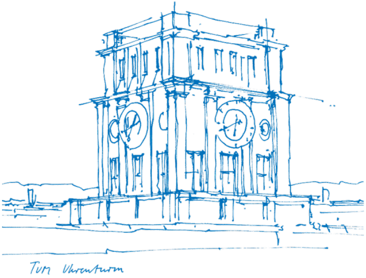
Algorithms of Scientific Computing

FFT on Real Data

Michael Bader
Technical University of Munich

Summer 2022



DFT and Symmetry – Agenda

	INPUT		TRANSFORM
real symmetry	$f_n \in \mathbb{R}$	$\rightarrow$	Real DFT (RDFT)
even symmetry	$f_n = f_{-n}$	$\rightarrow$	Discrete Cosine Transform (DCT)
odd symmetry	$f_n = -f_{-n}$	$\rightarrow$	Discrete Sine Transform (DST)

"QUARTER-WAVE"	INPUT		TRANSFORM
even symmetry	$f_n = f_{-n-1}$	$\rightarrow$	QW-DCT
odd symmetry	$f_n = -f_{-n-1}$	$\rightarrow$	QW-DST

Real-valued DFT (RDFT)

Consider real-valued input data $f_n \in \mathbb{R}$, i.e.: $f_n^* := \overline{f_n} = f_n$, then:

$$F_k = \frac{1}{N} \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_n e^{-i2\pi nk/N} = \frac{1}{N} \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_n \left(\cos\left(\frac{2\pi nk}{N}\right) - i \sin\left(\frac{2\pi nk}{N}\right)\right).$$

Properties:

- $\mathrm{Re}\{F_k\} = \frac{1}{N} \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_n \cos\left(\frac{2\pi nk}{N}\right), \quad \mathrm{Im}\{F_k\} = -\frac{1}{N} \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_n \sin\left(\frac{2\pi nk}{N}\right)$
- only N independent, real coefficients necessary, since:

$$F_k^* = \frac{1}{N} \sum f_n^* \left\{\omega_N^{-nk}\right\}^* = \frac{1}{N} \sum f_n \omega_N^{-n(-k)} = F_{-k}$$

Recall: $\left\{\omega_N^{-nk}\right\}^* \quad = \left\{e^{-i2\pi_n k/N}\right\}^* = \cos(-2\pi nk/N) - i \sin(-2\pi nk/N)$

$\qquad = \cos(2\pi nk/N) + i \sin(2\pi nk/N) = \cdots = \omega_N^{nk}$

Real DFT (2)

Map N values to N coefficients (and vice versa):

$$\left\{ f_{-\frac{N}{2}+1}, \ldots, \qquad f_0, \qquad \ldots, \quad f_{\frac{N}{2}} \right\}$$

$$\text{DFT} \quad \Downarrow \qquad \Uparrow \quad \text{IDFT}$$

$$\left\{ F_0, \mathrm{Re}\{F_1\}, \mathrm{Im}\{F_1\}, \qquad \ldots, \qquad \mathrm{Re}\{F_{\frac{N}{2}-1}\}, \mathrm{Im}\{F_{\frac{N}{2}-1}\}, F_{\frac{N}{2}} \right\}$$

Note: real and imaginary parts of F_{-k} correspond to those of F_k

Real DFT (3)

Situation:

- only N real input values (as all N imaginary parts are 0)
- **only N independent, real output values** (coefficient components) due to symmetry $F_{-k} = F_k^*$

Wanted $\rightarrow$ new transformation:

N real input values $\rightarrow$ N distinct real coefficient components

Hence: Insert symmetry $F_{-k} = F_k^*$ in IDFT!

