Numerical Modelling

Jonathan GULA
gula@univ-brest.fr

the anatomy of an ocean model

Outline

- **Lesson 1 :** [D109]
  - Introduction
  - Equations of motions
  - *Activity 1 [run an ocean model]*
- **Lesson 2 :** [D109]
  - Subgrid-scale parameterization
  - Dynamics of the ocean gyre
  - *Activity 2 [Dynamics of an ocean gyre]*
- **Lesson 3 :** [D109]
  - Horizontal Discretization
  - Vertical coordinates
  - *Activity 2 [Dynamics of an ocean gyre]*
  - *Activity 3 [Impacts of numerics / topography]*
- **Lesson 4 :** [D109]
  - Numerical schemes
  - *Activity 3 [Impacts of numerics / topography]*
- **Lesson 5 :** [D109]
  - Dynamics of the ocean gyre
  - Presentation of the model CROCO
  - *Activity 3 [Impacts of numerics / topography]*
- **Lesson 6 :** [D109]
  - Boundary Forcings
  - *Activity 4 [Design a realistic simulation]*
- **Lesson 7 :** [D109]
  - Diagnostics and validation
  - *Activity 4 [Analyze a realistic simulation]*

Presentations and material will be available at :

jgula.fr/ModNum/

#6
Numerical options in CROCO

Master's degree 2nd year Marine Physics

Jonathan GULA
gula@univ-brest.fr

Choice of numerics:

- cppdefs.h

```
#if defined REGIONAL
/*
!====================================================================
!               REGIONAL (realistic) Configurations
!====================================================================
!
!----------------------
! BASIC OPTIONS
!----------------------
!
*/
                      /* Configuration Name */
# define BENGUELA_LR
                      /* Parallelization */
# undef  OPENMP
# undef  MPI
                      /* Nesting */
# undef  AGRIF
# undef  AGRIF_2WAY
                      /* OA and OW Coupling via OASIS (MPI) */
# undef  OA_COUPLING
# undef  OW_COUPLING
                      /* I/O server */
# undef  XIOS
                      /* Open Boundary Conditions */
# undef  TIDES
# define OBC_EAST
# define OBC_WEST
# define OBC_NORTH
# define OBC_SOUTH
                      /* Applications */
# undef  BIOLOGY
# undef  FLOATS
# undef  STATIONS
# undef  PASSIVE_TRACER
# undef  SEDIMENT
# undef  BBL
/*!
```

- cppdefs_dev.h

```
======================================================================
    Select MOMENTUM LATERAL advection-diffusion scheme:
    (The default is third-order upstream biased)
======================================================================
*/
#ifdef UV_HADV_UP3     /* Check if options are defined in cppdefs.h */
#elif defined UV_HADV_C4
#elif defined UV_HADV_C2
#else
# define UV_HADV_UP3       /* 3rd-order upstream lateral advection */
# undef  UV_HADV_C4        /* 4th-order centered lateral advection */
# undef  UV_HADV_C2        /* 2nd-order centered lateral advection */
#endif
/*
   UV DIFFUSION: set default orientation
*/
#ifdef UV_MIX_S        /* Check if options are defined */
#elif defined UV_MIX_GEO
#else
# define UV_MIX_S      /* Default: diffusion along sigma surfaces */
#endif
/*
   Set keys related to Smagorinsky viscosity
*/
#ifdef UV_VIS_SMAGO
# define VIS_COEF_3D
#endif
/*
   Set UP3 scheme in barotropic equations for 2DH applications
*/
#if !defined SOLVE3D && !defined SOLITON
# define M2_HADV_UP3
#endif
/*
   If interior MOMENTUM LATERAL diffusion is defined, apply it
   over an anomaly with respect to a reference frame (climatology)
*/
#ifdef M3CLIMATOLOGY
# undef CLIMAT_UV_MIXH
#endif
```

cppdefs.h

```
#if defined REGIONAL
/*
!====================================================================
!              REGIONAL (realistic) Configurations
!====================================================================
!
!----------------------
! BASIC OPTIONS
!----------------------
!
*/
                        /* Configuration Name */
# define BENGUELA_LR
                        /* Parallelization */
# undef  OPENMP
# undef  MPI
                        /* Nesting */
# undef  AGRIF
# undef  AGRIF_2WAY
                        /* OA and OW Coupling via OASIS (MPI) */
# undef  OA_COUPLING
# undef  OW_COUPLING
                        /* I/O server */
# undef  XIOS
                        /* Open Boundary Conditions */
# undef  TIDES
# define OBC_EAST
# define OBC_WEST
# define OBC_NORTH
# define OBC_SOUTH
                        /* Applications */
# undef  BIOLOGY
# undef  FLOATS
# undef  STATIONS
# undef  PASSIVE_TRACER
# undef  SEDIMENT
# undef  BBL
/*!
```

Configuration name:

Used to define configuration specific part of the codes in param.h, ana_initial.F, analytical.F, etc.

```
#  elif defined PACIFIC
      parameter (LLm0=170,  MMm0=60,   N=30)   ! Pacific
#  elif defined  CORAL
      parameter (LLm0=81,   MMm0=77,   N=32)   ! CORAL sea
#  elif defined  BENGUELA_LR
      parameter (LLm0=41,   MMm0=42,   N=32)   ! BENGUELA_LR
#  elif defined  BENGUELA_HR
      parameter (LLm0=83,   MMm0=85,   N=32)   ! BENGUELA_HR
#  elif defined  BENGUELA_VHR
      parameter (LLm0=167,  MMm0=170,  N=32)   ! BENGUELA_VHR
#  elif defined  GULFSTREAM
      parameter (LLm0=134,  MMm0=112,  N=32)   ! GULFSTREAM
#  else
      parameter (LLm0=94,   MMm0=81,   N=40)
#  endif
```

cppdefs.h

```
#if defined REGIONAL
/*
!====================================================================
!                REGIONAL (realistic) Configurations
!====================================================================
!
!----------------------
! BASIC OPTIONS
!----------------------
!
*/
                        /* Configuration Name */
# define BENGUELA_LR
                        /* Parallelization */
# undef  OPENMP
# undef  MPI
                        /* Nesting */
# undef  AGRIF
# undef  AGRIF_2WAY
                        /* OA and OW Coupling via OASIS (MPI) */
# undef  OA_COUPLING
# undef  OW_COUPLING
                        /* I/O server */
# undef  XIOS
                        /* Open Boundary Conditions */
# undef  TIDES
# define OBC_EAST
# define OBC_WEST
# define OBC_NORTH
# define OBC_SOUTH
                        /* Applications */
# undef  BIOLOGY
# undef  FLOATS
# undef  STATIONS
# undef  PASSIVE_TRACER
# undef  SEDIMENT
# undef  BBL
/*!
```

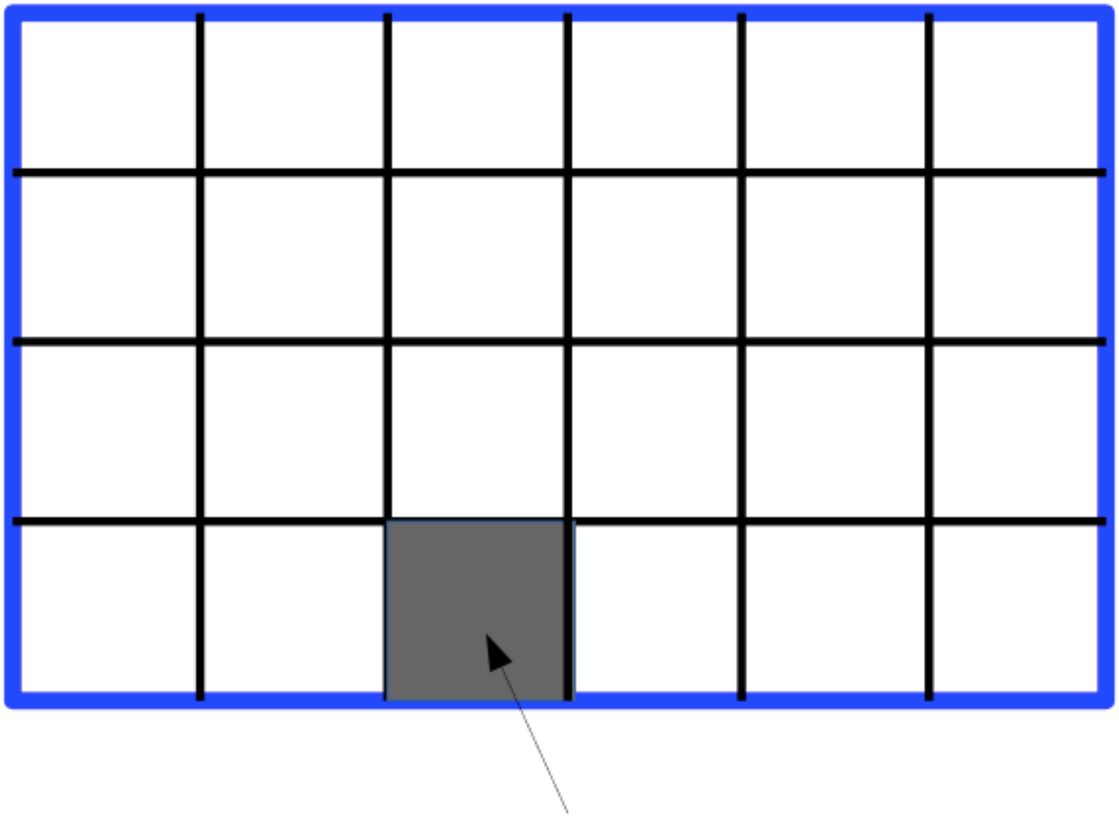
Parallelization options:

Activation of MPI and/or openMP parallelization:

Numbers of threads are defined in param.h

```
!
! Domain subdivision parameters
! ====== =========== ==========
!
! NPP           Maximum allowed number of parallel threads;
! NSUB_X,NSUB_E  Number of SHARED memory subdomains in XI- and
!                                                   ETA-directions;
! NNODES        Total number of MPI processes (nodes);
! NP_XI,NP_ETA  Number of MPI subdomains in XI- and ETA-directions;
!
      integer NSUB_X, NSUB_E, NPP
#ifdef MPI
      integer NP_XI, NP_ETA, NNODES
      parameter (NP_XI=2, NP_ETA=1,  NNODES=NP_XI*NP_ETA)
      parameter (NPP=1)
      parameter (NSUB_X=1, NSUB_E=1)
#elif defined OPENMP
      parameter (NPP=8)
```

define OPENMP



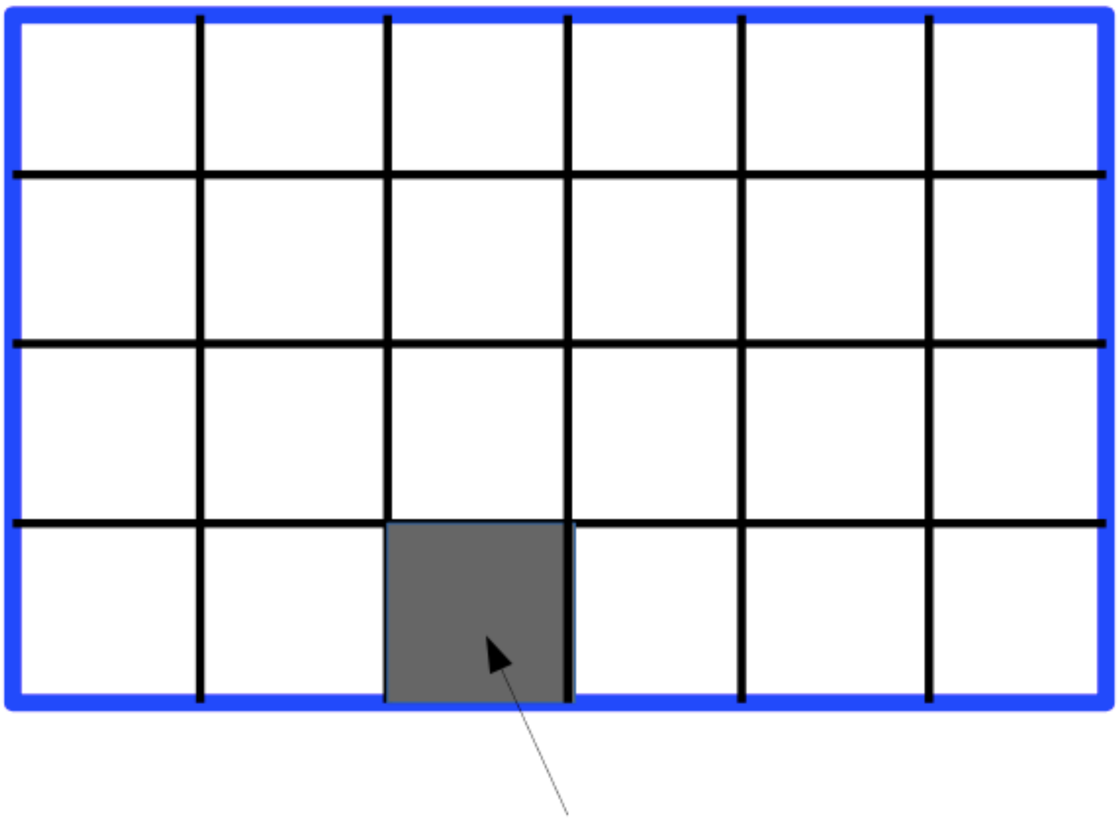
openMP = shared – memory

NPP = number of threads (cores)

NSUB_X , NSUB_E = number of tiles in both directions.

NSUB_X * NSUB_E has to be a multiple of NPP.

define OPENMP



Ex: NSUB_X = 6, NSUB_E = 4

openMP = shared – memory

NPP = number of threads (cores)

NSUB_X , NSUB_E = number of tiles in both directions.

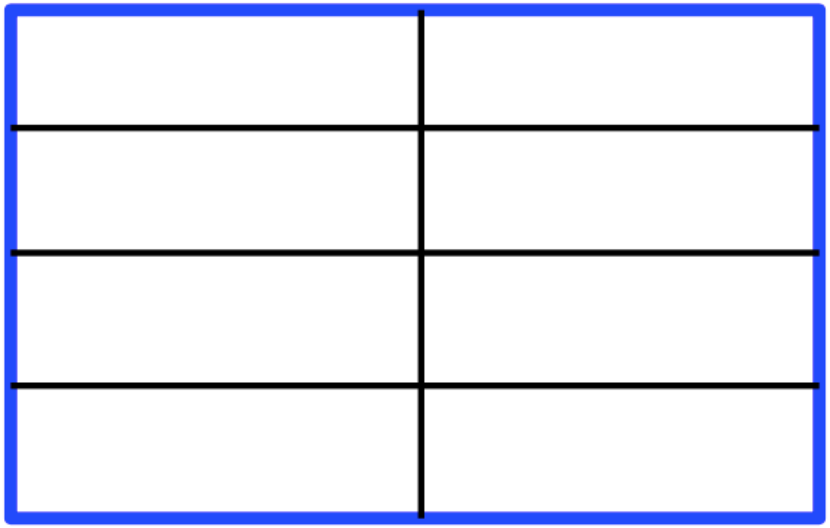
NSUB_X * NSUB_E has to be a multiple of NPP.

Default is (see param.h)

NSUB_X = 1, NSUB_E = NPP

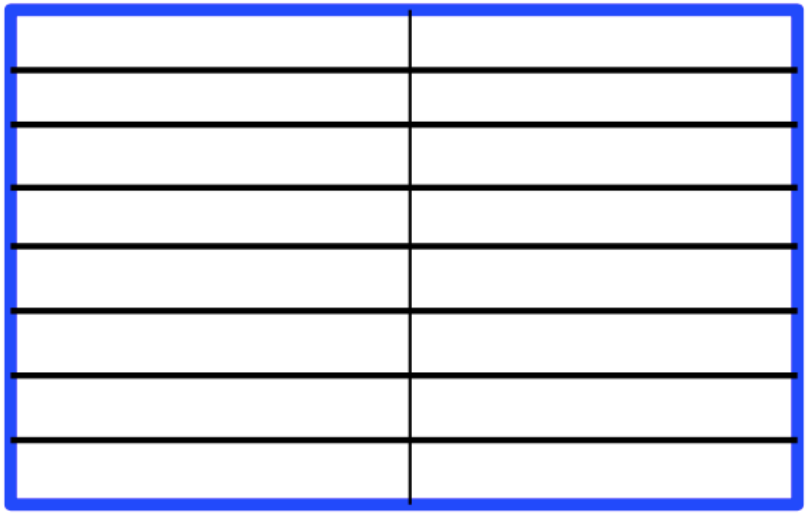
define OPENMP

Ex: 1 Node with 8 cores



NSUB_X = 2, NSUB_E = 4, NPP = 8

1 thread = 1 tile



NSUB_X = 2, NSUB_E = 8, NPP = 8

1 thread = 2 tiles

Multiple sub-domains can be assigned to each processor in order to optimize the use of processor cache memory.

define OPENMP

Each core can read/write global variables.

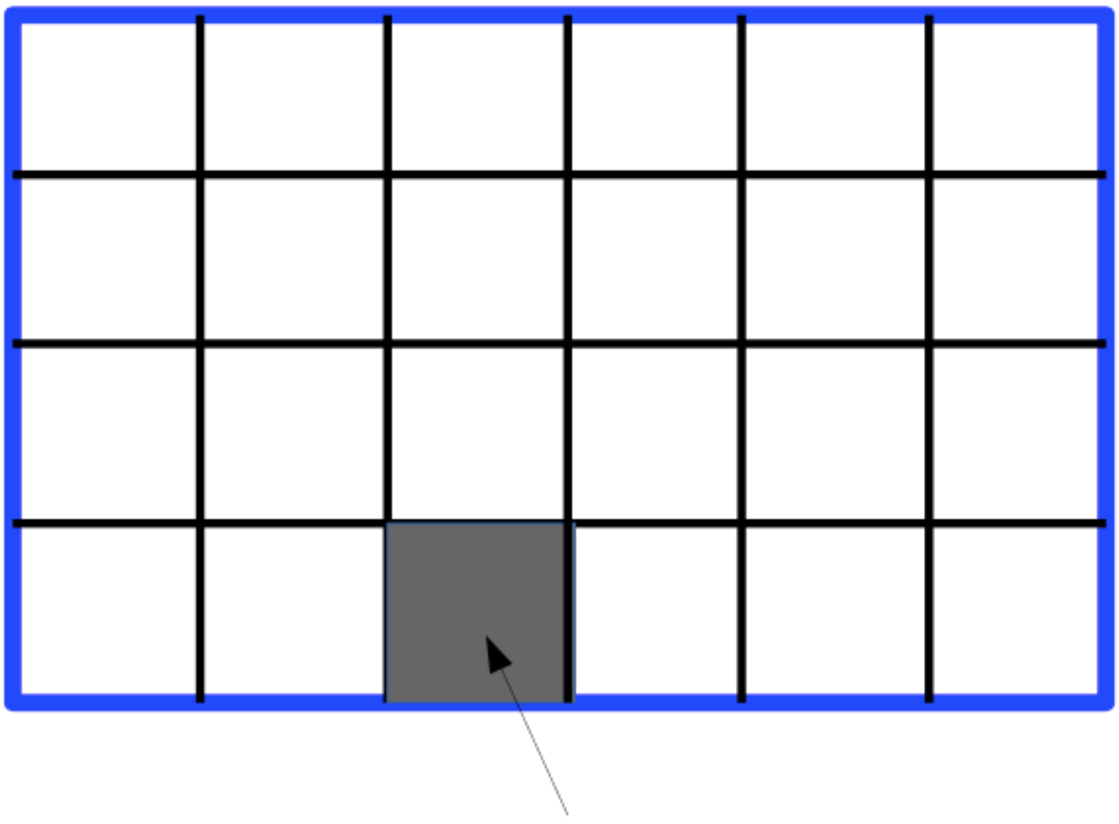
Has to ensure that different cores will not write the same indices of variables.

