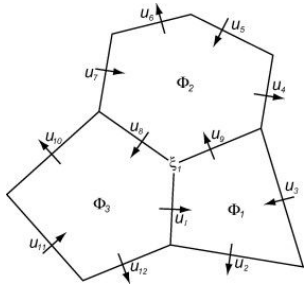
Discretizations of the Shallow Water on the Sphere on Voronoi Grids

Prof. Dr. Pedro da Silva Peixoto

Applied Mathematics
Instituto de Matemática e Estatística
Universidade de São Paulo

Oct 2023

Key references



“TRiSK” Papers:

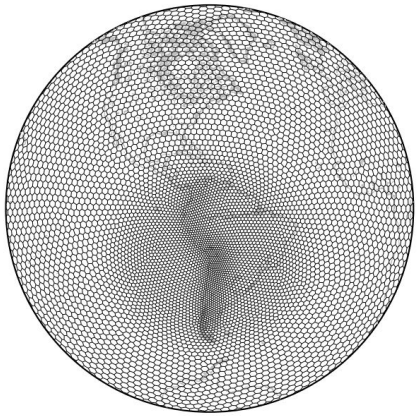
- Thuburn, J., T. Ringler, J. Klemp and W. Skamarock, 2009: Numerical representation of geostrophic modes on arbitrarily structured C-grids, Journal of Computational Physics, 2009
- Ringler, T., J. Thuburn, J. Klemp and W. Skamarock, 2010: A unified approach to energy conservation and potential vorticity dynamics on arbitrarily structured C-grids, Journal of Computational Physics

Analysis:

- Exploring a Multi-Resolution Modeling Approach within the Shallow-Water Equations. Ringler, T., D.W. Jacobsen, M. Gunzburger, L. Ju, M. Duda and W. Skamarock, 2011, Monthly Weather Review
- Peixoto, 2016: Accuracy analysis of mimetic finite volume operators on geodesic grids and a consistent alternative (Journal of Computational Physics) – PDF, DOI
- Yonggang Yu; Ning Wang; Jacques Middlecoff; Pedro Peixoto; Mark Govett, 2020: Comparing Numerical Accuracy of Icosahedral A-grid and C-grid Schemes in Solving the Shallow-Water Model (Monthly Weather Review) – PDF, DOI
- Santos, Peixoto, 2021: Topography based local spherical Voronoi grid refinement on classical and moist shallow-water finite volume models (Geoscientific Model Development). Open Access

Codes

MPAS
Model for Prediction Across Scales



- MPAS-BR:

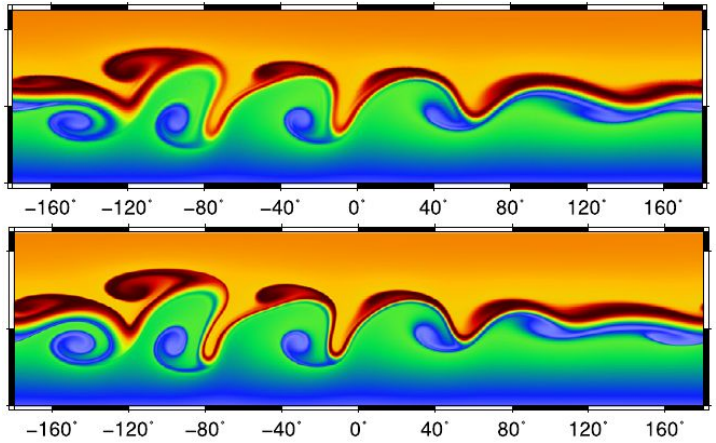
  https://github.com/pedrospeixoto/MPAS-BR

  Research code for MPAS Brazilian MONAN initiative.

- IMODEL:

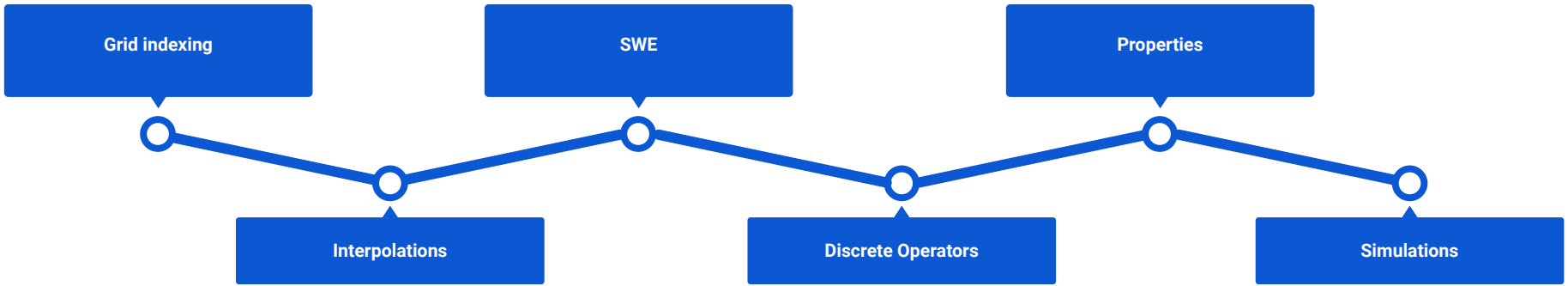
  https://github.com/pedrospeixoto/iModel

  Shallow Water Equation Model
  Icosahedral grid tools



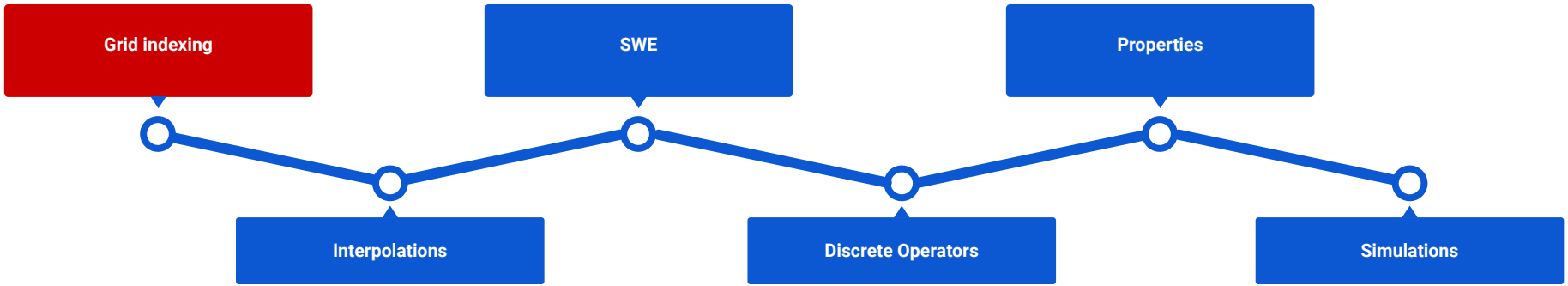
Overview

Today's class:

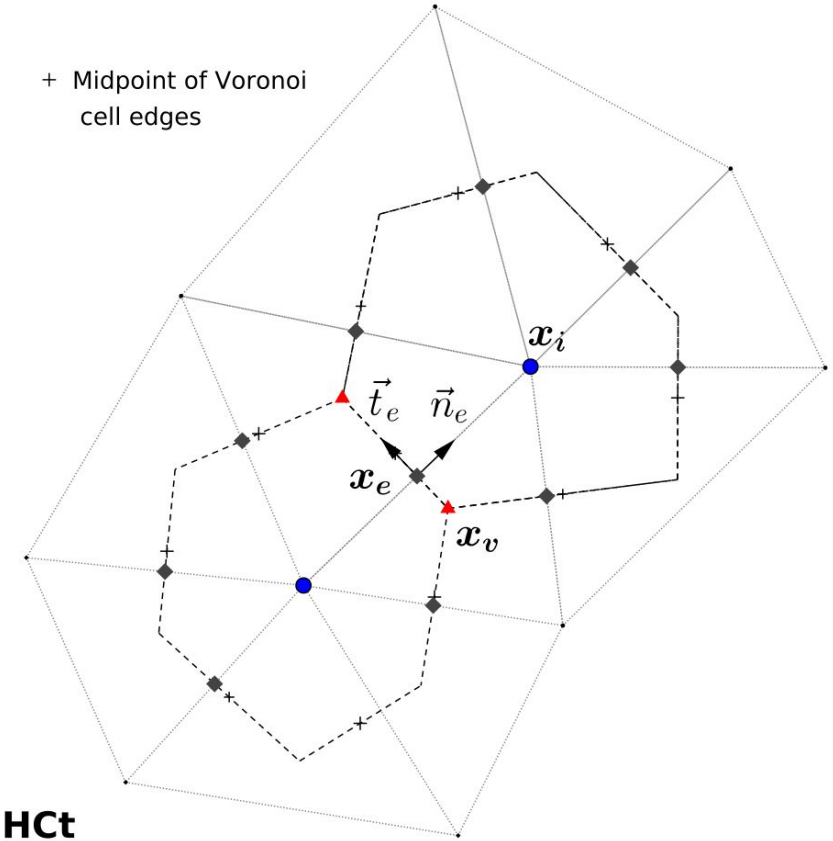


Overview

Today's class:



Grid Structures



HCt

Voronoi/Delaunay grid

Primal Grid (Voronoi):
- Cells Nodes (Voronoi generators)
- Cell Vertices (Triangle circumcenters)
- Cell Edges (connects triangle circumcenters)

Dual Grid (Delaunay):
- Triangles circumcenters
- Triangle vertices (Cell nodes)
- Triangle Edges (connects cell nodes)

Dual and Primal Edges:
- Same indexing
- Orthogonal
- Reference tangent and normal vectors

Grid Recall - Jigsaw

Load environment

- `$conda activate maps-tools`

Create an example grid with Jigsaw

- `(mpas-tools) /…/MPAS-BR/grids/utilities/jigsaw$ python3 spherical_grid.py -g unif -o unif240km -r 240 -l 240`

Check the grid created (resolutions) - Plot.

- `(mpas-tools) /…/MPAS-BR/post_proc/py/grid_maps$ python3 mpas_plot_grid.py -g ../../../grids/utilities/jigsaw/unif240km/unif240km_mpas.nc -o ../../../grids/utilities/jigsaw/unif240km/unif240km_mpas.jpg`

Netcdf grid files

```
pedrosp@ppeixoto:~/ppdata/Work/Reps/MPAS-BR/grids/grids/x1.2562$ ncdump -h x1.2562.grid.nc
netcdf x1.2562.grid {
dimensions:
        nCells = 2562 ;
        nVertices = 5120 ;
        nEdges = 7680 ;
        maxEdges = 10 ;
        maxEdges2 = 20 ;
        TWO = 2 ;
        vertexDegree = 3 ;
        codeLen = 1 ;
variables:
        int indexToCellID(nCells) ;
                indexToCellID:units = "-" ;
                indexToCellID:long_name = "Mapping from local array index to global cell
        double latCell(nCells) ;
                latCell:units = "rad" ;
                latCell:long_name = "Latitude of cells" ;
        double lonCell(nCells) ;
                lonCell:units = "rad" ;
                lonCell:long_name = "Longitude of cells" ;
        double xCell(nCells) ;
                xCell:units = "m" ;
                xCell:long_name = "Cartesian x-coordinate of cells" ;
        double yCell(nCells) ;
                yCell:units = "m" ;
                yCell:long_name = "Cartesian y-coordinate of cells" ;
        double zCell(nCells) ;
                zCell:units = "m" ;
                zCell:long_name = "Cartesian z-coordinate of cells" ;
        int indexToVertexID(nVertices) ;
                indexToVertexID:units = "-" ;
                indexToVertexID:long_name = "Mapping from local array index to global ver
        double latVertex(nVertices) ;
                latVertex:units = "rad" ;
                latVertex:long_name = "Latitude of vertices" ;
        double lonVertex(nVertices) ;
                lonVertex:units = "rad" ;
                lonVertex:long_name = "Longitude of vertices" ;
        double xVertex(nVertices) ;
                xVertex:units = "m" ;
                xVertex:long_name = "Cartesian x-coordinate of vertices" ;
        double yVertex(nVertices) ;
                yVertex:units = "m" ;
                yVertex:long_name = "Cartesian y-coordinate of vertices" ;
        double zVertex(nVertices) ;
```

```
                verticesOnEdge:long_name = "IDs of the two vertex endpoints of an edge" ;
        int nEdgesOnEdge(nEdges) ;
                nEdgesOnEdge:units = "-" ;
                nEdgesOnEdge:long_name = "Number of edges involved in reconstruction of tangential velocity for an edge" ;
        int edgesOnEdge(nEdges, maxEdges2) ;
                edgesOnEdge:units = "-" ;
                edgesOnEdge:long_name = "IDs of edges involved in reconstruction of tangential velocity for an edge" ;
        double areaCell(nCells) ;
                areaCell:units = "m^2" ;
                areaCell:long_name = "Spherical area of a Voronoi cell" ;
        double areaTriangle(nVertices) ;
                areaTriangle:units = "m^2" ;
                areaTriangle:long_name = "Spherical area of a Delaunay triangle" ;
        double kiteAreasOnVertex(nVertices, vertexDegree) ;
                kiteAreasOnVertex:units = "m^2" ;
                kiteAreasOnVertex:long_name = "Intersection areas between primal (Voronoi) and dual (triangular) mesh cells" ;
        double dvEdge(nEdges) ;
                dvEdge:units = "-" ;
                dvEdge:long_name = "Spherical distance between vertex endpoints of an edge" ;
        double dcEdge(nEdges) ;
                dcEdge:units = "m" ;
                dcEdge:long_name = "Spherical distance between cells separated by an edge" ;
        double angleEdge(nEdges) ;
                angleEdge:units = "rad" ;
                angleEdge:long_name = "Angle between local north and the positive tangential direction of an edge" ;
        double weightsOnEdge(nEdges, maxEdges2) ;
                weightsOnEdge:units = "-" ;
                weightsOnEdge:long_name = "Weights used in reconstruction of tangential velocity for an edge" ;
        double meshDensity(nCells) ;
                meshDensity:units = "unitless" ;
                meshDensity:long_name = "Mesh density function (used when generating the mesh) evaluated at a cell" ;
        double nominalMinDc ;
                nominalMinDc:units = "m" ;
                nominalMinDc:long_name = "Nominal minimum dcEdge value where meshDensity == 1.0" ;
        char densityFunctionCode(codeLen) ;
                densityFunctionCode:units = "-" ;
                densityFunctionCode:long_name = "Source code for mesh density function" ;

// global attributes:
                :mesh_spec = "1.1" ;
                :on_a_sphere = "YES" ;
                :sphere_radius = 1. ;
                :is_periodic = "NO" ;
                :x_period = 0. ;
                :y_period = 0. ;
                :file_id = "5d483ba2" ;
```

