Scallop Growth Density

0.1

Generated by Doxygen 1.9.8

Chapter 1

Scallop Population Density

This program is used to compute Scallop Density after a given growth period

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
- Save By Stratum: Used in GB to make break up region into smoother shapes, i.e. rather than clover leaf.

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 Abbreviation
  - MA, or
  - GB

1.1.2 Instantiate Growth Module

The simulation then instantiates parameters that define how growth occurs

1.1.2.1 Load Grid and Initial State

The initial state is defined by a hardcode 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.

1.1.2.2 For each class: Define shell_lengths weight conversion

Shell Length

Starting at 30mm to 150mm inclusive, in 5 mm steps.
That is (150 - 30) / 5 + 1, or 25 size classes

Weigth in grams

GB

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

MA

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

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

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

- L_{∞_μ}
- L_{∞_σ}
- K_μ
- K_σ

1.1.2.4 Compute G matrix for given growth parameters

From MN18 p. 1312, 1313

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

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

For each size class, *k*

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

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

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

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

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

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

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

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

1.1.3 Instantiate Recruitment

The simulation next instantiates how recruitment will be handled.

1.1.3.1 Recruitment data

For years start_year to stop_year

Data is read in from RecruitEstimates/RecruitEstimateDNYYYY.txt

1.1.3.2 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
  - year(year_index) = year
  - rec_start = Start Period, typically 0/365, or January 1st
  - rec_stop = Stop Period, typically 100/365, or April 10

1.1.3.3 It then quantizes recruitment,

For each grid, n

- L30mm = ($L_{\infty_{\mu}}$(n) - 30) $*$ exp(- K_{μ}(n))
- For each class, j
  - If (length(n) $<=$ L30mm) recruit(n).max_rec_ind = j

1.1.4 Instantiate Mortality

The simulation next sets mortality values.

Table 1.1 Mortality

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

1.1.4.1 Compute alpha

$$\alpha(l) = 1 - \frac{1}{1 + e^{-(l-l_0)/10.0}}$$

1.1.4.2 Compute Fishing Effort

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

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

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

Compute landings by weight

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

This is also considered total_catch

Total catch is used compute fishing effort

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

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

CAS Fishing Effort

Fishing effort is defined by year and region from past history *Data/FYrGBcGBoMA.csv*. Otherwise, fishing effort is computed.

Table 1.2 Fishing Effort (partial)

Year	GB Closed	GB Open	MA
2005	0.14	0.36	0.55
2006	0.24	0.94	0.25
2007	0.15	0.76	0.5
2008	0.07	0.73	0.57
2009	0.05	0.55	0.61
2010	0.09	0.28	0.53
2011	0.21	0.19	0.54
2012	0.31	0.44	0.43
2013	0.1	0.85	0.26
2014	0.07	0.48	0.33
2015	0	0.75	0.36
2016	0	0.51	0.4
2017	0.11	0.17	0.34

1.2 Main Loop

1.2.1 For each time step

1.2.1.1 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.

1.2.1.2 For each grid

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:

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

Georges Bank:

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

where M_{adult} is 0.25 if MA or 0.2 if GB
Finally

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

Adjust population state based on von Bertalanffy growth

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

Compute increase in population due to recruitment, R

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

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

Compute Overall Mortality

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

Compute effect of mortality to arrive at new state

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

Chapter 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 = 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.

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

WAS

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

where $f = \Phi_N$

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

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

Chapter 3

Recruit_Mod

3.1 Recruitment Class

Recruitment is treated as a spatially correlated random variable. Recruit estimates at each node are read in from files stored in directories RecruitEstimates/RecruitEstimateDNYYYY.txt, where DN is ['MA', 'GB'] and YY is the year 1979 - 2019
Within the Population Dynamics portion

3.1.1 Implementation of random recruitment

Spatial fields of recruitment are generated by the software located in directory "∼/UKsrc". The output files, such as: "KrigingEstimates/SimDNYYYY/RandomFieldxx.txt" contain n_{nodes} vectors of independent random fields conditioned on survey observations from "Data/RecruitsYYYYDN.csv", where xx runs from 1 to N_{rand} and $year$ 1979 to 2018.

3.1.2 Interpolation Algorithm

The interpolation of recruit data is carried out with a Universal Kriging (UK) algorithm allowing for sampling from the posterior distribution.

3.1.2.1 Universal Kriging

Universal Kriging (UK) is a generalization of ordinary kriging in which a set of spatial functions are used to model the trend of a set of point observations. The underlying model is:

$$f(x, y, H(x, y), \lambda) = \sum_{k=1}^{n_f} f_k(x, y, H(x, y), \lambda_k) + \epsilon(x, y)$$

where f_k are the known spatial functions and $\epsilon(x, y)$ is a zero mean, spatially correlated, stationary random process with semi-variogram $\gamma(s)$. For a summary of UK see Cressie 1993, pages 151 -180. The spatially variable x here is taken to include latitude, longitude and, bathymetric depth ($x = [lat, lon, z(lat, lon)]$).

3.1.2.2 Spatial functions

The spatial functions (SF) used here are a set of one dimensional, bounded, C-infinity functions with two parameters, \ Gaussian Bump:

$$f_a(s, \lambda, x_0) = \exp(-\left(\frac{s - x_0}{\lambda}\right)^2)$$

Logistic curve:

$$f_b(s,\lambda,x_0)=\frac{1}{1+\exp(-\frac{s-x_0}{\lambda})}$$

"Sin Exp" curve: \iline 48 \iline 49 _form#79@_fakenl@_fakenl "Cos Exp" curve:

$$f_c(s,\lambda,x_0)=\cos(\frac{s-x_0}{\lambda})\exp(-\left(\frac{s-x_0}{\lambda}\right)^2)$$

In all of the function form λ controls the width of the transition and x_0 the transition point.
After fitting these to the bathymetric variable (H) we can introduce interaction. Allowing interaction terms for the spatial functions depending on bathymetry only we can define, $g_j(x,H,\lambda^j,{x_0}^j,\lambda_k,{x_0}^k)=f_j(x)f_k(H)$

$$f(x,y,H)=\sum_i f_i(H,\lambda^i,z_0^i)+\sum_j f_{j_x}(x,\lambda^{j_x},x_0^{j_x})f_k(z,\lambda^k,x_0^k)+\sum_j f_{j_y}(y,\lambda^{j_y},x_0^{j_y})f_k(z,\lambda^k,x_0^k)$$

Some parametric functions for spatial fitting on the continental shelf. Here z is bathymetric depth. We start by fitting nonlinear parameters $\lambda^{c,s}$ and $x_0^{c,s}$ to log recruitment for "cross shelf" structure.
The non linear fitting is done with standard linear regression. i.e.

$$f(x,y,z)=\beta_0+\sum_i\beta_i f_i(z)+\sum_j\beta_j g_j(x,z)+\sum_k\beta_k g_k(y,z)+\epsilon$$

where β_i are coefficients for the spatial functions and ϵ is the zero mean noise process associated with UK.

3.1.2.3 Fitting non-linear parameters

A brute force approach is taken to fitting the nonlinear parameters x_0 and λ. A search range is determined based on the geographic range of the observations. The parameters are then fit to minimize the misfit to observations.
subroutine $NLSF_Fit_Function$ parameter np). The nonlinear parameters are fit by minimizing RMS misfit to the simple least squares fit with a smoothness penalty,

$$J(x_0,\lambda)=\sqrt{\frac{1}{n}\sum_i(d_i-a-bf(x_i|\lambda,x_0))^2}+S(\lambda,x_0)$$

Where $S(\lambda,x_0)=\int_{-\infty}^{\infty}f"(x)^2dx=S(\lambda)$ is a roughness penalty, a and b are temporarily assigned (by least squares) constants fit to minimize J. S is proportional to λ^{-3} for all examples used here (see subroutine $NLSFuncPen$). Other one dimensional function forms can be added to the software in subroutine NLSF_Eval_Semivariance and NLSFFuncPen.
A smoothness penalty is imposed for each function based on the analytic

3.1.3 Residual process

After performing an ordinary least squares fit for the SF coeficients, β, we have an estimate of ϵ. An empirical variogram is computed subroutine $variogramF$, and variogram parameters are fit (again by brute force).
The variogram forms allowed are "spherical", "exponential", and "gaussian". The form is hard-coded in the main program, UniversalKriging.f90.

3.1.3.1 Posterior sampling

With the fitting of the residual we have a covariance for ϵ and the estimation problem becomes one of Generalized Least Squares (LSF_Generalized_Least_Squares). Posterior sampling is then conducted achieved posterior sampling is Treating the

Chapter 4

Mortality_Mod

4.1 Mortality Class

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

4.1.1 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

4.1.2 Read_Configuration

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

4.1.3 Load_Fishing_Mortalities

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

4.1.4 Ring_Size_Selectivity

Assign size class fishing selectivity based on increasing logistic function

$$Selectivity = \frac{1}{1 + e^{a - b * length_{shell}}}$$

4.1.5 Set_Fishing_Effort

4.1.6 Dollars_Per_SqM

4.1.7 Scallops_To_Counts

4.1.8 Set_Fishing_Effort_Weight_USD

4.1.9 Set_Fishing_Effort_Weight_BMS

4.1.10 Compute_Natural_Mortality

4.1.11 Set_Fishing_Mortality

4.1.12 Set_Config_File_Name

4.1.13 Set_Fishing_Mort_File_Name

4.1.14 Mortality_Write_At_Timestep

4.1.15 Set_Discard

Chapter 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 Grid Manager Support Methods

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

Chapter 6

Common Parameters

Chapter 7

Modules Index

7.1 Modules List

Here is a list of all modules with brief descriptions:

Chapter 8

Data Type Index

8.1 Data Types List

Here are the data types with brief descriptions:

Chapter 9

File Index

9.1 File List

Here is a list of all files with brief descriptions:

Chapter 10

Module Documentation

10.1 globals Module Reference

Functions/Subroutines

- elemental real(dp) function logic_to_double (value)
- real(dp) function, dimension(n, n) matrixinv (x, n)

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 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

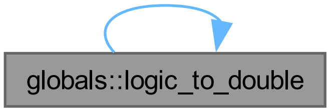
- real(dp), parameter zero_threshold = 1.0D-99
- real(dp), parameter pi = 3.14159265358979311599796346854D0
- real(dp), parameter grams_per_pound = 453.592_dp
- real(dp), parameter meters_per_naut_mile = 1852.D0
- real(dp), parameter feet_per_naut_mile = 6076.12
- real(dp), parameter grams_per_metric_ton = 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
- 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/'

10.1.1 Function/Subroutine Documentation

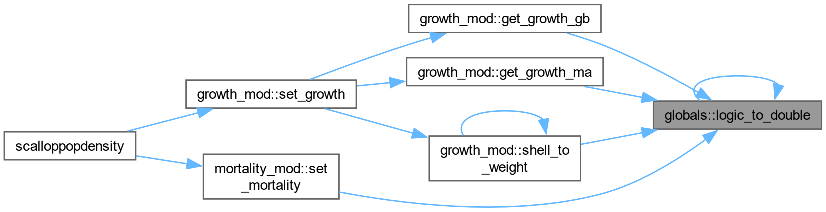
10.1.1.1 logic_to_double()

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

Here is the call graph for this function:



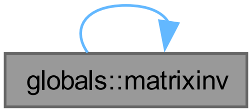
Here is the caller graph for this function:


growth_mod::get_growth_gb
growth_mod::get_growth_ma
growth_mod::set_growth
scalloppopdensity
growth_mod::shell_to_weight
mortality_mod::set_mortality
globals::logic_to_double

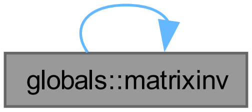
10.1.1.2 matrixinv()

```
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:



10.1.2 Variable Documentation

10.1.2.1 comment_len

```
integer, parameter globals::comment_len = 80
```

10.1.2.2 config_dir_interp

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

10.1.2.3 config_dir_sim

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

10.1.2.4 config_dir_special

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

10.1.2.5 csv_line_len

```
integer, parameter globals::csv_line_len = 2000
```

10.1.2.6 data_dir

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

10.1.2.7 domain_len

```
integer, parameter globals::domain_len = 2
```

10.1.2.8 dp

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

10.1.2.9 feet_per_naut_mile

```
real(dp), parameter globals::feet_per_naut_mile = 6076.12
```

10.1.2.10 fname_len

```
integer, parameter globals::fname_len = 100
```

10.1.2.11 form_len

```
integer, parameter globals::form_len = 20
```

10.1.2.12 grams_per_metric_ton

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

10.1.2.13 grams_per_pound

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

10.1.2.14 grid_area_sqm

```
real(dp), parameter globals::grid_area_sqm = meters_per_naut_mile**2
```

10.1.2.15 grid_dir

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

10.1.2.16 growth_out_dir

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

10.1.2.17 init_cond_dir

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

10.1.2.18 input_str_len

```
integer, parameter globals::input_str_len = 100
```

10.1.2.19 line_len

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

10.1.2.20 ma_gb_border

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

10.1.2.21 max_num_areas

```
integer, parameter globals::max_num_areas = 25
```

10.1.2.22 max_num_years

```
integer, parameter globals::max_num_years = 50
```

10.1.2.23 max_sides

```
integer, parameter globals::max_sides = 8
```

10.1.2.24 meters_per_naut_mile

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

10.1.2.25 ndim

```
integer, parameter globals::ndim = 12000
```

10.1.2.26 num_size_classes

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

10.1.2.27 one_scallop_per_tow

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

10.1.2.28 output_dir

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

10.1.2.29 pi

```
real(dp), parameter globals::pi = 3.14159265358979311599796346854D0
```

10.1.2.30 qp

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

10.1.2.31 read_dev

```
integer, parameter globals::read_dev = 69
```

10.1.2.32 rec_input_dir

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

10.1.2.33 rec_output_dir

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

10.1.2.34 region_ma

```
integer, parameter globals::region_ma =5
```

10.1.2.35 region_n

```
integer, parameter globals::region_n =1
```

10.1.2.36 region_none

```
integer, parameter globals::region_none =0
```

10.1.2.37 region_s

```
integer, parameter globals::region_s =2
```

10.1.2.38 region_sw

```
integer, parameter globals::region_sw =3
```

10.1.2.39 region_w

```
integer, parameter globals::region_w =4
```

10.1.2.40 shell_len_delta

```
integer, parameter globals::shell_len_delta = 5
```

10.1.2.41 shell_len_max

```
integer, parameter globals::shell_len_max = 150
```

10.1.2.42 shell_len_min

```
integer, parameter globals::shell_len_min = 30
```

10.1.2.43 sp

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

10.1.2.44 tag_len

```
integer, parameter globals::tag_len = 40
```

10.1.2.45 term_blk

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

10.1.2.46 term_blu

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

10.1.2.47 term_grn

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

10.1.2.48 term_red

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

10.1.2.49 term_yel

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

10.1.2.50 tow_area_sqm

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

10.1.2.51 value_len

```
integer, parameter globals::value_len = 30
```

10.1.2.52 write_dev

```
integer, parameter globals::write_dev = 63
```

10.1.2.53 zero_threshold

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

10.2 grid_manager_mod Module Reference

Data Types

- type grid_data_class
- type lonlatpoint
- type lonlatvector

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:
- integer function get_region (lat, lon, stratum_real)

  Returns a region number based on stratum 0: not used 1: region_N North region 2: region_S South region 3: region_SW Southwest region 4: region_W West region 5: region_MA Entire Mid-Atlantic Region Inputs: grid: locations of survey data.

Variables

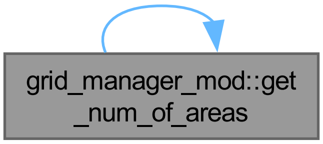
- type(lonlatvector), dimension(max_num_areas), private area
- integer, private num_areas
- integer, private num_grids
- logical, private use_spec_access_data
- character(domain_len), private domain_name
- character(fname_len), private config_file_name
- character(fname_len), private init_cond_fname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

10.2.1 Function/Subroutine Documentation

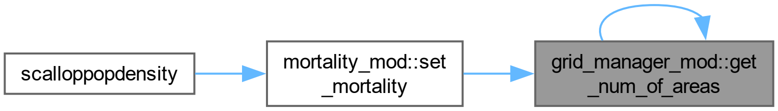
10.2.1.1 get_num_of_areas()

```
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:

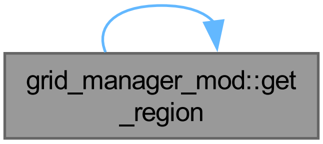


10.2.1.2 get_region()

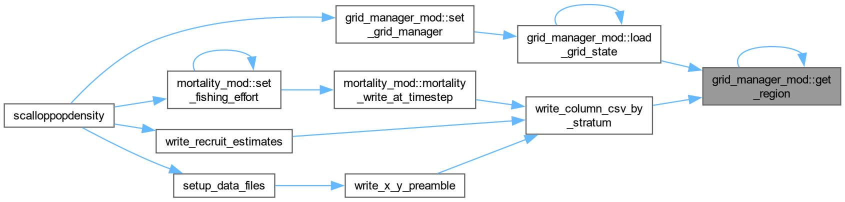
```
integer function grid_manager_mod::get_region (
            real(dp), intent(in) lat,
            real(dp), intent(in) lon,
            real(dp), intent(in) stratum_real )
```

Returns a region number based on stratum 0: not used 1: region_N North region 2: region_S South region 3: region↩ _SW Southwest region 4: region_W West region 5: region_MA Entire Mid-Atlantic Region Inputs: grid: locations of survey data.

Here is the call graph for this function:



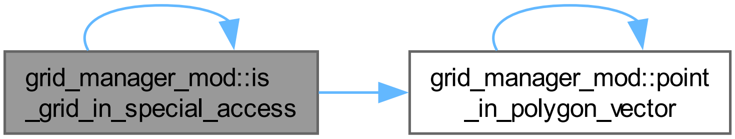
Here is the caller graph for this function:



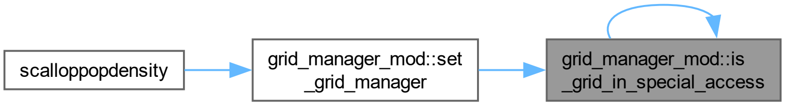
10.2.1.3 is_grid_in_special_access()

```
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:



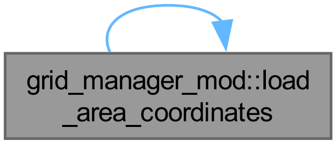
Here is the caller graph for this function:



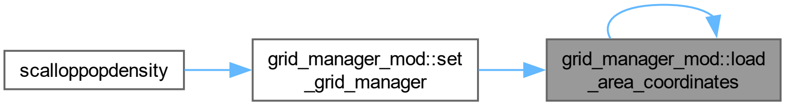
10.2.1.4 load_area_coordinates()

```
integer function grid_manager_mod::load_area_coordinates
```

Here is the call graph for this function:



Here is the caller graph for this function:


scalloppopdensity
grid_manager_mod::set_grid_manager
grid_manager_mod::load_area_coordinates

10.2.1.5 load_grid_state()

```
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 )
```

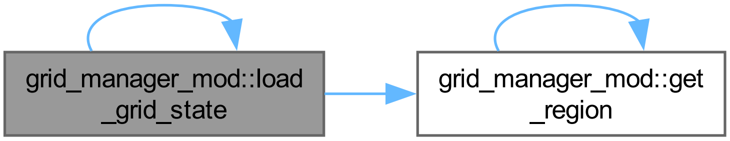
This function is used to set the grid parameters and the initial state to start the simulation.

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

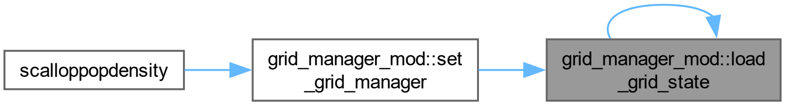
Parameters

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:


grid_manager_mod::load_grid_state
grid_manager_mod::get_region

Here is the caller graph for this function:


scalloppopdensity
grid_manager_mod::set_grid_manager
grid_manager_mod::load_grid_state

10.2.1.6 point_in_polygon_array()

```
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 )
```

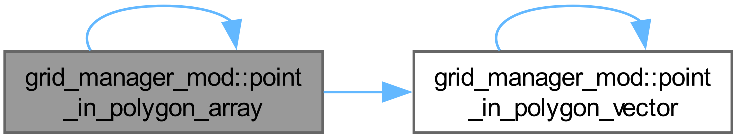
Parameters

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

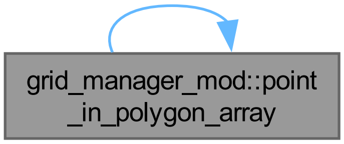
Returns

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:


grid_manager_mod::point_in_polygon_array
grid_manager_mod::point_in_polygon_vector

Here is the caller graph for this function:



10.2.1.7 point_in_polygon_points()

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

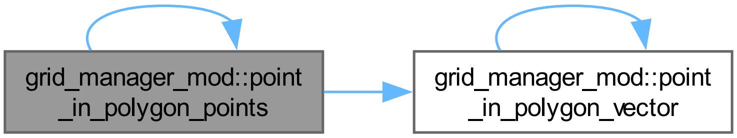
Parameters

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

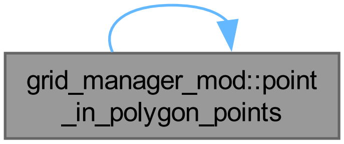
Returns

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:



10.2.1.8 point_in_polygon_vector()

```
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/

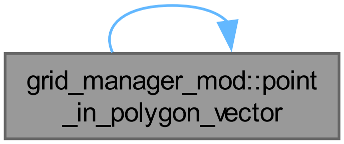
Parameters

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

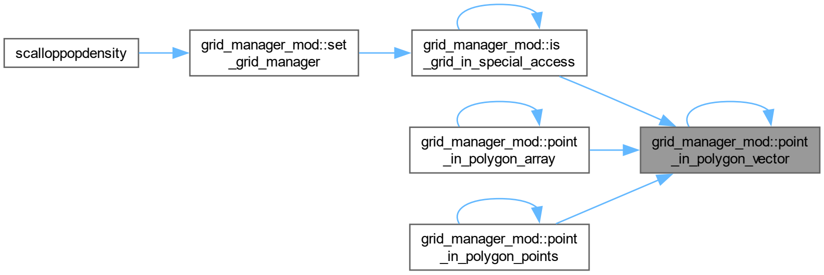
Returns

> 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:



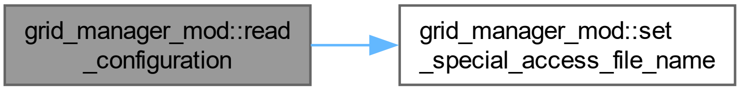
Here is the caller graph for this function:



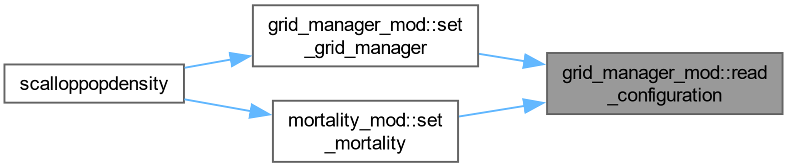
10.2.1.9 read_configuration()

```
subroutine grid_manager_mod::read_configuration
```

Read_Configuration.
Read Input File
Reads a configuration file Here is the call graph for this function:



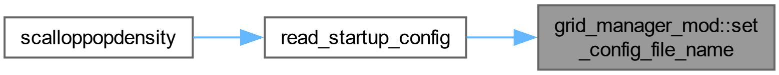
Here is the caller graph for this function:



10.2.1.10 set_config_file_name()

```
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:

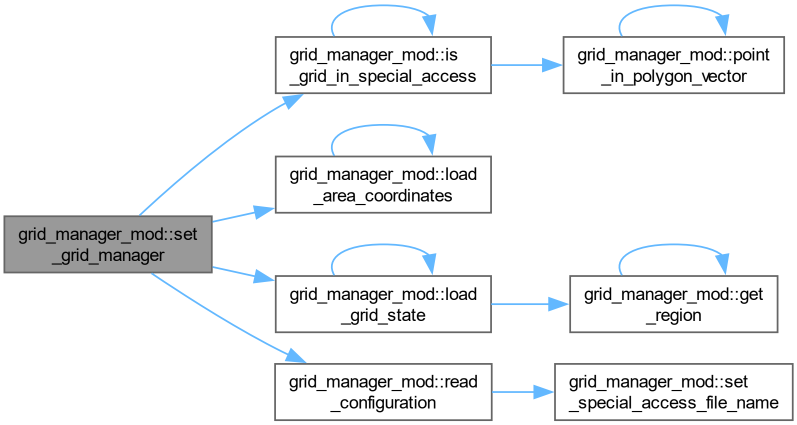


10.2.1.11 set_grid_manager()

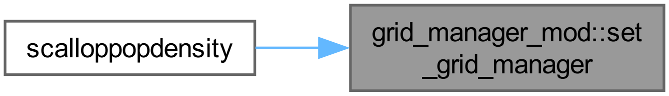
```
subroutine grid_manager_mod::set_grid_manager (
             real(dp), dimension(1:ngrids, 1:num_size_classes), intent(out) state_mat,
             type(grid_data_class), dimension(*), intent(out) grid,
             integer, intent(inout) ngrids,
             character(domain_len), intent(in) dom_name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

Initializes growth for startup.

Here is the call graph for this function:


grid_manager_mod::set _grid_manager
grid_manager_mod::is _grid_in_special_access
grid_manager_mod::point _in_polygon_vector
grid_manager_mod::load _area_coordinates
grid_manager_mod::load _grid_state
grid_manager_mod::get _region
grid_manager_mod::read _configuration
grid_manager_mod::set _special_access_file_name

Here is the caller graph for this function:



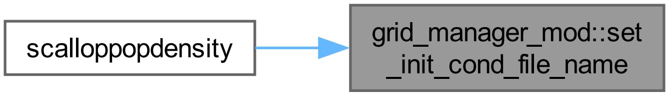
10.2.1.12 set_init_cond_file_name()

```
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:



10.2.1.13 set_num_grids()

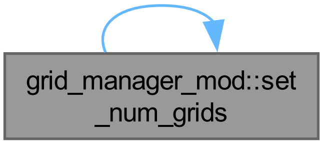
```
integer function grid_manager_mod::set_num_grids
```

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.

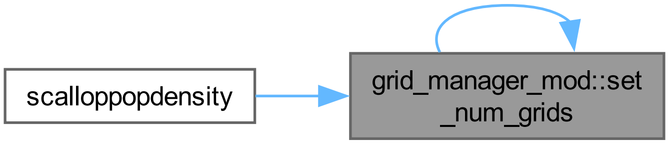
Returns

The expected number of grids to process.

Here is the call graph for this function:



Here is the caller graph for this function:



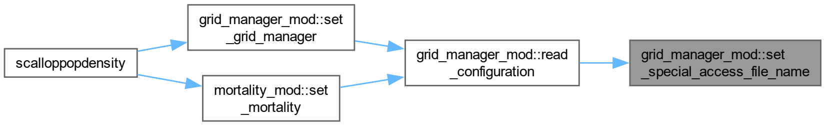
10.2.1.14 set_special_access_file_name()

```
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:



10.2.2 Variable Documentation

10.2.2.1 area

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

10.2.2.2 config_file_name

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

10.2.2.3 domain_name

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

10.2.2.4 init_cond_fname

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

10.2.2.5 num_areas

```
integer, private grid_manager_mod::num_areas  [private]
```

10.2.2.6 num_grids

```
integer, private grid_manager_mod::num_grids  [private]
```

10.2.2.7 special_accesss_fname

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

10.2.2.8 use_spec_access_data

```
logical, private grid_manager_mod::use_spec_access_data  [private]
```

10.3 growth_mod Module Reference

Data Types

- type growth_class

Functions/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)

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

10.3.1 Detailed Description

10.3.2 Function/Subroutine Documentation

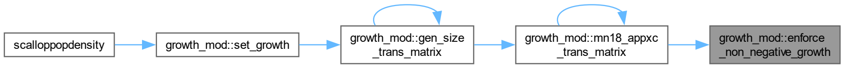
10.3.2.1 enforce_non_negative_growth()

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

Parameters

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

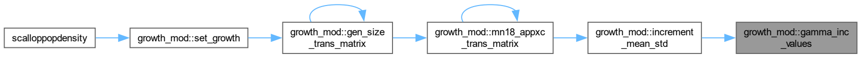
Here is the caller graph for this function:


scalloppopdensity
growth_mod::set_growth
growth_mod::gen_size _trans_matrix
growth_mod::mn18_appxc _trans_matrix
growth_mod::enforce _non_negative_growth

10.3.2.2 gamma_inc_values()

```
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:


scalloppopdensity
growth_mod::set_growth
growth_mod::gen_size_trans_matrix
growth_mod::mn18_appxc_trans_matrix
growth_mod::increment_mean_std
growth_mod::gamma_inc_values

10.3.2.3 gen_size_trans_matrix()

```
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.

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

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

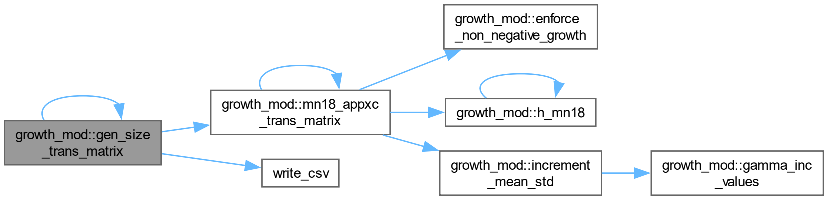
Returns

Transition Matrix

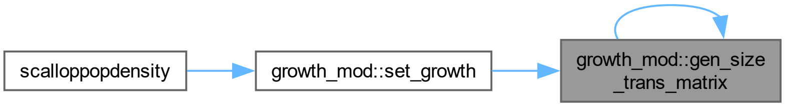
Author

Keston Smith (IBSS corp) June-July 2021

Here is the call graph for this function:



Here is the caller graph for this function:



10.3.2.4 get_growth_gb()

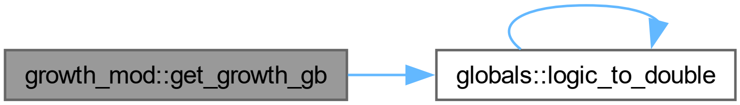
```
subroutine growth_mod::get_growth_gb (
            real(dp), intent(in) depth,
            real(dp), intent(in) lat,
            logical, intent(in) is_closed,
            real(dp), intent(out) l_inf_mu,
            real(dp), intent(out) k_mu,
            real(dp), intent(out) l_inf_sd,
            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.

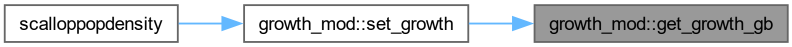
Parameters

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:



10.3.2.5 get_growth_ma()

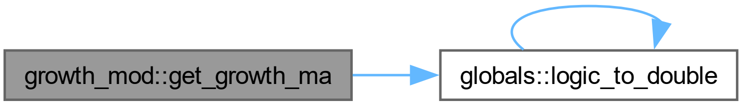
```
subroutine growth_mod::get_growth_ma (
             real(dp), intent(in) depth,
             real(dp), intent(in) lat,
             logical, intent(in) is_closed,
             real(dp), intent(out) l_inf_mu,
             real(dp), intent(out) k_mu,
             real(dp), intent(out) l_inf_sd,
             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.

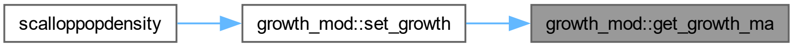
Parameters

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:



10.3.2.6 h_mn18()

```
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 $f = \Phi_N$

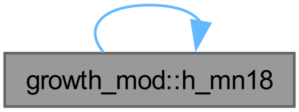
Parameters

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

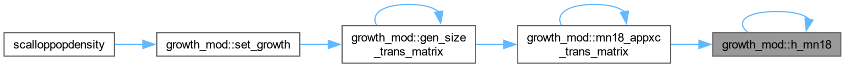
Returns

H - variable

Here is the call graph for this function:



Here is the caller graph for this function:


scalloppopdensity
growth_mod::set_growth
growth_mod::gen_size_trans_matrix
growth_mod::mn18_appxc_trans_matrix
growth_mod::h_mn18

10.3.2.7 increment_mean_std()

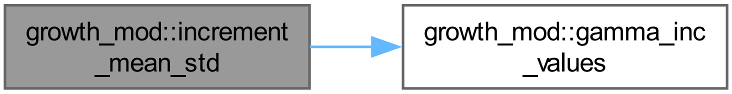
```
subroutine growth_mod::increment_mean_std (
                real(dp), intent(in) l_inf_mu,
                real(dp), intent(in) k_mu,
                real(dp), intent(in) l_inf_sd,
                real(dp), intent(in) k_sd,
                real(dp), intent(in) size,
                real(dp), intent(out) 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.