Also has to synchronize between different threads

Code structure example:

```
      Do tile=my_first,my_last

          Call compute_1(tile)
          Call compute_2(tile)
      Enddo
C$OMP BARRIER ! synchronisation

      Do tile=my_first,my_last

          Call compute_3(tile)
          Call compute_4(tile)
      Enddo
C$OMP BARRIER ! synchronisation
```

define MPI



Ex: *NP_XI = 6, NP_ETA = 4*

MPI = distributed – memory

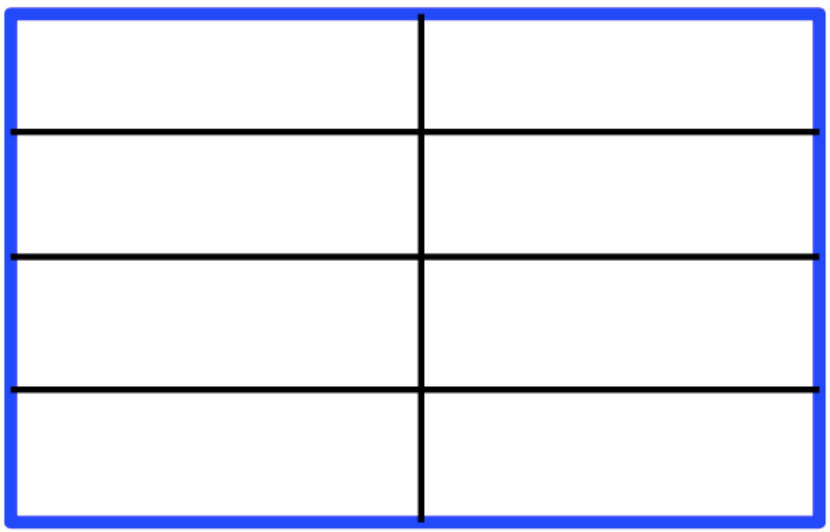
NP_XI*NP_ETA = number of threads (cores)

NPP = 1

NSUB_X * NSUB_E can be anything

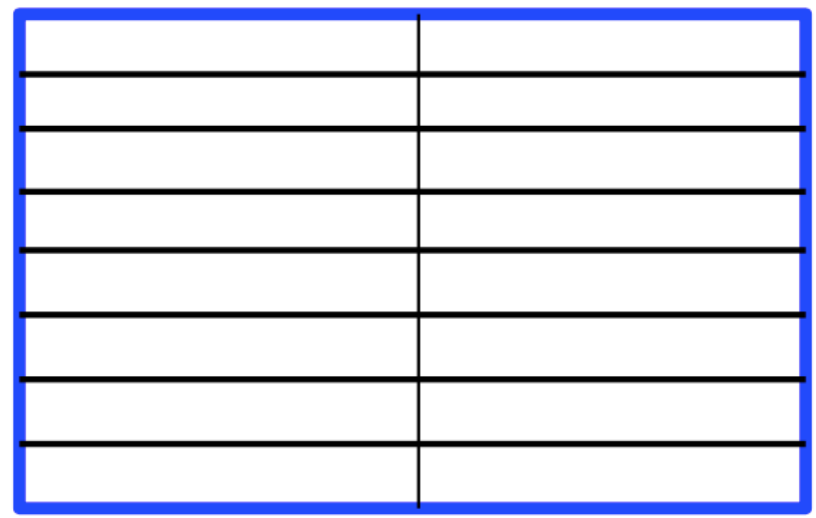
define MPI

Ex: 1 Node with 8 cores



NP_XI = 2, NP_ETA = 4, NPP = 1

1 thread = 1 tile



NP_XI = 2, NP_ETA = 4, NPP = 1

NSUB_X = 1, NSUB_E 2

1 thread = 2 tiles

define MPI

Each core has access to the variables only over the tile

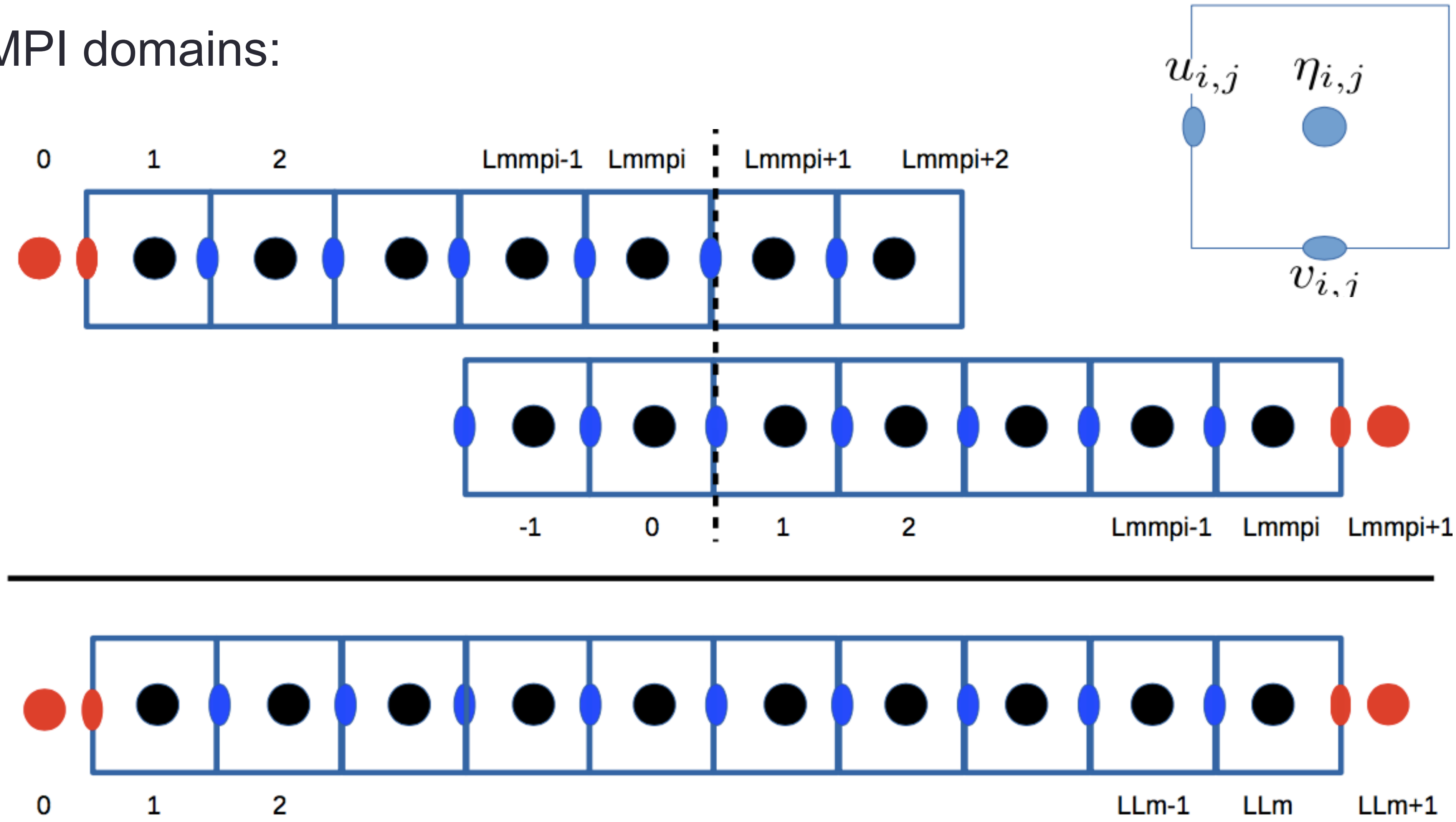
Cores have to communicate with each other to exchange information about boundaries

Code structure example:

```
# if defined EW_PERIODIC || defined NS_PERIODIC || defined MPI
      call exchange_u3d_tile (Istr,Iend,Jstr,Jend,
     &                                  u(START_2D_ARRAY,1,nnew))
      call exchange_v3d_tile (Istr,Iend,Jstr,Jend,
     &                                  v(START_2D_ARRAY,1,nnew))

      call exchange_u3d_tile (Istr,Iend,Jstr,Jend,
     &                                   Huon(START_2D_ARRAY,1))
      call exchange_v3d_tile (Istr,Iend,Jstr,Jend,
     &                                   Hvom(START_2D_ARRAY,1))

      call exchange_u2d_tile (Istr,Iend,Jstr,Jend,
     &                                   ubar(START_2D_ARRAY,knew))

      call exchange_v2d_tile (Istr,Iend,Jstr,Jend,
     &                                   vbar(START_2D_ARRAY,knew))
#   if defined TS_MIX_ISO || defined TS_MIX_GEO
      call exchange_u3d_tile (istr,iend,jstr,jend, dRdx  )
      call exchange_v3d_tile (istr,iend,jstr,jend, dRde  )
#  endif
```

define MPI

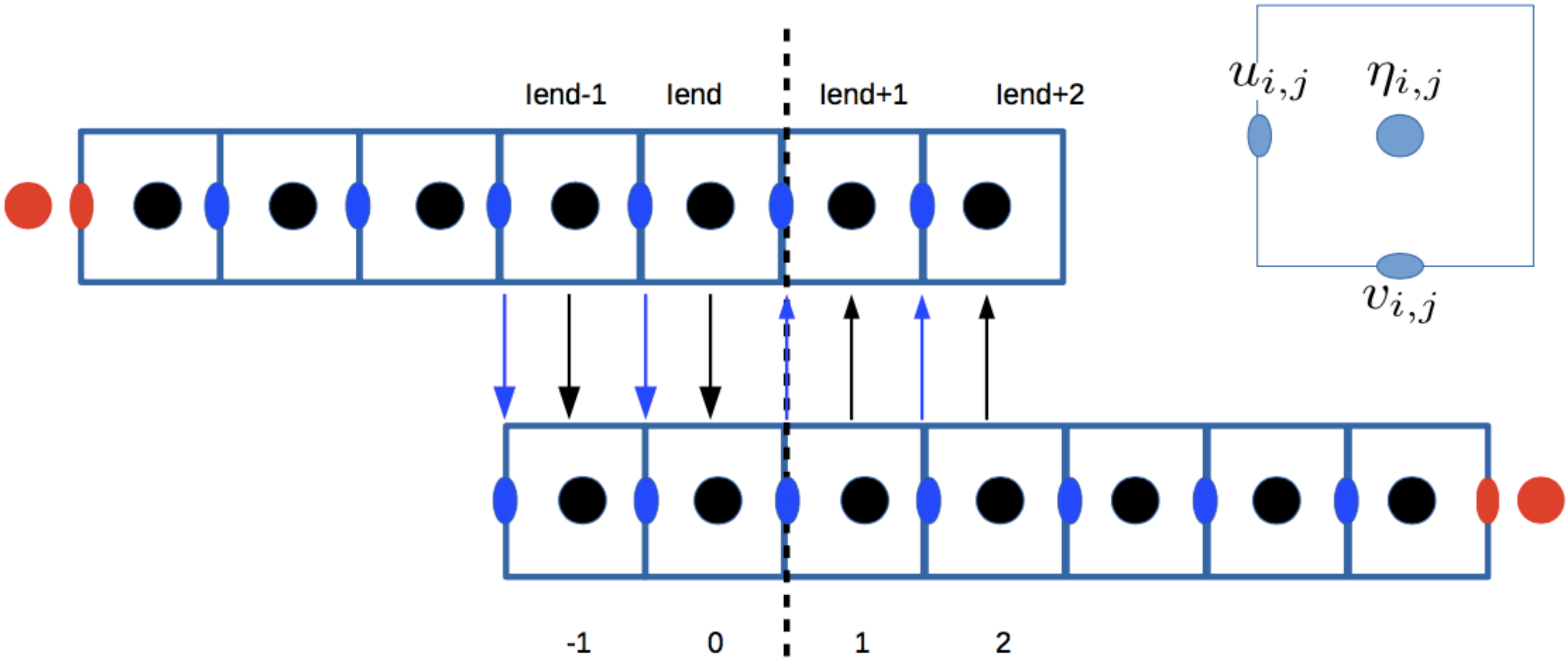
MPI domains:



