How to Write Fast Numerical Code

Spring 2017

Lecture: Discrete Fourier transform, fast Fourier transforms

Instructor: Markus Püschel

TA: Alen Stojanov, Georg Ofenbeck, Gagandeep Singh

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Linear Transforms

- **Overview: Transforms and algorithms**

- **Discrete Fourier transform**

- **Fast Fourier transforms**

- **After that:**
  - Optimized implementation and autotuning (FFTW)
  - Automatic program synthesis (Spiral)

2

Blackboard

- **Linear Transforms**
- **Discrete Fourier transform (DFT)**
- **Transform algorithms**
- **Fast Fourier transform, size 4**

FFT References

- **Complexity:** *Bürgisser, Clausen, Shokrollahi, **Algebraic Complexity Theory**, Springer, 1997*

- **History:** *Heideman, Johnson, Burrus: **Gauss and the History of the Fast Fourier Transform**, Arch. Hist. Sc. 34(3) 1985*

- **FFTs:**
  - *Cooley and Tukey, **An algorithm for the machine calculation of complex Fourier series,"** Math. of Computation, vol. 19, pp. 297–301, 1965*
  - *Nussbaumer, **Fast Fourier Transform and Convolution Algorithms**, 2nd ed., Springer, 1982*
  - *van Loan, **Computational Frameworks for the Fast Fourier Transform**, SIAM, 1992*
  - *Tolimieri, An, Lu, **Algorithms for Discrete Fourier Transforms and Convolution**, Springer, 2nd edition, 1997*
  - *Franchetti, Püschel, Voronenko, Chellappa and Moura, **Discrete Fourier Transform on Multicore**, IEEE Signal Processing Magazine, special issue on ``Signal Processing on Platforms with Multiple Cores'', Vol. 26, No. 6, pp. 90-102, 2009*

Linear Transforms

- **Very important class of functions: signal processing, scientific computing, …**

- *Mathematically:* **Change of basis = Multiplication by a fixed matrix T**

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} = y = Tx \quad \longleftarrow \quad \boxed{T \cdot} \quad \longleftarrow \quad x = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix}$$

$$T = [t_{k,\ell}]_{0 \le k, \ell < n}$$

Output   *Input*

- **Equivalent definition: Summation form**

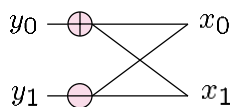$$y_k = \sum_{\ell=0}^{n-1} t_{k,\ell} x_\ell, \quad 0 \le k < n$$

5

Smallest Relevant Example: DFT, Size 2

Transform (matrix): $T = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

Computation: $y = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} x$ *or* $\begin{aligned} y_0 &= x_0 + x_1 \\ y_1 &= x_0 - x_1 \end{aligned}$

As graph (direct acyclic graph or DAG):

called a butterfly

http://charlottesmartypants.blogspot.com/
2011_02_01_archive.html

6

Transforms: Examples

- **A few dozen transforms are relevant**
- **Some examples**

$$\mathrm{DFT}_n = [e^{-2k\ell\pi i/n}]_{0 \le k,\ell < n}$$

$$\mathrm{RDFT}_n = [r_{k\ell}]_{0 \le k,\ell < n}, \quad r_{k\ell} = \begin{cases} \cos\frac{2\pi k\ell}{n}, & k \le \lfloor\frac{n}{2}\rfloor \\ -\sin\frac{2\pi k\ell}{n}, & k > \lfloor\frac{n}{2}\rfloor \end{cases} \qquad \textit{universal tool}$$

$$\mathrm{DHT} = \left[\cos(2k\ell\pi/n) + \sin(2k\ell\pi/n)\right]_{0 \le k,\ell < n}$$

$$\mathrm{WHT}_n = \begin{bmatrix} \mathrm{WHT}_{n/2} & \mathrm{WHT}_{n/2} \\ \mathrm{WHT}_{n/2} & -\mathrm{WHT}_{n/2} \end{bmatrix}, \quad \mathrm{WHT}_2 = \mathrm{DFT}_2$$

$$\mathrm{IMDCT}_n = \left[\cos((2k+1)(2\ell+1+n)\pi/4n)\right]_{0 \le k < 2n, 0 \le \ell < n} \qquad \textbf{\textit{MPEG}}$$

$$\mathrm{DCT\text{-}2}_n = \left[\cos(k(2\ell+1)\pi/2n)\right]_{0 \le k,\ell < n} \qquad \textbf{\textit{JPEG}}$$

$$\mathrm{DCT\text{-}3}_n = \mathrm{DCT\text{-}2}_n^T \quad \text{(transpose)}$$

$$\mathrm{DCT\text{-}4}_n = \left[\cos((2k+1)(2\ell+1)\pi/4n)\right]_{0 \le k,\ell < n}$$

