Universal Kriging

0.1

Generated by Doxygen 1.9.8

Chapter 1

Universal Kriging

This program is designed to interpolate geospatial data from a smaller, arbritrary grid to a larger, evenly spaced grid, albeit evenly is not required.

1.1 Universal Kriging, UK, Startup

UK begins by gathering parameters used to define how the program executes

1.1.1 UK Configuration File

This file is specified on the command line. It has the following parameters

- Kriging variogram form
- Log Transform
- Power Transform Parameter
- NLS Spatial Fcn File Name
- Save Data

1.1.1.1 High Limit Factor

This configuration item is used to set an upper limit for the observed data. That is,

- fmax = fmax_multiplier $*$ maxval(observed data)
  Used with logical IsHighLimit. IsHighLimit is the inverse of Log Transform. Typically runs as false. Not fully tested.

1.1.1.2 Kriging variogram form

- 'spherical'
- 'exponential'
- 'gaussian'
- 'matern'

Default value is 'spherical' with no entry in configuration file

1.1.1.3 Log Transform

If set to true, interpolation uses the log of the input data instead of linear transform.

1.1.1.4 NLS Spatial Fcn File Name

The name of the file that defines the spatial functions. Configuration files are expected to be in the ***./Configuration*** subdirectory

1.1.1.5 Save Data

This configuration item determines where the output is saved.

F: Written to the root directory
T: Written to the ***./Results*** with files named using the template Lat_Lon_Grid_PARM_DN_YYYY.csv

- PARM, includes but not limited to
  - EBMS: Exploitable BioMasS
  - LPUE: Landings Per Unit Effort
  - RECR: Recruitment
- DN: Domain Name
  - MA: Mid Atlantic
  - GB: Georges Bank

Chapter 2

Kriging Routines

2.1 Implementation of Interpolation

2.1.1 Interpolation Algorithm

The interpolation of geospatial data is carried out via an Universal Kriging (UK) algorithm.

2.1.1.1 Universal Kriging

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

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

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

2.1.1.2 Spatial functions

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

Gaussian Bump:

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

Logistic curve:

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

Sin Exp curve:

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

Cos Exp curve:

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

In all of the function form λ controls the width of the transition and x_0 the transition point.

After fitting these to the bathymetric variable (H) we can introduce interaction. Allowing interaction terms for the spatial functions depending on bathymetry only we can define, $g_j(x,H,\lambda^j,{x_0}^j,\lambda_k,{x_0}^k) = f_j(x)f_k(H)$

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

Here z is bathymetric depth. We start by fitting nonlinear parameters $\lambda^{c,s}$ and $x_0^{c,s}$ to log recruitment for "cross shelf" structure.

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

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

2.1.1.3 Fitting non-linear parameters

A brute force approach is taken to fitting the nonlinear parameters x_0 and λ. A search range is determined based on the geographic range of the observations. The parameters are then fit to minimize the misfit to observations.

Subroutine *NLSF_Fit_Function* parameter np). The nonlinear parameters are fit by minimizing RMS misfit to the simple least squares fit with a smoothness penalty,

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

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

A smoothness penalty is imposed for each function based on the analytic

2.1.2 Residual process

After performing an ordinary least squares fit for the SF coeficients, β, we have an estimate of ϵ. An empirical variogram is computed subroutine *Krig_Comp_Emp_Variogram*, and variogram parameters are fit (again by brute force).
The variogram forms allowed are "spherical", "exponential", and "gaussian". The form is defaults to 'spherical' if not specified by the UK configuration file.

2.1.2.1 Posterior sampling

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

2.2 Non Linear Spatial function fitting for UK

The universal kriging algorithm described above is used to build a distribution based on the historical recruitment data (1979-present). Spatial function forms of one variable were selected for smoothness and boundedness. We have:
Gaussian bump

The nonlinear parameters are fit by minimizing RMS misfit to the simple least squares fit.

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

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

Chapter 3

Non Linear Spatial Functions, NLSF

TBD - More information on what these methods do.

To easily change how the NLSF functions perform a configuration file is used. The NLSF configuration file is named by the UK configuration file, configuration item **NLS Spatial Fcn File Name**. The subroutine ***NLSF::Set_Config_File_Name*** is called to set this value. When UK first starts is must call ***NLSF_Count_Defined_Functions*** to determine how many spatial functions are defined.

The following configuration items are defined.

- Function String

3.1 Function String

3.1.1 Function

Define non linear spatial functions and paramater search range.

- "Function 1, dim=z, shape=Logistic, precon=0 "
- "Function 2, dim=z, shape=Gaussian, precon=0 "
- "Function 3, dim=x, shape=Logistic, precon=1 "

These define spatial functions for setting the spatial trend in the universal kriging algorithm.

The precon=0 term means that the function is not multiplied by another function. For example,

```
"Function 3, dim=x, shape=Logistic, precon=1 "\n
```

indicates that the third function is multiplied by the first function.
This is true for fitting the nonlinear parameters of function 3 hence the parameters of function 1 must be fit before the parameters of function 3.

3.1.2 dim

- 'x'
- 'y'
- 'z'
- 'x+y'

3.1.3 shape

Let $\vec{A} = \vec{dim} - f0$

- 'Gaussian': $exp(-(\vec{A}/\lambda)^2)$
- 'Logistic': $1/\left(1 + exp(-\vec{A}/\lambda)\right)$
- 'SinExp': $sin(\vec{A}/\lambda) * exp(-(\vec{A}/\lambda)^2)$
- 'CosExp': $cos(\vec{A}/\lambda) * exp(-(\vec{A}/(2\lambda))^2)$

3.2 IsTruncateRange

- Set to F to extrapolate beyond observation range
- Set to T to restrict within observation range

3.3 ZF0Max

This configuration item is optional. The default is blank which then allows the algorithm to set the maximum value for Z, i.e. depth. A nonzero value will limit the z value to that setting. This configuration item can also be set on the command line.

3.4 Use Original Data

On each loop for the least squares fit, if true, the algorithm restores the data to the original data. If false, the algorithm will used the residual data.

Chapter 4

Linear Spatial Functions, LSF

Chapter 5

Grid Manager

Chapter 6

Common Parameters

Chapter 7

Modules Index

7.1 Modules List

Here is a list of all modules with brief descriptions:

Chapter 8

Data Type Index

8.1 Data Types List

Here are the data types with brief descriptions:

Chapter 9

File Index

9.1 File List

Here is a list of all files with brief descriptions:

Chapter 10

Module Documentation

10.1 globals Module Reference

Functions/Subroutines

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

Variables

- integer, parameter sp = selected_real_kind(6, 37)
- integer, parameter dp = selected_real_kind(15, 307)
- integer, parameter qp = selected_real_kind(33, 4931)
- integer, parameter ndim = 12000
- integer, parameter shell_len_max = 150
- integer, parameter shell_len_min = 30
- integer, parameter shell_len_delta = 5
- integer, parameter num_size_classes = (shell_len_max - shell_len_min) / shell_len_delta + 1
- integer, parameter max_num_years = 50
- integer, parameter max_num_areas = 25
- integer, parameter max_sides = 8
- integer, parameter region_none =0
- integer, parameter region_n =1
- integer, parameter region_s =2
- integer, parameter region_sw =3
- integer, parameter region_w =4
- integer, parameter region_ma =5
- integer, parameter tag_len = 40
- integer, parameter value_len = 30
- integer, parameter comment_len = 80
- integer, parameter line_len = tag_len+value_len+comment_len
- integer, parameter fname_len = 100
- integer, parameter form_len = 20

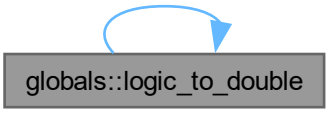
- integer, parameter input_str_len = 100
- integer, parameter csv_line_len = 2000
- integer, parameter domain_len = 2
- integer, parameter read_dev = 69
- integer, parameter write_dev = 63
- real(dp), parameter zero_threshold = 1.0D-99
- real(dp), parameter pi = 3.14159265358979311599796346854D0
- real(dp), parameter grams_per_pound = 453.592_dp
- real(dp), parameter meters_per_naut_mile = 1852.D0
- real(dp), parameter feet_per_naut_mile = 6076.12
- real(dp), parameter grams_per_metric_ton = 1000000._dp
- real(dp), parameter grid_area_sqm = meters_per_naut_mile**2
- real(dp), parameter tow_area_sqm = 4516._dp
- real(dp), parameter one_scallop_per_tow = 1.D0 / tow_area_sqm
- real(dp), parameter ma_gb_border = -70.5
- character(*), parameter term_red = "//achar(27)//'[31m'
- character(*), parameter term_yel = "//achar(27)//'[33m'
- character(*), parameter term_grn = "//achar(27)//'[92m'
- character(*), parameter term_blu = "//achar(27)//'[94m'
- character(*), parameter term_blk = "//achar(27)//'[0m'
- character(*), parameter init_cond_dir = 'InitialCondition/'
- character(*), parameter growth_out_dir = 'GrowthOutput/'
- character(*), parameter rec_input_dir = 'RecruitEstimates/'
- character(*), parameter rec_output_dir = 'RecruitField/'
- character(*), parameter output_dir = 'Results/'
- character(*), parameter config_dir_sim = 'Configuration/Simulation/'
- character(*), parameter config_dir_interp = 'Configuration/Interpolation/'
- character(*), parameter config_dir_special = 'Configuration/SpecialAccess/'
- character(*), parameter grid_dir = 'Grids/'
- character(*), parameter data_dir = 'Data/'

10.1.1 Function/Subroutine Documentation

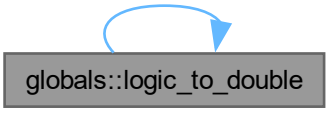
10.1.1.1 logic_to_double()

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

Here is the call graph for this function:



Here is the caller graph for this function:

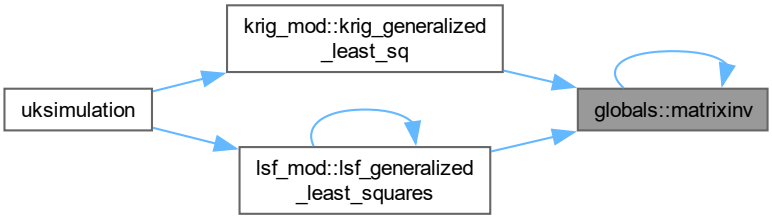


10.1.1.2 matrixinv()

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

Here is the call graph for this function:

Here is the caller graph for this function:



10.1.2 Variable Documentation

10.1.2.1 comment_len

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

10.1.2.2 config_dir_interp

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

10.1.2.3 config_dir_sim

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

10.1.2.4 config_dir_special

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

10.1.2.5 csv_line_len

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

10.1.2.6 data_dir

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

10.1.2.7 domain_len

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

10.1.2.8 dp

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

10.1.2.9 feet_per_naut_mile

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

10.1.2.10 fname_len

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

10.1.2.11 form_len

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

10.1.2.12 grams_per_metric_ton

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

10.1.2.13 grams_per_pound

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

10.1.2.14 grid_area_sqm

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

10.1.2.15 grid_dir

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

10.1.2.16 growth_out_dir

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

10.1.2.17 init_cond_dir

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

10.1.2.18 input_str_len

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

10.1.2.19 line_len

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

10.1.2.20 ma_gb_border

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

10.1.2.21 max_num_areas

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

10.1.2.22 max_num_years

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

10.1.2.23 max_sides

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

10.1.2.24 meters_per_naut_mile

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

10.1.2.25 ndim

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

10.1.2.26 num_size_classes

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

10.1.2.27 one_scallop_per_tow

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

10.1.2.28 output_dir

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

10.1.2.29 pi

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

10.1.2.30 qp

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

10.1.2.31 read_dev

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

10.1.2.32 rec_input_dir

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

10.1.2.33 rec_output_dir

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

10.1.2.34 region_ma

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

10.1.2.35 region_n

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

10.1.2.36 region_none

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

10.1.2.37 region_s

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

10.1.2.38 region_sw

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

10.1.2.39 region_w

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

10.1.2.40 shell_len_delta

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

10.1.2.41 shell_len_max

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

10.1.2.42 shell_len_min

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

10.1.2.43 sp

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

10.1.2.44 tag_len

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

10.1.2.45 term_blk

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

10.1.2.46 term_blu

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

10.1.2.47 term_grn

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

10.1.2.48 term_red

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

10.1.2.49 term_yel

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

10.1.2.50 tow_area_sqm

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

10.1.2.51 value_len

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

10.1.2.52 write_dev

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

10.1.2.53 zero_threshold

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

10.2 grid_manager_mod Module Reference

Data Types

- type grid_data_class

Functions/Subroutines

- subroutine gridmgr_set_grid_manager (obs, grid, alpha_obs, nobs, ngrid)

  Initialize survey and grid location data.
- subroutine gridmgr_set_grid_data_file_name (fname)

  Used during instantiation to set the name of the file to read to for main grid data points.
- subroutine gridmgr_set_obs_data_file_name (fname)

  Used during instantiation to set the name of the file to read to for observation data points.
- character(fname_len) function gridmgr_get_obs_data_file_name ()
- integer function gridmgr_load_grid (x, y, z, lat, lon)

  load grid coordinates and bathymetric depth from CSV file with 5 columns representing an x coordinate, y, bathymetric depth (z), latitude, and longitude.
- integer function gridmgr_load_observation_data (x, y, z, f)

  Load data from CSV file with 4 columns representing an x coordinate, y coordinate, bathymetric depth (z), and a scaller field f.

Variables

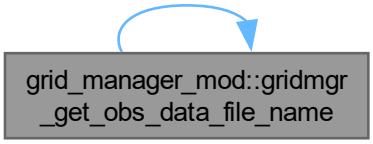
- character(fname_len), private grid_data_file_name
- character(fname_len), private obs_data_file_name

10.2.1 Function/Subroutine Documentation

10.2.1.1 gridmgr_get_obs_data_file_name()

```
character(fname_len) function grid_manager_mod::gridmgr_get_obs_data_file_name
```

Here is the call graph for this function:



Here is the caller graph for this function:

10.2.1.2 gridmgr_load_grid()

```
integer function grid_manager_mod::gridmgr_load_grid (
             real(dp), dimension(*) x,
             real(dp), dimension(*) y,
             real(dp), dimension(*) z,
             real(dp), dimension(*) lat,
             real(dp), dimension(*) lon )
```

load grid coordinates and bathymetric depth from CSV file with 5 columns representing an x coordinate, y, bathymetric depth (z), latitude, and longitude.

Inputs:

- none

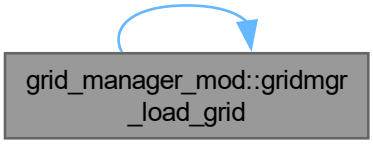
Outputs:

- x (real(dp)) x-coordinate of data
- y (real(dp)) y-coordinate of data
- z (real(dp)) bathymetric depth at (x, y)
- lat (real(dp)) latitude at (x, y)
- lon (real(dp)) longitude at (x, y)
- num_points (integer)length of x, y, z, lat, lon (number of data points)

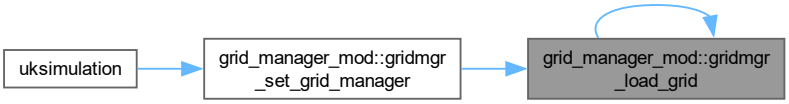
Note: At present lat and lon variabels are not used.

10.2.1.3 @author Keston Smith (IBSS corp) June-July 2021

Here is the call graph for this function:



Here is the caller graph for this function:



10.2.1.4 gridmgr_load_observation_data()

```
integer function grid_manager_mod::gridmgr_load_observation_data (
            real(dp), dimension(*), intent(out) x,
            real(dp), dimension(*), intent(out) y,
            real(dp), dimension(*), intent(out) z,
            real(dp), dimension(*), intent(out) f )
```

Load data from CSV file with 4 columns representing an x coordinate, y coordinate, bathymetric depth (z), and a scaller field f.

Inputs:

- none

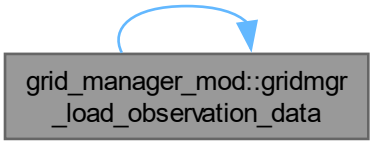
Outputs:

- x (real(dp)) x-coordinate of data
- y (real(dp)) y-coordinate of data
- z (real(dp)) bathymetric depth at (x, y)
- f (real(dp)) scalar data at (x, y)
- num_points (integer)length of x, y, and z (number of data points)

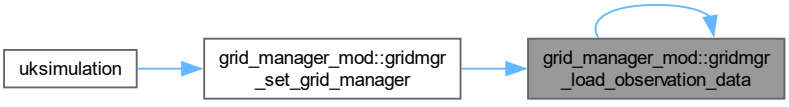
Author

Keston Smith (IBSS corp) June-July 2021

Here is the call graph for this function:



Here is the caller graph for this function:



10.2.1.5 gridmgr_set_grid_data_file_name()

```
subroutine grid_manager_mod::gridmgr_set_grid_data_file_name (
            character(*), intent(in) fname )
```

Used during instantiation to set the name of the file to read to for main grid data points.

Read Input File

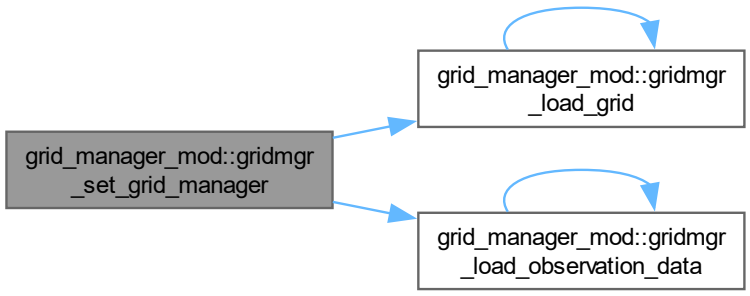
Sets file names for main grid parameters, x, y, lat, lon, depth

10.2.1.6 gridmgr_set_grid_manager()

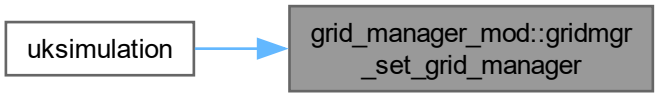
```
subroutine grid_manager_mod::gridmgr_set_grid_manager (
            type(grid_data_class), intent(out) obs,
            type(grid_data_class), intent(out) grid,
            real(dp), intent(in) alpha_obs,
            integer, intent(out) nobs,
            integer, intent(out) ngrid )
```

Initialize survey and grid location data.

Here is the call graph for this function:



Here is the caller graph for this function:



10.2.1.7 gridmgr_set_obs_data_file_name()

```
subroutine grid_manager_mod::gridmgr_set_obs_data_file_name (
            character(*), intent(in) fname )
```

Used during instantiation to set the name of the file to read to for observation data points.

