Adding Variables to ROMS

Kate Hedstrom

January 2011

Considerations

- **Input/initialization**
- **Output?**
- **Is it gridded or not?**
- **Where does it logically belong?**

Four Examples

- **Scalar variable**
- **A 3-D tracer**
- **A static 2-D gridded field**
- **A time-dependent 2-D gridded field**

Simplest Case

- **A scalar (not gridded), common to all processes**
- **No need for I/O**
- **In ROMS/Modules/mod_scalars.F:**

```
#ifdef ICE_MODEL
      logical, dimension(Ngrids) ::  Lice
#endif
#ifdef ICE_MODEL
      Lice(ng)=.TRUE.
#endif
```

Where?

- **Lice is a logical switch, similar to logicals Lbiology, Lfloats, and Lsediment**
- **Put it with the others, but keep the #ifdefs around so you can find the ICE_MODEL changes later**

Second Case

- The ecosystem models have NBT tracers
- If using NEMURO, the value of NBT comes from nemuro_mod.h, included by mod_biology.F
- The biology files are all under ROMS/Nonlinear/Biology

NEMURO Iron

- **Jerome Fiechter added iron to NEMURO, using three new tracers**

```
#  ifdef IRON_LIMIT
      NBT = 14
#  else
      NBT = 11
#  endif
```

New Tracer ID (nemuro_mod.h)

```
!  Set biological tracer indices.
      integer :: iopal
# ifdef IRON_LIMIT
      integer :: iFeSp    ! Small phyt Fe
      integer :: iFeLp    ! Large phyt Fe
      integer :: iFeD_    ! Dissolved Fe
# endif
```

Give them Values

```
      iopal=ic+11
# ifdef IRON_LIMIT
      iFeSp=ic+12
      iFeLp=ic+13
      iFeD_=ic+14
# endif
```

Other Iron Changes

- nemuro_mod.h – set up some rate constants
- nemuro_inp.h – read in rate constants
- Add the rate constants to a copy of nemuro.in called nemuro_iron.in
- Add code to nemuro.h to use the rate constants and iron fields

Iron I/O Changes

- nemuro_def.h – NetCDF definitions for writing iron vars (see the opal code)
- nemuro_wrt.h – output of iron vars
- nemuro_var.h – check for idTvar (iFeSp) and friends when reading varinfo.dat
- Add them to varinfo.dat
- Add iron fields to boundary and initial condition files (or ana_initial.h)

Finally…

- Adding passive tracers is even simpler, changing NPT in ocean.in
- To output them, change Hout(inert) in ocean.in
- To provide initial and boundary conditions, call the fields dye_01, dye_west_01, etc.

Third Case

- **Spatially variable bottom friction**
- **2-D field to be read in from the grid file, treated like the other grid variables**
- **I tried to add it live for a ROMS training in Maryland – and failed**

Label Your Code

- **Start by picking a cppdef: RDRG_GRID, also ANA_RDRG in the case of defining it analytically**
- **More later on the ANA_ business**
- **Add both RDRG_GRID and ANA_RDRG to checkdefs.F and cppdefs.h**

Which Module?

- **I put it in mod_forces.F, with the other bottom drag variables:**

```
#ifdef RDRG_GRID
      real(r8), pointer :: rdrg_grid(:,:)
#endif
#ifdef RDRG_GRID
      allocate(FORCES(ng)%rdrg_grid(LBi:UBi,LBj:UBj) )
#endif
# ifdef RDRG_GRID
      FORCES(ng) % rdrg_grid(i,j) = IniVal
# endif
```

mod_forces.F

- **The three chunks go in three different parts of mod_forces**
  - Define the data structure
  - Allocate the gridded variables
  - Initialize the gridded variables
- **If adding a gridded variable, make sure to add it to a module with other gridded variables**

Input via get_grid.F

- **Because I put the variable in mod_forces, we need to "use" it:**

```
#ifdef RDRG_GRID
      USE mod_forces
#endif
```

- **We can check to see if rdrg_grid is in the file, then read it, just like the 'h' variable, including communication**
- **Doesn't need to be in varinfo.dat**

Analytic function

- **You might want to read the grid file first, then call ana_rdrg instead of reading the field**
- **In initial.F, after the get_grid block:**

```
#ifdef ANA_RDRG
!$OMP PARALLEL DO PRIVATE(thread,subs,tile)
SHARED(ng,numthreads)
      DO thread=0,numthreads-1
        subs=NtileX(ng)*NtileE(ng)/numthreads
        DO tile=subs*thread,subs*(thread+1)-1
          CALL ana_rdrg (ng, TILE, iNLM)
        END DO
      END DO
!$OMP END PARALLEL DO
#endif
```