Full domain:

define MPI

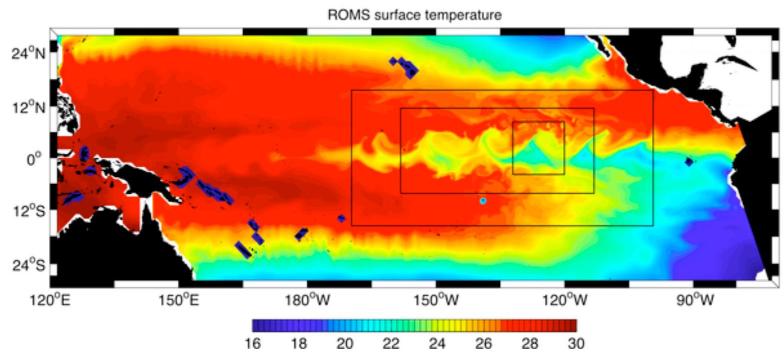
MPI exchanges:

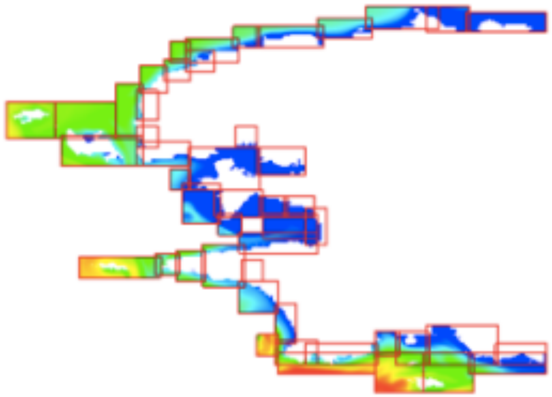


cppdefs.h

```
#if defined REGIONAL
/*
!====================================================================
!               REGIONAL (realistic) Configurations
!====================================================================
!
!----------------------
! BASIC OPTIONS
!----------------------
!
*/
                      /* Configuration Name */
# define BENGUELA_LR
                      /* Parallelization */
# undef  OPENMP
# undef  MPI
                      /* Nesting */
# undef  AGRIF
# undef  AGRIF_2WAY
                      /* OA and OW Coupling via OASIS (MPI) */
# undef  OA_COUPLING
# undef  OW_COUPLING
                      /* I/O server */
# undef  XIOS
                      /* Open Boundary Conditions */
# undef  TIDES
# define OBC_EAST
# define OBC_WEST
# define OBC_NORTH
# define OBC_SOUTH
                      /* Applications */
# undef  BIOLOGY
# undef  FLOATS
# undef  STATIONS
# undef  PASSIVE_TRACER
# undef  SEDIMENT
# undef  BBL
/*!
```

Online Nesting options:

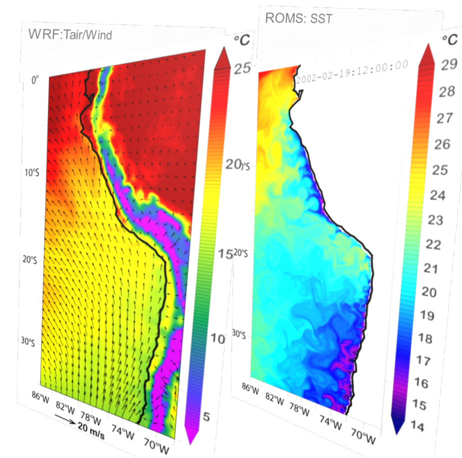




cppdefs.h

```
#if defined REGIONAL
/*
!====================================================================
!              REGIONAL (realistic) Configurations
!====================================================================
!
!----------------------
! BASIC OPTIONS
!----------------------
!
*/
                      /* Configuration Name */
# define BENGUELA_LR
                      /* Parallelization */
# undef  OPENMP
# undef  MPI
                      /* Nesting */
# undef  AGRIF
# undef  AGRIF_2WAY
                      /* OA and OW Coupling via OASIS (MPI) */
# undef  OA_COUPLING
# undef  OW_COUPLING
                      /* I/O server */
# undef  XIOS
                      /* Open Boundary Conditions */
# undef  TIDES
# define OBC_EAST
# define OBC_WEST
# define OBC_NORTH
# define OBC_SOUTH
                      /* Applications */
# undef  BIOLOGY
# undef  FLOATS
# undef  STATIONS
# undef  PASSIVE_TRACER
# undef  SEDIMENT
# undef  BBL
/*!
```

Ocean - Atmosphere coupling



Ocean – Wave coupling

cppdefs.h

```
#if defined REGIONAL
/*
!====================================================================
!               REGIONAL (realistic) Configurations
!====================================================================
!
!----------------------
! BASIC OPTIONS
!----------------------
!
*/
                      /* Configuration Name */
# define BENGUELA_LR
                      /* Parallelization */
# undef  OPENMP
# undef  MPI
                      /* Nesting */
# undef  AGRIF
# undef  AGRIF_2WAY
                      /* OA and OW Coupling via OASIS (MPI) */
# undef  OA_COUPLING
# undef  OW_COUPLING
                      /* I/O server */
# undef  XIOS
                      /* Open Boundary Conditions */
# undef  TIDES
# define OBC_EAST
# define OBC_WEST
# define OBC_NORTH
# define OBC_SOUTH
                      /* Applications */
# undef  BIOLOGY
# undef  FLOATS
# undef  STATIONS
# undef  PASSIVE_TRACER
# undef  SEDIMENT
# undef  BBL
/*!
```

XIOS = XML – IO – SERVER

Management of IO by dedicated cores

cppdefs.h

```
#if defined REGIONAL
/*
!====================================================================
!               REGIONAL (realistic) Configurations
!====================================================================
!
!----------------------
! BASIC OPTIONS
!----------------------
!
*/
                       /* Configuration Name */
# define BENGUELA_LR
                       /* Parallelization */
# undef  OPENMP
# undef  MPI
                       /* Nesting */
# undef  AGRIF
# undef  AGRIF_2WAY
                       /* OA and OW Coupling via OASIS (MPI) */
# undef  OA_COUPLING
# undef  OW_COUPLING
                       /* I/O server */
# undef  XIOS
                       /* Open Boundary Conditions */
# undef  TIDES
# define OBC_EAST
# define OBC_WEST
# define OBC_NORTH
# define OBC_SOUTH
                       /* Applications */
# undef  BIOLOGY
# undef  FLOATS
# undef  STATIONS
# undef  PASSIVE_TRACER
# undef  SEDIMENT
# undef  BBL
/*!
```

Add barotropic tides (from a global tidal model) at the boundaries

cppdefs.h

```
#if defined REGIONAL
/*
!====================================================================
!               REGIONAL (realistic) Configurations
!====================================================================
!
!----------------------
! BASIC OPTIONS
!----------------------
!
*/
                      /* Configuration Name */
# define BENGUELA_LR
                      /* Parallelization */
# undef  OPENMP
# undef  MPI
                      /* Nesting */
# undef  AGRIF
# undef  AGRIF_2WAY
                      /* OA and OW Coupling via OASIS (MPI) */
# undef  OA_COUPLING
# undef  OW_COUPLING
                      /* I/O server */
# undef  XIOS
                      /* Open Boundary Conditions */
# undef  TIDES
# define OBC_EAST
# define OBC_WEST
# define OBC_NORTH
# define OBC_SOUTH
                      /* Applications */
# undef  BIOLOGY
# undef  FLOATS
# undef  STATIONS
# undef  PASSIVE_TRACER
# undef  SEDIMENT
# undef  BBL
/*!
```

Use open boundary conditions

If undef then the domain is closed (see e.g. basin) or periodic.

cppdefs.h

```
#if defined REGIONAL
/*
!====================================================================
!               REGIONAL (realistic) Configurations
!====================================================================
!
!----------------------
! BASIC OPTIONS
!----------------------
!
*/
                      /* Configuration Name */
# define BENGUELA_LR
                      /* Parallelization */
# undef  OPENMP
# undef  MPI
                      /* Nesting */
# undef  AGRIF
# undef  AGRIF_2WAY
                      /* OA and OW Coupling via OASIS (MPI) */
# undef  OA_COUPLING
# undef  OW_COUPLING
                      /* I/O server */
# undef  XIOS
                      /* Open Boundary Conditions */
# undef  TIDES
# define OBC_EAST
# define OBC_WEST
# define OBC_NORTH
# define OBC_SOUTH
                      /* Applications */
# undef  BIOLOGY
# undef  FLOATS
# undef  STATIONS
# undef  PASSIVE_TRACER
# undef  SEDIMENT
# undef  BBL
/*!
```

Various applications

Activate biogeochemical modeling

Activate floats

Store high frequency model outputs at stations

Add a passive tracer

Activate sediment modeling

Activate bottom boundary layer parametrization

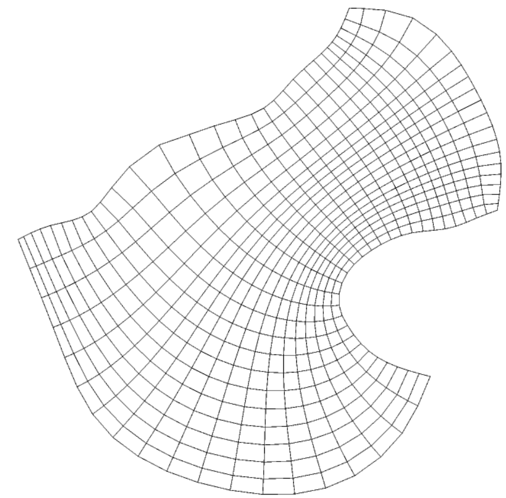
cppdefs.h

```
!----------------------------------------------------
! PRE-SELECTED OPTIONS
!
! ADVANCED OPTIONS ARE IN CPPDEFS_DEV.H
!----------------------------------------------------
*/
                      /* Parallelization */
# ifdef MPI
#  undef  PARALLEL_FILES
# endif
# undef  AUTOTILING
# undef  ETALON_CHECK
                      /* Grid configuration */
# define CURVGRID
# define SPHERICAL
# define MASKING
# undef  WET_DRY
# undef  NEW_S_COORD
                      /* Model dynamics */
# define SOLVE3D
# define UV_COR
# define UV_ADV
                      /* Equation of State */
# define SALINITY
# define NONLIN_EOS
# define SPLIT_EOS
```

