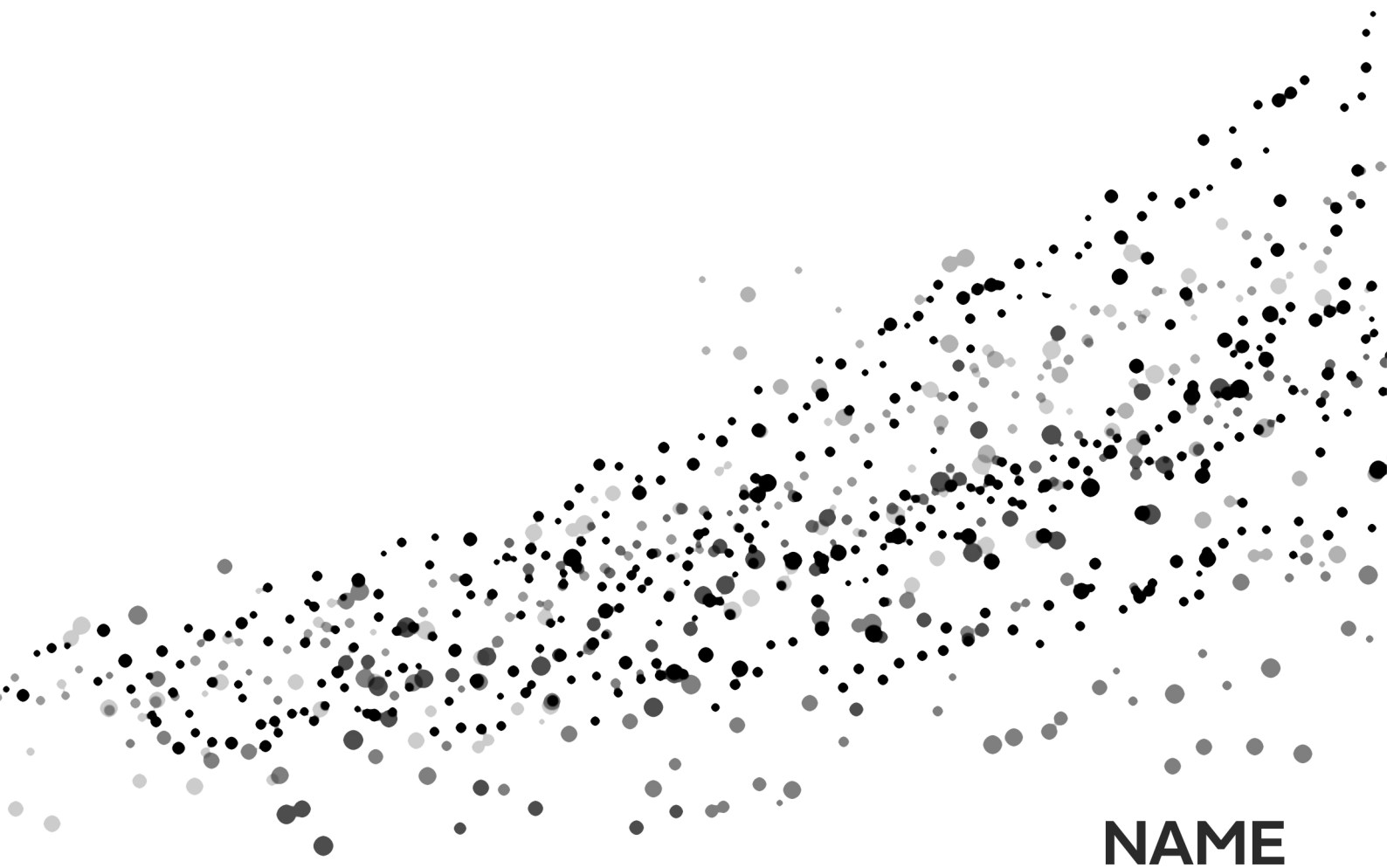
Met Office

NAME Technical Specification Document C05

Grids

David J. Thomson

Documentation issued with	**:**	NAME Version 8.7
Document last updated for	**:**	NAME Version 8.7
Last updated on	**:**	02/12/2024



NAME

Numerical Atmospheric-Dispersion Modelling Environment

Met Office, FitzRoy Road, Exeter, Devon, EX1 3PB, United Kingdom

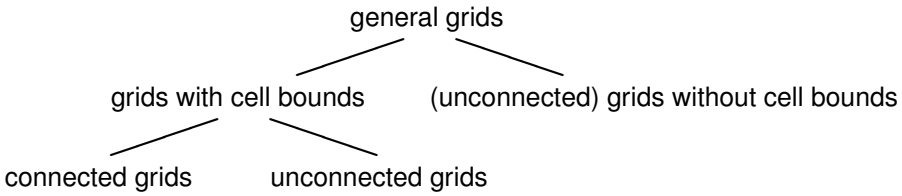
This document has been written primarily to document the use of cubed-sphere and unstructured connected (netCDF "UGRID") grids in NAME, but with the intention of extending it, in the future, to cover all grids in NAME. This means that some sections are currently incomplete. The missing material is however covered from a user perspective in the User Guide.

1 Grid types

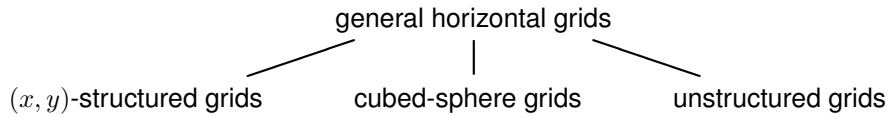
NAME uses horizontal, vertical, temporal and data grids. A grid can be regarded as a set of grid points, specified by the value of the horizontal position, of the vertical position, of the time, or the value of other types of data. An example of a data grid might be a grid of concentration values which is used to store the probability distribution of concentration.

The grids can be combined. For example quantities are often presented on a spatial grid which combines a horizontal and a vertical grid. Currently in NAME such grid combinations are always "product grids" with, for example, the vertical grid definition being independent of the horizontal position and vice versa. (The vertical grid might be defined in terms of a height-above-ground vertical coordinate system with the height of the ground above sea level varying in the horizontal, or a height-based eta coordinate system, or a pressure-based coordinate system. One might then think of the vertical grid as depending on the horizontal position. However the definition of the vertical grid in its defining coordinate system does not change with horizontal position.)[1]

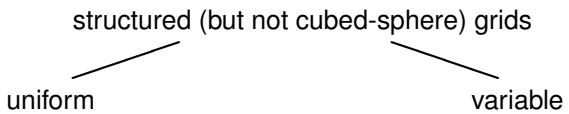
Grids can be classified in various ways. Sometimes each grid point will represent a range of values. We generally call this range a "cell" and we call the cell boundaries the "bounds" of the cell. An example of this might be a grid on which concentration is calculated by summing the mass of all particles in the cell. Sometimes these cells will join up and not overlap to give a "connected grid", and sometimes not. An example of the latter might be when one wants values of the concentration at some observation locations and requires a cell around each location. The relationships between these grid types are illustrated here:



Horizontal grids can also be classified as "(x, y)-structured grids", where the grid is a product grid formed from an x-grid and a y-grid with (x, y) being the horizontal coordinate system used to define the grid, as "cubed-sphere grids", which are structured but in a different way and which will be defined in detail below, and "unstructured grids":



Horizontal (x, y)-structured grids and vertical, temporal and data grids can be classified as having uniform or variable spacing (in terms of the defining coordinate(s)). For uniform grids with cell bounds, as well as the grid point spacing being uniform, the grid points are in the exact centre of the cell (again in terms of the defining coordinate(s)). Regarding 1-d grids as always being structured, this is illustrated here:



Not every combination of grid types is supported in NAME as there is no requirement for some combinations. However these grid types are perhaps a useful way of classifying the grids that are supported.

Nearly all NAME grids are fixed and have a finite number of points. However for data grids we allow grids with potentially infinitely many points, although only a finite number of points are active at the same time. This is useful for calculating probability distributions of e.g. concentration. The set of active levels can adapt as higher concentrations are found by adding grid points at higher concentrations and discarding lower values, on the

[1] At some point there may be a need to introduce "spatial grids" in NAME that are not products of horizontal and vertical grids, for example in connection with complicated geometries around buildings. However there is currently no support for such grids.

basis that the interest is usually in high concentration values. Which levels are active is stored outside the grid as one might want to use the same grid definition with many concentration pdfs (e.g. at different locations).[2]

2 Horizontal grids

2.1 (x,y)-structured grids

(x, y)-structured grids are product grids constructed from an x-grid and a y-grid defined using one of NAME's horizontal coordinate systems. In detail, the grid point coordinates in the chosen coordinate system take the form (x_i, y_j), $i = 1, \ldots n_x$, $j = 1, \ldots n_y$. Cell bounds are defined in the obvious way from x and y bounds x_{bi}, $i = 1, \ldots n_x + 1$ and y_{bj}, $j = 1, \ldots n_y + 1$. The x_i, y_j, x_{bi} and y_{bj} must all be strictly monotonic (although they can be decreasing) with the grid points lying within the bounds.

