Continuous and Discontinuous Galerkin Methods

Frank Giraldo
Department of Applied Mathematics
Naval Postgraduate School
Monterey CA 93943 USA

ECMWF Seminar on

Recent Developments in Numerical Methods for Atmosphere and Ocean Modelling

Motivation

Our goal is to construct nonhydrostatic atmospheric models with the following capabilities:

① Highly scalable on current and future computer architectures
② Permitting general grids (e.g., statically and dynamically adaptive)
③ Highly efficient
④ High-order accuracy in all the numerics: spatial and temporal
⑤ Unified regional and global NWP model
⑥ Conservative, minimally dissipative with excellent dispersion properties

This talk will only focus on Spatial Discretization methods

Talk Summary

1. Unified formulation of CGDG Methods
2. Positivity Preservation
3. The NUMA Model
4. Adaptive Mesh Refinement
5. Parallelization
6. Examples with CGDG
7. Closing Remarks

1. Unified CGDG

Desirable Properties of Numerical Methods

① Stability/Robustness

② Accuracy

③ Efficiency (e.g., exascale)

④ Geometric Flexibility (e.g., hp refinement)

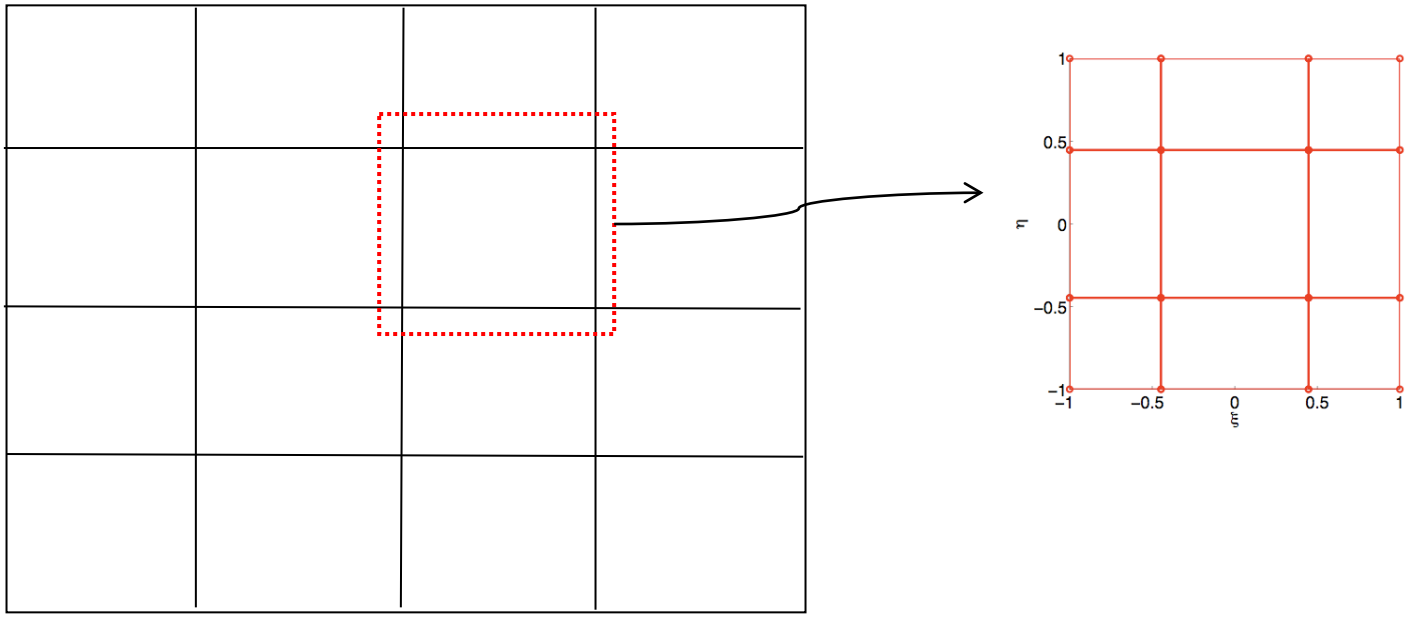
⑤ Long Shelf-life (adaptable to new requirements)

CGDG methods offer one possibility to achieve these goals

1. Unified CGDG

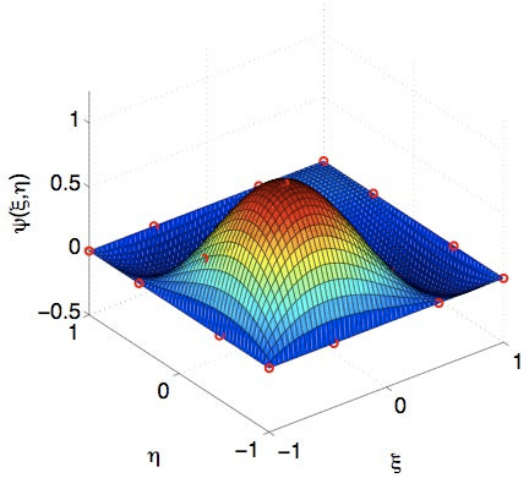
Question: What are CGDG methods?

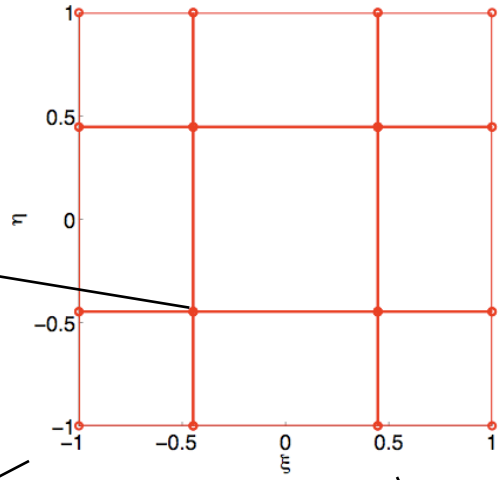
An element is chosen to be the basic building-block of the discretization and then a polynomial expansion is used to represent the solution inside the element. These are element-based Galerkin methods.

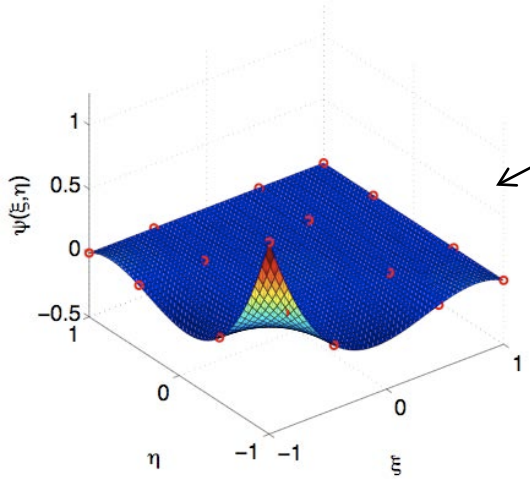


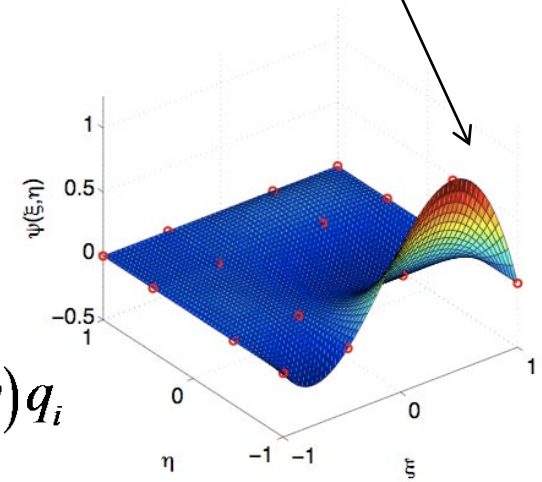
If there is only one element spanning the global domain then we recover spectral methods

1. Unified CGDG









$$q_N(\xi,\eta)=\sum_{i=1}^{M_N}\psi_i(\xi,\eta)q_i$$

1. Unified CGDG: General Idea

- Primitive Equations: $\frac{\partial q}{\partial t} + \nabla \cdot \mathbf{F} = S(q)$
- Approximate the solution as: $q_N = \sum_{i=1}^{M_N} \psi_i q_i \qquad \mathbf{F}_N = \mathbf{F}(q_N) \qquad S_N = S(q_N)$
  - **Interpolation O(N)**
- Write Primitive Equations as: $R(q_N) \equiv \frac{\partial q_N}{\partial t} + \nabla \cdot \mathbf{F}_N - S_N = \varepsilon$
- Weak Problem Statement: Find

$$q_N \in \Sigma(\Omega) \; \forall \; \psi \in \Sigma$$

$$\Sigma = \left\{ \psi \in H^1(\Omega) : \psi \in P_N(\Omega_e) \forall \Omega_e \right\} \quad \text{(CG)}$$

$$\Sigma = \left\{ \psi \in L^2(\Omega) : \psi \in P_N(\Omega_e) \forall \Omega_e \right\} \quad \text{(DG)}$$

  - such that (**Integration O(2N))**

$$\int_{\Omega_e} \psi R(q_N) d\Omega_e = 0$$

1. Unified CGDG

- Integral Form:

$$\int_{\Omega_e} \psi R(q_N)\, d\Omega_e = 0$$

- Matrix Form:

$$R_i^{(e)} \equiv M_{ij}^{(e)} \frac{dq_j^{(e)}}{dt} + \sum_{f=1}^{N_{face}} \left(\mathbf{M}_{ij}^{(e,f)}\right)^T \mathbf{F}_j^{(e,f)} - \left(\mathbf{D}_{ij}^{(e)}\right)^T \mathbf{F}_j^{(e)} - S_i^{(e)} = 0$$

- Where each matrix is:

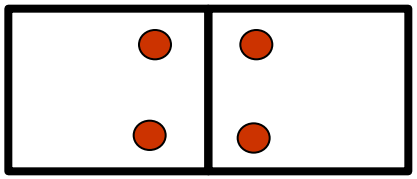
$$M_{ij}^{(e)} = \int_{\Omega_e} \psi_i \psi_j d\Omega_e$$

$$\mathbf{D}_{ij}^{(e)} = \int_{\Omega_e} \nabla \psi_i \psi_j d\Omega_e \quad \mathbf{M}_{ij}^{\Gamma} = \int_{\Gamma} \mathbf{n} \psi_i \psi_j d\Gamma$$

Integration O(2N)

For DG: $\mathbf{C}(R)=(M^{(e)})^{-1}\ \mathbf{R}_i^{(e)}$

For CG: $\mathbf{C}(R)=M^{-1}\, S(G(\mathbf{R}_i^{(e)}))$



$\mathbf{R}_i^{(e)}$

1. Unified CGDG

- Integral Form:

$$\int_{\Omega_e} \psi R(q_N)\, d\Omega_e = 0$$

- Matrix Form:

$$R_i^{(e)} \equiv M_{ij}^{(e)} \frac{dq_j^{(e)}}{dt} + \sum_{f=1}^{N_{face}} \left(\mathbf{M}_{ij}^{(e,f)}\right)^T \mathbf{F}_j^{(e,f)} - \left(\mathbf{D}_{ij}^{(e)}\right)^T \mathbf{F}_j^{(e)} - S_i^{(e)} = 0$$

- Where each matrix is:

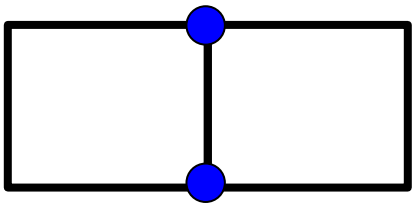
$$M_{ij}^{(e)} = \int_{\Omega_e} \psi_i \psi_j \, d\Omega_e$$

$$\mathbf{D}_{ij}^{(e)} = \int_{\Omega_e} \nabla \psi_i \psi_j \, d\Omega_e \qquad \mathbf{M}_{ij}^{\Gamma} = \int_{\Gamma} \mathbf{n} \psi_i \psi_j \, d\Gamma$$

Integration O(2N)

