GeoSAMS Growth

Thomas Callaghan
Version 0.1

Contents

1 Scallop Population Density

This program is used to compute Scallop Density after a given growth period The Growth year starts on June 1st, actually May 31 at 2400

Jun 1st @ 0600 is day 0.25 which is = 0.25 /365.2425 = 0.00068 years
June 1st @ 1200 is day 0.50 which is = 0.50 /365.2425 = 0.00137
June 1st @ 1800 is day 0.75 which is = 0.75 /365.2425 = 0.00205
June 1st @ 2359 is day 0.99 which is = 0.99931/365.2425 = 0.002736
Jun2 2nd @ 0000 is day 1 which is = 1.00000/365.2425 = 0.00274
Jun2 2nd @ 2400 is day 2 which is = 2.00000/365.2425 = 0.00548
Dec 31st @ 2400 is day 214 which is = 214. /365.2425 = 0.58591
Jan 1st @ 2400 is day 215 which is = 215. /365.2425 = 0.58865
= 1 + DayOfYear(12,31) - DayOfYear(5,31)
Apr 10 @ 2400 is day 314 which is = 314. /365.2425 = 0.85970
if leap year 315 which is = 315. /365.2425 = 0.86244 However, leap year will be handled in the main loop
in which it is considered only for the current year

GUI specifies 2022 to 2026, however, it passes to this program 2022 to 2025 as these are the years for which growth starts. The resulting files are still the same.

X_Y_BIOM_2022_DN Initial state as of June 1, 2022 @ 00:00, i.e. May 31, 2022 @ 24:00
X_Y_BIOM_2023_DN Growth state as of May 31, 2023 @ 24:00, results for 1st year growth
X_Y_BIOM_2024_DN Growth state as of May 31, 2024 @ 24:00, results for 2nd year growth
X_Y_BIOM_2025_DN Growth state as of May 31, 2025 @ 24:00, results for 3rd year growth
X_Y_BIOM_2026_DN Growth state as of May 31, 2026 @ 24:00, results for 4th year growth

1.1 Initialize Simulation Parameters

1.1.1 Read Input

Values are read in from file name given on command line, e.g.

ScallopPopDensity.exe **Scallop.cfg**

Time steps per Year: number of time steps each year

The following are used to name configuration files used by other modules

Mortality Config File
Recruit Config File
Grid Manager Config File

Additional parameters are placed on the command line to facilitate batch processing

Start Year
Stop Year
Domain Name
MA
GB
AL, for both MA and GB

1.2 Set Grid Manager

1.2.1 Load Grid and Initial State

The initial state is defined by a hardcoded data file named as follows:

Data/bin5mmYYYY[MA|GB].csv

where the year, YYYY, is defined by the Start Year and MA or GB is specified by the given domain name.

The data in each file, Data/bin5mmYYYY[MA|GB].csv has grid information for where each grid is located and its depth. Data in the same row is used for the initial state, in units of scallop count per square for each size classs.

1.2.2 Load Area Coordinates

After the initial state has been loaded, GeoSAMS will check if any of the grid locations are in a special access area. This will be used later to set fishing mortality based on these location settings.

1.3 Set Growth

The simulation then instantiates parameters that define how growth occurs

1.3.1 Shell Length

Starting at 30mm to 150mm inclusive, in 5 mm steps.

That is (150 - 30) / 5 + 1, or 25 size classes

1.3.2 Shell to Weigth, in grams

GB

$$\begin{aligned} ShellToWeight = exp(& - 6.69 + 2.878 * log(shell_{length}) \\ & - 0.0073 * depth - 0.073 * latitude \\ & + (1.28 - 0.25 * log(shell_{length}) * isClosed) \end{aligned}$$

MA

$$\begin{aligned} ShellToWeight = exp(& - 9.713394 + 2.62025 * log(shell_{length}) \\ & - 0.004665 * depth + 0.021 * latitude \\ & - 0.031 * isClosed) \end{aligned}$$

where *isClosed* is 1 if closed or 0 if open

1.3.3 Compute Growth Parameters, given depth, latitude, and isClosed

L_{∞_μ}
L_{∞_σ}
K_μ
K_σ

1.3.4 Compute G matrix for given growth parameters

From MN18 p. 1312, 1313, where $len = shell_{length}$

$$c = 1.0 - exp(-K_\mu * \delta_t)$$

$$\eta = c * L_{\infty_\mu}$$

For each size class, *k*

$$\omega_k = len_k - len_{k-1}, \text{typically 5mm}$$

$$\omega_{k_{avg}} = \frac{len_k + len_{k-1}}{2}, \text{typically 2.5mm}$$

$$\Omega = (1-c)\omega_k$$

$$X(y,k) = len_y - \eta - (1-c)len_k$$

$$\Phi(x,\mu,\sigma) = \frac{1}{2}\left(1 + Erf(\frac{x-\mu}{\sigma\sqrt{2}})\right)$$

$$\phi(x,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

$$H_{MN18}(x,\sigma,\omega) = \frac{1}{\omega}\left(x\Phi_N(x,0,\sigma^2) + \sigma^2\phi_N(x,0,\sigma^2)\right)$$

$$G(y,k,\sigma,\omega_k) = H_{MN18}(X(y,k-1),\sigma,\Omega) - H_{MN18}(X(y,k),\sigma,\Omega)$$

1.4 Set Recruitment

The simulation next instantiates how recruitment will be handled.

1.4.1 For years start_year to stop_year

Data is read randomly chosen from from RecruitEstimates/RecruitEstimateDNYYYY.txt. YYYY is the range of years of available recruit information as pulled from the survey data file.

1.4.2 This method is effectively setting

For year in start_year to stop_year

Year_index = year - start_year + 1
for year_index in [1..max]

recruitment(year_index) = RecruitEstimate(random year index)
year(year_index) = year
rec_start = Start Period, typically 0/365.2425, or January 1st
rec_stop = Stop Period, typically 100/365.2425, or April 10

1.4.3 It then quantizes recruitment,

For each grid, n

$$L30mm = \left(L_{\infty_\mu}(n) - 30\right) * exp(-K_\mu(n))$$

For each class, j
If (length(j) <= L30mm) recruit(n).max_rec_ind = j

1.5 Set Mortality

The simulation next sets mortality values.

Table 1 Mortality

Region	Adult	Incidental	Base Length l_0
MA	**25%**	**5%**	**65.0**
GB	**20%**	**10%**	**70.0**

1.5.1 Compute alpha

$$\alpha(shell_{length}) = 1 - \frac{1}{1 + exp(-(shell_{length} - length_0)/10.0)}$$

1.5.2 Compute Fishing Effort

1.5.2.1 Compute landings at size for each grid location

Given The number of scallops per square meter:

$$\vec{scallops} = \vec{selectivity}_{loc} \cdot \vec{state}_{loc}$$

and the exploitable biomass in grams per square meter

$$EBMS_{loc} = \vec{scallops} \cdot \vec{weight}_{loc}$$

$$\vec{landings}_{size} = (1.0 - exp((-F_{mort_{loc}} * \delta_)) * \vec{state}_{loc} * gridArea * \vec{selectivity}_{loc}$$

1.5.2.2 Compute landings by weight

$$\vec{landings}_{wgt} = \vec{landings}_{size} \cdot \vec{weight} = catch$$

This is also considered total_catch

1.5.2.3 Total catch is used compute fishing effort

$$rms = \sum_{loc=1}^{n} \frac{EBMS(loc)^2}{scallops(loc))}$$

$$FishingEffort = \frac{\vec{EBMS} * catch/\vec{scallops}}{rms * gridArea}$$

1.6 Main Loop

1.6.1 For each time step

1.6.1.1 Set Fishing Effort

Here there is defined a fishing effort that is independent of mortality. Whereas the mortality fishing effort is a function of region and historical data, this fishing effort is a function of cost, biomass or as a spatial constant within region.

1.6.2 For each grid

1.6.2.1 Compute natural mortality

Determine the number of scallops in millions, S, given the current state

$$S = state * domainArea$$

This is used to determine the juvenile mortality. Adult mortality was defined at module instantiation.

Mid-Atlantic:

$$M_{juv} = \begin{cases} exp(1.093 * log(S) - 9.701), & \text{if } S > 1400 \text{ million} \\ 0.25, & \text{otherwise} \end{cases}$$

Georges Bank:

$$M_{juv} = \begin{cases} exp((1.226 * log(S) - 10.49)), & \text{if } S > 1400 \text{ million} \\ 0.2, & \text{otherwise} \end{cases}$$

Finally

$$M_{nat} = \alpha * M_{juv} + (1 - \alpha) M_{adult}$$

1.6.2.2 Adjust population state based on von Bertalanffy growth

$$\vec{S} = |G| \times \vec{S}$$

1.6.2.3 Adjust population state based on von Bertalanffy growth

If within recruitment period, i.e. Jan 1st to April 10th

$$\vec{S} = \vec{S} + \delta_t \frac{\vec{R}}{RecruitDuration}$$

1.6.2.4 Compute Overall Mortality

$$\vec{M} = \vec{M}_{nat} + Fishing * (\vec{M}_{selectivity} + \vec{M}_{incidental} + \vec{M}_{discard})$$

1.6.2.5 Compute effect of mortality to arrive at new state

$$\vec{S}_{t+1} = \vec{S}_t * (1 - \delta_t * \vec{M})$$

2 Growth_Mod

2.1 Growth Class

The scallop state_vector at each node in the domain is a vector of length $N_{sc} = (150 - 30)/5 + 1 = 25$ representing the abundance of scallops in size classes $[30 - 35mm, 35 - 40mm, ...145 - 150mm, 150mm+]$.

Size class transition matrices are generated for each node based on the work of Millar and Nottingham 2018 Appendix C, henceforth MN18 **[1]**, although other methods are present in the code including direct Monte Carlo simulation. including(see subroutine $GenTransMat$).

Growth in GeoSAMS is based off of von Bertanlffy growth.

$$\delta(u) = (L_\infty - u)(1 - e^{-K})$$

Or from HC2009 **[2]**, equation (1)

$$L_t = L_{t-1}e^{-K} + L_\infty(1 - e^{-K})$$

We assume normal distribution on L_∞ and K with all distribution parameters independent.

The shell height of the ith individual at time $t + 1$, $L_{t+1,i}$ depends on the random effects (α_i and β_i) as well as the mean slope and intercept:

$$L_{t+1,i} = (m + \alpha_i)L_{t,i} + (b + \beta_i) + \epsilon,$$
where ϵ is a random error with expected value zero.

The values of the distribution means (μ_{L_∞} and μ_K) are taken from previous work of Hart, HC2009. The distribution of increments by size class as in MN18). Growth increment is given by the von Bertlanaffy growth curve

We begin by determining the scallop time to grow for a given year: Computes the overall growth of the scallop population over a time period of (num_time_steps * delta_time) in units of years, typically one year with delta_time as a decimal year, e.g. one day = 1/365.2425 = 0.00274

For each time step, δ_t

Computes mortality based on current state_vector.

Computes increase in population due to recruitment, $\vec{R}$,if within recruitment months, i.e. Jan to April 10th

$$\vec{S} = \vec{S} + \delta_t \frac{\vec{R}}{RecruitDuration}$$

Adjusts population based on von Bertalanffy growth

$$\vec{S} = |G| \times \vec{S}$$
where G is the transition matrix

Compute overall mortality, **M**

$$\vec{M} = \vec{M}_{nat} + Fishing * \left(\vec{M}_{selectivity} + \vec{M}_{incidental} + \vec{M}_{discard}\right)$$

Compute new state_vector

$$\vec{S_{t+1}} = \vec{S_t} * \left(1 - \delta_t * \vec{M}\right)$$

1. MN18 refers to Miller, R. B. and Nottingham, 2018, "Improved approximations for estimation of size-transition probabilities within size-structured models"

2. HC2009 refers to Hart, D. R. and Chute, A. S. 2009, "Estimating von Bertalanffy growth parameters from growth increment data using a linear mixed-effects model, with an application to the sea scallop Placopecten magellanicus."

2.1.1 Transition Matrix

A transition matrix, 25 by 25, is computed under the assumption of von Bertlanaffy growth. It is assumed that the parameters of von BernBertlanaffy growth K and L_inf have normal distributions.

From MN18 p. 1312, 1313

$$c = 1.0 - e^{-K_\mu * \delta_t}$$

$$\eta = c * L_{\infty_\mu}$$

$$\omega_k = l_k - l_{k-1}$$

$$\omega_{k_{avg}} = \frac{l_k + l_{k-1}}{2}$$

$$\Omega = (1 - c)\omega_k$$

$$X(y,k) = l_y - \eta - (1 - c)l_k$$

$$G(y,k,\sigma,\omega_k) = H_{MN18}(X(y,k-1),\sigma,\Omega) - H_{MN18}(X(y,k),\sigma,\Omega)$$

2.1.1.1 Function H(x, sigma, omega)

Given (MN18 Appendix B)

Φ_N denotes the normal **cumulative** distribution function.

ϕ_N denotes the normal **density** function.

$$H_{MN18}(x,\sigma,\omega) = \frac{1}{\omega}\left(x\Phi_N(x,0,\sigma^2) + \sigma^2\phi_N(x,0,\sigma^2)\right)$$

2.1.1.2 Normal Cumulative Distribution Function

$$\Phi(x,\mu,\sigma) = \frac{1}{2}(1 + Erf(\frac{x-\mu}{\sigma\sqrt{2}}))$$

2.1.1.3 Normal Density Function

$$\phi(x,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

3 Recruit_Mod

3.1 Recruitment Class

An array of weights is computed based on the number of recruitment years that favors more recent recruit estimates. The weighting is then used to randomly choose an index into the available recruit data. This index is used to preload the recruit data into the Recruitment Class structure.

recruitment() = data read in from randomly chosen recruit file
year() = simulation year
rec_start = a decimal value given as day of the year divided by 365.2425. Typically 0, which would be January 1
rec_stop = decimal value given as day of the year divided by 365.2425. Typically 100/365.2425, which is April 10th.

4 Mortality_Mod

The methods in this class are used to determine the selectiviy and discard of the scallops based on shell length and location.

Set_Mortality

Instantiates private members for this class.

Reads in its configuration parameters and stores to private members.
Loads Fishing Mortalities, if enabled by GridManager
Sets up a repository for key values to allow offline analysis
Loads historical data for Fishing Effort
Set selectivity as computed by Ring_Size_Selectivity based on shell length and grid location

4.1 Read_Configuration

Opens the configuration file specified in the simulation configuration file and as set by *Set_Config_File_Name*

4.2 Load_Fishing_Mortalities

Opens the configuration file specified in the Mortality configuration file and as set by *Set_Fishing_Mortality*

4.3 Set Mortality

Mortality = $Mortality_{adult}$

4.4 Compute Alpha

$$Alpha_{mortality} = 1.0 - \frac{1.0}{(1.0 + exp(-(shell_{lengths}(:) - length_0)/10.0))}$$

4.5 Compute Selectivity

Assign size class fishing selectivity based on increasing logistic function

$$selectivity = \frac{1}{1 + exp(F_{sel_a} - F_{sel_b} * (shell_{len} + shell_{len_{delta}}/2.0))}$$

4.6 Compute Discard

$$Discard = \begin{cases} 0.0, & \text{if } length > cullSize \text{ or gridIsClosed} \\ discard_{region} * selectivity, & \text{otherwise} \end{cases}$$

4.7 Mortality_Write_At_Timestep

5 Grid Manager Mod

5.1 Grid Manager Class

5.1.1 Brief

The Grid Manager is responsible for setting up the grid by reading in each grid's coordinates from the ***Initial_Conditions*** *file* named by the ***Grid_Manager_Config_File*** *in* Scallop.cfg.

The main program instantiates a Grid Manger by calling *Set_Grid_Manager*

5.1.2 Set Grid Manager

This routine initializes private variables. Calls ***Read_Configuration*** *that* reads in the Grid Manger configuration file as given by the main configuration and set via ***Set_Config_File_Name*** .

Load_Grid_State *loads* the grid data from the file defined by the start year and domain.
Data/bin5mmYYYYDN.csv
This establishes the number of grids, *num_grids* , and the initial state of the scallop density, @ state.

If a special access area definitions are provided, these are loaded via ***Load_Area_Coordinates*** . Each grid location if then checked if it is in a special access area and identified as such by setting *special_access_index* to the index of the corresponding access area.

5.1.3 Read_Configuration

Reads given file name and scans each line, input string, for tag and value characters. Also determines if special access areas are desired and if not sets *use_spec_access_data* to false

5.1.4 Load_Area_Coordinates

If *use_spec_access_data* is true then reads given file name. Scans each input line for an area longitude vector coordinates followed by latitude vector coordinates. The length of each vector must be equal and establishes the number of vertices, or edges i.e. @ n_sides that define the special access area. The number of such vector pairs establishes the *num_ares* defined

5.1.5 Is_Grid_In_Special_Access

This method uses a grids longitude and latitude coordinates are in a special area. It does so by using a point in polygram algorithm. The data vector representation is used when calling ***Point_In_Polygon_Vector*** *@section* p4p3 Grid Manager Support Methods

5.1.6 Set_Config_File_Name

Sets *config_file_name* for ***Read_Configuration*** *@subsection* p4p2p2 Set_Init_Cond_File_Name Sets *init_cond_fname* for ***Load_Grid_State*** *@subsection* p4p2p3 Set_Special_Access_File_Name

5.2 Point In Polygon

The Point_In_Polygon_Vector method is used to find if a point is in a polygon. The ***Grid_Manager*** *also* supports polygon data representation as an array of LonLatPoint points vial ***Point_In_Polygon_Points*** *or* as a n by 2, 2-dimensional array, where n is a maximum of *max_sides* edges.