MPAS grid files and coding structure

Data structure

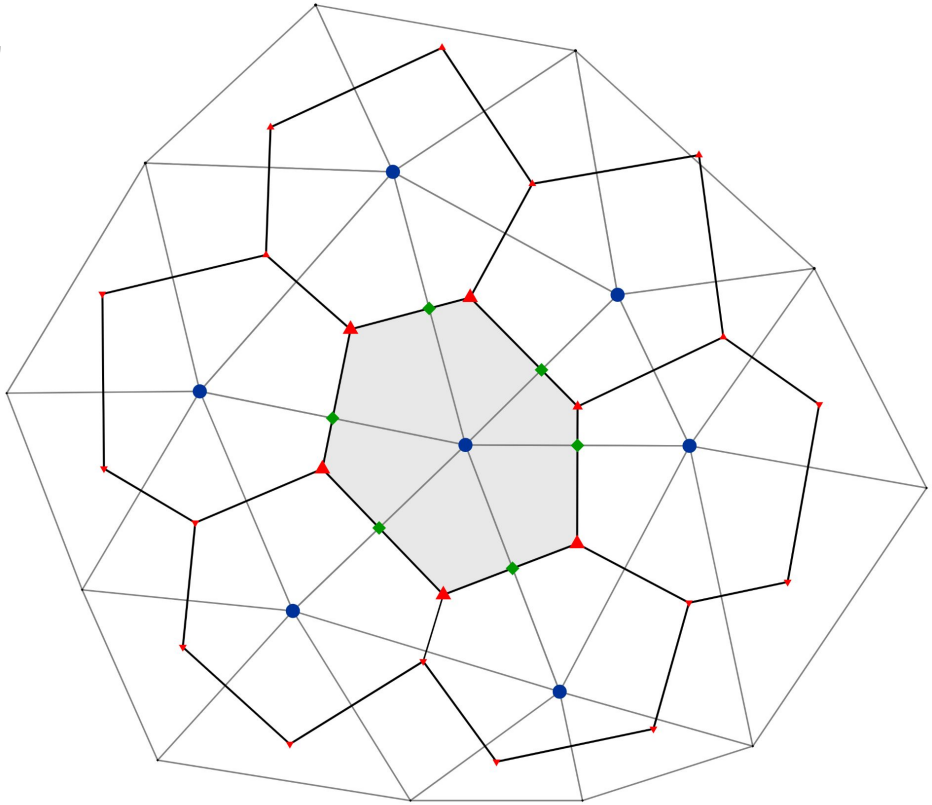
Geometric Point Locations

- Cell: 'latCell', 'lonCell', 'xCell', 'yCell', 'zCell', 'indexToCellID'

- Edges: 'latEdge', 'lonEdge', 'xEdge', 'yEdge', 'zEdge', 'indexToEdgeID'

- Vertices (Triangle circumcentres): 'latVertex', 'lonVertex', 'xVertex', 'yVertex', 'zVertex', 'indexToVertexID'

IndexToXXXX : Converts local indexing to global indexing (to allow parallelism, the grid is split into blocks)



Data structure

MPAS grid files and coding structure

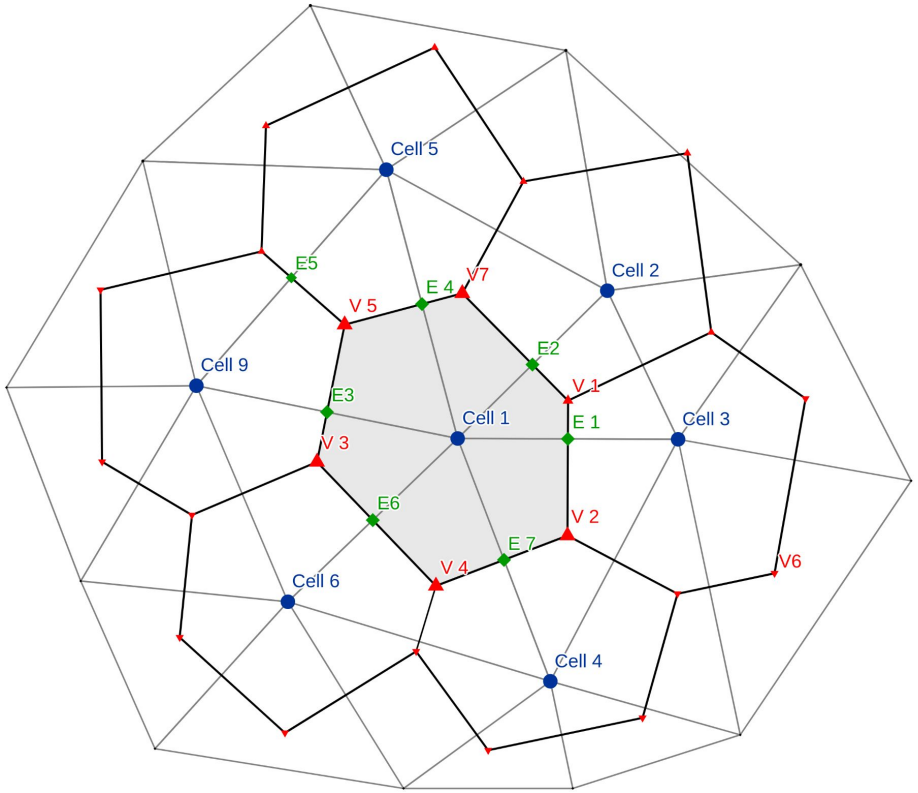
Connections:

- Quantities: 'nEdgesOnCell'

- Indexing: cellsOnCell', 'edgesOnCell', 'verticesOnCell', 'edgesOnEdge', 'cellsOnEdge', 'verticesOnEdge', 'cellsOnVertex', 'edgesOnVertex'

Example:

- nEdgesOnCell(1) = 6
- cellsOnCell(1) = (3, 2, 5, 9, 6, 4)
- edgesOnCell(1) = (1, 2, 4, 3, 6, 7)
- verticesOnCell(1) = (1, 7, 5, 3, 4, 2)
- cellsOnEdge(1) = (1, 3)
- verticesOnEdge(1) = (1, 2)
- cellsOnVertex(1) = (1, 3, 2)
- edgesOnVertex(5) = (4, 5, 3)

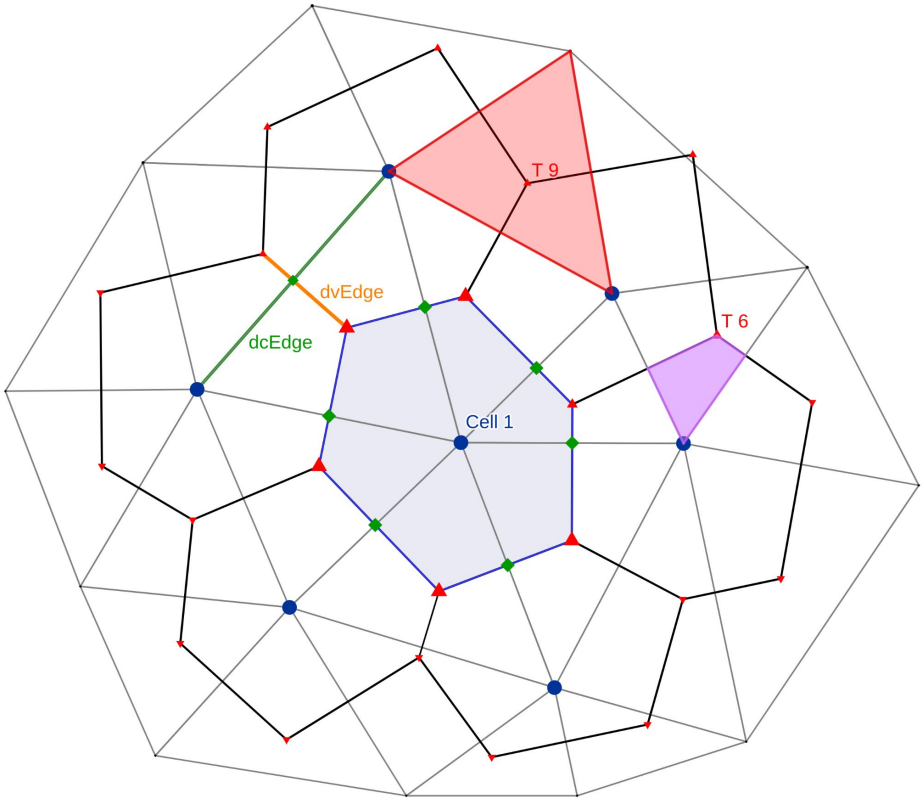


Data structure

MPAS grid files and coding structure

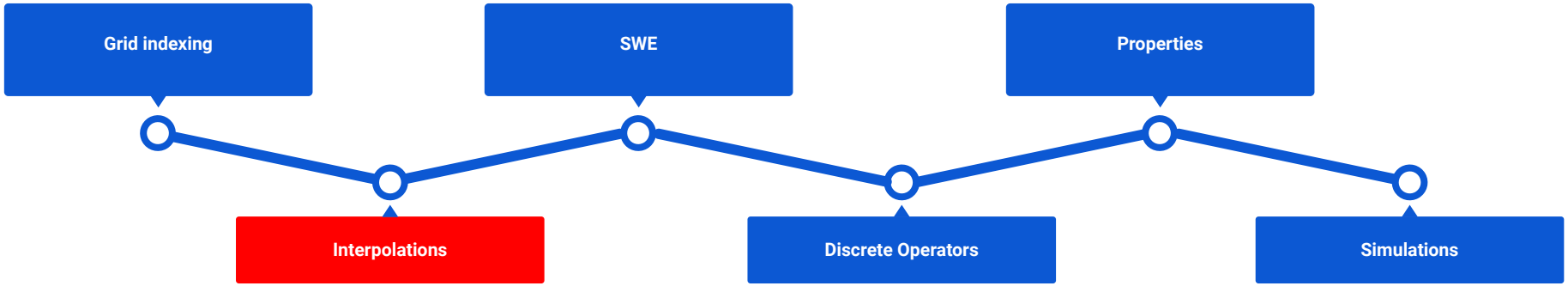
Properties:

- Area: 'areaCell', 'areaTriangle', 'kiteAreasOnVertex'
- Length: 'dcEdge', 'dvEdge'
- Quality: 'cellQuality', 'gridSpacing', 'triangleQuality', 'triangleAngleQuality', 'obtuseTriangle', 'meshDensity'
- Others: 'angleEdge', 'weightsOnEdge'



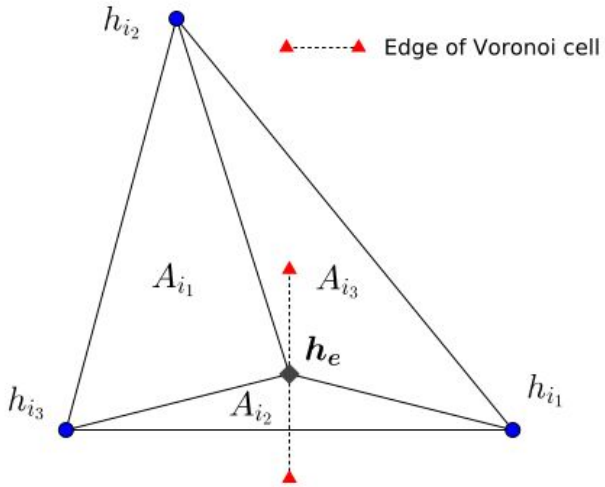
Overview

Today's class:

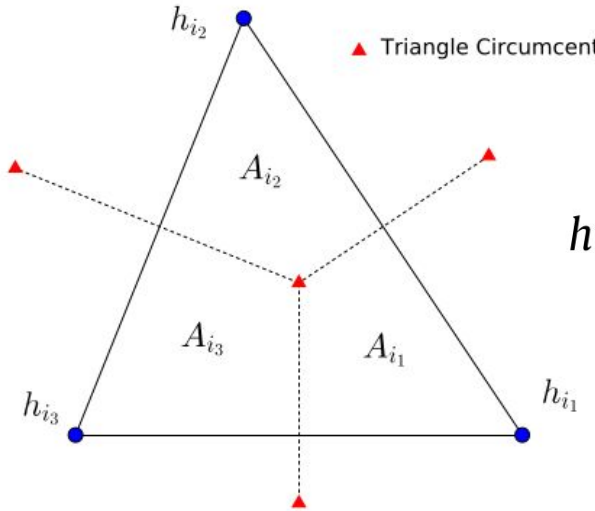


Scalar Data interpolation

From triangle vertices (cell nodes) to triangle centres:



Barycentric Coordinate Areas



Dual/Primal Intersection Areas

$$h_v = \frac{1}{A_k} \sum_i a_{iv} h_i,$$

Used in MPAS (Exact for constant fields)

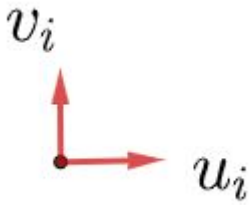
$$h_e = \sum_i \lambda_i(x_e) h_i, \quad \lambda_i(x_e) = \frac{a_i(x_e)}{A_k},$$

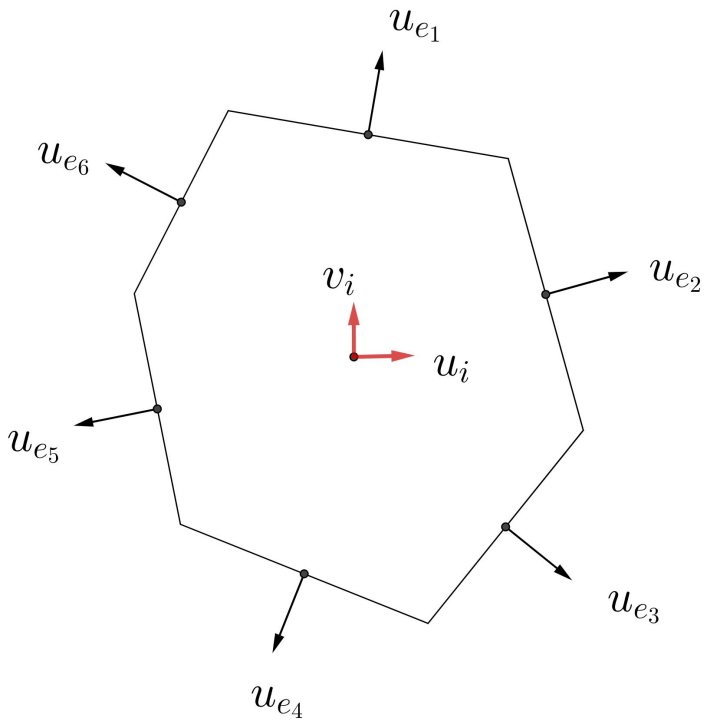
Barycentric: Exact for linear fields

Vector Reconstructions

Given normal edge velocities: u_e

How to reconstruct zonal/meridional velocities at cell centre?

