# Integration of the YAC regridding library into the OASIS3-MCT coupler


G. Jonville[1], A. Piacentini[1], S. Valcke[1], M. Hanke[2]

[1] CECI, Université de Toulouse, CNRS, CERFACS, Toulouse, France,
[2] DKRZ, Deutsches Klimarechenzentrum, Hamburg, Germany



TR-CMGC-24-182 Technical Report

December 2024


CERFACS

# 1. Introduction

YAC (Yet Another Coupler) is a coupling software for Earth system modelling developed by the German Climate Computing Center DKRZ (Deutsches Klimarechenzentrum).

YAC regridding code has been introduced in the OASIS coupler as a regridding library in addition to the SCRIP library. This new regridder is called OYAC in this report.

In Section 2, this report presents the interpolation methods proposed by YAC and available in OASIS, and explains the concept of the interpolation stack in YAC.

Section 3 details the integration of YAC into OASIS.

Section 4 shows how to create a weight file with OYAC and how to visualise weights and cells.

Section 5 presents an update of the qualitative comparisons of the regridding libraries produced in the 2021 benchmark (see Valcke et al. 2021). The focus here is on the evaluation of OYAC.

Finally, two tables in Appendix I and II present the number of residual destination cells after each method in several YAC interpolation stacks, involving different ocean-atmosphere grid pairs.

# 2. YAC interpolation

Interpolations in YAC are all in 2D on the sphere. If a mask is defined for a grid in OASIS auxiliary file masks.nc, it will be taken into account in the interpolation[1].

An interpolation consists of an interpolation stack which in turn is comprised of one or more interpolation methods (see Sections 2.1 and 2.2 for a more detailed description).

Interpolations are configured in the OASIS "*namcouple*" file.

The creation of weight files by YAC requires the description of grid points and corners (these last are the vertices defining a cell around each point) in OASIS auxiliary file grids.nc (see Section 5.1 of OASIS3-MCT User Guide *OASIS3-MCT_5.0*), whatever the interpolation method[2].

[1] To set up a consistent atmosphere-ocean system and have a well-posed coupled problem, we adopted the following best practice. The original sea-land mask of the ocean model is taken as is. For the atmosphere model, the fraction of water in each cell is defined by the conservative remapping of the ocean mask on the atmospheric grid performed with the specific regridder used in the current test. Then, the atmospheric coupling mask is adapted associating a valid/active index to cells containing at least a surface fraction (1/1000) of water; under 1/1000 of water, the atmospheric cell is considered completely masked. This method ensures that the total sea and land surfaces are the same in the ocean and atmosphere models, allowing global conservation of sea or land integrated quantities. The resulting mask and fractions are stored in file masks.nc.

[2] The requirement to define the corners for YAC interpolation will be lifted in OASIS3-MCT_6.0.

## 2.1. YAC interpolation methods

### 2.1.1. N-Nearest-Neighbour method (NNN)

For each target point, this method searches for the "n" nearest unmasked source points within a prescribed distance. Using the weighting method selected by the user (see *weighting* below), the interpolation stencil is generated from the found source points.

Remark: Here distances between two points on the surface of a unit sphere are defined as the angle between two vectors that point from the center of the sphere to the two points. Therefore, in the *namcouple* configuration file, parameters expressing distances, i.e. *search_distance* or *spread* (see Section 2.1.8) have to be provided in degrees.

NNN method is invoked in the OASIS *namcouple* by the following line and options:

```
NNN weighting n search_distance scale
```

- *weighting* `[string]`:
    - Arithmetic average `AVG`
    - Inverse distance weighting `DIST`
    - Gaussian filter `GAUSS`
      This method calculates weights by applying a Gauss function to the distance, with an additional optional *scale* parameter, see below.
    - Radial Basis Function `RBF`
      This method has also an additional optional *scale* parameter, see below. NNN with RBF weighting is strictly equivalent to RBF method detailed in Section 2.1.3.
    - Zero weight `ZERO`
      Strictly equivalent to the ZERO method (see Section 2.1.2). Sets the weight to zero (number of source points per target is necessarily `1`) and therefore sets target point values to `0.0`.
- *n* `[integer]`: Number of source points per target point (mandatory, except for the Zero weight)
- *search_distance* `[real]`: Maximum search distance in degree (optional, default `0.0`). Note: a value of `0.0` results in an unlimited search distance. Valid range is `[0.0, 180.0[`
  Warning: If there are less than the specified number of source points within the maximum search distance, then the respective target point will not be interpolated by this method.
- *scale* `[real]`: Scaling factor; optional, only for GAUSS and RBF weighting methods; empirical tests have led to default values of `0.1` and `1.487973e+01` respectively.

In the example below, we compare the impact of the weighting and the number of neighbours. The source grid is the coarser rectangular one and the target grid is the higher-resolution triangular one.

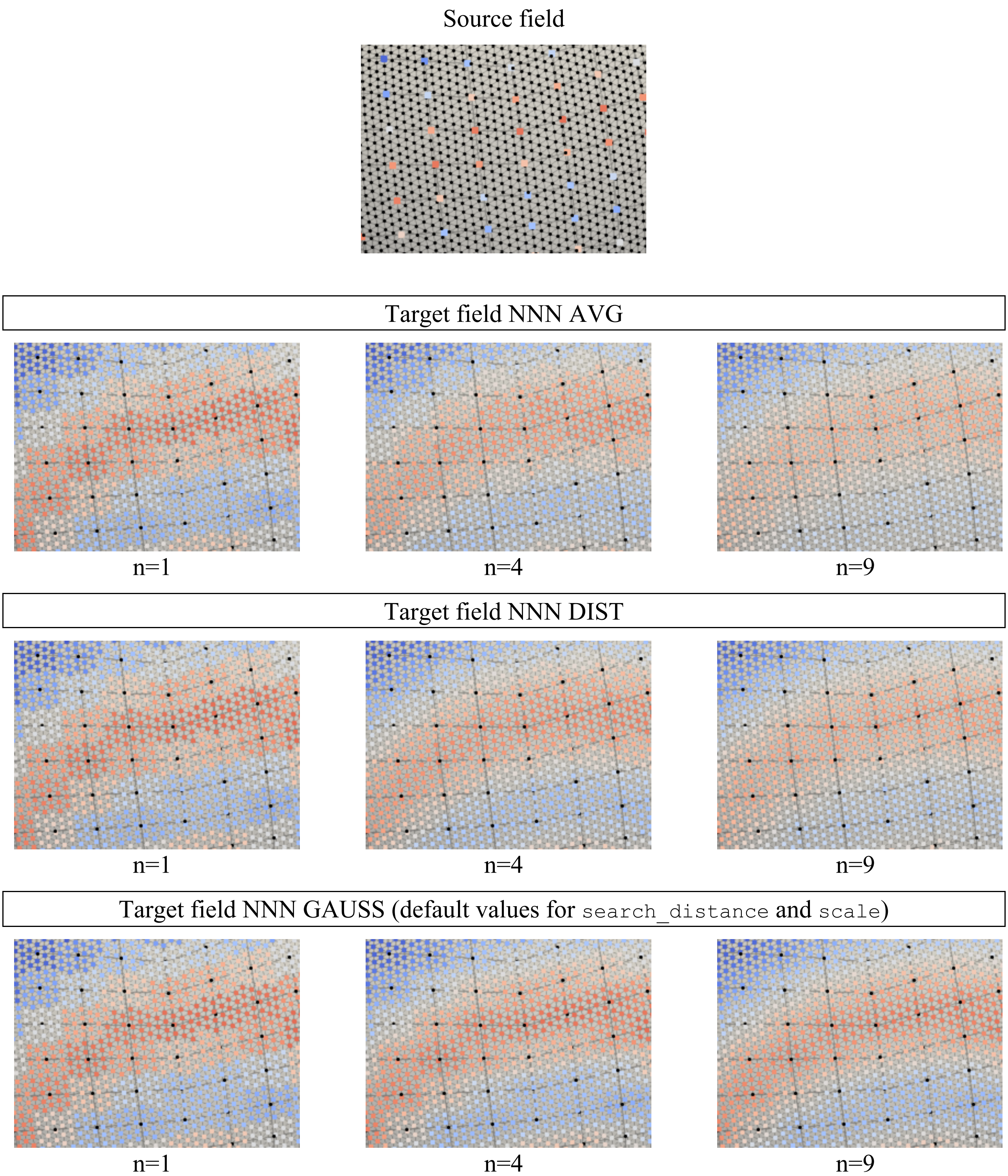



**Figure 1** – Source field and target fields of NNN interpolation for different weightings and numbers of neighbour (all weightings with n=1 give the same result)

### 2.1.2. ZERO method

This method is strictly equivalent to the N-Nearest-Neighbour method with Zero weighting. It sets the weight of one source neighbour to zero (therefore the address of the source neighbour is irrelevant). It can be used to set target point values to `0.0`.

Its description in the OASIS *namcouple* is reduced to the line:

```
ZERO
```

### 2.1.3. RBF method

This method is strictly equivalent to the N-Nearest-Neighbour method with RBF weighting. It is based on Reinheimer 2018.

For each target point, this method searches for the n nearest neighbours and then computes the weights based on the distances between the source points and the target point using gauss kernel radial basis functions. The search distance between the target and source points is not limited here.

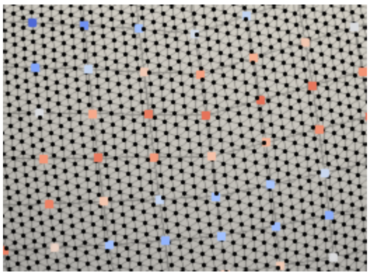
RBF method is invoked in the OASIS *namcouple* by the following line and options:

```
RBF n scale
```

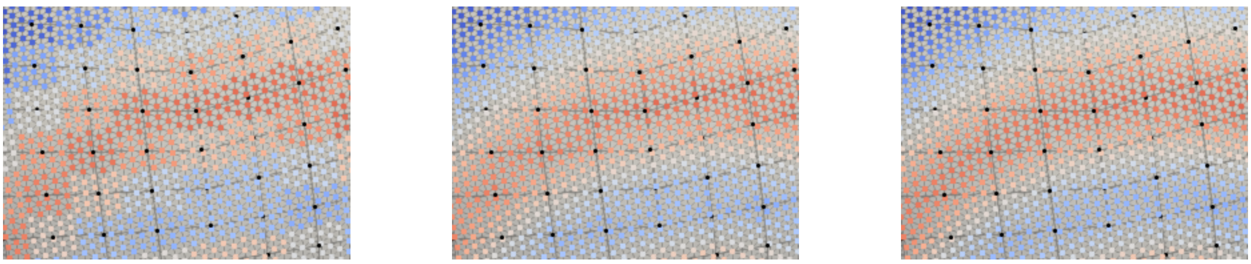
- *n* `[integer]`: Number of source points per target point
- *scale* `[real]`: Scaling factor; optional; empirical tests have led to a default value of `1.487973e+01`

In the following example we compare the impact of the number of neighbours.

Source field



Target field RBF



n=1 n=4 n=9

**Figure 2** – Target fields of RBF interpolation for different numbers of neighbours

#### 2.1.4. Average method (AVG)

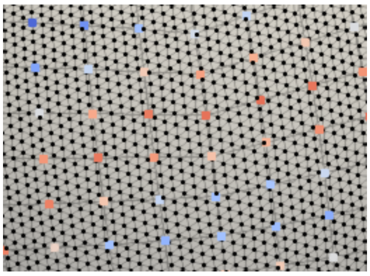
For this method, YAC generates cells based on the points of the original source grid. For each target point, this method searches for a matching source cell. It then applies the selected weighting type to compute the interpolation stencil for each target points using the non-masked corners of the matching source cell.

AVG method is invoked in the OASIS *namcouple* by the following line and option:

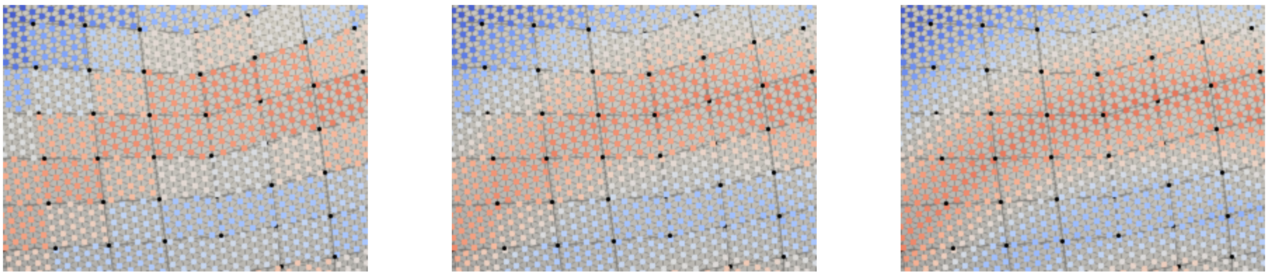
```
AVG weighting
```

- *`weighting`* `[string]`
  - Arithmetic average `ARITHMETIC`
  - Inverse distance weighting `DIST`
  - Barycentric coordinate `BARY`
    YAC first triangulates the source cell and then find the matching source triangle for the target point. The weights are then computed based on the barycentric coordinates of the target point within this triangle.

Source field



Target field AVG



ARITHMETIC DIST BARY

**Figure 3** – Target fields of AVG interpolation for different weightings

### 2.1.5. Nearest Corner Cells method (NCC)

For each target point, this method first searches for a matching source cell, as defined in the AVG method. Afterwards it determines the vertex of this cell that is the closest to the target point. And finally, all unmasked vertices of the source cells connected to this vertex are selected. The method then applies the selected weighting method to compute the interpolation stencil for all target points.