5.2.1 Point_In_Polygon_Points

5.2.2 Point_In_Polygon_Array

5.2.3 Point_In_Polygon_Vector

6 Common Parameters

7 Modules Index

7.1 Modules List

Here is a list of all modules with brief descriptions:

8 Data Type Index

8.1 Data Types List

Here are the data types with brief descriptions:

9 File Index

9.1 File List

Here is a list of all files with brief descriptions:

10 Module Documentation

10.1 globals Module Reference

10.1.1 Functions/Subroutines

elemental real(**dp**) function **logic_to_double** (value)
real(**dp**) function, dimension(n, n) **matrixinv** (x, n)
logical function **leap_year** (year)
logical function **divby** (y, val)
integer function **dayofyear** (m, d)
logical function **is_nan** (x)

10.1.2 Variables

integer, parameter **sp** = selected_real_kind(6, 37)
integer, parameter **dp** = selected_real_kind(15, 307)
integer, parameter **qp** = selected_real_kind(33, 4931)
integer, parameter **ndim** = 12000
integer, parameter **shell_len_max** = 150
integer, parameter **shell_len_min** = 30
integer, parameter **shell_len_delta** = 5
integer, parameter **num_size_classes** = (**shell_len_max** - **shell_len_min**) / **shell_len_delta** + 1
integer, parameter **max_num_years** = 50
integer, parameter **max_num_areas** = 25
integer, parameter **max_sides** = 8
integer, parameter **region_none** =0
integer, parameter **region_n** =1
integer, parameter **region_s** =2
integer, parameter **region_sw** =3
integer, parameter **region_w** =4
integer, parameter **region_ma** =5
integer, parameter **region_gbk** = 1
integer, parameter **region_mab** = 5
integer, parameter **tag_len** = 40
integer, parameter **value_len** = 30
integer, parameter **comment_len** = 80
integer, parameter **line_len** = **tag_len**+**value_len**+**comment_len**
integer, parameter **fname_len** = 100
integer, parameter **form_len** = 20
integer, parameter **input_str_len** = 100
integer, parameter **csv_line_len** = 2000
integer, parameter **domain_len** = 2
integer, parameter **read_dev** = 69
integer, parameter **write_dev** = 63
real(**dp**), parameter **zero_threshold** = 1.0D-99
real(**dp**), parameter **pi** = 3.14159265358979323846264338327950288D0
real(**dp**), parameter **grams_per_pound** = 453.592_dp
real(**dp**), parameter **meters_per_naut_mile** = 1852.D0
real(**dp**), parameter **grams_per_metric_ton** = 1000000._dp
real(**dp**), parameter **grid_area_sqm** = **meters_per_naut_mile****2
real(**dp**), parameter **tow_area_sqm** = 4516._dp
real(**dp**), parameter **one_scallop_per_tow** = 1.D0 / **tow_area_sqm**
real(**dp**), parameter **ma_gb_border** = -70.5

real(**dp**), parameter **days_in_year** =365+0.25-0.01+0.0025
character(*), parameter **term_red** = "//achar(27)//'[31m'
character(*), parameter **term_yel** = "//achar(27)//'[33m'
character(*), parameter **term_grn** = "//achar(27)//'[92m'
character(*), parameter **term_blu** = "//achar(27)//'[94m'
character(*), parameter **term_blk** = "//achar(27)//'[0m'
character(*), parameter **init_cond_dir** = 'InitialCondition/'
character(*), parameter **growth_out_dir** = 'GrowthOutput/'
character(*), parameter **rec_input_dir** = 'RecruitEstimates/'
character(*), parameter **rec_output_dir** = 'RecruitField/'
character(*), parameter **output_dir** = 'Results/'
character(*), parameter **config_dir_sim** = 'Configuration/Simulation/'
character(*), parameter **config_dir_interp** = 'Configuration/Interpolation/'
character(*), parameter **config_dir_special** = 'Configuration/SpecialAccess/'
character(*), parameter **grid_dir** = 'Grids/'
character(*), parameter **data_dir** = 'Data/'
character(*), parameter **anal_dir** = 'Analysis/'
integer, parameter **num_regions** = 2
character(3), dimension(**num_regions**) **rgn** = (/ '_GB', '_MA'/)

10.1.3 Function/Subroutine Documentation

10.1.3.1 integer function globals::dayofyear (integer, intent(in) *m*, integer, intent(in) *d*)

Here is the call graph for this function:

Here is the caller graph for this function:

10.1.3.2 logical function globals::divby (integer *y*, integer *val*)

Here is the call graph for this function:

Here is the caller graph for this function:

10.1.3.3 logical function globals::is_nan (real(dp), intent(in) *x*)

Here is the call graph for this function:

Here is the caller graph for this function:

10.1.3.4 logical function globals::leap_year (integer *year*)

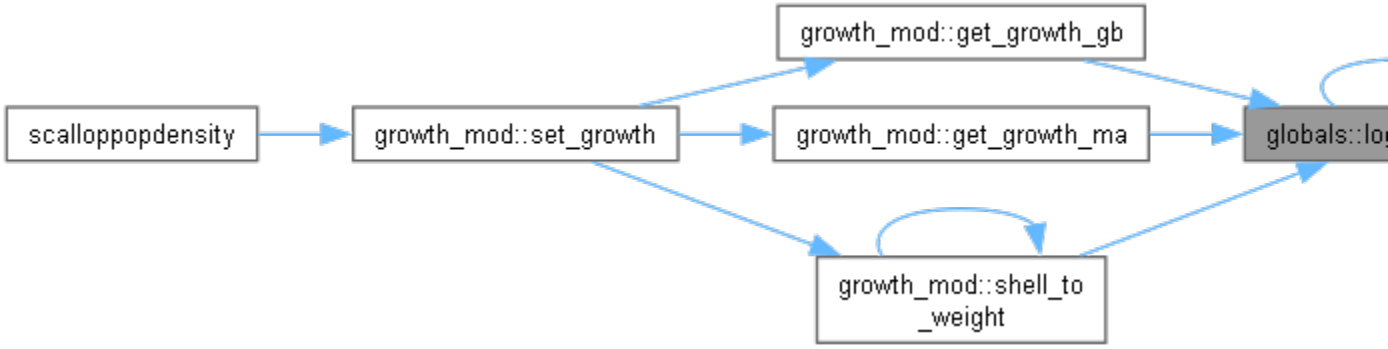
Here is the call graph for this function:

Here is the caller graph for this function:

10.1.3.5 elemental real(dp) function globals::logic_to_double (logical, intent(in) *value*)

Here is the call graph for this function:

Here is the caller graph for this function:



10.1.3.6 real(dp) function, dimension(n,n) globals::matrixinv (real(dp), dimension(n,n), intent(in) *x*, integer, intent(in) *n*)

Here is the call graph for this function:

Here is the caller graph for this function:

10.1.4 Variable Documentation

10.1.4.1 character(*), parameter globals::anal_dir = 'Analysis/'

10.1.4.2 integer, parameter globals::comment_len = 80

10.1.4.3 character(*), parameter globals::config_dir_interp = 'Configuration/Interpolation/'

10.1.4.4 character(*), parameter globals::config_dir_sim = 'Configuration/Simulation/'

10.1.4.5 character(*), parameter globals::config_dir_special = 'Configuration/SpecialAccess/'

10.1.4.6 integer, parameter globals::csv_line_len = 2000

10.1.4.7 character(*), parameter globals::data_dir = 'Data/'

10.1.4.8 real(dp), parameter globals::days_in_year =365+0.25-0.01+0.0025

10.1.4.9 integer, parameter globals::domain_len = 2

10.1.4.10 integer, parameter globals::dp = selected_real_kind(15, 307)

10.1.4.11 integer, parameter globals::fname_len = 100

10.1.4.12 integer, parameter globals::form_len = 20

10.1.4.13 real(dp), parameter globals::grams_per_metric_ton = 1000000._dp

10.1.4.14 real(dp), parameter globals::grams_per_pound = 453.592_dp

10.1.4.15 real(dp), parameter globals::grid_area_sqm = meters_per_naut_mile2**

10.1.4.16 character(*), parameter globals::grid_dir = 'Grids/'

10.1.4.17 character(*), parameter globals::growth_out_dir = 'GrowthOutput/'

10.1.4.18 character(*), parameter globals::init_cond_dir = 'InitialCondition/'

10.1.4.19 integer, parameter globals::input_str_len = 100

10.1.4.20 integer, parameter globals::line_len = tag_len+value_len+comment_len

10.1.4.21 real(dp), parameter globals::ma_gb_border = -70.5

10.1.4.22 integer, parameter globals::max_num_areas = 25

10.1.4.23 integer, parameter globals::max_num_years = 50

10.1.4.24 integer, parameter globals::max_sides = 8

10.1.4.25 real(dp), parameter globals::meters_per_naut_mile = 1852.D0

10.1.4.26 integer, parameter globals::ndim = 12000

10.1.4.27 integer, parameter globals::num_regions = 2

10.1.4.28 integer, parameter globals::num_size_classes = (shell_len_max - shell_len_min) / shell_len_delta + 1

10.1.4.29 real(dp), parameter globals::one_scallop_per_tow = 1.D0 / tow_area_sqm

10.1.4.30 character(*), parameter globals::output_dir = 'Results/'

10.1.4.31 real(dp), parameter globals::pi = 3.14159265358979323846264338327950288D0

10.1.4.32 integer, parameter globals::qp = selected_real_kind(33, 4931)

10.1.4.33 integer, parameter globals::read_dev = 69

10.1.4.34 character(*), parameter globals::rec_input_dir = 'RecruitEstimates/'

10.1.4.35 character(*), parameter globals::rec_output_dir = 'RecruitField/'

10.1.4.36 integer, parameter globals::region_gbk = 1

10.1.4.37 integer, parameter globals::region_ma =5

10.1.4.38 integer, parameter globals::region_mab = 5

10.1.4.39 integer, parameter globals::region_n =1

10.1.4.40 integer, parameter globals::region_none =0

10.1.4.41 integer, parameter globals::region_s =2

10.1.4.42 integer, parameter globals::region_sw =3

10.1.4.43 integer, parameter globals::region_w =4

10.1.4.44 character(3), dimension(num_regions) globals::rgn = (/ '_GB', '_MA'/)

10.1.4.45 integer, parameter globals::shell_len_delta = 5

10.1.4.46 integer, parameter globals::shell_len_max = 150

10.1.4.47 integer, parameter globals::shell_len_min = 30

10.1.4.48 integer, parameter globals::sp = selected_real_kind(6, 37)

10.1.4.49 integer, parameter globals::tag_len = 40

10.1.4.50 character(*), parameter globals::term_blk = ''//achar(27)//'[0m'

10.1.4.51 character(*), parameter globals::term_blu = ''//achar(27)//'[94m'

10.1.4.52 character(*), parameter globals::term_grn = ''//achar(27)//'[92m'

10.1.4.53 character(*), parameter globals::term_red = ''//achar(27)//'[31m'

10.1.4.54 character(*), parameter globals::term_yel = ''//achar(27)//'[33m'

10.1.4.55 real(dp), parameter globals::tow_area_sqm = 4516._dp

10.1.4.56 integer, parameter globals::value_len = 30

10.1.4.57 integer, parameter globals::write_dev = 63

10.1.4.58 real(dp), parameter globals::zero_threshold = 1.0D-99

10.1.4.59

10.2 grid_manager_mod Module Reference

10.2.1 Data Types

type **grid_data_class**type **lonlatpoint**
type **lonlatvector**

10.2.2 Functions/Subroutines

integer function **set_num_grids** ()

Determines the expected number of grids by simply counting the number of lines with text in the initial state file. It does not perform any error checking, only counting the number of lines with text and stopping at the first blank line.

subroutine **set_grid_manager** (state_mat, grid, ngrids, dom_name)

Initializes growth for startup.

subroutine **set_config_file_name** (fname)

Used during instantiation to set the name of the file to read to for configuration parameters.

subroutine **set_init_cond_file_name** (fname)

Used during instantiation to set the name of the file to read to for grid locations, state.

subroutine **set_special_access_file_name** (fname)

Used during instantiation to set the name of the file to special access coordinates.

integer function **get_num_of_areas** ()

Get'r function for private member num_areas.

subroutine **read_configuration** ()

Read_Configuration.

integer function **load_grid_state** (grid, state_mat)

This function is used to set the grid parameters and the initial state to start the simulation.

integer function **load_area_coordinates** ()
integer function **is_grid_in_special_access** (lon, lat)
logical function **point_in_polygon_points** (poly, point, nodes)
logical function **point_in_polygon_array** (poly, point, nodes)
logical function **point_in_polygon_vector** (polyx, polyy, x, y, nodes)

First of all, notice that each iteration considers two adjacent points and the target point. Then the if statement evaluates two conditions:

10.2.3 Variables

type(**lonlatvector**), dimension(**max_num_areas**), private **area**
integer, private **num_areas**
integer, private **num_grids**

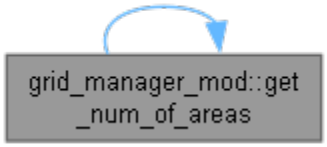
logical, private **use_spec_access_data**
character(**domain_len**), private **domain_name**
character(**fname_len**), private **config_file_name**
character(**fname_len**), private **init_cond_fname**
character(**fname_len**), private **special_accesss_fname**

10.2.4 Function/Subroutine Documentation

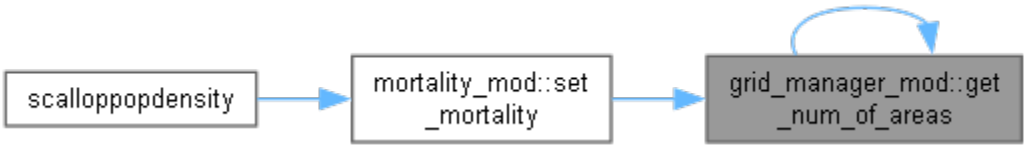
10.2.4.1 integer function grid_manager_mod::get_num_of_areas

Get'r function for private member num_areas.

Here is the call graph for this function:

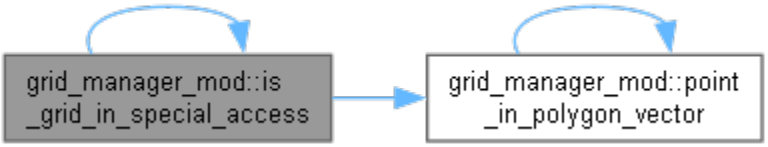


Here is the caller graph for this function:

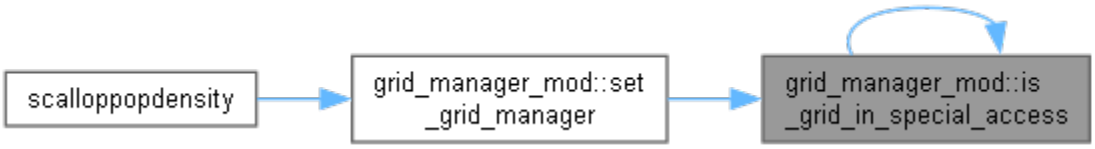


10.2.4.2 integer function grid_manager_mod::is_grid_in_special_access (real(dp), intent(in) *lon*, real(dp), intent(in) *lat*)

Here is the call graph for this function:

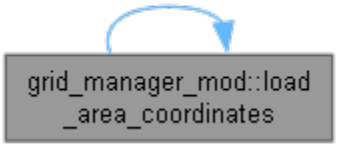


Here is the caller graph for this function:

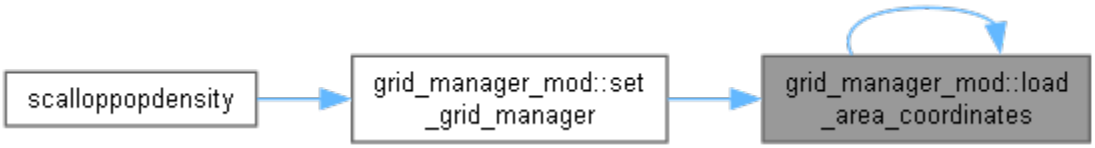


10.2.4.3 integer function grid_manager_mod::load_area_coordinates

Here is the call graph for this function:



Here is the caller graph for this function:



10.2.4.4 integer function grid_manager_mod::load_grid_state (type(grid_data_class), dimension(*), intent(out) *grid*, real(dp), dimension(1:num_grids, 1:num_size_classes), intent(out) *state_mat*)

This function is used to set the grid parameters and the initial state to start the simulation.

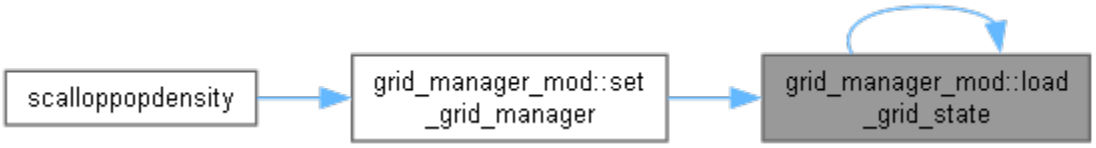
It does so by reading the CSV file at file_name. This file has been generated by the TrawlData5mm.m Matlab script. The format is for each grid in a row, the columns are Decimal Year, UTM X, UTM Y, Latitude, Longitude, UTM Z, Grid Is Closed, Followed by Scallop Density in Count/m^2 sorted by shell length 30 to 150 mm in 5mm increments for 25 columns

10.2.4.4.1 Parameters

in,out	*grid*	Holds position information
out	*state*	Holds the initial state at various location specified by grid
in	*file_name*	CSV name to be read in

Here is the call graph for this function:

Here is the caller graph for this function:



10.2.4.5 logical function grid_manager_mod::point_in_polygon_array (real(dp), dimension(max_sides,2), intent(in) *poly*, real(dp), dimension(2), intent(in) *point*, integer, intent(in) *nodes*)

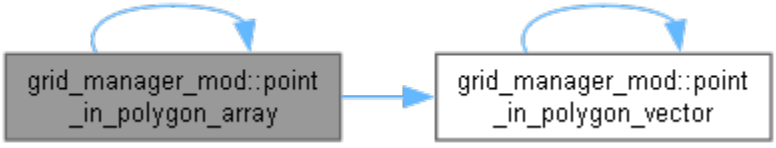
10.2.4.5.1 Parameters

poly	Array of x,y coordinates that define polygram,
point	x,y coordinate of point we wish to determine if inside polygram
nodes	the number of corners, edges, that define the polygon

10.2.4.5.2 Returns

true if point is inside polygram, false if outsied if point is on an edge then is may return true of false

Here is the call graph for this function:



Here is the caller graph for this function:

10.2.4.6 logical function grid_manager_mod::point_in_polygon_points (type(lonlatpoint), dimension(*), intent(in) *poly*, type(lonlatpoint), intent(in) *point*, integer, intent(in) *nodes*)

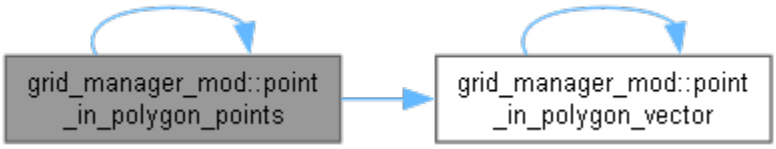
10.2.4.6.1 Parameters

poly	Array of LonLatPoint coordinates that define polygram,
point	LonLatPoint coordinate of point we wish to determine if inside polygram
nodes	the number of corners, edges, that define the polygon

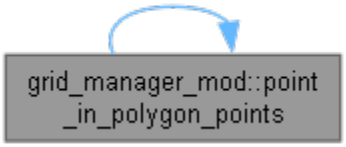
10.2.4.6.2 Returns

true if point is inside polygram, false if outsied if point is on an edge then is may return true of false

Here is the call graph for this function:



Here is the caller graph for this function:



10.2.4.7 logical function grid_manager_mod::point_in_polygon_vector (real(dp), dimension(*), intent(in) *polyx*, real(dp), dimension(*), intent(in) *polyy*, real(dp), intent(in) *x*, real(dp), intent(in) *y*, integer, intent(in) *nodes*)

First of all, notice that each iteration considers two adjacent points and the target point. Then the if statement evaluates two conditions:

1.) Y-value of our target point is within the range [verty[j], verty[i]). 2.) X-value of our target point is below the linear line connecting the point j and i. If you're having problems to see this second condition, just write down the linear equation of the line, reorganize the expression a little bit and place testy as the free variable.

Every time the above two conditions are met, we toggle the flag c. So we return true if above conditions are met odd number of times and false otherwise.

