

Numerical Modelling

Jonathan GULA
gula@univ-brest.fr

the anatomy of an ocean model

- **Lesson 1 : [D109]**
 - Introduction
 - Equations of motions
 - *Activity 1 [run an ocean model]*
 - **Lesson 2 : [B 012]**
 - Horizontal Discretization
 - *Activity 2 [Dynamics of an ocean gyre]*
 - **Lesson 3 : [D109]**
 - Presentation of the model CROCO
 - Dynamics of the ocean gyre
 - *Activity 2 [Dynamics of an ocean gyre]*
 - **Lesson 4 : [D109]**
 - Numerical schemes
 - *Activity 3 [Impacts of numerics]*
 - **Lesson 5 : [D109]**
 - Vertical coordinates
 - **Lesson 6 : [D109]**
 - Model parameterizations
 - *Activity 4 [Impact of topography]*
 - **Lesson 7 : [D109]**
 - Boundary Forcings
 - *Activity 4 [Design a realistic simulation]*
 - **Lesson 8 : [D109]**
 - *Work on your projet*
- Presentations and material will be available at :
- jgula.fr/ModNum/**

#4

Numerical options in CROCO

Choice of numerics:

- `cppdefs.h`

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

*/
#ifndef 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 */
#define UV_HADV_C4         /* 4th-order centered lateral advection */
#define UV_HADV_C2         /* 2nd-order centered lateral advection */
#endif
/*
   UV DIFFUSION: set default orientation
*/
#ifndef 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
*/
#ifndef UV_VIS_SMAGO
#define VIS_COEF_3D
#endif
/*
   Set UP3 scheme in barotropic equations for 2DH applications
*/
#ifndef 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)
*/
#ifndef M3CLIMATOLOGY
#define CLIMAT_UV_MIXH
#endif
=====
```

cppdefs.h

```
#if defined REGIONAL
/*
=====
!           REGIONAL (realistic) Configurations
=====
!
!
!-----+
! BASIC OPTIONS
!-----+
!
*/
# define BENGUELA_LR      /* Configuration Name */
# define BENGUELA_LR      /* Parallelization */

# undef OPENMP             /* Nesting */
# undef MPI                /* OA and OW Coupling via OASIS (MPI) */
# undef AGRIF              /* I/O server */
# undef AGRIF_2WAY          /* Open Boundary Conditions */
# undef OA_COUPLING         /* Applications */
# undef OW_COUPLING          */
# undef XIOS                */
# undef TIDES               */
# define OBC_EAST            */
# define OBC_WEST            */
# define OBC_NORTH           */
# define OBC_SOUTH           */
# 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.

```
# 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
!-----+
!
*/
# define BENGUELA_LR
                /* Configuration Name */
                /* Parallelization */
# undef OPENMP
# undef MPI
                /* Nesting */
# undef AGRIF
# undef AGRIF_2WAY
# undef OA_COUPLING
# undef OW_COUPLING
# undef XIOS
                /* I/O server */
# undef TIDES
# define OBC_EAST
# define OBC_WEST
# define OBC_NORTH
# define OBC_SOUTH
                /* Open Boundary Conditions */
# undef BIOLOGY
# undef FLOATS
# undef STATIONS
# undef PASSIVE_TRACER
# undef SEDIMENT
# undef BBL
                /* Applications */
/*!
```

Parallelization options:

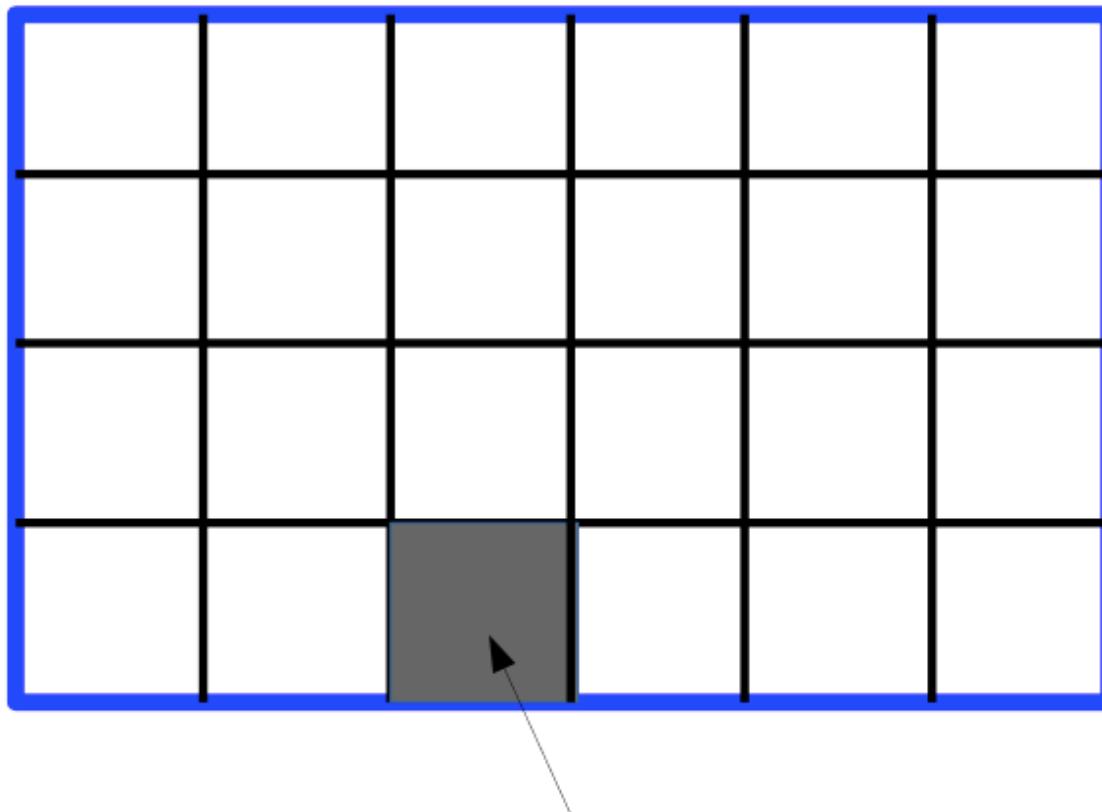
Activation of MPI and/or openMP parallelization:

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
#endif 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

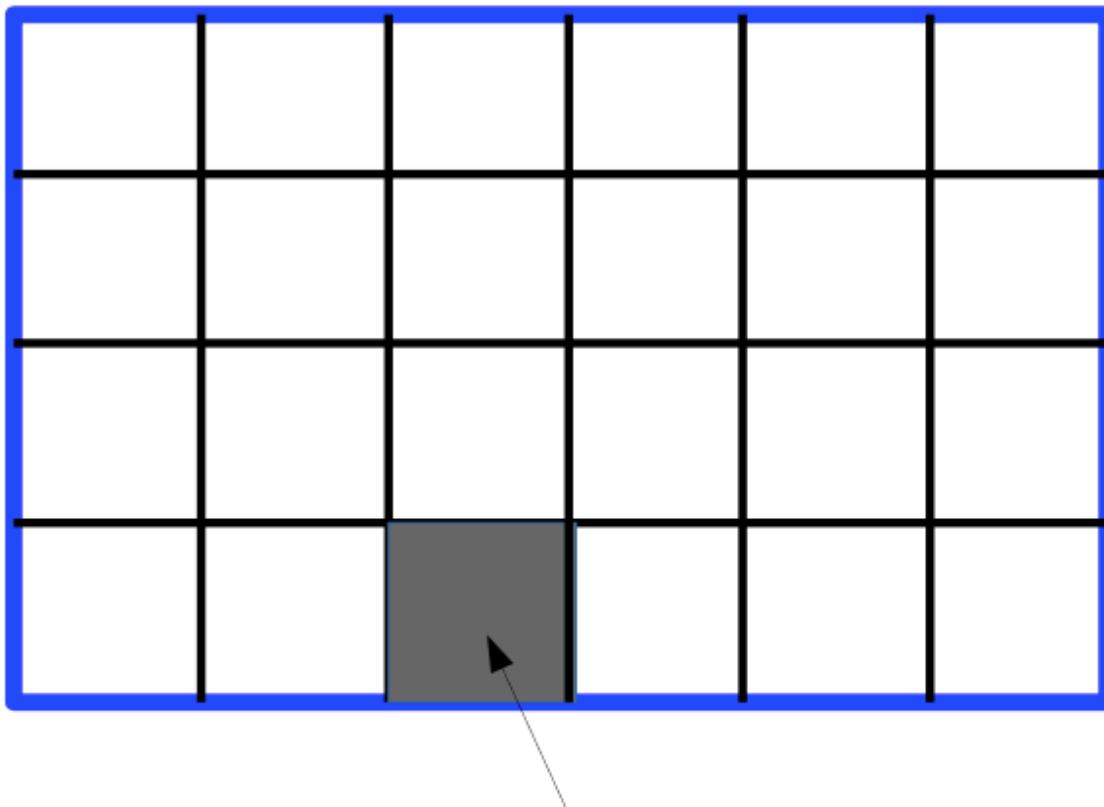


NPP = number of threads (cores)

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.

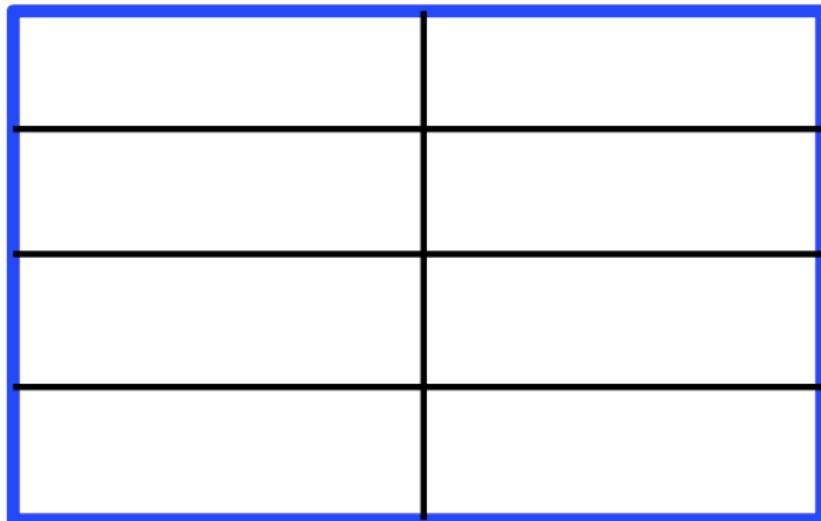
NSUB_X * NSUB_E has to be a multiple of NPP.

