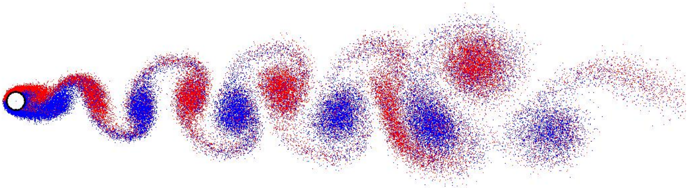
A new take at Adaptive Fast Multipole Methods: application, implementation, and hybrid CPU/GPU parallelism



Stefan Engblom

UPMARC @ TDB/IT, Uppsala University

KCSE seminar, Stockholm, November 6, 2013

Outline

- Background: design of vertical axis wind turbines
- Discretization using a vortex formulation
- Fast multipole methods...
  - ...with space adaptivity...
  - ...in parallel...
  - ...optimally on hybrid CPU/GPU-systems

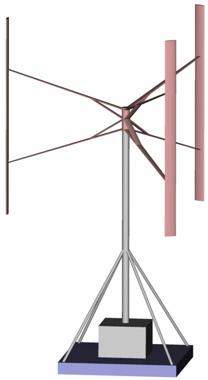
Joint work in part with **Paul Deglaire** and **Anders Goude** at the Division for Electricity and Lightning Research, and with **Marcus Holm** and **Sverker Holmgren** at the Division of Scientific Computing.

Background

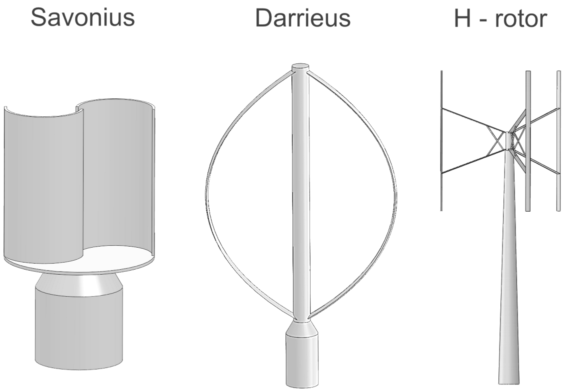
Pros/cons of VAWTs:

+ Generator at ground level
+ Less gravitational loads
+ No gears
+ Easier maintenance
+ Less noise
- Fatigue loads
- Start-up
- **Aerodynamics model**

YouTube: Vertical Wind 200kW (March 2010)



VAWTs



Vortex formulation

In 2D, let the velocity field $\mathbf{u}(z,t)$ solve the Navier-Stokes equations with BCs (identifying the complex number $z = x + iy$ with the space coordinate (x,y)). Introduce the *vorticity* $\omega \equiv \nabla \times \mathbf{u} \cdot \hat{k}$ and consider the two-step formulation:

$$\begin{aligned} \omega_t + \mathbf{u} \cdot \nabla\omega &= 0 && \text{(advection)}, \\ \omega_t &= \nu\Delta\omega && \text{(diffusion)}. \end{aligned}$$

-Hence; how do we obtain $\mathbf{u}$ from ω?

One can show that $\mathbf{u} = \mathbf{u}_\omega + \nabla\phi$ for some ϕ s.t. $\Delta\phi = 0$ accounting for the BCs. In turn,

$$\mathbf{u}_\omega(z,t) = \int_\Omega K(z - z')\omega(z', t)\, dz',$$

where $K = -i/(2\pi z)$ is the *Green's function* for $-\Delta$.
If the vorticity is discretized,

$$\omega(z,t) = \sum_j \delta(z - z_j)\Gamma_j,$$

with $z_j = z_j(t)$, then the velocity field is obtained from

$$\mathbf{u}_\omega(z,t) = \sum_j K(z - z_j)\Gamma_j.$$

To *advect*, evaluate the velocity field in all vorticity points z_j,

$$\mathbf{u}_\omega(z_j, t) = \sum_{i \neq j} K(z_j - z_i)\Gamma_i,$$

an N-body problem.

