```
!! ------------------
!! Namelist for SOSIE
!! ------------------
!!
!!
!!
!!*****************************************************************
**********************************
!! &ndom_src => info about source domain (the domain on which the
field we want to interpolate is given)
!!
!!*****************************************************************
**********************************
!!
!! csource  : short string to describe the origin grid
[char]
!!
!! ivect : vector correction control on treated field [integer]
!!         ivect = 0 : source field is not a component of a vector
!!         or the target grid is regular (l_reg_trg = .true.)
!!         * if non-regular distorted target grids (like ORCAX):
!!         ivect = 1 : source field is a zonal (X) component of a
vector
!!         ivect = 2 : source field is a meridional (Y) component of
a vector
!!
!! l_reg_src : is the source grid regular? [logical]
!!          (ie : are source longitude and latitude 1D?)
!!
!! cf_src   : file containing the source field to be interpolated
[char]
!! cv_src   : name of treated variable (in source field file) [char]
!!
!! cv_t_src : name of time record variable in the source file [char]
!!           or 'missing_rec' if no time record is present in the
source file
!!
!! cf_x_src   : file containing the source grid (usually = cf_src)
[char]
!! cv_lon_src : name of longitude in the source grid file [char]
!! cv_lat_src : name of latitude in the source grid file [char]
!!
!! cf_lsm_src : (only relevant if idrown>0)
!!              file containing the source land-sea mask [char]
!!              Alternatively:
!!              * specify " cf_lsm_src = 'missing_value' " if a
'missing_value' netcdf
!!                attribute defines the mask on the source data field
!!              * specify " cf_lsm_src = 'nan' " if mask is defined
with NaN values
!!              * specify " cf_lsm_src = 'value' if you want land
regions to be defined
!!                where field 'cv_src' is strictly equal to the
numeric value read into 'cv_lsm_src'
```

```
!!             * specify " cf_lsm_src = 'val+' if you want land
regions to be defined
!!               where field 'cv_src' is larger or equal to the
numeric value read into 'cv_lsm_src'
!!             * specify " cf_lsm_src = 'val-' if you want land
regions to be defined
!!               where field 'cv_src' is smaller or equal to the
numeric value read into 'cv_lsm_src'
!!               Ex: you want all points where your field is <= 0 to
become land mask,
!!                   then specify: cf_lsm_src = 'val-' and
cv_lsm_src = '0.00001'
!!
!! cv_lsm_src : (only relevant if idrown>0)
!!             name of land-sea mask variable [char]
!!             or if cf_lsm_src = 'missing_value'--> '')
!!             by default ocean is flagged with value 1
!!             and continents are flagged with value 0
!!             Alternatively:
!!             a string of numeric value when cf_lsm_src is 'value',
'val-', or 'val+'
!!
!! ewper : east-west periodicity on the source file/grid [integer]
!!         = -1 --> no periodicity
!!         >= 0 --> periodicity with overlap of ewper points
!!
!! Only required if 3D interpolation:
!!
!! cf_z_src  : file containing the source depth vector (associates a
depth to a
!!            given level). In most cases should be the same file
than cf_x_src.
!! cv_z_src  : name of the variable for the source depth vector
!!
!! ctype_z_src  : type of coordinates in source file (currently
available z/sigma)
!!
!! These are to be set ONLY if ctype_z_src = 'sigma'
!! cf_bathy_src : file containing the bathymetry on source grid
(usually ROMS grid file)
!! cv_bathy_src : name of bathymetry variable (usually h)
!! ssig_src     : structure with ROMS s-coordinates parameters on
source grid
!!------------------------------------------------------------------
--------
!!
&ndom_src
csource    = 'regular'
ivect      = 0
l_reg_src  = .true.
cf_src     = '/home/sofiad/SWR_MARCH_2D.nc'
cv_src     = 'msnswrf'
cv_t_src   = 'time'
cf_x_src   = '/home/sofiad/SWR_MARCH_2D.nc'
```

```
cv_lon_src = 'longitude'
cv_lat_src = 'latitude'
cf_lsm_src = 'missing_value'
cv_lsm_src = ''
ewper_src  = 0
!!
!! Only required if 3D interpolation (jplev==0):
!!cf_z_src  = '/data/toto/my_file.nc'
!!cv_z_src  = 'level'
!! ROMS s-coordinates stuff only:
!!ctype_z_src = 'sigma'
!!cf_bathy_src = ''
!!cv_bathy_src = ''
!!   ROMS s-coordinates parameters (if ctype_z_src or/and
ctype_z_trg = 'sigma' )
!!   Vtransform | Vstretching | Nlevels | theta_s | theta_b | Tcline
| hmin
!!ssig_src =  2,         4,         40,       7.,       2.,
250. ,   15.
/
!!
!!
!!
!!
