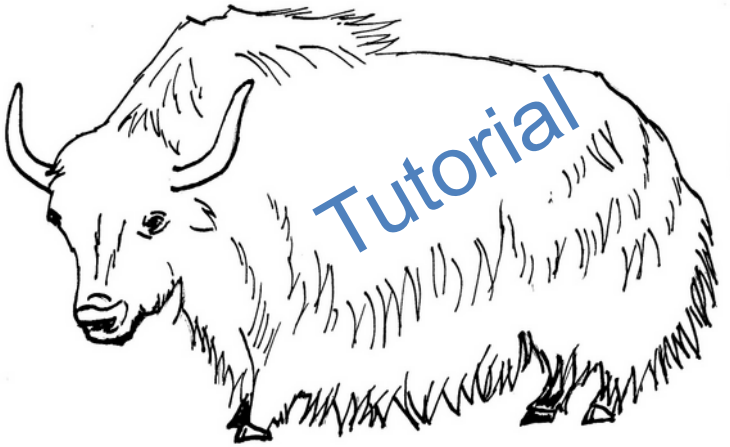
Yet Another Coupler – YAC

Version 2.4.2 – Jan 2022



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

Max-Planck-Institut für Meteorologie

DKRZ DEUTSCHES KLIMARECHENZENTRUM

YAC2 – a rewrite of YAC1

Moritz Hanke (DKRZ)

with contributions from

Sergey Kosukhin (MPI-M)
René Redler (MPI-M)

GUI:

Teresa Holfeld (MPI-M, student assistant)
Maxim Yastremsky (MPI-M, student assistant)

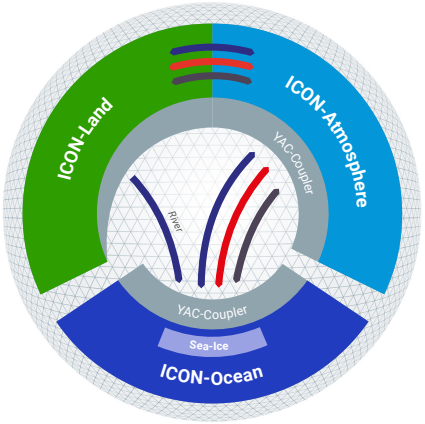
YAC – General remarks

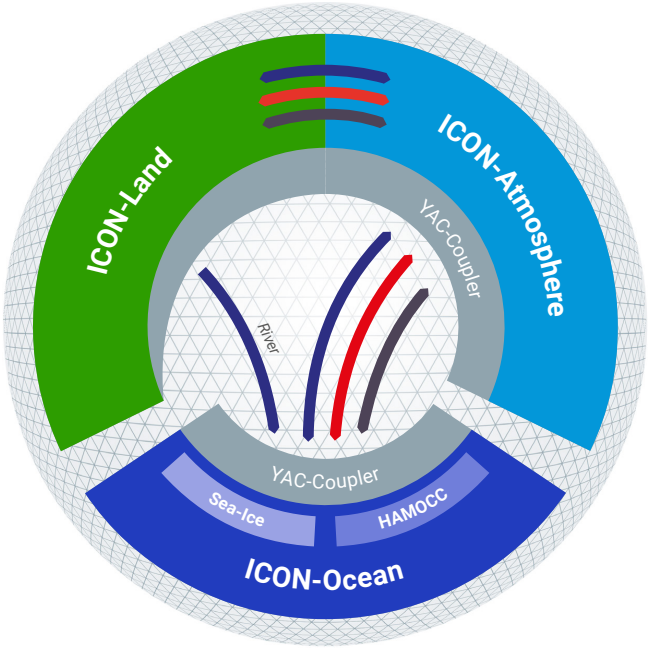
A coupling software ***not only*** for ICON

- Parallel weight computation 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 (>95% of lines covered)
- Fortran and C examples plus toy models
- XML coupling configuration file with GUI support