Default is (see param.h)

$NSUB_X = 1, NSUB_E = NPP$

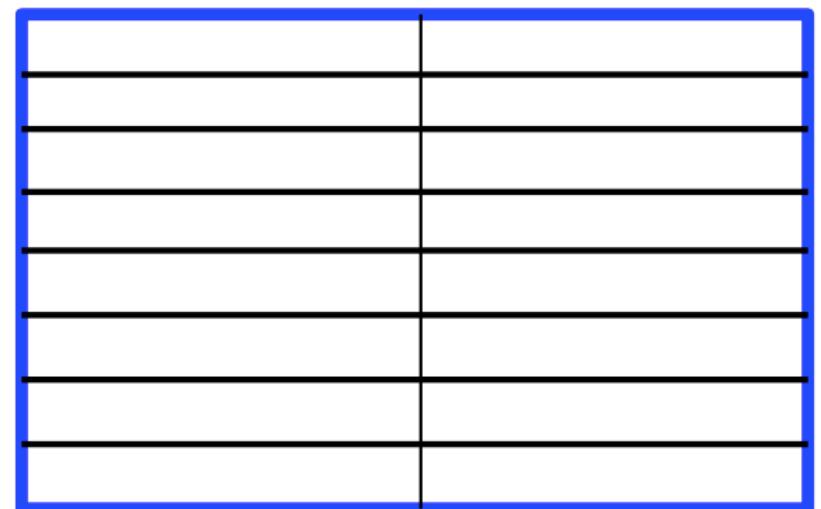
define OPENMP

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.

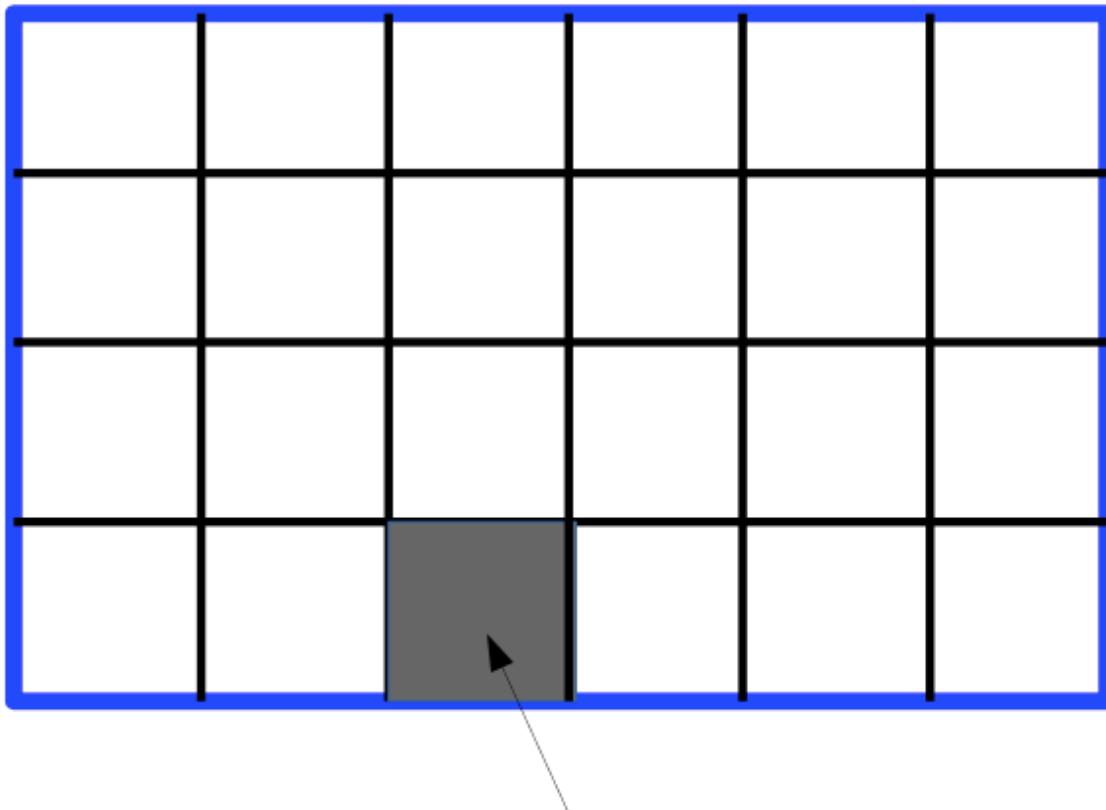
Also has to synchronize between different threads

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

MPI = distributed – memory



Ex: $NP_{XI} = 6, NP_{ETA} = 4$

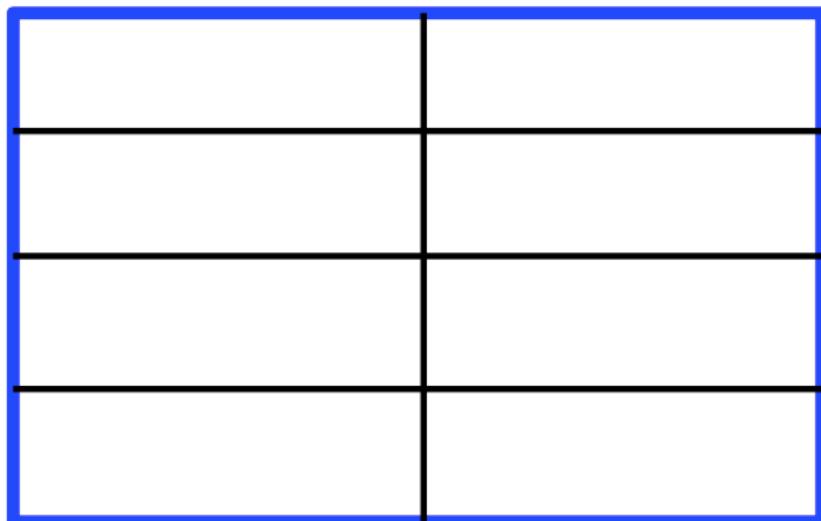
$NP_{XI} * NP_{ETA} =$ number of threads (cores)

NPP = 1

$NSUB_X * NSUB_E$ can be anything

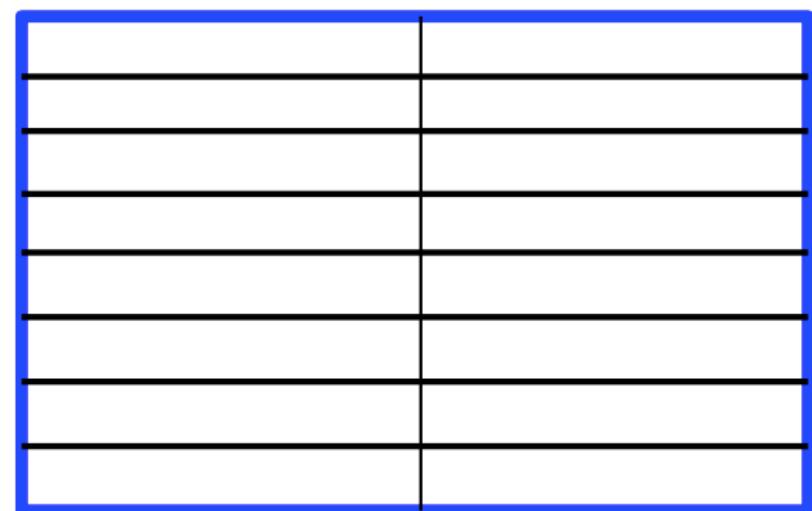
define MPI

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))
        &                                     v(START_2D_ARRAY,1,nnew))

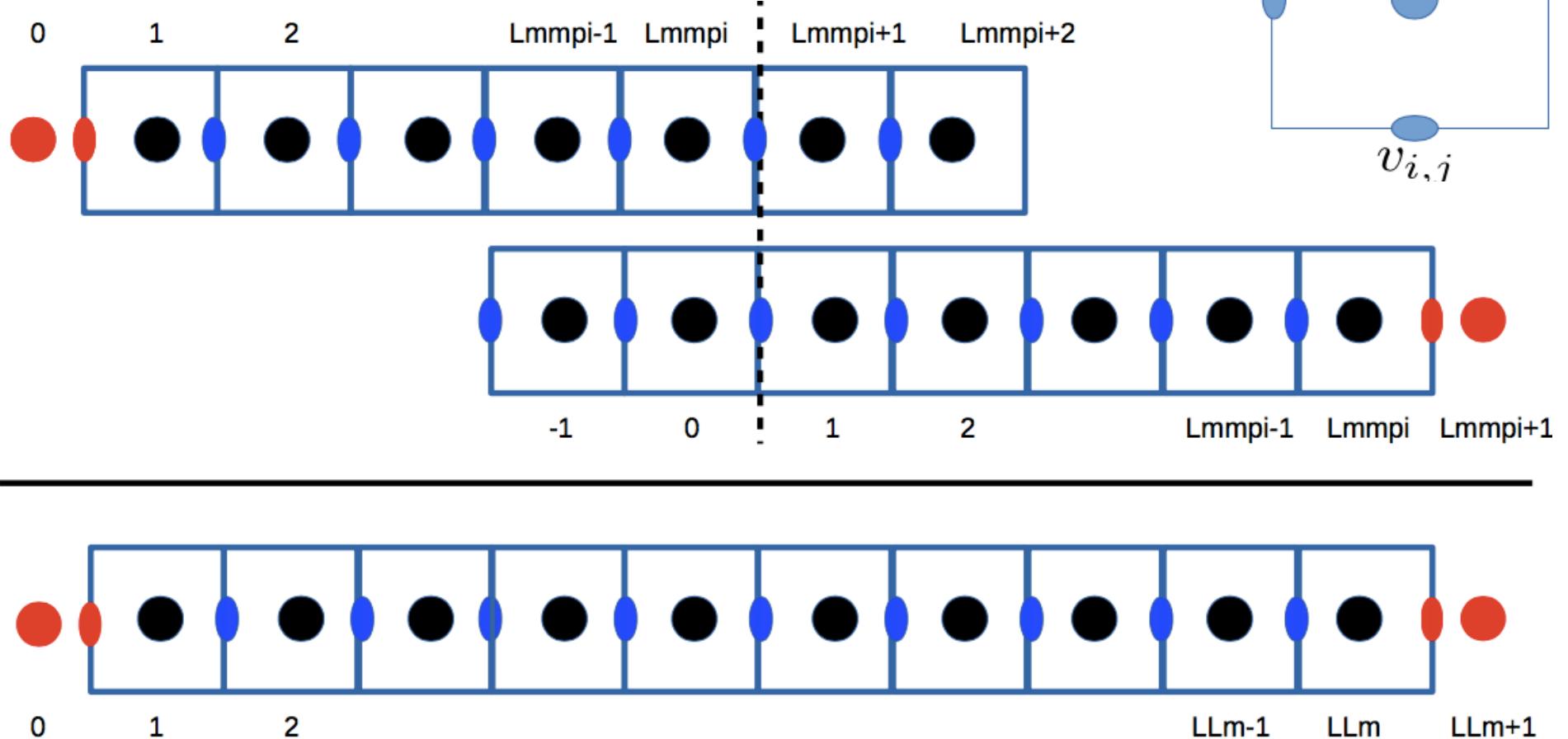
        call exchange_u3d_tile (Istr,Iend,Jstr,Jend,
        &                                     Huon(START_2D_ARRAY,1))
        &                                     Hvom(START_2D_ARRAY,1))

