https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

YAC v3.19.0: scalable online interpolation and flexible coupling for Earth system models

Moritz Hanke[1], Nils-Arne Dreier[1], René Redler[2], and Andrea Piacentini[3]

[1]Deutsches Klimarechenzentrum, Hamburg, Germany
[2]Max-Planck-Institut für Meteorologie, Hamburg, Germany
[3]Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique, Toulouse, France

Correspondence: Moritz Hanke (hanke@dkrz.de)

Abstract. Earth system models increasingly couple many components on heterogeneous grids at rising spatial resolution, which raises requirements on coupling software for scalability, interpolation quality, workflow integration, and reliable orchestration. We present the transition to YAC 3, described here for release v3.19.0, and summarise advances that address these requirements in production-relevant settings. First, YAC 3 updates its parallel online weight-generation workflow, enabling scalable setup for very-high-resolution coupled configurations and large MPI process counts. Second, interpolation-related capabilities have been extended and refined, including improvements to conservative remapping and new methods such as Nearest-Corner-Cells, Distance-Nearest-Neighbour, and Creep Fill. Third, workflow support is broadened through Python bindings, YAML-based configuration, and tooling for weight and grid inspection. Fourth, we introduce a generic MPI handshake mechanism that makes communicator construction in multi-component, multi-library setups more reliable, and propose it as a convention for other libraries and components. Fifth, YAC is separated into a self-contained core library and a component-oriented model coupling interface, which makes its interpolation stack reusable by software outside the classical model coupling context. Use cases from coupled ICON simulations (including kilometre-scale configurations), input/output server coupling, and direct integration in other software ecosystems demonstrate that YAC 3 supports a broad range of contemporary and emerging Earth system workflows.

1 Introduction

Starting with the pioneering work of Manabe and Bryan (1969), climate models have evolved into Earth system models (ESMs), now embedding a wide variety of physical components to simulate interactions among them. In parallel with advances in computing power, scientists have increased spatial resolution from several hundred kilometres to the kilometre scale (Hohenegger et al., 2023).

ESMs are inherently multi-physics models and are developed in a multidisciplinary way. Typically, individual model components, such as the atmosphere, ocean, ocean biogeochemistry, ocean waves, sea ice, land, hydrology, and others, are developed by scientists and specialists in their respective domains. These components require boundary conditions to drive the internal processes. In order to support efficient computation at high spatial resolution on modern supercomputing architectures, parallel programming follows the Message Passing Interface (MPI) paradigm, typically with parallelisation over the horizontal domain.

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

To allow close interaction during integration, the various components are coupled to exchange boundary conditions. For some components, this can be achieved by simple subroutine calls, in particular when the involved components run with identical spatial resolution and domain decomposition. Otherwise, an intermediate software layer, or coupling software, is required to handle interpolation and repartitioning of data that must be delivered to different numerical grids and/or MPI processes.

To handle this data exchange efficiently in ESMs employing thousands of MPI processes, coupling software must provide a sufficient degree of parallelism, both during the initial setup phase, which usually involves generating or reading interpolation weights from file, and during boundary data exchange in the model's integration time loop. To ease the assembly of external model components into an ESM, the coupling software must also provide sufficient flexibility for developers, including support for so-called MPI communicator splitting across groups of MPI processes. The physical fidelity of a coupled simulation depends on how the exchanged fields are remapped, so the coupling software must further provide interpolation methods that suit the quantity in question—conservative remapping for fluxes, for instance—and that remain accurate under the conditions found in real model configurations, such as masked regions, mismatched land-sea boundaries between components, and strongly varying grid resolution. Finally, model components must be able to run concurrently on dedicated resources, each of them internally parallelised over its own set of processes, which is a capability that established coupling solutions have provided for many years (Craig et al., 2017; Collins et al., 2005; Arakawa et al., 2020). What remains demanding at kilometre-scale resolution is therefore not the execution model itself, but the degree of parallelism required in the setup phase and in the data exchange.

With increasing resolution, the initial setup phase of coupling software can become problematic. Efficient generation of interpolation weights during this phase is often limited by time and memory constraints. Similar issues arise when interpolation weights are read from file, because the preprocessing step needed to provide these files increases the complexity of running a model (Moreno-Chamarro et al., 2024; Mogensen et al., 2023; Fladrich and Maisonnave, 2023).

The main contributions of this paper are as follows. First, we present advances in YAC 3, including a fully distributed parallelisation scheme for online interpolation weight computation that enables efficient coupling at extreme grid resolutions and large MPI process counts. Second, we describe methodological improvements and new interpolation techniques that enhance both the accuracy and versatility of regridding operations, validated through recent benchmark studies. Third, we introduce newly added workflow features, such as Python bindings and YAML-based configuration files, which can facilitate adoption in modern ESM environments. Fourth, we propose and implement a generic handshake mechanism for more reliable multi-component orchestration, addressing a key challenge in coupled model set-ups. Fifth, we report on the separation of YAC into a reusable Core library and a component-oriented Model Coupling Interface, which makes its interpolation stack available to software outside the classical model coupling context. Finally, we demonstrate the practical impact of these developments through real-world use cases, spanning coupled ICON simulations from the kilometre scale to century-long climate projections, the use of YAC as the communication layer for input and output server components, and the direct reuse of its interpolation capabilities by other software, namely OASIS3-MCT and UXarray.

The remainder of this paper is structured as follows. Section 2 provides background on existing coupling software within the Earth system modelling community and situates YAC within this landscape. This is followed by four sections that introduce

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

major challenges faced by coupling software and describe how YAC 3 has improved in these areas compared to YAC 1. These challenges are:

- Scalability to extreme resolution and large MPI process counts
- Accuracy and versatility of interpolation methods
- Integration of coupling software into modern workflows
- Reliable multi-component orchestration

In Section 7, multiple use cases are presented to demonstrate YAC's capabilities. Section 8 summarises the contributions, and the code and data availability statements provide the information needed for reproducibility.

2 Background and positioning

A variety of software solutions for the coupling of Earth system models exist. For a more exhaustive introduction to this topic, we refer the reader to Valcke et al. (2012) and the literature cited therein. We note here that new coupling software has emerged and existing solutions have been further improved.

OASIS3 has evolved into OASIS3-MCT (Craig et al., 2017), the version of the OASIS coupler interfaced with the Model Coupling Toolkit (MCT). Its latest release, OASIS3-MCT_6.0 (Valcke et al., 2025), provides support for parallel online weight computation, bindings for C/C++ and Python, a dedicated locally conservative interpolation method for runoff fields, and optional normalisation for conservative interpolation using user-provided cell areas.

The Earth System Modeling Framework (Collins et al., 2005) is a widely used coupling framework. It has a large user base, high-quality parallel online weight computation capabilities, and many other practical features. A major difference between ESMF and coupling libraries like OASIS3-MCT and YAC is that with ESMF, components are integrated into a coupling framework rather than using a coupling library. This distinction affects component development, where teams may choose one approach or the other.

Arakawa et al. (2020) introduce Jcup3, which is mainly used by the Japanese ESM community. In contrast to, e.g., OASIS3-MCT, Jcup3 does not provide any interpolation methods but offers an interface for application programmers (or users of Jcup3) to provide their own interpolation methods.

Liu et al. (2023) describe C-Coupler, now available as version 3.0 and targeted at the needs of the Chinese ESM community. In version 3.0, this classical coupling software has evolved into a module-integration framework, in particular to support integration of data assimilation frameworks into coupled model systems and to provide support for halo exchange. For its implementation of parallel online weight computation, it uses algorithms similar to those in YAC 3.

The first official version of YAC was released as version 1.2 in 2015 (Hanke et al., 2016). It already provided efficient parallel online interpolation weight calculation capabilities. However, especially for very high-resolution simulations, it had significant limitations. From the beginning, YAC has been developed as an independent library for parallel online data exchange and

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

interpolation. As a distinguishing feature at that time, YAC 1.2 introduced the concept of interpolation stacks, which allow users to freely configure a sequence of interpolation methods from the full set of available methods. Target points that cannot be interpolated by one method are passed to the next, which makes interpolation configuration much more flexible (see Hanke et al., 2016, for details). Starting with version 1.2, it has been used to couple model components in ICON. Later versions of YAC have already been used for global coupled ICON simulations at the kilometre scale (Hohenegger et al., 2023) and in the context of projects like nextGEMS (Segura et al., 2025), WarmWorld[1], and DestinE (Doblas-Reyes et al., 2025).

This paper presents YAC v3.19.0. Relative to the version described by Hanke et al. (2016), it broadens the set of interpolation methods, improves scalability of the setup phase, and adds interfaces that allow its interpolation stack to be reused outside classical model coupling. The remainder of the paper describes these developments and reports the performance of the resulting implementation.

3 Challenge 1 — scalable coupling at extreme resolution

The basis for parallel online remapping weight computation is an efficient parallelisation scheme. It must evenly distribute work across available resources, and memory consumption must scale with the number of processes.

YAC 1 largely achieved these goals (Hanke et al., 2016) but had important limitations. It depended on halo information provided by the user, restricting use to components able to supply it—a point raised in review (Anonymous Referee #2, 2016) of the original paper (Hanke et al., 2016). In addition, weight computation ran only on the source component's processes. When the source used fewer processes than the target, or when the target grid had higher resolution, this led to inefficient resource use. Finally, the parallelisation scheme's complexity and non-deterministic behaviour made development and debugging difficult.

To address these issues, the parallelisation scheme was rewritten for YAC 2 (released in 2021). As in YAC 1, each process provides grid information for its own domain-decomposed subdomain. The new scheme employs a rendezvous algorithm using a distributed directory (Pinar and Hendrickson, 2001; Plimpton et al., 2004), an approach also applied by Liu et al. (2023). For each coupled grid pair, YAC first builds an internal decomposition using the sphere part algorithm (Hanke and Redler, 2019), which operates similarly to a k-d tree on the surface of the sphere. It partitions the globe so that all processes receive roughly the same total number of source and target cells. Where grid density varies (e.g., local refinement or polar singularities), the partitions adapt to keep the cell counts per process similar. YAC then redistributes user-provided grid data according to this decomposition, forming the distributed grid pair used for weight computation.

Each process computes interpolation weights for the target points assigned to it by the internal decomposition. This design has three key advantages. First, each process holds only the source and target data of its own partition, so memory consumption scales with the number of processes, including in cases where process counts or grid sizes differ substantially between components. Second, when both grids cover approximately the same area on the sphere, all processes participate in weight computation, balancing the workload. Third, by design, the communication between processes is minimal, as the required source points for a target point are typically in close proximity and hence mostly available locally.

[1] https://www.warmworld.de/, last access: 18 May 2026

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

The distributed grid pair is encapsulated in a data structure that exposes query interfaces for interpolation methods. One such interface is a nearest neighbour search executed collectively by all processes. It returns references to data available locally; when needed, YAC augments the local partition with neighbouring grid entities, caching a small amount of ghost data. Subsequent queries can reuse this cache, reducing inter-process communication and improving latency. Consequently, interpolation methods operate on locally available data without explicit communication calls.

This approach removes the need for user-provided halo information and, during coupling setup, distributes the workload—distributed grid generation and weight computation—across all processes in the coupling, resolving the YAC 1 limitation where only source-component processes performed the weight computation.

3.1 Performance

Scalability and performance measurements illustrating the distributed grid-pair approach are shown in Figure 1. We used two synthetic benchmark components, each running with the same number of processes: one using a global icosahedral grid and the other a cubed-sphere grid. In the test setup, YAC was configured to compute the weights for the remapping from the icosahedral grid to the cubed-sphere grid. The reported times cover the generation of the distributed grid pair, the weight computation, and the setup of all internal data structures required for interpolation. The exchange of coupling fields during the model integration is not re-examined here, as it has not changed significantly with respect to the versions measured in Hanke et al. (2016) and Hanke and Redler (2019), where the time required for a ping-pong exchange was reported using the same benchmark programs. A direct quantitative comparison with YAC 1 and YAC 2 is likewise not provided: those versions were assessed on systems that have since been decommissioned, and absolute timings obtained on that hardware are not comparable to the measurements reported here. The results below therefore characterise the current implementation rather than a relative speed-up, while the advances over YAC 1 described above are structural, in particular the removal of the halo requirement and the involvement of all processes in the weight computation. The benchmark runs also record the total memory used per job, but this figure includes the memory required to read and hold the input grids and therefore does not isolate the coupling-related footprint; it is consequently not analysed here. Measurements were performed on the Levante system at the German Climate Computing Center (DKRZ), whose nodes have two AMD EPYC 7763 64-core CPUs and are connected via Mellanox InfiniBand HDR100 fabric. Benchmark programs were compiled with GCC 11.2.0 and linked against Open MPI 4.1.2. All measurements were performed with YAC v3.19.0. Every configuration was repeated three times; the reported values are averages over these repetitions. The measurements, the raw job output and the scripts that produced Figure 1 are archived (see the code and data availability statements).

