How to Write Fast Numerical Code

Spring 2013

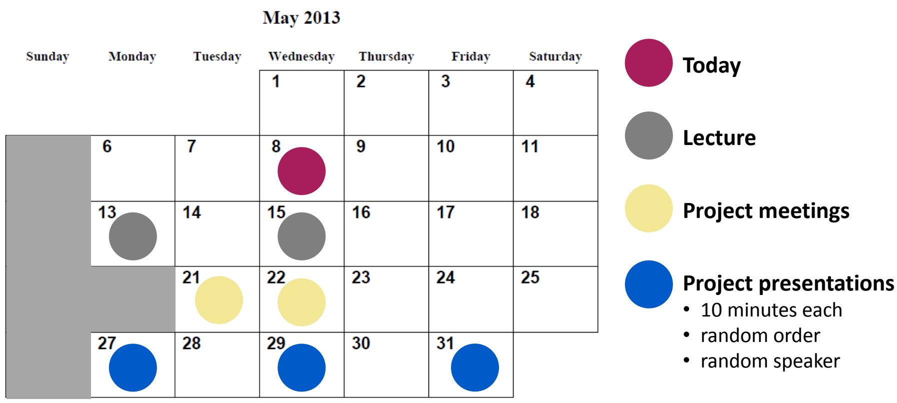
Lecture: Discrete Fourier transform, fast Fourier transforms

Instructor: Markus Püschel

TA: Georg Ofenbeck & Daniele Spampinato

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Rest of Semester



Linear Transforms

- **Overview: Transforms and algorithms**
- **Discrete Fourier transform**
- **Fast Fourier transforms**
- **After that:**
  - Optimized implementation and autotuning (FFTW)
  - Automatic program synthesis (Spiral)

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

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

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

Output *Input*

- Equivalent definition: Summation form

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

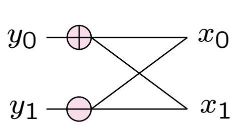
5

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

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

universal tool (DFT, RDFT)

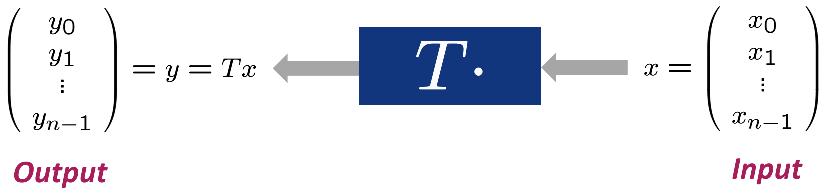
MPEG (IMDCT)

JPEG (DCT-2)

Blackboard

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

Linear Transforms: DFT



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

Algorithms: Example FFT, n = 4

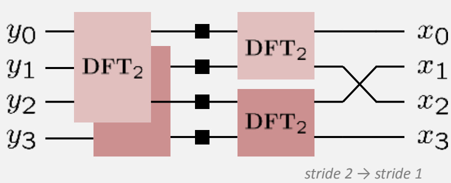
Fast Fourier transform (FFT)

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

Representation using matrix algebra

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

Data flow graph

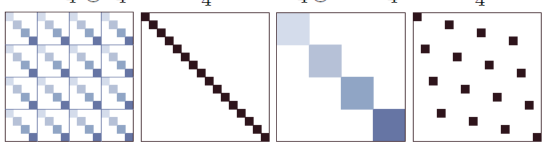


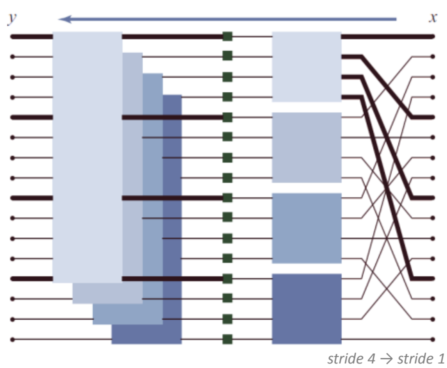
Cooley-Tukey FFT (Recursive, General-Radix)