For DG: $\mathbf{C}(R) = (M^{(e)})^{-1}\ \mathbf{R}_i^{(e)}$

For CG: $\mathbf{C}(R) = M^{-1}\, S(G(\mathbf{R}_i^{(e)}))$



$\mathbf{R}_I = G(\mathbf{R}_i^{(e)})$

1. Unified CGDG

- Integral Form:

$$\int_{\Omega_e} \psi R(q_N)\, d\Omega_e = 0$$

- Matrix Form:

$$R_i^{(e)} \equiv M_{ij}^{(e)} \frac{dq_j^{(e)}}{dt} + \sum_{f=1}^{N_{face}} \left(\mathbf{M}_{ij}^{(e,f)}\right)^T \mathbf{F}_j^{(e,f)} - \left(\mathbf{D}_{ij}^{(e)}\right)^T \mathbf{F}_j^{(e)} - S_i^{(e)} = 0$$

- Where each matrix is:

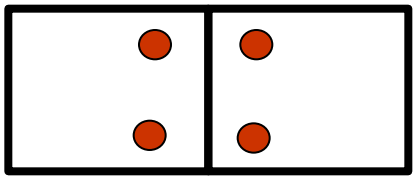
$$M_{ij}^{(e)} = \int_{\Omega_e} \psi_i \psi_j \, d\Omega_e$$

$$\mathbf{D}_{ij}^{(e)} = \int_{\Omega_e} \nabla \psi_i \psi_j \, d\Omega_e \qquad \mathbf{M}_{ij}^{\Gamma} = \int_{\Gamma} \mathbf{n} \psi_i \psi_j \, d\Gamma$$

Integration O(2N)

For DG: $\mathbf{C}(R) = (M^{(e)})^{-1}\ \mathbf{R}_i^{(e)}$

For CG: $\mathbf{C}(R) = M^{-1}\, S(G(\mathbf{R}_i^{(e)}))$



$\mathbf{R}_i^{(e)} = S(\mathbf{R}_l)$

1. Unified CGDG: Weak & Strong

What are the Strong and Weak forms?

- Let's start with the PDE: $\frac{\partial q}{\partial t} + \nabla \cdot F = 0$
- Integrate by Parts: $\int_{\Omega_e} \psi_i \frac{\partial q_N^{(e)}}{\partial t} d\Omega_e - \int_{\Omega_e} \nabla \psi_i \cdot F_N^{(e)} d\Omega_e + \int_{\Gamma_e} \psi_i \hat{\mathbf{n}} \cdot F_N^{(e)} d\Gamma_e = 0$
- Introducing a numerical flux (*) yields the Weak form:

$$\int_{\Omega_e} \psi_i \frac{\partial q_N^{(e)}}{\partial t} d\Omega_e - \int_{\Omega_e} \nabla \psi_i \cdot F_N^{(e)} d\Omega_e + \int_{\Gamma_e} \psi_i \hat{\mathbf{n}} \cdot F_N^{(*)} d\Gamma_e = 0$$

- Integrating the 2nd term by parts again yields:

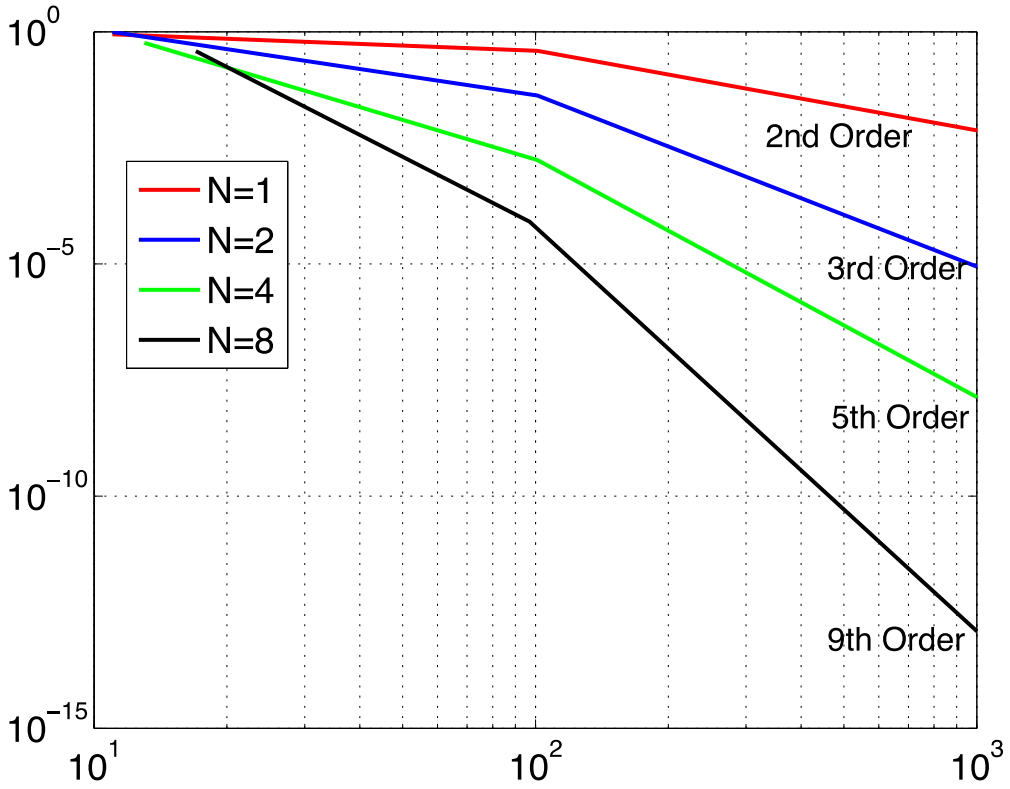
$$\int_{\Omega_e} \psi_i \frac{\partial q_N^{(e)}}{\partial t} d\Omega_e + \int_{\Omega_e} \psi_i \cdot \nabla F_N^{(e)} d\Omega_e - \int_{\Gamma_e} \psi_i \hat{\mathbf{n}} \cdot F_N^{(e)} d\Gamma_e + \int_{\Gamma_e} \psi_i \hat{\mathbf{n}} \cdot F_N^{(*)} d\Gamma_e = 0$$

- Combining the boundary integrals yields the Strong form:

$$\int_{\Omega_e} \psi_i \frac{\partial q_N^{(e)}}{\partial t} d\Omega_e + \int_{\Omega_e} \psi_i \cdot \nabla F_N^{(e)} d\Omega_e + \int_{\Gamma_e} \psi_i \hat{\mathbf{n}} \cdot \left[F_N^{(*)} - F^{(e)} \right] d\Gamma_e = 0$$

1. Unified CGDG: Convergence

Convergence for the 1D Wave Equation after one revolution

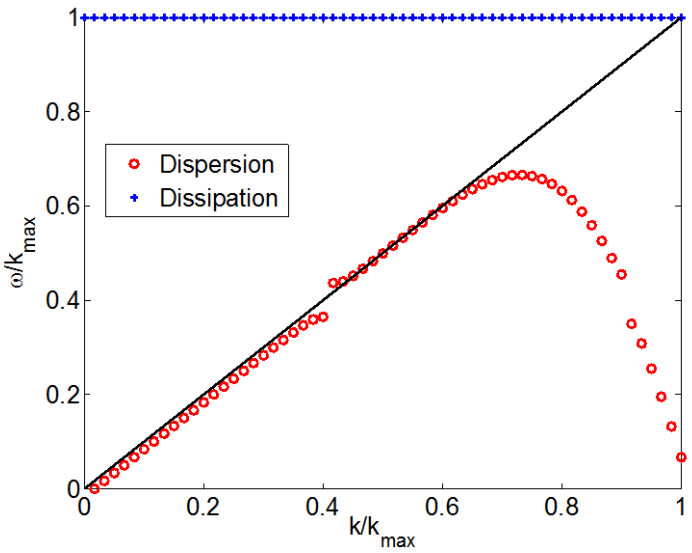


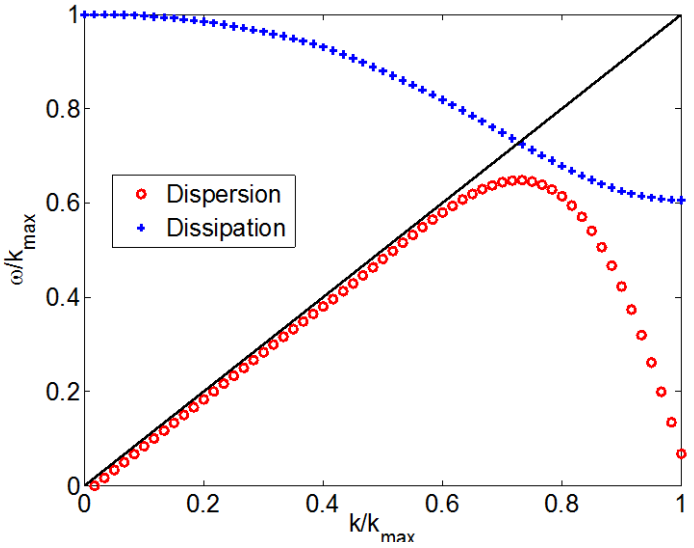
1. Unified CGDG: Dispersion Properties (N=4)

No Dissipation

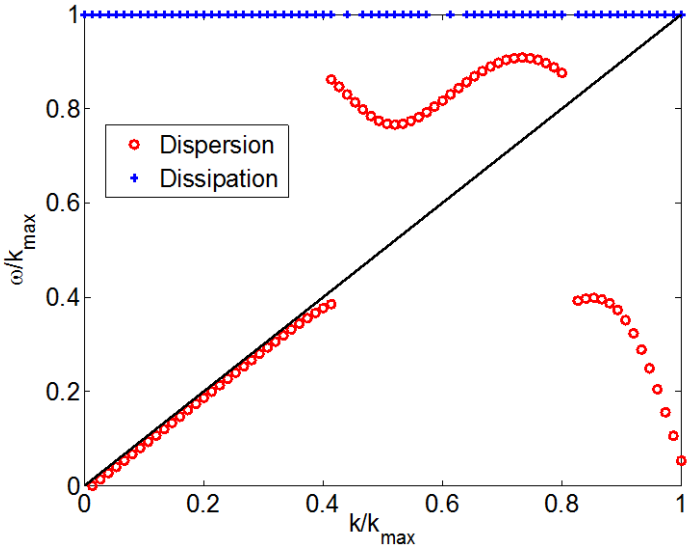
With Dissipation

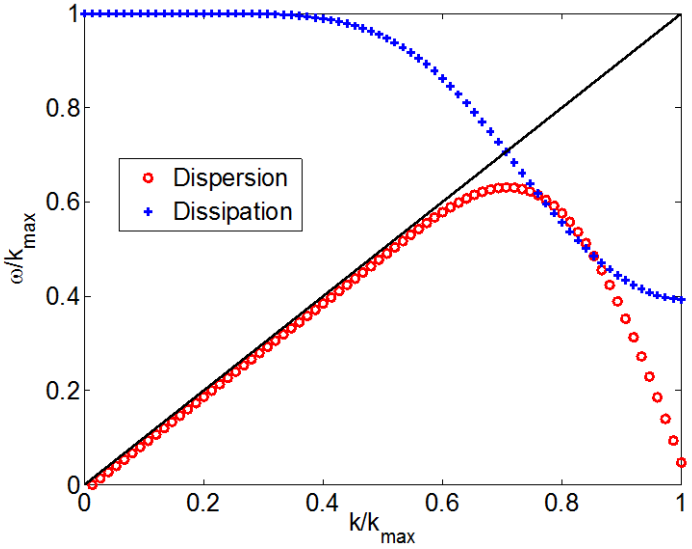
CG





DG





1. Unified CGDG: Vices and Virtues

CG

- High-order and local
- Requires inexact integration for efficiency (no global M matrix)
- Easy to construct MPI code
- Not efficient if non-tensor product basis functions are used (i.e., triangles, tri-prisms, tetrahedra)
- Easy to use with conforming adaptivity
- Tricky to use with non-conforming adaptivity
- Tricky to use P-adaptivity
- Easy to construct IMEX (Schur Complement) formulations
- Unclear how to make positivity preserving (VMS is best candidate)

DG

- High-order and local
- Can use exact integration for (block diagonal global M matrix)
- Very easy to construct MPI code
- Can be used efficienctly with tris, tri-prisms, tetrahedra and also quads, hexahedra, etc.
- Easy to use with conforming adaptivity
- Easy to use with non-conforming adaptivity
- Easty to use P-adaptivity
- Not easy to construct IMEX (Schur Complement) formulations
- There exist machinery for positivity preserving (numerical fluxes and limiters)

1. Benefits of Unified CGDG

- Allows for 4 different possible solutions within the same code.
  - CG/DG and Strong/Weak forms.
- Allows for a consistent approach for implementing boundary conditions.
- Allows improvement of CG by using DG framework, including:
  ① Performance on massively parallel architectures
  ② Positivity preserving mechanisms
  ③ Permitting the implementation of non-conforming grids (simplifies AMR)
  ④ Paves the way for P-refinement for CG
- Allows improvement of DG by using CG result:
  ① Inclusion of IMEX methods to DG
  ② Derivation of Schur Complement for DG

Talk Summary

1. Unified formulation of CGDG Methods
2. Positivity Preservation
3. The NUMA Model
4. Adaptive Mesh Refinement
5. Parallelization
6. Examples with CGDG
7. Closing Remarks

2. Positivity Preservation

How do we enforce positivity (e.g., tracers) with CGDG?