This method produces results that are very similar to AVG method. The major differences occur close to the mask edges, where the AVG method fails. This property makes the NCC method a good fallback option (e.g. for AVG method or Hybrid Cubic Spherical Bernstein-Bézier patch method).

NCC method is invoked in the OASIS *namcouple* by the following line and option:

```
NCC weighting
```

- `weighting` [string]
  - Arithmetic average `AVG`
  - Inverse distance weighting `DIST`

### 2.1.6. Hybrid Cubic Spherical Bernstein-Bézier patch method (HCSBB)

The Hybrid Cubic Spherical Bernstein-Bézier (HCSBB) patch method is based on Alfeld et al. 1996. This method first triangulates the source grid points. Then the derivatives of the source field across the edges of the triangles are estimated. Using these, triangular patches from a blend of spherical Bernstein-Bézier polynomials are constructed are then used for the interpolation of the target points. The resulting target field always has a continuous first derivative[3]. This method is computationally expensive and produces a quite large interpolation stencil.

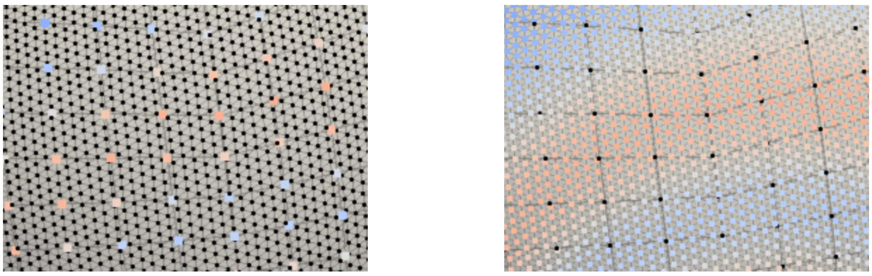


**Figure 4** – Source and target field of HCSBB interpolation

HCSBB method is simply invoked in the OASIS *namcouple* by the following line:

```
HCSBB
```

[3] The Patch recovery interpolation method (Zienkiewicz et al. 1992), which is implemented in the Earth System Modeling Framework does not guarantee the above mentioned property, which is why HCSBB is implemented in YAC.

### 2.1.7. Conservative method (CONSERV)

For CONSERV the source and target cells are defined by the source and target corners as defined by OASIS3-MCT. For each target cell, this method searches for all overlapping source cells and the weight of each source cell is defined by the overlap areas between the source cell and target cell.
A first order method following (Jones 1999) and a second order one following (Kritsikis et al. 2017) are supported.

When computing the intersection areas between source and target cells, YAC differentiates between edges defined on great circles (GC) and longitude-latitude circles (LL). This is defined by the user by specifying the source and target grid types (*srctype* and *tgttype*) in the *namcouple* (see examples in Section 2.2). This is especially important for cells from regular grids that are located close to the poles. Therefore, in contrast to other software that also provides this interpolation method, YAC does not require the usage of coordinate transformations close to the poles.

This method is invoked in the OASIS *namcouple* by the following line and options:

```
CONSERV order normalization
```

- *order* [string]
    - First conservative interpolation FIRST
    - Second conservative interpolation SECOND
      In contrast to first order conservative, this option generates a much bigger stencil because a gradient of the source field is computed for each source cell based on its neighbouring cells. This results in a much smoother field especially when going from a low- to a high-resolution grid. If the gradient computation fails, a zero-gradient is used for the respective source cell, which is basically a fallback to first order conservative interpolation.

- *normalization* [string]
  Selects the area normalization method as described in the SCRIP User manual (Jones, 1998).
  The different options produce different results only for target cells that are partially covered by non-masked source cells.

    - FRACAREA
      The sum of the area of the non-mask source cells overlapping with the target cell is used to normalize the respective target field value.
      This option gives reasonable flux values but may not be locally conservative.

    - DESTAREA
      The target cell area is used to normalize the target field value.
      This option might generate unreasonable flux values but it ensures local conservation of the source flux.

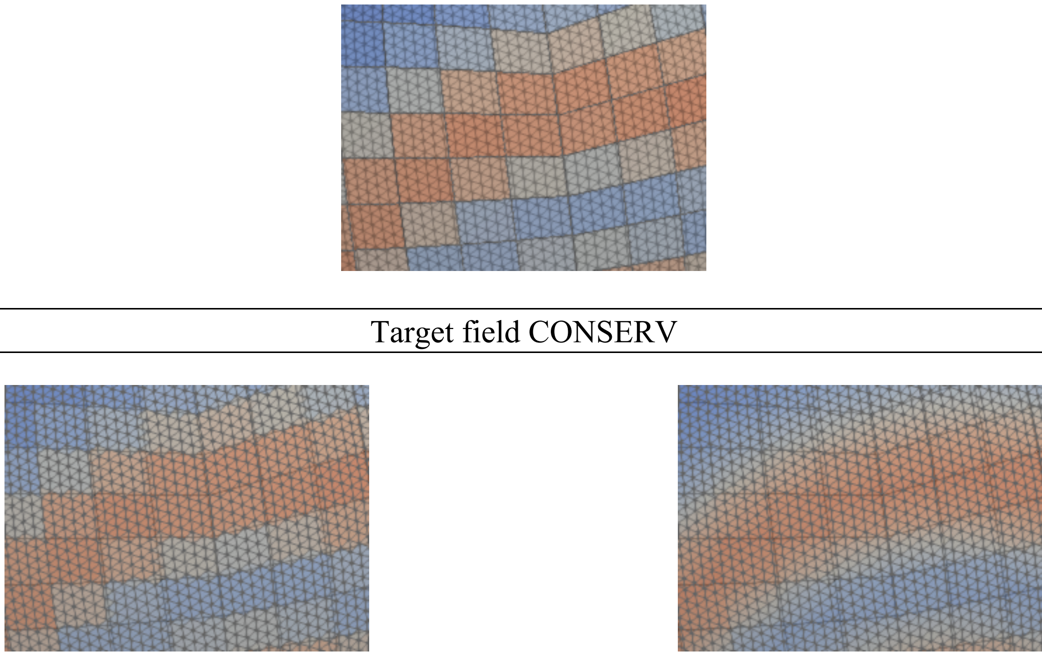



**Figure 5** – Target fields of CONSERV interpolation for the first and the second order

### 2.1.8. Source to Target mapping method (SPMAP)

For each source cell this method first searches for the nearest target cell within a maximum search distance (`search_distance`). The source cell data is then distributed to all non-masked target cells that are within the user-provided maximum spread distance (`spread`) around the initially found target cell respecting the grid connectivity around the sea-land mask. If multiple source cells contribute to a target cell, their contributions are summed up. Target cells not associated with any source cell will not get any value. There are also multiple normalization options (`scale`) available based on the cell areas of the associated source/target cells.

This method has been implemented in particular to cover the mapping of the hydrological runoff. This quantity is provided on selected cells at the coastline. The goal of all other interpolation schemes is to generate a value for all non-masked target cells. In contrast, this method aims to be locally mass conserving and not to lose any flux from the donor (source) cells.

SPMAP method is invoked in the OASIS *namcouple* by the following line and options:

```
SPMAP weighting spread search_distance scale src_radius tgt_radius
```

- *`weighting`* `[string]`
  - Arithmetic average `AVG`
  - Inverse distance weighting `DIST`
- *`spread`* `[real]`: Spread distance (optional, default: `0.0`).
  Note: a value of `0.0` results in the source field values being assigned to the single closest target point.
  Valid range is `[0.0, 90.0[`

- *search_distance* `[real]`: Maximum search distance (optional, default: `0.0`).
  Note: a value of `0.0` results in an unlimited search distance.
  Valid range is `[0.0, 180.0[`

- *scale* `[string]`: Scaling factor (optional, default: `NONE`)
  - `NONE` : No scaling. The sum of the weights for each source cell is `1.0`.
    This can be used in case the source and target field are not provided as a flux (for example, runoff source/target field unit is $m^3/s$).
  - `SRCAREA` : All weights are scaled by the area of the associated source cell.
    Therefore, weights for each source cell sum to its area.
    This can be used in case the source field is provided as a flux while the target field is not (for example, runoff source field unit is m/s; runoff target field unit is $m^3/s$).
  - `INVTGTAREA` : All weights are scaled by the inverse area of the associated target cell.
    This can be used in case the target field is provided as a flux while the source field is not (for example, runoff source field unit is $m^3/s$; runoff target field unit is m/s)
  - `FRACAREA` : All weights are scaled by the area of the associated source cell and the inverse area of the associated target cell.
    This can be used in case the source and target fields are provided as a flux (for example, runoff source/target field unit is m/s).

- *src_radius* [4] `[real]`: Sphere radius used for the area computation of the source cells (optional, default: `1.0`).

- *tgt_radius* `[real]`: Sphere radius used for the area computation of the target cells (optional, default: `1.0`).

[4] The possible values of thes options have evolved in YAC 3.5.2 and therefore in OASIS3-MCT_6.0 allowing the automatic use of the value of OASIS earth radius (src_radius/tgt_radius = “EARTH”) or of user-defined cell surfaces read from areas.nc (src_radius/tgt_radius = “FILE”)

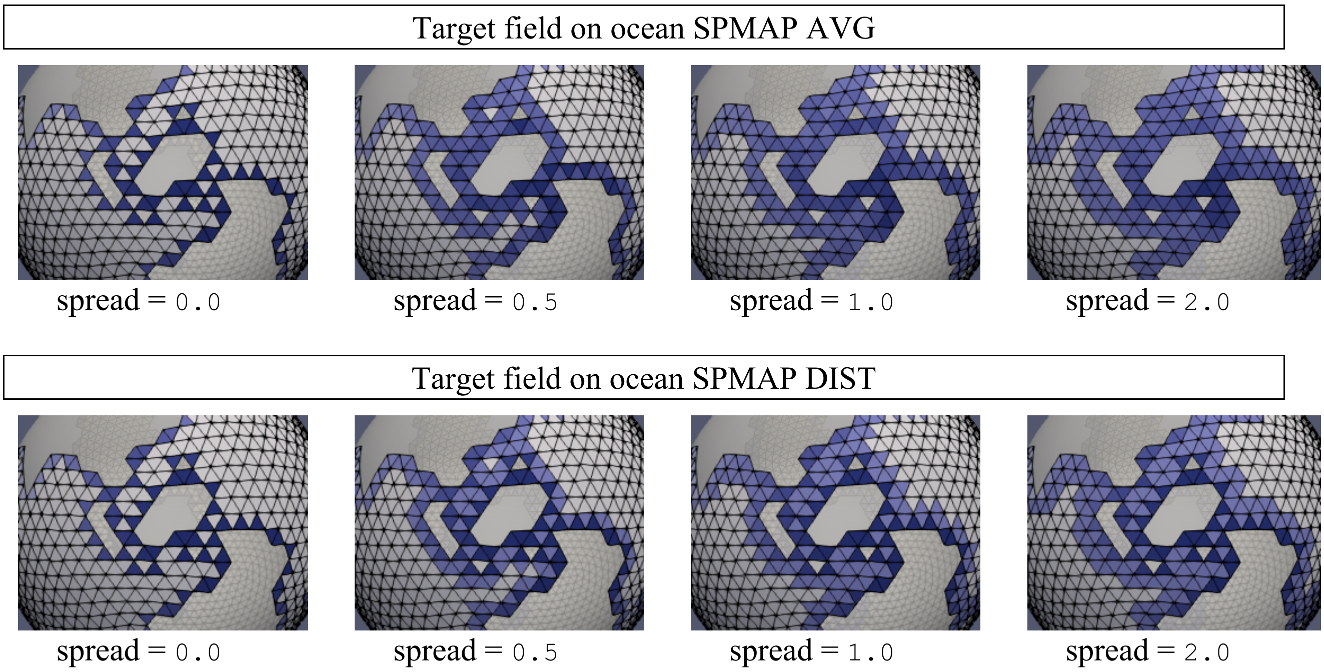


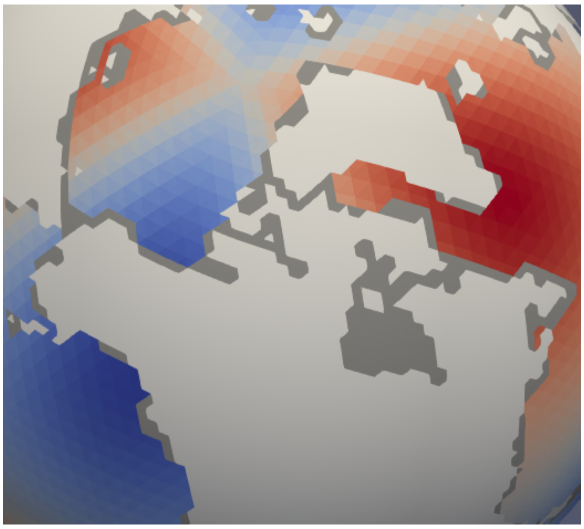

**Figure 6** – SPMAP remapping of a runoff field on a target ocean grid for AVG and DIST weightings and different spreads

### 2.1.9. CREEP method

Creep fill is an extrapolation method described in Kara et al. 2007. For each unmapped target point, the method checks the status of its neighbouring target points. If successfully-interpolated target neighbour points are available, the respective unmapped point value is calculated based on their results. This check is repeated until all target points are remapped or for a specified number of iterations.