YAC – General remarks

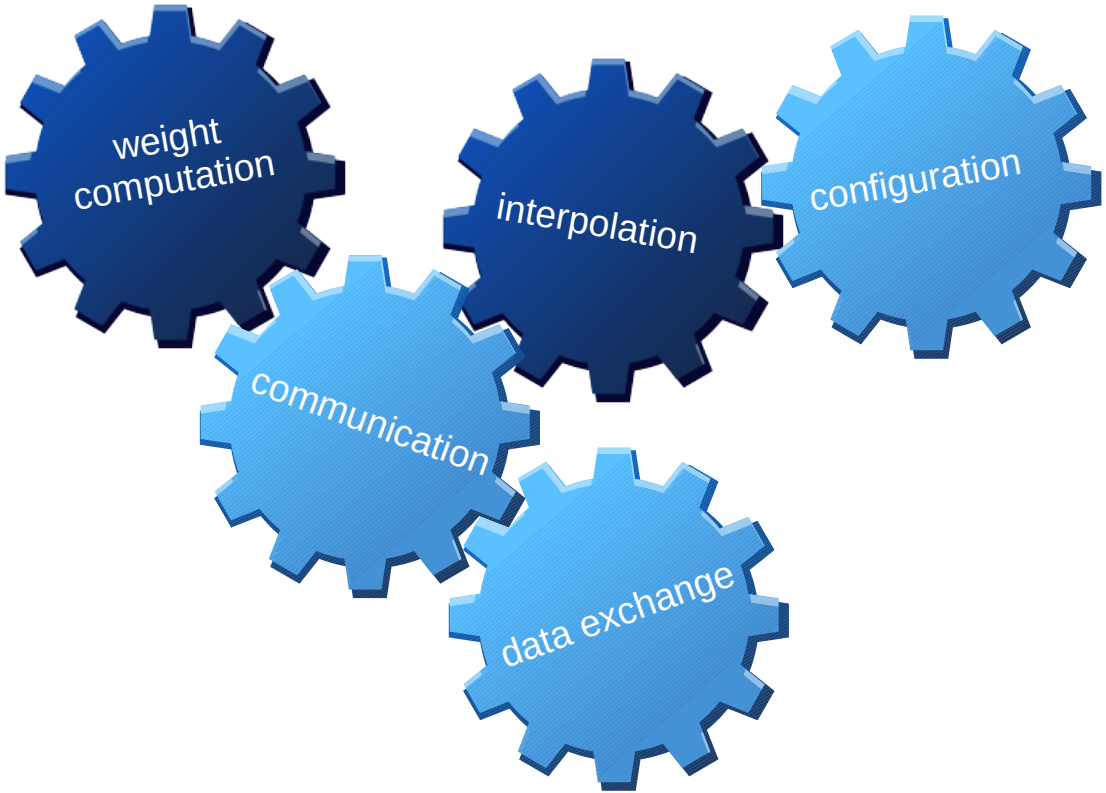




Legend:

- Energy, Momentum
- Water
- Carbon

YAC – General remarks



YAC – Search

required

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

provided

- Initial scalable computation of global mapping
- completely parallelised initialisation and interpolation weight computation with supp

features

- Support for circles of latitude/longitude and great circles
- Convex & moderately concave polygons
- Support for masked cells and points

YAC – Interpolation

Available 2-dimensional (horizontal) interpolation methods

- Conservative remapping (**conserv**)
  - 1st – order
  - 2nd – order
- Hybrid cubic spherical Bernstein-Bézier patch interpolation (**bernstein_bezier**)
- N-nearest-neighbour (**n-nearest_neighbour**)
  - Distance-weighted
  - N-nearest-neighbour average
  - Gauss-weighted N-nearest-neighbour
- Source Point to Target Point Mapping (**source_to_target_map**)
- Fixed value (**fixed**)

Available 2-dimensional (horizontal) interpolation methods (continued)

- Radial Basis Functions (**radial_basis_function**)
- Cell average (**average**)
  - simple
  - distance-weighted cell average
- Interpolation from precomputed NetCDF weight file (**user_file**)