- For DG, the obvious choice is limiters
- For CG, there are many (not-so-obvious) choices but artificial diffusion methods is a likely choice. To this end, we are exploring the general class known as Variational Multi-Scale (VMS) Methods.

2. Positivity Preservation: Limiters

- A straightforward approach to limiting is to use the Zhang-Shu Limiter as follows:

$$\tilde{q}_i^{(e)} = \bar{q}^{(e)} + \alpha\left(q_i^{(e)} - \bar{q}^{(e)}\right)$$

- Where the weight is defined as:

$$\alpha = \min\left(\frac{\bar{q}^{(e)}}{\bar{q}^{(e)} - q_{\min}^{(e)}}, 1\right)$$

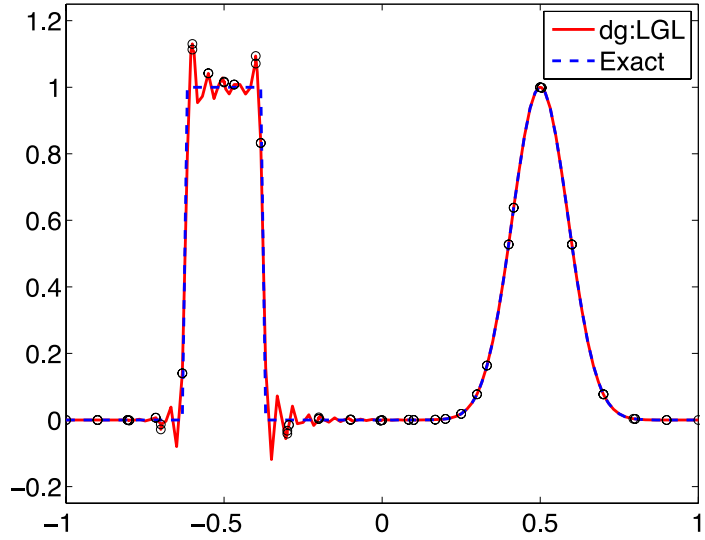
- Where

$$\bar{q}^{(e)} = \frac{1}{|\Omega_e|}\int_{\Omega_e} q_N^{(e)} d\Omega_e \equiv V_{0,j} q_j^{(e)}$$

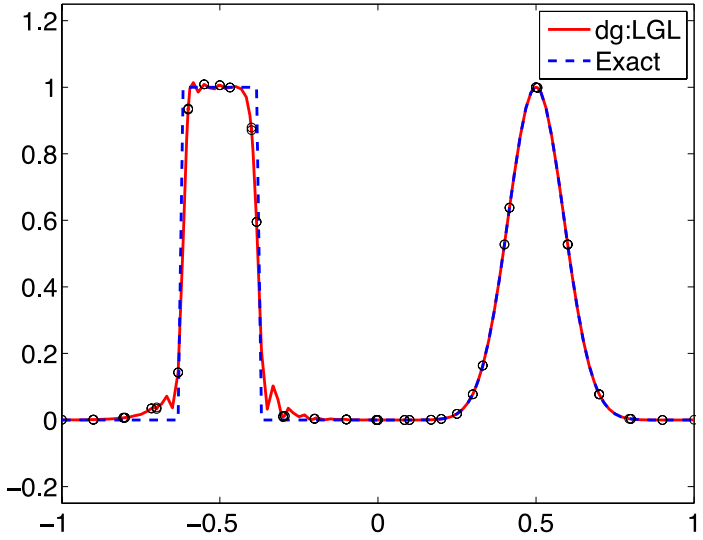
- and

$$q_{\min}^{(e)} = \min\left(q_i^{(e)}\right) \forall i = 0, \ldots, N$$

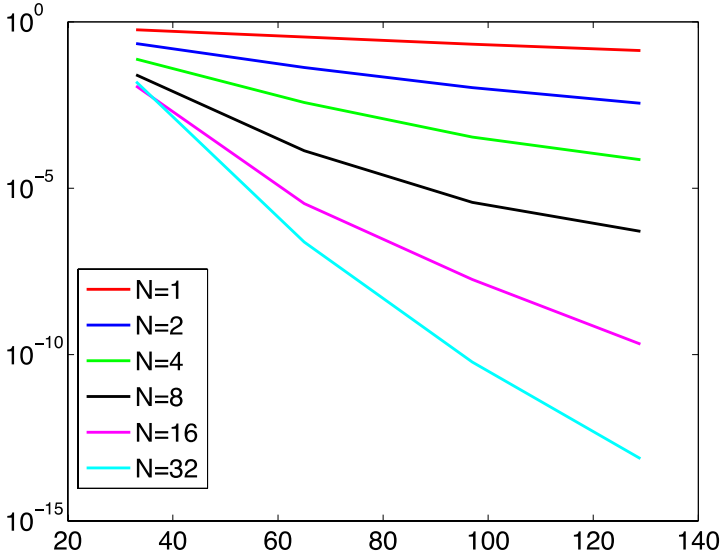
2. Positivity Preservation: Limiters

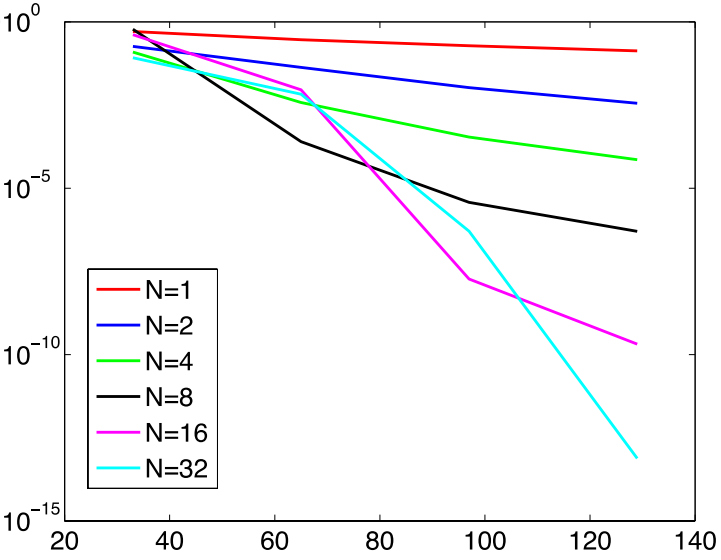


DG



DG with Limiter





2. Positivity Preservation: VMS

- To help explain VMS, let us write the general PDE

$$R \equiv \frac{\partial q}{\partial t} + L(q) - S(q) = 0$$

  - Where L(q) contains, e.g., the divergence of the flux tensor.

- The classical Galerkin formulation may be written as

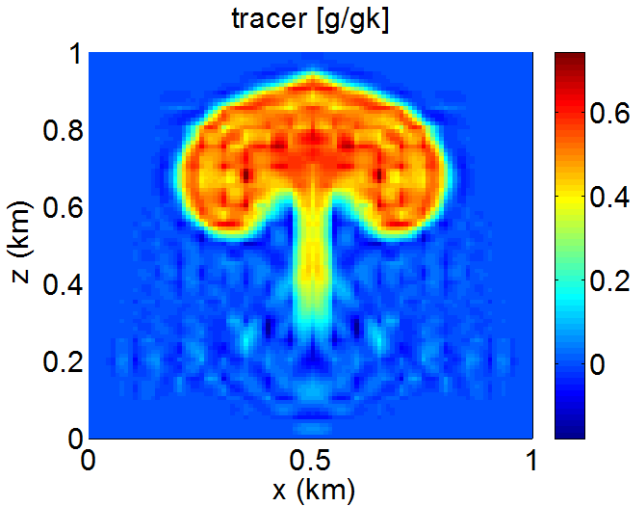
$$\left(\psi, \frac{\partial q}{\partial t}\right) + \left(\psi, L(q)\right) = \left(\psi, S(q)\right)$$

  - Where the inner product is defined as: $\left(u, v\right) = \bigcup_{e=1}^{N_e} \int_{\Omega_e} uv \, d\Omega_e$

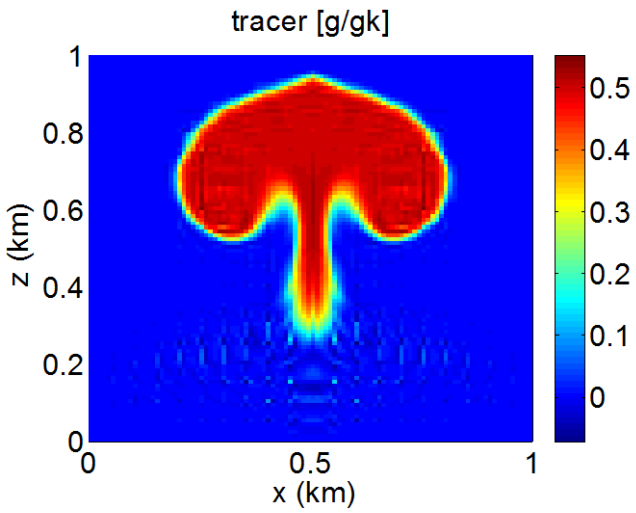
- With the VMS stabilization requiring an additional term

$$\left(\psi, \frac{\partial q}{\partial t}\right) + \left(\psi, L(q)\right) = \left(\psi, S(q)\right) + \left(R, \tau L^*(\psi)\right)$$