There is a uniformly spaced version in which the grid points and the bounds are equally spaced with the grid points in the exact centre of the cells in the (x, y)-plane. Identifying the cell that a point of interest is in is much more efficient for the uniformly spaced version.

In terms of the grid types introduced in section 1, (x, y)-structured grids are connected grids with cell bounds and might be uniform or variable. An (x, y)-structured grid is specified by giving a horizontal coordinate system, together with n_x, n_y, x_i, y_j, x_{bi} and y_{bj}.

2.2 Cubed-sphere grids

To construct a cubed-sphere grid, we start by taking a cube centred at the centre of our spherical Earth, and we project it from the centre of the sphere onto the sphere. The projected images of the six faces of the cube on the sphere are called the cubed-sphere "panels". Within each face of the cube we can define a two-dimensional cartesian coordinate system with axes aligned with the face edges and with the origin at the face centre.

We now consider how to construct coordinates and grids on each panel of the sphere. If we identify points on the face and on the panel using a projection from the centre of the sphere (Sadourny, 1972), and give identified points the same coordinates, then the coordinates on the panel are (modulo a choice of scaling factor) given by $\tan\lambda_1$ and $\tan\lambda_2$, where λ_1 and λ_2 are longitude coordinates defined using two longitude-latitude coordinate systems with poles at the centres of adjacent panels and with the zero longitude lines chosen to go through the panel centre. (In general we will use λ_1 and λ_2 to refer to the longitude coordinates of whichever of the panels is being considered.) A uniform cartesian grid on the cube's face (aligned with the face edges and with cell boundaries at the face edges) then maps to a grid on the panel and this grid is uniform when plotted on the $(\tan\lambda_1, \tan\lambda_2)$-plane. If this is done for each face and panel (each with the same resolution) this gives the "gnomonic cubed-sphere grid". The name arises from the term "gnomonic projection" for a projection from a point on to a plane. We note that straight lines on the cube face and great circles arcs on the sphere map to each other, and so straight lines on the $(\tan\lambda_1, \tan\lambda_2)$-plane and great circle arcs on the sphere do too.

More generally we can define coordinates $(f(\lambda_1), f(\lambda_2))$ on the panel for some odd monotonic function f. We can then define cubed-sphere grids where the grid on each panel is, on the $(f(\lambda_1), f(\lambda_2))$-plane, uniformly spaced and aligned with the coordinate directions, with cell boundaries along the panel edges and with grid points at the cell centres (on the $(f(\lambda_1), f(\lambda_2))$-plane). If $f(\lambda) \not\propto \tan\lambda$, this stretches the grid relative to the gnomonic cubed-sphere grid, but keeps the coordinate lines as lines of constant λ_1 or λ_2. The coordinate lines on the panel remain great circles but straight lines on the $(f(\lambda_1), f(\lambda_2))$-plane which are not aligned with the coordinate directions and great circles which are not coordinate lines no longer map to each other. While it's not necessary to map the panel to the cube face, some things are easier to illustrate on the cube than on the sphere (e.g. area-filling curves – see below), especially if we use a map that makes our grids on the sphere panels uniform on the cube faces. For this we choose to map each panel to the corresponding cube face by mapping the point with coordinates $(f(\lambda_1), f(\lambda_2))$ on the panel to the point with the same coordinates on the cube face (using our cartesian coordinates on the face and adding any scaling factor needed to map the panel edges to the cube-face edges). This means the points on the panel no longer project (using the gnomonic projection) from the centre of the sphere to the corresponding point on the face.

[2]There is also a potential need for temporal grids with infinitely many points. Such a grid could obviously not be used to store data on all points but could be useful for specifying the times for which met data might be available or for which output should be generated if the run progresses to that point. This is not yet supported in NAME. Instead, currently, met times are specified without using a NAME temporal grid and, in a long running air quality simulation, one needs to specify an arbitrary but large number of points in the output temporal grid.

If $f(\lambda) \propto \tan\lambda$ we have the gnomonic cubed sphere and if $f(\lambda) \propto \lambda$ we have the "equiangular cubed sphere" (Ronchi et al., 1996). For the "equidistant cubed sphere" (Auer, 2021) where the coordinates change at a constant rate along the panel edges, we need $f(\lambda) \propto \arctan(\frac{1}{\sqrt{2}}\tan\lambda)$ (this is derived below).

We now need to make these definitions more precise. In particular we need to provide a way to label the 6 cube sphere panels, clarify for each panel which of the two longitude coordinates is λ_1 and which is λ_2, and define the directions in which λ_1 and λ_2 increase. We define the cubed sphere relative to an "underlying" longitude-latitude coordinate system which can be rotated relative to the standard longitude-latitude coordinate system. We assume the cubed sphere has panels centred at 0°E 0°N, 90°E 0°N and 90°N in this coordinate system. We number the panels according to the numbering system for the cube faces shown in figure 1 with faces 1, 3 and 5 corresponding to the panels centred on 0°E 0°N, 90°E 0°N and 90°N. The cube faces are numbered 1 to 6 following the blue line which turns right and left on alternate faces from the point of view of someone walking along the blue line on the outside surface of the cube. We define cartesian coordinates on each cube face with the origin at the centre of the face using the axis orientations shown in the figure with the purple axis regarded as the first coordinate and the green axis as the second coordinate. These coordinates have the same relationship to the blue curve on each face. These coordinates are right-handed on odd numbered faces and left-handed on even numbered faces. We define longitude coordinates λ_1 and λ_2 on each cubed-sphere panel as longitudes defined using poles in the centres of the adjacent panels, with the zero longitude lines passing through the centre of the panel. To define which coordinate is λ_1 and which is λ_2, we define λ_1 as increasing in the direction corresponding to the purple arrow in figure 1 and λ_2 as increasing in the direction corresponding to the green arrow. As an example, for the cubed-sphere panel centred at 0°E 0°N, λ_1 is the normal longitude coordinate and λ_2 is the longitude coordinate defined with "north pole" at 90°W 0°N and "south pole" at 90°E 0°N (so that λ_2 increases across the panel towards the north of the underlying longitude-latitude coordinate system) and with $\lambda_2 = 0$ being the half great circle joining 90°W 0°N to 90°E 0°N through 0°E 0°N. The panel coordinates are then defined as $(f(\lambda_1), f(\lambda_2))$ and correspond to the cartesian coordinates on the cube face under the mapping between the cube and the sphere. Our motivation for this apparently complicated set of definitions is to ensure that the description of the cubed sphere relative to a given panel is the same for all panels apart from the minor inconvenience of the odd and even numbered panels being right and left handed. For example if you leave any panel in the direction of increasing λ_1, the panel index increases by 2 mod 6 and, at the panel edge, λ_1 on the new panel equals λ_2 on the old panel and λ_2 on the new panel and λ_1 on the old panel both equal $\pi/4$. These sort of relationships reduce the number of if-tests needed in the code and hopefully therefore the chances of errors in the coding. We also introduce a 3-d right-handed coordinate system (X, Y, Z) as shown in the figure. This coordinate system is centred at the centre of the cube / sphere with the X, Y and Z axes passing through 0°E 0°N, 90°E 0°N and 90°N.