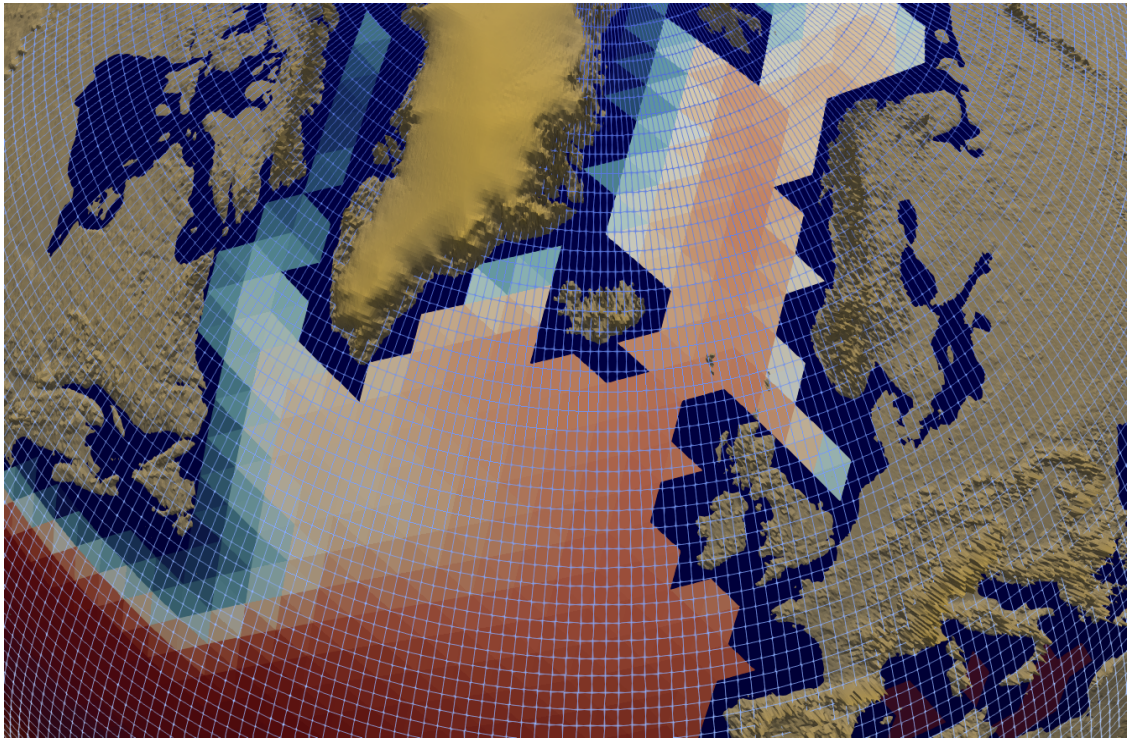
example

interpolation of 1º x 1º World Ocean Atlas 2009 sea surface salinity onto an ICON R2B04 atmosphere grid.

1st-order conservative remapping
plus **Hybrid cubic spherical Bernstein-Bézier interpolation**
plus **fixed value**

