Modification of the NEMO OASIS module for multiple 2D/3D coupling

Alexis Barge

4th September 2024

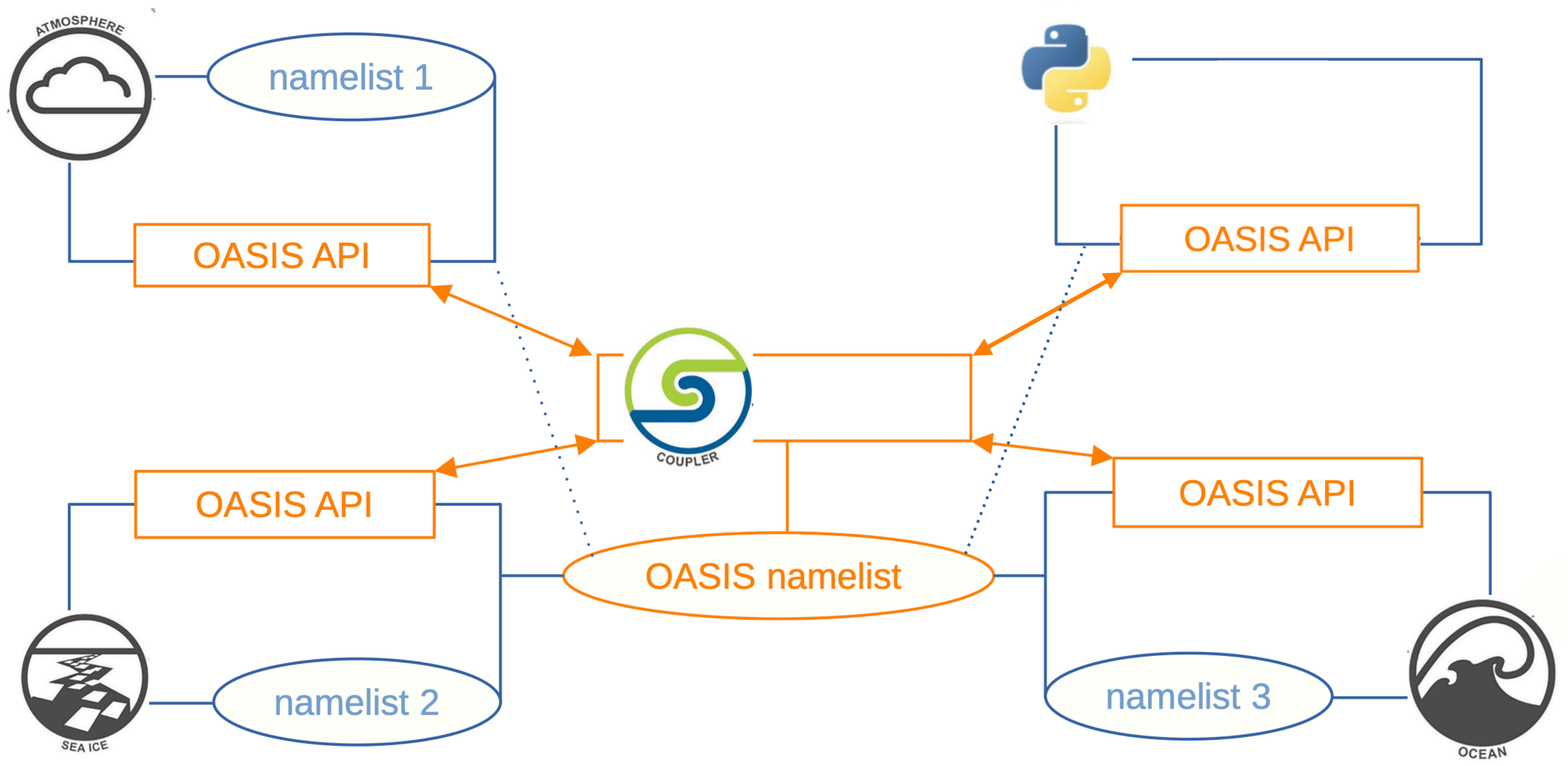
PROGRAMME PRIORITAIRE DE RECHERCHE OCÉAN & CLIMAT

MEDIATION

Motivation: multiple couplings

Coupling needs in NEMO

Surface with atmosphere
Lagrangian tracking of icebergs
PISCES
Hybrid Fortran / Python



