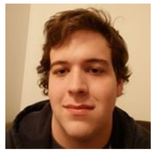
Massively Parallel Solvers for Computational Fluid Dynamics on Multi-block Cartesian Grids

BOISE STATE UNIVERSITY
COLLEGE OF ARTS AND SCIENCES
Department of Mathematics


Scott Aiton


Brenton Peck


Donna Calhoun


Grady B. Wright

Research supported by
NASA IDAHO Space Grant CONSORTIUM FPK 900-034
NSF ACI-1440638

Introduction

Computational fluid dynamics (CFD) plays a central role in numerical weather prediction (NWP) models for forecasting and understanding the Earth as a system. Often these models produce results with large uncertainty and can fail to predict the intensity of weather events with desired accuracy. This is especially true for forecasting microscale wind over arbitrarily complex terrain, which is one of the least understood subjects in the atmospheric sciences, but is fundamental to many crucial problems, such as predicting microscale contaminant dispersion. Growing evidence indicates that these shortcomings arise from a combination of inadequate modeling of microscale physical processes within the Earth's atmosphere and inherent numerical errors in the CFD methods.

To address these shortcomings, the PIs are developing a massively parallel, open-source CFD software package **GEM3D** to study microscale atmospheric flows over complex terrain with numerical resolutions that are much finer than any current software allows. Our package uses block structured Cartesian adaptive mesh refinement (AMR) methods to provide a practical, accurate method for geometrically modeling complex terrain and large-eddy simulation techniques with novel dynamic sub-grid scale models for physically modeling incompressible flows in complex terrain.

To make near-real time forecasting possible, the software is being designed and developed to exploit the massively parallel, multi-GPU (graphics processing unit) computing paradigm efficiently.

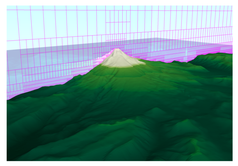


Figure. Example of embedded boundary structured Cartesian adaptive mesh refinement (AMR) over terrain generated by the GEM3D code. Only a vertical slice of the grid is shown in magenta.

Pressure Poisson equation

Solving the Pressure Poisson equation is one of the key challenges in CFD models for incompressible flows. The equation arises from the condition that the velocity of the fluid remain incompressible, or that the density of the fluid remain constant. Solving the equation is often the dominating computational cost, as it requires solving a massive linear system of equations every time-step of a simulation.

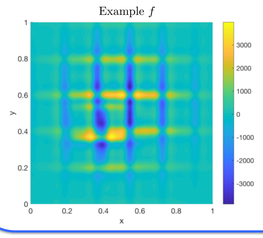
Pressure Poisson Problem on a domain Ω with boundary $\partial\Omega$:

$$\text{Solve: } \begin{cases} \nabla^2 p = f & \text{on } \Omega \\ \mathbf{n}\cdot\nabla p = 0 & \text{on } \partial\Omega \end{cases} \quad \text{for } p \text{ given } f, \tag{1}$$

where $\mathbf{n}$ is the unit outward normal vector to Ω.

GEM3D uses *embedded boundary methods* for dealing with complex geometries, which allows us to focus on solving the Pressure Poisson equation on rectangular domains.

Focus of this poster is on 2D rectangular domains



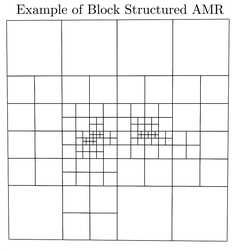
Block Structured Cartesian AMR

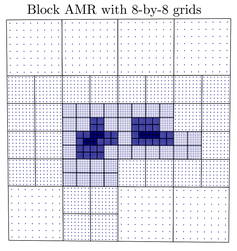
We use a nested hierarchy of fixed size, non-overlapping, Cartesian grids to discretize the rectangular domains.

Benefits:

- Use fine grids to capture localized, small-scale features of the problems and coarse grids in regions with large-scale features.
- Use standard finite difference/volume schemes to discretize the equations.

We enforce a constant 2:1 refinement ratio between grid levels.


Example of Block Structured AMR


Block AMR with 8-by-8 grids

130 blocks
8320 grid points
Finest resolution 1/512

Domain Decomposition

Block structured Cartesian AMR is naturally set up for a domain decomposition (DD) approach for solving the Pressure Poisson problem.