More ANA_RDRG

- **Add to analytical.F:**

```
# if defined ANA_RDRG
#  include <ana_rdrg.h>
# endif
```

- **Copy ana_mask.h to create ana_rdrg.h then edit it**

ana_rdrg.h

- **Add “USE mod_forces”**
- **Put “FORCES(ng) % rdrg_grid” in argument list to ana_rdrg_tile, also variables it depends on**
- **Strip out “mask” scratch array and all the umask, vmask stuff**

Passing to Tile Routine

```
CALL ana_rdrg_tile (ng, tile, model,    &
&                   LBi, UBi, LBj, UBj,          &
&                   IminS, ImaxS, JminS, JmaxS,  &
&                   FORCES(ng) % rdrg_grid,      &
&                   GRID(ng) % h)
```

- **Change where called and declaration:**

```
real(r8), intent(out) :: rdrg_grid(LBi:,LBj:)
real(r8), intent(out) :: h(LBi:,LBj:)
```

Set rdrg_grid

```
#ifdef NEP5
      DO j=JstrR,JendR
        DO i=IstrR,IendR
          rdrg_grid(i,j)=…
        END DO
      END DO
#else
      ana_rdrg.h: no values provided
for rdrg_grid.
#endif
```

Communication

```
#if defined EW_PERIODIC || defined NS_PERIODIC
      CALL exchange_r2d_tile (ng, tile,         &
     &                        LBi, UBi, LBj, UBj,   &
     &                        rdrg_grid)
#endif
#ifdef DISTRIBUTE
      CALL mp_exchange2d (ng, tile, model, 1, &
     &                    LBi, UBi, LBj, UBj,           &
     &                    NghostPoints, EWperiodic,   &
     &                    NSperiodic, rdrg_grid)
#endif
```

ANANAME

- **Ana_mask.h has:**

```
ANANAME(15)=__FILE__
```

- **ANANAME is defined in mod_ncparam.F:**

```
character (len=256), dimension(37) ::
ANANAME
```

- **Then used in def_info.F and close_io.F**
- **“Magic number” of 37 in all three places**

More ANANAME

- **We can change all three values to 38 and set it in ana_rdrg as:**

```
ANANAME(38)=__FILE__
```

- **Then next week/year Hernan adds ana_jellyfish and takes over 38**
- **There's got to be a better way…**

Using rdrg_grid

- **Used in set_vbc.F**
- **Need to pass it to set_vbc_tile routine – copy how bustr is passed**
- **Search on rdrg and add option:**

```
#ifdef RDRG_GRID
      cff1=0.5_r8*(rdrg_grid(i,j)                &
&                  +rdrg_grid(i-1,j))
#else
      cff1=rdrg2(ng)
#endif
```

Output

- It could be written to all output files along with the other grid variables unless NO_WRITE_GRID
- Put in def_info.F and wrt_info.F
- Still don't need to add to varinfo.dat

Fourth Case

- **Bio sediment variables for putting ammonia back into the water from falling detritus**
- **Code used in NEMURO for Bering Sea (from Enrique Curchitser)**
- **Time dependent – want to store averages, write to stations, etc.**

NEMURO_SED1

- We also tried a NEMURO_SED2, but it behaved badly
- Create two new variables in mod_ocean.F, PONsed and OPALsed
- 2-D so declare, allocate, initialize just like Hsbl
- This time we need to add it to varinfo.dat

Averages

- **Add avgPONsed and avgOPALsed to mod_averages, in three places as usual**
- **The allocate and initialize are protected by tests on**
  - Aout(idPONsed, ng)
  - Aout(idOPALsed)
- **Aout is array of average output choices**

Aout and Sout

- **To properly set Aout(idPONsed), we need to add code to inp_par, matching code for Aout(idHsbl), Sout(idHsbl)**

Set_avg.F

- **Again, mimic the code for Hsbl to accumulate PONsed values into avgPONsed and avgOPALsed**
  - Initialize at start of averaging duration
  - Add values until end of averaging duration
  - Scale by number of timesteps at end of averaging duration

I/O

- Add code to def_avg.F and wrt_avg.F to get averages output
- Add code to def_rst.F, wrt_rst.F and get_state.F to read it on restart
- Add code to def_station.F and wrt_station.F
- Add code to def_his.F and wrt_his.F – but we neglected to check for Hout (idPONsed) in inp_par.F
- We also left it out of ana_biology.h

Last bits

- **Don't forget to actually use and timestep your new variables!**
- **We need to add idOPALsed and idPONsed to mod_ncparam.F:**

```
integer   :: idOPALsed      ! opal in sediment
integer   :: idPONsed       ! PON in sediment
```

- **Files nemuro.diffs and drag.diffs are at:**

`http://www.arsc.edu/~kate/ROMS/HK`