2. Positivity Preservation: VMS

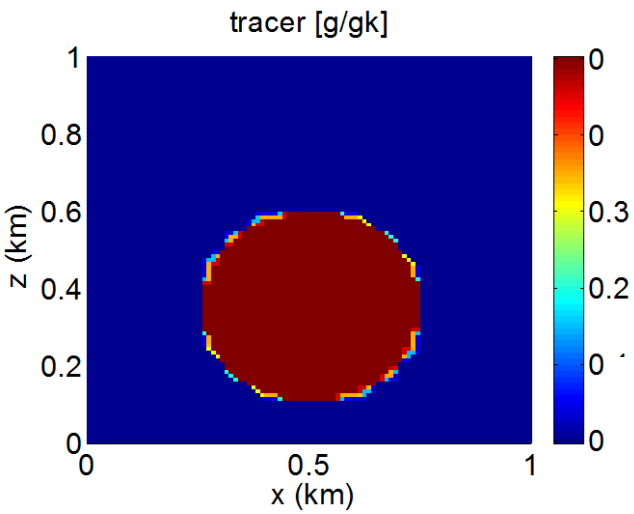


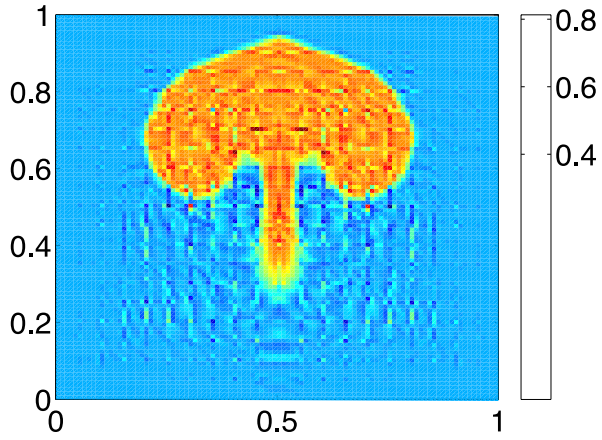
Filter



VMS

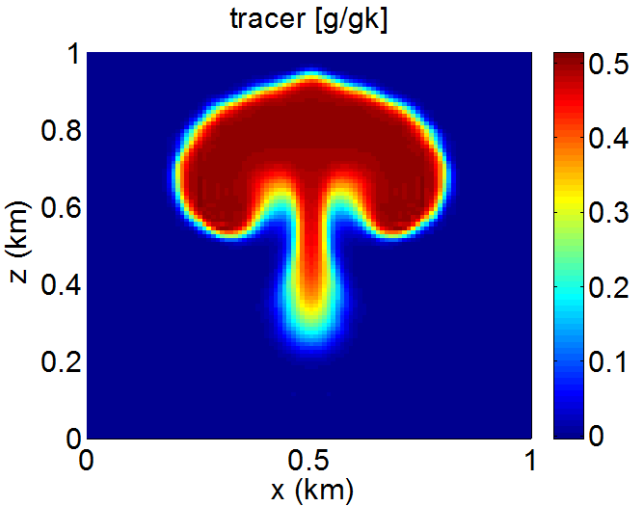
Initial Condition





Hyper Diffusion

VMS-DC



Talk Summary

1. Unified formulation of CGDG Methods
2. Positivity Preservation
3. The NUMA Model
4. Adaptive Mesh Refinement
5. Parallelization
6. Examples with CGDG
7. Closing Remarks

3. The NUMA Model

- NUMA=Nonhydrostatic Unified Model of the Atmosphere that can be used for both global and regional modeling.

- NUMA is based on CGDG Methods with a suite of IMEX time-integrators and Adaptive Mesh Refinement.

- NUMA is comprised of a collection of models that include modules for:

① NUMA3d is the nonhydrostatic atmospheric model (Euler equations)

② NUMACOM is the coastal ocean model (shallow water equations with an Inundation model)

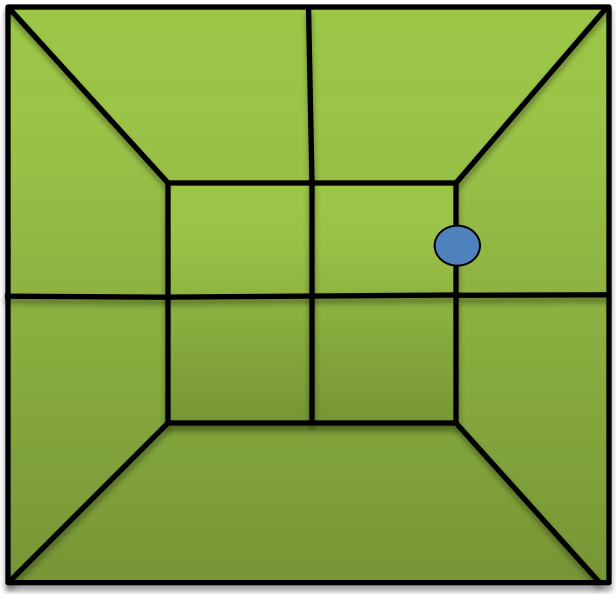
③ NUMACOM_SPHERE is the spherical version of NUMACOM

Talk Summary

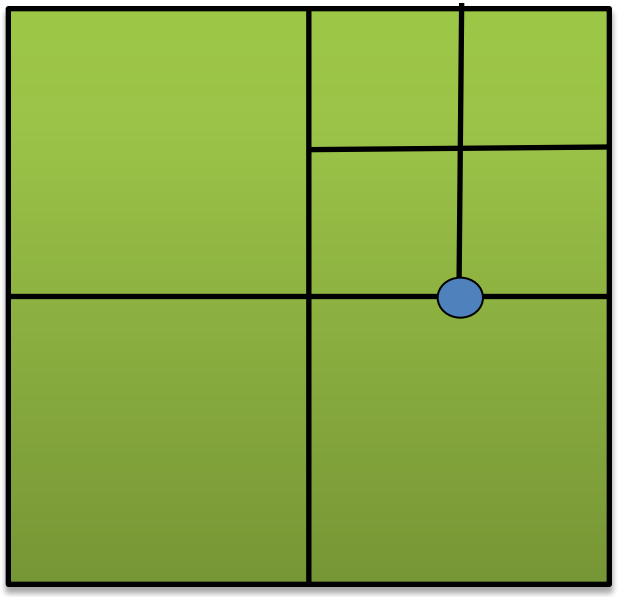
1. Unified formulation of CGDG Methods
2. Positivity Preservation
3. The NUMA Model
4. Adaptive Mesh Refinement
5. Parallelization
6. Examples with CGDG
7. Closing Remarks

4. Adaptive Mesh Refinement

Conforming Grid



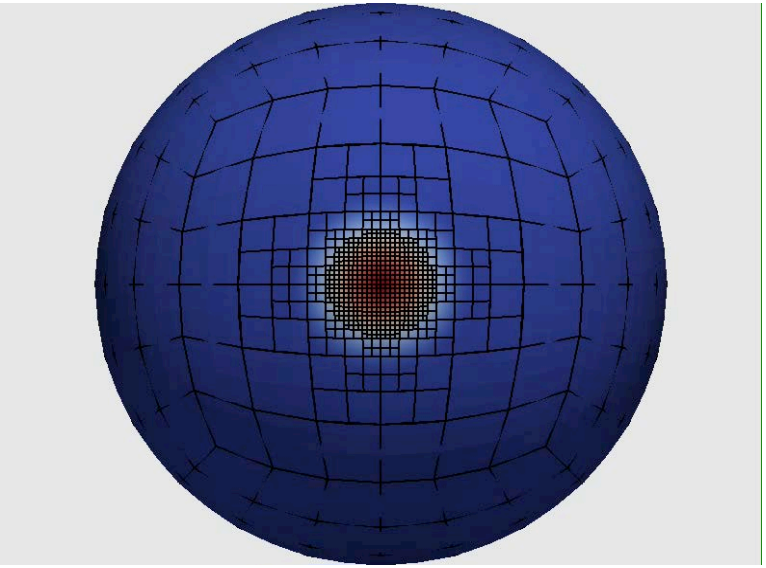
Non-Conforming Grid

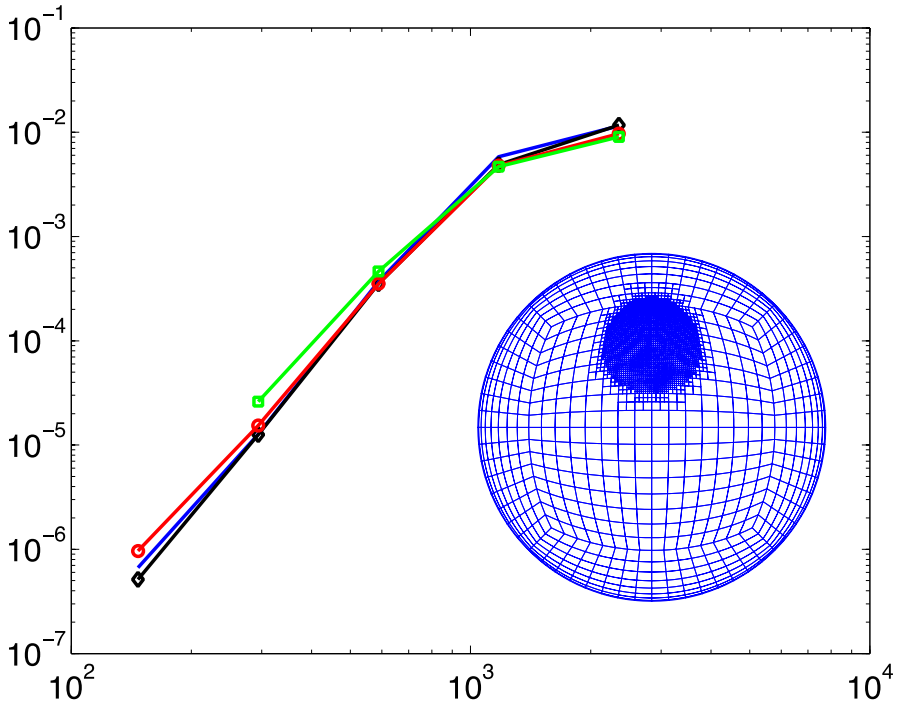


A grid point on an interface can be owned by more than two control volumes. This kind of adaptive method places MUCH OF THE BURDEN on the solver as it now needs to be able to handle non-conforming grids but greatly simplifies the adaptivity process. This kind of grid can produce very efficient adaptive methods and is the idea used in various AMR (e.g., M. Berger, P. Colella, etc.)

4. Adaptive Mesh Refinement

Case 1: Passive Advection





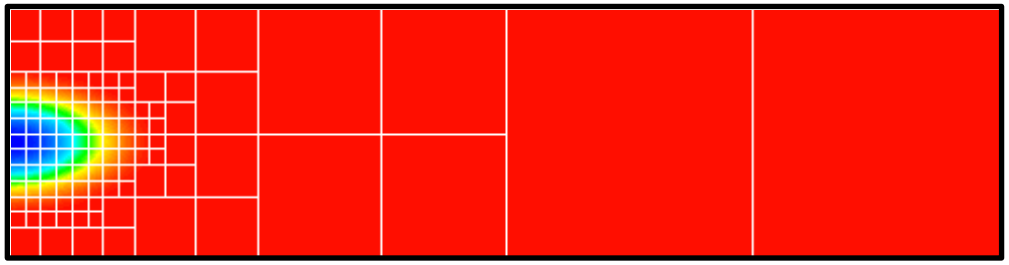
CGDG Shallow Water Model on the Sphere

4. Adaptive Mesh Refinement

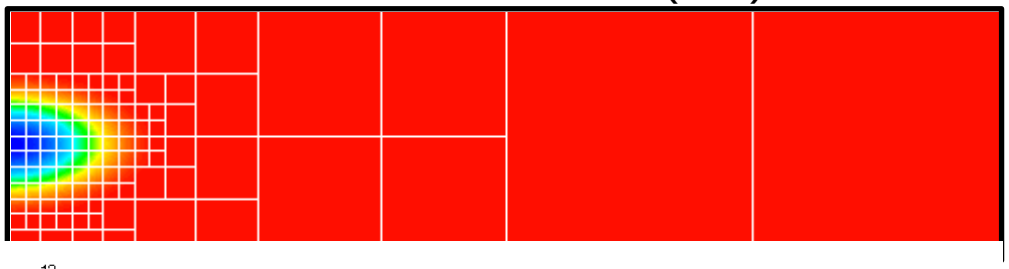
Dynamic AMR

Test case: **DENSITY CURRENT**
Base mesh: **4x1 elements**
Polynomial order: **10**
Max. AMR level: **4**