To *diffuse*, just add a normally distributed random number,

$$\mathbf{u}_\omega(z_j, t + \Delta t) = \mathbf{u}_\omega(z_j, t) + \sqrt{2\nu\Delta t}\mathcal{N}(0, 1).$$

In practice, there are also redistribution-type methods such that Γ_i is made time-dependent.

Fast multipole method

Main idea: all charges/potentials/bodies inside two well-separated sets can interact through an operator of low effective rank.

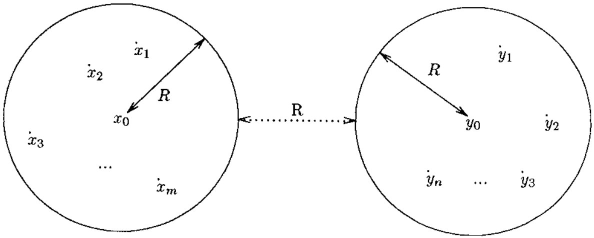


FIG. 1. Well-separated sets in the plane.

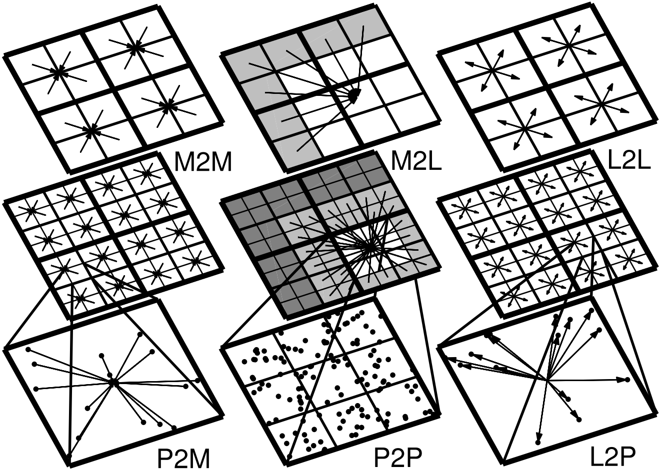
Figure: Found at p. 3 of Greengard and Rokhlin: "A Fast Algorithm for Particle Simulations" *J. Comput. Phys.* **73**(2):325–348 (1987).

Bottom-up, then top-down

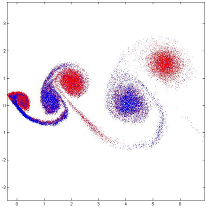
Distribute the points in a recursive tree of boxes where each box has 4 children (2D).

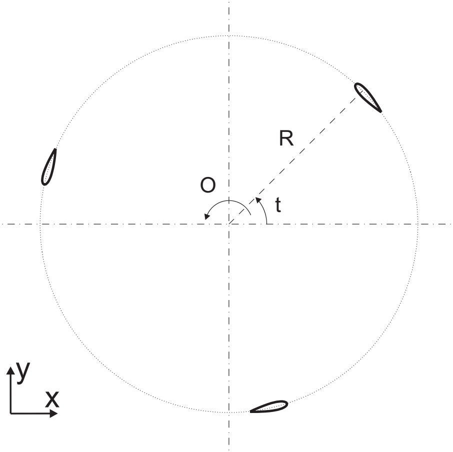
1. *Initialize* at the finest level in the tree, expanding each potential in a *multipole series* around the midpoint of the box.
2. *Go upwards* and *shift* all multipole expansions to parents, yielding a "top expansion" for the whole enclosing box.
3. *Go downwards* and, for all well-separated boxes, *shift-and-convert* all expansions into *local* expansions (eg. polynomials). Also, *shift* all such expansions to children, yielding a local field in each box.

Particles, Multipole, and Local...



Illustrations



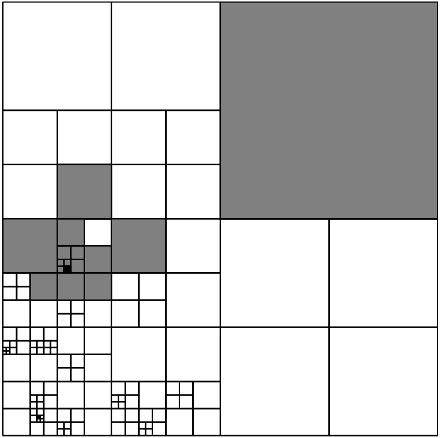


Test: flat plate.