`http://alienryderflex.com/polygon/`

10.2.4.7.1 Parameters

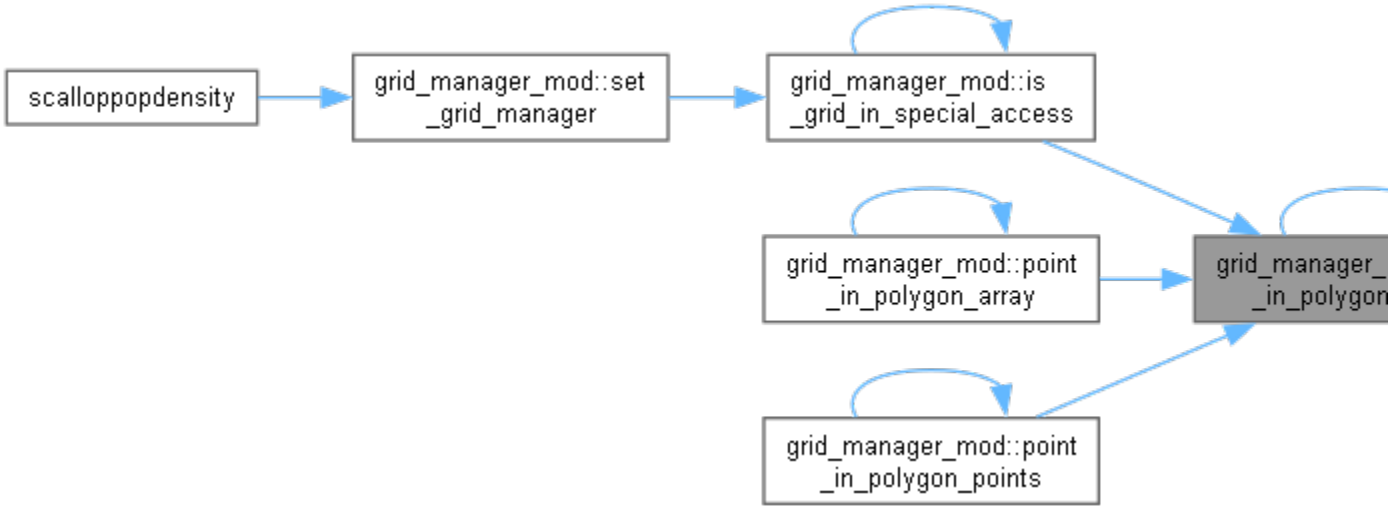
polyX	Array of horizontal, coordinates of corners
polyY	Array of vertical coordinates of corners
x	horizontal coordinate of point we wish to determine if inside polygram
y	vertical coordinate of point we wish to determine if inside polygram
nodes	the number of corners, edges, that define the polygon

10.2.4.7.2 Returns

true if point is inside polygram or if on vert or horiz edge, if point is on rise of falling edge then it may return true or false

Here is the call graph for this function:

Here is the caller graph for this function:



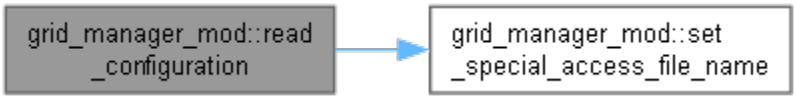
10.2.4.8 subroutine grid_manager_mod::read_configuration

Read_Configuration.

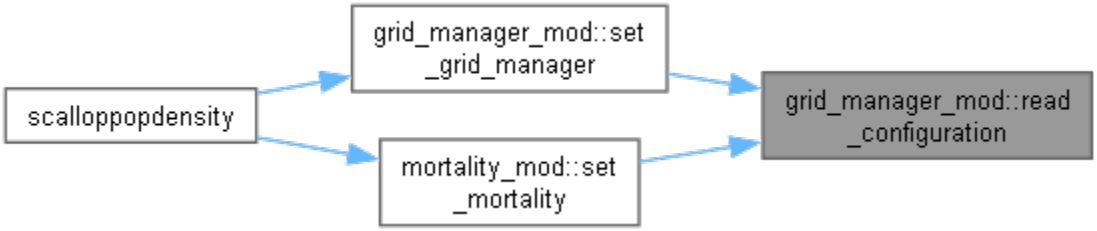
Read Input File

Reads a configuration file

Here is the call graph for this function:



Here is the caller graph for this function:



10.2.4.9 subroutine grid_manager_mod::set_config_file_name (character(*), intent(in) *fname*)

Used during instantiation to set the name of the file to read to for configuration parameters.

Read Input File

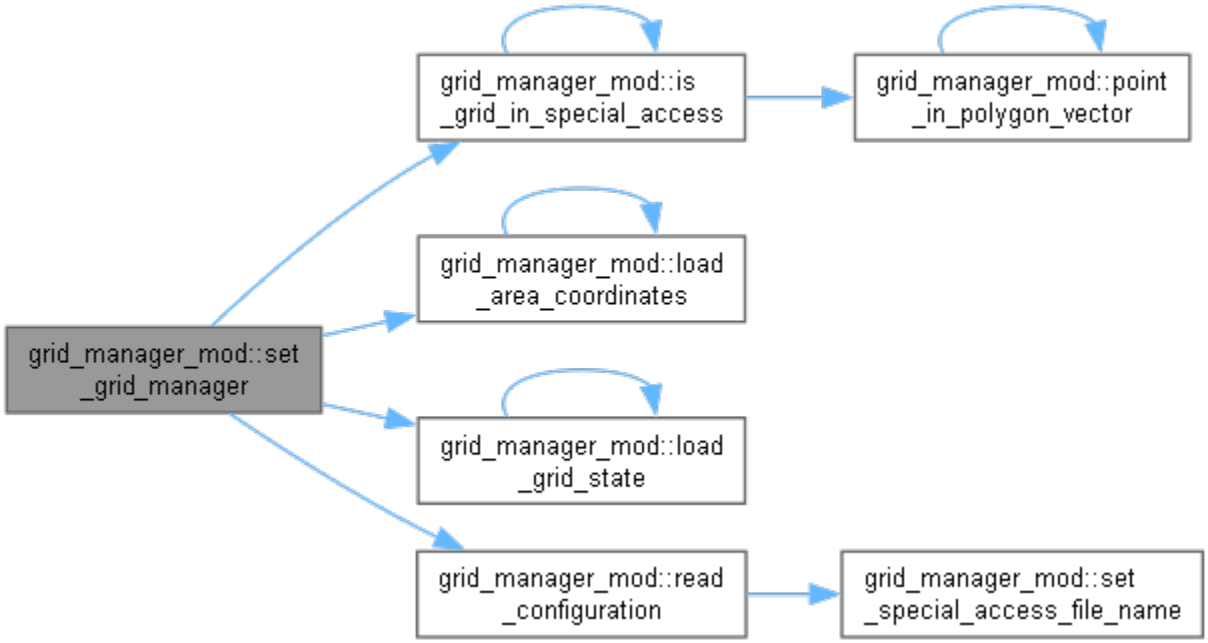
Sets file names for initial state data and special access data

Here is the caller graph for this function:

10.2.4.10 subroutine grid_manager_mod::set_grid_manager (real(dp), dimension(1:ngrids, 1:num_size_classes), intent(out) *state_mat*, type(grid_data_class), dimension(*), intent(out) *grid*, integer, intent(inout) *ngrids*, character(domain_len), intent(in) *dom_name*)

Initializes growth for startup.

Here is the call graph for this function:



Here is the caller graph for this function:

10.2.4.11 subroutine grid_manager_mod::set_init_cond_file_name (character(*), intent(in) *fname*)

Used during instantiation to set the name of the file to read to for grid locations, state.

Read Input File

Sets name of a configuration file, typical 'Data/bin5mmYYYY[MA|GB].csv'

Here is the caller graph for this function:

10.2.4.12 integer function grid_manager_mod::set_num_grids

Determines the expected number of grids by simply counting the number of lines with text in the initial state file. It does not perform any error checking, only counting the number of lines with text and stopping at the first blank line.

10.2.4.12.1 Returns

The expected number of grids to process.

Here is the call graph for this function:

Here is the caller graph for this function:

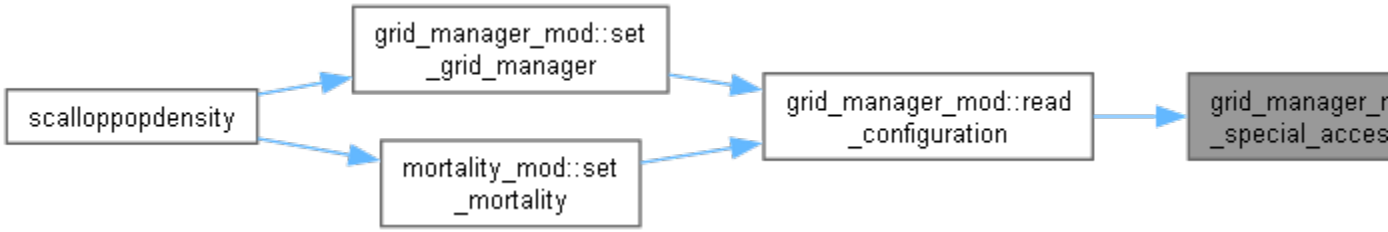
10.2.4.13 subroutine grid_manager_mod::set_special_access_file_name (character(*), intent(in) *fname*)

Used during instantiation to set the name of the file to special access coordinates.

Read Input File

Sets file name for special access coordinates

Here is the caller graph for this function:



10.2.5 Variable Documentation

10.2.5.1 type(lonlatvector), dimension(max_num_areas), private grid_manager_mod::area [private]

10.2.5.2 character(fname_len), private grid_manager_mod::config_file_name [private]

10.2.5.3 character(domain_len), private grid_manager_mod::domain_name [private]

10.2.5.4 character(fname_len), private grid_manager_mod::init_cond_fname [private]

10.2.5.5 integer, private grid_manager_mod::num_areas [private]

10.2.5.6 integer, private grid_manager_mod::num_grids [private]

10.2.5.7 character(fname_len), private grid_manager_mod::special_accesss_fname [private]

10.2.5.8 logical, private grid_manager_mod::use_spec_access_data [private]

10.2.5.9

10.3 growth_mod Module Reference

10.3.1 Data Types

10.3.2 type growth_classFunctions/Subroutines

subroutine **set_growth** (growth, grid, shell_lengths, num_ts, ts_per_year, dom_name, dom_area, state_mat, weight_grams, ngrids)

Initializes growth for startup.

real(**dp**) function, dimension(1:**num_size_classes**, 1:**num_size_classes**) **gen_size_trans_matrix** (l_inf_mu, l_inf_sd, k_mu, k_sd, shell_lengths, method)

Transition matrix used to determine the growth of the scallop. The equations are based on MN18, Appendix C.

real(**dp**) function, dimension(**num_size_classes**) **set_shell_lengths** (length_min, length_delta)

setup shell shell_lengths intervals

subroutine **get_growth_gb** (depth, lat, is_closed, l_inf_mu, k_mu, l_inf_sd, k_sd)

Provides growth parameters L and K parameters for Georges Bank. From R code sent by Dvora July 28th 2021.

subroutine **get_growth_ma** (depth, lat, is_closed, l_inf_mu, k_mu, l_inf_sd, k_sd)

Provides growth parameters L and K parameters for Mid Atlantic From R code sent by Dvora July 28th 2021.

real(**dp**) function, dimension(**num_size_classes**, **num_size_classes**) **mn18_appxc_trans_matrix** (l_inf_mu, k_mu, l_inf_sd, k_sd, shell_lengths)

Purpose: This subroutine computes a sizeclass transition matrix under the assumption of von Bertlanaffy growth. It is assumed that the parameters of von BernBertlanaffy growth K and L_inf have normal distributions.

subroutine **increment_mean_std** (l_inf_mu, k_mu, l_inf_sd, k_sd, size, mu, sigma)

Purpose: This subroutine computes a growth increment distribution parameters under the assumption of von Bertlanaffy growth and normally distributed growth increments. It is assumed that the parameters of von BernBertlanaffy growth K and L_inf have normal distributions.

real(**dp**) function **h_mn18** (x, sigma, w)

Given (MN18 Appendix B)

real(**dp**) function **norm_cumul_dist_fcn** (x, mu, sigma)

Computation of normal cumulative distribution function.

real(**dp**) function **norm_density_fcn** (x, mu, sigma)

Computation of normal density function.

subroutine **enforce_non_negative_growth** (g)
real(**dp**) function, dimension(**num_size_classes**) **time_to_grow** (ts, growth, mortality, recruit, state_vector, fishing_effort, year, longitude)
Computes growth in scallop population.

elemental real(**dp**) function **shell_to_weight** (shell_length_mm, is_closed, depth, latitude, longitude)
Computes weight given a shell height.

subroutine **gamma_inc_values** (n_data, a, x, fx)

10.3.3 Variables

integer, parameter **growth_param_size** = 4
integer, private **num_grids**
character(**domain_len**), private **domain_name**
real(**dp**), private **domain_area_sqm**
integer, private **num_time_steps**
integer, private **time_steps_year**
real(**dp**), private **delta_time**
logical, private **show_recruits_msg**

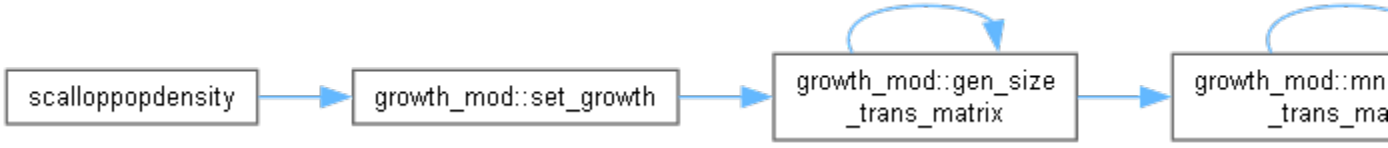
10.3.4 Function/Subroutine Documentation

10.3.4.1 subroutine growth_mod::enforce_non_negative_growth (real(dp), dimension(num_size_classes,*), intent(inout) *g*)

10.3.4.1.1 Parameters

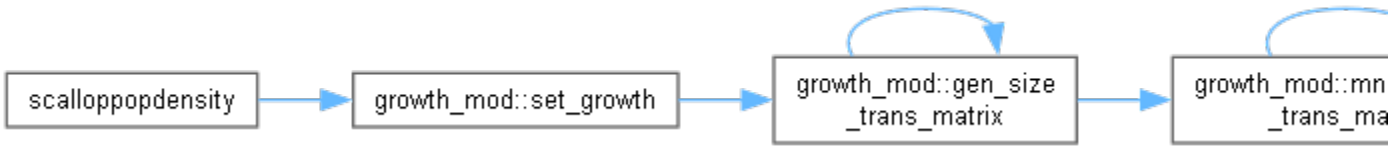
in,out	*G*	- growth transition matrix with negative growth lumped into 0 growth

Here is the caller graph for this function:



10.3.4.2 subroutine growth_mod::gamma_inc_values (integer (kind = 4) *n_data*, real (kind = 8) *a*, real (kind = 8) *x*, real (kind = 8) *fx*)

Here is the caller graph for this function:



10.3.4.3 real(dp) function, dimension(1:num_size_classes, 1:num_size_classes) growth_mod::gen_size_trans_matrix (real(dp), intent(in) *l_inf_mu*, real(dp), intent(in) *l_inf_sd*, real(dp), intent(in) *k_mu*, real(dp), intent(in) *k_sd*, real(dp), dimension(*), intent(in) *shell_lengths*, character(*), intent(in) *method*)

Transition matrix used to determine the growth of the scallop. The equations are based on MN18, Appendix C.

$$\vec{\mathbf{Size}}[Grid] = |\mathbf{GrowthMatrix}[Grid]| \times \vec{\mathbf{Size}}[Grid] \times \left| e^{-(Mort_{nat}[Grid,Height_{shell}]+Mort_{fish}[Grid,Height_{shell}])*timestep} \right|$$

10.3.4.3.1 Parameters

in	*L_inf_mu*	[real 1x1] = mean of von Bertlanaffy asymptotic growth parameter L_inf(see HC09 eqn 1)
in	*L_inf_std*	[real 1x1] = standard deviation of von Bertlanaffy asymptotic growth parameter L_inf(see HC09 eqn 1)
in	*K_mu*	[real 1x1] = mean of mean of von Bertlanaffy asymptotic growth parameter K(see HC09 eqn 1)
in	*K_sd*	[real 1x1] = standard deviation of von Bertlanaffy growth parameter K(see HC09 eqn 1)
in	*shell_lengths*	for each size class

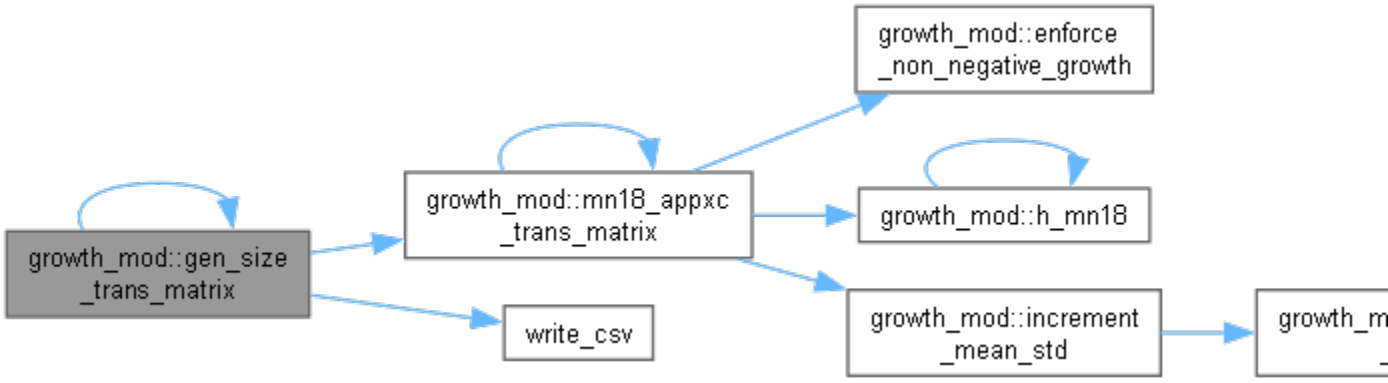
10.3.4.3.2 Returns

Transition Matrix

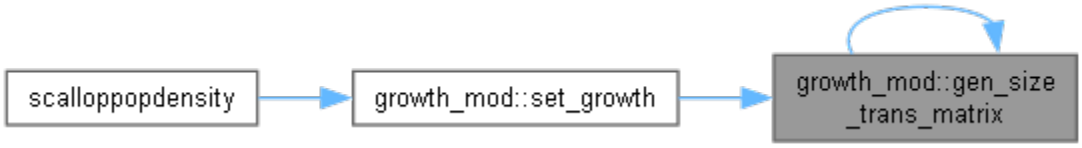
10.3.4.3.3 Author

Keston Smith (IBSS corp) June-July 2021

Here is the call graph for this function:



Here is the caller graph for this function:



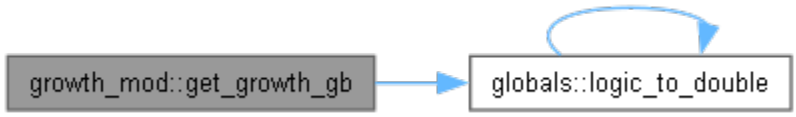
10.3.4.4 subroutine growth_mod::get_growth_gb (real(dp), intent(in) *depth*, real(dp), intent(in) *lat*, logical, intent(in) *is_closed*, real(dp), intent(out) *l_inf_mu*, real(dp), intent(out) *k_mu*, real(dp), intent(out) *l_inf_sd*, real(dp), intent(out) *k_sd*)

Provides growth parameters L and K parameters for Georges Bank. From R code sent by Dvora July 28th 2021.

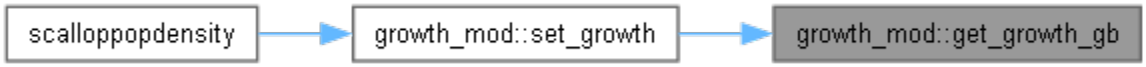
10.3.4.4.1 Parameters

in	*depth*	in meters
in	*lat*	Geospatial coordinate, Latitude
in	*is_closed*	Logical that indicates if grid is closed for fishing
out	*L_inf_mu*	von Bertlanaffy asymptotic growth parameter
out	*K_mu*	von Bertlanaffy asymptotic growth parameter
out	*L_inf_sd*	standard deviation von Bertlanaffy asymptotic growth parameter
out	*K_sd*	standard deviation von Bertlanaffy asymptotic growth parameter
in	*area_index*	index to indicate management area

Here is the call graph for this function:



Here is the caller graph for this function:



10.3.4.5 subroutine growth_mod::get_growth_ma (real(dp), intent(in) *depth*, real(dp), intent(in) *lat*, logical, intent(in) *is_closed*, real(dp), intent(out) *l_inf_mu*, real(dp), intent(out) *k_mu*, real(dp), intent(out) *l_inf_sd*, real(dp), intent(out) *k_sd*)

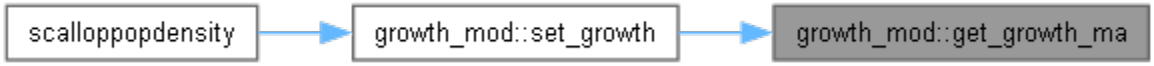
Provides growth parameters L and K parameters for Mid Atlantic From R code sent by Dvora July 28th 2021.

