Yet Another Coupler – YAC
Version 2.0.0 – Jan 2021

Contact:  Moritz Hanke (DKRZ)
           René Redler (MPI-M)
YAC – Team

**Moritz Hanke** (DKRZ)
René Redler (MPI-M)
Teresa Holfeld (MPI-M, student assistant)
Maxim Yastremsky (MPI-M, student assistant)

With contributions from

Thomas Jahns (DKRZ)
Uwe Schulzweida (MPI-M)
Hendrik Bockelmann (DKRZ)
Jörg Behrens (DKRZ)
Sergey Kosukhin (MPI-M)
YAC – General remarks
YAC – General remarks

ECHAM6

Energy Momentum

Water

Carbon

Energy Momentum

libOASIS3-MCT.a

HAMOCC

MPIOM

Water

JSBACH3

MPI-ESM1.1
YAC – General remarks

- ICON atmosphere
  - Carbon
  - Energy
  - Momentum
  - Water
- JSBACH4
  - Energy
  - Momentum
  - River Runoff
- ICON ocean
- ICON-ESM
- libyac.a
- HAMOCC
YAC – General remarks

- search
- interpolation
- configuration
- communication
- data exchange
A coupling software not only for ICON

- Parallel search on (almost) arbitrary grids on the sphere
- Parallel interpolation
- Parallel data exchange

- Library
- BSD License
- Programming Language C
- Fortran and C user API
- Programming based on standards (C, MPI, XML, NetCDF)
- Git repository
- Autotools
- Valgrind testing
- Unit tests (~90% of lines covered)
- Fortran and C examples plus toy models
- XML coupling configuration file with GUI support
YAC – Search

required

• Geographical positions (λ, φ) of vertices and points

provided

• Initial scalable computation of global mapping
• Final scalable parallel interpolation specific search and calculation of interpolation weights

features

• Support for circles of latitude/longitude and great circles
• Search and interpolation in Cartesian coordinates
• Convex & moderately concave polygons
• Support for masked cells and points
Available 2-dimensional (horizontal) interpolation methods

- 1\textsuperscript{st} – order conservative remapping (\texttt{conserv})
- 2\textsuperscript{nd} – order conservative remapping (\texttt{conserv})
- Hybrid cubic spherical Bernstein-Bézier patch interpolation (\texttt{bernstein_bezier})
- Distance-weighted N-nearest-neighbour (\texttt{n-nearest_neighbour})
- N-nearest-neighbour average (\texttt{n-nearest_neighbour})
- Gauss-weighted N-nearest-neighbour (\texttt{n-nearest_neighbour})
- Radial Basis Functions (\texttt{radial_basis_function})
- Source Point to Target Point Mapping (\texttt{source_to_target_map})
- Fixed value (\texttt{fixed})
Available 2-dimensional (horizontal) interpolation methods (continued)

- Patch recovery - polynomial fit (patch_recovery)
- Smoothed Patch recovery - polynomial fit (smooth_patch_recovery)
- Radial Basis Functions (radial_basis_function)
- Source Point to Target Point Mapping (source_to_target_map)
- Simple cell average (average)
- Distance-weighted cell average (average)
- File input (user_file)
example


1\textsuperscript{st}-order conservative remapping
plus patch recovery
plus fixed value
Step 1: 1\textsuperscript{st}- order conservative remapping
YAC – Interpolation stack

Step 1: … + patch recovery
YAC – Interpolation stack

Step 1: ... + fixed value
YAC – Graphical User Interface

[Image of a YAC graphical user interface showing options for Transients and Basic settings.]
YAC – Fortran Interface

**Initialisation Phase**
- `yac_finit`
- `yac_fdef_comp`
- `yac_fdef_datetime`
- `yac_fget_localcomm`

**Grid Definition**
- `yac_fdef_grid`
- `yac_fdef_points`
- `yac_fdef_index_location`
- `yac_fset_core_mask`
- `yac_fdef_mask`
- `yac_fdef_field`

**Search – End of Definition**
- `yac_fsearch`

**Data exchange**
- `yac_fget`
- `yac_fput`

**Termination**
- `yac_ffinalize`
YAC – Initialisation phase

component initialisation

CALL yac_finit ( "coupling.xml", "coupling.xsd" )
• will call MPI_INIT if not been called already

CALL yac_fdef_comp ( "component_name", component_id )
• local operations for initialising of YAC-internal data structures
• needs to be called by all processes

