

Contents

1	Routine/Function Prologues	2
1.1	Fortran: Module Interface mwmsub_mod	2
1.1.1	mwmsub_init	3
1.1.2	mwmsub_forward	6
1.1.3	mwmsub_getvar	8
1.1.4	mwmsub_getsize	11
1.1.5	mwmsub_final	11

1 Routine/Function Prologues

1.1 Fortran: Module Interface mwmsub_mod

INTERFACE:

```
module mwmsub_mod
```

DESCRIPTION:

This module contains subroutines needed for non-breaking wave mixing incorporated into circulation models. The results of bv could be added to moment diffusivity and tracer diffusivity calculated from those schemes without considering the wave effects. The information of grid/mesh including the longitude, latitude, ocean depth, depth of each level and dimension values of the mesh in the space of wave number (or frequency, or circular frequency) and wave direction will be input from initial routine. And the wave spectrum will be calculated using the same code of MASNUM wave model.

References:

* Qiao F, Yuan Y, Ezer T, et al. A three-dimensional surface wave-ocean circulation coupled model and its initial testing. Ocean Dynamics, 2010.

REVISION HISTORY:

2018/05/07 - Xunqiang Yin - Initial Version

REMARKS:

MWMSUB-2018

USES:

```
use mwmvar_mod, only: spdp,np,npc,kl,jl,im,jm,deltt,delttm,grdflag
use mwmvar_mod, only: nsp,pnb,cnb,d,alon,alat,ee,ea,wx,wy,h1_3,aet,ape,tpf,e
use mwmvar_mod, only: wamvar_mod_final,wamvar_mod_init, cnb,deg2rad,pi,rs

use mwmcor_mod, only: setwave,implsch,nlweight,setspec
use mwmcor_mod, only: check_timestep,intact_mixture_bv

use mwmpgt_mod, only: propagat,init_propagat,final_propagat
use mwmpgt_mod, only: cal_geo_distangl,setgeoinf
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: mwmsub_init           ! - Initialize the surface wave model.
public :: mwmsub_forward        ! - Forward wave model, curvilinear grid.
public :: mwmsub_getvar         ! - Get variable from this module.
public :: mwmsub_getsize        ! - Get grid size in wave number and direction.
public :: mwmsub_final          ! - Finalize the wave model.
private
```

```

!-----!
!                                     *** INTERFACES ***                                     !
!-----!

interface mwmsub_init
  module procedure &
    mwmsub_init_cuv,      &! - For normal partition of rectangle.
    mwmsub_init_blk      ! - For partition of multi-blocks.
end interface mwmsub_init

interface mwmsub_forward
  module procedure &
    mwmsub_forward_cuv,  &! - Forward wave model, partition by rectangle.
    mwmsub_forward_blk   ! - Forward wave model, partition by blocks.
end interface mwmsub_forward

interface mwmsub_getvar
  module procedure &
    mwmsub_getvar_byname_2d,&! - Get 2D var by name and return error flag.
    mwmsub_getvar_byname_3d,&! - Get 3D var by name and return error flag.
    mwmsub_getvar_byname_4d,&! - Get 4D var by name and return error flag.
    mwmsub_getvar_cuv      ,&! - Get variable from this module by a rectangle.
    mwmsub_getvar_blk      ! - Get variable from this module by blocks.
end interface mwmsub_getvar

```

1.1.1 mwmsub_init

INTERFACE:

```

subroutine mwmsub_init_cuv(tlon,tlat,mask,&! Grid locations and mask.
  depth,zlevel, &! Topography and vertical layers.
  nstep,delttime,&! Steps and time interval of winds.
  ksize,jsize,  &! Model size in physical space.
  vecdir,       &! Vector direction, default is along grid.
  wndtype,      &! Forcing type: 0-wind vector,1-windstress.
  istart,       &! Type of initial condition.
  halosize,     &! Halo size of this block.
  inrbonds,     &! Boundary index of this block.
  windx,windy   )! Initial wind vector 10m above surface.