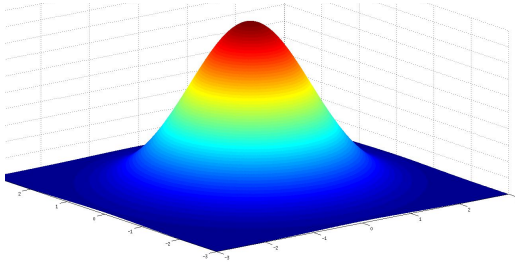


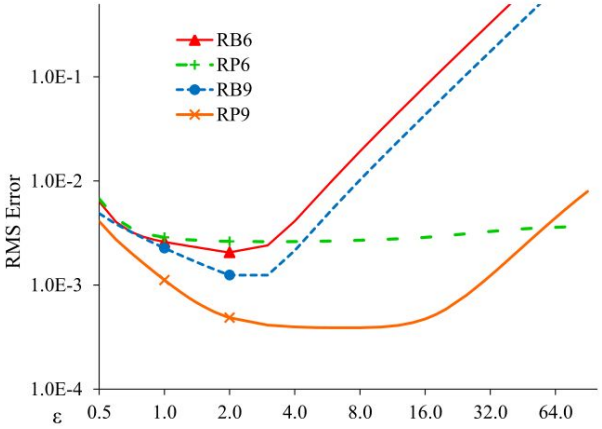


Vector Reconstruction

MPAS uses Radial Basis Functions (RBF)

$$\vec{u}(\vec{x}) = \sum_{i=1}^{k} \lambda_i \phi_i(\vec{x}) \vec{n}_i, \qquad \phi(r) = e^{-(\epsilon r)^2}$$





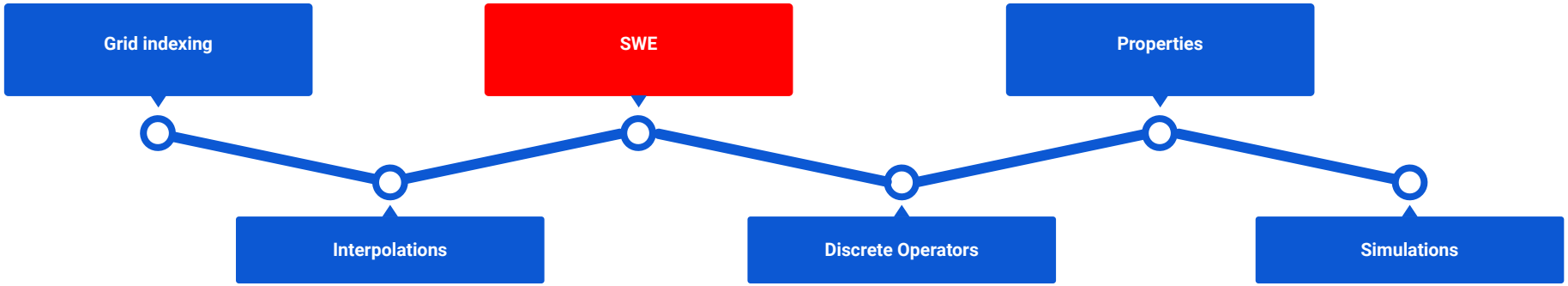
Solves a linear system:

$$\sum_{i=1}^{k} \lambda_i \phi_i(\vec{x}_j) \vec{n}_i \cdot \vec{n}_j = u_j$$

See: Peixoto, PS and Barros, SRM, 2014 : On vector field reconstructions for semi-Lagrangian transport methods on geodesic staggered grids (Journal of Computational Physics) – PDF, DOI

Overview

Today's class:



Ultimate goal

Hydrostatic (primitive) equations: inadequate below ~20-10km horizontal resolution

To capture explicit convection (resolution << 10km):

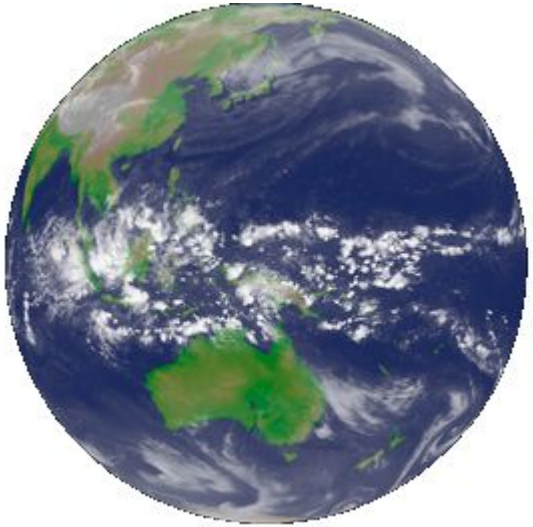


Image: NICAM Model (Japan)

Compressible Euler equations for atmosphere (ideal gas)

$$\frac{D\mathbf{u}}{Dt} = -2\,\boldsymbol{\Omega} \times \mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{g} + \mathbf{F}_r \quad \text{(Momentum)}$$

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{u} \quad \text{(Continuity)}$$

$$c_v \frac{DT}{Dt} = -\frac{p}{\rho}\nabla \cdot \mathbf{u} \quad \text{(Thermodynamics)}$$

- $\mathbf{u} = (u, v, w)$: wind velocity
- p: pressure
- ρ: density
- T: temperature
- $D/Dt = \partial/\partial t + \mathbf{u} \cdot \nabla$: Material derivative

Intermediate step

3D Incompressibility (constant density):

$$\nabla \cdot \vec{v} = 0, \quad (\rho = \rho_0)$$

Hydrostatic balance:

$$\frac{\partial p}{\partial z} = -\rho_0 g,$$

Free surface $h_T(x, y, t)$ where $h_T = h_0 + h$, with $h_0(x, y)$ topography, $h(x, y, t)$ fluid depth.

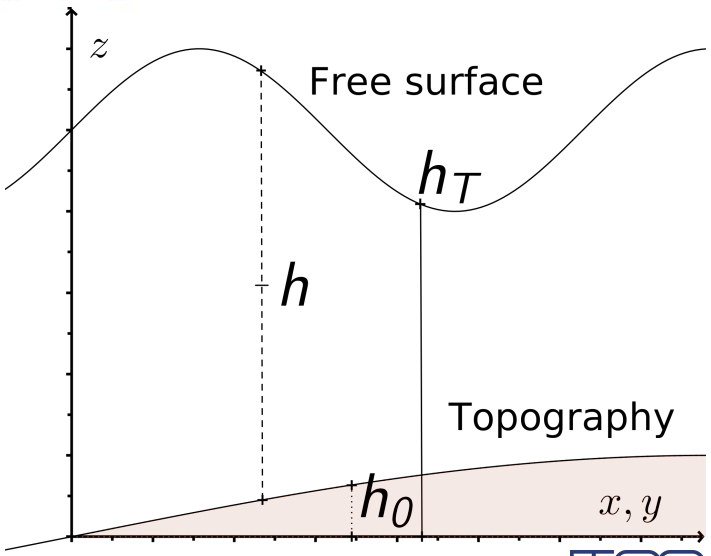
$$\int_z^{h_T} \frac{\partial p}{\partial z} dz = -\int_z^{h_T} \rho_0 g dz$$

$$p(z) = \rho_0 g(h_T - z) + \underbrace{p(h_T)}_{\text{Constant}}$$

Pressure gradient:

$$\nabla p = \rho_0 g \nabla h_T$$

Vertical Integration (Mean Flow)



Shallow Water Equations

Horizontal Momentum Equations ($\vec{v} = (u, v)$):

$$\frac{\partial \vec{v}}{\partial t} + \underbrace{(\vec{v} \cdot \nabla)\vec{v}}_{\text{nonlinear advection}} = \underbrace{-f\vec{k} \times \vec{v}}_{\text{Coriolis}} \underbrace{-g\nabla(h + h_0)}_{\text{Pressure}}$$

(f-plane approximation)

Shallow Water Equations

3D Incompressibility :

$$\nabla \cdot \vec{v} = \partial_x u + \partial_y v + \partial_z w = 0,$$

$$\partial_z w = -\partial_x u - \partial_y v$$

Free surfave must have w velocity:

$$\frac{Dh_T}{Dt} = w(x, y, h_T, t)$$

Topography height must have w_0 velocity:

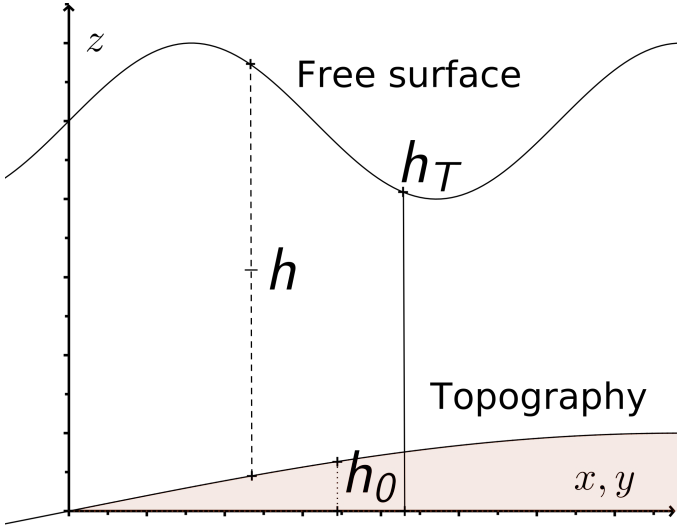
$$\frac{Dh_0}{Dt} = w(x, y, h_0, t) \Rightarrow w(x, y, h_0, t) = \vec{v} \cdot \nabla h_0$$

Integrate 3D incompressibility:

$$\int_{h_0}^{h} \partial_z w dz = -\int_{h_0}^{h} (\partial_x u + \partial_y v) dz$$

$$w_h - w_0 = -(h - h_0)\nabla \cdot \vec{v}$$

Vertical Integration (Mean Flow)



Shallow Water Equations

$$\vec{v}(x,y,t) = (u(x,y,t), v(x,y,t))$$
$$h(x,y,t) = h_T(x,y,t) - h_0(x,y)$$

2D Continuity equation:

$$\frac{\partial h}{\partial t} + \nabla \cdot (h\vec{v}) = 0$$

$$\frac{\partial h}{\partial t} + \underbrace{\vec{v} \cdot \nabla h}_{\text{transport}} = \underbrace{-h \nabla \cdot \vec{v}}_{\text{flow divergence}}$$

2D Horizontal Momentum Equations:

$$\frac{\partial \vec{v}}{\partial t} + \underbrace{(\vec{v} \cdot \nabla)\vec{v}}_{\text{nonlinear advection}} = \underbrace{-f\vec{k} \times \vec{v}}_{\text{Coriolis}} \underbrace{-g\nabla(h + h_0)}_{\text{Pressure}}$$

(f-plane approximation)

Shallow Water Equations on the Sphere

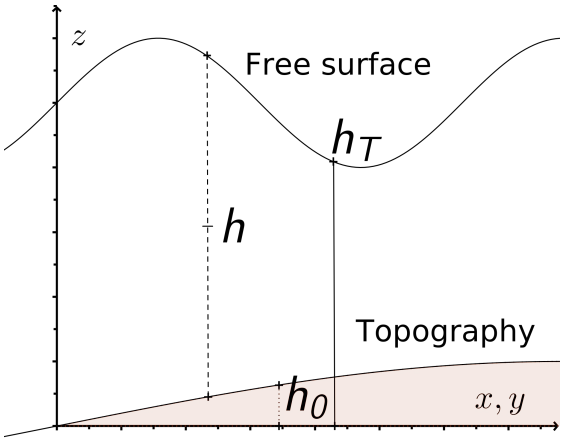
Using that:

$$(\vec{v}\cdot\nabla)\vec{v} = (\nabla\times\vec{v})\times\vec{v} + \frac{1}{2}\nabla(\vec{v}\cdot\vec{v})$$

Vector invariant form:

$$\frac{\partial h}{\partial t} + \nabla\cdot(h\vec{v}) = 0$$

$$\frac{\partial \vec{v}}{\partial t} + \eta\vec{k}\times\vec{v} = -\nabla K - g\nabla(h+h_0)$$



- $\vec{v}$ is the 3D velocity vector tangent to the sphere
- $f = 2\Omega\sin(\theta)$
- ∇ gradient on tangent plane
- $\vec{k}$ unit vector point out of the sphere

- Kinetic Energy $K = \frac{\vec{v}\cdot\vec{v}}{2}$
- Absolute Vorticity

$$\eta = \vec{k}\cdot\nabla\times\vec{u} + f$$

Many properties: Conserves mass, energy, enstrophy, Coriolis neutral in energy budget, normal modes - inertia-gravity waves, Rossby waves, etc...