We have now introduced twelve longitude coordinates, namely the λ_1 and λ_2 for each of the six panels. We note however that these twelve coordinates are all simply related to one of three longitude coordinates. The three longitude coordinates λ_x, λ_y and λ_z are defined with "north poles" at 0°E 0°N, 90°E 0°N and 90°N. For each of these three longitude coordinates, we take the zero longitude line as the longitude line running away from the "north pole" on the opposite side of the pole to the side on which the blue line approaches the pole, i.e., following the blue line, one approaches the north pole along longitude π. Note λ_z is the normal longitude coordinate (of the underlying longitude-latitude coordinate system). In the panels containing the north poles (panels 1, 3 and 5), the zero longitude line (for the longitude coordinate defined with north pole in the panel) runs away from the pole in the direction of increase of the first coordinate. The λ_1 and λ_2 on each panel can now be expressed in terms of λ_x, λ_y and λ_z as shown in table 1.

panel	λ_1	λ_2	first coordinate $f(\lambda_1)$	second coordinate $f(\lambda_2)$
1	λ_z	$-(\lambda_y - \pi/2)$	$f(\lambda_z)$	$f(-(\lambda_y - \pi/2))$
2	$\lambda_y \pm \pi$	$-(\lambda_x + \pi/2)$	$f(\lambda_y \pm \pi)$	$f(-(\lambda_x + \pi/2))$
3	λ_x	$-(\lambda_z - \pi/2)$	$f(\lambda_x)$	$f(-(\lambda_z - \pi/2))$
4	$\lambda_z \pm \pi$	$-(\lambda_y + \pi/2)$	$f(\lambda_z \pm \pi)$	$f(-(\lambda_y + \pi/2))$
5	λ_y	$-(\lambda_x - \pi/2)$	$f(\lambda_y)$	$f(-(\lambda_x - \pi/2))$
6	$\lambda_x \pm \pi$	$-(\lambda_z + \pi/2)$	$f(\lambda_x \pm \pi)$	$f(-(\lambda_z + \pi/2))$

Table 1: λ_1 and λ_2 and the panel coordinates $f(\lambda_1)$ and $f(\lambda_2)$ used in the cubed-sphere panels.

The various coordinate conversions required for the cubed-sphere grids are reasonably straightforward and we have coded these conversions with the grids code rather than with the coordinate systems code. This means the coordinate systems code can remain restricted to orthogonal horizontal coordinate systems and it doesn't need to consider coordinate systems consisting of two longitudinal coordinates or three longitudi-

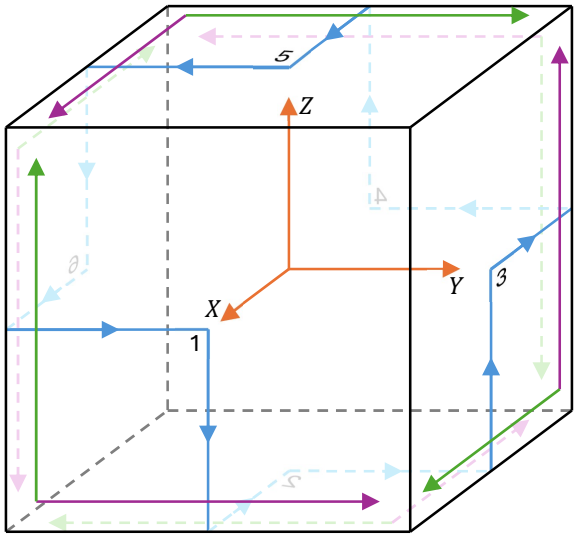


Figure 1: Illustration of some definitions linked to the cube. The cube faces are numbered 1 to 6 following the blue line which turns right and left on alternate faces. X, Y and Z form a right-handed cartesian coordinate system with origin at the centre of the cube. Each face has a cartesian coordinate system indicated by the purple and green axes which, on each face, are defined in the same way relative to the blue line. The axes indicate the direction of increase of the coordinates, but not the origin which is at the centre of the face (in the interests of clarity we haven't attempted to depict the origin). With the purple axis regarded as the first coordinate, the coordinate system is right handed on odd-numbered faces and left handed on even-numbered faces.

nal coordinates such as $(\lambda_x, \lambda_y, \lambda_z)$. It also means it's easy to support cubed spheres defined relative to an arbitrary rotated longitude-latitude coordinate system without the ability in the coordinate systems code to define rotated longitude (or longitude-latitude) coordinate systems relative to another rotated longitude-latitude coordinate system. Because of this we don't actually make direct use of $(\lambda_x, \lambda_y, \lambda_z)$ in the code. Instead we always work directly with a panel's λ_1 and λ_2 and convert between λ_1 and λ_2 and normal longitude and latitude coordinates without reference to $(\lambda_x, \lambda_y, \lambda_z)$.

Starting from the normal longitude (λ) and latitude (ϕ) coordinates of a point of interest (in the underlying longitude-latitude coordinate system), we can calculate X, Y and Z using

$$X = \cos\phi\cos\lambda, \quad Y = \cos\phi\sin\lambda, \quad Z = \sin\phi$$

(taking, for convenience, the sphere radius to be 1). We can also calculate which panel the point is in as follows

$$\begin{array}{lll} |X| \geq |Y|, |Z| & X > 0: & \text{panel 1} \\ |X| \geq |Y|, |Z| & X < 0: & \text{panel 4} \\ |Y| \geq |Z|, |X| & Y > 0: & \text{panel 3} \\ |Y| \geq |Z|, |X| & Y < 0: & \text{panel 6} \\ |Z| \geq |X|, |Y| & Z > 0: & \text{panel 5} \\ |Z| \geq |X|, |Y| & Z < 0: & \text{panel 2.} \end{array}$$

This follows because X, Y and Z are a measure, namely the cosine, of the angles between the position vector (relative to the centre of the sphere) of the point of interest and the X, Y and Z axes, and because, for points in the interior of a panel, the angle with the position vector of the panel centre is less than the angle with the position vectors of the centres of the neighbouring panels. (Suppose the point is in the interior of panel 1. Consider moving the point so that the gnomonic projection of the point on the cube moves parallel to the Y (Z) axis until it reaches the nearer of the two edges of the cube face parallel to the Z (Y) axis. This increases the angle with the X axis and decreases the angle with the Y (Z) axis while, on reaching the edge, the angles are equal.) Having determined which panel the point is in, we can calculate λ_1 and λ_2 as follows

$$\begin{array}{lll} \text{panel 1:} & \lambda_1 = \arctan(Y/X), & \lambda_2 = \arctan(Z/X) \\ \text{panel 2:} & \lambda_1 = \arctan(X/Z), & \lambda_2 = \arctan(Y/Z) \\ \text{panel 3:} & \lambda_1 = \arctan(Z/Y), & \lambda_2 = \arctan(X/Y) \\ \text{panel 4:} & \lambda_1 = \arctan(Y/X), & \lambda_2 = \arctan(Z/X) \\ \text{panel 5:} & \lambda_1 = \arctan(X/Z), & \lambda_2 = \arctan(Y/Z) \\ \text{panel 6:} & \lambda_1 = \arctan(Z/Y), & \lambda_2 = \arctan(X/Y). \end{array}$$

We can then determine the panel coordinates $f(\lambda_1)$ and $f(\lambda_2)$. These formulae can also be used to determine the values of a particular panel's coordinates for a point just outside the panel. However this might not work for points a long way outside the panel. For example, depending on the form of f, the result can blow up. We note that λ_x, λ_y and λ_z can be expressed as

$$\lambda_x = \arctan_2(Z, Y), \quad \lambda_y = \arctan_2(X, Z), \quad \lambda_z (\equiv \lambda) = \arctan_2(Y, X)$$

(where we define $\arctan_2$ as in Fortran with $\arctan_2(y, x) = \arctan(y/x) \mod \pi$ and $\arctan(r\sin\theta, r\cos\theta) = \theta \mod 2\pi$).

Conversely, given the two panel coordinates $f(\lambda_1)$ and $f(\lambda_2)$ for a point in a particular panel, we can apply f^{-1} to determine λ_1 and λ_2. Then, depending on the panel, we can calculate

$$\begin{aligned}
&\text{panel 1:} \quad X = +\frac{1}{\sqrt{\tan^2\lambda_1 + \tan^2\lambda_2 + 1}}, \quad Y = X\tan\lambda_1, \quad Z = X\tan\lambda_2 \\
&\text{panel 2:} \quad Z = -\frac{1}{\sqrt{\tan^2\lambda_1 + \tan^2\lambda_2 + 1}}, \quad X = Z\tan\lambda_1, \quad Y = Z\tan\lambda_2 \\
&\text{panel 3:} \quad Y = +\frac{1}{\sqrt{\tan^2\lambda_1 + \tan^2\lambda_2 + 1}}, \quad Z = Y\tan\lambda_1, \quad X = Y\tan\lambda_2 \\
&\text{panel 4:} \quad X = -\frac{1}{\sqrt{\tan^2\lambda_1 + \tan^2\lambda_2 + 1}}, \quad Y = X\tan\lambda_1, \quad Z = X\tan\lambda_2 \\
&\text{panel 5:} \quad Z = +\frac{1}{\sqrt{\tan^2\lambda_1 + \tan^2\lambda_2 + 1}}, \quad X = Z\tan\lambda_1, \quad Y = Z\tan\lambda_2 \\
&\text{panel 6:} \quad Y = -\frac{1}{\sqrt{\tan^2\lambda_1 + \tan^2\lambda_2 + 1}}, \quad Z = Y\tan\lambda_1, \quad X = Y\tan\lambda_2.
\end{aligned}$$

We can then compute λ and ϕ from

$$\lambda = \arctan_2(Y, X), \quad \phi = \arctan_2(Z, \sqrt{X^2 + Y^2}).$$

As above these formulae can be used if the location is just outside the panel whose panel coordinates λ_1 and λ_2 we are using. However they should not be used a long way outside the panel. For example λ_1 and λ_2 do not always determine the location uniquely a long way outside the panel.

