My Project

Generated by Doxygen 1.9.2

1	Namespace Index	1
	1.1 Namespace List	1
2	Hierarchical Index	3
	2.1 Class Hierarchy	3
3	Class Index	5
	3.1 Class List	5
4	File Index	7
	4.1 File List	7
5	Namespace Documentation	9
	5.1 LocallyStationaryModels::cd Namespace Reference	9
	5.1.1 Detailed Description	9
	5.2 LocallyStationaryModels::gf Namespace Reference	9
	5.2.1 Detailed Description	10
	5.2.2 Function Documentation	10
	5.2.2.1 make_grid()	10
	5.2.2.2 pizza()	10
	5.3 LocallyStationaryModels::kf Namespace Reference	10
	5.3.1 Detailed Description	11
	5.3.2 Function Documentation	11
	5.3.2.1 gaussian()	11
	5.3.2.2 identity()	11
	5.3.2.3 make kernel()	11
	3.5.2.6 mano_nemon()	
6	Class Documentation	13
	6.1 LocallyStationaryModels::Anchor Class Reference	13
	6.1.1 Detailed Description	13
	6.1.2 Constructor & Destructor Documentation	13
	6.1.2.1 Anchor()	13
	6.1.3 Member Function Documentation	14
	6.1.3.1 get_origin()	14
	6.1.3.2 get_tiles_dimensions()	14
	6.2 LocallyStationaryModels::Exponential Class Reference	14
	6.2.1 Member Function Documentation	15
	6.2.1.1 operator()()	15
	6.3 LocallyStationaryModels::Gaussian Class Reference	15
	6.3.1 Member Function Documentation	15
	6.3.1.1 operator()()	16
	6.4 LocallyStationaryModels::Grid Class Reference	16
	6.4.1 Detailed Description	16
	6.4.2 Constructor & Destructor Documentation	17

6.4.2.1 Grid()	17
6.4.3 Member Function Documentation	17
6.4.3.1 build_grid()	17
6.4.3.2 get_grid()	17
6.4.3.3 get_normh()	18
6.4.3.4 get_x()	18
6.4.3.5 get_y()	18
6.5 LocallyStationaryModels::Kernel Class Reference	18
6.5.1 Detailed Description	19
6.5.2 Constructor & Destructor Documentation	19
6.5.2.1 Kernel()	19
6.5.3 Member Function Documentation	19
6.5.3.1 build_kernel()	19
<b>6.5.3.2</b> build_simple_kernel() [1/2]	19
<b>6.5.3.3</b> build_simple_kernel() [2/2]	20
6.5.3.4 get_kernel()	20
6.5.3.5 operator()()	20
6.6 LocallyStationaryModels::Matern Class Reference	21
6.6.1 Member Function Documentation	21
6.6.1.1 operator()()	21
6.7 LocallyStationaryModels::MaternNuFixed Class Reference	21
6.7.1 Member Function Documentation	22
6.7.1.1 operator()()	22
6.8 LocallyStationaryModels::Opt Class Reference	22
6.8.1 Detailed Description	23
6.8.2 Constructor & Destructor Documentation	23
6.8.2.1 Opt()	23
6.8.3 Member Function Documentation	23
6.8.3.1 get_solutions()	24
6.9 LocallyStationaryModels::Predictor Class Reference	24
6.9.1 Detailed Description	24
6.9.2 Constructor & Destructor Documentation	24
6.9.2.1 Predictor()	24
6.10 LocallyStationaryModels::SampleVar Class Reference	25
6.10.1 Detailed Description	25
6.10.2 Constructor & Destructor Documentation	25
6.10.2.1 SampleVar()	25
6.10.3 Member Function Documentation	26
6.10.3.1 build_samplevar()	26
6.10.3.2 get_denominators()	26
6.10.3.3 get_grid()	26
6.10.3.4 get_kernel()	27

