Development of the Nonhydrostatic Unified Model of the Atmosphere (NUMA): Limited-Area Mode

James F. Kelly and Francis X. Giraldo

Department of Applied Mathematics, Naval Postgraduate School, Monterey, CA

Abstract

This paper describes a Nonhydrostatic Unified Model of the Atmosphere (NUMA) based on a spectral element, or high-order continuous Galerkin (CG) spatial discretization utilizing 3D hexahedral elements. The nonhydrostatic dynamical core, based on the compressible Euler equations, is appropriate for both limited-area and global atmospheric simulations. In this paper, we restrict our attention to 3D limited-area phenomena; global atmospheric simulations will be presented in a follow-up paper. A suite of explicit and semi-implicit time-integrators is presented. Domain decomposition and communication algorithms utilized by our distributed memory implementation is presented, allowing efficient evaluation of the the direct stiffness summation (DSS) operator. Numerical verification of the model is performed using four test cases: 1) 2D inertia-gravity waves, 2) flow past a 3D linear hydrostatic mountain, 3) flow past a 3D nonlinear mountain and 4) 3D buoyant convection of a bubble in a neutral atmosphere; these tests indicate that NUMA can simulate the necessary physics of a dry numerical weather prediction dynamical core. Scalability for the explicit dynamical core is demonstrated for 12288 cores on TACC's Ranger cluster, while the semi-implicit core is shown to scale to 4096 cores on the same architecture.

Keywords: compressible flow, Euler, Lagrange, Legendre, Navier-Stokes, nonhydrostatic, parallelization.

1. Introduction

As the resolution of numerical weather prediction (NWP) models increase, nonhydrostatic effects become relevant. Nonhydrostatic dynamical cores allow grid resolutions ranging from a few hundred kilometers (mesoscale) to

several kilometers (microscale) to approximately 100 meters (large eddy simulations). Almost all current limited-area models utilize a nonhydrostatic core, while global models are currently transitioning from the hydrostatic to the nonhydrostatic regime. A host of challenging, non-trivial numerical problems arise when one enters the nonhydrostatic regime: these include 1) choosing the appropriate equation set to ensure efficiency, accuracy, and conservation, 2) effectively resolving multi-scale flow features, that may require adaptive mesh refinement (AMR), 3) developing efficient time-integrators and/or "soundproof" equation sets to confront the fast acoustic and gravity waves present in non-hydrostatic models, and 4) developing scalable parallel codes for shared, distributed, and hybrid architectures based on 1) -3).

Virtually all current nonhydrostatic NWP models are based on a combination of finite-difference discretization in space and either split-explicit or semi-implicit discretization in time. Examples include WRF [24] (NCAR), Lokal Modell [30] (DWD), COAMPS [18] (US Navy), and UM [2] (UK Met Office). Although finite difference schemes are very efficient, they suffer from several problems, including: 1) dispersion error, 2) geometrical inflexibility, and 3) lacking of scalability to large (e.g. tens of thousands) of processors. To overcome some of the limitations of finite-difference approximations, several emerging NWP models utilize finite-volume spatial discretization, such as MPAS [38] and MCORE [40], which utilize polynomial reconstruction of the inter-element fluxes. To overcome these limitations, element-based Galerkin (EBG) methods have recently been proposed for several next generation NWP models [12, 14, 28] and [5] using both the continuous Galerkin (CG) and discontinuous Galerkin (DG) formulations.

EBG methods possess several desirable attributes, such as 1) higher-order accuracy, 2) geometrical flexibility, whereby the solver is completely independent of the grid, 3) excellent dispersion properties [26] and 4) minimal communication overhead within a parallel implementation. High-order accuracy, which allows fine-scale atmospheric flow features to be resolved, is achieved by representing the prognostic variables via an orthogonal basis function expansion within each element. Geometrical flexibility, which is inherent to all EBGs (both low and high order), is advantageous since any terrain-following coordinate may be utilized within the same solver; also, both static and dynamic adaptivity may be retro-fitted to the existing solver. Finally, in a distributed memory environment (e.g. cluster), low communication overhead is critical to maintain linear scalability to hundreds of thousands of processor cores. In our previous work, higher-order accuracy and geometrical flex-

ibility were addressed within an explicit framework [12] and semi-implicit framework [14] for 2D (x-z slices) problems using a serial implementation. However, parallel implementation was not explicitly addressed. The purpose of the present work is to extend the work begun in [12] and [14] to realistic 3D domains using a parallel, MPI-based implementation. Although we restrict our attention to limited-area (mesoscale and large-eddy scale) simulations, the dynamical core developed in this paper may also be applied to global atmospheric simulations. Hence, we are considering a Nonhydrostatic Unified Model of the Atmosphere (NUMA) that is being developed in conjunction with the Naval Research Laboratory (NRL) in Monterey, CA; to our knowledge, this is the first 3D spectral element model for a nonhydrostatic atmosphere.

This work is guided by a need for highly scalable models in distributed memory environments. In the past five years, clock speeds of processors have remained stagnant; to achieve increased floating-point performance, chip manufacturers have developed multiple core processors that allow many threads to execute in parallel. In tandem, high performance computing (HPC) has evolved towards clusters with processors counts exceeding 100,000. As we approach the exaflop era, core counts are expected to approach 1,000,000. Therefore, next-generation NWP models must be based upon scalable numerical methods that allow arbitrarily large processor counts with minimal communication overhead. EBG methods, both CG and DG, have proved effective in this respect in modeling massive biological flow [15], shallow water flows [7], incompressible flows using low-order finite elements [19], and geodynamical problems [41].

This paper presents a scalable, 3D nonhydrostatic spectral element atmospheric model targeted toward distributed memory architectures. We are developing a unified dynamical core appropriate for both mesoscale and global simulations. The current implementation utilizes tensor products of Lagrange polynomials in a hexahedral grid for maximum computational efficiency; however, more flexible grids based on either tetrahedra or triangular prisms may be incorporated into future versions with relative ease. The remainder of this paper is structured as follows. In Section 2, we formulate the nonhydrostatic compressible Euler equations (Set 2NC from [14]), which constitute the governing equations of our dynamical core. To ensure numerical stability, these nonhydrostatic equations are solved about a hydrostatic base state. In Section 3, we present the spectral element discretization, along with explicit and semi-implicit time-integration schemes and the nec-

essary boundary conditions. Section 4, which forms the core of the paper, outlines the parallelization algorithm for both the explicit and semi-implicit time-integrators. Numerical results for a 2D inertia-gravity wave, 3D linear hydrostatic and non-linear mountains, and a 3D rising thermal bubble are shown in Section 5 along with the results of scalability experiments. Scalability for the explicit code is demonstrated to 12288 processor cores for the explicit code and 4096 processor cores for the semi-implicit code.

2. Governing Equations

We consider the fully compressible, nonhydrostatic Euler equations in non-conservative form. These equations (Set 2NC), which are valid for spatial resolutions finer than 10 km, have previously been considered within a semi-implicit framework in [14] for 2D, limited-area atmospheric flows. Of the five equation sets considered in [14], set 2NC proved to be both computationally efficient and provides acceptable mass and energy conservation properties; in addition, replacing the advection operator in the momentum equation allows for formal conservation of energy up to time truncation error. Both diabatic forcing and the effects of moisture are neglected; in other words, we consider a *dry* dynamical core without sub-grid scale turbulence closure. In the present study, we consider three-dimensional flow in Cartesian coordinates (x-y-z) subject to gravitational and Coriolis forces, yielding

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{1a}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \frac{1}{\rho} \nabla P + g\hat{\mathbf{k}} + \mathbf{f} \times \mathbf{u} = 0 \tag{1b}$$

$$\frac{\partial \theta}{\partial t} + \mathbf{u} \cdot \nabla \theta = 0 \,. \tag{1c}$$

where the prognostic variables are $(\rho, \mathbf{u}^{\mathcal{T}}, \theta)$, where ρ is density, $\mathbf{u} = (u, v, w)^{\mathcal{T}}$ is velocity, and θ is potential temperature. In addition, P is pressure, g is the gravitational constant, $\mathbf{f} = 2\Omega\hat{\mathbf{k}}$ is a Coriolis parameter (with Ω the angular frequency of the earth), and $\hat{\mathbf{k}}$ is the unit vector in the z direction. Eq. (1a) enforces mass conservation, Eq. (1b) enforces conservation of momentum, and Eq. (1c) enforces conservation of entropy. To close the system of conservation laws given by Eq. (1), a thermodynamic equation of state is required.

We utilize the ideal gas law given by

$$P = P_A \left(\frac{\rho R \theta}{P_A} \right)^{\gamma} \tag{2}$$

where P_A is the atmospheric pressure at ground, R is the ideal gas constant, and $\gamma \approx 1.4$ is the ratio of specific heats. In three dimensions, Eqs. (1) and (2) constitute a closed system of nonlinear PDEs in five unknowns.

To facilitate the solution of the compressible Euler equations and maintain numerical stability, we split the density, pressure, and potential temperatures about their mean hydrostatic values:

$$\rho(x, z, y, t) = \rho_0(z) + \rho'(x, y, z, t) \tag{3a}$$

$$\theta(x, y, z, t) = \theta_0(z) + \theta'(x, y, z, t) \tag{3b}$$

$$P(x, y, z, t) = P_0(z) + P'(x, y, z, t) \tag{3c}$$

where ρ_0, θ_0, and P_0 are the hydrostatic reference states that depend only on the vertical distance z. Inserting Eq. (3) into Eq. (1) and applying hydrostatic balance

$$\frac{dP_0}{dz} = -\rho_0 g \tag{4}$$

yields the system

$$\frac{\partial \rho'}{\partial t} + \mathbf{u} \cdot \nabla \rho' + w \frac{d\rho_0}{dz} + (\rho' + \rho_0) \nabla \cdot \mathbf{u} = 0 \tag{5a}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \frac{1}{\rho' + \rho_0} \nabla P' + \frac{\rho'}{\rho' + \rho_0} g \hat{\mathbf{k}} + \mathbf{f} \times \mathbf{u} = 0 \tag{5b}$$

$$\frac{\partial \theta'}{\partial t} + \mathbf{u} \cdot \nabla \theta' + w \frac{d\theta_0}{dz} = 0. \tag{5c}$$

Defining a solution vector $\mathbf{q} = (\rho', \mathbf{u}^{\mathcal{T}}, \theta')^{\mathcal{T}}$, Eq. (5) is written in condensed form as

$$\frac{\partial \mathbf{q}}{\partial t} = S(\mathbf{q}) \tag{6}$$

where the source term $S(\mathbf{q})$ is a nonlinear, first-order differential operator.

3. Numerical Methods

In this section, we briefly discuss the numerical discretization used by NUMA: 1) spatial discretization of Eq. (5) using a continuous Galerkin (CG), or spectral element method, 2) a suite of explicit and semi-implicit time-integrators, and 3) the necessary boundary conditions required for limited-area problems.

3.1. Spatial Discretization

The spectral element method decomposes the spatial domain $\Omega \subset \mathcal{R}^3$ into N_e disjoint elements Ω_e via

$$\Omega = \bigcup_{e=1}^{N_e} \Omega_e. \tag{7}$$

In the current formulation of NUMA, we let Ω_e be hexahedra, which provides 1) simple grid generation and 2) efficient (fast) evaluation of the necessary differentiation and integration operators. We note, however, that Ω_e may be replaced by either 1) tetrahedra or 2) triangular prisms in a future version of NUMA.