Discrete shallow Water Equations

Vector invariant form

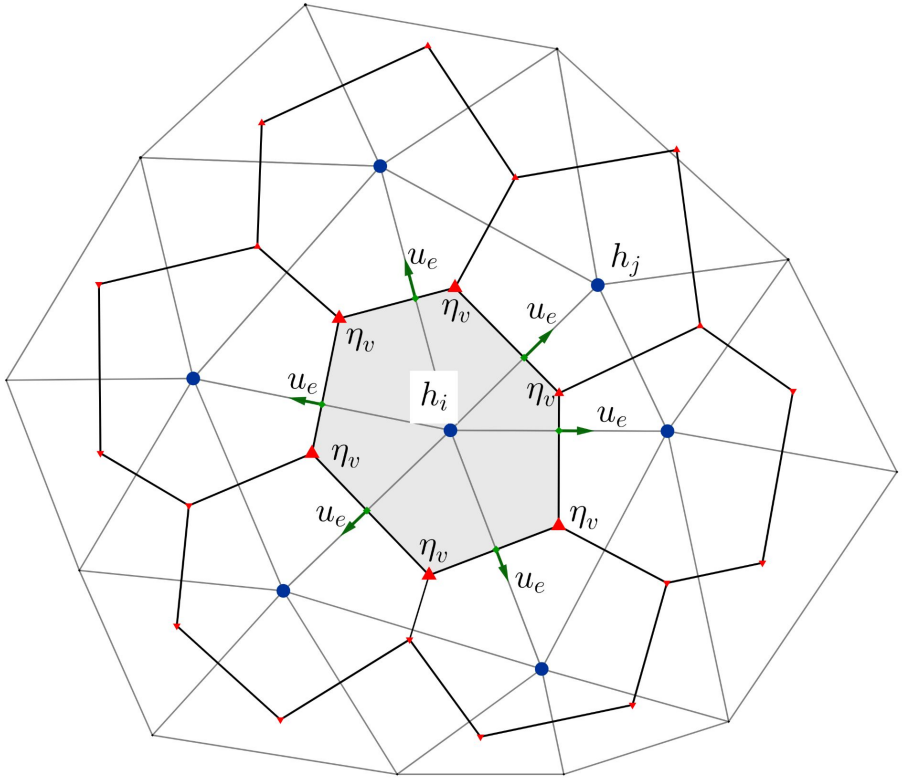
$$\frac{\partial h}{\partial t} + \nabla \cdot (h\vec{u}) = 0,$$

$$\frac{\partial \vec{u}}{\partial t} + qh\vec{u}^{\perp} = -g\nabla(h+b) - \nabla K,$$

Discrete model

$$\begin{aligned} \frac{\partial h_i}{\partial t} &= -D_i, \\ \frac{\partial u_e}{\partial t} &= -Q_e^{\perp} - G_e, \end{aligned}$$

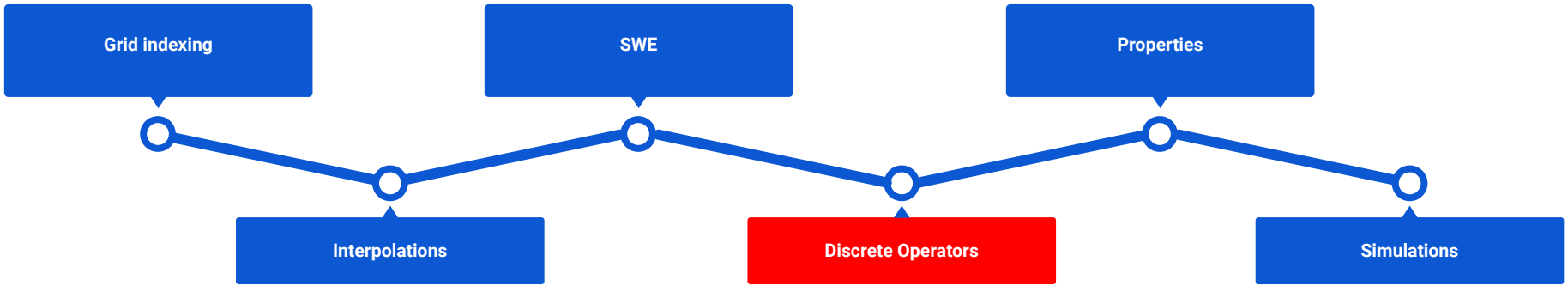
- D_i discrete version of $\nabla \cdot (h\vec{u})$
- G_e discrete version of $\nabla(g(h+b)+K)$
- $Q_e^{\perp}$ discrete version of $qh\vec{u}^{\perp}$



Use staggered Voronoi grid (C-Grid):
Cell centres: Depth h_i Edges: Velocity (normal) u_e Triangles CC: Vorticity (PV) $q = \eta/h$

Overview

Today's class:



Divergence

Wiki

$$\underbrace{\int \cdots \int_U}_{n} \nabla \cdot \mathbf{F}\, \mathrm{d}V = \underbrace{\oint \cdots \oint_{\partial U}}_{n-1} \mathbf{F} \cdot \mathbf{n}\, \mathrm{d}S$$

This equation is also known as the divergence theorem.

Finite Volume based:
Similar to what was done in 2D Cartesian grids!

Key concepts:

- Point value divergence is approximated by cell integral (average): $\nabla \cdot (\vec{v}h) \approx \frac{1}{\|\Omega\|} \int_\Omega \nabla \cdot (\vec{v}h)\, dA$
- Divergence Theorem - from area integral to edge integral:

$$\frac{1}{\|\Omega\|} \int_\Omega \nabla \cdot (\vec{v}h)\, dA = \frac{1}{\|\Omega\|} \int_{\partial\Omega} h\vec{v} \cdot \vec{n}\, dl = \frac{1}{\|\Omega\|} \sum_e \int_{\gamma_e} h\vec{v} \cdot \vec{n}\, dl$$

- Approximate each edge integral with midpoint rule:

$$\frac{1}{\|\Omega\|} \sum_e \int_{\gamma_e} h\vec{v} \cdot \vec{n}\, dl = \frac{1}{\|\Omega\|} \sum_e h_e u_e n_{ei} l_e$$

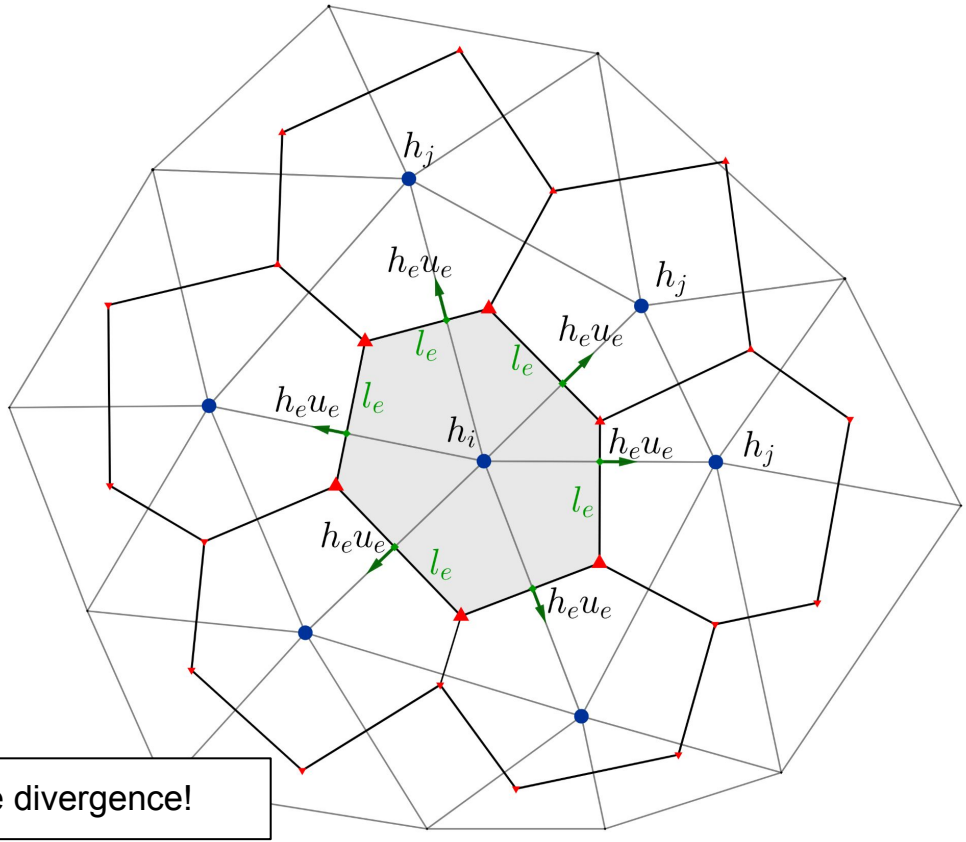
- Interpolate depth to edges: $h_e = (h_i + h_j)/2$
- Normal direction correction: $n_{ei} = \pm 1$

Divergence

$$D_i = \frac{1}{A_i} \sum_e h_e u_e n_{ei} l_e$$

Cell Area: A_i

Cell edge length: l_e



>> If the cell is a rectangle, we recover the 2D discrete divergence!

Divergence in MPAS

$$D_i = \frac{1}{A_i} \sum_e h_e u_e n_{ei} l_e$$

n_{ei}

l_e

$h_e u_e$

```
4728
4729        do iCell=cellStart,cellEnd
4730           h_divergence(1:nVertLevels,iCell) = 0.0
4731           do i=1,nEdgesOnCell(iCell)
4732              iEdge = edgesOnCell(i,iCell)
4733              edge_sign = edgesOnCell_sign(i,iCell) * dvEdge(iEdge)
4734 !DIR$ IVDEP
4735              do k=1,nVertLevels
4736                 h_divergence(k,iCell) = h_divergence(k,iCell) + edge_sign * ru(k,iEdge)
4737              end do
4738           end do
4739        end do
```

$\sum_e$

```
4743        do iCell = cellStart,cellEnd
4744           r = invAreaCell(iCell)
4745           do k = 1,nVertLevels
4746              h_divergence(k,iCell) = h_divergence(k,iCell) * r
4747           end do
4748        end do
```

$\frac{1}{A_i}$

https://github.com/pedrospeixoto/MPAS-BR/blob/master/src/core_atmosphere/dynamics/mpas_atm_time_integration.F

Gradient

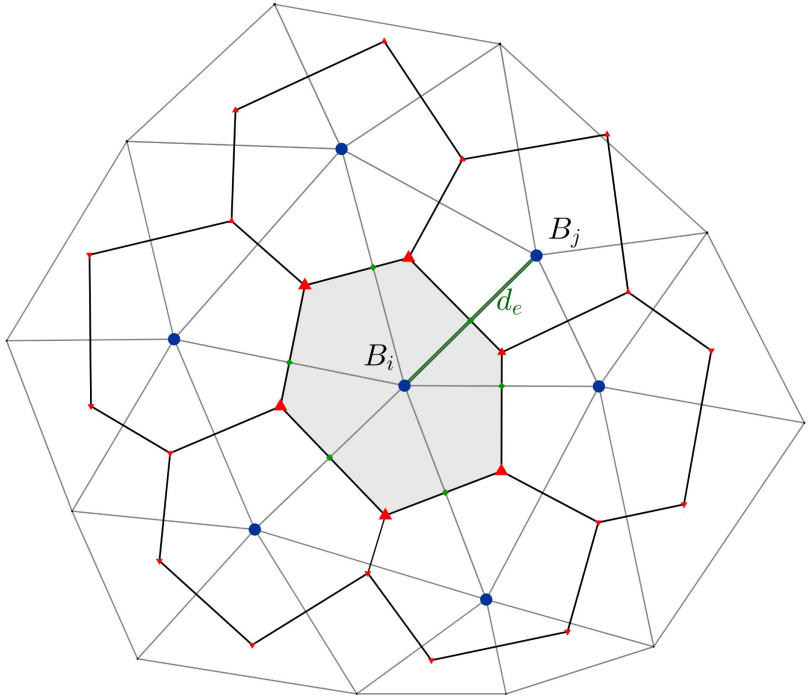
Finite Volume based: Similar to what was done in 2D Cartesian grids!

G_e Discrete gradient of the gradient of the Bernoulli potential $\nabla(g(h+b)+K) = \nabla B$

Key concepts:

- Integrate (average) gradient along a dual edge (that connects cell nodes)
- Use the fundamental theorem of Calculus (line integral converts to point values)

$$\nabla B \approx \frac{1}{d_e}\int_{e'} \nabla B \, dl = \frac{1}{d_e}(B_j - B_i)$$



Gradient in MPAS

In the 3D model, we have the gradient of Kinetic Energy at this point of the code:

$$\nabla K \approx \frac{1}{d_e} \int_{e'} \nabla K \, dl = \frac{1}{d_e}(K_j - K_i)$$

$$K_j - K_i$$

```
4823        do k=1,nVertLevels
4824
4825          ! horizontal ke gradient and vorticity terms in the vector invariant formulation
4826          ! of the horizontal momentum equation
4827          tend_u(k,iEdge) = tend_u(k,iEdge) + rho_edge(k,iEdge)* (q(k) - (ke(k,cell2) - ke(k,cell1))       &
4828                                                                 * invDcEdge(iEdge))                      &
4829                                          - u(k,iEdge)*0.5*(h_divergence(k,cell1)+h_divergence(k,cell2))
```

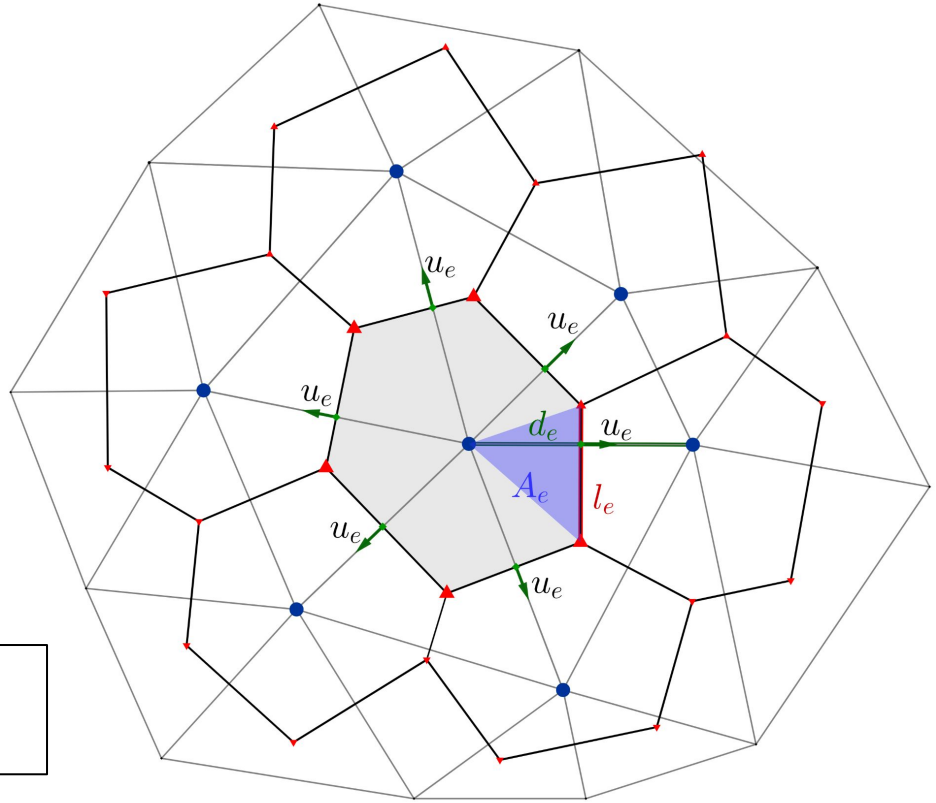
$$\frac{1}{d_e}$$

https://github.com/pedrospeixoto/MPAS-BR/blob/master/src/core_atmosphere/dynamics/mpas_atm_time_integration.F

Kinetic energy

$$K_i = \frac{1}{4A_i} \sum_e l_e \, d_e \, u_e^2,$$

$$A_e \approx \frac{l_e d_e}{4}$$ Edge “area”



The edge has an associated “area” (blue): this is key to energy conservation.

Vorticity

$$\eta = \vec{k} \cdot \nabla \times \vec{u} + f$$

Finite Volume based: Similar to what was done in 2D Cartesian grids on the dual grid!