CALL yac_fdef_datetime ( start_datetime = start_of_run_in_iso_format, end_datetime = end_of_run_in_iso_format )
• overwrites start and end date set in coupling.xml
• if required it has to be called before calling yac_fdef_field
• time management inside yac using mtime
YAC – Definition Phase

grid definition (example for an unstructured grid)

CALL yac_fdef_grid ( "grid_name“,
                        nbr_of_horizontal_vertices,
                        nbr_of_horizontal_cells,
                        nbr_vertices_per_cell,
                        array_of_vertex_longitudes,
                        array_of_vertex_latitudes,
                        connectivity,
                        grid_id )

overloaded with respect to
- data type for coordinate arrays
- grid types
CALL yac_fset_global_index ( array_of_global_indices,
YAC_LOCATION_CELL,
grid_id )
YAC – Definition Phase

**grid definition**

CALL yac_fset_core_mask ( core_mask_array, YAC_LOCATION_CELL, grid_id )
YAC – Definition Phase

mask definition

CALL yac_fset_mask ( mask_array, point_id )

overloaded with respect to data type (Integer or Logical) of mask array

mask_array
1 (.TRUE.) for valid data
0 (.FALSE.) for invalid data
CALL yac_fdef_field ( "field_name", component_id, grid_id, array_of_cell_point_ids, nbr_point_sets, field_id )
CALL yac_fsearch ( nbr_of_components, array_of_component_ids, nbr_of_fields, array_of_field_ids, error_status )

- includes collective MPI operations
- needs to be called by all processes
- accesses the coupling configuration
- invokes the neighbourhood search
- does the communicator splitting

CALL yac_fget_localcomm ( local_mpi_communicator, component_id )
YAC – Exchange Phase

Data exchange

As it is implemented in ICON

Atmosphere

Ocean
YAC – Exchange Phase

data exchange

CALL yac_fput(
    field_id,
    nbr_horizontal_points,
    collection_size,
    send_field,
    info,
    error_flag
)

- to be called at every time step
- at the “source timestep” interval specified in the xml file
- accumulation/averaging done inside yac_fput
YAC – Exchange Phase

data exchange as it is implemented in ICON

! `field_id(6)` : Temperature

DO i_blk = 1, patch_horz%nblks_c
  nn = (i_blk-1)*nproma
  DO n = 1, nproma
    buffer(nn+n,1) = &
    ocean_state%p_prog(nold(1))%tracer(n,1,i_blk,1) + tmelt
  ENDDO
ENDDO

CALL `yac_fput` ( `field_id(6)`, nbr_hor_points, 1, &
  & buffer(1:nbr_hor_points,1), &
  & info, ierror )
data exchange

CALL yac_fget ( field_id, collection_size, recv_field, info, error_flag )

• to be called at every time step
• at the “source timestep” interval specified in the xml file
• accumulation/averaging done inside yac_fput
YAC – Exchange Phase

data exchange as it is implemented in ICON

CALL `yac_fget` ( `field_id(1)`, `nbr_hor_points`, 2, &
    & `buffer(1:nbr_hor_points,1:2)`, &
    & `info`, `ierror` )

IF ( `info` > 0 .AND. `info` < 7 ) THEN
  DO `i_blk` = 1, `patch_horz%nblks_c`
    `nn` = (`i_blk`-1)*`nproma`
    DO `n` = 1, `nproma`
      `atmos_fluxes%stress_xw`(n,`i_blk`) = `buffer`(nn+n,1)
      `atmos_fluxes%stress_x`(n,`i_blk`) = `buffer`(nn+n,2)
    ENDDO
  ENDDO
CALL `sync_patch_array` ...
YAC – Exchange Phase

data exchange

Return values for the info argument

```c
enum, bind(c)
    enumerator :: NONE = 0
    enumerator :: COUPLING = 1
    enumerator :: RESTART = 2
    enumerator :: GET_FOR_RESTART = 3
    enumerator :: PUT_FOR_RESTART = 4
    enumerator :: GET_FOR_CHECKPOINT = 5
    enumerator :: PUT_FOR_CHECKPOINT = 6
    enumerator :: OUT_OF_BOUND = 7
end enum
```
YAC – Termination Phase

termination of coupling

CALL yac_ffinalize()

