How to Write Fast Numerical Code

Spring 2011
Lecture 21

Instructor: Markus Püschel

TA: Georg Ofenbeck

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Schedule

May 2011

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
24	25	26	27	28	29	30
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

Today

Lecture

Project presentations

- 10 minutes each
- random order
- random speaker

Final project paper and code due:

Friday, June 10th

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*
- **FFTW: www.fftw.org**
  - *Frigo and Johnson,* ***FFTW: An Adaptive Software Architecture for the FFT****, Proc. ICASSP, vol. 3, pp. 1381-1384*
  - *M. Frigo,* ***A fast Fourier transform compiler****, in Proc. PLDI, 1999*

Discrete Fourier Transform

- **Defined for all sizes n:**

$$y = \mathbf{DFT}_n\, x$$

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

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 $L_c(DFT_n) \geq ½\ n \log_c(n)$
  - 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, …

Example FFT, n = 4

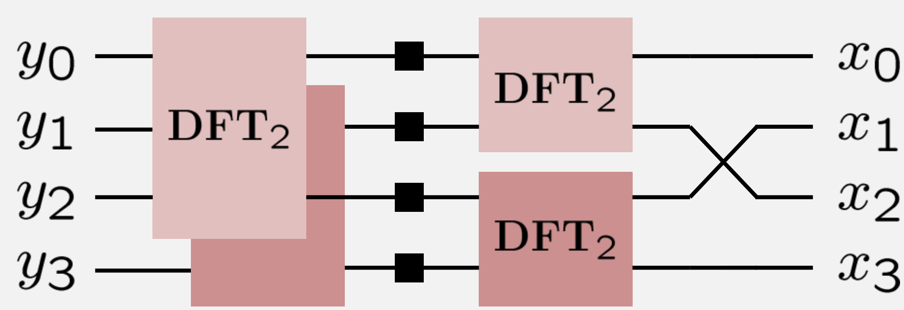
Fast Fourier transform (FFT)

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

Representation using matrix algebra

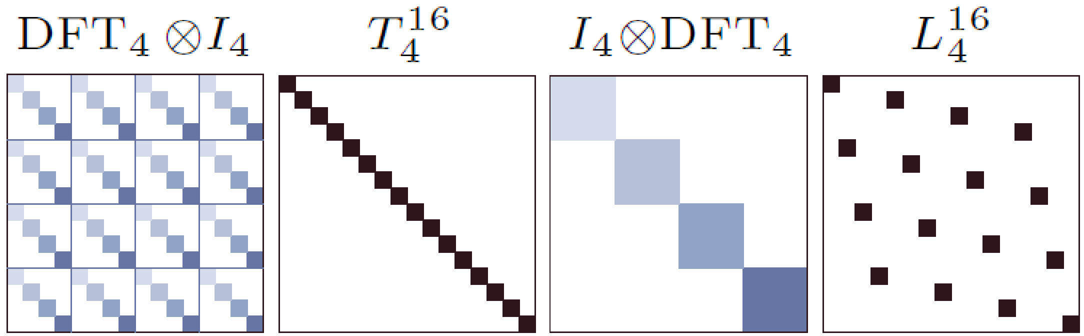
$$\mathbf{DFT}_4 = (\mathbf{DFT}_2 \otimes \mathrm{I}_2)\,\mathrm{diag}(1,1,1,i)(\mathrm{I}_2 \otimes \mathbf{DFT}_2)\,\mathrm{L}^4_2$$