Key concepts:

- Point value vorticity is approximated by triangle integral (average): $\vec{k} \cdot \nabla \times \vec{u} \approx \frac{1}{\|T\|} \int_T \vec{k} \cdot \nabla \times \vec{u}\, dA$
- “Divergence” (Circulation) Theorem - from area integral to edge integral:

$$\frac{1}{\|T\|} \int_T \vec{k} \cdot \nabla \times \vec{u}\, dA = \frac{1}{\|T\|} \int_{\partial T} \vec{v} \cdot \vec{t}\, dl = \frac{1}{\|T\|} \sum_{e'} \int_{e'} \vec{v} \cdot \vec{t}\, dl$$

- Approximate each edge integral with midpoint rule:

$$\frac{1}{\|T\|} \sum_{e'} \int_{e'} \vec{v} \cdot \vec{t}\, dl = \frac{1}{\|T\|} \sum_{e'} u_e t_{ei} d_e$$

- Counter-clockwise tangent direction correction: $t_{ei} = \pm 1$

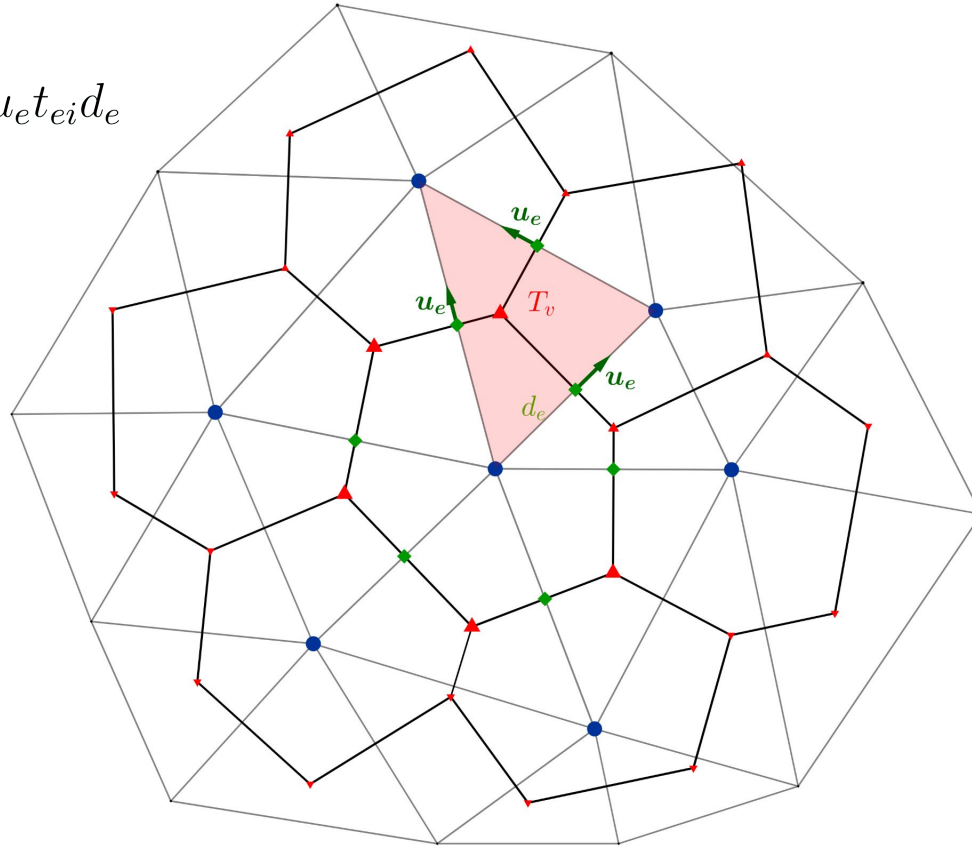
Vorticity

$$\vec{k} \cdot \nabla \times \vec{u} \approx \frac{1}{A_v} \int_T \vec{k} \cdot \nabla \times \vec{u} \, dA \approx \frac{1}{A_v} \sum_{e'} u_e t_{ei} d_e$$

A_v Triangle area

d_e Triangle edge length

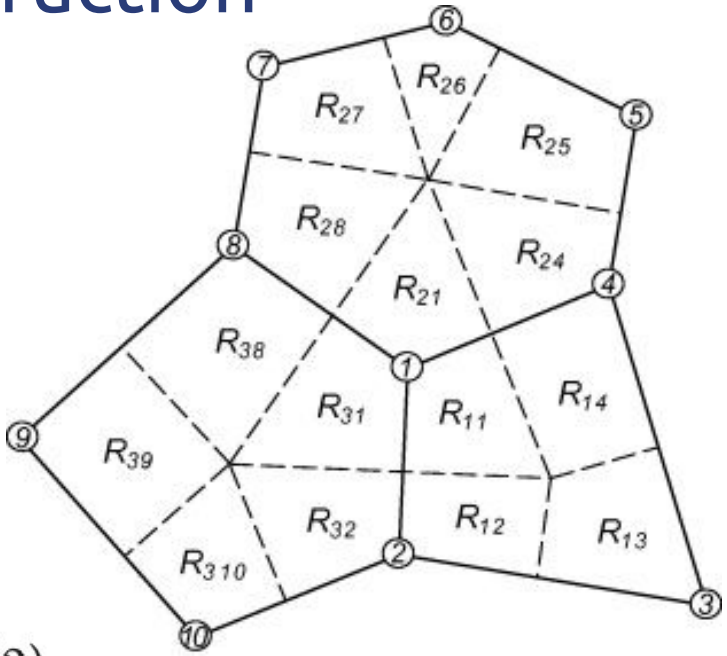
The normal velocity vectors relative to Voronoi cells are tangent velocity vectors on triangles!!!



Tangential Reconstruction

Given normal velocities, the Coriolis term requires a perpendicular (tangential) velocity!!!

$$u_e^{\perp} = \sum_{e' \in \mathrm{NB}(e)} w_{ee'} u_{e'}$$



$$w_{e\,e'}\,t_{e\,v_2} = \left(\sum_v R_{i\,v} - \alpha_{e'\,i\,v_1}\right) n_{e'\,i} = \left(\sum_v R_{i\,v} - 1/2\right) n_{e'\,i}$$

Thuburn, J., T. Ringler, J. Klemp and W. Skamarock, 2009: Numerical representation of geostrophic modes on arbitrarily structured C-grids, Journal of Computational Physics, 2009

Tangential velocity in MPAS

```
5716          !
5717          ! Compute v (tangential) velocities following Thuburn et al JCP 2009
5718          ! The tangential velocity is only used to compute the Smagorinsky coefficient
5719
5720          reconstruct_v = .true.
5721          if(present(rk_step)) then
5722            if(rk_step /= 3) reconstruct_v = .false.
5723          end if
5724
5725          if (reconstruct_v) then
5726            do iEdge = edgeStart,edgeEnd
5727              v(1:nVertLevels,iEdge) = 0.0
5728              do i=1,nEdgesOnEdge(iEdge)
5729                eoe = edgesOnEdge(i,iEdge)
5730      !DIR$ IVDEP
5731                do k = 1,nVertLevels
5732                  v(k,iEdge) = v(k,iEdge) + weightsOnEdge(i,iEdge) * u(k, eoe)
5733                end do
5734              end do
5735            end do
5736          end if
```

u_e

w_{ee}

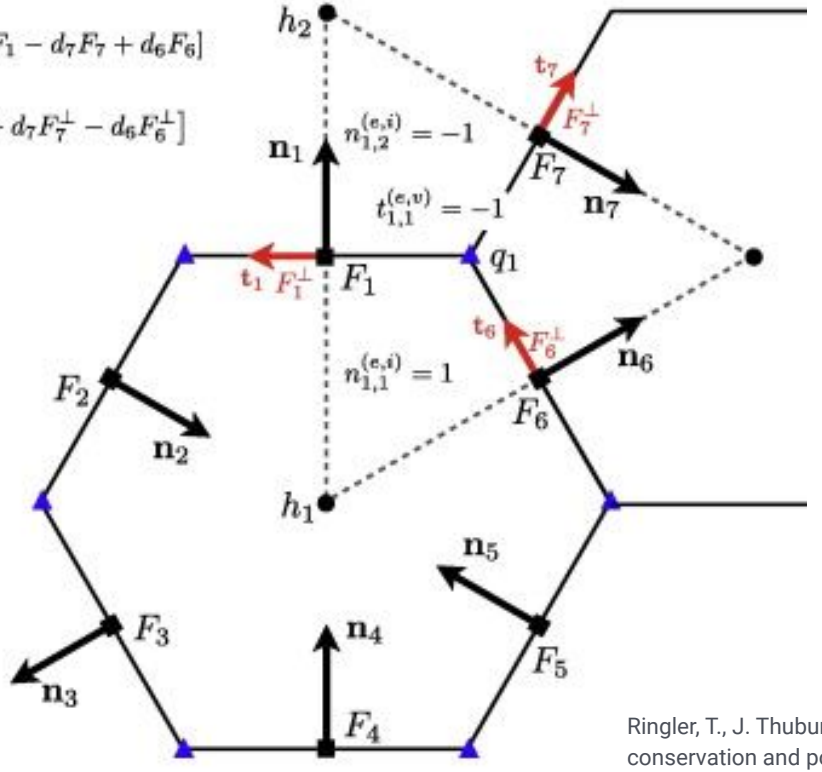
Summary of Operators

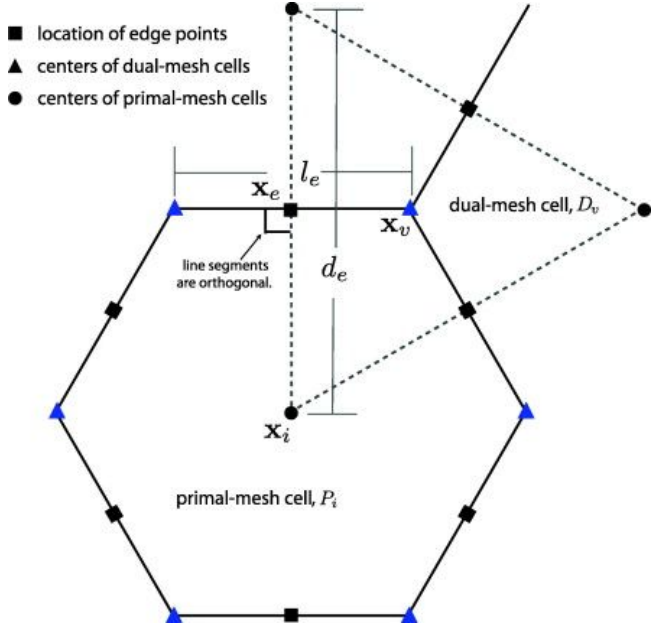
$$[\nabla \cdot F\mathbf{n}]_1^{(i)} = \frac{1}{A^{(i)}}\left[l_1F_1 - l_2F_2 + l_3F_3 - l_4F_4 - l_5F_5 + l_6F_6\right]$$

$$[\mathbf{k} \cdot \nabla \times F\mathbf{n}]_1^{(v)} = \frac{1}{A_1^{(v)}}\left[-d_1F_1 - d_7F_7 + d_6F_6\right]$$

$$\left[\nabla \cdot F^{\perp}\mathbf{t}\right]_1^{(v)} = \frac{1}{A_1^{(v)}}\left[d_1F_1^{\perp} + d_7F_7^{\perp} - d_6F_6^{\perp}\right]$$

$$[\nabla h]_1^{(e)} = \frac{1}{d_1}\left[h_2 - h_1\right]$$

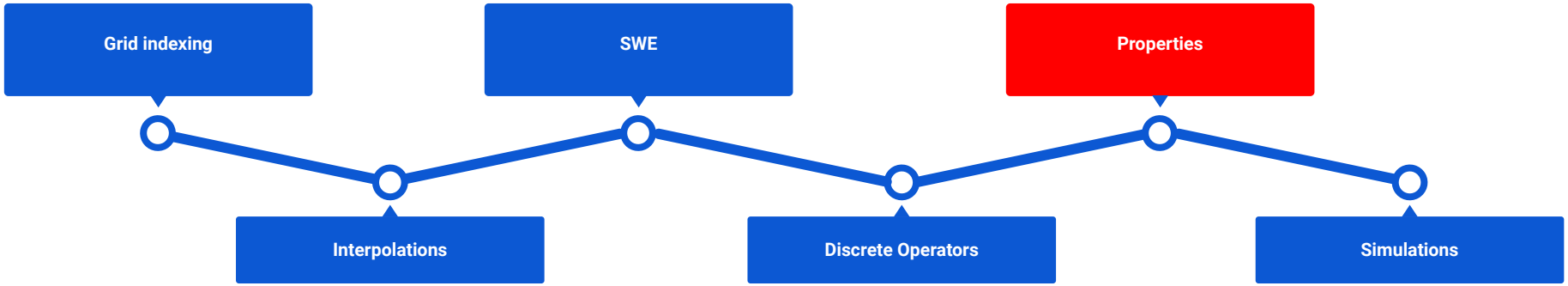




Ringler, T., J. Thuburn, J. Klemp and W. Skamarock, 2010: A unified approach to energy conservation and potential vorticity dynamics on arbitrarily structured C-grids, Journal of Computational Physics

Overview

Today's class:



Desired properties

- Example of desired properties for horizontal shallow water equations:
  - Accurate and stable
  - Scalable (Local operators - no global operations)
  - Mass and energy conservation
  - Accurate representation slow/fast waves (staggering)
  - Curl-free pressure gradient $\nabla \times \nabla \psi = 0$
  - Energy conservation of pressure terms $\vec{v} \cdot \nabla h + h \nabla \cdot \vec{v} = \nabla \cdot (h \vec{v})$
  - Energy conserving Coriolis term $\vec{v} \cdot \vec{v}^{\perp} = 0$

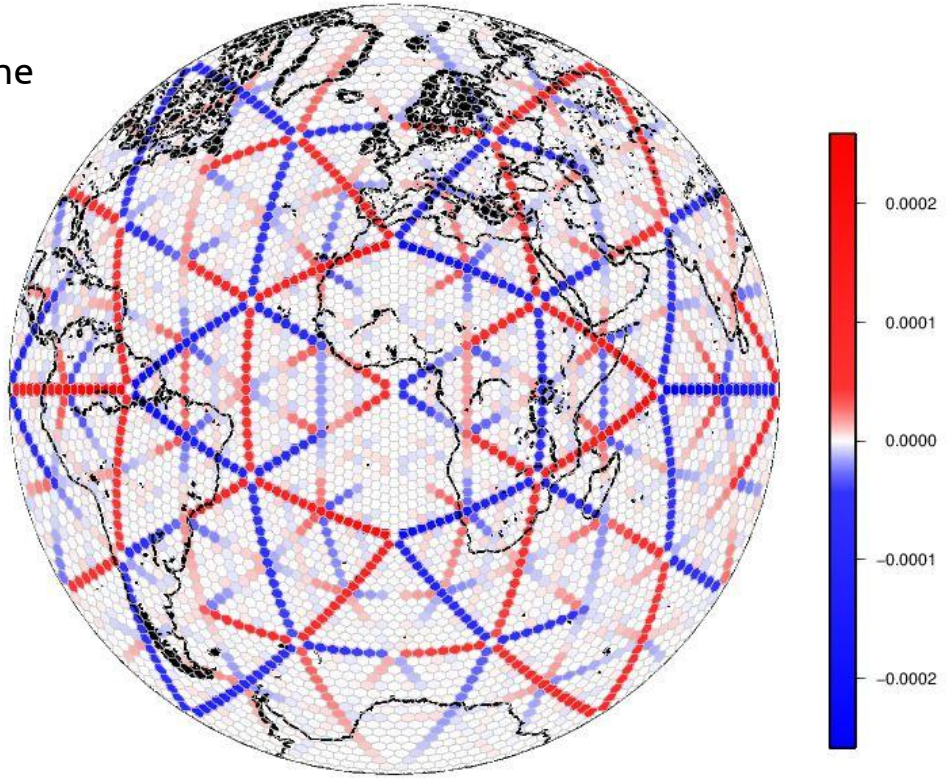
Solved for Finite Differences on Lat-Lon grids (apart from scalability!)
Open problem for Finite Volumes on arbitrary polygonal spherical grids

TRiSK Scheme: Ringler, T.D., Thuburn, J., Klemp, J.B. and Skamarock, W.C., 2010. A unified approach to energy conservation and potential vorticity dynamics for arbitrarily-structured C-grids. Journal of Computational Physics.