Letting the unit cube $(\xi, \eta, \zeta) \in E = [-1, 1]^3$ be the reference hexahedral element, a transformation $\mathcal{F}_e : \Omega_e \rightarrow E$ mapping physical space to computational space is defined for each element, yielding $(x, y, z) = \mathcal{F}_e(\xi, \eta, \zeta)$. Associated with $\mathcal{F}_e$ is a Jacobian J_e, that, in conjunction with the decomposition in Eq. (7) allows globally defined integrals. We express

$$\int_\Omega f(\mathbf{x})\, d\Omega = \bigwedge_{e=1}^{N_e} |J_e| f^{(e)}\left(\mathbf{x}_e\right)\, d\Omega_e \tag{8}$$

where $\bigwedge_{e=1}^{N_e}$ denotes the global assembly, or direct stiffness summation (DSS) operator and $|J_e|$ is the determinant of the Jacobian mapping. Eq. (8) forms the basis for all CG algorithms; moreover, Eq. (8) is responsible for the local nature of CG that facilitates parallelization. This aspect of CG will be explored in detail in Section 4.

Within each element Ω_e, a finite-dimensional approximation $\mathbf{q}_N$ is formed by expanding $\mathbf{q}(\mathbf{x}, t)$ in basis functions $\psi_j\left(\mathbf{x}\right)$ such that

$$\mathbf{q}_N(\mathbf{x}, t) = \sum_{j=1}^{M_N} q_j(t)\psi_j(\mathbf{x}) \tag{9}$$

where $M_N = (N+1)^3$ is the number of nodes per element and N is the order of the basis functions. The discrete solution $\mathbf{q}_N$ is assumed $C^0(\Omega)$ continuous across inter-element boundaries. For basis functions, we construct tensor products of Lagrange polynomials given by

$$\psi_i(\mathbf{x}) = h_\alpha(\xi) \otimes h_\beta(\eta) \otimes h_\gamma(\zeta) \tag{10}$$

where $h_\alpha(\xi)$ is the Lagrange polynomial associated with the Legendre-Gauss-Lobatto (LGL) points ξ_i and (ξ, η, ζ) are functions of the physical variable $\mathbf{x}$. These LGL points satisfy

$$(1 - \xi^2) P_N'(\xi) = 0 \tag{11}$$

where $P_N(\xi)$ is the N-th order Legendre polynomial. Approximating the prognostic vector $\mathbf{q}(\mathbf{x}, t)$ by a finite-dimensional approximation $\mathbf{q}_N$ in Eq. (9), multiplying by a basis function ψ_i and integrating over the domain Ω yields the weak form

$$\int_\Omega \psi_I \frac{\partial \mathbf{q}_N}{\partial t} \, d\Omega = \int_\Omega \psi_I S(\mathbf{q}_N) \, d\Omega \tag{12}$$

where we have replaced the index i with I to emphasize that Eq. (12) is now a global representation. Applying the Galerkin expansion given by Eq. (9) to Eq. (12) yields the matrix-vector equation

$$\frac{\partial q_I}{\partial t} = M_{IJ}^{-1} S(q_I) \tag{13}$$

where the mass-matrix $M_{IJ} = \int_\Omega \psi_I \psi_J \, d\Omega$ is diagonal if the interpolation and integration points are co-located. This approximation is valid for $N \geq 4$ while incurring a small error in integration [9]. Denoting the right-hand side (RHS) of Eq. (13) by $R_I(q_I)$, Eq. (13) is expressed as

$$\frac{\partial q_I}{\partial t} = \bigwedge_{e=1}^{N_e} R_i^{(e)} \left(q_i^{(e)} \right). \tag{14}$$

Note that the DSS operator maps local, element-wise coordinates i to global coordinates I. Eq. (14) forms the core of the spectral element method, allowing local, element-wise information $q_i^{(e)}$ to propagate to adjacent elements via the DSS operator. In Section 4, we discuss the efficient evaluation of Eq. (14) within a distributed memory architecture. Moreover, the choice of tensor product basis functions, coupled with inexact integration, allows the right-hand sides $S(\mathbf{q})$ to be evaluated with $\mathcal{O}(N_e N^4)$ work [6], thereby resulting in an efficient algorithm [6].

3.2. Temporal Discretization

In the following section, we discuss two schemes used to evolve Eq. (6) forward in time: an explicit Runge-Kutta RK-35 method and a semi-implicit BDF2 method. A fully-implicit, Jacobian-Free Newton Krylov (JFNK) time-integrator [25] and a family of semi-implicit RK methods were also implemented; however, we defer discussing these results and will report their relative strengths and weaknesses in future work.

3.2.1. Explicit RK methods

We implemented a strong stability preserving (SSP) Runge-Kutta third-order, five stage time-integrator proposed by [36]. This time-integrator is stable for Courant numbers of 1.3 or less.

3.2.2. Semi-Implicit BDF2 (Schur Form)

Semi-implicit, or IMEX schemes, decompose the operator $S(\mathbf{q})$ in Eq. (6) into two components: a linear term and and a nonlinear term. This decomposition exploits the underlying physics of the compressible Euler equations: namely, that the fast moving acoustic and gravity waves are linear, while the slower advective waves are nonlinear. Hence, the fast moving waves may be discretized implicitly, while the slower advective waves are discretized explicitly. Hence, the semi-implicit schemes allow much larger time-steps than explicit schemes, since the Courant number is only restricted by the advective dynamics. Semi-implicit discretizations of spectral-element atmospheric models was recently developed in [14]; a brief outline is provided in this section for completeness.

First, Eq. (6) is rewritten as follows:

$$\frac{\partial \mathbf{q}}{\partial t} = [S(\mathbf{q}) - \delta L(\mathbf{q})] + \delta L(\mathbf{q}) \tag{15}$$

where $L(\mathbf{q})$ is the linearized Euler operator specified in Appendix A of [14] and $\delta = 0,\ 1$ is a switch. The operator $L(\mathbf{q})$ is responsible for 1) acoustic, 2) barotropic gravity and 3) baroclinic gravity waves. By setting $\delta = 0$, we recover an explicit scheme, while for $\delta = 1$, we have the semi-implicit scheme. To discretize Eq. (15) in time, a general backward difference (BDF) formula is utilized, yielding

$$\mathbf{q}^{n+1} = \sum_{k=0}^{K-1} \alpha_k \mathbf{q}^{n-k} + \gamma \Delta t \sum_{k=0}^{K-1} \beta_k \left[S(\mathbf{q}^{n-k}) - \delta L(\mathbf{q}^{n-k}) \right] + \gamma \Delta t \delta L(\mathbf{q}^{n+1}) \tag{16}$$

where $1 \leq K \leq 6$ is the order of the time-integrator. Eq. (16) is rewritten in terms of the variables defined in Section 3 of [14] yielding

$$\mathbf{q}_{tt} + \lambda L\left(\mathbf{q}_{tt}\right) = \hat{\mathbf{q}} \tag{17}$$

where $\lambda = \delta\gamma\Delta t$. Eq. (17) is immediately realized as a system of linear equations $A\mathbf{x} = \mathbf{b}$, where the left-hand side (LHS) matrix A is given by $A = I + \lambda L$ and right-hand side (RHS) vector $\hat{\mathbf{q}}$. Examining Eq. (17) in conjunction with [14], the semi-implicit problem consists of the following three steps: 1) solve the explicit problem and form $\hat{\mathbf{q}}$, 2) solve the linear system given by Eq. (17) for the intermediate variable $\mathbf{q}_{tt}$, and 3) backsubstitute to determine the update $\mathbf{q}^{n+1}$.

The majority of the computational work is spent in solving Eq. (17) via an iterative Krylov subspace technique, such as GMRES [29]. Noting that this linear system is size $5N_g \times 5N_g$ for the 3D Euler equations, the cost of an iterative solve is $\mathcal{O}(25k^2N_g^2)$, where N_g is the number of global grid points and k is the number of Krylov iterations. Hence, the cost of solving Eq. (17) may become prohibitive even for relatively small 3D problems. To mitigate this problem, the size of the linear system may be reduced from $5N_g$ to N_g via the *Schur complement* approach, where the first-order system of $5N_g$ equations is reduced to a second-order system of N_g equations written in terms of the linearized discrete pressure P_{tt}. Symbolically, we solve

$$\mathcal{H}P_{tt} = R_{tt} \tag{18}$$

where $\mathcal{H} = \mathcal{H}\left(\mathbf{D}, \mathbf{D}^{\mathcal{T}}\right)$ is a linear, pseudo-Helmholtz operator consisting of gradient $\mathbf{D}$ and divergence $\mathbf{D}^{\mathcal{T}}$ operators operating on the discretized pressure P_{tt} and R_{tt} is an effective source term. Explicit expressions for $\mathcal{H}$ and R_{tt} are given by Eq. (A.10) in [14]. We note that the gradient operator may be written as a DSS of local gradient operators $\mathbf{D}^{(e)}$ using the notation

$$\mathbf{D} = \bigwedge_{e=1}^{N_e} \mathbf{D}^{(e)} \tag{19}$$

which is crucial for the construction of a parallel, semi-implicit algorithm (see Section 4).

3.3. Boundary Conditions

Limited-area atmospheric models, such as mesoscale codes, typically require two types of boundary conditions: 1) no-flux boundary conditions

(NFBCs), that mimic impenetrable objects (e.g., the ground) and 2) non-reflecting boundary conditions (NRBCs), that mimic an infinite domain (e.g. the top of the atmosphere) by allowing waves to propagate out of the computational domain without generating spurious reflections. In this section, we outline how NFBCs are imposed via projection matrices and how NRBCs are imposed via sponges. For additional details, see [12].

3.3.1. No-Flux Boundary Conditions

All of our test cases utilize NFBCs on the bottom boundary, while some test cases (such as the rising thermal bubble) utilize NFBCs on other boundaries as well. For NFBCs, we enforce

$$\hat{\mathbf{n}} \cdot \mathbf{u} = 0 \tag{20}$$

for all points on the boundary Γ, where $\hat{\mathbf{n}}$ is the outward pointing unit normal on Γ. In order to apply the NFBC to the prognostic vector $\mathbf{q}$, we augment $\hat{\mathbf{n}}$ to R^5 via $\hat{\mathbf{n}} = (0, \hat{\mathbf{n}}^{\mathcal{T}}, 0)^{\mathcal{T}}$, yielding $\hat{\mathbf{n}} \cdot \mathbf{q} = 0$. To apply theses boundary conditions in the *strong* sense, we construct a 3 by 3 projection matrix P via

$$P = \begin{pmatrix} 1 - n_x^2 & -n_x n_y & -n_x n_z \\ -n_y n_x & 1 - n_y^2 & -n_y n_z \\ -n_z n_x & -n_z n_y & 1 - n_z^2 \end{pmatrix}. \tag{21}$$

This matrix is constructed during the initialization phase and applied to the RHS of Eq. (16) after each time-step.