Read Input File

Sets file names for initial state data

10.2.2 Variable Documentation

10.2.2.1 grid_data_file_name

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

10.2.2.2 obs_data_file_name

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

10.3 krig_mod Module Reference

Data Types

- type krig_class

Functions/Subroutines

- real(dp) function, dimension(n_dim, k) krig_compute_distance (p, q, n_dim, k)

  Purpose: Computes distance between two vectors of points Inputs: p: vector 1 q: vector 2 n_dim: allocated length of vectors k: length of q vector, needed for 2nd dim of result Outputs distance.
- real(dp) function, dimension(n_dim, num_cols) krig_compute_variogram (num_points, num_cols, dist, n_dim, par)

  Purpose: Computes a variogram (variogram) given distances between points (dist).
- subroutine krig_comp_emp_variogram (num_points, dist, n_dim, f, par)

  Purpose: Computes a variogram (variogram) given distances between points (dist).
- real(dp) function, dimension(n_dim, num_spat_fcns) krig_eval_spatial_function (p, num_spat_fcns, n_dim, nlsf, save_data)

  Purpose: Computes value of spatial functions at x,y.
- subroutine krig_generalized_least_sq (grid, obs, num_spat_fcns, par, beta, covbeta, eps, cepsg, nlsf, save_data)

  Purpose: Calculates Universal Kriging (UK) estimate at grid points given by coordinates (x, y). Additionally this returns the statistics needed to simulate from the posterior distribution, that is to simulate random fields consistent with the observations, spatial functions, and variogram.

10.3.1 Function/Subroutine Documentation

10.3.1.1 krig_comp_emp_variogram()

```
subroutine krig_mod::krig_comp_emp_variogram (
            integer, intent(in) num_points,
            real(dp), dimension(n_dim,*), intent(in) dist,
            integer, intent(in) n_dim,
            real(dp), dimension(*), intent(in) f,
            type(krig_class), intent(inout) par )
```

Purpose: Computes a variogram (variogram) given distances between points (dist).

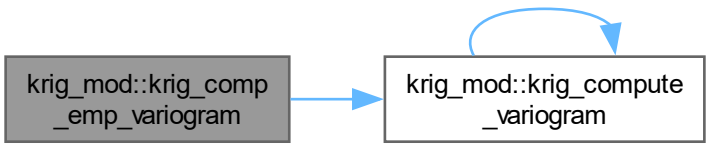
Parameters

in	*num_points*	(integer) number of rows in dist, Gamma
in	*n_dim*	(integer) allocated number of rows dist, Gamma
in	*dist*	(real(dp)) Size (num_points x num_cols) matrix distance between point pairs
in,out	*par*	

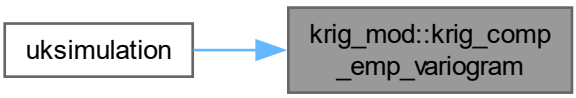
Author

Keston Smith (IBSS corp) June-July 2021

Here is the call graph for this function:



Here is the caller graph for this function:



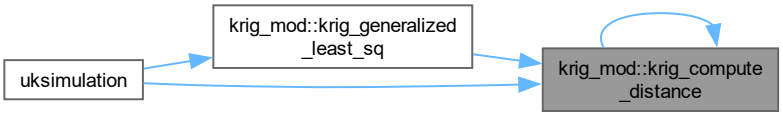
10.3.1.2 krig_compute_distance()

```
real(dp) function, dimension(n_dim,k) krig_mod::krig_compute_distance (
            type(grid_data_class), intent(in) p,
            type(grid_data_class), intent(in) q,
            integer, intent(in) n_dim,
            integer, intent(in) k )
```

Purpose: Computes distance between two vectors of points Inputs: p: vector 1 q: vector 2 n_dim: allocated length of vectors k: length of q vector, needed for 2nd dim of result Outputs distance.

Here is the call graph for this function:

Here is the caller graph for this function:



10.3.1.3 krig_compute_variogram()

```
real(dp) function, dimension(n_dim,num_cols) krig_mod::krig_compute_variogram (
            integer, intent(in) num_points,
            integer, intent(in) num_cols,
            real(dp), dimension(n_dim,*), intent(in) dist,
            integer, intent(in) n_dim,
            type(krig_class), intent(in) par )
```

Purpose: Computes a variogram (variogram) given distances between points (dist).

Parameters

in	*num_points*	(integer) number of rows in dist, Gamma
in	*num_cols*	(integer) number of columns in dist, Gamma
in	*dist*	(real(dp)) Size (num_points x num_cols) matrix distance between point pairs
out	*variogram*	(real(dp)) Size (num_points x num_cols) variogram matrix for point pairs represented in dist

Internal:

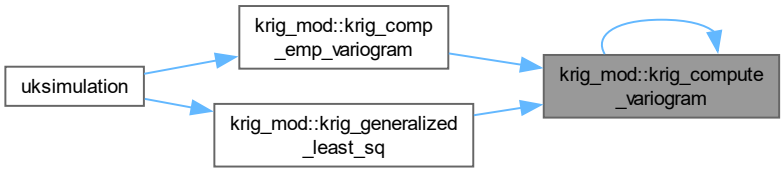
- sill (real(dp)) Variogram parameter sill+nugget = shelf
- nugget (real(dp)) Variogram parameter nugget
- alpha (real(dp)) Variogram length scale parameter
- form (charecter) functional form of variogram
  The above should be read in an input file but are written as constants for now

Author

Keston Smith (IBSS corp) June-July 2021

Here is the call graph for this function:

Here is the caller graph for this function:



10.3.1.4 krig_eval_spatial_function()

```
real(dp) function, dimension(n_dim,num_spat_fcns) krig_mod::krig_eval_spatial_function (
            type(grid_data_class), intent(in) p,
            integer, intent(in) num_spat_fcns,
            integer, intent(in) n_dim,
            type(nlsf_class), dimension(*), intent(in) nlsf,
            logical, intent(in) save_data )
```

Purpose: Computes value of spatial functions at x,y.

Inputs:

Parameters

in	*p*	(Grid_Data_Class) - Spatial points to evaluate functions at
in	*n_dim*	(integer) leading dimension of F
in	*nlsf*	(NLSF) vector of nonlinear spatial functions defined

outputs:

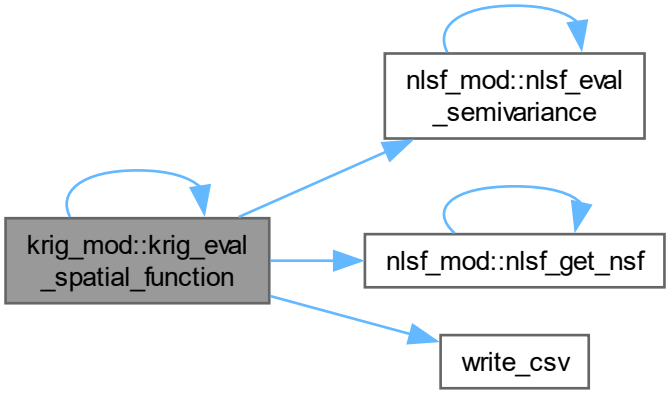
Parameters

out	*F*	(real(dp)) Size (num_points x num_spat_fcns) matrix containg values of spatial functions at points (x,y)
out	*num_spat_fcns*	(integer) number of spatial functions (be carefull allocating F)

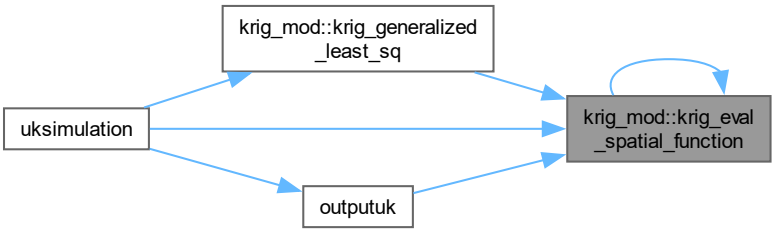
Author

Keston Smith (IBSS corp) June-July 2021

Here is the call graph for this function:



Here is the caller graph for this function:



10.3.1.5 krig_generalized_least_sq()

```
subroutine krig_mod::krig_generalized_least_sq (
             type(grid_data_class) grid,
             type(grid_data_class) obs,
             integer, intent(in) num_spat_fcns,
             type(krig_class) par,
             real(dp), dimension(*), intent(out) beta,
             real(dp), dimension(num_spat_fcns, num_spat_fcns), intent(out) covbeta,
             real(dp), dimension(*), intent(out) eps,
             real(dp), dimension(grid%num_points, grid%num_points), intent(out) cepsg,
```

```
type(nlsf_class), dimension(*) nlsf,
logical, intent(in) save_data )
```

Purpose: Calculates Universal Kriging (UK) estimate at grid points given by coordinates (x, y). Additionally this returns the statistics needed to simulate from the posterior distribution, that is to simulate random fields consistent with the observations, spatial functions, and variogram.

Inputs:

- x (real(dp)) length num_points vector of x-coordinates of grid
- y (real(dp)) length num_points vector of y-coordinates for grid
- z (real(dp)) length num_points vector of bathymetric depth at (x, y)
- num_points (integer) number of points in grid
- xo (real(dp)) length num_obs_points vector of x-coordinates of data
- yo (real(dp)) length num_obs_points vector of y-coordinates for data
- zo (real(dp)) length num_obs_points vector of bathymetric depth at data point (xo, yo)
- fo (real(dp)) length num_obs_points vector of observations at (xo, yo)
- n0 (integer) number of points in grid

Outputs:

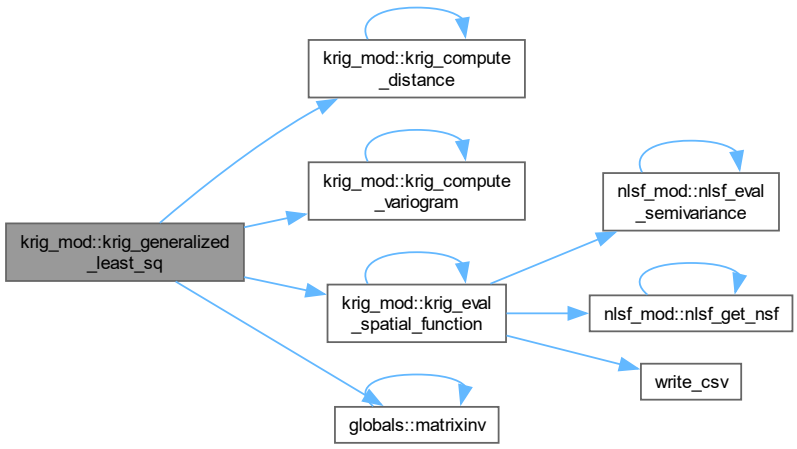
- f (real(dp)) UK estimate (linear in observations, fo) of the field
- beta (real(dp)) length num_spat_fcns vector, best estimate of spatial function coefficients
- CovBeta (real(dp)) (num_spat_fcns x num_spat_fcns) covariance matrix for the estimates of beta
- eps (real(dp)) length num_points vector, best estimate of residual process
- CepsPost(real(dp)) (num_points x num_points) covariance matrix for residual esimate (the spatially coorelated random field component).