Activate parallel I/O writing

cppdefs.h

```
!-----------------------------------------------
! PRE-SELECTED OPTIONS
!
! ADVANCED OPTIONS ARE IN CPPDEFS_DEV.H
!-----------------------------------------------
*/
                       /* Parallelization */
# ifdef MPI
#  undef  PARALLEL_FILES
# endif
# undef  AUTOTILING
# undef  ETALON_CHECK
                       /* Grid configuration */
# define CURVGRID
# define SPHERICAL
# define MASKING
# undef  WET_DRY
# undef  NEW_S_COORD
                       /* Model dynamics */
# define SOLVE3D
# define UV_COR
# define UV_ADV
                       /* Equation of State */
# define SALINITY
# define NONLIN_EOS
# define SPLIT_EOS
```



Activate curvilinear coordinate transformation

See rhs3d.F:

```
      do j=JstrV-1,Jend
        do i=IstrU-1,Iend
          cff=0.5*Hz(i,j,k)*(
#  ifdef UV_COR
     &              fomn(i,j)
#  endif
#  if (defined CURVGRID && defined UV_ADV)
     &          +0.5*( (v(i,j,k,nrhs)+v(i,j+1,k,nrhs))*dndx(i,j)
     &                -(u(i,j,k,nrhs)+u(i+1,j,k,nrhs))*dmde(i,j))
#  endif
     &                                                          )
          UFx(i,j)=cff*(v(i,j,k,nrhs)+v(i,j+1,k,nrhs))
          VFe(i,j)=cff*(u(i,j,k,nrhs)+u(i+1,j,k,nrhs))
        enddo
      enddo
```

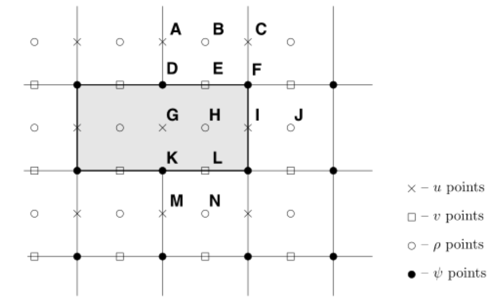
cppdefs.h

```
!---------------------------------------------------
! PRE-SELECTED OPTIONS
!
! ADVANCED OPTIONS ARE IN CPPDEFS_DEV.H
!---------------------------------------------------
*/
                      /* Parallelization */
# ifdef MPI
#  undef  PARALLEL_FILES
# endif
# undef  AUTOTILING
# undef  ETALON_CHECK
                      /* Grid configuration */
# define CURVGRID
# define SPHERICAL
# define MASKING
# undef  WET_DRY
# undef  NEW_S_COORD
                      /* Model dynamics */
# define SOLVE3D
# define UV_COR
# define UV_ADV
                      /* Equation of State */
# define SALINITY
# define NONLIN_EOS
# define SPLIT_EOS
```

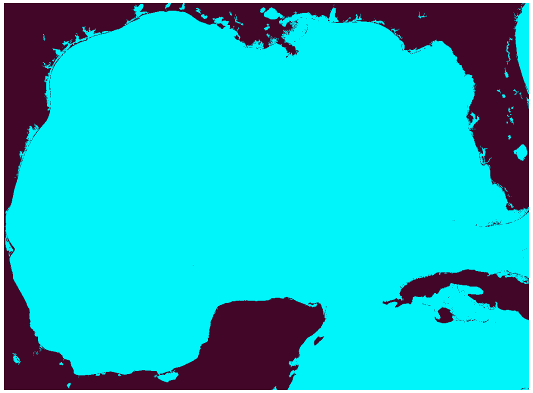
Activate longitude/latitude grid positioning

cppdefs.h

```
!-----------------------------------------------
! PRE-SELECTED OPTIONS
!
! ADVANCED OPTIONS ARE IN CPPDEFS_DEV.H
!-----------------------------------------------
*/
                          /* Parallelization */
# ifdef MPI
#  undef  PARALLEL_FILES
# endif
# undef  AUTOTILING
# undef  ETALON_CHECK
                          /* Grid configuration */
# define CURVGRID
# define SPHERICAL
# define MASKING
# undef  WET_DRY
# undef  NEW_S_COORD
                          /* Model dynamics */
# define SOLVE3D
# define UV_COR
# define UV_ADV
                          /* Equation of State */
# define SALINITY
# define NONLIN_EOS
# define SPLIT_EOS
```



Activate land masking



```
#  endif /* ZONAL_NUDGING */
#  ifdef MASKING
           u(i,j,k,nnew)=u(i,j,k,nnew)*umask(i,j)
#  endif
# ifdef WET_DRY
           u(i,j,k,nnew)=u(i,j,k,nnew)*umask_wet(i,j)
# endif
        enddo
```

cppdefs.h

```
!--------------------------------------------------
! PRE-SELECTED OPTIONS
!
! ADVANCED OPTIONS ARE IN CPPDEFS_DEV.H
!--------------------------------------------------
*/
                      /* Parallelization */
# ifdef MPI
#  undef  PARALLEL_FILES
# endif
# undef  AUTOTILING
# undef  ETALON_CHECK
                      /* Grid configuration */
# define CURVGRID
# define SPHERICAL
# define MASKING
# undef  WET_DRY
# undef  NEW_S_COORD
                      /* Model dynamics */
# define SOLVE3D
# define UV_COR
# define UV_ADV
                      /* Equation of State */
# define SALINITY
# define NONLIN_EOS
# define SPLIT_EOS
```

Activate wetting-Drying scheme

The Wetting-Drying scheme cancels the outgoing momentum flux (not the incoming) from a grid cell if its total depth is below a threshold value (5 cm).

```
#  endif /* ZONAL_NUDGING */
#  ifdef MASKING
            u(i,j,k,nnew)=u(i,j,k,nnew)*umask(i,j)
#  endif
# ifdef WET_DRY
            u(i,j,k,nnew)=u(i,j,k,nnew)*umask_wet(i,j)
# endif
          enddo
```

cppdefs.h

```
!----------------------------------------------
! PRE-SELECTED OPTIONS
!
! ADVANCED OPTIONS ARE IN CPPDEFS_DEV.H
!----------------------------------------------
*/
                      /* Parallelization */
# ifdef MPI
#  undef  PARALLEL_FILES
# endif
# undef  AUTOTILING
# undef  ETALON_CHECK
                      /* Grid configuration */
# define CURVGRID
# define SPHERICAL
# define MASKING
# undef  WET_DRY
# undef  NEW_S_COORD
                      /* Model dynamics */
# define SOLVE3D
# define UV_COR
# define UV_ADV
                      /* Equation of State */
# define SALINITY
# define NONLIN_EOS
# define SPLIT_EOS
```

Choose new vertical S-coordinates

cppdefs.h

```
!-----------------------------------------------
! PRE-SELECTED OPTIONS
!
! ADVANCED OPTIONS ARE IN CPPDEFS_DEV.H
!-----------------------------------------------
*/
                      /* Parallelization */
# ifdef MPI
#  undef  PARALLEL_FILES
# endif
# undef  AUTOTILING
# undef  ETALON_CHECK
                      /* Grid configuration */
# define CURVGRID
# define SPHERICAL
# define MASKING
# undef  WET_DRY
# undef  NEW_S_COORD
                      /* Model dynamics */
# define SOLVE3D
# define UV_COR
# define UV_ADV
                      /* Equation of State */
# define SALINITY
# define NONLIN_EOS
# define SPLIT_EOS
```

solve 3D primitive equations

See rhs3d.F:

```
#include "cppdefs.h"
#ifdef SOLVE3D

      subroutine rhs3d (tile)
!
      implicit none
      integer tile, trd, omp_get_thread_num
# include "param.h"
# include "private_scratch.h"
# include "compute_tile_bounds.h"
      trd=omp_get_thread_num()
      call rhs3d_tile (Istr,Iend,Jstr,Jend,
     &                          A3d(1,1,trd), A3d(1,2,trd),
     &              A2d(1,1,trd), A2d(1,2,trd), A2d(1,3,trd),
     &              A2d(1,1,trd), A2d(1,2,trd), A2d(1,3,trd),
     &              A2d(1,4,trd), A2d(1,5,trd), A2d(1,6,trd))
      return
      end
```

cppdefs.h

```
!-----------------------------------------------
! PRE-SELECTED OPTIONS
!
! ADVANCED OPTIONS ARE IN CPPDEFS_DEV.H
!-----------------------------------------------
*/
                        /* Parallelization */
# ifdef MPI
#  undef  PARALLEL_FILES
# endif
# undef  AUTOTILING
# undef  ETALON_CHECK
                        /* Grid configuration */
# define CURVGRID
# define SPHERICAL
# define MASKING
# undef  WET_DRY
# undef  NEW_S_COORD
                        /* Model dynamics */
# define SOLVE3D
# define UV_COR
# define UV_ADV
                        /* Equation of State */
# define SALINITY
# define NONLIN_EOS
# define SPLIT_EOS
```

$$\frac{\partial u}{\partial t} + \vec{u}\cdot\vec{\nabla}_H u + w\frac{\partial u}{\partial z} \boxed{- fv} = -\frac{\partial_x P}{\rho_0} + \mathcal{F}_u + \mathcal{D}_u$$

$$\frac{\partial v}{\partial t} + \vec{u}\cdot\vec{\nabla}_H v + w\frac{\partial v}{\partial z} \boxed{+ fu} = -\frac{\partial_y P}{\rho_0} + \mathcal{F}_v + \mathcal{D}_v$$