Simple Block AMR For Explaining DD

Ω_4	Ω_7	Ω_9
Ω_2	Ω_5	Ω_8
Ω_1	Ω_3	Ω_6

$$\Omega = \bigcup_{i=1}^{M} \Omega_i \qquad \Gamma_i = \text{Boundary of } \Omega_i$$

Transform the Pressure Poisson problem (1) to solving

$$\begin{cases} \nabla^2 p_i = f_i & \text{on } \Omega_i \\ p_i = \gamma_i & \text{on } \Gamma_i \backslash \partial\Omega \\ \mathbf{n}\cdot\nabla p_i = 0 & \text{on } \partial\Omega \cap \Gamma_i \end{cases} \tag{2}$$

Benefit:

Highly efficient fast Fourier transform (FFT) based methods can be used to solve each subdomain problem in $\mathcal{O}(n \log n)$

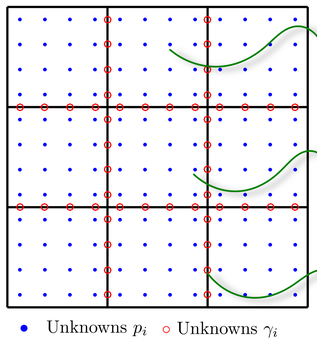
Issue:

Have to determine γ_i so that p_i also solves (1) in each Ω_i.

This couples all the domains together.

Finite-difference Discretization

Cell-centered n-by-n grid in each Ω_i



- Unknowns p_i ○ Unknowns γ_i

At interior grid points approximate $\nabla^2 p_i = f_i$ as:

$$-2(p_i)_{j,k} + (p_i)_{j-1,k} + (p_i)_{j+1,k}$$
$$-2(p_i)_{j,k} + (p_i)_{j,k-1} + (p_i)_{j,k+1} = h^2 (f_i)_{j,k}$$

Next to boundary of Ω_i approximate $\nabla^2 p_i = f_i$ as

$$-2(p_i)_{j,n} + (p_i)_{j-1,n} + (p_i)_{j+1,n}$$
$$-2(p_i)_{j,n} + (p_i)_{j,n-1} + 2(\gamma_i)_{j,n+\frac{1}{2}} = h^2 (f_i)_{j,n}$$

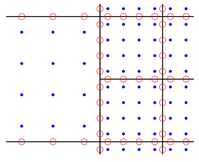
To determine γ_i we enforce smoothness across the domains:

$$\frac{(p_i)_{j,n} + (p_{i,r})_{j,1}}{2} - (\gamma_i)_{j,n+\frac{1}{2}} = 0$$

These approximations are all second order accurate, so that the local truncation error is $\mathcal{O}(h^2)$

We can extend these discretizations to more complicated block structured AMR settings:

- Use conservation of fluxes to derive stencils at the boundaries
- Use bi-cubic interpolation to define interface equations for γ_i

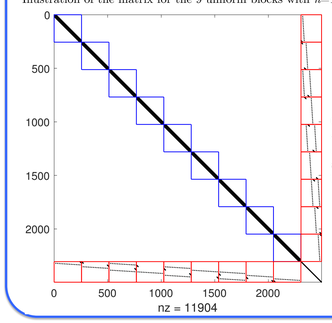


Linear System of Equations

We can arrange the finite difference and interface equations in to a sparse linear system of equations:

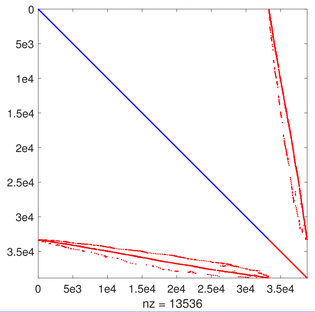
$$Lp + B\gamma = f \implies \text{Discretization of } \nabla^2 p = f \text{ with boundary conditions on subdomains domains}$$
$$\tilde{B}p + C\gamma = 0 \implies \text{Equations for determining the boundary conditions on the subdomains}$$

Illustration of the matrix for the 9 uniform blocks with n=16



Outlines show the equations for the 9 non-overlapping subdomains and interface conditions.

Illustration of the matrix for the 30 AMR blocks with n=16



Outlines of the blocks not included here since the resulting image is too messy.

Schur Complement Linear System

The interface values can first be computed directly by solving the Schur Complement Linear System:

$$\underbrace{\left(C - \tilde{B}L^{-1}B\right)}_{S}\gamma = -\tilde{B}L^{-1}f \tag{3}$$

Comments:

- The actual formation of S can be done *fast* by exploiting the block Cartesian AMR structure.
- At most $8n$ systems of size n-by-n have to be solved to determine S completely.
- Each of these systems can be solved in $\mathcal{O}(n \log n)$ operations using FFT based solvers.
- After computing γ, the values of p can be solved independently on each subdomain in $\mathcal{O}(n \log n)$ operations.