We can now derive the form of f for the equidistant cubed sphere. For, e.g., the east and west edges of the panel centred on 0°E 0°N (here $|X| = |Y|$), we want $f(\lambda_2) \propto$ latitude $= \arctan(Z/\sqrt{X^2+Y^2}) = \arctan(Z/\sqrt{2}|X|) = \arctan(\frac{1}{\sqrt{2}}\tan\lambda_2)$.

We note that it is possible to construct an area-filling Hilbert curve on the cube following the blue line. This is illustrated in figure 2. The curve can be mapped onto the sphere using the identification of panel coordinates on the sphere with cartesian coordinates on the cube face to obtain a Hilbert curve on the sphere.[3]

In terms of the grid types introduced in section 1, cubed-sphere grids are connected grids with cell bounds. A cubed-sphere grid is specified by giving a (possibly rotated) longitude-latitude coordinate system using units of radians, the number of grid points across each panel, the type of cubed sphere (gnomonic, equiangular or equidistant[4]), and the interpolation method (spherical or planar barycentric interpolation – see section 3 below).

2.3 Unstructured connected grids

Unstructured connected grids with cell bounds are essentially horizontal meshes as defined in the netCDF UGRID format, although currently we only support meshes where all the cells are quadrilaterals. Such a grid is specified by giving (i) the number of cells, cell edges and cell vertices, (ii) maps to define which vertices belong to which edges, which vertices belong to which cells, and which edges belong to which cells, (iii) a (possibly rotated) longitude-latitude coordinate system, and (iv) the locations, in that coordinate system, of the central grid point in each cell, of a central point in each edge and of the vertices.

Often such grids are actually structured in some way but are presented in a way that doesn't make this structure explicit. In this situation NAME can calculate a mapping from an equivalent structured grid to the unstructured grid which acts as a look-up table for the unstructured grid. For example, given a location of interest,

[3] This is not currently used in NAME but may be useful in future to extend the patch system used for grouping particles or for working with unstructured grids.

[4] Other types could be supported in NAME if this is required.

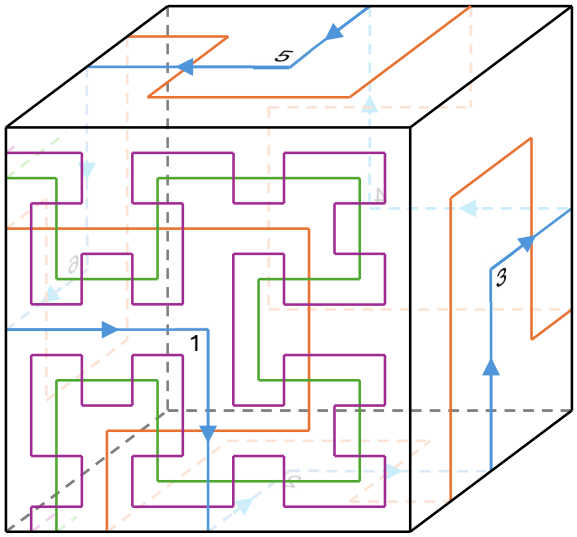


Figure 2: Illustration of an area-filling Hilbert curve on the cube. The Hilbert curve is constructed to follow the blue curve, which can be regarded as the zeroth iteration of the curve. The first, second and third iterations are shown in orange, green and purple with the second and third iterations shown on the front face only.

we can identify the structured grid cell in which it lies and then use the mapping to identify the unstructured grid cell.

If there is such a mapping from a structured to an unstructured grid, then the paths taken by the edges in the unstructured grid are assumed to follow the structured grid edges. Also the locations of the central grid points in each cell and edge and of the vertices should follow those in the unstructured grid. We check this is so when setting up the map, but we allow some errors and NAME will report on these errors if they are deemed significant. In this situation, if the unstructured grid is accessed via the map, it will be as if the correct values were those in the structured grid.

In the absence of such a mapping the edges are assumed to follow great circles. Here we check that the central grid point in each edge lies on the great circle, but again we allow some errors.

2.4 Unstructured unconnected grids

To be written.

3 Interpolation of data held on horizontal grids

We first discuss some general interpolation techniques and then describe how these are applied with specific types of horizontal grid. Interpolation is only possible for connected grids. Generally we are seeking interpolation methods which are continuous and local.

3.1 Interpolation methods

3.1.1 Bilinear interpolation

This is a straightforward and well known method which can be applied when the points holding the data form coordinate aligned rectangles in the (x, y)-coordinate plane for some horizontal coordinate system (x, y).

3.1.2 Spherical and planar barycentric interpolation

For cubed-sphere and unstructured grids, bilinear interpolation, at least in its basic form, can't be applied. Here we consider barycentric interpolation for triangular and quadrilateral cells with data on the vertices. Our main focus is spherical barycentric interpolation but we also consider barycentric interpolation on a projection

plane as this is a possible alternative approach for some grids. See Floater (2015) for a review of barycentric coordinates. Lomax (2021) developed the planar case for use in connection with NWP data assimilation.

In spherical barycentric interpolation we choose the weights so that the (weighted) centroid of the cell's vertices lies on a line between the point of interest and the centre of the sphere. In doing this we assume the cell can be entirely contained in a hemisphere to remove any ambiguity over which end of the line the point is on. We assume (or choose) the cell edges to be great circles. We use the centre of the sphere as the origin for position vectors, with the position vectors of the cell corners denoted by $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ for triangular cells and $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$ for quadrilateral cells, and with the position vector of the point of interest denoted by $\mathbf{p}$. This is illustrated together with the notation for the weights in figure 3. For the quadrilateral cells we need an extra constraint to fix the weights (in addition to the weights summing to one and the weighted barycentre lying on the line between the centre of the sphere and the point of interest). For this we choose the weights to take the usual bilinear interpolation form as shown in the figure. Any two of α, β and γ for triangular cells and α and β for quadrilateral cells can be regarded as coordinates within the cell. Provided the cell is convex (which is automatic for triangular cells), values of the two coordinates in the range $[0, 1]$ determine a unique point of interest $\mathbf{p}$ within the cell and a point of interest $\mathbf{p}$ determines unique coordinate values in the range $[0, 1]$. Here convexity can be defined as either convexity of the 3-d shape formed by the lines joining points of the cell to the centre of the sphere, or by requiring that the great circle arc joining any two points of the cell lies entirely within the cell. Figure 4 (left) shows why we need the convex restriction. In the situation shown there are coordinates in the range $[0, 1]$, namely $\alpha = 2/3$ and $0 < \beta < 1/2$, that correspond to points outside the cell. Also, the point where the line $\alpha = 2/3$ crosses $\mathbf{bc}$ can be represented in two ways with coordinates in the range $[0, 1]$, one with $\alpha = 2/3$ and one with $\alpha = 1$. In fact, even for convex cells, there are in general two choices for α and β corresponding to a given point in the cell, although only one with values in the range $[0, 1]$. This is illustrated in figure 4 (right) where the point on the intersection of $\beta = 1/2$ and $\beta = 2$ can be represented with either value of β. We will find below that these two solutions arise from a quadratic equation.

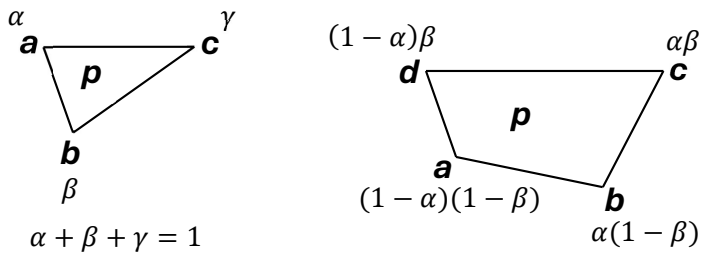


Figure 3: Illustration of the notation for interpolation to the point $\mathbf{p}$ for triangular and quadrilateral cells on a sphere or plane. In the spherical case the straight lines represent great circle arcs. The vertices and weights are indicated by roman and greek letters.