Real DFT (4)

Definition of **"Real discrete Fourier transform"** (RDFT)

- formulation 1

$$F_k = \frac{1}{N} \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_n \left(\cos\left(\frac{2\pi nk}{N}\right) - i \sin\left(\frac{2\pi nk}{N}\right)\right) \, , k = 0, \ldots, \frac{N}{2}$$

- formulation 2

$$\begin{aligned} \operatorname{Re}\{F_k\} &= \frac{1}{N} \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_n \cos\left(\frac{2\pi nk}{N}\right), k = 0, \ldots, \frac{N}{2} \\ \operatorname{Im}\{F_k\} &= -\frac{1}{N} \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_n \sin\left(\frac{2\pi nk}{N}\right), k = 1, \ldots, \frac{N}{2} - 1 \end{aligned}$$

Inverse Real DFT

Goal: compute a real representation of the DFT (i.e., no $e^{-i2\pi nk/N}$).

We start with an input vector (Fourier coefficients) of **length $2N$**, then:

$$
\begin{aligned}
f_n &= \sum_{k=-N+1}^{N} F_k e^{i2\pi nk/2N} \\
&= F_0 + \sum_{k=1}^{N-1} \left(F_k e^{i2\pi nk/2N} + F_{-k} e^{-i2\pi nk/2N} \right) + F_N e^{i2\pi nN/2N} \\
&= F_0 + \sum_{k=1}^{N-1} \left(F_k e^{i2\pi nk/2N} + \left\{ F_k e^{i2\pi nk/2N} \right\}^* \right) + F_N e^{i\pi n} \\
&= F_0 + 2 \sum_{k=1}^{N-1} \operatorname{Re} \left\{ F_k e^{i2\pi nk/2N} \right\} + F_N e^{i\pi n} \\
&= F_0 + 2 \sum_{k=1}^{N-1} \left(\operatorname{Re}\{F_k\} \cos\left(\frac{\pi nk}{N} \right) - \operatorname{Im}\{F_k\} \sin\left(\frac{\pi nk}{N} \right) \right) + F_N \cos(\pi n)
\end{aligned}
$$

Inverse Real DFT (2)

Set $a_k := 2\,\mathrm{Re}\{F_k\}$ and $b_k := -2\,\mathrm{Im}\{F_k\}$ (but $a_0 := \mathrm{Re}\{F_0\}$ and $a_N := \mathrm{Re}\{F_N\}$) to get :

$$f_n = a_0 + \sum_{k=1}^{N-1} \left(a_k \cos\left(\frac{\pi nk}{N}\right) + b_k \sin\left(\frac{\pi nk}{N}\right)\right) + a_N \cos(\pi n)$$

“Real inverse discrete Fourier transform”

Using the (real-valued) formula for F_k:

$$a_k = \frac{1}{N} \sum_{n=-N+1}^{N} f_n \cos\left(\frac{\pi nk}{N}\right), \qquad b_k = \frac{1}{N} \sum_{n=-N+1}^{N} f_n \sin\left(\frac{\pi nk}{N}\right)$$

but: $a_0 = \frac{1}{2N} \sum \ldots$ and $a_N = \frac{1}{2N} \sum \ldots$

Inverse Real DFT – Compare with Textbooks

Alternate (probably more frequent) notation in textbooks:
Set $a_k := 2\,\mathrm{Re}\{F_k\}$ and $b_k := -2\,\mathrm{Im}\{F_k\}$ to get

$$f_n = \frac{1}{2}a_0 + \sum_{k=1}^{N-1}\left(a_k \cos\left(\frac{\pi nk}{N}\right) + b_k \sin\left(\frac{\pi nk}{N}\right)\right) + \frac{1}{2}a_N \cos(\pi n)$$

Using the (real-valued) formula for F_k:

$$a_k = \frac{1}{N}\sum_{n=-N+1}^{N} f_n \cos\left(\frac{\pi nk}{N}\right), \qquad b_k = \frac{1}{N}\sum_{n=-N+1}^{N} f_n \sin\left(\frac{\pi nk}{N}\right)$$

Differences:

- no extra definition of a_0 and a_N
- but factors $\frac{1}{2}$ in formula for f_n $\rightarrow$ not as nicely related to interpolation

Real-valued Trigonometric Interpolation

Interpretation of the real DFT as an interpolation problem:

- $2N$ ansatz functions:

$$\begin{aligned} g_k(x) &:= \cos(kx) \qquad k = 0, \ldots, N \\ h_k(x) &:= \sin(kx) \qquad k = 1, \ldots, N-1 \end{aligned}$$

- $2N$ supporting points: $x_n := \frac{2\pi n}{2N} = \frac{\pi n}{N} \qquad n = -N+1, \ldots, N$
- $2N$ interpolation conditions:

$$f_n = a_0 + \sum_{k=1}^{N-1} \left(a_k \cos\left(\frac{\pi n k}{N}\right) + b_k \sin\left(\frac{\pi n k}{N}\right)\right) + a_N \cos(\pi n)$$

(cmp. exercises)

Fast Real DFT

Computation of a real-valued DFT using complex FFT is inefficient:

- N redundant components computed (symmetry)
- complex arithmetics with lots of real/imaginary parts being 0

Possibilities to improve the efficiency:

1. compute two real DFTs from one complex FFT
2. compute a real DFT of length $2N$ from one complex FFT of length N
3. "compact" real FFT – use symmetry of the data directly in the algorithm

Two Real DFTs from one complex FFT

Idea: for real-valued g_n and h_n, compute DFT of $f_n := g_n + ih_n$:

$$F_k = \frac{1}{N} \sum_n (g_n + ih_n)\omega_N^{-nk}$$

Comparison with coefficients G_k and H_k of the two real DFTs:

$$G_k = \frac{1}{N} \sum_n g_n \omega_N^{-nk} \qquad H_k = \frac{1}{N} \sum_n h_n \omega_N^{-nk}$$

Due to linearity of the Fourier transform:

$$F_k = G_k + iH_k$$

Two Real DFTs from one complex FFT (2)

Since g_n and h_n are real data, we have the following symmetry:

$$G_k = G^*_{-k} \qquad H_k = H^*_{-k} \ .$$

Hence, we get for F^*_{-k}:

$$F^*_{-k} = (G_{-k} + iH_{-k})^* = \left(G^*_{-k} + i^* H^*_{-k}\right) = G_k - iH_k \ .$$

Together with $F_k = G_k + iH_k$, we obtain

$$G_k = \frac{1}{2}\left(F_k + F^*_{-k}\right) \qquad \text{and} \qquad H_k = -\frac{i}{2}\left(F_k - F^*_{-k}\right) \ .$$

Two real DFTs from one complex FFT – Algorithm

Algorithm to compute two real DFTs:

(1) set $f_n := g_n + ih_n$

(2) compute F_k from FFT (using a library, e.g.)

(3) compute G_k and H_k according to

$$G_k = \frac{1}{2}\left(F_k + F^*_{-k}\right) \qquad \text{and} \qquad H_k = -\frac{i}{2}\left(F_k - F^*_{-k}\right)$$

$\Rightarrow$ "half" the costs compared to using complex FFT

but: additional operations for pre- and postprocessing

Real DFT of length $2N$ from complex FFT of length N

Compute DFT of a real-valued vector $(f_{-N+1}, \ldots, f_N)$:

$$F_k = \frac{1}{2N} \sum_{-N+1}^{N} f_n \omega_{2N}^{-nk} \qquad \text{for} \quad k = -\frac{N}{2} + 1, \ldots, \frac{N}{2}$$

Split up in $g_n := f_{2n}$ and $h_n := f_{2n-1}$; leads to butterfly scheme:

$$\begin{aligned} F_k &= \tfrac{1}{2}\left(G_k + \omega_{2N}^{k} H_k\right) , \\ F_{k \pm N} &= \tfrac{1}{2}\left(G_k - \omega_{2N}^{k} H_k\right) \end{aligned}$$

for $k = -\frac{N}{2} + 1, \ldots, \frac{N}{2}$, respectively.