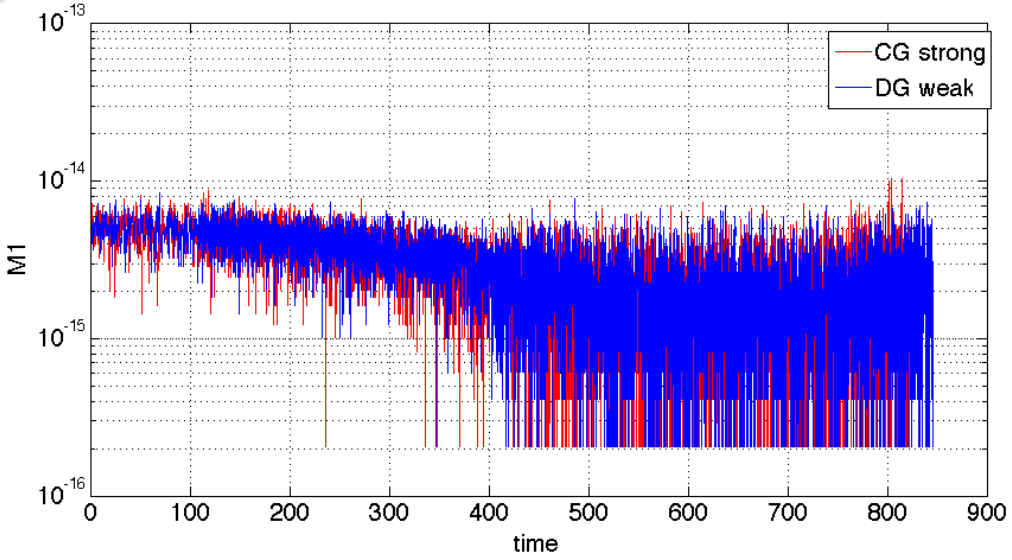
Discontinuous Galerkin (**DG**)



Continuous Galerkin (**CG**)

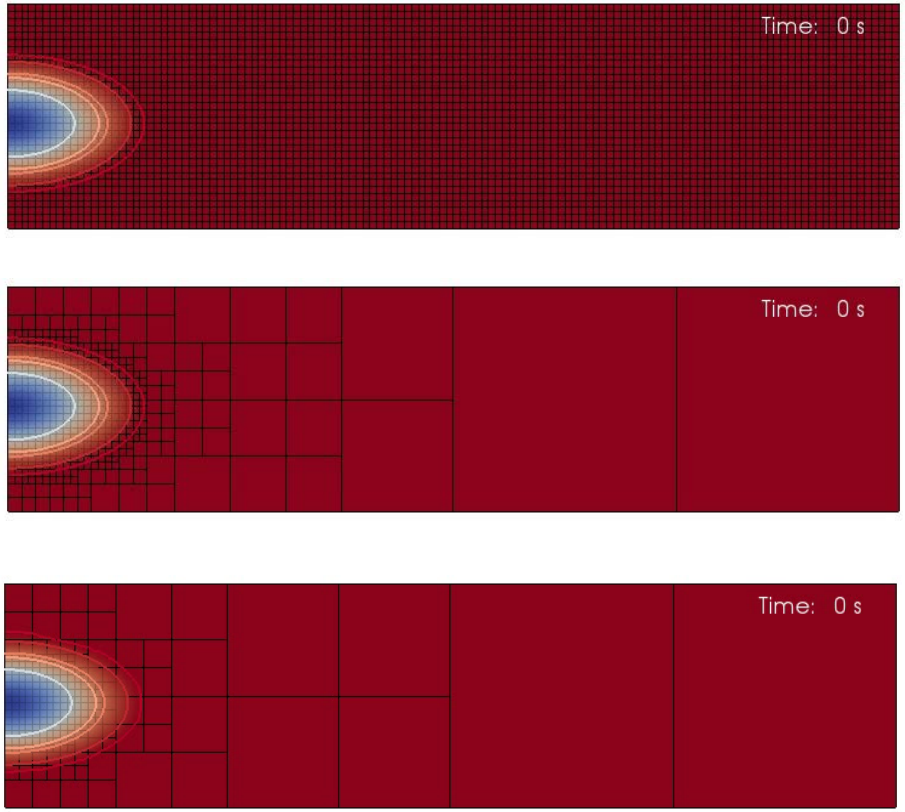


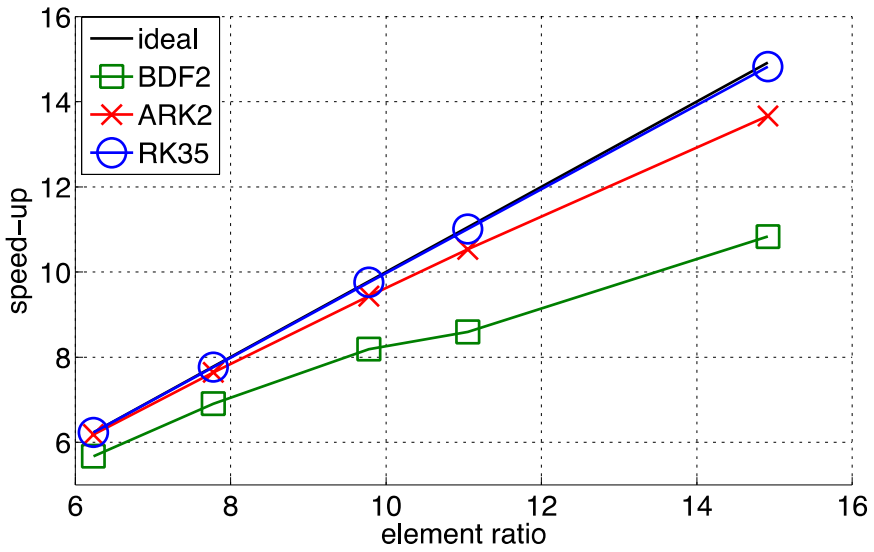
MASS CONSERVATION



Kopera-Giraldo [JCP 2013]

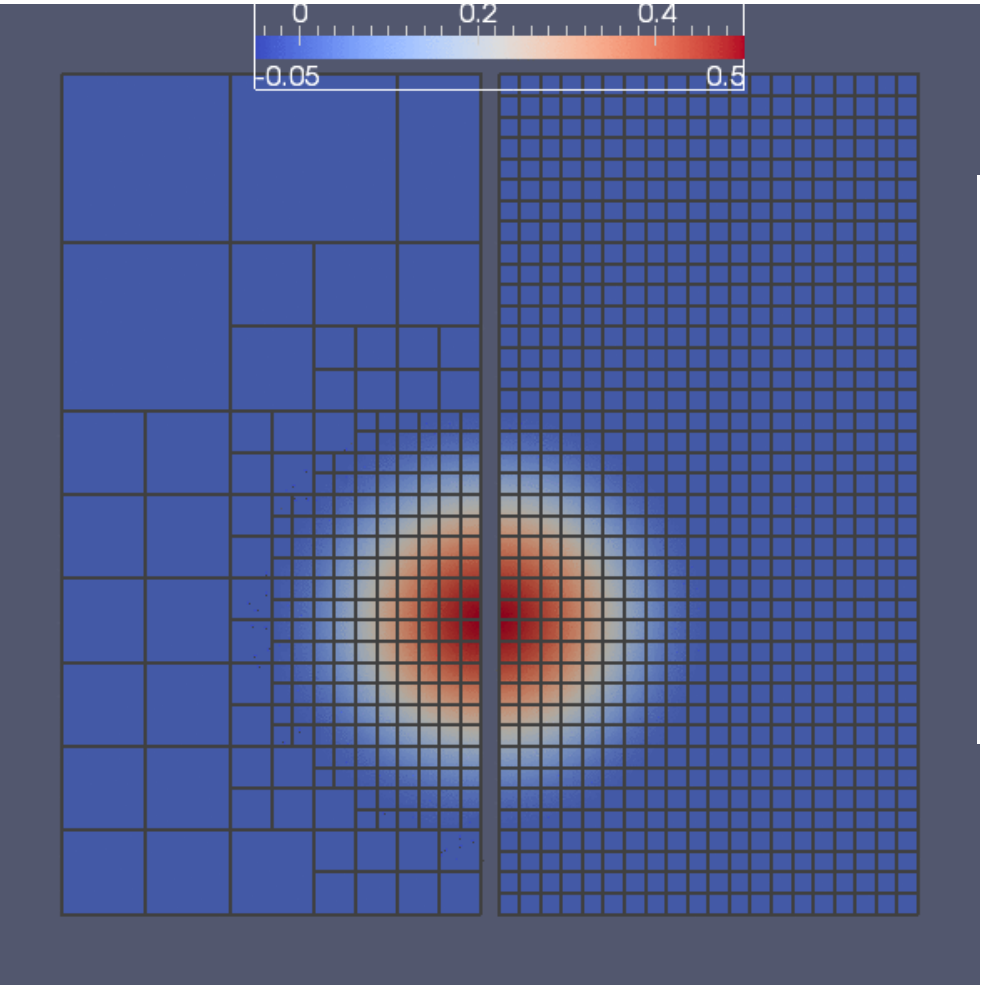
4. Adaptive Mesh Refinement

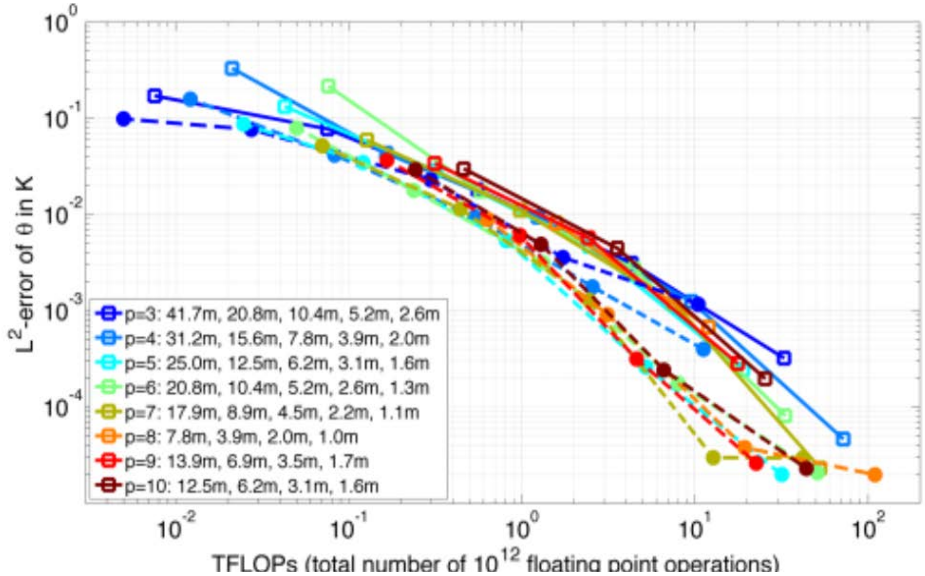




Kopera-Giraldo [JCP 2013]