Derivation for the UK algorithm and relation to generalized least squares estimation can be found in the book: N. Cressie (1993). Statistics for spatial data. Wiley, New York, 1993.

Author

Keston Smith (IBSS corp) June-July 2021

Here is the call graph for this function:



Here is the caller graph for this function:

10.4 lsf_mod Module Reference

Functions/Subroutines

- real(dp) function, dimension(n) lsf_generalized_least_squares (y, f, c, n, m, beta, cbeta, save_data)

  Purpose:Generalized Least Squares solver. Solve for beta to fit y = F∗beta +epsilon, for epsilon∼N(0, C). Return optimal beta, covariance of beta and residual.
- real(dp) function, dimension(1:n) lsf_simple_linear_regression (y, x, n)

  Purpose: Simple linear regresion minimize sum_{j=1:n} (y(j) - alpha + beta ∗ x(j))∗∗2 over alpha and beta.

10.4.1 Function/Subroutine Documentation

10.4.1.1 lsf_generalized_least_squares()

```
real(dp) function, dimension(n) lsf_mod::lsf_generalized_least_squares (
             real(dp), dimension(*), intent(in) y,
             real(dp), dimension(n,*), intent(in) f,
             real(dp), dimension(n,*), intent(in) c,
             integer, intent(in) n,
             integer, intent(in) m,
             real(dp), dimension(*), intent(out) beta,
             real(dp), dimension(m,m), intent(out) cbeta,
             logical, intent(in) save_data )
```

Purpose:Generalized Least Squares solver. Solve for beta to fit y = F∗beta +epsilon, for epsilon∼N(0, C). Return optimal beta, covariance of beta and residual.

Inputs:

- y (real(dp)) length n vector to fit
- F (real(dp)) size n x m matrix of j=1:m functions evaluated at points k=1:n
- C (real(dp)) size n x n covariance matrix for epsilon
- n (integer) number of points to fit
- m (integer) number of functions in fit

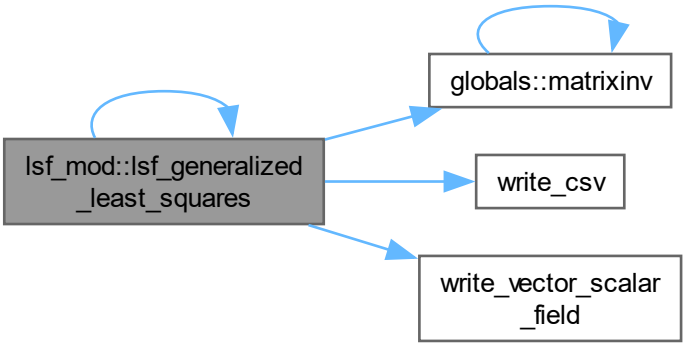
Outputs:

- beta (real(dp)) length m vector of optimal coefficients
- Cbeta (real(dp)) size m x m covariance matrix for beta
- r (real(dp)) length num_points vector of residuals r = y - F ∗ beta

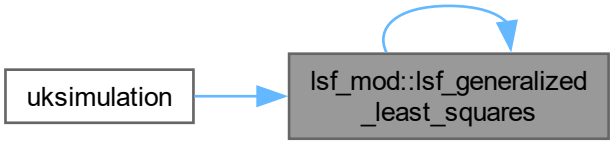
Author

Keston Smith (IBSS corp) June-July 2021

Here is the call graph for this function:



Here is the caller graph for this function:



10.4.1.2 lsf_simple_linear_regression()

```
real(dp) function, dimension(1:n) lsf_mod::lsf_simple_linear_regression (
            real(dp), dimension(*), intent(in) y,
            real(dp), dimension(*), intent(in) x,
            integer, intent(in) n )
```

Purpose: Simple linear regresion minimize sum_{j=1:n} (y(j) - alpha + beta * x(j))**2 over alpha and beta.

Simple Least Squares

Inputs:

- y (real) [n]
- x (real) [n]
- n (integer)

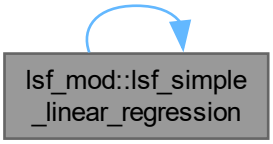
Outputs:

- alpha (real)
- beta (real)
- p (real) [n] p(1:n) = alpha + beta * x(1:n)

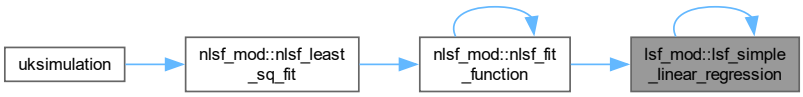
Author

Keston Smith (IBSS corp) June-July 2021

Here is the call graph for this function:



Here is the caller graph for this function:



10.5 nlsf_mod Module Reference

Data Types

- type nlsf_class

Functions/Subroutines

- logical function nlsf_get_is_truncate_range ()

  Getter functions.
- logical function nlsf_get_use_orig_data ()
- integer function nlsf_get_nsf_limit ()
- integer function nlsf_get_nsf ()
- real(dp) function nlsf_get_z_f0_max ()
- subroutine set_config_file_name (fname)

  Used during instantiation to set the name of the file to read to for configuration parameters.
- integer function nlsf_count_defined_functions ()
- integer function nlsf_define_functions (nlsf, p, f0_max)

  Define non linear spatial functions(NLSF) and paramater search range.
- subroutine nlsf_least_sq_fit (obs, nlsf, save_data)

  Purpose: Performs a brute force least squares fit of nonlinear spatial function parameters to data points in obs.
- real(dp) function, dimension(1:p%num_points) nlsf_eval_semivariance (p, nlsf)

  Purpose: Evaluates nonlinear spatial function with parameters x0 ad lambda at points in p. Values returned in vector f.
- real(dp) function nlsf_smooth_penalty (nlsf)

  Purpose: Calculate smoothing penalty for nonlinear spatial function fit. Penalty is based on symbolic integral from -inf to inf of the functions second derivative squared. i.e.: integral (d2 f /d x2)^ 2 from -inf to inf = smoothness penalty -> p see: SageScriptSmoothing.s for derivation input.
- real(dp) function, dimension(1:obs%num_points) nlsf_fit_function (obs, nlsf, y, f)

  Purpose: Fit nonlinear parameters nlsff0 and nlsflambda to observations at num_points points defined in obs with values y. f is a function defined at the observation points. The penalty function is
  sum_i (a+b∗nlsfg_i(x0,lambda)∗f_i - y_i)∗∗2
  where a and b are estimated using simple linear regresion for each nonlinear parameter pair x_0, lambda. The minimization is done with a brute force search over np=500 equally spaced values between (nlsff0_min and nlsff0_max) and (nlsff0_min and nlsff0_max) respectivly.

Variables

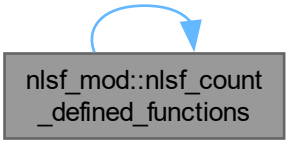
- character(fname_len), private config_file_name
- integer, private nsf
- integer, private nsflim
- logical, private is_truncate_range
- logical, private use_orig_data
- real(dp), private z_f0_max

10.5.1 Function/Subroutine Documentation

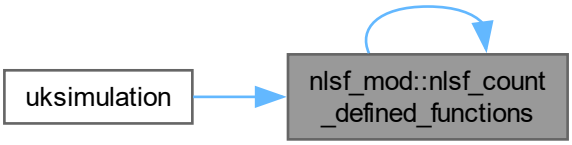
10.5.1.1 nlsf_count_defined_functions()

```
integer function nlsf_mod::nlsf_count_defined_functions
```

Here is the call graph for this function:



Here is the caller graph for this function:



10.5.1.2 nlsf_define_functions()

```
integer function nlsf_mod::nlsf_define_functions (
             type(nlsf_class), dimension(*) nlsf,
             type(grid_data_class) p,
             real(dp), intent(in) f0_max )
```

Define non linear spatial functions(NLSF) and paramater search range.

The file "SpatialFcns.cfg" contains a set of lines of the form

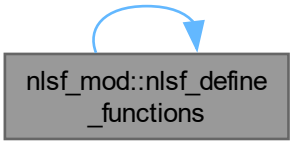
- "Function 1, dim=z, shape=Logistic, precon=0 "
- "Function 2, dim=z, shape=Gaussian, precon=0 "
- "Function 3, dim=x, shape=Logistic, precon=1 "

These define spatial functions for setting the spatial trend in the universal kriging algorithm.

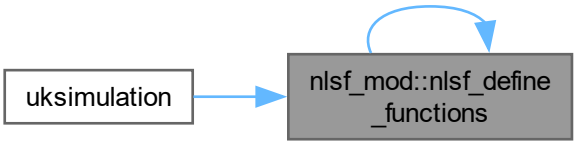
The precon=0 term means that the function is not multiplied by another function. "Function 3, dim=x, shape=Logistic, precon=1 " indicates that the third function is multiplied by the first function. This is true for fitting the nonlinear parameters of function 3 hence the parameters of function 1 must be fit before the parameters of function 3.

