GungHo: Designing a next generation dynamical core for weather & climate prediction

Thomas Melvin

Met Office

Overview

Part I

- Gungho Dynamical Core

Part II

- LFRic Atmosphere model
- Code generation
- Next generation modelling system

Acknowledgments

Nerc: Gungho partner Universities

STFC: PSyClone development

Met Office: LFRic development team

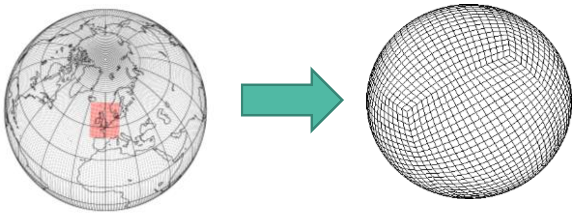
Dynamics Research

UM Physics developers

NGMS programme

What we are trying to do

- Move to quasi-uniform mesh to remove polar singularity
- Maintain ‘good’ aspects of current model
  - No computational modes
  - Accurate wave dispersion
  - Semi-Implicit timestepping
  - Reuse subgrid parametrizations
- Improve inherent conservation
- Improve scalability



Model Components

工合 Gungho: Mixed finite element dynamical core

LFRic: Model infrastructure for next generation modelling

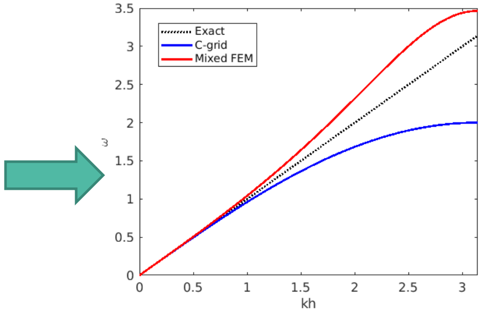
PSyclone: Parallel Systems code generation used in LFRic and Gungho

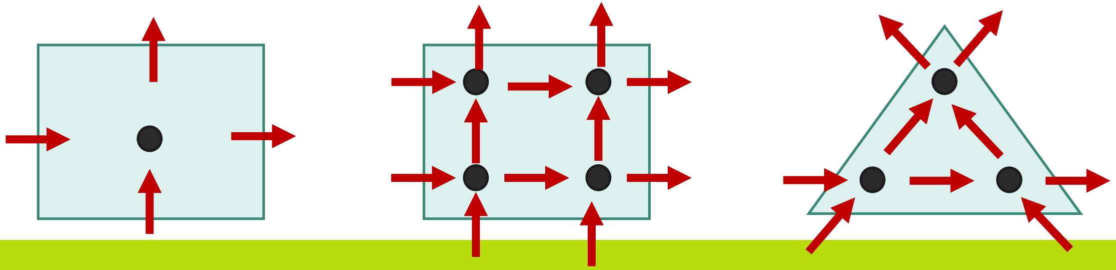
UM | Unified Model — UM: Current modelling environment (UM parametrisations are being reused in LFRic)

Mixed Finite Elements

Mixed Finite Element method gives

- Compatibility: $\nabla \times \nabla\varphi = 0, \nabla \cdot \nabla \times \boldsymbol{v} = 0$
- Accurate balance and adjustment properties
- No orthogonality constraints on the mesh
- Flexibility of choice mesh (quads,triangles) and accuracy (polynomial order)



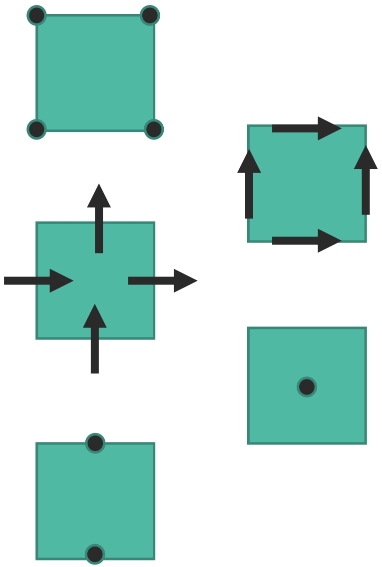


Met Office

Mixed Finite Element Method

$$\mathbb{W}_0 \xrightarrow{\nabla} \mathbb{W}_1 \xrightarrow{\nabla\times} \mathbb{W}_2 \xrightarrow{\nabla\cdot} \mathbb{W}_3. \qquad \mathbb{W}_\theta = \mathbf{k}\cdot\mathbb{W}_2$$

$\mathbb{W}_0$	Pointwise scalars	
$\mathbb{W}_1$	Circulation Vectors	
$\mathbb{W}_2$	Flux Vectors	Velocity
$\mathbb{W}_3$	Volume integrated Scalars	Pressure, Density
$\mathbb{W}_\theta$	Pointwise scalars	Potential Temperature, moisture



New developments

- Switch from vector invariant form to advective form momentum equation
  - Allows use of same explicit upwind advection scheme as for scalars: improves stability & accuracy
- Switch from Krylov (BiCGStab) to Multigrid preconditioner for the pressure solver (improves scalability)
- Move to 2x2 (advection, nonlinear) iterations in the timestep from 4x1 (reduces cost)
- Over relaxation ($\tau_\rho = 1$) in continuity equation (improves stability)

$$\rho' + \tau_\rho \Delta t \nabla \cdot (\rho^* \mathbf{u}') = -\left(\rho^{n+1} - \rho^n + \Delta t \nabla \cdot \boldsymbol{\mathcal{F}}\right)$$

Gungho Discretisation

- Inspired by iterative-semi-implicit semi-Lagrangian scheme used in UM
- Transport uses high-order, upwind, explicit Eulerain FV scheme
- Wave dynamics use iterative-semi-implicit, lowest order mixed finite element method (equivalent to C-grid/Charney-Phillips staggering)

$$\begin{aligned}
\int_{\mathrm{D}} \mathbf{v} \cdot \delta_t \mathbf{u}\, dV &= -\int_{\mathrm{D}} \mathbf{v} \cdot \left[\boldsymbol{\mathcal{A}}\left(\mathbf{u}^p, \overline{\mathbf{u}}^{1/2}\right) + \overline{2\boldsymbol{\Omega} \times \mathbf{u} + \nabla\Phi + c_p \theta \nabla \Pi}^{\alpha} \right] dV, \\
\int_{\mathrm{D}} \sigma \delta_t \rho\, dV &= -\int_{\mathrm{D}} \sigma \nabla \cdot \boldsymbol{\mathcal{F}}\left(\rho^p, \overline{\mathbf{u}}^{1/2}\right) dV, \\
\int_{\mathrm{D}} w \delta_t \theta\, dV &= -\int_{\mathrm{D}} w \mathcal{A}\left(\theta^p, \overline{\mathbf{u}}^{1/2}\right) dV, \\
\int_{\mathrm{D}} \sigma \left(\Pi^{n+1}\right)^{\frac{1-\kappa}{\kappa}} dV &= \int_{\mathrm{D}} \sigma \frac{R}{p_0} \rho^{n+1} \theta^{n+1}\, dV,
\end{aligned}$$

$$\overline{F}^{\alpha} \equiv \alpha F^{n+1} + (1-\alpha) F^n$$

$$\begin{aligned}
\mathcal{A}(q, \mathbf{u}) &\equiv \int_t^{t+\Delta t} \mathbf{u} \cdot \nabla q\, dt, \\
\mathcal{F}(q, \mathbf{u}) &\equiv \int_t^{t+\Delta t} \mathbf{u} q\, dt,
\end{aligned}$$

