Implementation of the Yin-Yang grid in PENCIL

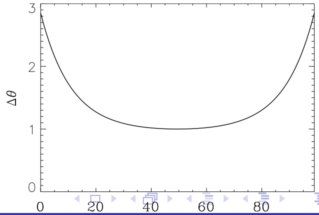
Matthias Rheinhardt

August 10, 2016

Motivation

Models in spherical geometry with full $\theta - \phi$ extent

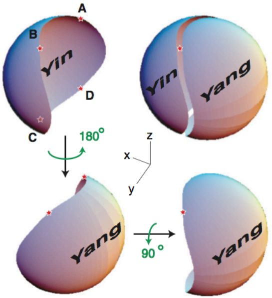
- $\theta = 0$ can't be a coordinate line
- ghost zones for θ boundaries lie beyond the poles
- hence: define grid as $\theta_i = \Delta\theta/2 + i\Delta\theta, i = 0, \dots, \pi/\Delta\theta - 1$,
  for grid point at ϕ_j fill ghost zones with values from $\phi_j + \pi$
  implemented by Dhruba/Fred
- problem: for θ grid lines close to poles
  stepsize in ϕ direction $r \sin\theta\Delta\phi$ gets small
  $\Longrightarrow \Delta t$ gets (too) small
- possible solution:
  make grid non-uniform in θ, e.g.:

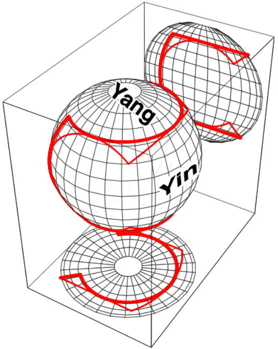


Yin-Yang grid for simulations over the full $\theta - \phi$ extent

Alternative:

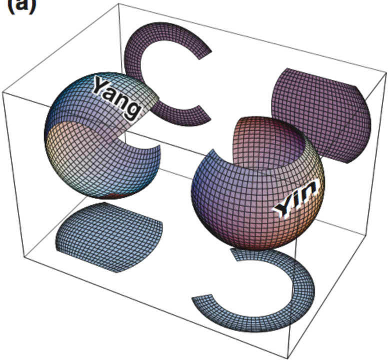
- cover spherical surface by 2 overlapping *identical* grids
- axis singularity of one grid covered regularly by the other
- grid cell size roughly uniform
- tb added: communication between Yin and Yang; algorithms internal to each grid untouched
  $\Longrightarrow$ code extensibility





decomposition without with overlap

(a)



coordinate ranges

$$\pi/4 \le \theta \le 3\pi/4, \quad \Delta\theta = \pi/2$$
$$\pi/4 \le \phi \le 7\pi/4, \quad \Delta\phi = 3\pi/2$$

transformation matrix

$$M = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$M = M^{\mathrm{T}} = M^{-1} \quad !$$

$\Longrightarrow$ only one transformation

Pencil Code: possible strategies

- double the variables
- *double the processors*
  $\Longrightarrow$ modification of communication only

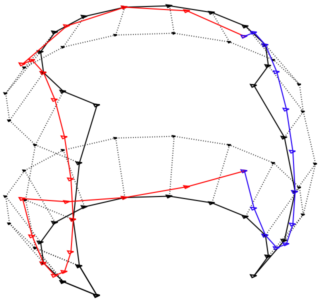
Implementation in Pencil Code

- switch on by `lyinyang=T`
- initialize
  - check constraint `nprocz = 3 nprocy`
  - set implicitly `nprocs = 2 ncpus`,
    create a MPI communicator for each grid:
    `MPI_COMM_GRID` $\neq$ `MPI_COMM_WORLD`
  - for boundary processors: set outer neighbours
  - transform and communicate ghost point coordinates
  - calculate interpolation parameters
  - transform global input data
- run
  - transform (vectors) and interpolate variables
  - communicate for ghostzone update
  - correct averages
- diagnostics
  - transform/interpolate data on Yang grid

Problems

standard layout:
each processor has exactly 8 neighbours in $\theta - \phi$ plane
$\Longrightarrow$ restrictions for processor numbers

4 x 12

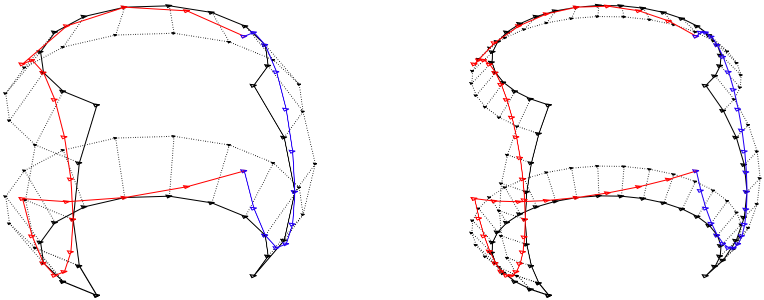


8 x 24

Problems

standard layout:
each processor has exactly 8 neighbours in $\theta - \phi$ plane
$\Longrightarrow$ restrictions for processor numbers

4 x 12 8 x 24



2nd layer entered !

Implementation in PENCIL CODE

in code:

- additional modules `yinyang, yinyang_mpi, noyinyang`
- + additional subroutines in `general, mpicomm`

in setup:

- `YINYANG = yinyang` (default `noyinyang`)
- `ncpus` — number of processors for one grid
  (but in submit script: 2*`ncpus`!)