Production runs: 3-bladed turbine, small turbine park.

Adaptivity

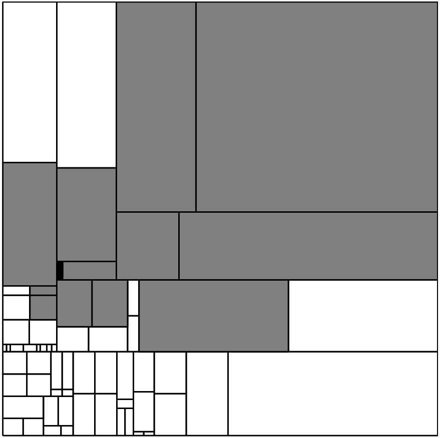
Want adaptivity, but *quite* complicated... The "C" in $\mathcal{O}(N)$ can be rather large.



(shaded boxes interact; different sizes means different levels in the multipole tree)

Asymmetric adaptivity/Balanced implementation

Idea: split around the median point instead of around the geometric midpoint. Easier to get the communication localized.



(all boxes shown here reside at the same level in the tree)

The θ-criterion

As the mesh looses regularity, it becomes important to keep track of what sets are really well-separated.

Criterion

Let the sets $S_1, S_2 \subset \mathbf{R}^D$ be contained inside two disjoint spheres such that $\|S_1 - x_0\| \le r_1$ and $\|S_2 - y_0\| \le r_2$. Given $\theta \in (0, 1)$, if $d \equiv \|x_0 - y_0\|$, $R \equiv \max\{r_1, r_2\}$, and $r \equiv \min\{r_1, r_2\}$, then the two sets are *well-separated* whenever $R + \theta r \le \theta d$.

In other words: any of the two sets may be expanded by a factor of $1/\theta$ and arbitrarily rotated about its center point without touching the other set.

Asymmetric adaptivity

Rules of Procedure

§1 A box is strongly connected to itself.

§2 By default, children to strongly connected boxes are also strongly connected.

§3 *However*, if two such children satisfy the θ-criterion, then they become weakly connected.

-At any level in the tree, weakly connected (= well-separated) boxes can interact through M2L-shifts. Strongly connected boxes, however, either have to interact on a more highly resolved level in the tree, or interact directly.

Thanks to a static data structure, these rules are highly implementable.

Asymmetric adaptivity

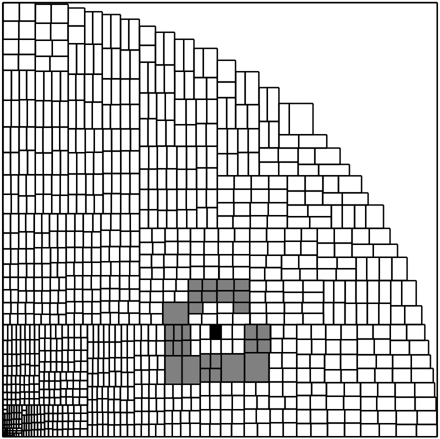


Figure: Typical M2L interaction list ($\theta = 1/2$).

Theory (1/2)

$$\Phi(x_i) = \sum_{j=1, j\neq i}^{N} G(x_i, x_j), \quad x_i \in \mathbf{R}^D, \quad i = 1 \ldots N.$$

(*Model:* the harmonic $1/r$ potential...)

Assumption (Kernel regularity)

For $\alpha, \beta \in \mathbf{Z}_+^D$ with $|\alpha|, |\beta| \leq p+1$,

$$|\partial_x^\alpha \partial_y^\beta G(x,y)| \leq C \frac{n!}{\|x-y\|^{n+1}}, \tag{1}$$

where $n \equiv |\alpha + \beta|$. Additionally, G is positive and satisfies

$$\|x-y\|^{-1} \leq cG(x,y). \tag{2}$$

Theory (2/2)

Assumption (Rotational invariance)

For any rotation T of the coordinate system,

$$G(x, y) = G(Tx, Ty). \tag{3}$$

$\Longrightarrow$ Then the relative error for the pth order adaptive fast multipole method under the θ-criterion is bounded by a constant $\times$ $\theta^{p+1}/(1-\theta)^2$.