Activate Coriolis terms

See rhs3d.F:

```
        do j=JstrV-1,Jend
          do i=IstrU-1,Iend
            cff=0.5*Hz(i,j,k)*(
#  ifdef UV_COR
     &              fomn(i,j)
#  endif
#  if (defined CURVGRID && defined UV_ADV)
     &          +0.5*( (v(i,j,k,nrhs)+v(i,j+1,k,nrhs))*dndx(i,j)
     &                -(u(i,j,k,nrhs)+u(i+1,j,k,nrhs))*dmde(i,j))
#  endif
     &                                                         )
            UFx(i,j)=cff*(v(i,j,k,nrhs)+v(i,j+1,k,nrhs))
            VFe(i,j)=cff*(u(i,j,k,nrhs)+u(i+1,j,k,nrhs))
          enddo
        enddo
```

cppdefs.h

```
!----------------------------------------------------
! PRE-SELECTED OPTIONS
!
! ADVANCED OPTIONS ARE IN CPPDEFS_DEV.H
!----------------------------------------------------
*/
                      /* Parallelization */
# ifdef MPI
#  undef  PARALLEL_FILES
# endif
# undef  AUTOTILING
# undef  ETALON_CHECK
                      /* Grid configuration */
# define CURVGRID
# define SPHERICAL
# define MASKING
# undef  WET_DRY
# undef  NEW_S_COORD
                      /* Model dynamics */
# define SOLVE3D
# define UV_COR
# define UV_ADV
                      /* Equation of State */
# define SALINITY
# define NONLIN_EOS
# define SPLIT_EOS
```

$$
\begin{aligned}
\frac{\partial u}{\partial t} + \vec{u}\cdot\vec{\nabla}_H u + w\,\frac{\partial u}{\partial z} - fv &= -\frac{\partial_x P}{\rho_0} + \mathcal{F}_u + \mathcal{D}_u \\
\frac{\partial v}{\partial t} + \vec{u}\cdot\vec{\nabla}_H v + w\,\frac{\partial v}{\partial z} + fu &= -\frac{\partial_y P}{\rho_0} + \mathcal{F}_v + \mathcal{D}_v
\end{aligned}
$$

Activate advection terms

See rhs3d.F:

```
# ifdef UV_ADV
!
!======================================================================
!
!  Add in horizontal advection of momentum.
!  Compute diagonal [UFx,VFe] and off-diagonal [UFe,VFx] components
!  of tensor of momentum flux due to horizontal advection; after that
!  add in horizontal advection terms.
!
!======================================================================
!
#  if !(defined UV_HADV_UP3 || defined UV_HADV_C4 || defined UV_HADV_C2)
#   define UV_HADV_UP3
#  endif

#  if defined UV_HADV_UP3 || defined UV_HADV_C4
!
!-----------------------------------------------------------
! Fourth or Third order advection scheme (default)
!-----------------------------------------------------------
!
#   define uxx wrk1
#   define Huxx wrk2
!
#   ifdef EW_PERIODIC
#    define IU_EXT_RANGE IstrU-1,Iend+1
#   else
#    ifdef MPI
        if (WEST_INTER) then
          imin=IstrU-1
        else
          imin=max(IstrU-1,2)
        endif
        if (EAST_INTER) then
          imax=Iend+1
        else
          imax=min(Iend+1,Lmmpi)
        endif
#     define IU_EXT_RANGE imin,imax
#    else
#     define IU_EXT_RANGE max(IstrU-1,2),min(Iend+1,Lm)
#    endif
#   endif

        do j=Jstr,Jend
          do i=IU_EXT_RANGE
            uxx(i,j)=(u(i-1,j,k,nrhs)-2.*u(i,j,k,nrhs)
     &                  +u(i+1,j,k,nrhs))  SWITCH umask(i,j)
            Huxx(i,j)=(Huon(i-1,j,k)-2.*Huon(i,j,k)
     &                   +Huon(i+1,j,k))   SWITCH umask(i,j)
          enddo
        enddo
```

cppdefs.h

```
!-----------------------------------------
! PRE-SELECTED OPTIONS
!
! ADVANCED OPTIONS ARE IN CPPDEFS_DEV.H
!-----------------------------------------
*/
                      /* Parallelization */
# ifdef MPI
#  undef  PARALLEL_FILES
# endif
# undef  AUTOTILING
# undef  ETALON_CHECK
                      /* Grid configuration */
# define CURVGRID
# define SPHERICAL
# define MASKING
# undef  WET_DRY
# undef  NEW_S_COORD
                      /* Model dynamics */
# define SOLVE3D
# define UV_COR
# define UV_ADV
                      /* Equation of State */
# define SALINITY
# define NONLIN_EOS
# define SPLIT_EOS
```

Define salinity as an active tracer

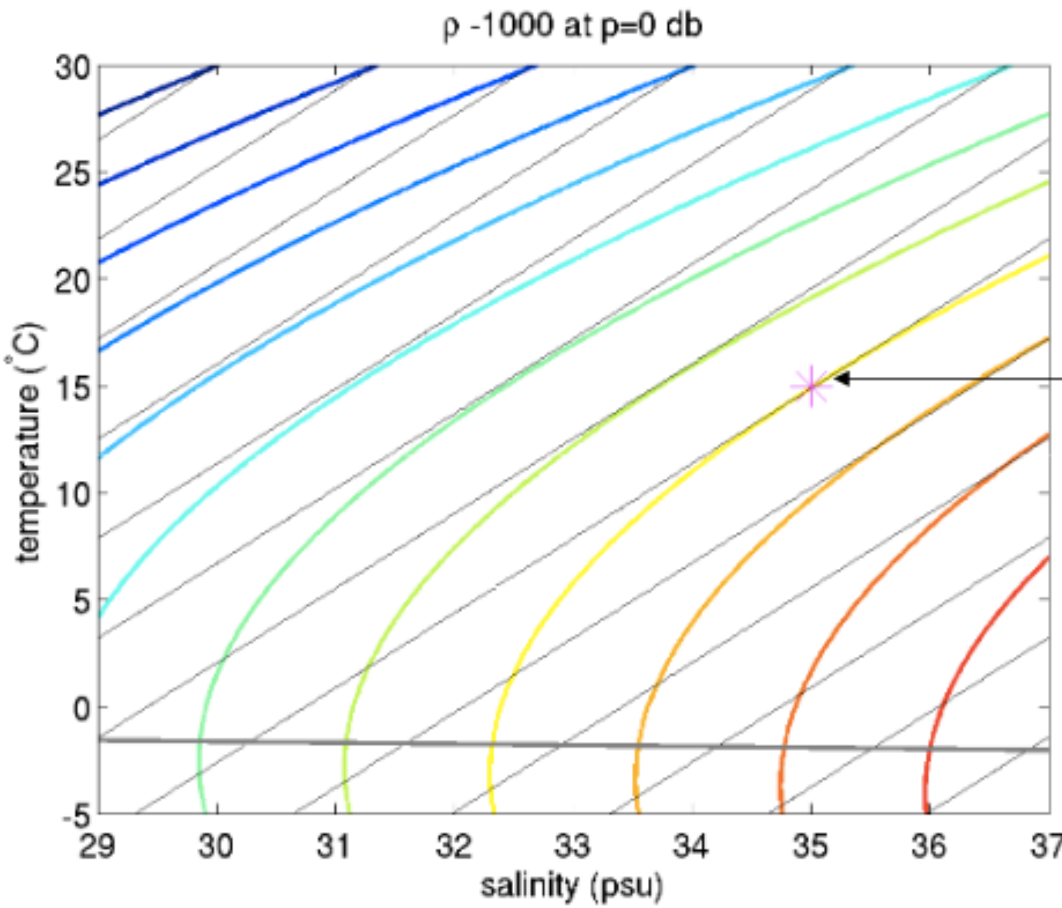
cppdefs.h

```
!-----------------------------------------------
! PRE-SELECTED OPTIONS
!
! ADVANCED OPTIONS ARE IN CPPDEFS_DEV.H
!-----------------------------------------------
*/
                      /* Parallelization */
# ifdef MPI
#  undef  PARALLEL_FILES
# endif
# undef  AUTOTILING
# undef  ETALON_CHECK
                      /* Grid configuration */
# define CURVGRID
# define SPHERICAL
# define MASKING
# undef  WET_DRY
# undef  NEW_S_COORD
                      /* Model dynamics */
# define SOLVE3D
# define UV_COR
# define UV_ADV
                      /* Equation of State */
# define SALINITY
# define NONLIN_EOS
# define SPLIT_EOS
```

Choose non linear equation of state. *See rho_eos.F:*

```
      subroutine rho_eos(Lm,Mm,N, T,S, z_r,z_w,rho0, rho)

!
! Compute density anomaly from T,S via Equation Of State (EOS) for
!-------- ------- ------- ----  for seawater. Following Jackett and
! McDougall, 1995, physical EOS is assumed to have form
!
!                           rho0 + rho1(T,S)
!           rho(T,S,z) = ------------------------                  (1)
!                          1 - 0.1*|z|/K(T,S,|z|)
!
! where rho1(T,S) is sea-water density perturbation[kg/m^3] at
! standard pressure of 1 Atm (sea surface); |z| is absolute depth,
! i.e. distance from free-surface to the point at which density is
! computed, and
!
!     K(T,S,|z|) = K00 + K01(T,S) + K1(T,S)*|z| + K2(T,S)*|z|^2.  (2)
!
! To reduce errors of pressure-gradient scheme associated with
! nonlinearity of compressibility effects, as well as to reduce
! roundoff errors, the dominant part of density profile,
!
!                           rho0
!                     ---------------                              (3)
!                      1 - 0.1|z|/K00
!
! is removed from from (1). [Since (3) is purely a function of z,
! it does not contribute to pressure gradient.]  This results in
!
!                  rho1 - rho0*[K01+K1*|z|+K2*|z|^2]/[K00-0.1|z|]
!    rho1 + 0.1|z| -----------------------------------------------
!                        K00 + K01 + (K1-0.1)*|z| + K2*|z|^2
!                                                                  (4)
! which is suitable for pressure-gradient calculation.
!
! Optionally, if CPP-switch SPLIT_EOS is defined, term proportional
! to |z| is linearized using smallness 0.1|z|/[K00 + K01] << 1 and
! the resultant EOS has form
!
!              rho(T,S,z) = rho1(T,S) + qp1(T,S)*|z|               (5)
!
! where
!                         rho1(T,S) - rho0*K01(T,S)/K00
!         qp1(T,S)= 0.1 ----------------------------------        (6)
!                              K00 + K01(T,S)
!
! is stored in a special array.
!
```

cppdefs.h



(S_o, T_o, σ_o)

=(35,15,26)