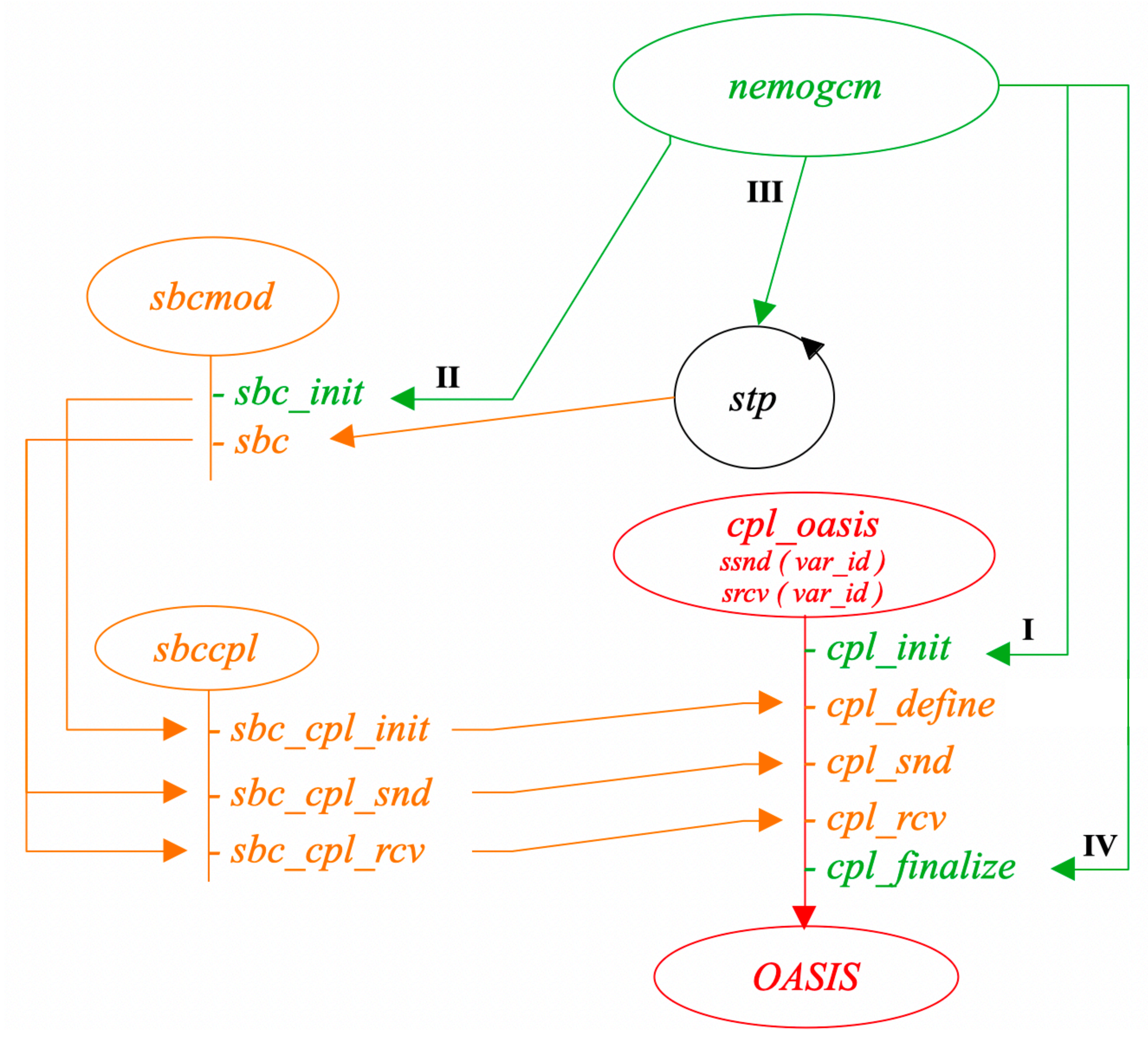
OASIS coupler

Interpolate and exchange 2D/3D fields
Fortran, C and Python API
Implemented in SBC for 2D exchanges only