Really accurate?

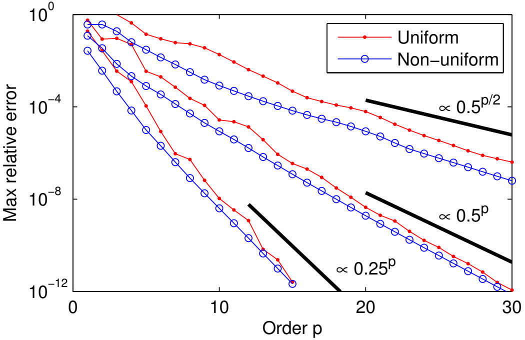


Figure: Errors for two different distribution of points and three distinct θs. *Note:* signed potential $1/(z_i - z_j)$.

Efficient way of handling adaptivity?

Theory: $\mathcal{O}\left(\theta^{-2}\log^{-2}\theta \cdot N\log^2 \mathrm{TOL}\right)$. ($\Longrightarrow \theta_{\mathrm{opt}} = \exp(-1) \approx 0.368$)

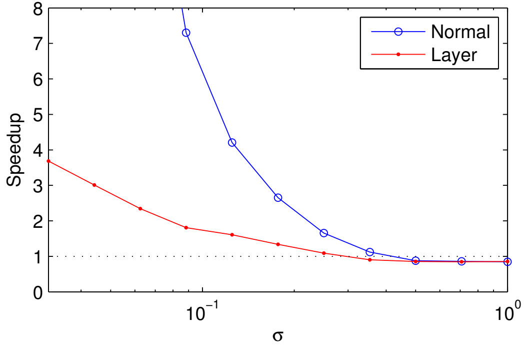
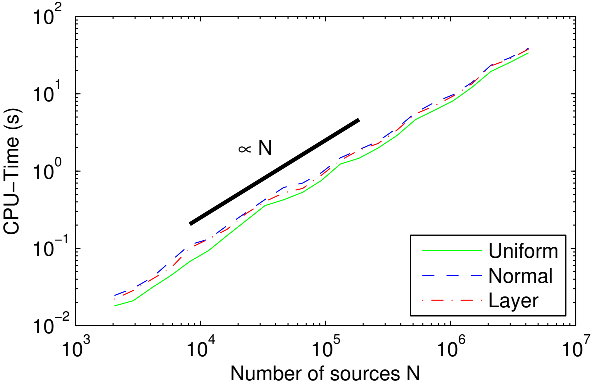


Figure: Adaptive vs. uniform FMM. Two different distribution of points.

"Normal" $:= \mathcal{N}(0,\sigma)$, but rejected to fit within the positive unit square.
"Layer" $:=$ the x-coordinate is $U[0,1]$ instead.

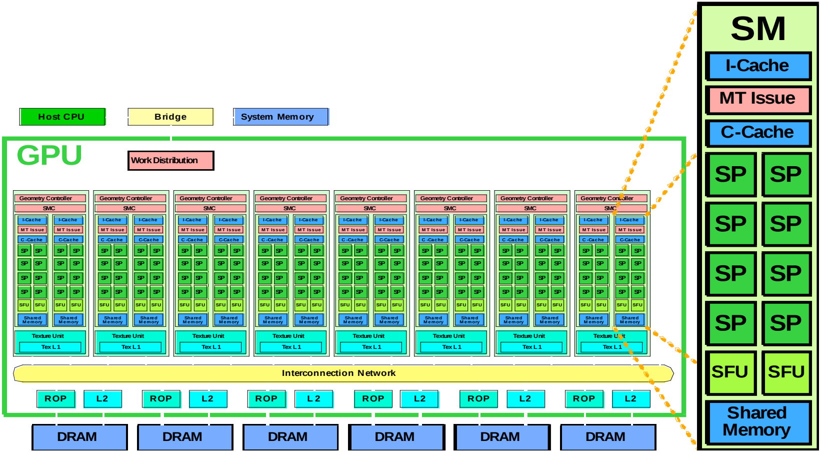
Scalable?