- Blackboard
- Kronecker products
- Stride permutations

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

$$\mathrm{DFT}_{16} = (\mathrm{DFT}_4 \otimes I_4)\; T_4^{16}\; (I_4 \otimes \mathrm{DFT}_4)\; L_4^{16}$$





Recursive Cooley-Tukey FFT

radix

$$\mathbf{DFT}_{km} = (\mathbf{DFT}_k \quad \mathrm{I}_m) T_m^{km} (\mathrm{I}_k \quad \mathbf{DFT}_m) L_k^{km}$$ *decimation-in-time*

$$\mathbf{DFT}_{km} = L_m^{km} (\mathrm{I}_k \quad \mathbf{DFT}_m) T_m^{km} (\mathbf{DFT}_k \quad \mathrm{I}_m)$$ *decimation-in-frequency*

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

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

- **Cost:**
  - (complex adds, complex mults) = (n $\log_2$(n), n $\log_2$(n)/2)
    independent of recursion
  - (real adds, real mults) ≤ (2n $\log_2$(n), 3n $\log_2$(n)) = 5n $\log_2$(n) flops
    depends on recursion: best is at least radix-8

Recursive vs. Iterative FFT

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

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

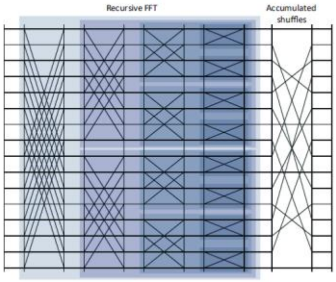
$$\mathbf{DFT}_{km} = L_m^{km} (\mathrm{I}_k \quad \mathbf{DFT}_m) T_m^{km} (\mathbf{DFT}_k \quad \mathrm{I}_m)$$

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

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

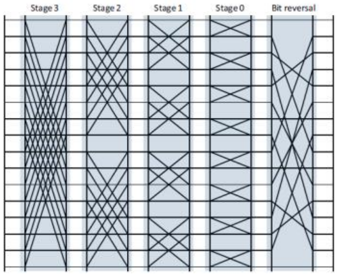
$$\mathbf{DFT}_{2^t} = R_{2^t} \cdot \left(\prod_{j=1}^{t} (\mathrm{I}_{2^{t-j}} \quad T_{2^{j-1}}^{2^j}) \cdot (\mathrm{I}_{2^{t-j}} \quad \mathbf{DFT}_2 \quad \mathrm{I}_{2^{j-1}}) \right)$$