[1] *A mixed finite-element, finite-volume, semi-implicit discretization for atmospheric dynamics: Cartesian geometry*, Melvin et al QJRMS19

Met Office

Timestep

Dynamics forcing (pgf, Coriolis)

Transport

$$Lq' = R = F^{n+1} + F^{n} + S^{n+1} + S^{n} + A(q^{p}, u)$$

Linear operator

Fast (convection, bl) and Slow (radiation, gwd) subgrid terms

Compute Slow Physics: $S^n(q^n)$

Compute dynamics forcing: $F^n(q^n)$

Advect fields: $A(q^p, u) = A(q^n + S^n + F^n, u)$

Compute Fast Physics: $S^{n+1}(q^{n+1})$

Compute dynamics forcings: $F^{n+1}(q^{n+1})$

Solve linear system: $Lq' = R$

2x outer loop

2x inner loop

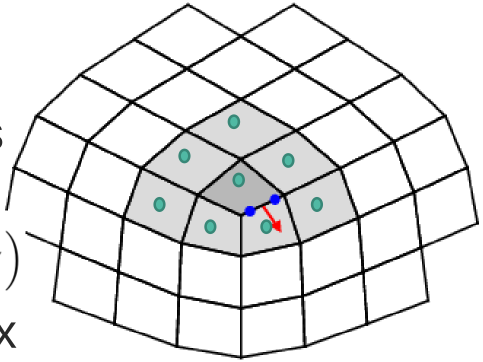
Met Office

Transport

$$\delta_t q + \sum_{s=1}^{n} a_s \overline{\mathbf{u}}^{1/2} \cdot \nabla \widetilde{q}^s = 0,$$

$$\delta_t q + \sum_{s=1}^{n} a_s \nabla \cdot \left(\overline{\mathbf{u}}^{1/2} \widetilde{q}^s \right) = 0,$$

- Advection equation solved using a multi-stage scheme
- Field $(\tilde{q})$ is reconstructed at each stage using a high order upwind polynomial
- Flux $(\mathcal{F})$ or advective $(\mathcal{A})$ increment used to advect field is diagnosed and used in iterative-implicit scheme
- Wind field is fixed during the advection step $\overline{\mathbf{u}}^{1/2} \equiv \frac{1}{2}\left(\mathbf{u}^{(k)} + \mathbf{u}^n\right)$
- Moisture is transported on a shifted vertical mesh using flux form scheme → moisture mass is conserved



Linear Solver

$\approx 10\times$

Iteratively (GCR) solve mixed system

$$\mathcal{L}\left(\mathbf{x}^*\right)\left[\mathbf{u}',\theta',\rho',\Pi'\right]^T = -\mathcal{R}^{(k)},$$

Form approximate Schur complement

$$\mathcal{L} \longrightarrow H, \quad \mathcal{R} \longrightarrow R_H,$$

Solve (Multigrid) Helmholtz system

$$H\Pi' = R_H,$$

$\approx 2\times$

Apply smoother on each level

$$\widetilde{\Pi}' \leftarrow \widetilde{\Pi}' + \omega \widehat{H}_z^{-1}\left(\mathcal{B} - H\widetilde{\Pi}'\right)$$

Error is determined by convergence of mixed system. Multigrid v-cycle used to speed up convergence

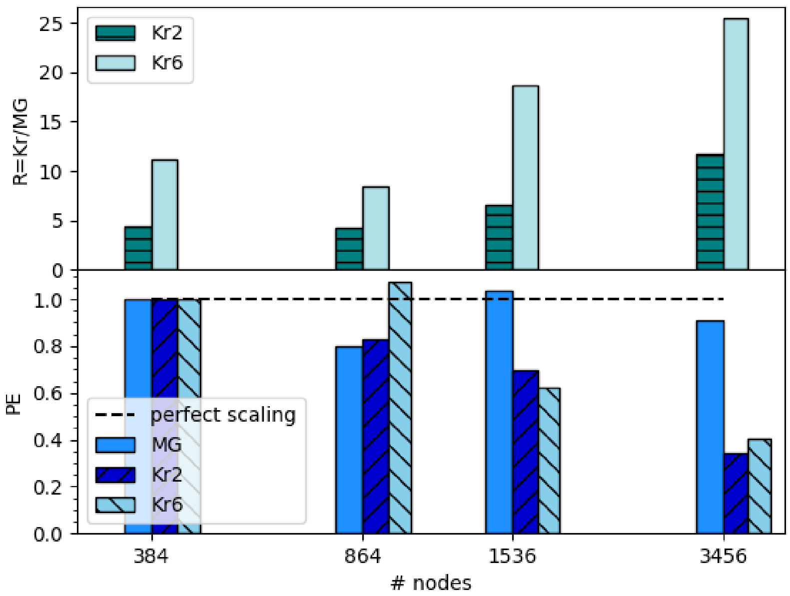
Multigrid Preconditioner

- Helmholtz system $H\Pi' = R$ solved using a single Geometric-Multi-Grid V-cycle with block-Jacobi smoother

$$H = M_3^{\Pi^*} + \left(P_{3\theta}^* \mathring{M}_\theta^{-1} P_{\theta 2}^{\theta^*,z} + M_3^{\rho^*} M_3^{-1} D^{\rho^*}\right)\left(\mathring{M}_2^{\mu,C}\right)^{-1} G$$

- Block-Jacobi smoother with small number (2) of iterations on each level
- Smoother: Exact vertical solve: $\hat{H}_z^{-1}$

$$\widetilde{\Pi}' \leftarrow \widetilde{\Pi}' + \omega \widehat{H}_z^{-1}\left(\mathcal{B} - H\widetilde{\Pi}'\right)$$