Illustration of the Schur complement matrix for the 9 uniform blocks with n=16

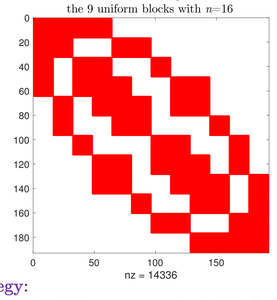
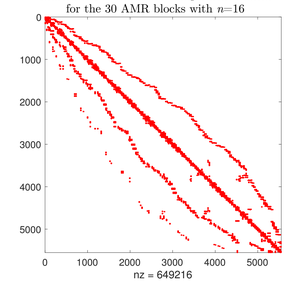


Illustration of the Schur complement matrix for the 30 AMR blocks with n=16



Our strategy:

Use the Generalized Minimum Residual(GMRES) iterative method preconditioned with Algebraic Multigrid (AMG) to solve the Schur Complement system for γ.

Numerical results

The following methods are compared:

(a) Domain decomposition as described above: solving the Schur complement system (3) with AMG preconditioned GMRES and then solving (2) on each subdomain using FFT based solvers.

(b) Constructing the full linear system for the discretization of (1) directly (i.e. without domain decomposition) and solving with AMG preconditioned GMRES.

We also compare these methods on CPU and GPU architectures. All runs were done on Kestrel.

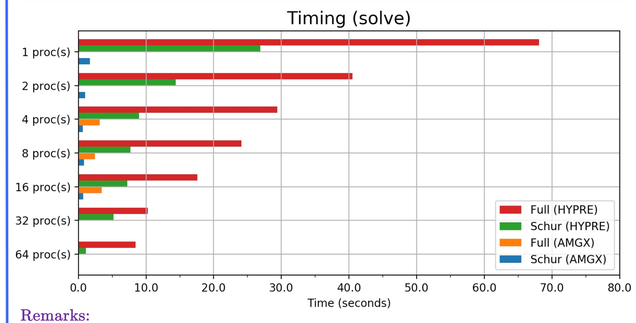
Software libraries:

CPU: HYPRE (BoomerAMG); PETSc (GMRES); FFTW (Fast Cosine/Sine transforms)

GPU: NVIDIA Linear Algebra Library (AMGx) and Signal Library (cuFFT); code compiled under CUDA 7.5.

Timing results for approximating (1)

	Full (HYPRE)		Full (AMGX)		Schur (HYPRE)		Schur (AMGX)	
p	Time (s)	Iterations	Time (s)	Iterations	Time (s)	Iterations	Time (s)	Iterations
1	68.07	27	—	—	26.91	17	1.69	19
2	40.49	25	—	—	14.38	17	1.01	20
4	29.41	26	3.17	53	8.99	16	0.63	20
8	24.12	26	2.44	54	7.67	16	0.83	41
16	17.56	25	3.47	100	7.25	16	0.69	43
32	10.27	25	—	—	5.20	16	—	—
64	8.43	25	—	—	1.11	16	—	—



Details:

8320 AMR blocks with 16-by-16 grids per block, which gives 2,129,920 degrees of freedom

Method of manufactured solution used to validate the code

For AMGx, proc(s) and p means number of GPUs used

GMRES relative tolerance= 10^{-12}, and restart is 40

AMG preconditioner uses 1 v-cycle with 1 pre/post smooth

CPU: 2.4 GHz Intel Xeon E5-2600; 2 processor per node; 64GB memory per node

GPU: NVIDIA Tesla K20; 2 GPUs per node; 5GB memory per GPU

Mellanox ConnectX-3FDR Infiniband interconnect

Remarks:

- The Schur complement domain decomposition method is faster than just using GMRES/AMG on the full system.
- The GPU versions of the solvers are anywhere from 5 to 16 times faster than the respective CPU versions.
- NVIDIA's GMRES and AMGx methods have inconsistent behavior in terms of iterations on multi-GPUs.
- We are in the process of extending the Schur complement domain decomposition method to 3D, developing better preconditioners than AMG for this system, and running the code on larger machines.
- All codes are available at https://github.com/GEM3D/pressurePoissonSolver.