We varied several parameters. We tested four grid configurations, spanning roughly one million to 335 million cells per globally defined grid (Table 1), representative of high and very high horizontal resolutions of around 1.25 to 10 km. We evaluated three commonly used interpolation methods: 4-Nearest-Neighbour (4NN), first-order conservative, and Hybrid Cubic Spherical Bernstein-Bézier patch (HCSBB; Alfeld et al., 1996). Total process counts ranged from 512 to 32,768 (4 to 256 nodes). For the highest resolutions at four nodes, HCSBB's memory footprint was too high, so those values were omitted; in practice, such low node counts are not used for very high-resolution coupled runs, so this is not a usability limitation.

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

Grid type	Resolution	Number of cells	Approx. hor. res.
ICON	R02B08	5,242,880	10 km
	R02B09	20,971,520	5 km
	R02B10	83,886,080	2.5 km
	R02B11	335,544,320	1.25 km
CUBE	C450	1,215,000	20 km
	C900	4,860,000	10 km
	C1800	19,440,000	5 km
	C3600	77,760,000	2.5 km

Table 1. Grid configurations used for the scalability measurements. The approximate horizontal resolution is given as the square root of the mean cell area on a sphere of radius 6371 km.

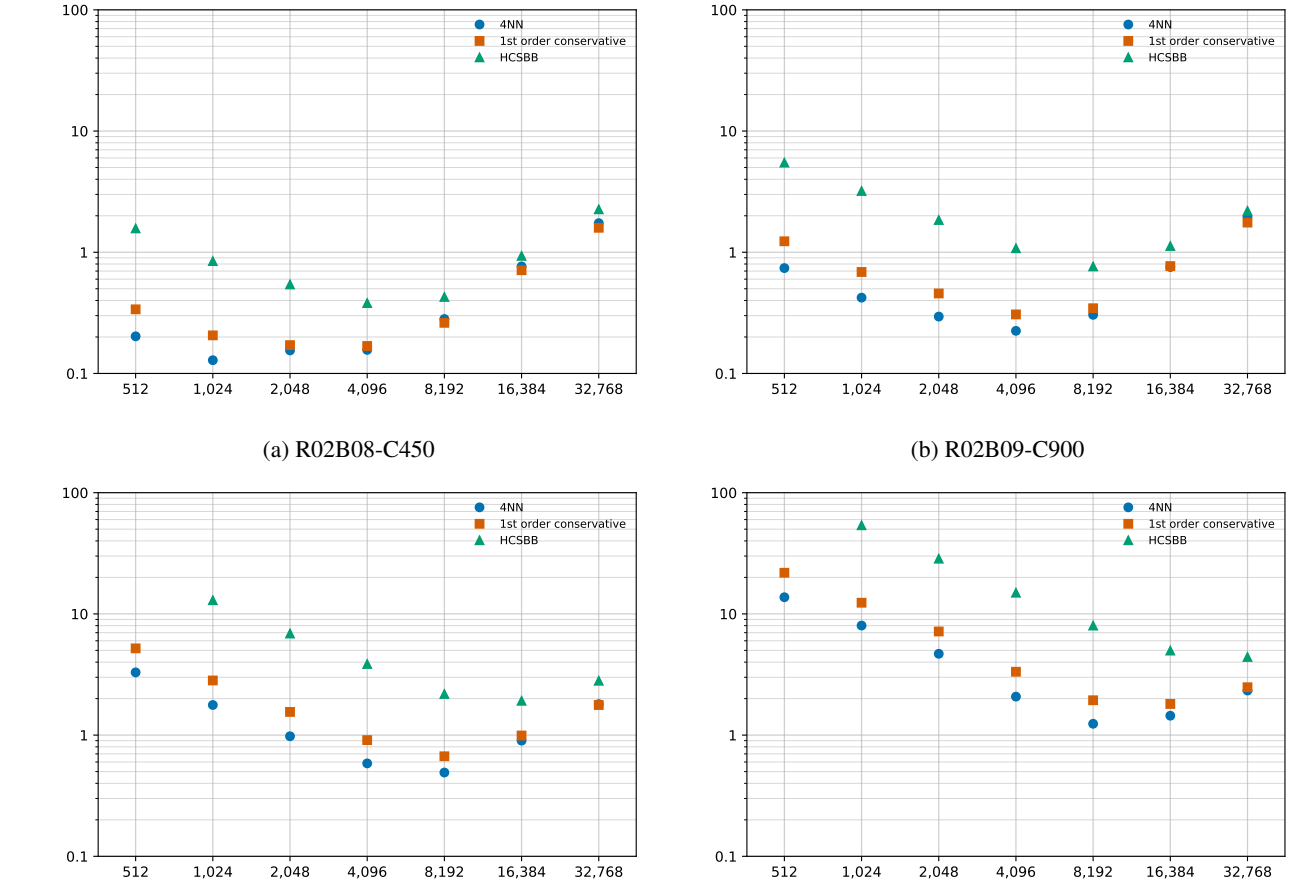
Two factors dominate runtime across configurations. The first is the actual computation of interpolation weights, which shows favourable scaling with process count in the evaluated range. Therefore, differences between interpolation methods are most pronounced at lower process counts. Consequently, at high process counts, runtimes converge across interpolation methods, as is especially evident in Figure 1a. The second and most impactful factor is the distributed grid generation, which involves significant communication. This also scales well with process count up to a certain threshold, beyond which runtime begins to increase. The exact threshold depends on problem size; in Figure 1a the setup time is lowest between about 1,024 and 4,096 processes and increases beyond that range. However, for practical applications the total runtime remains within reasonable bounds of a few seconds. Moreover, the recent move to GPU-based systems results in reduced process counts per run, which further mitigates the runtime increase at high process counts.

The three interpolation methods differ significantly in their computational intensity. While 4NN is relatively inexpensive, the HCSBB method is more complex, requiring significantly more computation and communication, which is why it is up to about eight times slower than 4NN at the lowest process counts. This difference shrinks as the process count grows, until the three methods differ by less than a factor of two at 32,768 processes. The nearly linear decrease in runtime for grid configurations with high workload indicates good scaling of the weight computation; it holds up to about 4,096 processes in Figure 1c and up to about 8,192 processes in Figure 1d.

3.2 Scalability summary and key messages

The results presented above show that YAC addresses the challenge of scalability to extreme resolution and large MPI process counts. The distributed grid-pair approach enables scalable online weight computation for coupled grids of varying resolution while removing key limitations of previous YAC versions, in particular the need for user-provided halo information and the restriction to source-side processes. By construction, memory and work are distributed across all participating processes, which keeps both within practical limits for the configurations relevant to current CPU- and GPU-oriented production systems. Although the communication-heavy setup phase can become a bottleneck at very high process counts, total setup times remain

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.



(a) R02B08-C450

(b) R02B09-C900

(c) R02B10-C1800

(d) R02B11-C3600

Figure 1. Setup time for the regridding from an icosahedral to a cubed-sphere grid, for the four grid pairs of Table 1. Each panel shows the time for distributed grid-pair generation, weight computation and interpolation setup (y-axis, seconds) against the total number of processes (x-axis), both on logarithmic scales, for the three interpolation methods: 4NN (circles), first-order conservative (squares) and HCSBB (triangles). Each symbol is the average of three repetitions. A missing symbol marks a configuration that could not be run; these are HCSBB at four nodes for the two highest resolutions.

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

acceptable for practical applications. Runtime also depends strongly on interpolation method, with computationally intensive methods such as HCSBB substantially slower than simpler methods such as 4NN, while high-workload cases still show near-linear scaling of the weight computation.

While the weight computation must be performed for each unique interpolation configuration, the distributed grid pairs need to be computed only once for each coupled grid configuration in a setup. They are then reused for interpolation configurations involving the same grid pair, which is commonly the case.

The efficient online weight generation allows for flexible coupling configurations by eliminating the need for pre-computed weights, which would otherwise have to be generated and managed for each unique interpolation and grid setup—a challenging task for high-resolution grids (Moreno-Chamarro et al., 2024). This online approach simplifies the setup process, as users do not need to handle weight files and can adjust interpolation configurations without an additional preparation step. It also reduces disk space requirements, which can be substantial for large grids and multiple coupling scenarios.

4 Challenge 2 — accurate and versatile interpolation

YAC's online weight computation capabilities are among its notable features. Numerous improvements and new developments in this area have been introduced with YAC 2 and YAC 3, addressing both methodological enhancements and emerging application requirements.

Equally important as feature availability is the quality of generated interpolation weights. A benchmark study by Valcke et al. (2022) compared multiple regridding libraries (including YAC v2.3.0) and concluded that YAC performed well in all test cases, producing high-quality weights. Based on these benchmark results, additional refinements were implemented that led to further quality improvements, particularly for second-order conservative remapping. These enhancements were validated in a follow-up study conducted by the Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique (CERFACS) (Jonville et al., 2024). Furthermore, Taylor (2024) demonstrated that accurate cell shape representation is critical for interpolation quality, especially for conservative remapping. The study highlighted that YAC supports cell edges following great circles and, in contrast to most other coupling libraries, also circles of constant longitude and latitude, which enables accurate representation of cell shapes like those in regular Gaussian grids.

This paper focuses on the availability and design of YAC's interpolation methods and the coupling scenarios they address; a comprehensive, per-method quality assessment is beyond its scope. Some evaluations of individual methods are available in the literature: besides the conservative remapping improvements noted above, the study by Jonville et al. (2024) also assessed the quality of the barycentric coordinate weighting option of the Average method (Section 4.1.1) and of the NCC fallback (Section 4.2.1). For several of the newer methods, in particular Source Point Mapping, DNN, and Creep Fill, interpolation quality is strongly application-dependent, and a meaningful assessment (for example, weighing an NNN against a Creep Fill fallback) requires evaluation within concrete user configurations. Such application-specific assessments are therefore left to the respective users of these methods.

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

Section 4.1 describes improvements to existing interpolation methods from YAC 1, while Section 4.2 presents new methods introduced in YAC 2 and YAC 3. Beyond algorithmic enhancements, YAC 3 introduces dynamic fractional masking for time-varying masks, improved support for unstructured longitude-latitude grids, and a new concept called "raw data exchange", each of which is described in the respective sections below.

4.1 Improvements

Multiple existing interpolation methods have been further improved and extended with additional configuration options and features. The most important changes are addressed in the following subsections.

4.1.1 Average

The average interpolation (AVG) method as described in Hanke et al. (2016) provides a simple interpolation with a compact stencil. For each target point, YAC identifies the source cell containing the target and constructs an interpolation stencil from the source values at the cell vertices. When the source field is defined on cell centres rather than vertices, YAC first generates the dual grid to enable this vertex-based interpolation. However, especially when interpolating from coarser to finer grids, an imprint of the source grid on the interpolated target field can be observed in the form of discontinuities along source grid edges. To address this, a new barycentric coordinate-based weighting option was introduced. For this method to work, the source grid is first triangulated. Then for each target point, the matching source triangle is identified and its barycentric coordinates within this triangle are computed. These coordinates are then used as weights. By construction, barycentric coordinate-based interpolation produces a continuous target field for the triangulated source geometry. A drawback of this approach is that the interpolation stencil for each target point will always have a maximum of three source points, even when the matching source cell has more corners. In addition, source edges introduced by the triangulation may leave an additional imprint on the target field data. To overcome these limitations, support for barycentric coordinates on general spherical polygons is planned for future YAC versions.

