Algorithms of Scientific Computing

Discrete Cosine Transform (DCT)

Michael Bader

Summer Term 2012

DFT and Symmetry

	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

Application Example: Compression of Image Data (JPEG)

Compression steps of the JPEG method

1. Conversion into a suitable colour model (YCbCr, e.g.), separation of brightness and colour information
2. Downsampling (in particular of the colour components)
3. **blockwise "quarter-wave discrete cosine transform"** (blocks of size 8×8)
4. **Quantification of the coefficients** ($\rightarrow$ reduce information)
5. run-length encoding, Huffman/arithmetic coding (loss-free compression of the quantified coefficients)

Example: jpeg on matlab central (see link on webpage)

Discrete Fourier Transform (DFT)

Definition:

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

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

Interpretation:

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

Fourier Coefficients and Numerical Quadrature

For a 2π-periodic function f, the corresponding **Fourier series** is defined as

$$f(x) \sim \sum_{k=-\infty}^{\infty} c_k e^{ikx}, \qquad \text{mit} \quad c_k = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-ikx}\, dx$$

The c_k are called (continuous) **Fourier coefficients**.

If f is piecewisely smooth, Fourier series converges pointwisely (i.e. for each x) towards

$$\tfrac{1}{2}\left(f(x^+) + f(x^-)\right),$$

i.e. in particular towards $f(x)$, if f is coninuously differentiable at x.

Approximate Computation of c_k

The continuous Fourier coefficients are given as

$$c_k = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-ikx} \, dx$$

Approaches to compute c_k approximately:

- compute c_k only for $\pm k = 0, \ldots, K$; then: $f(x) \approx \sum_{k=-K}^{K} c_k e^{ikx}$
- compute integral $\int_0^{2\pi} f(x) e^{-ikx} \, dx$ numerically

Computation of c_k via Trapezoidal Sum

Trapezoidal sum : for equidistant $x_n := \frac{2\pi n}{N}$:

$$\int_0^{2\pi} g(x)\,dx \approx T_N\{g\} := \frac{2\pi}{N}\left(\frac{1}{2}g(x_0) + \sum_{n=1}^{N-1} g(x_n) + \frac{1}{2}g(x_N)\right)$$

Use $g(x) := f(x)e^{-ikx}$ and $f_n := f(x_n)$; hence:

$$c_k \approx \frac{1}{2\pi} T_N\{f(x)e^{-ikx}\} = \frac{1}{N}\left(\frac{1}{2}f_0 e^0 + \sum_{n=1}^{N-1} f_n e^{-i2\pi nk/N} + \frac{1}{2}f_N e^{-i2\pi Nk/N}\right)$$

$$= \frac{1}{N}\left(\frac{f_0}{2} + \sum_{n=1}^{N-1} f_n e^{-i2\pi nk/N} + \frac{f_N}{2}\right)$$

Computation of c_k via Trapezoidal Sum (2)

If $f_0 = f_N$ (periodic data), we obtain

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

- ⇒ F_k are approximations of c_k
- ⇒ approximate computation leads to solution of the interpolation problem
- ⇒ approximation error is of order $\mathcal{O}(N^{-2})$

For $f_0 \neq f_N$, or for "discontinuities", we get a recommendation:

Average Values at Endpoints and Discontinuities (AVED)

Computation of c_k via Midpoint Rule

Midpoint rule: evaluate $g(x)$ at midpoints x_n:

$$\int_0^{2\pi} g(x)\,dx \approx \frac{2\pi}{N} \sum_{n=0}^{N-1} g(x_n) \qquad \text{with} \quad x_n := \frac{2\pi\left(n+\frac{1}{2}\right)}{N}\;.$$

With $g(x) := f(x)e^{-ikx}$ and $f_n := f(x_n)$, we obtain:

$$c_k \approx \widetilde{F}_k := \frac{1}{N} \sum_{n=0}^{N-1} f_n e^{-i2\pi\left(n+\frac{1}{2}\right)k/N}$$

"Quarter-Wave Discrete Fourier Transform"

Quarter-Wave Discrete Fourier Transform

- new variant of DFT:

$$\widetilde{F}_k := \frac{1}{N}\sum_{n=0}^{N-1} f_n e^{-i2\pi\left(n+\frac{1}{2}\right)k/N} \qquad f_n := \sum_{k=0}^{N-1} \widetilde{F}_k e^{i2\pi\left(n+\frac{1}{2}\right)k/N}$$

