Requiem for Moore’s Law
Moore’s Law

- Every 18 months, the speed of your computer is doubled
- Every 18 months, the memory on your computer is doubled
- At the same time, the cost of your computer goes down - not quite exponentially: the box does not become much cheaper!
- A good number to look at

\[ R_{1970} = \frac{\text{Cost of CPU time}}{\text{Cost of human time}} \]

- 1970 is the year
- Different CPUs, different humans, etc.
Observation

- $R_{1945} \gg 1000$
- $R_{1960} \gg 100$
- $R_{1970} \gg 10$
- $R_{1980} \sim 1$
- $R_{2000} \ll 0.01$
- $R_{2015} \ll 10^{-5}$ - a qualitative change

Unlike men, not all CPUs are created equal!

Most CPUs do not vote - but this seems to be changing
Consequences

- Cell phones and E-mail
- Internet
- Drones: military, civilian, research
- Experimental science
- Manufacturing
- Everything else: Sandy as an example
- Properly interpreted, Moore’s law simplified the life of an applied mathematician (or a numerical analyst)
Consider a numerical task: solving a PDE, evaluating fields generated by a collection of $N$ charges, etc.

The cost is normally represented in the form

$$T \sim \alpha N^\beta,$$

or perhaps

$$T \sim \alpha N^\beta \log(N)$$

In an environment dominated by Moore’s law, it often makes sense to (more or less) ignore $\alpha$ and concentrate on $\beta$ - especially for a young researcher.
To a large extent, the choice of the algorithm is determined by the power of one’s computer (!!)
Aim at the near (or not so near) future, as opposed to the immediate state of the computers: obvious submarine analogy

- Sometimes, ships zigzag - and so do problems (and humans!)
■ Obvious exceptions when the needs are relatively immediate - 
sometimes a problem for people working in industry

■ Especially important for younger people

■ Easy life for some of us

■ What do we do now that Moore’s law is limping on both feet  
  - and might be terminally ill?

■ The model: problems from physics, engineering, etc.
Some History:

- Sorting algorithms - prehistory of computing
- Given a set of elements $x_1, x_2, x_3, \ldots$, sort them with respect to some comparison function
- Naive sorts cost $O(n^2)$ operations
- Mergesort: Von Neumann in 1946 - birth of Moore's law (?)
- Quicksort in 1960 by Tony Hoare (for automatic translation!)
- Heapsort in 1964 by J.W.J. Williams and R.W. Floyd
- Algorithms from Computer Science: fast, exact in exact arithmetic, fragile
FFT - the granddaddy of them all

\[ f_k = \sum_{j=1}^{n} e^{i j \cdot k \cdot h} \cdot \varphi_j \]

- Discrete Fourier Transform
- Order $n \log(n)$ operations vs. order $n^2$
- Purely algebraic, “fragile”, small constant
- Cooley and Tukey circa 1965, Gauss circa 1865, Fichtengoltz in 1948 - up to order 72 - attributed to Runge 1903
FFT - Immediate Consequences

- “Fast” Convolution
- Separation of variables: Fast Poisson Solver, etc. - based on Fast Convolution
- “Fast” algorithms over finite fields
- Major influences: numerics, signal processing, etc.
- Early effects of Moore’s law?
Basic Tools for The Brave New World

- Hashing, Graphs, Searches, Trees, Heaps, Strings...
- Housekeeping within the computer
- Lots of competent work, careful asymptotic CPU time estimates: implicitly driven by Moore’s law
- Coupled with hardware, started transforming our lives
- Almost purely “algebra-based” - very few real numbers (!)
- Transformed scientific computing, applied analysis, etc.
- Only in a generic sense
Concept of a “Fast” Algorithm

- Must be asymptotically faster than the obvious one:
  - Order $<< n^3$ operations for solving a dense linear system
  - Applying a dense linear integral operator faster than in order $n^2$ operations
  - $n$-body problem in under $n^2$ operations
- No cheating: finite precision, finite memory, etc.
- Interaction of “fast” algorithms with Moore’s law
The very analytical techniques used depend on the presence or absence of a “fast” method.

