Melissa

Generated by Doxygen 1.8.13

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Module Documentation

4.1 misc functions

Functions

- void melissa_logo ()
- void ∗ melissa_malloc (size_t size)
- void ∗ melissa_calloc (size_t num, size_t size)
- void melissa_free (void ∗ptr)
- void melissa_bind (void ∗socket, char ∗port_name)
- void melissa_connect (void ∗socket, char ∗port_name)
- double melissa_get_time ()
- void melissa_get_node_name (char ∗node_name)
- void set_bit (int32_t ∗vect, int pos)
- void clear_bit (int32_t ∗vect, int pos)
- int test_bit (int32_t ∗vect, int pos)

4.1.1 Detailed Description

4.1.2 Function Documentation

4.1.2.1 clear_bit()

```
void clear_bit (
            int32_t * vect,
            int pos )
```

Sets a bit to 0 in an array of bits

Parameters

out	∗*vect*	pointer to the array of bits
in	∗*pos*	position in the array of bits to set to 0

4.1.2.2 melissa_bind()

```
void melissa_bind (
            void * socket,
            char * port_name )
```

Wraper around zmq_bind

Parameters

in	*socket	ZMQ socket handler
in	*port_name*	Port name

4.1.2.3 melissa_calloc()

```
void* melissa_calloc (
            size_t num,
            size_t size )
```

Wraper around melissa_calloc

Parameters

in	*num*	Number of elements to allocate
in	*size*	Size of one element

Returns

The pointer to the allocated memory

4.1.2.4 melissa_connect()

```
void melissa_connect (
            void * socket,
            char * port_name )
```

Wraper around zmq_connect

Parameters

in	*socket	ZMQ socket handler
in	*port_name*	Port name

4.1.2.5 melissa_free()

```
void melissa_free (
            void * ptr )
```

Free and nullify a pointer

Parameters

in	*ptr	The pointer to free and nullify

4.1.2.6 melissa_get_node_name()

```
void melissa_get_node_name (
            char * node_name )
```

Gets the name of the processus node

Parameters

out	*node_name	The node name

4.1.2.7 melissa_get_time()

```
double melissa_get_time ( )
```

Returns an elapsed time

Returns

elapsed time

4.1.2.8 melissa_logo()

```
void melissa_logo ( )
```

Prints Melissa logo

4.1.2.9 melissa_malloc()

```
void* melissa_malloc (
            size_t size )
```

Wraper around melissa_malloc

Parameters

in	*size*	Number of bytes to allocate

Returns

The pointer to the allocated memory

4.1.2.10 set_bit()

```
void set_bit (
            int32_t * vect,
            int pos )
```

Sets a bit to 1 in an array of bits

Parameters

out	**vect*	pointer to the array of bits
in	**pos*	position in the array of bits to set to 1

4.1.2.11 test_bit()

```
int test_bit (
            int32_t * vect,
            int pos )
```

Tests a value in an array of bits

Parameters

out	**vect*	pointer to the array of bits
in	**pos*	position to test in the array of bits

4.2 internal API

Functions

- void compute_stats (melissa_data_t ∗data, const int time_step, const int nb_vect, double ∗∗in_vect_tab, const int group_id)
- void finalize_stats (melissa_data_t ∗data)

4.2.1 Detailed Description

4.2.2 Function Documentation

4.2.2.1 compute_stats()

```
void compute_stats (
            melissa_data_t * data,
            const int time_step,
            const int nb_vect,
            double ** in_vect_tab,
            const int group_id )
```

This function updates the statistics stored in the data structure

Parameters

in	∗*data*	pointer to the structure containing global parameters
in	*time_step*	time step of the current simulation
in	*nb_vect*	number of input vectors
in	∗∗*in_vect_tab*	array of input vectors
in	*group_id*	ID of the simulation providing in_vect_tab

4.2.2.2 finalize_stats()

```
void finalize_stats (
            melissa_data_t * data )
```

This function finalize the statistics stored in the data structure

Parameters

in	∗*data*	pointer to the structure containing global parameters

4.3 Melissa fields

4.4 Melissa data

Functions

- void melissa_init_data (melissa_data_t *data, melissa_options_t *options, int vect_size)
- void melissa_free_data (melissa_data_t *data)

4.4.1 Detailed Description

4.4.2 Function Documentation

4.4.2.1 melissa_free_data()

```
void melissa_free_data (
            melissa_data_t * data )
```

This function frees the memory in the data structure

Parameters

in	*data	pointer to the structure containing global parameters

4.4.2.2 melissa_init_data()

```
void melissa_init_data (
            melissa_data_t * data,
            melissa_options_t * options,
            int vect_size )
```

This function initializes the data structure

Parameters

out	*data	pointer to the structure containing global parameters
in	*options	pointer to the structure containing options parsed from command line
in	vect_size	sise of the local input vector

4.5 input, output and checkpoint functions

Functions

- void write_client_data (int *client_comm_size, int *client_vect_sizes)
- int read_client_data (int *client_comm_size, int **client_vect_sizes, melissa_options_t *options)
- void save_stats (melissa_data_t *data, comm_data_t *comm_data, char *field_name)
- void read_saved_stats (melissa_data_t *data, comm_data_t *comm_data, char *field_name, int client_rank)
- void save_simu_states (int *simu_states, comm_data_t *comm_data, int size)
- void read_simu_states (int *simu_states, melissa_options_t *options, comm_data_t *comm_data, int size)
- void write_stats_bin (melissa_data_t **data, melissa_options_t *options, comm_data_t *comm_data, int *local_vect_sizes, char *field)
- void write_stats_ensight (melissa_data_t **data, melissa_options_t *options, comm_data_t *comm_data, int *local_vect_sizes, char *field)
- void read_ensight (melissa_options_t *options, comm_data_t *comm_data, double *in_vect, int *local_↩ vect_sizes, char *file_name)
- void write_stats_txt (melissa_data_t **data, melissa_options_t *options, comm_data_t *comm_data, int *local_vect_sizes, char *field)

4.5.1 Detailed Description

4.5.2 Function Documentation

4.5.2.1 read_client_data()

```
int read_client_data (
            int * client_comm_size,
            int ** client_vect_sizes,
            melissa_options_t * options )
```

This function reads a saved option structure on disc

Parameters

out	*client_comm_size	client MPI communicator size
out	**client_vect_sizes	client vector sizes
in	*options	Melissa option structure

4.5.2.2 read_ensight()

```
void read_ensight (
            melissa_options_t * options,
            comm_data_t * comm_data,
            double * in_vect,
```

```
            int * local_vect_sizes,
            char * file_name )
```

This function reads data from Ensight files

Parameters

in	*∗options*	option structure
in	*∗comm_data*	structure containing communications parameters
in	*∗in_vect*	input vector
in	*∗local_vect_sizes*	all local vector sises
in	*∗file_name*	name of the file

4.5.2.3 read_saved_stats()

```
void read_saved_stats (
            melissa_data_t * data,
            comm_data_t * comm_data,
            char * field_name,
            int client_rank )
```

This function reads stats saved on disc

Parameters

in	*∗data*	data structure to read
in	*∗comm_data*	communication structure
in	*∗field_name*	name of the field to read
in	*client_rank*	mpi rank of sending client process

4.5.2.4 read_simu_states()

```
void read_simu_states (
            int * simu_states,
            melissa_options_t * options,
            comm_data_t * comm_data,
            int size )
```

This function reads simulation states from disc

Parameters

out	*∗simu_states*	array of simulation states
in	*∗options*	Melissa option structure
in	*∗comm_data*	communication structure
in	*size*	size of ∗simu_states

4.5.2.5 save_simu_states()

```
void save_simu_states (
            int * simu_states,
            comm_data_t * comm_data,
            int size )
```

This function saves current simulation states on disc

Parameters

in	*∗simu_states*	array of simulation states
in	*∗comm_data*	communication structure
in	*size*	size of ∗simu_states

4.5.2.6 save_stats()

```
void save_stats (
            melissa_data_t * data,
            comm_data_t * comm_data,
            char * field_name )
```

This function saves stats on disc

Parameters

in	*∗data*	data structure to save
in	*∗comm_data*	communication structure
in	*∗field_name*	name of the field to write

4.5.2.7 write_client_data()

```
void write_client_data (
            int * client_comm_size,
            int * client_vect_sizes )
```

This function saves some client data on disc

Parameters

in	*∗client_comm_size*	client MPI communicator size
in	*∗client_vect_sizes*	client vector sizes

4.5.2.8 write_stats_bin()

```
void write_stats_bin (
            melissa_data_t ** data,
            melissa_options_t * options,
            comm_data_t * comm_data,
            int * local_vect_sizes,
            char * field )
```

This function writes the computed statistics on files

Parameters

in	∗∗*data*	pointer to the array of structures containing statistics data
in	∗*options*	Melissa option structure
in	*comm_data*	structure containing communications parameters
in	∗*local_vect_sizes*	all local vector sises
in	∗*field*	name of the field on which are computed the statistics

4.5.2.9 write_stats_ensight()

```
void write_stats_ensight (
            melissa_data_t ** data,
            melissa_options_t * options,
            comm_data_t * comm_data,
            int * local_vect_sizes,
            char * field )
```

This function writes the computed statistics on files

Parameters

in	∗∗*data*	pointer to the array of structures containing statistics data
in	∗*options*	option structure
in	∗*comm_data*	structure containing communications parameters
in	∗*local_vect_sizes*	all local vector sises
in	∗*field*	name of the field on which are computed the statistics

4.5.2.10 write_stats_txt()

```
void write_stats_txt (
            melissa_data_t ** data,
```

```
melissa_options_t * options,
comm_data_t * comm_data,
int * local_vect_sizes,
char * field )
```

This function writes the computed statistics on files

Parameters

in	∗∗*data*	pointer to the array of structures containing statistics data
in	∗*options*	Melissa option structure
in	*comm_data*	structure containing communications parameters
in	∗*local_vect_sizes*	all local vector sises
in	∗*field*	name of the field on which are computed the statistics

4.6 Get options from command line

Functions

- void melissa_print_options (melissa_options_t ∗options)
- void melissa_get_options (int argc, char ∗∗argv, melissa_options_t ∗options)
- void melissa_check_options (melissa_options_t ∗options)
- void melissa_write_options (melissa_options_t ∗options)
- int melissa_read_options (melissa_options_t ∗options)

4.6.1 Detailed Description

4.6.2 Function Documentation

4.6.2.1 melissa_check_options()

```
void melissa_check_options (
            melissa_options_t * options )
```

This function validates the option structure

Parameters

out	∗*options*	pointer to the structure containing options parameters

4.6.2.2 melissa_get_options()

```
void melissa_get_options (
            int argc,
            char ** argv,
            melissa_options_t * options )
```

This function parses command line options and fill the parameter structure

Parameters

in	*argc*	argc
in	∗∗*argv*	argv
out	∗*options*	pointer to the structure containing options parameters

4.6.2.3 melissa_print_options()

```
void melissa_print_options (
            melissa_options_t * options )
```

This function displays the global parameters on stdout

Parameters

in	*options	pointer to the structure containing the options parsed from command line

4.6.2.4 melissa_read_options()

```
int melissa_read_options (
            melissa_options_t * options )
```

This function reads a saved option structure on disc

Parameters

in,out	*options	pointer to the structure containing global options

4.6.2.5 melissa_write_options()

```
void melissa_write_options (
            melissa_options_t * options )
```