In the following example, a source field is interpolated to a finer ocean grid. A 1st-Order conservative interpolation is first being used. The target field has then a number of non-interpolated cells (grey cells in right panel on Fig. 7) due to mismatches between the source and target masks. The creep fill interpolation method is one option to fill these cells (see Fig. 8).

Source field with target grid overlayed



Target field after 1st order conservation

**Figure 7** – Example of field after a 1st order conservative interpolation on a fine target grid with mismatching source and target masks

CREEP extrapolation method is invoked in the OASIS *namcouple* by the following line and option:

```
CREEP iter
```

- `iter` `[integer]`: Number of iterations. A value of -1 results in the execution of the algorithm until no additional target points can be remapped by this method.
  Valid range is `[-1.0, ∞[`

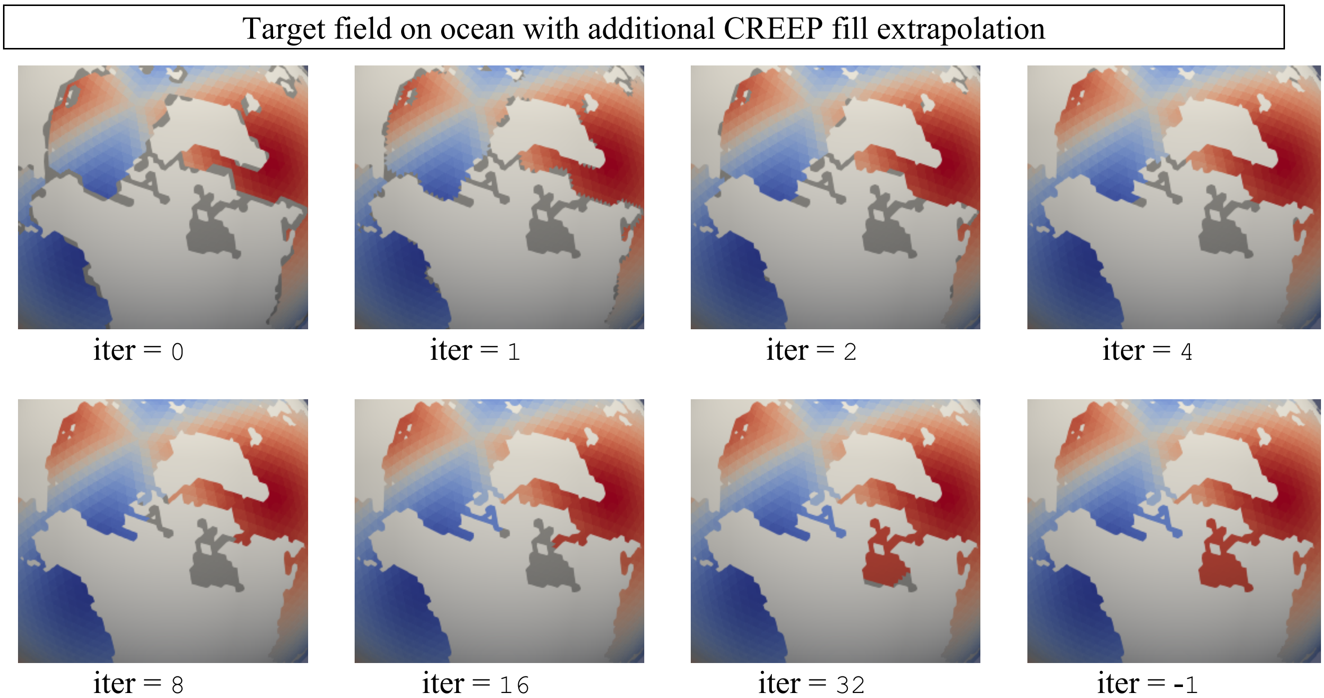



**Figure 8** – Target fields of CREEP extrapolation for different number of iterations

Another method to fill the missing target cells in the example would have been the N-Nearest-Neighbour interpolation.

However, this method does not take grid connectivity into account, which leads to obvious differences in the remapping and awkward results; see for example the pink point at the end of the blue thin peninsula at the center of Fig.9.

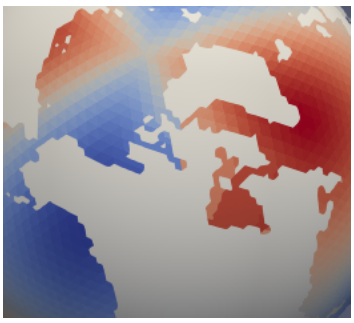


**Figure 9** – Target field after 1-NN fallback instead of CREEP fill extrapolation

## 2.2. YAC interpolation stack

Individual interpolation methods may have limitations that prevent them from assigning a value to all required target field values. A typical solution is the definition of a fallback solution that tries to handle these remaining points.

YAC introduced the concept of the interpolation stack. It is a more general approach to defining fallback solutions. An interpolation stack is comprised of a list of one or more interpolation methods. To compute an interpolation, all required target field points are passed to the first method in a stack. This will try to generate interpolated values for all target field points. The ones for which the method failed are passed to the next method in the stack. This is done until the end of the stack is reached.

There are no limitations on which interpolation methods can occur at which position in the stack as long as the respective methods are compatible with available source and target field data. This allows for a very flexible definition of an interpolation.

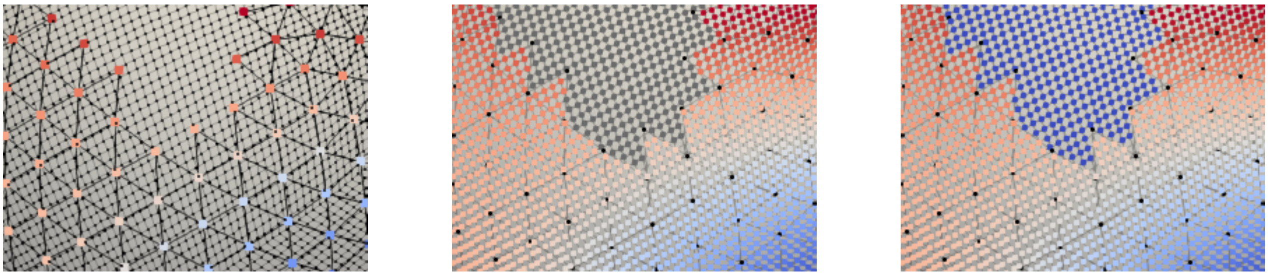
The Zero interpolation will assign a zero value to all remaining target points in a stack. It can be used for example to mark target points that could not be interpolated by the other methods in the stack. And since Zero value interpolation will always successfully process all remaining target point, it does not make sense to add any other interpolation method afterwards in the stack.

You may want to inspect the results of the interpolation stack. YAC further supports this by providing a tool that allows you to visualize a weight file (see Section 4).

#### 2.2.1. Example of interpolation stack

##### 2.2.1.1. Average + Zero stack

A simple stack consists of an Average interpolation combined with a Zero value interpolation. The Average interpolation will fail for target points that are located outside of the region covered by the source grid points or due to masked source points. The Zero value interpolation will assign a fixed value (`0.0`) to these target points.



Source field | Target field after AVG (grey points are not interpolated) | Target field after ZERO (these points, now in blue, are given a value of 0.0)

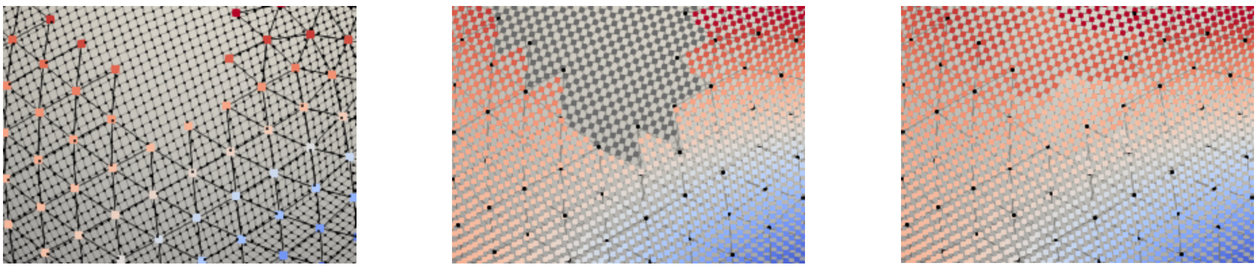
**Figure 10** – Example of Average + Zero interpolation stack

The OASIS *namcouple* file contains these lines:

```
YAC
srctype tgttype 2 wgt_filename npio
AVG BARY
ZERO
```

##### 2.2.1.2. HCSBB + N-Nearest-Neighbour stack

A higher order stack consisting of a Hybrid Cubic Spherical Bernstein-Bézier patch interpolation with a N-Nearest-Neighbour fallback, which is used for the extrapolation of target points not covered by the source grid.



Source field | Target field after HCSBB (grey points are not interpolated) | Target field after 1-NN (these points are given the source nearest-neighbour value)

**Figure 11** – Example of HCSBB + 1-NN interpolation stack

The OASIS *namcouple* file contains these lines:

```
YAC
srctype tgttype 2 wgt_filename npio
HCSBB
NNN AVG 1
```

#### 2.2.1.3. Source-Point-Mapping + Zero stack

A typical river-runoff stack can consist of Source to Target Mapping (SPMAP) combined with Zero value interpolation. The Zero value interpolation ensures that all target ocean cells receive a valid value in the get operation.

| Source field (red cells indicate masked source cells) | Target field after SPMAP (red cells do not receive any data) | Target field after ZERO (these cells are given a value of 0.0) |
|---|---|---|

**Figure 12** – Example of SPMAP + Zero interpolation stack

The OASIS *namcouple* file contains these lines:

```
YAC
srctype tgttype 2 wgt_filename npio
SPMAP AVG 0.5
ZERO
```

#### 2.2.1.4. Conservative 2nd + Conservative 1st + N-Nearest-Neighbour stack

An interpolation stack for conservative interpolation. It first tries to interpolate using a second order conservative method, which uses a bigger stencil, and therefore can fail at the edges of the source grid. As a backup, a first order conservative method is used, which is more robust than the 2nd Order, but does not provide values for target cells falling outside the source grid cells. Therefore, it is followed by a N-Nearest-Neighbour method for all remaining cells.

The OASIS *namcouple* file contains these lines:

```
YAC
srctype tgttype 3 wgt_filename npio
CONSERV SECOND DESTAREA
CONSERV FIRST FRACAREA
NNN DIST 4
```

# 3. Integration of YAC interpolation library into OASIS3-MCT

The sources of YAC interpolation library have been introduced in OASIS3-MCT as git submodules, in directories `oasis3-mct/lib/yaxt` and `oasis3-mct/lib/yac` . To obtain them the `--recurse-submodules` option must be added to the command for creating or updating the local git repository:

```
git clone --recurse-submodules https://nitrox.cerfacs.fr/globc/OASIS3-MCT/oasis3-mct
```

or `git pull --recurse-submodules` .

The versions of yaxt and yac recovered correspond to the tags mentioned in the file `oasis3-mct/.gitmodules` (`release-0.10.0` tag for yaxt and `release-3.4.0_p3` tag for yac, for the versions used for this report). If a tag is changed in this file the submodule can be updated with the command `git submodule update --remote` .

YAC interpolation library is interfaced with OASIS3-MCT by the new Fortran module `oasis3-mct/lib/psmile/src/mod_oasis_yac_map.F90`. All the YAC interfacing developments are conditioned under the CPP key YAC_REMAP, which is enabled with the ENABLE_YAC variable in the user's `make.inc` file. Files affected by the integration of YAC in OASIS3-MCT are `mod_oasis_coupler.F90` and `mod_oasis_namcouple.F90` .

The calculation of interpolation weights by the YAC library at OASIS execution is activated in the OASIS *namcouple* with the keyword YAC (see examples in Section 2.2). At the end, the format of the weight file produced by YAC is converted to the OASIS format and written in the run directory.

# 4. Plotting YAC weight files on grids in OASIS3-MCT

Based on an existing weight file, a YAC tool can be used to display the weights of the source grid cells contributing to the remapping for a given target grid cell. This tool is installed with OASIS3-MCT, and is called `plot_weights.py`.