Example: the dominant eigenvector of a large matrix: a “fast” algorithm (or Moore’s law, or both) can obviate the analysis.

The concept of “brute force” is redefined.

The fallacy of “engineering accuracy”

Greengard vs. Hamming

Convergence rates matter - integral vs. differential equations
Sufficiently rapid convergence tends to obviate the difference between “approximate” and “exact” algorithms.

- Constants matter.
- Convergence rates matter even more.
- A recent example: Laplace and Helmholtz equations on polygons.
- Lots of low-hanging fruit.
Wirth’s Law: Monster in the Attic

- "Software is getting slower more rapidly than hardware becomes faster", Wirth, 1995
- “The speed of software halves every 18 months” - Gates’ Law
- Extremely susceptible: Consumer software
- Scientific software has been more resistant, but...
- Good infrastructure is good - bad infrastructure is an abomination
A strange thing happened in the early 80’s:

- $n$-body solvers: evaluate the potential (field, etc.) created by a collection of heavy particles on the whole ensemble
Appel, Barnes-Hut, G-R, Rundensteiner & Van Dommelen - not an exhaustive list

All more or less simultaneously

Very elementary underlying idea

Newton knew it in 1666 (!!!)

Can be viewed as a trick - or a class of algorithms and a new paradigm
We do NOT evaluate the gravitational field of the Moon on Earth by adding up the fields one atom at a time.

The rank of interaction is small - to (very!) high accuracy.

The field generated by the center of mass approximation, perhaps dipole moment, quadrupole, etc.
A Simple Example

\[ V(Q_i) = \sum_{j=1}^{N} \frac{q_j}{\|Q_i - P_j\|} \]

Direct evaluation requires \(O(NM)\) work.
Multipole Expansion

\[ V(Q) = V(r, \theta, \phi) \approx \sum_{n=0}^{p} \sum_{m=-n}^{n} \frac{M_n^m Y_n^m(\theta, \phi)}{r^{n+1}}, \]

- The error in the multipole approximation decays like 
  \[ (R/|Q|)^{p+1}. \]

- For our simple example, \( R/|Q| < 1/2 \), so that setting 
  \[ p = \log_2\left(\frac{1}{\epsilon}\right) \] 
  yields a precision of \( \epsilon \).
Using multipole expansions

- Evaluate multipole coefficients $M_n^m$ for $n = 0, \ldots, p$
- Evaluate expansion at target points $Q_j$, for $j = 1, \ldots, M$
- Total operation count: $p^2 \cdot (N + M) = (N + M) \cdot \log^2(\frac{1}{\varepsilon})$
The General Case

- “Sources” and “targets” are squished together - often the same points
A Rudimentary “Fast” Scheme: Step 1

- Blank boxes are “separated” from the crossed one, and can be accounted for via multipole expansions
  - The same for each of the 16 boxes
Rudimentary Scheme: Step 2

- Light grey boxes have been accounted for on the preceding step; dark grey boxes are NOT separated from box $X$; blank boxes can be accounted for via multipole expansions

- Same for each of 64 boxes
Multipole expansions are used to account for all blank boxes, for all $4^M$ boxes $X$.

Cost: $O(N)$
Rudimentary Scheme: Final Step

- $O(1)$ points per box on the finest level
- Cost: $O(N)$
Rudimentary Scheme: Adaptive Version

- Quad-tree structure
- Total cost: $O(N \log(N))$, whatever the distribution
- Good clean fun
Two options

- Declare victory: asymptotically, this IS a “fast” scheme
- Run actual numerical tests: it is VERY slow - especially in three dimensions, and given high accuracy requirements
- At 6 digits, the number of spherical harmonics in three dimensions is of the order 1000
- “Engineering accuracy” - pontificate
- And the number of boxes in each interaction list, and \( \log(N) \)...
- Something needs to be done, starting with the \( \log(N) \), and the expansions, and the geometry...
The log($N$) Factor