Needs of multiple modules using OASIS with 3D exchanges

Current OASIS implementation

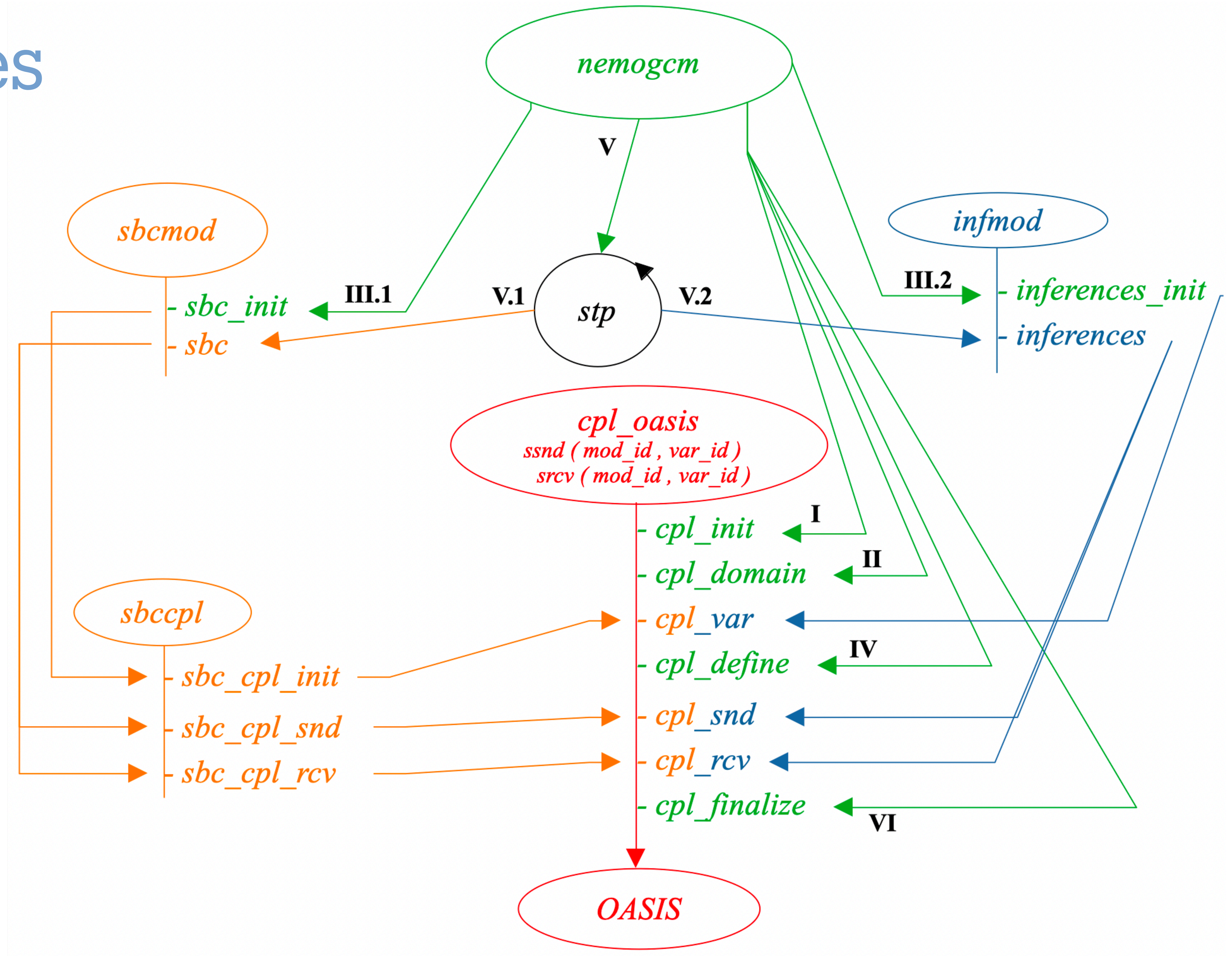
OASIS typical steps	NEMO routines
MPI environment	cpl_init()
Grid partitioning Fields to exchange Setup coupling	cpl_define()
Receive a field	cpl_rcv()
Send a field	cpl_snd()