Implementation in PENCIL CODE

in visualisation:

- object of `pc_read_var` contains usual variables, but with additional dimension of extent 2 for the two grids

  Yin-Yang specific:

  YZ, dimension(2,*) - a linear list of (θ, ϕ) coordinate pairs for the merged grids; technically an irregular grid

  TRIANGLES, dimension(3,*) - a list of triangles describing the triangulation of the merged grid

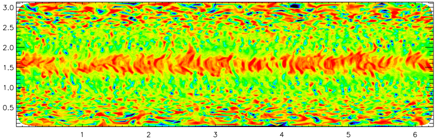
  UU_MERGE, dimension(nxgrid,(size(YZ))(2),3) - velocity defined on the merged grids

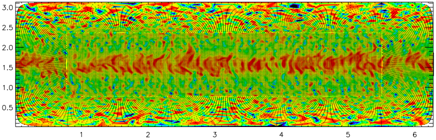
Implementation in PENCIL CODE

use merged data by, e.g.:

```
contour, reform(v.uu_merge(ir,*,0)), v.yz(1,*),
v.yz(0,*), /fill, nlev=30, tri=v.triangles
```

stellar convection:

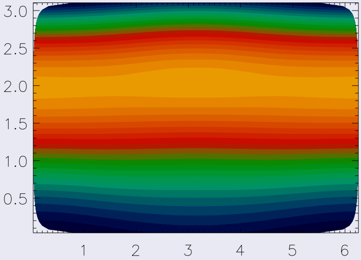




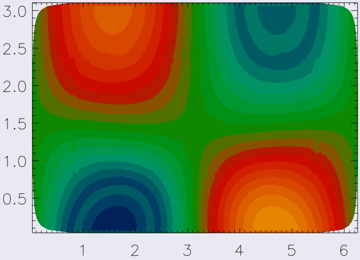
with grid

Problem

discontinuities/whiggles at grid interface
example: decay of dipolar meridional flow



u_r

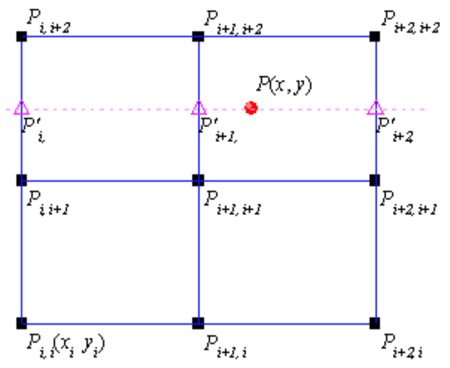


ω_r

Solution: biquadratic interpolation?

$$f(y,z) = a_0 + a_1 y + a_2 z + a_3 yz + a_4 y^2 + a_5 y^2 z + a_6 yz^2 + a_7 y^2 z^2$$

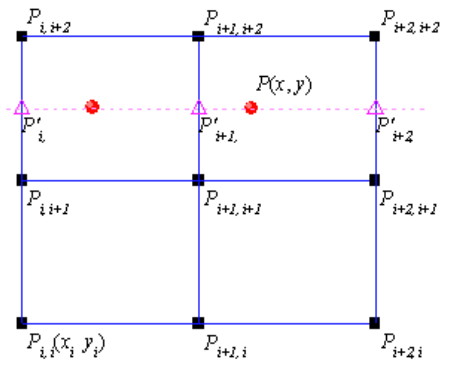
not unique:



Solution: biquadratic interpolation?

$$f(y,z) = a_0 + a_1 y + a_2 z + a_3 yz + a_4 y^2 + a_5 y^2 z + a_6 yz^2 + a_7 y^2 z^2$$

not unique:

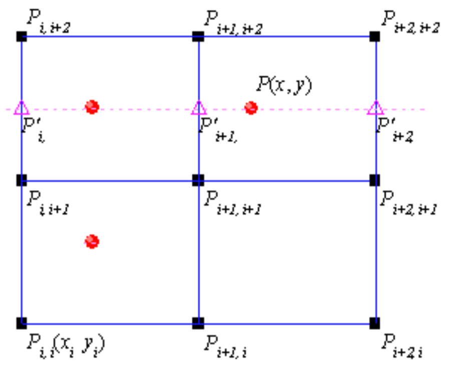


Implementation in PENCIL CODE

Solution: biquadratic interpolation?

$$f(y,z) = a_0 + a_1 y + a_2 z + a_3 yz + a_4 y^2 + a_5 y^2 z + a_6 yz^2 + a_7 y^2 z^2$$

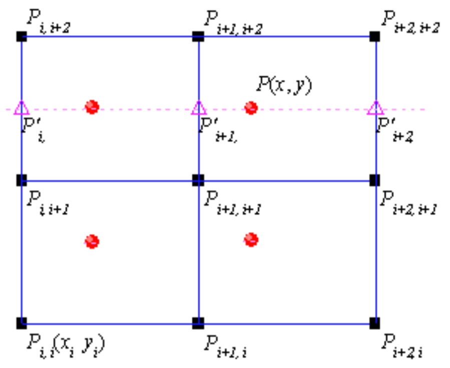
not unique:



Solution: biquadratic interpolation?

$$f(y,z) = a_0 + a_1 y + a_2 z + a_3 yz + a_4 y^2 + a_5 y^2 z + a_6 yz^2 + a_7 y^2 z^2$$

not unique:



How to weigh the 4 variants?

Implementation in PENCIL CODE

Status

- initialization & communication — done
- linear & quadratic interpolation — in testing
- z averages: for diagnostics — in debugging
  for PDEs — in coding
- y and volume averages — missing
- slices: yz – done, other — missing
- visualization: reading snapshots, z averages & yz slices — done