Grid Imprinting

- Grid influences numerical errors of Finite Volume

Classic Finite Volume Discretization error for 2D divergence of solid body rotation (should be zero everywhere!)

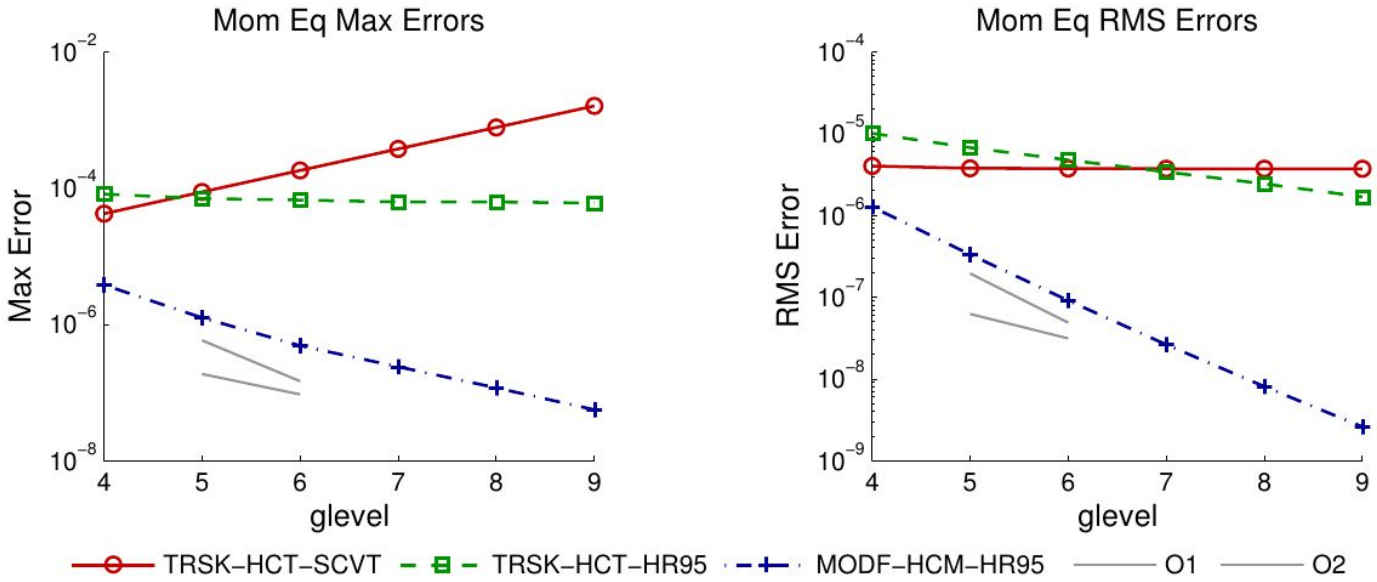


Peixoto, P.S. and Barros, S.R., 2013. Analysis of grid imprinting on geodesic spherical icosahedral grids. Journal of Computational Physics, 237, pp.61-78.

Accuracy

- Accurate and Stable Finite Volume Schemes

  ⇨ Finite Volume Schemes may lose consistency/convergence on irregular grids



Finite Volume scheme (TRiSK - used in MPAS model) truncation error for 2D Momentum Equation

Peixoto, P.S., 2016. Accuracy analysis of mimetic finite volume operators on geodesic grids and a consistent alternative. Journal of Computational Physics, 310, pp.127-160.

Numerical Stability

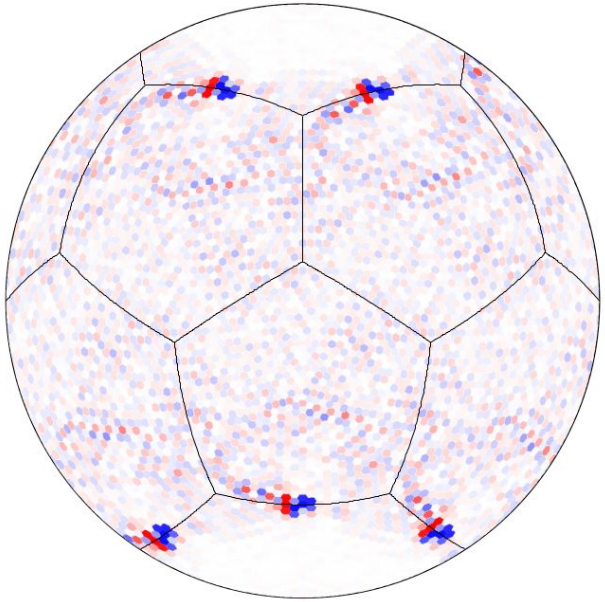
- Energy conserving schemes on polygonal grids use vector relation

$$\vec{v} \cdot \nabla \vec{v} = \nabla K + \zeta \vec{k} \times \vec{v}$$

Equivalently for 2D:

$$uu_x + vu_y = \left(\frac{u^2 + v^2}{2} \right)_x + (v_x - u_y)(-v)$$

$$uv_x + vv_y = \left(\frac{u^2 + v^2}{2} \right)_y + (v_x - u_y)(u)$$

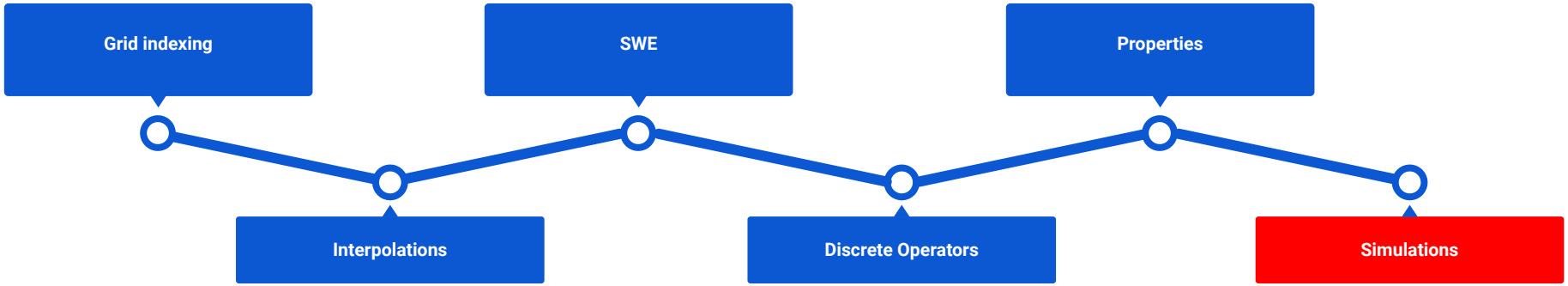


Terms in red cancel analytically, but maybe not numerically...
Lack of numerical cancellation may lead to instability.

Peixoto, P.S., Thuburn, J. and Bell, M.J., 2018. Numerical instabilities of spherical shallow-water models considering small equivalent depths. *Quarterly Journal of the Royal Meteorological Society, 144(710), pp.156-171.*

Overview

Today's class:



Simulations

We will use the iModel code: https://github.com/pedrospeixoto/iModel

Why?

- Simpler to install (requirements: linux and gfortran)
- Simpler dynamics (Advection, Shallow Water, no 3D dynamics)
- Simpler grid generation (native grid generator)

Download:

- https://github.com/pedrospeixoto/iModel/archive/refs/heads/master.zip

Install:

- In a linux terminal, assuming you have gfortran installed, type:
  $ make

Configuration (par/)

mesh.par

iModel > par > mesh.par

```
!Grid parameters
!Number of vertices (may be changed if icos/octg mesh is chosen)
160000
!Kind of mesh use: icos, octg, read, rand
read
!Position: eqs, pol, ran, ref, readref, readref_andes
eqs
!Optimization of mesh: nopt, sprg, scvt, salt, hr95
nopt
!Loadable grid? 0-No; 1-Yes
1
!Show ongoing process details on screen? 0-No; 1-Yes
1
!Test case to be done - uncomment method to be used in imodel.f90
15
!If mesh read from file give filename (Extensions: .xyz cartesian coords, .gmt lon, lat)
icos_pol_scvt_h1_6.xyz
!Hierarchical grid construction: 0-No; 1-Yes; 2-New only; 3-Icos0 Symmetry implied; 4- Both 2 and 3
1
```

Configuration (par/)

mesh.par

Grid

```
iModel > par > mesh.par
1   !Grid parameters
2   !Number of vertices (may be changed it
3   160000
4   !Kind of mesh use: icos, octg, read, r
5   read
6   !Position: eqs, pol, ran, ref, readref
7   eqs
8   !Optimization of mesh: nopt, sprg, scv
9   nopt
10  !Loadable grid? 0-No; 1-Yes
11  1
12  !Show ongoing process details on scree
13  1
14  !Test case to be done - uncomment meth
15  15
16  !If mesh read from file give filename
17  icos_pol_scvt_h1_6.xyz
18  !Hierarchical grid construction: 0-No;
19  1
```

Generate icosahedral grid or read from file

Simple xyz coordinates of grid points read from file

Pre-generated grids here:
https://pedrosp.ime.usp.br/grids/swm/

```
$ cd grid
$ wget https://pedrosp.ime.usp.br/grids/swm/icos_pol_scvt_h1_6.xyz
```

Configuration (par/)

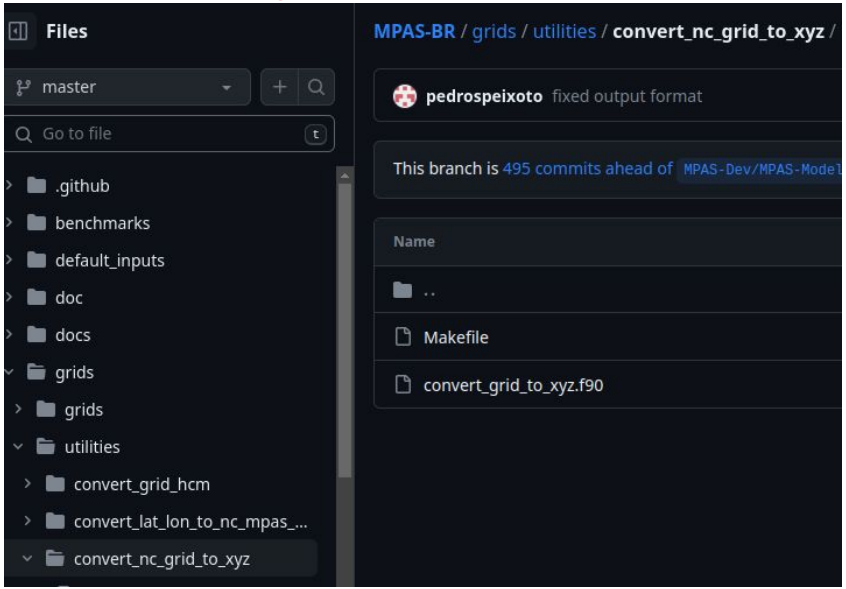
mesh.par

Grid

```
iModel > par > mesh.par
1   !Grid parameters
2   !Number of vertices (may be changed it
3   160000
4   !Kind of mesh use: icos, octg, read, r
5   read
6   !Position: eqs, pol, ran, ref, readref
7   eqs
8   !Optimization of mesh: nopt, sprg, scv
9   nopt
10  !Loadable grid? 0-No; 1-Yes
11  1
12  !Show ongoing process details on scree
13  1
14  !Test case to be done - uncomment meth
15  15
16  !If mesh read from file give filename
17  icos_pol_scvt_h1_6.xyz
18  !Hierarchical grid construction: 0-No;
19  1
```

Use MPAS grids

Convert grid to XYZ text file



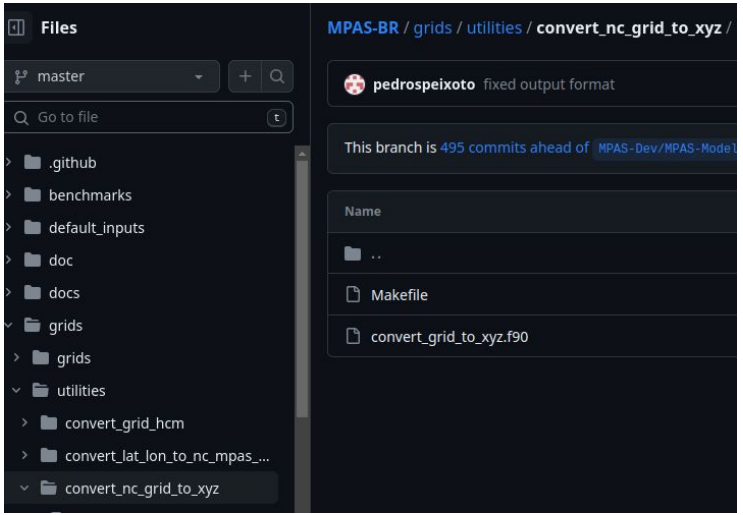
Simple xyz coordinates of grid points read from file