$\alpha_T = 2.1 \times 10^{-4}\ \mathrm{K}^{-1}$

$\beta_S = 7.5 \times 10^{-4}\ \mathrm{psu}^{-1}$

- The equation of state can be approximated using a linear function of temperature and salinity

$$\sigma = \sigma_o + \rho_{ref}\left[\beta_S(S - S_o) - \alpha_T(T - T_o)\right]$$

cppdefs.h

```
                      /* Lateral Momentum Advection (default UP3) */
# define UV_HADV_UP3
# undef  UV_HADV_C4
# undef  UV_HADV_C2
                      /* Lateral Tracer Advection (default UP3) */
# undef  TS_HADV_UP3
# define TS_HADV_RSUP3
# undef  TS_HADV_UP5
# undef  TS_HADV_C4
# undef  TS_HADV_WENO5
                      /* Lateral Explicit Momentum Mixing */
# undef  UV_VIS2
# ifdef UV_VIS2
#  define UV_VIS_SMAGO
# endif
                      /* Lateral Explicit Tracer Mixing */
# undef  TS_DIF2
# undef  TS_DIF4
# undef  TS_MIX_S
                      /* Sponge layers for UV and TS */
# define SPONGE
                      /* Semi-implicit Vertical Tracer/Mom Advection */
# undef  VADV_ADAPT_IMP
                      /* Vertical Mixing */
# undef  BODYFORCE
# undef  BVF_MIXING
# define LMD_MIXING
# undef  GLS_MIXING
# ifdef LMD_MIXING
#  define LMD_SKPP
#  define LMD_BKPP
#  define LMD_RIMIX
#  define LMD_CONVEC
#  undef  LMD_DDMIX
#  define LMD_NONLOCAL
# endif
# ifdef GLS_MIXING
#  define GLS_KKL
#  undef  GLS_KOMEGA
#  undef  GLS_KEPSILON
#  undef  GLS_GEN
#  undef  KANTHA_CLAYSON
#  undef  CRAIG_BANNER
#  undef  CANUTO_A
#  undef  ZOS_HSIG
# endif
```

cppdefs.h

```
                      /* Lateral Momentum Advection (default UP3) */
# define UV_HADV_UP3
# undef  UV_HADV_C4
# undef  UV_HADV_C2
                      /* Lateral Tracer Advection (default UP3) */
# undef  TS_HADV_UP3
# define TS_HADV_RSUP3
# undef  TS_HADV_UP5
# undef  TS_HADV_C4
# undef  TS_HADV_WENO5
                      /* Lateral Explicit Momentum Mixing */
# undef  UV_VIS2
# ifdef UV_VIS2
#  define UV_VIS_SMAGO
# endif
                      /* Lateral Explicit Tracer Mixing */
# undef  TS_DIF2
# undef  TS_DIF4
# undef  TS_MIX_S
                      /* Sponge layers for UV and TS */
# define SPONGE
                      /* Semi-implicit Vertical Tracer/Mom Advection */
# undef  VADV_ADAPT_IMP
                      /* Vertical Mixing */
# undef  BODYFORCE
# undef  BVF_MIXING
# define LMD_MIXING
# undef  GLS_MIXING
# ifdef LMD_MIXING
#  define LMD_SKPP
#  define LMD_BKPP
#  define LMD_RIMIX
#  define LMD_CONVEC
#  undef  LMD_DDMIX
#  define LMD_NONLOCAL
# endif
# ifdef GLS_MIXING
#  define GLS_KKL
#  undef  GLS_KOMEGA
#  undef  GLS_KEPSILON
#  undef  GLS_GEN
#  undef  KANTHA_CLAYSON
#  undef  CRAIG_BANNER
#  undef  CANUTO_A
#  undef  ZOS_HSIG
# endif
```

Choose horizontal advective scheme for momentum

See rhs3d.F:

cppdefs.h

```
                        /* Lateral Momentum Advection (default UP3) */
# define UV_HADV_UP3
# undef  UV_HADV_C4
# undef  UV_HADV_C2
                        /* Lateral Tracer Advection (default UP3) */
# undef  TS_HADV_UP3
# define TS_HADV_RSUP3
# undef  TS_HADV_UP5
# undef  TS_HADV_C4
# undef  TS_HADV_WENO5
                        /* Lateral Explicit Momentum Mixing */
# undef  UV_VIS2
# ifdef UV_VIS2
#  define UV_VIS_SMAGO
# endif
                        /* Lateral Explicit Tracer Mixing */
# undef  TS_DIF2
# undef  TS_DIF4
# undef  TS_MIX_S
                        /* Sponge layers for UV and TS */
# define SPONGE
                        /* Semi-implicit Vertical Tracer/Mom Advection */
# undef  VADV_ADAPT_IMP
                        /* Vertical Mixing */
# undef  BODYFORCE
# undef  BVF_MIXING
# define LMD_MIXING
# undef  GLS_MIXING
# ifdef LMD_MIXING
#  define LMD_SKPP
#  define LMD_BKPP
#  define LMD_RIMIX
#  define LMD_CONVEC
#  undef  LMD_DDMIX
#  define LMD_NONLOCAL
# endif
# ifdef GLS_MIXING
#  define GLS_KKL
#  undef  GLS_KOMEGA
#  undef  GLS_KEPSILON
#  undef  GLS_GEN
#  undef  KANTHA_CLAYSON
#  undef  CRAIG_BANNER
#  undef  CANUTO_A
#  undef  ZOS_HSIG
# endif
```

Choose horizontal advective scheme for tracers

See step3d_t.F and compute_horiz_tracer_fluxes.h

Horizontal Advective Schemes

- C2 = 2nd-order centered advection scheme
- UP3 = 3rd-order upstream-biased advection scheme
- C4 = 4th-order centered advection scheme
- UP5 = 5th-order upstream-biased advection scheme
- C6 = 6th-order centered advection scheme
- RSUP3 = Split and rotated 3rd-order upstream-biased advection scheme
- RSUP5 = Split and rotated 3rd-order upstream-biased advection scheme with reduced dispersion/diffusion
- HADV_WENO5 = 5th-order WENOZ quasi-monotone advection scheme for all tracers

Vertical Advective Schemes

- cppdefs_dev.h

```
/*
======================================================================
    Select MOMENTUM VERTICAL advection scheme:
======================================================================
*/
#ifdef UV_VADV_SPLINES  /* Check if options are defined in cppdefs.h */
#elif defined UV_VADV_C2
#else
# define UV_VADV_SPLINES   /* Splines vertical advection             */
# undef  UV_VADV_C2        /* 2nd-order centered vertical advection  */
#endif

#ifdef VADV_ADAPT_IMP      /* Semi-implicit vertical advection       */
# undef  VADV_ADAPT_PRED   /* apply to both pred/corr steps (choice) */
# define UV_VADV_SPLINES   /* Impose splines advection (no choice)   */
# undef  UV_VADV_C2
#endif
```

```
/*
======================================================================
    Select model dynamics for TRACER vertical advection
    (The default is 4th-order centered)
======================================================================
*/
#ifdef TS_VADV_SPLINES  /* Check if options are defined in cppdefs.h */
#elif defined TS_VADV_AKIMA
#elif defined TS_VADV_WENO5
#elif defined TS_VADV_C2
#else
# undef  TS_VADV_SPLINES   /* Splines vertical advection             */
# define TS_VADV_AKIMA     /* 4th-order Akima vertical advection     */
# undef  TS_VADV_WENO5     /* 5th-order WENOZ vertical advection     */
# undef  TS_VADV_C2        /* 2nd-order centered vertical advection */
#endif

#undef  TS_VADV_FCT        /* Flux correction of vertical advection */

#ifdef VADV_ADAPT_IMP
# define  TS_VADV_SPLINES
# undef   TS_VADV_AKIMA
# undef   TS_VADV_WENO5
# undef   TS_VADV_C2
#endif
```

cppdefs.h

```
                      /* Lateral Momentum Advection (default UP3) */
# define UV_HADV_UP3
# undef  UV_HADV_C4
# undef  UV_HADV_C2
                      /* Lateral Tracer Advection (default UP3) */
# undef  TS_HADV_UP3
# define TS_HADV_RSUP3
# undef  TS_HADV_UP5
# undef  TS_HADV_C4
# undef  TS_HADV_WENO5
                      /* Lateral Explicit Momentum Mixing */
# undef  UV_VIS2
# ifdef UV_VIS2
#  define UV_VIS_SMAGO
# endif
                      /* Lateral Explicit Tracer Mixing */
# undef  TS_DIF2
# undef  TS_DIF4
# undef  TS_MIX_S
                      /* Sponge layers for UV and TS */
# define SPONGE
                      /* Semi-implicit Vertical Tracer/Mom Advection */
# undef  VADV_ADAPT_IMP
                      /* Vertical Mixing */
# undef  BODYFORCE
# undef  BVF_MIXING
# define LMD_MIXING
# undef  GLS_MIXING
# ifdef LMD_MIXING
#  define LMD_SKPP
#  define LMD_BKPP
#  define LMD_RIMIX
#  define LMD_CONVEC
#  undef  LMD_DDMIX
#  define LMD_NONLOCAL
# endif
# ifdef GLS_MIXING
#  define GLS_KKL
#  undef  GLS_KOMEGA
#  undef  GLS_KEPSILON
#  undef  GLS_GEN
#  undef  KANTHA_CLAYSON
#  undef  CRAIG_BANNER
#  undef  CANUTO_A
#  undef  ZOS_HSIG
# endif
```

Activate explicit horizontal viscosity

See uv3dmix.F

UV_VIS2 = Laplacian

UV_VIS4 = bilaplacian

UV_VIS_SMAGO = Smagorinsky parametrization of turbulent viscosity

cppdefs.h

```
                     /* Lateral Momentum Advection (default UP3) */
# define UV_HADV_UP3
# undef  UV_HADV_C4
# undef  UV_HADV_C2
                     /* Lateral Tracer Advection (default UP3) */
# undef  TS_HADV_UP3
# define TS_HADV_RSUP3
# undef  TS_HADV_UP5
# undef  TS_HADV_C4
# undef  TS_HADV_WENO5
                     /* Lateral Explicit Momentum Mixing */
# undef  UV_VIS2
# ifdef UV_VIS2
#  define UV_VIS_SMAGO
# endif
                     /* Lateral Explicit Tracer Mixing */
# undef  TS_DIF2
# undef  TS_DIF4
# undef  TS_MIX_S
                     /* Sponge layers for UV and TS */
# define SPONGE
                     /* Semi-implicit Vertical Tracer/Mom Advection */
# undef  VADV_ADAPT_IMP
                     /* Vertical Mixing */
# undef  BODYFORCE
# undef  BVF_MIXING
# define LMD_MIXING
# undef  GLS_MIXING
# ifdef LMD_MIXING
#  define LMD_SKPP
#  define LMD_BKPP
#  define LMD_RIMIX
#  define LMD_CONVEC
#  undef  LMD_DDMIX
#  define LMD_NONLOCAL
# endif
# ifdef GLS_MIXING
#  define GLS_KKL
#  undef  GLS_KOMEGA
#  undef  GLS_KEPSILON
#  undef  GLS_GEN
#  undef  KANTHA_CLAYSON
#  undef  CRAIG_BANNER
#  undef  CANUTO_A
#  undef  ZOS_HSIG
# endif
```

