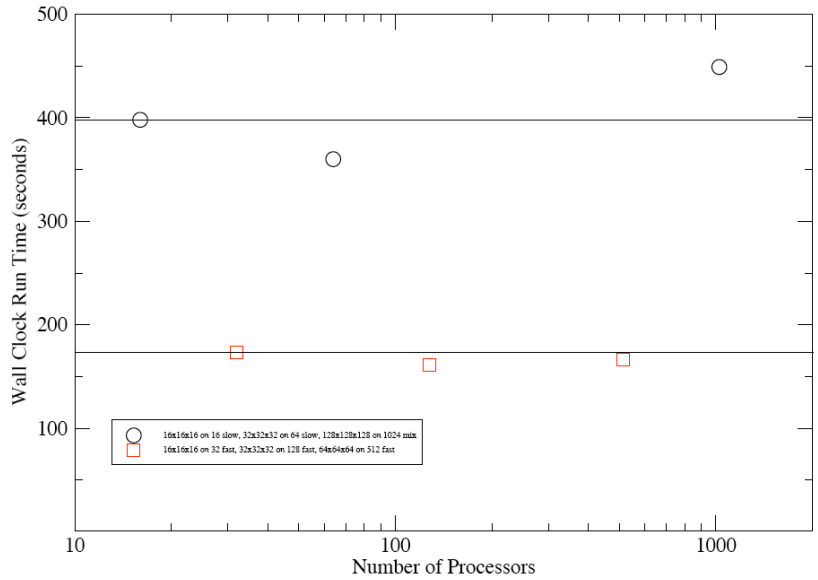
Scaling Results - Hyperbolic Conservation Laws

- O(10^3) Flops per grid point per time step; 5 unknowns per grid point.
- One exchange of ghost cells per operator evaluation.
- Refinement along a surface: refinement of base grid by 2X leads to a increase of 4X in the workload.
- 75% scaled efficiency up to 1024 processors.



Scaled speedup on HP Alpha / Quadrics (halem.gsfc.nasa.gov)

Prob size	Num Procs	Avg Memory MB	Min-Max Memory MB	AMR Run time (sec)
32x32x32	32	433	359-493	1558
32x32x32	64	249	195-285	811
32x32x32	128	148	113-171	432
64x64x64	128	411	340-473	1593
64x64x64	256	249	197-291	838
64x64x64	512	168	128-199	449
128x128x128	512	454	392-532	1713
128x128x128	1024	299	250-348	974

Scaling on SP3 (seaborg.lbl.gov)

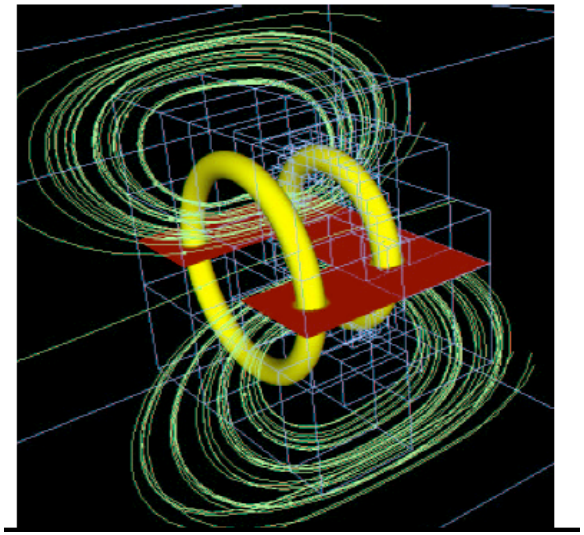
AMR for Incompressible Flow

• $O(10^3)$ Flops per grid point per time step; 15 unknowns per grid point.

• Dominated by elliptic solver: one exchange of ghost cells per operator evaluation, but O(100) operator evaluations.

• Refinement in a subvolume: refinement of base grid by 2X leads to 8X increase in the workload.

