Thomas Callaghan

Version 0.1

Contents

[GeoSAMS Growth i](#_Toc188507749)

[Thomas Callaghan i](#_Toc188507750)

[Version 0.1 i](#_Toc188507751)

[1 Scallop Population Density 1](#_Toc188507752)

[1.1 Initialize Simulation Parameters 1](#_Toc188507753)

[1.1.1 Read Input 1](#_Toc188507754)

[1.2 Set Grid Manager 2](#_Toc188507755)

[1.2.1 Load Grid and Initial State 2](#_Toc188507756)

[1.2.2 Load Area Coordinates 2](#_Toc188507757)

[1.3 Set Growth 2](#_Toc188507758)

[1.3.1 Shell Length 2](#_Toc188507759)

[1.3.2 Shell to Weigth, in grams 2](#_Toc188507760)

[1.3.3 Compute Growth Parameters, given depth, latitude, and isClosed 2](#_Toc188507761)

[1.3.4 Compute G matrix for given growth parameters 3](#_Toc188507762)

[1.4 Set Recruitment 3](#_Toc188507763)

[1.4.1 For years start\_year to stop\_year 3](#_Toc188507764)

[1.4.2 This method is effectively setting 4](#_Toc188507765)

[1.4.3 It then quantizes recruitment, 4](#_Toc188507766)

[1.5 Set Mortality 4](#_Toc188507767)

[1.5.1 Compute alpha 4](#_Toc188507768)

[1.5.2 Compute Fishing Effort 4](#_Toc188507769)

[1.6 Main Loop 5](#_Toc188507770)

[1.6.1 For each time step 5](#_Toc188507771)

[1.6.2 For each grid 5](#_Toc188507772)

[2 Growth\_Mod 7](#_Toc188507773)

[2.1 Growth Class 7](#_Toc188507774)

[2.1.1 Transition Matrix 8](#_Toc188507775)

[3 Recruit\_Mod 9](#_Toc188507776)

[3.1 Recruitment Class 9](#_Toc188507777)

[4 Mortality\_Mod 10](#_Toc188507778)

[4.1 Read\_Configuration 10](#_Toc188507779)

[4.2 Load\_Fishing\_Mortalities 10](#_Toc188507780)

[4.3 Set Mortality 10](#_Toc188507781)

[4.4 Compute Alpha 10](#_Toc188507782)

[4.5 Compute Selectivity 10](#_Toc188507783)

[4.6 Compute Discard 10](#_Toc188507784)

[4.7 Mortality\_Write\_At\_Timestep 11](#_Toc188507785)

[5 Grid Manager Mod 12](#_Toc188507786)

[5.1 Grid Manager Class 12](#_Toc188507787)

[5.1.1 Brief 12](#_Toc188507788)

[5.1.2 Set Grid Manager 12](#_Toc188507789)

[5.1.3 Read\_Configuration 12](#_Toc188507790)

[5.1.4 Load\_Area\_Coordinates 12](#_Toc188507791)

[5.1.5 Is\_Grid\_In\_Special\_Access 12](#_Toc188507792)

[5.1.6 Set\_Config\_File\_Name 12](#_Toc188507793)

[5.2 Point In Polygon 13](#_Toc188507794)

[5.2.1 Point\_In\_Polygon\_Points 13](#_Toc188507795)

[5.2.2 Point\_In\_Polygon\_Array 13](#_Toc188507796)

[5.2.3 Point\_In\_Polygon\_Vector 13](#_Toc188507797)

[6 Common Parameters 14](#_Toc188507798)

[7 Modules Index 15](#_Toc188507799)

[7.1 Modules List 15](#_Toc188507800)

[8 Data Type Index 16](#_Toc188507801)

[8.1 Data Types List 16](#_Toc188507802)

[9 File Index 17](#_Toc188507803)

[9.1 File List 17](#_Toc188507804)

[10 Module Documentation 18](#_Toc188507805)

[10.1 globals Module Reference 18](#_Toc188507806)

[10.1.1 Functions/Subroutines 18](#_Toc188507807)

[10.1.2 Variables 18](#_Toc188507808)

[10.1.3 Function/Subroutine Documentation 19](#_Toc188507809)

[10.1.4 Variable Documentation 21](#_Toc188507810)

[10.2 grid\_manager\_mod Module Reference 24](#_Toc188507811)

[10.2.1 Data Types 24](#_Toc188507812)

[10.2.2 Functions/Subroutines 24](#_Toc188507813)

[10.2.3 Variables 24](#_Toc188507814)

[10.2.4 Function/Subroutine Documentation 25](#_Toc188507815)

[10.2.5 Variable Documentation 31](#_Toc188507816)

[10.3 growth\_mod Module Reference 32](#_Toc188507817)

[10.3.1 Data Types 32](#_Toc188507818)

[10.3.2 type growth\_classFunctions/Subroutines 32](#_Toc188507819)

[10.3.3 Variables 33](#_Toc188507820)

[10.3.4 Function/Subroutine Documentation 33](#_Toc188507821)

[10.3.5 Variable Documentation 43](#_Toc188507822)

[10.4 mortality\_mod Module Reference 44](#_Toc188507823)

[10.4.1 Data Types 44](#_Toc188507824)

[10.4.2 Functions/Subroutines 44](#_Toc188507825)

[10.4.3 Variables 45](#_Toc188507826)

[10.4.4 Function/Subroutine Documentation 46](#_Toc188507827)

[10.4.5 Variable Documentation 53](#_Toc188507828)

[10.5 recruit\_mod Module Reference 56](#_Toc188507829)

[10.5.1 Data Types 56](#_Toc188507830)

[10.5.2 type recruitment\_classFunctions/Subroutines 56](#_Toc188507831)

[10.5.3 Variables 56](#_Toc188507832)

[10.5.4 Function/Subroutine Documentation 56](#_Toc188507833)

[10.5.5 Variable Documentation 59](#_Toc188507834)

[11 Data Type Documentation 60](#_Toc188507835)

[11.1 mortality\_mod::dataforplots Type Reference 60](#_Toc188507836)

[11.1.1 Public Attributes 60](#_Toc188507837)

[11.1.2 Member Data Documentation 61](#_Toc188507838)

[11.2 mortality\_mod::fishingmortality Type Reference 62](#_Toc188507839)

[11.2.1 Public Attributes 62](#_Toc188507840)

[11.2.2 Member Data Documentation 62](#_Toc188507841)

[11.3 Grid\_Data\_Class Module Reference 63](#_Toc188507842)

[11.4 grid\_manager\_mod::grid\_data\_class Type Reference 64](#_Toc188507843)

[11.4.1 Public Attributes 64](#_Toc188507844)

[11.4.2 Member Data Documentation 65](#_Toc188507845)

[11.5 Growth\_Class Module Reference 66](#_Toc188507846)

[11.5.1 Detailed Description 66](#_Toc188507847)

[11.6 growth\_mod::growth\_class Type Reference 67](#_Toc188507848)

[11.6.1 Public Attributes 67](#_Toc188507849)

[11.6.2 Member Data Documentation 67](#_Toc188507850)

[11.7 grid\_manager\_mod::lonlatpoint Type Reference 69](#_Toc188507851)

[11.7.1 Public Attributes 69](#_Toc188507852)

[11.7.2 Member Data Documentation 69](#_Toc188507853)

[11.8 grid\_manager\_mod::lonlatvector Type Reference 70](#_Toc188507854)

[11.8.1 Public Attributes 70](#_Toc188507855)

[11.8.2 Member Data Documentation 70](#_Toc188507856)

[11.9 Mortality\_Class Module Reference 71](#_Toc188507857)

[11.9.1 Detailed Description 71](#_Toc188507858)

[11.10 mortality\_mod::mortality\_class Type Reference 72](#_Toc188507859)

[11.10.1 Public Attributes 72](#_Toc188507860)

[11.10.2 Member Data Documentation 73](#_Toc188507861)

[11.11 recruit\_mod::recruitment\_class Type Reference 74](#_Toc188507862)

[11.11.1 Public Attributes 74](#_Toc188507863)

[11.11.2 Member Data Documentation 74](#_Toc188507864)

[11.12 Recruitment\_Class Module Reference 75](#_Toc188507865)

[11.12.1 Detailed Description 75](#_Toc188507866)

[12 File Documentation 76](#_Toc188507867)

[12.1 SRC/aaaPageOrder.f90 File Reference 76](#_Toc188507868)

[12.2 SRC/Globals.f90 File Reference 77](#_Toc188507869)

[12.2.1 Modules 77](#_Toc188507870)

[12.2.2 Functions/Subroutines 77](#_Toc188507871)

[12.2.3 Variables 77](#_Toc188507872)

[12.3 SRC/GridManager.f90 File Reference 79](#_Toc188507873)

[12.3.1 Data Types 79](#_Toc188507874)

[12.3.2 Modules 79](#_Toc188507875)

[12.3.3 Functions/Subroutines 79](#_Toc188507876)

[12.3.4 Variables 80](#_Toc188507877)

[12.4 SRC/IORoutines.f90 File Reference 81](#_Toc188507878)

[12.4.1 Functions/Subroutines 81](#_Toc188507879)

[12.4.2 Function/Subroutine Documentation 81](#_Toc188507880)

[12.5 SRC/ScallopGrowth.f90 File Reference 83](#_Toc188507881)

[12.5.1 Data Types 83](#_Toc188507882)

[12.5.2 type growth\_mod::growth\_classModules 83](#_Toc188507883)

[12.5.3 Functions/Subroutines 83](#_Toc188507884)

[12.5.4 Variables 84](#_Toc188507885)

[12.6 SRC/ScallopMortality.f90 File Reference 85](#_Toc188507886)

[12.6.1 Data Types 85](#_Toc188507887)

[12.6.2 Modules 85](#_Toc188507888)

[12.6.3 Functions/Subroutines 85](#_Toc188507889)

[12.6.4 Variables 86](#_Toc188507890)

[12.7 SRC/ScallopPopDensity.f90 File Reference 88](#_Toc188507891)

[12.7.1 Functions/Subroutines 88](#_Toc188507892)

[12.7.2 Function/Subroutine Documentation 88](#_Toc188507893)

[12.8 SRC/ScallopRecruit.f90 File Reference 93](#_Toc188507894)

[12.8.1 Data Types 93](#_Toc188507895)

[12.8.2 type recruit\_mod::recruitment\_classModules 93](#_Toc188507896)

[12.8.3 Functions/Subroutines 93](#_Toc188507897)

[12.8.4 Variables 93](#_Toc188507898)

[13 Index 94](#_Toc188507899)

# Scallop Population Density

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

Jun 1st @ 0600 is day 0.25 which is = 0.25 /365.2425 = 0.00068 years

June 1st @ 1200 is day 0.50 which is = 0.50 /365.2425 = 0.00137

June 1st @ 1800 is day 0.75 which is = 0.75 /365.2425 = 0.00205

June 1st @ 2359 is day 0.99 which is = 0.99931/365.2425 = 0.002736

Jun2 2nd @ 0000 is day 1 which is = 1.00000/365.2425 = 0.00274

Jun2 2nd @ 2400 is day 2 which is = 2.00000/365.2425 = 0.00548

Dec 31st @ 2400 is day 214 which is = 214. /365.2425 = 0.58591

Jan 1st @ 2400 is day 215 which is = 215. /365.2425 = 0.58865

= 1 + DayOfYear(12,31) - DayOfYear(5,31)