To use it, you can add the appropriate path in your environment via the variables defined for the use of OASIS3-MCT:

```
export PATH=${OASIS_COUPLE}/INSTALL_OASIS.${OASIS_ENV}/bin:$PATH
```

The usage of `plot_weights.py` is:

```
plot_weights.py [-h] [--center LON LAT] [--source_idx SOURCE_IDX] [--
target_idx TARGET_IDX] [--zoom ZOOM][--label_src_grid {vertex,edge,cell}] [--
label_tgt_grid {vertex,edge,cell}][--coast_res [{10m,50m,110m}]] [--projection
[{orthographic,stereographic,platecarree}]][--stencil_only] [--save_as
SAVE_AS] [--log-level LOG_LEVEL]source_grid [target_grid] [weights_file]

Plot grids and yac weights file.

positional arguments:
  source_grid           appearing as, when used with OASIS grid format,
                        scrip:grids.nc:masks.nc:ssss
                        where ssss is the source grid acronym
  target_grid           appearing as, when used with OASIS grid format,
                        scrip:grids.nc:masks.nc:tttt
                        where tttt is the target grid acronym
  weights_file          the weights file


optional arguments:
  -h, --help            show this help message and exit
  --center LON LAT, -c LON LAT
                        center of the orthographic projection
  --source_idx SOURCE_IDX, -s SOURCE_IDX
                        index of source cell to focus
  --target_idx TARGET_IDX, -t TARGET_IDX
                        index of target cell to focus
  --zoom ZOOM, -z ZOOM  zoom around the cell
  --label_src_grid {vertex, edge, cell}
                        Add labels at the source grid
  --label_tgt_grid {vertex, edge, cell}
                        Add labels at the source grid
  --coast_res [{10m,50m,110m}]
                        Resolution of coastlines (def 50m). Omit argument to
disable coastlines.
  --projection [{orthographic,stereographic,platecarree}]
                        Type of projection
  --stencil_only
  --save_as SAVE_AS     Save to file instead of showing the figure
  --log-level LOG_LEVEL
                        Configure the logging level.
```

An example of using `plot_weights.py`, in a directory containing the grid file, the mask file and the weight file, is:

```
plot_weights.py scrip:grids.nc:masks.nc:torc scrip:grids.nc:masks.nc:bggd
rmp_torc_to_bggd_YAC_AVGBARY_NCCDIST.nc -t 8840 --label_src_grid cell --
label_tgt_grid cell --zoom 1.0 --save_as
plt_weights_torc_to_bggd_YAC_AVGBARY_NCCDIST_t8840.png
```

The resulting plot is shown in the Fig. 13. The green source cell 10678 is the only source cell contributing to the blue target cell 8840 for the remapping.

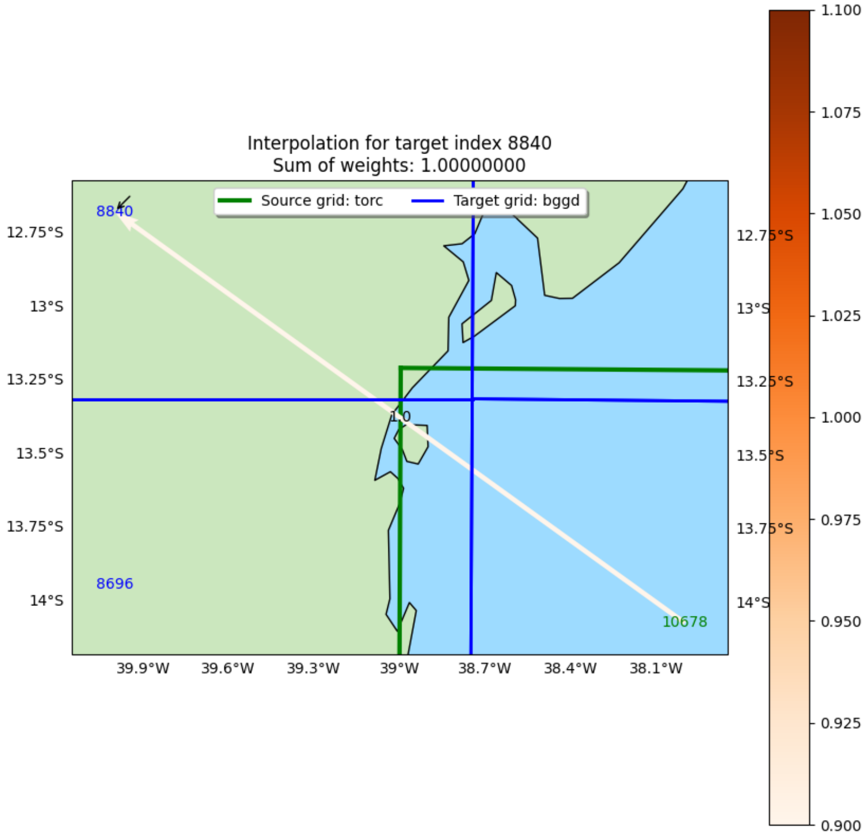



**Figure 13** – Contribution of a source cell to a target cell for the remapping

# 5. Qualitative comparisons of regridding libraries

The regridding libraries ESMF, SCRIP, XIOS and YAC are compared here.

The version of these libraries involved in the comparisons are ESMF-8.6.0, SCRIP in OASIS3-MCT_5.0 branch and XIOS2 r2134[5].

Two versions of the YAC library are involved in the comparisons:

- YAC2 : the version used offline in the 2021 benchmark (Valcke et al. 2021), release v2.3.0
- OYAC : the YAC release `v3.1.0` initially integrated into OASIS3-MCT branch `yac_remap`, on 17 January 2024

## 5.1. Algorithms

### 5.1.1. Nearest-Neighbour

The values of the nearest neighbour on the source grid gives the value for each target grid point.

In OYAC we use a single stack being

```
NNN AVG 1
```

### 5.1.2. Bilinear (or equivalent)

The regridding is based on a bilinear approximation, which uses the value of the coupling field at the four enclosing source grid points.

In YAC2, we used an interpolation stack consisting of the average method with inverse distance weighting (`AVG DIST`), followed by a 2-Nearest-Neighbour (`NNN AVG 2`) interpolation fallback:

```
AVG DIST
NNN AVG 2
```

In OYAC, we use an interpolation stack consisting of the average method again but with a weighting based on the barycentric coordinates of the target point (`AVG BARY`), followed by a Nearest-Corner-Cells interpolation fallback (`NCC DIST`), and again followed by a nearest-neighbour interpolation fallback (`NNN AVG 1`). This interpolation stack is useful because the AVG method can fail close to the mask edges and the NCC method is a good fallback option. It gives better overall results in the comparisons (see Section 5.3.2):

```
AVG BARY
NCC DIST
NNN AVG 1
```

---

[5] The sources used correspond to SVN revision 2134 dated 2021-04-29 that can be extracted with the SVN command:

```
svn co -r 2134 http://forge.ipsl.jussieu.fr/ioserver/svn/XIOS/trunk XIOS
```

### 5.1.3. Second-Order non-conservative

Different algorithms are used in the different regridders to evaluate second-Order non-conservative regriddings.

In YAC2 we used a Hybrid Cubic Spherical Bernstein-Bézier (HCSBB) patch interpolation method without defining any atmospheric mask (to obtain reasonable results); this method is based on a local triangular patch constructed from a blend of certain spherical Bernstein-Bézier polynomials.

In OYAC, HCSBB patch interpolation method is used taking into account the definition of an atmospheric mask, but adding an interpolation stack consisting of a Nearest-Corner-Cells interpolation fallback (`NCC DIST`), followed by a nearest-neighbour interpolation fallback (`NNN AVG 1`). This interpolation stack is useful because the HCSBB method can fail close to the mask edges.

```
HCSBB
NCC DIST
NNN AVG 1
```

### 5.1.4. First-Order conservative FRACAREA and DESTAREA

In a first-order conservative remapping, the value for each target cell is computed as a weighted sum of source cell values, with the contribution of a source cell being proportional to the fraction of the target cell intersected by the source cell. Different normalisation options exist. DESTAREA uses the whole target cell area for the normalisation, whereas FRACEARA uses the intersected area of the target cell.

In YAC2 and OYAC we use the same method without interpolation fallback:

```
CONSERV FIRST FRACAREA
```

and

```
CONSERV FIRST DESTAREA
```

### 5.1.5. Second-Order conservative FRACAREA

The basis of a second-order conservative remapping is the same than the first-order conservative remapping but additional terms proportional to the gradients of the source field are applied. The algorithm is based on Kritsikis et al. 2017.

In YAC2 and OYAC we use the same method without interpolation fallback:

```
CONSERV SECOND FRACAREA
```

## 5.2. Metrics

The 2021 benchmark (Valcke et al. 2021) implemented the calculation of the metrics proposed by the CANGA project.

If we note,

- $\Psi^s$: the analytical function on the source grid
- $\Psi^t$: the analytical function on the target grid
- $R\Psi^s$: the source analytical function regridded on the target grid
- $I_s$: the integral on the source grid
- $I_t$: the integral on the target grid

those metrics are defined as:

- Mean misfit: mean ( $|R\Psi^s - \Psi^t| / |\Psi^t|$ )
- Max misfit: maximum ( $|R\Psi^s - \Psi^t| / |\Psi^t|$ )
- RMS misfit : RMS ( $|R\Psi^s - \Psi^t| / |\Psi^t|$ )
- Lmin: ($\min \Psi^t - \min R\Psi^s$ ) / max ( $|\Psi^t|$ ): a positive Lmin detects an undershoot of the function minimum (i.e.it is reinforced) while a negative Lmin detects some smoothing of the function minimum
- Lmax : ( $\max R\Psi^s - \max \Psi^t$) / max ( $|\Psi^t|$ ) a positive Lmax detects an overshoot of the function maximum (i.e.it is reinforced) while a negative Lmax detects some smoothing of the function maximum
- Source global conservation: $|I_t(R\Psi^s) - I_s(\Psi^s)| / I_s(\Psi^s)$
- Target global conservation: $|I_t(R\Psi^s) - I_t(\Psi^t)| / I_t(\Psi^t)$

We calculated those metrics for all regridders (see Section 5.3), for all algorithms (see Section 5.3), except when the regridder did not support the algorithm (i.e. nearest-neighbour, bilinear and second-order non-conservative for XIOS), for all grid pairs (see Section 5.3) and for four analytical functions.

These 4 analytical functions are used to define the coupling fields to be regridded, named and characterised by[6]:

a) sinusoid: a slowly varying standard sinusoid over the globe
b) harmonic: a more rapidly varying function with 16 maximums and 16 minimums in northern and southern bands
c) vortex: a slowly varying function with two added vortices, one in the Atlantic and one over Indonesia
d) gulfstream: the slowly varying standard sinusoid with a mimicked Gulf Stream

They are illustrated on Fig. 14.

[6] Their exact definition is available at https://inle.cerfacs.fr/attachments/10233/function_ana.f90

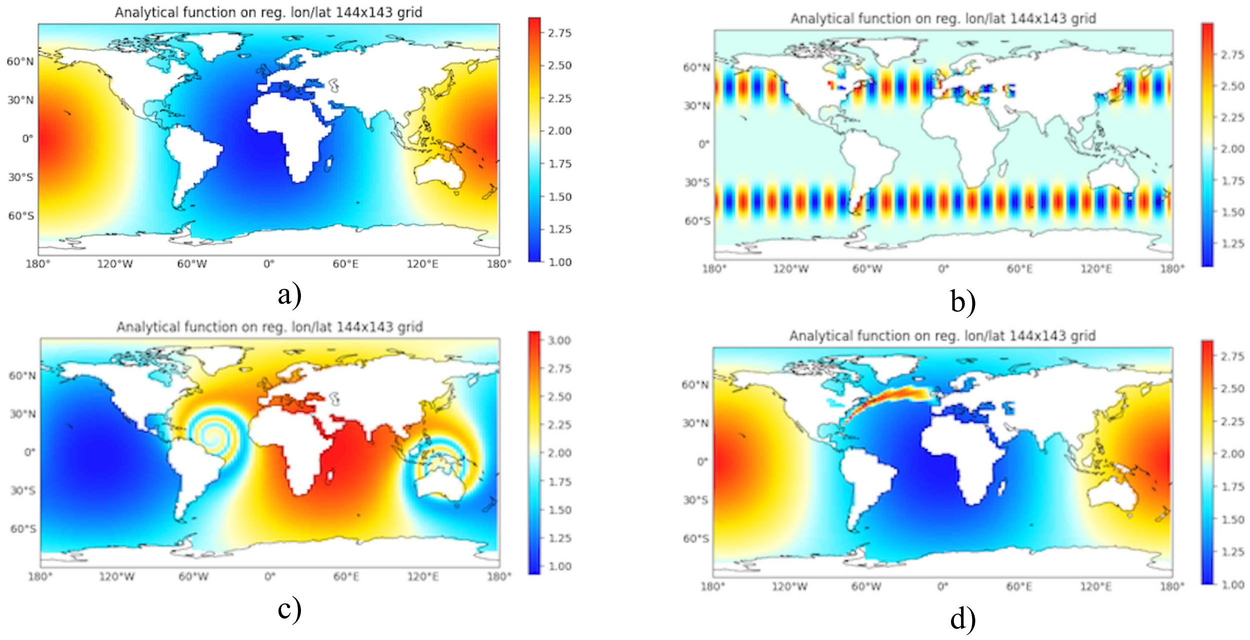



**Figure 14** – The 4 functions defining the analytical field to be regridded:
a) sinusoid, b) harmonic, c) vortex, d) gulfstream