```

INPUT PARAMETERS:

```

real(8),intent(in) :: tlon(:,:)      !(nx,ny)
! --- Longitude at each grid, in degree east (0-360).
real(8),intent(in) :: tlat(:,:)      !(nx,ny)
! --- Latitude at each grid, in degree north (-90-90).

```

```

integer,intent(in) :: mask(:,:)          !(nx,ny)
! --- Mask at each grid, the value will be 0~2.
! 0 land point.
! 1 water points, but not at open boundary
! 2 water point at open boundary.
real(8),intent(in) :: depth(:,:)        !(nx,ny)
! --- Depth at each grid, positive value in meter.
real(8),intent(in),optional :: zlevel(:)  !(kb)
! --- Depth of each level in vertical, positive value in meter.

```

INPUT/OUTPUT PARAMETERS:

```

integer,intent(inout) :: nstep
! --- Step number of wave model within the time interval of wind.
! If it is not proper, the suggested value will be returned.
! This could be used to test if this module is initiated successfully.
! If (input nstep) /= (output nstep), the initial of MWMSUB is failed.

```

INPUT PARAMETERS:

```

real(8),intent(in) :: deltime
! --- Time interval of wind forcing, in seconds.
integer,intent(in),optional :: ksize,jsize
! --- The grid size in physical space (wavenumber,wavedirection).
integer,intent(in),optional :: vecdir
! --- Type of vector direction.
! 0 Default, same direction with the grid system.
! 1 direction of normal east/north.
integer,intent(in),optional :: wndtype
! --- Forcing type/flag:
! 0 Default, wind vectors at 10m above the surface.
! 1 Windstress will be given from outside. (under develop)
integer,intent(in),optional :: istart
! --- Type of initial condition.
! 0 Default, it will be cool start.
! 1 The wave model is continue run from a restart.
integer,intent(in),optional :: halosize
! --- Size of halo region of this box.
integer,intent(in),optional :: inrbonds(4)
! --- Local index in this box to indicate the inner points.
real(8),intent(in),optional :: windx(:,:)  !(nx,ny)
real(8),intent(in),optional :: windy(:,:)  !(nx,ny)
! --- Wind vectors on the time of inquiring the mixing coefficients (bv).
! The unit is m/s and they are at 10m above the sea surface with the
! same grid with the initialization.

```

DESCRIPTION:

This routine will be used to initialize this module. The step number of the wave model within the time interval of wind forcing is required. The horizontal/vertical grid information are also inquired. The horizontal grid information includes the longitude, latitude, mask and depth in each grid. The Vertical grid information includes the depth of each layer. The grid sizes in geo-space will inquired by checking the array size. The other parameters are optional and default value will be used if they are not given.

INTERFACE:

```
subroutine mwmsub_init_blk(tlon,tlat,mask,&! Grid locations and mask.
                        depth,zlevel, &! Topography and vertical layers.
                        nstep,delttime,&! Steps and time interval of winds.
                        ksize,jsize, &! Model size in physical space.
                        vecdir,      &! Vector direction, default is along grid.
                        wndtype,     &! Forcing type: 0-wind vector,1-windstress.
                        istart,      &! Type of initial condition.
                        halosize,    &! Halo size of this block.
                        inrbonds,    &! Boundary index of this block.
                        windx,windy  )! Initial wind vector 10m above surface.
```

INPUT PARAMETERS:

```
real(8),intent(in) :: tlon(:,:,:)      ! (nx,ny,nblock)
! --- Longitude at each grid, in degree east (0-360).
real(8),intent(in) :: tlat(:,:,:)      ! (nx,ny,nblock)
! --- Latitude at each grid, in degree north (-90-90).
integer,intent(in) :: mask(:,:,:)      ! (nx,ny,nblock)
! --- Mask at each grid, the value will be 0~2.
! 0 land point.
! 1 water points, but not at open boundary
! 2 water point at open boundary.
real(8),intent(in) :: depth(:,:,:)      ! (nx,ny,nblock)
! --- Depth at each grid, positive value in meter.
real(8),intent(in),optional :: zlevel(:) ! (kb)
! --- Depth of each level in vertical.
```