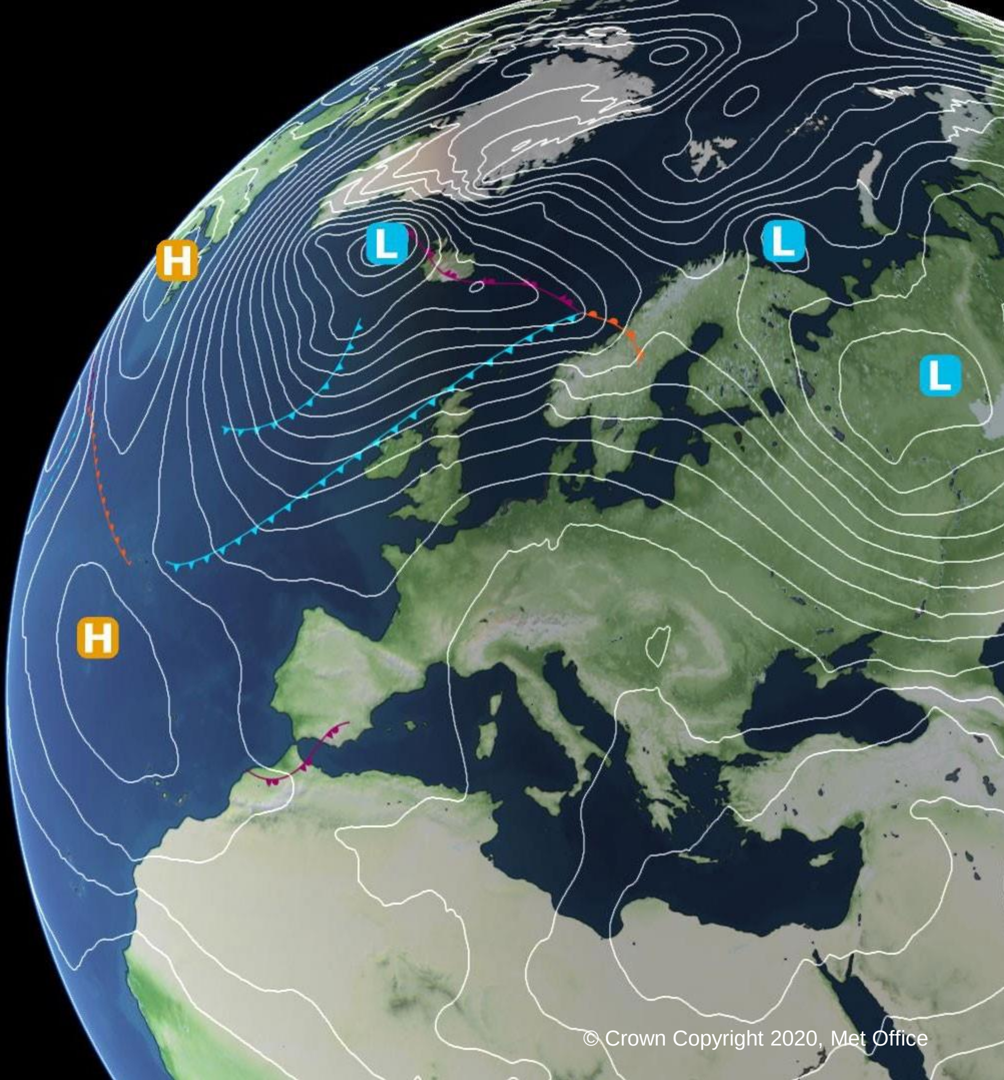


C1152 (~9km)

- No global sums
- Reduced number of operator (H) applications

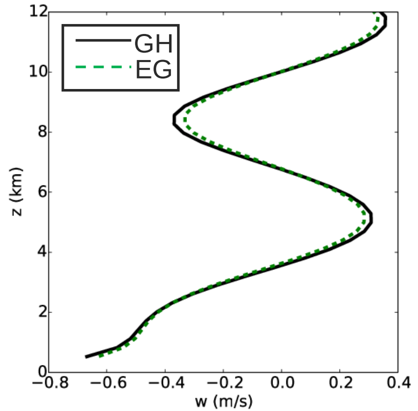
Met Office

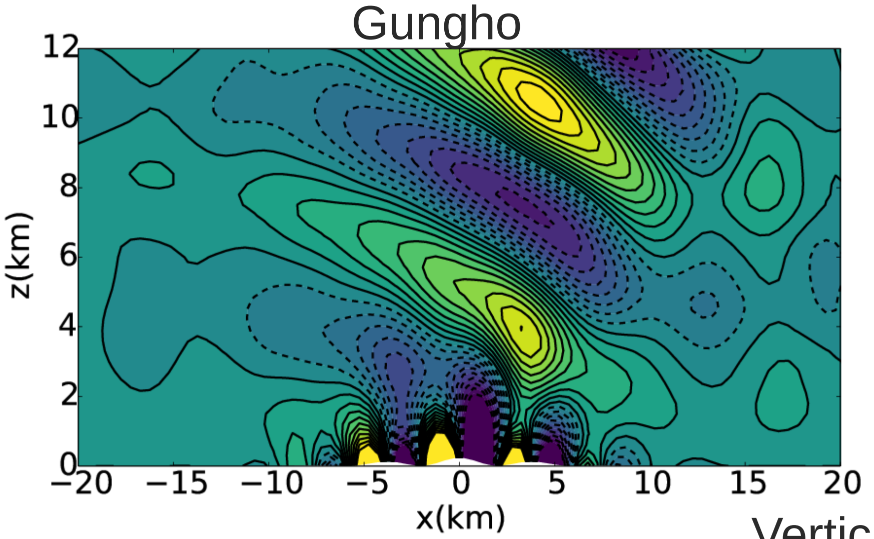
Questions?