6.10.3.5 get_normh()	27
6.10.3.6 get_squaredweights()	27
6.10.3.7 get_variogram()	27
6.10.3.8 get_x()	27
6.10.3.9 get_y()	28
6.11 LocallyStationaryModels::Smt Class Reference	28
6.11.1 Detailed Description	28
6.11.2 Constructor & Destructor Documentation	28
6.11.2.1 Smt() [1/2]	28
<b>6.11.2.2 Smt()</b> [2/2]	29
6.11.3 Member Function Documentation	29
6.11.3.1 get_anchorpos()	29
6.11.3.2 get_optimal_delta()	29
6.11.3.3 get_solutions()	30
6.11.3.4 smooth_vector()	30
6.12 LocallyStationaryModels::TargetFunction Struct Reference	30
6.12.1 Detailed Description	31
6.12.2 Constructor & Destructor Documentation	31
6.12.2.1 TargetFunction()	31
6.12.3 Member Function Documentation	31
6.12.3.1 operator()() [1/2]	32
6.12.3.2 operator()() [2/2]	32
6.13 LocallyStationaryModels::Tolerances Struct Reference	32
6.13.1 Detailed Description	33
6.14 LocallyStationaryModels::VariogramFunction Class Reference	33
6.14.1 Member Function Documentation	33
6.14.1.1 operator()()	34
7 File Documentation	35
7.1 anchor.hpp	35
7.2 grid.hpp	36
7.3 gridfunctions.hpp	36
7.4 kernel.hpp	37
7.5 kernelfunctions.hpp	37
7.6 kriging.hpp	37
7.7 samplevar.hpp	38
7.8 smooth.hpp	39
7.9 tolerances.hpp	39
7.10 traits.hpp	39
7.11 variogramfit.hpp	40
7.12 variogramfunctions.hpp	41
Index	43

# **Chapter 1**

# Namespace Index

## 1.1 Namespace List

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

LocallyStationaryModels::cd	
Collects all the typedef used inside the code	9
LocallyStationaryModels::gf	
Collect the grid functions and the corresponding factory function	9
LocallyStationaryModels::kf	
Collect the kernel functions and the corresponding factory function	0

2 Namespace Index

# Chapter 2

# **Hierarchical Index**

## 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

LocallyStationaryModels::Anchor	13
LocallyStationaryModels::Grid	16
LocallyStationaryModels::Kernel	18
LocallyStationaryModels::Opt	22
LocallyStationaryModels::Predictor	<b>2</b> 4
LocallyStationaryModels::SampleVar	25
LocallyStationaryModels::Smt	28
LocallyStationaryModels::TargetFunction	30
LocallyStationaryModels::Tolerances	32
LocallyStationaryModels::VariogramFunction	33
LocallyStationaryModels::Exponential	14
LocallyStationaryModels::Gaussian	15
LocallyStationaryModels::Matern	21
LocallyStationaryModels::MaternNuFixed	21

4 Hierarchical Index

# **Chapter 3**

# **Class Index**

## 3.1 Class List

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

LocallyStationaryModelsAnchor	
Simple class to find the anchor points given the data	13
LocallyStationaryModels::Exponential	14
LocallyStationaryModels::Gaussian	15
LocallyStationaryModels::Grid	
Class to build the grid	16
LocallyStationaryModels::Kernel	
Class to compute the kernel matrix	18
LocallyStationaryModels::Matern	21
LocallyStationaryModels::MaternNuFixed	21
LocallyStationaryModels::Opt	
Class to estimate the value of the parameters of the variogram in each point by optimizing the	
correspondent funzionedaottimizzare relying on the library LBFGSpp	22
LocallyStationaryModels::Predictor	
Class to perform kriging on the data	24
LocallyStationaryModels::SampleVar	
Class to build and store the empiric variogram in all the anchor points	25
LocallyStationaryModels::Smt	
Class to perform kernel smoothing of the paramters estimated in the anchor points to get the non	
stationary value of the parameters in any position of the domain	28
LocallyStationaryModels::TargetFunction	
Functor to pass to the optimizer that contains the wls to be minimized	30
LocallyStationaryModels::Tolerances	
Collects all the tolerances and the constants used inside the code	32
LocallyStationaryModels::VariogramFunction	33