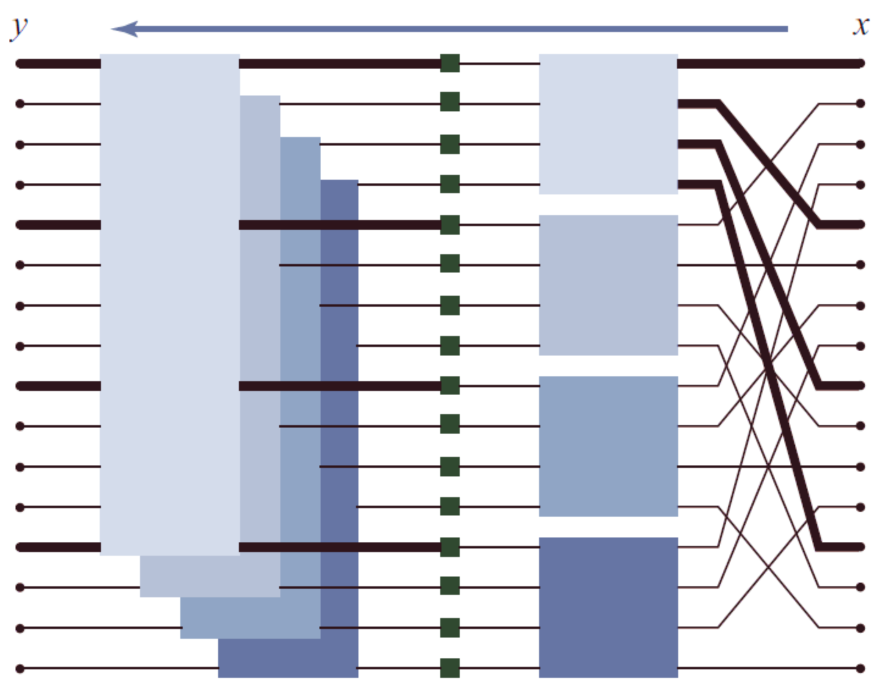
Data flow graph



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

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





FFTs

- **Recursive, general radix, decimation-in-time/decimation-in-frequency**
  radix = k

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

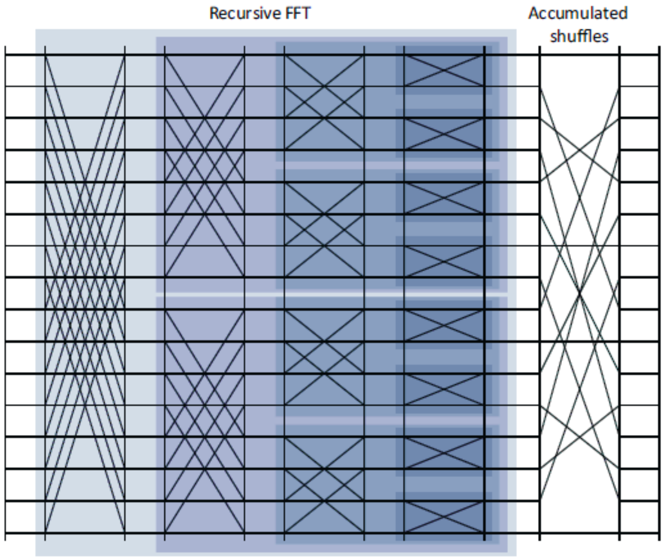
$$\mathbf{DFT}_{km} = L_m^{km} (\mathrm{I}_k \quad \mathbf{DFT}_m) T_m^{km} (\mathbf{DFT}_k \quad \mathrm{I}_m)$$

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

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

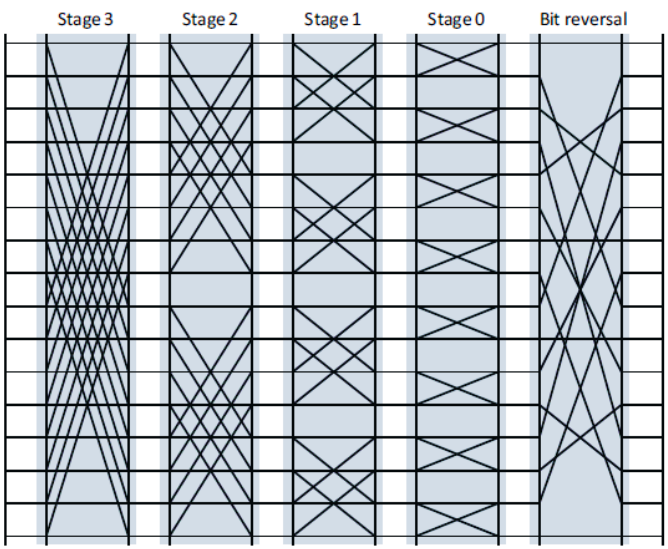
$$\mathbf{DFT}_{2^t} = R_{2^t} \cdot \left(\prod_{j=1}^{t} (\mathrm{I}_{2^{t-j}} \quad T_{2^{j-1}}^{2^j}) \cdot (\mathrm{I}_{2^{t-j}} \quad \mathbf{DFT}_2 \quad \mathrm{I}_{2^{j-1}}) \right)$$

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

Fast Implementation (≈ FFTW 2.x)

- Choice of algorithm
- Locality optimization
- Constants
- Fast basic blocks
- Adaptivity

- Blackboard