Linear Transforms: DFT

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} = y = Tx \quad \longleftarrow \quad \boxed{T\cdot} \quad \longleftarrow \quad x = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix}$$

$$T = [t_{k,\ell}]_{0 \le k,\ell < n}$$

Output   *Input*

Example: $\quad T = \mathrm{DFT}_n = [e^{-2k\ell\pi i/n}]_{0 \le k,\ell < n}$

$$= [\omega_n^{k\ell}]_{0 \le k,\ell < n}, \quad \omega_n = e^{-2\pi i/n}$$
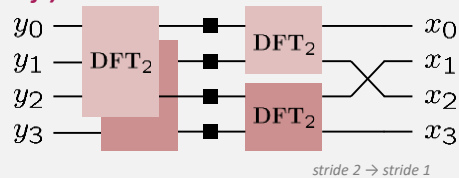
Algorithms: Example FFT, n = 4

Fast Fourier transform (FFT)

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} x = \begin{bmatrix} 1 & \cdot & 1 & \cdot \\ \cdot & 1 & \cdot & 1 \\ 1 & \cdot & -1 & \cdot \\ \cdot & 1 & \cdot & -1 \end{bmatrix} \begin{bmatrix} 1 & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & i \end{bmatrix} \begin{bmatrix} 1 & 1 & \cdot & \cdot \\ 1 & -1 & \cdot & \cdot \\ \cdot & \cdot & 1 & 1 \\ \cdot & \cdot & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot \\ \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 \end{bmatrix} x$$

Representation using matrix algebra

$$\mathrm{DFT}_4 = (\mathrm{DFT}_2 \otimes \mathrm{I}_2)\, \mathrm{diag}(1,1,1,i)(\mathrm{I}_2 \otimes \mathrm{DFT}_2)\, \mathsf{L}_2^4$$

Data flow graph (right to left)
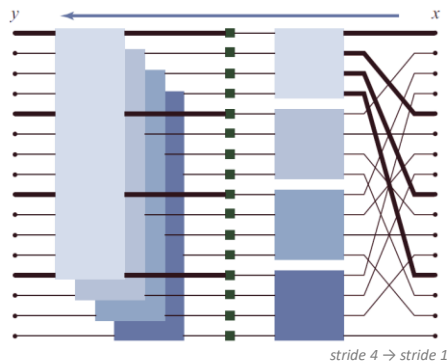

stride 2 → stride 1

Cooley-Tukey FFT (Recursive, General-Radix)

- **Blackboard**
- **Kronecker products**
- **Stride permutations**

Example FFT, n = 16 *(Recursive, Radix 4)*

$$\mathrm{DFT}_{16} = $$

$\mathrm{DFT}_4 \otimes I_4$ T_4^{16} $I_4 \otimes \mathrm{DFT}_4$ L_4^{16}

y   *x*

stride 4 → stride 1

Recursive Cooley-Tukey FFT

radix

$$\mathrm{DFT}_{km} = (\mathrm{DFT}_k \otimes \mathrm{I}_m) T_m^{km} (\mathrm{I}_k \otimes \mathrm{DFT}_m) L_k^{km} \qquad \textit{decimation-in-time}$$

$$\mathrm{DFT}_{km} = L_m^{km} (\mathrm{I}_k \otimes \mathrm{DFT}_m) T_m^{km} (\mathrm{DFT}_k \otimes \mathrm{I}_m) \qquad \textit{decimation-in-frequency}$$

- **For powers of two n = 2ᵗ sufficient together with base case**

$$\mathrm{DFT}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

- **Cost:**
  - (complex adds, complex mults) = (n log₂(n), n log₂(n)/2)
    independent of recursion
  - (real adds, real mults) ≤ (2n log₂(n), 3n log₂(n)) = 5n log₂(n) flops
    depends on recursion: best is at least radix-8