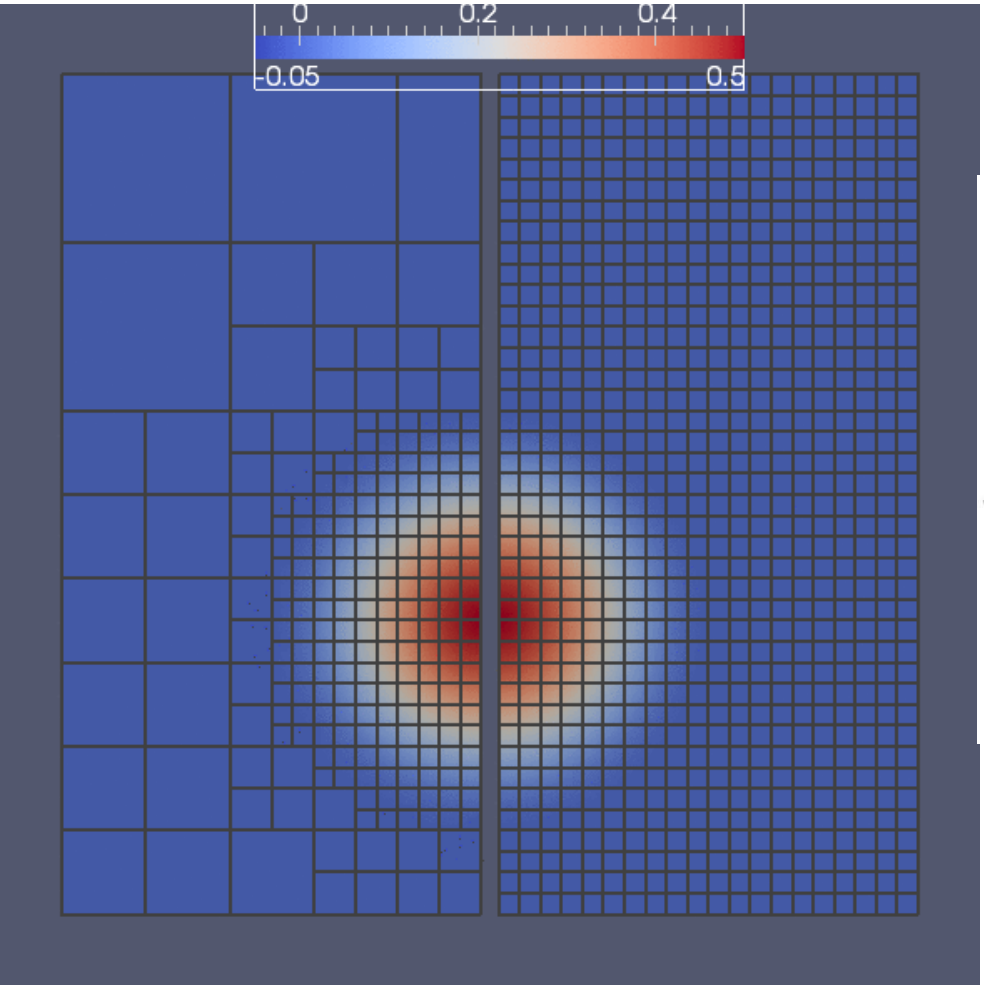
4. Adaptive Mesh Refinement

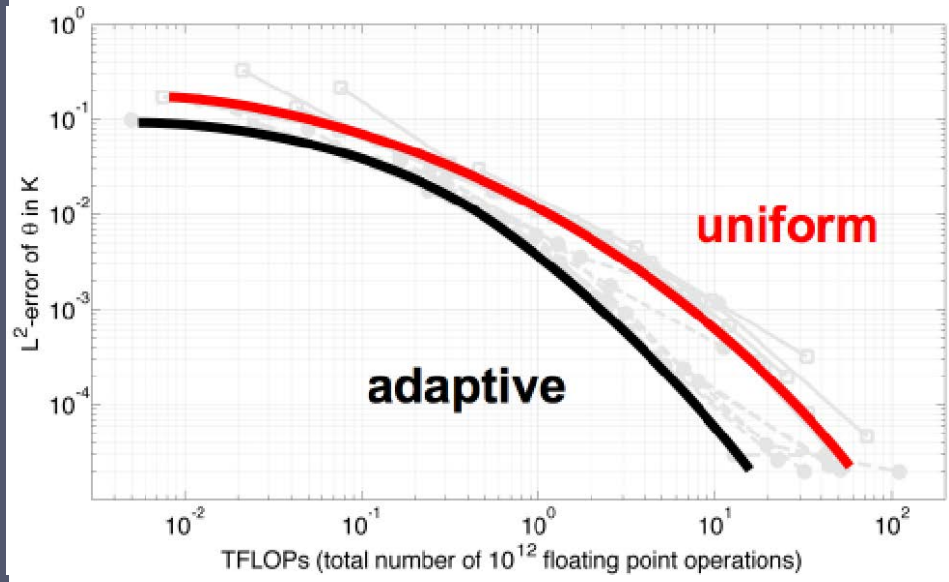




Mueller-Kopera-Giraldo [JCP 2013]

4. Adaptive Mesh Refinement

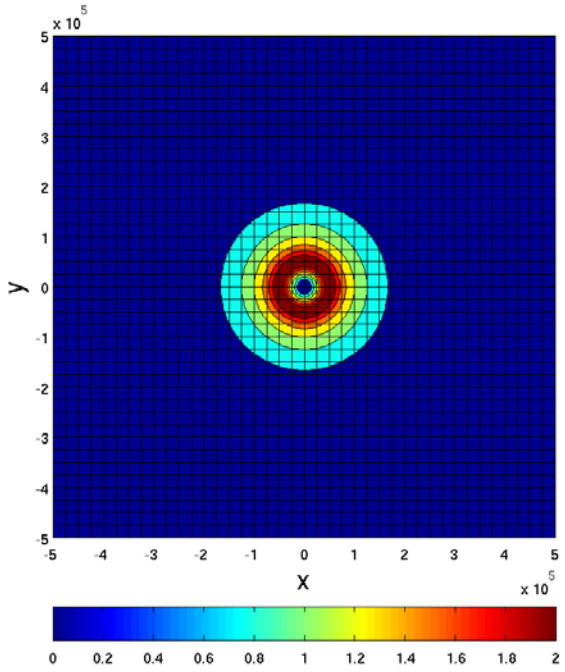




Mueller-Kopera-Giraldo [JCP 2013]

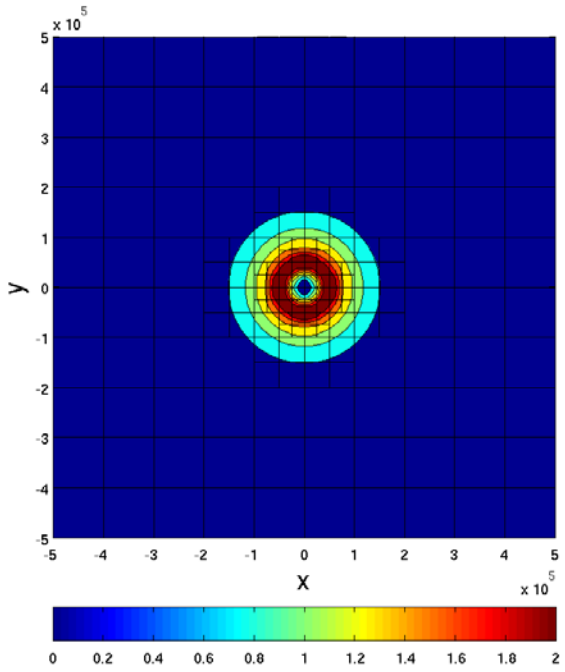
4. Adaptive Mesh Refinement

Wind Speed (m/s), NO AMR



CPU time: 9.11 h

Wind Speed (m/s), AMR



CPU time: 1.45 h

8th order polynomials, adapt mesh to potential vorticity

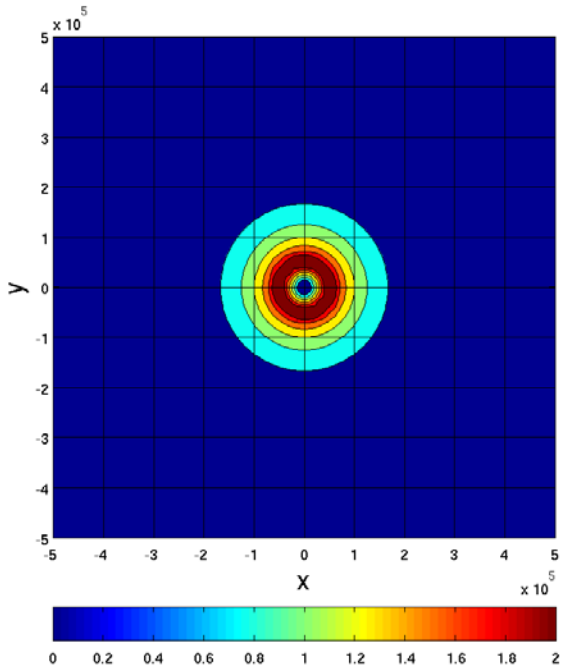
Advecting vortex in 10 m/s x-aligned flow, doubly periodic BCs, 1000 km domain

With same resolution over vortex, AMR simulation is over 6 times as fast NO AMR

Courtesy of Eric Hendricks (NRL)

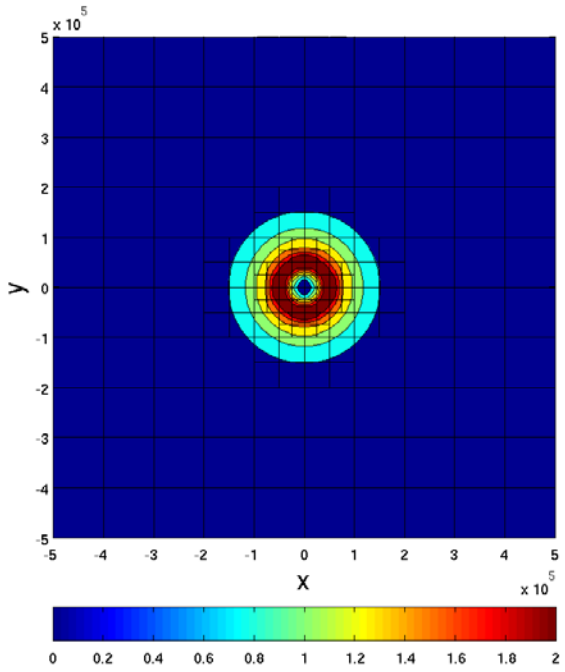
4. Adaptive Mesh Refinement

Wind Speed (m/s), NO AMR



CPU time: 0.44 h

Wind Speed (m/s), AMR



CPU time: 1.45 h

8th order polynomials, adapt mesh to potential vorticity

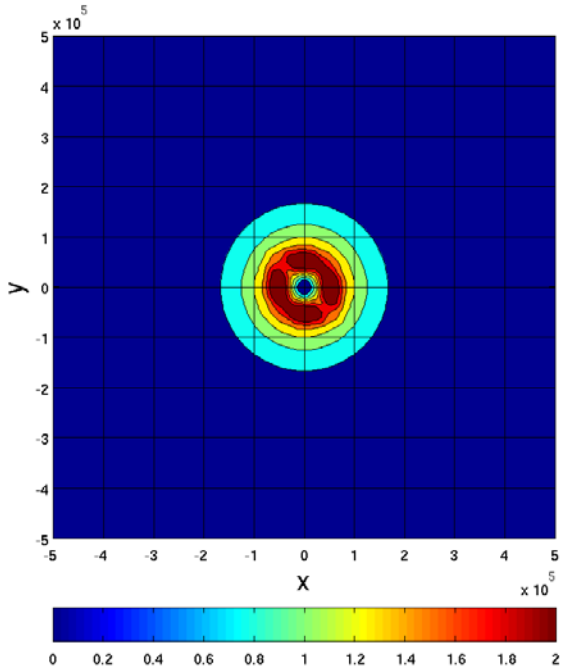
Advecting vortex in 10 m/s x-aligned flow, doubly periodic BCs, 1000 km domain

With coarse resolution over domain, NO AMR is faster than AMR, but vortex is not as well resolved

Courtesy of Eric Hendricks (NRL)