- The reason for the log($N$)
- An obvious remedy: “translation operators”
- Every charge is accessed once - elaborate
Other Translations

- Far-field to local expansion

![Diagram of far-field to local expansion]

- Local to local expansion

![Diagram of local to local expansion]
The Whole Scheme

- Construct far-field expansions for all childless boxes
- Merge the expansions bottom to top
- On each level, convert far-field expansions into local ones for all elements in the interaction list for each box
- Starting at the top, split the local expansion for each box and add the obtained expansions to those of the children
- At the finest level, evaluate the local expansion for each box at all nodes in that box
- Evaluate directly all “local” interactions
The Whole Scheme II

- Total CPU time adds to $O(N)$ - problem solved?
- In two dimensions sufficiently fast as is
- In three dimensions, too slow above three digits or so
- Diagonal forms of translation operators, playing games with interaction lists, etc.
- The final scheme ends up using separation of variables in the spherical, cylindrical, and rectangular coordinates
- A lot of fuss for a finite factor in speed!
Several Examples

- Random distribution inside a cube, 3-digit accuracy
- Times in seconds on a 3-GHz LENOVO laptop

<table>
<thead>
<tr>
<th>$N$</th>
<th>Levels</th>
<th>$T_{fmm}$</th>
<th>$T_{dir}$</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>20000</td>
<td>4</td>
<td>0.74</td>
<td>12.9</td>
<td>$7.9 \cdot 10^{-4}$</td>
</tr>
<tr>
<td>50000</td>
<td>4</td>
<td>1.36</td>
<td>81.5</td>
<td>$5.2 \cdot 10^{-4}$</td>
</tr>
<tr>
<td>200000</td>
<td>5</td>
<td>8.66</td>
<td>1343</td>
<td>$8.4 \cdot 10^{-4}$</td>
</tr>
<tr>
<td>500000</td>
<td>5</td>
<td>14.6</td>
<td>8655</td>
<td>$7.0 \cdot 10^{-4}$</td>
</tr>
<tr>
<td>1000000</td>
<td>6</td>
<td>31.8</td>
<td>34863</td>
<td>$7.1 \cdot 10^{-4}$</td>
</tr>
</tbody>
</table>
Random distribution inside a cube, 9-digit accuracy

<table>
<thead>
<tr>
<th>$N$</th>
<th>Levels</th>
<th>$T_{fmm}$</th>
<th>$T_{dir}$</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>20000</td>
<td>3</td>
<td>2.08</td>
<td>12.9</td>
<td>$2.8 \cdot 10^{-10}$</td>
</tr>
<tr>
<td>50000</td>
<td>3</td>
<td>5.21</td>
<td>81.5</td>
<td>$1.6 \cdot 10^{-10}$</td>
</tr>
<tr>
<td>200000</td>
<td>4</td>
<td>20.2</td>
<td>1343</td>
<td>$1.6 \cdot 10^{-10}$</td>
</tr>
<tr>
<td>500000</td>
<td>4</td>
<td>63.7</td>
<td>8655</td>
<td>$1.2 \cdot 10^{-10}$</td>
</tr>
<tr>
<td>1000000</td>
<td>4</td>
<td>129</td>
<td>34863</td>
<td>$1.2 \cdot 10^{-10}$</td>
</tr>
</tbody>
</table>
Examples III

- Distribution on a complicated surface, 9-digit accuracy

<table>
<thead>
<tr>
<th>$N$</th>
<th>Levels</th>
<th>$T_{fmm}$</th>
<th>$T_{dir}$</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>20880</td>
<td>6</td>
<td>2.51</td>
<td>16.7</td>
<td>$3.6 \cdot 10^{-12}$</td>
</tr>
<tr>
<td>51900</td>
<td>7</td>
<td>5.46</td>
<td>120</td>
<td>$1.1 \cdot 10^{-10}$</td>
</tr>
<tr>
<td>20320</td>
<td>8</td>
<td>17.6</td>
<td>1696</td>
<td>$6.5 \cdot 10^{-12}$</td>
</tr>
<tr>
<td>503775</td>
<td>9</td>
<td>46.5</td>
<td>10712</td>
<td>$1.0 \cdot 10^{-11}$</td>
</tr>
</tbody>
</table>
Observations

