Algorithms and Computation in Signal Processing

special topic course 18-799B
spring 2005
10th Lecture Feb. 10, 2005

Instructor: Markus Pueschel
TA: Srinivas Chellappa

Discrete Signal Transforms

Signal Transforms

- Mathematically: Matrix-vector multiplication

$$x \mapsto y = M \cdot x$$

Input vector (signal)

Output vector (signal)

Transform = a fixed matrix

- Is often written like that:

$$y_k = \sum_{\ell=0}^{n-1} m_{k,\ell} x_\ell, \qquad M = [m_{k,\ell}]_{0 \le k,\ell < n}$$

- Idea behind transforms: represent signal in other basis which enables better manipulation

Transforms: Examples

$$
\begin{aligned}
\mathbf{DFT}_n &= [e^{-2k\ell\pi i/n}]_{0\le k,\ell<n} \\
\mathbf{DCT\text{-}2}_n &= \left[\cos(k(2\ell+1)\pi/2n)\right]_{0\le k,\ell<n}, \\
\mathbf{DCT\text{-}3}_n &= \mathbf{DCT\text{-}2}_n^T \quad \text{(transpose)}, \\
\mathbf{DCT\text{-}4}_n &= \left[\cos((2k+1)(2\ell+1)\pi/4n)\right]_{0\le k,\ell<n}, \\
\mathbf{IMDCT}_n &= \left[\cos((2k+1)(2\ell+1+n)\pi/4n)\right]_{0\le k<2n,0\le\ell<n}, \\
\mathbf{RDFT}_n &= [r_{k\ell}]_{0\le k,\ell<n}, \quad r_{k\ell} = \begin{cases} \cos\frac{2\pi k\ell}{n}, & k \le \lfloor\frac{n}{2}\rfloor \\ -\sin\frac{2\pi k\ell}{n}, & k > \lfloor\frac{n}{2}\rfloor \end{cases}, \\
\mathbf{WHT}_n &= \begin{bmatrix} \mathbf{WHT}_{n/2} & \mathbf{WHT}_{n/2} \\ \mathbf{WHT}_{n/2} & -\mathbf{WHT}_{n/2} \end{bmatrix}, \quad \mathbf{WHT}_2 = \mathbf{DFT}_2, \\
\mathbf{DHT} &= \left[\cos(2k\ell\pi/n) + \sin(2k\ell\pi/n)\right]_{0\le k,\ell<n}.
\end{aligned}
$$

Fast Algorithms

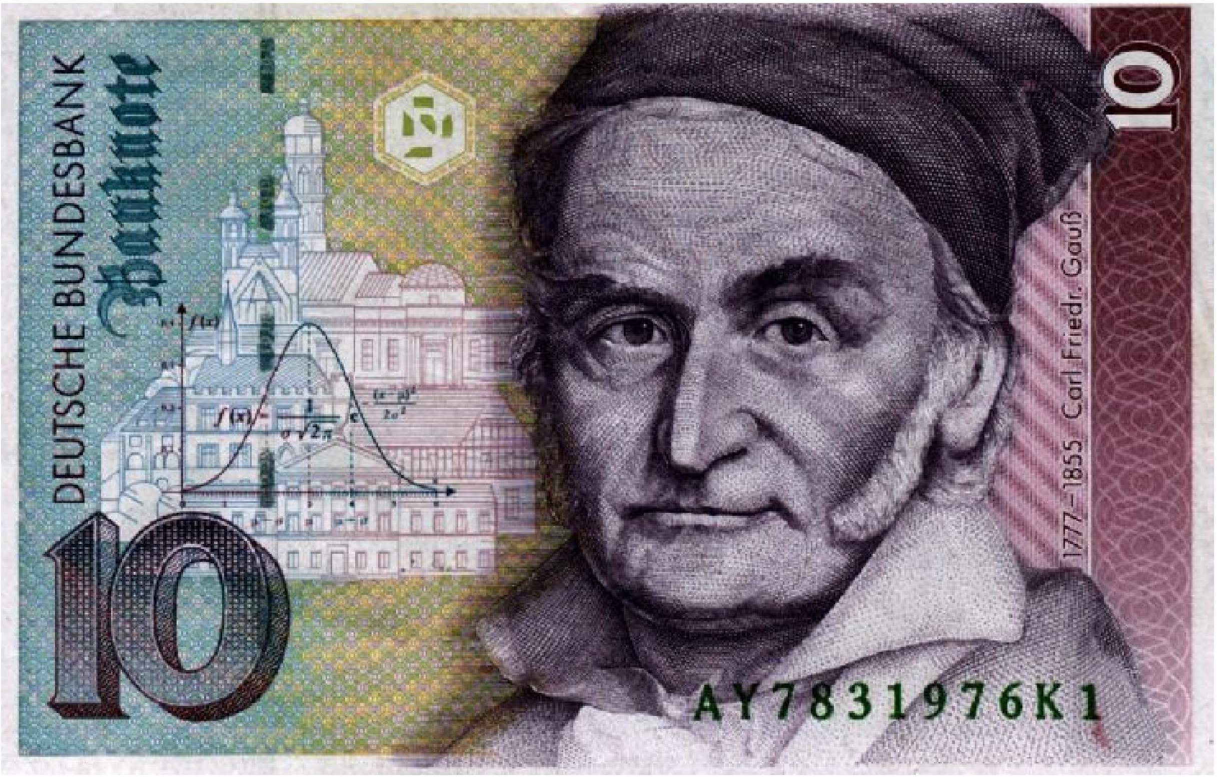
- Transforms by definition: $O(n^2)$
- Transforms have fast algorithm, typically: $O(n \log(n))$
- Example: DFT and FFT (fast Fourier transform)
- These fast algorithms exist because of “symmetries” or “redundancy” in the matrix (however, the connection is not straightforward)