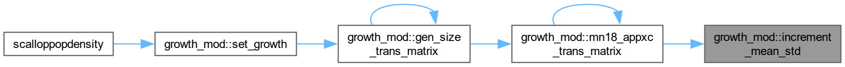
Parameters

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:



10.3.2.8 mn18_appxc_trans_matrix()

```
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

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

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

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

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

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

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

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

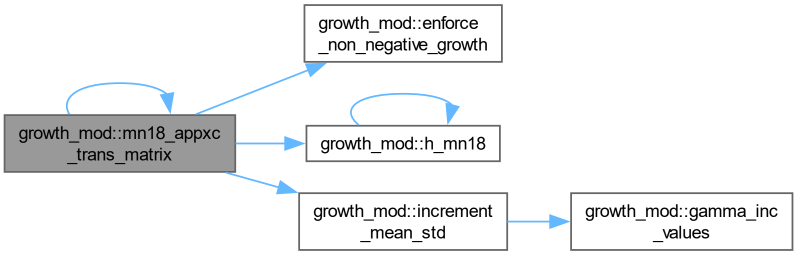
Parameters

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

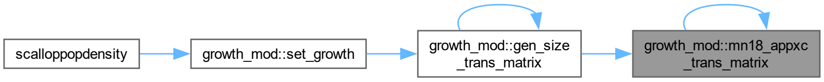
Returns

> 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:


growth_mod::enforce _non_negative_growth
growth_mod::mn18_appxc _trans_matrix
growth_mod::h_mn18
growth_mod::increment _mean_std
growth_mod::gamma_inc _values

Here is the caller graph for this function:


scalloppopdensity
growth_mod::set_growth
growth_mod::gen_size _trans_matrix
growth_mod::mn18_appxc _trans_matrix

10.3.2.9 norm_cumul_dist_fcn()

```
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.

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

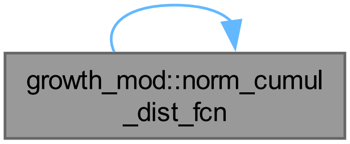
Parameters

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

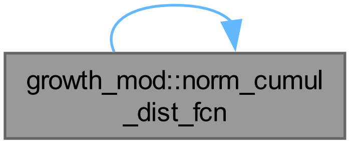
Returns

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:



10.3.2.10 norm_density_fcn()

```
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.

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

Parameters

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

Returns

normal density function at x

Here is the call graph for this function:

Here is the caller graph for this function:

10.3.2.11 set_growth()

```
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.

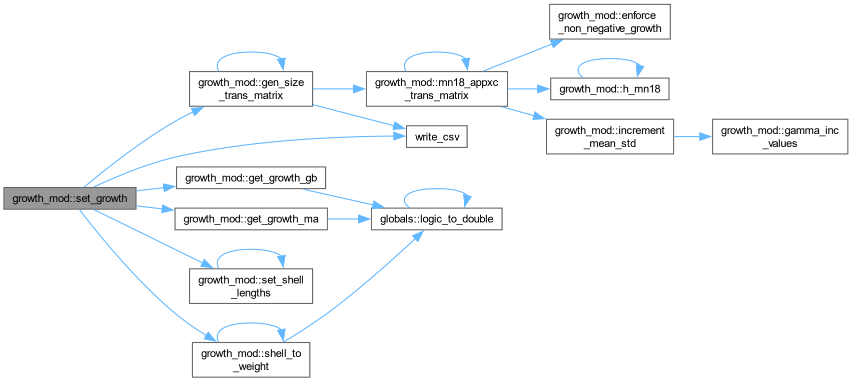
Parameters

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

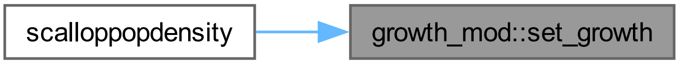
Parameters

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:



10.3.2.12 set_shell_lengths()

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

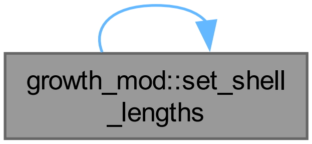
Parameters

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

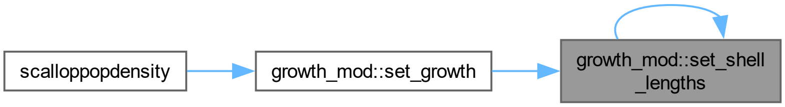
Returns

shell length in millimeters

Here is the call graph for this function:



Here is the caller graph for this function:



10.3.2.13 shell_to_weight()

```
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

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

$$weight_g = e^x$$

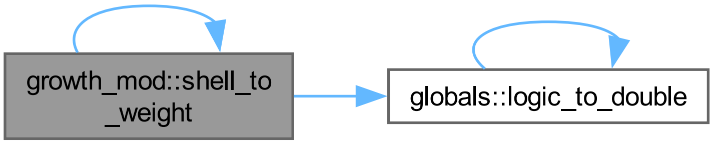
Parameters

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???

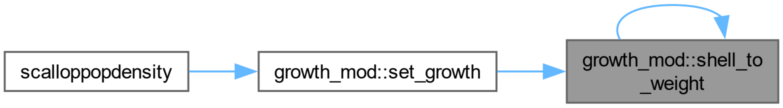
Returns

weight in grams

Here is the call graph for this function:



Here is the caller graph for this function:



10.3.2.14 time_to_grow()

```
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**

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

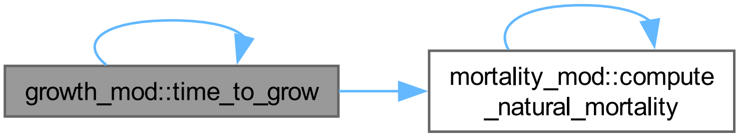
- Compute new state

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

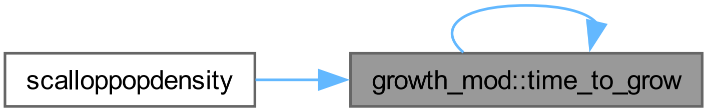
Parameters

`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:



10.3.3 Variable Documentation

10.3.3.1 delta_time

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

10.3.3.2 domain_area_sqm

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

10.3.3.3 domain_name

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

10.3.3.4 growth_param_size

```
integer, parameter growth_mod::growth_param_size = 4
```

10.3.3.5 num_grids

```
integer, private growth_mod::num_grids  [private]
```

10.3.3.6 num_time_steps

```
integer, private growth_mod::num_time_steps  [private]
```

10.3.3.7 time_steps_year

```
integer, private growth_mod::time_steps_year  [private]
```

10.4 mortality_mod Module Reference

Data Types

- type dataforplots
- type fishingmortality
- type mortality_class

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, save_by↩ _strat)
- 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 dollars_per_sqm (year, meat_weight_grams)

  Compute value of scallop population at a specific grid location.
- subroutine scallops_to_counts (meat_weight_grams, cnt10, cnt10to20, cnt20to30, cnt30plus)

  Purpose: Convert Scallop density by shell height and meat wieght to count data. The count data are divided into cnt10- 10 or less scallops per pound. cnt10to20 10-20 scallops per pound. cnt20to30 20-30 scallops per pound. cnt30+ 30 or more scallops per pound.
- 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, mortality, 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 @parma[in] expl_biomass ! Expl biomass @parma[in] expl_scallops ! Expl Number of Scallops.

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
- logical, private save_by_stratum
- 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 usd_per_sqm
- real(dp), dimension(:), allocatable, private expl_scallops_psqm
- real(dp), dimension(:), allocatable, private expl_num
- real(dp), dimension(:), allocatable, private f_mort
- real(dp), dimension(:), allocatable, private f_mort_raw
- real(dp), dimension(:), allocatable, private landings_by_num
- real(dp), dimension(:), allocatable, private landings_wgt_grams
- real(dp), dimension(:), allocatable, private landings_wgt_grams_open
- real(dp), dimension(:), allocatable, private landings_wgt_grams_closed
- real(dp), dimension(:), allocatable, private lpue

- real(dp), dimension(:), allocatable, private fishing_effort
- real(dp), dimension(num_size_classes), private expl_scallops_psqm_at_size
- real(dp), dimension(num_size_classes), private landings_at_size
- real(dp), dimension(num_size_classes), private landings_at_size_open
- real(dp), dimension(num_size_classes), private landings_at_size_closed
- type(dataforplots), private data_select

10.4.1 Function/Subroutine Documentation

10.4.1.1 calc_lpue()

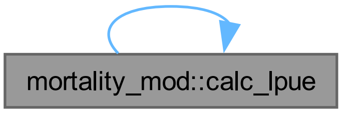
```
elemental real(dp) function mortality_mod::calc_lpue (
             real(dp), intent(in) expl_biomass,
             real(dp), intent(in) expl_scallops )
```

Computes catch as pounds per day @parma[in] expl_biomass ! Expl biomass @parma[in] expl_scallops ! Expl Number of Scallops.

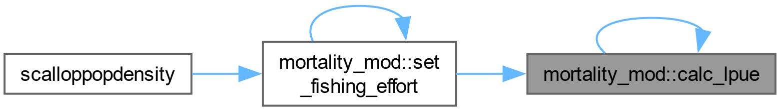
Parameters

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 = expl_weight_g xxxx xxxx Here is the call graph for this function:



Here is the caller graph for this function:



10.4.1.2 compute_natural_mortality()

```
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.

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

A similar formula for GB Open:

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

where M_{adult} is 0.25 if MA or 0.2 if GB
TODO: At present the computation does not use the conditional but rather whichever is greater
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

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

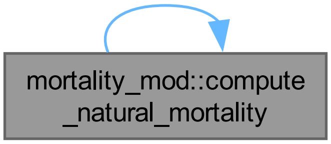
Parameters

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

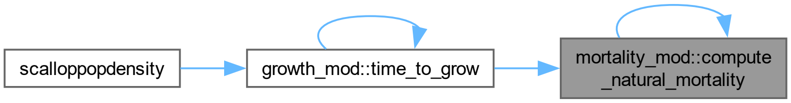
Returns

natural_mortality and juvenile mortality

Here is the call graph for this function:



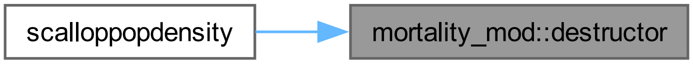
Here is the caller graph for this function:



10.4.1.3 destructor()

```
subroutine mortality_mod::destructor
```

Here is the caller graph for this function:



10.4.1.4 dollars_per_sqm()

```
real(dp) function mortality_mod::dollars_per_sqm (
             integer, intent(in) year,
             real(dp), dimension(*), intent(in) meat_weight_grams )
```

Compute value of scallop population at a specific grid location.

Value is based on population structure. The population is sorted into size count bucket classes U10, 10-20, 20-30, 30+ and the value based on these classes and the year is read from the file "Data/ScallopPrice.csv".

Parameters

in	*year*	- current year
in	*meat_weight_grams*	[num_size_classes] - Weight meat per individual scallop in each size class

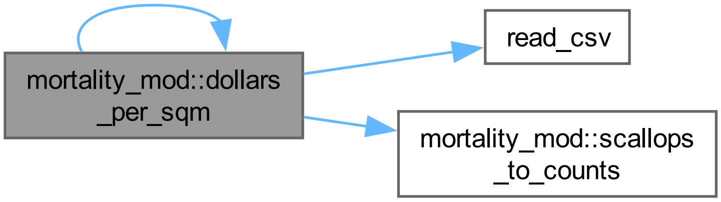
Returns

dollars per square meter

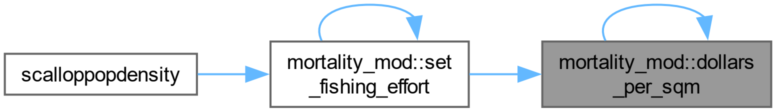
Author

Keston Smith 2022

Here is the call graph for this function:



Here is the caller graph for this function:

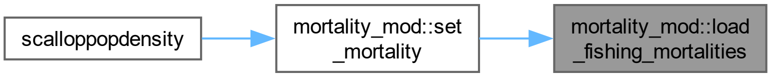


10.4.1.5 load_fishing_mortalities()

```
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:

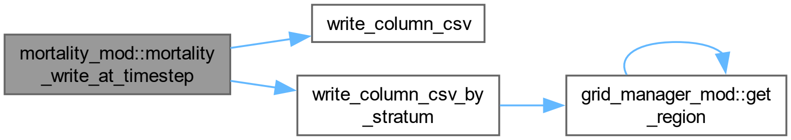


10.4.1.6 mortality_write_at_timestep()

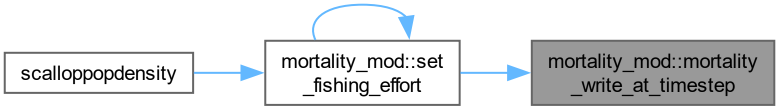
```
subroutine mortality_mod::mortality_write_at_timestep (
            integer, intent(in) year,
            integer, intent(in) ts,
            real(dp), dimension(1:num_grids, 1:num_size_classes), intent(in) state_mat,
            real(dp), dimension(1:num_grids, 1:num_size_classes), intent(in) weight_grams,
            type(mortality_class), dimension(*), intent(in) mortality,
            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:



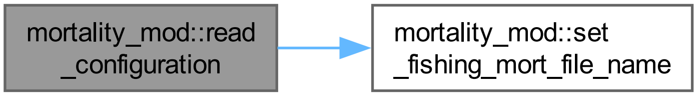
10.4.1.7 read_configuration()

```
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:



10.4.1.8 ring_size_selectivity()

```
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)

Parameters

in	*is_closed*	true if grid is closed to fishing

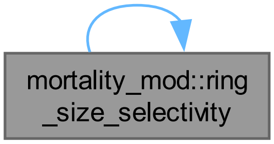
Author

Keston Smith (IBSS corp) May 2022

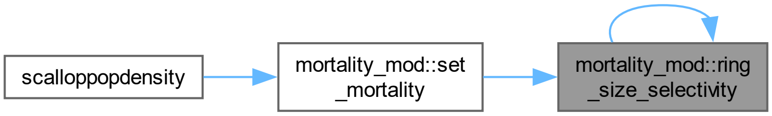
Returns

length num_size_classes vector of selectivity

Here is the call graph for this function:



Here is the caller graph for this function:



10.4.1.9 scallops_to_counts()

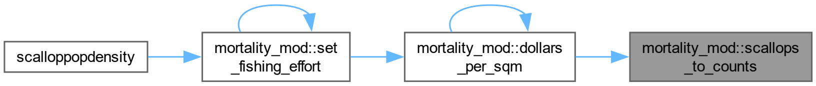
```
subroutine mortality_mod::scallops_to_counts (
             real(dp), dimension(*), intent(in) meat_weight_grams,
             real(dp), intent(out) cnt10,
             real(dp), intent(out) cnt10to20,
             real(dp), intent(out) cnt20to30,
             real(dp), intent(out) cnt30plus )
```

Purpose: Convert Scallop density by shell height and meat wieght to count data. The count data are divided into cnt10-10 or less scallops per pound. cnt10to20 10-20 scallops per pound. cnt20to30 20-30 scallops per pound. cnt30+ 30 or more scallops per pound.

Parameters

in	*meat_weight_grams*	(real) length num_size_classes vector of weight of individual scallops by size class
out	*cnt10*	number of scallops wich get binned into U10
out	*cnt10to20*	number of scallops wich get binned into U10-20
out	*cnt20to30*	number of scallops wich get binned into U20-30
out	*cnt30*	number of scallops wich get binned into U30+

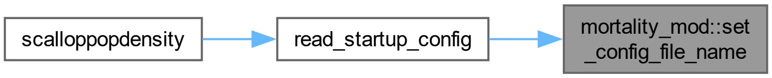
Here is the caller graph for this function:


scalloppopdensity
mortality_mod::set_fishing_effort
mortality_mod::dollars_per_sqm
mortality_mod::scallops_to_counts

10.4.1.10 set_config_file_name()

```
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:



10.4.1.11 set_discard()

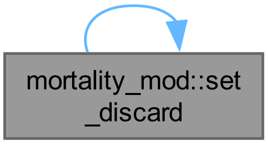
```
elemental real(dp) function mortality_mod::set_discard (
             real(dp), intent(in) length,
             real(dp), intent(in) selectivity,
             real(dp), intent(in) cull_size,
             real(dp), intent(in) discard,
             logical, intent(in) is_closed )
```

Computes element of discard vector.

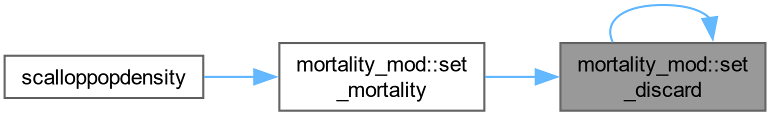
Parameters

in	*length,vector*	element for shell length @parma[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:



10.4.1.12 set_fishing_effort()

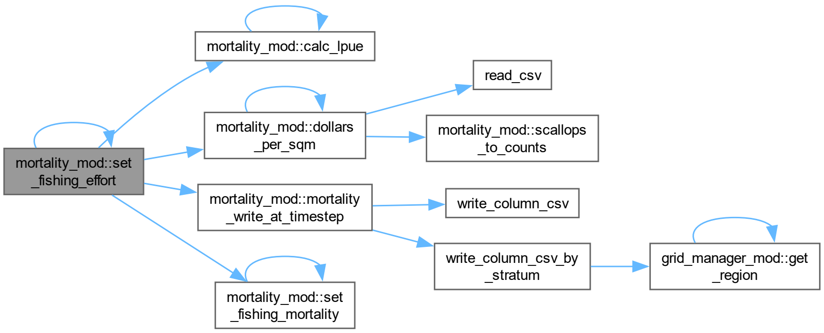
```
real(dp) function, dimension(num_grids) mortality_mod::set_fishing_effort (
            integer, intent(in) year,
            integer, intent(in) ts,
            real(dp), dimension(1:num_grids, 1:num_size_classes), intent(in) state_mat,
            real(dp), dimension(1:num_grids, 1:num_size_classes ), 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.

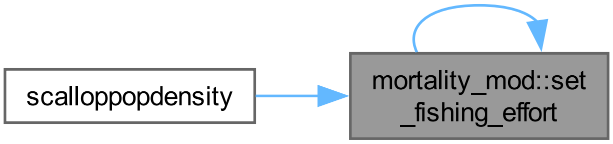
Parameters

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:



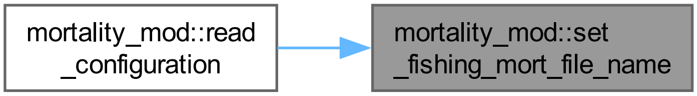
Here is the caller graph for this function:



10.4.1.13 set_fishing_mort_file_name()

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

Here is the caller graph for this function:



10.4.1.14 set_fishing_mortality()

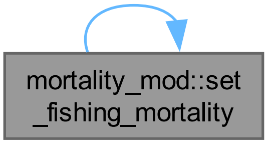
```
elemental real(dp) function mortality_mod::set_fishing_mortality (
             type(grid_data_class), intent(in) grid,
             integer, intent(in) year,
             logical, intent(in) use_f_loc,
             real(dp), intent(in) f_loc )
```

Computes Fishing Mortality.

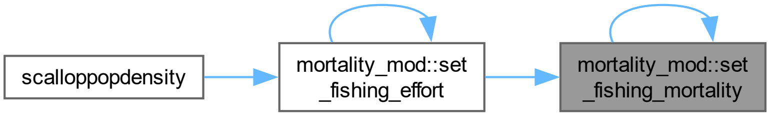
There is a year list for each year of interest, up to a total number of years of max_num_years For each list item there are two vectors.

- 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:


scalloppopdensity
mortality_mod::set_fishing_effort
mortality_mod::set_fishing_mortality

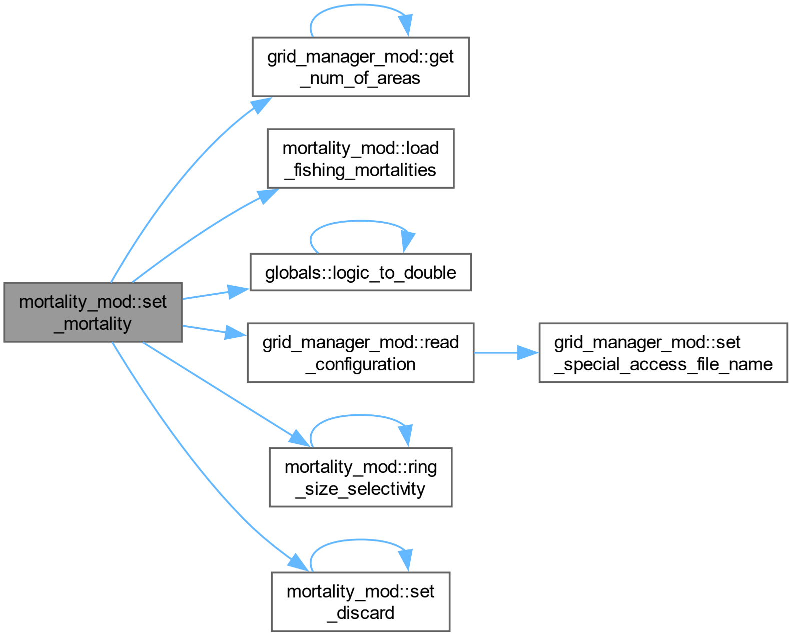
10.4.1.15 set_mortality()

```
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,
             logical, intent(in) save_by_strat )
```

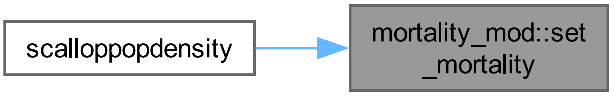
Parameters

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:



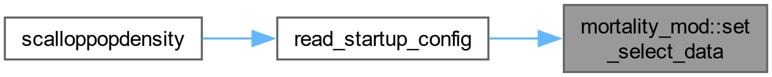
Here is the caller graph for this function:



10.4.1.16 set_select_data()

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

Here is the caller graph for this function:



10.4.2 Variable Documentation

10.4.2.1 alpha_mort

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

10.4.2.2 config_file_name

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

10.4.2.3 data_select

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

10.4.2.4 delta_time

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

10.4.2.5 domain_area_sqm

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

10.4.2.6 domain_name

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

10.4.2.7 dredge_width_m

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

10.4.2.8 expl_biomass_gpsqm

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

10.4.2.9 expl_num

```
real(dp), dimension(:), allocatable, private mortality_mod::expl_num  [private]
```

10.4.2.10 expl_scallops_psqm

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

10.4.2.11 expl_scallops_psqm_at_size

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

10.4.2.12 f_mort

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

10.4.2.13 f_mort_raw

```
real(dp), dimension(:), allocatable, private mortality_mod::f_mort_raw  [private]
```

10.4.2.14 fishing_effort

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

10.4.2.15 fishing_mort

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

10.4.2.16 fishing_mort_fname

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

10.4.2.17 fmort_list

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

10.4.2.18 gb_cull_size_mm

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

10.4.2.19 gb_discard

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

10.4.2.20 gb_incidental

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

10.4.2.21 gb_length_0

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

10.4.2.22 gb_mort_adult

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

10.4.2.23 gbc_fselect_a

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

10.4.2.24 gbc_fselect_b

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

10.4.2.25 gbo_fselect_a

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

10.4.2.26 gbo_fselect_b

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

10.4.2.27 landings_at_size

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

10.4.2.28 landings_at_size_closed