INPUT/OUTPUT PARAMETERS:

```
integer,intent(inout) :: nstep
! --- Step number of wave model within the time interval of wind.
! If it is not proper, the suggested value will be returned.
! This could be used to test if this module is initiated successfully.
! If (input nstep) /= (output nstep), the initial of MWMSUB is failed.
```

INPUT PARAMETERS:

```
real(8),intent(in) :: deltime
! --- Time interval of wind forcing, in seconds.
integer,intent(in),optional :: ksize,jsize
```

```

! --- The grid size in physical space (wavenumber,wavedirection).
integer,intent(in),optional :: vecdir
! --- Type of vector direction.
! 0 Default, same direction with the grid system.
! 1 direction of normal east/north.
integer,intent(in),optional :: wndtype
! --- Forcing type/flag:
! 0 Default, wind vectors at 10m above the surface.
! 1 Windstress will be given from outside. (under develop)
integer,intent(in),optional :: istart
! --- Type of initial condition.
! 0 Default, it will be cool start.
! 1 The wave model is continue run from a restart.
integer,intent(in),optional :: halosize
! --- Size of halo region of this box.
integer,intent(in),optional :: inrbonds(4)
! --- Local index in this box to indicate the inner points.
real(8),intent(in),optional :: windx(:, :, :) !(nx,ny,nblock)
real(8),intent(in),optional :: windy(:, :, :) !(nx,ny,nblock)
! --- Wind vectors on the time of inquiring the mixing coefficients (bv).
! The unit is m/s and they are at 10m above the sea surface with the
! same grid with the initialization.

```

DESCRIPTION:

This routine will be used to initialize this module. The step number of the wave model within the time interval of wind forcing is required. The horizontal/vertical grid information are also inquired. The horizontal grid information includes the longitude, latitude, mask and depth in each grid. The Vertical grid information includes the depth of each layer. The grid sizes in geo-space will inquired by checking the array size. The other parameters are optional and default value will be used if they are not given.

1.1.2 mwmsub_forward

INTERFACE:

```

subroutine mwmsub_forward_cuv(istep,    & ! Step index between 2 wind fields.
                             spectrum,  & ! Wave energy spectrum.
                             windx,windy, & ! Wind vector 10m above surface.
                             newmask,    & ! Ice mask for each grid, 0 or 1.
                             uvel,vvel   ) ! Circulation currents, m/s.

```

INPUT PARAMETERS:

```

integer,intent(in) :: istep
! --- Index of steps within twice inquire of the mixing coefficients (bv)

```

```

!      for coupling of the wave-circulation coupling. If 0 is given for it,
!      there will be no wind interpolation along time and the wind provided
!      into this subroutine will be used directly.
real(8),intent(in) :: windx(:,,:)      !(nx,ny)
real(8),intent(in) :: windy(:,,:)      !(nx,ny)
! --- Wind vectors on the time of inquiring the mixing coefficients (bv).
!      The unit is m/s and they are at 10m above the sea surface with the
!      same grid with the initialization.
real(8),intent(in),optional :: newmask(:,)  !(nx,ny)
! --- Mask for ice coverage, 0 for land/ice grid and 1 for water grid.
real(8),intent(in),optional :: uvel(:,)     !(nx,ny)
real(8),intent(in),optional :: vvel(:,)     !(nx,ny)
! --- Background circulation (vertical averaged or at surface),
!      the unit is m/s. (under develop)

```

INPUT/OUTPUT PARAMETERS:

```

real(8),intent(inout) :: spectrum(:,,:,:)  !(nx,ny,kl,jl)
! --- Energy spectrum of surface wave. This will need to readin from restart file
!      or output for restart the model.