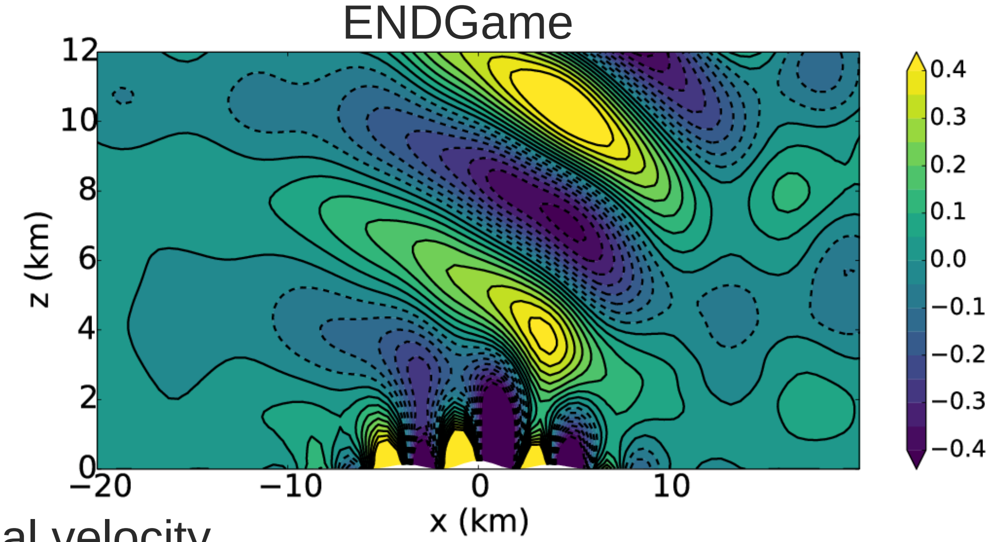


Results

2D Schar hill





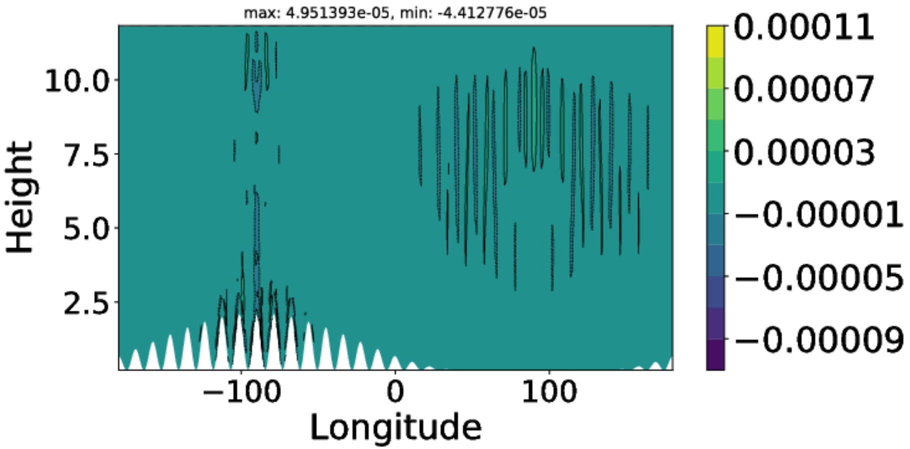


Vertical velocity

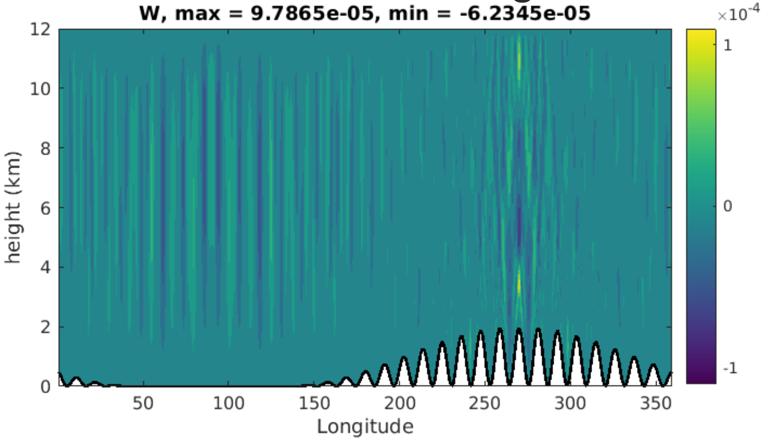
Balance

DCMIP2012 Test 200: Balanced atmosphere over a large mountain

Gungho C96



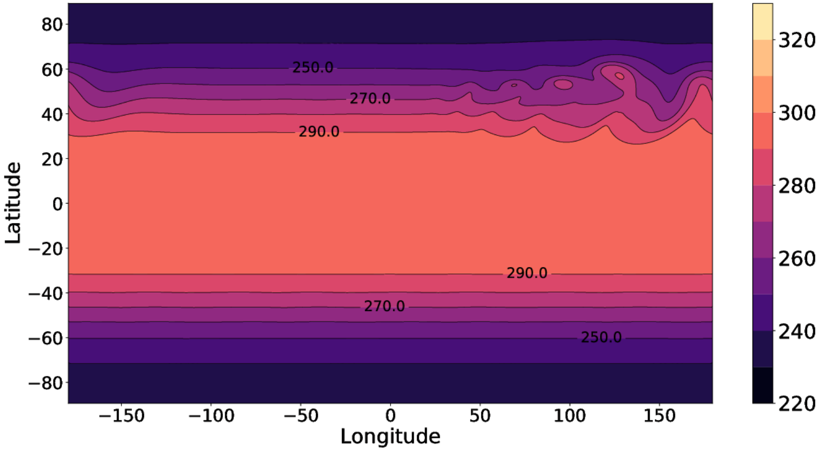
ENDGame 1degree



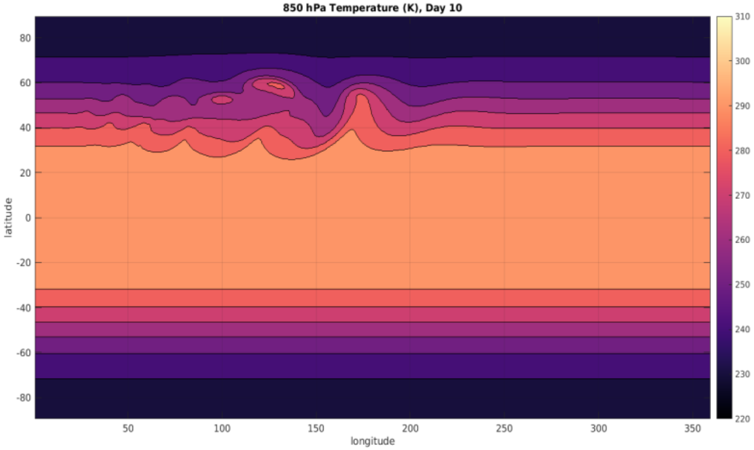
Vertical velocity

Baroclinic Wave: Day 10

Gungho C96



ENDGame (1deg)