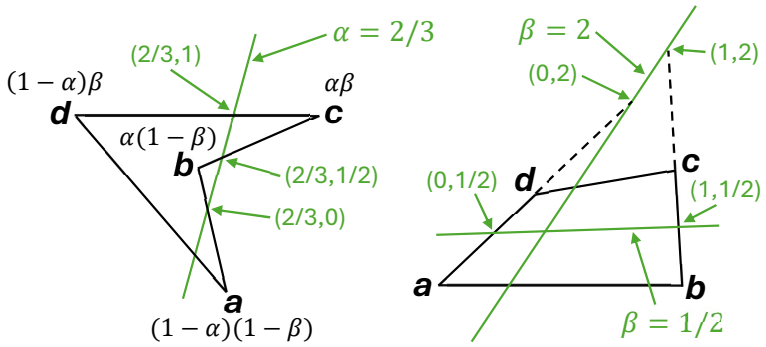


Figure 4: Some examples of the geometry of quadrilateral cells which are discussed in detail in the text. The left diagram shows a problematic non-convex example and the right diagram a convex example. The weights applied to $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$ are shown in the left hand diagram. Various points are labelled with their (α, β) coordinates.

For triangular cells we need

$$\alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c} \propto \mathbf{p}. \tag{1}$$

We will use the subscript h for the part of a vector orthogonal to $\mathbf{p}$. Then

$$\alpha\mathbf{a}_h + \beta\mathbf{b}_h + \gamma\mathbf{c}_h = 0.$$

Hence $\alpha\mathbf{a}_h + \beta\mathbf{b}_h$ and $\mathbf{c}_h$ are linearly dependent which implies $(\alpha\mathbf{a}_h + \beta\mathbf{b}_h) \times \mathbf{c}_h = 0$ and so

$$[(\alpha\mathbf{a} + \beta\mathbf{b}) \times \mathbf{c}].\mathbf{p} = 0.$$

[Alternatively we can derive this by taking the triple product of (1) with $\mathbf{c}$ and $\mathbf{p}$.] It follows that

$$\alpha(\mathbf{c} \times \mathbf{a}).\mathbf{p} = \beta(\mathbf{b} \times \mathbf{c}).\mathbf{p}$$

with similar results obtained by permuting $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ and α, β and γ appropriately. Using $\alpha + \beta + \gamma = 1$ we can deduce that

$$(\alpha, \beta, \gamma) = \frac{((\mathbf{b} \times \mathbf{c}).\mathbf{p}, (\mathbf{c} \times \mathbf{a}).\mathbf{p}, (\mathbf{a} \times \mathbf{b}).\mathbf{p})}{(\mathbf{a} \times \mathbf{b}).\mathbf{p} + (\mathbf{b} \times \mathbf{c}).\mathbf{p} + (\mathbf{c} \times \mathbf{a}).\mathbf{p}}.$$

For quadrilateral cells we need

$$((1-\alpha)\mathbf{a} + \alpha\mathbf{b})\,(1-\beta) + ((1-\alpha)\mathbf{d} + \alpha\mathbf{c})\,\beta \propto \mathbf{p}. \tag{2}$$

Then

$$((1-\alpha)\mathbf{a}_h + \alpha\mathbf{b}_h)(1-\beta) + ((1-\alpha)\mathbf{d}_h + \alpha\mathbf{c}_h)\beta = 0.$$

For this to have a solution for β we need $(1-\alpha)\mathbf{a}_h + \alpha\mathbf{b}_h$ and $(1-\alpha)\mathbf{d}_h + \alpha\mathbf{c}_h$ to be linearly dependent which implies $((1-\alpha)\mathbf{a}_h + \alpha\mathbf{b}_h) \times ((1-\alpha)\mathbf{d}_h + \alpha\mathbf{c}_h) = 0$ and so

$$[((1-\alpha)\mathbf{a} + \alpha\mathbf{b}) \times ((1-\alpha)\mathbf{d} + \alpha\mathbf{c})].\mathbf{p} = 0$$

which is a quadratic equation for α. [Alternatively we can derive this by taking the triple product of (2) with $(1-\alpha)\mathbf{a} + \alpha\mathbf{b}$ and $\mathbf{p}$ and with $(1-\alpha)\mathbf{d} + \alpha\mathbf{c}$ and $\mathbf{p}$ and adding the results.] However, instead of solving this quadratic equation, it is simpler to solve and easier to see which of the two solutions we want if we first convert it into a quadratic equation for $\hat{\alpha} = \frac{1-\alpha}{\alpha}$ (see Floater (2015, section 3) for the similar planar case). This leads to

$$[(\hat{\alpha}\mathbf{a} + \mathbf{b}) \times (\hat{\alpha}\mathbf{d} + \mathbf{c})].\mathbf{p} = 0$$

i.e.

$$(\mathbf{b} \times \mathbf{c}).\mathbf{p} + \hat{\alpha}((\mathbf{a} \times \mathbf{c}).\mathbf{p} + (\mathbf{b} \times \mathbf{d}).\mathbf{p}) + \hat{\alpha}^2(\mathbf{a} \times \mathbf{d}).\mathbf{p} = 0$$

with solution

$$\hat{\alpha} = \frac{(\mathbf{a} \times \mathbf{c}).\mathbf{p} + (\mathbf{b} \times \mathbf{d}).\mathbf{p} + \sqrt{D}}{2(\mathbf{d} \times \mathbf{a}).\mathbf{p}}.$$

Here $D = [(\mathbf{a} \times \mathbf{c}).\mathbf{p} + (\mathbf{b} \times \mathbf{d}).\mathbf{p}]^2 + 4[(\mathbf{b} \times \mathbf{c}).\mathbf{p}][(\mathbf{d} \times \mathbf{a}).\mathbf{p}]$ and, by noting that $(\mathbf{b} \times \mathbf{c}).\mathbf{p} \geq 0$ and $(\mathbf{d} \times \mathbf{a}).\mathbf{p} \geq 0$ and that we want $0 \leq \hat{\alpha} \leq +\infty$, we have determined which of the two solutions of the quadratic equation is required. The expression for D can be expressed more symmetrically as

$$D = [(\mathbf{a} \times \mathbf{c}).\mathbf{p}]^2 + [(\mathbf{b} \times \mathbf{d}).\mathbf{p}]^2 + 2[(\mathbf{a} \times \mathbf{b}).\mathbf{p}][(\mathbf{c} \times \mathbf{d}).\mathbf{p}] + 2[(\mathbf{b} \times \mathbf{c}).\mathbf{p}][(\mathbf{d} \times \mathbf{a}).\mathbf{p}]$$

by using

$$[(\mathbf{a} \times \mathbf{b}).\mathbf{p}][(\mathbf{c} \times \mathbf{d}).\mathbf{p}] + [(\mathbf{a} \times \mathbf{c}).\mathbf{p}][(\mathbf{d} \times \mathbf{b}).\mathbf{p}] + [(\mathbf{a} \times \mathbf{d}).\mathbf{p}][(\mathbf{b} \times \mathbf{c}).\mathbf{p}] = 0 \tag{3}$$

which is an extension of a result presented by Floater (2015, section 3) for the planar case. (3) can be derived in various ways: we can choose coordinates with $\mathbf{p} = (1, 0, 0)$ and evaluate the l.h.s. of (3) in terms of the components of $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$; or we can note that, because we are in three dimensions, $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$ can't be linearly independent, and so one can be expressed as a linear combination of the others, say $\mathbf{d} = A\mathbf{a} + B\mathbf{b} + C\mathbf{c}$, and then we can substitute this into the l.h.s. of (3) and show that the contributions due to each of A, B and C are zero. (3) is also equivalent to

$$[(\mathbf{b} \times \mathbf{c}).\mathbf{p}]\mathbf{a} + [(\mathbf{c} \times \mathbf{a}).\mathbf{p}]\mathbf{b} + [(\mathbf{a} \times \mathbf{b}).\mathbf{p}]\mathbf{c} \propto \mathbf{p} \tag{4}$$

while (4) can be deduced from the analysis of the triangular case above. To show the equivalence of (3) and (4), we note that taking the triple product of (4) with $\mathbf{d}$ and $\mathbf{p}$ yields (3) and, conversely, (3) implies that taking the triple product of the left-hand side of (4) with $\mathbf{d}$ and $\mathbf{p}$ is zero (where $\mathbf{d}$ is not involved in the left-hand side of (4) and can be any vector) which implies that the left-hand side of (4) is proportional to $\mathbf{p}$. The expression for $\hat{\alpha}$ can be rearranged to give

$$\alpha = \frac{2(\mathbf{d} \times \mathbf{a}).\mathbf{p}}{2(\mathbf{d} \times \mathbf{a}).\mathbf{p} + (\mathbf{a} \times \mathbf{c}).\mathbf{p} + (\mathbf{b} \times \mathbf{d}).\mathbf{p} + \sqrt{D}}.$$

