National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

University of Reading

UPSCALE K-scale project: work so far towards an inter-scale energy transfer tool

Sadie Bartholomew

CMS Weekly Meetings

2025-03-14

Context

- I am funded ~~3~~ 2 months on UPSCALE (funding project) K-scale (Reading-specific project)
- Delayed due to MO funding delay! By a whole month, so 3 → 2 months work for me
- Several Reading (mostly non-NCAS) colleagues involved, but I am working mostly only with Dan Shipley on a sub-goal of the project, namely to develop a Python tool which can perform a locality-based integral calculation for energy transfer in a grid-aware and efficient way
- It's going well, with lots of progress, though I am only on the project until end of March so only have 2-3 weeks left - eek
- This presentation: part hand-wavy talking, (possibly / time permitting) part live demo of prototype

Context

Pulling out relevant quotes from the 'K-Scale Simulations for AI Training' research plan:

- "UPSCALE has *two main goals*: (1) Evaluation of the newly developed Met Office CPM hierarchy of simulations. This will focus on the impact of resolving convective scales on the larger scale circulation (100 km to planetary scales) … (2) Building on the knowledge gained from the evaluation studies, we will develop and apply novel process-based diagnostics and propose sensitivity experiments to understand the mechanisms of up and down scale interactions in the CPMs vs. current 10km+ simulations with parameterised convection …"
- "We will *develop an open source Python package for diagnosing local upscale energy transfer that is suitable for use across the modelling hierarchy* (method and code development by Dan). Software tool development will be assisted by 20% FTE of a specialist Python programmer (Sadie in FY1) from NCAS CMS, who will help to optimise the diagnostic for use on large amounts of high-resolution data (the calculations are computationally intensive involving repeated application of a kernel-based convolution which can be time consuming). A pilot version of this tool will be produced (O1b)."

Our* task

*Our == Dan S and I

University of Reading

National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

Task

- Ultimately, Dan and I need to develop a Python tool which can implement the following integral calculation (for gridded model data) i**n a grid-aware and efficient way**
- Some code exists already (LoSSETT tool) but it is assumes planar geometry (flat Earth)!

- Velocity increments $\delta\mathbf{u}(\mathbf{r},\mathbf{x}) = \mathbf{u}(\mathbf{x}+\mathbf{r}) - \mathbf{u}(\mathbf{x}) \equiv \delta\mathbf{u}(\mathbf{r})$.
- Filtering function $G_\ell(\mathbf{r}) = \frac{1}{N}\exp\{-1/(1-[\mathbf{r}/(2\ell)^2])\}$ is smooth, non-negative and localised such that
  - N is a normalisation constant such that $\int d^3\mathbf{r} G(\mathbf{r}) = 1$,
  - $\int d\mathbf{r}\, G(\mathbf{r}) = 1$,
  - $\int d\mathbf{r}|\mathbf{r}|^2 G(\mathbf{r}) \approx 1$,
  - $G(\mathbf{r})$ is rescaled with ℓ as $G_\ell(\mathbf{r}) = \ell^{-3} G(\mathbf{r}/\ell)$.

the $\mathcal{D}$ term which represents energy transfer across scale ℓ is expressed as:

$$D_\ell(\mathbf{u}) = \frac{1}{4\ell}\int_V d\mathbf{r}(\nabla G_\ell)(\mathbf{r})\cdot\delta\mathbf{u}(\mathbf{r})|\delta\mathbf{u}(\mathbf{r})|^2.$$

$$\mathcal{D}_\ell := \int \nabla G_\ell(\mathbf{r})\cdot\delta\mathbf{u}|\delta\mathbf{u}|^2 \mathrm{d}^d\mathbf{r}.$$