The function is called twice from the main program. In the first call InitiialCallFlag= T and the number of nonlinear spatial functions is returned.

In the second call InitiialCallFlag= F and all of the functions are defined. with precon=0 the nonlinear paraters function is fit in a least Here is the call graph for this function:



Here is the caller graph for this function:



10.5.1.3 nlsf_eval_semivariance()

```
real(dp) function, dimension(1:p%num_points) nlsf_mod::nlsf_eval_semivariance (
            type(grid_data_class), intent(in) p,
            type(nlsf_class), intent(in) nlsf )
```

Purpose: Evaluates nonlinear spatial function with parameters x0 ad lambda at points in p. Values returned in vector f.

inputs:

- p: Grid_Manager_Mod(see UniversalKriging.f90) Defines spatial point grid/field

- nlsf: Nonlinear spatial function(see UniversalKriging.f90) Defines a nonlinear spatial function

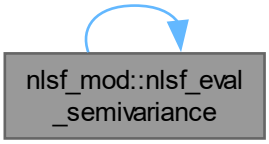
outputs:

- f: pnum_points length vector of values of nlsf at points defined in p.

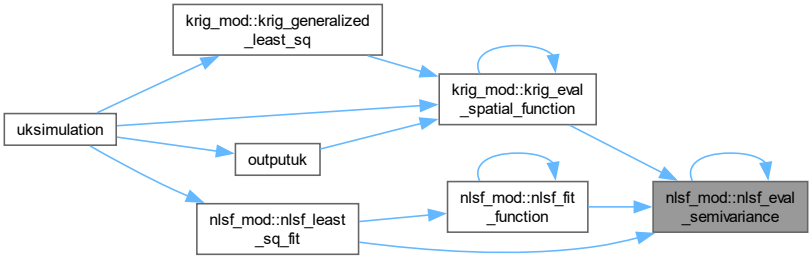
Author

keston Smith (IBSS corp) 2022

Here is the call graph for this function:



Here is the caller graph for this function:



10.5.1.4 nlsf_fit_function()

```
real(dp) function, dimension(1:obs%num_points) nlsf_mod::nlsf_fit_function (
            type(grid_data_class), intent(in) obs,
            type(nlsf_class), intent(inout) nlsf,
            real(dp), dimension(*), intent(in) y,
            real(dp), dimension(*), intent(in) f )
```

Purpose: Fit nonlinear parameters nlsff0 and nlsflambda to observations at num_points points defined in obs with values y. f is a function defined at the observation points. The penalty function is
sum_i (a+b∗nlsfg_i(x0,lambda)∗f_i - y_i)∗∗2
where a and b are estimated using simple linear regresion for each nonlinear parameter pair x_0, lambda. The minimization is done with a brute force search over np=500 equally spaced values between (nlsff0_min and nlsff0_max) and (nlsff0_min and nlsff0_max) respectivly.

inputs:

- obs: Grid_Manager_Mod(see UniversalKriging.f90) Defines spatial observation points
- y: vector of observation values
- f: vector of function values (a preconditioning function) at obs.

input/output

- nlsf: Nonlinear spatial function(see UniversalKriging.f90) Defines a nonlinear spatial function. On exit optimal values of x_0 and lambda are specified

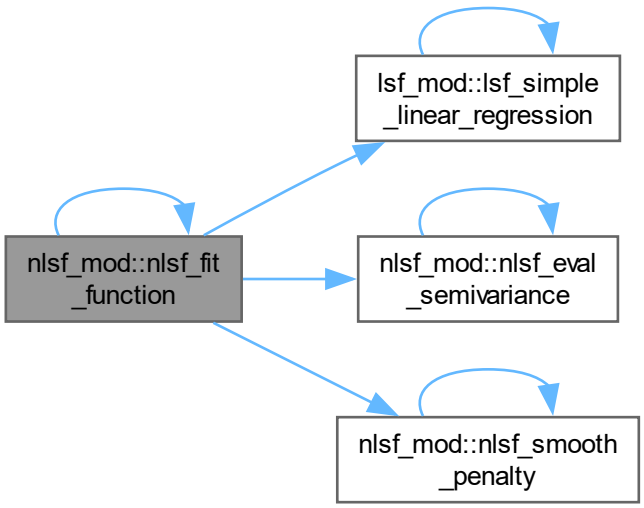
outputs:

- residual: residual, r_i=y_i - nlsf(x_i |x0,lambda) where i=1..# of observations.

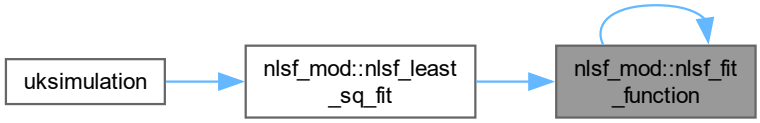
Author

keston Smith (IBSS corp) 2022

Here is the call graph for this function:



Here is the caller graph for this function:

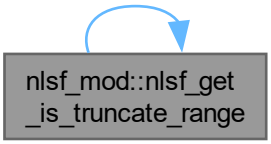


10.5.1.5 nlsf_get_is_truncate_range()

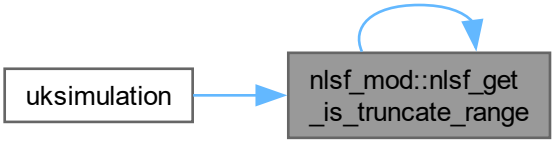
```
logical function nlsf_mod::nlsf_get_is_truncate_range
```

Getter functions.

Here is the call graph for this function:



Here is the caller graph for this function:

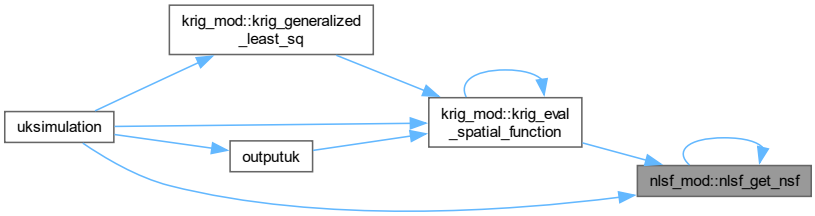


10.5.1.6 nlsf_get_nsf()

```
integer function nlsf_mod::nlsf_get_nsf
```

Here is the call graph for this function:

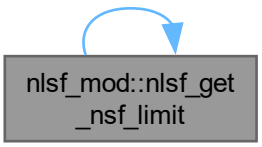
Here is the caller graph for this function:



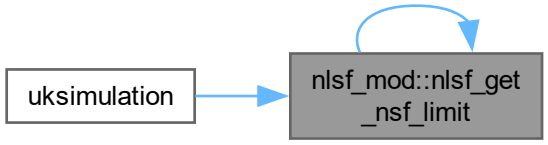
10.5.1.7 nlsf_get_nsf_limit()

```
integer function nlsf_mod::nlsf_get_nsf_limit
```

Here is the call graph for this function:



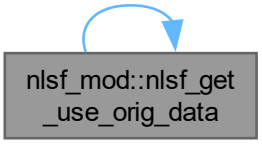
Here is the caller graph for this function:



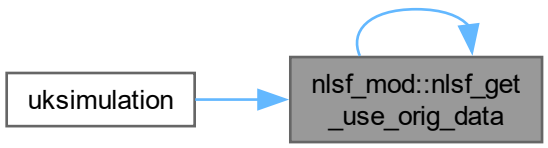
10.5.1.8 nlsf_get_use_orig_data()

```
logical function nlsf_mod::nlsf_get_use_orig_data
```

Here is the call graph for this function:



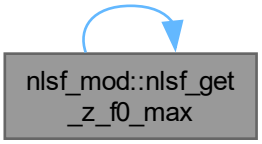
Here is the caller graph for this function:



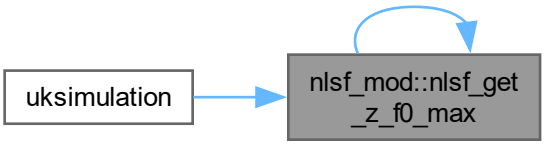
10.5.1.9 nlsf_get_z_f0_max()

```
real(dp) function nlsf_mod::nlsf_get_z_f0_max
```

Here is the call graph for this function:



Here is the caller graph for this function:



10.5.1.10 nlsf_least_sq_fit()

```
subroutine nlsf_mod::nlsf_least_sq_fit (
            type(grid_data_class), intent(in) obs,
            type(nlsf_class), dimension(*), intent(inout) nlsf,
            logical, intent(in) save_data )
```

Purpose: Performs a brute force least squares fit of nonlinear spatial function parameters to data points in obs.

inputs:

- obs: Grid_Manager_Mod(see UniversalKriging.f90) Defines observations to be fit.
- the residual from the preceeding function is fit.

inputs/output:

- nlsf: Nonlinear spatial function(see UniversalKriging.f90). Defines a vector of nonlinear spatial functions. On return nlsf(1:nsf)f0 and nlsf(1:nsf)lambda are specified.

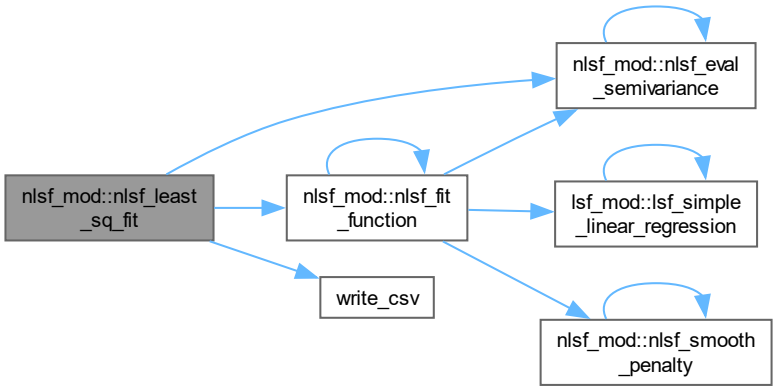
outputs:

- f: pnum_points length vector of values of nlsf at points defined in p.