Apr 10 @ 2400 is day 314 which is = 314. /365.2425 = 0.85970

if leap year 315 which is = 315. /365.2425 = 0.86244 However, leap year will be handled in the main loop in which it is considered only for the current year

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

X\_Y\_BIOM\_2022\_DN Initial state as of June 1, 2022 @ 00:00, i.e. May 31, 2022 @ 24:00

X\_Y\_BIOM\_2023\_DN Growth state as of May 31, 2023 @ 24:00, results for 1st year growth

X\_Y\_BIOM\_2024\_DN Growth state as of May 31, 2024 @ 24:00, results for 2nd year growth

X\_Y\_BIOM\_2025\_DN Growth state as of May 31, 2025 @ 24:00, results for 3rd year growth

X\_Y\_BIOM\_2026\_DN Growth state as of May 31, 2026 @ 24:00, results for 4th year growth

## Initialize Simulation Parameters

### Read Input

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

ScallopPopDensity.exe **Scallop.cfg**

Time steps per Year: number of time steps each year

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

Mortality Config File

Recruit Config File

Grid Manager Config File

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

Start Year

Stop Year

Domain Name

MA

GB

AL, for both MA and GB

## Set Grid Manager

### Load Grid and Initial State

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

Data/bin5mmYYYY[MA|GB].csv

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

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

### Load Area Coordinates

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

## Set Growth

The simulation then instantiates parameters that define how growth occurs

### Shell Length

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

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

### Shell to Weigth, in grams

GB

MA

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

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

### Compute G matrix for given growth parameters

From MN18 p. 1312, 1313, where

For each size class, *k*

## Set Recruitment

The simulation next instantiates how recruitment will be handled.

### For years start\_year to stop\_year

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

### This method is effectively setting

For year in start\_year to stop\_year

Year\_index = year - start\_year + 1

for year\_index in [1..max]

recruitment(year\_index) = RecruitEstimate(random year index)

year(year\_index) = year

rec\_start = Start Period, typically 0/365.2425, or January 1st

rec\_stop = Stop Period, typically 100/365.2425, or April 10

### It then quantizes recruitment,

For each grid, n

For each class, j

If (length(j) <= L30mm) recruit(n).max\_rec\_ind = j

## Set Mortality

The simulation next sets mortality values.

**Table 1 Mortality**

|  |  |  |  |
| --- | --- | --- | --- |
| **Region** | **Adult** | **Incidental** | **Base Length**  **l0** |
| **MA** | **25%** | **5%** | **65.0** |
| **GB** | **20%** | **10%** | **70.0** |

### Compute alpha

### Compute Fishing Effort

#### Compute landings at size for each grid location

Given The number of scallops per square meter:

and the exploitable biomass in grams per square meter

#### Compute landings by weight

This is also considered total\_catch

#### Total catch is used compute fishing effort

## Main Loop

### For each time step

#### 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.

### For each grid

#### Compute natural mortality

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

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

Mid-Atlantic:

Georges Bank:

Finally

#### Adjust population state based on von Bertalanffy growth

#### Adjust population state based on von Bertalanffy growth

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

#### Compute Overall Mortality

#### Compute effect of mortality to arrive at new state

# Growth\_Mod

## Growth Class

The scallop state\_vector at each node in the domain is a vector of length representing the abundance of scallops in size classes .

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 ).

Growth in GeoSAMS is based off of von Bertanlffy growth.

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

We assume normal distribution on and with all distribution parameters independent.

The shell height of the ith individual at time , depends on the random effects ( and ) as well as the mean slope and intercept:

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

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

For each time step,

Computes mortality based on current state\_vector.

Computes increase in population due to recruitment, ,if within recruitment months, i.e. Jan to April 10th

Adjusts population based on von Bertalanffy growth

where G is the transition matrix

Compute overall mortality, **M**

Compute new state\_vector

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."

### 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

#### Function H(x, sigma, omega)

Given (MN18 Appendix B)

denotes the normal **cumulative** distribution function.

denotes the normal **density** function.

#### Normal Cumulative Distribution Function

#### Normal Density Function

# Recruit\_Mod

## Recruitment Class

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

recruitment() = data read in from randomly chosen recruit file

year() = simulation year

rec\_start = a decimal value given as day of the year divided by 365.2425. Typically 0, which would be January 1

rec\_stop = decimal value given as day of the year divided by 365.2425. Typically 100/365.2425, which is April 10th.

# Mortality\_Mod

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

Set\_Mortality

Instantiates private members for this class.

Reads in its configuration parameters and stores to private members.

Loads Fishing Mortalities, if enabled by GridManager

Sets up a repository for key values to allow offline analysis

Loads historical data for Fishing Effort

Set selectivity as computed by Ring\_Size\_Selectivity based on shell length and grid location

## Read\_Configuration

Opens the configuration file specified in the simulation configuration file and as set by *Set\_Config\_File\_Name*

## Load\_Fishing\_Mortalities

Opens the configuration file specified in the Mortality configuration file and as set by *Set\_Fishing\_Mortality*

## Set Mortality

Mortality =

## Compute Alpha

## Compute Selectivity

Assign size class fishing selectivity based on increasing logistic function

## Compute Discard

## Mortality\_Write\_At\_Timestep

# Grid Manager Mod

## Grid Manager Class

### 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*

### 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.

### 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

### 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

### 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

### 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

## Point In Polygon

The Point\_In\_Polygon\_Vector method is used to find if a point is in a polygon. The ***Grid\_Manager*** *also* supports polygon data representation as an array of LonLatPoint points vial ***Point\_In\_Polygon\_Points*** *or* as a n by 2, 2-dimensional array, where n is a maximum of *max\_sides* edges.

### Point\_In\_Polygon\_Points

### Point\_In\_Polygon\_Array

### Point\_In\_Polygon\_Vector

# Common Parameters

# Modules Index

## Modules List

Here is a list of all modules with brief descriptions:

**globals**

**grid\_manager\_mod**

**growth\_mod**

**mortality\_mod**

**recruit\_mod**

# Data Type Index

## Data Types List

Here are the data types with brief descriptions:

**mortality\_mod::dataforplots**

**mortality\_mod::fishingmortality**

**Grid\_Data\_Class**

**grid\_manager\_mod::grid\_data\_class**

**Growth\_Class (Subroutines that determine expected growth of scallops )**

**growth\_mod::growth\_class**

**grid\_manager\_mod::lonlatpoint**

**grid\_manager\_mod::lonlatvector**

**Mortality\_Class (Subroutines that determine expected mortality of scallops )**

**mortality\_mod::mortality\_class**

**recruit\_mod::recruitment\_class**

**Recruitment\_Class (Subroutines that determine expected growth of scallops )**

# File Index

## File List

Here is a list of all files with brief descriptions:

**SRC/aaaPageOrder.f90**

**SRC/Globals.f90**

**SRC/GridManager.f90**

**SRC/IORoutines.f90**

**SRC/ScallopGrowth.f90**

**SRC/ScallopMortality.f90**

**SRC/ScallopPopDensity.f90**

**SRC/ScallopRecruit.f90**

# Module Documentation

## globals Module Reference

### Functions/Subroutines

elemental real(**dp**) function **logic\_to\_double** (value)

real(**dp**) function, dimension(n, n) **matrixinv** (x, n)

logical function **leap\_year** (year)

logical function **divby** (y, val)

integer function **dayofyear** (m, d)

logical function **is\_nan** (x)

### Variables

integer, parameter **sp** = selected\_real\_kind(6, 37)

integer, parameter **dp** = selected\_real\_kind(15, 307)

integer, parameter **qp** = selected\_real\_kind(33, 4931)

integer, parameter **ndim** = 12000

integer, parameter **shell\_len\_max** = 150

integer, parameter **shell\_len\_min** = 30

integer, parameter **shell\_len\_delta** = 5

integer, parameter **num\_size\_classes** = (**shell\_len\_max** - **shell\_len\_min**) / **shell\_len\_delta** + 1

integer, parameter **max\_num\_years** = 50

integer, parameter **max\_num\_areas** = 25

integer, parameter **max\_sides** = 8

integer, parameter **region\_none** =0

integer, parameter **region\_n** =1

integer, parameter **region\_s** =2

integer, parameter **region\_sw** =3

integer, parameter **region\_w** =4

integer, parameter **region\_ma** =5

integer, parameter **region\_gbk** = 1

integer, parameter **region\_mab** = 5

integer, parameter **tag\_len** = 40

integer, parameter **value\_len** = 30

integer, parameter **comment\_len** = 80

integer, parameter **line\_len** = **tag\_len**+**value\_len**+**comment\_len**

integer, parameter **fname\_len** = 100

integer, parameter **form\_len** = 20

integer, parameter **input\_str\_len** = 100

integer, parameter **csv\_line\_len** = 2000

integer, parameter **domain\_len** = 2

integer, parameter **read\_dev** = 69

integer, parameter **write\_dev** = 63

real(**dp**), parameter **zero\_threshold** = 1.0D-99

real(**dp**), parameter **pi** = 3.14159265358979323846264338327950288D0

real(**dp**), parameter **grams\_per\_pound** = 453.592\_dp

real(**dp**), parameter **meters\_per\_naut\_mile** = 1852.D0

real(**dp**), parameter **grams\_per\_metric\_ton** = 1000000.\_dp

real(**dp**), parameter **grid\_area\_sqm** = **meters\_per\_naut\_mile**\*\*2

real(**dp**), parameter **tow\_area\_sqm** = 4516.\_dp

real(**dp**), parameter **one\_scallop\_per\_tow** = 1.D0 / **tow\_area\_sqm**

real(**dp**), parameter **ma\_gb\_border** = -70.5

real(**dp**), parameter **days\_in\_year** =365+0.25-0.01+0.0025

character(\*), parameter **term\_red** = ''//achar(27)//'[31m'

character(\*), parameter **term\_yel** = ''//achar(27)//'[33m'

character(\*), parameter **term\_grn** = ''//achar(27)//'[92m'

character(\*), parameter **term\_blu** = ''//achar(27)//'[94m'

character(\*), parameter **term\_blk** = ''//achar(27)//'[0m'

character(\*), parameter **init\_cond\_dir** = 'InitialCondition/'

character(\*), parameter **growth\_out\_dir** = 'GrowthOutput/'

character(\*), parameter **rec\_input\_dir** = 'RecruitEstimates/'

character(\*), parameter **rec\_output\_dir** = 'RecruitField/'

character(\*), parameter **output\_dir** = 'Results/'

character(\*), parameter **config\_dir\_sim** = 'Configuration/Simulation/'

character(\*), parameter **config\_dir\_interp** = 'Configuration/Interpolation/'

character(\*), parameter **config\_dir\_special** = 'Configuration/SpecialAccess/'

character(\*), parameter **grid\_dir** = 'Grids/'

character(\*), parameter **data\_dir** = 'Data/'

character(\*), parameter **anal\_dir** = 'Analysis/'

integer, parameter **num\_regions** = 2

character(3), dimension(**num\_regions**) **rgn** = (/ '\_GB', '\_MA'/)