• 75% scaled efficiency up to 128 processors



Base Problem Size	Num Procs	Large Problem Size	Large num processors	Scaled Efficiency
32x32x48	1	64x64x96	8	1.13
	2		16	1.03
	4		32	0.96
	8		64	0.75
32x32x48	1	128x128x192	64	1.07
	2		128	0.86
64x64x96	8	128x128x192	64	0.99
	16		128	0.83

Scaling on HP Alpha / Quadrics (halem.gsfc.nasa.gov)

Differences in Scaling

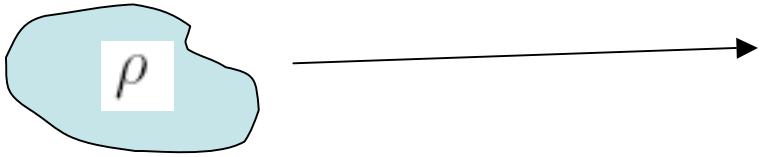
- Incompressible AMR performance dominated by elliptic performance.

- Iterative solvers such as multigrid are dominated by communications costs: 100X as many messages as in the hyperbolic case for about the same number of floating point operations.

- Is this intrinsic to the problem, or an artifact of our solution method ?

Analysis-Based Poisson Solvers

$$\Delta\varphi = \rho \ , \ \varphi(\boldsymbol{x}) = \int_{\Omega} G(\boldsymbol{x}-\boldsymbol{y})\rho(\boldsymbol{y})d\boldsymbol{y} \ , \ \boldsymbol{x} \in \Omega$$



Real analytic, with rapidly convergent Taylor expansion

- Straightforward evaluation of the convolution operator leads to an $O(N^2)$ calculation, where N is the number of points in the discretization of the domain.
- Idea: solution is a local function of the data, modulo analytic functions. Exploit this property to reduce cost to O(N), design domain decomposition algorithms for parallel computers.

Analysis-Based Poisson Solvers

$$\varphi(\boldsymbol{x}) = \int_{\Omega} G(\boldsymbol{x}-\boldsymbol{y})\rho(\boldsymbol{y})d\boldsymbol{y} \ , \ \boldsymbol{x} \in \Omega \qquad \varphi = \varphi_{\text{local}} + \varphi_{\text{global}}$$

- Fourier Methods: φ_{local} represented by high-wavenumber modes, φ_{global} represented by low-wavenumber modes. Advantage: extremely fast. Disadvantage: only works for single rectangular grid, does not localize communication.
- Multigrid: φ_{local} represented by solution to associated heat equation, φ_{global} represented by solution to heat equation on coarser grid. Advantage: works for more general grids, variable coefficient problems. Disadvantage: localizes computation, but not communication.
- Fast multipole method: φ_{local} represented by local N^2 calculation, φ_{global} represented by multipole (Taylor) expansion plus interpolation. Advantage: works for general grids, localizes communication as well as computation. Disadvantage: cost of computation increases dramatically as with dimensionality.

All applied recursively to obtain O(N) - O(NlogN) algorithms.

Method of Local Corrections (Anderson, 1986)

MLC is a non-iterative domain decomposition method for computing

$$\varphi(\boldsymbol{x}) = \int_{\Omega} G(\boldsymbol{x}-\boldsymbol{y})\rho(\boldsymbol{y})d\boldsymbol{y} \ , \ \ \boldsymbol{x} \in \Omega \qquad (\ \varphi = G * \rho\ )$$

(1) Solve local problems on overlapping local patches:

$$\rho = \sum_l \rho^l \ , \ supp(\rho^l) \subset \Omega_0^l \qquad \varphi^l = G * \rho^l \ \text{ on } \ \Omega^l \ , \ \Omega^l \supset \Omega_0^l$$

(2) Solve a single coarse-grained problem to represent the nonlocal coupling:

$$R_{\boldsymbol{i}}^{H,l} = \begin{cases} (\Delta^H \varphi^l)_{\boldsymbol{i}} & \text{if } \boldsymbol{i}H \in \Omega^l \\ 0 & \text{otherwise} \end{cases} \qquad R^H = \sum_l R^{H,l} \qquad \varphi^H = G^H * R^H$$

(3) Compute composite solution as combination of local fields and interpolated corrected global field:

$$\varphi(\boldsymbol{x}) = \sum_{l:\boldsymbol{x}\in\Omega^l} \varphi^l(\boldsymbol{x}) + \mathcal{I}\left(\varphi^H - \sum_{l:\boldsymbol{x}\in\Omega^l} \varphi^l\right)(\boldsymbol{x})$$