4.1.2 Conservative

The previously mentioned benchmark study by Valcke et al. (2022) showed that the XML I/O Server (XIOS) library [2] generated slightly better interpolation weights for second-order conservative remapping. This could be attributed to a difference in the handling of target cells that are close to grid or mask boundaries of the source component. YAC now handles these cases similarly to XIOS, so that both libraries now produce nearly identical results.

As global grids advance towards sub-kilometre horizontal resolutions, accurate geometric calculations become increasingly critical, particularly for conservative remapping which relies on precise great circle distance and spherical polygon area computations. Following the methodology of Chen et al. (2026b), YAC has enhanced these calculations to improve numerical robustness at fine resolutions.

[2] http://forge.ipsl.jussieu.fr/ioserver/wiki, last access: 18 May 2026

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

4.1.3 Source Point Mapping

This method assigns each unmask source point to the closest target point. The target points receive the sum of the values of all source points assigned to them. The typical application for this method is the mapping of river discharge off from a hydrology model to the closest ocean point.

This technique can potentially trigger numerical instabilities in the ocean model, particularly at horizontal resolutions finer than 10 km. For example, river discharge from the Amazon delta may be mapped to only a few ocean cells, causing water to accumulate faster than it can be transported away. To mitigate this problem, YAC provides a spread distance option that distributes each source point's value across multiple nearby targets. For each source-target match, YAC identifies all target points within the specified spread distance around the initial target point and distributes the source value among them. Users can choose between different weighting options.

Another issue arises when a high-resolution hydrology model includes coastal areas which are not represented in the target ocean grid. For instance, if the hydrology model represents the Caspian Sea while a coarser ocean model does not, river discharge from the Caspian region may be incorrectly mapped to distant ocean cells in the Black Sea. To prevent such spurious mappings, YAC allows users to specify a maximum matching distance between source and target points. While this constraint prevents physically unrealistic couplings, users must still explicitly handle the conservation implications of unmapped discharge within their models.

For all source points that are successfully mapped to target points, the sum of the weights associated with each source point is one by default. This ensures field conservation. However, some models provide or expect river discharge as a flux rather than an integral quantity (Maisonnave, 2020). To accommodate this, YAC offers a scaling option that multiplies weights by the source cell area and/or divides by the target cell area. Area computations assume spherical cells; if source and target components use different sphere radii, these can be specified to YAC. When YAC's computed cell areas differ from those assumed by the coupled models, users may alternatively provide cell areas via external files.

By default, spread and maximum search distances are applied uniformly to all source points. However, YAC also supports spatially varying parameters through user-defined regions. Users can define spherical regions and assign custom distance values to all source cells within each region. This capability enables targeted configurations, such as applying larger spread distances near major river deltas while maintaining smaller values elsewhere. Similarly, when models do not explicitly simulate iceberg drift, an increased spread distance in Antarctic regions can better represent the transport of iceberg-derived freshwater into the open ocean.

4.2 New methods

While YAC 1 already supported most common interpolation use cases, several new methods have been implemented in YAC 2 and YAC 3 to address emerging requirements and enhance results for existing applications.

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

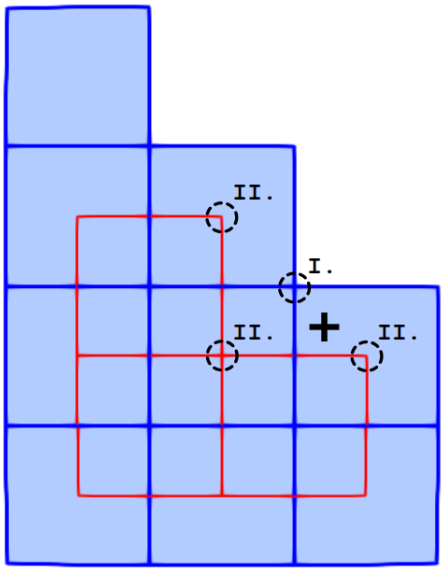


Figure 2. Artificial source grid (filled blue cells) with YAC-generated dual grid (empty red cells) overlaid with example target point (black cross), nearest source cell corner (I.), and associated source points (II.) used for NCC interpolation.

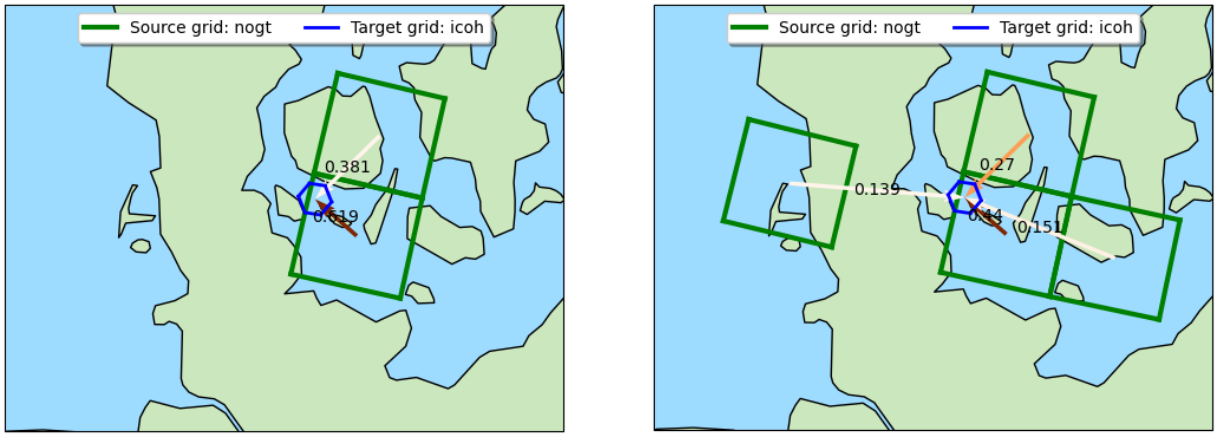
4.2.1 Nearest-Corner-Cells

When applying the AVG interpolation method to source fields defined at cell centres, YAC generates a dual grid where cell centres become vertices. However, this dual grid typically covers less of the sphere than the original grid, particularly along mask boundaries and grid edges where it cannot fully reach the original grid boundary. Consequently, target points near these boundaries may require fallback interpolation methods such as N-Nearest-Neighbour (NNN), which may not produce optimal results.

To address this limitation, the Nearest-Corner-Cells (NCC) method was implemented for cell-centre-based source data as an additional fallback method. Figure 2 illustrates the NCC algorithm using a simple artificial grid configuration. The filled blue cells represent the original source grid with cell-centre data, while the empty red cells show the dual grid generated by YAC. For a given target point (marked with a black cross), NCC first identifies the matching source cell and then determines the nearest corner of that cell (marked as I. in the figure). The algorithm constructs the interpolation stencil from all source cell centres surrounding this corner (marked as II.) using a user-specified weighting scheme. In this example, three source points contribute to the interpolation. This approach enables NCC to interpolate target points that would fall outside the dual grid coverage, particularly near grid boundaries and mask edges where the dual grid does not extend to the original grid's full extent.

Figure 3 compares interpolation stencils for a hexagonal target cell centre (blue) using the NCC and 4NN methods (Figure 3a and Figure 3b, respectively). Arrows indicate interpolation weights, with values labelled adjacent to each arrow. In this example, land masking is applied to the source ocean grid. The nearest corner to the target cell centre lies directly on the land-ocean

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.



(a) Nearest-Corner-Cells

(b) 4-Nearest-Neighbour

Figure 3. Interpolation stencils for a single hexagonal target cell (blue) on a source ocean grid with land masking, obtained with (a) Nearest-Corner-Cells and (b) 4-Nearest-Neighbour. Arrows point from the contributing source cells to the target cell and are labelled with the corresponding interpolation weights. In (a) the nearest corner lies on the land-ocean boundary, so only two unmasked source cells contribute, whereas (b) selects the four nearest source cell centres irrespective of cell connectivity and therefore draws on source points on either side of the land mass.

boundary, resulting in only two unmasked source cells adjacent to that corner that contribute to the NCC stencil. In contrast, the 4NN method selects the four nearest source cell centres without considering cell connectivity, resulting in source points located both east and west of the land mass. This non-local selection may compromise interpolation accuracy compared to the connectivity-aware NCC approach.

When target cell centres lie outside all unmasked source cells, NCC cannot provide an interpolation stencil. In such cases, an additional NNN fallback method may be required in the interpolation stack.

4.2.2 Distance-Nearest-Neighbour

The Distance-Nearest-Neighbour (DNN) interpolation method was introduced as an alternative to NNN. While NNN searches for a user-specified fixed number of nearest source points for each target point (constant stencil size, variable search distance), DNN searches within a specified distance around each target point (variable stencil size, fixed or adaptive search distance). This results in a variable number of source points depending on the local source point density, making DNN particularly suitable for grids with highly variable resolution or when adaptive stencil sizes based on target cell geometry are desired.

DNN supports two approaches for defining the search distance. The user can either provide a fixed distance for all target points, or YAC computes the search distance based on the target cell area. In the latter case, the distance equals the polar angle of a spherical cap with the same area as the target cell, multiplied by a user-defined scale factor.

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

DNN is particularly useful when the source point density varies significantly across the domain or when the interpolation should be based on physical distance rather than a fixed number of neighbours. Another application arises when target cell sizes vary and an adaptive search distance is required. There are also cases in which small-scale features from a high-resolution source field may interfere with the target component. Here DNN can be used to smooth the interpolated target field by selecting a search distance larger than these features. This can reduce high-frequency noise in the interpolated target field.

4.2.3 Creep Fill

Inspired by the Creep Fill method in ESMF (Collins et al., 2005), YAC now implements this extrapolation technique as well. The method assigns each non-interpolated target point the average value of its neighbouring target points that already possess valid interpolation stencils. This process iterates until either all target points have valid stencils or a user-specified maximum iteration count is reached.

Figure 4 depicts a synthetic source field over Canada and Greenland. Grey regions represent masked areas where the source grid provides no data. The figure is overlaid with a target grid. To demonstrate the Creep Fill functionality, the source field was remapped to a target grid without any mask. The interpolation results using four different interpolation stacks are shown in Figure 5, where dark grey areas indicate target cells that receive no data. First-order conservative interpolation was used as the primary method in all stacks. Figure 5a shows results without any fallback method. Figure 5b and Figure 5c employ Creep Fill with maximum iteration counts of 16 and unlimited, respectively. For comparison, Figure 5d shows results using a 1-Nearest-Neighbour (1NN) fallback.

Without any fallback, coastal regions and features such as Hudson Bay receive no data because these areas are absent in the source field. With limited iterations, Creep Fill successfully populates coastal cells but cannot reach Hudson Bay. Using unlimited iterations for Creep Fill or applying 1NN enables YAC to assign data to the entire target field. The key difference is that Creep Fill exploits cell connectivity information for extrapolation while 1NN does not, accounting for the differences between Figure 5c and Figure 5d.

4.2.4 User Callback

When applications require specialised interpolation methods unavailable in YAC, or when existing methods lack the needed generality, users can compute weights externally. Traditionally, such weights are provided via weight files that YAC reads during initialisation. YAC now also supports runtime weight computation through user-provided callback functions via the "User Callback" interpolation method.

For each target point, YAC first identifies the containing source cell, then invokes the user-provided callback function on the source process that owns that cell. The callback receives the target point coordinates and source cell identifier as input, computes the interpolation weights, and returns both the weights and their associated source points.

An initial application of this method enabled a finite element model to couple with YAC for output to a finite volume component. The callback directly computed weights that remap source fields, stored as discontinuous Galerkin coefficients, onto the Gaussian output grid.

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.



Figure 4. Synthetic source field over Canada and Greenland, overlaid with the target grid. Grey marks masked areas in which the source grid provides no data.

4.2.5 Zero

YAC's fixed interpolation method assigns a user-defined value to all target points and is often used as the final fallback in an interpolation stack. However, weight files generated with this method may be incompatible with other coupling software, as user-defined fixed values are not a common feature.

As a workaround, the Zero method selects a single source point for all target points and assigns weights with value zero. This provides compatibility with other coupling software while setting all remaining target points to zero as the final fallback in an interpolation stack. Weight files generated using this method can be read by other coupling libraries that do not support user-defined fixed values.