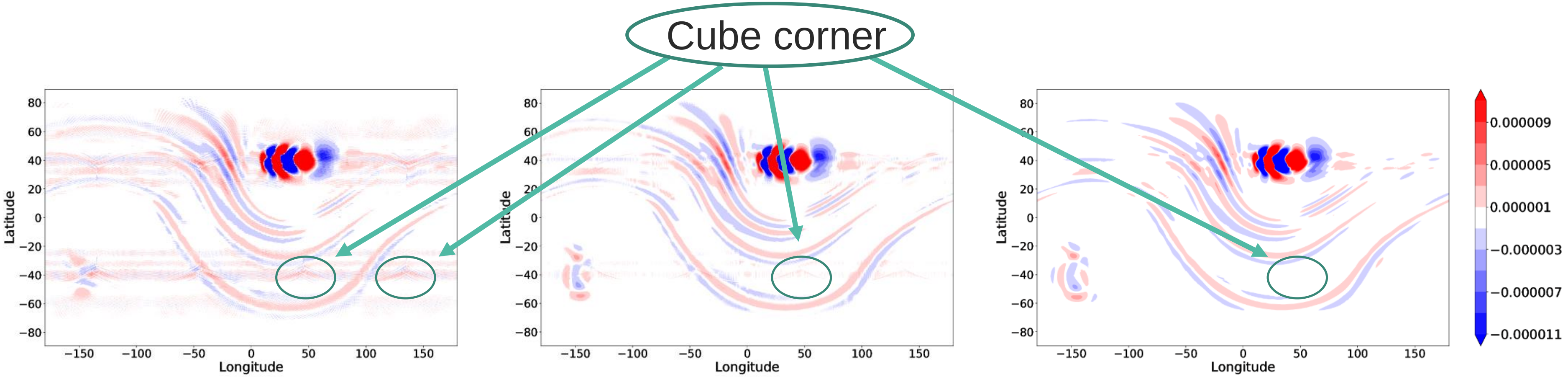


850hPa Temperature
Surface Pressure

Met Office

Grid Imprinting

Vertical velocity (with zonal mean removed) after 1 day of baroclinic wave simulation



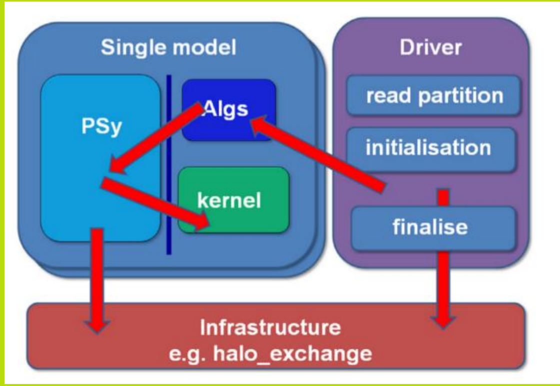
C96, Dt=900s

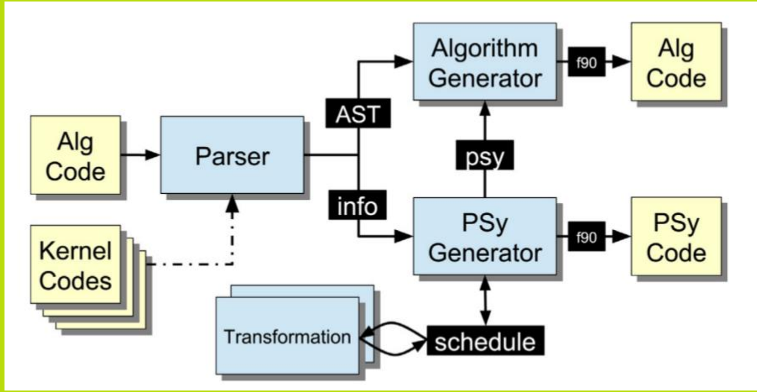
C192, Dt=450s

C384, Dt=225s

Code Generation: PSyclone

- LFRic [3] uses the PSyclone [4] code generation tool to create the parallel layer
- PSyclone reads the code and generates interfaces based upon defined rules
- Parallelisation & optimisation strategies are defined via a simple script





[3] *LFRic: Meeting the challenges of scalability and performance portability in Weather and Climate models*, Adams et al J. Par. Dist. Comp. 19

[4] github.com/stfc/PSyclone

Met Office

Code generation: User code

Compute: $u + \Delta t \nabla p$

```
! Compute rhs = u + dt*grad(p)
call rhs%initialise( u%get_function_space() )
call grad%initialise( u%get_function_space() )
call invoke( setval_c(grad, 0.0_r_def),                        &
             gradient_kernel_type(grad, pressure, qr),         &
             aX_plus_Y(rhs, dt, grad, u) )
```

Algorithm

Kernel

```
subroutine gradient_code(nlayers,                                     &
                         gradient, pressure,                          &
                         ndf_w2, undf_w2, map_w2, w2_diff_basis,      &
                         ndf_w3, undf_w3, map_w3, w3_basis,           &
                         nqp_h, nqp_v, w_h, w_v )

  implicit none

  ...

  ! Loop over layers in the cell
  do k = 0, nlayers-1

    ! Loop over quadrature points
    do qp2 = 1, nqp_v
      do qp1 = 1, nqp_h

        ! Compute the gradient
        grad_p = 0.0_r_def
        do df = 1, ndf_w3
          grad_p = grad_p + pressure(map_w2(df)+k)*w3_basis(1,df,qp1,qp2)
        end do

        ! Multiply by test function and integrate
        do df = 1, ndf_w2
          gradient(map_w2(df)+k) = gradient(map_w2(df)+k) &
                       + w_h(qp1)*w_v(qp2)*w2_diff_basis(1,df,qp1,qp2)*grad_p
        end do

      end do
    end do

  end do

end subroutine gradient_code
```

Met Office

Code generation: Auto generation

```
! Compute rhs = u + dt*grad(p)
call rhs%initialise( u%get_function_space() )
call grad%initialise( u%get_function_space() )
call invoke( setval_c(grad, 0.0_r_def),                      &
             gradient_kernel_type(grad, pressure, qr),       &
             aX_plus_Y(rhs, dt, grad, u) )
```

```
!-------------------------------------------------------------
!> The type declaration for the kernel.
!> Contains the metadata needed by the Psy layer.
!-------------------------------------------------------------
type, public, extends(kernel_type) :: gradient_kernel_type
  private
  type(arg_type) :: meta_args(2) = (/    &
      arg_type(GH_FIELD,   GH_INC,  W2), &
      arg_type(GH_FIELD,   GH_READ, W3)  &
      /)
  type(func_type) :: meta_funcs(2) = (/ &
      func_type(W2, GH_DIFF_BASIS),     &
      func_type(W3, GH_BASIS)           &
      /)
  integer :: iterates_over = CELLS
  integer :: gh_shape = GH_QUADRATURE_XYoZ
contains
  procedure, nopass :: gradient_code
end type
```

Psyclone parses:

Algorithm

Kernel metadata

and produces:

New algorithm

& Psy Layer

```
CALL rhs % initialise(u % get_function_space())
CALL grad % initialise(u % get_function_space())
CALL invoke_0(grad, pressure, rhs, dt, u, qr)
```

```
SUBROUTINE invoke_0(grad, pressure, rhs, dt, u, qr)

....

!$omp end parallel
IF (pressure_proxy%is_dirty(depth=1)) THEN
  CALL pressure_proxy%halo_exchange(depth=1)
END IF
!
DO colour=1,ncolour
  !$omp parallel default(shared), private(cell)
  !$omp do schedule(static)
  DO cell=1,mesh%get_last_halo_cell_per_colour(colour,1)
    !
    CALL gradient_code(nlayers,                                      &
                       grad_proxy%data, pressure_proxy%data,         &
                       ndf_w2, undf_w2, map_w2(:,cmap(colour, cell)), &
                       diff_basis_w2,                                &
                       ndf_w3, undf_w3, map_w3(:,cmap(colour, cell)), &
                       basis_w3, np_xy, np_z, weights_xy, weights_z)
  END DO
  !$omp end do
  !$omp end parallel
END DO

! Set halos dirty/clean for fields modified in the above loop
CALL grad_proxy%set_dirty()

!$omp parallel default(shared), private(df)
!$omp do schedule(static)
DO df=1,rhs_proxy%vspace%get_last_dof_annexed()
  rhs_proxy%data(df) = dt*grad_proxy%data(df) + u_proxy%data(df)
END DO
!$omp end do

! Set halos dirty/clean for fields modified in the above loop
!$omp master
CALL rhs_proxy%set_dirty()
!$omp end master
!
!$omp end parallel
END SUBROUTINE invoke_0
```

Met Office

Code generation: Auto generation

```
! Compute rhs = u + dt*grad(p)
call rhs%initialise( u%get_function_space() )
call grad%initialise( u%get_function_space() )
call invoke( setval_c(grad, 0.0_r_def),                &
             gradient_kernel_type(grad, pressure, qr), &
             aX_plus_Y(rhs, dt, grad, u) )
```

All parallel directives appear in the Psylayer

```
SUBROUTINE invoke_0(grad, pressure, rhs, dt, u, qr)

....

!$omp end parallel
```

MPI

```
IF (pressure_proxy%is_dirty(depth=1)) THEN
  CALL pressure_proxy%halo_exchange(depth=1)
END IF
!
```

Horizontal Looping

```
DO colour=1,ncolour
  !$omp parallel default(shared), private(cell)
  !$omp do schedule(static)
  DO cell=1,mesh%get_last_halo_cell_per_colour(colour,1)
    !
```

Kernel Call

```
    CALL gradient_code(nlayers,                                       &
                       grad_proxy%data, pressure_proxy%data,          &
                       ndf_w2, undf_w2, map_w2(:,cmap(colour, cell)), &
                       diff_basis_w2,                                 &
                       ndf_w3, undf_w3, map_w3(:,cmap(colour, cell)), &
                       basis_w3, np_xy, np_z, weights_xy, weights_z)
  END DO
  !$omp end do
  !$omp end parallel
END DO

! Set halos dirty/clean for fields modified in the above loop
CALL grad_proxy%set_dirty()
```

OpenMP

```
!$omp parallel default(shared), private(df)
!$omp do schedule(static)
```

Built-in

```
DO df=1,rhs_proxy%vspace%get_last_dof_annexed()
  rhs_proxy%data(df) = dt*grad_proxy%data(df) + u_proxy%data(df)
END DO
!$omp end do

! Set halos dirty/clean for fields modified in the above loop
```

MPI

```
!$omp master
CALL rhs_proxy%set_dirty()
!$omp end master
!
!$omp end parallel
END SUBROUTINE invoke_0
```

Met Office

LFRic Atmosphere

工合
+
Unified Model

- Gungho dynamical core
- Unified model subgrid physics
- Within the LFRic infrastructure

UM Physics

- LFRic uses existing UM Physics (+ code repository)
- Can run in single column model (SCM) mode to compare output
- LFRic uses k-first indexing while UM uses i-first
- i-first allows physics to work on a large segment of data giving a performance boost. LFRic has to use single columns as a segment
- Investigating transposing data in LFRic

```
do ij = 1, ncells
  do k = 0, nlayers - 1
    q(map(1,ij) + k) = rhs
  end do
end do
```

LFRic

```
do k = 1, nlayers
  do j = 1, nrows
    do i = 1, ncols
      q(i,j,k) = rhs
    end do
  end do
end do
```

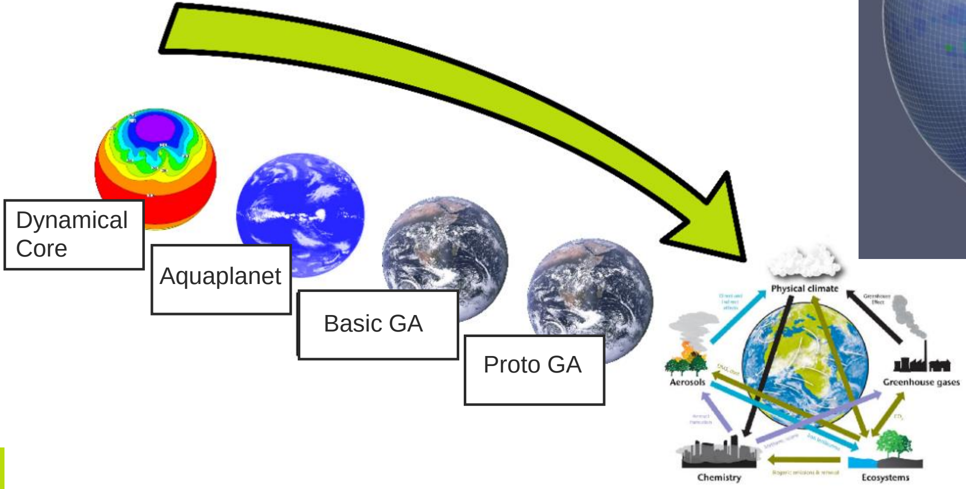
UM

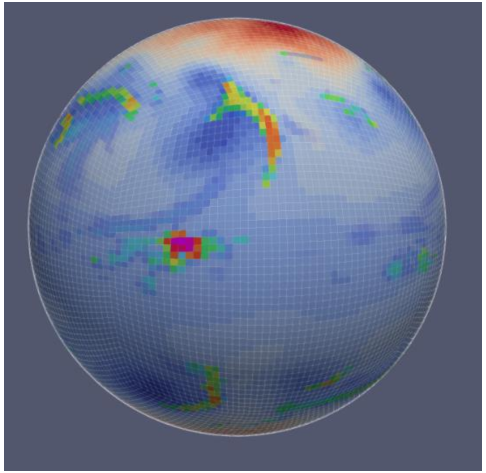
Met Office

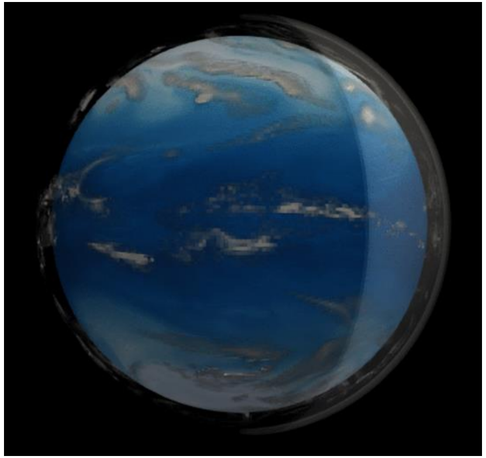
Aquaplanet & beyond

- Couple UM physics to Gungho dynamical core
- Read in reconfigured UM start dump
- I/O using XIOS
- Parallel code layer automatically generated

