Dynamic Memory in NEMO

Implications for developers

Andy Porter,
STFC Daresbury Laboratory,
UK

Overview

1. What's changed
2. Implications for coding a module
3. Implications for coding a subroutine
4. The wrk_nemo module

1. What's Changed

par_oce.F90: *jpi*, *jpj* and *jpk* are now variables

Module arrays that were previously declared with e.g.

DIMENSION(jpi,jpj,jpk)

are now declared with

SAVE, ALLOCATABLE, DIMENSION(:,:,:)

and are ALLOCATED in a new ..._alloc() function that belongs to the module.

1. What's Changed continued...

ALL of these module ..._alloc() functions are called from nemo_alloc() in nemogcm.F90 once jpi and jpj are known.

If any of these calls fail then the program is halted.

2. Implications for coding a module

Any array involving *jpi*, *jpj* or *jpk* in one of its extents must be allocatable.

Any module containing one or more allocatable arrays must CONTAIN a PUBLIC ..._alloc() function that allocates them.

This function must be called from within nemo_alloc().

3. Implications for coding a subroutine

- All dynamic module arrays are now SAVE'd so be aware of what an array contains before it is used
- Work-space arrays must now be declared differently
  - If extent of array <= (*jpi*,*jpj*,*jpk*) then use new **wrk_nemo** module
  - Otherwise, use a module variable and ensure it is allocated in the ..._alloc() function

4. MODULE wrk_nemo

- Contains arrays for use as temporary workspaces in subroutines
- REAL(wp), INTEGER and LOGICAL.
- 1, 2, 3 and 4 dimensions (jpi, jpj, jpk and jpts)
- Removes need to ALLOCATE workspace arrays in each subroutine
- Should reduce overall memory usage
- Arrays accessed using standard module 'use' syntax:
  - USE wrk_nemo, ONLY: zwrk => wrk_3d_1
- Module contains helper routines that ensure requested workspaces not already in use
  - wrk_use(ndims, array_id_1, array_id_2,...)
  - wrk_release(ndims, array_id_1, array_id_2,...)

4. Using the **wrk_nemo** module

```
 USE wrk_nemo, ONLY: wrk_use, wrk_release
USE wrk_nemo, ONLY: ztemp1 => wrk_2d_1
USE wrk_nemo, ONLY: ztemp2 => wrk_3d_2

IF( (.NOT. wrk_use(2, 1)) .OR. &
    (.NOT. wrk_use(3, 2)) )THEN
   CALL ctl_stop('sbc_fwb: workspace arrays in use.')
   RETURN
END IF

ztemp1(:,:) = .....

IF( (.NOT. wrk_release(2, 1)) .OR. &
    (.NOT. wrk_release(3, 2)) )THEN
   CALL ctl_stop('sbc_fwb: failed to release..')
END IF
```

4. Workspaces with extent < (jpi,jpj,jpk)

1. Either, use explicit index range throughout
   (e.g. zwrk(1:n,:,:) where n < jpi)
2. or, use a pointer to access a sub-array of the workspace:

```
USE wrk_nemo, ONLY: wrk_3d_4

REAL(wp), POINTER, DIMENSION(:,:,:) :: zwrk
INTEGER, PARAMETER :: n = 4
!----------------------------------
...
zwrk => wrk_3d_4(1:n,:,:)
```

and can then use zwrk(:,:,:) as usual.

Comments/discussion...