Coupling definition exclusive to SBC

Independent OASIS module

OASIS typical steps	Splited routines
MPI environment	cpl_init()
Grid partitioning	cpl_domain()
Fields to exchange	cpl_var()
Setup coupling	cpl_define()
Receive a field	cpl_rcv()
Send a field	cpl_snd()



Other modules may participate to coupling definition

3D coupling scattered among modules

Coupling fields properties

```
TYPE, PUBLIC ::   FLD_CPL                       !: Type for coupled field informations
   LOGICAL                    ::    laction    ! To be coupled or not
   CHARACTER(len = 8)         ::    clname     ! Field alias used by OASIS
   CHARACTER(len = 1)         ::    clgrid     ! Grid type
   REAL(wp)                   ::    nsgn       ! Control of the sign change
   INTEGER, DIMENSION(nmaxcat,nmaxcpl) ::   nid    ! Id of the field (no more than 9 categories
   INTEGER                    ::    nct        ! Number of categories in field
   INTEGER                    ::    ncplmodel ! Maximum number of models to/from which this variable
   INTEGER                    ::    nlvl       ! Number of grid level to exchange, set 1 for 2D fields
END TYPE FLD_CPL
```

New attribute for 3D

Arrays of coupling properties ssnd / srcv

Filled during initialization

Sort with modules

ssnd / srcv : now batches of coupling properties arrays

Two-level dimensions with modules and fields IDs

Different size allocation for memory optimization

```
ssnd( mod_ID_1 )%fld( field_ID_1 )%field_property_1
srcv( mod_ID_1 )%fld( field_ID_2 )%field_property_2
ssnd( mod_ID_2 )%fld( field_ID_3 )%field_property_3
```

Module ID passed through OASIS interface

Develop with new OASIS interface

Register a new module

```
vi cpl_oasis3.F90
##   [...]
INTEGER, PUBLIC, PARAMETER ::   nmodmax=2    ! Maximum number of identified modules
INTEGER, PUBLIC, PARAMETER ::   nmodsbc=1    ! module ID #1 : surface boundary condition
INTEGER, PUBLIC, PARAMETER ::   nmodext=2    ! module ID #2 : external communication modul
```

Defined coupled fields

In new module initialization routine

Between cpl_domain() and cpl_define() during NEMO init

Fill up ssnd / srvc and call cpl_var()

```
! default values for ssnd and srcv batches
srcv(nmodext)%fld(:)%laction = .FALSE.  ;  srcv(nmodext)%fld(:)%clgrid = 'T'
srcv(nmodext)%fld(:)%nct = 1  ;  srcv(nmodext)%fld(:)%nlvl = 1
!
ssnd(nmodext)%fld(:)%laction = .FALSE.  ;  ssnd(nmodext)%fld(:)%clgrid = 'T'
ssnd(nmodext)%fld(:)%nct = 1  ;  ssnd(nmodext)%fld(:)%nlvl = 1
```

```
CALL cpl_var( krcv, ksnd, kcplmodel, kmod)
! INTEGER :: krcv      ! number of fields to receive for current module
! INTEGER :: ksnd      ! number of fields to send for current module
! INTEGER :: kcplmodel ! Maximum number of models to/from which NEMO is ,
! INTEGER :: kmod      ! calling module ID
```

Send and Receive

```
CALL cpl_snd( kmod, kid, kstep, pdata, kinfo)
! INTEGER                  :: kmod   ! calling module ID
! INTEGER                  :: kid    ! field index in ssnd for calling module
! INTEGER                  :: kstep  ! ocean time-step in second
! REAL, DIMENSION(:,:,:) :: pdata  ! field to send (shape is (:,:,1) for 2D)
! INTEGER                  :: kinfo  ! OAIS3 info argument
```

```
CALL cpl_rcv( kmod, kid, kstep, pdata, kinfo, pmask)
! INTEGER                  :: kmod   ! calling module ID
! INTEGER                  :: kid    ! field index in srcv for calling module
! INTEGER                  :: kstep  ! ocean time-step in second
! INTEGER                  :: kinfo  ! returned OASIS3 info
! REAL, DIMENSION(:,:,:) :: pdata  ! returned received field (shape is (:,:,1) for 2D)
! REAL, DIMENSION(:,:,:) :: pmask  ! coupling mask, optional
```