## 5.3. Results

All benchmark metrics of the regridding were calculated

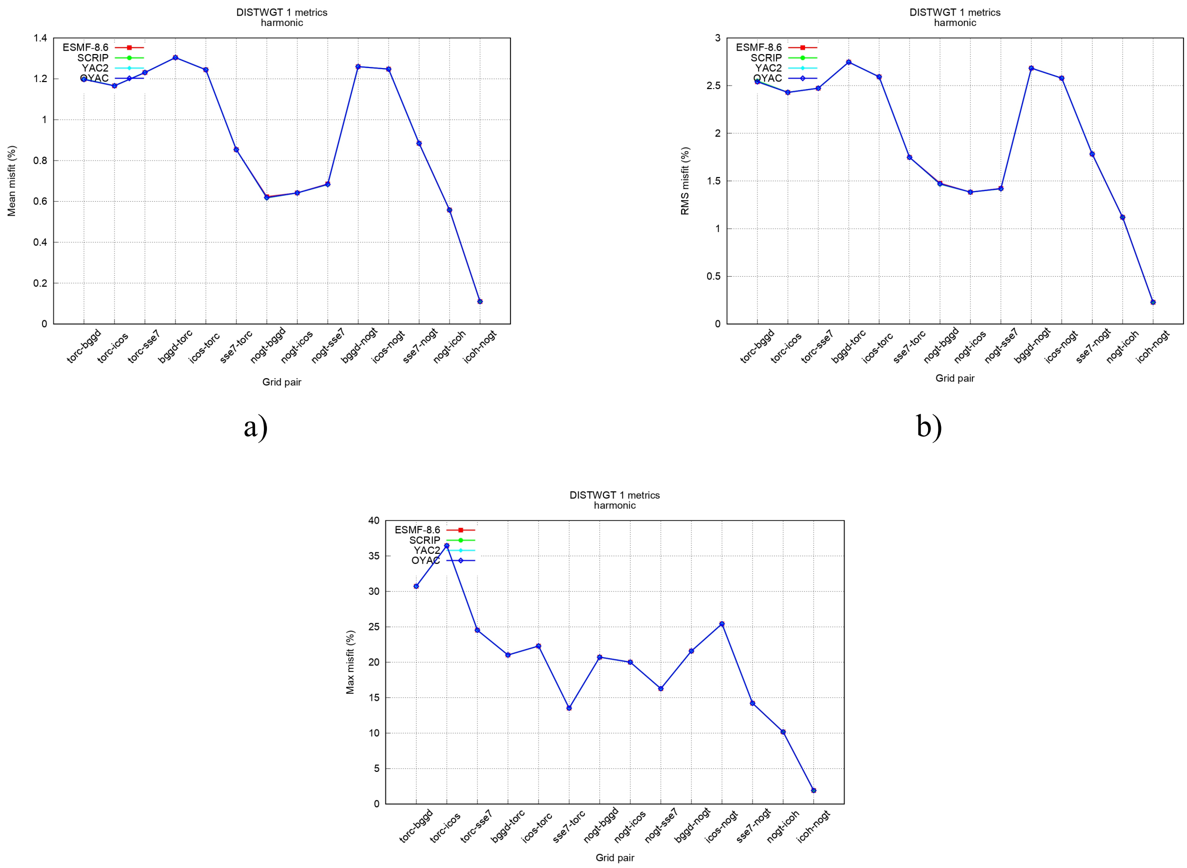
- for the regridders (presented in Valcke et al. 2021):
  - ESMF,
  - SCRIP only for non-conservative algorithms,
  - YAC2,
  - OYAC,
  - XIOS only for conservative algorithms,
- for the algorithms:
  - Nearest-Neighbour,
  - Bilinear (or equivalent),
  - Second-Order non-conservative
  - First-Order conservative,
  - Second-Order conservative
- for the grid pairs (described in Valcke et al. 2021):
  - torc-bggd, torc-icos, torc-sse7, in both ways,
  - nogt-bggd, nogt -icos, nogt -sse7, nogt-icoh, in both ways,
- for the analytics functions: sinusoid, harmonic, vortex and gulfstream.

We cannot discuss all metrics obtained but we illustrate here the main conclusions of our analysis with specific examples[7].

[7] Unlike the layout of the graphs in the 2021 benchmark (Valcke et al. 2021), here the y-axis of the graphs (the metric) is not logarithmic.

### 5.3.1. Nearest-Neighbour

The nearest-neighbour algorithm is available in ESMF, SCRIP, YAC2 and OYAC regridders.




a) b)

c)

**Figure 15** – a) Mean, b) RMS and c) Max misfit of the 1-Nearest-Neighbour algorithm for different grid pairs with the harmonic function

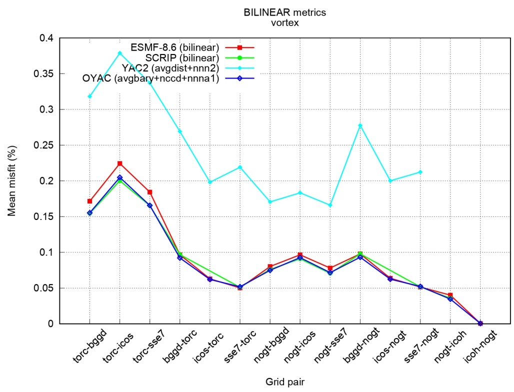
Fig. 15 shows the Mean, RMS and Max misfit for these regridders for the different grid pairs with the harmonic function. The regridders produce the same results: the curves are superimposed and not distinguishable; this is also true for the other functions (not shown). The results are identical for the different regridders as in the 2021 benchmark (see Valcke et al. 2021).

### 5.3.2. Bilinear

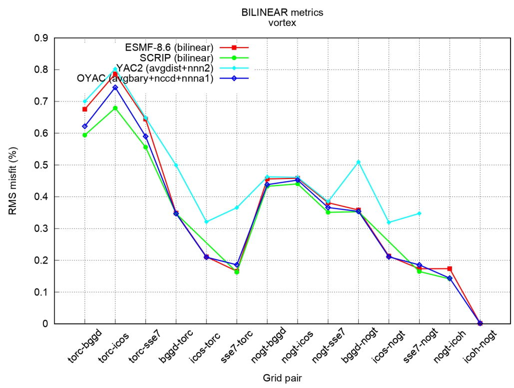
The Bilinear algorithm (or equivalent) is available in ESMF, SCRIP, YAC2 and OYAC regridders (see Section 5.1.2).

Fig. 16 and Fig. 17 show the Mean, RMS and Max misfit and Lmax for the different grid pairs for these regridders, respectively with the vortex function and for the gulfstream function.

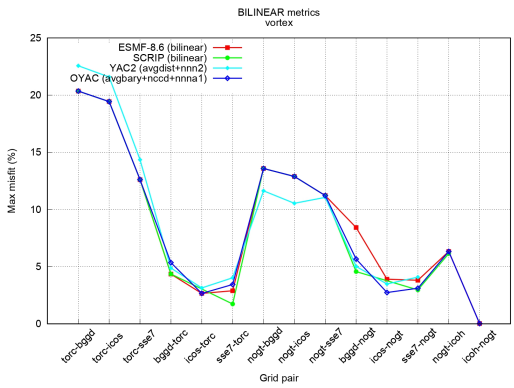
The algorithm in YAC2 and OYAC is not a pure bilinear algorithm but a stack of different methods, see Section 5.1.2. The OYAC interpolation stack offers a better Mean misfit with the vortex and gulfstream functions than the YAC2 interpolation stack. But we note a high Max misfit for OYAC for torc->sse7 with the gulfstream function, as high as for the SCRIP (see Fig. 17c, to be investigated).



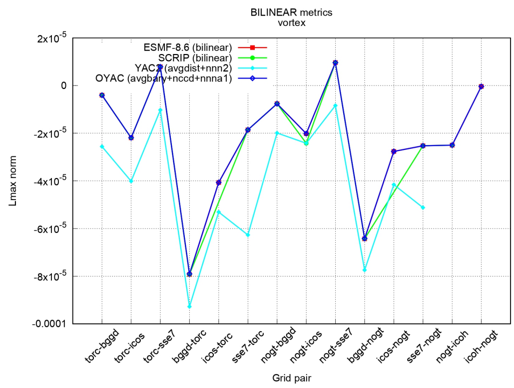

a)




b)




c)




d)

**Figure 16** – a) Mean, b) RMS and c) Max misfit and d) Lmax of the Bilinear or equivalent algorithm for different grid pairs with the vortex function

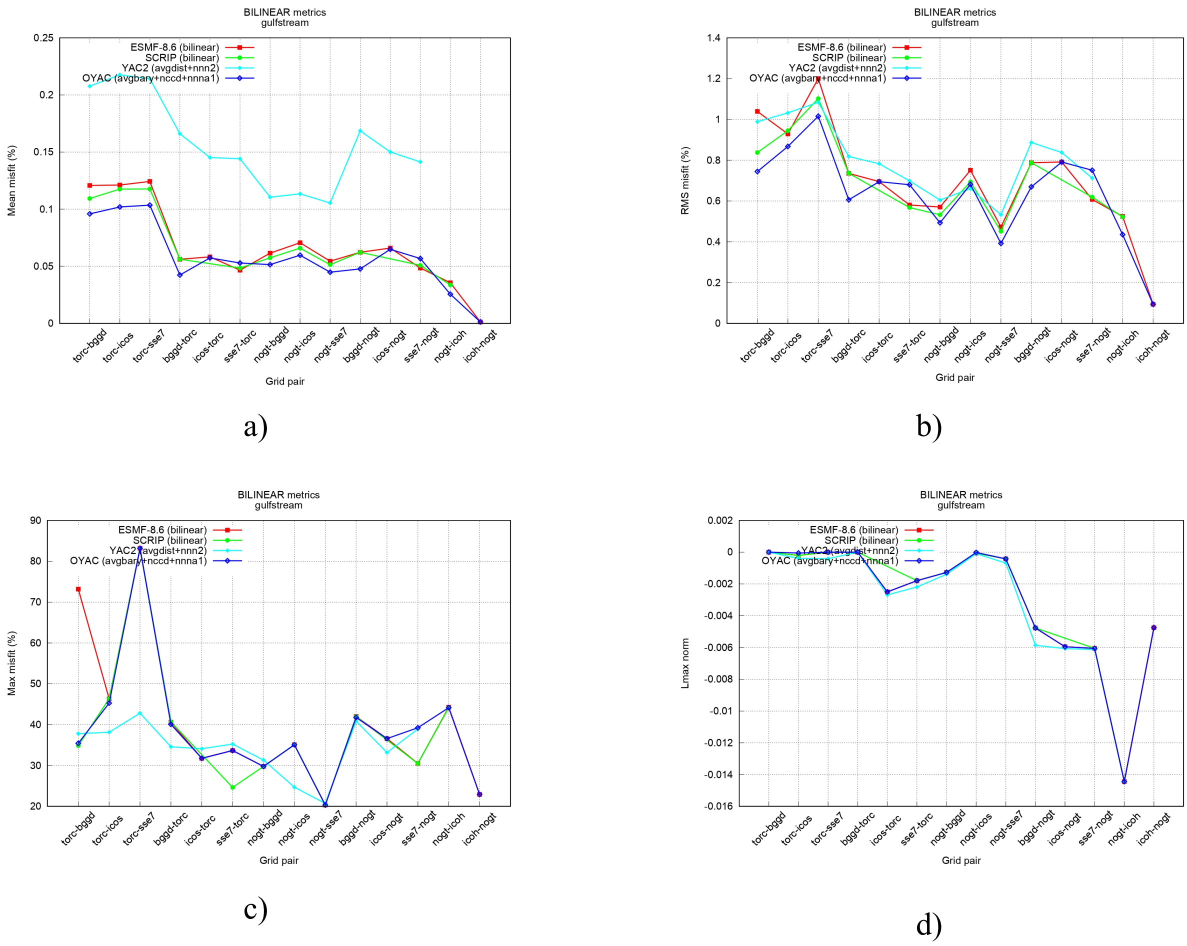


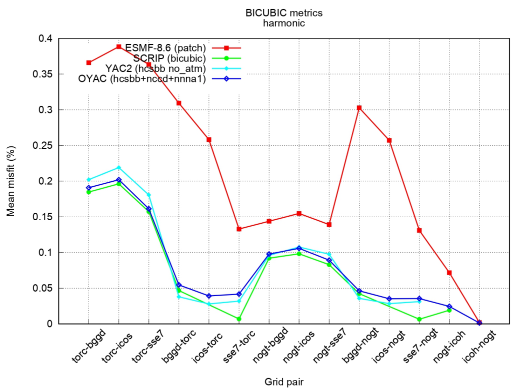

**Figure 17** – a) Mean, b) RMS and c) Max misfit and d) Lmax of the Bilinear or equivalent algorithm for different grid pairs with the gulfstream function

### 5.3.3. Second-Order non-conservative

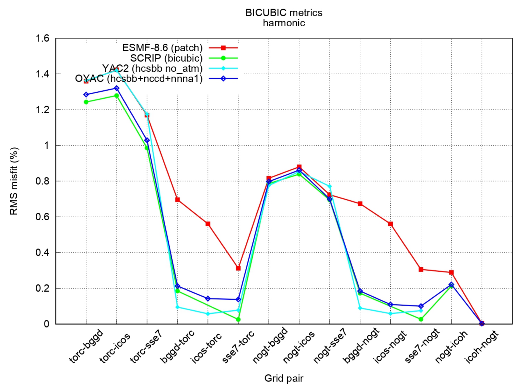
2nd-Order non-conservative algorithms are available in ESMF, SCRIP, YAC2 and OYAC regridders (see Section 5.1.3).

Fig. 18 and Fig. 19 show the Mean, RMS and Max misfit and Lmax for the different grid pairs for these regridders, respectively for the harmonic function and for the gulfstream function.

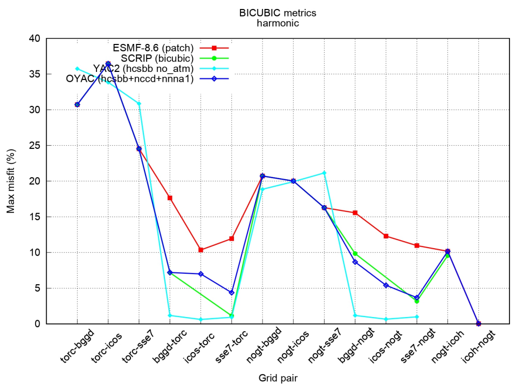
The algorithm in YAC2 and OYAC is the HCSBB patch method, see Section 2.1.6. Note here that YAC2 implements this method without fallback but not defining any atmospheric mask, which is clearly to its advantage. The OYAC interpolation stack offers good results, similar to those of YAC2. But again, we note a high Max misfit for OYAC for torc->sse7 with the gulfstream function (see Fig. 19c, to be investigated).