3.3.2. Non-Reflecting Boundary Conditions

In an operational NWP model, the four lateral and the top boundary of any mesoscale model should mimic an open domain. That is, waves should smoothly exit the domain without reflection; in addition, information from outside the domain should be allowed to enter the domain of interest. Mathematically modeling this behavior is non-trivial and has attracted the attention of researchers in many domains [3, 17, 27]. In our model, we utilize a simple, albeit, effective absorbing sponge layer method. The computational domain is surrounded by a layer with Newtonian relaxation coefficients $\alpha(x, y, z)$ and $\beta(x, y, z)$ such that $\alpha = 1$ and $\beta = 0$ in the domain of interest, while $\alpha \to 0$ and $\beta \to 1$ at the boundary. Specifically, for the top boundary, we choose

$$\beta = \left(\frac{z - z_s}{z_t - z_s} \right)^4 \tag{22}$$

where z_t is the vertical height of the domain and z_s is the bottom of the sponge layer. Similar functions are used for the lateral boundaries of the domain. Once the sponge layer is constructed, the numerical solution $\tilde{\mathbf{q}}$ given by the RHS of Eq. (16) is relaxed to some known solution at the boundary $\mathbf{q}_b$ via

$$\mathbf{q} = \alpha(x, y, z)\tilde{\mathbf{q}} + \beta(x, y, z)\mathbf{q}_b. \tag{23}$$

For problems under consideration in this paper, $\mathbf{q}_b$ is a far-field condition (e.g. known wind velocity).

4. Parallel Implementation

NUMA is an MPI-based code targeted toward distributed memory architectures (e.g. clusters). In this section, we discuss the parallel implementation of NUMA, including a description of the domain-decomposition, necessary data structures (e.g. local to global mappings), and communication algorithms.

4.1. Domain Decomposition

We decompose Ω into N_p processor elements (PE) Ω_p that consist of local elements $\Omega_{e'}^{(p)}$. Mathematically, we rewrite Eq. (7) as

$$\Omega = \bigcup_{p=1}^{N_p} \bigcup_{e'=1}^{N_e^{(p)}} \Omega_{e'}^{(p)} \tag{24}$$

where $N_e^{(p)}$ is the number of local elements residing on PE p. Since the DSS operator acts on global elements e, we must also construct local to global mappings $e = LG^{(p)}(e')$ that map local elements e' on processor p to global elements e residing on the global domain Ω.

A guiding principle in the construction of NUMA is to maintain independence between grid generation and the spectral element solver; hence, domain decomposition should be as general as possible and not constrained by the underlying grid connectivity. Therefore, we have implemented a domain decomposition strategy based on the widely used METIS graph partitioning library [22]. METIS requires an adjacency graph where the N_e vertices are elements Ω_e and the "edges" denote the connectivity between elements. Since the DSS operator requires information from elements that share nodes, the

"edges" include all forms of geometric connectivity (faces, edges, and vertices). For maximum flexibility, we constructed a weighted adjacency graph $G' = (V, E)$ with adjacency matrix A' of size N_e by N_e defined via

$$a'_{ij} = \begin{cases} 1 & \text{if i and j are vertex neighbors} \\ 2 & \text{if i and j are edge neighbors} \\ 4 & \text{if i and j are faces neighbors} \end{cases} \tag{25}$$

The weights in Eq. (25) represent the number of common points between neighbors assuming linear ($N = 1$) elements; these weights are arbitrary and may be altered to construct a machine-optimal weighted adjacency matrix. Once A' is constructed, the adjacency matrix A for the graph $G = (V, F)$, where F are geometrical faces is simply $a_{ij} = 1$ if $a'_{ij} = 4$ and $a_{ij} = 0$ otherwise. An example connectivity graph for a 2D grid is shown in Figure 1, showing both edge and vertex connectivity. The associated 9 by 9 adjacency matrix has nodes with degree ranging from 3 (for the corner nodes) to 8 (for the central node). This standard adjacency matrix A may be utilized within a discontinuous Galerkin (DG) framework [11], where the only inter-element communication is between adjacent faces via flux operators.

Since we are considering a local method, both A and A' are sparse. Therefore, these matrices are stored in compressed storage (CSR) format using an adjacency list `adjncy` of length $2|E|$ and array `xadj` of length $|V| + 1$. These arrays, along with the number of processor N_p are then passed to METIS, which returns a partition $\mathcal{P} : V \to \{1, 2, ..., N_p\}$, that maps global elements to processors. This mapping is then used to construct the local to global mappings LG necessary for global assembly in Eq. (14).

In addition to the element adjacency graph G, a processor-element adjacency graph $G_P = (V_P, E_P)$ is constructed, where the vertices V_P are processor elements and the edges E_P are the connectivity between processor elements. Again, since we are considering a CG method, E_P includes vertex, edge, and face connections between processor elements. The adjacency matrix N_P by N_P adjacency matrix $A^{(P)}$ associated with G_P is derived from A' as follows: the element $a^{(P)}_{ij} = 1$ if the intersection of all rows i' and columns j' of A' such that $i = \mathcal{P}(i')$ and $j = \mathcal{P}(j')$ has at least one nonzero element; otherwise, $a^{(P)}_{ij} = 0$. From the inter-processor adjacency matrix $A^{(P)}$, the necessary communication data structures are constructed with ease. Specifically, the neighbors of processor element i are simply the non-zero columns of row i, while the number of neighbors for element i is given by $\sum_{j=1}^{N_P} a^{(P)}_{ij}$.

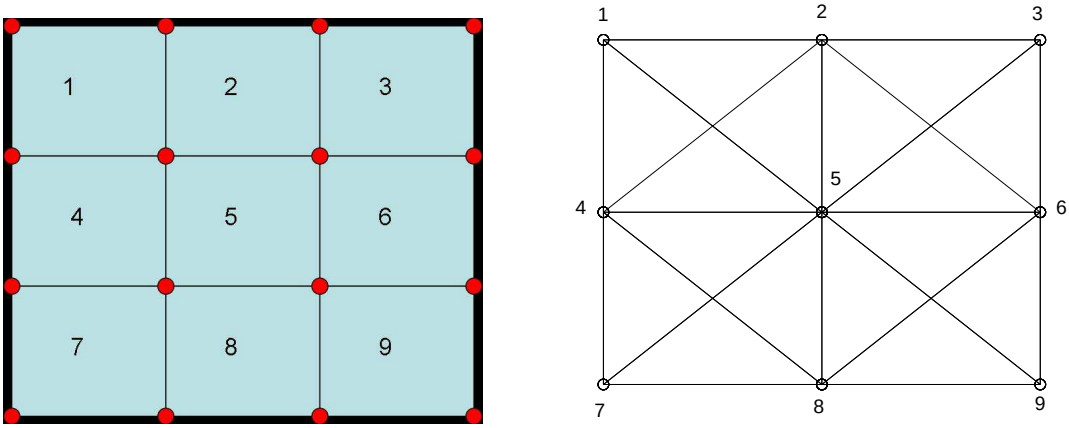


Figure 1: Example 2D grid (left) and the associated adjacency graph. Since spectral element methods utilize nodal communication, the adjacency graph includes both edge and vertex connectivity, with a maximum degree of eight. For 3D, structured, Cartesian grids, the maximum degree is 26.

Thus, a low-communication partition will have an adjacency matrix $A^{(P)}$ that is as sparse as possible.

4.2. Parallelization: Explicit TI

We first consider the parallelization of the explicit time-integrator, due to its simplicity. The global DSS operator in Eq. (14) requires inter-processor communication due to the overlap of elements at processor element boundaries. A global DSS is required in two parts of the code: 1) construction of the mass matrix and 2) construction of the right-hand side (RHS). To perform this communication, we first construct the boundary nodes of each processor (excluding the physical boundary where BCs are applied). Denote the boundary of processor element i by $\partial\Omega_i$. In order to communicate between processor elements, we construct two data structures *send* and *recv*. Consider a particular processor i and neighboring processor j. *send* contains the local nodes on processor i that are sent to each neighboring processor j, while *recv* contains the local nodes on processor j that must be sent to i. In order to construct *send*, we utilize the method shown in Algorithm 1. The corresponding *recv* structure contains the same grid points as *send*, although they may be ordered differently.

Once these data structures have been constructed, the global mass matrix

Algorithm 1 Construction of MPI send/receive communication data structures.

for all NBHs j of i **do**
  MPISEND $\partial\Omega_i^G \leftarrow LG^{(i)}(\partial\Omega_i)$ to proc j
  MPIRECV $\partial\Omega_j^G \leftarrow LG^{(j)}(\partial\Omega_j)$ to proc i
  $B \leftarrow \partial\Omega_i^G \cap \partial\Omega_j^G$
  $send(j) \leftarrow \left[LG^{(i)}\right]^{-1}(B)$
end for

and RHS operators may be constructed via a two step process. The global mass matrix M_{IJ} and RHS operator in Eq. (14) is decomposed as

$$M_{IJ} = \bigwedge_{e=1}^{N_e} M_{ij}^{(e)} = \bigwedge_{n_p=1}^{N_p} \bigwedge_{e'=1}^{N_e^{(p)}} M_{ij}^{(e')} \text{ and} \tag{26a}$$

$$R_I = \bigwedge_{e=1}^{N_e} R_i^{(e)} = \bigwedge_{n_p=1}^{N_p} \bigwedge_{e'=1}^{N_e^{(p)}} R_i^{(e')}. \tag{26b}$$

Hence, the global DSS is decomposed into a local, on-processor DSS and a global DSS, that requires inter-processor communication. In this way, global continuity between elements is preserved during the construction of each RHS. Note that the communication stencil is simply the boundary of each processor element $\partial\Omega^{(i)}$ and is independent of the polynomial order N; that is, unlike higher-order finite difference or finite volume methods, the spectral-element method is *halo-free*. In summary, the global DSS procedure may be summarized as follows:

1. Perform a local DSS on processor i.
2. Exchange boundary points between processors i and all processor neighbors j using $send$ and $recv$.
3. Perform a global DSS using the boundary data received from neighbors j.

The global DSS operation is represented schematically in Figure 2 in a 2D setting. To simplify the discussion, each processor is assumed to own one element. In order to construct the RHS operator on the boundary (red dots), the element E (green) requires nodal information for its 8 nodal neighbors.

These neighbors include both edge neighbors (2, 4, 6, and 8) and vertex neighbors (1, 3, 5, and 7). In a 3D setting, an element may have (a maximum) of 6 face neighbors, 12 edge neighbors, and 8 vertex neighbors, for a total of 26 nodal neighbors. To reduce this communication cost, METIS is utilized to reduce the total number of vertex neighbors; in a typical 3D partition, a processor element lying away from a physical boundary has 16 or less nodal neighbors; hence, the METIS-based decomposition further reduces communication cost relative to a naive geometric domain decomposition.

Spectral elements, and in fact all Galerkin based methods, possess purely local communication stencils. Referring to Fig. 2, the interior nodes (yellow dots) do not need to be communicated to adjacent processors, resulting in a *halo-free* algorithm. Thus unlike high-order finite difference and finite volume schemes, spectral elements may achieve spectral accuracy without sacrificing their local character. We note in passing that discontinuous Galerkin (DG) methods further reduce communication costs, since elements need only communicate with face neighbors; we will report the results of our DG implementation in a future paper.