**********************************************************************
**
!! &ndom_trg => info about target domain (the domain we interpolate
to!)
!!
**********************************************************************
**
!!
!! ctarget  : short string to describe the target grid
[char]
!!
!! l_reg_trg : is the target grid regular ? [logical]
!!           (ie : are target longitude and latitude 1D?)
!!
!! cf_x_trg   : file containing the target grid [char]
!! cv_lon_trg : name of longitude variable [char]
!! cv_lat_trg : name of latitude variable [char]
!!
!! TRICK:  for interpolating onto a global regular spherical grid
!! ------  with a resolution of dx deg. of longitude and dy deg. of
latitude
!!         * cf_x_trg   = 'spheric' ! tells SOSIE to build a
spherical target grid
!!         * cv_lon_trg = '1.0'  ! your dx, here 1.0 deg.
!!         * cv_lat_trg = '1.0'  ! your dy, here 1.0 deg.
!!
!!
!! cf_lsm_trg : (not needed if "lmout = .FALSE." --> '')
!!              file containing TARGET land-sea mask [char]
!!              land-sea mask MUST BE 3D for 3D interpolation!
```

```
!!              - or specify 'missing_value' if you want to "scan"
field "cv_lsm_trg"
!!                into file "cf_x_trg" after values that seem to
define a masked region!
!!
!! cv_lsm_trg : (not needed if "lmout = .FALSE." --> '')
!!              name of land-sea mask variable in
'cf_lsm_trg'     [char]
!!              - or name of field 'X' in 'cf_x_trg' that seem to
define a masked
!!                region (based on NaN/missing value) IF you
specified
!!                cf_lsm_trg = 'missing_value'
!!
!! ewper_trg : east-west periodicity on the target file/grid
[integer]
!!             = -1 --> no periodicity
!!             >= 0 --> periodicity with overlap of ewper points
!!
!!
!! Only required if 3D interpolation (jplev==0):
!!
!!    cf_z_trg : file containing the target depth vector (associates
a depth to a
!!               given level). In most cases should be the same file
than cf_x_src.
!!    cv_z_trg : name of the variable for the target depth vector in
file 'cf_z_trg'
!!    ctype_z_trg : type of coordinates in target file (currently
available z/sigma)
!!
!!    ROMS s-coordinates stuff only:
!!    These are to be set ONLY if ctype_z_trg = 'sigma'
!!    cf_bathy_trg : file containing the bathymetry on target grid
(usually ROMS grid file)
!!    cv_bathy_trg : name of bathymetry variable (usually h)
!!    ssig_trg     : structure with ROMS s-coordinates parameters on
target grid (see above)
!!------------------------------------------------------------------
--------
!!------------------------------------------------------------------
--------
&ndom_trg
ctarget    = 'regular'
l_reg_trg  = .true.
cf_x_trg   = '/home/sofiad/CMEMS_thetao_depth_0_2017-03.nc'
cv_lon_trg = 'longitude'
cv_lat_trg = 'latitude'
cf_lsm_trg = '/home/sofiad/mask_CMEMS_2D_FIN.nc'
cv_lsm_trg = 'thetao'
ewper_trg  = 2
!!
!! Only required if 3D interpolation (jplev==0):
!!cf_z_trg = '/data/toto/mesh_zgr.nc'
```

```
!!cv_z_trg = 'nav_lev'
!!ctype_z_trg = 'z'
!!
!!cf_bathy_trg = '/data/toto/roms_grid.nc'
!!cv_bathy_trg = 'h'
!! ROMS s-coordinates parameters (if ctype_z_src or/and ctype_z_trg
= 'sigma' )
!! Vtransform | Vstretching | Nlevels | theta_s | theta_b | Tcline |
hmin
!!ssig_trg = 2,         4,         40,       7.,       2.,
250. ,   15.
/
!!
!!
!! ******************************************************************
!! &ninterp => stuff related to interpolation and pre/post
processing
!!             of data
!! ******************************************************************
!!
!! cmethod  : the 2D interpolation method to be used
!!
!!            * use 'akima' if your source domain is regular (non-
distorted grid)
!!
!!            * use 'bilin' otherwise (bilinear 2D interpolation)
!!
!!            * use 'no_xy' to only perform vertical interpolation,
i.e. interpolate a
!!              a 3D field given on ni*nj and nk_src levels to the
same ni*nj 2D domain
!!              but on nk_trg levels!
!!              => for example interpolates a 3D field from grid
ORCAX.L46 to ORCAX.L75
!!
!! idrown : Three values in this structure: np_drown, nt_smooth,
l_msk_chg ([integer],[integer],[boolean]):
!!          * np_drown : whether we call DROWN land filling
procedure (>0) or not (=0)
!!            which propagates/extrapolates sea values (defined
where lsm==1) of field
!!            cv_src by "np_drown" grid points over land regions
(defined where lsm==0)
!!          * nt_smooth : the number of time smoothing is applied on
drowned regions
!!            (only performed on land regions)
!!          * l_msk_chg : wether the missing/masked points region
(land) on source field is changing with time!