10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
CPU–Time (s)
$\propto$ N
Uniform
Normal
Layer
10^{3}
10^{4}
10^{5}
10^{6}
10^{7}
Number of sources N

Implementability?

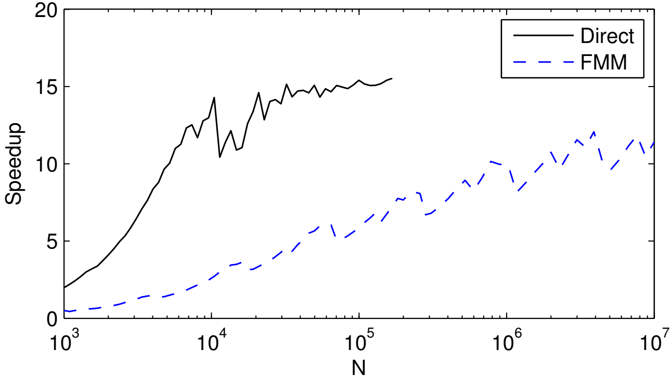
GPU-card



GPU implementation

Intel Xeon 6c W3680 @3.33GHz vs. Nvidia Tesla C2075 448c

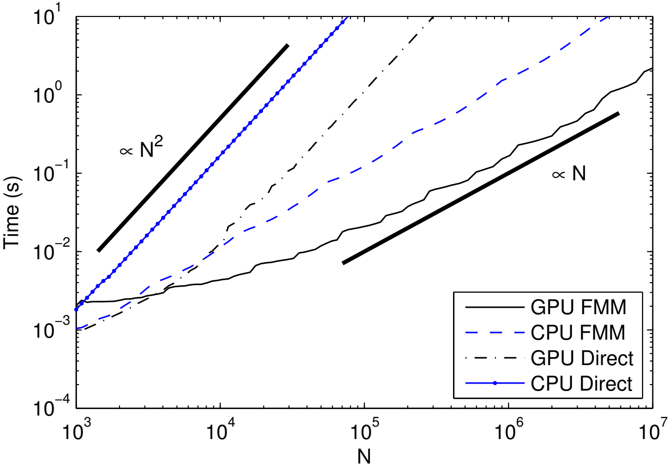
GPU speedup for (1) direct $O(N^2)$ all-pairs interaction, and (2) FMM.



Note: single threaded CPU.

GPU implementation

Intel Xeon 6c W3680 @3.33GHz vs. Nvidia Tesla C2075 448c



Note: single threaded CPU.

Lessons learnt from the GPU $\Longrightarrow$ Hybrid implementation

- Good speedup obtained across all operations in the FMM
- Coding complexity $\sim \times 4$ for topological parts

-For a 4+ core computer it seems reasonable that a hybrid approach would be beneficial. *For example:* offloading the local and perfectly data-parallel P2P evaluation to the GPU. Loadbalance?

Lessons learnt from the GPU $\Longrightarrow$ Hybrid implementation

- Good speedup obtained across all operations in the FMM
- Coding complexity $\sim \times 4$ for topological parts

-For a 4+ core computer it seems reasonable that a hybrid approach would be beneficial. *For example:* offloading the local and perfectly data-parallel P2P evaluation to the GPU. Loadbalance?

-*Idea:* the FMM is most often used in a dynamic/iterative context.
$\Longrightarrow$ Measure the performance and adjust the parameters of the algorithm for the next iteration.

Lessons learnt from the GPU $\Longrightarrow$ Hybrid implementation

- Good speedup obtained across all operations in the FMM
- Coding complexity $\sim \times 4$ for topological parts

-For a 4+ core computer it seems reasonable that a hybrid approach would be beneficial. *For example:* offloading the local and perfectly data-parallel P2P evaluation to the GPU. Loadbalance?

-*Idea:* the FMM is most often used in a dynamic/iterative context.
$\Longrightarrow$ Measure the performance and adjust the parameters of the algorithm for the next iteration.