10.3.4.5.1 Parameters

in	*depth*	in meters
in	*lat*	Geospatial coordinate, Latitude
in	*is_closed*	Logical that indicates if grid is closed for fishing
out	*L_inf_mu*	von Bertlanaffy asymptotic growth parameter
out	*K_mu*	von Bertlanaffy asymptotic growth parameter
out	*L_inf_sd*	standard deviation von Bertlanaffy asymptotic growth parameter
out	*K_sd*	standard deviation von Bertlanaffy asymptotic growth parameter

Here is the call graph for this function:

Here is the caller graph for this function:



10.3.4.6 real(dp) function growth_mod::h_mn18 (real(dp), intent(in) *x*, real(dp), intent(in) *sigma*, real(dp), intent(in) *w*)

Given (MN18 Appendix B)

Φ_N denotes the normal **cumulative** distribution function.

ϕ_N denotes the normal **density** function.

$$H_{MN18}(x,\sigma,\omega) = \frac{1}{\omega}[x\Phi_N(x,0,\sigma^2) + \sigma^2\phi_N(x,0,\sigma^2)]$$

WAS

$$H_{MN18}(x,\sigma,\omega) = \frac{1}{\omega}(x * f + \sigma^2 * f)$$

where $f = \Phi_N$

10.3.4.6.1 Parameters

in	*x*	- evaluation point
in	*sigma*	- paramaters defined within MN18
in	*w*	- paramaters defined within MN18

10.3.4.6.2 Returns

H - variable

Here is the call graph for this function:

Here is the caller graph for this function:

10.3.4.7 subroutine growth_mod::increment_mean_std (real(dp), intent(in) *l_inf_mu*, real(dp), intent(in) *k_mu*, real(dp), intent(in) *l_inf_sd*, real(dp), intent(in) *k_sd*, real(dp), intent(in) *size*, real(dp), intent(out) *mu*, real(dp), intent(out) *sigma*)

Purpose: This subroutine computes a growth increment distribution parameters under the assumption of von Bertlanaffy growth and normally distributed growth increments. It is assumed that the parameters of von BernBertlanaffy growth K and L_inf have normal distributions.

10.3.4.7.1 Parameters

in	*L_inf_mu*	[real 1x1] = mean of von Bertlanaffy asymptotic growth parameterL_inf(see HC09 eqn 1)
in	*K_mu*	[real 1x1] = mean of von Bertlanaffy growth parameter K(see HC09 eqn 1)
in	*L_inf_sd*	[real 1x1] = standard deviation of von Bertlanaffy asymptoticgrowth parameter L_inf(see HC09 eqn 1)
in	*K_sd*	[real 1x1] = standard deviation of von Bertlanaffy growth parameter (see HC09 eqn 1)
in	*size*	[real 1x1] = size to estimate increment stats
out	*mu*	[1x1] = mean of increment at size
out	*sigma*	[1x1] = standard deviation of increment at size

history: Written by keston Smith (IBSS corp) May 2021

Here is the call graph for this function:

Here is the caller graph for this function:

10.3.4.8 real(dp) function, dimension(num_size_classes, num_size_classes) growth_mod::mn18_appxc_trans_matrix (real(dp), intent(in) *l_inf_mu*, real(dp), intent(in) *k_mu*, real(dp), intent(in) *l_inf_sd*, real(dp), intent(in) *k_sd*, real(dp), dimension(num_size_classes), intent(in) *shell_lengths*)

Purpose: This subroutine computes a sizeclass transition matrix under the assumption of von Bertlanaffy growth. It is assumed that the parameters of von BernBertlanaffy growth K and L_inf have normal distributions.

From MN18 p. 1312, 1313

$$c = 1.0 - e^{-K_\mu * \delta_t}$$

$$\eta = c * L_{\infty_\mu}$$

$$\omega_k = l_k - l_{k-1}$$

$$\omega_{k_{avg}} = \frac{l_k + l_{k-1}}{2}$$

$$\Omega = (1 - c)\omega_k$$

$$X(y,k) = l_y - \eta - (1 - c)l_k$$

$$G(y,k,\sigma,\omega_k) = H_{MN18}(X(y,k-1),\sigma,\Omega) - H_{MN18}(X(y,k),\sigma,\Omega)$$

10.3.4.8.1 Parameters

in	*L_inf_mu*	[real 1x1] = mean of von Bertlanaffy asymptotic growth parameter L_inf(see HC09 eqn 1)
in	*K_mu*	[real 1x1] = mean of mean of von Bertlanaffy asymptotic growth parameter K(see HC09 eqn 1)
in	*L_inf_std*	[real 1x1] = standard deviation of von Bertlanaffy asymptotic growth parameter L_inf(see HC09 eqn 1)
in	*K_std*	[real 1x1] = standard deviation of von Bertlanaffy growth parameter K(see HC09 eqn 1)
in	*shell_lengths*	[real nx1] = shell_lengths for each size class

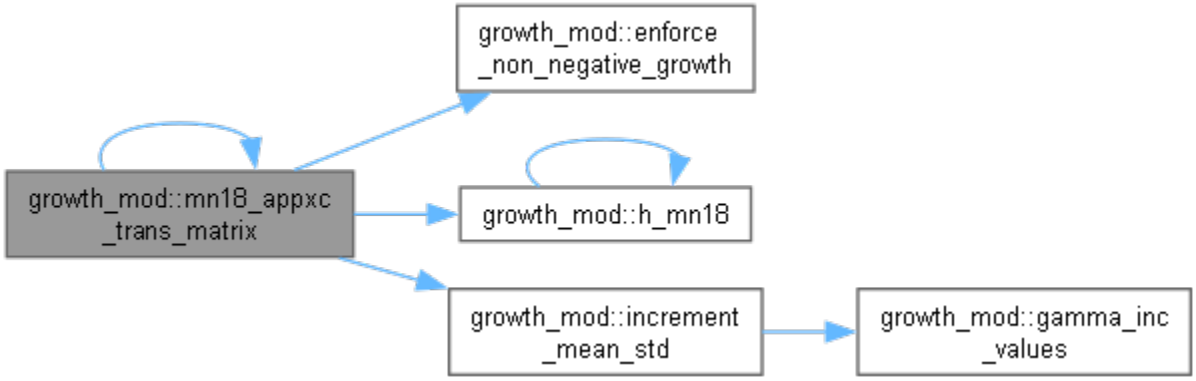
10.3.4.8.2 Returns

G [real n x n] = size transition matrix estimated under the assumption of uniform size distribution within size interval and growth distribution evaluated at mid point of size interval. Derivation is from MN18 appendix C.

Derivation of formula for growth increment mean and variance is in MN18eq7.pdf

history: Written by keston Smith (IBSS corp) May 2021

Here is the call graph for this function:



Here is the caller graph for this function:

10.3.4.9 real(dp) function growth_mod::norm_cumul_dist_fcn (real(dp), intent(in) *x*, real(dp), intent(in) *mu*, real(dp), intent(in) *sigma*)

Computation of normal cumulative distribution function.

$$\Phi(x,\mu,\sigma) = \frac{1}{2}(1 + Erf(\frac{x-\mu}{\sigma\sqrt{2}}))$$

10.3.4.9.1 Parameters

in	*mu*	- mean
in	*sigma*	- standard deviation
in	*x*	- evaluation point

10.3.4.9.2 Returns

normal cdf value at x, f(x|mu,sigma)

Here is the call graph for this function:

Here is the caller graph for this function:

10.3.4.10 real(dp) function growth_mod::norm_density_fcn (real(dp), intent(in) *x*, real(dp), intent(in) *mu*, real(dp), intent(in) *sigma*)

Computation of normal density function.

$$\phi(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

10.3.4.10.1 Parameters

in	*mu*	- mean
in	*sigma*	- standard deviation
in	*x*	- evaluation point

10.3.4.10.2 Returns

normal density function at x

Here is the call graph for this function:

Here is the caller graph for this function:

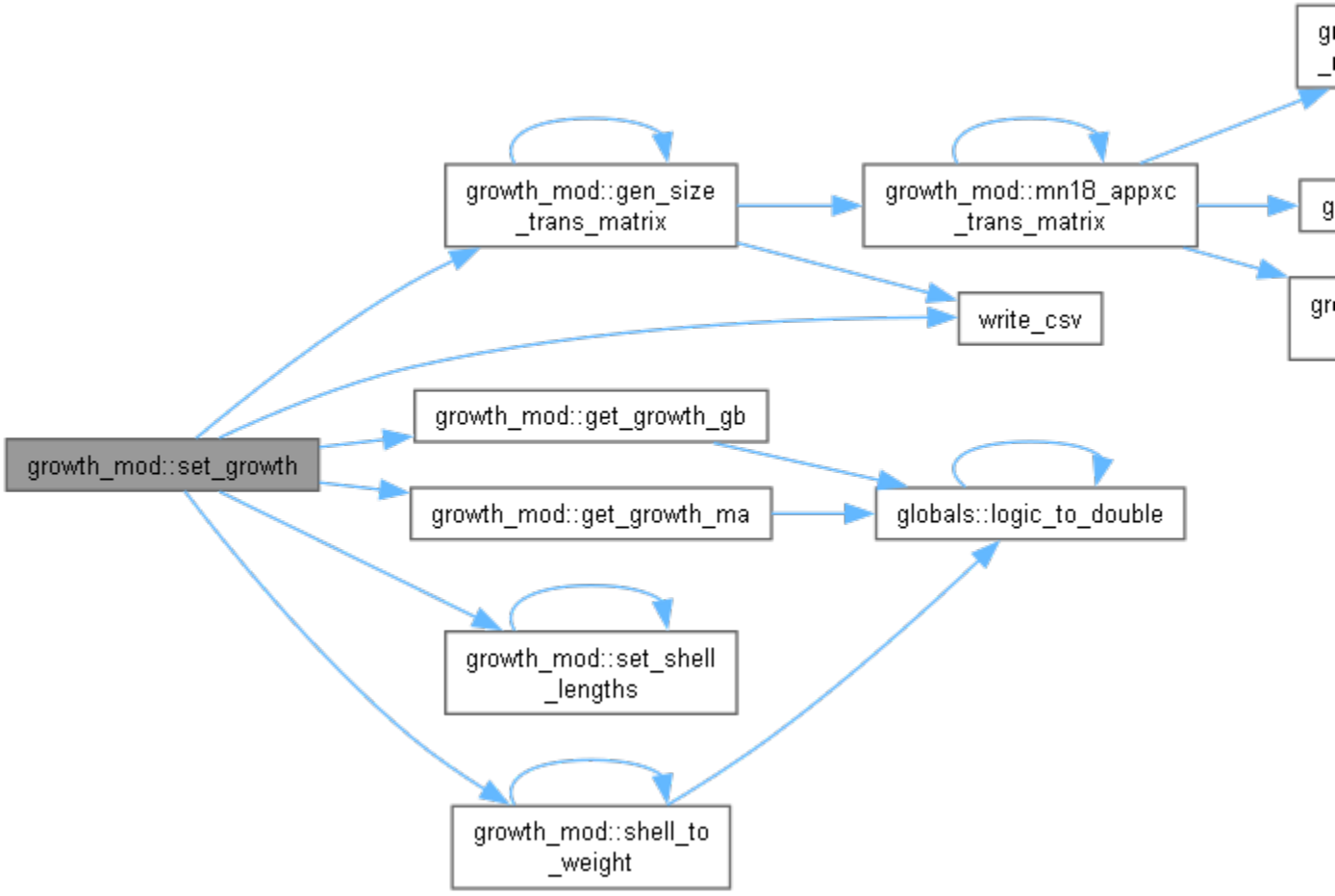
10.3.4.11 subroutine growth_mod::set_growth (type(growth_class), dimension(*), intent(inout) *growth*, type(grid_data_class), dimension(*), intent(in) *grid*, real(dp), dimension(*), intent(inout) *shell_lengths*, integer, intent(in) *num_ts*, integer, intent(in) *ts_per_year*, character(domain_len), intent(in) *dom_name*, real(dp), intent(out) *dom_area*, real(dp), dimension(1:ngrids, 1:num_size_classes), intent(inout) *state_mat*, real(dp), dimension(1:ngrids, 1:num_size_classes), intent(inout) *weight_grams*, integer, intent(in) *ngrids*)

Initializes growth for startup.

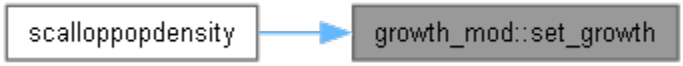
10.3.4.11.1 Parameters

in,out	*growth*	Parameters that identify how the scallop should grow
in	*grid*	Vector that identifies the geospatial locations under simulation
in,out	*shell_lengths*	Vector of the size, length, of scallops
in	*num_ts*	number of time steps per year for simulation
in	*num_sz_classes*	Number of size classes to set private member
in	*domain_name*	Name of domain being simulate, 'MA' or 'GB'
out	*domain_area,Size*	of domain under consideration in square meters
in	*file_name*	The name of the file with initial state, i.e. scallops per sq meter
in	*start_year*	Year in which to start simulation
out	*state*	Initial state as set by initial conditions
in,out	*weight_grams*	Computed combined scallop weight

Here is the call graph for this function:



Here is the caller graph for this function:



10.3.4.12 real(dp) function, dimension(num_size_classes) growth_mod::set_shell_lengths (real(dp), intent(in) *length_min*, real(dp), intent(in) *length_delta*)

setup shell shell_lengths intervals

length_min
length_min + length_delta
$length_{shell}(n) = length_{min} + (n-1) * length_{delta}$

10.3.4.12.1 Parameters

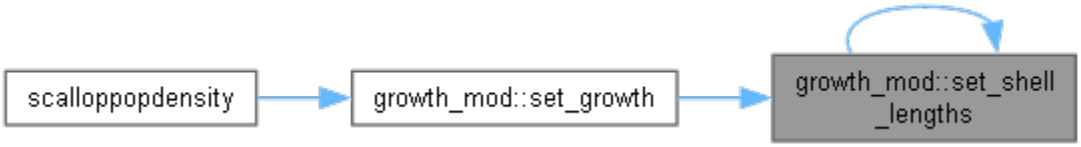
in	*length_min*	Size of smallest size class
in	*length_delta*	amount between size classes

10.3.4.12.2 Returns

shell length in millimeters

Here is the call graph for this function:

Here is the caller graph for this function:



10.3.4.13 elemental real(dp) function growth_mod::shell_to_weight (real(dp), intent(in) *shell_length_mm*, logical, intent(in) *is_closed*, real(dp), intent(in) *depth*, real(dp), intent(in) *latitude*, real(dp), intent(in) *longitude*)

Computes weight given a shell height.

For Mid-Atlantic

$$\begin{aligned} x &= -9.48 + 2.51 * log(length_{mm}) \\ &- 0.1743 - 0.059094 \\ &- 0.0033 * depth \\ &+ 0.021 * latitude \\ &- 0.031 * isClosed \\ &+ 0.00525 * log(length_{mm} * 21.0 \\ &- 0.000065 * 21.0 * depth \end{aligned}$$

For Georges Bank

$$\begin{aligned} x &= -6.69 + 2.878 * log(length_{mm}) \\ &- 0.0073 * depth \\ &- 0.073 * latitude \\ &+ 1.28 * isClosed \\ &- 0.25 * log(length_{mm}) * isClosed \end{aligned}$$

$$weight_g = e^x$$

10.3.4.13.1 Parameters

in	*shell_length_mm*	The shell height, or length, in millimeters
in	*is_closed*	Logic to indicate if grid is open (F) or closed (T) to fishing
in	*depth*	The depth of the grid in meters
in	*latitude*	Geographic coordinate
in	*domain*	

MA for Mid-Atlantic
GB for Georges Bank

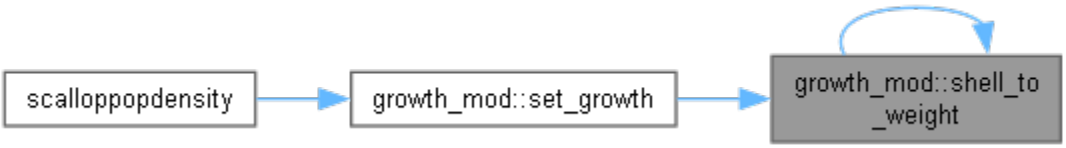
in	*ispp*	Logic to indiate is Peter Pan

10.3.4.13.2 Returns

weight in grams

Here is the call graph for this function:

Here is the caller graph for this function:



10.3.4.14 real(dp) function, dimension(num_size_classes) growth_mod::time_to_grow (integer, intent(in) *ts*, type(growth_class), intent(in) *growth*, type(mortality_class), intent(inout) *mortality*, type(recruitment_class), intent(inout) *recruit*, real(dp), dimension(*), intent(inout) *state_vector*, real(dp), intent(in) *fishing_effort*, integer, intent(in) *year*, real(dp), intent(in) *longitude*)

Computes growth in scallop population.

Computes the overall growth of the scallop population over a time period of (num_time_steps * delta_time) in units of years, typically one year with delta_time as a percent of year.

For each time step, δ_t

Computes mortality based on current state.

Computes increase in population due to recruitment, $\vec{R}$,if within recruitment months, i.e. Jan to April 10th

$$\vec{S} = \vec{S} + \delta_t \frac{\vec{R}}{RecruitDuration}$$

Adjusts population based on von Bertalanffy growth

$$\vec{S} = |G| \times \vec{S}$$

Compute overall mortality, **M**

$$\vec{M} = \vec{M}_{nat} + Fishing * \left(\vec{M}_{selectivity} + \vec{M}_{incidental} + \vec{M}_{discard}\right)$$

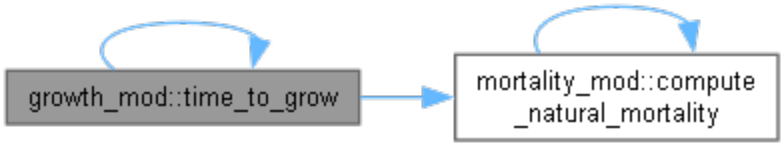
Compute new state

$$\vec{S_{t+1}} = \vec{S_t} * \left(1 - \delta_t * \vec{M}\right)$$

10.3.4.14.1 Parameters

in	*growth*	object to hold growth simulation paramters
in,out	*mortality*	object to hold mortality simulation parameters
in,out	*recruit*	object to hold recruitment simulation parameters
in,out	*state*	vector of the current state in scallops per square meter
in	*fishing_effort*	vector of fishing effort by location
out	*state_time_steps*	State at each time step
in	*start_year*	under considration

Here is the call graph for this function:



Here is the caller graph for this function:

10.3.5 Variable Documentation

10.3.5.1 real(dp), private growth_mod::delta_time [private]

10.3.5.2 real(dp), private growth_mod::domain_area_sqm [private]

10.3.5.3 character(domain_len), private growth_mod::domain_name [private]

10.3.5.4 integer, parameter growth_mod::growth_param_size = 4

10.3.5.5 integer, private growth_mod::num_grids [private]

10.3.5.6 integer, private growth_mod::num_time_steps [private]

10.3.5.7 logical, private growth_mod::show_recruits_msg [private]

10.3.5.8 integer, private growth_mod::time_steps_year [private]

10.3.5.9

10.4 mortality_mod Module Reference

10.4.1 Data Types

type **dataforplots**type **fishingmortality**
type **mortality_class**

10.4.2 Functions/Subroutines

subroutine **set_select_data** (value)
subroutine **destructor** ()
subroutine **set_mortality** (mortality, grid, shell_lengths, dom_name, dom_area, num_ts, ts_py, ngrids)
subroutine **load_fishing_mortalities** ()

Open file given by fishing_mort_fname. Reads in the year and number of entries in the list. If the number of entries exceeds the number of areas loaded by the GridManager then show the error and stop. Otherwise reads in the vectors for area list indices and for fishing mortality.

elemental real(**dp**) function **ring_size_selectivity** (shell_length, is_closed, longitude)

Purpose: Assign size class fishing selectivity based on increasing logistic function.

real(**dp**) function, dimension(**num_grids**) **set_fishing_effort** (year, ts, state_mat, weight_grams, mortality, grid)

Determines a real value of mortality due to fishing given a fishing type.

real(**dp**) function, dimension(1:**num_size_classes**) **compute_natural_mortality** (max_rec_ind, mortality, state_vector, longitude)

*Computes the total number of scallops, **S** , in millions. Then recomputes juvenile mortality as a function of S.*

elemental real(**dp**) function **set_fishing_mortality** (grid, year, use_f_loc, f_loc)

Computes Fishing Mortality.

subroutine **set_config_file_name** (fname)

Used during instantiation to set the name of the file to read to for configuration parameters.

subroutine **set_fishing_mort_file_name** (fname)
subroutine **read_configuration** ()

Read_Configuration.

subroutine **mortality_write_at_timestep** (year, ts, state_mat, weight_grams, grid)

Initializes growth for startup.

elemental real(**dp**) function **set_discard** (length, selectivity, cull_size, discard, is_closed)

Computes element of discard vector.

elemental real(**dp**) function **calc_lpue** (expl_biomass, expl_scallops)

Computes catch as pounds per day.