        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

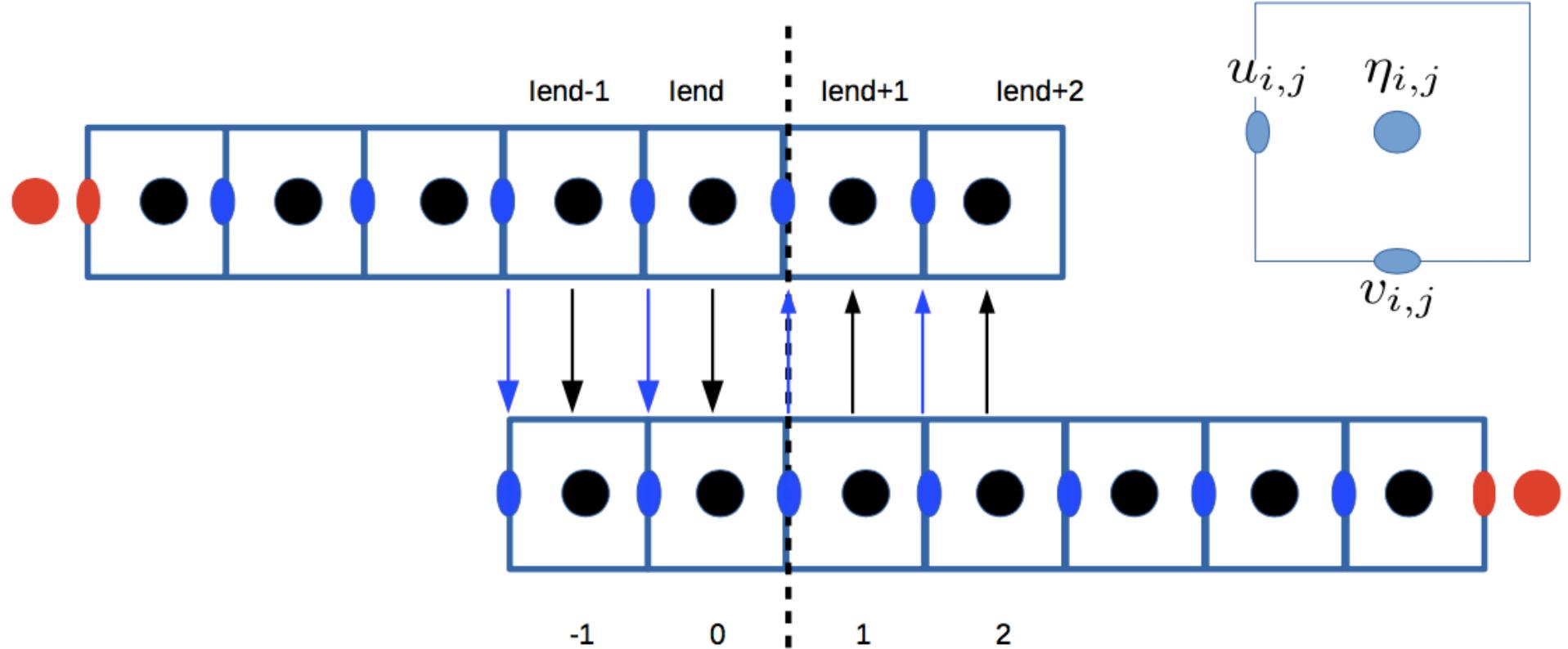
MPI domains:



Full domain:

define MPI

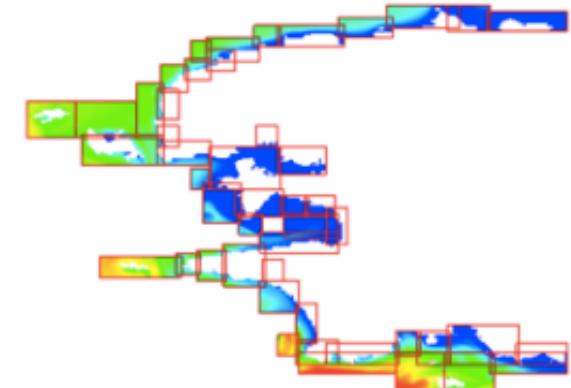
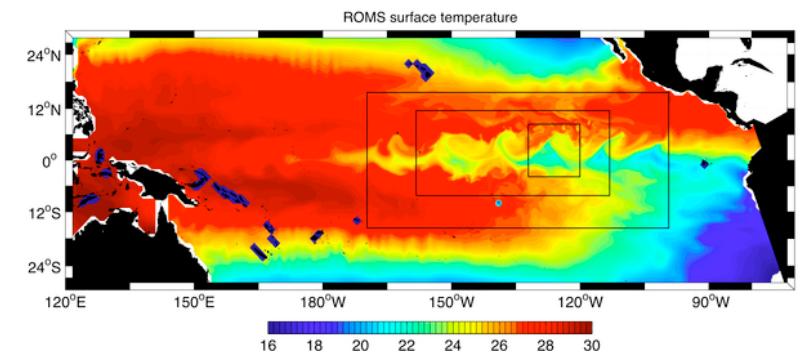
MPI exchanges:



cppdefs.h

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

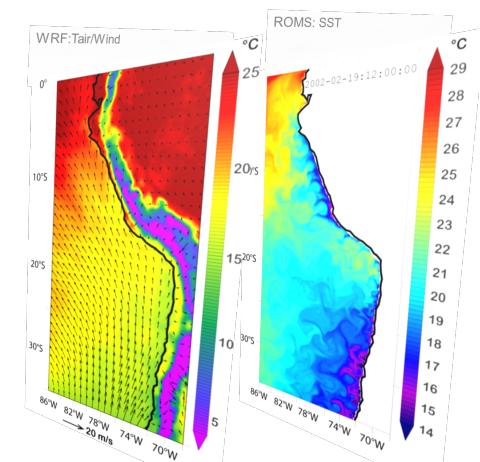
Online Nesting options:



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
#define XIOS                  /* I/O server */
/* 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 */

#define OPENMP                /* Nesting */
#define MPI                   /* OA and OW Coupling via OASIS (MPI) */

#define AGRIF                 /* I/O server */
#define AGRIF_2WAY             /* Open Boundary Conditions */

#define OA_COUPLING           /* TIDES */
#define OW_COUPLING            /* OBC_EAST */
#define XIOS                  /* OBC_WEST */
#define OBC_NORTH              /* OBC_SOUTH */

#define BIOLOGY               /* Applications */
#define FLOATS
#define STATIONS
#define PASSIVE_TRACER
#define SEDIMENT
#define 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 */

#define OPENMP                /* Nesting */
#define MPI                   /* OA and OW Coupling via OASIS (MPI) */

#define AGRIF                /* I/O server */
#define AGRIF_2WAY             /* Open Boundary Conditions */

#define OA_COUPLING           /* Applications */
#define OW_COUPLING           

#define XIOS                  /* TTDIFES */

#define OBC_EAST               /* BIOLOGY */
#define OBC_WEST               /* FLOATS */
#define OBC_NORTH              /* STATIONS */
#define OBC_SOUTH              /* PASSIVE_TRACER */
#define SEDIMENT               /* SEDIMENT */
#define BBL                     /* 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

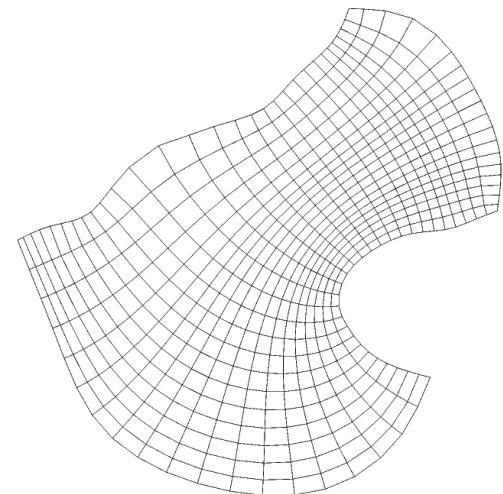
cppdefs.h

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

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  
  
# define CURVGRID  
# define SPHERICAL  
# define MASKING  
# undef WET_DRY  
# undef NEW_S_COORD  
  
                  /* Grid configuration */  
  
# define SOLVE3D  
# define UV_COR  
# define UV_ADV  
  
# define SALINITY  
# define NONLIN_EOS  
# define SPLIT_EOS  
... - - - - - ...  
                  /* Model dynamics */  
  
                  /* Equation of State */
```



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

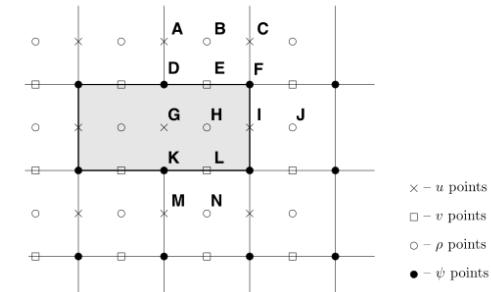
```
!-----  
! PRE-SELECTED OPTIONS  
!  
! ADVANCED OPTIONS ARE IN CPPDEFS_DEV.H  
!-----  
*/  
/* Parallelization */  
# ifdef MPI  
# undef PARALLEL_FILES  
# endif  
# undef AUTOTILING  
# undef ETALON_CHECK  
/* Grid configuration */  
# define CURVGRID  
# define SPHERICAL  
# define MASKING  
# 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



```
# ifndef _LAMBERT_CONVENTION_#  
# 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  
#endif
```