Single Patch Case

Consider the case where only one of the local densities is nonzero.

$$\rho = \sum_l \rho^l \ , \ supp(\rho^l) \subset \Omega_0^l$$

$$\varphi^l = G * \rho^l \ \text{ on } \ \Omega^l \ , \ \Omega^l \supset \Omega_0^l$$

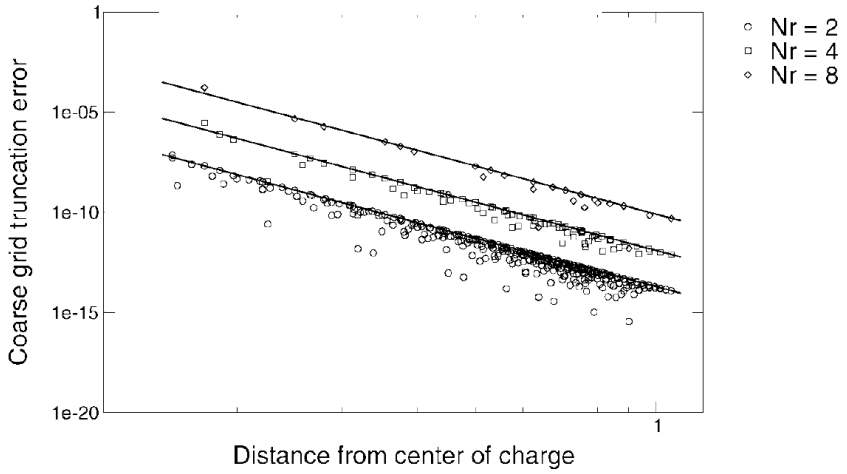
(2) If we don't truncate the coarsened right-hand side, the global solve is just the identity operator.

$$R_\infty^H = \Delta^H \varphi^l$$

$$\varphi^H = (\Delta^H)^{-1} R_\infty^H = (\Delta^H)^{-1}(\Delta^H \varphi^l) = \varphi^l$$

However, the solution satisfies $\Delta\varphi \equiv 0$ away from the support of the right-hand side, so that truncating it introduces a small error, and an enormous reduction of cost.

$$\tilde{R}^H = 3.64 \times 10^{-6} \frac{H^6}{|\vec{x}|^8}$$



$$R^H - R_\infty^H = O(H^p)$$

$$\Rightarrow \varphi^H = \varphi^l + O(H^p)$$

Single Patch Case

(3) To a high degree of accuracy, the final local corrections step

$$\varphi(\boldsymbol{x}) = \sum_{l:\boldsymbol{x}\in\Omega^l} \varphi^l(\boldsymbol{x}) + \mathcal{I}\left(\varphi^H - \sum_{l:\boldsymbol{x}\in\Omega^l} \varphi^l\right)(\boldsymbol{x})$$

reduces either

- to evaluating the local solution near the support of the right-hand side,

$$\varphi(\boldsymbol{x}) = \varphi^l(\boldsymbol{x})$$

- or to interpolating a smooth function away from the support of the right-hand side

$$\varphi(\boldsymbol{x}) = \mathcal{I}(\varphi^H)(\boldsymbol{x})$$

Method of Local Corrections

Anderson's original algorithm was a PPPM method for vortex transport in 2D - local calculation was a N-body method, coarse calculation a Mehrstellen finite difference method. Baden (1986) used the method as the basis of parallel domain-decomposition algorithm for particle methods.

What is needed to apply this idea to 3D multiresolution gridded calculations?