This function writes the option structure on disc

Parameters

in	*options	pointer to the structure containing global options

4.7 API

Classes

- struct zmq_data_s

Functions

- void melissa_init (const int *local_vect_size, const int *comm_size, const int *rank, const int *sobol_rank, const int *sample_id, MPI_Comm *comm, const int *coupling)
- void melissa_init_no_mpi (const int *vect_size, const int *sobol_rank, const int *sample_id)
- void melissa_send (const int *time_step, const char *field_name, double *send_vect, const int *rank, const int *sobol_rank, const int *sample_id)
- void melissa_send_no_mpi (const int *time_step, const char *field_name, double *send_vect, const int *sobol_rank, const int *sample_id)
- void melissa_finalize ()

4.7.1 Detailed Description

4.7.2 Function Documentation

4.7.2.1 melissa_finalize()

```
void melissa_finalize ( )
```

This function disconects the simulation from the statistic library

4.7.2.2 melissa_init()

```
void melissa_init (
            const int * local_vect_size,
            const int * comm_size,
            const int * rank,
            const int * sobol_rank,
            const int * sample_id,
            MPI_Comm * comm,
            const int * coupling )
```

This function initialise connexion with the stats library

Parameters

in	*local_vect_size	sise of the local data vector to send to the library
in	*comm_size	sise of the MPI communicator comm
in	*rank	MPI rank
in	*sobol_rank	Sobol indice rank in Sobol group
in	*sample_id	ID of the parameter set defining the simulation
in	*comm	MPI communicator
in	*coupling	1 if simulation are coupled in the same MPI_COMM_WORLD, 0 otherwhise

4.7.2.3 melissa_init_no_mpi()

```
void melissa_init_no_mpi (
            const int * vect_size,
            const int * sobol_rank,
            const int * sample_id )
```

This function initialise connexion with Melissa Server for sequential simulations

Parameters

in	*vect_size	sise of the data vector to send to the library
in	*sobol_rank	Sobol indice rank in Sobol group
in	*sample_id	ID of the parameter set defining the simulation

4.7.2.4 melissa_send()

```
void melissa_send (
            const int * time_step,
            const char * field_name,
            double * send_vect,
            const int * rank,
            const int * sobol_rank,
            const int * sample_id )
```

This function sends data to Melissa Server

Parameters

in	*time_step*	current time step of the simulation
in	*field_name	name of the field to send to Melissa Server
in	*send_vect	local data array to send to the statistic library
in	*rank	MPI rank
in	*sobol_rank	Sobol indice rank in Sobol group
in	*sample_id	ID of the parameter set defining the simulation

4.7.2.5 melissa_send_no_mpi()

```
void melissa_send_no_mpi (
            const int * time_step,
            const char * field_name,
            double * send_vect,
```

```
            const int * sobol_rank,
            const int * sample_id )
```

This function sends data to the stats library

Parameters

in	*time_step*	current time step of the simulation
in	∗*field_name*	name of the field to send to Melissa Server
in	∗*send_vect*	local data array to send to the statistic library
in	∗*sobol_rank*	Sobol indice rank in Sobol group
in	∗*sample_id*	ID of the parameter set defining the simulation

Chapter 5

Class Documentation

5.1 comm_data_s Struct Reference

`#include <melissa_data.h>`

Public Attributes

- int rank
- int comm_size
- int client_comm_size
- int * rcounts
- int * rdispls

5.1.1 Detailed Description

Structure to store communications parameters

5.1.2 Member Data Documentation

5.1.2.1 client_comm_size

`int comm_data_s::client_comm_size`

size of the clients communicators

5.1.2.2 comm_size

`int comm_data_s::comm_size`

size of the MPI communicator (1 if sequential)

5.1.2.3 rank

```
int comm_data_s::rank
```

rank of the MPI process (0 if sequential)

5.1.2.4 rcounts

```
int* comm_data_s::rcounts
```

counts for receiving datas

5.1.2.5 rdispls

```
int* comm_data_s::rdispls
```

displacements for receiving datas

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

- /home/tterraz/avido/melissa/Melissa/source/server/melissa_data.h

5.2 covariance_s Struct Reference

```
#include <covariance.h>
```

Collaboration diagram for covariance_s:

Public Attributes

- double * covariance
- mean_t mean1
- mean_t mean2
- int increment

5.2.1 Detailed Description

Structure containing an array of covariances and the corresponding mean structures

5.2.2 Member Data Documentation

5.2.2.1 covariance

```
double* covariance_s::covariance
```

covariance[vect_size]

5.2.2.2 increment

```
int covariance_s::increment
```

increment

5.2.2.3 mean1

```
mean_t covariance_s::mean1
```

corresponding mean

5.2.2.4 mean2

```
mean_t covariance_s::mean2
```

corresponding mean

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

- /home/tterraz/avido/melissa/Melissa/source/stats/covariance.h

5.3 mean_s Struct Reference

```
#include <mean.h>
```

Public Attributes

- double * mean
- int increment

5.3.1 Detailed Description

Structure containing an array of means, and the corresponding increment

5.3.2 Member Data Documentation

5.3.2.1 increment

`int mean_s::increment`

increment of this mean

5.3.2.2 mean

`double* mean_s::mean`

mean[vect_size]

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

- /home/tterraz/avido/melissa/Melissa/source/stats/mean.h

5.4 melissa_data_s Struct Reference

`#include <melissa_data.h>`

Collaboration diagram for melissa_data_s:

Public Attributes

- int vect_size
- melissa_options_t * options
- int is_valid
- mean_t * means
- variance_t * variances
- min_max_t * min_max
- int ** thresholds
- quantile_t * quantiles
- sobol_array_t * sobol_indices
- void(* init_sobol)(sobol_array_t *, int, int)
- void(* read_sobol)(sobol_array_t *, int, int, int, FILE *)
- void(* save_sobol)(sobol_array_t *, int, int, int, FILE *)
- void(* increment_sobol)(sobol_array_t *, int, double **, int)
- void(* free_sobol)(sobol_array_t *, int)
- int nb_simu
- int32_t ** step_simu

5.4.1 Detailed Description

Structure to store global parameters

5.4.2 Member Data Documentation

5.4.2.1 free_sobol

```
void(* melissa_data_s::free_sobol) (sobol_array_t *, int)
```

pointer to Sobol free function

5.4.2.2 increment_sobol

```
void(* melissa_data_s::increment_sobol) (sobol_array_t *, int, double **, int)
```

pointer to Sobol increment function

5.4.2.3 init_sobol

```
void(* melissa_data_s::init_sobol) (sobol_array_t *, int, int)
```

pointer to Sobol initialization function

5.4.2.4 is_valid

```
int melissa_data_s::is_valid
```

1 if the structure has been checked

5.4.2.5 means

```
mean_t* melissa_data_s::means
```

array of mean structures, size nb_time_steps

5.4.2.6 min_max

```
min_max_t* melissa_data_s::min_max
```

array of min and max structures, size nb_time_steps

5.4.2.7 nb_simu

```
int melissa_data_s::nb_simu
```

number of simulation that have sent a message

5.4.2.8 options

```
melissa_options_t* melissa_data_s::options
```

pointer to an option structure

5.4.2.9 quantiles

```
quantile_t* melissa_data_s::quantiles
```

array of quantile structures, size nb_time_steps

5.4.2.10 read_sobol

```
void(* melissa_data_s::read_sobol) (sobol_array_t *, int, int, int, FILE *)
```

pointer to Sobol read function

5.4.2.11 save_sobol

```
void(* melissa_data_s::save_sobol) (sobol_array_t *, int, int, int, FILE *)
```

pointer to Sobol save function

5.4.2.12 sobol_indices

```
sobol_array_t* melissa_data_s::sobol_indices
```

array of sobol array structures, size nb_time_steps

5.4.2.13 step_simu

```
int32_t** melissa_data_s::step_simu
```

arrays of bits, size nb_groups

5.4.2.14 thresholds

```
int** melissa_data_s::thresholds
```

array of threshold exceedance vectors, size nb_time_steps

5.4.2.15 variances

```
variance_t* melissa_data_s::variances
```

array of variance structures, size nb_time_steps

5.4.2.16 vect_size

```
int melissa_data_s::vect_size
```

local size of input vectors

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

- /home/tterraz/avido/melissa/Melissa/source/server/melissa_data.h

5.5 melissa_field_s Struct Reference

```
#include <melissa_fields.h>
```

Collaboration diagram for melissa_field_s:

Public Attributes

- char name [MAX_FIELD_NAME]
- melissa_data_t * stats_data

5.5.1 Detailed Description

< Structure for a linked list of output fields

Field structure

5.5.2 Member Data Documentation

5.5.2.1 name

```
char melissa_field_s::name[MAX_FIELD_NAME]
```

name of the field

5.5.2.2 stats_data

```
melissa_data_t* melissa_field_s::stats_data
```

stats_data structure

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

- /home/tterraz/avido/melissa/Melissa/source/server/melissa_fields.h

5.6 melissa_options_s Struct Reference

`#include <melissa_options.h>`

Public Attributes

- int nb_time_steps
- int nb_parameters
- int sampling_size
- int nb_simu
- int nb_fields
- int mean_op
- int variance_op
- int min_and_max_op
- int threshold_op
- double threshold
- int quantile_op
- int sobol_op
- int sobol_order
- int global_vect_size
- int restart
- char restart_dir [256]
- char launcher_name [256]

5.6.1 Detailed Description

Structure to store options parsed from command line

5.6.2 Member Data Documentation

5.6.2.1 global_vect_size

`int melissa_options_s::global_vect_size`

global size of input vector

5.6.2.2 launcher_name

`char melissa_options_s::launcher_name[256]`

Melissa master node name

5.6.2.3 mean_op

```
int melissa_options_s::mean_op
```

1 if the user needs to compute the means, 0 otherwise.

5.6.2.4 min_and_max_op

```
int melissa_options_s::min_and_max_op
```

1 if the user needs to compute min and max, 0 otherwise.

5.6.2.5 nb_fields

```
int melissa_options_s::nb_fields
```

nb of fields of the simulations

5.6.2.6 nb_parameters

```
int melissa_options_s::nb_parameters
```

nb of variables parameters of the study

5.6.2.7 nb_simu

```
int melissa_options_s::nb_simu
```

nb of simulation of the study

5.6.2.8 nb_time_steps

```
int melissa_options_s::nb_time_steps
```

number of time steps of the study

5.6.2.9 quantile_op

```
int melissa_options_s::quantile_op
```

1 if the user needs to compute quantiles, 0 otherwise

5.6.2.10 restart

```
int melissa_options_s::restart
```

1 if restart, 0 otherwise

5.6.2.11 restart_dir

```
char melissa_options_s::restart_dir[256]
```

Melissa restart files directory

5.6.2.12 sampling_size

```
int melissa_options_s::sampling_size
```

nb of randomly drawn simulation parameter sets

5.6.2.13 sobol_op

```
int melissa_options_s::sobol_op
```

1 if the user needs to compute sobol indices, 0 otherwise

5.6.2.14 sobol_order

```
int melissa_options_s::sobol_order
```

max order of the computes sobol indices

5.6.2.15 threshold

```
double melissa_options_s::threshold
```

threshold used to compute threshold exceedance

5.6.2.16 threshold_op

```
int melissa_options_s::threshold_op
```

1 if the user needs to compute threshold exceedance, 0 otherwise

5.6.2.17 variance_op

```
int melissa_options_s::variance_op
```

1 if the user needs to compute the variances, 0 otherwise.