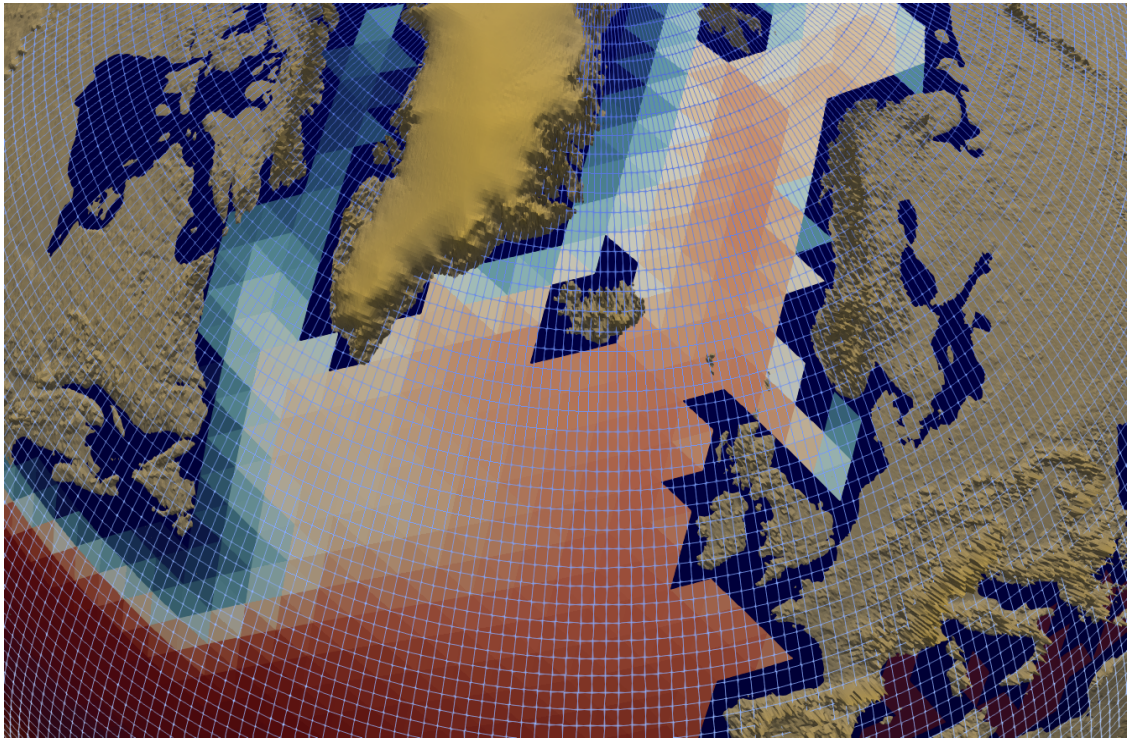
YAC – Interpolation stack

Step 1: 1st- order conservative remapping



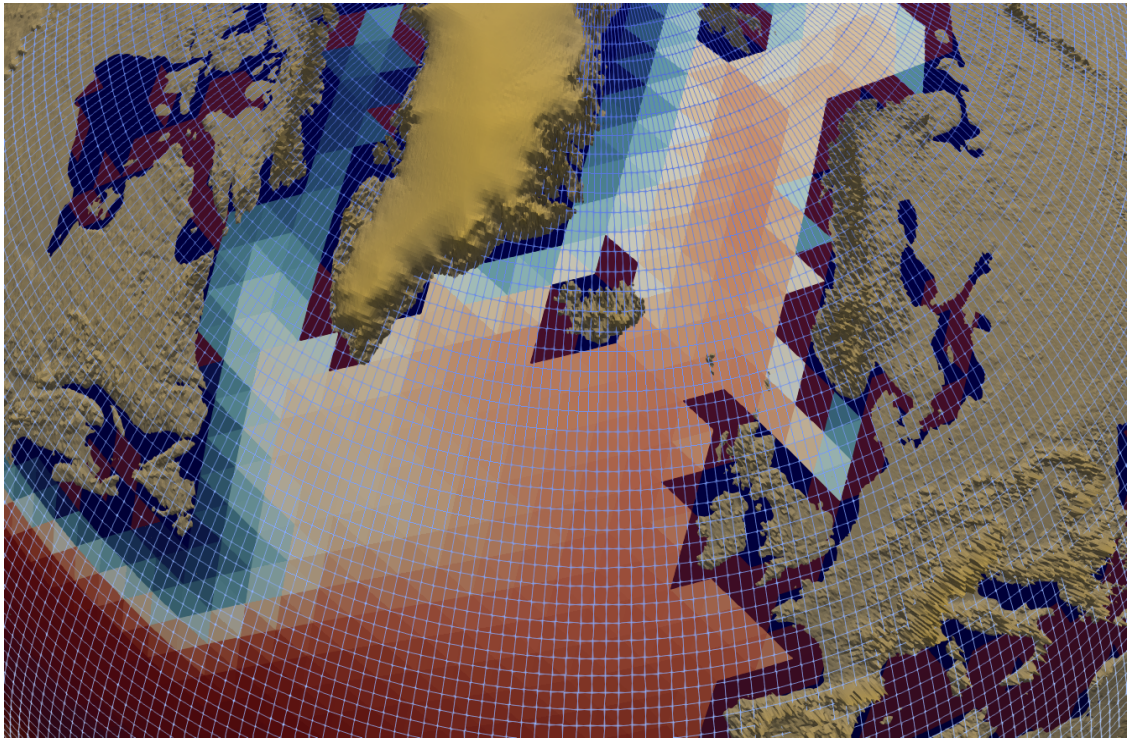
YAC – Interpolation stack

Step 1: … + Hybrid cubic spherical Bernstein-Bézier interpolation

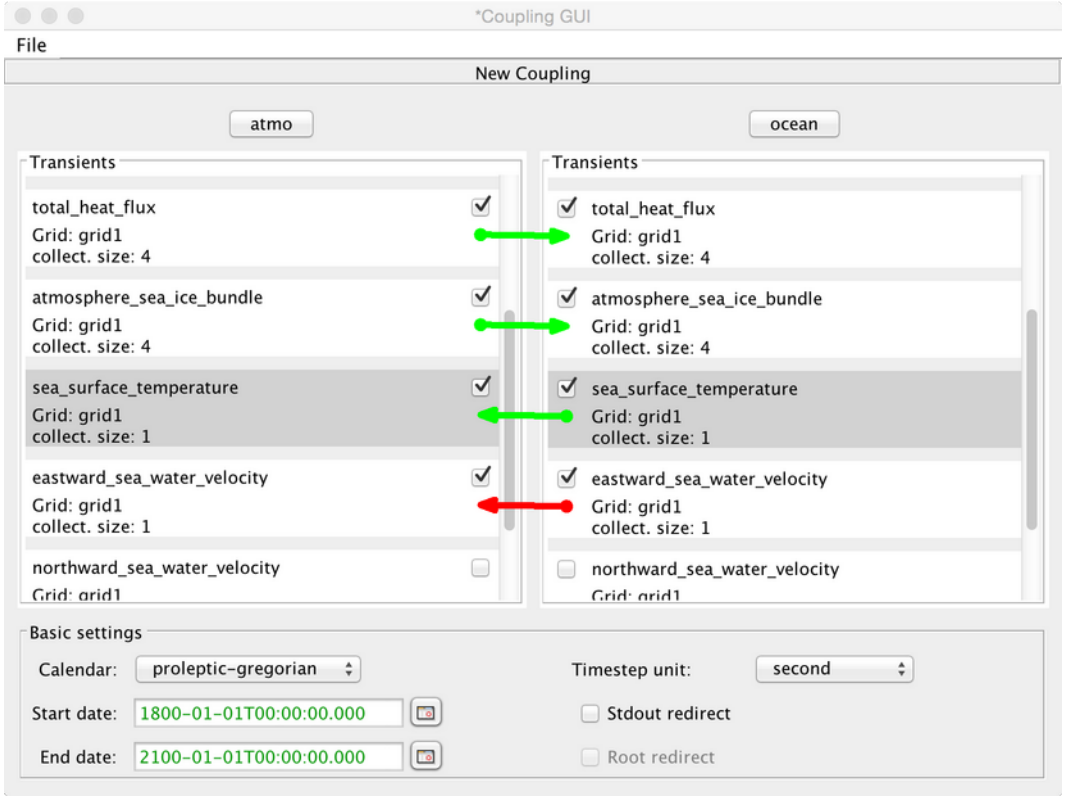


YAC – Interpolation stack

Step 1: … + … + fixed value



YAC – Graphical User Interface



YAC – Fortran Interface

Initialisation Phase
- yac_finit
- yac_fdef_comp
- yac_fdef_datetime

Grid Definition
- yac_fdef_grid
- yac_fdef_points
- yac_fset_global_index
- yac_fset_core_mask
- yac_fdef_mask
- yac_fdef_field_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

YAC – Definition Phase

grid definition *(example for an unstructured grid)*

```
CALL yac_fdef_points ( grid_id,
                       patch_horz%n_patch_cells,
                       YAC_LOCATION_CELL,
                       array_of_cellcenter_longitudes,
                       array_of_cellcenter_latitudes,
                       cell_point_ids )
```

overloaded with respect to

- data type for coordinate arrays
- grid types

YAC – Definition Phase

grid definition

```
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 )
```

mask definition

CALL yac_fdef_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

mask definition

```
CALL yac_fdef_field_mask  (  “field_name”,
                             comp_id,
                             point_id,
                             mask_id,
                             nbr_point_sets,
                             field_id )
```

field definition

```
CALL yac_fdef_field   (   "field_name",
                          component_id,
                          grid_id,
                          array_of_cell_point_ids,
                          nbr_point_sets,
                          point_id    )
```

calculation of weights

CALL yac_fsearch (error_status)