Dynamical core + subgrid physics
No land surface or orography
Proscribed surface temperature

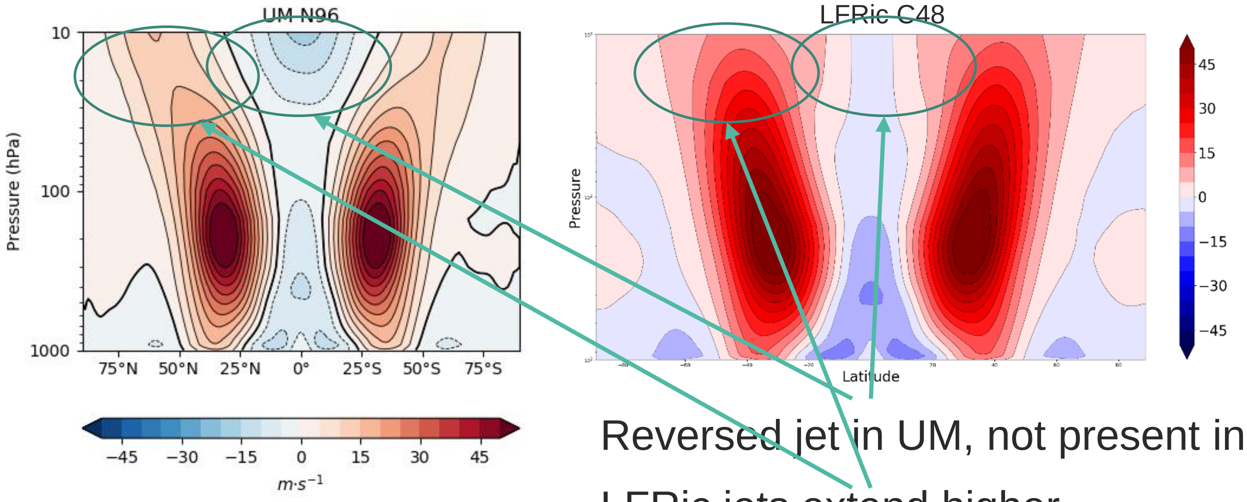






Aquaplanet

- Comparison to current ENDGame (SISL) dynamical core over 2yr simulation
- Same subgrid physics, vertical mesh & timestep

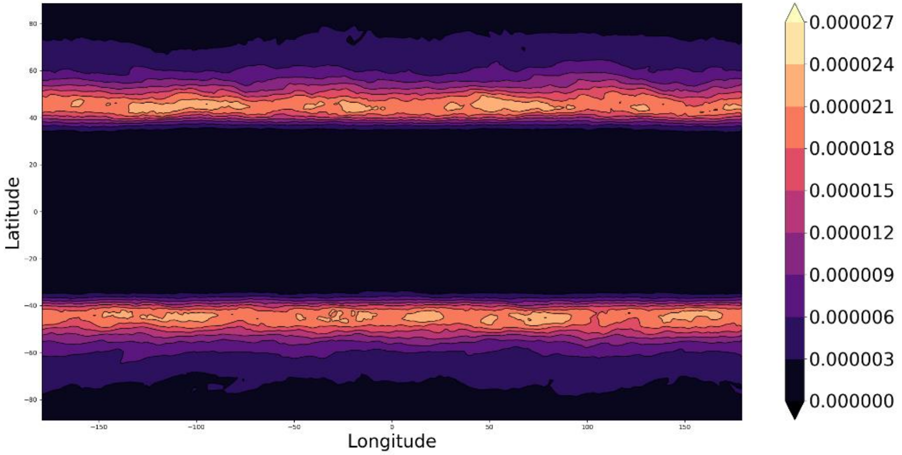


Reversed jet in UM, not present in LFRic

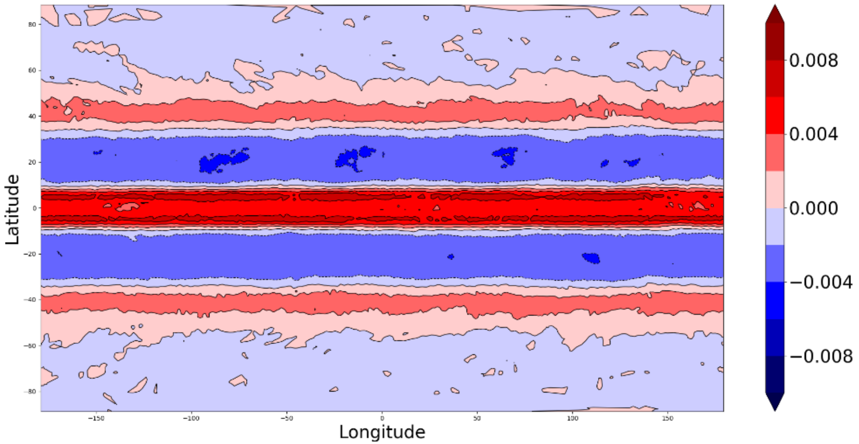
LFRic jets extend higher

Aquaplanet

850hPa Cloud Ice



850hPa Vertical velocity



Time averages over 800 days of simulation

Next Generation Modelling System (NGMS)

"*Reformulating and redesigning our complete weather and climate research and operational/production systems, including oceans and the environment, to allow us to fully exploit future generations of supercomputer for the benefits of society*"

Met Office

NGMS Projects

Gungho Atmospheric Science Project
- Develop atmospheric science aspects & deliver model scientifically as good as UM

LFRic Infrastructure Development
- Deliver infrastructure to replace the UM scalable for future platforms

LFRic Inputs
- Tools to ingest fixed & time-varying fields.
- Include initial conditions, ancillary fields and LBCs

LFRic Diagnostic Infrastructure
- Development of research diagnostics and research workflow capabilities

NG-Marine Systems
- Deliver scalable marine systems including ocean, sea-ice & wave models

NG-Coupling
- OASIS3-MCT coupled components

NG-DA
- NGMS-ready coupled atmos/ocean DA
- JEDI as a DA framework

NG-OPS
- Processing of NWP observational data for NG-DA

NG-VAT (Visualisation Analysis Tools)
- Support for visualisation and evaluation tools used by scientists

FAB Build System
- Development of new build systems for NGMS components

NG-R2O
- Support transition of NGMS capability from research to NWP operations

NG-Composition
- Coordination of aerosol & chemistry development within NGMS

NG-R2C
- Support transition of capability from research to climate production

NG-ADAQ
- Development of dispersions models (e.g. NAME) for next generation computing