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).



```
# ifndef _WETTING_DRYING_  
# 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  
#endif
```

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 */

#ifndef MPI
#define UNDEF_PARALLEL_FILES
#endif
#define UNDEF_AUTOTILING
#define UNDEF_ETALON_CHECK
                    /* Grid configuration */

#define CURVGRID
#define SPHERICAL
#define MASKING
#define UNDEF_WET_DRY
#define 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 Coriolis terms

See *rhs3d.F*:



```

do j=JstrV-1,Jend
    do i=IstrU-1,Iend
        cff=0.5*Hz(i,j,k)*(
#endif
        & 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))
        VFx(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 */

#ifndef MPI
#define UNDEF_PARALLEL_FILES
#endif
#ifndef AUTOTILING
#define UNDEF_ETALON_CHECK
#endif

                    /* Grid configuration */

#define CURVGRID
#define SPHERICAL
#define MASKING
#ifndef WET_DRY
#define NEW_S_COORD
#endif

                    /* 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

```

#ifndef UV_ADV
!-----
! Add in horizontal advection of momentum.
! Compute diagonal [UFX,VFX] and off-diagonal [UFX,VFX] components
! of tensor of momentum flux due to horizontal advection; after that
! add in horizontal advection terms.
!-----
#endif

#ifndef UV_HADV_UP3 || defined UV_HADV_C4 || defined UV_HADV_C2
#define UV_HADV_UP3
#endif

#ifndef UV_HADV_UP3 || defined UV_HADV_C4
!-----
! Fourth or Third order advection scheme (default)
!-----

#define UXX_WRK1
#define HUXX_WRK2
#ifndef EW_PERIODIC
#define IU_EXT_RANGE IstrU-1,Iend+1
#else
#define 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,Lmmp)
    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
        ux(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

```

See *rhs3d.F*

cppdefs.h

```
!---
! PRE-SELECTED OPTIONS
!
! ADVANCED OPTIONS ARE IN CPPDEFS_DEV.H
!---
*/
                    /* Parallelization */
#ifndef MPI
#define UNDEF_PARALLEL_FILES
#endif
#ifndef AUTOTILING
#define UNDEF_ETALON_CHECK
                    /* Grid configuration */
#define CURVGRID
#define SPHERICAL
#define MASKING
#define UNDEF_WET_DRY
#define UNDEF_NEW_S_COORD
                    /* Model dynamics */
#define SOLVE3D
#define UV_COR
#define UV_ADV
                    /* Equation of State */
#define SALINITY
#define NONILTN_EOS
#define SPLIT_EOS
```

Define salinity as an active tracer

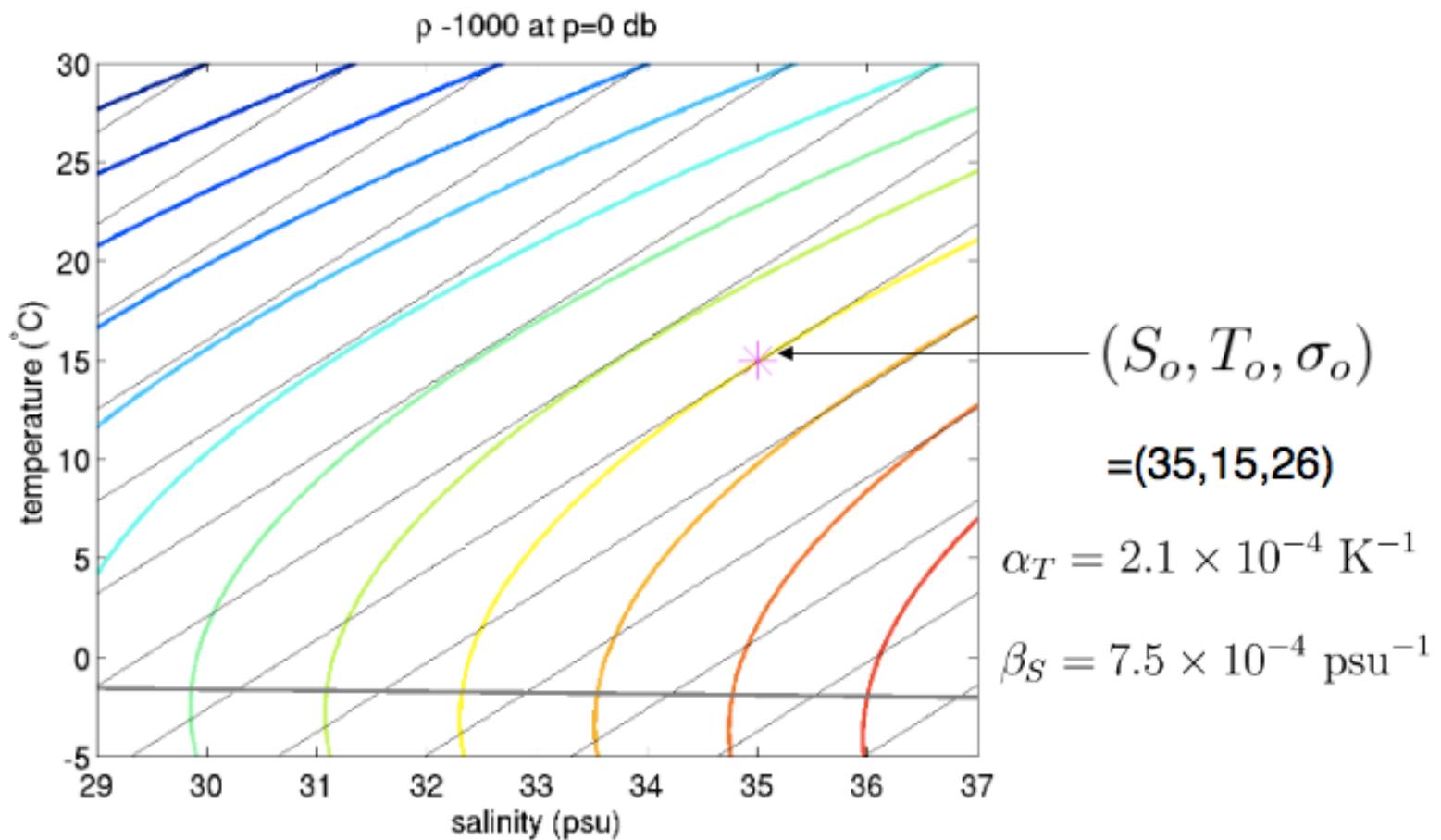
cppdefs.h

```
!-----  
! 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  
  
    rho(T,S,z) =  $\frac{\rho_0 + \rho_1(T,S)}{1 - 0.1|z|/K(T,S,|z|)}$  (1)  
  
where  $\rho_1(T,S)$  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  
  
 $K(T,S,|z|) = K_00 + K_01(T,S) + K_1(T,S)*|z| + K_2(T,S)*|z|^2$ . (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,  
  
 $\frac{\rho_0}{1 - 0.1|z|/K_00}$  (3)  
  
is removed from (1). [Since (3) is purely a function of  $z$ ,  
it does not contribute to pressure gradient.] This results in  
  
 $\frac{\rho_1 - \rho_0*[K_01+K_1*|z|+K_2*|z|^2]/[K_00-0.1|z|]}{K_00 + K_01 + (K_1-0.1)*|z| + K_2*|z|^2}$  (4)  
which is suitable for pressure-gradient calculation.  
  
Optionally, if CPP-switch SPLIT_EOS is defined, term proportional  
to  $|z|$  is linearized using smallness  $0.1|z|/[K_00 + K_01] \ll 1$  and  
the resultant EOS has form  
  
 $\rho(T,S,z) = \rho_1(T,S) + q\rho_1(T,S)*|z|$  (5)  
  
where  
 $q\rho_1(T,S) = 0.1 \frac{\rho_0*K_01(T,S)/K_00}{K_00 + K_01(T,S)}$  (6)  
  
is stored in a special array.
```

cppdefs.h



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

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

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
#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 */

/* 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
#endif
```

Choose horizontal advective scheme for momentum

See *rhs3d.F*:

cppdefs.h

```
# define UV_HADV_UP3
# undef  UV_HADV_C4
# undef  UV_HADV_C2
/* Lateral Momentum Advection (default UP3) */

# undef  TS_HADV_UP3
# define TS_HADV_RSUP3
# undef  TS_HADV_UP5
# undef  TS_HADV_C4
# undef  TS_HADV_WENO5
/* Lateral Tracer Advection (default UP3) */

# undef  UV_VIS2
# ifdef UV_VIS2
#  define UV_VIS_SMAGO
# endif
/* Lateral Explicit Momentum Mixing */

# undef  TS_DIF2
# undef  TS_DIF4
# undef  TS_MIX_S
/* Lateral Explicit Tracer Mixing */

# define SPONGE
/* Sponge layers for UV and TS */

# undef  VADV_ADAPT_IMP
/* Semi-implicit Vertical Tracer/Mom Advection */

# 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 */
```