Radix 2, recursive



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

Radix 2, iterative



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

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**

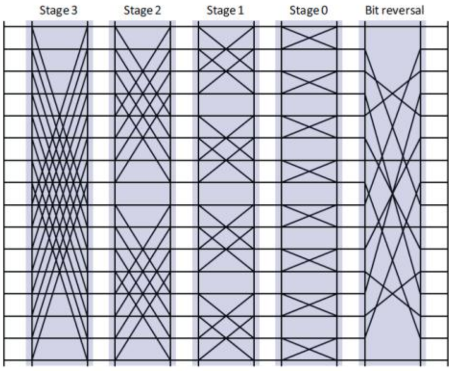
- **Rough analogy:**

MMM	DFT
Triple loop	Iterative FFT
Blocked	Recursive FFT

The FFT Is Very Malleable

17

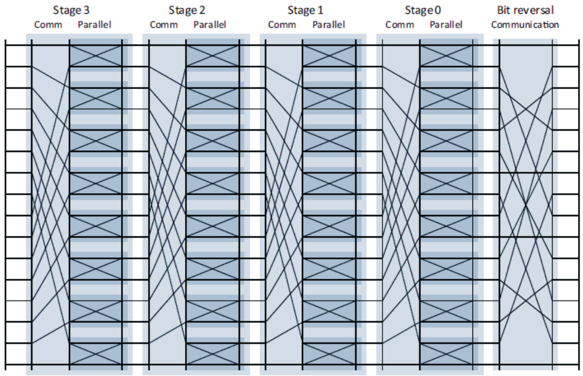
Iterative FFT, Radix 2



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

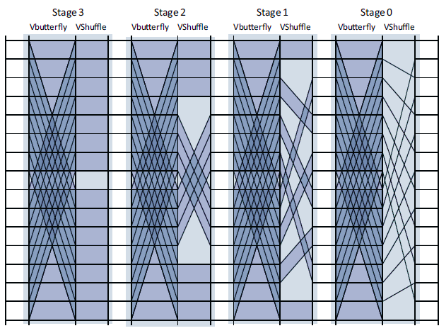
18

Pease FFT, Radix 2



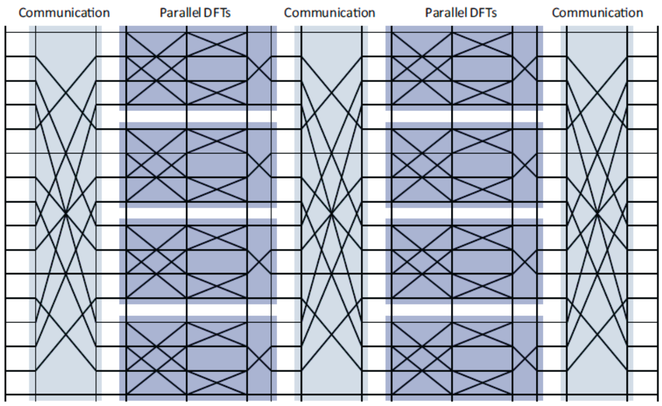
$$\left(L_2^{16}(I_8\otimes \mathrm{DFT}_2)D_0^{16}\right)\left(L_2^{16}(I_8\otimes\mathrm{DFT}_2)D_1^{16}\right)\left(L_2^{16}(I_8\otimes\mathrm{DFT}_2)D_2^{16}\right)\left(L_2^{16}(I_8\otimes\mathrm{DFT}_2)D_3^{16}\right)R_2^{16}$$

Stockham FFT, Radix 2



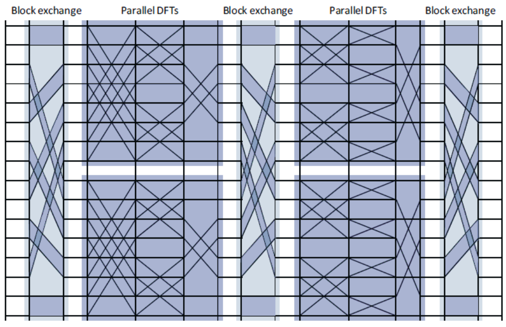
$$\left((\mathrm{DFT}_2\otimes I_8)D_0^{16}(L_2^2\otimes I_8)\right)\left((\mathrm{DFT}_2\otimes I_8)D_1^{16}(L_2^4\otimes I_4)\right)\left((\mathrm{DFT}_2\otimes I_8)D_2^{16}(L_2^8\otimes I_2)\right)\left((\mathrm{DFT}_2\otimes I_8)D_3^{16}(L_2^{16}\otimes I_1)\right)$$

Six-Step FFT



$$L^{16}_4 \left(I_4 \otimes ((\mathrm{DFT}_2 \otimes I_2) T^4_2 (I_2 \otimes \mathrm{DFT}_2) L^4_2)\right) L^{16}_4 T^{16}_4 \left(I_4 \otimes ((\mathrm{DFT}_2 \otimes I_2) T^4_2 (I_2 \otimes \mathrm{DFT}_2) L^4_2)\right) L^{16}_4$$

Multi-Core FFT



$$(L^8_4 \otimes I_2) \left(I_2 \otimes \left((\mathrm{DFT}_2 \otimes I_2) T^4_2 (I_2 \otimes \mathrm{DFT}_2) L^4_2\right) \otimes I_2\right) (L^8_2 \otimes I_2)\, T^{16}_4 \left(I_2 \otimes \left(I_2 \otimes (\mathrm{DFT}_2 \otimes I_2) T^4_2 (I_2 \otimes \mathrm{DFT}_2)\right) R^8_2\right) (L^8_2 \otimes I_2)$$