- Comparison with coefficients F_k of the "usual" DFT:

$$F_k = \widetilde{F}_k e^{i\pi k/N} = \widetilde{F}_k \omega_N^{k/2}$$

- Supporting points compared to "usual" DFT shifted by a "quarter wave length" (midpoints of intervals).
- Derivation via midpoint rule motivates usage for piecewise constant data

$\Rightarrow$ **Transformation of image data**

Quarter-Wave DFT on Symmetric Data

Given $2N$ real-valued input data $f_0, \dots, f_{2N-1}$ with symmetry

$$f_{2N-n-1} = f_n$$

Inserting the symmetric data in Quarter-Wave DFT results in

$$\begin{aligned}
\widetilde{F}_k &= \frac{1}{2N} \sum_{n=0}^{2N-1} f_n \omega_{2N}^{-k\left(n+\frac{1}{2}\right)} \\
&= \frac{1}{2N} \sum_{n=0}^{N-1} f_n \omega_{2N}^{-k\left(n+\frac{1}{2}\right)} + \frac{1}{2N} \sum_{n=0}^{N-1} f_{2N-n-1} \omega_{2N}^{-k\left(2N-n-1+\frac{1}{2}\right)} \\
&= \frac{1}{2N} \sum_{n=0}^{N-1} f_n \left(\omega_{2N}^{-k\left(n+\frac{1}{2}\right)} + \omega_{2N}^{-k\left(-n-\frac{1}{2}\right)} \right) = \frac{1}{N} \sum_{n=0}^{N-1} f_n \cos\left(\frac{\pi k \left(n+\frac{1}{2}\right)}{N} \right)
\end{aligned}$$

Quarter-Wave DFT on Symmetric Data (2)

Quarter-Wave DFT of symmetric data results in **real-valued** coefficients:

$$\widetilde{F}_k = \frac{1}{N} \sum_{n=0}^{N-1} f_n \cos\left(\frac{\pi k \left(n + \frac{1}{2}\right)}{N} \right) \qquad \text{for } k = 0, \ldots, 2N-1$$

Additional symmetry:

$$\begin{aligned} \widetilde{F}_{2N-k} &= \frac{1}{N} \sum_{n=0}^{N-1} f_n \cos\left(\frac{\pi(2N-k)\left(n + \frac{1}{2}\right)}{N} \right) \\ &= \frac{1}{N} \sum_{n=0}^{N-1} f_n \cos\left(2\pi n + \pi - \frac{\pi k \left(n + \frac{1}{2}\right)}{N} \right) = -\widetilde{F}_k \end{aligned}$$

$\Rightarrow$ again: only N independent coefficients

Quarter-Wave Even Discrete Cosine Transform

backward transform:

$$f_n := \sum_{k=0}^{2N-1} \widetilde{F}_k e^{i2\pi\left(n+\frac{1}{2}\right)k/2N} \quad \xrightarrow{\widetilde{F}_{2N-k}=-\widetilde{F}_k} \quad f_n = \widetilde{F}_0 + 2\sum_{k=1}^{N-1} \widetilde{F}_k \cos\left(\frac{\pi k\left(n+\frac{1}{2}\right)}{N}\right)$$

Definition of the quarter-wave even DCT:

$$\widetilde{F}_k = \frac{1}{N}\sum_{n=0}^{N-1} f_n \cos\left(\frac{\pi k\left(n+\frac{1}{2}\right)}{N}\right) \qquad f_n = \widetilde{F}_0 + 2\sum_{k=1}^{N-1} \widetilde{F}_k \cos\left(\frac{\pi k\left(n+\frac{1}{2}\right)}{N}\right)$$

N **real-valued data** $\longleftrightarrow$ N **real-valued coefficients**

(no symmetry any more in data/coefficients!)

2D Cosine Transform

Definition of the 2D-DCT:

$$\widetilde{F}_{kl} = \frac{1}{N \cdot M} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f_{nm} \cos\left(\frac{\pi k \left(n + \frac{1}{2}\right)}{N} \right) \cos\left(\frac{\pi l \left(m + \frac{1}{2}\right)}{M} \right)$$

$$f_{nm} = 4 \sum_{k=0}^{N-1}{}' \sum_{l=0}^{M-1}{}' \widetilde{F}_{kl} \cos\left(\frac{\pi k \left(n + \frac{1}{2}\right)}{N} \right) \cos\left(\frac{\pi l \left(m + \frac{1}{2}\right)}{M} \right)$$

shortened notation: $\sum_{k=0}^{N-1}{}' x_k := \frac{x_0}{2} + \sum_{k=1}^{N-1} x_k$