$$\int_{S^2(R)} \nabla G_\ell(\mathbf{r})\cdot\delta\mathbf{u}|\delta\mathbf{u}|^2\mathrm{d}^2\mathbf{r} = \int_0^{\pi R}\int_0^{2\pi} \frac{\partial G_\ell(r)}{\partial r}\hat{\mathbf{r}}\cdot\delta\mathbf{u}|\delta\mathbf{u}|^2 R\sin\left(\frac{r}{R}\right)\mathrm{d}\phi\,\mathrm{d}r$$

$$= R\int_0^{\pi R}\frac{\partial G_\ell(r)}{\partial r}\sin\left(\frac{r}{R}\right)\left[\int_0^{2\pi}\hat{\mathbf{r}}\cdot\delta\mathbf{u}|\delta\mathbf{u}|^2\mathrm{d}\phi\right]\mathrm{d}r.$$

National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

Task

- Ultimately, Dan and I need to develop a Python tool which can implement the following integral calculation for model gridded data **in a grid-aware and efficient way**
- **Key computational difficulty/grind: finding the spherical (great circle) distances and bearing angles from grid point to each other grid point, in order to do the integral over r, ϕ which sums u field vectors based on distance from the origin grid point and weights by bearing angle:**

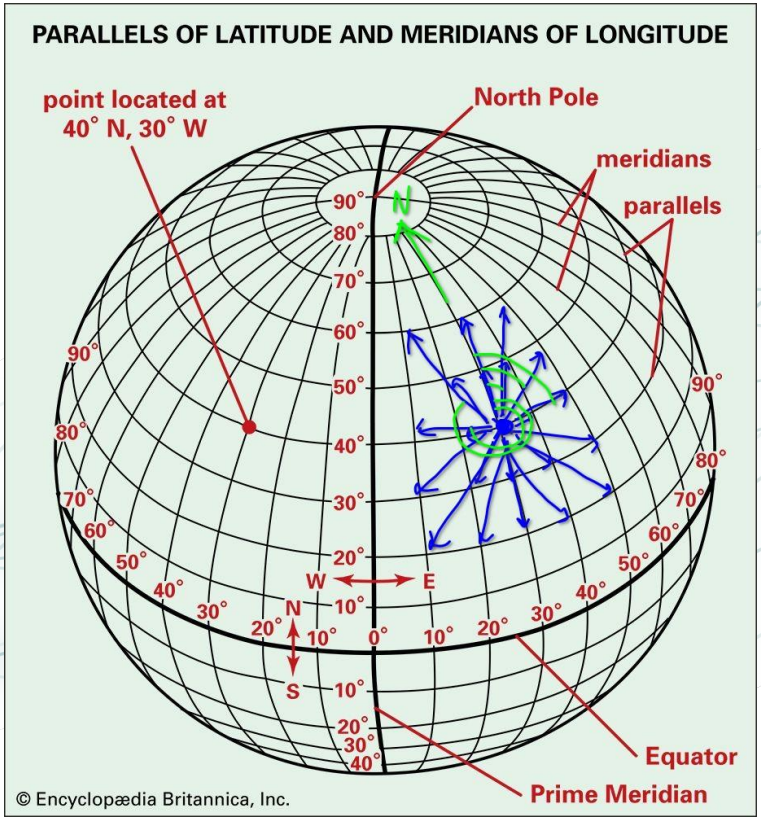
$$\int_{S^2(R)} \nabla G_\ell(\mathbf{r}) \cdot \delta\mathbf{u}|\delta\mathbf{u}|^2 \mathrm{d}^2\mathbf{r} = \int_0^{\pi R} \int_0^{2\pi} \frac{\partial G_\ell(r)}{\partial r} \hat{\mathbf{r}} \cdot \delta\mathbf{u}|\delta\mathbf{u}|^2 R \sin\left(\frac{r}{R}\right) \mathrm{d}\phi \, \mathrm{d}r$$

$$= R \int_0^{\pi R} \frac{\partial G_\ell(r)}{\partial r} \sin\left(\frac{r}{R}\right) \left[\int_0^{2\pi} \hat{\mathbf{r}} \cdot \delta\mathbf{u}|\delta\mathbf{u}|^2 \mathrm{d}\phi\right] \mathrm{d}r.$$