6 Class Index

# **Chapter 4**

# File Index

## 4.1 File List

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

anchor.hpp										 							 							 	??
grid.hpp										 														 	??
gridfunctions.hpp										 													 	 	??
kernel.hpp										 							 						 	 	??
kernelfunctions.h	pp									 													 	 	??
kriging.hpp										 													 	 	??
samplevar.hpp										 													 	 	??
smooth.hpp										 													 	 	??
tolerances.hpp										 							 						 	 	??
traits.hpp										 													 	 	??
variogramfit.hpp										 													 	 	??
variogramfunction	ıs.	h	gc	)						 													 	 	??

8 File Index

## **Chapter 5**

# **Namespace Documentation**

## 5.1 LocallyStationaryModels::cd Namespace Reference

collects all the typedef used inside the code

## **Typedefs**

- using vector = Eigen::VectorXd
- using matrix = Eigen::MatrixXd
- using matrixI = Eigen::MatrixXi
- using vectorptr = std::shared\_ptr< vector >
- using matrixptr = std::shared\_ptr< matrix >
- using matrixlptr = std::shared\_ptr< matrixl >
- using  $vectorind = std::vector < size_t >$
- using **kernelfunction** = std::function < double(const vector &, const vector &, const double &)>
- using gridfunction = std::function< matrix|ptr(const matrixptr &, const size\_t &, const size\_t &, const double &)>

## 5.1.1 Detailed Description

collects all the typedef used inside the code

Namespace cd

## 5.2 LocallyStationaryModels::gf Namespace Reference

collect the grid functions and the corresponding factory function

### **Functions**

 matrixlptr pizza (const cd::matrixptr &data, const size\_t &n\_angles, const size\_t &n\_intervals, const double &epsilon)

this function builds a 2D-grid using a "a fette di pizza" (slices-of-pizza like) algorithm to partition the domain

• gridfunction make\_grid (const std::string &id)

allow to select between the preferred method to build the grid

## 5.2.1 Detailed Description

collect the grid functions and the corresponding factory function

Namespace gf

## 5.2.2 Function Documentation

## 5.2.2.1 make\_grid()

allow to select between the preferred method to build the grid

#### **Parameters**

```
id name of the function of choice
```

## 5.2.2.2 pizza()

this function builds a 2D-grid using a "a fette di pizza" (slices-of-pizza like) algorithm to partition the domain

#### **Parameters**

data	a shared pointer to the matrix of the coordinates
n_angles	number of slices of the pizza
n_intervals	number of the pieces for each slice of the pizza
epsilon	bandwidth parameter epsilon. Same of the kernel

## 5.3 LocallyStationaryModels::kf Namespace Reference

collect the kernel functions and the corresponding factory function

## **Functions**

- double gaussian (const vector &x, const vector &y, const double &epsilon)
- · double identity (const vector &x, const vector &y, const double &epsilon)
- kernelfunction make\_kernel (const std::string &id)

allow to select between pre-built kernel functions

## 5.3.1 Detailed Description

collect the kernel functions and the corresponding factory function

Namespace kf

## 5.3.2 Function Documentation

## 5.3.2.1 gaussian()

```
double LocallyStationaryModels::kf::gaussian ( const cd::vector & x, const cd::vector & y, const double & epsilon )
```

#### Returns

```
e^{(-norm(x-y)^2/epsilon^2)}
```

#### 5.3.2.2 identity()

#### Returns

1 only if the norm of the difference between x and y is less the the square of epsilon