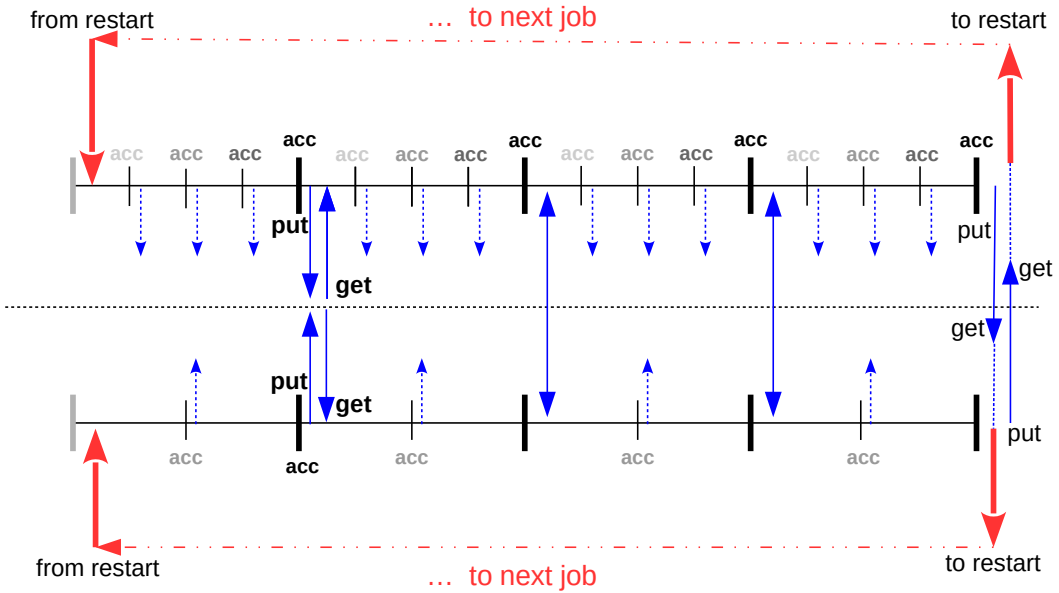
- includes collective MPI operations
- needs to be called by all processes
- accesses the coupling configuration
- invokes the neighbourhood search
- does the communicator splitting

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 )
```

YAC – Exchange Phase

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  …
ENDIF
```

data exchange

Return values for the info argument

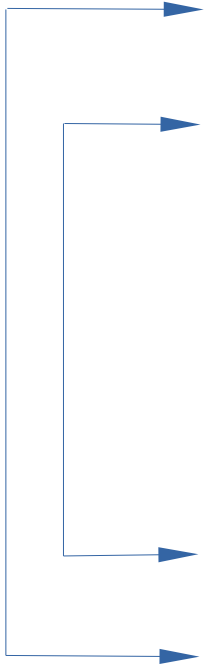
```
enum, bind(c)
  enumerator :: NONE                = 0
  enumerator :: COUPLING            = 1
  enumerator :: RESTART             = 2
  enumerator :: GET_FOR_RESTART     = 3
  enumerator :: PUT_FOR_RESTART     = 4
  enumerator :: GET_FOR_CHECKPOINT  = 5
  enumerator :: PUT_FOR_CHECKPOINT  = 6
  enumerator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

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_ffinalize ( )

CALL MPI_finalize (...)

YAC – Component XML Configuration

```
<?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>
        ...
```

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>
        <grids>
            <grid id="1" alias_name="atmos_grid"/>
            <grid id="2" alias_name="ocean_grid"/>
        </grids>
</component>
```

YAC – Component XML Configuration

```
<?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>
        ...
```

YAC – Component XML Configuration

```
<name>atmo</name>
<model>ICON</model>
<simulated>atmosphere</simulated>
```

```
CALL yac_fdef_comp ( "atmo", comp_id )
```

YAC – Component XML Configuration

```
<?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>
        ...
```