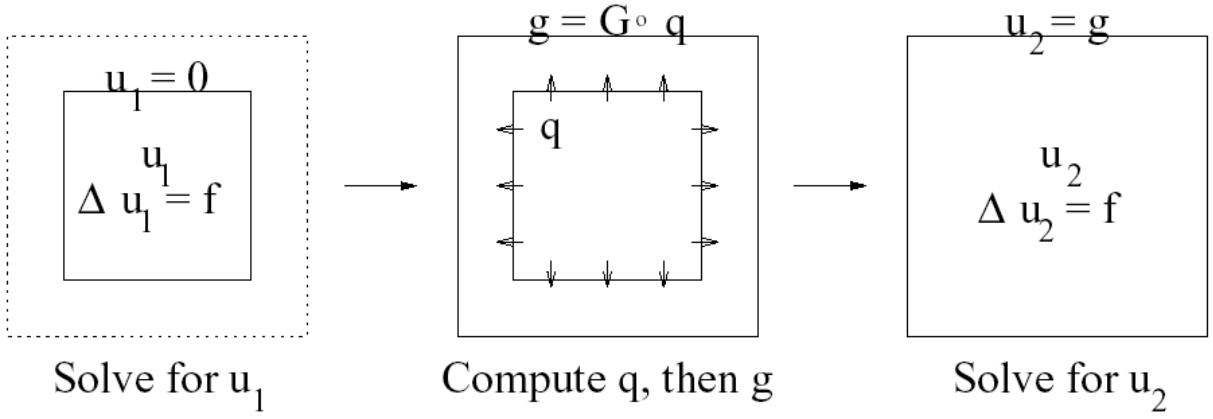
- Efficient grid-based infinite domain solver.
- Mehrstellen discretizations play an essential role:

$$\tau \equiv \Delta^h \varphi - \rho = h^2 \mathcal{L}(\rho) + O(h^p) \quad , \quad p = 4, 6$$

High-order accuracy in regions where the solution is harmonic.

Ref: Balls and Colella, 2002; Baden, Balls, Colella, McCorquodale, 2005.

James' Algorithm for Infinite-Domain Boundary Conditions



- Two Dirichlet solves are performed using FFTs on a single processor - very fast.
- In 3D, the direct calculation of the surface-surface potential is too expensive - $O(N^4)$. We reduce this to a small fraction of the solver cost using a simplified multipole expansion.

$$\int G(\boldsymbol{x}-\boldsymbol{y})\rho(\boldsymbol{y})d\boldsymbol{y} \approx \sum_{\boldsymbol{p}} \frac{1}{\boldsymbol{p}!}\frac{\partial^{\boldsymbol{p}} G}{\partial \boldsymbol{y}^{\boldsymbol{p}}}\Big|_{\boldsymbol{y}=\boldsymbol{y}_0} \int \rho(\boldsymbol{y})(\boldsymbol{y}-\boldsymbol{y}_0)^{\boldsymbol{p}} d\boldsymbol{y}$$

If m_y is the number of charges, and m_x is the number of evaluation points, replacing direct evaluation with multipole reduces computation from $O(m_x m_y)$ to $O(m_x + m_y)$.

James' Algorithm for Infinite-Domain Boundary Conditions

We compute $g(\vec{x}) = \int_{\partial D_1} G(\vec{x} - \vec{y}) q(\vec{y}) dA_{\vec{y}}$

for $\vec{x} \in \partial D_2$ with 2-D fast multipole method.

Example: for patch on face of ∂D_1 with $z = c$, set

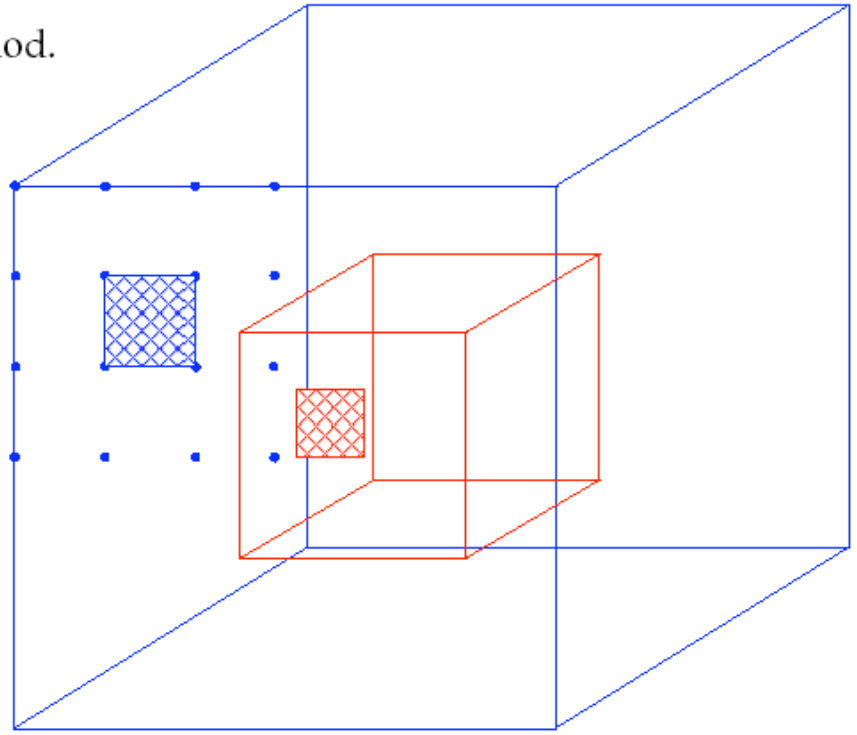
$$Q_{a,b} = \frac{(-1)^{a+b}}{a!b!} \int_{\text{patch}} q(x, y, c) x^a y^b dx dy$$