https://morays-doc.readthedocs.io/en/latest/nemo.api_5.html

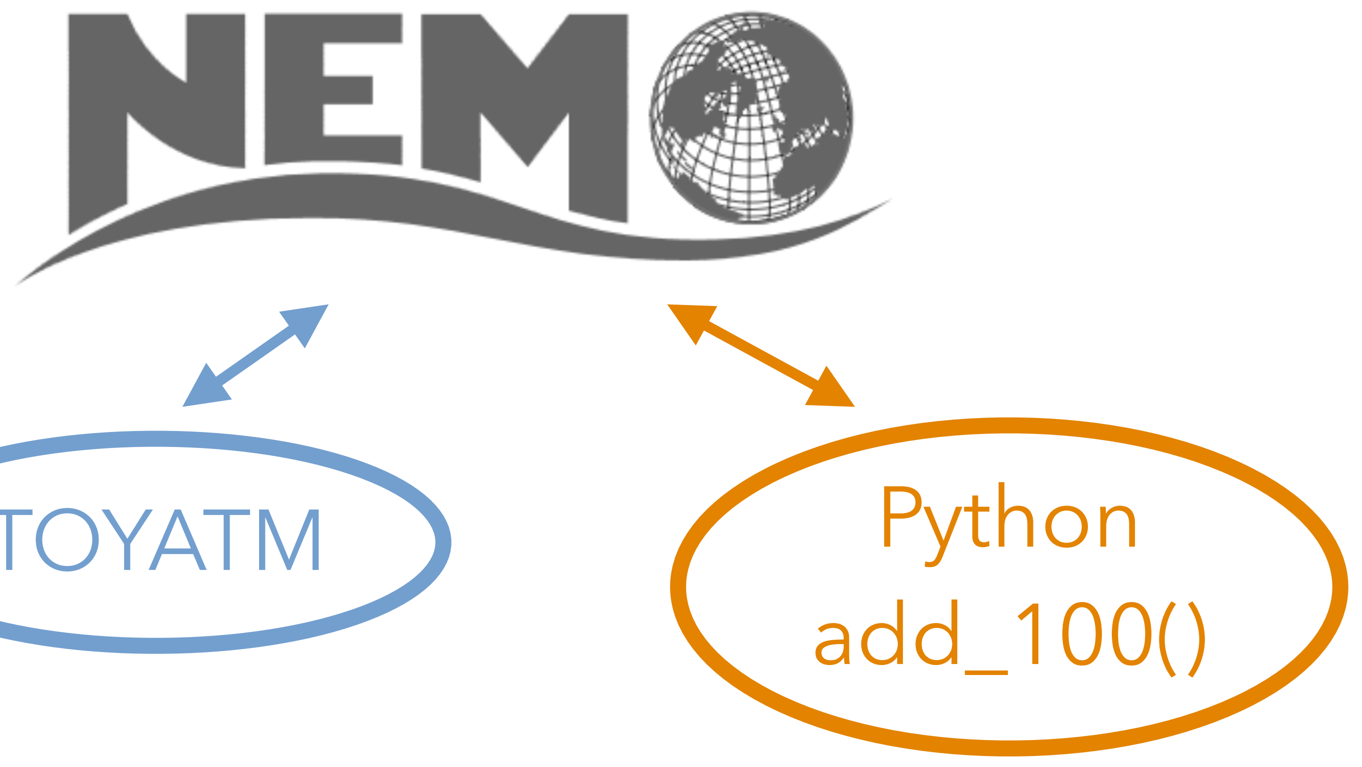
Demonstration

Simple TOYATM coupling

CPL_OASIS test case