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

- /home/tterraz/avido/melissa/Melissa/source/server/melissa_options.h

5.7 melissa_simulation_s Struct Reference

Public Attributes

- int **id**
- int **status**
- int **timeout**
- int **last_message**

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

- /home/tterraz/avido/melissa/Melissa/source/server/fault_tolerance.h

5.8 min_max_s Struct Reference

```
#include <min_max.h>
```

Public Attributes

- double * min
- double * max
- int is_init

5.8.1 Detailed Description

Structure containing two arrays of min and max values

5.8.2 Member Data Documentation

5.8.2.1 is_init

```
int min_max_s::is_init
```

0 before the first update, 1 otherwhise

5.8.2.2 max

```
double* min_max_s::max
```

max[vect_size]

5.8.2.3 min

```
double* min_max_s::min
```

min[vect_size]

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

- /home/tterraz/avido/melissa/Melissa/source/stats/min_max.h

5.9 pull_data_s Struct Reference

```
#include <server.h>
```

Public Attributes

- int * pull_rank
- int * push_rank
- int * message_sizes
- int total_nb_messages
- int local_nb_messages
- int buff_size

5.9.1 Detailed Description

< Helper structure for push pull socket

5.9.2 Member Data Documentation

5.9.2.1 buff_size

```
int pull_data_s::buff_size
```

recieve buffer size

5.9.2.2 local_nb_messages

```
int pull_data_s::local_nb_messages
```

local number of messages

5.9.2.3 message_sizes

```
int* pull_data_s::message_sizes
```

messages sizes, size total_nb_messages

5.9.2.4 pull_rank

```
int* pull_data_s::pull_rank
```

array of receiver ranks, size total_nb_messages

5.9.2.5 push_rank

```
int* pull_data_s::push_rank
```

array of sender ranks, size total_nb_messages

5.9.2.6 total_nb_messages

```
int pull_data_s::total_nb_messages
```

total number of messages

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

- /home/tterraz/avido/melissa/Melissa/source/server/server.h

5.10 quantile_s Struct Reference

```
#include <quantile.h>
```

Public Attributes

- double * quantile
- int increment
- double alpha

5.10.1 Detailed Description

Structure containing an array of quantiles, the corresponding increment, alpha (the quantile partition) and gamma parameters.

5.10.2 Member Data Documentation

5.10.2.1 alpha

```
double quantile_s::alpha
```

alpha

5.10.2.2 increment

```
int quantile_s::increment
```

increment of this quantile

5.10.2.3 quantile

```
double* quantile_s::quantile
```

quantile[vect_size]

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

- /home/tterraz/avido/melissa/Melissa/source/stats/quantile.h

5.11 sobol_array_s Struct Reference

```
#include <sobol.h>
```

Collaboration diagram for sobol_array_s:

Public Attributes

- sobol_jansen_t * sobol_jansen
- sobol_martinez_t * sobol_martinez
- variance_t variance_a
- variance_t variance_b
- int iteration

5.11.1 Detailed Description

Structure containing an array of sobol index structures

5.11.2 Member Data Documentation

5.11.2.1 iteration

```
int sobol_array_s::iteration
```

number of computed groups

5.11.2.2 sobol_jansen

```
sobol_jansen_t* sobol_array_s::sobol_jansen
```

array of sobol indices, size nb_parameters

5.11.2.3 sobol_martinez

```
sobol_martinez_t* sobol_array_s::sobol_martinez
```

array of sobol indices, size nb_parameters

5.11.2.4 variance_a

```
variance_t sobol_array_s::variance_a
```

first set variance needed by Martinez formula

5.11.2.5 variance_b

```
variance_t sobol_array_s::variance_b
```

second set variance needed by Martinez formula

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

- /home/tterraz/avido/melissa/Melissa/source/stats/sobol.h

5.12 sobol_jansen_s Struct Reference

```
#include <sobol.h>
```

Public Attributes

- double * summ_a
- double * summ_b
- double * first_order_values
- double * total_order_values

5.12.1 Detailed Description

Structure containing a sobol indices vector with all structures needed by Jansen update formula

5.12.2 Member Data Documentation

5.12.2.1 first_order_values

```
double* sobol_jansen_s::first_order_values
```

values of the sobol indices

5.12.2.2 summ_a

```
double* sobol_jansen_s::summ_a
```

summ needed by Jansen formula

5.12.2.3 summ_b

```
double* sobol_jansen_s::summ_b
```

summ needed by Jansen formula

5.12.2.4 total_order_values

```
double* sobol_jansen_s::total_order_values
```

values of the sobol indices

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

- /home/tterraz/avido/melissa/Melissa/source/stats/sobol.h

5.13 sobol_martinez_s Struct Reference

`#include <sobol.h>`

Collaboration diagram for sobol_martinez_s:

Public Attributes

- covariance_t first_order_covariance
- covariance_t total_order_covariance
- variance_t variance_k
- double * first_order_values
- double * total_order_values
- double confidence_interval [2]

5.13.1 Detailed Description

Structure containing a sobol indices vector with all structures needed by Martinez update formula

5.13.2 Member Data Documentation

5.13.2.1 confidence_interval

```
double sobol_martinez_s::confidence_interval[2]
```

interval for 95% confidence level

5.13.2.2 first_order_covariance

```
covariance_t sobol_martinez_s::first_order_covariance
```

covariance needed by Martinez formula

5.13.2.3 first_order_values

```
double* sobol_martinez_s::first_order_values
```

values of the sobol indices

5.13.2.4 total_order_covariance

```
covariance_t sobol_martinez_s::total_order_covariance
```

covariance needed by Martinez formula

5.13.2.5 total_order_values

```
double* sobol_martinez_s::total_order_values
```

values of the sobol indices

5.13.2.6 variance_k

```
variance_t sobol_martinez_s::variance_k
```

variance needed by Martinez formula

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

- /home/tterraz/avido/melissa/Melissa/source/stats/sobol.h

5.14 variance_s Struct Reference

```
#include <variance.h>
```

Collaboration diagram for variance_s:

Public Attributes

- double * variance
- mean_t mean_structure

5.14.1 Detailed Description

Structure containing an array of variances and the corresponding mean structure

5.14.2 Member Data Documentation

5.14.2.1 mean_structure

```
mean_t variance_s::mean_structure
```

corresponding mean

5.14.2.2 variance

```
double* variance_s::variance
```

variance[vect_size]

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

- /home/tterraz/avido/melissa/Melissa/source/stats/variance.h

5.15 zmq_data_s Struct Reference

Public Attributes

- void * context
- void * connexion_requester
- void * init_requester
- void ** data_pusher
- void ** sobol_requester
- int rinit_tab [3]
- int sobol
- int sobol_rank
- int sinit_tab [2]
- int nb_proc_server
- int nb_parameters
- int * server_vect_size
- char * buffer
- int buff_size
- int send_buff_size
- double * buffer_sobol
- int * send_counts
- int * local_vect_sizes
- int * sdispls
- int * pull_rank
- int * push_rank
- int * message_sizes
- int total_nb_messages
- int local_nb_messages
- int coupling
- MPI_Comm comm_sobol

5.15.1 Detailed Description

Structure containing some data needed by zmq

5.15.2 Member Data Documentation

5.15.2.1 buff_size

```
int zmq_data_s::buff_size
```

size of this buffer

5.15.2.2 buffer

```
char* zmq_data_s::buffer
```

buffer used to send data to the library

5.15.2.3 buffer_sobol

```
double* zmq_data_s::buffer_sobol
```

buffer used to store data on sobol rank 0

5.15.2.4 comm_sobol

```
MPI_Comm zmq_data_s::comm_sobol
```

inter-groups communicator

5.15.2.5 connexion_requester

```
void* zmq_data_s::connexion_requester
```

connexion ZeroMQ port

5.15.2.6 context

```
void* zmq_data_s::context
```

ZeroMQ context

5.15.2.7 coupling

```
int zmq_data_s::coupling
```

coupled simulations or not

5.15.2.8 data_pusher

```
void** zmq_data_s::data_pusher
```

push data ZeroMQ ports

5.15.2.9 init_requester

```
void* zmq_data_s::init_requester
```

initialization ZeroMQ port

5.15.2.10 local_nb_messages

```
int zmq_data_s::local_nb_messages
```

local number of messages

5.15.2.11 local_vect_sizes

```
int* zmq_data_s::local_vect_sizes
```

local vector size

5.15.2.12 message_sizes

```
int* zmq_data_s::message_sizes
```

size of the message i

5.15.2.13 nb_parameters

```
int zmq_data_s::nb_parameters
```

number of parameters of the study

5.15.2.14 nb_proc_server

```
int zmq_data_s::nb_proc_server
```

number of MPI processes of the library

5.15.2.15 pull_rank

```
int* zmq_data_s::pull_rank
```

rank of the pulling process for the message i

5.15.2.16 push_rank

```
int* zmq_data_s::push_rank
```

rank of the pushing process for the message i

5.15.2.17 rinit_tab

```
int zmq_data_s::rinit_tab[3]
```

array used to receive data

5.15.2.18 sdispls

```
int* zmq_data_s::sdispls
```

displacement to which data should be sent to server rank i

5.15.2.19 send_buff_size

```
int zmq_data_s::send_buff_size
```

size of send buffer

5.15.2.20 send_counts

```
int* zmq_data_s::send_counts
```

number of elements to send to server rank i

5.15.2.21 server_vect_size

```
int* zmq_data_s::server_vect_size
```

local vect size for the library

5.15.2.22 sinit_tab

```
int zmq_data_s::sinit_tab[2]
```

array used to send data

5.15.2.23 sobol

```
int zmq_data_s::sobol
```

1 if sobol computation, 0 otherwhise

5.15.2.24 sobol_rank

```
int zmq_data_s::sobol_rank
```

sobol rank

5.15.2.25 sobol_requester

```
void** zmq_data_s::sobol_requester
```

data ZeroMQ Sobol port

5.15.2.26 total_nb_messages

```
int zmq_data_s::total_nb_messages
```

total number of messages

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

- /home/tterraz/avido/melissa/Melissa/source/api/melissa_api.c

Chapter 6

File Documentation

6.1 /home/tterraz/avido/melissa/Melissa/source/api/melissa_api.c File Reference

API Functions.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <zmq.h>
#include "melissa_api_no_mpi.h"
#include "melissa_utils.h"
```

Include dependency graph for melissa_api.c:

6.2 /home/tterraz/avido/melissa/Melissa/source/api/melissa_api.h File Reference

6.2.1 Detailed Description

Author

Terraz Théophile

Date

2016-09-05

6.3 /home/tterraz/avido/melissa/Melissa/source/api/melissa_api_no_mpi.h File Reference

This graph shows which files directly or indirectly include this file:

Functions

- void melissa_init_no_mpi (const int *vect_size, const int *sobol_rank, const int *sample_id)
- void melissa_send_no_mpi (const int *time_step, const char *field_name, double *send_vect, const int *sobol_rank, const int *sample_id)
- void melissa_finalize ()

6.3.1 Detailed Description

Author

Terraz Théophile

Date

2016-09-05

6.4 /home/tterraz/avido/melissa/Melissa/source/server/compute_stats.c File Reference

Functions called by the server.