```
real(dp), dimension(num_size_classes), private mortality_mod::landings_at_size_closed  [private]
```

10.4.2.29 landings_at_size_open

```
real(dp), dimension(num_size_classes), private mortality_mod::landings_at_size_open  [private]
```

10.4.2.30 landings_by_num

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

10.4.2.31 landings_wgt_grams

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

10.4.2.32 landings_wgt_grams_closed

```
real(dp), dimension(:), allocatable, private mortality_mod::landings_wgt_grams_closed  [private]
```

10.4.2.33 landings_wgt_grams_open

```
real(dp), dimension(:), allocatable, private mortality_mod::landings_wgt_grams_open  [private]
```

10.4.2.34 lpue

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

10.4.2.35 lpue_intercept

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

10.4.2.36 lpue_slope

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

10.4.2.37 lpue_slope2

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

10.4.2.38 ma_cull_size_mm

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

10.4.2.39 ma_discard

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

10.4.2.40 ma_fselect_a

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

10.4.2.41 ma_fselect_b

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

10.4.2.42 ma_incidental

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

10.4.2.43 ma_length_0

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

10.4.2.44 ma_mort_adult

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

10.4.2.45 max_per_day

```
integer, private mortality_mod::max_per_day  [private]
```

10.4.2.46 max_time_hpd

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

10.4.2.47 num_areas

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

10.4.2.48 num_grids

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

10.4.2.49 num_in_list

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

10.4.2.50 num_time_steps

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

10.4.2.51 save_by_stratum

```
logical, private mortality_mod::save_by_stratum  [private]
```

10.4.2.52 towing_speed_knots

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

10.4.2.53 ts_per_year

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

10.4.2.54 usd_per_sqm

```
real(dp), dimension(:), allocatable, private mortality_mod::usd_per_sqm  [private]
```

10.4.2.55 use_spec_access_data

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

10.5 recruit_mod Module Reference

Data Types

- type recruitment_class

Functions/Subroutines

- subroutine set_recruitment (recruit, n_grids, dom_name, dom_area, l_inf_mu, k_mu, shell_length_mm, yr_start, yr_stop)

  Set_Recruitment.
- 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.

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 recr_start_year
- integer, private recr_stop_year
- real(dp), private recr_period_start
- real(dp), private recr_period_stop

10.5.1 Function/Subroutine Documentation

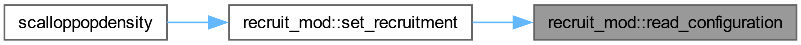
10.5.1.1 read_configuration()

```
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:



10.5.1.2 set_config_file_name()

```
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'

10.5.1.3 set_recruitment()

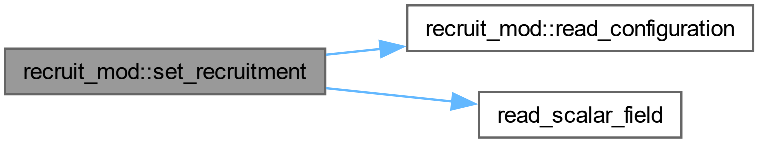
```
subroutine recruit_mod::set_recruitment (
             type(recruitment_class), dimension(*), intent(inout) recruit,
             integer, intent(in) n_grids,
             character(domain_len), intent(in) dom_name,
             real(dp), intent(in) dom_area,
             real(dp), dimension(*), 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

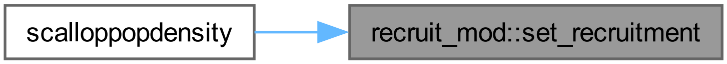
Parameters

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	*num_sz_classes*	
in	*L_inf_mu*	asymptotic size, average
in	*K_mu*	Brody growth coefficient K, average
in	*shell_length_mm*	Shell height in millimeters

Here is the call graph for this function:



Here is the caller graph for this function:



10.5.2 Variable Documentation

10.5.2.1 config_file_name

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

10.5.2.2 domain_area_sqm

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

10.5.2.3 domain_name

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

10.5.2.4 max_n_year

```
integer, parameter recruit_mod::max_n_year = 50
```

10.5.2.5 num_grids

```
integer, private recruit_mod::num_grids  [private]
```

10.5.2.6 recr_period_start

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

10.5.2.7 recr_period_stop

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

10.5.2.8 recr_start_year

```
integer, private recruit_mod::recr_start_year  [private]
```

10.5.2.9 recr_stop_year

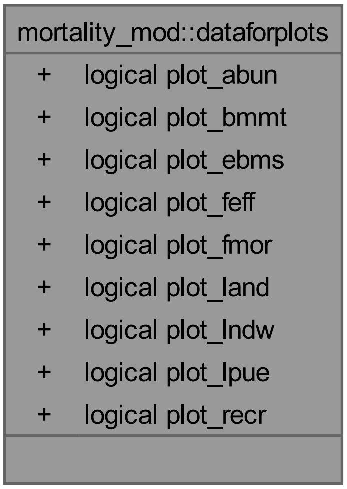
```
integer, private recruit_mod::recr_stop_year  [private]
```

Chapter 11

Data Type Documentation

11.1 mortality_mod::dataforplots Type Reference

Collaboration diagram for mortality_mod::dataforplots:



Public Attributes

- logical plot_abun
- logical plot_bmmt
- logical plot_ebms
- logical plot_feff
- logical plot_fmor
- logical plot_land
- logical plot_lndw
- logical plot_lpue
- logical plot_recr

11.1.1 Member Data Documentation

11.1.1.1 plot_abun

```
logical mortality_mod::dataforplots::plot_abun
```

11.1.1.2 plot_bmmt

```
logical mortality_mod::dataforplots::plot_bmmt
```

11.1.1.3 plot_ebms

```
logical mortality_mod::dataforplots::plot_ebms
```

11.1.1.4 plot_feff

```
logical mortality_mod::dataforplots::plot_feff
```

11.1.1.5 plot_fmor

```
logical mortality_mod::dataforplots::plot_fmor
```

11.1.1.6 plot_land

```
logical mortality_mod::dataforplots::plot_land
```

11.1.1.7 plot_lndw

```
logical mortality_mod::dataforplots::plot_lndw
```

11.1.1.8 plot_lpue

```
logical mortality_mod::dataforplots::plot_lpue
```

11.1.1.9 plot_recr

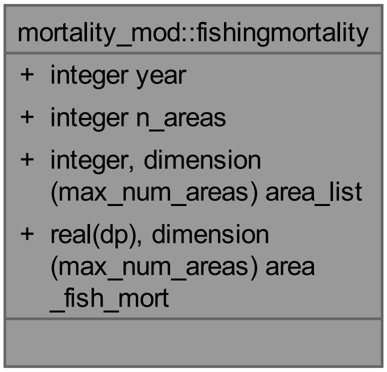
```
logical mortality_mod::dataforplots::plot_recr
```

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

- SRC/ScallopMortality.f90

11.2 mortality_mod::fishingmortality Type Reference

Collaboration diagram for mortality_mod::fishingmortality:


mortality_mod::fishingmortality
+ integer year
+ integer n_areas
+ integer, dimension (max_num_areas) area_list
+ real(dp), dimension (max_num_areas) area _fish_mort

Public Attributes

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

11.2.1 Member Data Documentation

11.2.1.1 area_fish_mort

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

11.2.1.2 area_list

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

11.2.1.3 n_areas

```
integer mortality_mod::fishingmortality::n_areas
```

11.2.1.4 year

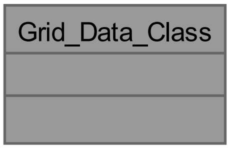
```
integer mortality_mod::fishingmortality::year
```

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

- SRC/ScallopMortality.f90

11.3 Grid_Data_Class Module Reference

Collaboration diagram for Grid_Data_Class:

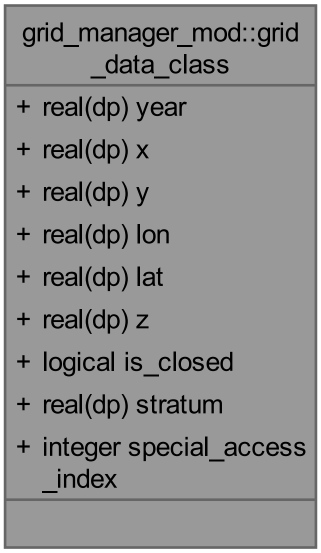


The documentation for this module was generated from the following file:

- SRC/GridManager.f90

11.4 grid_manager_mod::grid_data_class Type Reference

Collaboration diagram for grid_manager_mod::grid_data_class:



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

11.4.1 Member Data Documentation

11.4.1.1 is_closed

```
logical grid_manager_mod::grid_data_class::is_closed
```

11.4.1.2 lat

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

11.4.1.3 lon

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

11.4.1.4 special_access_index

```
integer grid_manager_mod::grid_data_class::special_access_index
```

11.4.1.5 stratum

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

11.4.1.6 x

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

11.4.1.7 y

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

11.4.1.8 year

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

11.4.1.9 z

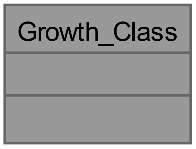
```
real(dp) grid_manager_mod::grid_data_class::z
```

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

- SRC/GridManager.f90

11.5 Growth_Class Module Reference

Subroutines that determine expected growth of scallops.
Collaboration diagram for Growth_Class:



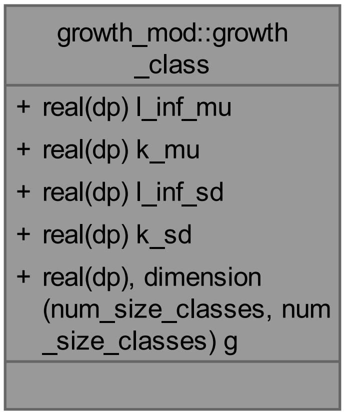
11.5.1 Detailed Description

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:



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.

11.6.1 Member Data Documentation

11.6.1.1 g

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

Growth matrix.

11.6.1.2 k_mu

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

Growth coefficient mean.

11.6.1.3 k_sd

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

Growth coefficient standard deviation.

11.6.1.4 l_inf_mu

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

Asymptotic size mean.

11.6.1.5 l_inf_sd

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

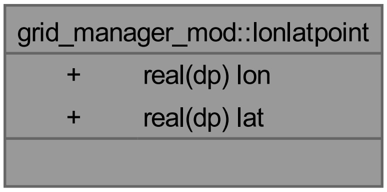
Asymptotic size standard deviation.

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

- SRC/ScallopGrowth.f90

11.7 grid_manager_mod::lonlatpoint Type Reference

Collaboration diagram for grid_manager_mod::lonlatpoint:



Public Attributes

- real(dp) lon
- real(dp) lat

11.7.1 Member Data Documentation

11.7.1.1 lat

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

11.7.1.2 lon

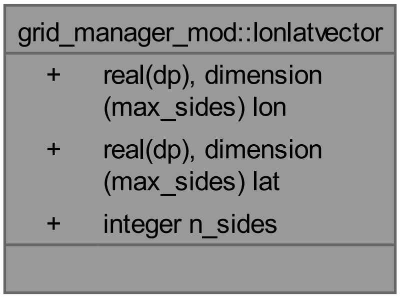
```
real(dp) grid_manager_mod::lonlatpoint::lon
```

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

- SRC/GridManager.f90

11.8 grid_manager_mod::lonlatvector Type Reference

Collaboration diagram for grid_manager_mod::lonlatvector:


grid_manager_mod::lonlatvector
+ real(dp), dimension (max_sides) lon
+ real(dp), dimension (max_sides) lat
+ integer n_sides

Public Attributes

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

11.8.1 Member Data Documentation

11.8.1.1 lat

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

11.8.1.2 lon

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

11.8.1.3 n_sides

```
integer grid_manager_mod::lonlatvector::n_sides
```

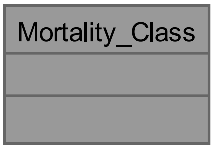
The documentation for this type was generated from the following file:

- SRC/GridManager.f90

11.9 Mortality_Class Module Reference

Subroutines that determine expected mortality of scallops.