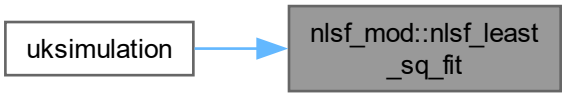
Author

keston Smith (IBSS corp) 2022

Here is the call graph for this function:



Here is the caller graph for this function:



10.5.1.11 nlsf_smooth_penalty()

```
real(dp) function nlsf_mod::nlsf_smooth_penalty (
            type(nlsf_class) nlsf )
```

Purpose: Calculate smoothing penalty for nonlinear spatial function fit. Penalty is based on symbolic integral from -inf to inf of the functions second derivative squared. i.e.: integral (d2 f /d x2)^2 from -inf to inf = smoothness penalty -> p see: SageScriptSmoothing.s for derivation input.

- nlsf: Nonlinear spatial function(see UniversalKriging.f90)

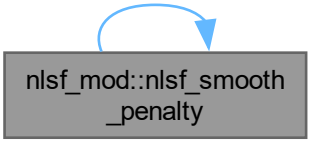
output

- p: smoothness penalty

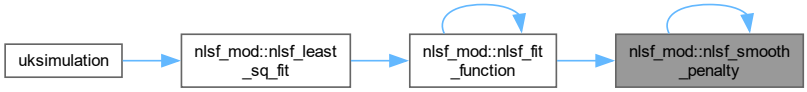
Author

keston Smith (IBSS corp) 2022

Here is the call graph for this function:



Here is the caller graph for this function:



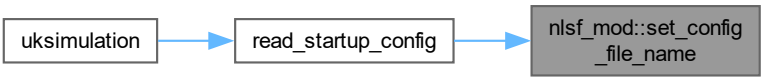
10.5.1.12 set_config_file_name()

```
subroutine nlsf_mod::set_config_file_name (
            character(*), intent(in) fname )
```

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

Read Input File

Sets name of a configuration file, 'config_file_name.cfg' Here is the caller graph for this function:



10.5.2 Variable Documentation

10.5.2.1 config_file_name

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

10.5.2.2 is_truncate_range

```
logical, private nlsf_mod::is_truncate_range  [private]
```

10.5.2.3 nsf

```
integer, private nlsf_mod::nsf  [private]
```

10.5.2.4 nsflim

```
integer, private nlsf_mod::nsflim  [private]
```

10.5.2.5 use_orig_data

```
logical, private nlsf_mod::use_orig_data  [private]
```

10.5.2.6 z_f0_max

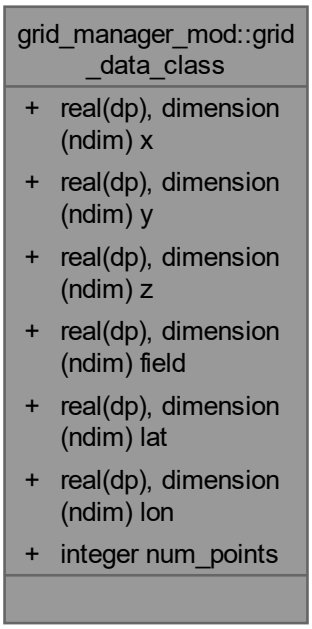
```
real(dp), private nlsf_mod::z_f0_max  [private]
```

Chapter 11

Data Type Documentation

11.1 grid_manager_mod::grid_data_class Type Reference

Collaboration diagram for grid_manager_mod::grid_data_class:



Public Attributes

- real(dp), dimension(ndim) x
- real(dp), dimension(ndim) y
- real(dp), dimension(ndim) z
- real(dp), dimension(ndim) field
- real(dp), dimension(ndim) lat
- real(dp), dimension(ndim) lon
- integer num_points

11.1.1 Member Data Documentation

11.1.1.1 field

```
real(dp), dimension(ndim) grid_manager_mod::grid_data_class::field
```

11.1.1.2 lat

```
real(dp), dimension(ndim) grid_manager_mod::grid_data_class::lat
```

11.1.1.3 lon

```
real(dp), dimension(ndim) grid_manager_mod::grid_data_class::lon
```

11.1.1.4 num_points

```
integer grid_manager_mod::grid_data_class::num_points
```

11.1.1.5 x

```
real(dp), dimension(ndim) grid_manager_mod::grid_data_class::x
```

11.1.1.6 y

```
real(dp), dimension(ndim) grid_manager_mod::grid_data_class::y
```

11.1.1.7 z

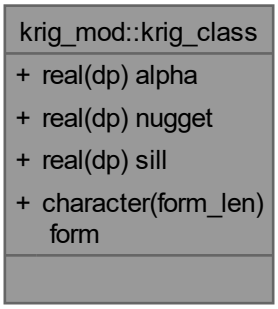
```
real(dp), dimension(ndim) grid_manager_mod::grid_data_class::z
```

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

- UKsrc/UK_GridManager.f90

11.2 krig_mod::krig_class Type Reference

Collaboration diagram for krig_mod::krig_class:



Public Attributes

- real(dp) alpha
- real(dp) nugget
- real(dp) sill
- character(form_len) form

11.2.1 Member Data Documentation

11.2.1.1 alpha

```
real(dp) krig_mod::krig_class::alpha
```

11.2.1.2 form

```
character(form_len) krig_mod::krig_class::form
```

11.2.1.3 nugget

```
real(dp) krig_mod::krig_class::nugget
```

11.2.1.4 sill

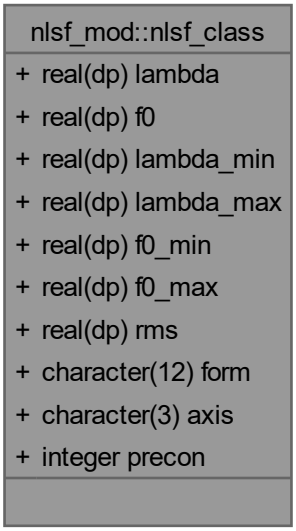
```
real(dp) krig_mod::krig_class::sill
```

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

- UKsrc/KrigingRoutines.f90

11.3 nlsf_mod::nlsf_class Type Reference

Collaboration diagram for nlsf_mod::nlsf_class:



Public Attributes

- real(dp) lambda
- real(dp) f0
- real(dp) lambda_min
- real(dp) lambda_max
- real(dp) f0_min
- real(dp) f0_max
- real(dp) rms
- character(12) form
- character(3) axis
- integer precon

11.3.1 Member Data Documentation

11.3.1.1 axis

```
character(3) nlsf_mod::nlsf_class::axis
```

11.3.1.2 f0

```
real(dp) nlsf_mod::nlsf_class::f0
```

11.3.1.3 f0_max

```
real(dp) nlsf_mod::nlsf_class::f0_max
```

11.3.1.4 f0_min

```
real(dp) nlsf_mod::nlsf_class::f0_min
```

11.3.1.5 form

```
character(12) nlsf_mod::nlsf_class::form
```

11.3.1.6 lambda

```
real(dp) nlsf_mod::nlsf_class::lambda
```

11.3.1.7 lambda_max

```
real(dp) nlsf_mod::nlsf_class::lambda_max
```

11.3.1.8 lambda_min

```
real(dp) nlsf_mod::nlsf_class::lambda_min
```

11.3.1.9 precon

```
integer nlsf_mod::nlsf_class::precon
```

11.3.1.10 rms

```
real(dp) nlsf_mod::nlsf_class::rms
```

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

- UKsrc/NonLinearSpatialFcn.f90

Chapter 12

File Documentation

12.1 UKsrc/aaaUKOrder.f90 File Reference

12.2 UKsrc/Globals.f90 File Reference

Modules

- module globals

Functions/Subroutines

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

Variables

- integer, parameter globals::sp = selected_real_kind(6, 37)
- integer, parameter globals::dp = selected_real_kind(15, 307)
- integer, parameter globals::qp = selected_real_kind(33, 4931)
- integer, parameter globals::ndim = 12000
- integer, parameter globals::shell_len_max = 150
- integer, parameter globals::shell_len_min = 30
- integer, parameter globals::shell_len_delta = 5
- integer, parameter globals::num_size_classes = (shell_len_max - shell_len_min) / shell_len_delta + 1
- integer, parameter globals::max_num_years = 50
- integer, parameter globals::max_num_areas = 25
- integer, parameter globals::max_sides = 8
- integer, parameter globals::region_none =0
- integer, parameter globals::region_n =1
- integer, parameter globals::region_s =2
- integer, parameter globals::region_sw =3