```
#include <stdio.h>
#include <stdlib.h>
#include "melissa_data.h"
#include "melissa_utils.h"
```

Include dependency graph for compute_stats.c:

Functions

- void compute_stats (melissa_data_t *data, const int time_step, const int nb_vect, double **in_vect_tab, const int group_id)
- void finalize_stats (melissa_data_t *data)

6.4.1 Detailed Description

Functions called by the server.

Author

Terraz Théophile

Date

2016-15-03

6.5 /home/tterraz/avido/melissa/Melissa/source/server/compute_stats.h File Reference

```
#include "melissa_data.h"
```

Include dependency graph for compute_stats.h: This graph shows which files directly or indirectly include this file:

Functions

- void compute_stats (melissa_data_t *data, const int time_step, const int nb_vect, double **in_vect_tab, const int group_id)
- void finalize_stats (melissa_data_t *data)

6.5.1 Detailed Description

Author

Terraz Théophile

Date

2017-15-01

6.6 /home/tterraz/avido/melissa/Melissa/source/server/fault_tolerance.c File Reference

```
#include <string.h>
#include "fault_tolerance.h"
#include "zmq.h"
```

Include dependency graph for fault_tolerance.c:

Functions

- melissa_simulation_t * **add_simulation** (int id, int nb_time_steps)
- void **simu_push_to** (vector_t *v, int pos, int nb_time_steps)
- void **free_simu_vector** (vector_t v)
- int **check_timeouts** (int *simu_state, int *simu_timeouts, double *last_message_simu, int nb_simu)
- void **send_timeouts** (int detected_timeouts, int *simu_timeouts, int nb_simu, char *txt_buffer, void *python_pusher)

6.6.1 Detailed Description

Author

Terraz Théophile

Date

2017-30-06

6.7 /home/tterraz/avido/melissa/Melissa/source/server/fault_tolerance.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <melissa_utils.h>
#include <vector.h>
```

Include dependency graph for fault_tolerance.h: This graph shows which files directly or indirectly include this file:

Classes

- struct melissa_simulation_s

Typedefs

- typedef struct field_status_s **field_status_t**
- typedef struct melissa_simulation_s **melissa_simulation_t**

Functions

- melissa_simulation_t * **add_simulation** (int id, int nb_time_steps)
- void **simu_push_to** (vector_t *v, int pos, int nb_time_steps)
- void **free_simu_vector** (vector_t v)
- int **check_timeouts** (int *simu_state, int *simu_timeouts, double *last_message_simu, int nb_simu)
- void **send_timeouts** (int detected_timeouts, int *simu_timeouts, int nb_simu, char *txt_buffer, void *python_requester)

6.7.1 Detailed Description

Author

Terraz Théophile

Date

2017-30-06

6.8 /home/tterraz/avido/melissa/Melissa/source/server/melissa_data.c File Reference

Routines related to the melissa_data structure.

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "melissa_data.h"
#include "melissa_utils.h"
#include "sobol.h"
```

Include dependency graph for melissa_data.c:

Functions

- void melissa_init_data (melissa_data_t *data, melissa_options_t *options, int vect_size)
- void melissa_check_data (melissa_data_t *data)
- void melissa_free_data (melissa_data_t *data)
- long int mem_conso (melissa_options_t *options)

6.8.1 Detailed Description

Routines related to the melissa_data structure.

Author

Terraz Théophile

Date

2016-24-05

6.8.2 Function Documentation

6.8.2.1 melissa_check_data()

```
void melissa_check_data (
            melissa_data_t * data )
```

This function checks the data structure. It terminates the program if one of the mandatory option is missing or invalid, and correct mistakes for other options

Parameters

in,out	*data	pointer to the structure containing global parameters

6.8.2.2 mem_conso()

```
long int mem_conso (
            melissa_options_t * options )
```

This function computes and displays the memory consumption of the library

Parameters

in	*options	pointer to the structure containing global options

6.9 /home/tterraz/avido/melissa/Melissa/source/server/melissa_data.h File Reference

```
#include <stdint.h>
#include "melissa_utils.h"
```

```
#include "melissa_options.h"
#include "mean.h"
#include "variance.h"
#include "min_max.h"
#include "threshold.h"
#include "quantile.h"
#include "covariance.h"
#include "sobol.h"
#include "vector.h"
```

Include dependency graph for melissa_data.h: This graph shows which files directly or indirectly include this file:

Classes

- struct comm_data_s
- struct melissa_data_s

Typedefs

- typedef struct comm_data_s comm_data_t
- typedef struct melissa_data_s melissa_data_t

Functions

- void melissa_init_data (melissa_data_t *data, melissa_options_t *options, int vect_size)
- void melissa_check_data (melissa_data_t *data)
- void melissa_free_data (melissa_data_t *data)
- long int mem_conso (melissa_options_t *options)

6.9.1 Detailed Description

Author

Terraz Théophile

Date

2017-15-01

6.9.2 Typedef Documentation

6.9.2.1 comm_data_t

```
typedef struct comm_data_s comm_data_t
```

type corresponding to comm_data_s

6.9.2.2 melissa_data_t

```
typedef struct melissa_data_s melissa_data_t
```

type corresponding to melissa_data_s

6.9.3 Function Documentation

6.9.3.1 melissa_check_data()

```
void melissa_check_data (
            melissa_data_t * data )
```

This function checks the data structure. It terminates the program if one of the mandatory option is missing or invalid, and correct mistakes for other options

Parameters

in,out	*data	pointer to the structure containing global parameters

6.9.3.2 mem_conso()

```
long int mem_conso (
            melissa_options_t * options )
```

This function computes and displays the memory consumption of the library

Parameters

in	*options	pointer to the structure containing global options

6.10 /home/tterraz/avido/melissa/Melissa/source/server/melissa_fields.c File Reference

Routines related to the melissa_fields structure.

```
#include <getopt.h>
#include <string.h>
#include "melissa_fields.h"
#include "melissa_data.h"
#include "melissa_io.h"
#include "compute_stats.h"
```

Include dependency graph for melissa_fields.c:

Functions

- void **melissa_get_fields** (int argc, char **argv, melissa_field_t *fields, int nb_fields)
- void **add_fields** (melissa_field_t *field, int data_size, int nb_fields)
- melissa_data_t * **get_data_ptr** (melissa_field_t *field, int nb_fields, char *field_name)
- void **finalize_field_data** (melissa_field_t *field, comm_data_t *comm_data, melissa_options_t *options, int *local_vect_sizes)

6.10.1 Detailed Description

Routines related to the melissa_fields structure.

Author

Terraz Théophile

Date

2017-01-09

6.11 /home/tterraz/avido/melissa/Melissa/source/server/melissa_fields.h File Reference

```
#include "melissa_data.h"
```

Include dependency graph for melissa_fields.h: This graph shows which files directly or indirectly include this file:

Classes

- struct melissa_field_s

Typedefs

- typedef struct melissa_field_s melissa_field_t

Functions

- void **melissa_get_fields** (int argc, char **argv, melissa_field_t *fields, int nb_fields)
- void **add_fields** (melissa_field_t *field, int data_size, int nb_fields)
- melissa_data_t * **get_data_ptr** (melissa_field_t *field, int nb_fields, char *field_name)
- void **finalize_field_data** (melissa_field_t *field, comm_data_t *comm_data, melissa_options_t *options, int *local_vect_sizes)

6.11.1 Detailed Description

Author

Terraz Théophile

Date

2017-15-01

6.11.2 Typedef Documentation

6.11.2.1 melissa_field_t

```
typedef struct melissa_field_s melissa_field_t
```

type corresponding to field_s

6.12 /home/tterraz/avido/melissa/Melissa/source/server/melissa_io.c File Reference

Inputs, outputs and checkpoints.

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <stdint.h>
#include <math.h>
#include "melissa_data.h"
#include "melissa_utils.h"
```

Include dependency graph for melissa_io.c:

Functions

- void write_client_data (int *client_comm_size, int *client_vect_sizes)
- int read_client_data (int *client_comm_size, int **client_vect_sizes, melissa_options_t *options)
- void save_stats (melissa_data_t *data, comm_data_t *comm_data, char *field_name)
- void read_saved_stats (melissa_data_t *data, comm_data_t *comm_data, char *field_name, int client_rank)
- void save_simu_states (int *simu_states, comm_data_t *comm_data, int size)
- void read_simu_states (int *simu_states, melissa_options_t *options, comm_data_t *comm_data, int size)
- void write_stats_bin (melissa_data_t **data, melissa_options_t *options, comm_data_t *comm_data, int *local_vect_sizes, char *field)
- void write_stats_ensight (melissa_data_t **data, melissa_options_t *options, comm_data_t *comm_data, int *local_vect_sizes, char *field)
- void read_ensight (melissa_options_t *options, comm_data_t *comm_data, double *in_vect, int *local_vect_sizes, char *file_name)
- void write_stats_txt (melissa_data_t **data, melissa_options_t *options, comm_data_t *comm_data, int *local_vect_sizes, char *field)

6.12.1 Detailed Description

Inputs, outputs and checkpoints.

Author

Terraz Théophile

Date

2017-19-01

6.13 /home/tterraz/avido/melissa/Melissa/source/server/melissa_io.h File Reference

```
#include "melissa_data.h"
#include "melissa_options.h"
```

Include dependency graph for melissa_io.h: This graph shows which files directly or indirectly include this file:

Functions

- void write_stats_bin (melissa_data_t **data, melissa_options_t *options, comm_data_t *comm_data, int *local_vect_sizes, char *field)
- void write_stats_txt (melissa_data_t **data, melissa_options_t *options, comm_data_t *comm_data, int *local_vect_sizes, char *field)
- void write_stats_ensight (melissa_data_t **data, melissa_options_t *options, comm_data_t *comm_data, int *local_vect_sizes, char *field)
- void write_client_data (int *client_comm_size, int *client_vect_sizes)
- int read_client_data (int *client_comm_size, int **client_vect_sizes, melissa_options_t *options)
- void save_stats (melissa_data_t *data, comm_data_t *comm_data, char *field_name)
- void read_saved_stats (melissa_data_t *data, comm_data_t *comm_data, char *field_name, int client_rank)
- void save_simu_states (int *simu_states, comm_data_t *comm_data, int size)
- void read_simu_states (int *simu_states, melissa_options_t *options, comm_data_t *comm_data, int size)
- void read_ensight (melissa_options_t *options, comm_data_t *comm_data, double *in_vect, int *local_vect_sizes, char *file_name)

6.13.1 Detailed Description

Author

Terraz Théophile

Date

2017-15-01

6.14 /home/tterraz/avido/melissa/Melissa/source/server/melissa_options.c File Reference

Parse commande line to get stats options.

```
#include <unistd.h>
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include <getopt.h>
#include "melissa_options.h"
#include "melissa_data.h"
#include "melissa_utils.h"
```

Include dependency graph for melissa_options.c:

Functions

- void **get_fields** (char ∗name, melissa_options_t ∗options)
- void melissa_print_options (melissa_options_t ∗options)
- void melissa_get_options (int argc, char ∗∗argv, melissa_options_t ∗options)
- void melissa_check_options (melissa_options_t ∗options)
- void melissa_write_options (melissa_options_t ∗options)
- int melissa_read_options (melissa_options_t ∗options)

6.14.1 Detailed Description

Parse commande line to get stats options.