### Function/Subroutine Documentation

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

Here is the call graph for this function:

Here is the caller graph for this function:

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

Here is the call graph for this function:

Here is the caller graph for this function:

#### logical function globals::is\_nan (real(dp), intent(in) *x*)

Here is the call graph for this function:

Here is the caller graph for this function:

#### logical function globals::leap\_year (integer *year*)

Here is the call graph for this function:

Here is the caller graph for this function:

#### 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:

#### 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:

### Variable Documentation

#### character(\*), parameter globals::anal\_dir = 'Analysis/'

#### integer, parameter globals::comment\_len = 80

#### character(\*), parameter globals::config\_dir\_interp = 'Configuration/Interpolation/'

#### character(\*), parameter globals::config\_dir\_sim = 'Configuration/Simulation/'

#### character(\*), parameter globals::config\_dir\_special = 'Configuration/SpecialAccess/'

#### integer, parameter globals::csv\_line\_len = 2000

#### character(\*), parameter globals::data\_dir = 'Data/'

#### real(dp), parameter globals::days\_in\_year =365+0.25-0.01+0.0025

#### integer, parameter globals::domain\_len = 2

#### integer, parameter globals::dp = selected\_real\_kind(15, 307)

#### integer, parameter globals::fname\_len = 100

#### integer, parameter globals::form\_len = 20

#### real(dp), parameter globals::grams\_per\_metric\_ton = 1000000.\_dp

#### real(dp), parameter globals::grams\_per\_pound = 453.592\_dp

#### real(dp), parameter globals::grid\_area\_sqm = meters\_per\_naut\_mile\*\*2

#### character(\*), parameter globals::grid\_dir = 'Grids/'

#### character(\*), parameter globals::growth\_out\_dir = 'GrowthOutput/'

#### character(\*), parameter globals::init\_cond\_dir = 'InitialCondition/'

#### integer, parameter globals::input\_str\_len = 100

#### integer, parameter globals::line\_len = tag\_len+value\_len+comment\_len

#### real(dp), parameter globals::ma\_gb\_border = -70.5

#### integer, parameter globals::max\_num\_areas = 25

#### integer, parameter globals::max\_num\_years = 50

#### integer, parameter globals::max\_sides = 8

#### real(dp), parameter globals::meters\_per\_naut\_mile = 1852.D0

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

#### integer, parameter globals::num\_regions = 2

#### integer, parameter globals::num\_size\_classes = (shell\_len\_max - shell\_len\_min) / shell\_len\_delta + 1

#### real(dp), parameter globals::one\_scallop\_per\_tow = 1.D0 / tow\_area\_sqm

#### character(\*), parameter globals::output\_dir = 'Results/'

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

#### integer, parameter globals::qp = selected\_real\_kind(33, 4931)

#### integer, parameter globals::read\_dev = 69

#### character(\*), parameter globals::rec\_input\_dir = 'RecruitEstimates/'

#### character(\*), parameter globals::rec\_output\_dir = 'RecruitField/'

#### integer, parameter globals::region\_gbk = 1

#### integer, parameter globals::region\_ma =5

#### integer, parameter globals::region\_mab = 5

#### integer, parameter globals::region\_n =1

#### integer, parameter globals::region\_none =0

#### integer, parameter globals::region\_s =2

#### integer, parameter globals::region\_sw =3

#### integer, parameter globals::region\_w =4

#### character(3), dimension(num\_regions) globals::rgn = (/ '\_GB', '\_MA'/)

#### integer, parameter globals::shell\_len\_delta = 5

#### integer, parameter globals::shell\_len\_max = 150

#### integer, parameter globals::shell\_len\_min = 30

#### integer, parameter globals::sp = selected\_real\_kind(6, 37)

#### integer, parameter globals::tag\_len = 40

#### character(\*), parameter globals::term\_blk = ''//achar(27)//'[0m'

#### character(\*), parameter globals::term\_blu = ''//achar(27)//'[94m'

#### character(\*), parameter globals::term\_grn = ''//achar(27)//'[92m'

#### character(\*), parameter globals::term\_red = ''//achar(27)//'[31m'

#### character(\*), parameter globals::term\_yel = ''//achar(27)//'[33m'

#### real(dp), parameter globals::tow\_area\_sqm = 4516.\_dp

#### integer, parameter globals::value\_len = 30

#### integer, parameter globals::write\_dev = 63

#### real(dp), parameter globals::zero\_threshold = 1.0D-99

#### 

## 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:*

### 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**

### Function/Subroutine Documentation

#### 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:

#### 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:

#### integer function grid\_manager\_mod::load\_area\_coordinates

Here is the call graph for this function:

Here is the caller graph for this function:

#### 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:

Here is the caller graph for this function:

#### 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:

Here is the caller graph for this function:

#### 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:

#### 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:

#### 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:

#### 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:

#### subroutine grid\_manager\_mod::set\_grid\_manager (real(dp), dimension(1:ngrids, 1:num\_size\_classes), intent(out) *state\_mat*, type(grid\_data\_class), dimension(\*), intent(out) *grid*, integer, intent(inout) *ngrids*, character(domain\_len), intent(in) *dom\_name*)

Initializes growth for startup.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 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:

#### 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:

#### 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:

### Variable Documentation

#### type(lonlatvector), dimension(max\_num\_areas), private grid\_manager\_mod::area[private]

#### character(fname\_len), private grid\_manager\_mod::config\_file\_name[private]

#### character(domain\_len), private grid\_manager\_mod::domain\_name[private]

#### character(fname\_len), private grid\_manager\_mod::init\_cond\_fname[private]

#### integer, private grid\_manager\_mod::num\_areas[private]

#### integer, private grid\_manager\_mod::num\_grids[private]

#### character(fname\_len), private grid\_manager\_mod::special\_accesss\_fname[private]

#### logical, private grid\_manager\_mod::use\_spec\_access\_data[private]

#### 

## growth\_mod Module Reference

### Data Types

### type growth\_classFunctions/Subroutines

subroutine **set\_growth** (growth, grid, shell\_lengths, num\_ts, ts\_per\_year, dom\_name, dom\_area, state\_mat, weight\_grams, ngrids)

*Initializes growth for startup.*

real(**dp**) function, dimension(1:**num\_size\_classes**, 1:**num\_size\_classes**) **gen\_size\_trans\_matrix** (l\_inf\_mu, l\_inf\_sd, k\_mu, k\_sd, shell\_lengths, method)

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

real(**dp**) function, dimension(**num\_size\_classes**) **set\_shell\_lengths** (length\_min, length\_delta)

*setup shell shell\_lengths intervals*

subroutine **get\_growth\_gb** (depth, lat, is\_closed, l\_inf\_mu, k\_mu, l\_inf\_sd, k\_sd)

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

subroutine **get\_growth\_ma** (depth, lat, is\_closed, l\_inf\_mu, k\_mu, l\_inf\_sd, k\_sd)

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

real(**dp**) function, dimension(**num\_size\_classes**, **num\_size\_classes**) **mn18\_appxc\_trans\_matrix** (l\_inf\_mu, k\_mu, l\_inf\_sd, k\_sd, shell\_lengths)

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

subroutine **increment\_mean\_std** (l\_inf\_mu, k\_mu, l\_inf\_sd, k\_sd, size, mu, sigma)

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

real(**dp**) function **h\_mn18** (x, sigma, w)

*Given (MN18 Appendix B)*

real(**dp**) function **norm\_cumul\_dist\_fcn** (x, mu, sigma)

*Computation of normal cumulative distribution function.*

real(**dp**) function **norm\_density\_fcn** (x, mu, sigma)

*Computation of normal density function.*

subroutine **enforce\_non\_negative\_growth** (g)

real(**dp**) function, dimension(**num\_size\_classes**) **time\_to\_grow** (ts, growth, mortality, recruit, state\_vector, fishing\_effort, year, longitude)

*Computes growth in scallop population.*

elemental real(**dp**) function **shell\_to\_weight** (shell\_length\_mm, is\_closed, depth, latitude, longitude)

*Computes weight given a shell height.*

subroutine **gamma\_inc\_values** (n\_data, a, x, fx)

### Variables

integer, parameter **growth\_param\_size** = 4

integer, private **num\_grids**

character(**domain\_len**), private **domain\_name**

real(**dp**), private **domain\_area\_sqm**

integer, private **num\_time\_steps**

integer, private **time\_steps\_year**

real(**dp**), private **delta\_time**

logical, private **show\_recruits\_msg**

### Function/Subroutine Documentation

#### 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:

#### 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:

#### 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.

##### 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:

#### 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:

#### 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:

#### 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)

denotes the normal **cumulative** distribution function.

denotes the normal **density** function.

WAS

where

##### 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:

#### 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:

#### 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

##### 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:

Here is the caller graph for this function:

#### 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.

##### 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:

#### 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.

##### 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:

#### 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 |
| 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:

#### 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

##### 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:

#### 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

For Georges Bank

##### 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:

#### 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,

Computes mortality based on current state.

Computes increase in population due to recruitment, ,if within recruitment months, i.e. Jan to April 10th

Adjusts population based on von Bertalanffy growth

Compute overall mortality, **M**

Compute new state

##### 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:

### Variable Documentation

#### real(dp), private growth\_mod::delta\_time[private]

#### real(dp), private growth\_mod::domain\_area\_sqm[private]

#### character(domain\_len), private growth\_mod::domain\_name[private]

#### integer, parameter growth\_mod::growth\_param\_size = 4

#### integer, private growth\_mod::num\_grids[private]

#### integer, private growth\_mod::num\_time\_steps[private]

#### logical, private growth\_mod::show\_recruits\_msg[private]

#### integer, private growth\_mod::time\_steps\_year[private]

#### 

## 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)

subroutine **load\_fishing\_mortalities** ()

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

elemental real(**dp**) function **ring\_size\_selectivity** (shell\_length, is\_closed, longitude)

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

real(**dp**) function, dimension(**num\_grids**) **set\_fishing\_effort** (year, ts, state\_mat, weight\_grams, mortality, grid)

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

real(**dp**) function, dimension(1:**num\_size\_classes**) **compute\_natural\_mortality** (max\_rec\_ind, mortality, state\_vector, longitude)

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

elemental real(**dp**) function **set\_fishing\_mortality** (grid, year, use\_f\_loc, f\_loc)

*Computes Fishing Mortality.*

subroutine **set\_config\_file\_name** (fname)

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

subroutine **set\_fishing\_mort\_file\_name** (fname)

subroutine **read\_configuration** ()

*Read\_Configuration.*

subroutine **mortality\_write\_at\_timestep** (year, ts, state\_mat, weight\_grams, grid)