YAC – Component XML Configuration

```
<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, ierror )

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

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>
        <grids>
            <grid id="1" alias_name="atmos_grid"/>
            <grid id="2" alias_name="ocean_grid"/>
        </grids>
</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="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

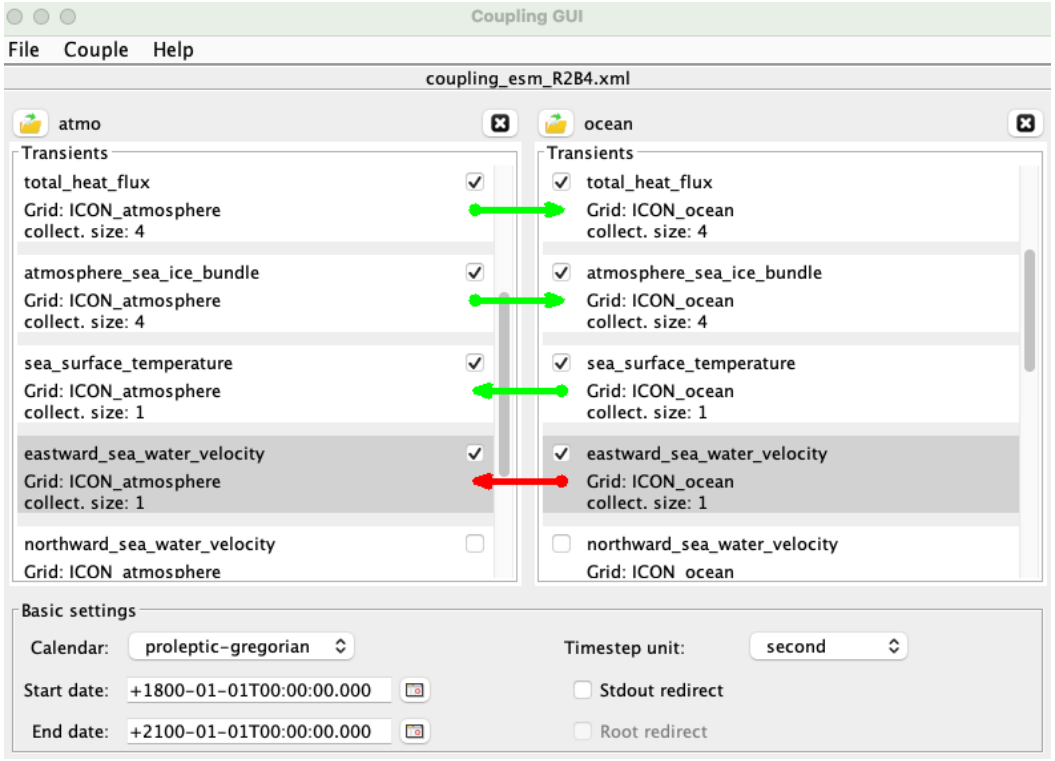
```
<grids>
   <grid id="1" alias_name="atmos_grid"/>
</grids>
```

```
CALL yac_fdef_grid ( "atmos_grid",

                     [ … ],

                     grid_id )
```

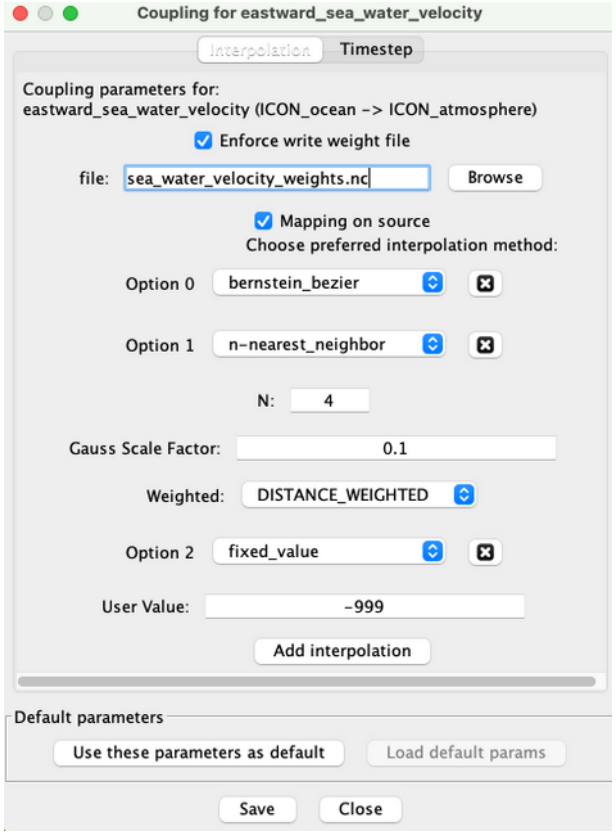
YAC – Component XML Configuration