Author

Terraz Théophile

Date

2016-03-03

6.15 /home/tterraz/avido/melissa/Melissa/source/server/melissa_options.h File Reference

This graph shows which files directly or indirectly include this file:

Classes

- struct melissa_options_s

Typedefs

- typedef struct melissa_options_s melissa_options_t

Functions

- void melissa_get_options (int argc, char ∗∗argv, melissa_options_t ∗options)
- void melissa_check_options (melissa_options_t ∗options)
- void melissa_print_options (melissa_options_t ∗options)
- void melissa_write_options (melissa_options_t ∗options)
- int melissa_read_options (melissa_options_t ∗options)

6.15.1 Detailed Description

Author

Terraz Théophile

Date

2017-15-01

6.15.2 Typedef Documentation

6.15.2.1 melissa_options_t

```
typedef struct melissa_options_s melissa_options_t
```

type corresponding to melissa_options_s

6.16 /home/tterraz/avido/melissa/Melissa/source/server/server.h File Reference

```
#include "melissa_fields.h"
#include "melissa_data.h"
#include "melissa_utils.h"
```

Include dependency graph for server.h:

Classes

- struct pull_data_s

Typedefs

- typedef struct pull_data_s pull_data_t

Functions

- void **comm_n_to_m_init** (int *rcounts, int *rdispls, const int global_vect_size, const int *server_vect_size, int *client_vect_size, const int nb_proc_client, const int rank, pull_data_t *pull_data)
- void **increment_step_simu** (melissa_field_t *field, char *field_name, int group_id)
- int **check_simu_state** (melissa_field_t *field, int nb_fields, int group_id, int nb_time_steps, comm_data_t *comm_data)
- long int **count_mbytes_written** (melissa_options_t *options)
- int **string_recv** (void *socket, char *recv_buff)
- void global_confidence_sobol_martinez (melissa_field_t *field, int nb_fields, comm_data_t *comm_data, double *interval1, double *interval_tot)

6.16.1 Detailed Description

Author

Terraz Théophile

Date

2016-15-03

6.16.2 Typedef Documentation

6.16.2.1 pull_data_t

```
typedef struct pull_data_s pull_data_t
```

type corresponding to pull_data_s

6.16.3 Function Documentation

6.16.3.1 global_confidence_sobol_martinez()

```
void global_confidence_sobol_martinez (
            melissa_field_t * field,
            int nb_fields,
            comm_data_t * comm_data,
            double * interval1,
            double * interval_tot )
```

This function computes the confidence interval for Martinez Sobol indices

Parameters

in	*field*	array of field structures
in	*nb_fields*	size of field array
out	∗*comm_data*	comm data structure
out	∗*interval1*	worst confidence interval (first order)
out	∗*interval_tot*	worst confidence interval (total order)

6.17 /home/tterraz/avido/melissa/Melissa/source/stats/covariance.c File Reference

Functions needed to compute covariances.

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "mean.h"
#include "variance.h"
#include "covariance.h"
#include "melissa_utils.h"
```

Include dependency graph for covariance.c:

Functions

- void init_covariance (covariance_t *covariance, const int vect_size)
- void increment_covariance (covariance_t *covariance, double in_vect1[ ], double in_vect2[ ], const int vect_size)
- void update_covariance (covariance_t *covariance1, covariance_t *covariance2, covariance_t *updated_covariance, const int vect_size)
- void save_covariance (covariance_t *covars, int vect_size, int nb_time_steps, FILE *f)
- void read_covariance (covariance_t *covars, int vect_size, int nb_time_steps, FILE *f)
- void free_covariance (covariance_t *covariance)

6.17.1 Detailed Description

Functions needed to compute covariances.

Author

Terraz Théophile

Date

2016-01-07

6.17.2 Function Documentation

6.17.2.1 free_covariance()

```
void free_covariance (
            covariance_t * covariance )
```

This function frees a covariance structure.

Parameters

in	*∗covariance*	the covariance structure to free

6.17.2.2 increment_covariance()

```
void increment_covariance (
            covariance_t * covariance,
            double in_vect1[],
            double in_vect2[],
            const int vect_size )
```

This function updates the incremental covariance.

Parameters

in,out	*∗covariance*	input: previously computed covariance, output: incremented covariance
in	*in_vect1[]*	first input vector of double values
in	*in_vect2[]*	second input vector of double values
in	*vect_size*	size of the input vectors

6.17.2.3 init_covariance()

```
void init_covariance (
            covariance_t * covariance,
            const int vect_size )
```

This function initializes a covariance structure.

Parameters

in,out	*∗covariance*	the covariance structure to initialize
in	*vect_size*	size of the covariance vector

6.17.2.4 read_covariance()

```
void read_covariance (
            covariance_t * covars,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function reads an array of covariances structures on disc

Parameters

in	*∗covars*	covariance structures to read, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.17.2.5 save_covariance()

```
void save_covariance (
            covariance_t * covars,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function writes an array of covariances structures on disc

Parameters

in	*∗covars*	covariance structures to save, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.17.2.6 update_covariance()

```
void update_covariance (
            covariance_t * covariance1,
            covariance_t * covariance2,
            covariance_t * updated_covariance,
            const int vect_size )
```

This function updates the incremental covariance.

Parameters

in	*∗covariance1*	first input partial covariance
in	*∗covariance2*	second input partial covariance
out	*∗updated_covariance*	updated covariance
in	*vect_size*	size of the input vectors

6.18 /home/tterraz/avido/melissa/Melissa/source/stats/covariance.h File Reference

```
#include "mean.h"
```

Include dependency graph for covariance.h: This graph shows which files directly or indirectly include this file:

Classes

- struct covariance_s

Typedefs

- typedef struct covariance_s covariance_t

Functions

- void init_covariance (covariance_t *covariance, const int vect_size)
- void increment_covariance (covariance_t *partial_covariance, double in_vect1[ ], double in_vect2[ ], const int vect_size)
- void update_covariance (covariance_t *covariance1, covariance_t *covariance2, covariance_t *updated_↩ covariance, const int vect_size)
- void save_covariance (covariance_t *covars, int vect_size, int nb_time_steps, FILE *f)
- void read_covariance (covariance_t *covars, int vect_size, int nb_time_steps, FILE *f)
- void free_covariance (covariance_t *covariance)

6.18.1 Detailed Description

Author

Terraz Théophile

Date

2017-15-01

6.18.2 Typedef Documentation

6.18.2.1 covariance_t

```
typedef struct covariance_s covariance_t
```

type corresponding to covariance_s

6.18.3 Function Documentation

6.18.3.1 free_covariance()

```
void free_covariance (
            covariance_t * covariance )
```

This function frees a covariance structure.

Parameters

in	*∗covariance*	the covariance structure to free

6.18.3.2 increment_covariance()

```
void increment_covariance (
            covariance_t * covariance,
            double in_vect1[],
            double in_vect2[],
            const int vect_size )
```

This function updates the incremental covariance.

Parameters

in,out	*∗covariance*	input: previously computed covariance, output: incremented covariance
in	*in_vect1[]*	first input vector of double values
in	*in_vect2[]*	second input vector of double values
in	*vect_size*	size of the input vectors

6.18.3.3 init_covariance()

```
void init_covariance (
            covariance_t * covariance,
            const int vect_size )
```

This function initializes a covariance structure.

Parameters

in,out	*∗covariance*	the covariance structure to initialize
in	*vect_size*	size of the covariance vector

6.18.3.4 read_covariance()

```
void read_covariance (
            covariance_t * covars,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function reads an array of covariances structures on disc

Parameters

in	*∗covars*	covariance structures to read, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.18.3.5 save_covariance()

```
void save_covariance (
            covariance_t * covars,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function writes an array of covariances structures on disc

Parameters

in	*∗covars*	covariance structures to save, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.18.3.6 update_covariance()

```
void update_covariance (
            covariance_t * covariance1,
            covariance_t * covariance2,
            covariance_t * updated_covariance,
            const int vect_size )
```

This function updates the incremental covariance.

Parameters

in	*∗covariance1*	first input partial covariance
in	*∗covariance2*	second input partial covariance
out	*∗updated_covariance*	updated covariance
in	*vect_size*	size of the input vectors

6.19 /home/tterraz/avido/melissa/Melissa/source/stats/mean.c File Reference

Mean related functions.

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <math.h>
#include "mean.h"
#include "melissa_utils.h"
```

Include dependency graph for mean.c:

Functions

- void init_mean (mean_t *mean, const int vect_size)
- void increment_mean (mean_t *mean, double in_vect[ ], const int vect_size)
- void update_mean (mean_t *mean1, mean_t *mean2, mean_t *updated_mean, const int vect_size)
- void save_mean (mean_t *means, int vect_size, int nb_time_steps, FILE *f)
- void read_mean (mean_t *means, int vect_size, int nb_time_steps, FILE *f)
- void free_mean (mean_t *mean)

6.19.1 Detailed Description

Mean related functions.

Author

Terraz Théophile

Date

2016-15-02

6.19.2 Function Documentation

6.19.2.1 free_mean()

```
void free_mean (
            mean_t * mean )
```

This function frees a mean structure.

Parameters

in	*mean	the mean structure to free

6.19.2.2 increment_mean()

```
void increment_mean (
            mean_t * mean,
            double in_vect[],
            const int vect_size )
```

This function updates the incremental mean.

Parameters

in,out	*mean	input: previously computed iterative mean, output: updated mean
in	*in_vect[]*	input vector of double values
in	*vect_size*	size of the input vectors

6.19.2.3 init_mean()

```
void init_mean (
            mean_t * mean,
            const int vect_size )
```

This function initializes a mean structure.

Parameters

in,out	*mean	the mean structure to initialize
in	*vect_size*	size of the mean vector

6.19.2.4 read_mean()

```
void read_mean (
            mean_t * means,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function reads an array of mean structures on disc

Parameters

in	*means	mean structures to read, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.19.2.5 save_mean()

```
void save_mean (
            mean_t * means,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function writes an array of mean structures on disc

Parameters

in	*means	mean structures to save, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.19.2.6 update_mean()

```
void update_mean (
            mean_t * mean1,
            mean_t * mean2,
            mean_t * updated_mean,
            const int vect_size )
```

This function agregates two partial means.

Parameters

in	*mean1	first input vector of partial means
in	*mean2	second input vector of partial means
out	*updated_mean	the updated mean
in	*vect_size*	size of the input and output vectors

6.20 /home/tterraz/avido/melissa/Melissa/source/stats/mean.h File Reference

`#include <stdio.h>`
Include dependency graph for mean.h: This graph shows which files directly or indirectly include this file:

Classes

- struct mean_s

Typedefs

- typedef struct mean_s mean_t

Functions

- void init_mean (mean_t *mean, const int vect_size)
- void increment_mean (mean_t *mean, double in_vect[ ], const int vect_size)
- void update_mean (mean_t *mean1, mean_t *mean2, mean_t *updated_mean, const int vect_size)
- void save_mean (mean_t *means, int vect_size, int nb_time_steps, FILE *f)
- void read_mean (mean_t *means, int vect_size, int nb_time_steps, FILE *f)
- void free_mean (mean_t *mean)

6.20.1 Detailed Description

Author

Terraz Théophile

Date

2017-15-01

6.20.2 Typedef Documentation

6.20.2.1 mean_t

```
typedef struct mean_s mean_t
```

type corresponding to mean_s

6.20.3 Function Documentation

6.20.3.1 free_mean()

```
void free_mean (
            mean_t * mean )
```

This function frees a mean structure.

Parameters

in	*mean	the mean structure to free

6.20.3.2 increment_mean()

```
void increment_mean (
            mean_t * mean,
            double in_vect[],
            const int vect_size )
```

This function updates the incremental mean.

Parameters

in,out	*mean	input: previously computed iterative mean, output: updated mean
in	*in_vect[]*	input vector of double values
in	*vect_size*	size of the input vectors

6.20.3.3 init_mean()

```
void init_mean (
            mean_t * mean,
            const int vect_size )
```

This function initializes a mean structure.

Parameters

in,out	*mean	the mean structure to initialize
in	*vect_size*	size of the mean vector

6.20.3.4 read_mean()

```
void read_mean (
            mean_t * means,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function reads an array of mean structures on disc

Parameters

in	*∗means*	mean structures to read, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.20.3.5 save_mean()

```
void save_mean (
            mean_t * means,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function writes an array of mean structures on disc

Parameters

in	*∗means*	mean structures to save, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.20.3.6 update_mean()

```
void update_mean (
            mean_t * mean1,
            mean_t * mean2,
            mean_t * updated_mean,
            const int vect_size )
```

This function agregates two partial means.