*Initializes growth for startup.*

elemental real(**dp**) function **set\_discard** (length, selectivity, cull\_size, discard, is\_closed)

*Computes element of discard vector.*

elemental real(**dp**) function **calc\_lpue** (expl\_biomass, expl\_scallops)

*Computes catch as pounds per day.*

### Variables

character(**fname\_len**), private **config\_file\_name**

character(**fname\_len**), private **fishing\_mort\_fname**

type(**fishingmortality**), dimension(**max\_num\_years**), private **fmort\_list**

logical, private **use\_spec\_access\_data**

integer, private **num\_in\_list**

integer, private **num\_grids**

integer, private **num\_areas**

character(**domain\_len**), private **domain\_name**

real(**dp**), private **domain\_area\_sqm**

integer, private **num\_time\_steps**

integer, private **ts\_per\_year**

real(**dp**), private **delta\_time**

real(**dp**), private **fishing\_mort**

real(**dp**), private **alpha\_mort**

real(**dp**), private **ma\_cull\_size\_mm**

real(**dp**), private **ma\_discard**

real(**dp**), private **gb\_cull\_size\_mm**

real(**dp**), private **gb\_discard**

real(**dp**), private **ma\_fselect\_a**

real(**dp**), private **ma\_fselect\_b**

real(**dp**), private **gbc\_fselect\_a**

real(**dp**), private **gbc\_fselect\_b**

real(**dp**), private **gbo\_fselect\_a**

real(**dp**), private **gbo\_fselect\_b**

real(**dp**), private **ma\_mort\_adult**

real(**dp**), private **ma\_incidental**

real(**dp**), private **ma\_length\_0**

real(**dp**), private **gb\_mort\_adult**

real(**dp**), private **gb\_incidental**

real(**dp**), private **gb\_length\_0**

real(**dp**), private **lpue\_slope**

real(**dp**), private **lpue\_slope2**

real(**dp**), private **lpue\_intercept**

integer, private **max\_per\_day**

real(**dp**), private **max\_time\_hpd**

real(**dp**), private **dredge\_width\_m**

real(**dp**), private **towing\_speed\_knots**

real(**dp**), dimension(:), allocatable, private **expl\_biomass\_gpsqm**

real(**dp**), dimension(:), allocatable, private **expl\_scallops\_psqm**

real(**dp**), dimension(:), allocatable, private **f\_mort**

real(**dp**), dimension(:), allocatable, private **landings\_by\_num**

real(**dp**), dimension(:), allocatable, private **landings\_wgt\_grams**

real(**dp**), dimension(:), allocatable, private **lpue**

real(**dp**), dimension(:), allocatable, private **fishing\_effort**

real(**dp**), dimension(:), allocatable, private **landings\_accum**

real(**dp**), dimension(:), allocatable, private **landings\_wgt\_accum**

real(**dp**), dimension(:), allocatable, private **lpue\_accum**

real(**dp**), dimension(**num\_size\_classes**), private **expl\_scallops\_psqm\_at\_size**

real(**dp**), dimension(**num\_size\_classes**), private **landings\_at\_size**

type(**dataforplots**), private **data\_select**

### Function/Subroutine Documentation

#### 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.

##### Parameters

|  |  |  |
| --- | --- | --- |
| in | *expl\_biomass* | ! Expl biomass |
| in | *expl\_scallops* | ! Expl Number of Scallops |
| out | *dredge\_time\_hrs* | ! dredge bottom time |
| out | *dredge\_area\_sqnm* | ! area swept per day |

EBiomass/ENumber = ESize Total Weight of a Tow / Number of scallops caught = mean weight of individual scallop **expl\_biomass\_gpsqm(grid)** \* 4516 / **expl\_scallops\_psqm(grid)** \* 4516 = mean\_expl\_wght\_g xxxx xxxx

Here is the call graph for this function:

Here is the caller graph for this function:

#### 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.

A similar formula for GB Open:

Decreasing logistic function,

TODO, current alpha equation is:

where is 65 if MA or 70 if GB

Finally

##### 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:

#### subroutine mortality\_mod::destructor

Here is the caller graph for this function:

#### 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:

#### subroutine mortality\_mod::mortality\_write\_at\_timestep (integer, intent(in) *year*, integer, intent(in) *ts*, real(dp), dimension(1:num\_grids, 1:num\_size\_classes), intent(in) *state\_mat*, real(dp), dimension(1:num\_grids, 1:num\_size\_classes), intent(in) *weight\_grams*, type(grid\_data\_class), dimension(\*), intent(in) *grid*)

Initializes growth for startup.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 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:

#### 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.

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:

#### 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:

#### 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 |
| 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:

#### 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:

#### subroutine mortality\_mod::set\_fishing\_mort\_file\_name (character(\*), intent(in) *fname*)

Here is the caller graph for this function:

#### 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:

#### 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*)

##### 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:

#### subroutine mortality\_mod::set\_select\_data (type(dataforplots), intent(in) *value*)

Here is the caller graph for this function:

### Variable Documentation

#### real(dp), private mortality\_mod::alpha\_mort[private]

#### character(fname\_len), private mortality\_mod::config\_file\_name[private]

#### type(dataforplots), private mortality\_mod::data\_select[private]

#### real(dp), private mortality\_mod::delta\_time[private]

#### real(dp), private mortality\_mod::domain\_area\_sqm[private]

#### character(domain\_len), private mortality\_mod::domain\_name[private]

#### real(dp), private mortality\_mod::dredge\_width\_m[private]

#### real(dp), dimension(:), allocatable, private mortality\_mod::expl\_biomass\_gpsqm[private]

#### real(dp), dimension(:), allocatable, private mortality\_mod::expl\_scallops\_psqm[private]

#### real(dp), dimension(num\_size\_classes), private mortality\_mod::expl\_scallops\_psqm\_at\_size[private]

#### real(dp), dimension(:), allocatable, private mortality\_mod::f\_mort[private]

#### real(dp), dimension(:), allocatable, private mortality\_mod::fishing\_effort[private]

#### real(dp), private mortality\_mod::fishing\_mort[private]

#### character(fname\_len), private mortality\_mod::fishing\_mort\_fname[private]

#### type(fishingmortality), dimension(max\_num\_years), private mortality\_mod::fmort\_list[private]

#### real(dp), private mortality\_mod::gb\_cull\_size\_mm[private]

#### real(dp), private mortality\_mod::gb\_discard[private]

#### real(dp), private mortality\_mod::gb\_incidental[private]

#### real(dp), private mortality\_mod::gb\_length\_0[private]

#### real(dp), private mortality\_mod::gb\_mort\_adult[private]

#### real(dp), private mortality\_mod::gbc\_fselect\_a[private]

#### real(dp), private mortality\_mod::gbc\_fselect\_b[private]

#### real(dp), private mortality\_mod::gbo\_fselect\_a[private]

#### real(dp), private mortality\_mod::gbo\_fselect\_b[private]

#### real(dp), dimension(:), allocatable, private mortality\_mod::landings\_accum[private]

#### real(dp), dimension(num\_size\_classes), private mortality\_mod::landings\_at\_size[private]

#### real(dp), dimension(:), allocatable, private mortality\_mod::landings\_by\_num[private]

#### real(dp), dimension(:), allocatable, private mortality\_mod::landings\_wgt\_accum[private]

#### real(dp), dimension(:), allocatable, private mortality\_mod::landings\_wgt\_grams[private]

#### real(dp), dimension(:), allocatable, private mortality\_mod::lpue[private]

#### real(dp), dimension(:), allocatable, private mortality\_mod::lpue\_accum[private]

#### real(dp), private mortality\_mod::lpue\_intercept[private]

#### real(dp), private mortality\_mod::lpue\_slope[private]

#### real(dp), private mortality\_mod::lpue\_slope2[private]

#### real(dp), private mortality\_mod::ma\_cull\_size\_mm[private]

#### real(dp), private mortality\_mod::ma\_discard[private]

#### real(dp), private mortality\_mod::ma\_fselect\_a[private]

#### real(dp), private mortality\_mod::ma\_fselect\_b[private]

#### real(dp), private mortality\_mod::ma\_incidental[private]

#### real(dp), private mortality\_mod::ma\_length\_0[private]

#### real(dp), private mortality\_mod::ma\_mort\_adult[private]

#### integer, private mortality\_mod::max\_per\_day[private]

#### real(dp), private mortality\_mod::max\_time\_hpd[private]

#### integer, private mortality\_mod::num\_areas[private]

#### integer, private mortality\_mod::num\_grids[private]

#### integer, private mortality\_mod::num\_in\_list[private]

#### integer, private mortality\_mod::num\_time\_steps[private]

#### real(dp), private mortality\_mod::towing\_speed\_knots[private]

#### integer, private mortality\_mod::ts\_per\_year[private]

#### logical, private mortality\_mod::use\_spec\_access\_data[private]

#### 

## recruit\_mod Module Reference

### Data Types

### type recruitment\_classFunctions/Subroutines

subroutine **set\_recruitment** (recruit, n\_grids, dom\_name, dom\_area, recr\_yr\_strt, recr\_yr\_stop, recruit\_avg, l\_inf\_mu, k\_mu, shell\_length\_mm, yr\_start, yr\_stop)

*Set\_Recruitment.*

integer function **random\_index** ()

*Defines a weighted distribution as defined in weights.*

subroutine **set\_config\_file\_name** (fname)

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

subroutine **read\_configuration** ()

*Read\_Configuration.*

### Variables

integer, parameter **max\_n\_year** = 50

character(**fname\_len**), private **config\_file\_name**

integer, private **num\_grids**

character(**domain\_len**), private **domain\_name**

real(**dp**), private **domain\_area\_sqm**

integer, private **recruit\_yr\_strt**

integer, private **recruit\_yr\_stop**

integer, private **recruit\_avg\_num**

integer, private **n\_rand\_yrs**

integer, private **sim\_start\_year**

integer, private **sim\_stop\_year**

real(**dp**), private **recr\_period\_start**

real(**dp**), private **recr\_period\_stop**

real(**dp**), dimension(:), allocatable, private **weights**

real(**dp**), private **wsum**

### Function/Subroutine Documentation

#### integer function recruit\_mod::random\_index

Defines a weighted distribution as defined in weights.

##### Returns

a value 1 <= x <= n\_rand\_yrs

Here is the call graph for this function:

Here is the caller graph for this function:

#### 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:

#### 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'

#### subroutine recruit\_mod::set\_recruitment (type(recruitment\_class), dimension(\*), intent(inout) *recruit*, integer, intent(in) *n\_grids*, character(domain\_len), intent(in) *dom\_name*, real(dp), intent(in) *dom\_area*, integer, intent(out) *recr\_yr\_strt*, integer, intent(out) *recr\_yr\_stop*, integer, intent(out) *recruit\_avg*, real(dp), dimension(\*), intent(in) *l\_inf\_mu*, real(dp), dimension(\*), intent(in) *k\_mu*, real(dp), dimension(\*), intent(in) *shell\_length\_mm*, integer, intent(in) *yr\_start*, integer, intent(in) *yr\_stop*)

Set\_Recruitment.