## 5.3.2.3 make\_kernel()

allow to select between pre-built kernel functions

## **Parameters**

id a string with the name of the kernel function you want to use

## **Chapter 6**

## **Class Documentation**

## 6.1 LocallyStationaryModels::Anchor Class Reference

a simple class to find the anchor points given the data

```
#include <anchor.hpp>
```

## **Public Member Functions**

- Anchor (const cd::matrixptr &data, const double &n\_pieces)
   constructor
- const cd::matrix find\_anchorpoints ()

this function returns the coordinates of the anchor points in a way such that every anchor point has at least one point of the domain in its neighbourhood

- std::pair< double, double > get\_origin () const
- std::pair< double, double > get\_tiles\_dimensions () const

## 6.1.1 Detailed Description

a simple class to find the anchor points given the data

## 6.1.2 Constructor & Destructor Documentation

## 6.1.2.1 Anchor()

constructor

#### **Parameters**

data	shared pointer to the matrix with the coordinates of the dataset points
n_pieces	the number of tiles per row and column of the grid

## 6.1.3 Member Function Documentation

## 6.1.3.1 get\_origin()

std::pair< double, double > LocallyStationaryModels::Anchor::get\_origin ( ) const [inline]

## Returns

the coordinates of the origin of the grid

## 6.1.3.2 get\_tiles\_dimensions()

std::pair< double, double > LocallyStationaryModels::Anchor::get\_tiles\_dimensions ( ) const
[inline]

#### Returns

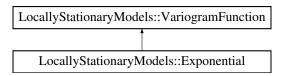
the dimensions (height and width) of each cell of the grid

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

· anchor.hpp

## 6.2 LocallyStationaryModels::Exponential Class Reference

Inheritance diagram for LocallyStationaryModels::Exponential:



#### **Public Member Functions**

• double operator() (const cd::vector &params, const double &x, const double &y) override

## **Additional Inherited Members**

## **6.2.1 Member Function Documentation**

## 6.2.1.1 operator()()

## sigma \* sigma \* (1 - exp(-h))

#### **Parameters**

params a vector with lambda1, lambda2, phi and sigma in this exact order

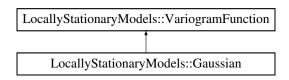
 $Implements\ Locally Stationary Models:: Variogram Function.$ 

The documentation for this class was generated from the following files:

- · variogramfunctions.hpp
- · variogramfunctions.cpp

## 6.3 LocallyStationaryModels::Gaussian Class Reference

Inheritance diagram for LocallyStationaryModels::Gaussian:



#### **Public Member Functions**

• double operator() (const cd::vector &params, const double &x, const double &y) override

## **Additional Inherited Members**

## 6.3.1 Member Function Documentation

#### 6.3.1.1 operator()()

#### Returns

```
sigma * sigma * (1 - exp(-h*h))
```

#### **Parameters**

params a vector with lambda1, lambda2, phi and sigma in this exact order

Implements LocallyStationaryModels::VariogramFunction.

The documentation for this class was generated from the following files:

- · variogramfunctions.hpp
- · variogramfunctions.cpp

## 6.4 LocallyStationaryModels::Grid Class Reference

class to build the grid

```
#include <grid.hpp>
```

#### **Public Member Functions**

- Grid (const std::string &id, const double &epsilon)
   constructor
- Grid ()

constructor. Use the Pizza style by default

- void build\_grid (const cd::matrixptr &data, const size\_t &n\_angles, const size\_t &n\_intervals)
   build the grid
- · const cd::matrixlptr get grid () const
- const cd::vectorptr get\_normh () const
- const cd::vectorptr get\_x () const
- const cd::vectorptr get\_y () const

## 6.4.1 Detailed Description

class to build the grid

## 6.4.2 Constructor & Destructor Documentation

## 6.4.2.1 Grid()

#### constructor

#### **Parameters**

id	name of the grid function
epsilon	the same epsilon regulating the kernel

## 6.4.3 Member Function Documentation

## 6.4.3.1 build\_grid()

## build the grid

#### **Parameters**

data	a shared pointer to the matrix of the coordinates
n_angles	number of slices of the pizza
n_intervals number of the pieces for each slice of the pizz	

