complex unit polynomials),
  $k = 0, \ldots, N - 1$
- $N$ supporting points: $z_n := e^{i2\pi n/N} = \omega^N_n$, where $\omega_N := e^{i2\pi/N}$
- $N$ interpolation values $f_n$, $n = 0, \ldots, N - 1$, respectively.
- find the $N$ weights $F_k$ such that at all supporting points

$$f_n = \sum_{k=0}^{N-1} F_k \tilde{g}_k(z_n) \iff f_n = \sum_{k=0}^{N-1} F_k e^{i2\pi nk/N}.$$  

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

\[ s_1 = e^{3it}, \quad s_2 = 0.2i + 0.8e^{it}, \quad s_3 = e^{it} + 0.2e^{12it} \]
Solution of the Interpolation Problem

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

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

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 \( \mathbf{f} := (f_0, \ldots, f_{N-1}) \) and \( \mathbf{F} := (F_0, \ldots, F_{N-1}) \), 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

$$W = \begin{pmatrix}
\omega_0^N & \omega_0^N & \omega_0^N & \cdots & \omega_0^N \\
\omega_0^N & \omega_1^N & \omega_2^N & \cdots & \omega_{(N-1)}^N \\
\omega_0^N & \omega_2^N & \omega_4^N & \cdots & \omega_{2(N-1)}^N \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\omega_0^N & \omega_{(N-1)}^N & \omega_{2(N-1)}^N & \cdots & \omega_{(N-1)(N-1)}^N
\end{pmatrix}$$

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

$$[WW^H]_{kl} = \sum_{j=0}^{N-1} \omega_N^{kj} \left(\omega_N^{lj}\right)^* = \sum_{j=0}^{N-1} \omega_N^{(k-l)j} = \begin{cases} N & \text{if } k = l \\ 0 & \text{if } k \neq l. \end{cases}$$
Properties of the Fourier Matrix $W$ (details)

- Important property #1: $(\omega^k_N)^N = e^{N \cdot i 2\pi k/N} = e^{i 2\pi k} = 1$

- Important property #2: $(\omega^k_N)^* = (e^{i 2\pi k/N})^* = (\cos(2\pi k/N) + i \sin(2\pi k/N))^* = \cos(2\pi k/N) - i \sin(2\pi k/N) = \cos(-2\pi k/N) + i \sin(-2\pi k/N) = e^{-i 2\pi k/N} = \omega^{-k}_N$

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

\[
\left[W W^H\right]_{kl} = \sum_{j=0}^{N-1} \omega^{kj}_N \left(\omega^l_j\right)^* = \sum_{j=0}^{N-1} \omega^{kj}_N \omega^{-lj}_N = \sum_{j=0}^{N-1} \omega^{(k-l)j}_N
\]

- If $k = l$ then

\[
\sum_{j=0}^{N-1} \omega^{(k-l)j}_N = \sum_{j=0}^{N-1} \omega^0_N = N
\]

- If $k \neq l$ then

\[
\sum_{j=0}^{N-1} \omega^{(k-l)j}_N = \sum_{j=0}^{N-1} \xi^j \text{ where } \xi = \omega^{(k-l)}_N \text{ and then:}
\]

\[
(1 - \xi) \sum_{j=0}^{N-1} \xi^j = \sum_{j=0}^{N-1} \xi^j - \sum_{j=0}^{N-1} \xi^{(j+1)} = 1 - \xi^N, \text{ thus } \sum_{j=0}^{N-1} \xi^j = \frac{1 - \xi^N}{1 - \xi}
\]

Remember that $(\omega^k_N)^N = 1$ for any $k$ (“unit roots”!), thus

\[
\sum_{j=0}^{N-1} \omega^{(k-l)j}_N = 0
\]
Computation of the Fourier Coefficients $F_k$

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

$$W^{-1} = \frac{1}{N} \begin{pmatrix} \omega_0^0 & \omega_0^0 & \omega_0^0 & \ldots & \omega_0^0 \\ \omega_0^0 & \omega_0^{-1} & \omega_0^{-2} & \ldots & \omega_0^{-(N-1)} \\ \omega_0^0 & \omega_0^{-2} & \omega_0^{-4} & \ldots & \omega_0^{-2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega_0^0 & \omega_0^{-(N-1)} & \omega_0^{-(2(N-1))} & \ldots & \omega_0^{-(N-1)(N-1)} \end{pmatrix}$$

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

$$F = \frac{1}{N}W^Hf \quad \text{or} \quad F_k = \frac{1}{N} \sum_{n=0}^{N-1} f_n\omega_N^{-nk}.$$
Inverse Discrete Fourier Transform (IDFT)

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

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

**Observation:**
DFT and IDFT are inverse operations:

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

\[
\text{F} = \text{DFT(IDFT(F))} \quad \text{or} \quad \text{f} = \text{IDFT(DFT(f))}.
\]
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 \( f := (f_0, \ldots, f_{N-1})^T \) and \( F := (F_0, \ldots, F_{N-1})^T \), we denote (and compute) the DFT and IDFT as matrix-vector products

\[
F = \frac{1}{N} W^H f \quad f = W F,
\]

where the elements of the Fourier matrix \( W \) are \( W_{nk} := \omega_N^{nk} \).
Properties of the DFT

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

\[
\begin{align*}
\text{DFT}(\alpha f + \beta g) &= \alpha \text{DFT}(f) + \beta \text{DFT}(g) \\
\text{IDFT}(\alpha f + \beta g) &= \alpha \text{IDFT}(f) + \beta \text{IDFT}(g)
\end{align*}
\]

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

\[
\begin{align*}
f_{n+N} &= f_n \\
F_{k+N} &= F_k
\end{align*}
\]
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: \( e^{-i2\pi n k/N} = e^{-i2\pi n(k\pm N)/N} \)
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


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

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

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