Parameters

in	*∗mean1*	first input vector of partial means
in	*∗mean2*	second input vector of partial means
out	*∗updated_mean*	the updated mean
in	*vect_size*	size of the input and output vectors

6.21 /home/tterraz/avido/melissa/Melissa/source/stats/min_max.c File Reference

Min and max related functions.

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "min_max.h"
#include "melissa_utils.h"
```

Include dependency graph for min_max.c:

Functions

- void init_min_max (min_max_t *min_max, const int vect_size)
- void min_and_max (min_max_t *min_max, double in_vect[ ], const int vect_size)
- void save_min_max (min_max_t *minmax, int vect_size, int nb_time_steps, FILE *f)
- void read_min_max (min_max_t *minmax, int vect_size, int nb_time_steps, FILE *f)
- void free_min_max (min_max_t *min_max)

6.21.1 Detailed Description

Min and max related functions.

Author

Terraz Théophile

Date

2016-15-02

6.21.2 Function Documentation

6.21.2.1 free_min_max()

```
void free_min_max (
            min_max_t * min_max )
```

This function frees a min and max structure.

Parameters

in	*min_max	the min and max structure to free

6.21.2.2 init_min_max()

```
void init_min_max (
            min_max_t * min_max,
            const int vect_size )
```

This function initializes a min and max structure.

Parameters

in,out	*min_max	the min and max structure to initialize
in	*vect_size*	size of the vectors

6.21.2.3 min_and_max()

```
void min_and_max (
            min_max_t * min_max,
            double in_vect[],
            const int vect_size )
```

This function updates the min and the max values of min and max vectors using the input vector.

Parameters

in,out	*min_max	the min and max structure
in	*in_vect[]*	input vector of double values
in	*vect_size*	size of the input vectors

6.21.2.4 read_min_max()

```
void read_min_max (
            min_max_t * minmax,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function reads an array of min and max structures on disc

Parameters

in	*minmax	min and max structures to read, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.21.2.5 save_min_max()

```
void save_min_max (
            min_max_t * minmax,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function writes an array of min and max structures on disc

Parameters

in	*minmax	min and max structures to save, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.22 /home/tterraz/avido/melissa/Melissa/source/stats/min_max.h File Reference

This graph shows which files directly or indirectly include this file:

Classes

- struct min_max_s

Typedefs

- typedef struct min_max_s min_max_t

Functions

- void init_min_max (min_max_t *min_max, const int vect_size)
- void min_and_max (min_max_t *min_max, double in_vect[ ], const int vect_size)
- void save_min_max (min_max_t *minmax, int vect_size, int nb_time_steps, FILE *f)
- void read_min_max (min_max_t *minmax, int vect_size, int nb_time_steps, FILE *f)
- void free_min_max (min_max_t *min_max)

6.22.1 Detailed Description

Author

Terraz Théophile

Date

2017-15-01

6.22.2 Typedef Documentation

6.22.2.1 min_max_t

```
typedef struct min_max_s min_max_t
```

type corresponding to min_max_s

6.22.3 Function Documentation

6.22.3.1 free_min_max()

```
void free_min_max (
            min_max_t * min_max )
```

This function frees a min and max structure.

Parameters

in	*min_max	the min and max structure to free

6.22.3.2 init_min_max()

```
void init_min_max (
            min_max_t * min_max,
            const int vect_size )
```

This function initializes a min and max structure.

Parameters

in,out	*min_max	the min and max structure to initialize
in	vect_size	size of the vectors

6.22.3.3 min_and_max()

```
void min_and_max (
            min_max_t * min_max,
```

```
            double in_vect[],
            const int vect_size )
```

This function updates the min and the max values of min and max vectors using the input vector.

Parameters

in,out	*min_max	the min and max structure
in	*in_vect[]*	input vector of double values
in	*vect_size*	size of the input vectors

6.22.3.4 read_min_max()

```
void read_min_max (
            min_max_t * minmax,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function reads an array of min and max structures on disc

Parameters

in	*minmax	min and max structures to read, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.22.3.5 save_min_max()

```
void save_min_max (
            min_max_t * minmax,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function writes an array of min and max structures on disc

Parameters

in	*minmax	min and max structures to save, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.23 /home/tterraz/avido/melissa/Melissa/source/stats/quantile.c File Reference

Quantile related functions.

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <math.h>
#include "quantile.h"
#include "melissa_utils.h"
```

Include dependency graph for quantile.c:

Functions

- void init_quantile (quantile_t *quantile, const int vect_size, const double alpha)
- void increment_quantile (quantile_t *quantile, const int nmax, double in_vect[ ], const int vect_size)
- void save_quantile (quantile_t *quantiles, int vect_size, int nb_time_steps, FILE *f)
- void read_quantile (quantile_t *quantiles, int vect_size, int nb_time_steps, FILE *f)
- void free_quantile (quantile_t *quantile)

6.23.1 Detailed Description

Quantile related functions.

Author

Terraz Théophile

Date

2017-18-05

6.23.2 Function Documentation

6.23.2.1 free_quantile()

```
void free_quantile (
            quantile_t * quantile )
```

This function frees a quantile structure.

Parameters

in,out	*quantile	the quantile structure to free

6.23.2.2 increment_quantile()

```
void increment_quantile (
            quantile_t * quantile,
            const int nmax,
            double in_vect[],
            const int vect_size )
```

This function updates the incremental quantile.

Parameters

in,out	*quantile	input: previously computed iterative quantile, output: updated partial quantile
in	*nmax*	maximum number of iterations
in	*in_vect[]*	input vector of double values
in	*vect_size*	size of the input vectors

6.23.2.3 init_quantile()

```
void init_quantile (
            quantile_t * quantile,
            const int vect_size,
            const double alpha )
```

This function initializes a quantile structure.

Parameters

in,out	*quantile	the quantile structure to initialize
in	*vect_size*	size of the quantile vector
in	*alpha*	alpha parameter of the algotithm

6.23.2.4 read_quantile()

```
void read_quantile (
            quantile_t * quantiles,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function reads an array of quantile structures on disc

Parameters

in	*∗quantiles*	quantile structures to read, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.23.2.5 save_quantile()

```
void save_quantile (
            quantile_t * quantiles,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function writes an array of quantile structures on disc

Parameters

in	*∗quantiles*	quantile structures to save, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.24 /home/tterraz/avido/melissa/Melissa/source/stats/quantile.h File Reference

```
#include <stdio.h>
```

Include dependency graph for quantile.h: This graph shows which files directly or indirectly include this file:

Classes

- struct quantile_s

Typedefs

- typedef struct quantile_s quantile_t

Functions

- void init_quantile (quantile_t ∗quantile, const int vect_size, const double alpha)
- void increment_quantile (quantile_t ∗quantile, const int nmax, double in_vect[ ], const int vect_size)
- void save_quantile (quantile_t ∗quantile, int vect_size, int nb_time_steps, FILE ∗f)
- void read_quantile (quantile_t ∗quantile, int vect_size, int nb_time_steps, FILE ∗f)
- void free_quantile (quantile_t ∗quantile)

6.24.1 Detailed Description

Author

Terraz Théophile

Date

2017-18-05

6.24.2 Typedef Documentation

6.24.2.1 quantile_t

```
typedef struct quantile_s quantile_t
```

type corresponding to quantile_s

6.24.3 Function Documentation

6.24.3.1 free_quantile()

```
void free_quantile (
            quantile_t * quantile )
```

This function frees a quantile structure.

Parameters

in,out	*quantile	the quantile structure to free

6.24.3.2 increment_quantile()

```
void increment_quantile (
            quantile_t * quantile,
            const int nmax,
            double in_vect[],
            const int vect_size )
```

This function updates the incremental quantile.

Parameters

in,out	*quantile	input: previously computed iterative quantile, output: updated partial quantile
in	*nmax*	maximum number of iterations
in	*in_vect[]*	input vector of double values
in	*vect_size*	size of the input vectors

6.24.3.3 init_quantile()

```
void init_quantile (
            quantile_t * quantile,
            const int vect_size,
            const double alpha )
```

This function initializes a quantile structure.

Parameters

in,out	*quantile	the quantile structure to initialize
in	*vect_size*	size of the quantile vector
in	*alpha*	alpha parameter of the algotithm

6.24.3.4 read_quantile()

```
void read_quantile (
            quantile_t * quantiles,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function reads an array of quantile structures on disc

Parameters

in	*quantiles	quantile structures to read, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.24.3.5 save_quantile()

```
void save_quantile (
            quantile_t * quantiles,
```

```
int vect_size,
int nb_time_steps,
FILE * f )
```

This function writes an array of quantile structures on disc

Parameters

in	*quantiles	quantile structures to save, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.25 /home/tterraz/avido/melissa/Melissa/source/stats/sobol.c File Reference

Functions needed to compute sobol indices.

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <math.h>
#include "mean.h"
#include "variance.h"
#include "covariance.h"
#include "sobol.h"
#include "melissa_utils.h"
```

Include dependency graph for sobol.c:

Functions

- void init_sobol_jansen (sobol_array_t *sobol_array, int nb_parameters, int vect_size)
- void init_sobol_martinez (sobol_array_t *sobol_array, int nb_parameters, int vect_size)
- void increment_sobol_jansen (sobol_array_t *sobol_array, int nb_parameters, double **in_vect_tab, int vect_size)
- void increment_sobol_martinez (sobol_array_t *sobol_array, int nb_parameters, double **in_vect_tab, int vect_size)
- void confidence_sobol_martinez (sobol_array_t *sobol_array, int nb_parameters, int vect_size)
- int check_convergence_sobol_martinez (sobol_array_t **sobol_array, double confidence_value, int nb_↩ time_steps, int nb_parameters)
- void save_sobol_jansen (sobol_array_t *sobol_array, int vect_size, int nb_time_steps, int nb_parameters, FILE *f)
- void save_sobol_martinez (sobol_array_t *sobol_array, int vect_size, int nb_time_steps, int nb_parameters, FILE *f)
- void read_sobol_jansen (sobol_array_t *sobol_array, int vect_size, int nb_time_steps, int nb_parameters, FILE *f)
- void read_sobol_martinez (sobol_array_t *sobol_array, int vect_size, int nb_time_steps, int nb_parameters, FILE *f)
- void free_sobol_jansen (sobol_array_t *sobol_array, int nb_parameters)
- void free_sobol_martinez (sobol_array_t *sobol_array, int nb_parameters)

6.25.1 Detailed Description

Functions needed to compute sobol indices.