Transform Algorithms

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

$$\begin{vmatrix}\mathbf{RDFT}_n\\ \mathbf{RDFT}'_n\\ \mathbf{DHT}_n\\ \mathbf{DHT}'_n\end{vmatrix} \rightarrow (P^{\top}_{k/2,m}\quad I_2)\left(\begin{vmatrix}\mathbf{RDFT}_{2m}\\ \mathbf{RDFT}'_{2m}\\ \mathbf{DHT}_{2m}\\ \mathbf{DHT}'_{2m}\end{vmatrix} \oplus \left(I_{k/2-1}\quad {}_i\, D_{2m}\begin{vmatrix}\mathbf{rDFT}_{2m}(i/k)\\ \mathbf{rDFT}_{2m}(i/k)\\ \mathbf{rDHT}_{2m}(i/k)\\ \mathbf{rDHT}_{2m}(i/k)\end{vmatrix}\right)\right)\left(\begin{vmatrix}\mathbf{RDFT}'_k\\ \mathbf{RDFT}'_k\\ \mathbf{DHT}'_k\\ \mathbf{DHT}'_k\end{vmatrix}\quad I_m\right), \quad k \text{ even},$$

$$\begin{vmatrix}\mathbf{rDFT}_{2n}(u)\\ \mathbf{rDHT}_{2n}(u)\end{vmatrix} \rightarrow L^{2n}_m\left(I_k\quad {}_i\begin{vmatrix}\mathbf{rDFT}_{2m}((i+u)/k)\\ \mathbf{rDHT}_{2m}((i+u)/k)\end{vmatrix}\right)\left(\begin{vmatrix}\mathbf{rDFT}_{2k}(u)\\ \mathbf{rDHT}_{2k}(u)\end{vmatrix}\quad I_m\right),$$

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

$$\mathbf{DCT\text{-}2}_n \rightarrow P^{\top}_{k/2,2m}\left(\mathbf{DCT\text{-}2}_{2m}\,K^{2m}_2 \oplus \left(I_{k/2-1}\quad N_{2m}\,\mathbf{RDFT\text{-}3}^{\top}_{2m}\right)\right)B_n(L^{n/2}_{k/2}\quad I_2)(I_m\quad \mathbf{RDFT}'_k)Q_{m/2,k},$$

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

$$\mathbf{DCT\text{-}4}_n \rightarrow Q^{\top}_{k/2,2m}\left(I_{k/2}\quad N_{2m}\,\mathbf{RDFT\text{-}3}^{\top}_{2m}\right)B'_n(L^{n/2}_{k/2}\quad I_2)(I_m\quad \mathbf{RDFT\text{-}3}_k)Q_{m/2,k}.$$

$$\mathbf{DFT}_n \rightarrow (\mathbf{DFT}_k\quad \mathrm{I}_m)\,\mathrm{T}^n_m(\mathrm{I}_k\quad \mathbf{DFT}_m)\,\mathrm{L}^n_k, \quad n = km$$ — *Cooley-Tukey FFT*

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

$$\mathbf{DFT}_p \rightarrow R^T_p(\mathrm{I}_1 \oplus \mathbf{DFT}_{p-1})D_p(\mathrm{I}_1 \oplus \mathbf{DFT}_{p-1})R_p, \quad p \text{ prime}$$ — *Rader FFT*

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

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

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

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

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

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

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

Complexity of the DFT

- **Measure: L_c, $2 \le c$**
  - 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:**
  - $n = 2^k$: $L_2(DFT_n) \le 3/2\ n \log_2(n)$ *(using Cooley-Tukey FFT)*
  - General n: $L_2(DFT_n) \le 8\ n \log_2(n)$ *(needs Bluestein FFT)*
- **Lower bound:**
  - Theorem by Morgenstern: If $c < \infty$, then $L_c(DFT_n) \ge ½\ n \log_c(n)$
  - 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, …