Similarly we have

$$1-\alpha = \frac{2(\mathbf{b}\times\mathbf{c}).\mathbf{p}}{2(\mathbf{b}\times\mathbf{c}).\mathbf{p}+(\mathbf{c}\times\mathbf{a}).\mathbf{p}+(\mathbf{d}\times\mathbf{b}).\mathbf{p}+\sqrt{D}},$$

$$\beta = \frac{2(\mathbf{a}\times\mathbf{b}).\mathbf{p}}{2(\mathbf{a}\times\mathbf{b}).\mathbf{p}+(\mathbf{c}\times\mathbf{a}).\mathbf{p}+(\mathbf{b}\times\mathbf{d}).\mathbf{p}+\sqrt{D}}$$

and

$$1-\beta = \frac{2(\mathbf{c}\times\mathbf{d}).\mathbf{p}}{2(\mathbf{c}\times\mathbf{d}).\mathbf{p}+(\mathbf{a}\times\mathbf{c}).\mathbf{p}+(\mathbf{d}\times\mathbf{b}).\mathbf{p}+\sqrt{D}}.$$

We note that the signs of the triple products (or of $\sqrt{D}$) need changing if the vertices $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$ are labelled clockwise instead of anticlockwise as in figure 3. Of course we can also calculate $1-\alpha$ and $1-\beta$ from α and β, although with different numerical error characteristics. Using alternative calculation methods (including the different expressions for D) provides some possibilities for testing some aspects of the code. As a check we note that $\alpha+(1-\alpha)=1$ is equivalent to

$$\frac{2(\mathbf{d}\times\mathbf{a}).\mathbf{p}}{2(\mathbf{d}\times\mathbf{a}).\mathbf{p}+(\mathbf{a}\times\mathbf{c}).\mathbf{p}+(\mathbf{b}\times\mathbf{d}).\mathbf{p}+\sqrt{D}}+\frac{2(\mathbf{b}\times\mathbf{c}).\mathbf{p}}{2(\mathbf{b}\times\mathbf{c}).\mathbf{p}+(\mathbf{c}\times\mathbf{a}).\mathbf{p}+(\mathbf{d}\times\mathbf{b}).\mathbf{p}+\sqrt{D}}=1$$

i.e.

$$2(\mathbf{d}\times\mathbf{a}).\mathbf{p}[2(\mathbf{b}\times\mathbf{c}).\mathbf{p}+(\mathbf{c}\times\mathbf{a}).\mathbf{p}+(\mathbf{d}\times\mathbf{b}).\mathbf{p}+\sqrt{D}]+2(\mathbf{b}\times\mathbf{c}).\mathbf{p}[2(\mathbf{d}\times\mathbf{a}).\mathbf{p}+(\mathbf{a}\times\mathbf{c}).\mathbf{p}+(\mathbf{b}\times\mathbf{d}).\mathbf{p}+\sqrt{D}]$$

$$=[2(\mathbf{d}\times\mathbf{a}).\mathbf{p}+(\mathbf{a}\times\mathbf{c}).\mathbf{p}+(\mathbf{b}\times\mathbf{d}).\mathbf{p}+\sqrt{D}][2(\mathbf{b}\times\mathbf{c}).\mathbf{p}+(\mathbf{c}\times\mathbf{a}).\mathbf{p}+(\mathbf{d}\times\mathbf{b}).\mathbf{p}+\sqrt{D}]$$

i.e.

$$4(\mathbf{d}\times\mathbf{a}).\mathbf{p}(\mathbf{b}\times\mathbf{c}).\mathbf{p}=[(\mathbf{a}\times\mathbf{c}).\mathbf{p}+(\mathbf{b}\times\mathbf{d}).\mathbf{p}+\sqrt{D}][(\mathbf{c}\times\mathbf{a}).\mathbf{p}+(\mathbf{d}\times\mathbf{b}).\mathbf{p}+\sqrt{D}]$$

i.e.

$$4(\mathbf{d}\times\mathbf{a}).\mathbf{p}(\mathbf{b}\times\mathbf{c}).\mathbf{p}=-[(\mathbf{a}\times\mathbf{c}).\mathbf{p}+(\mathbf{b}\times\mathbf{d}).\mathbf{p}]^2+D$$

which is true from the original expression for D given above.

We now consider the planar case. Much of the discussion and analysis of the spherical case above applies here too with some straightforward modifications. In the planar case we choose weights so that the centroid of the vertices in the projection plane coincides with the point of interest. The hemispheric restriction becomes the simple condition that the cell must lie entirely within the projection plane. The cell edges are straight lines. Position vectors $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{d}$ and $\mathbf{p}$ now refer to positions in the projection plane and we do not need to consider the choice of origin for position vectors as we will only consider differences between position vectors. The convexity condition now refers to convexity in the projection plane. Where we had $\mathbf{a}_h$ etc for the spherical case, we now have $\mathbf{a}'=\mathbf{a}-\mathbf{p}$ and, where we had vector triple products such as $(\mathbf{a}\times\mathbf{b}).\mathbf{p}$ etc, we now have $(\mathbf{a}'\times\mathbf{b}').\hat{\mathbf{z}}$ with $\hat{\mathbf{z}}$ a unit vector orthogonal to the plane. For triangular cells the weights are given by

$$(\alpha,\beta,\gamma)=\frac{((\mathbf{b}'\times\mathbf{c}').\hat{\mathbf{z}},(\mathbf{c}'\times\mathbf{a}').\hat{\mathbf{z}},(\mathbf{a}'\times\mathbf{b}').\hat{\mathbf{z}})}{(\mathbf{a}'\times\mathbf{b}').\hat{\mathbf{z}}+(\mathbf{b}'\times\mathbf{c}').\hat{\mathbf{z}}+(\mathbf{c}'\times\mathbf{a}').\hat{\mathbf{z}}}$$

etc. and for quadilateral cells we have

$$\alpha = \frac{2(\mathbf{d}'\times\mathbf{a}').\hat{\mathbf{z}}}{2(\mathbf{d}'\times\mathbf{a}').\hat{\mathbf{z}}+(\mathbf{a}'\times\mathbf{c}').\hat{\mathbf{z}}+(\mathbf{b}'\times\mathbf{d}').\hat{\mathbf{z}}+\sqrt{D}},$$

$$1-\alpha = \frac{2(\mathbf{b}'\times\mathbf{c}').\hat{\mathbf{z}}}{2(\mathbf{b}'\times\mathbf{c}').\hat{\mathbf{z}}+(\mathbf{c}'\times\mathbf{a}').\hat{\mathbf{z}}+(\mathbf{d}'\times\mathbf{b}').\hat{\mathbf{z}}+\sqrt{D}},$$

$$\beta = \frac{2(\mathbf{a}'\times\mathbf{b}').\hat{\mathbf{z}}}{2(\mathbf{a}'\times\mathbf{b}').\hat{\mathbf{z}}+(\mathbf{c}'\times\mathbf{a}').\hat{\mathbf{z}}+(\mathbf{b}'\times\mathbf{d}').\hat{\mathbf{z}}+\sqrt{D}},$$

and

$$1-\beta = \frac{2(\mathbf{c}'\times\mathbf{d}').\hat{\mathbf{z}}}{2(\mathbf{c}'\times\mathbf{d}').\hat{\mathbf{z}}+(\mathbf{a}'\times\mathbf{c}').\hat{\mathbf{z}}+(\mathbf{d}'\times\mathbf{b}').\hat{\mathbf{z}}+\sqrt{D}}$$

with

$$D=[(\mathbf{a}'\times\mathbf{c}').\hat{\mathbf{z}}]^2+[(\mathbf{b}'\times\mathbf{d}').\hat{\mathbf{z}}]^2+2[(\mathbf{a}'\times\mathbf{b}').\hat{\mathbf{z}}][(\mathbf{c}'\times\mathbf{d}').\hat{\mathbf{z}}]+2[(\mathbf{b}'\times\mathbf{c}').\hat{\mathbf{z}}][(\mathbf{d}'\times\mathbf{a}').\hat{\mathbf{z}}].$$

3.1.3 Interpolation of horizontal velocities

There are extra complications in interpolating the horizontal part of the velocity because it is a vector defined in the local tangent plane of the sphere. (There is no such complication for vertical velocity which we treat as a scalar.)