10.4.3 Variables

character(**fname_len**), private **config_file_name**
character(**fname_len**), private **fishing_mort_fname**
type(**fishingmortality**), dimension(**max_num_years**), private **fmort_list**
logical, private **use_spec_access_data**
integer, private **num_in_list**
integer, private **num_grids**
integer, private **num_areas**
character(**domain_len**), private **domain_name**
real(**dp**), private **domain_area_sqm**
integer, private **num_time_steps**
integer, private **ts_per_year**
real(**dp**), private **delta_time**
real(**dp**), private **fishing_mort**
real(**dp**), private **alpha_mort**
real(**dp**), private **ma_cull_size_mm**
real(**dp**), private **ma_discard**
real(**dp**), private **gb_cull_size_mm**
real(**dp**), private **gb_discard**
real(**dp**), private **ma_fselect_a**
real(**dp**), private **ma_fselect_b**
real(**dp**), private **gbc_fselect_a**
real(**dp**), private **gbc_fselect_b**
real(**dp**), private **gbo_fselect_a**
real(**dp**), private **gbo_fselect_b**
real(**dp**), private **ma_mort_adult**
real(**dp**), private **ma_incidental**
real(**dp**), private **ma_length_0**
real(**dp**), private **gb_mort_adult**
real(**dp**), private **gb_incidental**
real(**dp**), private **gb_length_0**
real(**dp**), private **lpue_slope**
real(**dp**), private **lpue_slope2**
real(**dp**), private **lpue_intercept**
integer, private **max_per_day**
real(**dp**), private **max_time_hpd**
real(**dp**), private **dredge_width_m**
real(**dp**), private **towing_speed_knots**
real(**dp**), dimension(:), allocatable, private **expl_biomass_gpsqm**
real(**dp**), dimension(:), allocatable, private **expl_scallops_psqm**
real(**dp**), dimension(:), allocatable, private **f_mort**
real(**dp**), dimension(:), allocatable, private **landings_by_num**
real(**dp**), dimension(:), allocatable, private **landings_wgt_grams**
real(**dp**), dimension(:), allocatable, private **lpue**
real(**dp**), dimension(:), allocatable, private **fishing_effort**
real(**dp**), dimension(:), allocatable, private **landings_accum**
real(**dp**), dimension(:), allocatable, private **landings_wgt_accum**
real(**dp**), dimension(:), allocatable, private **lpue_accum**
real(**dp**), dimension(**num_size_classes**), private **expl_scallops_psqm_at_size**
real(**dp**), dimension(**num_size_classes**), private **landings_at_size**
type(**dataforplots**), private **data_select**

10.4.4 Function/Subroutine Documentation

10.4.4.1 elemental real(dp) function mortality_mod::calc_lpue (real(dp), intent(in) *expl_biomass*, real(dp), intent(in) *expl_scallops*)

Computes catch as pounds per day.

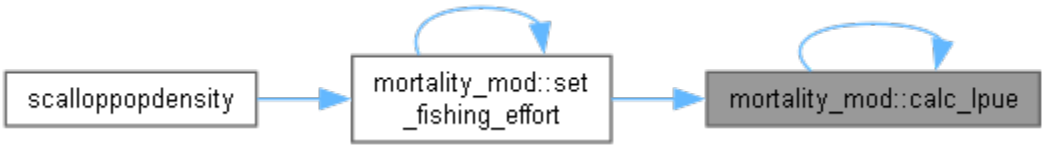
10.4.4.1.1 Parameters

in	*expl_biomass*	! Expl biomass
in	*expl_scallops*	! Expl Number of Scallops
out	*dredge_time_hrs*	! dredge bottom time
out	*dredge_area_sqnm*	! area swept per day

EBiomass/ENumber = ESize Total Weight of a Tow / Number of scallops caught = mean weight of individual scallop **expl_biomass_gpsqm(grid)** * 4516 / **expl_scallops_psqm(grid)** * 4516 = mean_expl_wght_g xxxx xxxx

Here is the call graph for this function:

Here is the caller graph for this function:



10.4.4.2 real(dp) function, dimension(1:num_size_classes) mortality_mod::compute_natural_mortality (integer, intent(in) *max_rec_ind*, type(mortality_class), intent(inout) *mortality*, real(dp), dimension(*), intent(in) *state_vector*, real(dp), intent(in) *longitude*)

Computes the total number of scallops, **S** , in millions. Then recomputes juvenile mortality as a function of S.

$$M_{juv} = \begin{cases} exp(1.093 * log(S) - 9.701), & \text{if } S > 1400 \text{ million } (2030?) \\ 0.25, & \text{otherwise} \end{cases}$$

A similar formula for GB Open:

$$M_{juv} = \begin{cases} exp((1.226 * log(S) - 10.49)), & \text{if } S > 1400 \text{ million } (2030?) \\ 0.2, & \text{otherwise} \end{cases}$$

Decreasing logistic function,

$$\alpha(length) = 1 - \frac{1}{1 + e^{-length_0[length-a]}}$$

TODO, current alpha equation is:

$$\alpha(length) = 1 - \frac{1}{1 + e^{-a*(length/10.0 - length_0)}}$$

where h_0 is 65 if MA or 70 if GB

Finally

$$M_{nat} = \alpha * M_{juv} + (1 - \alpha) M_{adult}$$

10.4.4.2.1 Parameters

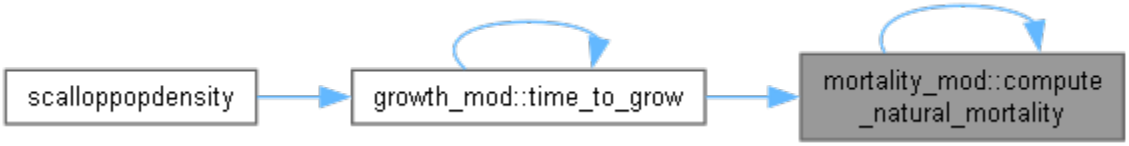
in	*recruit*	
in,out	*mortality*	
in	*state_vector*	Current state_vector of scallop population in scallops/m^2

10.4.4.2.2 Returns

natural_mortality and juvenile mortality

Here is the call graph for this function:

Here is the caller graph for this function:



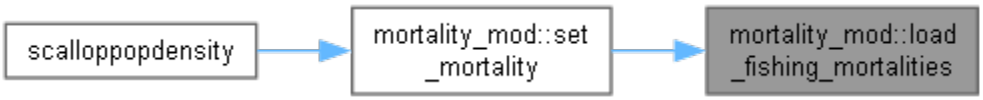
10.4.4.3 subroutine mortality_mod::destructor

Here is the caller graph for this function:

10.4.4.4 subroutine mortality_mod::load_fishing_mortalities

Open file given by fishing_mort_fname. Reads in the year and number of entries in the list. If the number of entries exceeds the number of areas loaded by the GridManager then show the error and stop. Otherwise reads in the vectors for area list indices and for fishing mortality.

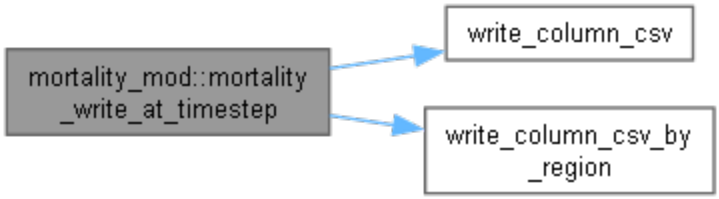
Here is the caller graph for this function:



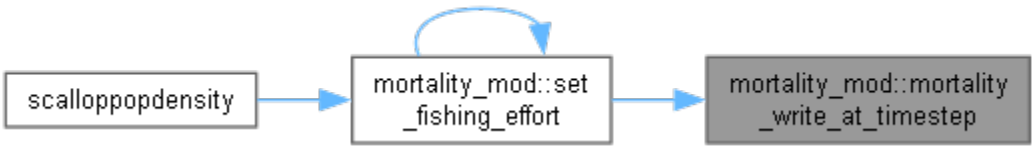
10.4.4.5 subroutine mortality_mod::mortality_write_at_timestep (integer, intent(in) *year*, integer, intent(in) *ts*, real(dp), dimension(1:num_grids, 1:num_size_classes), intent(in) *state_mat*, real(dp), dimension(1:num_grids, 1:num_size_classes), intent(in) *weight_grams*, type(grid_data_class), dimension(*), intent(in) *grid*)

Initializes growth for startup.

Here is the call graph for this function:



Here is the caller graph for this function:



10.4.4.6 subroutine mortality_mod::read_configuration

Read_Configuration.

Read Input File

Reads a configuration file, 'config_file_name.cfg', to set data parameters for Mortality

Here is the call graph for this function:

10.4.4.7 elemental real(dp) function mortality_mod::ring_size_selectivity (real(dp), intent(in) *shell_length*, logical, intent(in) *is_closed*, real(dp), intent(in) *longitude*)

Purpose: Assign size class fishing selectivity based on increasing logistic function.

$$Selectivity = \frac{1}{1 + exp(a - b * length_{shell})}$$

3.5" rings were used from 1996-2004, 3.25" rings in 1995, and 3" rings through 1994. We don't have curves for the 3 and 3.25" ring dredges. To estimate these selectivity curves, I would simply shift the 3.5" ring curve to the left by 13 (for 3" rings) or 6 mm (for 3.25" rings). The primary purpose of GEOSAMS is for forecasting, where all of this is irrelevant, to do some hindcasting as a way of testing the model, in which case getting the historical selectivity right is important.@param [in] shell_length (real(dp)) length n vector of shell lengths@param [in] a, b (real(dp)) parameters of logistic selectivity curve@param [in] year (integer) Year to determine if this is before 2005 (3.5" rings) or after (4" rings)

10.4.4.7.1 Parameters

in	*is_closed*	true if grid is closed to fishing

10.4.4.7.2 Author

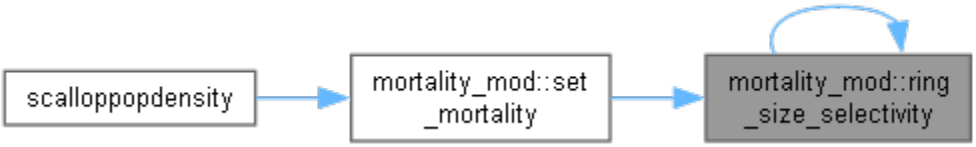
Keston Smith (IBSS corp) May 2022

10.4.4.7.3 Returns

length num_size_classes vector of selectivity

Here is the call graph for this function:

Here is the caller graph for this function:



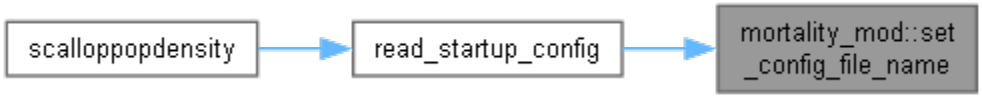
10.4.4.8 subroutine mortality_mod::set_config_file_name (character(*), intent(in) *fname*)

Used during instantiation to set the name of the file to read to for configuration parameters.

Read Input File

Sets name of a configuration file, 'config_file_name.cfg'

Here is the caller graph for this function:



10.4.4.9 elemental real(dp) function mortality_mod::set_discard (real(dp), intent(in) *length*, real(dp), intent(in) *selectivity*, real(dp), intent(in) *cull_size*, real(dp), intent(in) *discard*, logical, intent(in) *is_closed*)

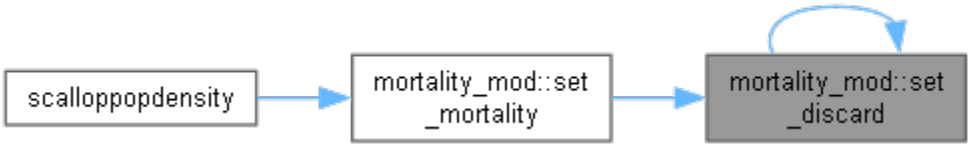
Computes element of discard vector.

10.4.4.9.1 Parameters

in	*length,vector*	element for shell length
in	*cull_size,determins*	shell length below which are discarded
in	*discard,percentage*	of selectivity that will be discarded
in	*selectivity,vector*	element that determines scallops harvested

Here is the call graph for this function:

Here is the caller graph for this function:



10.4.4.10 real(dp) function, dimension(num_grids) mortality_mod::set_fishing_effort (integer, intent(in) *year*, integer, intent(in) *ts*, real(dp), dimension(1:num_grids, 1:num_size_classes), intent(in) *state_mat*, real(dp), dimension(1:num_grids, 1:num_size_classes), intent(in) *weight_grams*, type(mortality_class),

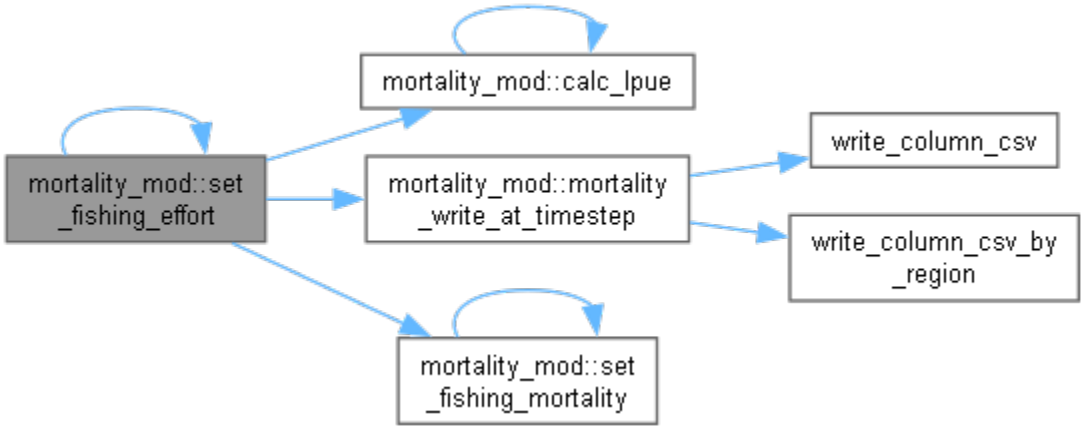
dimension(*), intent(in) *mortality*, type(grid_data_class), dimension(*), intent(in) *grid*)

Determines a real value of mortality due to fishing given a fishing type.

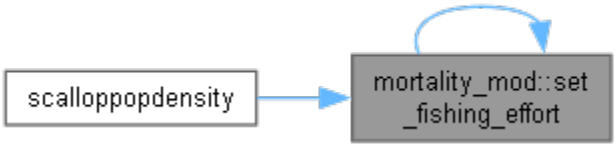
10.4.4.10.1 Parameters

in	*year*	
in	*state*	matrix|num_grids by num_size classes| current state in scallops per square meter
in	*weight_grams*	matrix|num_grids by num_size classes|
in	*mortality*	vector(num_grids) @results fishing mortality

Here is the call graph for this function:

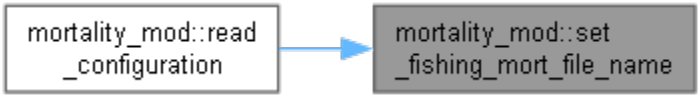


Here is the caller graph for this function:



10.4.4.11 subroutine mortality_mod::set_fishing_mort_file_name (character(*), intent(in) *fname*)

Here is the caller graph for this function:



10.4.4.12 elemental real(dp) function mortality_mod::set_fishing_mortality (type(grid_data_class), intent(in) *grid*, integer, intent(in) *year*, logical, intent(in) *use_f_loc*, real(dp), intent(in) *f_loc*)

Computes Fishing Mortality.

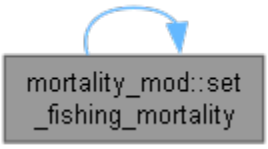
There is a year list for each year of interest, up to a total number of years of max_num_years For each list item there are two vectors.

The first vector is a list of special access by index.