In the present context: N_{levels} controls the amount of work left at the finest level (P2P), θ controls connectivity and the number of expansion coefficients. A regulating autotuner can be designed to wisely "crawl" this parameter space and continuously find the performance sweet spot.

Results

"Superglue" task-based programming model.

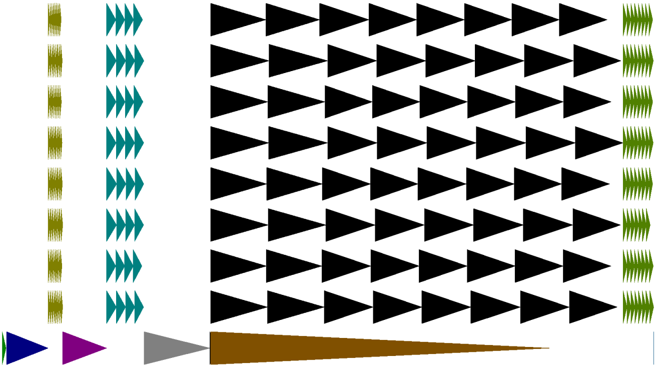
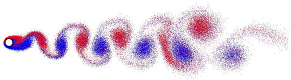
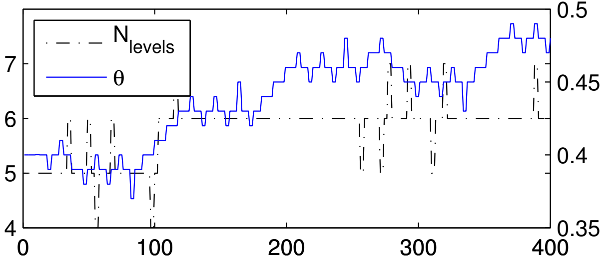


Figure: Superglue Execution Visualizator

Results

Note: 8c CPU vs. 8c CPU+448c GPU





	Vortex instability	Galaxy	Cylinder flow
CPU (s)	425	1094	1451
CPU+GPU (s)	379	324	367
Speedup	1.12	3.37	**3.95**

Conclusions

- Academic sw-project driven by a concrete problem.
- Nice aspect: the steady and controlled growth of performance and complexity:

Conclusions

- Academic sw-project driven by a concrete problem.
- Nice aspect: the steady and controlled growth of performance and complexity:
  1. Stand-alone *recursive* uniform FMM implementation ($N_{\text{max}} \sim 20,000$).
  2. Matlab-interface to a *direct* N-body evaluation ($N \sim 10,000$).
  3. First efficient working copy of uniform FMM; later heavily optimized, eg. BLAS L3 ($N \sim 1,000,000$, memory bound!).
  4. Added adaptivity – took the time to investigate a novel approach.
  5. Fully data-parallel GPU-implementation CUDA/C++.
  6. Hybrid-implementation: threaded (task-based) version on the CPU which runs concurrently with perfectly data-parallel tasks on the GPU ($N \sim 10,000,000$).
  7. *Todo:* 3D...?
- Increasingly sophisticated regression tests.

Conclusions

- Academic sw-project driven by a concrete problem.
- Nice aspect: the steady and controlled growth of performance and complexity:
  1. Stand-alone *recursive* uniform FMM implementation ($N_{\text{max}} \sim 20,000$).
  2. Matlab-interface to a *direct* N-body evaluation ($N \sim 10,000$).
  3. First efficient working copy of uniform FMM; later heavily optimized, eg. BLAS L3 ($N \sim 1,000,000$, memory bound!).
  4. Added adaptivity – took the time to investigate a novel approach.
  5. Fully data-parallel GPU-implementation CUDA/C++.
  6. Hybrid-implementation: threaded (task-based) version on the CPU which runs concurrently with perfectly data-parallel tasks on the GPU ($N \sim 10,000,000$).
  7. *Todo:* 3D...?
- Increasingly sophisticated regression tests.
- Academic environment; clarity wrt to goals is *very* important.

Conclusions

Programs, Papers, and Preprints are available from my web-page.
Thank you for the attention and for the lunch!