for $a + b \leq p =$ degree.

On coarse points of ∂D_2, do

$$g(x, y, z) = \sum_{a,b} Q_{a,b} \frac{\partial^{a+b} G}{\partial x^a \partial y^b}(x, y, z)$$

and then interpolate in 2-D.



Cost: $O(N^{2/3}(p^2 + q))$

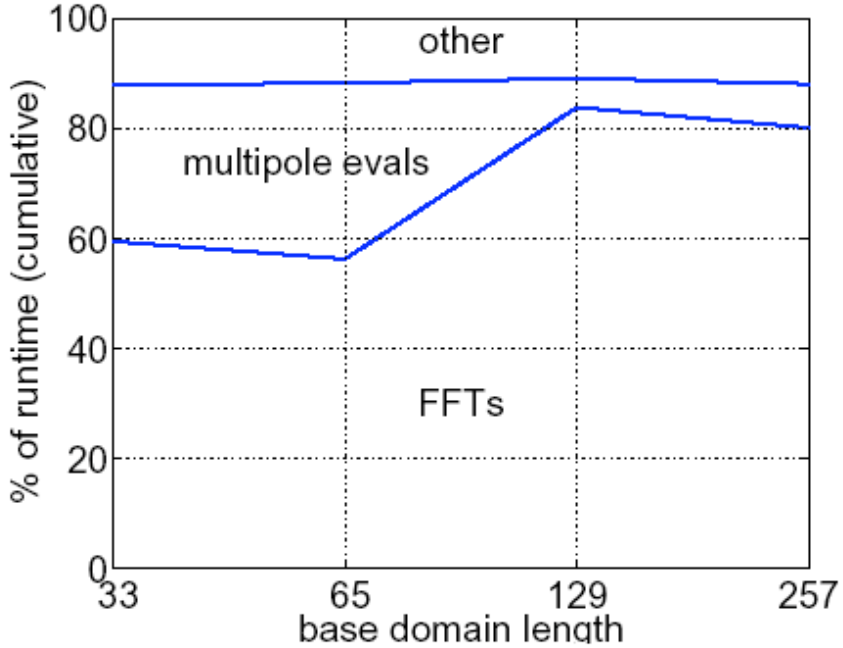
Accuracy: $O(h^2)$ when $p = 3, q = 3$

Performance of Fast James' Algorithm

D is $n \times n \times n$, D_1 is $n_1 \times n_1 \times n_1$, D_2 is $n_2 \times n_2 \times n_2$. Patch size is r.