Surface coupled ORCA2_ICE_PISCES

Test passed



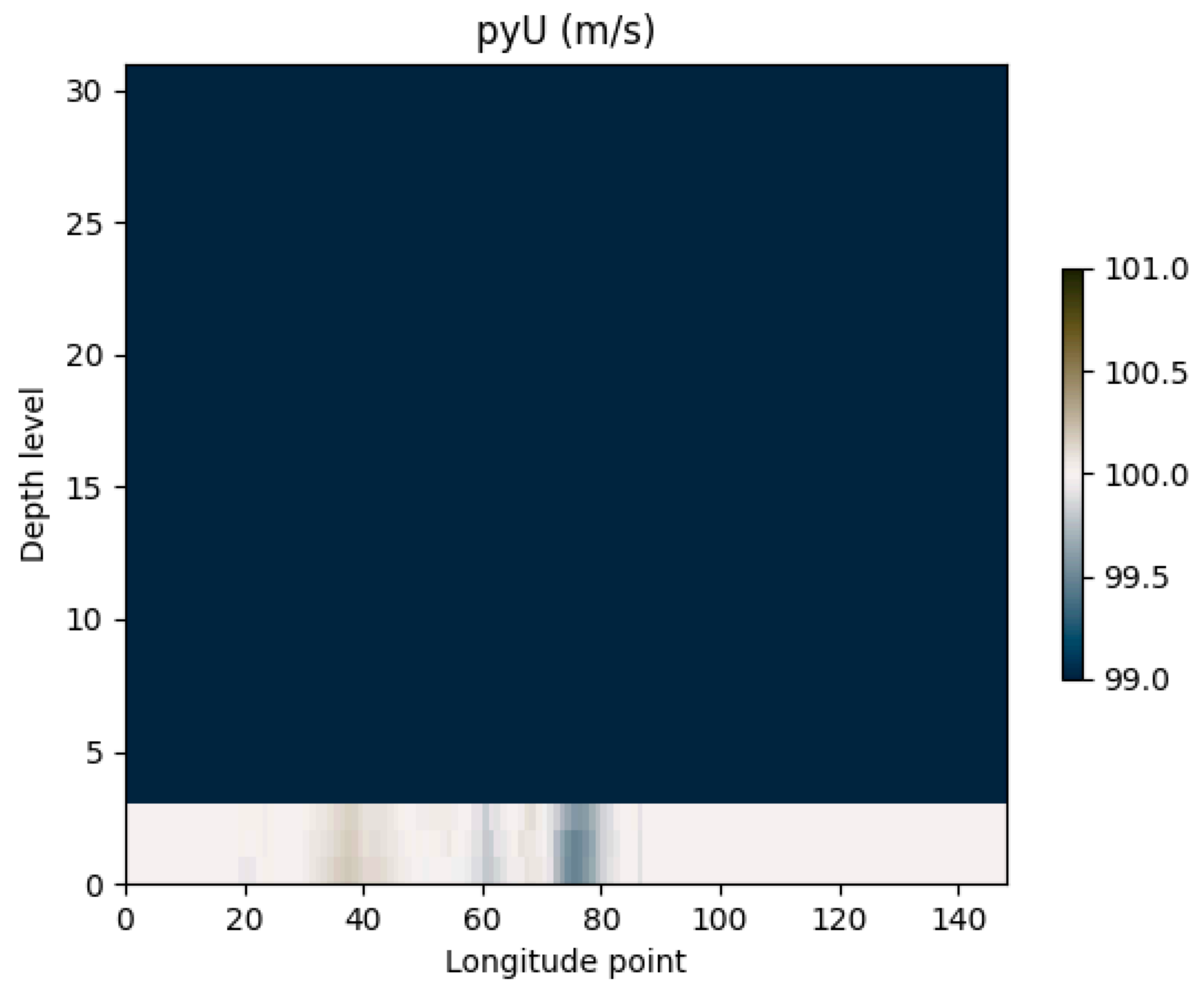
Simple Python coupling

Forced ORCA2_ICE_PISCES config

Demo Python module called after sbc()