4.3. Parallelization: Semi-Implicit TI

Unlike the explicit RK-35 TI, the semi-implicit TI requires the solution of the linear system given by Eq. (18) at each time-step. First, the effective source term R_{tt} must be constructed, which requires applying the divergence operator $\mathbf{D}^{\mathcal{T}}$. To solve the linear system, the Krylov-space solver GMRES is utilized, which requires constructing a set of orthonormal basis vectors $\mathbf{v}_i$ that spans the solution space by 1) constructing the matrix-vector product $\tilde{\mathbf{v}} = \mathcal{H}\mathbf{v}_i$ at each iteration, and 2) constructing the orthonormal vector $\mathbf{v}_{i+1}$ from $\tilde{\mathbf{v}}$ via orthogonalization of Krylov vectors. After the discrete pressure P_{tt} is constructed, additional differential operators $\mathbf{D}$ must be applied to construct the solution $\mathbf{q}^{n+1}$. Each of these operations require MPI communication, which is outlined in this section.

In constructing both the LHS and RHS operators in Eq. (18), the differential operator given by Eq. (19) must be applied to the Krylov vector $\mathbf{v}_i$. Analogous to the construction of the RHS operator in Eq. (26b), the differential operator is decomposed into a local, or on-processor DSS, and a global DSS that requires inter-processor communication. The communication stencil for the global DSS is the same as the explicit RHS; that is, the computation of derivatives is halo-free within the continuous Galerkin paradigm.

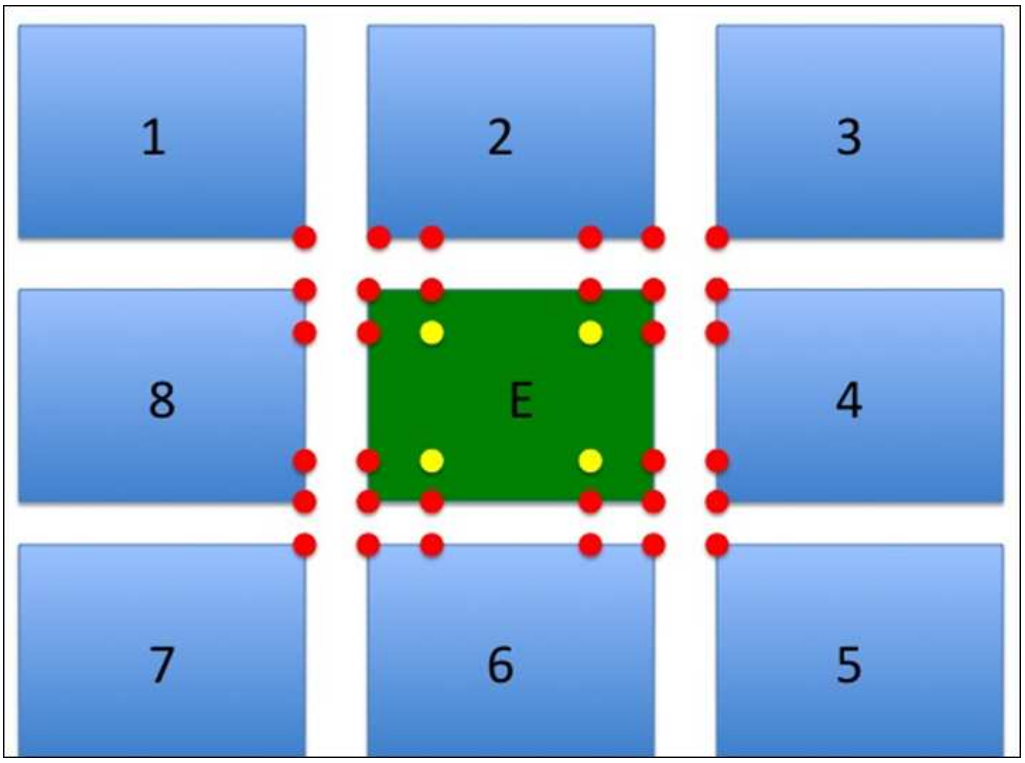


Figure 2: Global DSS operator in a 2D setting, where each processor owns one element. The element E (green) requires nodal information for its 8 nodal neighbors in order to construct the RHS operator on the boundary (red dots). However, the interior nodes (yellow dots) do not need to be communicated to adjacent processors, resulting in a *halo-free* algorithm.

In addition to differentiation operators, the orthogonalization procedure requires communication. In the modified Gram-Schmidt procedure, k global dot products $\alpha_i = \tilde{\mathbf{v}}^{\mathcal{T}}\mathbf{v}_i$ must be performed at the k-th Krylov iteration, where $1 \leq i \leq k$. These dot products 1) depend on the global vector, and 2) are required by each processor element; hence, the computation of dot products is an all-to-all communication that becomes prohibitive for large processor counts. To mitigate this communication overhead, a communication-avoiding GMRES solver [4] will be incorporated in our next-generation semi-implicit solver.

5. Results

5.1. Test Cases

Although a standard set of 2D mesoscale test cases has been proposed [31] and later utilized within an element-based Galerkin framework [12], an analogous suite of 3D test cases has not yet been developed. Fortunately, a 3D dynamical core can be run in 2D mode by imposing symmetry in the y-dimension and periodic boundary conditions in the y-lateral boundaries. For initial verification of NUMA, we utilized the test cases proposed in [31]; later, we ran full 3D test cases with no y-symmetry. In the following analysis, we consider four test cases: 1) 2D inertia-gravity waves, 2) a 3D linear hydrostatic mountain, 3) a 3D nonlinear mountain and 4) a 3D rising thermal bubble.

5.1.1. 2D Inertia-Gravity Waves

The 2D nonhydrostatic inertia-gravity wave simulates the evolution of a potential temperature perturbation in a channel with periodic boundary conditions; this test case was originally proposed in [32]. To adapt this problem to 3D, symmetry in the y-dimension was enforced along with periodic boundary conditions in the y-dimension. The domain is defined as $(x, y, z) \in [0, 300000] \times [0, 300000] \times [0, 10000]$ m. No-flux boundary conditions are applied at the top and bottom of the domain, while periodic boundary conditions are applied on the lateral boundaries.

5.1.2. 3D Linear Hydrostatic Mountain

To test the 3D capabilities of NUMA and the NRBCs, we consider stratified flow past an isolated mountain as outlined in [33]. An initial horizontal

flow $U = 20$ m/s blows past a mountain with orography given by

$$h(x,y) = \frac{h_0}{(x^2/a^2 + y^2/a^2 + 1)^{3/2}} \tag{27}$$

with mountain half-width $a = 10$ km and height $h_0 = 1$ m. A profile of the linear hydrostatic mountain is shown in Figure 3, where the z-axis has been amplified for illustrative purposes. The hydrostatic background is specified by a constant Brunt-Väisälä frequency $N_{bv} = g/\sqrt{cpT_0}$ with ground temperature $T_0 = 250$ K. In other words, we consider an isothermal atmosphere. No flux boundary conditions are imposed on the bottom of the domain, while NRBCs are imposed on the four lateral boundaries and the top boundary. In order to verify our numerical results, several analytical results from [33] and [34] are utilized. First, a contour integral solution for the density perturbations ρ' valid under a linear Boussinesq approximation is utilized. Also, the velocity perturbations parallel and perpendicular to the flow (u' and v') are known for observation points near the ground ($z = 0$) (see Eqs. (39) and (41) in [33]):

$$u'(x,y,0) = hN_{bv}\frac{x/a}{1 + x^2/a^2 + y^2/a^2} \tag{28a}$$

$$v'(x,y,0) = hN_{bv}\frac{y/a}{1 + x^2/a^2 + y^2/a^2} \tag{28b}$$

under the same approximation.

5.1.3. 3D Nonlinear Mountain

Additional tests were performed by increasing the height of the mountain in Eq. (27) from $h_0 = 1$ m to $h_0 = 1000$ m, causing nonlinear effects such as flow splitting. All other parameters were held constant.

5.1.4. 3D Rising Thermal Bubble

We consider a 3D buoyant thermal bubble rising in a neutrally stratified atmosphere [37], which is the 3D extension of a 2D thermal bubble originally considered in [35]. The hydrostatic potential temperature $\theta_0(z) = 300$ K (neutral atmosphere) is perturbed with a sphere of radius $r_c = 250$ m centered at $(x_c, y_c, z_c) = (500, 500, 260)$ m by a cosine taper given by

$$\theta' = A\left[1 + \cos\left(\frac{\pi r}{r_c}\right)\right] \tag{29}$$

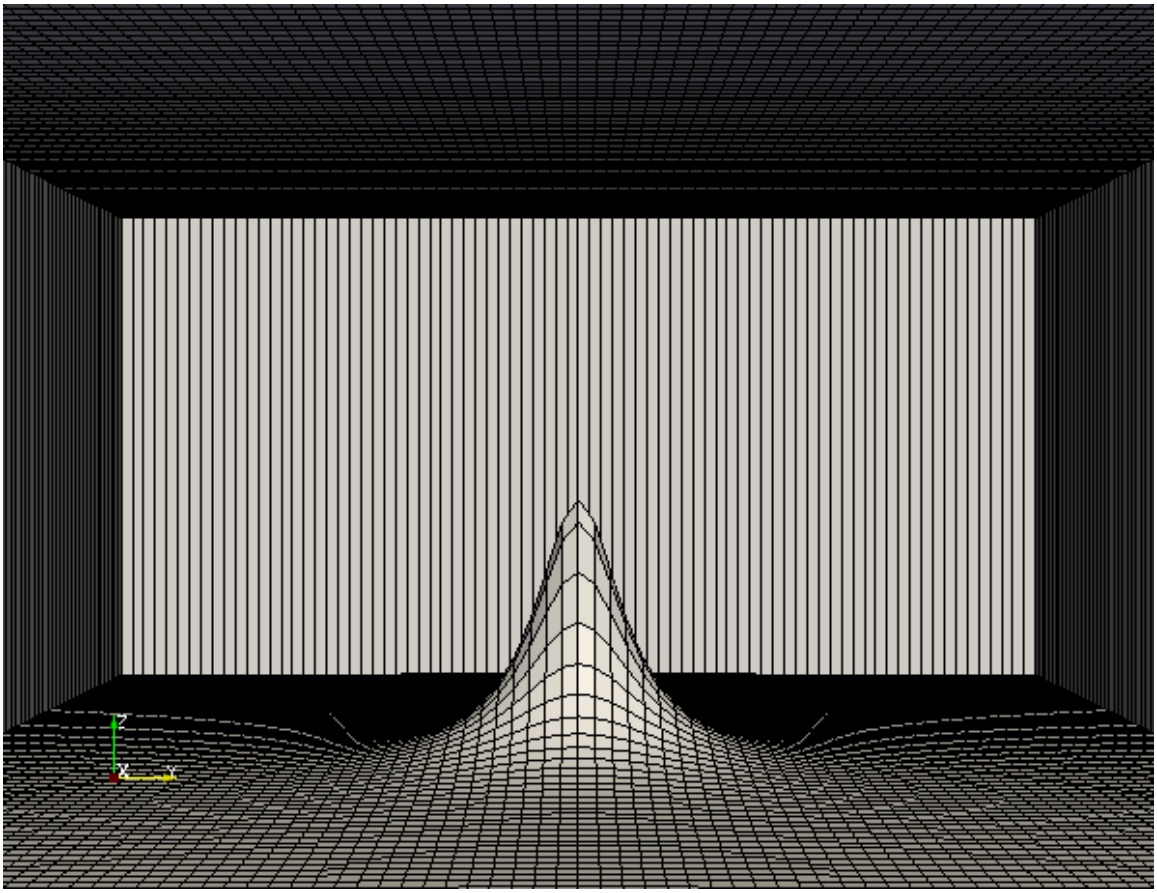


Figure 3: Profile of the isolated, 3D linear hydrostatic mountain defined by Eq. (27) originally considered in [33]

where $r = \sqrt{(x-x_c)^2 + (y-y_c)^2 + (z-z_c)^2}$ and A is a constant. Unlike the test case utilized in [37], our problem has a C^1 initial condition, thus mitigating unphysical oscillations associated with spectral element methods. The domain is defined as $(x, y, z) \in [0, 1000] \times [0, 1000] \times [0, 1500]$ m. No-flux boundary conditions are applied on all six boundaries.