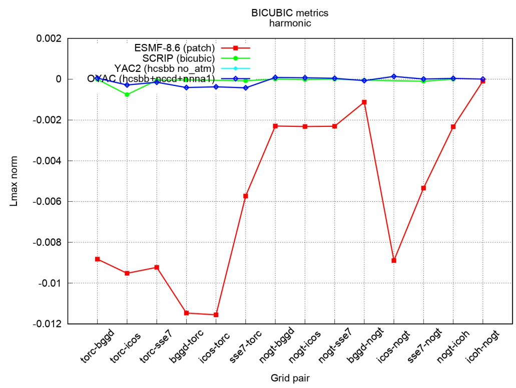

a)




b)




c)




d)

**Figure 18** – a) Mean, b) RMS and c) Max misfit and d) Lmax of the 2nd-Order non-conservative algorithm for different grid pairs with the harmonic function

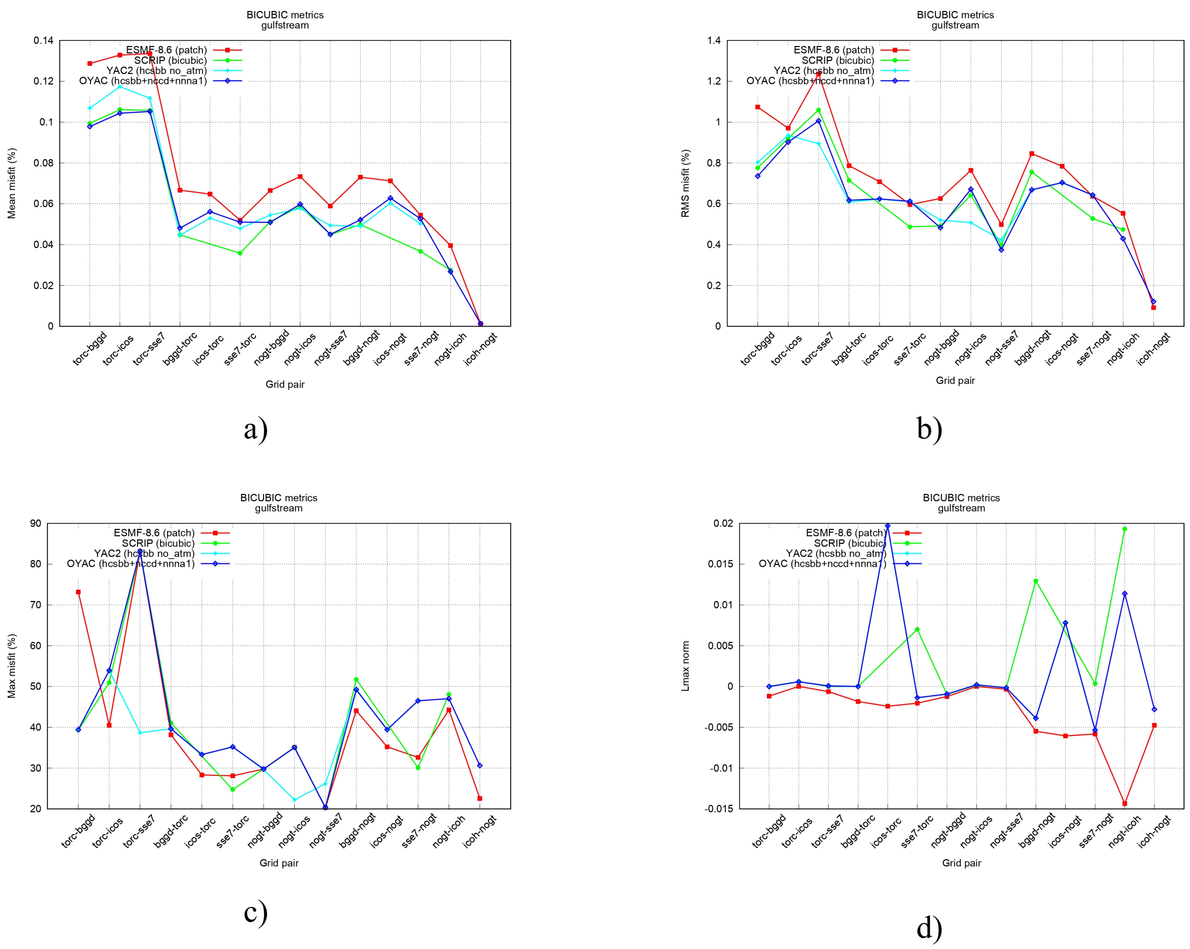
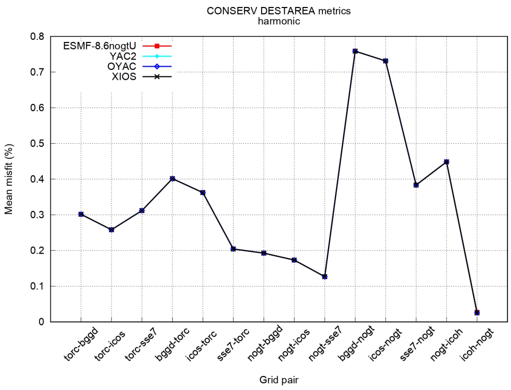



**Figure 19** – a) Mean, b) RMS and c) Max misfit and d) Lmax of the 2nd-Order non-conservative algorithm for different grid pairs with the gulfstream function

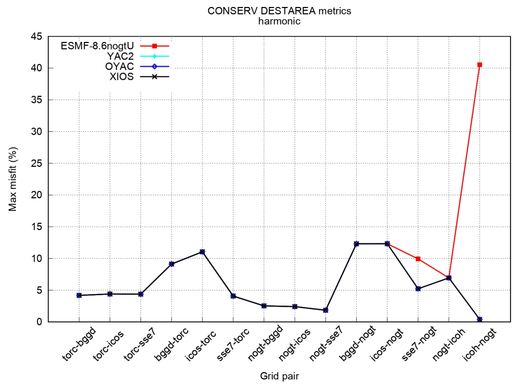
### 5.3.4. First-Order conservative with Destarea normalization

For the comparison of conservative algorithms, we have excluded the SCRIP regridder, which gave poor results, particularly with the Destarea normalization (see Valcke et al. 2021). The XIOS regridder is included in the comparison, unlike for the non-conservative algorithms that are not available in XIOS. The problems previously detected with ESMF when nogt is the source grid are resolved by passing the description of this grid in an unstructured format (see Valcke et al. 2021). The associated legend in the graphics is “ESMF-8.6nogtU”

The 1st-Order conservative algorithm with Destarea normalization is thus presented for ESMF, YAC2, OYAC and XIOS regridders.




a)




b)

**Figure 20** – a) Mean and b) Max misfit of the 1st-Order conservative algorithm with Destarea normalization for different grid pairs with the harmonic function

Fig. 20 shows the Mean and Max misfit for the different grid pairs for these regridders, with the harmonic function. The regridders produce almost exactly the same results, with one exception for ESMF, as a new fault appears with an anomalous Max misfit for sse7->nogt and icoh->nogt. Note that this fault, that still has to be investigated, also appears with the sinusoid and vortex functions, but not with the gulfstream function (not shown here).

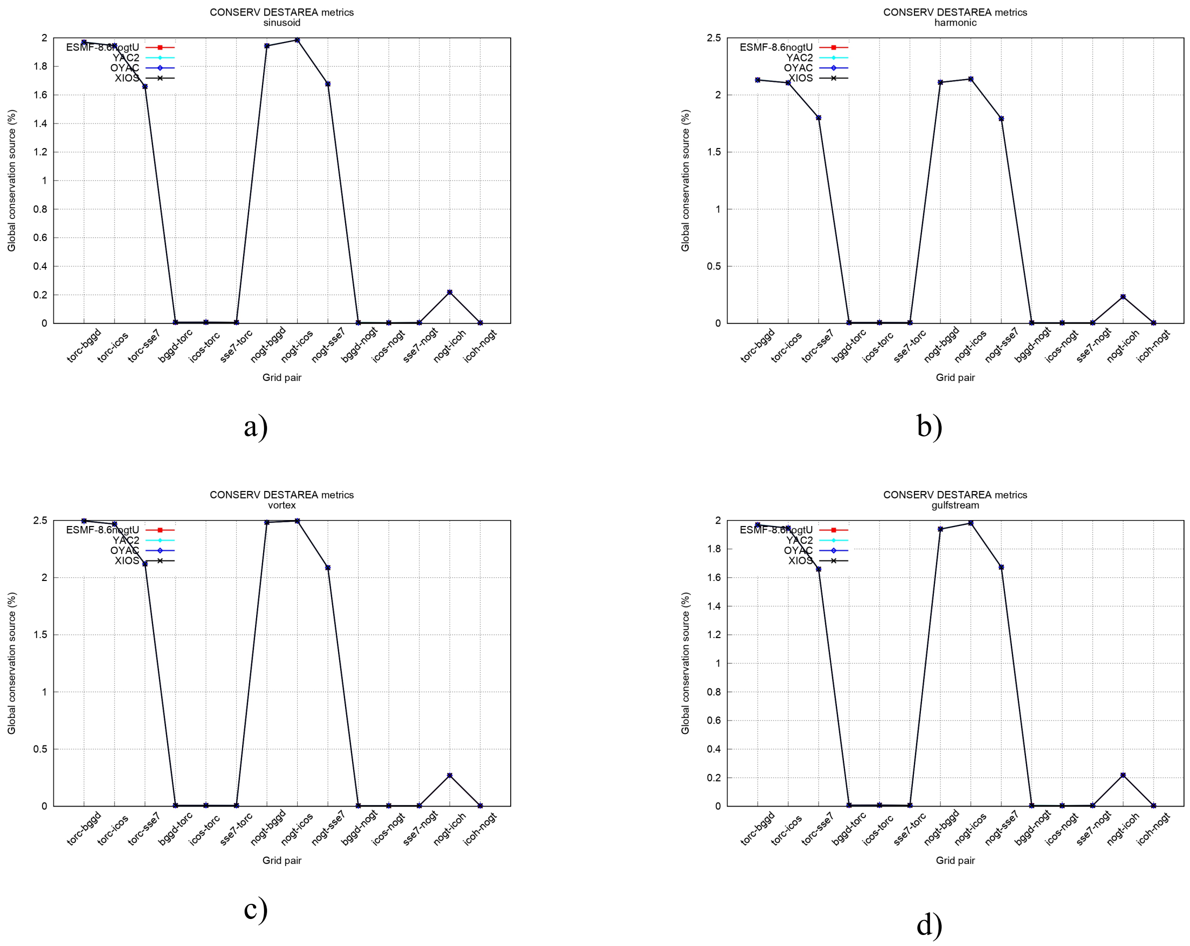
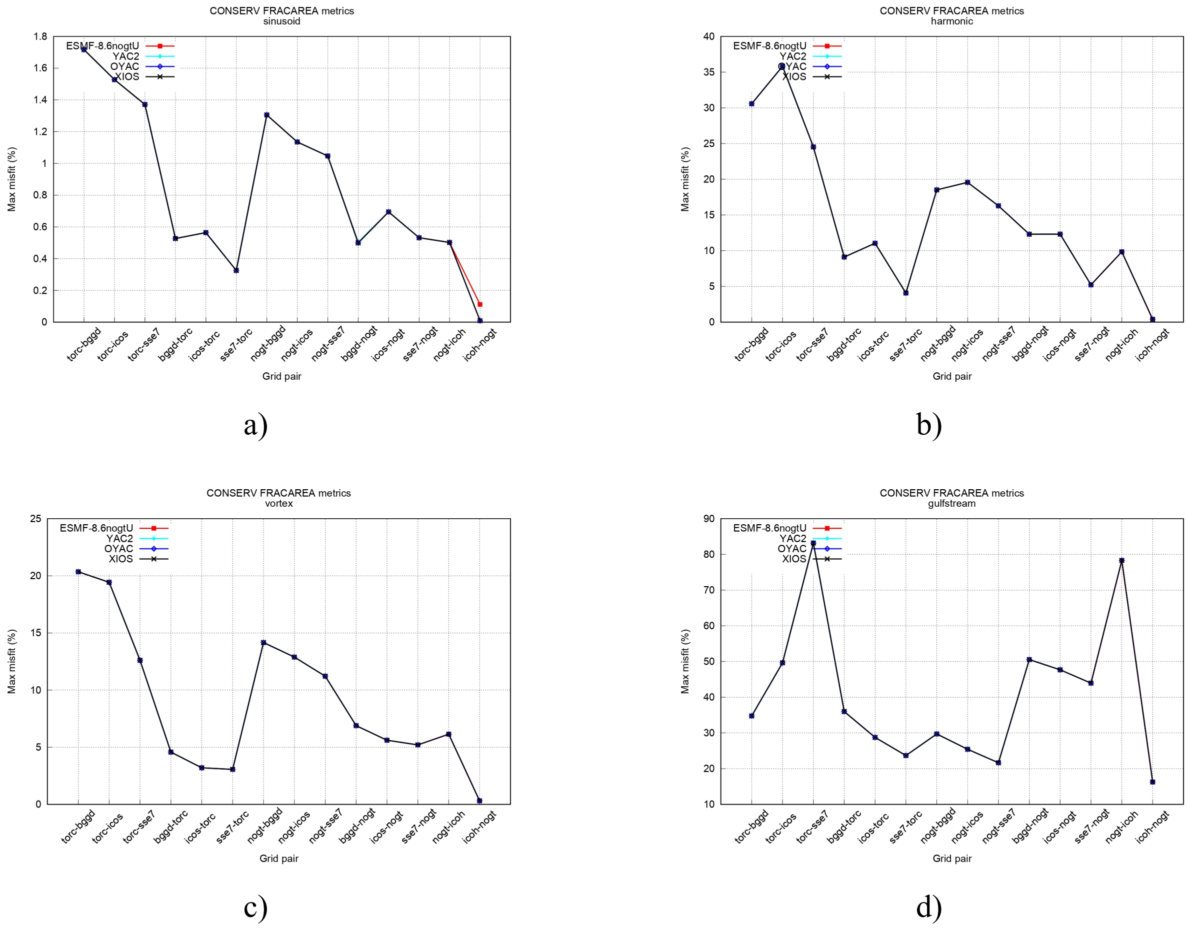



**Figure 21** – Conservation (%) of the 1st-Order conservative algorithm with Destarea normalization for different grid pairs for the a) sinusoid, b) harmonic, c) vortex, d) gulfstream functions

Fig. 21 shows the global conservation metric for the different grid pairs for these regridders for the four functions. All these regridders present almost exactly the same and very reasonable results.