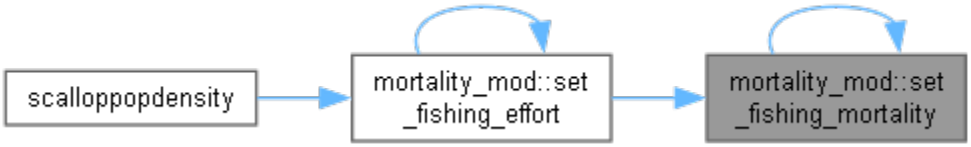
The second vector is a list of corresponding fishing mortalities for that area Thus, if the current simulation year is in the year list

Check if the grids

Here is the call graph for this function:



Here is the caller graph for this function:

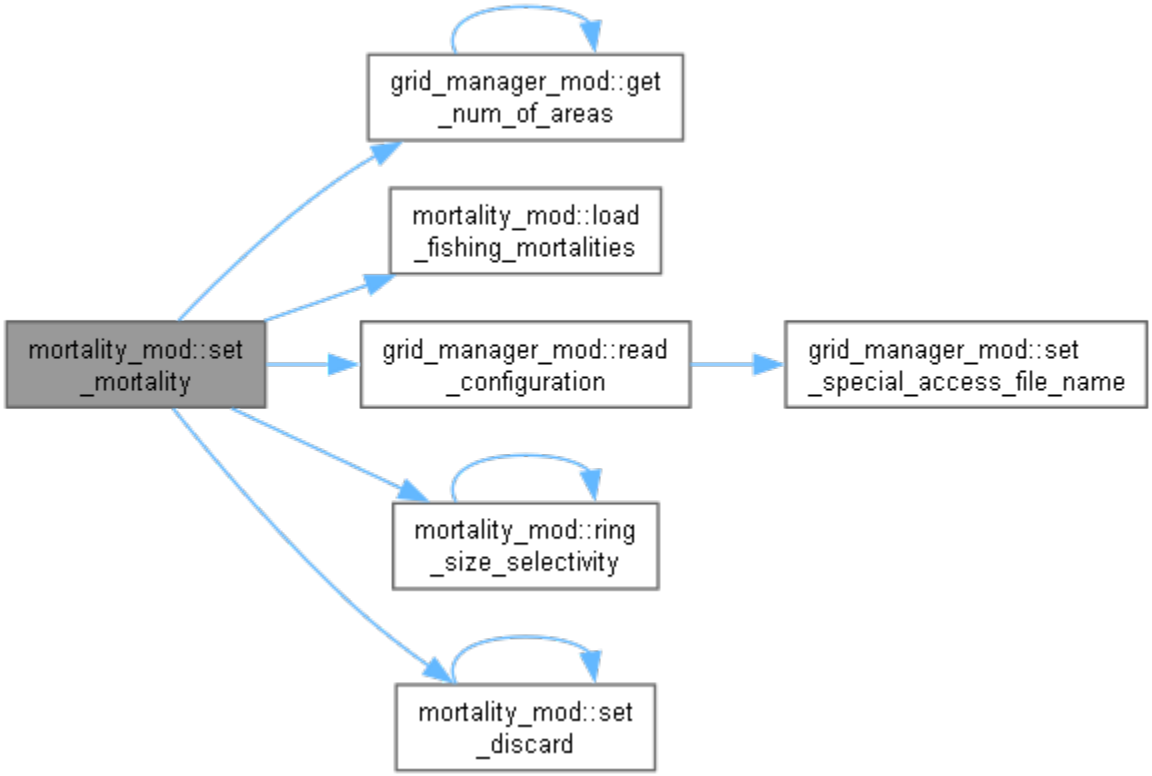


10.4.4.13 subroutine mortality_mod::set_mortality (type(mortality_class), dimension(*), intent(inout) *mortality*, type(grid_data_class), dimension(*), intent(in) *grid*, real(dp), dimension(*), intent(in) *shell_lengths*, character(domain_len), intent(in) *dom_name*, real(dp), intent(in) *dom_area*, integer, intent(in) *num_ts*, integer, intent(in) *ts_py*, integer, intent(in) *ngrids*)

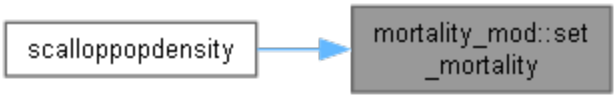
10.4.4.13.1 Parameters

in,out	*mortality*	Parameters that identify how the scallop should reaches mortality
in	*grid*	Vector that identifies the geospatial locations under simulation
in	*shell_lengths*	Vector of the size, or length, of scallops
in	*num_sz_classes*	Number of size classes to set private member
in	*domain_name*	Name of domain being simulate, 'MA' or 'GB'
in	*domain_area,Size*	of domain under consideration in square meters

Here is the call graph for this function:

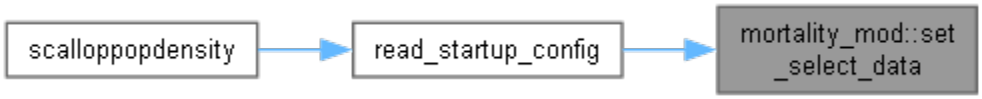


Here is the caller graph for this function:



10.4.4.14 subroutine mortality_mod::set_select_data (type(dataforplots), intent(in) *value*)

Here is the caller graph for this function:



10.4.5 Variable Documentation

10.4.5.1 real(dp), private mortality_mod::alpha_mort [private]

10.4.5.2 character(fname_len), private mortality_mod::config_file_name [private]

10.4.5.3 type(dataforplots), private mortality_mod::data_select [private]

10.4.5.4 real(dp), private mortality_mod::delta_time [private]

10.4.5.5 real(dp), private mortality_mod::domain_area_sqm [private]

10.4.5.6 character(domain_len), private mortality_mod::domain_name [private]

10.4.5.7 real(dp), private mortality_mod::dredge_width_m [private]

10.4.5.8 real(dp), dimension(:), allocatable, private mortality_mod::expl_biomass_gpsqm [private]

10.4.5.9 real(dp), dimension(:), allocatable, private mortality_mod::expl_scallops_psqm [private]

10.4.5.10 real(dp), dimension(num_size_classes), private mortality_mod::expl_scallops_psqm_at_size [private]

10.4.5.11 real(dp), dimension(:), allocatable, private mortality_mod::f_mort [private]

10.4.5.12 real(dp), dimension(:), allocatable, private mortality_mod::fishing_effort [private]

10.4.5.13 real(dp), private mortality_mod::fishing_mort [private]

10.4.5.14 character(fname_len), private mortality_mod::fishing_mort_fname [private]

10.4.5.15 type(fishingmortality), dimension(max_num_years), private mortality_mod::fmort_list [private]

10.4.5.16 real(dp), private mortality_mod::gb_cull_size_mm [private]

10.4.5.17 real(dp), private mortality_mod::gb_discard [private]

10.4.5.18 real(dp), private mortality_mod::gb_incidental [private]

10.4.5.19 real(dp), private mortality_mod::gb_length_0 [private]

10.4.5.20 real(dp), private mortality_mod::gb_mort_adult [private]

10.4.5.21 real(dp), private mortality_mod::gbc_fselect_a [private]

10.4.5.22 real(dp), private mortality_mod::gbc_fselect_b [private]

10.4.5.23 real(dp), private mortality_mod::gbo_fselect_a [private]

10.4.5.24 real(dp), private mortality_mod::gbo_fselect_b [private]

10.4.5.25 real(dp), dimension(:), allocatable, private
mortality_mod::landings_accum [private]

10.4.5.26 real(dp), dimension(num_size_classes), private
mortality_mod::landings_at_size [private]

10.4.5.27 real(dp), dimension(:), allocatable, private
mortality_mod::landings_by_num [private]

10.4.5.28 real(dp), dimension(:), allocatable, private
mortality_mod::landings_wgt_accum [private]

10.4.5.29 real(dp), dimension(:), allocatable, private
mortality_mod::landings_wgt_grams [private]

10.4.5.30 real(dp), dimension(:), allocatable, private mortality_mod::lpue [private]

10.4.5.31 real(dp), dimension(:), allocatable, private mortality_mod::lpue_accum [private]

10.4.5.32 real(dp), private mortality_mod::lpue_intercept [private]

10.4.5.33 real(dp), private mortality_mod::lpue_slope [private]

10.4.5.34 real(dp), private mortality_mod::lpue_slope2 [private]

10.4.5.35 real(dp), private mortality_mod::ma_cull_size_mm [private]

10.4.5.36 real(dp), private mortality_mod::ma_discard [private]

10.4.5.37 real(dp), private mortality_mod::ma_fselect_a [private]

10.4.5.38 real(dp), private mortality_mod::ma_fselect_b [private]

10.4.5.39 real(dp), private mortality_mod::ma_incidental [private]

10.4.5.40 real(dp), private mortality_mod::ma_length_0 [private]

10.4.5.41 real(dp), private mortality_mod::ma_mort_adult [private]

10.4.5.42 integer, private mortality_mod::max_per_day [private]

10.4.5.43 real(dp), private mortality_mod::max_time_hpd [private]

10.4.5.44 integer, private mortality_mod::num_areas `[private]`

10.4.5.45 integer, private mortality_mod::num_grids `[private]`

10.4.5.46 integer, private mortality_mod::num_in_list `[private]`

10.4.5.47 integer, private mortality_mod::num_time_steps `[private]`

10.4.5.48 real(dp), private mortality_mod::towing_speed_knots `[private]`

10.4.5.49 integer, private mortality_mod::ts_per_year `[private]`

10.4.5.50 logical, private mortality_mod::use_spec_access_data `[private]`

10.4.5.51

10.5 recruit_mod Module Reference

10.5.1 Data Types

10.5.2 type recruitment_class Functions/Subroutines

subroutine **set_recruitment** (recruit, n_grids, dom_name, dom_area, recr_yr_strt, recr_yr_stop, recruit_avg, l_inf_mu, k_mu, shell_length_mm, yr_start, yr_stop)
Set_Recruitment.

integer function **random_index** ()
Defines a weighted distribution as defined in weights.

subroutine **set_config_file_name** (fname)
Used during instantiation to set the name of the file to read to for configuration parameters.

subroutine **read_configuration** ()
Read_Configuration.

10.5.3 Variables

integer, parameter **max_n_year** = 50
character(**fname_len**), private **config_file_name**
integer, private **num_grids**
character(**domain_len**), private **domain_name**
real(**dp**), private **domain_area_sqm**
integer, private **recruit_yr_strt**
integer, private **recruit_yr_stop**
integer, private **recruit_avg_num**
integer, private **n_rand_yrs**
integer, private **sim_start_year**
integer, private **sim_stop_year**
real(**dp**), private **recr_period_start**
real(**dp**), private **recr_period_stop**
real(**dp**), dimension(:), allocatable, private **weights**
real(**dp**), private **wsum**

10.5.4 Function/Subroutine Documentation

10.5.4.1 integer function recruit_mod::random_index

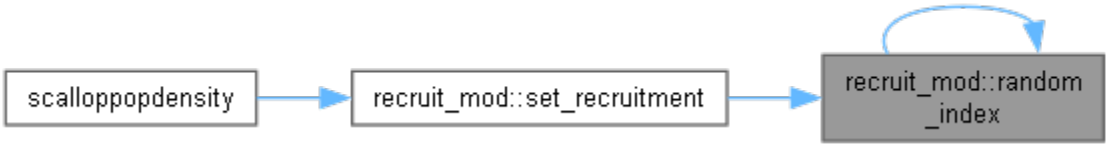
Defines a weighted distribution as defined in weights.

10.5.4.1.1 Returns

a value 1 <= x <= n_rand_yrs

Here is the call graph for this function:

Here is the caller graph for this function:



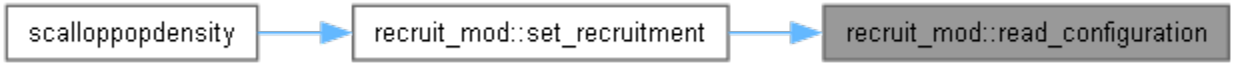
10.5.4.2 subroutine recruit_mod::read_configuration

Read_Configuration.

Read Input File

Reads a configuration file, 'config_file_name.cfg', to set data parameters for Recruitment

Here is the caller graph for this function:



10.5.4.3 subroutine recruit_mod::set_config_file_name (character(*), intent(in) *fname*)

Used during instantiation to set the name of the file to read to for configuration parameters.

Read Input File

Sets name of a configuration file, 'config_file_name.cfg'

10.5.4.4 subroutine recruit_mod::set_recruitment (type(recruitment_class), dimension(*), intent(inout) *recruit*, integer, intent(in) *n_grids*, character(domain_len), intent(in) *dom_name*, real(dp), intent(in) *dom_area*, integer, intent(out) *recr_yr_strt*, integer, intent(out) *recr_yr_stop*, integer, intent(out) *recruit_avg*, real(dp), dimension(*), intent(in) *l_inf_mu*, real(dp), dimension(*), intent(in) *k_mu*, real(dp), dimension(*), intent(in) *shell_length_mm*, integer, intent(in) *yr_start*, integer, intent(in) *yr_stop*)

Set_Recruitment.

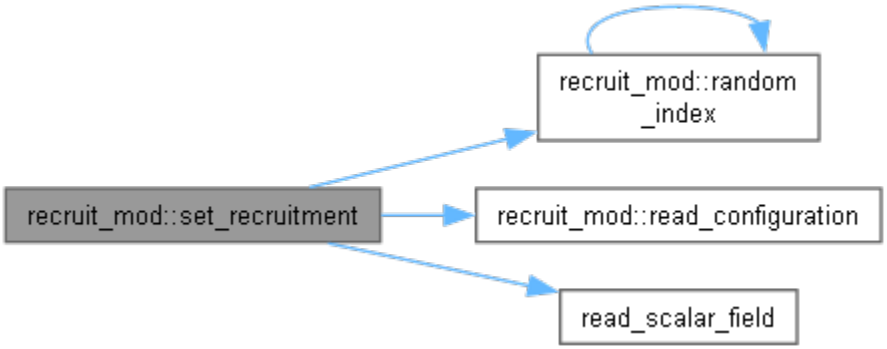
Sets recruitment parameters

10.5.4.4.1 Parameters

in,out	*recruit*	
in	*n_grids,The*	number of grids under consideration, sets private value num_grids
in	*dom_name,The*	doomain being simulated, sets private value domain_name. Should be MA MidAtlantic or GB GeorgesBank
in	*dom_area*	the total area in square meters, sets domain_area_sqm
in	*L_inf_mu*	asymptotic size, average
in	*K_mu*	Brody growth coefficient K, average
in	*shell_length_mm*	Shell height in millimeters
out	*recr_yr_strt*	year start of available data
out	*recr_yr_stop*	year stop of available data

in	*yr_start*	simulation start year
in	*yr_stop*	simulation end year

Here is the call graph for this function:



Here is the caller graph for this function:

10.5.5 Variable Documentation

10.5.5.1 character(fname_len), private recruit_mod::config_file_name [private]

10.5.5.2 real(dp), private recruit_mod::domain_area_sqm [private]

10.5.5.3 character(domain_len), private recruit_mod::domain_name [private]

10.5.5.4 integer, parameter recruit_mod::max_n_year = 50

10.5.5.5 integer, private recruit_mod::n_rand_yrs [private]

10.5.5.6 integer, private recruit_mod::num_grids [private]

10.5.5.7 real(dp), private recruit_mod::recr_period_start [private]

10.5.5.8 real(dp), private recruit_mod::recr_period_stop [private]

10.5.5.9 integer, private recruit_mod::recruit_avg_num [private]

10.5.5.10 integer, private recruit_mod::recruit_yr_stop [private]

10.5.5.11 integer, private recruit_mod::recruit_yr_strt [private]

10.5.5.12 integer, private recruit_mod::sim_start_year [private]

10.5.5.13 integer, private recruit_mod::sim_stop_year [private]

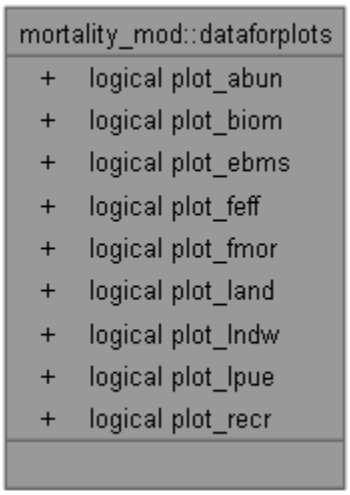
10.5.5.14 real(dp), dimension(:), allocatable, private recruit_mod::weights [private]

10.5.5.15 real(dp), private recruit_mod::wsum [private]

11 Data Type Documentation

11.1 mortality_mod::dataforplots Type Reference

Collaboration diagram for mortality_mod::dataforplots:



11.1.1 Public Attributes

logical **plot_abun**
logical **plot_biom**
logical **plot_ebms**
logical **plot_feff**
logical **plot_fmor**
logical **plot_land**
logical **plot_lndw**
logical **plot_lpue**
logical **plot_recr**

11.1.2 Member Data Documentation

11.1.2.1 logical mortality_mod::dataforplots::plot_abun

11.1.2.2 logical mortality_mod::dataforplots::plot_biom

11.1.2.3 logical mortality_mod::dataforplots::plot_ebms

11.1.2.4 logical mortality_mod::dataforplots::plot_feff

11.1.2.5 logical mortality_mod::dataforplots::plot_fmor

11.1.2.6 logical mortality_mod::dataforplots::plot_land

11.1.2.7 logical mortality_mod::dataforplots::plot_lndw

11.1.2.8 logical mortality_mod::dataforplots::plot_lpue

11.1.2.9 logical mortality_mod::dataforplots::plot_recr

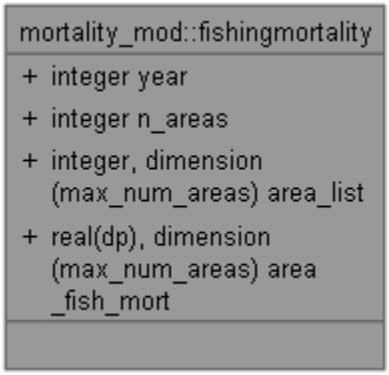
11.1.2.10 The documentation for this type was generated from the following file:

SRC/**ScallopMortality.f90**

11.1.2.11

11.2 mortality_mod::fishingmortality Type Reference

Collaboration diagram for mortality_mod::fishingmortality:



11.2.1 Public Attributes

integer **year**
integer **n_areas**
integer, dimension(**max_num_areas**) **area_list**
real(**dp**), dimension(**max_num_areas**) **area_fish_mort**

11.2.2 Member Data Documentation

11.2.2.1 real(dp), dimension(max_num_areas) mortality_mod::fishingmortality::area_fish_mort

11.2.2.2 integer, dimension(max_num_areas) mortality_mod::fishingmortality::area_list

11.2.2.3 integer mortality_mod::fishingmortality::n_areas

11.2.2.4 integer mortality_mod::fishingmortality::year

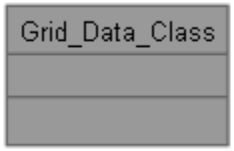
11.2.2.5 The documentation for this type was generated from the following file:

SRC/**ScallopMortality.f90**

11.2.2.6

11.3 Grid_Data_Class Module Reference

Collaboration diagram for Grid_Data_Class:

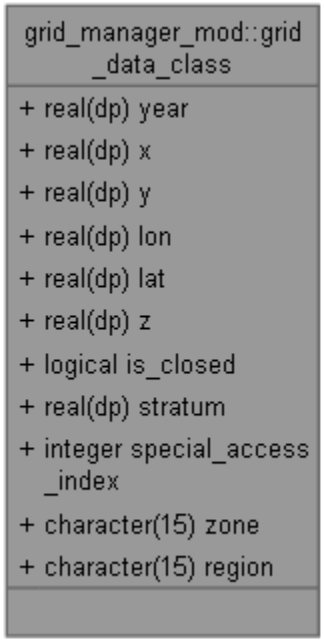


The documentation for this module was generated from the following file:

SRC/**GridManager.f90**

11.4 grid_manager_mod::grid_data_class Type Reference

Collaboration diagram for grid_manager_mod::grid_data_class:



11.4.1 Public Attributes

real(**dp**) **year**
real(**dp**) **x**
real(**dp**) **y**
real(**dp**) **lon**
real(**dp**) **lat**
real(**dp**) **z**
logical **is_closed**
real(**dp**) **stratum**
integer **special_access_index**
character(15) **zone**
character(15) **region**

11.4.2 Member Data Documentation

11.4.2.1 logical grid_manager_mod::grid_data_class::is_closed

11.4.2.2 real(dp) grid_manager_mod::grid_data_class::lat

11.4.2.3 real(dp) grid_manager_mod::grid_data_class::lon

11.4.2.4 character(15) grid_manager_mod::grid_data_class::region

11.4.2.5 integer grid_manager_mod::grid_data_class::special_access_index

11.4.2.6 real(dp) grid_manager_mod::grid_data_class::stratum

11.4.2.7 real(dp) grid_manager_mod::grid_data_class::x

11.4.2.8 real(dp) grid_manager_mod::grid_data_class::y

11.4.2.9 real(dp) grid_manager_mod::grid_data_class::year

11.4.2.10 real(dp) grid_manager_mod::grid_data_class::z

11.4.2.11 character(15) grid_manager_mod::grid_data_class::zone

---

11.4.2.12 The documentation for this type was generated from the following file:

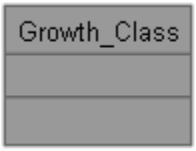
SRC/**GridManager.f90**

11.4.2.13

11.5 Growth_Class Module Reference

Subroutines that determine expected growth of scallops.

Collaboration diagram for Growth_Class:



11.5.1 Detailed Description

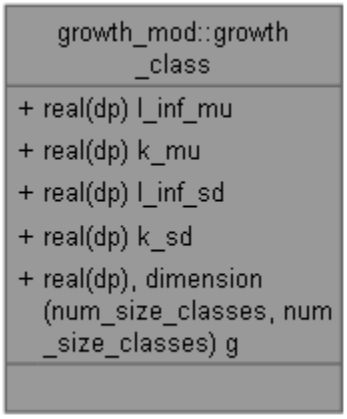
Subroutines that determine expected growth of scallops.