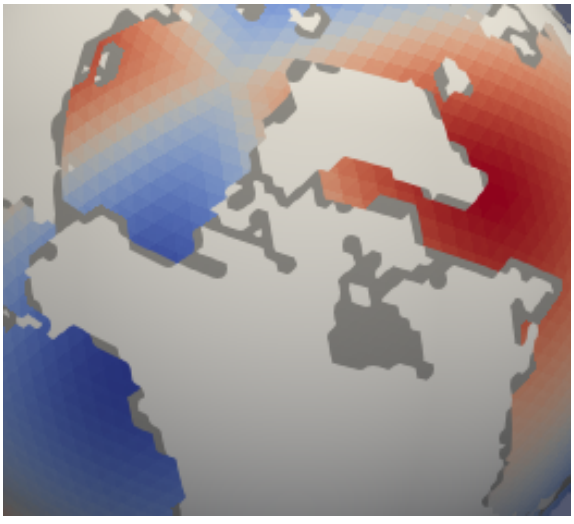
A limitation is that, because the method can only set values to zero, users cannot distinguish between target points that receive zero values through fallback interpolation and those that would legitimately interpolate to zero from the source field.

4.3 Dynamic fractional masking

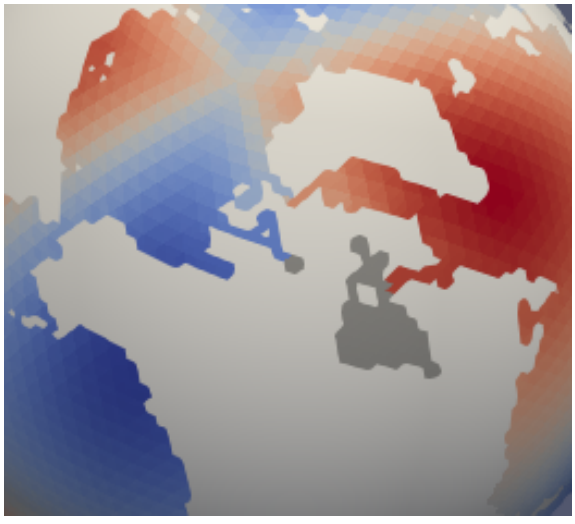
Traditional masks in YAC completely activate or deactivate individual source or target data points. However, for fields such as ice height, it is important to account for the fractional coverage of each cell when applying interpolation weights.

Based on a similar concept in OASIS3-MCT_6.0 (Valcke et al., 2025), YAC 3 introduces support for dynamic fractional masking, which allows the source component during the put operation to provide along with the source field a fractional

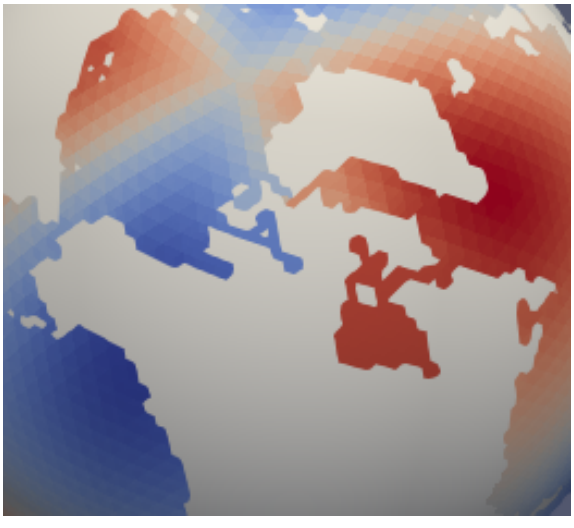
https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.



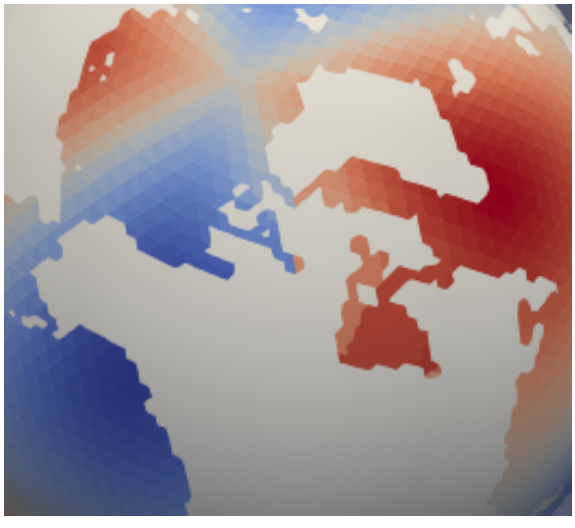
(a) 1st order conservative



(b) 1st order conservative + creep 16



(c) 1st order conservative + creep unlimited



(d) 1st order conservative + 1-Nearest-Neighbour

Figure 5. Remapping of the synthetic source field of Figure 4 onto an unmasked target grid using four interpolation stacks, each with first-order conservative interpolation as the primary method: (a) without a fallback, (b) with Creep Fill limited to 16 iterations, (c) with Creep Fill and an unlimited number of iterations, and (d) with a 1-Nearest-Neighbour fallback. Dark grey marks target cells that receive no data.

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

mask (with values $0 \leq f_i \leq 1$) for each source point. This enables time-varying masks without requiring recomputation of interpolation weights.

In each exchange, YAC first multiplies the source field values $\boldsymbol{F_s}$ by the corresponding fractional mask values $\boldsymbol{f}$. These masked values are then combined with the interpolation weights $\boldsymbol{w}$ to compute preliminary target values. To avoid undesired scaling of the target field, the result is normalised: each preliminary target value is divided by the frac-weighted sum of the weights and multiplied by the plain sum of the weights. If no source points contribute (i.e., the frac-weighted sum is zero) to a target point, a user-defined fallback value F_f is used.

Let F_t be a target field value, $\boldsymbol{F_s}$ the associated source field values, $\boldsymbol{w}$ the interpolation weights, $\boldsymbol{f}$ the fractional mask values, and F_f the fallback value. Then:

$$f' = \boldsymbol{f} \cdot \boldsymbol{w} \tag{1}$$

$$s = \sum_i w_i \tag{2}$$

$$F_t' = (\boldsymbol{F_s} \odot \boldsymbol{f}) \cdot \boldsymbol{w} \tag{3}$$

$$F_t = \begin{cases} F_t'/f' \cdot s & f' \neq 0 \\ F_f & f' = 0 \end{cases} \tag{4}$$

When all mask values are one, $f' = s$ and the formula reduces to the standard interpolation result $F_t = \boldsymbol{F_s} \cdot \boldsymbol{w}$, regardless of whether the weights sum to one.

When dynamic fractional masking is combined with a time aggregation scheme (i.e. the source field is provided at a finer time resolution than the coupling timestep), YAC independently accumulates the masked source field values $\boldsymbol{F_s} \odot \boldsymbol{f}$ and the fractional mask values $\boldsymbol{f}$ over all source timesteps. At the actual coupling timestep, the configured aggregation operation is applied to both accumulators separately, and the resulting quantities replace $\boldsymbol{F_s} \odot \boldsymbol{f}$ and $\boldsymbol{f}$ in the formula above. This approach differs from first computing a target field value at every source timestep and then aggregating those target values; the two are generally not equivalent when the fractional mask changes between source timesteps. Communication between source and target components is still only required once per coupling timestep, preserving efficiency.

4.4 Unstructured longitude-latitude grids

A recent study by Taylor (2024) demonstrated that accurate cell shape representation is essential for preserving global mean and integral properties of fields during interpolation. The study noted that YAC may be the only interpolation software capable of correctly representing regular Gaussian grid cells, whose edges follow circles of constant longitude or latitude on the sphere.

Beyond the issues identified by Taylor (2024), Reduced Gaussian grids present additional challenges related to cell shape representation (Valcke and Piacentini, 2019). Workarounds using seven corners instead of four per cell have been proposed and are currently employed (Jonville and Valcke, 2019).

In YAC, Reduced Gaussian grids must be defined as unstructured grids. To improve interpolation accuracy for these grids, an interface was added that allows defining unstructured grids whose cell edges follow circles of constant longitude or lati-

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

tude. When using a Reduced Gaussian grid with four corners per cell, this approach produces accurate results for first-order conservative interpolation.

YAC deduces cell connectivity relationships from shared corners. For Reduced Gaussian grids, this can result in missing connectivity information between cells from adjacent latitude rows. This limitation prevents second-order conservative remapping from functioning correctly, as it requires connectivity information to neighbouring cells for gradient computation. By combining the user-provided seven-corner cells with edges aligned to constant longitude or latitude, second-order conservative interpolation achieves accurate results.

4.5 Raw data exchange

A common approach to handling data exchange between different model grids is the "supermesh" or "exchange grid" approach as described in Balaji et al. (2006). It involves creating an intermediate grid representing the union of the source and target grids. Interpolation is then performed in two steps: first from the source to the intermediate grid, and then from the intermediate grid to the target. This approach is particularly useful for coupling fluxes.

Because YAC does not implement this feature, Bauer et al. (2021) chose not to use the available coupling infrastructure within ICON, but instead relied on the exchange grid functionality provided by ESMF to interpolate fluxes between the ICON and the General Estuarine Transport Model (GETM).

Supporting the supermesh approach directly would have required considerable additional infrastructure. YAC's conservative remapping already computes the individual intersection cells that such a mesh consists of, but these are temporary intermediate results that are never assembled into a complete grid. Providing a supermesh to users would therefore require both the construction and the management of that grid, as well as new interfaces to return it. Instead, a new concept called "raw data exchange" was introduced, which yields comparable results for the coupling of fluxes with limited changes on the user side.

With raw data exchange, users can enable this feature for specific field pairs to be coupled. When enabled, the source component sends data as usual. However, instead of receiving an already interpolated field, each target process receives all unaltered source field values required for interpolation of its local target points. Additionally, through a user interface, the target process can access the information needed to compute its target field values from the received source values, including the computed interpolation weights. This means that the target component is responsible for performing the interpolation itself, using the raw source field values and provided weights, rather than relying on YAC to deliver the final interpolated values.

While this increases the complexity of the target component, it provides greater flexibility and control over the interpolation process. For example, if first-order conservative interpolation is selected, the weights available to the target process represent the coverage of the target cell by the respective source cell. With this information, the target component can achieve results similar to the supermesh approach for the interpolation of fluxes, without requiring an intermediate grid.

Stochastic coupling methods, such as those described in Rackow and Juricke (2020), can also benefit from this approach. By receiving the raw source field values and associated weights, the target component can apply stochastic perturbations to the source values before performing the interpolation.

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

4.6 Interpolation summary and key messages

Overall, the developments in YAC 2 and YAC 3 broaden the interpolation toolbox and, based on published benchmarks and case studies, improve numerical quality for the tested configurations. Existing methods were refined, including conservative remapping and source point mapping, and new methods such as NCC, DNN, Creep Fill, User Callback, and Zero were added to address previously uncovered coupling scenarios. Together with dynamic fractional masking, improved support for unstructured longitude-latitude grids, and raw data exchange, these additions extend YAC's ability to handle heterogeneous model interfaces and demanding Earth system workflows.

At the same time, the expanded method portfolio increases configuration flexibility and therefore the need for informed method selection. In practice, stable interpolation setups are achieved by matching method stacks to the target application, in particular if masks or varying grid resolutions are involved, and by combining primary methods with appropriate fallback strategies. Example interpolation stack configurations for such setups are provided by Jonville et al. (2024). Consistent with this, the quality of the more recently added methods is best assessed within concrete user configurations rather than in isolation, and such application-specific evaluations are left to their respective users.

5 Challenge 3 — flexible integration into modern workflows

Modern Earth system modelling increasingly relies on flexible, user-friendly tools that integrate into diverse computational workflows. To meet these evolving requirements, YAC 3 has undergone substantial enhancements to its user interface and integration capabilities. This section describes the changes introduced to expand YAC's applicability and improve usability.