Author

Terraz Théophile

Date

2016-29-02

6.25.2 Function Documentation

6.25.2.1 check_convergence_sobol_martinez()

```
int check_convergence_sobol_martinez (
            sobol_array_t ** sobol_array,
            double confidence_value,
            int nb_time_steps,
            int nb_parameters )
```

This function check if the Sobol indice convergence has been reached

Parameters

out	***sobol_array*	Sobol indices
in	*confidence_value*	value to reach for the worst confidence interval
in	*nb_time_steps*	number of time steps of the study
in	*nb_parameters*	size of sobol_array->sobol_martinez

Returns

[out] int 0 if convergence is not reached 1 if convergence is reached

6.25.2.2 confidence_sobol_martinez()

```
void confidence_sobol_martinez (
            sobol_array_t * sobol_array,
            int nb_parameters,
            int vect_size )
```

This function computes the confidence interval for Martinez Sobol indices

Parameters

out	*∗sobol_array*	Sobol indices
in	*nb_parameters*	size of sobol_array->sobol_martinez
in	*vect_size*	size of input vectors

6.25.2.3 free_sobol_jansen()

```
void free_sobol_jansen (
            sobol_array_t * sobol_array,
            int nb_parameters )
```

This function frees a Jansen Sobol array structure

Parameters

in	*∗sobol_array*	reference or pointer to a sobol index structure to free
in	*nb_parameters*	number of parameters of the study

6.25.2.4 free_sobol_martinez()

```
void free_sobol_martinez (
            sobol_array_t * sobol_array,
            int nb_parameters )
```

This function frees a Martinez Sobol indices structure

Parameters

in	*∗sobol_array*	reference or pointer to a sobol array structure to free
in	*nb_parameters*	number of parameters of the study

6.25.2.5 increment_sobol_jansen()

```
void increment_sobol_jansen (
            sobol_array_t * sobol_array,
            int nb_parameters,
            double ** in_vect_tab,
            int vect_size )
```

This function computes Sobol indices using Jansen formula

Parameters

out	*sobol_array	computed sobol indices, using Jansen formula
in	*nb_parameters*	size of sobol_array->sobol_jansen
in	**in_vect_tab*	array of input vectors
in	*vect_size*	size of input vectors

6.25.2.6 increment_sobol_martinez()

```
void increment_sobol_martinez (
            sobol_array_t * sobol_array,
            int nb_parameters,
            double ** in_vect_tab,
            int vect_size )
```

This function computes Sobol indices using Martinez formula

Parameters

out	*sobol_array	computed sobol indices, using Martinez formula
in	*nb_parameters*	size of sobol_array->sobol_martinez
in	**in_vect_tab*	array of input vectors
in	*vect_size*	size of input vectors

6.25.2.7 init_sobol_jansen()

```
void init_sobol_jansen (
            sobol_array_t * sobol_array,
            int nb_parameters,
            int vect_size )
```

This function initialise a Jansen Sobol indices structure

Parameters

in,out	*sobol_array	input: reference or pointer to an uninitialised sobol indices structure, output: initialised structure, with values and variances set to 0
in	*nb_parameters*	number of parameters of the study
in	*vect_size*	size of the input vectors

6.25.2.8 init_sobol_martinez()

```
void init_sobol_martinez (
```

```
            sobol_array_t * sobol_array,
            int nb_parameters,
            int vect_size )
```

This function initialise a Martinez Sobol indices structure

Parameters

in,out	*sobol_array	input: reference or pointer to an uninitialised sobol indices structure, output: initialised structure, with values and variances set to 0
in	*nb_parameters*	number of parameters of the study
in	*vect_size*	size of the input vectors

6.25.2.9 read_sobol_jansen()

```
void read_sobol_jansen (
            sobol_array_t * sobol_array,
            int vect_size,
            int nb_time_steps,
            int nb_parameters,
            FILE * f )
```

This function reads an array of sobol_jansen structures on disc

Parameters

in	*sobol_array	sobol_array structures to read, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*nb_parameters*	number of parameters of the study
in	*f*	file descriptor

6.25.2.10 read_sobol_martinez()

```
void read_sobol_martinez (
            sobol_array_t * sobol_array,
            int vect_size,
            int nb_time_steps,
            int nb_parameters,
            FILE * f )
```

This function reads an array of sobol_martinez structures on disc

Parameters

in	*sobol_array	sobol_array structures to read, size nb_time_steps
in	*vect_size*	size of double vectors

Parameters

in	*nb_time_steps*	number of time_steps of the study
in	*nb_parameters*	number of parameters of the study
in	*f*	file descriptor

6.25.2.11 save_sobol_jansen()

```
void save_sobol_jansen (
            sobol_array_t * sobol_array,
            int vect_size,
            int nb_time_steps,
            int nb_parameters,
            FILE * f )
```

This function writes an array of sobol_jansen structures on disc

Parameters

in	*∗sobol_array*	sobol_array structures to save, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*nb_parameters*	number of parameters of the study
in	*f*	file descriptor

6.25.2.12 save_sobol_martinez()

```
void save_sobol_martinez (
            sobol_array_t * sobol_array,
            int vect_size,
            int nb_time_steps,
            int nb_parameters,
            FILE * f )
```

This function writes an array of sobol_martinez structures on disc

Parameters

in	*∗sobol_array*	sobol_array structures to save, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*nb_parameters*	number of parameters of the study
in	*f*	file descriptor

6.26 /home/tterraz/avido/melissa/Melissa/source/stats/sobol.h File Reference

This graph shows which files directly or indirectly include this file:

Classes

- struct sobol_jansen_s
- struct sobol_martinez_s
- struct sobol_array_s

Typedefs

- typedef struct sobol_jansen_s sobol_jansen_t
- typedef struct sobol_martinez_s sobol_martinez_t
- typedef struct sobol_array_s sobol_array_t

Functions

- void init_sobol_jansen (sobol_array_t *sobol_array, int nb_parameters, int vect_size)
- void init_sobol_martinez (sobol_array_t *sobol_array, int nb_parameters, int vect_size)
- void increment_sobol_jansen (sobol_array_t *sobol_array, int nb_parameters, double **in_vect_tab, int vect_size)
- void increment_sobol_martinez (sobol_array_t *sobol_array, int nb_parameters, double **in_vect_tab, int vect_size)
- void confidence_sobol_martinez (sobol_array_t *sobol_array, int nb_parameters, int vect_size)
- int check_convergence_sobol_martinez (sobol_array_t **sobol_array, double confidence_value, int nb_time_steps, int nb_parameters)
- void save_sobol_jansen (sobol_array_t *sobol_array, int vect_size, int nb_time_steps, int nb_parameters, FILE *f)
- void save_sobol_martinez (sobol_array_t *sobol_array, int vect_size, int nb_time_steps, int nb_parameters, FILE *f)
- void read_sobol_jansen (sobol_array_t *sobol_array, int vect_size, int nb_time_steps, int nb_parameters, FILE *f)
- void read_sobol_martinez (sobol_array_t *sobol_array, int vect_size, int nb_time_steps, int nb_parameters, FILE *f)
- void free_sobol_jansen (sobol_array_t *sobol_array, int nb_parameters)
- void free_sobol_martinez (sobol_array_t *sobol_array, int nb_parameters)

6.26.1 Detailed Description

Author

Terraz Théophile

Date

2017-15-01

6.26.2 Typedef Documentation

6.26.2.1 sobol_array_t

```
typedef struct sobol_array_s sobol_array_t
```

type corresponding to sobol_array_s

6.26.2.2 sobol_jansen_t

```
typedef struct sobol_jansen_s sobol_jansen_t
```

type corresponding to sobol_martinez_s

6.26.2.3 sobol_martinez_t

```
typedef struct sobol_martinez_s sobol_martinez_t
```

type corresponding to sobol_martinez_s

6.26.3 Function Documentation

6.26.3.1 check_convergence_sobol_martinez()

```
int check_convergence_sobol_martinez (
            sobol_array_t ** sobol_array,
            double confidence_value,
            int nb_time_steps,
            int nb_parameters )
```

This function check if the Sobol indice convergence has been reached

Parameters

out	***sobol_array*	Sobol indices
in	*confidence_value*	value to reach for the worst confidence interval
in	*nb_time_steps*	number of time steps of the study
in	*nb_parameters*	size of sobol_array->sobol_martinez

Returns

[out] int 0 if convergence is not reached 1 if convergence is reached

6.26.3.2 confidence_sobol_martinez()

```
void confidence_sobol_martinez (
            sobol_array_t * sobol_array,
            int nb_parameters,
            int vect_size )
```

This function computes the confidence interval for Martinez Sobol indices

Parameters

out	*∗sobol_array*	Sobol indices
in	*nb_parameters*	size of sobol_array->sobol_martinez
in	*vect_size*	size of input vectors

6.26.3.3 free_sobol_jansen()

```
void free_sobol_jansen (
            sobol_array_t * sobol_array,
            int nb_parameters )
```

This function frees a Jansen Sobol array structure

Parameters

in	*∗sobol_array*	reference or pointer to a sobol index structure to free
in	*nb_parameters*	number of parameters of the study

6.26.3.4 free_sobol_martinez()

```
void free_sobol_martinez (
            sobol_array_t * sobol_array,
            int nb_parameters )
```

This function frees a Martinez Sobol indices structure

Parameters

in	*∗sobol_array*	reference or pointer to a sobol array structure to free
in	*nb_parameters*	number of parameters of the study

6.26.3.5 increment_sobol_jansen()

```
void increment_sobol_jansen (
            sobol_array_t * sobol_array,
            int nb_parameters,
            double ** in_vect_tab,
            int vect_size )
```

This function computes Sobol indices using Jansen formula

Parameters

out	*sobol_array	computed sobol indices, using Jansen formula
in	*nb_parameters*	size of sobol_array->sobol_jansen
in	**in_vect_tab*	array of input vectors
in	*vect_size*	size of input vectors

6.26.3.6 increment_sobol_martinez()

```
void increment_sobol_martinez (
            sobol_array_t * sobol_array,
            int nb_parameters,
            double ** in_vect_tab,
            int vect_size )
```

This function computes Sobol indices using Martinez formula

Parameters

out	*sobol_array	computed sobol indices, using Martinez formula
in	*nb_parameters*	size of sobol_array->sobol_martinez
in	**in_vect_tab*	array of input vectors
in	*vect_size*	size of input vectors

6.26.3.7 init_sobol_jansen()

```
void init_sobol_jansen (
            sobol_array_t * sobol_array,
            int nb_parameters,
            int vect_size )
```

This function initialise a Jansen Sobol indices structure

Parameters

in,out	*sobol_array	input: reference or pointer to an uninitialised sobol indices structure, output: initialised structure, with values and variances set to 0
in	*nb_parameters*	number of parameters of the study
in	*vect_size*	size of the input vectors

6.26.3.8 init_sobol_martinez()

```
void init_sobol_martinez (
            sobol_array_t * sobol_array,
            int nb_parameters,
            int vect_size )
```

This function initialise a Martinez Sobol indices structure

Parameters

in,out	*sobol_array	input: reference or pointer to an uninitialised sobol indices structure, output: initialised structure, with values and variances set to 0
in	*nb_parameters*	number of parameters of the study
in	*vect_size*	size of the input vectors

6.26.3.9 read_sobol_jansen()

```
void read_sobol_jansen (
            sobol_array_t * sobol_array,
            int vect_size,
            int nb_time_steps,
            int nb_parameters,
            FILE * f )
```

This function reads an array of sobol_jansen structures on disc

Parameters

in	*sobol_array	sobol_array structures to read, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*nb_parameters*	number of parameters of the study
in	*f*	file descriptor

6.26.3.10 read_sobol_martinez()

```
void read_sobol_martinez (
            sobol_array_t * sobol_array,
            int vect_size,
            int nb_time_steps,
            int nb_parameters,
            FILE * f )
```

This function reads an array of sobol_martinez structures on disc

Parameters

in	**sobol_array*	sobol_array structures to read, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*nb_parameters*	number of parameters of the study
in	*f*	file descriptor

6.26.3.11 save_sobol_jansen()

```
void save_sobol_jansen (
            sobol_array_t * sobol_array,
            int vect_size,
            int nb_time_steps,
            int nb_parameters,
            FILE * f )
```

This function writes an array of sobol_jansen structures on disc

Parameters

in	**sobol_array*	sobol_array structures to save, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*nb_parameters*	number of parameters of the study
in	*f*	file descriptor

6.26.3.12 save_sobol_martinez()

```
void save_sobol_martinez (
            sobol_array_t * sobol_array,
            int vect_size,
            int nb_time_steps,
            int nb_parameters,
            FILE * f )
```

This function writes an array of sobol_martinez structures on disc

Parameters

in	*∗sobol_array*	sobol_array structures to save, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*nb_parameters*	number of parameters of the study
in	*f*	file descriptor

6.27 /home/tterraz/avido/melissa/Melissa/source/stats/threshold.c File Reference

Threshold exceedance related functions.

```
#include <stdlib.h>
#include <stdio.h>
#include "threshold.h"
#include "melissa_utils.h"
```

Include dependency graph for threshold.c:

Functions

- void update_threshold_exceedance (int threshold_exceedance[ ], double threshold, double in_vect[ ], const int vect_size)
- void save_threshold (int ∗∗threshold_exceedance, int vect_size, int nb_time_steps, FILE ∗f)
- void read_threshold (int ∗∗threshold_exceedance, int vect_size, int nb_time_steps, FILE ∗f)

6.27.1 Detailed Description

Threshold exceedance related functions.

