How to Write Fast Numerical Code
Spring 2011
Lecture 21

Instructor: Markus Püschel
TA: Georg Ofenbeck
Final project paper and code due:

*Friday, June 10th*
FFT References


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

- **FFTs:**
  - van Loan, *Computational Frameworks for the Fast Fourier Transform*, SIAM, 1992

- **FFTW:** [www.fftw.org](http://www.fftw.org)
Discrete Fourier Transform

- Defined for all sizes $n$:

$$y = \text{DFT}_n x$$

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

Carl-Friedrich Gauss

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)**

\[
\begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & i & -1 & -i \\
1 & -1 & 1 & -1 \\
1 & -i & -1 & i \\
\end{bmatrix}
\begin{bmatrix}
x_0 \\
x_1 \\
x_2 \\
x_3 \\
\end{bmatrix}
= 
\begin{bmatrix}
1 & 1 & 1 & 1 \\
. & 1 & . & 1 \\
. & 1 & -1 & . \\
. & 1 & . & -1 \\
\end{bmatrix}
\begin{bmatrix}
1 & 1 & 1 & 1 \\
. & 1 & . & 1 \\
. & 1 & 1 & . \\
. & 1 & . & 1 \\
\end{bmatrix}
\begin{bmatrix}
x_0 \\
x_1 \\
x_2 \\
x_3 \\
\end{bmatrix}
\]

**Representation using matrix algebra**

\[
\text{DFT}_4 = (\text{DFT}_2 \otimes I_2) \, \text{diag}(1, 1, 1, i) \, (I_2 \otimes \text{DFT}_2) \, L_2^4
\]

**Data flow graph**
Example FFT, $n = 16$ (Recursive, Radix 4)

\[
\text{DFT}_{16} = \text{DFT}_4 \otimes I_4 \quad T_4^{16} \quad I_4 \otimes \text{DFT}_4 \quad L_4^{16}
\]
## FFTs

- **Recursive, general radix, decimation-in-time/decimation-in-frequency**
  
  \[
  DFT_{km} = (DFT_k \cdot I_m)T_m^{km}(I_k \cdot DFT_m)L_m^{km}
  \]

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

  \[
  DFT_{2^{t}} = \left(\prod_{j=1}^{t} (I_{2^{t-j}} - DFT_2 \cdot I_{2^{t-j}}) \cdot (I_{2^{t-j}} - T_{2^{t-j}}^{2^{t-j}+1})\right) \cdot R_{2^{t}}
  \]

  \[
  DFT_{2^{t}} = R_{2^{t}} \cdot \left(\prod_{j=1}^{t} (I_{2^{t-j}} - T_{2^{j-1}}^{2^{2j}}) \cdot (I_{2^{t-j}} - DFT_2 \cdot I_{2^{t-j}})\right)
  \]
Radix 2, recursive

Radix 2, iterative
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:

<table>
<thead>
<tr>
<th>MMM</th>
<th>DFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triple loop</td>
<td>Iterative FFT</td>
</tr>
<tr>
<td>Blocked</td>
<td>Recursive FFT</td>
</tr>
</tbody>
</table>
Fast Implementation (≈ FFTW 2.x)

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