```

DESCRIPTION:

This routine will forward one/two step of the wave model. The wind vector will be given, but the ice mask and current velocity are optional. The vector could be agree with north/east (longitude/latitude) direction along or along with the grid lines. The option of windstrees accepted in the updated version.

INTERFACE:

```

subroutine mwmsub_forward_blk(istep,      &!Step index between 2 wind fields.
                             spectrum,    &!Wave energy spectrum.
                             windx,windy,&!Wind vector 10m above surface.
                             newmask,     &!Ice mask for each grid, 0 or 1.
                             uvel,vvel    )!Circulation currents, m/s.

```

INPUT PARAMETERS:

```

integer,intent(in) :: istep
! --- Index of steps within twice inquire of the mixing coefficients (bv)
!      for coupling of the wave-circulation coupling. If 0 is given for it,
!      there will be no wind interpolation along time and the wind provided
!      into this subroutine will be used directly.
real(8),intent(in) :: windx(:,,:,:)      !(nx,ny)
real(8),intent(in) :: windy(:,,:,:)      !(nx,ny)
! --- Wind vectors on the time of inquiring the mixing coefficients (bv).
!      The unit is m/s and they are at 10m above the sea surface with the
!      same grid with the initialization.

```

```

real(8),intent(in),optional :: newmask(:,:,:)    !(nx,ny)
! --- Mask for ice coverage, 0 for land/ice grid and 1 for water grid.
real(8),intent(in),optional :: uvel(:,:,:)       !(nx,ny)
real(8),intent(in),optional :: vvel(:,:,:)       !(nx,ny)
! --- Background circulation (vertical averaged or at surface),
!      the unit is m/s.

```

INPUT/OUTPUT PARAMETERS:

```

real(8),intent(inout) :: spectrum(:,:,:,,:)      !(nx,ny,kl,jl)
! --- Energy spectrum of surface wave. This will need to readin from restart file
!      or output for restart the model.

```

DESCRIPTION:

This routine will forward one/two step of the wave model. The wind vector will be given, but the ice mask and current velocity are optional. The vector could be agree with north/east (longitude/latitude) direction along or along with the grid lines. The option of windstrees accepted in the updatedversion.

1.1.3 mwmsub_getvar**INTERFACE:**

```

subroutine mwmsub_getvar_byname_2d(var,vname,ierr)

```

INPUT PARAMETERS:

```

character(len=*),intent(in) :: vname

```

OUTPUT PARAMETERS:

```

real(8),intent(out) :: var(:,:)
integer,intent(out) :: ierr

```

DESCRIPTION:

Get variables according to its name. For 2D variables, the name could be "hs", "th", "tp" or "tz". For 3D variables, the name could be "bv", "hs", "th", "tp" or "tz". For 4D variables, the name could be only "bv". The meaning of "bv", "hs", "th", "tp" or "tz" are "Mixing coefficients induced by non-breaking wave", "Significant wave height", "Mean wave direction", "Zero-crossing wave period", and "Spectrum peak wave period" respectively. This name is case-insensitive but must not contain blank space characters.

INTERFACE:

```

subroutine mwmsub_getvar_byname_3d(var,vname,ierr)
character(len=*),intent(in) :: vname

```

OUTPUT PARAMETERS:

```
real(8),intent(out) :: var(:,:,:)
integer,intent(out) :: ierr
```

DESCRIPTION:

Get variables according to its name. For 2D variables, the name could be "hs", "th", "tp" or "tz". For 3D variables, the name could be "bv", "hs", "th", "tp" or "tz". For 4D variables, the name could be only "bv". The meaning of "bv", "hs", "th", "tp" or "tz" are "Mixing coefficients induced by non-breaking wave", "Significant wave height", "Mean wave direction", "Zero-crossing wave period", and "Spectrum peak wave period" respectively. This name is case-insensitive but must not contain blank space characters.