Recursive vs. Iterative FFT

■ **Recursive, radix-k Cooley-Tukey FFT**

$$\mathrm{DFT}_{km} = (\mathrm{DFT}_k \otimes \mathrm{I}_m) T_m^{km} (\mathrm{I}_k \otimes \mathrm{DFT}_m) L_k^{km}$$

$$\mathrm{DFT}_{km} = L_m^{km} (\mathrm{I}_k \otimes \mathrm{DFT}_m) T_m^{km} (\mathrm{DFT}_k \otimes \mathrm{I}_m)$$

■ **Iterative, radix 2, decimation-in-time/decimation-in-frequency**

$$\mathrm{DFT}_{2^t} = \left(\prod_{j=1}^{t} (\mathrm{I}_{2^{j-1}} \otimes \mathrm{DFT}_2 \otimes \mathrm{I}_{2^{t-j}}) \cdot (\mathrm{I}_{2^{j-1}} \otimes T_{2^{t-j}}^{2^{t-j+1}}) \right) \cdot R_{2^t}$$
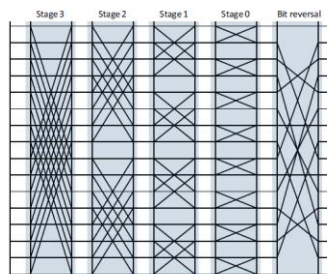
$$\mathrm{DFT}_{2^t} = R_{2^t} \cdot \left(\prod_{j=1}^{t} (\mathrm{I}_{2^{t-j}} \otimes T_{2^{j-1}}^{2^j}) \cdot (\mathrm{I}_{2^{t-j}} \otimes \mathrm{DFT}_2 \otimes \mathrm{I}_{2^{j-1}}) \right)$$

13
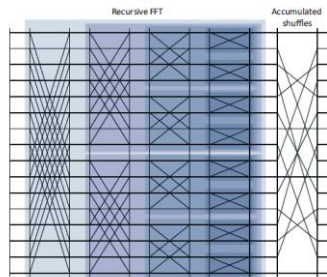
Radix 2, recursive

$$(\mathrm{DFT}_2 \otimes I_8) T_8^{16} \Big(I_2 \otimes \big((\mathrm{DFT}_2 \otimes I_4) T_4^8 \big(I_2 \otimes \big((\mathrm{DFT}_2 \otimes I_2) T_2^4 (I_2 \otimes \mathrm{DFT}_2) L_2^4 \big) \big) L_2^8 \big) \Big) L_2^{16}$$

Radix 2, iterative

$$\Big((I_1 \otimes \mathrm{DFT}_2 \otimes I_8) D_0^{16} \Big) \Big((I_2 \otimes \mathrm{DFT}_2 \otimes I_4) D_1^{16} \Big) \Big((I_4 \otimes \mathrm{DFT}_2 \otimes I_2) D_2^{16} \Big) \Big((I_8 \otimes \mathrm{DFT}_2 \otimes I_1) D_3^{16} \Big) R_2^{16}$$

© Markus Püschel
ETH
Computer Science Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

How to write fast numerical code
Spring 2017

Recursive vs. Iterative

- **Iterative FFT computes in stages of butterflies =**
  $log_2(n)$ passes through the data

- **Recursive FFT reduces passes through data =**
  better locality

- **Same computation graph but different topological sorting**

- **Rough analogy:**

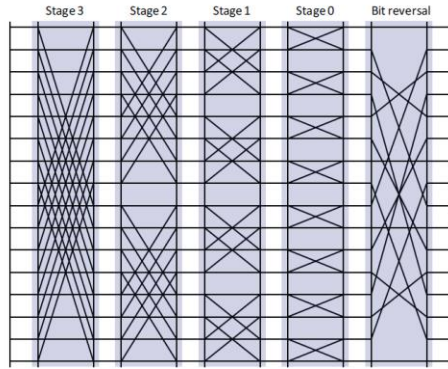| MMM	DFT
Triple loop	Iterative FFT
Blocked	Recursive FFT

The FFT Is Very Malleable

Iterative FFT, Radix 2

$$\Big((I_1 \otimes \mathrm{DFT}_2 \otimes I_8)D_0^{16}\Big)\Big((I_2 \otimes \mathrm{DFT}_2 \otimes I_4)D_1^{16}\Big)\Big((I_4 \otimes \mathrm{DFT}_2 \otimes I_2)D_2^{16}\Big)\Big((I_8 \otimes \mathrm{DFT}_2 \otimes I_1)D_3^{16}\Big)R_2^{16}$$