Choose horizontal advective scheme for tracers

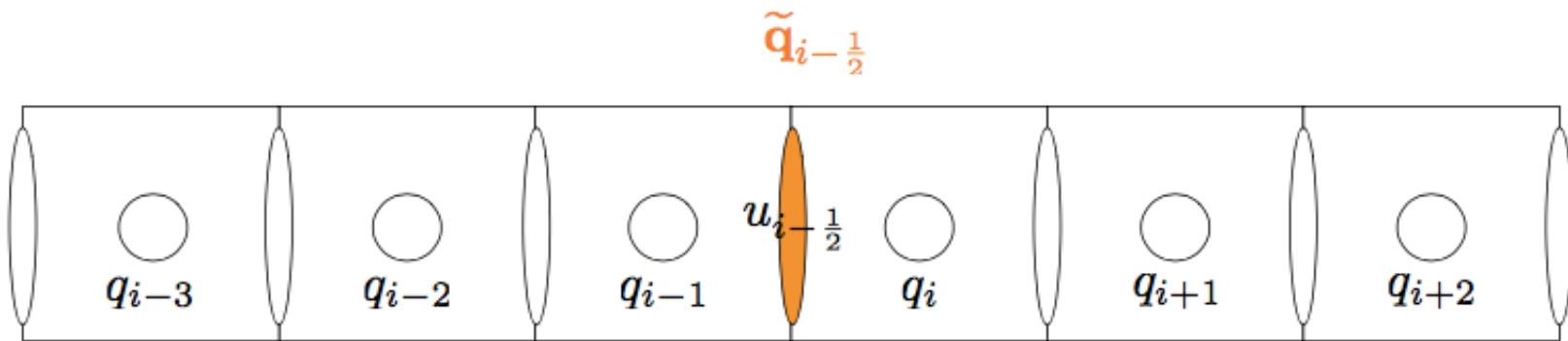
See *step3d_t.F* and
compute_horiz_tracer_fluxes.h

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

Horizontal Advection Schemes

[HADV_C2, HADV_UP3, HADV_C4, HADV_UP5, HADV_C6]



$$\partial_x(uq)|_{x=x_i} = \frac{1}{\Delta x_i} \{ u_{i+1/2} \tilde{q}_{i+1/2} - u_{i-1/2} \tilde{q}_{i-1/2} \}$$

$$\tilde{q}_{i-1/2}^{\text{C2}} = \frac{q_i + q_{i-1}}{2}$$

$$\tilde{q}_{i-1/2}^{\text{C4}} = (7/6)\tilde{q}_{i-1/2}^{\text{C2}} - (1/12)(q_{i+1} + q_{i-2})$$

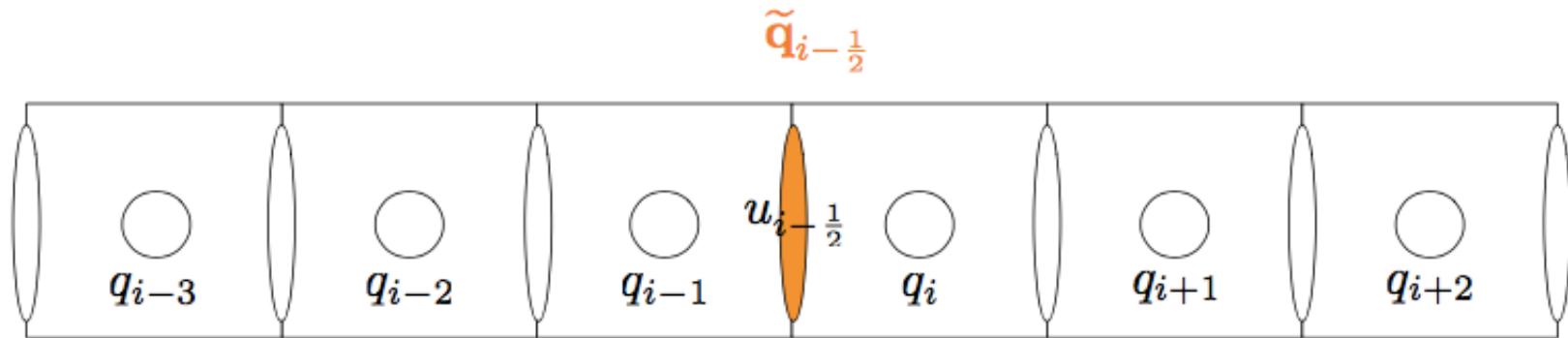
$$\tilde{q}_{i-1/2}^{\text{UP3}} = \tilde{q}_{i-1/2}^{\text{C4}} + \text{sign}(1/12, u_{i-1/2})(q_{i+1} - 3q_i + 3q_{i-1} - q_{i-2})$$

$$\tilde{q}_{i-1/2}^{\text{C6}} = (8/5)\tilde{q}_{i-1/2}^{\text{C4}} - (19/60)\tilde{q}_{i-1/2}^{\text{C2}} + (1/60)(q_{i+2} + q_{i-3})$$

$$\tilde{q}_{i-1/2}^{\text{UP5}} = \tilde{q}_{i-1/2}^{\text{C6}} - \text{sign}(1/60, u_{i-1/2})(q_{i+2} - 5q_{i+1} + 10q_i - 10q_{i-1} + 5q_{i-2} - q_{i-3})$$

Horizontal Advection Schemes

[HADV_RSUP3,HADV_RSUP5]



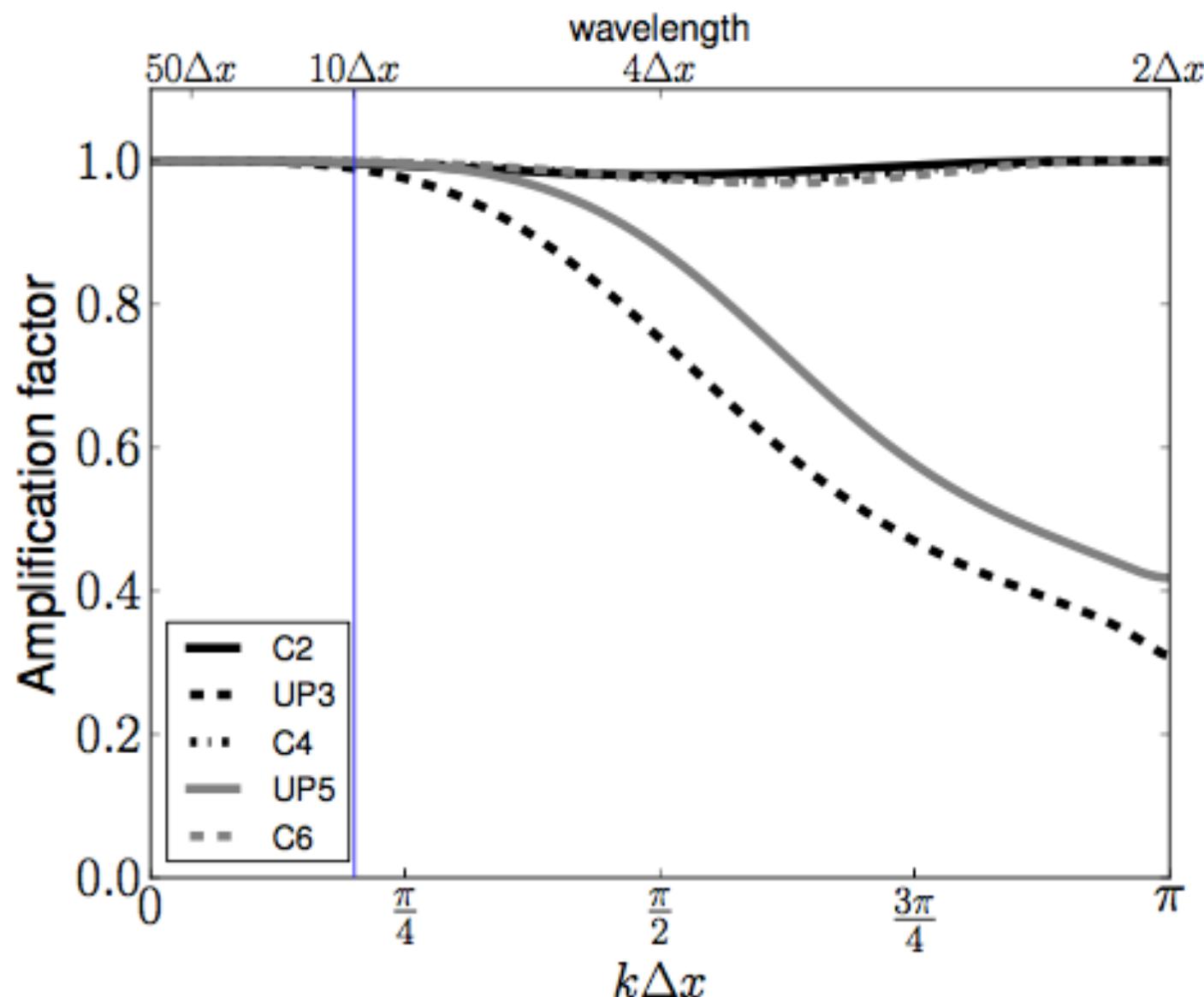
$$\partial_x(uq)|_{x=x_i} = \frac{1}{\Delta x_i} \{ u_{i+1/2} \tilde{q}_{i+1/2} - u_{i-1/2} \tilde{q}_{i-1/2} \}$$

$$\tilde{q}_{i-1/2}^{\text{UP3}} = \tilde{q}_{i-1/2}^{\text{C4}} + \text{sign}(1/12, u_{i-1/2})(q_{i+1} - 3q_i + 3q_{i-1} - q_{i-2})$$

$$\tilde{q}_{i-1/2}^{\text{UP5}} = \tilde{q}_{i-1/2}^{\text{C6}} - \text{sign}(1/60, u_{i-1/2})(q_{i+2} - 5q_{i+1} + 10q_i - 10q_{i-1} + 5q_{i-2} - q_{i-3})$$

+ SPLIT – UP3 and SPLIT – UP5 schemes ()

Horizontal Advective Schemes



Vertical Advective Schemes

- `cppdefs_dev.h`

```
=====
Select MOMENTUM VERTICAL advection scheme:
=====

*/
#ifndef UV_VADV_SPLINES /* Check if options are defined in cppdefs.h */
#elif defined UV_VADV_C2
#else
#define UV_VADV_SPLINES /* Splines vertical advection */
#define UV_VADV_C2 /* 2nd-order centered vertical advection */
#endif

#ifndef VADV_ADAPT_IMP
#define VADV_ADAPT_PRED
#define UV_VADV_SPLINES
#define UV_VADV_C2
#endif
```

```
/*
=====
Select model dynamics for TRACER vertical advection
(The default is 4th-order centered)
=====

*/
#ifndef 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 */

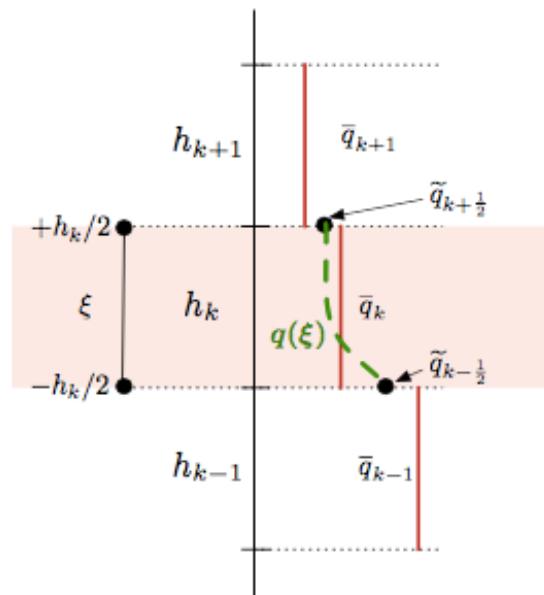
#ifndef VADV_ADAPT_IMP
#define TS_VADV_SPLINES
#undef TS_VADV_AKIMA
#undef TS_VADV_WENO5
#undef TS_VADV_C2
#endif
```