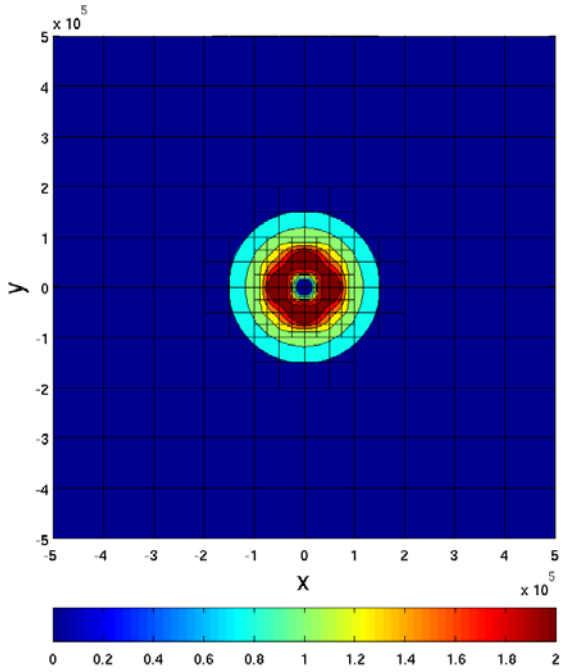
4. Adaptive Mesh Refinement

Wind Speed (m/s), NO AMR



CPU time: 0.11 h

Wind Speed (m/s), AMR



CPU time: 0.41 h

4th order polynomials, adapt mesh to potential vorticity

Advecting vortex in 10 m/s x-aligned flow, doubly periodic BCs, 1000 km domain

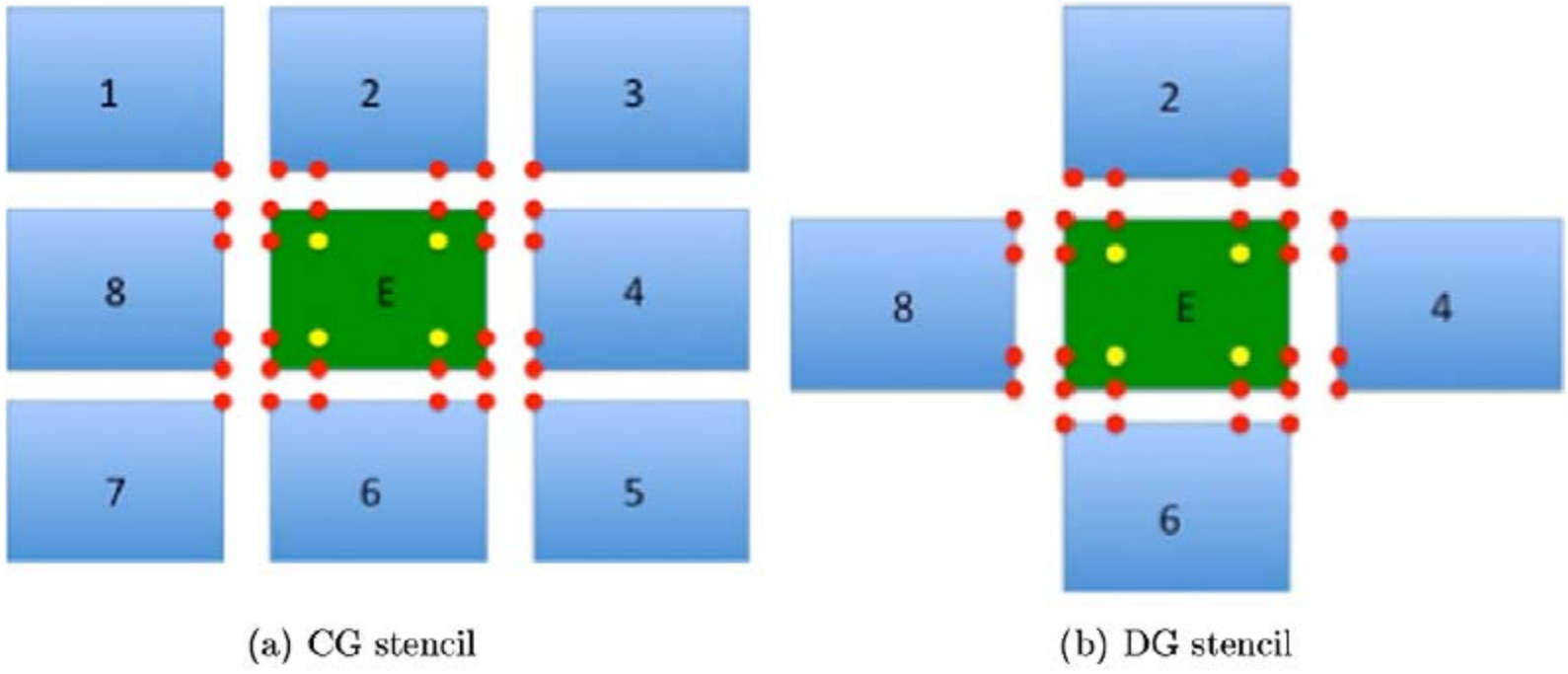
With coarse resolution over domain and lower polynomial order, NO AMR vortex becomes highly distorted, while AMR vortex does not.

Courtesy of Eric Hendricks (NRL)

Talk Summary

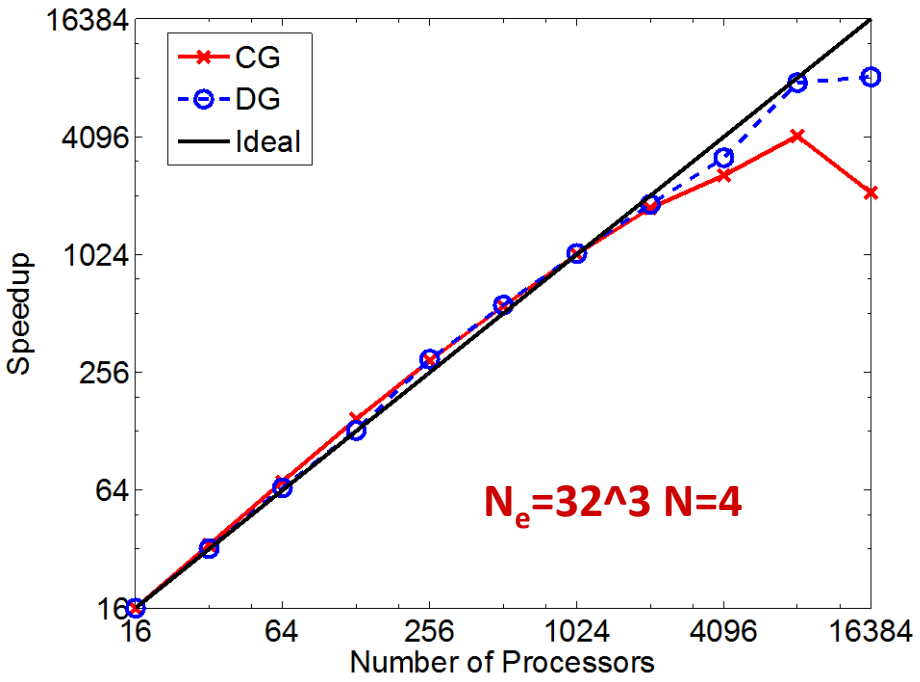
1. Unified formulation of CGDG Methods
2. Positivity Preservation
3. The NUMA Model
4. Adaptive Mesh Refinement
5. Parallelization
6. Examples with CGDG
7. Closing Remarks

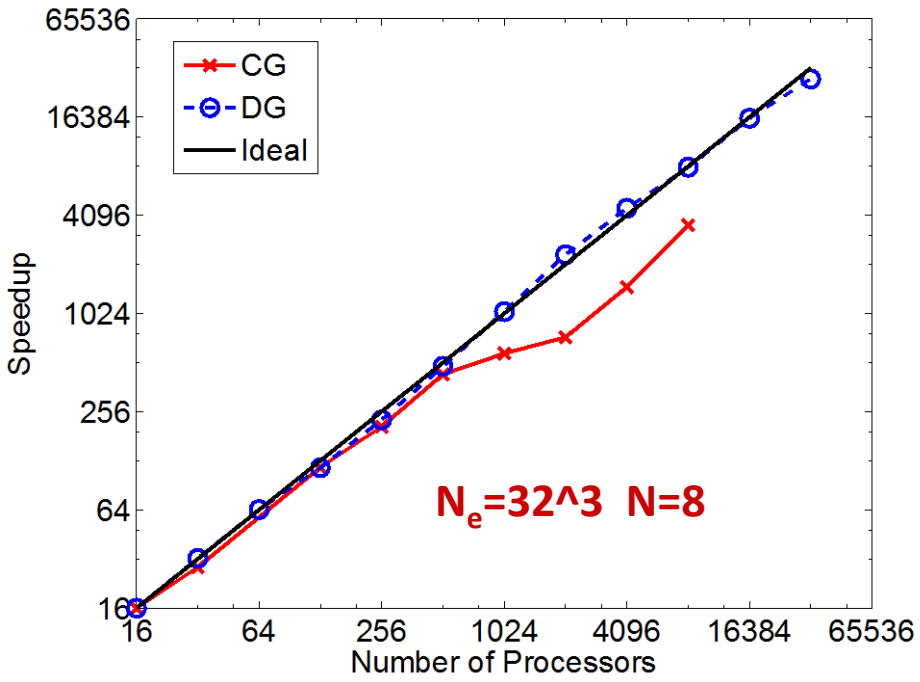
5. Parallelization: Communication Stencil



- On 2D (quads) CG requires information from 8 vertex neighbors whereas DG only requires information from its 4 face neighbors.
- On 3D (hexahedra) CG requires information from 26 vertex neighbors whereas DG only requires information from 6 face neighbors.

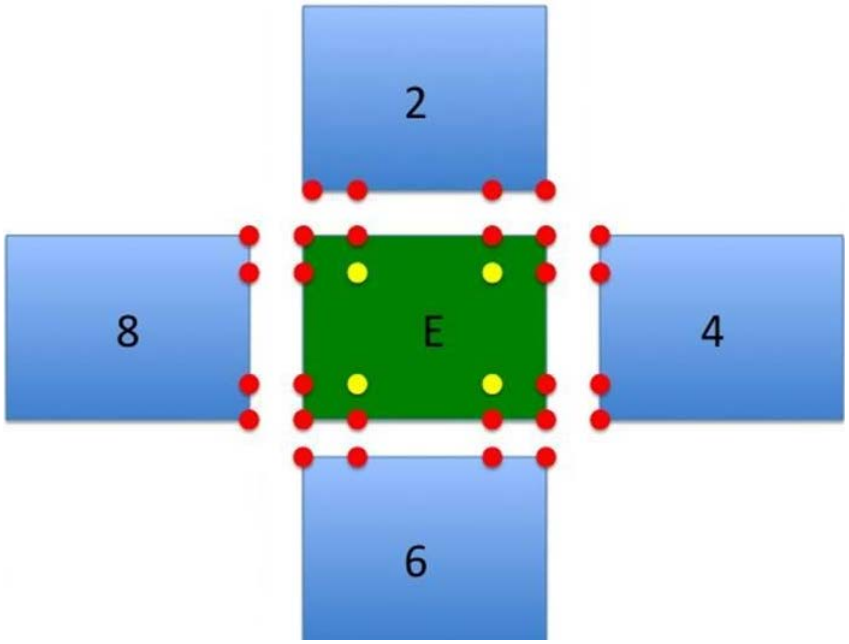
5. Parallelization: Scalability





- Local high-order CG and DG methods are known to be very accurate.
- CG and DG Methods scale impressively on massively parallel computers.
- Local high-order is beneficial for scalability because there is more on-processor work compared to the off-processor work (details can be found in Kelly-Giraldo JCP 2012)
- This plot is representative for explicit time-integration and implicit in the vertical.