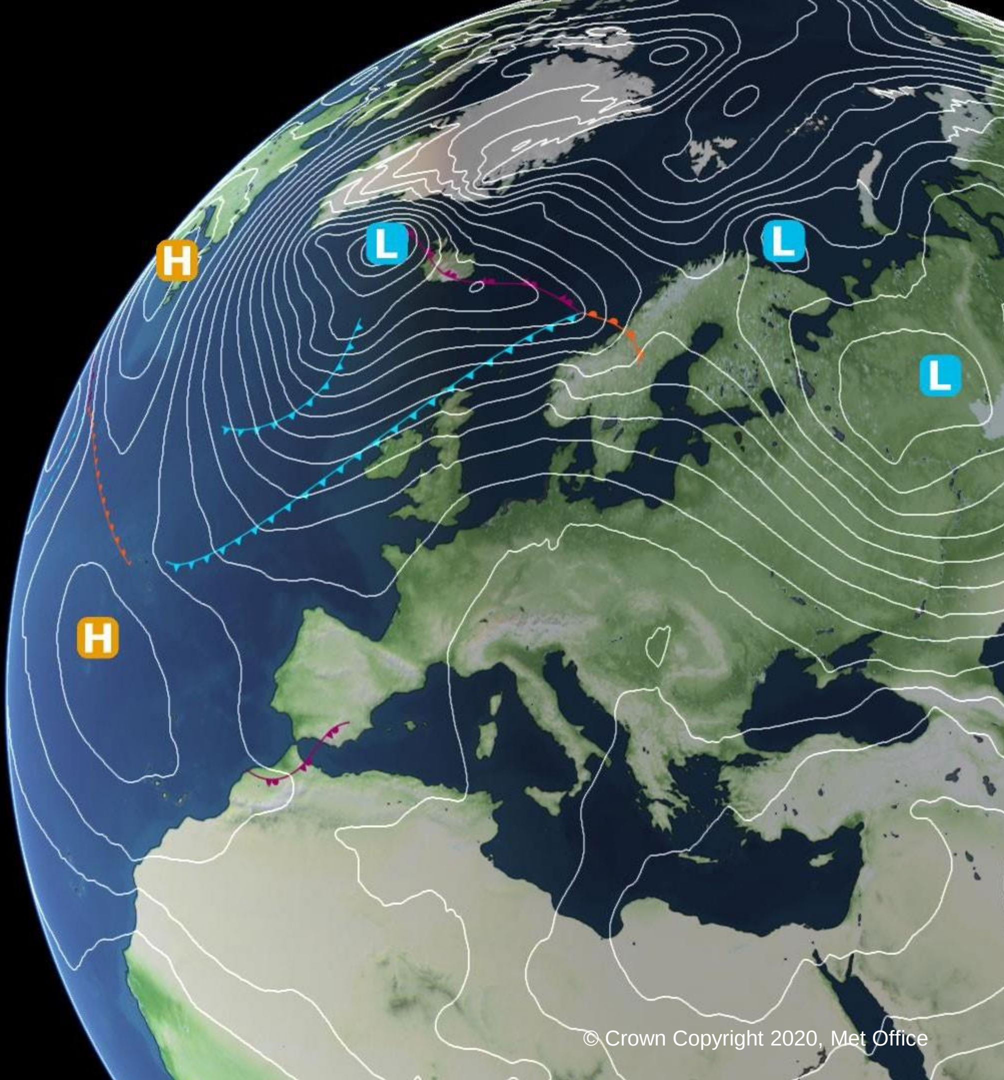
NG-VER
- Development of NWP verification capability for NGMS

Summary

- Gungho uses a mixed finite element + finite volume transport on a cubed sphere
- Shows similar accuracy to ENDGame
- Has been coupled to UM subgrid physics
  - Run in aquaplanet mode
  - Functionality to run in global atmosphere more exists but needs to be tested
- Only one part of NGMS

Met Office

Questions?



Linear Solver

- Quasi-Newton Method: $\mathcal{L}\left(\mathbf{x}^{*}\right)\mathbf{x}' = -\mathcal{R}\left(\mathbf{x}^{(k)}\right).$
- Linearized around reference state (previous timestep state) $\boldsymbol{x}^{*} \equiv \boldsymbol{x}^{n}$
- Solve for increments on latest state: $\boldsymbol{x}' \equiv \boldsymbol{x}^{(\boldsymbol{k}+\mathbf{1})} - \boldsymbol{x}^{(\boldsymbol{k})}$
- Semi-Implicit system contains terms needed for acoustic and buoyancy terms

$$\mathcal{L}\left(\mathbf{x}^{*}_{\text{phys}}\right)\mathbf{x}'_{\text{phys}} = \begin{cases} \mathbf{u}' - \mu\left(\frac{\mathbf{n}_{\text{b}}\cdot\mathbf{u}'}{\mathbf{n}_{\text{b}}\cdot\mathbf{z}_{\text{b}}}\right)\mathbf{z}_{\text{b}} \\ \quad +\tau_{u}\Delta t c_{p}\left(\theta'\nabla\Pi^{*} + \theta^{*}\nabla\Pi'\right), \\ \rho' + \tau_{\rho}\Delta t\nabla\cdot\left(\rho^{*}\mathbf{u}'\right), \\ \theta' + \tau_{\theta}\Delta t\mathbf{u}'\cdot\nabla\theta^{*}, \\ \frac{1-\kappa}{\kappa}\frac{\Pi'}{\Pi^{*}} - \frac{\rho'}{\rho^{*}} - \frac{\theta'}{\theta^{*}}, \end{cases}$$

[2] *Multigrid preconditioners for the mixed finite element dynamical core of the LFRic atmospheric model*, Maynard et al QJRMS20

Linear Solver

- Solver Outer system with Iterative (GCR) solver

$$\begin{pmatrix} M_2^{\mu,C} & & -P_{2\theta}^{\Pi^*} & -G^{\theta^*} \\ D^{\rho^*} & M_3 & & \\ P_{\theta 2}^{\theta^*} & & M_\theta & \\ & -M_3^{\rho^*} & -P_{3\theta}^{*} & M_3^{\Pi^*} \end{pmatrix} \begin{pmatrix} \widetilde{u}' \\ \widetilde{\rho}' \\ \widetilde{\theta}' \\ \widetilde{\Pi}' \end{pmatrix} = \begin{pmatrix} -\mathcal{R}_u \\ -\mathcal{R}_\rho \\ -\mathcal{R}_\theta \\ -\mathcal{R}_\Pi \end{pmatrix}$$

- Contains all couplings
- Preconditioned by approximate Schur complement for the pressure increment
- Velocity and potential temperature mass matrices are lumped
    - Drops non-orthogonal terms (cubed-sphere, terrain following)