- The final (relatively fast) algorithm involves some technical work: both book-keeping and analysis.
- Most *non-oscillatory* kernels encountered: easy in 1-D, a little work in two, quite a mess in three - but it works.
- *Low-frequency* electromagnetics - computer chips; hydrodynamics at low Reynolds numbers; “fast” Gauss transform; integral formulation of parabolic problems...
- Purely numerical applications: Hilbert matrices, interpolation, “Fast” Laplace Transform, “fast” Legendre transform, etc.
Observations II

- In the end, a combination of classical expansions and “Computer Science”
- Large submatrices of the matrix involved are of low rank
- A purely numerical version of the scheme: use SVD (or, more efficiently, Interpolative Decomposition) to compress the submatrices: numerical translation operators
- “Kernel-independent” algorithms
- After all, in a physical system, long-range interactions tend to be low-rank - or do they?
Looking at the Moon in its optical field

The rank of interactions $\geq$ the number of features visible

“The ranks of submatrices are of the same order as their sizes”

Straightforward rank-based algorithms can not work

With a subtle exception: “butterfly” schemes
For all standard scattering environments (acoustic, Maxwell, elastic solids, etc.) all translation operators are diagonalized simultaneously by a “fast” unitary operator (!!)

A direct consequence of the Sommerfeld (radiation) condition

Resulting schemes are somewhat involved - elaborate

No “incantational” solution, even on paper - elaborate
- 50 wavelengths in size
- Smallest triangle: $1.06 \times 10^{-6} \lambda$
- Largest triangle: $2.86 \times 10^{-1} \lambda$
- Number of triangles: 706,300
- Single node per triangle
## Application of the Operator

### A-10 - Helmholtz

<table>
<thead>
<tr>
<th>T (dir.)</th>
<th>Acc.</th>
<th>Error (pot.)</th>
<th>Error (grad.)</th>
<th>T (sec.)</th>
<th>Mem. (Mb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>18234</td>
<td>$10^{-3}$</td>
<td>0.43E-3</td>
<td>0.56E-3</td>
<td>25.5</td>
<td>300</td>
</tr>
<tr>
<td>18234</td>
<td>$10^{-6}$</td>
<td>0.48E-6</td>
<td>0.50E-6</td>
<td>67.6</td>
<td>790</td>
</tr>
<tr>
<td>18234</td>
<td>$10^{-9}$</td>
<td>0.11E-9</td>
<td>0.95E-10</td>
<td>152.6</td>
<td>1143</td>
</tr>
</tbody>
</table>

### A-10 - Laplace

<table>
<thead>
<tr>
<th>T (dir.)</th>
<th>Acc.</th>
<th>Error (pot.)</th>
<th>Error (grad.)</th>
<th>T (sec.)</th>
<th>Mem. (Mb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3348</td>
<td>$10^{-3}$</td>
<td>0.27E-3</td>
<td>0.37E-4</td>
<td>2.55</td>
<td>211</td>
</tr>
<tr>
<td>3348</td>
<td>$10^{-6}$</td>
<td>0.19E-6</td>
<td>0.43E-7</td>
<td>6.36</td>
<td>292</td>
</tr>
<tr>
<td>3348</td>
<td>$10^{-9}$</td>
<td>0.85E-10</td>
<td>0.61E-11</td>
<td>12.8</td>
<td>376</td>
</tr>
</tbody>
</table>
- 50 wavelengths in size
- Smallest triangle: $9.34 \times 10^{-3} \lambda$
- Largest triangle: $3.27 \times 10^{-1} \lambda$
- Number of triangles: 872,694
- Single node per triangle
## Application of the Operator