The documentation for this module was generated from the following file:

SRC/**ScallopGrowth.f90**

11.6 growth_mod::growth_class Type Reference

Collaboration diagram for growth_mod::growth_class:



11.6.1 Public Attributes

real(**dp**) **l_inf_mu**
Asymptotic size mean.

real(**dp**) **k_mu**
Growth coefficient mean.

real(**dp**) **l_inf_sd**
Asymptotic size standard deviation.

real(**dp**) **k_sd**
Growth coefficient standard deviation.

real(**dp**), dimension(**num_size_classes**, **num_size_classes**) **g**
Growth matrix.

11.6.2 Member Data Documentation

11.6.2.1 real(dp), dimension(num_size_classes, num_size_classes) growth_mod::growth_class::g

Growth matrix.

11.6.2.2 real(dp) growth_mod::growth_class::k_mu

Growth coefficient mean.

11.6.2.3 real(dp) growth_mod::growth_class::k_sd

Growth coefficient standard deviation.

11.6.2.4 real(dp) growth_mod::growth_class::l_inf_mu

Asymptotic size mean.

11.6.2.5 real(dp) growth_mod::growth_class::l_inf_sd

Asymptotic size standard deviation.

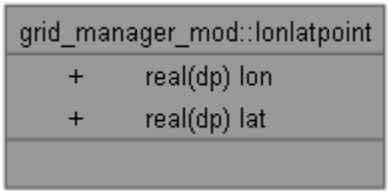
11.6.2.6 The documentation for this type was generated from the following file:

SRC/**ScallopGrowth.f90**

11.6.2.7

11.7 grid_manager_mod::lonlatpoint Type Reference

Collaboration diagram for grid_manager_mod::lonlatpoint:



11.7.1 Public Attributes

real(**dp**) **lon**
real(**dp**) **lat**

11.7.2 Member Data Documentation

11.7.2.1 real(dp) grid_manager_mod::lonlatpoint::lat

11.7.2.2 real(dp) grid_manager_mod::lonlatpoint::lon

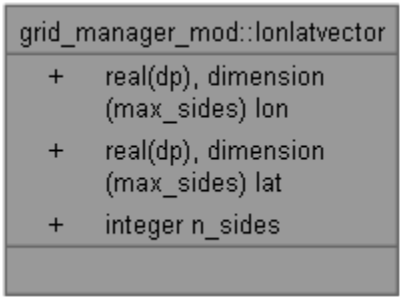
11.7.2.3 The documentation for this type was generated from the following file:

SRC/**GridManager.f90**

11.7.2.4

11.8 grid_manager_mod::lonlatvector Type Reference

Collaboration diagram for grid_manager_mod::lonlatvector:



11.8.1 Public Attributes

real(**dp**), dimension(**max_sides**) **lon**
real(**dp**), dimension(**max_sides**) **lat**
integer **n_sides**

11.8.2 Member Data Documentation

11.8.2.1 real(dp), dimension(max_sides) grid_manager_mod::lonlatvector::lat

11.8.2.2 real(dp), dimension(max_sides) grid_manager_mod::lonlatvector::lon

11.8.2.3 integer grid_manager_mod::lonlatvector::n_sides

11.8.2.4 The documentation for this type was generated from the following file:

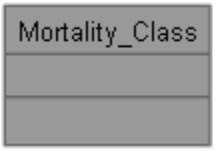
SRC/**GridManager.f90**

11.8.2.5

11.9 Mortality_Class Module Reference

Subroutines that determine expected mortality of scallops.

Collaboration diagram for Mortality_Class:



11.9.1 Detailed Description

Subroutines that determine expected mortality of scallops.

The documentation for this module was generated from the following file:

SRC/**ScallopMortality.f90**

11.10 mortality_mod::mortality_class Type Reference

Collaboration diagram for mortality_mod::mortality_class:

11.10.1 Public Attributes

real(**dp**), dimension(**num_size_classes**) **natural_mortality**
real(**dp**) **incidental**
real(**dp**), dimension(**num_size_classes**) **discard**
real(**dp**), dimension(**num_size_classes**) **selectivity**
real(**dp**) **natural_mort_adult**
real(**dp**) **natural_mort_juv**
real(**dp**), dimension(1:**num_size_classes**) **alpha**

11.10.2 Member Data Documentation

11.10.2.1 real(dp), dimension(1:num_size_classes) mortality_mod::mortality_class::alpha

11.10.2.2 real(dp), dimension(num_size_classes) mortality_mod::mortality_class::discard

11.10.2.3 real(dp) mortality_mod::mortality_class::incidental

11.10.2.4 real(dp) mortality_mod::mortality_class::natural_mort_adult

11.10.2.5 real(dp) mortality_mod::mortality_class::natural_mort_juv

11.10.2.6 real(dp), dimension(num_size_classes)
mortality_mod::mortality_class::natural_mortality

11.10.2.7 real(dp), dimension(num_size_classes)
mortality_mod::mortality_class::selectivity

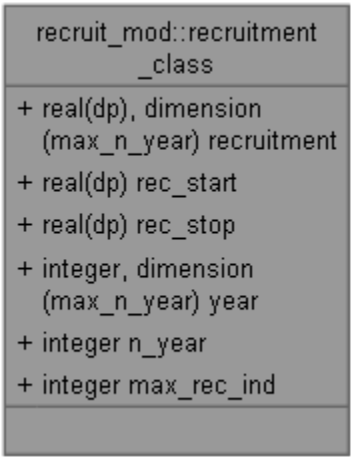
11.10.2.8 The documentation for this type was generated from the following file:

SRC/**ScallopMortality.f90**

11.10.2.9

11.11 recruit_mod::recruitment_class Type Reference

Collaboration diagram for recruit_mod::recruitment_class:



11.11.1 Public Attributes

real(**dp**), dimension(**max_n_year**) **recruitment**
real(**dp**) **rec_start**
real(**dp**) **rec_stop**
integer, dimension(**max_n_year**) **year**
integer **n_year**
integer **max_rec_ind**

11.11.2 Member Data Documentation

11.11.2.1 integer recruit_mod::recruitment_class::max_rec_ind

11.11.2.2 integer recruit_mod::recruitment_class::n_year

11.11.2.3 real(dp) recruit_mod::recruitment_class::rec_start

11.11.2.4 real(dp) recruit_mod::recruitment_class::rec_stop

11.11.2.5 real(dp), dimension(max_n_year) recruit_mod::recruitment_class::recruitment

11.11.2.6 integer, dimension(max_n_year) recruit_mod::recruitment_class::year

---

11.11.2.7 The documentation for this type was generated from the following file:

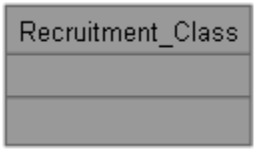
SRC/**ScallopRecruit.f90**

11.11.2.8

11.12 Recruitment_Class Module Reference

Subroutines that determine expected growth of scallops.

Collaboration diagram for Recruitment_Class:



11.12.1 Detailed Description

Subroutines that determine expected growth of scallops.

The documentation for this module was generated from the following file:

SRC/**ScallopRecruit.f90**

12 File Documentation

12.1 SRC/aaaPageOrder.f90 File Reference

12.2 SRC/Globals.f90 File Reference

12.2.1 Modules

module **globals**

12.2.2 Functions/Subroutines

elemental real(**dp**) function **globals::logic_to_double** (value)
real(**dp**) function, dimension(n, n) **globals::matrixinv** (x, n)
logical function **globals::leap_year** (year)
logical function **globals::divby** (y, val)
integer function **globals::dayofyear** (m, d)
logical function **globals::is_nan** (x)

12.2.3 Variables

integer, parameter **globals::sp** = selected_real_kind(6, 37)
integer, parameter **globals::dp** = selected_real_kind(15, 307)
integer, parameter **globals::qp** = selected_real_kind(33, 4931)
integer, parameter **globals::ndim** = 12000
integer, parameter **globals::shell_len_max** = 150
integer, parameter **globals::shell_len_min** = 30
integer, parameter **globals::shell_len_delta** = 5
integer, parameter **globals::num_size_classes** = (**shell_len_max** - **shell_len_min**) / **shell_len_delta** + 1
integer, parameter **globals::max_num_years** = 50
integer, parameter **globals::max_num_areas** = 25
integer, parameter **globals::max_sides** = 8
integer, parameter **globals::region_none** =0
integer, parameter **globals::region_n** =1
integer, parameter **globals::region_s** =2
integer, parameter **globals::region_sw** =3
integer, parameter **globals::region_w** =4
integer, parameter **globals::region_ma** =5
integer, parameter **globals::region_gbk** = 1
integer, parameter **globals::region_mab** = 5
integer, parameter **globals::tag_len** = 40
integer, parameter **globals::value_len** = 30
integer, parameter **globals::comment_len** = 80
integer, parameter **globals::line_len** = **tag_len**+**value_len**+**comment_len**
integer, parameter **globals::fname_len** = 100
integer, parameter **globals::form_len** = 20
integer, parameter **globals::input_str_len** = 100
integer, parameter **globals::csv_line_len** = 2000
integer, parameter **globals::domain_len** = 2
integer, parameter **globals::read_dev** = 69
integer, parameter **globals::write_dev** = 63
real(**dp**), parameter **globals::zero_threshold** = 1.0D-99
real(**dp**), parameter **globals::pi** = 3.14159265358979323846264338327950288D0
real(**dp**), parameter **globals::grams_per_pound** = 453.592_dp
real(**dp**), parameter **globals::meters_per_naut_mile** = 1852.D0
real(**dp**), parameter **globals::grams_per_metric_ton** = 1000000._dp
real(**dp**), parameter **globals::grid_area_sqm** = **meters_per_naut_mile****2
real(**dp**), parameter **globals::tow_area_sqm** = 4516._dp
real(**dp**), parameter **globals::one_scallop_per_tow** = 1.D0 / **tow_area_sqm**
real(**dp**), parameter **globals::ma_gb_border** = -70.5

real(**dp**), parameter **globals::days_in_year** =365+0.25-0.01+0.0025
character(*), parameter **globals::term_red** = "//achar(27)//'[31m'
character(*), parameter **globals::term_yel** = "//achar(27)//'[33m'
character(*), parameter **globals::term_grn** = "//achar(27)//'[92m'
character(*), parameter **globals::term_blu** = "//achar(27)//'[94m'
character(*), parameter **globals::term_blk** = "//achar(27)//'[0m'
character(*), parameter **globals::init_cond_dir** = 'InitialCondition/'
character(*), parameter **globals::growth_out_dir** = 'GrowthOutput/'
character(*), parameter **globals::rec_input_dir** = 'RecruitEstimates/'
character(*), parameter **globals::rec_output_dir** = 'RecruitField/'
character(*), parameter **globals::output_dir** = 'Results/'
character(*), parameter **globals::config_dir_sim** = 'Configuration/Simulation/'
character(*), parameter **globals::config_dir_interp** = 'Configuration/Interpolation/'
character(*), parameter **globals::config_dir_special** = 'Configuration/SpecialAccess/'
character(*), parameter **globals::grid_dir** = 'Grids/'
character(*), parameter **globals::data_dir** = 'Data/'
character(*), parameter **globals::anal_dir** = 'Analysis/'
integer, parameter **globals::num_regions** = 2
character(3), dimension(**num_regions**) **globals::rgn** = (/ '_GB', '_MA'/)

12.3 SRC/GridManager.f90 File Reference

12.3.1 Data Types

type **grid_manager_mod::grid_data_class**type **grid_manager_mod::lonlatpoint**
type **grid_manager_mod::lonlatvector**

12.3.2 Modules

module **grid_manager_mod**

12.3.3 Functions/Subroutines

integer function **grid_manager_mod::set_num_grids** ()

Determines the expected number of grids by simply counting the number of lines with text in the initial state file. It does not perform any error checking, only counting the number of lines with text and stopping at the first blank line.

subroutine **grid_manager_mod::set_grid_manager** (state_mat, grid, ngrids, dom_name)

Initializes growth for startup.

subroutine **grid_manager_mod::set_config_file_name** (fname)

Used during instantiation to set the name of the file to read to for configuration parameters.

subroutine **grid_manager_mod::set_init_cond_file_name** (fname)

Used during instantiation to set the name of the file to read to for grid locations, state.

subroutine **grid_manager_mod::set_special_access_file_name** (fname)

Used during instantiation to set the name of the file to special access coordinates.

integer function **grid_manager_mod::get_num_of_areas** ()

Get'r function for private member num_areas.

subroutine **grid_manager_mod::read_configuration** ()

Read_Configuration.

integer function **grid_manager_mod::load_grid_state** (grid, state_mat)

This function is used to set the grid parameters and the initial state to start the simulation.

integer function **grid_manager_mod::load_area_coordinates** ()
integer function **grid_manager_mod::is_grid_in_special_access** (lon, lat)
logical function **grid_manager_mod::point_in_polygon_points** (poly, point, nodes)
logical function **grid_manager_mod::point_in_polygon_array** (poly, point, nodes)
logical function **grid_manager_mod::point_in_polygon_vector** (polyx, polyy, x, y, nodes)

First of all, notice that each iteration considers two adjacent points and the target point. Then the if statement evaluates two conditions:

12.3.4 Variables

type(**lonlatvector**), dimension(**max_num_areas**), private **grid_manager_mod::area**
integer, private **grid_manager_mod::num_areas**
integer, private **grid_manager_mod::num_grids**
logical, private **grid_manager_mod::use_spec_access_data**
character(**domain_len**), private **grid_manager_mod::domain_name**
character(**fname_len**), private **grid_manager_mod::config_file_name**
character(**fname_len**), private **grid_manager_mod::init_cond_fname**
character(**fname_len**), private **grid_manager_mod::special_accesss_fname**

12.4 SRC/IORoutines.f90 File Reference

12.4.1 Functions/Subroutines

subroutine **read_scalar_field** (file_name, m, vector_len)
subroutine **write_2d_scalar_field** (nn, nsim, f, flnm, nndim)
Purpose: Write columns of a matrix (f) to a series of text files in exponential format. Inputs:

subroutine **write_vector_scalar_field** (vector_len, f, file_name)
subroutine **write_csv** (n, m, f, file_name, nndim, append)
Purpose: Write values of a matrix (f) to a csv file in exponential format. Inputs:

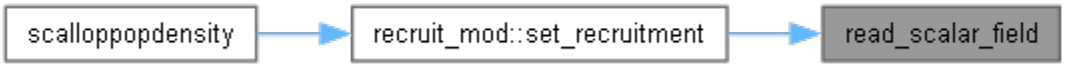
subroutine **write_column_csv** (n, f, header, file_name, append)
subroutine **read_csv** (num_rows, num_cols, file_name, m, nndim)

12.4.2 Function/Subroutine Documentation

12.4.2.1 subroutine read_csv (integer, intent(out) *num_rows*, integer, intent(in) *num_cols*, character (*), intent(in) *file_name*, real(dp), dimension(nndim,*), intent(out) *m*, integer, intent(in) *nndim*)

12.4.2.2 subroutine read_scalar_field (character(*), intent(in) *file_name*, real(dp), dimension(*), intent(out) *m*, integer, intent(inout) *vector_len*)

Here is the caller graph for this function:



12.4.2.3 subroutine write_2d_scalar_field (integer, intent(in) *nn*, integer, intent(in) *nsim*, real(dp), dimension(nndim,*), intent(in) *f*, character (*), intent(in) *flnm*, integer, intent(in) *nndim*)

Purpose: Write columns of a matrix (f) to a series of text files in exponential format. Inputs:

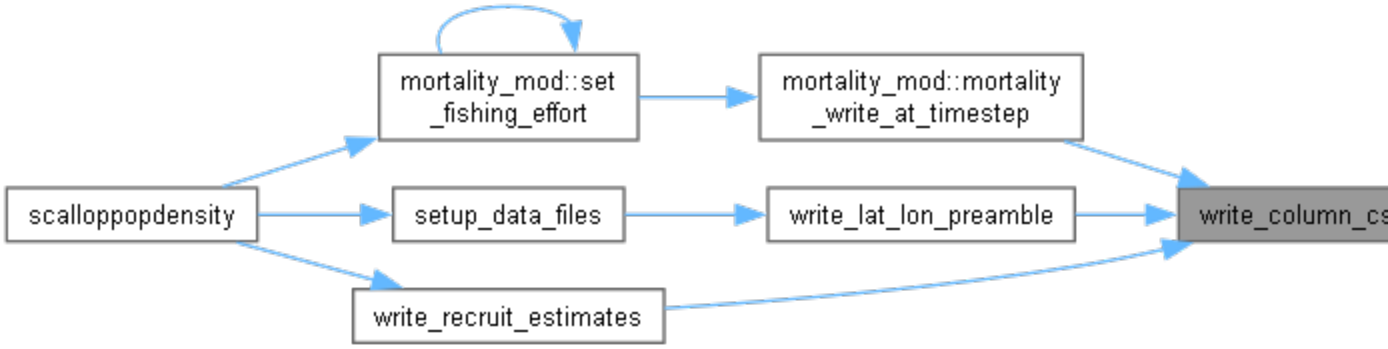
nn (integer) number of rows in f
nsim(integer) number of columns in f
f (real(dp)) values to write to text file
flnm(character(72)) filename to write f to in csv format
nndim(integer) leading dimension of f

12.4.2.3.1 Author

Keston Smith (IBSS corp) June-July 2021

12.4.2.4 subroutine write_column_csv (integer, intent(in) *n*, real(dp), dimension(*), intent(in) *f*, character(*), intent(in) *header*, character(*), intent(in) *file_name*, logical, intent(in) *append*)

Here is the caller graph for this function:



12.4.2.5 subroutine write_csv (integer, intent(in) *n*, integer, intent(in) *m*, real(dp), dimension(nndim,*), intent(in) *f*, character(*), intent(in) *file_name*, integer, intent(in) *nndim*, logical, intent(in) *append*)

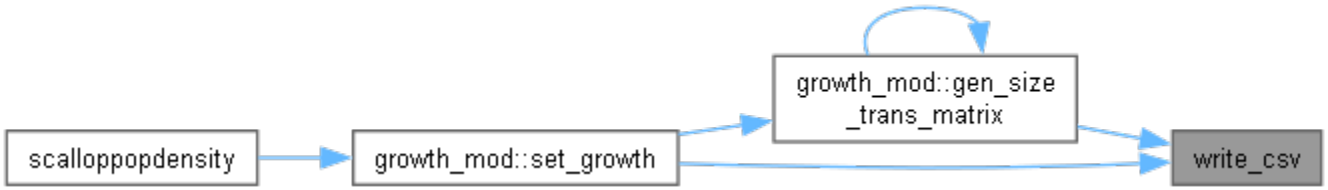
Purpose: Write values of a matrix (f) to a csv file in exponential format. Inputs:

n (integer) number of rows in f
m (integer) number of columns in f
f (real(dp)) values to write to csv file
flnm (character(72)) filename to write f to in csv format

12.4.2.5.1 Author

Keston Smith (IBSS corp) June-July 2021

Here is the caller graph for this function:



12.4.2.6 subroutine write_vector_scalar_field (integer, intent(in) *vector_len*, real(dp), dimension(*), intent(in) *f*, character (*), intent(in) *file_name*)

12.4.2.7

12.5 SRC/ScallopGrowth.f90 File Reference

12.5.1 Data Types

12.5.2 type growth_mod::growth_class Modules

module **growth_mod**

12.5.3 Functions/Subroutines

subroutine **growth_mod::set_growth** (growth, grid, shell_lengths, num_ts, ts_per_year, dom_name, dom_area, state_mat, weight_grams, ngrids)