Collaboration diagram for Mortality_Class:



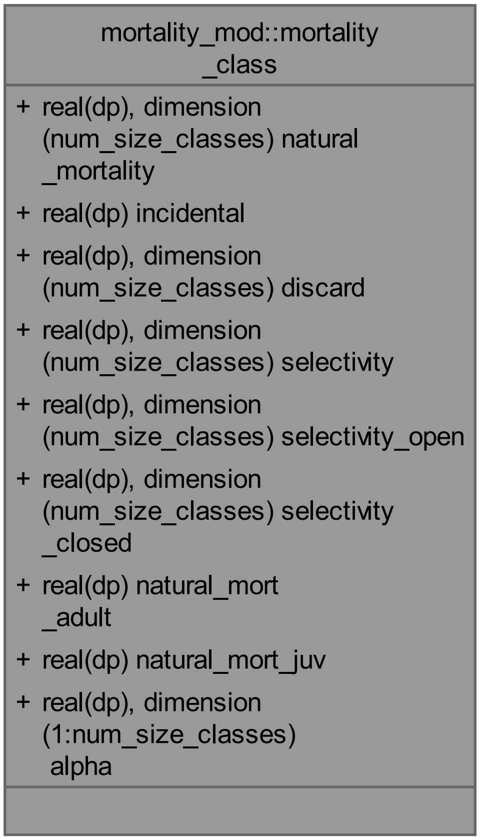
11.9.1 Detailed Description

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:


mortality_mod::mortality_class
+ 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), dimension (num_size_classes) selectivity_open
+ real(dp), dimension (num_size_classes) selectivity_closed
+ real(dp) natural_mort_adult
+ real(dp) natural_mort_juv
+ real(dp), dimension (1:num_size_classes) alpha

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), dimension(num_size_classes) selectivity_open
- real(dp), dimension(num_size_classes) selectivity_closed
- real(dp) natural_mort_adult
- real(dp) natural_mort_juv
- real(dp), dimension(1:num_size_classes) alpha

11.10.1 Member Data Documentation

11.10.1.1 alpha

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

11.10.1.2 discard

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

11.10.1.3 incidental

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

11.10.1.4 natural_mort_adult

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

11.10.1.5 natural_mort_juv

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

11.10.1.6 natural_mortality

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

11.10.1.7 selectivity

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

11.10.1.8 selectivity_closed

```
real(dp), dimension(num_size_classes) mortality_mod::mortality_class::selectivity_closed
```

11.10.1.9 selectivity_open

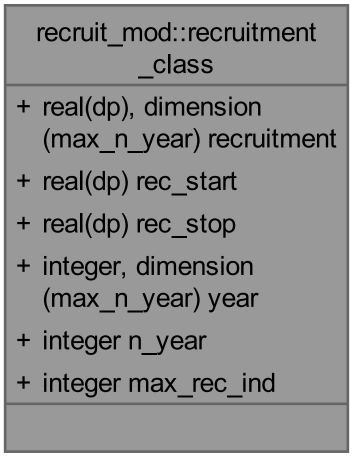
```
real(dp), dimension(num_size_classes) mortality_mod::mortality_class::selectivity_open
```

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

- SRC/ScallopMortality.f90

11.11 recruit_mod::recruitment_class Type Reference

Collaboration diagram for recruit_mod::recruitment_class:


recruit_mod::recruitment_class
+ 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

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

11.11.1 Member Data Documentation

11.11.1.1 max_rec_ind

```
integer recruit_mod::recruitment_class::max_rec_ind
```

11.11.1.2 n_year

```
integer recruit_mod::recruitment_class::n_year
```

11.11.1.3 rec_start

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

11.11.1.4 rec_stop

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

11.11.1.5 recruitment

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

11.11.1.6 year

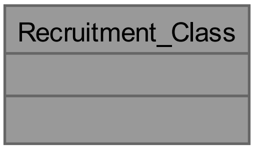
```
integer, dimension(max_n_year) recruit_mod::recruitment_class::year
```

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

- SRC/ScallopRecruit.f90

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

Chapter 12

File Documentation

12.1 SRC/aaaPageOrder.f90 File Reference

12.2 SRC/Globals.f90 File Reference

Modules

- module globals

Functions/Subroutines

- elemental real(dp) function globals::logic_to_double (value)
- real(dp) function, dimension(n, n) globals::matrixinv (x, n)

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::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 = 3.14159265358979311599796346854D0
- real(dp), parameter globals::grams_per_pound = 453.592_dp
- real(dp), parameter globals::meters_per_naut_mile = 1852.D0
- real(dp), parameter globals::feet_per_naut_mile = 6076.12
- 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
- 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/'

12.3 SRC/GridManager.f90 File Reference

Data Types

- type grid_manager_mod::grid_data_class
- type grid_manager_mod::lonlatpoint
- type grid_manager_mod::lonlatvector

Modules

- module grid_manager_mod

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:

- integer function grid_manager_mod::get_region (lat, lon, stratum_real)

  Returns a region number based on stratum 0: not used 1: region_N North region 2: region_S South region 3: region_SW Southwest region 4: region_W West region 5: region_MA Entire Mid-Atlantic Region Inputs: grid: locations of survey data.

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

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:

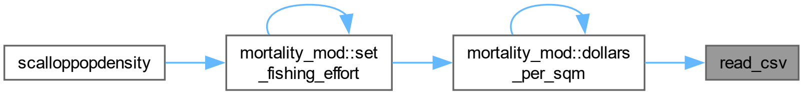
- subroutine write_column_csv (n, f, header, file_name, append)
- subroutine write_csv_h (n, m, f, file_name, nndim, header)
- subroutine read_csv (num_rows, num_cols, file_name, m, nndim)

12.4.1 Function/Subroutine Documentation

12.4.1.1 read_csv()

```
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 )
```

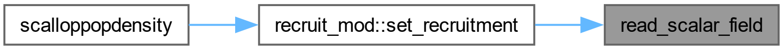
Here is the caller graph for this function:


scalloppopdensity
mortality_mod::set_fishing_effort
mortality_mod::dollars_per_sqm
read_csv

12.4.1.2 read_scalar_field()

```
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:



12.4.1.3 write_2d_scalar_field()

```
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

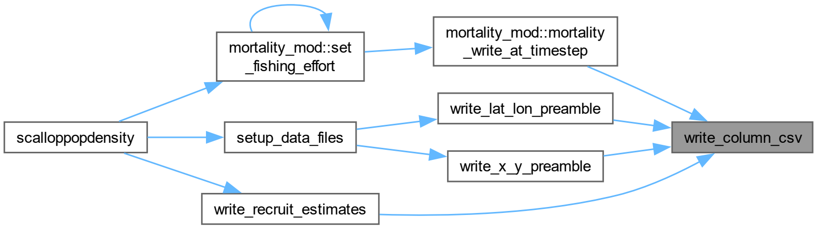
Author

Keston Smith (IBSS corp) June-July 2021

12.4.1.4 write_column_csv()

```
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:



12.4.1.5 write_csv()

```
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 )
```

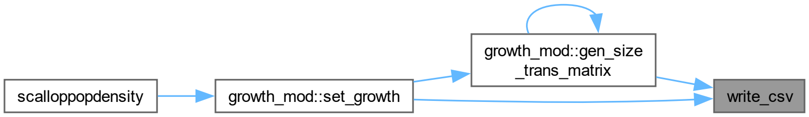
Purpose: Write values of a matrix (f) to a csv file in exponential format. Inputs:

- 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

Author

Keston Smith (IBSS corp) June-July 2021

Here is the caller graph for this function:



12.4.1.6 write_csv_h()

```
subroutine write_csv_h (
             integer, intent(in) n,
             integer, intent(in) m,
             real(dp), dimension(nndim,*), intent(in) f,
             character(*), intent(in) file_name,
             integer, intent(in) nndim,
             character(*), intent(in) header )
```

12.4.1.7 write_vector_scalar_field()

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

12.5 SRC/ScallopGrowth.f90 File Reference

Data Types

- type growth_mod::growth_class

Modules

- module growth_mod

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)

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

12.6 SRC/ScallopMortality.f90 File Reference

Data Types

- type mortality_mod::mortality_class
- type mortality_mod::fishingmortality
- type mortality_mod::dataforplots

Modules

- module mortality_mod

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, save_by_strat)
- 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 mortality_mod::dollars_per_sqm (year, meat_weight_grams)

  Compute value of scallop population at a specific grid location.
- subroutine mortality_mod::scallops_to_counts (meat_weight_grams, cnt10, cnt10to20, cnt20to30, cnt30plus)

  Purpose: Convert Scallop density by shell height and meat wieght to count data. The count data are divided into cnt10-10 or less scallops per pound. cnt10to20 10-20 scallops per pound. cnt20to30 20-30 scallops per pound. cnt30+ 30 or more scallops per pound.
- 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, mortality, 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 @parma[in] expl_biomass ! Expl biomass @parma[in] expl_scallops ! Expl Number of Scallops.

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
- logical, private mortality_mod::save_by_stratum
- 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::usd_per_sqm
- real(dp), dimension(:), allocatable, private mortality_mod::expl_scallops_psqm
- real(dp), dimension(:), allocatable, private mortality_mod::expl_num
- real(dp), dimension(:), allocatable, private mortality_mod::f_mort
- real(dp), dimension(:), allocatable, private mortality_mod::f_mort_raw
- 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::landings_wgt_grams_open
- real(dp), dimension(:), allocatable, private mortality_mod::landings_wgt_grams_closed
- real(dp), dimension(:), allocatable, private mortality_mod::lpue
- real(dp), dimension(:), allocatable, private mortality_mod::fishing_effort
- 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
- real(dp), dimension(num_size_classes), private mortality_mod::landings_at_size_open
- real(dp), dimension(num_size_classes), private mortality_mod::landings_at_size_closed
- type(dataforplots), private mortality_mod::data_select

12.7 SRC/ScallopPopDensity.f90 File Reference

Functions/Subroutines

- program scalloppopdensity
- subroutine read_startup_config (time_steps_per_year, save_by_stratum, 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, save_by_stratum)

  Writes year, UTM-X, UTM-Y, and Depth columns with headers to named file.
- subroutine write_column_csv_by_stratum (n, f, lat, lon, stratum, header, file_name, append)

  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, save_by_stratum, start_year, stop↩_year)
- subroutine write_recruit_estimates (ts, ts_per_year, num_grids, grid, domain_name, save_by_stratum, year, start_year, recruit)

12.7.1 Function/Subroutine Documentation

12.7.1.1 read_startup_config()