5.2. Numerical Verification

The numerical verification proceeds in three steps. In phase one, we ran the code in pseudo-2D mode using 1 element and periodic boundary conditions in the y-direction. The numerical results for the standard mesoscale suite [31] are directly compared to the results of our existing 2D model. In phase two, we considered the linear isolated mountain problem, which possess an approximate analytical solution in the form of a contour integral. In phase three, we consider flows over nonlinear mountains with height of 1 km and three-dimensional buoyant convection. Although these problems do not have analytical solutions, their qualitative behavior is understood.

Figure 4 displays results from NUMA for the 2D Inertia-Gravity Waves test case. Fig. 4(a) shows the potential temperature perturbation θ' after 2500 s for 250 m resolution (120 by 1 by 4 elements) and 10-th order polynomials for a slice in the center of the domain. Fig. 4(b) shows profiles along $z = 5000$ m; note the symmetry about $x = 150000$ m. These 3D results agree to eight decimal places of precision with earlier results generated by a 2D spectral element model. From this initial test, we conclude that NUMA is capable of 1) capturing the propagation of gravity waves in a stratified medium and 2) mimicking the results of a purely 2D model.

In order to test the 3D operators, orography, and non-reflecting boundary conditions, flow over a 3D isolated linear hydrostatic mountain was considered. This problem may be solved under the linear Bosusinesq approximation via a contour integral technique as described in [34]. In addition, closed form expressions for both the down-stream and cross-stream velocity components may be compared to the numerical solution. We used 20 spectral elements in the x and y dimensions and 10 in the z direction with eighth-order polynomials yielding effective resolutions of $\bar{\Delta x} = \bar{\Delta y} = 1.5$ km and $\bar{\Delta z} = 300$ m. The semi-implicit TI was run with a time-step of $\Delta t = 1$ s on an Apple XServe cluster using 10 processors.

Figure 6 compare the density perturbations ρ' using both of these approaches. In panel a), contours for ρ' are shown after 6 hours of simulation time and compared to the contour integral solution. Agreement is very good

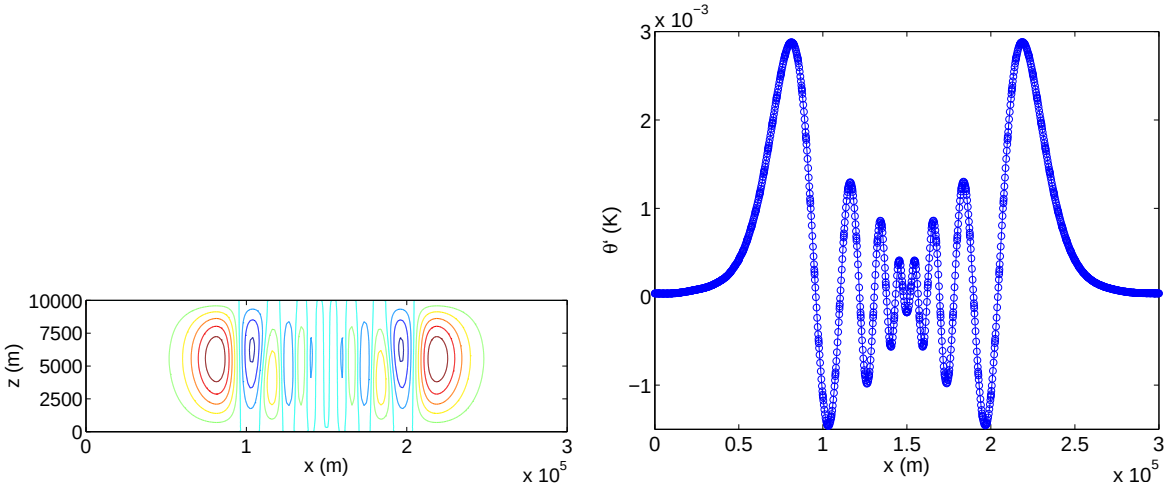


Figure 4: Case 1: 2D Inertia-gravity waves. Potential temperature perturbation after 2500 s for 250 m resolution (120 by 1 by 4 elements) and 10-th order polynomials. a) shows a slice in the center of the domain and b) shows profiles along $z = 5000$ m.

near the mountain, while the two solutions begin to deviate as z increases due to the influence of the sponge. Agreement between NUMA and Smith's results may be brought into closer agreement by either 1) increasing the resolution of the model or 2) running the model longer. Additional verification was performed by comparing the velocity on the surface of the mountain. The down-stream velocity perturbation u' and cross-stream velocity perturbation v' at the ground are compared with the analytical formulas in Eq. (28). Figure 5 displays the results of this comparison after $t = 4$ hours of integration. Agreement between the numerical spectral element model and the analytical formulas given by Eq. (28) is satisfactory, especially for the cross-stream velocity perturbation.

After verification in an isothermal atmosphere, the hydrostatic isolated mountain problem was also tested in a neutral atmosphere with $\theta_0(z) = 250$ K. To our knowledge, there is no analytical solution for this problem, but the results are qualitatively similar to the wave patterns generated for a 2D linear hydrostatic ridge. The three velocity components are shown in Figure 7 after 20 and 60 minutes. The initially horizontal flow creates a discontinuity and triggers vertically propagating acoustic waves. Note that the cross-stream velocity is symmetric with respect to the y-axis, that is expected since the initial cross-stream velocity is zero. As time evolves, a

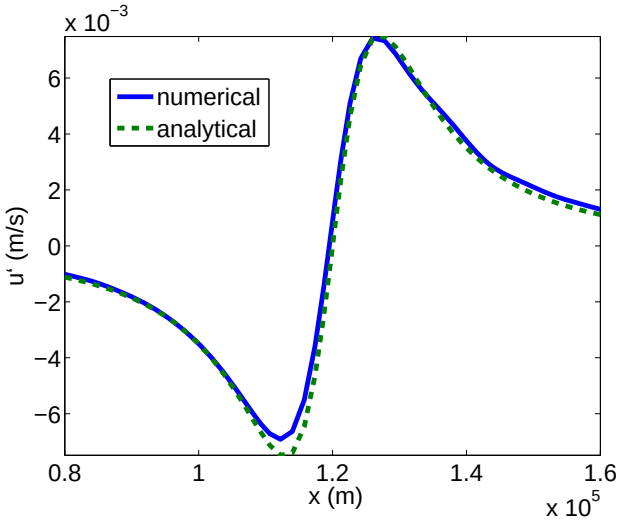
steady state mountain wave pattern develops, which begins to converge after 3600 seconds of simulation time. These tests show the development of eddies, especially on the lee side of the mountain. Unlike flows in atmospheres with constant N_{bv}, the velocity components do not attenuate as z increases. Since the mountain has a small height (1 m), there is no splitting of the flow around the base of the mountain. Qualitatively, these 3D mountain results are very similar to the standard 2D mountain (or ridge).

Additional tests on a nonhydrostatic mountain $h_0 = 1000$ m were also performed using the same initial horizontal velocity. By increasing the height of the mountain from $h_0 = 1$ m to $h_0 = 1000$ m, nonlinear phenomena, such as flow splitting emerge. Figure 8 shows the flow (i.e. velocity vectors) produced by the nonlinear mountain. Unlike the linear hydrostatic mountain analyzed above, this problem cannot be solved by analytical means. In this test case, the velocity perturbations are much larger than the 1 m mountain and significant eddy mixing develops on the lee side of the mountain. However, since the maximum velocity is an order of magnitude smaller than the speed of sound, a relatively large time-step may be utilized within a semi-implicit solver, demonstrating the efficiency of the Schur SI-integrator. This test confirms that NUMA is capable of handling topography; in future tests, we will consider flow over realistic orography. To compare the velocity fields of the nonlinear mountain with the linear mountain, the horizontal (u) and vertical (w) velocity perturbations in the plane $y = 0$ are shown in Figure 9. Note that the magnitude of the vertical velocities between the LHM and the nonlinear mountain differ by over three orders of magnitude.

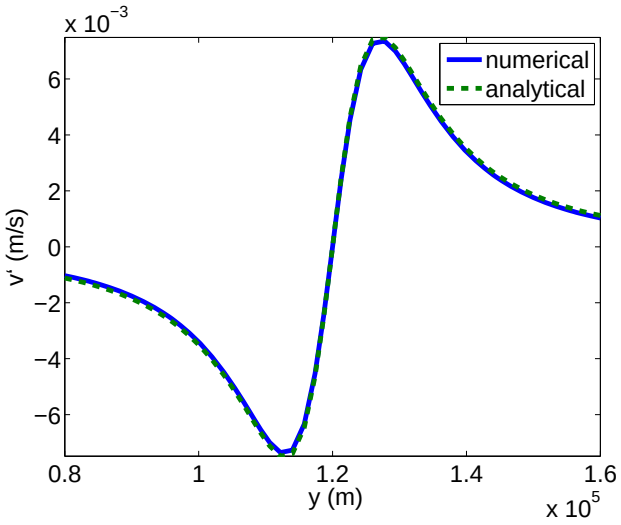
Finally, the results of the buoyant convection (i.e., rising 3D thermal bubble) experiment are shown in Figure 10. A smooth potential temperature perturbation in an initially neutral atmosphere generates vertical updrafts and shear velocities, which cause the bubble to rise, deform, and transition to turbulence. The rising thermal bubble problem tests the model's ability to capture 1) the effects of turbulence and 2) turbulent convection. Although no analytical solution exists for this problem, the numerical results are physically plausible and resemble previous 3D bubble experiments in [37] and [1]. Similar results are seen for both the explicit and semi-implicit codes.