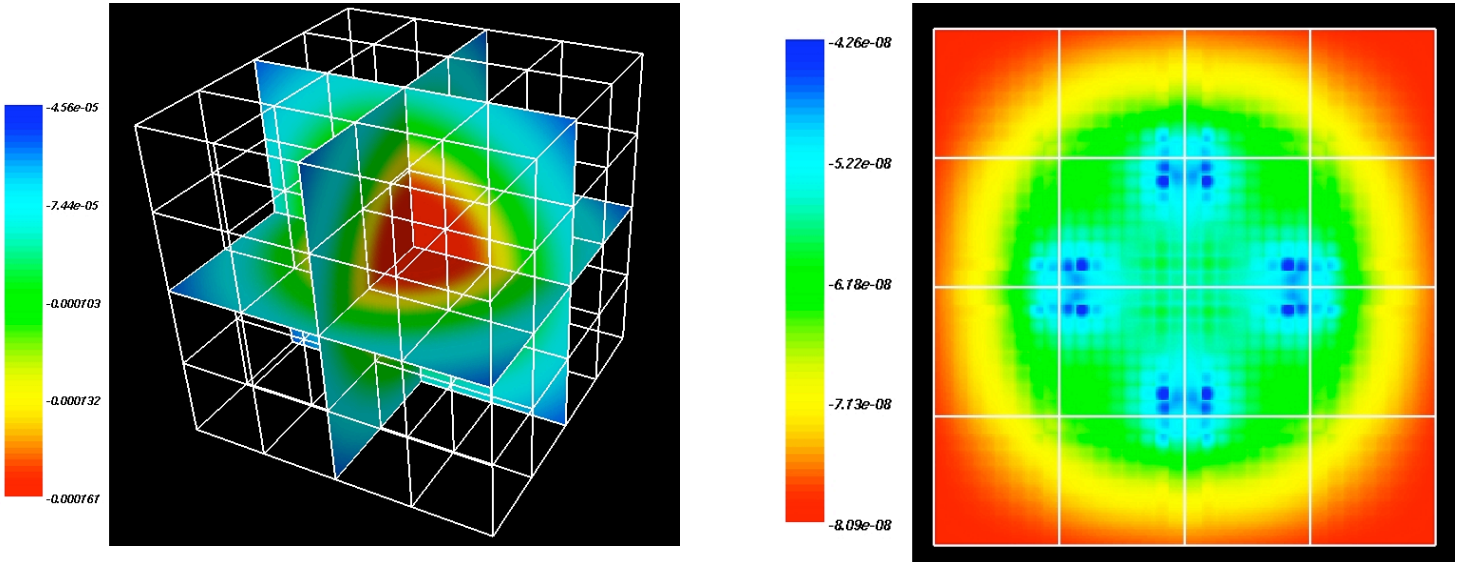
n	n_1	n_2	r	total time
33	41	49	8	0.23
65	73	81	8	1.28
129	153	177	16	10.88
257	289	321	16	63.98

Most time is spent in FFTs:



MLC Results

Input Parameters				Times for Each Stage (seconds)						Total	Proc-time/pt.	Percent
P	q	C	N^3	Local	Red.	Global	GC	Bnd.	Final	(sec)	(μsec)	Comm.
128	8	6	768^3	28.44	0.61	12.12	0.13	1.86	4.66	47.37	13.33	5.5
256	8	8	1024^3	35.31	0.69	12.22	0.20	1.66	5.83	55.31	13.15	4.6
512	8	10	1280^3	33.78	0.85	12.81	0.31	1.78	7.5	56.37	13.73	5.2
1024	16	12	1536^3	37.03	0.98	13.14	0.58	3.55	4.79	59.51	16.78	8.6



3D MLC calculation of Poisson's equation, with 256^3 mesh broken into 64^3 blocks. Image on right shows solution error through slice at mid-plane.

MLC Status and Plans

Finish final performance squeeze of two-level MLC: parallelize FFT global solves, squeeze down the overlap in the local solves.

Multilevel MLC.

- "Non-iterative multigrid" - communication comparable to a single AMR V-cycle.
- Reduction in mesh spacing by a fixed factor between levels. Overlap conditions are less onerous, and fewer terms in multipole expansion - local solves less expensive.

Other problems: heat equation in 3D; cell-centered solvers; nontrivial boundary conditions …

Goal: AMR Poisson solvers at cost per grid point = 5X uniform grid serial FFT (right now at 6X-8X).

Conclusions

• Successful parallel algorithms for adaptive algorithms need to exploit of the tools and structure available (Mathematics and CS).