### 5.3.5. First-Order conservative with Fracarea normalization

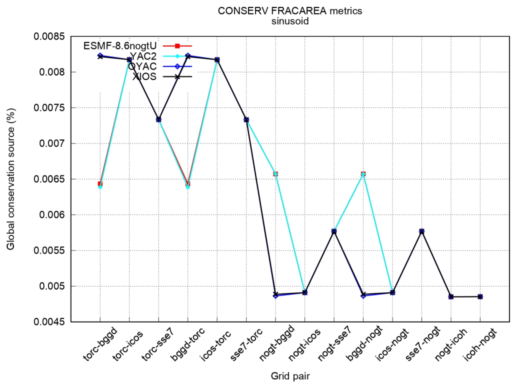
The 1st-Order conservative algorithm with Fracarea normalization is also presented for ESMF, YAC2, OYAC and XIOS regridders.



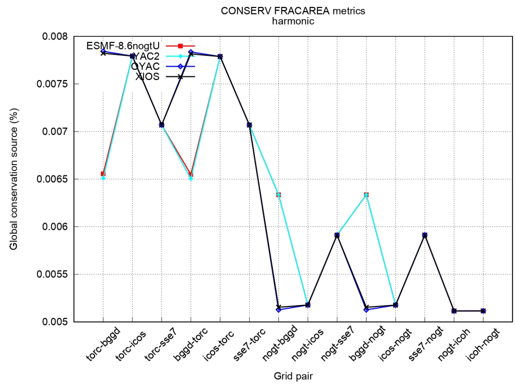

a) b) c) d)

**Figure 22** – Max misfit (%) of the 1st-Order conservative algorithm with Fracarea normalization for different grid pairs for the a) sinusoid, b) harmonic, c) vortex, d) gulfstream functions

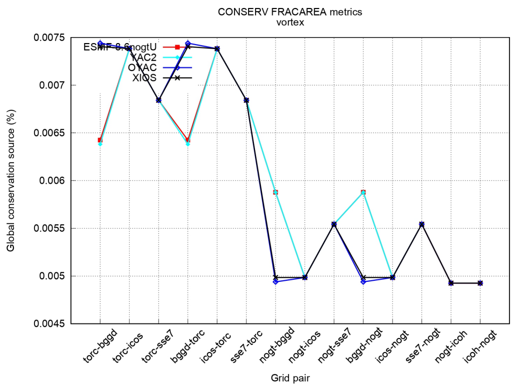
Fig. 22 shows the Max misfit for the 1st-Order conservative regridding with Fracarea normalization for the different grid pairs for the four functions. All regridders give almost exactly the same and very reasonable results.



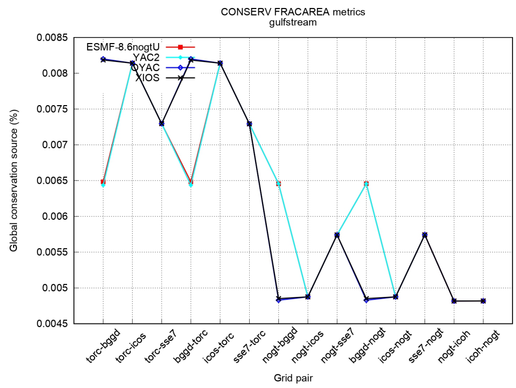

a)




b)




c)




d)

**Figure 23** – Conservation (%) of the 1st-Order conservative algorithm with Fracarea normalization for different grid pairs for the a) sinusoid, b) harmonic, c) vortex, d) gulfstream functions

Fig. 23 shows the Global conservation metric for the 1st-Order conservative regridding with Fracarea normalization for the different grid pairs for the four functions. The conservation calculation uses the areas supplied by the user (areas.nc file), multiplied by the unmasked fraction available in the masks.nc file (see Section 2).

All regridders for all functions present pretty reasonable results. We note better conservation with OYAC than with YAC2 for the nogt-bggd grid pair in both directions, for all functions.

### 5.3.6. Second-Order conservative Fracarea normalization

The 2$^{nd}$-Order conservative algorithm with Fracarea normalization is also presented for ESMF, YAC2, OYAC and XIOS regridders.

Fig. 24 and Fig. 25 show the Mean, RMS and Max misfit and Global conservation for the different grid pairs for these regridders, respectively with the harmonic function and for the gulfstream function. We see that all regridders show more or less the same behaviour with good global conservation, except always for ESMF when the source grid is the icosahedral grid (icos) which shows a relatively high mean misfit for both functions. There is a marked improvement in Mean and Max misfit for OYAC compared to YAC2 in most cases for both functions.

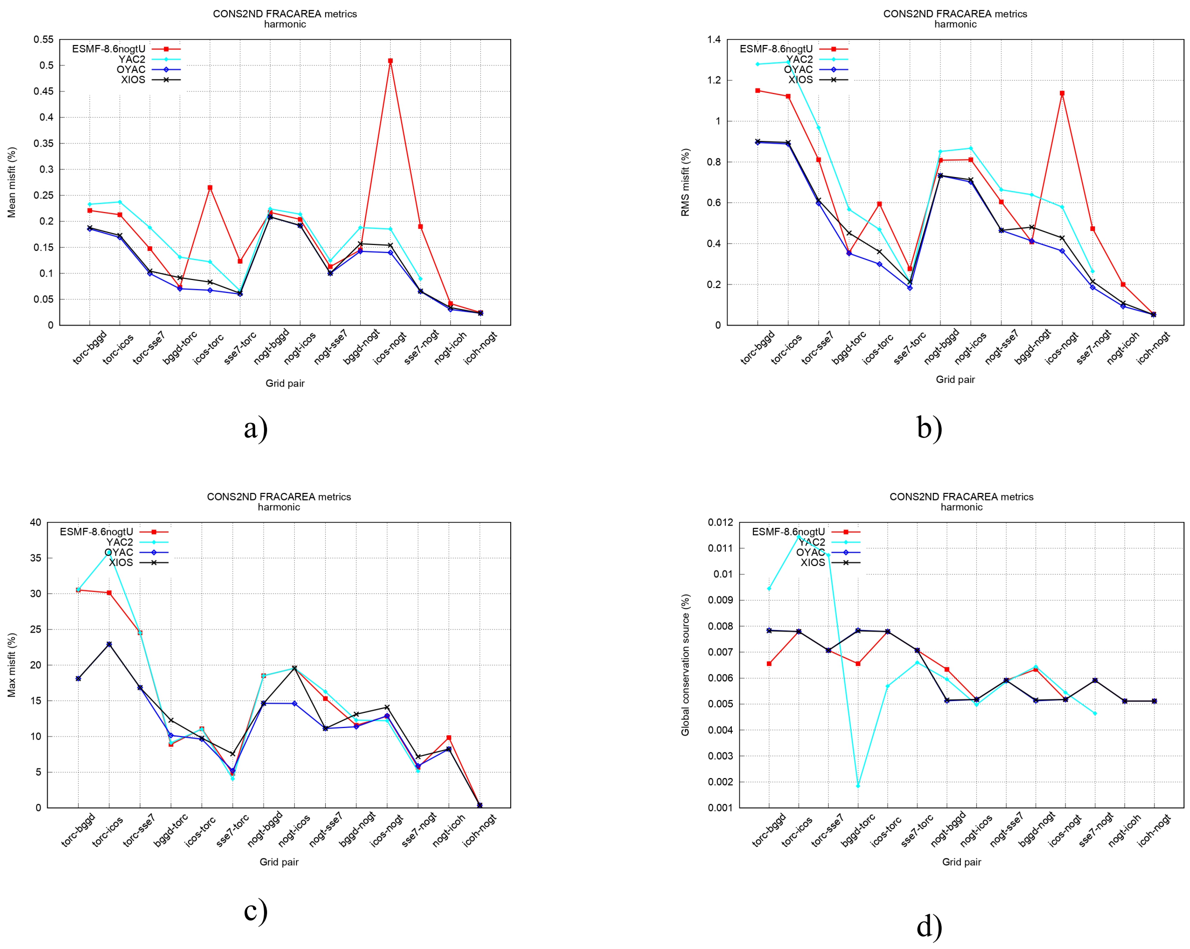



**Figure 24** – a) Mean, b) RMS and c) Max misfit and d) Global conservation of the 2$^{nd}$-Order conservative algorithm for different grid pairs with the harmonic function

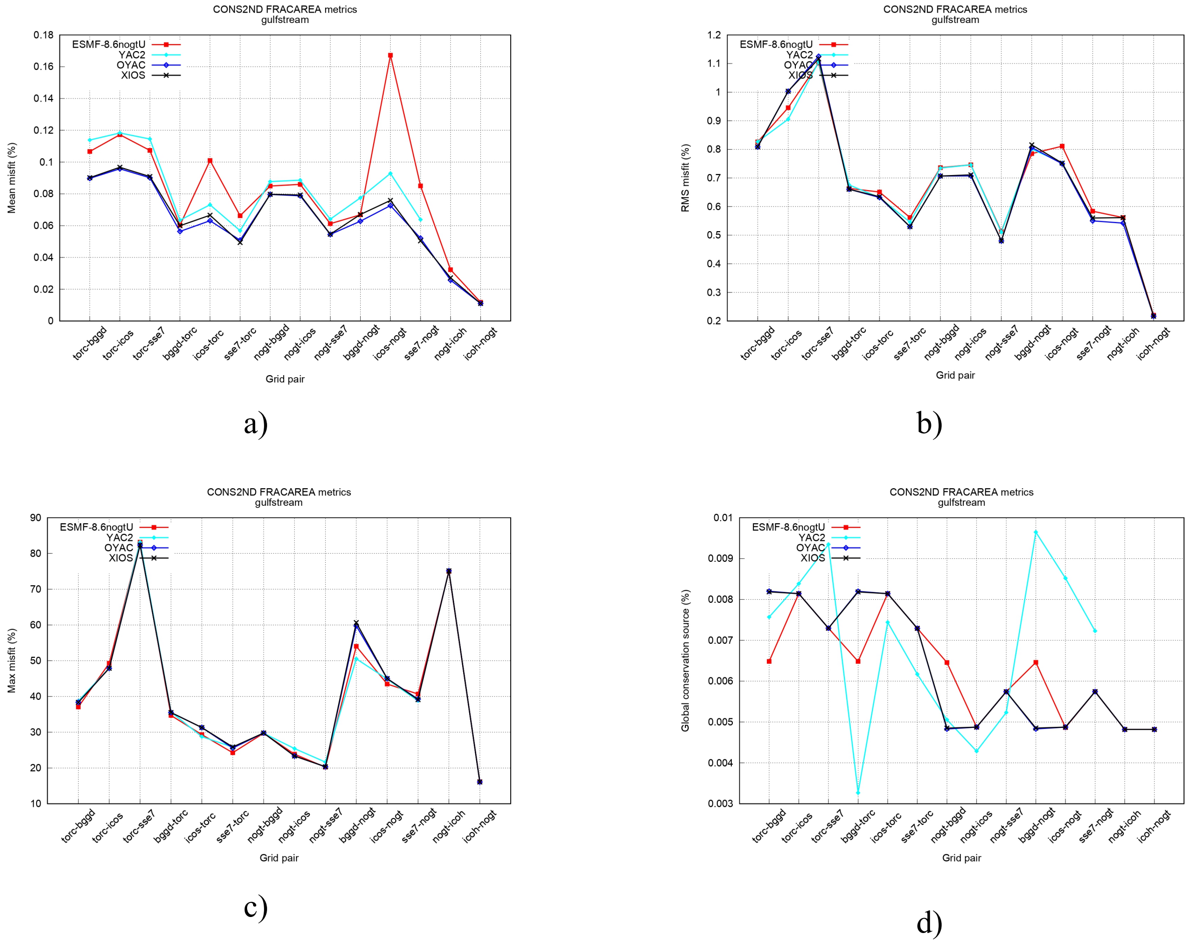
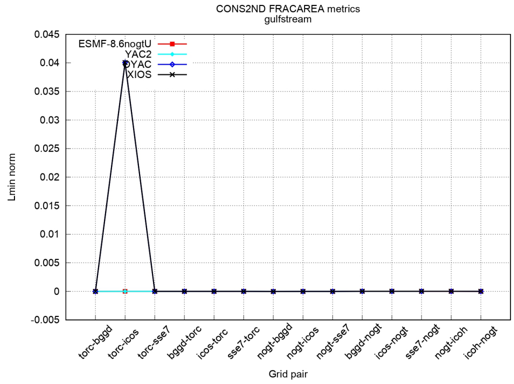
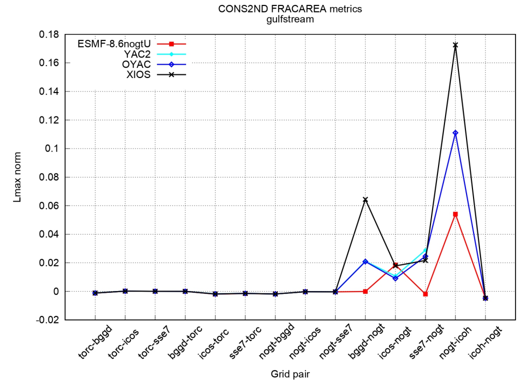



**Figure 25** – a) Mean, b) RMS and c) Max misfit and d) Global conservation of the 2nd-Order conservative algorithm for different grid pairs with the gulfstream function




a)




b)

**Figure 26** – a) Lmin and b) Lmax misfit of the 2nd-Order conservative algorithm for different grid pairs with the gulfstream function

Fig. 26 shows Lmin and Lmax for the 2nd-Order conservative remapping for the gulfstream function, which present some outstanding values (this is not the case with the other functions). The XIOS shoots were discussed in the 2021 benchmark (see Valcke et al. 2021). OYAC shows two medium-level overshoots, as for YAC2 in the 2021 benchmark but less important than XIOS.