The simplest approach is to just interpolate the easterly component u and the northerly component v using the same weights as for scalars. Here 'easterly' and 'northerly' might be defined relative to a rotated longitude-latitude coordinate system. This interpolation method is suitable if the data is held on an (x, y)-structured grid defined using the same longitude-latitude system. This is because, near the coordinate system's poles where the direction of east and north varies rapidly with position, the grid also becomes very fine in the east-west direction.

An alternative method which is suitable for more general grids is to transport the velocity vectors from the grid points to the point of interest using parallel transport along great circles and then combine them using the weights (Lomax et al., 2024). Parallel transport, also known as parallel displacement or parallel translation, is a general concept for moving a tangent vector along a curve on a Riemannian manifold, often so that it can be compared with a vector defined at another point (see e.g. Frankel (2004)). Without transporting the vector it can't be directly compared in an intrinsic coordinate-free way as it is in a different tangent space. Parallel transport along geodesics is a special case of this and can be thought of as producing a tangent vector at the new point which is in some sense as most like the vector at the old point as possible. In our case this corresponds to parallel transport along great circles and the resulting vector at the new point has the same magnitude and makes the same angle with the great circle as the vector at the old point.

Suppose we know the wind vector at the "source" point $\mathbf{S}$ and want to transport the vector to a point of interest $\mathbf{P}$ as illustrated in figure 5. The angles θ_s and θ between the geodesic $\mathbf{SP}$ and north at $\mathbf{S}$ and $\mathbf{P}$, measured from the geodesic to north with anticlockwise angles positive, can be expressed using spherical triangle formulae as

$$\theta_s = \arctan_2(\cos\phi\ \sin(\lambda - \lambda_s),\ \sin\phi\ \cos\phi_s - \cos\phi\ \sin\phi_s\ \cos(\lambda - \lambda_s))$$

and

$$\theta = \arctan_2(\cos\phi_s\ \sin(\lambda - \lambda_s),\ -\sin\phi_s\ \cos\phi + \cos\phi_s\ \sin\phi\ \cos(\lambda - \lambda_s)).$$

We then need to "rotate" the wind vector by $\Delta\theta = \theta_s - \theta$ using

$$\begin{pmatrix} u_p \\ v_p \end{pmatrix} = \begin{pmatrix} \cos\Delta\theta & -\sin\Delta\theta \\ \sin\Delta\theta & \cos\Delta\theta \end{pmatrix} \begin{pmatrix} u_s \\ v_s \end{pmatrix}$$

where (u_s, v_s) and (u_p, v_p) are the east-north components of the vector at $\mathbf{S}$ and $\mathbf{P}$. "Rotate" is in quotes because this only looks like a rotation due to our choice of coordinate system. "Unrotate" might be a bit more accurate – if we use east-north components defined in a rotated longitude-latitude coordinate system with the equator running through $\mathbf{S}$ and $\mathbf{P}$, then there is no need for this rotation.

3.2 Implementation

3.2.1 Interpolation of data on (x,y)-structured grids

For (x, y)-structured grids we only support interpolating data held on the central grid points within each cell.[5] For points outside the data area (i.e. outside the rectangle of cell centres plotted in the grid's (x, y) coordinate system) we calculate a value interpolated to the nearest point on the edge of the data area (with nearest defined in terms of distance on the (x, y)-plane). Like all the other data, horizontal velocity components must be held on the central grid points of their grid, but can be held on separate grids, and must be expressed as components along the x and y directions.

Interpolation is carried out using bilinear interpolation and horizontal velocities are interpolated using bilinear interpolation of the velocity components u and v in the x and y directions.

[5] Currently, to interpolate data held on the cell boundaries, we need to use a separate staggered grid.

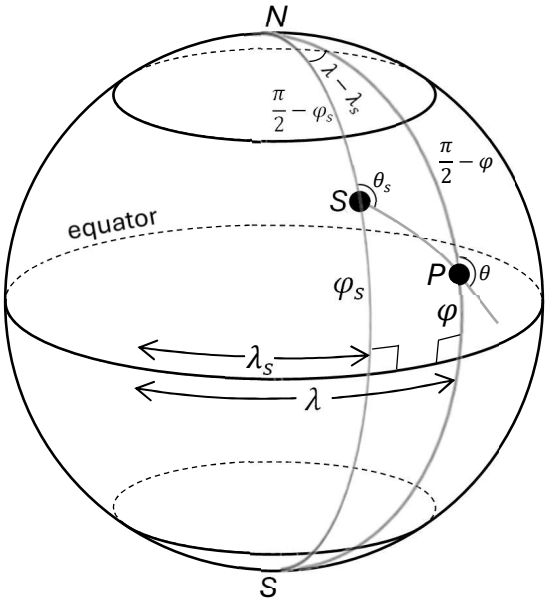


Figure 5: Illustration of the spherical triangle used in the parallel transport calculation. $\mathbf{S}$ is the "source" point at which we know the wind vector and $\mathbf{P}$ is the position of interest where we want to estimate the wind vector, with $\mathbf{SP}$ being the great circle along which we want to transport the vector. λ_s and ϕ_s are the longitude and latitude of $\mathbf{S}$ and λ and ϕ are the longitude and latitude of $\mathbf{P}$. θ_s and θ are the angles between the great circle $\mathbf{SP}$ and north at $\mathbf{S}$ and $\mathbf{P}$.

3.2.2 Interpolation of data on cubed-sphere grids

For cubed-sphere grids we only support interpolating data held on the central grid points within each cell.[6] Horizontal velocities must be held as east and north components on the central grid points within each cell, where east and north refer to the underlying longitude-latitude system used to define the cubed-sphere grid.[7]

Our main interpolation method is spherical barycentric interpolation with parallel transport of the horizontal velocities. However we also have the option of planar barycentric interpolation on the $(f(\lambda_1), f(\lambda_2))$-plane of each panel, again with parallel transport, with some minor modifications for cases other than the equiangular cubed sphere in order to ensure continuity. These modifications are described below.

The spherical barycentric method will be described first. To interpolate data held on the grid points we need to consider the "dual grid" consisting of new cells formed by joining the points on which the data is held whenever these are in cells with a common edge. The data points are at the vertices of these new cells. To do this we define the cell edges for the dual grid to be great circle arcs. In the interior of a panel these are straight lines on the $(f(\lambda_1), f(\lambda_2))$-plane aligned with the panel coordinates. However the lines joining two points on different panels are more complicated. This is illustrated in figure 6. To interpolate the data to a point of interest we need to decide in which cell of the dual grid the point of interest lies. Points in the interior of the panel are easily allocated to dual cells using the panel coordinates. For points near the panel edge we also initially (and sometimes incorrectly) allocate the points to cells by assuming that the dual cell boundaries follow panel coordinate lines. We then correct for this by testing whether the point is the wrong side of one of the cell's edges (there is only one edge that needs considering) and, if so, we assign the point to the neighbouring cell. For example, the red area in figure 6 shows an area where points would be allocated to cell ABED when they should be in BCFE. We test for this by checking on which side of the great circle arc BE the point lies and we do this by looking at the sign of the triple product of the position vectors (relative to the centre of the sphere) of B, E and the point of interest.

We assume that, if the test finds that the point is not in the first guess cell, then assigning the point to the adjacent cell is always sufficient to correct the situation. Provided we don't have a situation such as the

[6] Because cubed-sphere grids are global the issue of estimating values outside the grid area doesn't arise. However it is likely that in future we will allow 'masked cubed spheres' in connection with splitting LFRic global data into regional files. In this situation we will not allow the estimation of values outside the data area (i.e. outside the union of the cells in the dual grid formed from the unmasked points).

[7] There may be value in future in allowing horizontal velocity components to be stored on cell edges as this is how velocity is handled internally in LFRic. However standard LFRic met data output has velocity stored at cell centres.