Sets recruitment parameters

##### Parameters

|  |  |  |
| --- | --- | --- |
| in,out | *recruit* |  |
| in | *n\_grids,The* | number of grids under consideration, sets private value num\_grids |
| in | *dom\_name,The* | doomain being simulated, sets private value domain\_name. Should be MA MidAtlantic or GB GeorgesBank |
| in | *dom\_area* | the total area in square meters, sets domain\_area\_sqm |
| in | *L\_inf\_mu* | asymptotic size, average |
| in | *K\_mu* | Brody growth coefficient K, average |
| in | *shell\_length\_mm* | Shell height in millimeters |
| out | *recr\_yr\_strt* | year start of available data |
| out | *recr\_yr\_stop* | year stop of available data |
| in | *yr\_start* | simulation start year |
| in | *yr\_stop* | simulation end year |

Here is the call graph for this function:

Here is the caller graph for this function:

### Variable Documentation

#### character(fname\_len), private recruit\_mod::config\_file\_name[private]

#### real(dp), private recruit\_mod::domain\_area\_sqm[private]

#### character(domain\_len), private recruit\_mod::domain\_name[private]

#### integer, parameter recruit\_mod::max\_n\_year = 50

#### integer, private recruit\_mod::n\_rand\_yrs[private]

#### integer, private recruit\_mod::num\_grids[private]

#### real(dp), private recruit\_mod::recr\_period\_start[private]

#### real(dp), private recruit\_mod::recr\_period\_stop[private]

#### integer, private recruit\_mod::recruit\_avg\_num[private]

#### integer, private recruit\_mod::recruit\_yr\_stop[private]

#### integer, private recruit\_mod::recruit\_yr\_strt[private]

#### integer, private recruit\_mod::sim\_start\_year[private]

#### integer, private recruit\_mod::sim\_stop\_year[private]

#### real(dp), dimension(:), allocatable, private recruit\_mod::weights[private]

#### real(dp), private recruit\_mod::wsum[private]

# Data Type Documentation

## mortality\_mod::dataforplots Type Reference

Collaboration diagram for mortality\_mod::dataforplots:

### Public Attributes

logical **plot\_abun**

logical **plot\_biom**

logical **plot\_ebms**

logical **plot\_feff**

logical **plot\_fmor**

logical **plot\_land**

logical **plot\_lndw**

logical **plot\_lpue**

logical **plot\_recr**

### Member Data Documentation

#### logical mortality\_mod::dataforplots::plot\_abun

#### logical mortality\_mod::dataforplots::plot\_biom

#### logical mortality\_mod::dataforplots::plot\_ebms

#### logical mortality\_mod::dataforplots::plot\_feff

#### logical mortality\_mod::dataforplots::plot\_fmor

#### logical mortality\_mod::dataforplots::plot\_land

#### logical mortality\_mod::dataforplots::plot\_lndw

#### logical mortality\_mod::dataforplots::plot\_lpue

#### logical mortality\_mod::dataforplots::plot\_recr

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

SRC/**ScallopMortality.f90**

## mortality\_mod::fishingmortality Type Reference

Collaboration diagram for mortality\_mod::fishingmortality:

### Public Attributes

integer **year**

integer **n\_areas**

integer, dimension(**max\_num\_areas**) **area\_list**

real(**dp**), dimension(**max\_num\_areas**) **area\_fish\_mort**

### Member Data Documentation

#### real(dp), dimension(max\_num\_areas) mortality\_mod::fishingmortality::area\_fish\_mort

#### integer, dimension(max\_num\_areas) mortality\_mod::fishingmortality::area\_list

#### integer mortality\_mod::fishingmortality::n\_areas

#### integer mortality\_mod::fishingmortality::year

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

SRC/**ScallopMortality.f90**

## 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**

## 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**

character(15) **zone**

character(15) **region**

### Member Data Documentation

#### logical grid\_manager\_mod::grid\_data\_class::is\_closed

#### real(dp) grid\_manager\_mod::grid\_data\_class::lat

#### real(dp) grid\_manager\_mod::grid\_data\_class::lon

#### character(15) grid\_manager\_mod::grid\_data\_class::region

#### integer grid\_manager\_mod::grid\_data\_class::special\_access\_index

#### real(dp) grid\_manager\_mod::grid\_data\_class::stratum

#### real(dp) grid\_manager\_mod::grid\_data\_class::x

#### real(dp) grid\_manager\_mod::grid\_data\_class::y

#### real(dp) grid\_manager\_mod::grid\_data\_class::year

#### real(dp) grid\_manager\_mod::grid\_data\_class::z

#### character(15) grid\_manager\_mod::grid\_data\_class::zone

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

SRC/**GridManager.f90**

## Growth\_Class Module Reference

Subroutines that determine expected growth of scallops.

Collaboration diagram for Growth\_Class:

### Detailed Description

Subroutines that determine expected growth of scallops.

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

SRC/**ScallopGrowth.f90**

## 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.*

### Member Data Documentation

#### real(dp), dimension(num\_size\_classes, num\_size\_classes) growth\_mod::growth\_class::g

Growth matrix.

#### real(dp) growth\_mod::growth\_class::k\_mu

Growth coefficient mean.

#### real(dp) growth\_mod::growth\_class::k\_sd

Growth coefficient standard deviation.

#### real(dp) growth\_mod::growth\_class::l\_inf\_mu

Asymptotic size mean.

#### 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**

## grid\_manager\_mod::lonlatpoint Type Reference

Collaboration diagram for grid\_manager\_mod::lonlatpoint:

### Public Attributes

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

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

### Member Data Documentation

#### real(dp) grid\_manager\_mod::lonlatpoint::lat

#### real(dp) grid\_manager\_mod::lonlatpoint::lon

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

SRC/**GridManager.f90**

## grid\_manager\_mod::lonlatvector Type Reference

Collaboration diagram for grid\_manager\_mod::lonlatvector:

### Public Attributes

real(**dp**), dimension(**max\_sides**) **lon**

real(**dp**), dimension(**max\_sides**) **lat**

integer **n\_sides**

### Member Data Documentation

#### real(dp), dimension(max\_sides) grid\_manager\_mod::lonlatvector::lat

#### real(dp), dimension(max\_sides) grid\_manager\_mod::lonlatvector::lon

#### integer grid\_manager\_mod::lonlatvector::n\_sides

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

SRC/**GridManager.f90**

## Mortality\_Class Module Reference

Subroutines that determine expected mortality of scallops.

Collaboration diagram for Mortality\_Class:

### Detailed Description

Subroutines that determine expected mortality of scallops.

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

SRC/**ScallopMortality.f90**

## mortality\_mod::mortality\_class Type Reference

Collaboration diagram for mortality\_mod::mortality\_class:

### Public Attributes

real(**dp**), dimension(**num\_size\_classes**) **natural\_mortality**

real(**dp**) **incidental**

real(**dp**), dimension(**num\_size\_classes**) **discard**

real(**dp**), dimension(**num\_size\_classes**) **selectivity**

real(**dp**) **natural\_mort\_adult**

real(**dp**) **natural\_mort\_juv**

real(**dp**), dimension(1:**num\_size\_classes**) **alpha**

### Member Data Documentation

#### real(dp), dimension(1:num\_size\_classes) mortality\_mod::mortality\_class::alpha

#### real(dp), dimension(num\_size\_classes) mortality\_mod::mortality\_class::discard

#### real(dp) mortality\_mod::mortality\_class::incidental

#### real(dp) mortality\_mod::mortality\_class::natural\_mort\_adult

#### real(dp) mortality\_mod::mortality\_class::natural\_mort\_juv

#### real(dp), dimension(num\_size\_classes) mortality\_mod::mortality\_class::natural\_mortality

#### real(dp), dimension(num\_size\_classes) mortality\_mod::mortality\_class::selectivity

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

SRC/**ScallopMortality.f90**

## recruit\_mod::recruitment\_class Type Reference

Collaboration diagram for recruit\_mod::recruitment\_class:

### 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**

### Member Data Documentation

#### integer recruit\_mod::recruitment\_class::max\_rec\_ind

#### integer recruit\_mod::recruitment\_class::n\_year

#### real(dp) recruit\_mod::recruitment\_class::rec\_start

#### real(dp) recruit\_mod::recruitment\_class::rec\_stop

#### real(dp), dimension(max\_n\_year) recruit\_mod::recruitment\_class::recruitment

#### integer, dimension(max\_n\_year) recruit\_mod::recruitment\_class::year

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

SRC/**ScallopRecruit.f90**

## Recruitment\_Class Module Reference

Subroutines that determine expected growth of scallops.

Collaboration diagram for Recruitment\_Class:

### Detailed Description

Subroutines that determine expected growth of scallops.

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

SRC/**ScallopRecruit.f90**

# File Documentation

## SRC/aaaPageOrder.f90 File Reference

## 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)

logical function **globals::leap\_year** (year)

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

integer function **globals::dayofyear** (m, d)

logical function **globals::is\_nan** (x)

### Variables

integer, parameter **globals::sp** = selected\_real\_kind(6, 37)

integer, parameter **globals::dp** = selected\_real\_kind(15, 307)

integer, parameter **globals::qp** = selected\_real\_kind(33, 4931)

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

integer, parameter **globals::shell\_len\_max** = 150

integer, parameter **globals::shell\_len\_min** = 30

integer, parameter **globals::shell\_len\_delta** = 5

integer, parameter **globals::num\_size\_classes** = (**shell\_len\_max** - **shell\_len\_min**) / **shell\_len\_delta** + 1

integer, parameter **globals::max\_num\_years** = 50

integer, parameter **globals::max\_num\_areas** = 25

integer, parameter **globals::max\_sides** = 8

integer, parameter **globals::region\_none** =0

integer, parameter **globals::region\_n** =1

integer, parameter **globals::region\_s** =2

integer, parameter **globals::region\_sw** =3

integer, parameter **globals::region\_w** =4

integer, parameter **globals::region\_ma** =5

integer, parameter **globals::region\_gbk** = 1

integer, parameter **globals::region\_mab** = 5

integer, parameter **globals::tag\_len** = 40

integer, parameter **globals::value\_len** = 30

integer, parameter **globals::comment\_len** = 80

integer, parameter **globals::line\_len** = **tag\_len**+**value\_len**+**comment\_len**

integer, parameter **globals::fname\_len** = 100

integer, parameter **globals::form\_len** = 20

integer, parameter **globals::input\_str\_len** = 100

integer, parameter **globals::csv\_line\_len** = 2000

integer, parameter **globals::domain\_len** = 2

integer, parameter **globals::read\_dev** = 69

integer, parameter **globals::write\_dev** = 63

real(**dp**), parameter **globals::zero\_threshold** = 1.0D-99

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

real(**dp**), parameter **globals::grams\_per\_pound** = 453.592\_dp

real(**dp**), parameter **globals::meters\_per\_naut\_mile** = 1852.D0

real(**dp**), parameter **globals::grams\_per\_metric\_ton** = 1000000.\_dp

real(**dp**), parameter **globals::grid\_area\_sqm** = **meters\_per\_naut\_mile**\*\*2