### Horse - Helmholtz

<table>
<thead>
<tr>
<th>$T$ (dir.)</th>
<th>Acc.</th>
<th>Error (pot.)</th>
<th>Error (grad.)</th>
<th>$T$ (sec.)</th>
<th>Mem. (Mb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>36097</td>
<td>$10^{-3}$</td>
<td>0.65E-3</td>
<td>0.31E-3</td>
<td>36.5</td>
<td>549</td>
</tr>
<tr>
<td>36097</td>
<td>$10^{-6}$</td>
<td>0.66E-6</td>
<td>0.92E-7</td>
<td>98.2</td>
<td>1111</td>
</tr>
<tr>
<td>36097</td>
<td>$10^{-9}$</td>
<td>0.33E-9</td>
<td>0.33E-11</td>
<td>184.9</td>
<td>2027</td>
</tr>
</tbody>
</table>

### Horse - Laplace

<table>
<thead>
<tr>
<th>$T$ (dir.)</th>
<th>Acc.</th>
<th>Error (pot.)</th>
<th>Error (grad.)</th>
<th>$T$ (sec.)</th>
<th>Mem. (Mb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6044</td>
<td>$10^{-3}$</td>
<td>0.91E-3</td>
<td>0.57E-3</td>
<td>3.5</td>
<td>328</td>
</tr>
<tr>
<td>6044</td>
<td>$10^{-6}$</td>
<td>0.46E-6</td>
<td>0.31E-6</td>
<td>7.6</td>
<td>322</td>
</tr>
<tr>
<td>6044</td>
<td>$10^{-9}$</td>
<td>0.25E-9</td>
<td>0.10E-9</td>
<td>15.8</td>
<td>584</td>
</tr>
</tbody>
</table>
Direct Solvers

- “Fast” Application of operators to functions
- Often, we want to apply inverses of operators
- Iterative techniques - their own can of worms: condition numbers, resonances, etc.
- Direct solvers - elaborate
- Other “fast” operations of linear algebra
- Progress during the last decade: 1-D, 2-D, 3-D
- Rapidly evolving subject
Mundane desktop workstation

Each cavity is 5 wavelengths in diameter and is a body of revolution; a “nearly resonant” structure

Practically unsolvable by purely iterative techniques

Good clean fun

<table>
<thead>
<tr>
<th>$N$</th>
<th>$T_{solve}$</th>
<th>$E_{∞}^{rel}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>323 200</td>
<td>2.19e+04</td>
<td>3.813e-04</td>
</tr>
<tr>
<td>643 200</td>
<td>4.11e+04</td>
<td>8.145e-06</td>
</tr>
<tr>
<td>1 283 200</td>
<td>9.75e+04</td>
<td>6.884e-07</td>
</tr>
</tbody>
</table>
Fast Algorithms in Time Domain

- Time vs. frequency domain: separation of variables
- One 4-dimensional hyperbolic problem vs. a sequence of 3-dimensional ones
- Some problems are non-linear - no frequency-domain version
- Time-domain integral equations - well-known analytically; can be defined on boundaries only - but *for all times past*
- Computationally, too horrible to even look at - elaborate
- Eric Michielssen about 10 years ago
- A couple of movies is all I have time for
Most of these algorithms would make no sense in 1980

Just looking ahead, one could get an advantage of perhaps 10 years - without even trying!

And now, Moore’s law is dying - or at least comatose

While Gates’ Law is as vigorous as ever - and coming after us!
I have come to praise Moore’s law, not to bury it

Algorithms in current use are at least a decade behind the curve specified by Moore’s law

The “infrastructure” is farther behind

It would take at least a decade for us to catch up - if computers stopped evolving

No such thing: memory, multi-CPU systems, “clouds”...

A caution - especially to a young researcher

And perhaps the horse will learn to sing!