We begin by discussing the new Python bindings (Section 5.1) that enable integration with the extensive Python ecosystem commonly used in geosciences. Section 5.2 introduces a visualisation tool for interpolation weights that aids in debugging and understanding coupling configurations. Section 5.3 details the comprehensive overhaul of YAC's configuration system, including the transition to YAML-based configuration files, enhanced configuration interfaces, and name-based mask definitions. Finally, Section 5.4 describes the restructuring of YAC into separate Core and Model Coupling Interface (MCI) libraries, which facilitates integration with other coupling frameworks.

5.1 Python bindings

The Python programming language has evolved into a widely adopted language in computational geosciences. Beyond its applications in visualisation and data analytics, Python stands out as a preferred choice for rapid prototyping and is often the first programming language learned by students due to its straightforward syntax and readability.

To integrate the extensive Python package environment into YAC (and consequently the ESM using YAC), we have developed Python bindings for the YAC API. This was achieved using the Python package `cython`[3]. The design of these Python

[3]https://cython.org, last access: 18 May 2026

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

bindings closely aligns with the C interface, facilitating ease of use for individuals already acquainted with YAC, though it may deviate slightly where a more Pythonic style is more appropriate.

Specific applications of Python in this context include:

- **In-situ Visualisation:** Python boasts a diverse range of visualisation packages, particularly within the ESM context. YAC can now leverage Python to perform in-situ visualisation, such as receiving data on a regular, regional grid and dynamically visualising it using `cartopy` [4] during the model's runtime, eliminating the need to store intermediate data.
- **Output Handling:** While Python is commonly employed for analysing ESM data, it can also play a crucial role in writing out data. Notably, the `xarray` [5] package is popular for handling ESM data. Within the ICON community, the `hiopy`[6] (HEALPix I/O with PYthon) output component has been developed based on the Python bindings of YAC. It facilitates writing output on a hierarchy of HEALPix grids (Górski et al., 2005). See Section 7.2.
- **Input Processing:** Python's convenience in data handling extends to reading input data for an ESM during simulation and transferring it to the model through the YAC coupler. In the ICON model, we have implemented infrastructure to handle this process for ozone, aerosol, and ERA5 data.
- **Coupling of AI-based applications:** The growing importance of AI-based models offers new opportunities for Earth system modelling, and YAC can be leveraged to couple atmospheric or climate models to Python processes that run these AI models. By integrating with Python's extensive machine learning ecosystem—including popular frameworks such as `pytorch` and `tensorflow`—YAC supports coupling of models with AI capabilities, which can extend the functionality of Earth system simulations. This approach has already been used by Arnold et al. (2024).

This integration of Python within YAC extends the range of applications for which YAC can be used, including those that draw on Python packages common in Earth system modelling.

5.2 Weight visualisation tool

Interpolation weight files store the complete sparse mapping required for field exchange, including interpolation coefficients and the corresponding source and target addresses. Coupling software can use these files to perform interpolation without recomputing weights at runtime. If online weight generation is sufficiently fast for a given setup, however, persistent weight files may be unnecessary. In practice, such files can either be written by YAC after online weight computation or produced externally with tools such as the `ESMF_RegridWeightGen` utility[7].

YAC can write grid and weight information to netCDF files. The writing of grids using the format of the Spherical Coordinate Remapping and Interpolation Package (SCRIP) (Jones, 1998) can be enabled in the YAML configuration with the

[4] https://cartopy.readthedocs.io, last access: 18 May 2026
[5] https://xarray.dev/, last access: 18 May 2026
[6] https://gitlab.dkrz.de/dkrz-sw/hiopy, last access: 18 May 2026
[7] https://earthsystemmodeling.org/docs/release/latest/ESMF_refdoc, last access: 18 May 2026

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

`debug.output_grids` setting. To generate weight files, users can specify the `weight_file_name` parameter in the YAML coupling configuration.

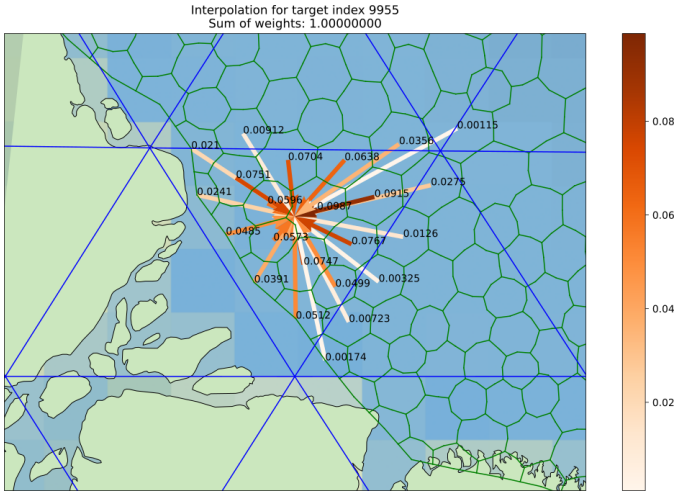


Figure 6. Visualisation of interpolation weights from a FESOM2 (Danilov et al., 2017) unstructured grid to an ICON grid using 1st order conservative interpolation. The target cell with index 9955 is focused.

To visualise the information that is contained in these files, the `yac_plot_weights.py` script provided with YAC facilitates the plotting of interpolation weights stored in a weights file. Leveraging the capabilities of Python, specifically `cartopy`, this tool enables the illustration of interpolation weights between source and target cells on their specific grids. Each weight is represented as an arrow originating from the source cell and pointing towards the corresponding target cell.

The script serves as a debugging tool that can help in understanding how target points are interpolated from the source data. It provides essential features such as the ability to select individual source and target cells, label cells with their respective global indices, annotate arrows with interpolation weights, and display coastlines and background maps for enhanced contextual understanding. Figure 3 and Figure 6 show example visualisations generated by this tool.

5.3 YAC configuration

Based on user feedback and new usage patterns, the configuration interface of YAC underwent a major overhaul. This encompassed both the configuration files and the user interface.

5.3.1 YAML configuration files

Originally, YAC used XML as the file format for its configuration files, which was considered state of the art when YAC development started in 2011. However, multiple user complaints about its verbosity, complexity, and low readability prompted a switch to YAML in YAC 3. Compared to XML, YAML offers a more concise syntax, and its advanced features such as anchors and references further improve the compactness of configuration files. In addition, the layout and contents of the configuration file were revised, further enhancing readability. Listing 1 shows the configuration for coupling `sea_surface_temperature` between an ocean and an atmosphere component at a coupling frequency of once every 8 minutes. It uses an interpolation stack that begins with first-order conservative interpolation, followed by a fixed user-defined value that is assigned to all target points that did not receive a value from the conservative interpolation method.

Listing 1. Example configuration file for the coupling of a single field

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

```
start_date: 2008-03-09T16:00:00
end_date: 2008-03-10T16:00:00
timestep_unit: ISO_format
calendar: proleptic-gregorian
coupling:
  - src_component: ocn_comp
    tgt_component: atm_comp
    src_grid: ocn_grid
    tgt_grid: atm_grid
    field: sea_surface_temperature
    coupling_period: PT8M
    interpolation:
      - conservative:
          partial_coverage: true
          normalisation: fracarea
      - fixed:
          user_value: -999.9
```

A program with a graphical user interface was originally implemented to aid in the generation of XML configuration files. However, due to limited usage and the introduction of simpler YAML-based configuration files, support for this tool has been discontinued.

In previous YAC versions, all processes of a coupled run were required to read a single configuration file to ensure a consistent view of the coupling configuration. To improve usability, a synchronisation step was added to the initialisation phase in which all processes exchange and merge their configurations to produce a unified view. This relaxes the restrictions on reading configuration files, which can now be read by an arbitrary set of processes from any number of files, provided they do not contain contradictory information (e.g., different start dates), which would automatically be detected and reported as an error during the synchronisation step.

5.3.2 Configuration user interface

New use cases for YAC required an extension of the user interface with regard to the coupling configuration. Reading a configuration file is now optional. For all information that can be set via a configuration file, corresponding user interfaces have been added that allow configuration during the initialisation phase. New interfaces for querying the global configuration have also been added. After the above-mentioned synchronisation step, these interfaces can provide information about definitions made by any process. Possible queries include: which components or grids are defined and which fields are defined for a given

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

grid. This information can then be used to extend the existing coupling configuration, for example by adding new coupling of additional fields. Such additional information is then distributed to all processes in a second and final synchronisation step.

A typical use case for these new interfaces is output components (Section 7.2), which can request and write data based on the information provided by other model components.

5.3.3 Name-based mask definition

With YAC 3 a new concept called "named masks" was introduced. This allows components to define masks with associated identifiers in the form of character strings. When defining a coupling between two fields, either via a configuration file or through the user interface, it is possible to specify the mask to be used for the source or target field based on this identifier.

Currently, to our knowledge, this feature is not yet being used by any application. A potential use case could be an output component that receives data from an atmosphere component. For some fields, the output component may require a global view of the source field, while for others, only data available over land may be desired. This would require the atmosphere component to define the various masks in advance, which can then be referenced by the output component.

5.4 Restructuring into Core and MCI

Originally, YAC was a single monolithic library that contained the interfaces needed for coupling between Earth system model components. However, over time requirements evolved and access to internal functionality (e.g., direct access to weight computation) by external software was requested. This led to a restructuring of the YAC source code base, which is now split into two parts.

The Core library contains basic functionality for the computation and handling of weights. It allows users to generate distributed grid pairs, set up interpolation stacks, compute weights, and perform distributed matrix-vector multiplications using these weights as the matrix.

The second part is the Model Coupling Interface (MCI), which contains the original user interface, configuration handling (including configuration files), and management of the Core. Both parts are separate libraries: the Core library is self-contained, while the MCI library depends on the Core.

This restructuring enables the integration of YAC Core into other libraries (Section 7.3).

5.5 Workflow summary and key messages

This section described how YAC 3 was adapted to better support modern, heterogeneous ESM workflows. Python bindings extend YAC into the broader scientific Python ecosystem and enable practical integrations such as in-situ visualisation, flexible input/output handling, and coupling to AI-based components. The weight visualisation tooling complements these capabilities by improving transparency and debugging of interpolation behaviour.

In parallel, the configuration system was redesigned to reduce setup friction: YAML replaces XML, configuration can be assembled across processes and files, and new query and programmatic interfaces support dynamic, configuration-driven

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

workflows such as output services. The introduction of named masks further prepares YAC for more selective and context-aware coupling patterns.

Finally, the separation into Core and MCI provides a cleaner architectural boundary between reusable interpolation functionality and high-level coupling orchestration. Together, these changes lower integration barriers while retaining the flexibility required for both model coupling and embedding of YAC functionality in external frameworks.

6 Challenge 4 — reliable multi-component orchestration

A common task in multi-executable coupled systems is the initial splitting of the global MPI communicator `MPI_COMM_WORLD` into component-specific and shared communicators. Because communicator creation requires collective MPI operations across all processes, orchestrating this consistently across heterogeneous libraries is challenging. The complexity of this task increases when a setup involves multiple components using multiple libraries that each require their own communicators. This issue typically arises when using both a coupling library and an output server library, as documented in Maisonnave and Valcke (2022).

This is conventionally handled by a custom, setup-specific initialisation sequence, which ensures that all components and libraries perform their communicator splits in a consistent order. However, this approach is fragile and can easily break when components or libraries are added, removed, or reordered. It also requires careful coordination between all parties involved in the coupled setup, which can be difficult to achieve in practice. Alternatively, the task of splitting `MPI_COMM_WORLD` can be delegated to a library (i.e., coupling library), but this requires that all components and libraries involved in the coupled setup use the same library for communicator management.

6.1 A generic handshake mechanism for reliable communicator orchestration

To address this challenge, we have developed a generic handshake mechanism and propose it as a common convention for splitting `MPI_COMM_WORLD`: a single collective call that accepts a list of strings per process and returns an equal-length list of intra-communicators, where each communicator groups all processes that provided the same string. The order is arbitrary across processes. When this algorithm is used by all components involved in a coupled setup to split the global communicator, each component can reliably obtain the communicator it needs by providing the same string on all processes that should be part of that communicator. Communicators spanning multiple components, as required by coupling libraries, can be created by providing the same library-specific string in all components that should participate in the shared communicator. This algorithm can be implemented either in the components themselves or by the coupling or similar libraries.