5.3. Comparison of Time-Integrators

An extensive comparison of high-order RK35 time-integrators and the semi-implicit BDF2 integrator was performed in [14] for 2D nonhydrostatic problems. In this section, we briefly extend this analysis to 3D, where we

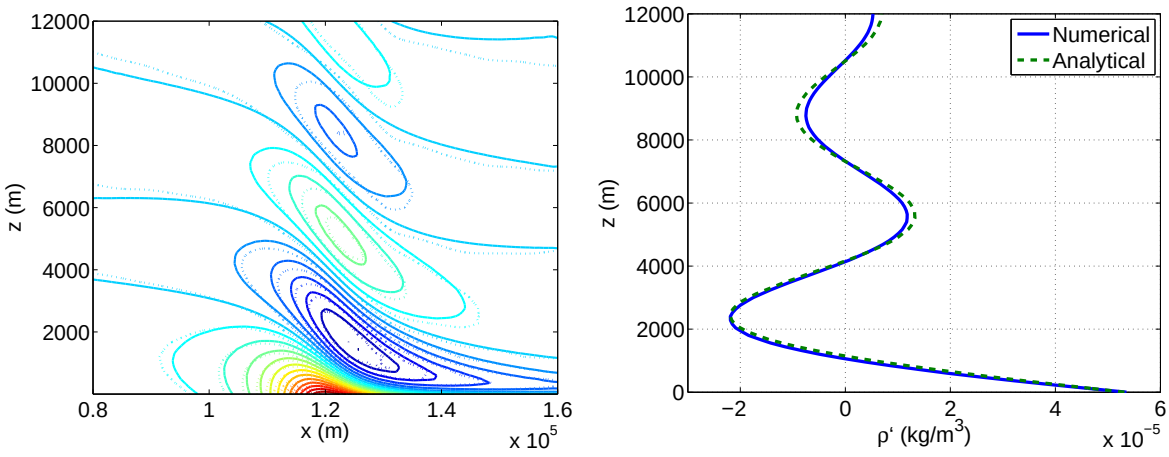


(a) down-stream velocity



(b) cross-stream velocity

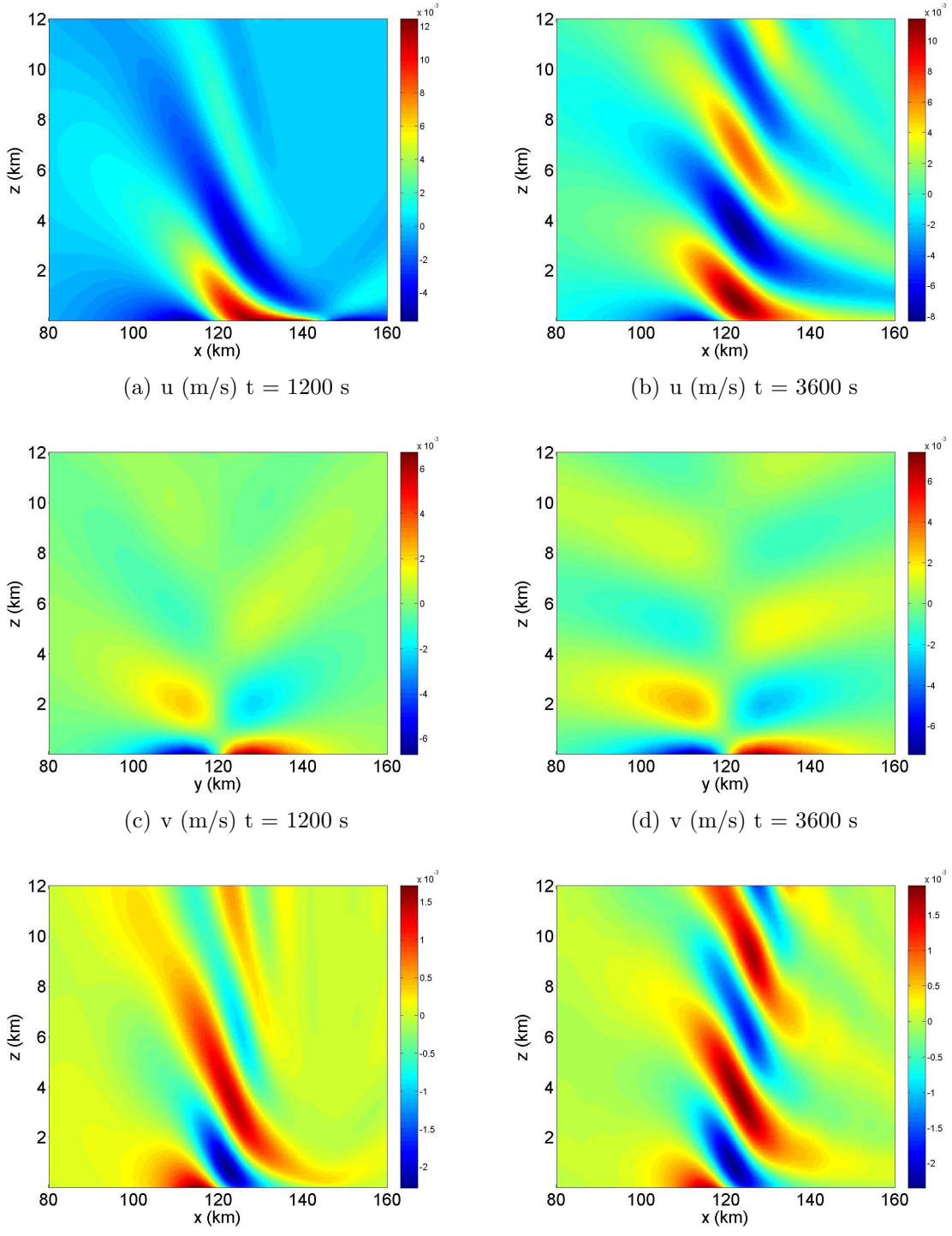
Figure 5: Comparison of the a) down-stream velocity perturbation u' and b) cross-stream velocity perturbation v' for the isolated mountain at ground level. Agreement between the numerical spectral element model and the analytical formulas given by Eq. (28) is satisfactory, especially for the cross-stream velocity perturbation.



(a) (b)

Figure 6: Comparison of the density perturbations ρ' for the isolated linear hydrostatic mountain using 1) NUMA and 2) a contour integral solution [34]. In panel a), the ρ' contours in the plane $y = 0$ are shown. In panel b), ρ' is shown as a function of z using $x = y = 0$.

show that the semi-implicit Schur integrator is more efficient than the RK35 integrator in serial mode for *all* applicable Courant numbers. Figure 11 displays CPU times versus Courant numbers for both the a) linear hydrostatic mountain and b) the 3D rising thermal bubble using the explicit RK35 time-integrator and the semi-implicit BDF2 integrator. Examining panel a), we see the BDF2 integrator continues to gain efficiency until we reach the maximum Courant number of 4.88; for Courant numbers larger than 4.88, the semi-implicit integrator becomes unstable due to the explicit discretization of the non-linear advective terms. Comparing the CPU times of the RK35 and BDF2 integrator at their maximum Courant number, we see that the semi-implicit integrator is faster by a factor of 2.6. A similar trend holds for the 3D rising thermal bubble problem; however, since the velocities are much smaller for the bubble problem, the semi-implicit integrator may be run with a much larger time-step. Note that the semi-implicit integrator begins to lose efficiency when the Courant number is increased from 4.02 to 8.04; in this case, the number of required GMRES iterations increases from 15 to 30. We attribute this behavior to the GMRES solver, which scales quadratically with respect to the number of Krylov iterations. Hence, for both test problems, the optimal Courant numbers is about 4.



(a) u (m/s) t = 1200 s

(b) u (m/s) t = 3600 s

(c) v (m/s) t = 1200 s

(d) v (m/s) t = 3600 s

(e) w (m/s) t = 1200 s

(f) w (m/s) t = 3600 s

Figure 7: Down-stream, cross-stream, and vertical velocities for the 3D linear hydrostatic mountain problem in a neutral atmosphere.

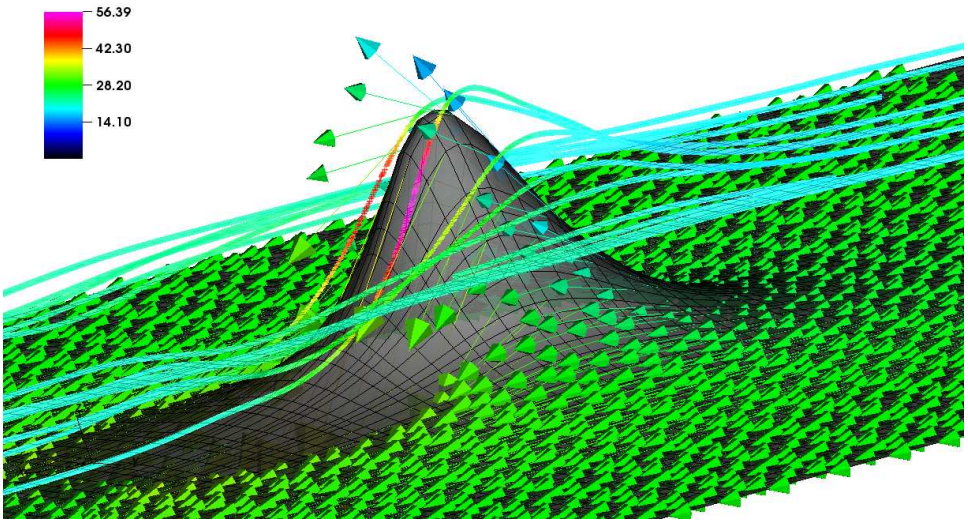
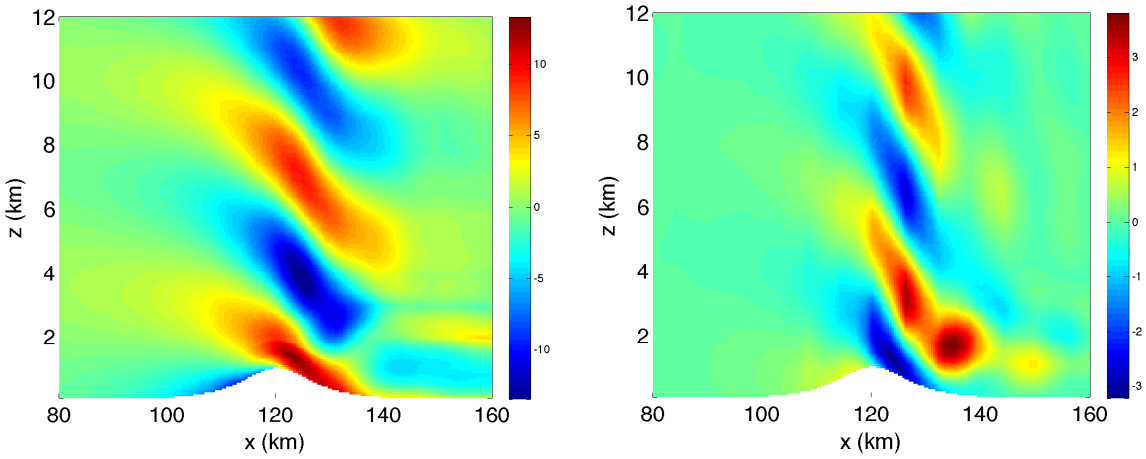


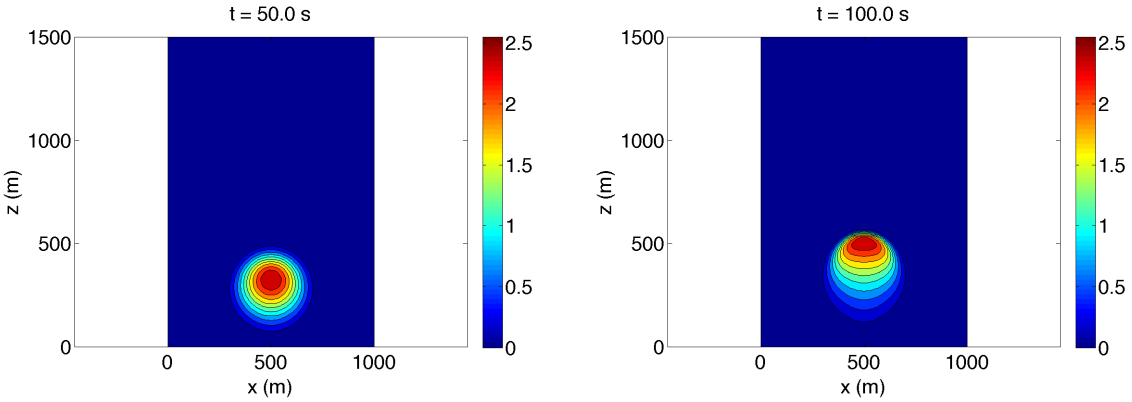
Figure 8: Streamlines and velocity vectors produced by flow about a nonlinear mountain with $h = 1000$ m.