University of Reading

National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

- We are **focusing on a regular lat-lon grid of arbitrary resolution**, first. A grid of other form can, for now, possibly be **regridded** to this form so it can be supported still
- Result is static so we can provide datasets with these pre-calculated for given resolution grids!
- Datasets don't need to be global e.g. could be over a given region - but it makes sense to consider whole globe as most general case
- For an example grid point shown here, blue lines show some illustrative, near origin, spherical (i.e. great circle) distances **r** and green arcs the bearing angles **ϕ** (relative to North) for some cases



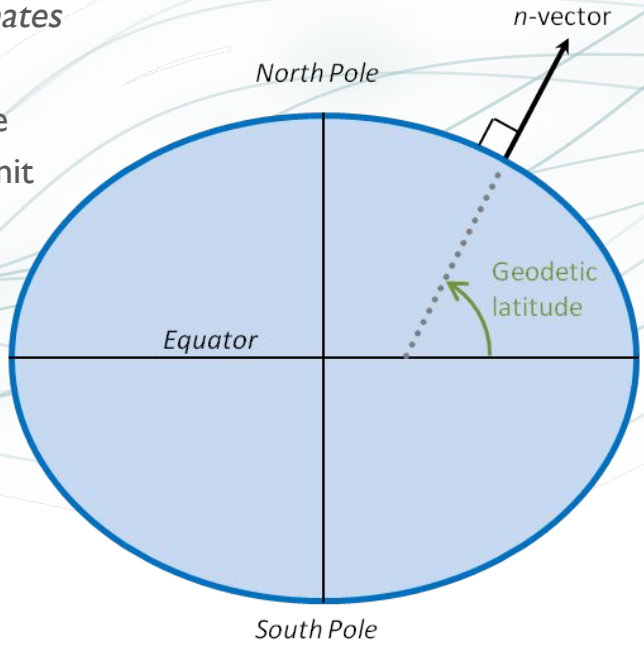
University of Reading

National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

Approach

Simplifying the geodesy calculations: N-vectors!

- [Wikipedia entry for n-vector:] "The n-vector representation (also called geodetic normal or ellipsoid normal vector) is a *three-parameter non-singular representation well-suited for replacing geodetic coordinates (latitude and longitude) for horizontal position representation in mathematical calculations and computer algorithms*. Geometrically, the n-vector for a given position on an ellipsoid is the outward-pointing unit vector that is normal in that position to the ellipsoid."
- Parameterise by 3-component vector instead of (lat, lon) coordinates. Quick to convert between the two forms.
- Python nvector library is proving really useful and is fast for the r, ϕ computations: see https://github.com/pbrod/nvector or the language agnostic spec. at https://www.ffi.no/en/research/n-vector/