17

Pease FFT, Radix 2
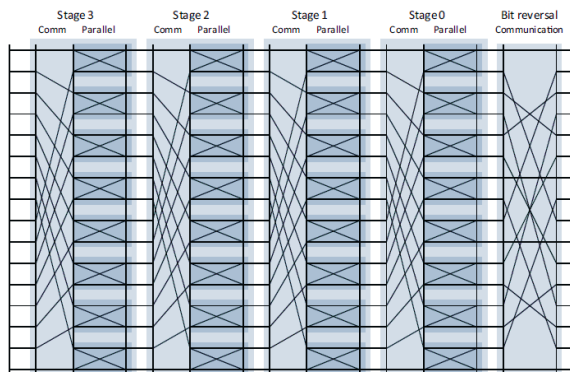
$$\Big(L_2^{16}(I_8 \otimes \mathrm{DFT}_2)D_0^{16}\Big)\Big(L_2^{16}(I_8 \otimes \mathrm{DFT}_2)D_1^{16}\Big)\Big(L_2^{16}(I_8 \otimes \mathrm{DFT}_2)D_2^{16}\Big)\Big(L_2^{16}(I_8 \otimes \mathrm{DFT}_2)D_3^{16}\Big)R_2^{16}$$

18

© Markus Püschel
Computer Science

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

How to write fast numerical code
Spring 2017

Stockham FFT, Radix 2
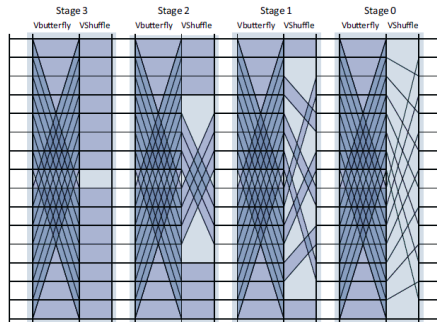
$$\Big(\big(\mathrm{DFT}_2 \otimes I_8 \big) D_0^{16} \big(L_2^2 \otimes I_8 \big) \Big) \Big(\big(\mathrm{DFT}_2 \otimes I_8 \big) D_1^{16} \big(L_2^4 \otimes I_4 \big) \Big) \Big(\big(\mathrm{DFT}_2 \otimes I_8 \big) D_2^{16} \big(L_2^8 \otimes I_2 \big) \Big) \Big(\big(\mathrm{DFT}_2 \otimes I_8 \big) D_3^{16} \big(L_2^{16} \otimes I_1 \big) \Big)$$

19

Six-Step FFT
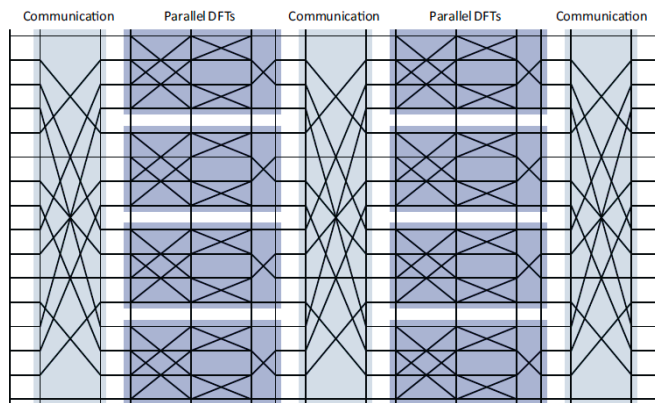
$$L_4^{16} \Big(I_4 \otimes \big((\mathrm{DFT}_2 \otimes I_2) T_2^4 (I_2 \otimes \mathrm{DFT}_2) L_2^4 \big) \Big) L_4^{16} T_4^{16} \Big(I_4 \otimes \big((\mathrm{DFT}_2 \otimes I_2) T_2^4 (I_2 \otimes \mathrm{DFT}_2) L_2^4 \big) \Big) L_4^{16}$$