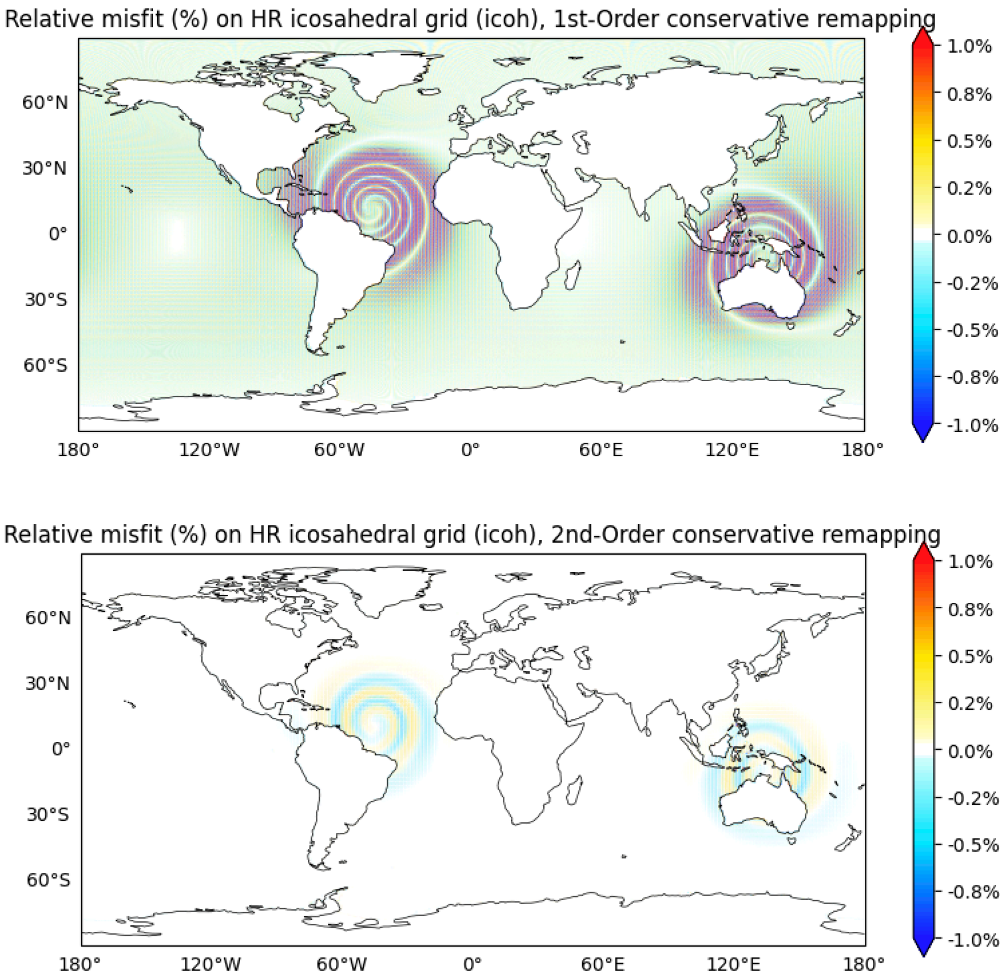



**Figure 27** – Relative misfit (%) on HR icosahedral grid (icoh), for 1st-Order conservative remapping (top) and 2nd-Order conservative remapping (bottom) from a low-resolution grid (nogt) of a vortex function defined field

Finally, Fig. 27 shows 2D plots of the relative misfit for the remapping of the vortex function from a low-resolution grid (nogt) to a high-resolution grid (icoh) with OYAC for the 1st-Order conservative algorithm (top) and for the 2nd-Order conservative algorithm (bottom). We see the clear benefit of the 2nd-Order as compared to the 1st-Order when this remapping involves two grids with very different resolutions. The 2nd-Order allowing the reconstitution of gradients on the higher-resolution target grid. The same conclusion can be drawn for YAC2 and XIOS but not for ESMF, see the discussion related Fig. 23 in Valcke et al. 2021.

This behaviour is very similar results to YAC2 for the case icos->icoh observed in the 2021 benchmark (see Fig. 23 in Valcke et al. 2021).

# 6. Performance of YAC weight calculation

A first study of the performance of the weight calculation shows that YAC is performing much better than the SCRIP.

Some performances of YAC[8] in calculating weights have been measured and compared with SCRIP on one regridding case. They are obtained by activating the writing of time statistics in OASIS3-MCT (the third number of `NLOGPRT` in OASIS *namcouple*) and setting the `local_timers_on` variables to `True` and 2 in the `mod_oasis_map` and `mod_oasis_yac_map` modules respectively. The regridding is the 1st-Order conservative remapping from the NEMO ORCA12 source grid with 3,145x4,322 grid points to the icosahedral grid icos with 15,212 grid points.
The number of cores ranges from 1 to 64. For YAC, the number of I/O MPI processes (*npio* in OASIS *namcouple*, see examples in Section 2.2.1) is systematically set to the number of MPI processes.

**Table 1** – Weight calculation time (sec) by SCRIP and YAC (`cpl_genmap` counter)

| N cores | 1 | 2 | 4 | 8 | 16 | 32 | 64 |
|---|---|---|---|---|---|---|---|
| SCRIP | 883 | 467 | 243 | 131 | 76 | 59 | 43 |
| YAC | 107 | 77 | 57 | 34 | 17 | 12 | 7 |

# 7. Conclusions

In summary, OYAC shows good results for all cases tested (except for two specific ones discussed with Fig. 19c and Fig. 26b, still to be investigated), very similar to those of YAC2, with improvements, in particular for the equivalent bilinear method and for the 2nd-Order conservative algorithm. We can therefore be confident about the quality of the integration of YAC into OASIS3-MCT.

[8] The version of YAC used for these performance calculations is `release-3.6.2`.

**Table 2** – OYAC regridding interpolation stack involving torc grid

| **Grid pairs** | **torc-bggd** | **torc-icos** | **torc-sse7** | **bggd-torc** | **icos-torc** | **sse7-torc** |
|---|---|---|---|---|---|---|
| Total unmasked destination cells | 14324 | 11468 | 17987 | 16501 | 16501 | 16501 |
| **AVGBARY** | Residual destination cells after each method | | | | | |
| method 01: AVG BARY | 1760 | 1452 | 2017 | 187 | 85 | 154 |
| method 02: NCC DIST | 924 | 772 | 1001 | 0 | 0 | 0 |
| method 03: NNN AVG 1 | 0 | 0 | 0 | | | |
| **HCSBB** | Residual destination cells after each method | | | | | |
| method 01: HCSBB | 1758 | 1452 | 2016 | 187 | 85 | 154 |
| method 02: NCC DIST | 924 | 772 | 1001 | 0 | 0 | 0 |
| method 03: NNN AVG 1 | 0 | 0 | 0 | | | |
| **DISTWGT** | Residual destination cells after each method | | | | | |
| method 01: NNN AVG 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **CONSERV_DESTAREA** | Residual destination cells after each method | | | | | |
| method 01: CONSERV FIRST DESTAREA | 0 | 0 | 0 | 0 | 0 | 0 |
| **CONSERV_FRACAREA** | Residual destination cells after each method | | | | | |
| method 01: CONSERV FIRST FRACAREA | 0 | 0 | 0 | 0 | 0 | 0 |
| **CONS2ND_FRACAREA** | Residual destination cells after each method | | | | | |
| method 01: CONSERV SECOND FRACAREA | 0 | 0 | 0 | 0 | 0 | 0 |

**Table 3** – OYAC regridding interpolation stack involving nogt grid

| **Grid pairs** | **nogt-bggd** | **nogt-icos** | **nogt-sse7** | **bggd-nogt** | **icos-nogt** | **sse7-nogt** | **nogt-icoh** | **icoh-nogt** |
|---|---|---|---|---|---|---|---|---|
| Total unmasked destination cells | 14579 | 11588 | 18282 | 65087 | 65087 | 65087 | 1443881 | 65087 |
| **AVGBARY** | Residual destination cells after each method | | | | | | | |
| method 01: AVG BARY | 1421 | 1166 | 1681 | 728 | 380 | 647 | 62141 | 0 |
| method 02: NCC DIST | 959 | 769 | 1019 | 0 | 0 | 0 | 11646 | |
| method 03: NNN AVG 1 | 0 | 0 | 0 | | | | 0 | |
| **HCSBB** | Residual destination cells after each method | | | | | | | |
| method 01: HCSBB | 1418 | 1166 | 1681 | 725 | 380 | 647 | 62140 | 0 |
| method 02: NCC DIST | 959 | 769 | 1019 | 0 | 0 | 0 | 11646 | |
| method 03: NNN AVG 1 | 0 | 0 | 0 | | | | 0 | |
| **DISTWGT** | Residual destination cells after each method | | | | | | | |
| method 01: NNN AVG 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **CONSERV_DESTAREA** | Residual destination cells after each method | | | | | | | |
| method 01: CONSERV FIRST DESTAREA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **CONSERV_FRACAREA** | Residual destination cells after each method | | | | | | | |
| method 01: CONSERV FIRST FRACAREA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **CONS2ND_FRACAREA** | Residual destination cells after each method | | | | | | | |
| method 01: CONSERV SECOND FRACAREA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

REFERENCES


Valcke, S., Piacentini, A. and Jonville, G. 2021: Benchmarking of regridding libraries used in Earth System Modelling SCRIP, YAC, ESMF and XIOS, Technical Report TR/CMGC/21-145, Cerfacs, France, https://cerfacs.fr/wp-content/uploads/2021/11/Globc_TR_Valcke_TR_CMGC_21_145_regridding_analysis_final.pdf.

Valcke, S., Craig, A., Maisonnave, E. and Coquart, L. (2021), *OASIS3-MCT User Guide, OASIS3-MCT 5.0*, CECI, Université de Toulouse, CNRS, CERFACS, Toulouse, France - TR-CMGC-21-161, Technical report, https://cerfacs.fr/wp-content/uploads/2022/03/GLOBC_TR_Valcke_oasis3mct50_2021.pdf.

Reinheimer, J., 2018: Vector Field Interpolation using Radial Basis Functions, pp 68.

Alfeld, P., Neamtu, M., Schumaker, L. L., 1996: Bernstein-Bézier polynomials on spheres and sphere-like surfaces, Computer Aided Geometric Design, https://doi.org/10.1016/0167-8396(95)00030-5.

Zienkiewicz, O. C., Zhu, J. Z., 1995: Superconvergence and the superconvergent patch recovery, Finite Elements in Analysis and Design, https://doi.org/10.1016/0168-874X(94)00054-J.

Jones, P. W., 1999: First- and Second-Order Conservative Remapping Schemes for Grids in Spherical Coordinates. *Mon. Wea. Rev.*, **127**, 2204–2210, https://doi.org/10.1175/1520-0493(1999)127<2204:FASOCR>2.0.CO;2.

Kritsikis, E., Aechtner, M., Meurdesoif, Y., and Dubos, T., 2017: Conservative interpolation between general spherical meshes, Geosci. Model Dev., 10, 425–431, https://doi.org/10.5194/gmd-10-425-2017.

Jones, P., 1998, A User's Guide for SCRIP: A Spherical Coordinate Remapping and Interpolation Package, Version 1.4, https://www.researchgate.net/publication/258437800_A_User's_Guide_for_SCRIP_A_Spherical_Coordinate_Remapping_and_Interpolation_Package_Version_14.

Kara, A. Birol, Alan J. Wallcraft, and Harley E. Hurlburt., 2007: A Correction for Land Contamination of Atmospheric Variables near Land–Sea Boundaries, Journal of Physical Oceanography 37.4: 803-818, https://doi.org/10.1175/JPO2984.1.