- frees all internal data structures related to coupling
- MPI communicators may no longer be available
- will call MPI_FINALIZE
  - if MPI_INIT has been called by yac_finit
  - if MPI_FINALIZE has not already been called
Recommended calling sequence

CALL MPI_init ( ... )

CALL yac_finit ( ... )

CALL yac_finit_comp ( ... )

CALL yac_fsearch ( ... )

CALL yac_fget_local_comm ( ... )

CALL yac_fffinalize ( )

CALL MPI_finalize ( ... )
<?xml version="1.0" encoding="UTF-8"?>
<component
    xmlns="http://www.w3schools.com"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3schools.component.xsd">
    <id>1</id>
    <name>atmo</name>
    <model>ICON</model>
    <simulated>atmosphere</simulated>
    <transient_grid_refs>
        <transient_grid_ref collection_size="2" grid_ref="1" id="1" transient_ref="1"/>
        <transient_grid_ref collection_size="2" grid_ref="1" id="2" transient_ref="2"/>
        <transient_grid_ref collection_size="3" grid_ref="1" id="3" transient_ref="3"/>
        <transient_grid_ref collection_size="4" grid_ref="1" id="4" transient_ref="4"/>
        <transient_grid_ref collection_size="4" grid_ref="1" id="5" transient_ref="5"/>
        <transient_grid_ref collection_size="1" grid_ref="1" id="6" transient_ref="6"/>
        <transient_grid_ref collection_size="1" grid_ref="1" id="7" transient_ref="7"/>
        <transient_grid_ref collection_size="1" grid_ref="1" id="8" transient_ref="8"/>
        <transient_grid_ref collection_size="5" grid_ref="1" id="9" transient_ref="9"/>
    </transient_grid_refs>
    ...
</component>
YAC – Component XML Configuration

... 

<transients>
    <transient id="1" transient_standard_name="surface_downward_eastward_stress"/>
    <transient id="2" transient_standard_name="surface_downward_northward_stress"/>
    <transient id="3" transient_standard_name="surface_fresh_water_flux"/>
    <transient id="4" transient_standard_name="total_heat_flux"/>
    <transient id="5" transient_standard_name="atmosphere_sea_ice_bundle"/>
    <transient id="6" transient_standard_name="sea_surface_temperature"/>
    <transient id="7" transient_standard_name="eastward_sea_water_velocity"/>
    <transient id="8" transient_standard_name="northward_sea_water_velocity"/>
    <transient id="9" transient_standard_name="ocean_sea_ice_bundle"/>
</transients>
<brackets>
    <grid id="1" alias_name="atmos_grid"/>
</brackets>
<?xml version="1.0" encoding="UTF-8"?>
<component
 xmlns="http://www.w3schools.com"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.w3schools.component.xsd">
  <id>1</id>
  <name>atmo</name>
  <model>ICON</model>
  <simulated>atmosphere</simulated>
  <transient_grid.refs>
    <transient_grid_ref collection_size="2" grid_ref="1" id="1" transient_ref="1"/>
    <transient_grid_ref collection_size="2" grid_ref="1" id="2" transient_ref="2"/>
    <transient_grid_ref collection_size="3" grid_ref="1" id="3" transient_ref="3"/>
    <transient_grid_ref collection_size="4" grid_ref="1" id="4" transient_ref="4"/>
    <transient_grid_ref collection_size="4" grid_ref="1" id="5" transient_ref="5"/>
    <transient_grid_ref collection_size="1" grid_ref="1" id="6" transient_ref="6"/>
    <transient_grid_ref collection_size="1" grid_ref="1" id="7" transient_ref="7"/>
    <transient_grid_ref collection_size="1" grid_ref="1" id="8" transient_ref="8"/>
    <transient_grid_ref collection_size="5" grid_ref="1" id="9" transient_ref="9"/>
  </transient_grid.refs>
  ...
</component>
YAC – Component XML Configuration

CALL yac_fdef_comp ( "atmo", comp_id )