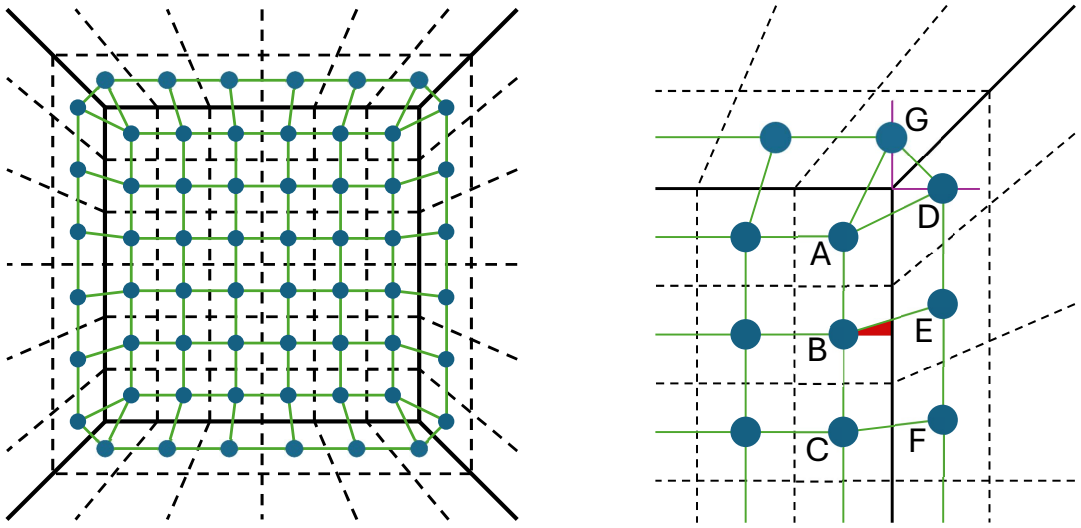


Figure 6: Illustration of a cubed-sphere panel showing its cells together with adjacent cells on neighbouring panels, viewed on the $(f(\lambda_1), f(\lambda_2))$-coordinate plane. The central grid points of each cell are shown in blue with the cell edges in black and with the cell edges of the dual grid in green. The solid black lines are the cell edges which are also panel edges. The dotted cell edges outside the panel which intersect our panel's boundary and the dual cell edges which cross between panels are drawn as straight lines, but this is only accurate for the gnomonic cubed sphere. (The other lines shown as straight are indeed straight.) A zoomed-in image of the top right corner is shown, labelled with some features that are discussed in the text.

bottom right-hand point of the red triangle crossing the great circle arc CF, this is a correct assumption. The assumption seems likely to be correct for the cubed spheres of interest but is not correct for cubed spheres with arbitrary coordinate mappings f. A strongly stretching form of f could produce very elongated cells near the panel edges which could violate this assumption. It seems likely that the assumption is correct for gnomonic, equiangular and equidistant cubed spheres, but we have not attempted to demonstrate this. Instead we test any cubed spheres within NAME to check that points like the bottom right-hand corner of the red triangle are the right side of the relevant great circle (using the vector triple product test).

Having established which cell of the dual grid contains the point of interest, we apply the spherical barycentric interpolation formulae as described in section 3.1.2 to calculate the weights. The two required conditions, namely that the cells can be contained in one hemisphere and that quadrilateral cells are convex, are automatically satisfied for cubed-sphere dual grids. The 'rotation' of the velocity vectors needed for parallel transport is also applied.

We note that if one continues a great circle arc making up a panel edge beyond the end of the panel, then, by the symmetry of the cubed sphere, the continuation runs diagonally across the adjacent panel. This means, in particular, that the extension of the panel edge runs through the central grid point of the corner cell of the adjacent panel, as illustrated by the purple lines in figure 6. When viewed in panel coordinates as in the figure, this seems an unexpected coincidence. The extension should also of course go through the opposite corner of the cell. The sketch in the figure is deficient in this regard, mainly because fixing it would require altering the sketch in different ways for different types of cubed sphere. For example in the gnomonic case the vertical lines in the adjacent panel to the right should become further apart as one moves to the right (with in fact the centre line of the panel at infinity). This would enable the "opposite corner" to lie on the purple line. However it would be misleading for the equiangular case. Here the vertical lines remain equidistant but the cell edges running left-right (in the upper part of the adjacent panel) should curve towards the top edge of the panel and meet it when the centre line of the panel is reached. These are mostly curiosities which are mentioned just in case the figure is misleading. However the fact that the extension of the panel edge runs through the central grid point of the corner cell of the adjacent panel will be used below.

We now consider the implementation of planar barycentric interpolation on the $(f(\lambda_1), f(\lambda_2))$-plane of each panel.[8] The approach follows the spherical barycentric interpolation method above with the following differences. The dual cell boundaries are defined as straight lines on the $(f(\lambda_1), f(\lambda_2))$-plane, vectors now refer

[8]This is not yet supported in NAME but it would be useful to enable this in order to compare results with our main approach.

to position on the $(f(\lambda_1), f(\lambda_2))$-plane, and, in the notation of section 3.1.2, vector triple products like $(\mathbf{a} \times \mathbf{b}).\mathbf{p}$ are replaced with products like $(\mathbf{a}' \times \mathbf{b}').\hat{\mathbf{z}}$ in testing which cell the point of interest is in and in calculating the weights.

However there is an additional complication. For spherical barycentric interpolation it is straightforward to show that the interpolated field is continuous. In contrast, for the planar case, we need to investigate whether there is continuity as the point moves from one panel to another, i.e. whether we get the same result for a point on the panel edge whether we regard the point as being in one panel or the other. We will consider this for the right hand edge of the panel shown in figure 6. We can think of the interpolation as involving a two step process. First we consider the set of points consisting of the points on the panel edge where the green lines in the figure cross the edge and the points at the ends of the edge. This set of points is the same by symmetry no matter which of the two adjacent panels we are considering. We interpolate to the two points in the set immediately above and below our point of interest. Then we linearly interpolate between these two points along the edge. This has the same effect as doing the barycentric interpolation in one step, although this is only true because the panel edge and the two rows of points either side are parallel on the $(f(\lambda_1), f(\lambda_2))$-plane. Otherwise, with this two step interpolation method, the implied weights for the quadrilateral cells would not take the required form shown in figure 3. The interpolation along the edge will have the same effect no matter which panel is considered, so we only need to investigate whether the first step gives results at our set of points which are also the same no matter which panel is considered. This is so for the equiangular cubed sphere because, for this case, the panel edge and the two rows of points either side are equally spaced on the $(f(\lambda_1), f(\lambda_2))$-plane. For example the value at the point on the edge between B and E in figure 6 will be the average of the values at B and E and will be the same no matter which panel is considered. Also, for the point at the end of the edge, points A and D have equal weight. To see this we need the fact that G lies on the upward extension of the edge. Similarly A and G have equal weight and so the result is the average of the values at A, D and G. This doesn't work for other types of cubed-sphere grid but we correct this by moving the line of points outside the panel in a direction perpendicular to the panel edge (on the $(f(\lambda_1), f(\lambda_2))$-plane) so that the panel edge and the two rows of points either side are equally spaced. This is equivalent to working not on the $(f(\lambda_1), f(\lambda_2))$-plane but on the $(g(\lambda_1), g(\lambda_2))$-plane where

$$g(\lambda) = \begin{cases} f(\lambda + \pi/2) - 2f(\pi/4) & \text{if } \lambda \leq -\pi/4 \\ f(\lambda) & \text{if } -\pi/4 \leq \lambda \leq \pi/4 \\ f(\lambda - \pi/2) + 2f(\pi/4) & \text{if } \pi/4 \leq \lambda. \end{cases}$$

3.2.3 Interpolation of data on unstructured connected grids

Currently data on unstructured connected grids can only be interpolated if a structured connected grid is mapped to it as described in section 2.3. The interpolation is then carried out on the structured grid.[9]

4 Vertical grids

To be written.

5 Temporal grids

To be written.

6 Data grids

To be written.

[9] In future it is possible we may want to work directly with data on unstructured connected grids. Various approaches are possible for locating a point of interest in such a grid. If the resolution does not vary strongly one could use a structured grid which is different from the unstructured grid but has similar resolution, and store a list of overlapping unstructured grid cells with each structured grid cell. There are also approaches that are possible using the k-d tree algorithm or the area-filling Hilbert curve described in section 2.2.

References

Auer, B., 2021, CS description. https://gmao.gsfc.nasa.gov/gmaoftp/ops/GEOSIT_sample/doc/CS_Description_c180_v1.pdf. 3

Floater, M. S., 2015, Generalized barycentric coordinates and applications, Acta Numerica **24**, 161-214. 7, 8

Frankel, T., 2004, The geometry of physics, 2nd edition, Cambridge University Press. 10

Lomax, O. S., 2021, Atlas-based H(x) interpolation, unpublished report. 7

Lomax, O., Thomson D. and Deconinck W. 2024, Accurate vector field interpolation in NG-PAO and beyond, Science Newsletter May 2024, Met Office. 10

Ronchi, C., Iacono, R., and Paolucci P. S., 1996, The “cubed sphere”: a new method for the solution of partial differential equations in spherical geometry, J. Comp. Phys. **124**, 93-114. 3

Sadourny, R., 1972, Conservative finite-difference approximations of the primitive equations on quasi-uniform spherical grids, Mon. Wea. Rev. **100**, 136-144. 2