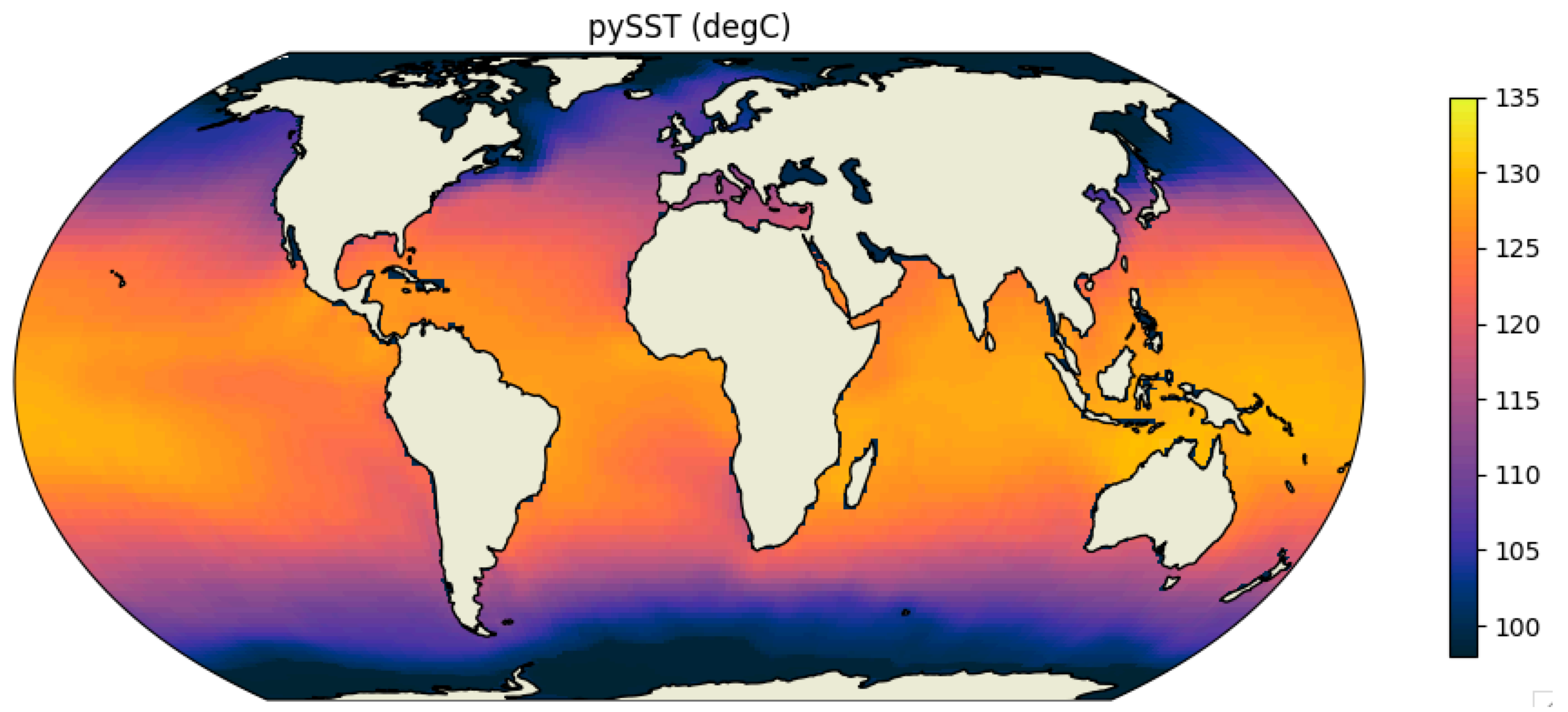
Send *sst* and 3 first levels of *U*

Receive modified *pysst* and *pyU*



Double coupling

CPL_OASIS test case with additional Python module



Conclusion

Summary

Gathered variables related to OASIS module in *cpl_oasis3.F90*
Scattered OASIS API in several NEMO subroutines
Attribute and routines to perform 3D coupling
Module ID in *ssnd/srcv* and *cpl_oasis3.F90* routines
Update *cpl_oasis3.F90* routines

Development

Branch *Fork-14-independent-3D-OASIS-module*
Above modifications pushed without demo Python module
SETTE passed

Branch *Fork-12-oasis-interface-for-python-scripts*
Dedicated to flexible communication module with Python