YAC implements this mechanism. Components that only use YAC and no other libraries requiring their own communicators can simply initialise YAC, which will internally perform the handshake and create the required communicators. The components can then query their component-specific communicator from YAC. For more complex setups, YAC provides an interface to perform the handshake directly within the component code. In addition, an interface returning the YAC-specific string is also provided. Once the component has performed the handshake and generated its own communicator and all other communicators

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

required by the libraries it uses (e.g., YAC), it can initialise the libraries with the communicators obtained from the handshake. After this point, usage of `MPI_COMM_WORLD` should no longer be necessary, which avoids incompatibilities between libraries and components.

The mechanism itself is independent of YAC, and we propose that other coupling and I/O libraries, such as OASIS3-MCT and XIOS, as well as model components, use it for their own communicator construction. Compared with the two established alternatives discussed above, it requires neither a fixed, setup-specific initialisation order nor a commitment by all participants to a single library for communicator management. Each participant only has to agree on the strings identifying the communicators it shares with others. Wider adoption would therefore improve compatibility between independently developed libraries and components. To support this, we provide a public reference implementation with compatible C, Fortran, and Python interfaces.[8]

The mechanism has already seen adoption beyond YAC. The ICON community interface (ComIn, Hartung et al., 2025) has adapted the handshake, and ICON itself now calls the algorithm explicitly and passes the resulting communicator to YAC. The configuration in which several independent libraries depend on communicators derived from `MPI_COMM_WORLD` has also been demonstrated: in a proof-of-concept setup, a component using XIOS for output was successfully coupled to ICON. In this setup, an adapted version of XIOS was used, which uses the handshake to obtain its communicators. This confirms that the mechanism works across independently developed libraries and is not restricted to setups in which YAC performs the split.

6.2 Orchestration summary and key messages

In summary, this section addresses a recurring orchestration problem in multi-component ESMs: reliable communicator construction across components and libraries that must collectively split `MPI_COMM_WORLD`. The proposed handshake pattern provides a generic and interoperable mechanism to construct both component-local and cross-component communicators, reducing the fragility of setup-specific initialisation orders.

The main practical requirement is that participating components and libraries agree on the handshake strings for communicators that should be shared. Under this convention, initialisation can proceed in a predictable sequence in which all required communicators are created first and libraries are initialised afterwards, avoiding ambiguous post-startup use of `MPI_COMM_WORLD` and thereby reducing integration conflicts in heterogeneous coupled configurations.

7 Use cases

Building on the advances described in previous sections, YAC 3 represents a substantial extension over YAC 1, enabling its use in a broader range of applications and configurations. ICON remains the primary model employing YAC for coupling, demonstrating its capabilities in high-resolution coupled simulations (Section 7.1). Its increased flexibility also makes it suitable as a communication layer between model- and file-access-oriented components, such as input and output servers (Section 7.2). Beyond coupling of components, YAC's interpolation capabilities can now be accessed directly by other libraries (Section 7.3).

[8] https://gitlab.dkrz.de/dkrz-sw/mpi-handshake, last access: 18 May 2026

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

7.1 ICON simulations

YAC's high-performance online weight computation and fast regridding capabilities (Hanke and Redler, 2019) have enabled high-resolution coupled simulations with the ICON model. The Earth system model configuration used for such simulations consists of several components, including atmosphere, ocean, and land. YAC-based coupling is primarily used between the atmosphere and ocean, and between land and ocean for river runoff (Fig. 1 in Hohenegger et al., 2023). A notable property of the ICON model is that its main components share the same basic infrastructure for grid-related operations. When the same grid resolution is used for both atmosphere and ocean, cells can overlap perfectly, allowing YAC to use a simple 1-Nearest-Neighbour (1NN) interpolation for remapping fields between grids. When grid resolutions or cell alignments differ, ICON configurations use a broader set of methods from the YAC interpolation stack. Runoff is remapped using the source point mapping technique described in Section 4.1.3. All remapping and interpolation weights are computed online during initialisation. Rather than requiring preprocessed files, the entire coupling is configured through a concise YAML configuration file read at runtime.

7.1.1 Kilometre-scale ICON configurations

The ICON-Sapphire model configuration (Hohenegger et al., 2023) demonstrates that the coupled global ICON model with YAC supports grid spacings ranging from 1.25 km to 10 km. Configurations with matching atmosphere and ocean grid resolutions use simple 1NN interpolation. For configurations where the ocean has higher resolution in certain regions, more complex approaches such as HCSBB and first-order conservative interpolation are employed. Bockelmann et al. (2022) reports a coupled simulation reproducing the weather on 7 December 1972, when the iconic Blue Marble photograph of Earth was taken from the Apollo 17 lunar mission. This simulation used an R02B11 grid (approximately 1.25 km horizontal resolution) for both atmosphere and ocean components, running on 900 nodes of Levante (Section 3.1). With matching grids, 1NN interpolation sufficed for all fields. Each multi-step simulation run took approximately 5.5 hours, with YAC initialisation requiring only one to two minutes. Due to different grid configurations and more complex coupling configurations this figure is not directly comparable to the performance measurements in Section 3.1; in both cases the setup cost is negligible relative to the overall runtime. Despite using exceptionally high-resolution grids, these simulations employed the same coupling configuration as more common ICON setups. All configurations executed without coupling issues or dedicated support requirements.

7.1.2 ICON-based climate projections

While YAC demonstrates its potential at high resolutions in terms of scalability, it is also used for coarser-resolution simulations targeting CMIP applications. The ICON-XPP configuration (Müller et al., 2025) is an example of such a setup. This configuration uses coarser grids of 160 km for the atmosphere and 40 km for the ocean, suitable for long-term climate simulations as a contribution to the Coupled Model Intercomparison Project (CMIP). YAC's flexibility and efficiency, together with its computational accuracy and conservation properties while handling different grid resolutions, make it a suitable choice for these simulations with simulated times of several centuries.

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

7.2 Coupled input/output server

In ESM components, file-based tasks such as reading input data or writing results often become bottlenecks. This is primarily due to the significantly different performance characteristics of the underlying hardware for computation and I/O, which can lead to inefficient resource utilisation. To address this issue, a functional decomposition of the model into compute and I/O components is applied. In this approach, I/O-related tasks are offloaded to dedicated input and output server components, which typically use fewer processes. Communication between the compute and I/O components involves redistributing data between them. Additionally, users may opt to use different grids for the compute and I/O components, necessitating data remapping during the transfer.

YAC has been used as a communication layer between compute and I/O components in coupled simulations. Its data redistribution and remapping capabilities can help limit the impact of I/O tasks on overall simulation performance. With support for Python bindings, I/O components can leverage Python's extensive ecosystem for data processing and file handling. Examples of such applications are presented in the following sections.

7.2.1 Input servers

ESM components often rely on reading forcing data from files. Depending on the origin of this data, its resolution or grid may differ from that of the model. A common approach is to apply a preprocessing step that remaps the data to the model grid. During model integration, the preprocessed data is read from file and used directly by the model. However, this workflow has several drawbacks: it requires additional storage for the preprocessed data—which can be substantial for high-resolution simulations—introduces performance bottlenecks due to I/O operations during model execution, and adds complexity through the extra preprocessing step.

An alternative is to use an input server component that reads the original data from file and provides it to the model component via YAC. The input server can perform the necessary remapping on-the-fly before sending the data to the model, eliminating the need for preprocessed files. This can reduce storage requirements and, by reading data asynchronously to the model's computation, can effectively hide I/O latency—thereby minimising the impact of I/O on overall simulation performance. The workflow is also simplified, as the separate preprocessing step is no longer required. Redler et al. (2026) presents a Python-based implementation of such an input server for the ICON model.

7.2.2 Output servers

The YAC-Output (YACO) server is an example of an output server component. It is a tool developed at the Max Planck Institute for Meteorology for the DestinE project (Doblas-Reyes et al., 2025) that writes ICON output to the Fields DataBase (FDB) on a HEALPix (Górski et al., 2005) grid. Since the FDB requires global fields, YACO benefits from data redistribution by assembling each global field on a single process. Designed for high-resolution settings, YACO runs on dedicated processes and can be parallelised over multiple fields, with every process holding the global fields of the subset of output variables assigned to it. Another example of ICON output in the NextGEMS project (Segura et al., 2025) is `hiopy` (Section 5.1), a

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

hierarchical output server that uses the HEALPix grid with varying resolutions and stores data in Zarr format, implemented entirely in Python. Through YAC, `hiopy` processes receive model data already interpolated onto the HEALPix grid. The dynamic configuration capability simplifies the creation of a component hierarchy in `hiopy`, enabling output generation at different grid resolutions without complex configuration files.

Using YAC to couple output server components to model components, rather than a dedicated output server library like XIOS, offers several advantages. First, it reduces dependencies and system complexity by requiring only one coupling library (YAC) for both model coupling and output server communication. Second, YAC's general coupling interface enables different output components to be used without modifying the model coupling code. Third, using the coupling library for communication between model and output server components simplifies the output component itself, which no longer needs to implement its own communication scheme for receiving model data. Finally, this approach makes the YAC interpolation stack available for output processing.

7.3 Direct access to YAC interpolation capabilities

By providing a well-defined interface to its interpolation stack, YAC enables other libraries to directly access its interpolation capabilities. This allows users to leverage its advanced interpolation methods in a broader range of applications even beyond model coupling. Two examples of such applications are OASIS3-MCT and UXarray, which are described in the following sections.

7.3.1 YAC in OASIS3-MCT

Like YAC, OASIS3-MCT provides parallel online weight computation capabilities. Through version 5.0, these were primarily based on the SCRIP library (Jones, 1998). However, several CERFACS studies (Jonville et al., 2018; Valcke and Piacentini, 2019; Jonville and Valcke, 2019) identified significant limitations, prompting evaluation of various alternatives (Valcke et al., 2022).

Through collaboration between CERFACS and DKRZ, YAC-based online weight computation support has been integrated into OASIS3-MCT version 6.0 (Valcke et al., 2025). This application was the primary driver for the restructuring described in Section 5.4, which enables OASIS3-MCT to access YAC Core interpolation capabilities directly without the user-oriented MCI layer. During initialisation, OASIS3-MCT reads its configuration file and, upon detecting a YAC-based interpolation request, provides the required grid data to YAC Core. The interpolation stack is then configured via the Core interface, and weights are computed in parallel and written to a netCDF-formatted weight file. OASIS3-MCT uses this file to configure its internal data structures for subsequent online regridding during model integration. From the user's perspective, the workflow mirrors the SCRIP-based weight generation process. While direct weight transfer without an intermediate file would improve performance, this has not yet been implemented.

By supporting YAC's interpolation stack concept and most of its interpolation methods, OASIS3-MCT version 6.0 expands its online interpolation capabilities, thereby exposing YAC to a broader user community.

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

7.3.2 YAC in UXarray

UXarray (Chen et al., 2026a) is a Python library that extends `xarray` to support unstructured grid data commonly produced by climate and global weather models. Built within the Pangeo ecosystem, it provides `xarray`-styled access to datasets stored in a variety of unstructured grid formats, including UGRID, MPAS, ICON, SCRIP, ESMF, FESOM2, and Exodus, enabling standard data analysis techniques to operate directly on unstructured mesh data. This makes UXarray well suited for exploratory data analysis and visualisation tasks, such as inspecting spatial patterns, comparing fields across simulations, and producing mesh-aware diagnostics.

For example, a user may load ICON or MPAS model output into UXarray, calculate temporal means or regional statistics using `xarray`-style operations, and subsequently generate visualisations on the native unstructured grid without requiring an intermediate regridding step.

UXarray now directly interfaces with the YAC Core interface, enabling YAC-based interpolation workflows to be configured and executed from within Python. This integration extends UXarray's remapping capabilities beyond its previous scope, in particular by adding access to conservative remapping methods for applications that require improved integral consistency.