## 6.4.3.2 get\_grid()

```
const matrixIptr LocallyStationaryModels::Grid::get_grid ( ) const
```

## Returns

a shared pointer to the grid

#### 6.4.3.3 get\_normh()

```
const vectorptr LocallyStationaryModels::Grid::get_normh ( ) const
```

#### Returns

a shared pointer to m\_normh

#### 6.4.3.4 get x()

```
const vectorptr LocallyStationaryModels::Grid::get_x ( ) const
```

#### Returns

a pointer to the vector containing the xs of the centers of the cells of the grid

#### 6.4.3.5 get\_y()

```
const vectorptr LocallyStationaryModels::Grid::get_y ( ) const
```

#### Returns

a pointer to the vector containing the ys of the centers of the cells of the grid

The documentation for this class was generated from the following files:

- grid.hpp
- grid.cpp

## 6.5 LocallyStationaryModels::Kernel Class Reference

class to compute the kernel matrix

```
#include <kernel.hpp>
```

## **Public Member Functions**

- Kernel (const std::string &id, const double &epsilon)
   constructor
- · Kernel ()

default constuctor with a gaussian kernel and epsilon equal to 1.

- double operator() (const cd::vector &x, const cd::vector &y) const
- void build\_kernel (const cd::matrixptr &data, const cd::matrixptr &anchorpoints)

build the "star" version of the kernel that contains the standardized kernel weights in such a way that each row sums to one

void build\_simple\_kernel (const cd::matrixptr &coordinates)

build the "standard" version of the kernel needed for smoothing

• void build\_simple\_kernel (const cd::matrixptr &coordinates, const double &epsilon)

build the "standard" version of the kernel needed for smoothing

const cd::matrixptr get\_kernel () const

## 6.5.1 Detailed Description

class to compute the kernel matrix

## 6.5.2 Constructor & Destructor Documentation

## 6.5.2.1 Kernel()

#### constructor

#### **Parameters**

id	name of the kernel function
epsilon	value of the bandwidth parameter epsilon

## 6.5.3 Member Function Documentation

#### 6.5.3.1 build\_kernel()

build the "star" version of the kernel that contains the standardized kernel weights in such a way that each row sums to one

## **Parameters**

data	a shared pointer to the matrix with the coordinates of the original dataset
anchorpoints	a shared pointer to the matrix with the coordinates of the anchor points

## 6.5.3.2 build\_simple\_kernel() [1/2]

build the "standard" version of the kernel needed for smoothing

#### **Parameters**

coordinates	a shared pointer to the matrix with the coordinates
-------------	---

## 6.5.3.3 build\_simple\_kernel() [2/2]

build the "standard" version of the kernel needed for smoothing

#### **Parameters**

coordinates	a shared pointer to the matrix with the coordinates	
epsilon replace the old epsilon with a new value		

## 6.5.3.4 get\_kernel()

```
\verb|const| \verb|matrixptr| LocallyStationaryModels::Kernel::get\_kernel () | const| \\
```

## Returns

a shared pointer to the matrix pointed by m\_k

## 6.5.3.5 operator()()

## Returns

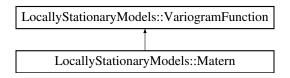
m\_f(x ,y) where m\_f is the kernel function

The documentation for this class was generated from the following files:

- · kernel.hpp
- · kernel.cpp

## 6.6 LocallyStationaryModels::Matern Class Reference

Inheritance diagram for LocallyStationaryModels::Matern:



#### **Public Member Functions**

• double operator() (const cd::vector &params, const double &x, const double &y) override

#### **Additional Inherited Members**

#### 6.6.1 Member Function Documentation

#### 6.6.1.1 operator()()

## Returns

```
sigma * sigma *(1 - std::pow(std::sqrt(2*nu)*h, nu)*std::cyl_bessel_k(nu, std::sqrt(2*nu)*h)/(std\leftrightarrow::tgamma(nu)*std::pow(2,nu-1)))
```

## Parameters

```
params a vector with lambda1, lambda2, phi, sigma and nu in this exact order
```

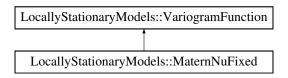
Implements LocallyStationaryModels::VariogramFunction.