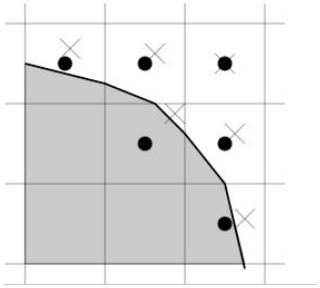
• Software design is a process of mapping mathematical abstractions onto programming abstractions. Done correctly, one obtains substantial reuse across applications.

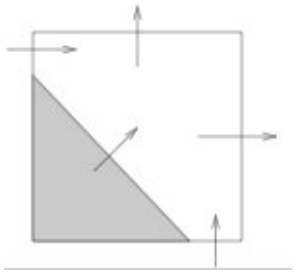
• AMR is too complicated to write everything from scratch. Progress will be made only if we are willing to build on existing software efforts.

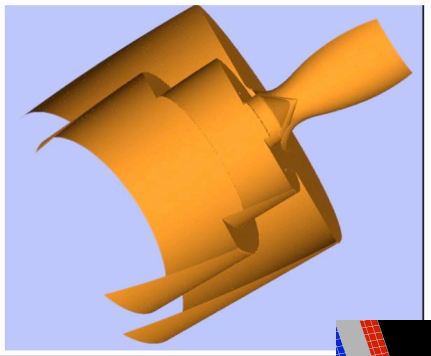
http://seesar.lbl.gov/anag , http://davis.lbl.gov/APDEC

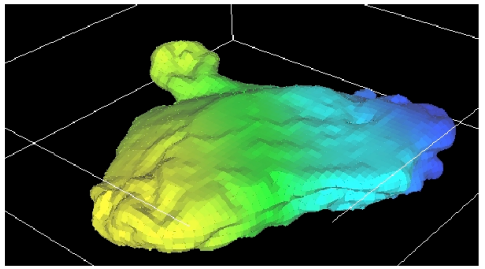
Embedded Boundary Discretization of Complex Geometries

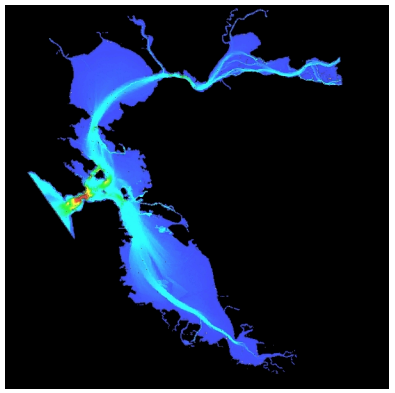
$$\nabla \cdot \vec{F} = \rho \ , \ \vec{F} = \nabla \phi \qquad \frac{\partial U}{\partial t} + \nabla \cdot \vec{F}(U) = 0$$

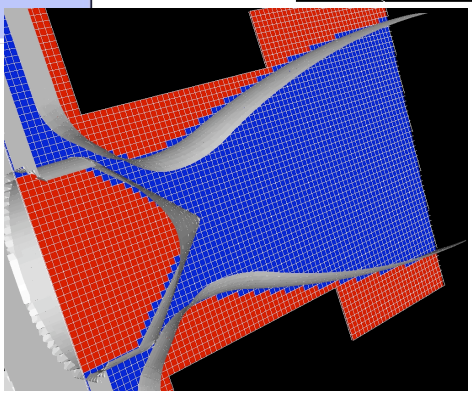


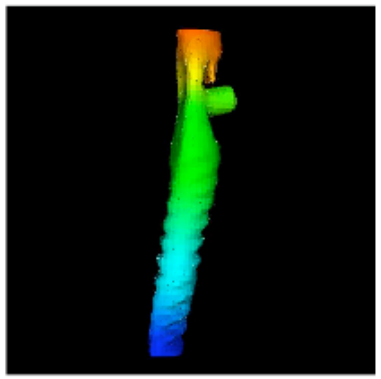


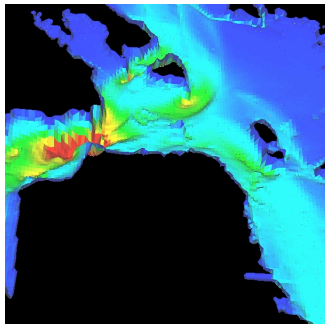












Office of Science
U.S. DEPARTMENT OF ENERGY

SciDAC
Scientific Discovery through Advanced Computing