```
<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="atmos_grid"/>
   <grid id="2" alias_name="ocean_grid"/>
</grids>
```

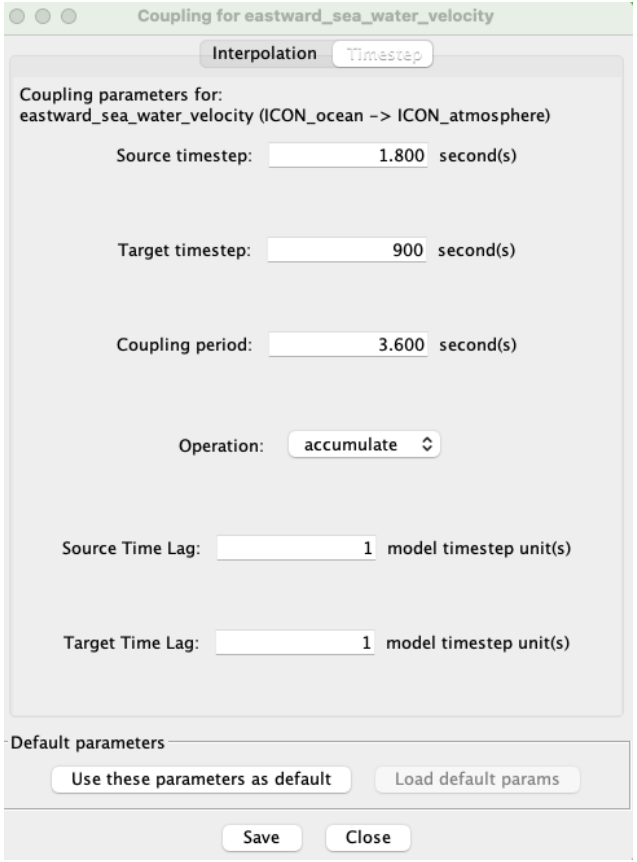
YAC – XML Configuration



YAC – XML Configuration



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.

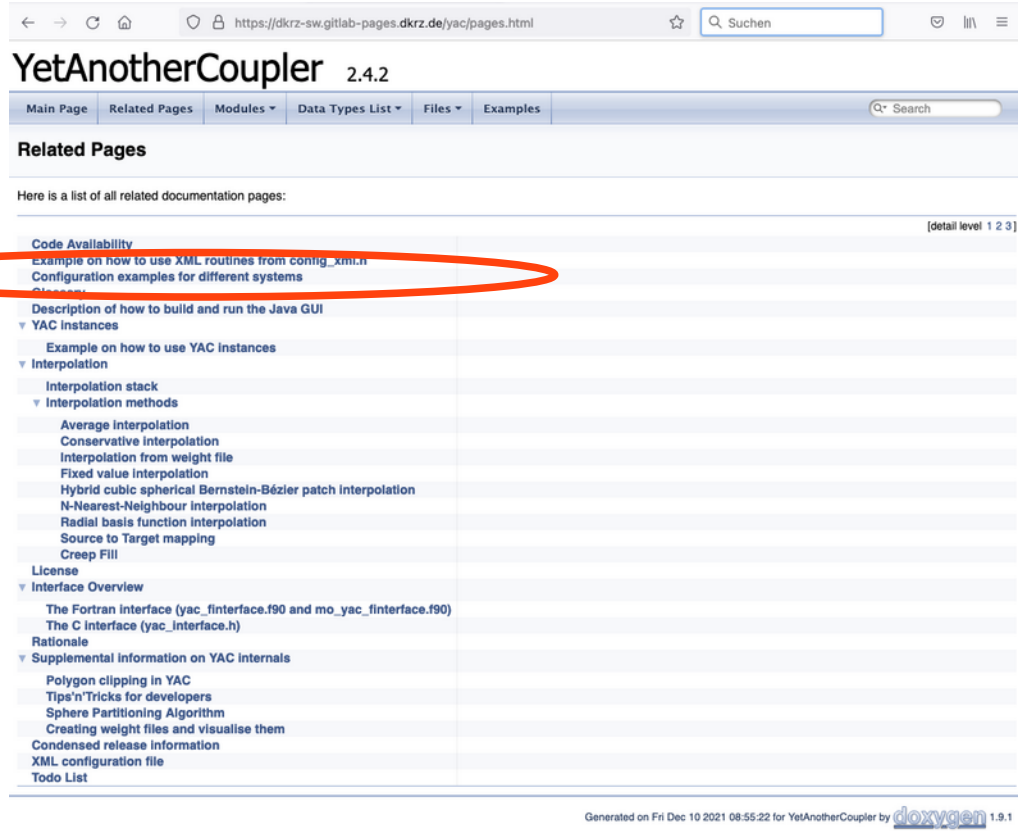
Coupling period

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

Requirement

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

YAC – Building



YAC – Documentation & Code

Doxygen

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

Source Code (version 2.4.2)

git clone -b 'release-2.4.2' --single-branch --depth 1 https://gitlab.dkrz.de/dkrz-sw/yac

Documentation with further Links

https://www.dkrz.de/dienste/softwareentwicklung