<name>atmo</name>
<model>ICON</model>
<simulated>atmosphere</simulated>
<?xml version="1.0" encoding="UTF-8"?>
<component
    xmlns="http://www.w3schools.com"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3schools.component.xsd">
  <id>1</id>
  <name>atmo</name>
  <model>ICON</model>
  <simulated>atmosphere</simulated>
  <transient_grid_refs>
    <transient_grid_ref collection_size="2" grid_ref="1" id="1" transient_ref="1"/>
    <transient_grid_ref collection_size="2" grid_ref="1" id="2" transient_ref="2"/>
    <transient_grid_ref collection_size="3" grid_ref="1" id="3" transient_ref="3"/>
    <transient_grid_ref collection_size="4" grid_ref="1" id="4" transient_ref="4"/>
    <transient_grid_ref collection_size="4" grid_ref="1" id="5" transient_ref="5"/>
    <transient_grid_ref collection_size="1" grid_ref="1" id="6" transient_ref="6"/>
    <transient_grid_ref collection_size="1" grid_ref="1" id="7" transient_ref="7"/>
    <transient_grid_ref collection_size="1" grid_ref="1" id="8" transient_ref="8"/>
    <transient_grid_ref collection_size="5" grid_ref="1" id="9" transient_ref="9"/>
  </transient_grid_refs>
  ...
</component>
YAC – Component XML Configuration

```xml
<transient_grid_refs>
  <transient_grid_ref collection_size="2" grid_ref="1" id="1" transient_ref="1"/>
  <transient_grid_ref collection_size="2" grid_ref="1" id="2" transient_ref="2"/>
  <transient_grid_ref collection_size="3" grid_ref="1" id="3" transient_ref="3"/>
  ...
  <transient_grid_ref collection_size="5" grid_ref="1" id="9" transient_ref="9"/>
</transient_grid_refs>

CALL yac_fput ( field_id, nbr_hor_points, 5, &
  & buffer(1:nbr_hor_points,1:5), &
  & info, ierrror )

CALL yac_fget ( field_id, nbr_hor_points, 2, &
  & buffer(1:nbr_hor_points,1:2), &
  & info, ierrror )
```
<transients>
  <transient id="1" transient_standard_name="surface_downward_eastward_stress"/>
  <transient id="2" transient_standard_name="surface_downward_northward_stress"/>
  <transient id="3" transient_standard_name="surface_fresh_water_flux"/>
  <transient id="4" transient_standard_name="total_heat_flux"/>
  <transient id="5" transient_standard_name="atmosphere_sea_ice_bundle"/>
  <transient id="6" transient_standard_name="sea_surface_temperature"/>
  <transient id="7" transient_standard_name="eastward_sea_water_velocity"/>
  <transient id="8" transient_standard_name="northward_sea_water_velocity"/>
  <transient id="9" transient_standard_name="ocean_sea_ice_bundle"/>
</transients>
<grids>
  <grid id="1" alias_name="atmos_grid"/>
</grids>
YAC – Component XML Configuration

```xml
<transients>
  <transient id="1" transient_standard_name="surface_downward_eastward_stress"/>
  <transient id="2" transient_standard_name="surface_downward_northward_stress"/>
  <transient id="3" transient_standard_name="surface_fresh_water_flux"/>
  ...
  <transient id="9" transient_standard_name="ocean_sea_ice_bundle"/>
</transients>

CALL yac_fdef_field &
  & ( "surface_downward_eastward_stress", &
  & component_id, grid_id, point_id, &
  & 1, field_id(1) )

...  

CALL yac_fdef_field &
  & ( "ocean_sea_ice_bundle", &
  & component_id, grid_id, point_id, &
  & 1, field_id(9) )
```
YAC – Component XML Configuration

```xml
<grids>
  <grid id="1" alias_name="atmos_grid"/>
</grids>

CALL yac_fdef_grid ( "atmos_grid",

[ ... ],

grid_id )
```
<transient_grid_refs>
    <transient_grid_ref collection_size="2" grid_ref="1" id="1" transient_ref="1"/>
    <transient_grid_ref collection_size="2" grid_ref="1" id="2" transient_ref="2"/>
    <transient_grid_ref collection_size="3" grid_ref="1" id="3" transient_ref="3"/>
    <transient_grid_ref collection_size="4" grid_ref="1" id="4" transient_ref="4"/>
    ...
    <transient_grid_ref collection_size="5" grid_ref="1" id="9" transient_ref="9"/>
</transient_grid_refs>

<transients>
    <transient id="1" transient_standard_name="surface_downward_eastward_stress"/>
    <transient id="2" transient_standard_name="surface_downward_northward_stress"/>
    <transient id="3" transient_standard_name="surface_fresh_water_flux"/>
    <transient id="4" transient_standard_name="total_heat_flux"/>
    ...
    <transient id="9" transient_standard_name="ocean_sea_ice_bundle"/>
</transients>