The documentation for this class was generated from the following files:

- · variogramfunctions.hpp
- · variogramfunctions.cpp

## 6.7 LocallyStationaryModels::MaternNuFixed Class Reference

Inheritance diagram for LocallyStationaryModels::MaternNuFixed:



## **Public Member Functions**

- MaternNuFixed (const double &nu)
- double operator() (const cd::vector &params, const double &x, const double &y) override

### **Additional Inherited Members**

#### 6.7.1 Member Function Documentation

## 6.7.1.1 operator()()

#### Returns

```
sigma * sigma *(1 - std::pow(std::sqrt(2*nu)*h, nu)*std::cyl_bessel_k(nu, std::sqrt(2*nu)*h)/(std\leftrightarrow::tgamma(nu)*std::pow(2,nu-1)))
```

#### **Parameters**

```
params a vector with lambda1, lambda2, phi and sigma in this exact order
```

 $Implements\ Locally Stationary Models:: Variogram Function.$ 

The documentation for this class was generated from the following files:

- · variogramfunctions.hpp
- · variogramfunctions.cpp

## 6.8 LocallyStationaryModels::Opt Class Reference

a class to estimate the value of the parameters of the variogram in each point by optimizing the correspondent funzionedaottimizzare relying on the library LBFGSpp

```
#include <variogramfit.hpp>
```

## **Public Member Functions**

 Opt (const cd::matrixptr &empiricvariogram, const cd::matrixptr &squaredweights, const cd::vectorptr &mean\_x, const cd::vectorptr &mean\_y, const std::string &id, const cd::vector &initialparameters, const cd::vector &lowerbound, const cd::vector &upperbound)

constructor

• void findallsolutions ()

find the optimal solution in all the position

cd::matrixptr get solutions () const

## 6.8.1 Detailed Description

a class to estimate the value of the parameters of the variogram in each point by optimizing the correspondent funzionedaottimizzare relying on the library LBFGSpp

#### 6.8.2 Constructor & Destructor Documentation

## 6.8.2.1 Opt()

#### constructor

#### **Parameters**

empiricvariogram	a shared pointer to the empiric variogram
squaredweights	a shared pointer to the squared weights
mean_x	a shared pointer to the vector of the abscissas of the centers
mean_y	a shared pointer to the vector of the ordinates of the centers
id	the name of the variogram of your choice
initialparameters	the initial value of the parameters required from the optimizer to start the search for a minimum
lowerbound	the lower bounds for the parameters in the nonlinear optimization problem
upperbound	the upper bounds for the parameters in the nonlinear optimization problem

#### 6.8.3 Member Function Documentation

#### 6.8.3.1 get\_solutions()

```
cd::matrixptr LocallyStationaryModels::Opt::get_solutions ( ) const
```

#### Returns

the solutions found by solving the problem of nonlinear optimization

The documentation for this class was generated from the following files:

- · variogramfit.hpp
- · variogramfit.cpp

## 6.9 LocallyStationaryModels::Predictor Class Reference

class to perform kriging on the data

```
#include <kriging.hpp>
```

## **Public Member Functions**

Predictor (const std::string &id, const cd::vectorptr &z, const Smt &mysmt, const double &b, const cd
 ::matrixptr &data)

constructor

• Predictor ()

gammaiso set by default to exponential

template<typename Input , typename Output >

Output predict\_mean (const Input &pos) const

predict the mean

• template<typename Input , typename Output >

Output predict\_z (const Input &pos) const

predict Z

template<> double predict\_mean (const size\_t &pos) const

## 6.9.1 Detailed Description

class to perform kriging on the data

#### 6.9.2 Constructor & Destructor Documentation

## 6.9.2.1 Predictor()

constructor

#### **Parameters**

id	name of the variogram function associated with the problem
Z	the vector with the value of the function Y in the known points
mysmt	the one used to previously smooth the variogram
b	the radius of the neighbourhood of the point where to perform kriging
data	a shared pointer to the matrix with the coordinates of the original dataset

The documentation for this class was generated from the following files:

- · kriging.hpp
- · kriging.cpp

## 6.10 LocallyStationaryModels::SampleVar Class Reference

a class to build and store the empiric variogram in all the anchor points

```
#include <samplevar.hpp>
```

#### **Public Member Functions**

 SampleVar (const std::string &kernel\_id, const size\_t &n\_angles, const size\_t &n\_intervals, const double &epsilon)

constructor

· SampleVar ()

a default constructor for the class which calls the default constructors for both the kernel and the grid

- void build\_samplevar (const cd::matrixptr &data, const cd::matrixptr &anchorpoints, const cd::vectorptr &z)
  - build the matrix of the empiric variogram
- const cd::matrixptr get\_variogram () const
- const cd::matrixptr get\_denominators () const
- const cd::matrixptr get\_squaredweights () const
- const cd::vectorptr get\_x () const
- const cd::vectorptr get\_y () const
- const cd::matrixptr get\_kernel () const
- const cd::matrixIptr get\_grid () const
- const cd::vectorptr get\_normh () const

## 6.10.1 Detailed Description

a class to build and store the empiric variogram in all the anchor points

## 6.10.2 Constructor & Destructor Documentation

## 6.10.2.1 SampleVar()

constructor

#### **Parameters**

kernel_id the name of the function you want to use for the	
n_angles the number of angles to be passed to the grid	
n_intervals the number of inervals to be passed to the grid	
epsilon the bandwidth parameter regulating the kernel	

## 6.10.3 Member Function Documentation

## 6.10.3.1 build\_samplevar()

build the matrix of the empiric variogram

#### **Parameters**

data	a shared pointer to the matrix of the coordinates of the original dataset
anchorpoints	a shared pointer to the matrix of the coordinates of the anchor poitns
Z	a shared pointer to the vector of the value of Z

## 6.10.3.2 get\_denominators()

```
\verb|const| \verb| matrixptr| LocallyStationaryModels::SampleVar::get_denominators () | const| \\
```

#### Returns

a shared pointer to the matrix of the denominators

## 6.10.3.3 get\_grid()

```
const matrixIptr LocallyStationaryModels::SampleVar::get_grid ( ) const
```

### Returns

m\_grid.m\_g

## 6.10.3.4 get\_kernel()

```
\verb|const| matrixptr| LocallyStationaryModels::SampleVar::get\_kernel () const|
```

#### Returns

m\_kernel.m\_k

## 6.10.3.5 get\_normh()

```
const vectorptr LocallyStationaryModels::SampleVar::get_normh ( ) const
```

#### Returns

m\_grid.m\_normh

## 6.10.3.6 get\_squaredweights()

const matrixptr LocallyStationaryModels::SampleVar::get\_squaredweights ( ) const

## Returns

a shared pointers to the squaredweigths required to evaluate the function to be optimized

## 6.10.3.7 get\_variogram()

```
\verb|const| \verb|matrixptr| LocallyStationaryModels::SampleVar::get\_variogram () const|
```

#### Returns

a shared pointer to the sample variogram

## 6.10.3.8 get\_x()

```
const vectorptr LocallyStationaryModels::SampleVar::get_x ( ) const
```

## Returns

 $m\_grid.m\_mean\_x$ 

#### 6.10.3.9 get\_y()

```
const vectorptr LocallyStationaryModels::SampleVar::get_y ( ) const

Returns
    m_grid.m_mean_y
```