real(**dp**), parameter **globals::tow\_area\_sqm** = 4516.\_dp

real(**dp**), parameter **globals::one\_scallop\_per\_tow** = 1.D0 / **tow\_area\_sqm**

real(**dp**), parameter **globals::ma\_gb\_border** = -70.5

real(**dp**), parameter **globals::days\_in\_year** =365+0.25-0.01+0.0025

character(\*), parameter **globals::term\_red** = ''//achar(27)//'[31m'

character(\*), parameter **globals::term\_yel** = ''//achar(27)//'[33m'

character(\*), parameter **globals::term\_grn** = ''//achar(27)//'[92m'

character(\*), parameter **globals::term\_blu** = ''//achar(27)//'[94m'

character(\*), parameter **globals::term\_blk** = ''//achar(27)//'[0m'

character(\*), parameter **globals::init\_cond\_dir** = 'InitialCondition/'

character(\*), parameter **globals::growth\_out\_dir** = 'GrowthOutput/'

character(\*), parameter **globals::rec\_input\_dir** = 'RecruitEstimates/'

character(\*), parameter **globals::rec\_output\_dir** = 'RecruitField/'

character(\*), parameter **globals::output\_dir** = 'Results/'

character(\*), parameter **globals::config\_dir\_sim** = 'Configuration/Simulation/'

character(\*), parameter **globals::config\_dir\_interp** = 'Configuration/Interpolation/'

character(\*), parameter **globals::config\_dir\_special** = 'Configuration/SpecialAccess/'

character(\*), parameter **globals::grid\_dir** = 'Grids/'

character(\*), parameter **globals::data\_dir** = 'Data/'

character(\*), parameter **globals::anal\_dir** = 'Analysis/'

integer, parameter **globals::num\_regions** = 2

character(3), dimension(**num\_regions**) **globals::rgn** = (/ '\_GB', '\_MA'/)

## 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:*

### 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**

## 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 **read\_csv** (num\_rows, num\_cols, file\_name, m, nndim)

### Function/Subroutine Documentation

#### 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*)

#### 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:

#### 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

#### 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:

#### 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:

#### subroutine write\_vector\_scalar\_field (integer, intent(in) *vector\_len*, real(dp), dimension(\*), intent(in) *f*, character (\*), intent(in) *file\_name*)

#### 

## SRC/ScallopGrowth.f90 File Reference

### Data Types

### type growth\_mod::growth\_classModules

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**

logical, private **growth\_mod::show\_recruits\_msg**

## 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)

subroutine **mortality\_mod::load\_fishing\_mortalities** ()

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

elemental real(**dp**) function **mortality\_mod::ring\_size\_selectivity** (shell\_length, is\_closed, longitude)

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

real(**dp**) function, dimension(**num\_grids**) **mortality\_mod::set\_fishing\_effort** (year, ts, state\_mat, weight\_grams, mortality, grid)

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

real(**dp**) function, dimension(1:**num\_size\_classes**) **mortality\_mod::compute\_natural\_mortality** (max\_rec\_ind, mortality, state\_vector, longitude)

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

elemental real(**dp**) function **mortality\_mod::set\_fishing\_mortality** (grid, year, use\_f\_loc, f\_loc)

*Computes Fishing Mortality.*

subroutine **mortality\_mod::set\_config\_file\_name** (fname)

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

subroutine **mortality\_mod::set\_fishing\_mort\_file\_name** (fname)

subroutine **mortality\_mod::read\_configuration** ()

*Read\_Configuration.*

subroutine **mortality\_mod::mortality\_write\_at\_timestep** (year, ts, state\_mat, weight\_grams, grid)

*Initializes growth for startup.*

elemental real(**dp**) function **mortality\_mod::set\_discard** (length, selectivity, cull\_size, discard, is\_closed)

*Computes element of discard vector.*

elemental real(**dp**) function **mortality\_mod::calc\_lpue** (expl\_biomass, expl\_scallops)

*Computes catch as pounds per day.*

### Variables

character(**fname\_len**), private **mortality\_mod::config\_file\_name**

character(**fname\_len**), private **mortality\_mod::fishing\_mort\_fname**

type(**fishingmortality**), dimension(**max\_num\_years**), private **mortality\_mod::fmort\_list**

logical, private **mortality\_mod::use\_spec\_access\_data**

integer, private **mortality\_mod::num\_in\_list**

integer, private **mortality\_mod::num\_grids**

integer, private **mortality\_mod::num\_areas**

character(**domain\_len**), private **mortality\_mod::domain\_name**

real(**dp**), private **mortality\_mod::domain\_area\_sqm**

integer, private **mortality\_mod::num\_time\_steps**

integer, private **mortality\_mod::ts\_per\_year**

real(**dp**), private **mortality\_mod::delta\_time**

real(**dp**), private **mortality\_mod::fishing\_mort**

real(**dp**), private **mortality\_mod::alpha\_mort**

real(**dp**), private **mortality\_mod::ma\_cull\_size\_mm**

real(**dp**), private **mortality\_mod::ma\_discard**

real(**dp**), private **mortality\_mod::gb\_cull\_size\_mm**

real(**dp**), private **mortality\_mod::gb\_discard**

real(**dp**), private **mortality\_mod::ma\_fselect\_a**

real(**dp**), private **mortality\_mod::ma\_fselect\_b**

real(**dp**), private **mortality\_mod::gbc\_fselect\_a**

real(**dp**), private **mortality\_mod::gbc\_fselect\_b**

real(**dp**), private **mortality\_mod::gbo\_fselect\_a**

real(**dp**), private **mortality\_mod::gbo\_fselect\_b**

real(**dp**), private **mortality\_mod::ma\_mort\_adult**

real(**dp**), private **mortality\_mod::ma\_incidental**

real(**dp**), private **mortality\_mod::ma\_length\_0**

real(**dp**), private **mortality\_mod::gb\_mort\_adult**

real(**dp**), private **mortality\_mod::gb\_incidental**

real(**dp**), private **mortality\_mod::gb\_length\_0**

real(**dp**), private **mortality\_mod::lpue\_slope**

real(**dp**), private **mortality\_mod::lpue\_slope2**

real(**dp**), private **mortality\_mod::lpue\_intercept**

integer, private **mortality\_mod::max\_per\_day**

real(**dp**), private **mortality\_mod::max\_time\_hpd**

real(**dp**), private **mortality\_mod::dredge\_width\_m**

real(**dp**), private **mortality\_mod::towing\_speed\_knots**

real(**dp**), dimension(:), allocatable, private **mortality\_mod::expl\_biomass\_gpsqm**

real(**dp**), dimension(:), allocatable, private **mortality\_mod::expl\_scallops\_psqm**

real(**dp**), dimension(:), allocatable, private **mortality\_mod::f\_mort**

real(**dp**), dimension(:), allocatable, private **mortality\_mod::landings\_by\_num**

real(**dp**), dimension(:), allocatable, private **mortality\_mod::landings\_wgt\_grams**

real(**dp**), dimension(:), allocatable, private **mortality\_mod::lpue**

real(**dp**), dimension(:), allocatable, private **mortality\_mod::fishing\_effort**

real(**dp**), dimension(:), allocatable, private **mortality\_mod::landings\_accum**

real(**dp**), dimension(:), allocatable, private **mortality\_mod::landings\_wgt\_accum**

real(**dp**), dimension(:), allocatable, private **mortality\_mod::lpue\_accum**

real(**dp**), dimension(**num\_size\_classes**), private **mortality\_mod::expl\_scallops\_psqm\_at\_size**

real(**dp**), dimension(**num\_size\_classes**), private **mortality\_mod::landings\_at\_size**

type(**dataforplots**), private **mortality\_mod::data\_select**

## SRC/ScallopPopDensity.f90 File Reference

### Functions/Subroutines

program **scalloppopdensity**

subroutine **read\_startup\_config** (time\_steps\_per\_year, start\_year, stop\_year, domain\_name, plot\_data\_sel)

*Read Input File.*

subroutine **write\_lat\_lon\_preamble** (num\_grids, grid, fname)

*Writes lat and lon columns with headers to named file.*

subroutine **write\_x\_y\_preamble** (num\_grids, grid, yr\_offset, fname)

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

subroutine **write\_column\_csv\_by\_region** (n, f, c, lon, header, file\_name, append, use\_c)

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

subroutine **setup\_data\_files** (plot\_data\_sel, num\_grids, grid, domain\_name, start\_year, stop\_year)

subroutine **write\_recruit\_estimates** (ts, ts\_per\_year, num\_grids, grid, domain\_name, year, start\_year, recruit)

### Function/Subroutine Documentation

#### subroutine read\_startup\_config (integer, intent(out) *time\_steps\_per\_year*, integer, intent(out) *start\_year*, integer, intent(out) *stop\_year*, character(domain\_len), intent(out) *domain\_name*, type(dataforplots), intent(out) *plot\_data\_sel*)

Read Input File.

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

##### Parameters

|  |  |  |
| --- | --- | --- |
| out | *domain\_name* | can be either MA MidAtlantic or GB GeorgesBank |
| out | *init\_cond\_file\_name* | File name that contains intial simulation conditions |
| out | *start\_year* | Starting year for simulation read from config file |
| out | *stop\_year* | End year for simulation read from config file |
| out | *time\_steps\_per\_year* | Number of times steps to evaluate growth |
| out | *num\_monte\_carlo\_iter* | Number of iterations for Monte Carlo simulation |

Here is the call graph for this function:

Here is the caller graph for this function:

#### program scalloppopdensity

Here is the call graph for this function:

#### subroutine setup\_data\_files (type(dataforplots), intent(in) *plot\_data\_sel*, integer, intent(in) *num\_grids*, type(grid\_data\_class), dimension(\*), intent(in) *grid*, character(domain\_len), intent(out) *domain\_name*, integer, intent(in) *start\_year*, integer, intent(in) *stop\_year*)

Here is the call graph for this function:

Here is the caller graph for this function:

#### subroutine write\_column\_csv\_by\_region (integer, intent(in) *n*, real(dp), dimension(\*), intent(in) *f*, character(15), dimension(\*), intent(in) *c*, real(dp), dimension(\*), intent(in) *lon*, character(\*), intent(in) *header*, character(\*), intent(in) *file\_name*, logical, intent(in) *append*, logical, intent(in) *use\_c*)

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

Here is the caller graph for this function:

#### 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:

#### subroutine write\_recruit\_estimates (integer, intent(in) *ts*, integer, intent(in) *ts\_per\_year*, integer, intent(in) *num\_grids*, type(grid\_data\_class), dimension(\*), intent(in) *grid*, character(domain\_len), intent(out) *domain\_name*, integer, intent(in) *year*, integer, intent(in) *start\_year*, type(recruitment\_class), dimension(\*), intent(in) *recruit*)

Here is the call graph for this function:

Here is the caller graph for this function:

#### subroutine write\_x\_y\_preamble (integer, intent(in) *num\_grids*, type(grid\_data\_class), dimension(\*), intent(in) *grid*, real(dp), intent(in) *yr\_offset*, character(\*), intent(in) *fname*)

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

Here is the call graph for this function:

Here is the caller graph for this function:

## SRC/ScallopRecruit.f90 File Reference

### Data Types

### type recruit\_mod::recruitment\_classModules

module **recruit\_mod**

### Functions/Subroutines

subroutine **recruit\_mod::set\_recruitment** (recruit, n\_grids, dom\_name, dom\_area, recr\_yr\_strt, recr\_yr\_stop, recruit\_avg, l\_inf\_mu, k\_mu, shell\_length\_mm, yr\_start, yr\_stop)

*Set\_Recruitment.*

integer function **recruit\_mod::random\_index** ()

*Defines a weighted distribution as defined in weights.*

subroutine **recruit\_mod::set\_config\_file\_name** (fname)

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

subroutine **recruit\_mod::read\_configuration** ()

*Read\_Configuration.*

### Variables

integer, parameter **recruit\_mod::max\_n\_year** = 50

character(**fname\_len**), private **recruit\_mod::config\_file\_name**

integer, private **recruit\_mod::num\_grids**

character(**domain\_len**), private **recruit\_mod::domain\_name**

real(**dp**), private **recruit\_mod::domain\_area\_sqm**

integer, private **recruit\_mod::recruit\_yr\_strt**

integer, private **recruit\_mod::recruit\_yr\_stop**

integer, private **recruit\_mod::recruit\_avg\_num**

integer, private **recruit\_mod::n\_rand\_yrs**

integer, private **recruit\_mod::sim\_start\_year**

integer, private **recruit\_mod::sim\_stop\_year**

real(**dp**), private **recruit\_mod::recr\_period\_start**

real(**dp**), private **recruit\_mod::recr\_period\_stop**

real(**dp**), dimension(:), allocatable, private **recruit\_mod::weights**

real(**dp**), private **recruit\_mod::wsum**

# Index

alpha

mortality\_mod::mortality\_class, 73

alpha\_mort

mortality\_mod, 53

anal\_dir

globals, 21

area

grid\_manager\_mod, 31

area\_fish\_mort

mortality\_mod::fishingmortality, 62

area\_list

mortality\_mod::fishingmortality, 62

calc\_lpue

mortality\_mod, 46

comment\_len

globals, 21

Common Parameters, 14

compute\_natural\_mortality

mortality\_mod, 46

config\_dir\_interp

globals, 21

config\_dir\_sim

globals, 21

config\_dir\_special

globals, 21

config\_file\_name

grid\_manager\_mod, 31

mortality\_mod, 53

recruit\_mod, 59

csv\_line\_len

globals, 21

data\_dir

globals, 21

data\_select

mortality\_mod, 53

dayofyear

globals, 19

days\_in\_year

globals, 21

delta\_time

growth\_mod, 43

mortality\_mod, 53

destructor

mortality\_mod, 47

discard

mortality\_mod::mortality\_class, 73

divby

globals, 19

domain\_area\_sqm

growth\_mod, 43

mortality\_mod, 53

recruit\_mod, 59

domain\_len

globals, 21

domain\_name

grid\_manager\_mod, 31

growth\_mod, 43

mortality\_mod, 53

recruit\_mod, 59

dp

globals, 21

dredge\_width\_m

mortality\_mod, 53

enforce\_non\_negative\_growth

growth\_mod, 33

expl\_biomass\_gpsqm

mortality\_mod, 53

expl\_scallops\_psqm

mortality\_mod, 53

expl\_scallops\_psqm\_at\_size

mortality\_mod, 53

f\_mort

mortality\_mod, 53

fishing\_effort

mortality\_mod, 53

fishing\_mort

mortality\_mod, 53

fishing\_mort\_fname

mortality\_mod, 53

fmort\_list

mortality\_mod, 53

fname\_len

globals, 21

form\_len

globals, 21

g

growth\_mod::growth\_class, 67

gamma\_inc\_values

growth\_mod, 33

gb\_cull\_size\_mm

mortality\_mod, 53

gb\_discard

mortality\_mod, 53

gb\_incidental

mortality\_mod, 53

gb\_length\_0

mortality\_mod, 53

gb\_mort\_adult

mortality\_mod, 53

gbc\_fselect\_a

mortality\_mod, 53

gbc\_fselect\_b

mortality\_mod, 54

gbo\_fselect\_a

mortality\_mod, 54

gbo\_fselect\_b

mortality\_mod, 54

gen\_size\_trans\_matrix

growth\_mod, 33

get\_growth\_gb

growth\_mod, 34

get\_growth\_ma

growth\_mod, 35

get\_num\_of\_areas

grid\_manager\_mod, 25

globals, 18

anal\_dir, 21

comment\_len, 21

config\_dir\_interp, 21

config\_dir\_sim, 21

config\_dir\_special, 21

csv\_line\_len, 21

data\_dir, 21

dayofyear, 19

days\_in\_year, 21

divby, 19

domain\_len, 21

dp, 21

fname\_len, 21

form\_len, 21

grams\_per\_metric\_ton, 21

grams\_per\_pound, 21

grid\_area\_sqm, 21

grid\_dir, 21

growth\_out\_dir, 21

init\_cond\_dir, 21

input\_str\_len, 21

is\_nan, 19

leap\_year, 20

line\_len, 21

logic\_to\_double, 20

ma\_gb\_border, 21

matrixinv, 20

max\_num\_areas, 21

max\_num\_years, 21

max\_sides, 22

meters\_per\_naut\_mile, 22

ndim, 22

num\_regions, 22

num\_size\_classes, 22

one\_scallop\_per\_tow, 22

output\_dir, 22

pi, 22

qp, 22

read\_dev, 22

rec\_input\_dir, 22

rec\_output\_dir, 22

region\_gbk, 22

region\_ma, 22

region\_mab, 22

region\_n, 22

region\_none, 22

region\_s, 22

region\_sw, 22

region\_w, 22

rgn, 22

shell\_len\_delta, 22

shell\_len\_max, 22

shell\_len\_min, 22

sp, 23

tag\_len, 23

term\_blk, 23

term\_blu, 23

term\_grn, 23

term\_red, 23

term\_yel, 23

tow\_area\_sqm, 23

value\_len, 23

write\_dev, 23

zero\_threshold, 23

grams\_per\_metric\_ton

globals, 21

grams\_per\_pound

globals, 21

Grid Manager Mod, 12

grid\_area\_sqm

globals, 21

Grid\_Data\_Class, 63

grid\_dir

globals, 21

grid\_manager\_mod, 24

area, 31

config\_file\_name, 31

domain\_name, 31

get\_num\_of\_areas, 25

init\_cond\_fname, 31

is\_grid\_in\_special\_access, 25

load\_area\_coordinates, 25

load\_grid\_state, 26

num\_areas, 31

num\_grids, 31

point\_in\_polygon\_array, 26

point\_in\_polygon\_points, 27

point\_in\_polygon\_vector, 27

read\_configuration, 28

set\_config\_file\_name, 28

set\_grid\_manager, 29

set\_init\_cond\_file\_name, 29

set\_num\_grids, 29

set\_special\_access\_file\_name, 30

special\_accesss\_fname, 31

use\_spec\_access\_data, 31

grid\_manager\_mod::grid\_data\_class, 64

is\_closed, 65

lat, 65

lon, 65

region, 65

special\_access\_index, 65

stratum, 65

x, 65

y, 65

year, 65

z, 65

zone, 65

grid\_manager\_mod::lonlatpoint, 69

lat, 69

lon, 69

grid\_manager\_mod::lonlatvector, 70

lat, 70

lon, 70

n\_sides, 70

Growth\_Class, 66

growth\_mod, 32

delta\_time, 43

domain\_area\_sqm, 43

domain\_name, 43

enforce\_non\_negative\_growth, 33

gamma\_inc\_values, 33

gen\_size\_trans\_matrix, 33

get\_growth\_gb, 34

get\_growth\_ma, 35

growth\_param\_size, 43

h\_mn18, 35

increment\_mean\_std, 36

mn18\_appxc\_trans\_matrix, 37

norm\_cumul\_dist\_fcn, 38

norm\_density\_fcn, 39

num\_grids, 43

num\_time\_steps, 43

set\_growth, 39

set\_shell\_lengths, 40

shell\_to\_weight, 41

show\_recruits\_msg, 43

time\_steps\_year, 43

time\_to\_grow, 42

Growth\_Mod, 7

growth\_mod::growth\_class, 67

g, 67

k\_mu, 67

k\_sd, 68

l\_inf\_mu, 68

l\_inf\_sd, 68

growth\_out\_dir

globals, 21

growth\_param\_size

growth\_mod, 43

h\_mn18

growth\_mod, 35

incidental

mortality\_mod::mortality\_class, 73

increment\_mean\_std

growth\_mod, 36

init\_cond\_dir

globals, 21

init\_cond\_fname

grid\_manager\_mod, 31

input\_str\_len

globals, 21

IORoutines.f90

read\_csv, 81

read\_scalar\_field, 81

write\_2d\_scalar\_field, 81

write\_column\_csv, 81

write\_csv, 82

write\_vector\_scalar\_field, 82

is\_closed

grid\_manager\_mod::grid\_data\_class, 65

is\_grid\_in\_special\_access

grid\_manager\_mod, 25

is\_nan

globals, 19

k\_mu

growth\_mod::growth\_class, 67

k\_sd

growth\_mod::growth\_class, 68

l\_inf\_mu

growth\_mod::growth\_class, 68

l\_inf\_sd

growth\_mod::growth\_class, 68

landings\_accum

mortality\_mod, 54

landings\_at\_size

mortality\_mod, 54

landings\_by\_num

mortality\_mod, 54

landings\_wgt\_accum

mortality\_mod, 54

landings\_wgt\_grams

mortality\_mod, 54

lat

grid\_manager\_mod::grid\_data\_class, 65

grid\_manager\_mod::lonlatpoint, 69

grid\_manager\_mod::lonlatvector, 70

leap\_year

globals, 20

line\_len

globals, 21

load\_area\_coordinates

grid\_manager\_mod, 25

load\_fishing\_mortalities

mortality\_mod, 47

load\_grid\_state

grid\_manager\_mod, 26

logic\_to\_double

globals, 20

lon

grid\_manager\_mod::grid\_data\_class, 65

grid\_manager\_mod::lonlatpoint, 69

grid\_manager\_mod::lonlatvector, 70

lpue

mortality\_mod, 54

lpue\_accum

mortality\_mod, 54

lpue\_intercept

mortality\_mod, 54

lpue\_slope

mortality\_mod, 54

lpue\_slope2

mortality\_mod, 54

ma\_cull\_size\_mm

mortality\_mod, 54