- integer, parameter globals::region_w =4
- integer, parameter globals::region_ma =5
- integer, parameter globals::tag_len = 40
- integer, parameter globals::value_len = 30
- integer, parameter globals::comment_len = 80
- integer, parameter globals::line_len = tag_len+value_len+comment_len
- integer, parameter globals::fname_len = 100
- integer, parameter globals::form_len = 20
- integer, parameter globals::input_str_len = 100
- integer, parameter globals::csv_line_len = 2000
- integer, parameter globals::domain_len = 2
- integer, parameter globals::read_dev = 69
- integer, parameter globals::write_dev = 63
- real(dp), parameter globals::zero_threshold = 1.0D-99
- real(dp), parameter globals::pi = 3.14159265358979311599796346854D0
- real(dp), parameter globals::grams_per_pound = 453.592_dp
- real(dp), parameter globals::meters_per_naut_mile = 1852.D0
- real(dp), parameter globals::feet_per_naut_mile = 6076.12
- real(dp), parameter globals::grams_per_metric_ton = 1000000._dp
- real(dp), parameter globals::grid_area_sqm = meters_per_naut_mile∗∗2
- real(dp), parameter globals::tow_area_sqm = 4516._dp
- real(dp), parameter globals::one_scallop_per_tow = 1.D0 / tow_area_sqm
- real(dp), parameter globals::ma_gb_border = -70.5
- character(∗), parameter globals::term_red = ''//achar(27)//'[31m'
- character(∗), parameter globals::term_yel = ''//achar(27)//'[33m'
- character(∗), parameter globals::term_grn = ''//achar(27)//'[92m'
- character(∗), parameter globals::term_blu = ''//achar(27)//'[94m'
- character(∗), parameter globals::term_blk = ''//achar(27)//'[0m'
- character(∗), parameter globals::init_cond_dir = 'InitialCondition/'
- character(∗), parameter globals::growth_out_dir = 'GrowthOutput/'
- character(∗), parameter globals::rec_input_dir = 'RecruitEstimates/'
- character(∗), parameter globals::rec_output_dir = 'RecruitField/'
- character(∗), parameter globals::output_dir = 'Results/'
- character(∗), parameter globals::config_dir_sim = 'Configuration/Simulation/'
- character(∗), parameter globals::config_dir_interp = 'Configuration/Interpolation/'
- character(∗), parameter globals::config_dir_special = 'Configuration/SpecialAccess/'
- character(∗), parameter globals::grid_dir = 'Grids/'
- character(∗), parameter globals::data_dir = 'Data/'

12.3 UKsrc/IORoutines.f90 File Reference

Functions/Subroutines

- subroutine read_scalar_field (file_name, m, vector_len)
- subroutine write_2d_scalar_field (nn, nsim, f, flnm, nndim)

  Purpose: Write columns of a matrix (f) to a series of text files in exponential format. Inputs:
- subroutine write_vector_scalar_field (vector_len, f, file_name)
- subroutine write_csv (n, m, f, file_name, nndim, append)

  Purpose: Write values of a matrix (f) to a csv file in exponential format. Inputs:
- subroutine write_column_csv (n, f, header, file_name, append)
- subroutine write_csv_h (n, m, f, file_name, nndim, header)
- subroutine read_csv (num_rows, num_cols, file_name, m, nndim)

12.3.1 Function/Subroutine Documentation

12.3.1.1 read_csv()

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

12.3.1.2 read_scalar_field()

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

12.3.1.3 write_2d_scalar_field()

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

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

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

Author

Keston Smith (IBSS corp) June-July 2021

12.3.1.4 write_column_csv()

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

12.3.1.5 write_csv()

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

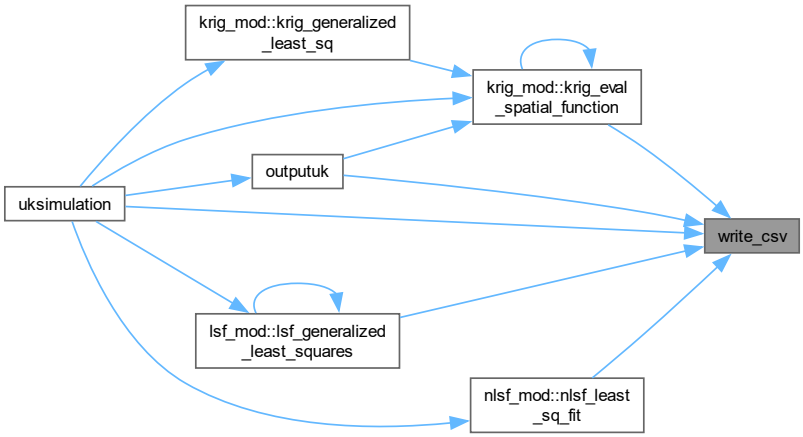
Purpose: Write values of a matrix (f) to a csv file in exponential format. Inputs:

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

Author

Keston Smith (IBSS corp) June-July 2021

Here is the caller graph for this function:



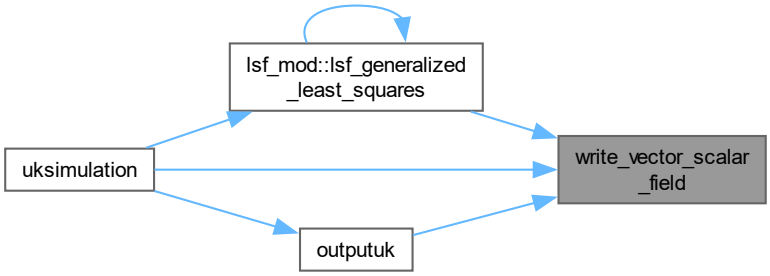
12.3.1.6 write_csv_h()

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

12.3.1.7 write_vector_scalar_field()

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

Here is the caller graph for this function:



12.4 UKsrc/KrigingRoutines.f90 File Reference

Data Types

- type krig_mod::krig_class

Modules

- module krig_mod

Functions/Subroutines

- real(dp) function, dimension(n_dim, k) krig_mod::krig_compute_distance (p, q, n_dim, k)

  Purpose: Computes distance between two vectors of points Inputs: p: vector 1 q: vector 2 n_dim: allocated length of vectors k: length of q vector, needed for 2nd dim of result Outputs distance.
- real(dp) function, dimension(n_dim, num_cols) krig_mod::krig_compute_variogram (num_points, num_cols, dist, n_dim, par)

  Purpose: Computes a variogram (variogram) given distances between points (dist).
- subroutine krig_mod::krig_comp_emp_variogram (num_points, dist, n_dim, f, par)

  Purpose: Computes a variogram (variogram) given distances between points (dist).
- real(dp) function, dimension(n_dim, num_spat_fcns) krig_mod::krig_eval_spatial_function (p, num_spat_fcns, n↩ _dim, nlsf, save_data)

  Purpose: Computes value of spatial functions at x,y.
- subroutine krig_mod::krig_generalized_least_sq (grid, obs, num_spat_fcns, par, beta, covbeta, eps, cepsg, nlsf, save_data)

  Purpose: Calculates Universal Kriging (UK) estimate at grid points given by coordinates (x, y). Additionally this returns the statistics needed to simulate from the posterior distribution, that is to simulate random fields consistent with the observations, spatial functions, and variogram.

12.5 UKsrc/LinearSpatialFcn.f90 File Reference

Modules

- module lsf_mod

Functions/Subroutines

- real(dp) function, dimension(n) lsf_mod::lsf_generalized_least_squares (y, f, c, n, m, beta, cbeta, save_data)

  Purpose:Generalized Least Squares solver. Solve for beta to fit y = F∗beta +epsilon, for epsilon∼N(0, C). Return optimal beta, covariance of beta and residual.
- real(dp) function, dimension(1:n) lsf_mod::lsf_simple_linear_regression (y, x, n)

  Purpose: Simple linear regresion minimize sum_{j=1:n} (y(j) - alpha + beta ∗ x(j))∗∗2 over alpha and beta.

12.6 UKsrc/NonLinearSpatialFcn.f90 File Reference

Data Types

- type nlsf_mod::nlsf_class

Modules

- module nlsf_mod

Functions/Subroutines

- logical function nlsf_mod::nlsf_get_is_truncate_range ()

  Getter functions.
- logical function nlsf_mod::nlsf_get_use_orig_data ()
- integer function nlsf_mod::nlsf_get_nsf_limit ()
- integer function nlsf_mod::nlsf_get_nsf ()
- real(dp) function nlsf_mod::nlsf_get_z_f0_max ()
- subroutine nlsf_mod::set_config_file_name (fname)

  Used during instantiation to set the name of the file to read to for configuration parameters.
- integer function nlsf_mod::nlsf_count_defined_functions ()
- integer function nlsf_mod::nlsf_define_functions (nlsf, p, f0_max)

  Define non linear spatial functions(NLSF) and paramater search range.
- subroutine nlsf_mod::nlsf_least_sq_fit (obs, nlsf, save_data)

  Purpose: Performs a brute force least squares fit of nonlinear spatial function parameters to data points in obs.
- real(dp) function, dimension(1:p%num_points) nlsf_mod::nlsf_eval_semivariance (p, nlsf)

  Purpose: Evaluates nonlinear spatial function with parameters x0 ad lambda at points in p. Values returned in vector f.
- real(dp) function nlsf_mod::nlsf_smooth_penalty (nlsf)

  Purpose: Calculate smoothing penalty for nonlinear spatial function fit. Penalty is based on symbolic integral from -inf to inf of the functions second derivative squared. i.e.: integral (d2 f /d x2)^ 2 from -inf to inf = smoothness penalty -> p see: SageScriptSmoothing.s for derivation input.
- real(dp) function, dimension(1:obs%num_points) nlsf_mod::nlsf_fit_function (obs, nlsf, y, f)

  Purpose: Fit nonlinear parameters nlsff0 and nlsflambda to observations at num_points points defined in obs with values y. f is a function defined at the observation points. The penalty function is
  sum_i (a+b∗nlsfg_i(x0,lambda)∗f_i - y_i)∗∗2
  where a and b are estimated using simple linear regresion for each nonlinear parameter pair x_0, lambda. The minimization is done with a brute force search over np=500 equally spaced values between (nlsff0_min and nlsff0_max) and (nlsff0_min and nlsff0_max) respectivly.