!!          => this treatment has absolutely no impact over sea
regions, only over land
!!          regions, it is intended to prevent land values to
polute/contaminate sea
!!          regions during the interpolation process!
!!
```

```
!! l_save_drwn : save the input field that has been drowned (idrown
must be > 0!)
!!
!! ismooth : if ismooth > 0 the field to be interpolated will be
smoothed
!!           prior to interpolation. By applying ismooth times a
type of
!!           closest neighboors boxcar smoothing algorithm
!!           (check "SMOOTH" of mod_drown.f90)
!!         => this is usefull to avoid sub-sampling when your target
!!            grid is much coarser than your source grid
!!            (i.e. when interpolating from high-res to low-res)
!!         => start with a multiple of 10, typically 20, and adjust
depending
!!            on the result
!!
!! jt1     : first time record to be interpolated
!! jt2     : last  time record to be interpolated
!!           => set jt1 and jt2 to 0 if you want to skip this option
!!              and interpolate the nt time steps of the current
field
!!
!! jplev : level to treat if your file is 3D (spatial), has no
influence if
!!         your file is 2D in space !
!!
-----------------------------------------------------------------
!!
-----------------------------------------------------------------
!!        jplev > 0 = level to treat (ex : jplev = 1 will
interpolate only
!!                    surface field corresponding to the 1st level )
!!
-----------------------------------------------------------------
!!        jplev = 0 : perform 3D interpolation (if source file is
3D) !!! |
!!
-----------------------------------------------------------------
!!        jplev = -1 : overrides good sense and forces sosie to
understand that
!!                     your field to interpolate is 2D with a time
record
!!                     (usually the case if the time record
dimension in your
!!                      source file is not declared as UNLIMITED =>
bad! )
!!                     => so SOSIE doesn't mistake this time record
with a depth!
!!
-----------------------------------------------------------------
!!
!! vmax    : upper bound not to exceed for treated variable [real]
!! vmin    : lower bound not to exceed for treated variable [real]
!!
```

```
!! ismooth_out : smooth the freshly interpolated field on the target
grid
!!               ismooth_out times! (see ismooth above)
!!
&ninterp
cmethod     = 'bilin'
!!
idrown      = 100,50,.false.
l_save_drwn = .false.
ismooth     = 0
jt1         = 0
jt2         = 0
jplev       = 1
vmax        =  1.E6
vmin        = -1.E6
ismooth_out = 0
ibx_xtra_sm = 0, 0,0, 0,0  ! Extra-smoothing on a given rectangular
region: ibx_xtra_sm = ntimes, i1,j1, i2,j2
/
!!
!!
!! *********************************************************************
!! &noutput => info on the (horizontal) interpolation method to use
!!             and the netcdf file to generate
!! *********************************************************************
!!
!! This mostly deals with how the output file to be created is going
to look like!
!!
!!
!! *** Into the netcdf file to be created : ***
!! cv_out   : name for treated variable in the output file
[char]
!! cu_out   : if not = '': then change the unit of treated variable
units [char]
!! cln_out  : if not = '': then change the long name treated
variable     [char]
!! cv_t_out : name of time record vector in the output file [char]
!!            => set to cv_t_out='' if no time dimension
!! cd_out   : directory to create output file to
[char]
!!
!! *** Naming of the output file : ***
!! cextra   : short extra indication about the file
[char]
!!
!! lmout : whether to mask the interpolated field on the target file
[logical]
!!         if lmout is set to .FALSE. and cf_lsm_trg is different
than '' the target
!!         field will be drowned using the mask defined by
cf_lsm_trg (and cv_lsm_trg)
!!
!! rmiss_val : missing value given to target field (for continents)
```

```
[real]
!!
!! lct   : whether to control or not time variable [logical]
!!         TRUE -> specify time array with starting time 't0' and
step 't_stp'
!!                 usefull if you do not have a "time" variable in
your source netcdf file !
!!         FALSE -> same time array as in source file is used
!! t0    : time to start (if lct is set to .TRUE.) [real]
!! t_stp : time step (if lct is set to .TRUE.) [real]
!!
!! Only required if 3D interpolation (jplev==0):
!!    cv_z_out: name you wish to give to depth variable in file to
be created...
!!
!!
&noutput
cv_out    = 'temp'
cu_out    = ''
cln_out   = ''
cv_t_out  = 'time_counter'
cd_out    = '.'
cextra    = 'monthly_1998'
lmout     = .true.
rmiss_val = -9999.
lct       = .false.
t0        = 0.
t_stp     = 0.
!! Only required if 3D interpolation (jplev==0):
cv_z_out  = 'depth'
/
!!
```