```
subroutine read_startup_config (
             integer, intent(out) time_steps_per_year,
             logical, intent(out) save_by_stratum,
             integer, intent(out) start_year,
             integer, intent(out) stop_year,
             character(domain_len), intent(out) domain_name,
             type(dataforplots), intent(out) plot_data_sel )
```

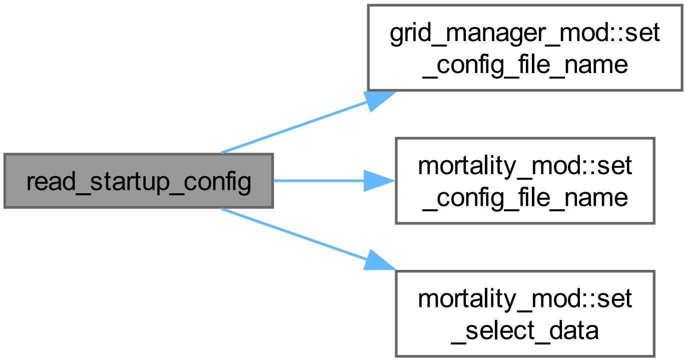
Read Input File.

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

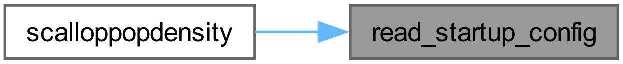
Parameters

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:


read_startup_config
grid_manager_mod::set_config_file_name
mortality_mod::set_config_file_name
mortality_mod::set_select_data

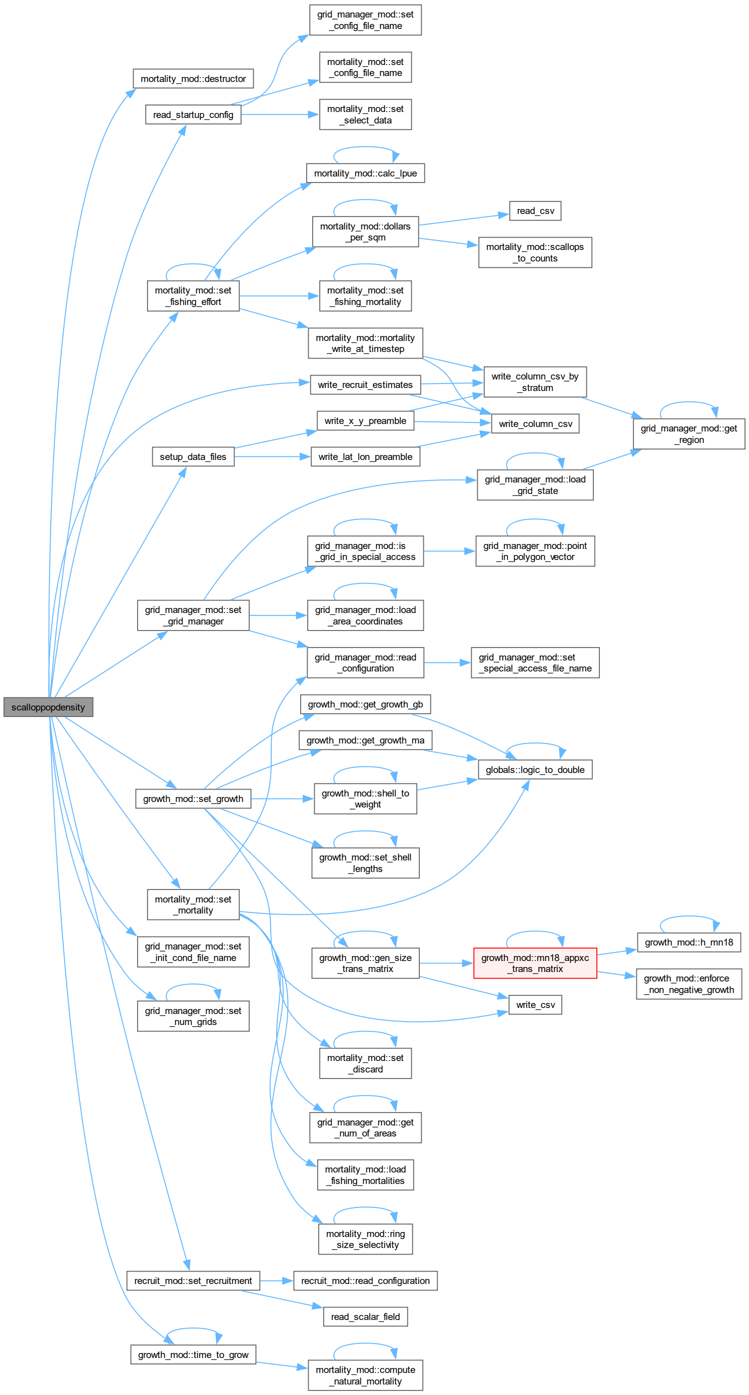
Here is the caller graph for this function:



12.7.1.2 scalloppopdensity()

```
program scalloppopdensity
```

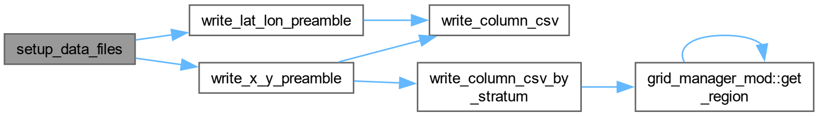
Here is the call graph for this function:


grid_manager_mod::set_config_file_name
mortality_mod::destructor
mortality_mod::set_config_file_name
read_startup_config
mortality_mod::set_select_data
mortality_mod::calc_lpue
read_csv
mortality_mod::dollars_per_sqm
mortality_mod::scallops_to_counts
mortality_mod::set_fishing_effort
mortality_mod::set_fishing_mortality
mortality_mod::mortality_write_at_timestep
write_column_csv_by_stratum
write_recruit_estimates
write_x_y_preamble
write_column_csv
grid_manager_mod::get_region
setup_data_files
write_lat_lon_preamble
grid_manager_mod::load_grid_state
grid_manager_mod::is_grid_in_special_access
grid_manager_mod::point_in_polygon_vector
grid_manager_mod::set_grid_manager
grid_manager_mod::load_area_coordinates
grid_manager_mod::read_configuration
grid_manager_mod::set_special_access_file_name
scalloppopdensity
growth_mod::get_growth_gb
growth_mod::get_growth_ma
globals::logic_to_double
growth_mod::set_growth
growth_mod::shell_to_weight
growth_mod::set_shell_lengths
mortality_mod::set_mortality
growth_mod::h_mn18
grid_manager_mod::set_init_cond_file_name
growth_mod::gen_size_trans_matrix
growth_mod::mn18_appxc_trans_matrix
growth_mod::enforce_non_negative_growth
write_csv
grid_manager_mod::set_num_grids
mortality_mod::set_discard
grid_manager_mod::get_num_of_areas
mortality_mod::load_fishing_mortalities
mortality_mod::ring_size_selectivity
recruit_mod::set_recruitment
recruit_mod::read_configuration
read_scalar_field
growth_mod::time_to_grow
mortality_mod::compute_natural_mortality

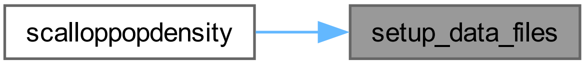
12.7.1.3 setup_data_files()

```
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,
             logical, intent(in) save_by_stratum,
             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:

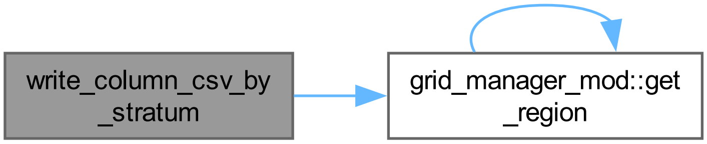


12.7.1.4 write_column_csv_by_stratum()

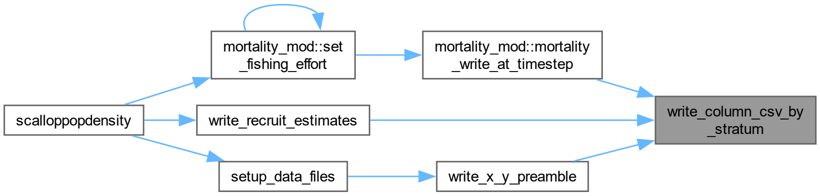
```
subroutine write_column_csv_by_stratum (
            integer, intent(in) n,
            real(dp), dimension(*), intent(in) f,
            real(dp), dimension(*), intent(in) lat,
            real(dp), dimension(*), intent(in) lon,
            real(dp), dimension(*), intent(in) stratum,
            character(*), intent(in) header,
            character(*), intent(in) file_name,
            logical, intent(in) append )
```

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 call graph for this function:



Here is the caller graph for this function:

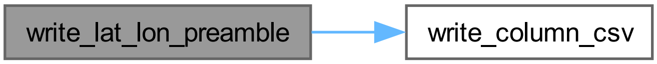


12.7.1.5 write_lat_lon_preamble()

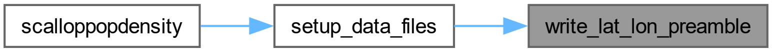
```
subroutine write_lat_lon_preamble (
             integer, intent(in) num_grids,
             type(grid_data_class), dimension(*), intent(in) grid,
             character(*), intent(in) fname )
```

Writes lat and lon columns with headers to named file.

Here is the call graph for this function:



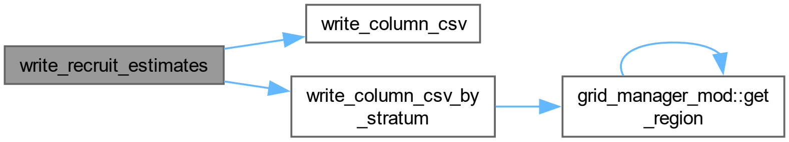
Here is the caller graph for this function:



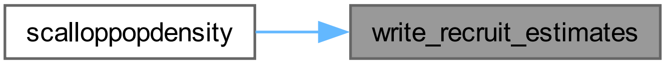
12.7.1.6 write_recruit_estimates()

```
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,
             logical, intent(in) save_by_stratum,
             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:

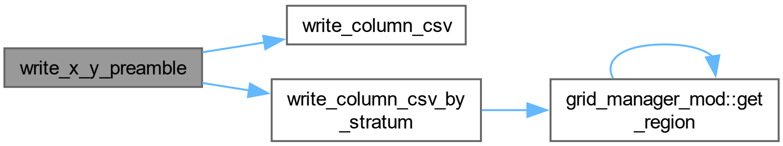


12.7.1.7 write_x_y_preamble()

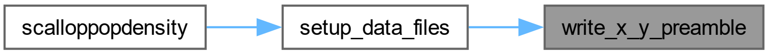
```
subroutine write_x_y_preamble (
             integer, intent(in) num_grids,
             type(grid_data_class), dimension(*), intent(in) grid,
             real(dp), intent(in) yr_offset,
             character(*), intent(in) fname,
             logical, intent(in) save_by_stratum )
```

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:



12.8 SRC/ScallopRecruit.f90 File Reference

Data Types

- type recruit_mod::recruitment_class

Modules

- module recruit_mod

Functions/Subroutines

- subroutine recruit_mod::set_recruitment (recruit, n_grids, dom_name, dom_area, l_inf_mu, k_mu, shell_length_mm, yr_start, yr_stop)

    Set_Recruitment.
- 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.

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::recr_start_year
- integer, private recruit_mod::recr_stop_year
- real(dp), private recruit_mod::recr_period_start
- real(dp), private recruit_mod::recr_period_stop

Index