Convert nc grid to XYZ

Use MPAS grids

Convert MPAS grid to XYZ text file

```
10394
      0.0038371408869309            0.0080082619234591            0.9999605712680770
      0.2234697114213758           -0.8323528203382676            0.5071982556675234
      0.8983912821830898            0.0030217708665120            0.4391855792239248
      0.7200291617762784            0.5346702875287391           -0.4423637528388660
      0.2728006195110721            0.8517651783350705            0.4472984495504100
     -0.2708353213026356            0.8508032983905797           -0.4503131978774422
     -0.7239333097203067            0.5258120608778889            0.4465895651633046
```

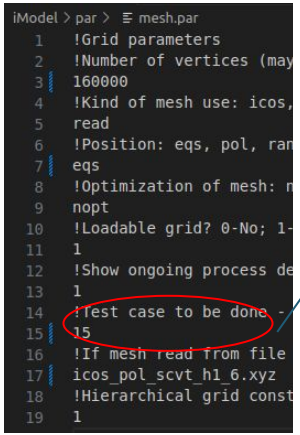



```
(mpas-tools) pedrosp@ppeixoto:/ppdata/ppdata/Work/Reps/MPAS-BR/grids/utilities/convert_nc_grid_to_xyz$ make
gfortran convert_grid_to_xyz.f90 -o convert_grid_to_xyz -ffree-line-length-none -I/home/pedrosp/anaconda3/envs/mpas-tools/include -L/home/pedrosp/anaconda3/envs/mpas-tools/lib -lnetcdff -lnetcdf
(mpas-tools) pedrosp@ppeixoto:/ppdata/ppdata/Work/Reps/MPAS-BR/grids/utilities/convert_nc_grid_to_xyz$ export LD_LIBRARY_PATH=`nc-config --libdir`:$LD_LIBRARY_PATH
(mpas-tools) pedrosp@ppeixoto:/ppdata/ppdata/Work/Reps/MPAS-BR/grids/utilities/convert_nc_grid_to_xyz$ ./convert_grid_to_xyz ../jigsaw/unif240km/unif240km_mpas.nc ../jigsaw/unif240km/unif240km_mpas.xyz
(mpas-tools) pedrosp@ppeixoto:/ppdata/ppdata/Work/Reps/MPAS-BR/grids/utilities/convert_nc_grid_to_xyz$
```

Copy file to imodel grids folder!

Configuration (par/)

mesh.par



Test case

```
!Possible test cases
!   case(1) !Test geodesic to regular grid conversion tool\\
!   case(2) !Test grid point search methods\\
!   case(3) !Mesh quality and distortion tests\\
!   case(4) !Divergence Tests\\
!   case(5) !Laplacian Tests\\
!   case(6) !Test scalar interpolations\\
!   case(7) !Test vector interpolation\\
!   case(8) !Test vector reconstruction\\
!   case(9) !Passive advection simulation\\
!   case(10)!Transport flow simulation\\
!   case(11)!Multigrid tests\\
!   case(12)!Tg reconstruction test\\
!   case(13)!Rotational discretization test\\
!   case(14)!Horizontal Discret Shallow water model diagnostics\\
!   case(15)!Shallow water model test cases\\
!   case(16)!Superconv tests\\
!   case(17)!Highorder advection tests\\
!   case(18)!Moist shallow water model test cases\\
```

Configuration (par/)

swm.par

iModel > par > swm.par

```
1  !Shallow water model parameters file - see below for details
2  !Test case / Test local truncation 0/1 (only possible in some test cases)
3  21 0
4  !Total period definition (ex:12days) / integrations stopping time (5 days)
5  8 8
6  !dt (in sec) / number of time steps (if dt=0) / adjust by grid level 5 (0 or 1)
7  60 0 0
8  !Scalar/vector location (HC or HTC)
9  HTC
10 !Wrapper (overrides next 4 options: none, trsk10, pxt16, gass18)
11 trsk10
12 !Cell vec reconstruction method (Kenergy) / Gassmann parameter (if
13 trsk 1.0
14 !Coriolis vec reconstruction method
15 trsk
16 !Scalar interpolations
17 trsk
18 !Gradient discrete method
19 trsk
20 !Area used for triangles and voronoi cells (geo=geodesic (default),
21 geo
22 !Number of plots on animation (ex: 10 plots) / print on screen step
23 8 60 0
24 !Diffusion coef - Leave zero for no diffusion / coeficient functior
25 0 const
26 !Hyperdiffusion maximum coef - Leave zero for no hyperdiffusion / c
27 0 const
28 !Hollingsworth coef - Only for test case 32, 33, 34
29 2.0
30 !Reference threshold for hybrid method (only if 'hyb' is set on cor
31 0.0
32 !Potential vorticity correction (none, apvm, clust) / CLUST paramet
33 none 0.5
34 !Level model vs layer model (Use PV = 0 , Do not use PV, use just v
```

```
!Test Cases implemented
1: Solid body rotation
2 : Steady State Will92
5 : Flow over mountain Will92
6 : Rossby-Haurwitz Will92
11 : Linearized equations - for mode calculation - fsphere
12 : Linearized equations - for mode calculation - f variable
21 : Galewski et al Barotropically unstable jet with perturbat
22 : Galewski et al Barotropically unstable jet without pertur
32 : Hollingsworth instability with tc2
33 : Hollingsworth instability with constant h and non rotatin
34 : Hollingsworth instability with tc2 suppressing nonlineari
35 : Hollingsworth instability with tc2 and non rotationg fram
42 : Rotated Steady state localised test on f-sphere
51 : Flow over mountain Will92 smooth mountain (Gaussian)
56 : Matsuno baroclinic wave - Eastward inertial gravity wave
57 : Matsuno baroclinic wave - Rossby wave (Paldor et al 2019)
```

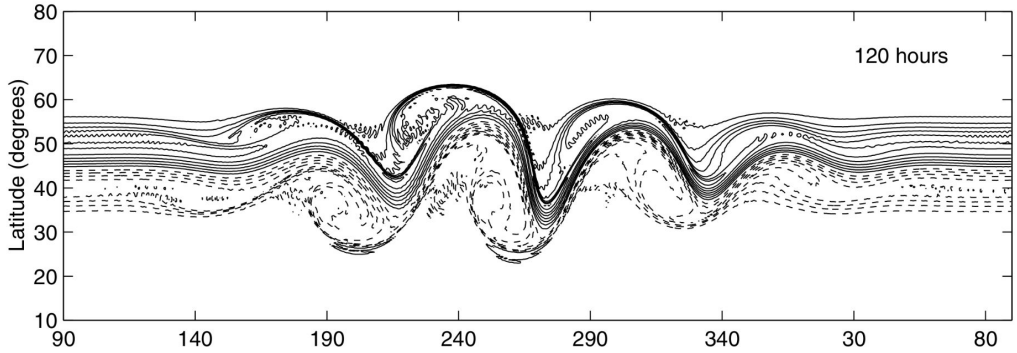
Test Case

Tellus (2004), 56A, 429–440

TELLUS

An initial-value problem for testing numerical models of the global shallow-water equations

By JOSEPH GALEWSKY*, RICHARD K. SCOTT and LORENZO M. POLVANI, *Department o*
Applied Physics and Applied Mathematics, Columbia University, New York, NY 10027, USA



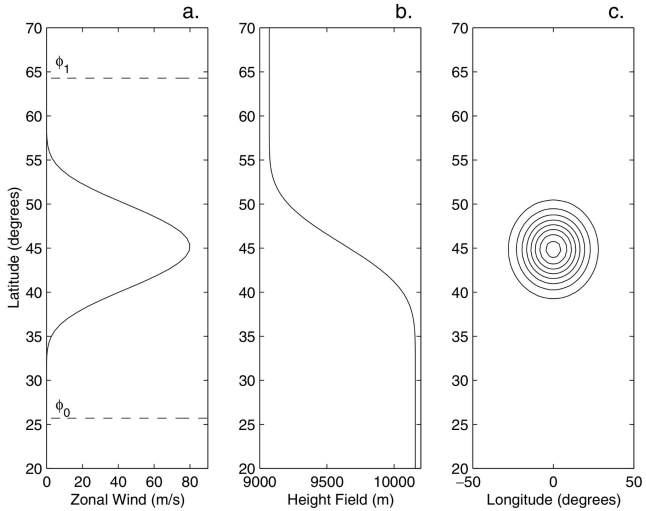


Fig 1. The initial conditions for the new test case. (a) The zonal wind, as defined in eq. (2); (b) the corresponding, balanced height field, calculated using eq. (3); (c) the height field perturbation, as defined in eq. (4), with a contour interval of 10 m; the outermost contour is at 10 m.

Configuration (par/)

swm.par

iModel > par > swm.par

```
1  !Shallow water model parameters file - see below for details
2  !Test case / Test local truncation 0/1 (only possible in some test cases)
3  21 0
4  !Total period definition (ex:12days) / integrations stopping time (5 days)
5  8 8
6  !dt (in sec) / number of time steps (if dt=0) / adjust by grid level 5 (0 or 1)
7  60 0 0
8  !Scalar/vector location (HC or HTC)
9  HTC
10 !Wrapper (overrides next 4 options: none, trsk10, pxt16, gass18)
11 trsk10
12 !Cell vec reconstruction method (Kenergy) / Gassmann parameter (if gass set - default
13 trsk 1.0
14 !Coriolis vec reconstruction method
15 trsk
16 !Scalar interpolations
17 trsk
18 !Gradient discrete method
19 trsk
20 !Area used for triangles and voronoi cells (geo=geodesic (default), tile= tiled (prod
21 geo
22 !Number of plots on animation (ex: 10 plots) / print on screen step (ex:every 100 time
23 8 60 0
24 !Diffusion coef - Leave zero for no diffusion / coeficient function (possible values:
25 0 const
26 !Hyperdiffusion maximum coef - Leave zero for no hyperdiffusion / coeficient function
27 0 const
28 !Hollingsworth coef - Only for test case 32, 33, 34
29 2.0
30 !Reference threshold for hybrid method (only if 'hyb' is set on coriolis method - zero
31 0.0
32 !Potential vorticity correction (none, apvm, clust) / CLUST parameter b or apvm parame
33 none 0.5
34 !Level model vs layer model (Use PV = 0 , Do not use PV, use just vorticity = 1)
```

Discretization options

HCT = MPAS grid staggering

TRSK = Ringler et 2010 paper (basis for MPAS)

Run

```
(imodel) pedrosp@ppeixoto:~/ppdata/Work/Reps/iModel$ ./imodel
 ------------------------------------------------------------
   Numerical Analysis on a Geodesical Icosahedral Sphere


   Pedro Peixoto and collaborators
     Oct  2018
 ------------------------------------------------------------

 Mesh parameters: par/mesh.par


 Loading mesh...


 Mesh        : Read from file icos_pol_scvt_h1_6.xyz
 nlat:   730
 dlat:  0.0043
 Max Quad Triangle intersection:      7
 Max Vertices Neighbours:             6
 Min Max Mean Vertice Distances (degrees):   0.8728  1.1435  1.0806

 Mesh loaded.

 Mesh created or loaded: icos_pol_scvt_h1_6
 ------------------------------------------------------
```

Outputs

```
Shallow Water Model parameters (file): par/swm.par

Test Case (Will1994)    :           21
Integration period (dys):    8.0000000000000000
Stopping time (dys)     :    8.0000000000000000
dt  (sec)               :    60.000000000000000
Number of timesteps     :        11520
Staggering              : HTC
Method wrapper          : trsk10
Scalar interpolation    : trsk
Vector recon method     : trsk
Coriolis recon method   : trsk
Gradient method         : trsk
Area method             : geo
Hollingsworth depth     :    2.0000000000000000
PV stabilization mtd    : none
Using layer model (with PV)

SWM Name for Plots: swm_tc21_dt60_HTC_trsk10_areageo

There is no reference file to read
CFL:   6.1821640109593814E-004
 Plotting scalarfield ( with neartrv ):
  data/swm_tc21_dt60_HTC_trsk10_areageo_h_t0_icos_pol_scvt_h1_6.dat
 Plotting scalarfield ( with neartrc ):
  data/swm_tc21_dt60_HTC_trsk10_areageo_eta_t0_icos_pol_scvt_h1_6.dat
 Plotting scalarfield ( with neartrv ):
  data/swm_tc21_dt60_HTC_trsk10_areageo_Kenergy_t0_icos_pol_scvt_h1_6.dat
 Plotting scalarfield ( with neartrc ):
  data/swm_tc21_dt60_HTC_trsk10_areageo_pv_t0_icos_pol_scvt_h1_6.dat
 Plotting scalarfield ( with neartred ):
  data/swm_tc21_dt60_HTC_trsk10_areageo_pv_ed_t0_icos_pol_scvt_h1_6.dat
 Plotting scalarfield ( with neartrv ):
  data/swm_tc21_dt60_HTC_trsk10_areageo_divuh_t0_icos_pol_scvt_h1_6.dat
```