Author

Terraz Théophile

Date

2016-15-02

6.27.2 Function Documentation

6.27.2.1 read_threshold()

```
void read_threshold (
            int ** threshold_exceedance,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function reads an array of threshold exceedance vectors on disc

Parameters

in	***threshold_exceedance*	threshold exceedance array of vectors to read, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.27.2.2 save_threshold()

```
void save_threshold (
            int ** threshold_exceedance,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function writes an array of threshold exceedance vectors on disc

Parameters

in	***threshold_exceedance*	threshold exceedance array of vectors to save, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.27.2.3 update_threshold_exceedance()

```
void update_threshold_exceedance (
            int threshold_exceedance[],
            double threshold,
            double in_vect[],
            const int vect_size )
```

This function updates the number of values exceeding a given threshold

Parameters

in,out	*threshold_exceedance[]*	number of threshold exceedance occurences
in	*threshold*	
in	*in_vect[]*	input vector of double values
in	*vect_size*	size of the input vector

6.28 /home/tterraz/avido/melissa/Melissa/source/stats/threshold.h File Reference

This graph shows which files directly or indirectly include this file:

Functions

- void update_threshold_exceedance (int threshold_exceedance[ ], double threshold, double in_vect[ ], const int vect_size)
- void save_threshold (int ∗∗threshold_exceedance, int vect_size, int nb_time_steps, FILE ∗f)
- void read_threshold (int ∗∗threshold_exceedance, int vect_size, int nb_time_steps, FILE ∗f)

6.28.1 Detailed Description

Author

Terraz Théophile

Date

2017-15-01

6.28.2 Function Documentation

6.28.2.1 read_threshold()

```
void read_threshold (
            int ** threshold_exceedance,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function reads an array of threshold exceedance vectors on disc

Parameters

in	∗∗*threshold_exceedance*	threshold exceedance array of vectors to read, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.28.2.2 save_threshold()

```
void save_threshold (
```

```
            int ** threshold_exceedance,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function writes an array of threshold exceedance vectors on disc

Parameters

in	***threshold_exceedance*	threshold exceedance array of vectors to save, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.28.2.3 update_threshold_exceedance()

```
void update_threshold_exceedance (
            int threshold_exceedance[],
            double threshold,
            double in_vect[],
            const int vect_size )
```

This function updates the number of values exceeding a given threshold

Parameters

in,out	*threshold_exceedance[]*	number of threshold exceedance occurences
in	*threshold*	
in	*in_vect[]*	input vector of double values
in	*vect_size*	size of the input vector

6.29 /home/tterraz/avido/melissa/Melissa/source/stats/variance.c File Reference

Variance related functions.

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "mean.h"
#include "variance.h"
#include "melissa_utils.h"
```

Include dependency graph for variance.c:

Functions

- void init_variance (variance_t *variance, const int vect_size)

- void increment_mean_and_variance (variance_t ∗partial_variance, double in_vect[ ], const int vect_size)
- void increment_variance (variance_t ∗partial_variance, double in_vect[ ], const int vect_size)
- void update_variance (variance_t ∗variance1, variance_t ∗variance2, variance_t ∗updated_variance, const int vect_size)
- void save_variance (variance_t ∗vars, int vect_size, int nb_time_steps, FILE ∗f)
- void read_variance (variance_t ∗vars, int vect_size, int nb_time_steps, FILE ∗f)
- void free_variance (variance_t ∗variance)

6.29.1 Detailed Description

Variance related functions.

Author

Terraz Théophile

Date

2016-15-02

6.29.2 Function Documentation

6.29.2.1 free_variance()

```
void free_variance (
            variance_t * variance )
```

This function frees a variance structure.

Parameters

in	∗*variance*	the variance structure to free

6.29.2.2 increment_mean_and_variance()

```
void increment_mean_and_variance (
            variance_t * partial_variance,
            double in_vect[],
            const int vect_size )
```

This function updates the incremental mean and variance.

Parameters

in,out	*partial_variance	input: previously computed partial variance, output: updated partial variance
in	*in_vect[]*	input vector of double values
in	*vect_size*	size of the input vectors

6.29.2.3 increment_variance()

```
void increment_variance (
            variance_t * partial_variance,
            double in_vect[],
            const int vect_size )
```

This function updates the incremental variance.

Parameters

in,out	*partial_variance	input: previously computed partial variance, output: updated partial variance
in	*in_vect[]*	input vector of double values
in	*vect_size*	size of the input vectors

6.29.2.4 init_variance()

```
void init_variance (
            variance_t * variance,
            const int vect_size )
```

This function initializes a variance structure.

Parameters

in,out	*variance	the variance structure to initialize
in	*vect_size*	size of the variance vector

6.29.2.5 read_variance()

```
void read_variance (
            variance_t * vars,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function reads an array of variances structures on disc

Parameters

in	*∗vars*	variance structures to read, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.29.2.6 save_variance()

```
void save_variance (
            variance_t * vars,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function writes an array of variances structures on disc

Parameters

in	*∗vars*	variance structures to save, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.29.2.7 update_variance()

```
void update_variance (
            variance_t * variance1,
            variance_t * variance2,
            variance_t * updated_variance,
            const int vect_size )
```

This function agregates two partial variances.

Parameters

in	*∗variance1*	first input partial variances
in	*∗variance2*	second input partial variances
out	*∗updated_variance*	the updated variances
in	*vect_size*	size of the input and output vectors

6.30 /home/tterraz/avido/melissa/Melissa/source/stats/variance.h File Reference

This graph shows which files directly or indirectly include this file:

Classes

- struct variance_s

Typedefs

- typedef struct variance_s variance_t

Functions

- void init_variance (variance_t *variance, const int vect_size)
- void increment_mean_and_variance (variance_t *partial_variance, double in_vect[ ], const int vect_size)
- void increment_variance (variance_t *partial_variance, double in_vect[ ], const int vect_size)
- void update_variance (variance_t *variance1, variance_t *variance2, variance_t *updated_variance, const int vect_size)
- void save_variance (variance_t *vars, int vect_size, int nb_time_steps, FILE *f)
- void read_variance (variance_t *vars, int vect_size, int nb_time_steps, FILE *f)
- void free_variance (variance_t *variance)

6.30.1 Detailed Description

Author

Terraz Théophile

Date

2017-15-01

6.30.2 Typedef Documentation

6.30.2.1 variance_t

```
typedef struct variance_s variance_t
```

type corresponding to variance_s

6.30.3 Function Documentation

6.30.3.1 free_variance()

```
void free_variance (
            variance_t * variance )
```

This function frees a variance structure.

Parameters

in	*variance	the variance structure to free

6.30.3.2 increment_mean_and_variance()

```
void increment_mean_and_variance (
            variance_t * partial_variance,
            double in_vect[],
            const int vect_size )
```

This function updates the incremental mean and variance.

Parameters

in,out	*partial_variance	input: previously computed partial variance, output: updated partial variance
in	*in_vect[]*	input vector of double values
in	*vect_size*	size of the input vectors

6.30.3.3 increment_variance()

```
void increment_variance (
            variance_t * partial_variance,
            double in_vect[],
            const int vect_size )
```

This function updates the incremental variance.

Parameters

in,out	*partial_variance	input: previously computed partial variance, output: updated partial variance
in	*in_vect[]*	input vector of double values
in	*vect_size*	size of the input vectors

6.30.3.4 init_variance()

```
void init_variance (
            variance_t * variance,
            const int vect_size )
```

This function initializes a variance structure.

Parameters

in,out	*variance	the variance structure to initialize
in	*vect_size*	size of the variance vector

6.30.3.5 read_variance()

```
void read_variance (
            variance_t * vars,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function reads an array of variances structures on disc

Parameters

in	*vars	variance structures to read, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.30.3.6 save_variance()

```
void save_variance (
            variance_t * vars,
            int vect_size,
            int nb_time_steps,
            FILE * f )
```

This function writes an array of variances structures on disc

Parameters

in	*vars	variance structures to save, size nb_time_steps
in	*vect_size*	size of double vectors
in	*nb_time_steps*	number of time_steps of the study
in	*f*	file descriptor

6.30.3.7 update_variance()

```
void update_variance (
            variance_t * variance1,
```

```
variance_t * variance2,
variance_t * updated_variance,
const int vect_size )
```

This function agregates two partial variances.

Parameters

in	*variance1	first input partial variances
in	*variance2	second input partial variances
out	*updated_variance	the updated variances
in	vect_size	size of the input and output vectors

6.31 /home/tterraz/avido/melissa/Melissa/source/utils/melissa_utils.c File Reference

Functions used in Melissa.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <stdint.h>
#include <time.h>
#include <sys/timeb.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <ifaddrs.h>
#include <zmq.h>
#include "melissa_utils.h"
```

Include dependency graph for melissa_utils.c:

Functions

- void melissa_logo ()
- void * melissa_malloc (size_t size)
- void * melissa_calloc (size_t num, size_t size)
- void melissa_free (void *ptr)
- void melissa_bind (void *socket, char *port_name)
- void melissa_connect (void *socket, char *port_name)
- double melissa_get_time ()
- void melissa_get_node_name (char *node_name)
- void set_bit (int32_t *vect, int pos)
- void clear_bit (int32_t *vect, int pos)
- int test_bit (int32_t *vect, int pos)

6.31.1 Detailed Description

Functions used in Melissa.

Author

Terraz Théophile

Date

2017-15-01

6.32 /home/tterraz/avido/melissa/Melissa/source/utils/melissa_utils.h File Reference

`#include <stdio.h>`
Include dependency graph for melissa_utils.h: This graph shows which files directly or indirectly include this file:

Macros

- #define MPI_MAX_PROCESSOR_NAME 256
- #define MAX_FIELD_NAME 128

Functions

- void melissa_logo ()
- void ∗ melissa_malloc (size_t size)
- void ∗ melissa_calloc (size_t num, size_t size)
- void melissa_free (void ∗ptr)
- void melissa_bind (void ∗socket, char ∗port_name)
- void melissa_connect (void ∗socket, char ∗port_name)
- double melissa_get_time ()
- void melissa_get_node_name (char ∗node_name)
- void set_bit (int32_t ∗vect, int pos)
- void clear_bit (int32_t ∗vect, int pos)
- int test_bit (int32_t ∗vect, int pos)

6.32.1 Detailed Description

Author

Terraz Théophile

Date

2016-15-02

6.32.2 Macro Definition Documentation

6.32.2.1 MAX_FIELD_NAME

```
#define MAX_FIELD_NAME 128
```

Define name size if not defined

6.32.2.2 MPI_MAX_PROCESSOR_NAME

```
#define MPI_MAX_PROCESSOR_NAME 256
```

Define the macro if mpi.h not present

Index