Real $2N$-DFT from complex N-FFT

Now: compute G_k and H_k (two real DFTs) from one complex FFT
$\rightarrow$ applied to $z_n := g_n + ih_n = f_{2n} + if_{2n-1}$:

$$G_k = \frac{1}{2}\left(Z_k + Z^*_{-k}\right) \qquad \text{and} \qquad H_k = -\frac{i}{2}\left(Z_k - Z^*_{-k}\right)$$

Combine both schemes to:

$$\begin{aligned}
F_k &= \frac{1}{4}Z_k\left(1 - i\omega_{2N}^k\right) + \frac{1}{4}Z^*_{-k}\left(1 + i\omega_{2N}^k\right), \qquad k = 0, \ldots, \frac{N}{2} \\
F_{k+N} &= \frac{1}{4}Z_k\left(1 + i\omega_{2N}^k\right) + \frac{1}{4}Z^*_{-k}\left(1 - i\omega_{2N}^k\right), \qquad k = -\frac{N}{2} + 1, \ldots, 0
\end{aligned}$$

Real $2N$-DFT from complex N-FFT – Algorithm

Algorithm for a real $2N$-DFT:

(1) set $z_n := f_{2n} + i f_{2n-1}$

(2) compute Z_k from FFT applied on z_n (using a library, e.g.)

(3) compute F_k according to

$$
\begin{aligned}
F_k &= \tfrac{1}{4} Z_k \left(1 - i\omega_{2N}^k\right) + \tfrac{1}{4} Z_{-k}^* \left(1 + i\omega_{2N}^k\right), \qquad && k = 0, \ldots, \tfrac{N}{2} \\
F_{k+N} &= \tfrac{1}{4} Z_k \left(1 + i\omega_{2N}^k\right) + \tfrac{1}{4} Z_{-k}^* \left(1 - i\omega_{2N}^k\right), \qquad && k = -\tfrac{N}{2} + 1, \ldots, 0
\end{aligned}
$$

$\Rightarrow$ Complexity determined by complex N-FFT

plus: additional operations for pre- and postprocessing

Compact Real FFT

Compute DFT of a real-valued vector $(f_{-N+1}, \dots, f_N)$:

$$F_k = \frac{1}{2N} \sum_{-N+1}^{N} f_n \omega_{2N}^{-nk} \qquad \text{for} \quad k = 0, \dots, N$$

Split up in $g_n := f_{2n}$ and $h_n := f_{2n-1}$; leads to butterfly scheme:

$$\begin{aligned} F_k &= \tfrac{1}{2}\left(G_k + \omega_{2N}^{k} H_k\right) && \text{for } k = 0, \dots, \tfrac{N}{2}\ , \\ F_{k+N} &= \tfrac{1}{2}\left(G_k - \omega_{2N}^{k} H_k\right) && \text{for } k = -\tfrac{N}{2} + 1, \dots, 0\ . \end{aligned}$$

G_k and H_k are coefficients of a real-valued DFT of length N; hence:

$$G_k = G_{-k}^{*} \qquad \text{and} \qquad H_k = H_{-k}^{*} \qquad \text{for } k = 0, \dots, \tfrac{N}{2} - 1$$

Compact Real-valued FFT (2)

Use symmetry of G_k and H_k for the computation of F_k:

$$
\begin{aligned}
F_k &= \frac{1}{2}\left(G_k + \omega_{2N}^{k} H_k\right) && \text{für } k = 0, \dots, \tfrac{N}{2}\,, \\
F_{N-k} &= \frac{1}{2}\left(G_{-k} - \omega_{2N}^{-k} H_{-k}\right) \\
&= \frac{1}{2}\left(G_k - \omega_{2N}^{k} H_k\right)^{*} && \text{for } k = 0, \dots, \tfrac{N}{2} - 1
\end{aligned}
$$

$\Rightarrow$ Computation of F_k (for $k = 0, \dots, N$) reduced to the computation of G_k and H_k (for $k = 0, \dots, \frac{N}{2}$, respectively).

"Edson's algorithm" (1968)