University of Reading

National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

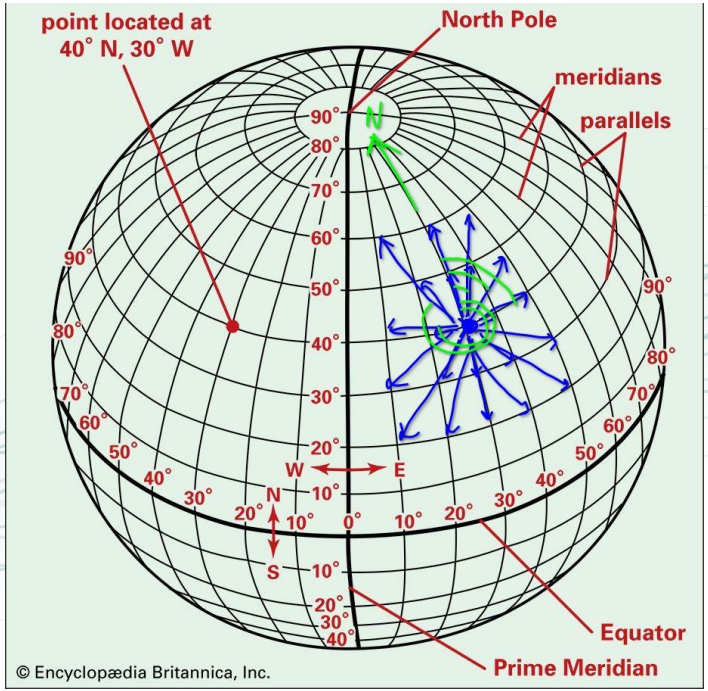
Basic approach

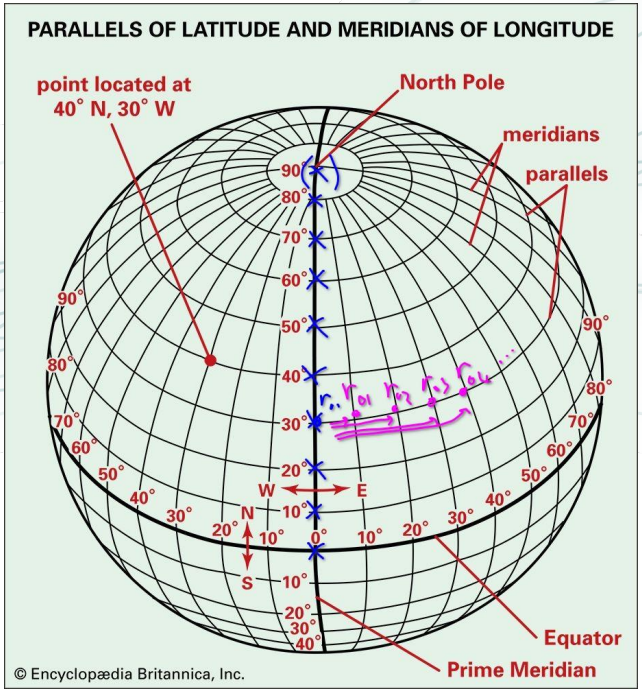
Using code ANVIL in https://github.com/sadielbartholomew/ANVIL ('Application of N-Vectors for Integration by Locality') - might be more generally useful than for energy transfer specific calculation. Using cf-python and numpy, dask array operations etc. For a given input u field:

1. Find cf field with *great circle distance* from origin point to every other grid point, for every point in grid: required for dr in integral
2. Find another cf field with *bearing (azimuth angle)* from origin point to every other grid point, for every point in grid: required for $d\phi$ in integral
3. Use *cf.where() conditional* to mask out all points not within a given r + dr distance from the origin point, and weight with the phi, and basic field arithmetic on the u field otherwise, to perform the integral incrementally

Reducing the number of calculations via symmetry of grid

- In order to perform the integral on a discrete basis for all grid points, we need to know the great circle distance and bearing angle from each grid point to every other.
- To avoid duplicate calculation, can use symmetry to reduce down the number of fields we need to find: reflection in equator for latitude, and rotation about earth's axis for longitude. Therefore only need to calculate grid of r, phi for half the latitude points, and can reflect or rotate to get result from those





National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

Progress

- *[In-terminal and in-browser code demo and view]*
- For a 160 lat x 320 lon grid, as example, need to find 160/2 = 80 fields of great circle distances and 80 fields of bearing azimuth angles.
- After a little optimisation, it is taking ~1 second per distance field and ~7s per angle field => around 10 minutes to calculate all of the fields needed for the integral
- Will take a little more time to perform the rotations and reflections to get each r, phi field and then masking over r and weighting by angle to get the result of the integral, but shouldn't take more time than the r, phi field calculations
- Scaling with grid size - obviously will take longer for higher-resolution grids but more precise scaling pattern TBC