How to Write Fast Numerical Code

Spring 2012
Lecture 18

Instructor: Markus Püschel

TAs: Georg Ofenbeck & Daniele Spampinato

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Planning

May 2012

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
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	5	6	7	8	9

Today

Lecture

Project meetings

Project presentations

- 10 minutes each
- random order
- random speaker

Reports due ~7-10 days after semester end

Linear Transforms

- Overview: Transforms and algorithms
- Discrete Fourier transform
- Fast Fourier transforms
- 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*
- **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*

Linear Transforms

- **Very important class of functions: signal processing, scientific computing, …**
- ***Mathematically:*** **Change of basis = Multiplication by a fixed matrix T**

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

T.

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

Output ***Input***

- **Equivalent definition: Summation form**

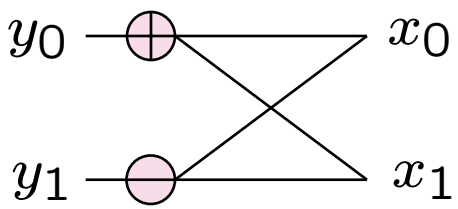
$$y_k = \sum_{\ell=0}^{n-1} t_{k,\ell} x_\ell, \qquad 0 \le k < n$$

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

Transforms: Examples

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

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

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

universal tool

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

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

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

MPEG

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

JPEG

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

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

Blackboard

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

Linear Transforms

Output ***Input***

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

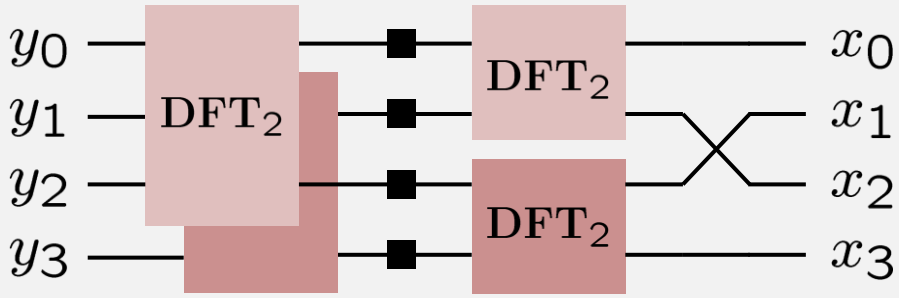
Fast Fourier transform (FFT)

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

Representation using matrix algebra

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

Data flow graph



Structured Matrices

- Useful for representing transform algorithms
- Cooley-Tukey FFT
- Blackboard