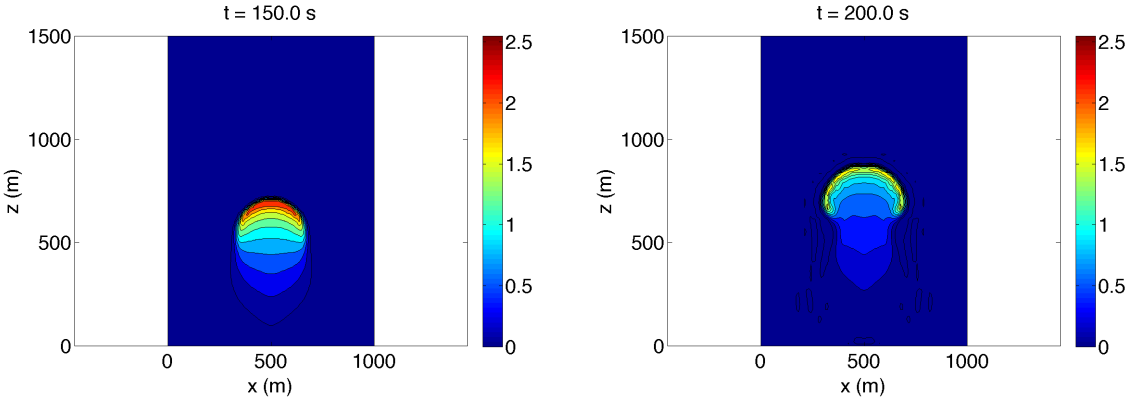
(a) u (m/s) t = 3600 s

(b) w (m/s) t = 3600 s

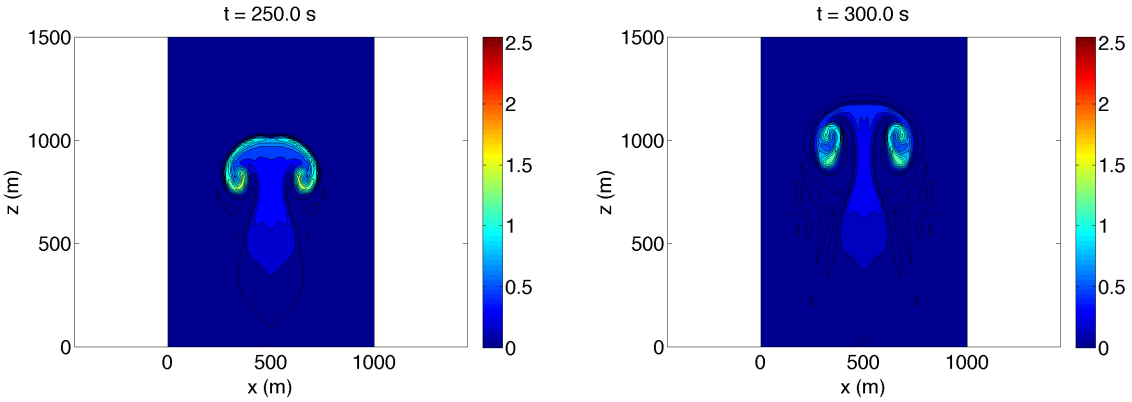
Figure 9: Horizontal and vertical velocities in the plane $y = 0$ for flow over a nonlinear mountain with $h = 1000$ m.



(a) t = 50 s (b) t = 100 s

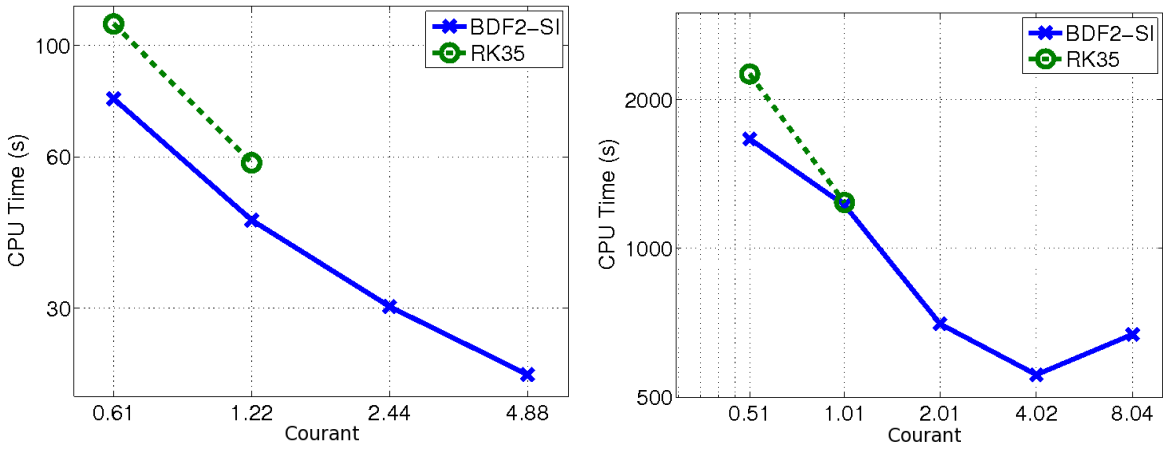


(c) t = 150 s (d) t = 200 s



(e) t = 250 s (f) t = 300 s

Figure 10: Evolution of a 3D rising thermal bubble problem ($x-z$-slices of the potential temperature perturbation θ') in the $y = 500$ m plane for $t = 50$, 100, 150, 200, 250, and 300 seconds.



(a) LHM (b) RTB

Figure 11: Comparison of the explicit RK35 and semi-implicit BDF2 time-integrators in serial mode for a) the linear hydrostatic mountain problem and 2) the 3D rising thermal bubble problem. In both cases and for all applicable Courant numbers, the BDF2-SI integrator is more efficient.

5.4. Parallel Performance

Optimized explicit and semi-implicit versions of NUMA have been deployed on TACC's Ranger Sun Constellation cluster. Ranger consists of 3,936 16-way SMP compute nodes (blades) with 4 AMD quad-core Opteron processors per blade for a total of 62,976 compute cores and a theoretical peak performance of 579 Tera FLOPS. Two test problems utilizing the rising thermal bubble with fourth order polynomials were executed: a $48^3 = 110592$ element problem for processor counts ranging from 16 to 512 and a $64^3 = 262144$ element problem for processor counts ranging from 512 to 12288. The wall clock time associated with time-integration was then recorded for each run; the startup times required for constructing grids and data structures were omitted, as well as the time required for model output. A time-step of 0.001 s was utilized for the explicit code, whereas a time step of 0.01 s was used for the semi-implicit code.

Figure 12 displays CPU times associated with these scaling problems. In panel a), note the nearly linear scaling of NUMA for processor counts ranging from 32 to 512. In panel b), note the near-linear scaling of the MPI code for processor counts ranging from 512 to 4096 (for semi-implicit) and 12288 (for explicit); this linear scaling is attributed to 1) the minimal

communication costs associated with spectral elements, and 2) the near-optimal implementation of the communication algorithm. Secondly, note the super-linear speedup for processor counts ranging from 1024 to 2048. This super-linear speedup is observed once the local problem size fits inside the L2 cache memory.

Comparing the computation times of the explicit and semi-implicit codes in Fig. 12, we see the semi-implicit code is more than an order of magnitude faster than the explicit code for the time-steps chosen. This speed-up is attributed to two factors: 1) the SI time-integrator admits a larger time-step and 2) the BDF2-based discretization required only 2 GMRES iterations per time-step, whereas the RK35 integrator requires constructing 5 RHS's. In addition, each of the iterations associated with the Schur-complement SI has $\mathcal{O}(N_p)$ operations, while the explicit time-integrator requires $\mathcal{O}(5N_p)$ operations to construct a RHS where N_p is the number of global grid points. Hence, we conclude the SI integrator is more efficient, for both single and multiple processors, relative to the explicit time-integrator.

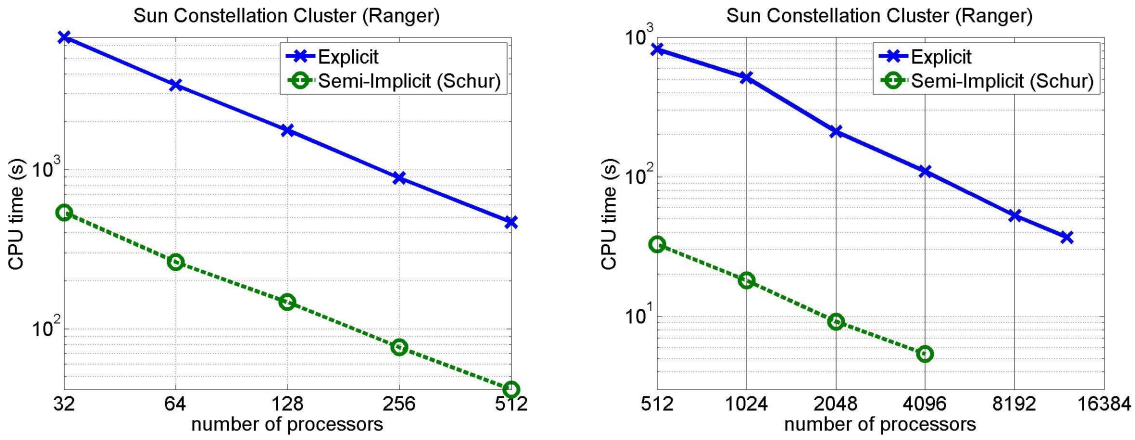
We note that the scaling performance of the semi-implicit code depends on the number of GMRES iterations required for convergence. For a time step of 0.01 s, an average of 2 GMRES iterations per time step are required for convergence. Increasing the time step increases the number of required GMRES iterations, which reduces the scalability of the semi-implicit code to large processor counts. Hence, increasing the time-step may make the semi-implicit algorithm more efficient for small processor counts but less efficient for higher processor counts. The optimal time-step for our semi-implicit code is hence a function of both the test problem at hand (which determines the number of GMRES iterations) and the available architecture (e.g. number of processors). The choice of this optimal time-step will be studied in a future work.

6. Discussion and Conclusion

6.1. Future Work

6.1.1. Microphysical Parameterizations

In conjunction with NRL Monterey, we are incorporating microphysical parameterizations into NUMA. Preliminary experiments using the Kessler scheme [23] have been conducted within a 2D, serial implementation [8]. Since physical parameterizations operate on columns of data independently of adjacent data, the problem is embarrassingly parallel *provided* the domain



(a) 110592 elements (b) 262144 elements

Figure 12: Optimized, explicit RK35 and semi-implicit BDF2 METIS-based spectral element code, displaying CPU times versus processor counts ranging from a) 16 to 512 and b) 512 to 12288. Near linear scaling to 12288 processors on TACCs Ranger Sun Constellation cluster with super-linear speedup observed once the local problem size fits inside the L2 cache memory.

is decomposed in the horizontal only such that all z values reside on processor. To facilitate scaling on hybrid shared-distributed memory architectures (such as TACC's Ranger), hierarchical domain decomposition is desirable, whereby MPI communication based on the algorithm developed in the present paper is utilized in the horizontal and either OpenMP parallelization, appropriate for shared memory, or graphical processor units (GPUs) are employed for fine-grained parallelism in the vertical.