Application: blockwise 2D-DCT in JPEG/MPEG compression

Reduction of the 2D-FCT to 1D-FCTs

In the 2D cosine transform, we can rearrange:

$$\begin{aligned}
\widetilde{F}_{kl} &= \frac{1}{N^2}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} f_{nm}\cos\left(\frac{\pi k\left(n+\frac{1}{2}\right)}{N}\right)\cos\left(\frac{\pi l\left(m+\frac{1}{2}\right)}{N}\right)\\
&= \frac{1}{N}\sum_{n=0}^{N-1}\underbrace{\left(\frac{1}{N}\sum_{m=0}^{N-1} f_{nm}\cos\left(\frac{\pi l\left(m+\frac{1}{2}\right)}{N}\right)\right)}_{:=\widehat{F}_{nl}}\cos\left(\frac{\pi k\left(n+\frac{1}{2}\right)}{N}\right).
\end{aligned}$$

- For each n, $\widehat{F}_{nl}$ are computed via N 1D transforms
- we may first 1D-transform all rows and then all columns to get the 2D-transform

Application Example: Compression of Image Data (JPEG)

Compression steps of the JPEG method

1. Conversion into a suitable colour model (YCbCr, e.g.), separation of brightness and colour information
2. Downsampling (in particular of the colour components)
3. **blockwise "quarter-wave discrete cosine transform"** (blocks of size 8×8)
4. **Quantification of the coefficients** ($\rightarrow$ reduce information)
5. run-length encoding, Huffman/arithmetic coding (loss-free compression of the quantified coefficients)

Example: jpeg on matlab central (see link on webpage)

QW-DCT – Algorithm

Reduce to Real FFT:

(1) for $n = 0, \ldots, N-1$:

$$g_n = f_n \qquad g_{2N-n-1} = f_n$$

(2) $2N$-Real-FFT: compute G_k from g_n (for $k = 0, \ldots, N$)

(3) for $k = 0, \ldots, N-1$:

$$\widetilde{F}_k = G_k e^{-i\pi k/2N}$$

Further optimisations:

- substitute real-valued $2N$-FFT by complex N-FFT
- compact (divide-and-conquer) real FFT
- compact Fast DCT $\longrightarrow$ paper *Swarztrauber*

Compact Fast DCT

Consider DCT: with symmetry $f_{2N-n-1} = f_n$

$$\widetilde{F}_k = \frac{1}{2N} \sum_{n=0}^{2N-1} f_n \omega_{2N}^{-k\left(n+\frac{1}{2}\right)} \quad \longrightarrow \quad \widetilde{F}_k = \frac{1}{N} \sum_{n=0}^{N-1} f_n \cos\left(\frac{\pi k \left(n+\frac{1}{2}\right)}{N}\right) .$$

Split into even and odd indices: $g_n := f_{2n}$ and $h_n := f_{2n+1}$ (as in FFT)

- $g_n := f_{2n}$:

$$g_n = f_{2n} = f_{2N-2n-1} = f_{2(N-n)-1} = f_{2(N-n-1)+1} = h_{N-n-1}$$

- $h_n := f_{2n+1}$:

$$h_n = f_{2n+1} = f_{2N-(2n+1)-1} = f_{2(N-n-1)} = g_{N-n-1}$$

- thus: **two real DFTs** with symmetric data sets

Compact Fast IDCT

Consider backward transform: with symmetry $\widetilde{F}_{2N-k} = -\widetilde{F}_k$

$$f_n := \sum_{k=0}^{2N-1} \widetilde{F}_k e^{i2\pi\left(n+\frac{1}{2}\right)k/2N} \quad \longrightarrow \quad f_n = \widetilde{F}_0 + 2\sum_{k=1}^{N-1} \widetilde{F}_k \cos\left(\frac{\pi k\left(n+\frac{1}{2}\right)}{N}\right)$$

Split into even and odd indices: (as in FFT)

- $G_k := \widetilde{F}_{2k}$: **leads to IDCT**

$$-G_k = -\widetilde{F}_{2k} = \widetilde{F}_{2N-2k} = \widetilde{F}_{2(N-k)} = G_{N-k}$$

- $H_k := \widetilde{F}_{2k+1}$: **leads to "DCT with negative exponent"??**

$$-H_k = -\widetilde{F}_{2k+1} = \widetilde{F}_{2N-(2k+1)} = \widetilde{F}_{2(N-k)-1} = \widetilde{F}_{2(N-k-1)+1} = H_{N-k-1}$$

$\longrightarrow$ to be continued . . .