<grids>
    <grid id="1" alias_name="grid1"/>
</grids>
YAC – XML Configuration

Coupling GUI

File

New Coupling

Transients

<table>
<thead>
<tr>
<th>Atmo</th>
<th>Ocean</th>
</tr>
</thead>
<tbody>
<tr>
<td>total_heat_flux</td>
<td>total_heat_flux</td>
</tr>
<tr>
<td>Grid: grid1</td>
<td>Grid: grid1</td>
</tr>
<tr>
<td>collect. size: 4</td>
<td>collect. size: 4</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>atmosphere_sea_ice_bundle</td>
<td>atmosphere_sea_ice_bundle</td>
</tr>
<tr>
<td>Grid: grid1</td>
<td>Grid: grid1</td>
</tr>
<tr>
<td>collect. size: 4</td>
<td>collect. size: 4</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>sea_surface_temperature</td>
<td>sea_surface_temperature</td>
</tr>
<tr>
<td>Grid: grid1</td>
<td>Grid: grid1</td>
</tr>
<tr>
<td>collect. size: 1</td>
<td>collect. size: 1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>eastward_sea_water_velocity</td>
<td>eastward_sea_water_velocity</td>
</tr>
<tr>
<td>Grid: grid1</td>
<td>Grid: grid1</td>
</tr>
<tr>
<td>collect. size: 1</td>
<td>collect. size: 1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>northward_sea_water_velocity</td>
<td>northward_sea_water_velocity</td>
</tr>
<tr>
<td>Grid: grid1</td>
<td>Grid: grid1</td>
</tr>
<tr>
<td>collect. size: 1</td>
<td>collect. size: 1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Basic settings

Calendar: proleptic-gregorian
Start date: 1800-01-01T00:00:00.000000
End date: 2100-01-01T00:00:00.000000
Timestep unit: second
Stdout redirect
Root redirect
YAC – XML Configuration
YAC – XML Configuration

Coupling parameters for:
eastward_sea_water_velocity (grid1 -> grid1)

Source timestep: 3600 seconds

Target timestep: 1200 seconds

Coupling period: 3600 seconds

Operation: accumulate

Source Time Lag: 1 model timestep

Target Time Lag: 2 model timestep

Forget default parameters

Save Close
YAC – XML Configuration

Source time step

• time interval between two consecutive calls to yac_fput

Target time step

• time interval between two consecutive calls to yac_fget

Requirement

Source or target time step must be equal to or an integer multiple of the other.
Coupling period

- Time interval at which data are exchanged (with internal calls to MPI_SEND and MPI_RECV)

Requirement

Coupling period must be an integer multiple of the source/target time step
### Related Pages

Here is a list of all related documentation pages:

<table>
<thead>
<tr>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere Partitioning Algorithm</td>
</tr>
<tr>
<td>Polygon clipping in YAC</td>
</tr>
<tr>
<td>Example on how to use XML routines from config_xml.h</td>
</tr>
<tr>
<td><strong>Configuration examples for different systems</strong></td>
</tr>
<tr>
<td>Tips 'n Tricks for developers</td>
</tr>
<tr>
<td>Description of how to build and run the Java GUI</td>
</tr>
<tr>
<td>The c interface (yac_interface.h)</td>
</tr>
<tr>
<td>The Fortran interface (yac_finterface.f90 and mo_yac_finterface.f90)</td>
</tr>
<tr>
<td>Patch Recovery in YAC</td>
</tr>
<tr>
<td>Issue with Patch Recovery in YAC</td>
</tr>
<tr>
<td>Condensed release information</td>
</tr>
<tr>
<td>Todo List</td>
</tr>
</tbody>
</table>
Doxygen

http://dkrz-sw.gitlab-pages.dkrz.de/yac/

Source Code (version 2.0.0)

  git clone -b 'release-2.0.0' --single-branch --depth 1 git@gitlab.dkrz.de:YAC/YAC.git

Latest version (untagged)

  git clone git@gitlab.dkrz.de:YAC/YAC.git

Documentation with further Links

  - https://www.geosci-model-dev.net/9/2755/2016/
  - https://doi.org/10.5676/dwd_pub/nwv/icon_003
  - https://code.zmaw.de/projects/mpiesm-2/wiki/ICON_Coupled_Model_Development