6.1.2. Global Model

As stated in the Introduction, NUMA is a unified nonhydrostatic dynamical core appropriate for both limited-area and global atmospheric simulations. In this respect, NUMA may be viewed as the nonhydrostatic successor to the Naval Spectral Element Atmospheric Model (NSEAM) [13, 10]. At present, we have developed 3D spherical grids based on the cubed sphere and icosahedral geometries and have successfully performed rising thermal bubble, acoustic wave propagation and inertia-gravity wave propagation experiments on these grids. In a companion paper, we will present numerical results for the standard dry dynamical core tests such as the Jablonowski-Williamson baroclinic instability [21], linear and non-linear mountains [39] ,

and Held-Suarez [16] test cases.

6.2. Conclusion

In this paper, we have developed a nonhydrostatic unified model of the atmosphere (NUMA) based on a spectral element (or continuous Galerkin) discretization in space and a suite of explicit and semi-implicit time-integrators. This model is suitable for simulations of both limited-area (mesoscale) and global-area simulations of atmospheric phenomena; this paper has subjected NUMA to a battery of limited-area simulations, including inertia-gravity waves, orographic flow, and buoyant convection problems. The results of these test problems are in agreement with either 1) previous simulations, 2) analytical results, or 3) physical intuition.

NUMA is targeted towards distributed memory architectures, such as Sun constellation clusters, and hence requires implementing both domain decomposition and communication algorithms. These parallelization strategies are based on a general graph-theoretic approach utilizing METIS graph partitioning. Since spectral elements are *halo-free*, only the boundary of each processor element needs to be communicated during the DSS operation. Denoting the number of local elements by N_e, the communication bandwidth is $\mathcal{O}\left(N_e^{2/3}N^2\right)$ while the amount of on-processor work is $\mathcal{O}\left(N_eN^4\right)$, yielding a work-to-communication ratio of $\mathcal{O}\left(N_e^{1/3}N^2\right)$. Hence, the spectral element formulation is optimal for high orders of discretization. NUMA has been deployed on TACC's Ranger Sun Constellation cluster and scalability experiments have been conducted. These experiments reveal near linear scaling to 12288 processors for the explicit code and 4096 processors for the semi-implicit code. We note that the semi-implicit algorithm scalability is adversely affected by increasing the time-step and the associated number of GMRES iterations. We are currently working on scalable preconditioners that are expected to improve scalability on the semi-implicit method [20].

Acknowledgment

The authors acknowledge Gabriele Jost, Texas Advanced Computing Center (TACC), for conducting the semi-implicit scalability study reported in Section 5, Simone Marras, Barcelona Supercomputing Center, and Lester E. Carr, Dept. of Applied Mathematics, Naval Postgraduate School. The authors also acknowledge TeraGrid for providing resources on TACC's Ranger Sun Constellation cluster. This work was funded by ONR Grant PE-0602435N.

References

[1] N. Ahmad, J. Lindeman, A Godunov-type finite volume scheme for meso- and micro-scale flows in three dimensions, Pure and Applied Geophysics 165 (2008) 1929–1939.

[2] T. Davies, M.J.P. Cullen, A.J. Malcolm, M.H. Mawson, A. Staniforth, A.A. White, N. Wood, A new dynamical core for the Met Office's global and regional modelling of the atmosphere, Q. J. R. Meteorol. Soc. 131 (205) 1759–1782.

[3] J.R. Dea, F.X. Giraldo, B. Neta, High-order non-reflecting boundary conditions for the linearized 2-D Euler equations: No mean flow case , Wave Motion 46 (2009) 210–220.

[4] J. Demmel, M. Hoemmen, M. Mohiyuddin, K. Yelick, Avoiding communication in sparse matrix computations, Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on (2008) 1 –12.

[5] J.M. Dennis, R.D. Nair, H.M. Tufo, M. Levy, T. Voran, Development of a scalable global discontinuous Galerkin atmospheric model, Int. J. of Comput. Sci. Eng. In Press (2008).

[6] M.O. Deville, P.F. Fischer, E.H. Mund, High-Order Methods for Incompressible Fluid Flow, Cambridge University Press, 2002.

[7] A. Fournier, M.A. Taylor, J.J. Tribbia, The spectral element atmosphere model (SEAM): High-resolution parallel computation and localized resolution of regional dynamics, Mon. Wea. Rev. 132 (2004) 726–748.

[8] S. Gabersek, F.X. Giraldo, J.D. Doyle, Simple microphysics experiments with a spectral element model, Mon. Wea. Rev. (in preparation) (2010).

[9] F.X. Giraldo, The Lagrange-Galerkin spectral element method on unstructured quadrilateral grids, J. Comp. Phys. 147 (1998) 114–146.

[10] F.X. Giraldo, Semi-implicit time-integrators for a scalable spectral element atmospheric model, Q. J. R. Meteorol. Soc. 131 (2005) 2431–2454.

[11] F.X. Giraldo, J.S. Hesthaven, T. Warburton, Nodal high-order discontinuous Galerkin methods for the spherical shallow water equations, J. Comp. Phys. 181 (2002) 499–525.

[12] F.X. Giraldo, M. Restelli, A study of spectral element and discontinuous Galerkin methods for the Navier–Stokes equations in nonhydrostatic mesoscale atmospheric modeling: Equation sets and test cases, J. Comp. Phys. 227 (2008) 3849–3877.

[13] F.X. Giraldo, T.E. Rosmond, A scalable spectral element Eulerian atmospheric model (SEE-AM) for NWP: Dynamical core tests, Mon. Wea. Rev. 132 (2004) 133–153.

[14] Giraldo, Francis Xavier and Restelli, M. and L äuter, M., Semi-implicit formulations of the Navier-Stokes equations: Applications to non-hydrostatic atmospheric modeling, SIAM J. Sci. Comp. 32 (2010) 3394–3425.

[15] L. Grinberg, G.E. Karniadakis, A new domain decomposition method with overlapping patches for ultrascale simulations: Application to biological flows , J. Comput. Phys. 229 (2010) 5541–5563.

[16] I.M. Held, M.J. Suarez, A proposal for the intercomparison of the dynamical cores of atmospheric general circulation models, Bull. Amer. Meteor. Soc. 75 (1994) 1825–1830.

[17] R.L. Higdon, Radiation boundary condtions for dispersive waves , Siam J. Numer. Anal. 31 (1994) 64–100.

[18] R. Hodur, The Naval Research Laboratory's coupled ocean/atmosphere mesoscale prediction system (COAMPS), Mon. Wea. Rev. 125 (1997) 1414–1430.

[19] G. Houzeaux, M. Vázquez, R. Aubry, J.M. Cela, A massively parallel fractional step solver for incompressible flows, J. Comp. Phys. 228 (2009) 6316–6332.

[20] L.E.C. III, C.F. Borges, F.X. Giraldo, An element-based, spectrally-optimized, approximate inverse preconditioner for the Euler equations, SIAM J. Sci. Comp. Submitted (2011).

[21] C. Jablonowski, D.L. Williamson, A baroclinic instability test case for atmospheric model dynamical cores , Q. J. R. Meteorol. Soc. 132 (2006) 2943–2975.

[22] G. Karypis, V. Kuman, A fast and highly quality multilevel scheme for partitioning irregular graphs, SIAM J. Sci. Comp. 20 (1998) 359–392.

[23] E. Kessler, On the Distribution and Continuity of Water Substance in Atmospheric Circulation, AMS, 1969.

[24] J. Klemp, W. Skamarock, J. Dudhia, Conservative split-explicit time integration methods for the compressible nonhydrostatic equations, Mon. Wea. Rev. 135 (2007) 2897–2913.

[25] D.A. Knoll, D.E. Keyes, Jacobian-free Newton-Krylov methods: a survey of approaches and applications, J. Comp. Phys. 193 (2004) 357 – 397.

[26] D.Y. Le Roux, Dispersion relation analysis of the $p^{NC_1} - p^1$ finite-element pair in shallow-water models, SIAM J. Sci. Comput. 27 (2005) 394–414.

[27] J.M. Lindquist, B. Neta, F.X. Giraldo, A spectral element solution of the Klein-Gordon equation with high-order treatment of time and non-reflecting boundary, Wave Motion 47 (2010) 289 – 298.

[28] M. Restelli, F.X. Giraldo, A conservative discontinuous galerkin semi-implicit formulation for the navier-stokes equations in nonhydrostatic mesoscale modeling, SIAM J. Sci. Comp. 31 (2009) 2231–2257.

[29] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing, Boston, 1996.

[30] U. Schattler, G. Doms, J. Steppeler, Requirements and problems in parallel model development at DWD, Scientific Programming 8 (2000) 13–22.

[31] W.C. Skamarock, J.D. Doyle, P. Clark, N. Wood, A standard test set for nonhydrostatic dynamical cores of NWP models , AMS NWP-WAF Conference Poster (2004) P2.17.

[32] W.C. Skamarock, J.B. Klemp, Efficiency and accuracy of the Klemp-Wilhelmson time-splitting technique, Mon. Wea. Rev. 122 (1994) 2623–2630.

[33] R.B. Smith, Linear theory of stratified hydrostatic flow past an isolated mountain , Tellus 32 (1980) 348–364.

[34] R.B. Smith, Linear theory of stratified flow past an isolated mountain in isoteric coordinates , J. Atmos. Sci. 45 (1988) 3889–3896.

[35] P.K. Smolarkiewicz, J.A. Pudykiewicz, A class of semi-Lagrangian approximations for fluids, J. Atmos. Sci. 49 (1992) 2082–2096.

[36] R.J. Spiteri, S.J. Ruuth, A new class of optimal high-order strong-stability-preserving time discretization methods, SIAM J. Numer. Anal. 40 (2002) 469–491.

[37] S.J. Thomas, J.P. Hacker, P.K. Smolarkiewicz, Spectral preconditioners for nonhydrostatic atmospheric models, Mon. Wea. Rev. 131 (2003) 2464–2478.

[38] J. Thuburn, T.D. Ringler, W.C. Skamarock, J.B. Klemp, Numerical representation of geostrophic modes on arbitrarily structured c-grids, J. Comp. Phys. 228 (2009) 8321–8335.

[39] H. Tomita, M. Satoh, A new dynamical framework of nonhydrostatic global model using the icosahedral grid, Fluid Dynamics Research 34 (2004) 357–400.

[40] P.A. Ullrich, C. Jablonowski, B. van Leer, High-order finite-volume methods for the shallow-water equations on the sphere, J. Comp. Phys. 229 (2010) 6104–6134.

[41] L.C. Wilcox, G. Stadler, C. Burstedde, O. Ghattas, A high-order discontinuous Galerkin method for wave propagation through coupled elastic-acoustic media, J. Comp. Phys. 229 (2010) 9373–9396.