5. Parallelization: DG Scalability



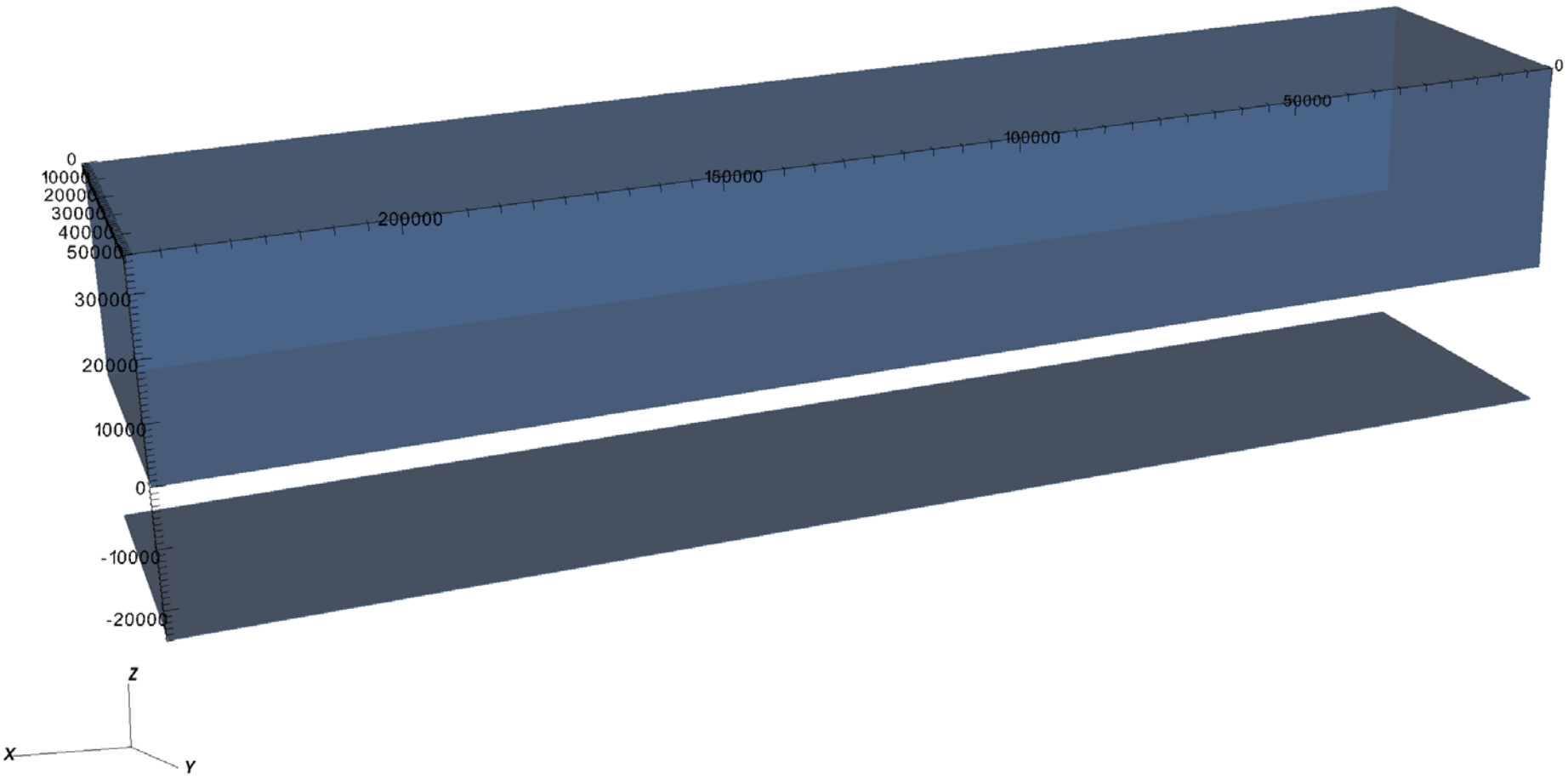
1. Send/Receive Boundary Data via non-blocking send/receive $O(N_e^{2/3}N^2)$. While waiting for boundary data…
2. Compute Volume integrals $O(N_eN^4)$.
3. Compute Intra-Processor fluxes.
4. MPI_Waitall()
5. Compute Inter-Processor fluxes.
6. Multiply by element-wise inverse mass matrix.

- The boundary calculation (fluxes) and interior calculation (volume integrals) are naturally decoupled. Hence DG is relatively easy to parallelize.
- R=Volume/Surface Area ratio = $O(N_e^{1/3} N^2)$. As N_e goes to 1, only for large N will there be sufficient work to maintain perfect scalability.
- On TACC Ranger we have found R=60 which means that N>=7 satisfies this.
- For IMEX, the behavior of DG is the same as in explicit for each iteration of an implicit solve.

Talk Summary

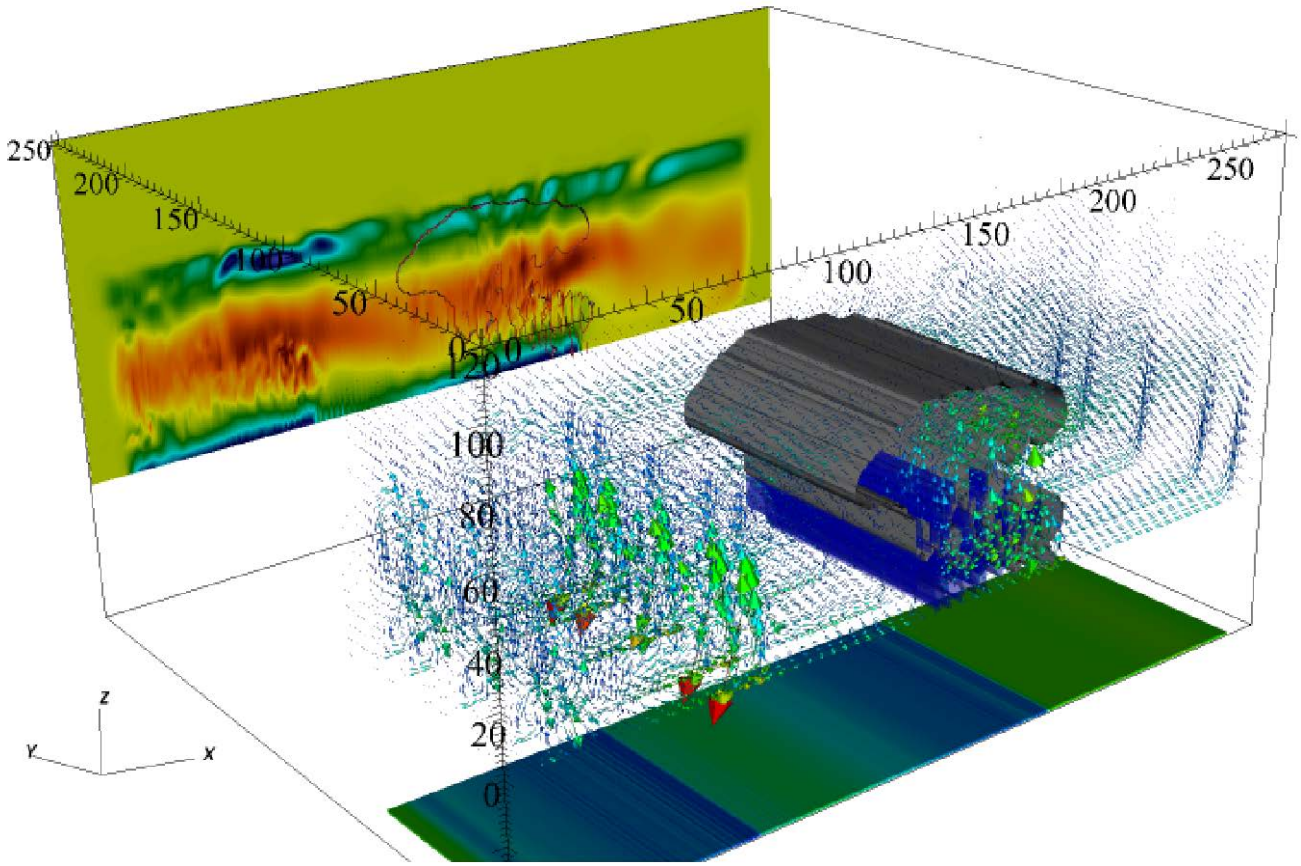
1. Unified formulation of CGDG Methods
2. Positivity Preservation
3. The NUMA Model
4. Adaptive Mesh Refinement
5. Parallelization
6. Examples with CGDG (NUMA)
7. Closing Remarks

NUMA: Moist Physics in Channel



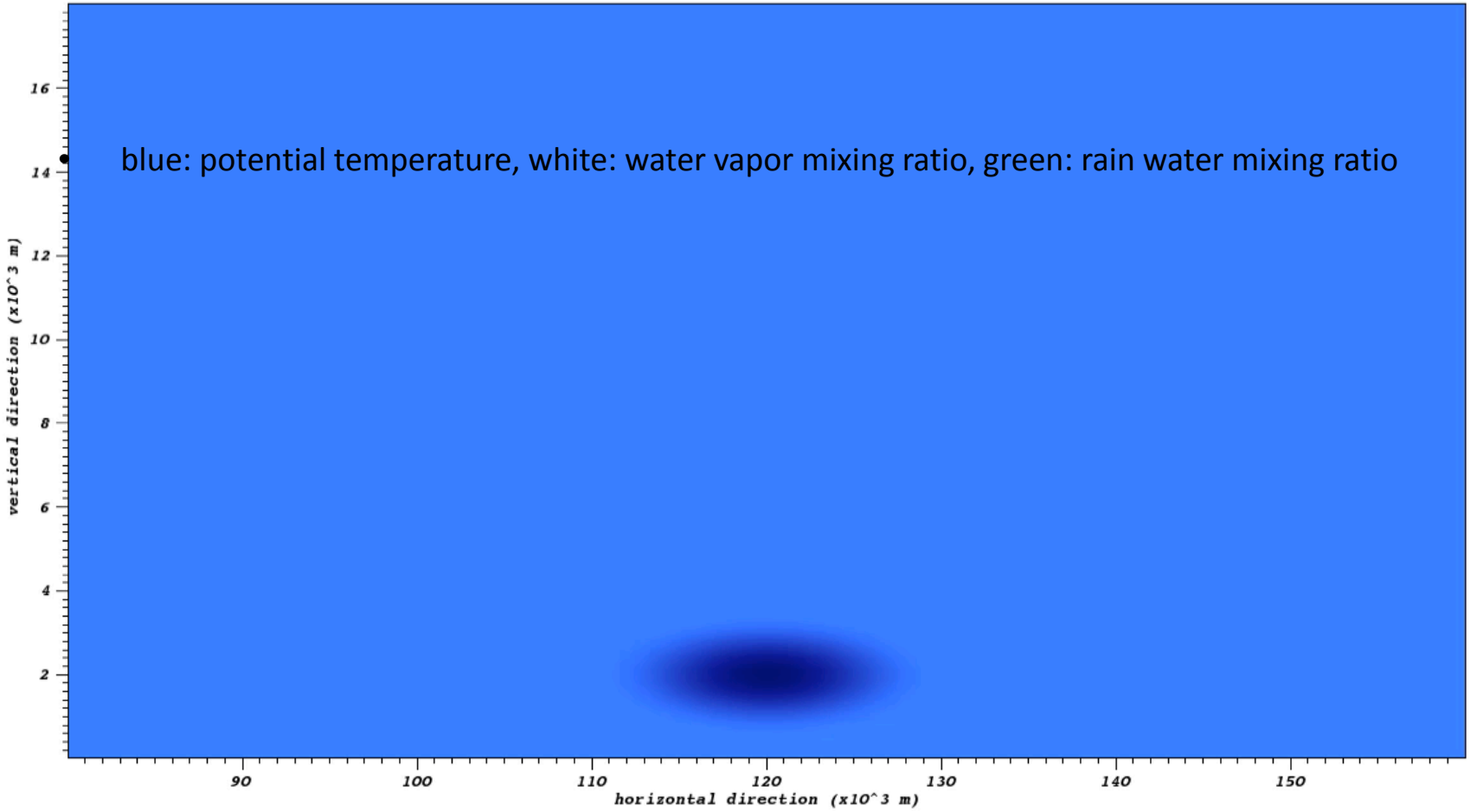
Courtesy of Simone Marras (NPS)

NUMA: Moist Physics in Channel



Courtesy of Simone Marras (NPS)

NUMA: Moist Physics (DG)



7. Closing Remarks

- CG or DG?
  - Both methods have their advantages
  - It is easy to carry both with little overhead
  - Comparison of CG and DG can be done fairly within the same model
  - For the moment CG is more efficient (wallclock time considerations only) due to the IMEX Schur complement (not discussed).
  - If HEVI is the only IMEX form desired then this advantage for CG vanishes.
  - We are running *Cost versus Accuracy* to determine which method wins in both static and dynamically adaptive mesh refinement.
  - In terms of dynamic AMR, DG is much more natural with nonconforming grids although it is possible to make CG work with nonconforming grids (NUMA has both). However, the computations required to get CG to conserve mass is not trivial (with nonconformity).
  - Performance on exascale computing hardware will always favor DG because of its data locality and the smaller communication stencil.
  - We are setting up to run massively parallel simulations to compare the scalability of both CG and DG on both CPU and GPU-based computers.
  - CG and DG can be combined quite easily within the same code – imagine regions that require IMEX time-integrators (use Schur form CG here) with areas that may require explicit time-integrators (use DG here).