20

Multi-Core FFT
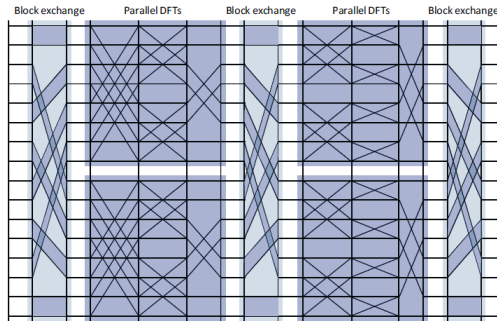
$$\left(L_4^8 \otimes I_2\right)\left(I_2 \otimes\left(\left(\mathrm{DFT}_2 \otimes I_2\right)T_2^4\left(I_2 \otimes \mathrm{DFT}_2\right)L_2^4\right)\otimes I_2\right)\left(L_2^8 \otimes I_2\right)T_4^{16}\left(I_2 \otimes\left(I_2 \otimes\left(\mathrm{DFT}_2 \otimes I_2\right)T_2^4\left(I_2 \otimes \mathrm{DFT}_2\right)\right)R_2^8\right)\left(L_2^8 \otimes I_2\right)$$

Transform Algorithms

$$\mathrm{DFT}_n \rightarrow P_{k/2,2m}^\top\left(\mathrm{DFT}_{2m} \oplus \left(I_{k/2-1} \otimes_i C_{2m}\,\mathrm{rDFT}_{2m}(i/k)\right)\right)\left(\mathrm{RDFT}_k' \otimes I_m\right), \quad k \text{ even,}$$

$$\left|\begin{matrix}\mathrm{RDFT}_n \\ \mathrm{RDFT}_n' \\ \mathrm{DHT}_n \\ \mathrm{DHT}_n'\end{matrix}\right| \rightarrow (P_{k/2,m}^\top \otimes I_2)\left(\left|\begin{matrix}\mathrm{RDFT}_{2m} \\ \mathrm{RDFT}_{2m}' \\ \mathrm{DHT}_{2m} \\ \mathrm{DHT}_{2m}'\end{matrix}\right| \oplus \left(I_{k/2-1} \otimes_i D_{2m}\left|\begin{matrix}\mathrm{rDFT}_{2m}(i/k) \\ \mathrm{rDFT}_{2m}(i/k) \\ \mathrm{rDHT}_{2m}(i/k) \\ \mathrm{rDHT}_{2m}(i/k)\end{matrix}\right|\right)\right)\left(\left|\begin{matrix}\mathrm{RDFT}_k' \\ \mathrm{RDFT}_k' \\ \mathrm{DHT}_k' \\ \mathrm{DHT}_k'\end{matrix}\right| \otimes I_m\right), \quad k \text{ even,}$$

$$\left|\begin{matrix}\mathrm{rDFT}_{2n}(u) \\ \mathrm{rDHT}_{2n}(u)\end{matrix}\right| \rightarrow L_m^{2n}\left(I_k \otimes_i \left|\begin{matrix}\mathrm{rDFT}_{2m}((i+u)/k) \\ \mathrm{rDHT}_{2m}((i+u)/k)\end{matrix}\right|\right)\left(\left|\begin{matrix}\mathrm{rDFT}_{2k}(u) \\ \mathrm{rDHT}_{2k}(u)\end{matrix}\right| \otimes I_m\right),$$

$$\mathrm{RDFT\text{-}3}_n \rightarrow (Q_{k/2,m}^\top \otimes I_2)(I_k \otimes_i \mathrm{rDFT}_{2m})(i+1/2)/k)(\mathrm{RDFT\text{-}3}_k \otimes I_m), \quad k \text{ even,}$$

$$\mathrm{DCT\text{-}2}_n \rightarrow P_{k/2,2m}^\top\left(\mathrm{DCT\text{-}2}_{2m}K_2^{2m} \oplus \left(I_{k/2-1} \otimes N_{2m}\,\mathrm{RDFT\text{-}3}_{2m}^\top\right)\right)B_n(L_{k/2}^{n/2} \otimes I_2)(I_m \otimes \mathrm{RDFT}_k')Q_{m/2,k},$$

$$\mathrm{DCT\text{-}3}_n \rightarrow \mathrm{DCT\text{-}2}_n^\top,$$

$$\mathrm{DCT\text{-}4}_n \rightarrow Q_{k/2,2m}^\top\left(I_{k/2} \otimes N_{2m}\,\mathrm{RDFT\text{-}3}_{2m}^\top\right)B_n'(L_{k/2}^{n/2} \otimes I_2)(I_m \otimes \mathrm{RDFT\text{-}3}_k)Q_{m/2,k}.$$