INTERFACE:

```
subroutine mwmsub_getvar_byname_4d(var,vname,ierr)
```

INPUT PARAMETERS:

```
character(len=*),intent(in) :: vname
```

OUTPUT PARAMETERS:

```
real(8),intent(out) :: var(:,:,:)
integer,intent(out) :: ierr
```

DESCRIPTION:

Get variables according to its name. For 2D variables, the name could be "hs", "th", "tp" or "tz". For 3D variables, the name could be "bv", "hs", "th", "tp" or "tz". For 4D variables, the name could be only "bv". The meaning of "bv", "hs", "th", "tp" or "tz" are "Mixing coefficients induced by non-breaking wave", "Significant wave height", "Mean wave direction", "Zero-crossing wave period", and "Spectrum peak wave period" respectively. This name is case-insensitive but must not contain blank space characters.

INTERFACE:

```
subroutine mwmsub_getvar_cuv(bv,& ! Mixing coefficients induced by non-breaking wave.
                             hs,& ! Significant wave height (m)
                             th,& ! Mean wave direction (Deg)
                             tp,& ! Zero-crossing wave period (s)
                             tz ) ! Spectrum peak wave period (s)
```

OUTPUT PARAMETERS:

```
real(8),intent(out),optional :: bv(:,:,:) ! (nx,ny,kb)
! ---- The mixing coefficients (bv) for coupling of the wave-circulation coupling
```

```

real(8),intent(out),optional :: hs(:,,:)  !(nx,ny)
! --- Significant wave height (m)
real(8),intent(out),optional :: th(:,,:)  !(nx,ny)
! --- Mean wave direction (Deg)
real(8),intent(out),optional :: tp(:,,:)  !(nx,ny)
! --- Spectrum peak wave period (s)
real(8),intent(out),optional :: tz(:,,:)  !(nx,ny)
! --- Zero-crossing wave period (s)

```

DESCRIPTION:

Get variables for the coupling of wave and circulation, or get some variables of the wave model for output.

INTERFACE:

```

subroutine mwmsub_getvar_blk(nblock,& ! Block size.
                           bv,& ! Non-breaking wave induced Mixing coefficients.
                           hs,& ! Significant wave height (m)
                           th,& ! Mean wave direction (Deg)
                           tp,& ! Zero-crossing wave period (s)
                           tz ) ! Spectrum peak wave period (s)

```

INPUT PARAMETERS:

```

integer,intent(in) :: nblock
! --- Block size.

```

OUTPUT PARAMETERS:

```

real(8),intent(out),optional :: bv(:,::,,:) !(nx,ny,kb,nblock)
! ---- The mixing coefficients induced by nonebreaking wave. (m2/s)
real(8),intent(out),optional :: hs(:,::,,:) !(nx,ny,nblock)
! --- Significant wave height (m)
real(8),intent(out),optional :: th(:,::,,:) !(nx,ny,nblock)
! --- Mean wave direction (Deg)
real(8),intent(out),optional :: tp(:,::,,:) !(nx,ny,nblock)
! --- Spectrum peak wave period (s)
real(8),intent(out),optional :: tz(:,::,,:) !(nx,ny,nblock)
! --- Zero-crossing wave period (s)

```

DESCRIPTION:

Get variables for the coupling of wave and circulation, or get some variables of the wave model for output.

1.1.4 mwmsub_getsize

INTERFACE:

```
subroutine mwmsub_getsize(ksize,jsize)
```

OUTPUT PARAMETERS:

```
integer,intent(out) :: ksize  ! - grid size in dimension of wavenumber.  
integer,intent(out) :: jsize  ! - grid size in diemnsion of wave direction.
```

DESCRIPTION:

Get the model size in physical space with the dimensions of wavenumber and wave direction.

1.1.5 mwmsub_final

INTERFACE:

```
subroutine mwmsub_final
```

DESCRIPTION:

This routine will release the memory occupied by this wave model.