Variables

- character(fname_len), private nlsf_mod::config_file_name
- integer, private nlsf_mod::nsf
- integer, private nlsf_mod::nsflim
- logical, private nlsf_mod::is_truncate_range
- logical, private nlsf_mod::use_orig_data
- real(dp), private nlsf_mod::z_f0_max

12.7 UKsrc/UK_GridManager.f90 File Reference

Data Types

- type grid_manager_mod::grid_data_class

Modules

- module grid_manager_mod

Functions/Subroutines

- subroutine grid_manager_mod::gridmgr_set_grid_manager (obs, grid, alpha_obs, nobs, ngrid)

  Initialize survey and grid location data.
- subroutine grid_manager_mod::gridmgr_set_grid_data_file_name (fname)

  Used during instantiation to set the name of the file to read to for main grid data points.
- subroutine grid_manager_mod::gridmgr_set_obs_data_file_name (fname)

  Used during instantiation to set the name of the file to read to for observation data points.
- character(fname_len) function grid_manager_mod::gridmgr_get_obs_data_file_name ()
- integer function grid_manager_mod::gridmgr_load_grid (x, y, z, lat, lon)

  load grid coordinates and bathymetric depth from CSV file with 5 columns representing an x coordinate, y, bathymetric depth (z), latitude, and longitude.
- integer function grid_manager_mod::gridmgr_load_observation_data (x, y, z, f)

  Load data from CSV file with 4 columns representing an x coordinate, y coordinate, bathymetric depth (z), and a scaller field f.

Variables

- character(fname_len), private grid_manager_mod::grid_data_file_name
- character(fname_len), private grid_manager_mod::obs_data_file_name

12.8 UKsrc/UniversalKriging.f90 File Reference

Functions/Subroutines

- program uksimulation
- subroutine read_startup_config (domain_name, par, alpha_obs, save_data, f0_max, overflow_thresh, use_↩ saturate)

  Purpose: Read parameter values, flags, etc. from an ascii text input file:"UK.cfg". Parameters etc. to be read from UK.cfg are identified by tag before '='. Values are read from the line to the right of an "=" character. Logical variables are read from 'T','F'.
- subroutine outputuk (num_points, num_spat_fcns, grid, nlsf, beta, eps, ceps, cbeta, sf, alpha_obs)

  "KrigingEstimate.txt" Predictor standard deviation is output to "KrigSTD.txt". Function coefficient
- subroutine outputestimates (num_points, grid, ceps, sf, alpha_obs, overflow_thresh, use_saturate)

  prediction. The file name is take for the observation file name that resides in the Data subdirectory and then written to the Results subdirectory.

12.8.1 Function/Subroutine Documentation

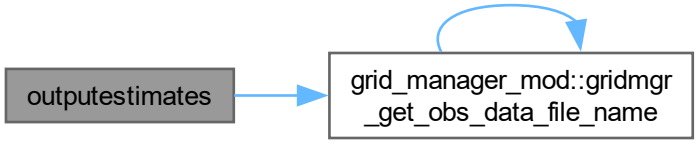
12.8.1.1 outputestimates()

```
subroutine outputestimates (
             integer, intent(in) num_points,
             type(grid_data_class), intent(inout) grid,
             real(dp), dimension(num_points,*), intent(in) ceps,
             real(dp), intent(in) sf,
             real(dp), intent(in) alpha_obs,
             real(dp), intent(in) overflow_thresh,
             logical, intent(in) use_saturate )
```

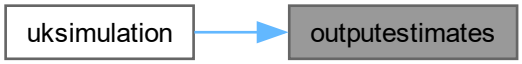
prediction. The file name is take for the observation file name that resides in the Data subdirectory and then written to the Results subdirectory.

Also want to change the name from X_Y_ (UTM) to Lat_Lon_Grid_ (Navigation)

That is moving from survey data locations to MA/GB grid locations Here is the call graph for this function:



Here is the caller graph for this function:

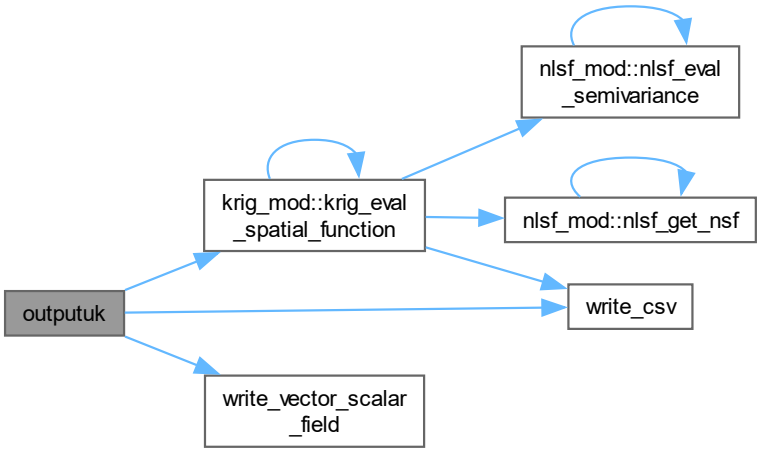


12.8.1.2 outputuk()

```
subroutine outputuk (
             integer, intent(in) num_points,
             integer, intent(in) num_spat_fcns,
             type(grid_data_class), intent(inout) grid,
             type(nlsf_class), dimension(*), intent(in) nlsf,
             real(dp), dimension(*), intent(in) beta,
             real(dp), dimension(*), intent(in) eps,
             real(dp), dimension(num_points,*), intent(inout) ceps,
             real(dp), dimension(num_spat_fcns,*), intent(in) cbeta,
             real(dp), intent(in) sf,
             real(dp), intent(in) alpha_obs )
```

"KrigingEstimate.txt" Predictor standard deviation is output to "KrigSTD.txt". Function coefficient

Here is the call graph for this function:



Here is the caller graph for this function:

12.8.1.3 read_startup_config()

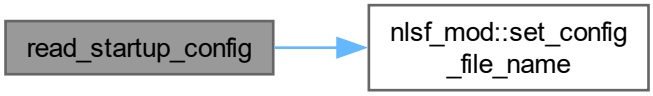
```
subroutine read_startup_config (
             character(2), intent(out) domain_name,
             type(krig_class), intent(out) par,
             real(dp), intent(out) alpha_obs,
             logical, intent(out) save_data,
             real(dp), intent(out) f0_max,
             real(dp), intent(out) overflow_thresh,
             logical, intent(out) use_saturate )
```

Purpose: Read parameter values, flags, etc. from an ascii text input file:"UK.cfg". Parameters etc. to be read from UK.cfg are identified by tag before '='. Values are read from the line to the right of an "=" character. Logical variables are read from 'T','F'.

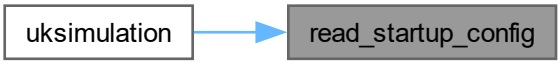
This method also considers arguments passed on the command line that determine

- Name of the configuration file
- Name of the observation file Or to override config file settings used in batch mode
- Domain
- Grid File
- Z f0 max

outputs: all variables Here is the call graph for this function:



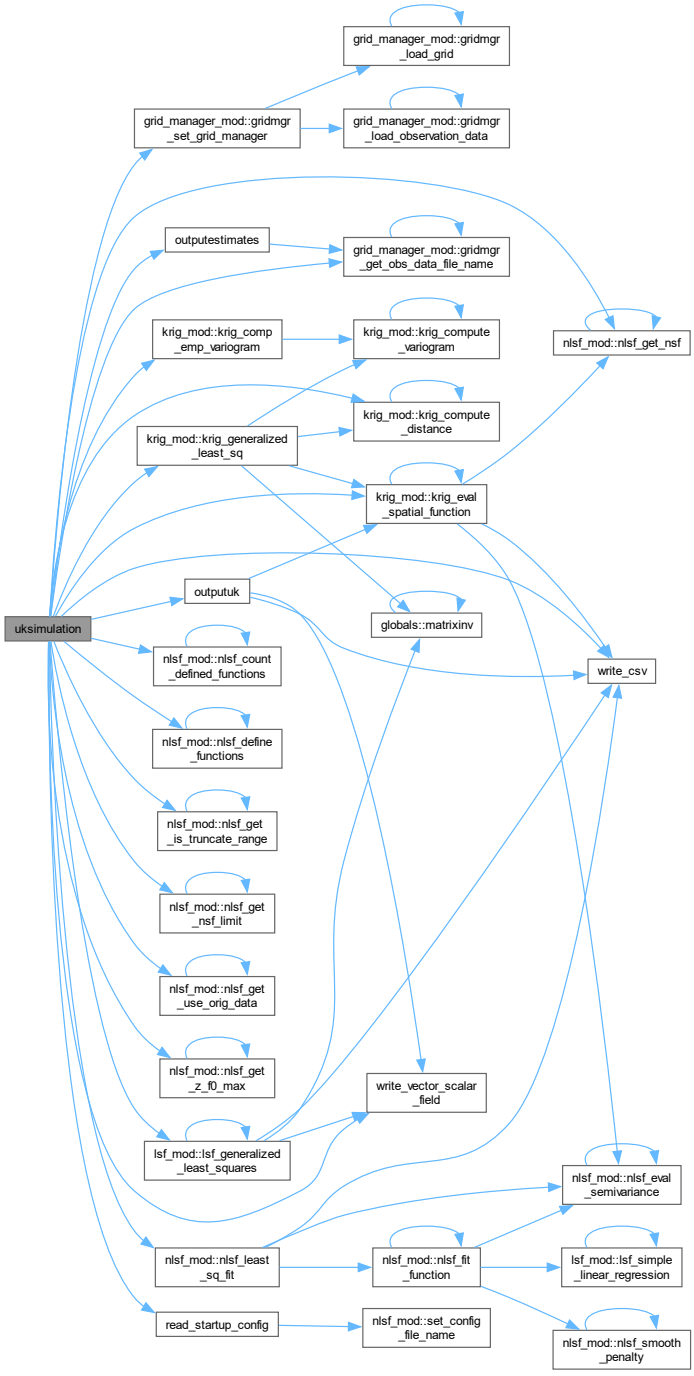
Here is the caller graph for this function:



12.8.1.4 uksimulation()

```
program uksimulation
```

Here is the call graph for this function:



Index