Towards a generic zooplankton IBM module in NORWECOM.E2E

Morten D. Skogen & Geir Huse

MEECE Copenhagen, 24-26/1-2011
NORWECOM 1992
2P + 3N +D
(Aksnes, Ulvestad, Berntsen, Skogen)

NORWECOM 200X
2P + 3N +3D +O
(Skogen, Søiland)

(oxygen, sis, detP, sedimentation,++)

Recoding, structuring, new modules

NORWECOM 2010

Micro/meso-zoo

Contaminants

Calanus IBM

FishIBM

Online (POM North Sea)
(without IBMs)

Offline ROMS

Full Offline

NORWECOM.E2E
IBM Calanus module

Feeding: functional response, type 2 (Campbell 2001)

Growth: bioenergetics (Carlotti & Wolf 1998)

Mortality: predation, starvation, spawning

Reproduction: mature adults above weight and fat thres.

Vertical movement: ontogeny, dvm

Horizontal movement: by currents
Object oriented approach

*Object oriented programming* - is a programming paradigm that uses “objects”- data structures consisting of data fields and methods together with their interactions - to design applications and computer programs

1) define a datatype with all necessary information about a calanus (or fish or krill) superindividual and use it inside a data type that contains all information about a stock/species…..

and

2) define routines that operates on these data types
module zoo_generic

! Type with attributes and strategy vector ++
type superind
  logical :: alive ! True if alive, false if dead
  integer :: stage ! 0 egg, 1-6 nauplii, 7-13 copepod
  integer :: inumb ! Number of individuals in superind.
  real :: x, y, z ! Position (x, y, z)
......
end type superind

! Type for a whole population. Superind+ population specific parameters
type zoopop
  character(len=6) :: species ! Name of species
  integer :: pop ! Pointer to last super ind.

......population specific parameters...........

  type(superind), dimension(:), allocatable :: zoopl
end type zoopop
type zooplankton

! Attribute vector

  logical :: alive          ! true if alive, false if dead
  integer :: stage        ! 0 is egg, 1-6 nauplia, 7-11 copepods, 12 adult
  real    :: inumb        ! Internal number in individuals
  real    :: sweight      ! Structural weight in micrograms
  real    :: lstage       ! Stage longevity
  real    :: fat           ! Fat energy level in KJ
  real    :: moult        ! Moult cycle fraction (Egg,N2,N3) | cum. egg number (adult)
  real    :: maxegg       ! Total number of eggs in a super individual
  real    :: xpos,ypos,zpos ! Position (x,y,z)
  integer :: diapause    ! 0 diapause, 1 active, 2 move down, 3 move up
  real    :: growth       ! Summing of daily growth
  real    :: egestion     ! Summing of hourly egestion
  integer :: mynumb      ! Unique identifier
  real    :: ingestion    ! Ingestion rate
  real    :: grate        ! Growth rate
  integer :: numegg       ! Number of spawned eggs
  real    :: inumb_int    ! Internal number in individuals at birth

! Strategy vector

  integer :: wud            ! Wake_Up_Day
  real    :: fsr            ! FSR
  integer :: afd            ! Allocation_to_Fat_Day
  integer :: owd            ! Over_Wintering_Depth
  integer :: vm1
  real    :: vm2

! Predation risk

  real    :: prisk(4)       ! Predation risk. Species dependent. Level set in CALCPRISK
Population specific parameters *C. finmarchicus*:

```plaintext
real, dimension(0:13) :: st_wgt ! Stage weight
real, dimension(0:13) :: mort ! Mortality rates
real, dimension(0:13) :: a ! Stage longevity parameter
real :: stloli ! Lower limit stage weight factor before starvation
real :: strate ! Starvation rate
integer :: maxmoult ! Maximum number of eggs to be spawned
integer :: maxlstage ! Maximum number of days in each stage
integer :: mthresh ! Mortality threshold, predation or MonteCarlo
real :: cmat ! Structural weight for spawning
real :: eggw ! Egg weight
real :: cs ! Fat/eggw ratio for spawning
real :: initw ! Initial weight at spawning
real :: starvation ! Rate of starvation at low stage_weight
real :: meffect ! Mutation effect
real :: mprob ! Mutation probability
```

Fortran 90 modules:

- **zoo_generic**: typedefinitions and subroutines that operates on these (and are assumed general)
  - kill_zoopl, init_zoopl, copy_zoopl, spawn_zoopl, count_zoopl, popzip, phytomort, calmort, calrep, vertmov, combind
- **tools** (bad name..): useful routines that does not operate on the zooplankton types
  - val3d, getr, easyr, deriv, surlig
- **ladim**: Lagrangian Advection DIffusion Model (Ådlandsvik)

- **calanus_mod** (one module for each species): uses the other modules and (until now) the remaining species specific routines
  - calanus (**mainprogram** should be *identical* for all zooplankton)
    - calsta, initialise, calcprisk
First test - implementing a Krill module (Strand + Huse, IMR)......

• swimming behaviour (to be implemented in ladim)
• new/additional strategies:
  • night-depth
  • meters above bottom
  • first and last day for spawning
• additional parameters:
  • maxmig = max daily vertical migration
  • first copepod stage (krill=5, calfin=7)
  • stage for spawning (krill=12, calfin=13)
  • a few other minor details...

• conclusion: can easily adapt the generic module, perhaps with a few less elegant: if(zoopls%species.eq.'krill')then
Plan to extend model with:

• calanus helgolandicus, glacialis, hyperboreus
• generic fish-module

Remaining issues:

• general coupling between different IBMs (today only two-way hardcoded coupling between phyto and calfin IBM, and coupling between IBM fish model and IBM calfin implemented)
• order of predation - who to eat first - both between superindividuals and between stocks/species
• ........
Thank you for your attention

Bergen seen from Mt. Ulriken