$$\mathrm{DFT}_n \rightarrow (\mathrm{DFT}_k \otimes I_m)\,\mathsf{T}_m^n(\mathsf{I}_k \otimes \mathrm{DFT}_m)\,\mathsf{L}_k^n, \quad n = km \text{ ——————— } \textit{Cooley-Tukey FFT}$$

$$\mathrm{DFT}_n \rightarrow P_n(\mathrm{DFT}_k \otimes \mathrm{DFT}_m)Q_n, \quad n = km,\ \gcd(k,m) = 1 \text{ ——————— } \textit{Prime-factor FFT}$$

$$\mathrm{DFT}_p \rightarrow R_p^T(\mathsf{I}_1 \oplus \mathrm{DFT}_{p-1})D_p(\mathsf{I}_1 \oplus \mathrm{DFT}_{p-1})R_p, \quad p \text{ prime } \text{ ——————— } \textit{Rader FFT}$$

$$\mathrm{DCT\text{-}3}_n \rightarrow (\mathsf{I}_m \oplus \mathsf{J}_m)\,\mathsf{L}_m^n(\mathrm{DCT\text{-}3}_m(1/4) \oplus \mathrm{DCT\text{-}3}_m(3/4))$$
$$\cdot (\mathsf{F}_2 \otimes \mathsf{I}_m)\begin{bmatrix}\mathsf{I}_m & 0 \oplus -\mathsf{J}_{m-1} \\ & \frac{1}{\sqrt{2}}(\mathsf{I}_1 \oplus 2\mathsf{I}_m)\end{bmatrix}, \quad n = 2m$$

$$\mathrm{DCT\text{-}4}_n \rightarrow S_n\mathrm{DCT\text{-}2}_n\,\mathrm{diag}_{0 \le k < n}(1/(2\cos((2k+1)\pi/4n)))$$

$$\mathrm{IMDCT}_{2m} \rightarrow (\mathsf{J}_m \oplus \mathsf{I}_m \oplus \mathsf{I}_m \oplus \mathsf{J}_m)\left(\left(\begin{bmatrix}1 \\ -1\end{bmatrix} \otimes \mathsf{I}_m\right) \oplus \left(\begin{bmatrix}-1 \\ -1\end{bmatrix} \otimes \mathsf{I}_m\right)\right)\mathsf{J}_{2m}\,\mathrm{DCT\text{-}4}_{2m}$$

$$\mathrm{WHT}_{2^k} \rightarrow \prod_{i=1}^t (\mathsf{I}_{2^{k_1+\cdots+k_{i-1}}} \otimes \mathrm{WHT}_{2^{k_i}} \otimes \mathsf{I}_{2^{k_{i+1}+\cdots+k_t}}), \quad k = k_1 + \cdots + k_t$$

$$\mathrm{DFT}_2 \rightarrow \mathsf{F}_2$$

$$\mathrm{DCT\text{-}2}_2 \rightarrow \mathrm{diag}(1, 1/\sqrt{2})\,\mathsf{F}_2$$

$$\mathrm{DCT\text{-}4}_2 \rightarrow \mathsf{J}_2\,\mathsf{R}_{13\pi/8}$$

How to write fast numerical code
Spring 2017

Complexity of the DFT

- **Measure: L_c, $2 \leq c$**
    - Complex adds count 1
    - Complex mult by a constant a with $|a| < c$ counts 1
    - L_2 is strictest, L_∞ the loosest (and most natural)
- **Upper bounds:**
    - $n = 2^k$: $L_2(DFT_n) \leq 3/2 \, n \log_2(n)$ *(using Cooley-Tukey FFT)*
    - General n: $L_2(DFT_n) \leq 8 \, n \log_2(n)$ *(needs Bluestein FFT)*
- **Lower bound:**
    - Theorem by Morgenstern: If $c < \infty$, then $L_c(DFT_n) \geq \frac{1}{2} \, n \log_c(n)$
    - Implies: in the measure L_c, the DFT is $\Theta(n \log(n))$

History of FFTs

- **The advent of digital signal processing is often attributed to the FFT**
  (Cooley-Tukey 1965)
- **History:**
    - Around 1805: FFT discovered by Gauss [1]
      (Fourier publishes the concept of Fourier analysis in 1807!)
    - 1965: Rediscovered by Cooley-Tukey

[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:** Modular arithmetic, least square analysis, normal distribution, fundamental theorem of algebra, Gauss elimination, Gauss quadrature, Gauss-Seidel, non-Euclidean geometry, …

25