8 Conclusion

This paper described the transition from YAC 1 to YAC 3 along four challenges that coupling software faces, some driven by the move to kilometre-scale resolution and others by the growing complexity of coupled workflows. For scalability (Section 3), the parallelisation scheme was rewritten around distributed grid pairs. This removes the need for user-provided halo information, involves the processes of both components in the weight computation, and keeps setup times within a few seconds for the configurations tested. For interpolation (Section 4), existing methods were refined and new ones added—Nearest-Corner-Cells, Distance-Nearest-Neighbour, Creep Fill, User Callback and Zero—complemented by dynamic fractional masking, support for unstructured longitude-latitude grids and raw data exchange. For workflow integration (Section 5), Python bindings, YAML-based configuration with a synchronisation step, tooling for weight and grid inspection, and the separation into a self-contained Core library and the Model Coupling Interface lower the barrier to adopting YAC, within and beyond classical model coupling. For orchestration (Section 6), a generic MPI handshake mechanism replaces setup-specific initialisation orders and is proposed as a convention for other libraries and components.

YAC is now used across a broad range of projects, from the classical coupling of physical model components—including coupled global ICON configurations with horizontal resolutions ranging from 10 to 1.25 km—to uses well beyond it. Notably, YAC was employed in the 1.25 km simulations that earned the ACM Gordon Bell Prize for climate modelling in 2025 (Klocke et al., 2025). Within DestinE, YAC has been effectively used to transfer data to output processes that receive data on a HEALPix grid and store it in a database. Similarly, it serves as a transport layer for data from files provided by external input processes. Currently, ICON remains the primary user of YAC. Other models, such as the FESOM2 ocean model (Danilov et al., 2017), the Elmer/Ice land-ice model (Gagliardini et al., 2013), the ParFlow hydrology model (Kuffour et al., 2020), and the Hydrological Discharge (HD) model (Hagemann et al., 2026), have so far adopted YAC mainly in order to couple to ICON. That models

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

as diverse as ocean, land-ice and hydrological models could be connected in this way nevertheless indicates that the interface itself is not specific to a single model.

Integrating YAC into a new model requires explicit code modifications within the model source. The model must call the appropriate YAC interface routines to register its coupling components, provide grid and field information, and manage the sending and receiving of data during runtime. If the model already employs another coupling library, this integration can become a challenging software development task. In such cases, introducing a coupling abstraction layer may be necessary to decouple model code from a specific coupling library and facilitate a smoother transition or coexistence between different coupling solutions.

For real-world applications, which often run for several hours, the runtime penalty for the online computation of weights (as presented in Section 3.1) is small in the discussed benchmark and use case configurations. The added benefit of avoiding preprocessing steps for generating and preparing weight files tends to be more significant in the studied configurations.

Taken together, the developments described here have turned YAC from a coupler for classical model components into a general-purpose framework for parallel data exchange that scales to kilometre-scale grids, offers a broad and extensible set of interpolation methods, and integrates into increasingly complex workflows. As Earth system modelling continues to move towards higher resolutions and more heterogeneous, loosely coupled workflows, we expect these capabilities to become ever more relevant, and we will continue to develop YAC together with its growing user community to meet these evolving needs.

Code availability. The source code of YAC is developed at https://gitlab.dkrz.de/dkrz-sw/yac and is distributed under the BSD 3-Clause license. YAC v3.19.0 has been archived on Zenodo (https://doi.org/10.5281/zenodo.21992085, Hanke et al., 2026). The archived release includes the full HTML documentation in `doc/html`. The performance measurements described in Section 3.1 were carried out with the program in `examples/perf_toy`, which is also included in the release.

Data availability. The timing measurements shown in Figure 1, together with the raw job output, the runscripts and the post-processing and plotting scripts used to produce the figure, are archived on Zenodo (https://doi.org/10.5281/zenodo.21993441, Hanke, 2026). No other datasets were used in this article.

Author contributions. MH developed the YAC software and prepared the manuscript. ND contributed to the development of YAC and co-authored the manuscript. RR provided scientific guidance and contributed to the conceptualisation and writing of the manuscript. AP developed the integration of YAC in OASIS and supported the restructuring into Core and MCI. All authors contributed to the final manuscript.

Competing interests. The authors declare that they have no conflict of interest.

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.



Acknowledgements. The authors acknowledge the use of AI-based tools, including GitHub Copilot, ChatGPT, and Claude Code, to assist in refining the syntax, grammar, and readability of the manuscript, and in maintaining its build configuration and bibliography metadata. While these tools supported the editing process, the authors retain full responsibility for the content, accuracy, and integrity of the final work.

Financial support. The development of YAC was supported by the institutional resources of the German Climate Computing Center (DKRZ).

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

References

Alfeld, P., Neamtu, M., and Schumaker, L. L.: Bernstein-Bézier polynomials on spheres and sphere-like surfaces, Comput. Aided Geom. Des., 13, 333–349, https://doi.org/10.1016/0167-8396(95)00030-5, 1996.

Anonymous Referee #2: Review of YAC 1.2.0: An extendable coupling software for Earth system modelling, Interactive comment on gmd-2015-267, https://doi.org/10.5194/gmd-2015-267-RC2, 2016.

Arakawa, T., Inoue, T., Yashiro, H., and Satoh, M.: Coupling library Jcup3: its philosophy and application, Prog Earth Planet Sci, 7, https://doi.org/10.1186/s40645-019-0320-z, 2020.

Arnold, C., Sharma, S., Weigel, T., and Greenberg, D. S.: Efficient and stable coupling of the SuperdropNet deep-learning-based cloud microphysics (v0.1.0) with the ICON climate and weather model (v2.6.5), Geoscientific Model Development, 17, 4017–4029, https://doi.org/10.5194/gmd-17-4017-2024, 2024.

Balaji, V., Anderson, J., Held, I., Winton, M., Durachta, J., Malyshev, S., and Stouffer, R. J.: The Exchange Grid: A mechanism for data exchange between Earth System components on independent grids, in: Parallel Computational Fluid Dynamics 2005, edited by Deane, A., Ecer, A., McDonough, J., Satofuka, N., Brenner, G., Emerson, D. R., Periaux, J., and Tromeur-Dervout, D., pp. 179–186, Elsevier, Amsterdam, ISBN 978-0-444-52206-1, https://doi.org/10.1016/B978-044452206-1/50021-5, 2006.

Bauer, T. P., Holtermann, P., Heinold, B., Radtke, H., Knoth, O., and Klingbeil, K.: ICONGETM v1.0 – flexible NUOPC-driven two-way coupling via ESMF exchange grids between the unstructured-grid atmosphere model ICON and the structured-grid coastal ocean model GETM, Geoscientific Model Development, 14, 4843–4863, https://doi.org/10.5194/gmd-14-4843-2021, 2021.

Bockelmann, H., Röber, N., and Stevens, B.: Revisiting the Blue Marble: ICON simulating the coupled climate system at 1 km, https://mpimet.mpg.de/en/communication/news/neuer-blick-auf-das-blue-marble-foto-icon-simuliert-das-gekoppelte-klimasystem-mit-1-km-aufloesung [Accessed: 2024/09/30], 2022.

Chen, H., Chmielowiec, P., Clyne, J., Eroglu, O., Jacob, R., Jain, R., Medeiros, B., Ullrich, P., and Zarzycki, C.: uxarray, https://doi.org/10.5281/zenodo.5655065, Zenodo [code], 2026a.

Chen, H., Ullrich, P. A., Panetta, J., Marsico, D., Hanke, M., Jain, R., Zhang, C., and Jacob, R. L.: Accurate and Robust Geometric Algorithms for Regridding on the Sphere, EGUsphere [preprint], https://doi.org/10.5194/egusphere-2026-636, 2026b.

Collins, N., Theurich, G., DeLuca, C., Suarez, M., Trayanov, A., Balaji, V., Li, P., Yang, W., Hill, C., and da Silva, A.: Design and Implementation of Components in the Earth System Modeling Framework, The International Journal of High Performance Computing Applications, 19, 341–350, https://doi.org/10.1177/1094342005056120, 2005.

Craig, A., Valcke, S., and Coquart, L.: Development and performance of a new version of the OASIS coupler, OASIS3-MCT_3.0, Geoscientific Model Development, 10, 3297–3308, https://doi.org/10.5194/gmd-10-3297-2017, 2017.

Danilov, S., Sidorenko, D., Wang, Q., and Jung, T.: The Finite-volumE Sea ice–Ocean Model (FESOM2), Geoscientific Model Development, 10, 765–789, https://doi.org/10.5194/gmd-10-765-2017, 2017.

Doblas-Reyes, F. J., Kontkanen, J., Sandu, I., Acosta, M., Al Turjmam, M. H., Alsina-Ferrer, I., Andrés-Martínez, M., Arriola, L., Axness, M., Batlle Martín, M., Bauer, P., Becker, T., Beltrán, D., Beyer, S., Bockelmann, H., Bretonnière, P.-A., Cabaniols, S., Caprioli, S., Castrillo, M., Chandrasekar, A., Cheedela, S., Correal, V., Danovaro, E., Davini, P., Enkovaara, J., Frauen, C., Früh, B., Gaya Àvila, A., Ghinassi, P., Ghosh, R., Ghosh, S., González, I., Grayson, K., Griffith, M., Hadade, I., Haine, C., Hartick, C., Haus, U.-U., Hearne, S., Järvinen, H., Jiménez, B., John, A., Juchem, M., Jung, T., Kegel, J., Kelbling, M., Keller, K., Kinoshita, B., Kiszler, T., Klocke, D., Kluft, L., Koldunov, N., Kölling, T., Kolstela, J., Kornblueh, L., Kosukhin, S., Lacima-Nadolnik, A., Leal Rojas, J. J., Lehtiranta, J., Lunttila, T., Luoma, A.,

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

Manninen, P., Medvedev, A., Milinski, S., Mohammed, A. O. A., Müller, S., Naryanappa, D., Nazarova, N., Niemelä, S., Niraula, B., Nortamo, H., Nummelin, A., Nurisso, M., Ortega, P., Paronuzzi, S., Pedruzo Bagazgoitia, X., Pelletier, C., Peña, C., Polade, S., Pradhan, H., Quintanilla, R., Quintino, T., Rackow, T., Räisänen, J., Rajput, M. M., Redler, R., Reuter, B., Rocha Monteiro, N., Roura-Adserias, F., Ruppert, S., Sayed, S., Schnur, R., Sharma, T., Sidorenko, D., Sievi-Korte, O., Soret, A., Steger, C., Stevens, B., Streffing, J., Sunny, J., Tenorio, L., Thober, S., Tigerstedt, U., Tinto, O., Tonttila, J., Tuomenvirta, H., Tuppi, L., Van Thielen, G., Vitali, E., von Hardenberg, J., Wagner, I., Wedi, N., Wehner, J., Willner, S., Yepes-Arbós, X., Ziemen, F., and Zimmermann, J.: The Destination Earth digital twin for climate change adaptation, EGUsphere, 2025, 1–41, https://doi.org/10.5194/egusphere-2025-2198, 2025.

Fladrich, U. and Maisonnave, E.: Ready-to-couple (rdy2cpl), https://raw.githubusercontent.com/IS-ENES3/IS-ENES-Portal-Website/master/pdf_documents/CW2023_Fladrich.pdf [Accessed: 2024/12/05], 2023.

Gagliardini, O., Zwinger, T., Gillet-Chaulet, F., Durand, G., Favier, L., de Fleurian, B., Greve, R., Malinen, M., Martín, C., Råback, P., Ruokolainen, J., Sacchettini, M., Schäfer, M., Seddik, H., and Thies, J.: Capabilities and performance of Elmer/Ice, a new-generation ice sheet model, Geoscientific Model Development, 6, 1299–1318, https://doi.org/10.5194/gmd-6-1299-2013, 2013.

Górski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M., and Bartelmann, M.: HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere, The Astrophysical Journal, 622, 759–771, https://doi.org/10.1086/427976, 2005.

Hagemann, S., Ho-Hagemann, H. T. M., and Hanke, M.: The Hydrological Discharge Model – a river runoff component for offline and coupled model applications, https://doi.org/10.5281/zenodo.19068931, Zenodo [code], 2026.

Hanke, M.: Setup-scaling measurements for YAC 3.19.0, https://doi.org/10.5281/zenodo.21993441, Zenodo [data], 2026.

Hanke, M. and Redler, R.: New Features with YAC 1.5.0, Reports on ICON, No. 3, Deutscher Wetterdienst, https://doi.org/10.5676/DWD_pub/nwv/icon_003, 2019.