The History of Fast Transform Algorithms …

- … starts with the FFT
- The advent of digital signal processing is often attributed to the FFT (Cooley-Tukey 1965)
- History:
  - ~1805: FFT discovered by Gauss (nach [1])
    (Fourier publishes the concept of Fourier analysis in 1807!)
  - 1965: Rediscovered by Cooley-Tukey
  - 2002: James W. Cooley receives the IEEE Jack S. Kilby Signal Processing Medal
    "For pioneering the Fast Fourier Transform (FFT) algorithm."

[1]: Heideman, Johnson, Burrus: "Gauss and the History of the Fast Fourier Transform" Arch. Hist. Sc. 34(3) 1985

Carl-Friedrich Gauss



1777 - 1855

- Contender for the greatest mathematician of all times
- Some contributions: Least square analysis, normal distribution, fundamental theorem of algebra, Gauss elimination, Gauss quadrature, Gauss-Seidel, non-euclidean geometry, …

How are Transform Algorithms Being Found?

- Staring at the matrix for a long time
- Example: Cooley-Tukey FFT
- Other Example: *G. Bi "Fast Algorithms for the Type-III DCT of Composite Sequence Lengths" IEEE Trans. SP 47(7) 1999* *link*

Representation of Transform Algorithms

- Representation of algorithms: two schools
  - Sequence of summations
  - Structured matrix factorization: $M = M_1 \cdot M_2 \ldots M_k$
- Example Bi's algorithm:

$$\mathrm{DCT}_n = K_m^n \Big(\bigoplus_{0 \le i < k} \mathrm{DCT}_m(r_i) \Big) (\mathrm{DCT}_k \otimes I_m) B_{n,k}$$

Example Cooley-Tukey FFT: What is Better?

This?:

$$\mathrm{DFT}_n = L^n_{n_2}(I_{n_1} \otimes \mathrm{DFT}_{n_2}) T^n_{n_1} (\mathrm{DFT}_{n_1} \otimes I_{n_2})$$

Or this?: $k = n_1 k_1 + k_2, \; j = n_2 j_1 + j_2$

$$y_{n_2 j_1 + j_2} = \sum_{k_1=0}^{n_1-1} \left(\omega_n^{j_2 k_1}\right) \left(\sum_{k_2=0}^{n_2-1} x_{n_1 k_2 + k_1} \omega_{n_2}^{j_2 k_2} \right) \omega_{n_1}^{j_1 k_1}$$

FFT References

- Nussbaumer (1982): Fast Fourier Transforms and Convolution Algorithms
- Van Loan (1992): Computational Frameworks for the Fast Fourier Transform
- Clausen/Baum (1993): Fast Fourier Transforms
- Tolimieri/An/Lu (1997): Algorithms for discrete Fourier transform and convolution