Vertical Advection Schemes

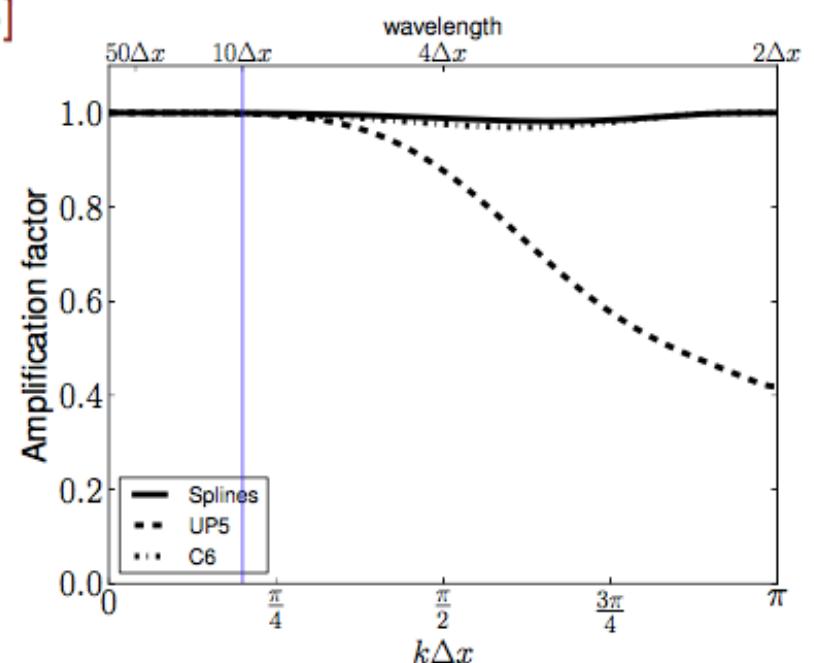
[TS_VADV_SPLINES, UV_VADV_SPLINES]

Les flux sont obtenus comme solution du problème tridiagonal

$$\begin{aligned} \text{Hz}_{k+1}\tilde{q}_{k-1/2} + 2(\text{Hz}_k + \text{Hz}_{k+1})\tilde{q}_{k+1/2} + \text{Hz}_k\tilde{q}_{k+3/2} \\ = 3(\text{Hz}_k\bar{q}_{k+1} + \text{Hz}_{k+1}\bar{q}_k) \end{aligned}$$



$$\begin{aligned} q(\xi) = \bar{q}_k + \frac{\tilde{q}_{k+\frac{1}{2}} - \tilde{q}_{k-\frac{1}{2}}}{h_k} \xi \\ + 6 \left(\frac{\tilde{q}_{k+\frac{1}{2}} + \tilde{q}_{k-\frac{1}{2}}}{2} - \bar{q}_k \right) \left(\frac{\xi^2}{h_k^2} - \frac{1}{12} \right) \end{aligned}$$



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 */

/* 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
#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
#define UV_HADV_C4
#define 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_WEN05
/* Lateral Explicit Momentum Mixing */
#undef UV_VIS2
#ifndef UV_VIS2
#define UV_VIS_SMAGO
#endif
#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
#ifndef LMD_MIXING
#define LMD_SKPP
#define LMD_BKPP
#define LMD_RIMIX
#define LMD_CONVEC
#undef LMD_DDMIX
#define LMD_NONLOCAL
#endif
#endif
#ifndef 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

$$-\nabla_h \cdot \langle \mathbf{u}'_h \mathbf{u}'_h \rangle = \frac{1}{\text{Hz}} \nabla_h \cdot (A_M \text{ Hz } \boldsymbol{\sigma}), \quad A_M \leftrightarrow \text{visc2}$$

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 ($B_M \leftrightarrow \text{visc4}$)

$$-\nabla_h \cdot \langle \mathbf{u}'_h \mathbf{u}'_h \rangle = -\frac{1}{\text{Hz}} \nabla_h \cdot (B_M \text{ Hz } \boldsymbol{\sigma}'), \quad \boldsymbol{\sigma}' = \boldsymbol{\sigma}(\nabla_h \cdot \boldsymbol{\sigma}(\mathbf{u}_h))$$

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 $C_M = 1/10$ (paramètre *horcon* dans la routine *hvisc_coef*)

[TS_DIF_SMAGO, TS_DIF2]

Coefficient de diffusion turbulente

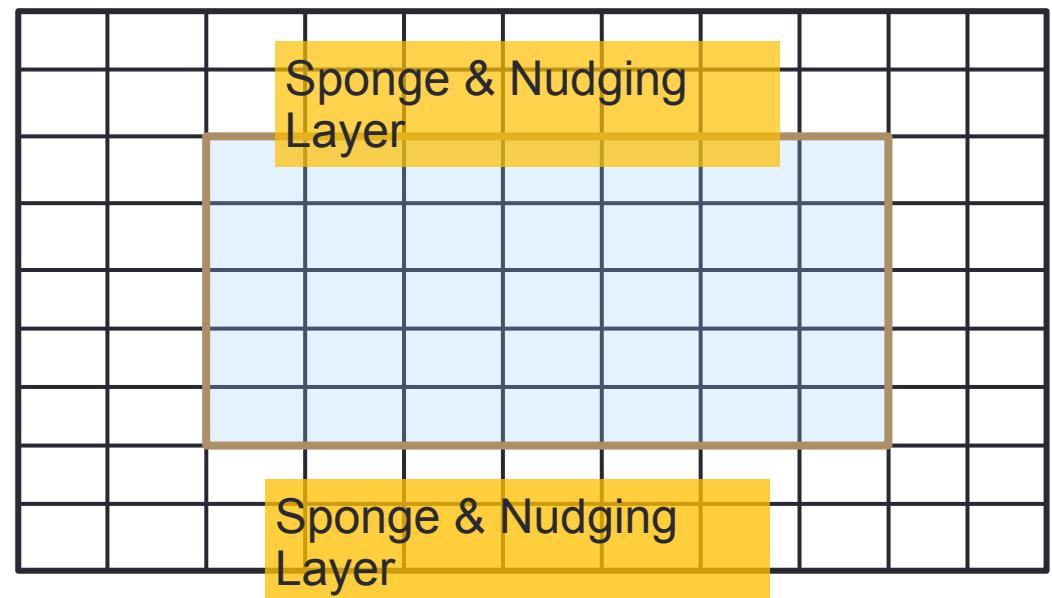
$$A_S = C_S (\Delta x \Delta y) \sqrt{(\partial_x u)^2 + (\partial_y v)^2 + 2(\partial_y u + \partial_x v)^2}$$

Par défaut $C_S = 1/12$ (paramètre *horcon* dans la routine *hdifff_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
#ifndef 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 DVT_MIXING
#define LMD_MIXING
#undef GLS_MIXING
#ifndef LMD_MIXING
#define LMD_SKPP
#define LMD_BKPP
#define LMD_RIMIX
#define LMD_CONVEC
#undef LMD_DDMIX
#define LMD_NONLOCAL
#endif
#ifndef 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
#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
#ifndef UV_VIS2
#define UV_VIS_SMAGO
#endif

/* Lateral Explicit Tracer Mixing */

/* Sponge layers for UV and TS */
#define SPONGE

/* Semi-implicit Vertical Tracer/Mom Advection */
#undef VADV_ADAPT_IMP

/* Vertical Mixing */
#undef BODYFORCE

#define BVF_MIXING
#define LMD_MIXING
#undef GLS_MIXING
#ifndef LMD_MIXING
#define LMD_SKPP
#define LMD_BKPP
#define LMD_RIMIX
#define LMD_CONVEC
#undef LMD_DDMIX
#define LMD_NONLOCAL
#endif
#ifndef 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
#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

#2 Subgrid-scale parameterization

Incompressible Navier-Stokes Equations:

- Dissipation of energy/momentum in the NS equations:

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \vec{\nabla} \vec{u} + f \vec{k} \times \vec{u} + g \vec{k} = -\frac{\vec{\nabla} P}{\rho} + \nu \vec{\nabla}^2 \vec{u}$$

Viscosity

Type of models

Navier
Stokes

- DNS = Direct Numerical Simulation
- LES = Large Eddy Simulation

CFD

Process
studies

PE

- PE = Primitive Equations models

Ocean
Circulation
Models

SW

- SW = Shallow-Water models

SQG

- SQG = Surface Quasi-Geostrophic models

QG

- QG = Quasi-Geostrophic models

Idealized
models

- Etc.

Incompressible Navier-Stokes Equations:

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \vec{\nabla} \vec{u} + f \vec{k} \times \vec{u} + g \vec{k} = -\frac{\vec{\nabla} P}{\rho} + \nu \vec{\nabla}^2 \vec{u}$$

Non-linear
terms

Viscosity

- Importance of NL terms and viscosity = Reynolds Number

$$Re = \frac{UL}{\nu}$$

Where U is a typical velocity of the flow and L is a typical length describing the flow.