Activate explicit horizontal mixing of tracers

See t3dmix.F

TS_DIFF2 = Laplacian

TS_DIFF4 = bilaplacian

TS_MIX_S = mixing along iso-sigma surfaces

TS_MIX_GEO = mixing along geopotential surfaces

TS_MIX_ISO = mixing along isopycnal surfaces

Horizontal viscosity/mixing

[UV_VIS2] Tenseur visqueux (uv3dmix.F)

$$\boldsymbol{\sigma}(\mathbf{u}_h) = \begin{pmatrix} \partial_x u - \partial_y v & \partial_y u + \partial_x v \\ \partial_x v + \partial_y u & -(\partial_x u - \partial_y v) \end{pmatrix}$$

l'opérateur de viscosité est donc donné par

$$-\boldsymbol{\nabla}_h \cdot \langle \mathbf{u}'_h \mathbf{u}'_h \rangle = \frac{1}{\mathrm{Hz}} \boldsymbol{\nabla}_h \cdot \left(A_M \;\mathrm{Hz}\; \boldsymbol{\sigma} \right), \qquad A_M \leftrightarrow \mathrm{visc2}$$

Cette formulation assure

1. La conservation de la quantité de mouvement
2. La conservation du moment angulaire
3. Le terme visqueux est strictement dissipatif

[UV_VIS4] Même logique appliquée 2 fois ($B_M \leftrightarrow \mathrm{visc4}$)

$$-\boldsymbol{\nabla}_h \cdot \langle \mathbf{u}'_h \mathbf{u}'_h \rangle = -\frac{1}{\mathrm{Hz}} \boldsymbol{\nabla}_h \cdot \left(B_M \;\mathrm{Hz}\; \boldsymbol{\sigma}' \right), \qquad \boldsymbol{\sigma}' = \boldsymbol{\sigma}(\boldsymbol{\nabla}_h \cdot \boldsymbol{\sigma}(\mathbf{u}_h))$$

Horizontal viscosity/mixing

[UV_VIS_SMAGO, UV_VIS2]

Coefficient de viscosité turbulente

$$A_M = C_M\,(\Delta x \Delta y)\,\sqrt{(\partial_x u)^2 + (\partial_y v)^2 + 2(\partial_y u + \partial_x v)^2}$$

Par défaut $C_M = 1/10$ (paramétre *horcon* dans la routine hvisc_coef)

[TS_DIF_SMAGO, TS_DIF2]

Coefficient de diffusion turbulente

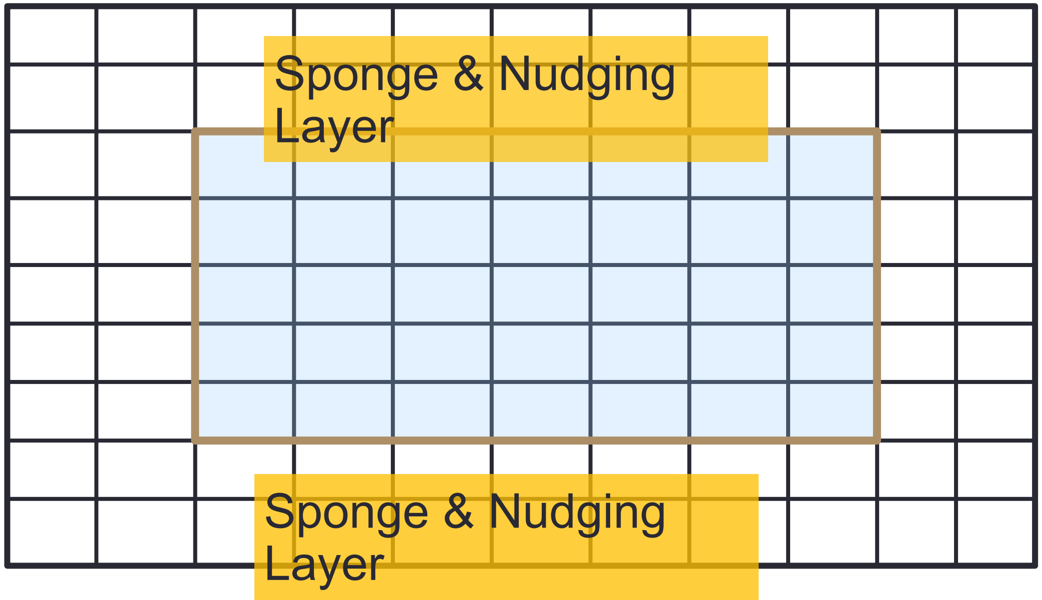
$$A_S = C_S\,(\Delta x \Delta y)\,\sqrt{(\partial_x u)^2 + (\partial_y v)^2 + 2(\partial_y u + \partial_x v)^2}$$

Par défaut $C_S = 1/12$ (paramétre *horcon* dans la routine hdiff_coef)

cppdefs.h

```
                                    /* Lateral Momentum Advection (default UP3) */
# define UV_HADV_UP3
# undef  UV_HADV_C4
# undef  UV_HADV_C2
                                    /* Lateral Tracer Advection (default UP3) */
# undef  TS_HADV_UP3
# define TS_HADV_RSUP3
# undef  TS_HADV_UP5
# undef  TS_HADV_C4
# undef  TS_HADV_WENO5
                                    /* Lateral Explicit Momentum Mixing */
# undef  UV_VIS2
# ifdef UV_VIS2
#  define UV_VIS_SMAGO
# endif
                                    /* Lateral Explicit Tracer Mixing */
# undef  TS_DIF2
# undef  TS_DIF4
# undef  TS_MIX_S
                                    /* Sponge layers for UV and TS */
# define SPONGE
                                    /* Semi-implicit Vertical Tracer/Mom Advection */
# undef  VADV_ADAPT_IMP
                                    /* Vertical Mixing */
# undef  BODYFORCE
# undef  BVF_MIXING
# define LMD_MIXING
# undef  GLS_MIXING
# ifdef LMD_MIXING
#  define LMD_SKPP
#  define LMD_BKPP
#  define LMD_RIMIX
#  define LMD_CONVEC
#  undef  LMD_DDMIX
#  define LMD_NONLOCAL
# endif
# ifdef GLS_MIXING
#  define GLS_KKL
#  undef  GLS_KOMEGA
#  undef  GLS_KEPSILON
#  undef  GLS_GEN
#  undef  KANTHA_CLAYSON
#  undef  CRAIG_BANNER
#  undef  CANUTO_A
#  undef  ZOS_HSIG
# endif
```

Activate areas of enhanced viscosity and diffusivity near lateral open boundaries.



cppdefs.h

```
                      /* Lateral Momentum Advection (default UP3) */
# define UV_HADV_UP3
# undef  UV_HADV_C4
# undef  UV_HADV_C2
                      /* Lateral Tracer Advection (default UP3) */
# undef  TS_HADV_UP3
# define TS_HADV_RSUP3
# undef  TS_HADV_UP5
# undef  TS_HADV_C4
# undef  TS_HADV_WENO5
                      /* Lateral Explicit Momentum Mixing */
# undef  UV_VIS2
# ifdef UV_VIS2
#  define UV_VIS_SMAGO
# endif
                      /* Lateral Explicit Tracer Mixing */
# undef  TS_DIF2
# undef  TS_DIF4
# undef  TS_MIX_S
                      /* Sponge layers for UV and TS */
# define SPONGE
                      /* Semi-implicit Vertical Tracer/Mom Advection */
# undef  VADV_ADAPT_IMP
                      /* Vertical Mixing */
# undef  BODYFORCE
# undef  BVF_MIXING
# define LMD_MIXING
# undef  GLS_MIXING
# ifdef LMD_MIXING
#  define LMD_SKPP
#  define LMD_BKPP
#  define LMD_RIMIX
#  define LMD_CONVEC
#  undef  LMD_DDMIX
#  define LMD_NONLOCAL
# endif
# ifdef GLS_MIXING
#  define GLS_KKL
#  undef  GLS_KOMEGA
#  undef  GLS_KEPSILON
#  undef  GLS_GEN
#  undef  KANTHA_CLAYSON
#  undef  CRAIG_BANNER
#  undef  CANUTO_A
#  undef  ZOS_HSIG
# endif
```

Apply surface and bottom stresses as body-forces

cppdefs.h

```
                      /* Lateral Momentum Advection (default UP3) */
# define UV_HADV_UP3
# undef  UV_HADV_C4
# undef  UV_HADV_C2
                      /* Lateral Tracer Advection (default UP3) */
# undef  TS_HADV_UP3
# define TS_HADV_RSUP3
# undef  TS_HADV_UP5
# undef  TS_HADV_C4
# undef  TS_HADV_WENO5
                      /* Lateral Explicit Momentum Mixing */
# undef  UV_VIS2
# ifdef UV_VIS2
#  define UV_VIS_SMAGO
# endif
                      /* Lateral Explicit Tracer Mixing */
# undef  TS_DIF2
# undef  TS_DIF4
# undef  TS_MIX_S
                      /* Sponge layers for UV and TS */
# define SPONGE
                      /* Semi-implicit Vertical Tracer/Mom Advection */
# undef  VADV_ADAPT_IMP
                      /* Vertical Mixing */
# undef  BODYFORCE
# undef  BVF_MIXING
# define LMD_MIXING
# undef  GLS_MIXING
# ifdef LMD_MIXING
#  define LMD_SKPP
#  define LMD_BKPP
#  define LMD_RIMIX
#  define LMD_CONVEC
#  undef  LMD_DDMIX
#  define LMD_NONLOCAL
# endif
# ifdef GLS_MIXING
#  define GLS_KKL
#  undef  GLS_KOMEGA
#  undef  GLS_KEPSILON
#  undef  GLS_GEN
#  undef  KANTHA_CLAYSON
#  undef  CRAIG_BANNER
#  undef  CANUTO_A
#  undef  ZOS_HSIG
# endif
```

Vertical Mixing Parameterization

BVF_MIXING = Simple mixing scheme based on the Brunt-Väisälä frequency

LMD_MIXING = Large/McWilliams/Doney mixing (turbulent closure for interior and planetary boundary layers) = KPP

GLS_MIXING = Generic Length Scale scheme