Initializes growth for startup.

real(**dp**) function, dimension(1:**num_size_classes**, 1:**num_size_classes**) **growth_mod::gen_size_trans_matrix** (l_inf_mu, l_inf_sd, k_mu, k_sd, shell_lengths, method)

Transition matrix used to determine the growth of the scallop. The equations are based on MN18, Appendix C.

real(**dp**) function, dimension(**num_size_classes**) **growth_mod::set_shell_lengths** (length_min, length_delta)

setup shell shell_lengths intervals

subroutine **growth_mod::get_growth_gb** (depth, lat, is_closed, l_inf_mu, k_mu, l_inf_sd, k_sd)

Provides growth parameters L and K parameters for Georges Bank. From R code sent by Dvora July 28th 2021.

subroutine **growth_mod::get_growth_ma** (depth, lat, is_closed, l_inf_mu, k_mu, l_inf_sd, k_sd)

Provides growth parameters L and K parameters for Mid Atlantic From R code sent by Dvora July 28th 2021.

real(**dp**) function, dimension(**num_size_classes**, **num_size_classes**) **growth_mod::mn18_appxc_trans_matrix** (l_inf_mu, k_mu, l_inf_sd, k_sd, shell_lengths)

Purpose: This subroutine computes a sizeclass transition matrix under the assumption of von Bertlanaffy growth. It is assumed that the parameters of von BernBertlanaffy growth K and L_inf have normal distributions.

subroutine **growth_mod::increment_mean_std** (l_inf_mu, k_mu, l_inf_sd, k_sd, size, mu, sigma)

Purpose: This subroutine computes a growth increment distribution parameters under the assumption of von Bertlanaffy growth and normally distributed growth increments. It is assumed that the parameters of von BernBertlanaffy growth K and L_inf have normal distributions.

real(**dp**) function **growth_mod::h_mn18** (x, sigma, w)

Given (MN18 Appendix B)

real(**dp**) function **growth_mod::norm_cumul_dist_fcn** (x, mu, sigma)

Computation of normal cumulative distribution function.

real(**dp**) function **growth_mod::norm_density_fcn** (x, mu, sigma)

Computation of normal density function.

subroutine **growth_mod::enforce_non_negative_growth** (g)

real(**dp**) function, dimension(**num_size_classes**) **growth_mod::time_to_grow** (ts, growth, mortality, recruit, state_vector, fishing_effort, year, longitude)

Computes growth in scallop population.

elemental real(**dp**) function **growth_mod::shell_to_weight** (shell_length_mm, is_closed, depth, latitude, longitude)

Computes weight given a shell height.

subroutine **growth_mod::gamma_inc_values** (n_data, a, x, fx)

12.5.4 Variables

integer, parameter **growth_mod::growth_param_size** = 4
integer, private **growth_mod::num_grids**
character(**domain_len**), private **growth_mod::domain_name**
real(**dp**), private **growth_mod::domain_area_sqm**
integer, private **growth_mod::num_time_steps**
integer, private **growth_mod::time_steps_year**
real(**dp**), private **growth_mod::delta_time**
logical, private **growth_mod::show_recruits_msg**

12.6 SRC/ScallopMortality.f90 File Reference

12.6.1 Data Types

type **mortality_mod::mortality_class**type **mortality_mod::fishingmortality**
type **mortality_mod::dataforplots**

12.6.2 Modules

module **mortality_mod**

12.6.3 Functions/Subroutines

subroutine **mortality_mod::set_select_data** (value)
subroutine **mortality_mod::destructor** ()
subroutine **mortality_mod::set_mortality** (mortality, grid, shell_lengths, dom_name, dom_area, num_ts, ts_py, ngrids)
subroutine **mortality_mod::load_fishing_mortalities** ()

Open file given by fishing_mort_fname. Reads in the year and number of entries in the list. If the number of entries exceeds the number of areas loaded by the GridManager then show the error and stop. Otherwise reads in the vectors for area list indices and for fishing mortality.

elemental real(**dp**) function **mortality_mod::ring_size_selectivity** (shell_length, is_closed, longitude)

Purpose: Assign size class fishing selectivity based on increasing logistic function.

real(**dp**) function, dimension(**num_grids**) **mortality_mod::set_fishing_effort** (year, ts, state_mat, weight_grams, mortality, grid)

Determines a real value of mortality due to fishing given a fishing type.

real(**dp**) function, dimension(1:**num_size_classes**) **mortality_mod::compute_natural_mortality** (max_rec_ind, mortality, state_vector, longitude)

*Computes the total number of scallops, **S** , in millions. Then recomputes juvenile mortality as a function of S.*

elemental real(**dp**) function **mortality_mod::set_fishing_mortality** (grid, year, use_f_loc, f_loc)

Computes Fishing Mortality.

subroutine **mortality_mod::set_config_file_name** (fname)

Used during instantiation to set the name of the file to read to for configuration parameters.

subroutine **mortality_mod::set_fishing_mort_file_name** (fname)
subroutine **mortality_mod::read_configuration** ()

Read_Configuration.

subroutine **mortality_mod::mortality_write_at_timestep** (year, ts, state_mat, weight_grams, grid)

Initializes growth for startup.

elemental real(**dp**) function **mortality_mod::set_discard** (length, selectivity, cull_size, discard, is_closed)

Computes element of discard vector.

elemental real(**dp**) function **mortality_mod::calc_lpue** (expl_biomass, expl_scallops)

Computes catch as pounds per day.

12.6.4 Variables

character(**fname_len**), private **mortality_mod::config_file_name**
character(**fname_len**), private **mortality_mod::fishing_mort_fname**
type(**fishingmortality**), dimension(**max_num_years**), private **mortality_mod::fmort_list**
logical, private **mortality_mod::use_spec_access_data**
integer, private **mortality_mod::num_in_list**
integer, private **mortality_mod::num_grids**
integer, private **mortality_mod::num_areas**
character(**domain_len**), private **mortality_mod::domain_name**
real(**dp**), private **mortality_mod::domain_area_sqm**
integer, private **mortality_mod::num_time_steps**
integer, private **mortality_mod::ts_per_year**
real(**dp**), private **mortality_mod::delta_time**
real(**dp**), private **mortality_mod::fishing_mort**
real(**dp**), private **mortality_mod::alpha_mort**
real(**dp**), private **mortality_mod::ma_cull_size_mm**
real(**dp**), private **mortality_mod::ma_discard**
real(**dp**), private **mortality_mod::gb_cull_size_mm**
real(**dp**), private **mortality_mod::gb_discard**
real(**dp**), private **mortality_mod::ma_fselect_a**
real(**dp**), private **mortality_mod::ma_fselect_b**
real(**dp**), private **mortality_mod::gbc_fselect_a**
real(**dp**), private **mortality_mod::gbc_fselect_b**
real(**dp**), private **mortality_mod::gbo_fselect_a**
real(**dp**), private **mortality_mod::gbo_fselect_b**
real(**dp**), private **mortality_mod::ma_mort_adult**
real(**dp**), private **mortality_mod::ma_incidental**
real(**dp**), private **mortality_mod::ma_length_0**
real(**dp**), private **mortality_mod::gb_mort_adult**
real(**dp**), private **mortality_mod::gb_incidental**
real(**dp**), private **mortality_mod::gb_length_0**
real(**dp**), private **mortality_mod::lpue_slope**
real(**dp**), private **mortality_mod::lpue_slope2**
real(**dp**), private **mortality_mod::lpue_intercept**
integer, private **mortality_mod::max_per_day**
real(**dp**), private **mortality_mod::max_time_hpd**
real(**dp**), private **mortality_mod::dredge_width_m**
real(**dp**), private **mortality_mod::towing_speed_knots**
real(**dp**), dimension(:), allocatable, private **mortality_mod::expl_biomass_gpsqm**
real(**dp**), dimension(:), allocatable, private **mortality_mod::expl_scallops_psqm**
real(**dp**), dimension(:), allocatable, private **mortality_mod::f_mort**
real(**dp**), dimension(:), allocatable, private **mortality_mod::landings_by_num**
real(**dp**), dimension(:), allocatable, private **mortality_mod::landings_wgt_grams**
real(**dp**), dimension(:), allocatable, private **mortality_mod::lpue**
real(**dp**), dimension(:), allocatable, private **mortality_mod::fishing_effort**
real(**dp**), dimension(:), allocatable, private **mortality_mod::landings_accum**
real(**dp**), dimension(:), allocatable, private **mortality_mod::landings_wgt_accum**
real(**dp**), dimension(:), allocatable, private **mortality_mod::lpue_accum**
real(**dp**), dimension(**num_size_classes**), private **mortality_mod::expl_scallops_psqm_at_size**

real(**dp**), dimension(**num_size_classes**), private **mortality_mod::landings_at_size**
type(**dataforplots**), private **mortality_mod::data_select**

12.7 SRC/ScallopPopDensity.f90 File Reference

12.7.1 Functions/Subroutines

program **scalloppopdensity**
subroutine **read_startup_config** (time_steps_per_year, start_year, stop_year, domain_name, plot_data_sel)
Read Input File.

subroutine **write_lat_lon_preamble** (num_grids, grid, fname)
Writes lat and lon columns with headers to named file.

subroutine **write_x_y_preamble** (num_grids, grid, yr_offset, fname)
Writes year, UTM-X, UTM-Y, and Depth columns with headers to named file.

subroutine **write_column_csv_by_region** (n, f, c, lon, header, file_name, append, use_c)
Inputs: n (integer) number of rows in f m (integer) number of columns in f header string to write as a column header f (real(dp)) values to write to csv file file_name (character(72)) filename to write f to in csv format.

subroutine **setup_data_files** (plot_data_sel, num_grids, grid, domain_name, start_year, stop_year)
subroutine **write_recruit_estimates** (ts, ts_per_year, num_grids, grid, domain_name, year, start_year, recruit)

12.7.2 Function/Subroutine Documentation

12.7.2.1 subroutine read_startup_config (integer, intent(out) *time_steps_per_year*, integer, intent(out) *start_year*, integer, intent(out) *stop_year*, character(domain_len), intent(out) *domain_name*, type(dataforplots), intent(out) *plot_data_sel*)

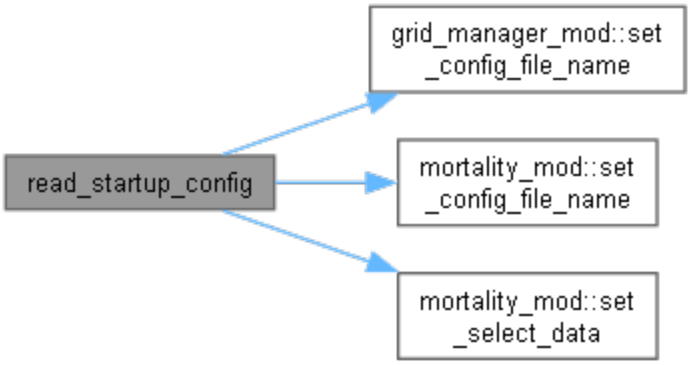
Read Input File.

Reads a configuration file, 'Scallop.inp', to set data parameters for simulation

12.7.2.1.1 Parameters

out	*domain_name*	can be either MA MidAtlantic or GB GeorgesBank
out	*init_cond_file_name*	File name that contains intial simulation conditions
out	*start_year*	Starting year for simulation read from config file
out	*stop_year*	End year for simulation read from config file
out	*time_steps_per_year*	Number of times steps to evaluate growth
out	*num_monte_carlo_iter*	Number of iterations for Monte Carlo simulation

Here is the call graph for this function:



Here is the caller graph for this function:

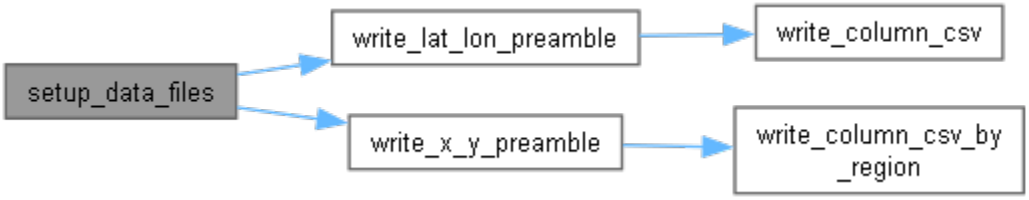
12.7.2.2 program scalloppopdensity

Here is the call graph for this function:


grid_manager_mod::set
_config_file_name
mortality_mod::set
_config_file_name
mortality_mod::destructor
read_startup_config
mortality_mod::set
_select_data
mortality_mod::calc_lpue
mortality_mod::set
_fishing_mortality
mortality_mod::set
_fishing_effort
mortality_mod::mortality
_write_at_timestep
write_x_y_preamble
write_recruit_estimates
setup_data_files
write_lat_lon_preamble
grid_manager_mod::is
_grid_in_special_access
grid_manager_mod::load
_area_coordinates
grid_manager_mod::set
_grid_manager
grid_manager_mod::load
_grid_state
grid_manager_mod::read
_configuration
scalloppopdensity

12.7.2.3 subroutine setup_data_files (type(dataforplots), intent(in) *plot_data_sel*, integer, intent(in) *num_grids*, type(grid_data_class), dimension(*), intent(in) *grid*, character(domain_len), intent(out) *domain_name*, integer, intent(in) *start_year*, integer, intent(in) *stop_year*)

Here is the call graph for this function:

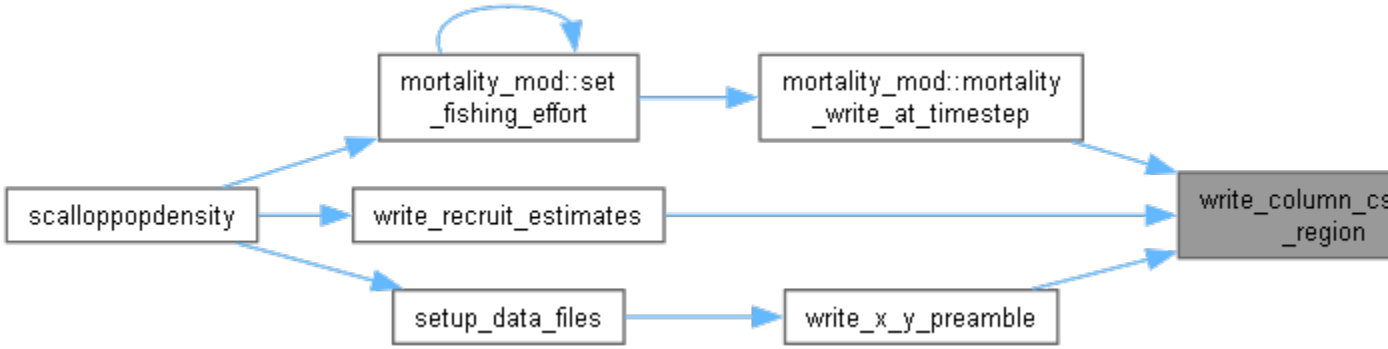


Here is the caller graph for this function:

12.7.2.4 subroutine write_column_csv_by_region (integer, intent(in) *n*, real(dp), dimension(*), intent(in) *f*, character(15), dimension(*), intent(in) *c*, real(dp), dimension(*), intent(in) *lon*, character(*), intent(in) *header*, character(*), intent(in) *file_name*, logical, intent(in) *append*, logical, intent(in) *use_c*)

Inputs: n (integer) number of rows in f m (integer) number of columns in f header string to write as a column header f (real(dp)) values to write to csv file file_name (character(72)) filename to write f to in csv format.

Here is the caller graph for this function:

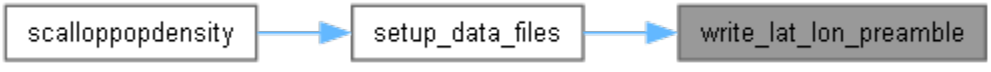


12.7.2.5 subroutine write_lat_lon_preamble (integer, intent(in) *num_grids*, type(grid_data_class), dimension(*), intent(in) *grid*, character(*), intent(in) *fname*)

Writes lat and lon columns with headers to named file.

Here is the call graph for this function:

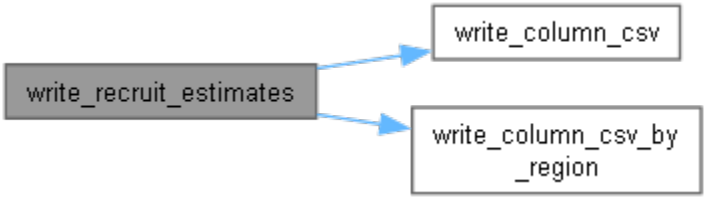
Here is the caller graph for this function:



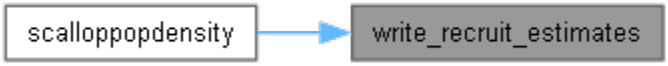
12.7.2.6 subroutine write_recruit_estimates (integer, intent(in) *ts*, integer, intent(in) *ts_per_year*, integer, intent(in) *num_grids*, type(grid_data_class), dimension(*),

intent(in) *grid*, character(domain_len), intent(out) *domain_name*, integer, intent(in) *year*, integer, intent(in) *start_year*, type(recruitment_class), dimension(*), intent(in) *recruit*)

Here is the call graph for this function:



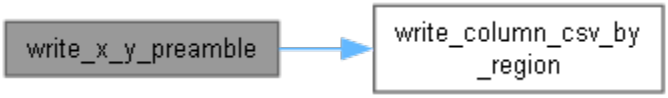
Here is the caller graph for this function:



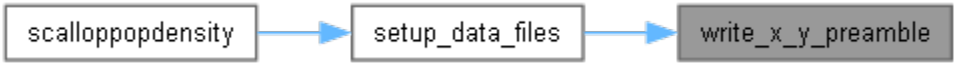
12.7.2.7 subroutine write_x_y_preamble (integer, intent(in) *num_grids*, type(grid_data_class), dimension(*), intent(in) *grid*, real(dp), intent(in) *yr_offset*, character(*), intent(in) *fname*)

Writes year, UTM-X, UTM-Y, and Depth columns with headers to named file.

Here is the call graph for this function:



Here is the caller graph for this function:



12.7.2.8

12.8 SRC/ScallopRecruit.f90 File Reference

12.8.1 Data Types

12.8.2 type recruit_mod::recruitment_class**Modules**

module **recruit_mod**

12.8.3 Functions/Subroutines

subroutine **recruit_mod::set_recruitment** (recruit, n_grids, dom_name, dom_area, recr_yr_strt, recr_yr_stop, recruit_avg, l_inf_mu, k_mu, shell_length_mm, yr_start, yr_stop)

Set_Recruitment.

integer function **recruit_mod::random_index** ()

Defines a weighted distribution as defined in weights.

subroutine **recruit_mod::set_config_file_name** (fname)

Used during instantiation to set the name of the file to read to for configuration parameters.

subroutine **recruit_mod::read_configuration** ()

Read_Configuration.

12.8.4 Variables

integer, parameter **recruit_mod::max_n_year** = 50
character(**fname_len**), private **recruit_mod::config_file_name**
integer, private **recruit_mod::num_grids**
character(**domain_len**), private **recruit_mod::domain_name**
real(**dp**), private **recruit_mod::domain_area_sqm**
integer, private **recruit_mod::recruit_yr_strt**
integer, private **recruit_mod::recruit_yr_stop**
integer, private **recruit_mod::recruit_avg_num**
integer, private **recruit_mod::n_rand_yrs**
integer, private **recruit_mod::sim_start_year**
integer, private **recruit_mod::sim_stop_year**
real(**dp**), private **recruit_mod::recr_period_start**
real(**dp**), private **recruit_mod::recr_period_stop**
real(**dp**), dimension(:), allocatable, private **recruit_mod::weights**
real(**dp**), private **recruit_mod::wsum**

13 Index