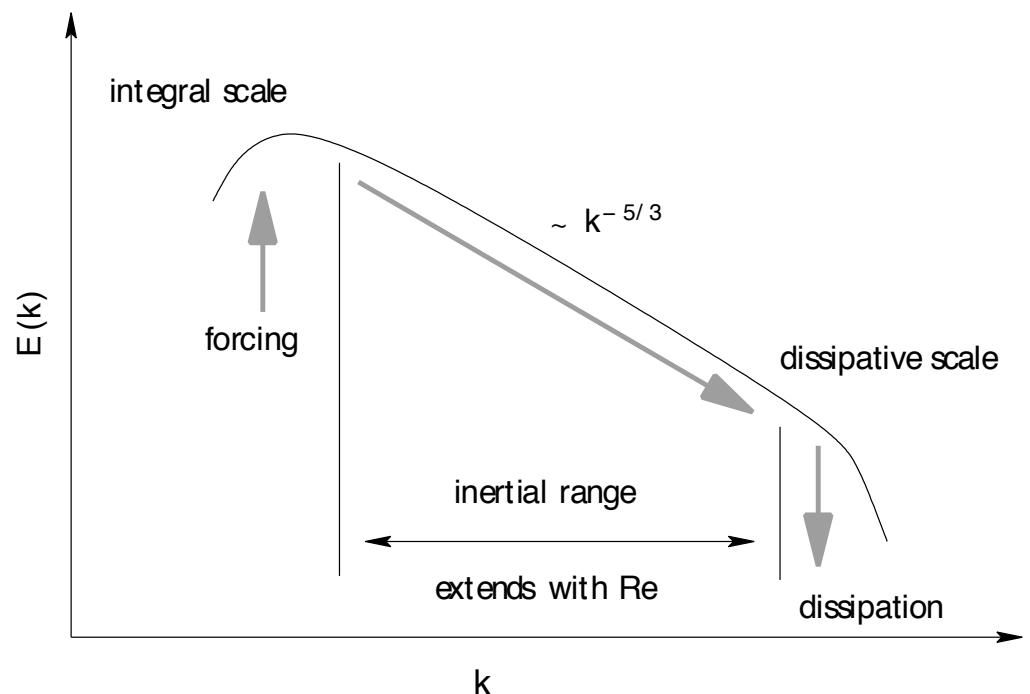
Direct numerical simulation (DNS)

DNS resolves the entire range of turbulent length scales down to the smallest dissipative scales (Kolmogorov scale):

$$\eta = \left(\frac{\nu^3}{\epsilon} \right)^{1/4} \approx \left(\frac{\nu^3 L}{U^3} \right)^{1/4} = Re^{-3/4} L$$

where ν is the kinematic viscosity

And ϵ the rate of kinetic energy dissipation



Direct numerical simulation (DNS)

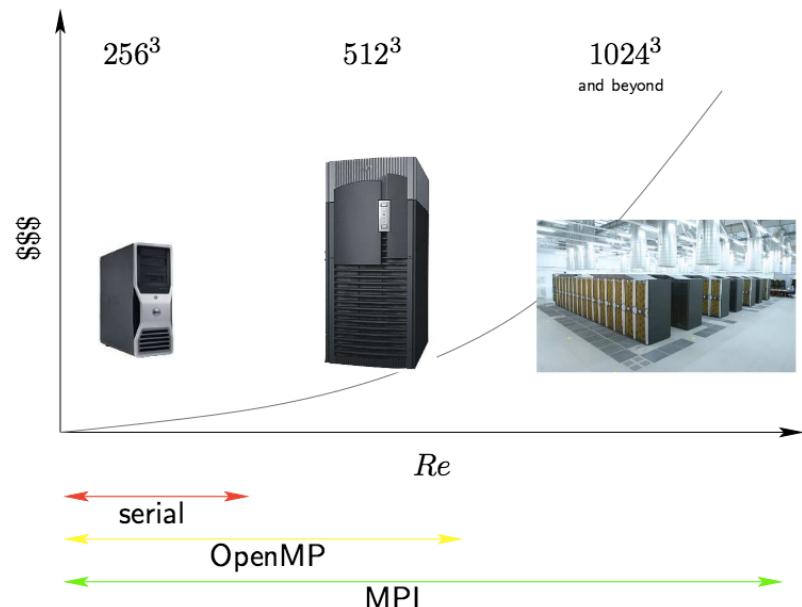
The number of floating-point operations required to complete the simulation is proportional to the number of mesh points:

$$N_x = \frac{L}{\eta} = Re^{3/4}$$

$$\frac{T}{\Delta t} = \frac{TU}{\eta} = \frac{TU}{L} Re^{3/4}$$

and the number of time steps:

It is extremely expensive as the computational cost scales as Re^3



Type of models

Navier
Stokes

- DNS = Direct Numerical Simulation
- LES = Large Eddy Simulation

PE

- PE = Primitive Equations models

SW

- SW = Shallow-Water models

SQG

- SQG = Surface Quasi-Geostrophic models

QG

- QG = Quasi-Geostrophic models

- Etc.

CFD

Process
studies

Ocean
Circulation
Models

Idealized
models

Reynolds averaging

To reduce the computational cost, one need to reduce the range of time- and length-scales that are being solved for.

The idea is based on separation of mean and turbulent component:

$$u = \bar{u} + u'$$

Where

$$\bar{u} = \frac{1}{T} \int_0^T u \, dt \text{ or } \bar{u} = \frac{1}{X} \int_0^X u \, dx$$

With by definition

$$\overline{u'} = 0$$

Reynolds averaging

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \vec{\nabla} \vec{u} + f \vec{k} \times \vec{u} + g \vec{k} = -\frac{\vec{\nabla} P}{\rho} + \nu \vec{\nabla}^2 \vec{u}$$

- Activity:

Adapt the momentum equation:

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} + f \mathbf{k} \times u_i + \frac{\rho}{\rho_0} g \mathbf{k} = -\frac{1}{\rho_0} \frac{\partial P}{\partial x_j} + \nu \frac{\partial^2 u_i}{\partial x_i \partial x_j}$$

For the mean velocity: $\frac{\partial \bar{u}_i}{\partial t} = ?$

Reynolds averaging

So we resolve only the equations for the mean variables:

$$\frac{\partial \bar{u}_i}{\partial t} + \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j} + f \mathbf{k} \times \bar{u}_i + \frac{\bar{\rho}}{\rho_0} g \mathbf{k} = - \frac{1}{\rho_0} \frac{\partial \bar{P}}{\partial x_j} + \nu \frac{\partial^2 \bar{u}_i}{\partial x_i \partial x_j}$$



$$\frac{\partial \bar{u}_i}{\partial t} + \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j} + f \mathbf{k} \times \bar{u}_i + \frac{\bar{\rho}}{\rho_0} g \mathbf{k} = - \frac{1}{\rho_0} \frac{\partial \bar{P}}{\partial x_j} + \nu \frac{\partial^2 \bar{u}_i}{\partial x_i \partial x_j} - \frac{\partial \bar{u}'_i \bar{u}'_j}{\partial x_j}$$

Advection for the averaged flow

Reynolds stress
= effect of subgrid-scale turbulence

Turbulence closure

The Closure Problem :

- The number of unknowns is larger than the number of equations
- When we start deriving equations for unknowns, more variables appear.

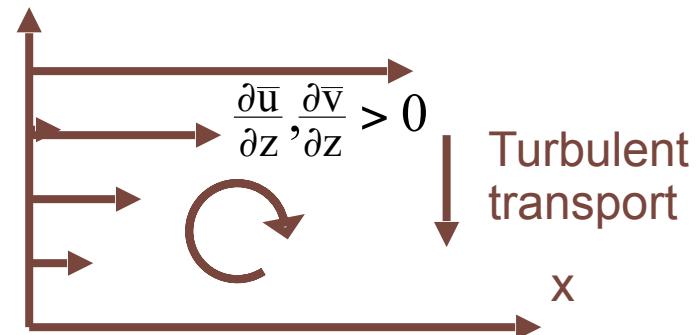
Progn. Eq. for:	Moment	Equation	No. Eqs.	No. Unknowns
$\overline{U_i}$	First	$\frac{\partial \overline{U_i}}{\partial t} = \dots - \frac{\partial u'_i u'_j}{\partial x_j}$	3	6
$\overline{u'_i u'_j}$	First	$\frac{\partial \overline{u'_i u'_j}}{\partial t} = \dots - \frac{\partial u'_k u'_i u'_j}{\partial x_k}$	6	10
$\overline{u'_i u'_j u'_k}$	First	$\frac{\partial \overline{u'_i u'_j u'_k}}{\partial t} = \dots - \frac{\partial u'_k u'_i u'_j u'_m}{\partial x_m}$	10	15

Turbulence closure

- In PE models the equations are closed by parameterizing the Reynolds stresses as:

$$\overline{u'w'} = -K_M v \frac{\partial u}{\partial z}$$

$$\overline{v'w'} = -K_M v \frac{\partial v}{\partial z}$$



Turbulence closure

In ROMS:

$$\mathcal{F}_u = \frac{\partial}{\partial z} \left(K_{Mv} \frac{\partial u}{\partial z} \right) + \nabla_h (K_{Mh} \cdot \nabla_h u)$$

$$\mathcal{F}_v = \frac{\partial}{\partial z} \left(K_{Mv} \frac{\partial v}{\partial z} \right) + \nabla_h (K_{Mh} \cdot \nabla_h v)$$

$$\mathcal{S}_T = \frac{\partial}{\partial z} \left(K_{Tv} \frac{\partial T}{\partial z} \right) + \nabla_h (K_{Th} \cdot \nabla_h T)$$

$$\mathcal{S}_S = \frac{\partial}{\partial z} \left(K_{Sv} \frac{\partial S}{\partial z} \right) + \nabla_h (K_{Sh} \cdot \nabla_h S)$$

Vertical mixing

Horizontal diffusion

Turbulence closure

Vertical Mixing:

$$K_{Mv}, K_{Tv}, K_{Sv}$$

- Local Turbulence closure: GLS, k-kl (MY2.5), k- ϵ , κ - ω , etc. [e.g. *Warner et al, 2005, Ocean Modelling*]
- Non local K-profile parameterization (KPP) [*Large et al, 1994, Rev. of Geophysics*]

Horizontal diffusion:

• Explicit diffusion

$$K_{Mh}, K_{Th}, K_{Sh}$$

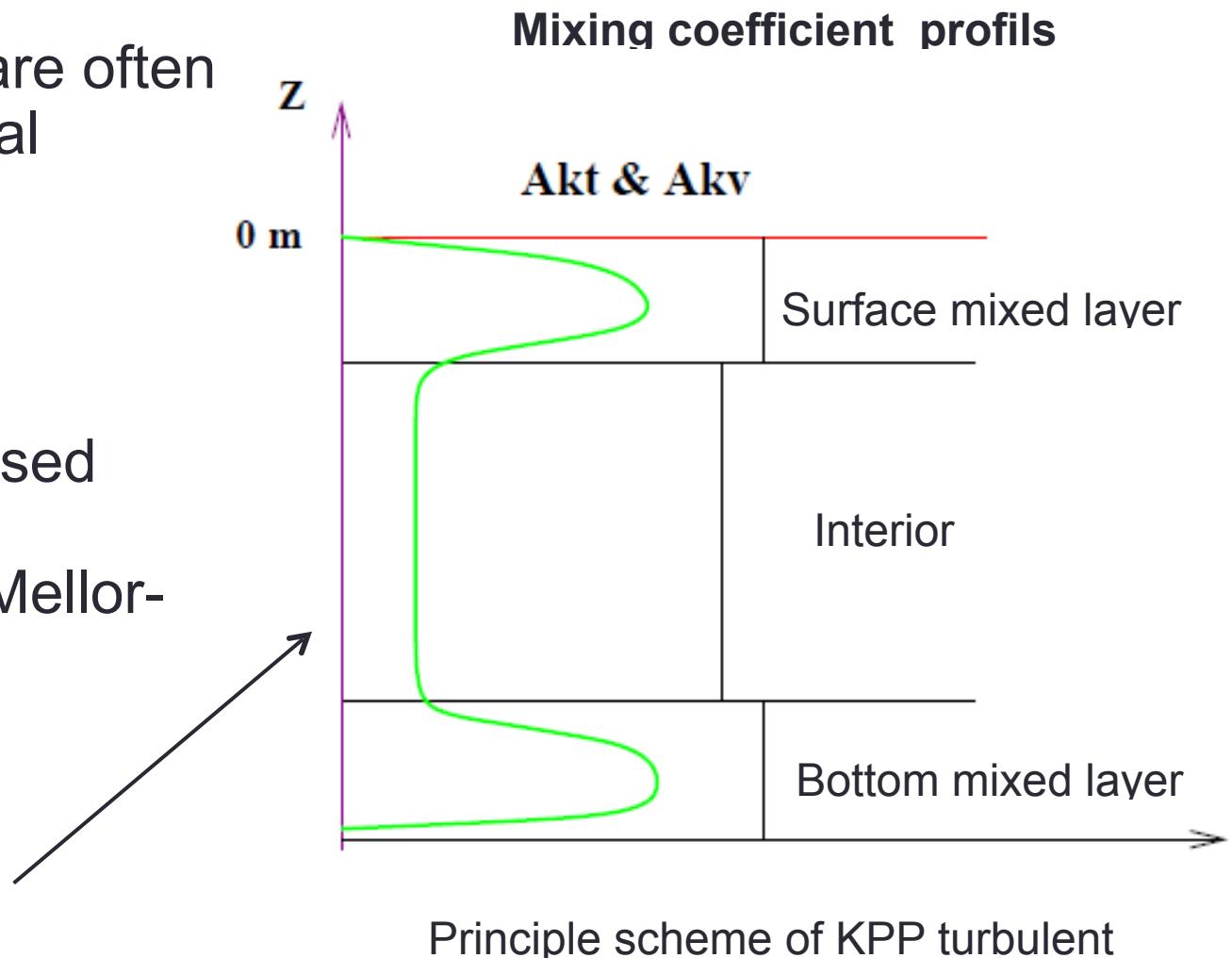
• Implicit (comes from the advective scheme)

Non local K-profile parameterization

- Mixed layer schemes are often based on one-dimensional « column physics »

- Boundary layer parameterizations are based either on:

- Turbulent closure (Mellor-Yamada, TKE)
- **K profile (KPP)**



Vertical Mixing Parameterization

[BVF_MIXING]

- Si $N^2(z) < 0$:

$$K_{m,s}(z) = 0.1 \text{ m}^2 \text{ s}^{-1}$$

- Si $N^2(z) > 0$:

$$K_{m,s}(z) = 10^{-7} / \sqrt{N^2(z)}, \quad K_{m,s}^{\min} \leq K_{m,s}(z) \leq K_{m,s}^{\max}$$

Bornes par défaut très restrictives :

$$K_{m,s}^{\min} = 3 \times 10^{-5} \text{ m}^2 \text{ s}^{-1}, \quad K_{m,s}^{\max} = 4 \times 10^{-4} \text{ m}^2 \text{ s}^{-1}$$

[ANA_VMIX]

Possibilité de spécifier analytiquement $K_{m,s}(z)$
(subroutine ana_vmix appelée à chaque pas de temps)

Vertical Mixing Parameterization

LMD_MIXING at surface

[LMD_SKPP1994, LMD_SKPP_MONOB, HBL_SMOOTH]

- ① Calcul de la profondeur de couche limite h_{bl} ($z_r \rightarrow z_N$)

$$\text{Ri}_b(z) = \frac{g(z_r - z)(\rho(z) - \rho_r)/\rho_0}{|\mathbf{u}(z) - (\mathbf{u}_h)_r|^2 + V_t^2(z)}, \quad \text{Ri}_b(-h_{bl}) = \text{Ri}_{cr}$$

- ② Dans le cas stable ($B_f > 0$) : $h_{bl} = \min(h_{bl}, h_{ek}, h_{mo})$

$$h_{ek} = 0.7u_\star/f, \quad h_{mo} = u_\star^3/(\kappa B_f).$$

- ③ Calcul des viscosités et diffusivités turbulentes

$$K_{m,s}(z) = w_{m,s} h_{bl} G(z/h_{bl}), \quad w_{m,s} = \kappa u_\star \psi_{m,s}(zB_f/u_\star^3)$$

Choix de Ri_{cr} ? $\text{Ri}_{cr} \in [0.15, 0.45]$

Vertical Mixing Parameterization

LMD_MIXING at surface

[LMD_SKPP2005, LIMIT_UNSTABLE_ONLY, HBL_SMOOTH]

- Critère pour h_{bl} : couche intégrale où la production de turbulence par cisaillement équilibre la dissipation due à la stratification

$$\text{Cr}(z) = \int_z^\zeta \mathcal{K}(z') \left\{ |\partial_{z'} \mathbf{u}_h|^2 - \frac{N^2}{\text{Ri}_{cr}} - C_{Ek} f^2 \right\} dz' + \frac{V_t^2(z)}{(\zeta - z)}, \quad \text{Cr}(-h_{bl}) = 0$$

- Consistant avec la physique originelle de KPP

$$\text{Cr}(-h_{bl}) = 0 \Rightarrow \frac{(\zeta - z) \int_z^\zeta \mathcal{K}(z') N^2(z') dz'}{(\zeta - z) \int_z^\zeta \mathcal{K}(z') \left\{ |\partial_z \mathbf{u}_h|^2 - C_{Ek} f^2 \right\} dz' + V_t^2(z)} = \text{Ri}_{cr}$$

→ cohérent avec problème d'Ekman

→ tend à produire des couches limites plus profondes

$$(\zeta - z) \int_z^\zeta |\partial_{z'} \mathbf{u}_h|^2 dz' \geq |\mathbf{u}_h(z) - \mathbf{u}_h(\zeta)|^2.$$

Vertical Mixing Parameterization

LMD_MIXING in the interior

$$K_{m,s}(z) = K_{m,s}^{\text{sh}}(z) + K_{m,s}^{\text{iw}}(z) + K_{m,s}^{\text{dd}}(z)$$

- [LMD_RIMIX, RI_(H - V)SMOOTH] (Large et al., 1994)

$$\text{Ri}_g = N^2 / [(\partial_z u)^2 + (\partial_z v)^2]$$

$$K_{m,s}^{\text{sh}}(z) = \begin{cases} K_{0,c} & \text{Ri}_g < 0 \\ K_0 [1 - (\text{Ri}_g/\text{Ri}_0)^3] & 0 < \text{Ri}_g < \text{Ri}_0 \\ 0 & \text{Ri}_0 < \text{Ri}_g \end{cases} \leftarrow [\text{LMD_CONVEC}]$$

$$K_0 = 5 \times 10^{-3} \text{ m}^2 \text{ s}^{-1}, \text{Ri}_0 = 0.7.$$

- [LMD_NUW_GARGETT] (Gargett & Holloway)

$$K_m^{\text{iw}}(z) = 10^{-6} / \sqrt{\max(N^2(z), 10^{-7})}, \quad K_s^{\text{iw}}(z) = 10^{-7} / \sqrt{\max(N^2(z), 10^{-7})}$$

- [LMD_DDMIX] (cf Large et al., 1994, eqns (31))

Vertical Mixing Parameterization

LMD_MIXING at the bottom

[LMD_BKPP, HBBL_SMOOTH]

- Même principe que pour le KPP de surface mais cette fois-ci on recherche la valeur critique Ri_{cr} (≈ 0.3) en partant du fond

$$h_{bbbl} = \min \left(h_{bbbl}, \frac{0.7 u_{\star,b}}{|f|} \right)$$

$$K_{m,s}(z) = \kappa u_{\star,b} h_{bbbl} G(\sigma), \quad \sigma = (z-h)/h_{bbbl}$$

Vertical Mixing Parameterization

[GLS_MIXING]

$$\begin{aligned}\partial_t e &= \partial_z (K_e \partial_z e) + P + B - \varepsilon, & K_e &= K_M / \text{Sc}_e \\ \partial_t \psi &= \partial_z (K_\psi \partial_z \psi) + \psi e^{-1} (\beta_1 P + \beta_3^\pm B - \beta_2 \varepsilon), & K_\psi &= K_M / \text{Sc}_\psi\end{aligned}$$

avec

$$\psi = (c_\mu^0)^p e^m l^n, \quad P = K_m [(\partial_z u)^2 + (\partial_z v)^2], \quad B = -K_s N^2$$

et

$$\varepsilon = (c_\mu^0)^{3+p/n} e^{3/2+m/n} \psi^{-1/n}, \quad l = (c_\mu^0)^3 e^{3/2} \varepsilon^{-1}$$

$$K_m = \mathbf{c}_\mu (e^2 / \varepsilon), \quad K_s = \mathbf{c}_\mu' (e^2 / \varepsilon)$$

Une instance de GLS se caractérise par :

- Un choix pour les exposants m, n et p (e.g. $k - \varepsilon$, $k - \omega$, $k - kl$, gen)
- Une valeur pour les coefficients β_1 , β_2 et β_3^\pm
- Une valeur pour les nombres de Schmidt Sc_e et Sc_ψ