Hanke, M., Redler, R., Holfeld, T., and Yastremsky, M.: YAC 1.2.0: new aspects for coupling software in Earth system modelling, Geosci. Model Dev., 9, 2755–2769, https://doi.org/10.5194/gmd-9-2755-2016, 2016.

Hanke, M., Dreier, N.-A., Rodenberg, B., Ibach, M., Redler, R., and Kosukhin, S.: YAC v3.19.0, https://doi.org/10.5281/zenodo.21992085, Zenodo [code], 2026.

Hartung, K., Kern, B., Dreier, N.-A., Geisbüsch, J., Haghighatnasab, M., Jöckel, P., Kerkweg, A., Loch, W. J., Prill, F., and Rieger, D.: ICON ComIn – the ICON Community Interface (ComIn version 0.1.0, with ICON version 2024.01-01), Geoscientific Model Development, 18, 1001–1015, https://doi.org/10.5194/gmd-18-1001-2025, 2025.

Hohenegger, C., Korn, P., Linardakis, L., Redler, R., Schnur, R., Adamidis, P., Bao, J., Bastin, S., Behravesh, M., Bergemann, M., Biercamp, J., Bockelmann, H., Brokopf, R., Brüggemann, N., Casaroli, L., Chegini, F., Datseris, G., Esch, M., George, G., Giorgetta, M., Gutjahr, O., Haak, H., Hanke, M., Ilyina, T., Jahns, T., Jungclaus, J., Kern, M., Klocke, D., Kluft, L., Kölling, T., Kornblueh, L., Kosukhin, S., Kroll, C., Lee, J., Mauritsen, T., Mehlmann, C., Mieslinger, T., Naumann, A. K., Paccini, L., Peinado, A., Praturi, D. S., Putrasahan, D., Rast, S., Riddick, T., Roeber, N., Schmidt, H., Schulzweida, U., Schütte, F., Segura, H., Shevchenko, R., Singh, V., Specht, M., Stephan, C. C., von Storch, J.-S., Vogel, R., Wengel, C., Winkler, M., Ziemen, F., Marotzke, J., and Stevens, B.: ICON-Sapphire: simulating the components of the Earth system and their interactions at kilometer and subkilometer scales, Geoscientific Model Development, 16, 779–811, https://doi.org/10.5194/gmd-16-779-2023, 2023.

Jones, P. W.: A User's Guide for SCRIP: A Spherical Coordinate Remapping and Interpolation Package, http://oceans11.lanl.gov/svn/SCRIP/trunk/SCRIP/doc/SCRIPusers.pdf, 1998.

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.

Jonville, G. and Valcke, S.: Analysis of SCRIP conservative remapping in OASIS3-MCT – Part B, Tech. Rep. TR/CMGC/19/155, CERFACS, 2019.

Jonville, G., Valcke, S., Coquart, L., Maisonnave, E., and Piacentini, A.: Specific analyses of the SCRIP interpolation library for OASIS coupler, Tech. Rep. WN/CMGC/18/93, CERFACS, 2018.

Jonville, G., Piacentini, Valcke, S., and Hanke, M.: Integration of the YAC coupling library in OASIS3-MCT, Tech. Rep. TR-CMGC-24-182, CERFACS, https://cerfacs.fr/wp-content/uploads/2024/12/TR-CMGC-24-182_yac_in_oasis.pdf, 2024.

Klocke, D., Frauen, C., Engels, J. F., Alexeev, D., Redler, R., Schnur, R., Haak, H., Kornblueh, L., Brüggemann, N., Chegini, F., Römmer, M., Hoffmann, L., Griessbach, S., Bode, M., Coles, J., Gila, M., Sawyer, W., Calotoiu, A., Budanaz, Y., Mazumder, P., Copik, M., Weber, B., Herten, A., Bockelmann, H., Hoefler, T., Hohenegger, C., and Stevens, B.: Computing the Full Earth System at 1km Resolution, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '25, p. 125–136, Association for Computing Machinery, New York, NY, USA, ISBN 9798400714665, https://doi.org/10.1145/3712285.3771789, 2025.

Kuffour, B. N. O., Engdahl, N. B., Woodward, C. S., Condon, L. E., Kollet, S., and Maxwell, R. M.: Simulating coupled surface–subsurface flows with ParFlow v3.5.0: capabilities, applications, and ongoing development of an open-source, massively parallel, integrated hydrologic model, Geoscientific Model Development, 13, 1373–1397, https://doi.org/10.5194/gmd-13-1373-2020, 2020.

Liu, L., Sun, C., Yu, X., Yu, H., Jiang, Q., Li, X., Li, R., Wang, B., Shen, X., and Yang, G.: C-Coupler3.0: an integrated coupler infrastructure for Earth system modelling, Geoscientific Model Development, 16, 2833–2850, https://doi.org/10.5194/gmd-16-2833-2023, 2023.

Maisonnave, E.: Locally conservative OASIS interpolation using target grid nearest neighbours, Tech. Rep. TR/CMGC/20/166, CERFACS, 2020.

Maisonnave, E. and Valcke, S.: Joint usage of OASIS3-MCT and XIOS, Tech. rep., CERFACS, https://oasis.cerfacs.fr/wp-content/uploads/sites/114/2022/02/Joint_usage_OASIS3-MCT_XIOS_2022.pdf, 2022.

Manabe, S. and Bryan, K.: Climate Calculations with a Combined Ocean-Atmosphere Model, Journal of the Atmospheric Sciences, 26, 786—-789, https://doi.org/10.1175/1520-0469(1969)026<0786:CCWACO>2.0.CO;2, 1969.

Mogensen, K., Polichtchouk, I., Hadade, I., Hatfield, S., Wedi, N., and Anantharaj, V.: Coupling at scale: Experiences from the IN-CITE project, https://raw.githubusercontent.com/IS-ENES3/IS-ENES-Portal-Website/master/pdf_documents/CW2023_Morgensen.pdf [Accessed: 2024/12/05], 2023.

Moreno-Chamarro, E., Arsouze, T., Acosta, M., Bretonnière, P.-A., Castrillo, M., Ferrer, E., Frigola, A., Kuznetsova, D., Martin-Martinez, E., Ortega, P., and Palomas, S.: The very-high resolution configuration of the EC-Earth global model for HighResMIP, Geoscientific Model Development Discussions, 2024, 1–49, https://doi.org/10.5194/gmd-2024-119, 2024.

Müller, W. A., Lorenz, S., Pham, T. V., Schneidereit, A., Brokopf, R., Brovkin, V., Brüggemann, N., Chegini, F., Dommenget, D., Fröhlich, K., Früh, B., Gayler, V., Haak, H., Hagemann, S., Hanke, M., Ilyina, T., Jungclaus, J., Köhler, M., Korn, P., Kornblueh, L., Kroll, C. A., Krüger, J., Castro-Morales, K., Niemeier, U., Pohlmann, H., Polkova, I., Potthast, R., Riddick, T., Schlund, M., Stacke, T., Wirth, R., Yu, D., and Marotzke, J.: The ICON-based Earth System Model for climate predictions and projections (ICON XPP v1.0), Geoscientific Model Development, 18, 9385–9415, https://doi.org/10.5194/gmd-18-9385-2025, 2025.

Pinar, A. and Hendrickson, B.: Communication Support for Adaptive Computation, in: Proceedings of the Tenth SIAM Conference on Parallel Processing for Scientific Computing, PP 2001, Portsmouth, Virginia, USA, March 12-14, 2001, SIAM, 2001.

Plimpton, S. J., Hendrickson, B., and Stewart, J. R.: A parallel rendezvous algorithm for interpolation between multiple grids, Journal of Parallel and Distributed Computing, 64, 266–276, https://doi.org/10.1016/j.jpdc.2003.11.006, 2004.

https://doi.org/10.5194/egusphere-2026-5008
Preprint. Discussion started: 2 November 2026
© Author(s) 2026. CC BY 4.0 License.



Rackow, T. and Juricke, S.: Flow-dependent stochastic coupling for climate models with high ocean-to-atmosphere resolution ratio, Quarterly Journal of the Royal Meteorological Society, 146, 284–300, https://doi.org/https://doi.org/10.1002/qj.3674, 2020.

Redler, R., Dreier, N.-A., Schneidereit, A., Haak, H., and Hanke, M.: ICON input via Python, Reports on ICON, No. 12, Deutscher Wetterdienst, https://doi.org/10.5676/DWD_pub/nwv/icon_012, 2026.

Segura, H., Pedruzo-Bagazgoitia, X., Weiss, P., Müller, S. K., Rackow, T., Lee, J., Dolores-Tesillos, E., Benedict, I., Aengenheyster, M., Aguridan, R., Arduini, G., Baker, A. J., Bao, J., Bastin, S., Baulenas, E., Becker, T., Beyer, S., Bockelmann, H., Brüggemann, N., Brunner, L., Cheedela, S. K., Das, S., Denissen, J., Dragaud, I., Dziekan, P., Ekblom, M., Engels, J. F., Esch, M., Forbes, R., Frauen, C., Freischem, L., García-Maroto, D., Geier, P., Gierz, P., González-Cervera, A., Grayson, K., Griffith, M., Gutjahr, O., Haak, H., Hadade, I., Haslehner, K., ul Hasson, S., Hegewald, J., Kluft, L., Koldunov, A., Koldunov, N., Kölling, T., Koseki, S., Kosukhin, S., Kousal, J., Kuma, P., Kumar, A. U., Li, R., Maury, N., Meindl, M., Milinski, S., Mogensen, K., Niraula, B., Nowak, J., Praturi, D. S., Proske, U., Putrasahan, D., Redler, R., Santuy, D., Sármány, D., Schnur, R., Scholz, P., Sidorenko, D., Spät, D., Sützl, B., Takasuka, D., Tompkins, A., Uribe, A., Valentini, M., Veerman, M., Voigt, A., Warnau, S., Wachsmann, F., Wacławczyk, M., Wedi, N., Wieners, K.-H., Wille, J., Winkler, M., Wu, Y., Ziemen, F., Zimmermann, J., Bender, F. A.-M., Bojovic, D., Bony, S., Bordoni, S., Brehmer, P., Dengler, M., Dutra, E., Faye, S., Fischer, E., van Heerwaarden, C., Hohenegger, C., Järvinen, H., Jochum, M., Jung, T., Jungclaus, J. H., Keenlyside, N. S., Klocke, D., Konow, H., Klose, M., Malinowski, S., Martius, O., Mauritsen, T., Mellado, J. P., Mieslinger, T., Mohino, E., Pawłowska, H., Peters-von Gehlen, K., Sarré, A., Sobhani, P., Stier, P., Tuppi, L., Vidale, P. L., Sandu, I., and Stevens, B.: nextGEMS: entering the era of kilometer-scale Earth system modeling, Geoscientific Model Development, 18, 7735–7761, https://doi.org/10.5194/gmd-18-7735-2025, 2025.

Taylor, K. E.: Truly conserving with conservative remapping methods, Geoscientific Model Development, 17, 415–430, https://doi.org/10.5194/gmd-17-415-2024, 2024.

Valcke, S. and Piacentini, A.: Analysis of SCRIP conservative remapping in OASIS3-MCT – Part A, Tech. Rep. TR/CMGC/19/129, CERFACS, 2019.

Valcke, S., Balaji, V., Craig, A., DeLuca, C., Dunlap, R., Ford, R. W., Jacob, R., Larson, J., O'Kuinghttons, R., Riley, G. D., and Vertenstein, M.: Coupling technologies for Earth System Modelling, Geoscientific Model Development, 5, 1589–1596, https://doi.org/10.5194/gmd-5-1589-2012, 2012.

Valcke, S., Piacentini, A., and Jonville, G.: Benchmarking Regridding Libraries Used in Earth System Modelling, Mathematical and Computational Applications, 27, https://doi.org/10.3390/mca27020031, 2022.

Valcke, S., Craig, A., Piacentini, A., and Jonville, G.: OASIS3-MCT_6.0 User Guide, Tech. Rep. TR/CMGC/25/161, CERFACS, https://cerfacs.fr/wp-content/uploads/2025/12/oasis3mct_UserGuide.pdf, 2025.