Fields for plotting

Outputs

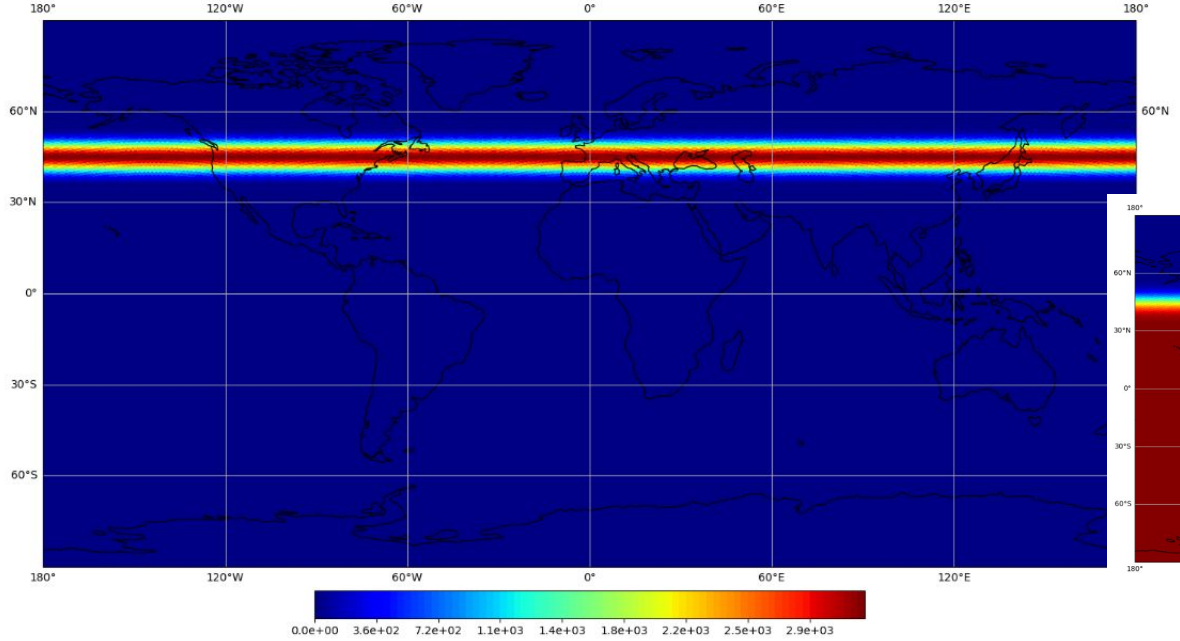
Time evolution

n	mvdist	tcase	k	nt	dt(s)	time(dys)	mass	Penergy	Kenergy	Tenergy	Availenergy	RMSdiv
40962	1.081	21	0	5760	120.0000	0.0000	0.100003E+05	0.616985E+10	0.187217E+08	0.618857E+10	0.267654E+08	0.000000E+00
40962	1.081	21	1	5760	120.0000	0.0014	-0.145514E-14	-0.255051E-08	0.837500E-06	-0.917727E-11	-0.212189E-08	0.436689E-07
40962	1.081	21	2	5760	120.0000	0.0028	-0.109136E-14	-0.100846E-07	0.332655E-05	0.937730E-11	0.216796E-08	0.856873E-07
40962	1.081	21	60	5760	120.0000	0.0833	-0.727572E-15	-0.205217E-06	0.676356E-04	0.159522E-10	0.368922E-08	0.210682E-06
40962	1.081	21	120	5760	120.0000	0.1667	-0.327407E-14	0.107492E-05	-0.354242E-03	0.143053E-10	0.330911E-08	0.210457E-06
40962	1.081	21	180	5760	120.0000	0.2500	0.618436E-14	0.202580E-05	-0.667609E-03	0.135653E-10	0.314161E-08	0.209209E-06
40962	1.081	21	240	5760	120.0000	0.3333	0.181893E-15	0.250622E-05	-0.825934E-03	0.165294E-10	0.382312E-08	0.207689E-06
40962	1.081	21	300	5760	120.0000	0.4167	-0.727572E-14	0.118571E-05	-0.390766E-03	-0.233149E-10	-0.539700E-08	0.197017E-06
40962	1.081	21	360	5760	120.0000	0.5000	-0.236461E-14	0.588133E-06	-0.193819E-03	0.106368E-10	0.245653E-08	0.195019E-06
40962	1.081	21	420	5760	120.0000	0.5833	0.100041E-13	0.128493E-05	-0.423453E-03	0.105657E-10	0.244221E-08	0.196028E-06
40962	1.081	21	480	5760	120.0000	0.6667	-0.272839E-14	0.103310E-05	-0.340461E-03	0.101030E-10	0.233553E-08	0.193976E-06
40962	1.081	21	540	5760	120.0000	0.7500	-0.218271E-14	0.331132E-06	-0.109123E-03	0.902687E-11	0.208661E-08	0.228686E-06
40962	1.081	21	600	5760	120.0000	0.8333	0.125506E-13	-0.116392E-06	0.383602E-04	0.743668E-11	0.171520E-08	0.194378E-06
40962	1.081	21	660	5760	120.0000	0.9167	0.327407E-14	-0.146626E-06	0.483237E-04	0.701475E-11	0.161828E-08	0.194849E-06
40962	1.081	21	720	5760	120.0000	1.0000	0.118230E-13	0.167361E-06	-0.551524E-04	0.718195E-11	0.166157E-08	0.195642E-06

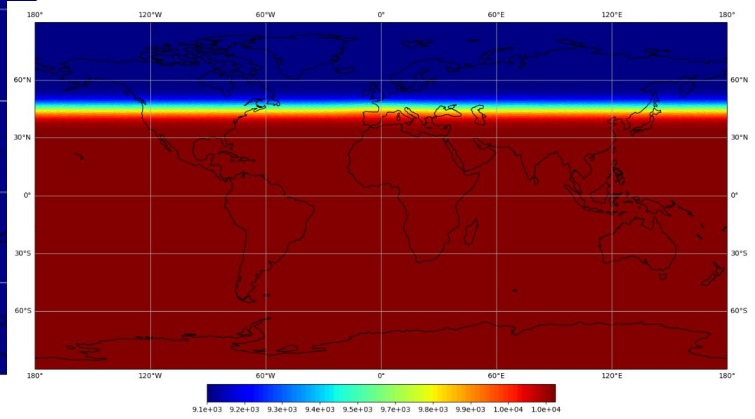
Plots

Initial conditions

Kinetic Energy



Depth

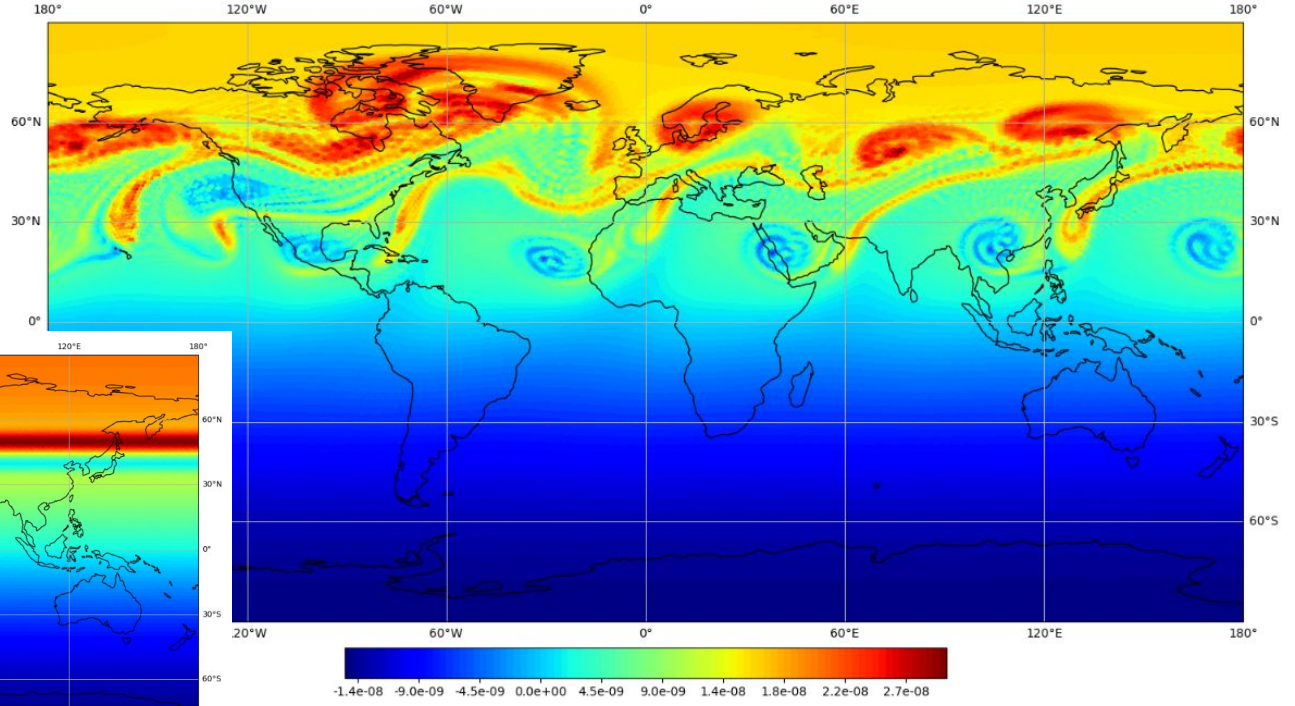



```
(imodel) pedrosp@ppeixoto:~/ppdata/Work/Reps/iModel/pyscripts$ python3 plot_scalar_field.py ../data/swm_tc21_dt120_HTC_trsk10_areageo_Kenergy_t0_icos_pol_scvt_h1_6.dat
Figure has been saved in ../data/../graphs/swm_tc21_dt120_HTC_trsk10_areageo_Kenergy_t0_icos_pol_scvt_h1_6.dat.png
```

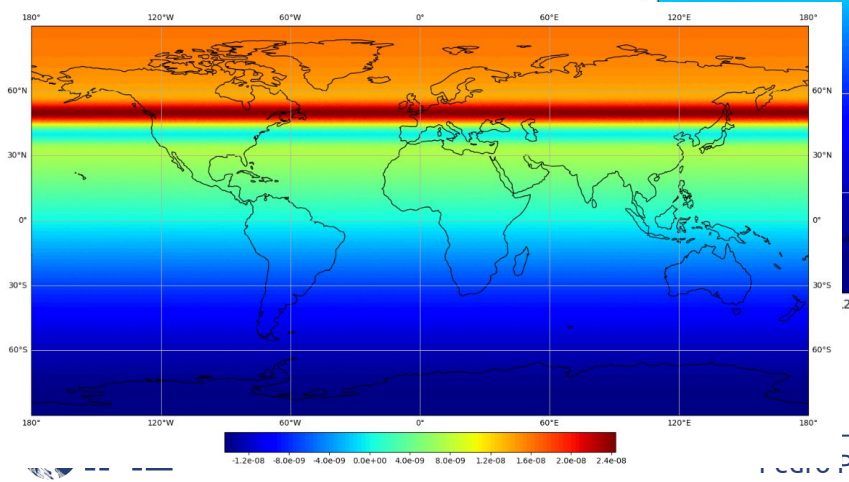
Plots

```
(imodel) pedrosp@ppeixoto:~/ppdata/Work/Reps/iModel/pyscripts$ python plot_scalar_field.py ../data/swm_tc21_dt120_HTC_trsk10_areageo_pv_ed_t691200_icos_pol_scvt_h1_6.dat
Figure has been saved in ../data/../graphs/swm_tc21_dt120_HTC_trsk10_areageo_pv_ed_t691200_icos_pol_scvt_h1_6.dat.png
(imodel) pedrosp@ppeixoto:~/ppdata/Work/Reps/iModel/pyscripts$
```

PV 8 days



PV Init



Code

Key features :
(simpler to understand iModel than MPAS)

- Grid data structure:
  https://github.com/pedrospeixoto/iModel/blob/master/src/datastruct.f90
- Time integration:
  https://github.com/pedrospeixoto/iModel/blob/master/src/swm.f90
- FV operators:
  https://github.com/pedrospeixoto/iModel/blob/master/src/swm_operators.f90

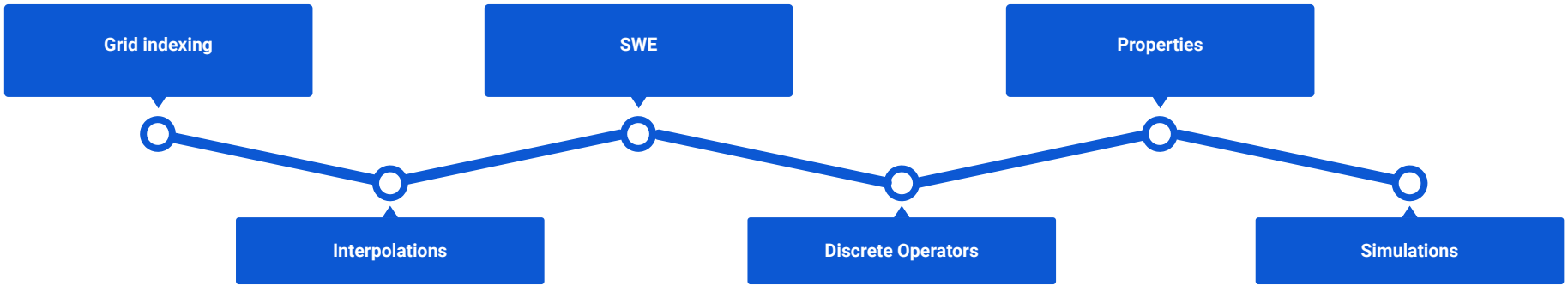
```
!$omp parallel do &
!$omp default(none) &
!$omp shared(mesh, u, div) &
!$omp shared(useStagHTC, useStagHC, useTiledAreas ) &
!$omp private(j, l, signcor) &
!$omp schedule(static)
do i=1,mesh%nv
  !Divergence of uh on unit sphere using div_cell_Cgrig
  !For all edges forming the hexagon
  div%f(i)=0._r8
  do j=1, mesh%v(i)%nnb
    !Get edge index
    l=mesh%v(i)%ed(j)
    !Get edge outer normal related to the hexagon
    if(useStagHTC)then
      signcor=real(mesh%hx(i)%ttgout(j), r8)
    elseif(useStagHC)then
      signcor=real(mesh%hx(i)%nr(j), r8)
    end if
    !Calculate numerical integration
    !lengths are multiplied by erad, and area by erad**2, therefore /erad
    div%f(i)=div%f(i)+signcor*u%f(l)*mesh%edhx(l)%leng
     !divu%f(i)=divu%f(i)+signcor*u%f(l)*mesh%edhx(l)%leng
  end do
  if(useTiledAreas)then
    div%f(i)=div%f(i)/mesh%hx(i)%areat/erad
  else
    div%f(i)=div%f(i)/mesh%hx(i)%areag/erad
  end if
  !divu%f(i)=divu%f(i)/mesh%hx(i)%areag/erad
  !divuh%f(i)=div_cell_Cgrid(i,uh,mesh)/erad
end do
!$omp end parallel do
```

Discrete Divergence (with OpenMP Parallelism)

$$D_i = \frac{1}{A_i} \sum_e h_e u_e n_{ei} l_e$$

Overview

Today's class:



Today's activity

iModel Simulation

Main goal:

Learn the processing steps in a simpler environment

Main steps:

- Install/Make
- Set parameter files
- Run
- Post-process

Shallow Water Equation Simulation!

Remark: MPAS has a Shallow Water Model in the repository. I have never tested it and prefer to use imodel. You could give the SW-MPAS code a try if you wish!

Today's activity

iModel Simulation

What to do?

1. Convert a grid generated with Jigsaw to xyz grid using the utility:
   MPAS-BR/grids/utilities/convert_nc_grid_to_xyz
   (if you have issues, download a grid from here: https://pedrosp.ime.usp.br/grids/swm/)
2. Configure iModel to read this grid and run the shallow water test case 21 - unstable jet test case
3. Plot the potential vorticity at day 7

Hand in:

1. An image of the grid used (a grid representation or resolution plot)
2. An image of the PV at day 7 from your simulation

Caution!

1. The test case is dynamically unstable and requires a relatively high global resolution (~30km or less)

That is all for today…

“All models are wrong, but some are useful”

— George Box

More at: www.ime.usp.br/~pedrosp

Thanks!