ma\_discard

mortality\_mod, 54

ma\_fselect\_a

mortality\_mod, 54

ma\_fselect\_b

mortality\_mod, 54

ma\_gb\_border

globals, 21

ma\_incidental

mortality\_mod, 54

ma\_length\_0

mortality\_mod, 54

ma\_mort\_adult

mortality\_mod, 54

matrixinv

globals, 20

max\_n\_year

recruit\_mod, 59

max\_num\_areas

globals, 21

max\_num\_years

globals, 21

max\_per\_day

mortality\_mod, 54

max\_rec\_ind

recruit\_mod::recruitment\_class, 74

max\_sides

globals, 22

max\_time\_hpd

mortality\_mod, 54

meters\_per\_naut\_mile

globals, 22

mn18\_appxc\_trans\_matrix

growth\_mod, 37

Mortality\_Class, 71

mortality\_mod, 44

alpha\_mort, 53

calc\_lpue, 46

compute\_natural\_mortality, 46

config\_file\_name, 53

data\_select, 53

delta\_time, 53

destructor, 47

domain\_area\_sqm, 53

domain\_name, 53

dredge\_width\_m, 53

expl\_biomass\_gpsqm, 53

expl\_scallops\_psqm, 53

expl\_scallops\_psqm\_at\_size, 53

f\_mort, 53

fishing\_effort, 53

fishing\_mort, 53

fishing\_mort\_fname, 53

fmort\_list, 53

gb\_cull\_size\_mm, 53

gb\_discard, 53

gb\_incidental, 53

gb\_length\_0, 53

gb\_mort\_adult, 53

gbc\_fselect\_a, 53

gbc\_fselect\_b, 54

gbo\_fselect\_a, 54

gbo\_fselect\_b, 54

landings\_accum, 54

landings\_at\_size, 54

landings\_by\_num, 54

landings\_wgt\_accum, 54

landings\_wgt\_grams, 54

load\_fishing\_mortalities, 47

lpue, 54

lpue\_accum, 54

lpue\_intercept, 54

lpue\_slope, 54

lpue\_slope2, 54

ma\_cull\_size\_mm, 54

ma\_discard, 54

ma\_fselect\_a, 54

ma\_fselect\_b, 54

ma\_incidental, 54

ma\_length\_0, 54

ma\_mort\_adult, 54

max\_per\_day, 54

max\_time\_hpd, 54

mortality\_write\_at\_timestep, 47

num\_areas, 55

num\_grids, 55

num\_in\_list, 55

num\_time\_steps, 55

read\_configuration, 48

ring\_size\_selectivity, 48

set\_config\_file\_name, 49

set\_discard, 49

set\_fishing\_effort, 49

set\_fishing\_mort\_file\_name, 50

set\_fishing\_mortality, 50

set\_mortality, 51

set\_select\_data, 52

towing\_speed\_knots, 55

ts\_per\_year, 55

use\_spec\_access\_data, 55

Mortality\_Mod, 10

mortality\_mod::dataforplots, 60

plot\_abun, 61

plot\_biom, 61

plot\_ebms, 61

plot\_feff, 61

plot\_fmor, 61

plot\_land, 61

plot\_lndw, 61

plot\_lpue, 61

plot\_recr, 61

mortality\_mod::fishingmortality, 62

area\_fish\_mort, 62

area\_list, 62

n\_areas, 62

year, 62

mortality\_mod::mortality\_class, 72

alpha, 73

discard, 73

incidental, 73

natural\_mort\_adult, 73

natural\_mort\_juv, 73

natural\_mortality, 73

selectivity, 73

mortality\_write\_at\_timestep

mortality\_mod, 47

n\_areas

mortality\_mod::fishingmortality, 62

n\_rand\_yrs

recruit\_mod, 59

n\_sides

grid\_manager\_mod::lonlatvector, 70

n\_year

recruit\_mod::recruitment\_class, 74

natural\_mort\_adult

mortality\_mod::mortality\_class, 73

natural\_mort\_juv

mortality\_mod::mortality\_class, 73

natural\_mortality

mortality\_mod::mortality\_class, 73

ndim

globals, 22

norm\_cumul\_dist\_fcn

growth\_mod, 38

norm\_density\_fcn

growth\_mod, 39

num\_areas

grid\_manager\_mod, 31

mortality\_mod, 55

num\_grids

grid\_manager\_mod, 31

growth\_mod, 43

mortality\_mod, 55

recruit\_mod, 59

num\_in\_list

mortality\_mod, 55

num\_regions

globals, 22

num\_size\_classes

globals, 22

num\_time\_steps

growth\_mod, 43

mortality\_mod, 55

one\_scallop\_per\_tow

globals, 22

output\_dir

globals, 22

pi

globals, 22

plot\_abun

mortality\_mod::dataforplots, 61

plot\_biom

mortality\_mod::dataforplots, 61

plot\_ebms

mortality\_mod::dataforplots, 61

plot\_feff

mortality\_mod::dataforplots, 61

plot\_fmor

mortality\_mod::dataforplots, 61

plot\_land

mortality\_mod::dataforplots, 61

plot\_lndw

mortality\_mod::dataforplots, 61

plot\_lpue

mortality\_mod::dataforplots, 61

plot\_recr

mortality\_mod::dataforplots, 61

point\_in\_polygon\_array

grid\_manager\_mod, 26

point\_in\_polygon\_points

grid\_manager\_mod, 27

point\_in\_polygon\_vector

grid\_manager\_mod, 27

qp

globals, 22

random\_index

recruit\_mod, 56

read\_configuration

grid\_manager\_mod, 28

mortality\_mod, 48

recruit\_mod, 57

read\_csv

IORoutines.f90, 81

read\_dev

globals, 22

read\_scalar\_field

IORoutines.f90, 81

read\_startup\_config

ScallopPopDensity.f90, 88

rec\_input\_dir

globals, 22

rec\_output\_dir

globals, 22

rec\_start

recruit\_mod::recruitment\_class, 74

rec\_stop

recruit\_mod::recruitment\_class, 74

recr\_period\_start

recruit\_mod, 59

recr\_period\_stop

recruit\_mod, 59

recruit\_avg\_num

recruit\_mod, 59

recruit\_mod, 56

config\_file\_name, 59

domain\_area\_sqm, 59

domain\_name, 59

max\_n\_year, 59

n\_rand\_yrs, 59

num\_grids, 59

random\_index, 56

read\_configuration, 57

recr\_period\_start, 59

recr\_period\_stop, 59

recruit\_avg\_num, 59

recruit\_yr\_stop, 59

recruit\_yr\_strt, 59

set\_config\_file\_name, 57

set\_recruitment, 57

sim\_start\_year, 59

sim\_stop\_year, 59

weights, 59

wsum, 59

Recruit\_Mod, 9

recruit\_mod::recruitment\_class, 74

max\_rec\_ind, 74

n\_year, 74

rec\_start, 74

rec\_stop, 74

recruitment, 74

year, 74

recruit\_yr\_stop

recruit\_mod, 59

recruit\_yr\_strt

recruit\_mod, 59

recruitment

recruit\_mod::recruitment\_class, 74

Recruitment\_Class, 75

region

grid\_manager\_mod::grid\_data\_class, 65

region\_gbk

globals, 22

region\_ma

globals, 22

region\_mab

globals, 22

region\_n

globals, 22

region\_none

globals, 22

region\_s

globals, 22

region\_sw

globals, 22

region\_w

globals, 22

rgn

globals, 22

ring\_size\_selectivity

mortality\_mod, 48

Scallop Population Density, 1

scalloppopdensity

ScallopPopDensity.f90, 89

ScallopPopDensity.f90

read\_startup\_config, 88

scalloppopdensity, 89

setup\_data\_files, 91

write\_column\_csv\_by\_region, 91

write\_lat\_lon\_preamble, 91

write\_recruit\_estimates, 91

write\_x\_y\_preamble, 92

selectivity

mortality\_mod::mortality\_class, 73

set\_config\_file\_name

grid\_manager\_mod, 28

mortality\_mod, 49

recruit\_mod, 57

set\_discard

mortality\_mod, 49

set\_fishing\_effort

mortality\_mod, 49

set\_fishing\_mort\_file\_name

mortality\_mod, 50

set\_fishing\_mortality

mortality\_mod, 50

set\_grid\_manager

grid\_manager\_mod, 29

set\_growth

growth\_mod, 39

set\_init\_cond\_file\_name

grid\_manager\_mod, 29

set\_mortality

mortality\_mod, 51

set\_num\_grids

grid\_manager\_mod, 29

set\_recruitment

recruit\_mod, 57

set\_select\_data

mortality\_mod, 52

set\_shell\_lengths

growth\_mod, 40

set\_special\_access\_file\_name

grid\_manager\_mod, 30

setup\_data\_files

ScallopPopDensity.f90, 91

shell\_len\_delta

globals, 22

shell\_len\_max

globals, 22

shell\_len\_min

globals, 22

shell\_to\_weight

growth\_mod, 41

show\_recruits\_msg

growth\_mod, 43

sim\_start\_year

recruit\_mod, 59

sim\_stop\_year

recruit\_mod, 59

sp

globals, 23

special\_access\_index

grid\_manager\_mod::grid\_data\_class, 65

special\_accesss\_fname

grid\_manager\_mod, 31

SRC/aaaPageOrder.f90, 76

SRC/Globals.f90, 77

SRC/GridManager.f90, 79

SRC/IORoutines.f90, 81

SRC/ScallopGrowth.f90, 83

SRC/ScallopMortality.f90, 85

SRC/ScallopPopDensity.f90, 88

SRC/ScallopRecruit.f90, 93

stratum

grid\_manager\_mod::grid\_data\_class, 65

tag\_len

globals, 23

term\_blk

globals, 23

term\_blu

globals, 23

term\_grn

globals, 23

term\_red

globals, 23

term\_yel

globals, 23

time\_steps\_year

growth\_mod, 43

time\_to\_grow

growth\_mod, 42

tow\_area\_sqm

globals, 23

towing\_speed\_knots

mortality\_mod, 55

ts\_per\_year

mortality\_mod, 55

use\_spec\_access\_data

grid\_manager\_mod, 31

mortality\_mod, 55

value\_len

globals, 23

weights

recruit\_mod, 59

write\_2d\_scalar\_field

IORoutines.f90, 81

write\_column\_csv

IORoutines.f90, 81

write\_column\_csv\_by\_region

ScallopPopDensity.f90, 91

write\_csv

IORoutines.f90, 82

write\_dev

globals, 23

write\_lat\_lon\_preamble

ScallopPopDensity.f90, 91

write\_recruit\_estimates

ScallopPopDensity.f90, 91

write\_vector\_scalar\_field

IORoutines.f90, 82

write\_x\_y\_preamble

ScallopPopDensity.f90, 92

wsum

recruit\_mod, 59

x

grid\_manager\_mod::grid\_data\_class, 65

y

grid\_manager\_mod::grid\_data\_class, 65

year

grid\_manager\_mod::grid\_data\_class, 65

mortality\_mod::fishingmortality, 62

recruit\_mod::recruitment\_class, 74

z

grid\_manager\_mod::grid\_data\_class, 65

zero\_threshold

globals, 23

zone

grid\_manager\_mod::grid\_data\_class, 65