The documentation for this class was generated from the following files:

- · samplevar.hpp
- · samplevar.cpp

## 6.11 LocallyStationaryModels::Smt Class Reference

a class to perform kernel smoothing of the paramters estimated in the anchor points to get the non stationary value of the parameters in any position of the domain

```
#include <smooth.hpp>
```

#### **Public Member Functions**

 Smt (const cd::matrixptr &solutions, const cd::matrixptr &anchorpos, const double &min\_delta, const double &max\_delta, const std::string &kernel\_id)

constructor

 Smt (const cd::matrixptr &solutions, const cd::matrixptr &anchorpos, const double delta, const std::string &kernel id)

constructor

• Smt ()

constructor. Call the default constructor for m\_kernel

template<class Input >

cd::vector smooth\_vector (const Input &pos) const

smooth all the parameters for a point in position pos

- const cd::matrixptr get\_solutions () const
- · double get\_optimal\_delta () const
- const cd::matrixptr get\_anchorpos () const

## 6.11.1 Detailed Description

a class to perform kernel smoothing of the paramters estimated in the anchor points to get the non stationary value of the parameters in any position of the domain

### 6.11.2 Constructor & Destructor Documentation

### 6.11.2.1 Smt() [1/2]

constructor

#### **Parameters**

solutions	a shared pointer to the solutions of the optimization
anchorpos	a vector containing the indeces of the anchor position obtained by clustering
d	a shared pointer to the matrix of the coordinates
min_delta	the minimum exponent for the cross-validation of the delta bandwidth parameter for gaussian kernel smoothing
max_delta	the maximum exponent for the cross-validation of the delta bandwidth parameter for gaussian kernel smoothing

## 6.11.2.2 Smt() [2/2]

#### constructor

#### **Parameters**

solutions	a shared pointer to the solutions of the optimization
anchorpos	a vector containing the indeces of the anchor position obtained by clustering
d	a shared pointer to the matrix of the coordinates
delta	a user-chosen value for delta

## 6.11.3 Member Function Documentation

## 6.11.3.1 get\_anchorpos()

```
const cd::matrixptr LocallyStationaryModels::Smt::get_anchorpos ( ) const
```

## Returns

a shared pointer the coordinates of the anchorpoints

## 6.11.3.2 get\_optimal\_delta()

```
double LocallyStationaryModels::Smt::get_optimal_delta ( ) const
```

## Returns

the delta found by cross-validation evaluated on sigma, the same delta is used for all the parameters

#### 6.11.3.3 get\_solutions()

```
const cd::matrixptr LocallyStationaryModels::Smt::get_solutions ( ) const
```

#### Returns

a shared pointer to the solutions found by the optimizer

## 6.11.3.4 smooth\_vector()

smooth all the parameters for a point in position pos

#### **Parameters**

pos a vector of coordinates or the index of the position of the point where to find the smoothed value of the parameters

The documentation for this class was generated from the following files:

- · smooth.hpp
- · smooth.cpp

## 6.12 LocallyStationaryModels::TargetFunction Struct Reference

functor to pass to the optimizer that contains the wls to be minimized

```
#include <variogramfit.hpp>
```

## **Public Member Functions**

- TargetFunction (const cd::matrixptr &empiricvariogram, const cd::matrixptr &squaredweights, const cd
  ::vectorptr &mean\_x, const cd::vectorptr &mean\_y, const size\_t &x0, const std::string &id)
  - constructor
- double operator() (const cd::vector &params, cd::vector &grad)
- double operator() (const cd::vector &params)

#### **Public Attributes**

· const cd::matrixptr m\_empiricvariogram

sample variogram matrix

const cd::matrixptr m\_squaredweights

matrix of the squared weights

· const cd::vectorptr m\_mean\_x

vector with the x of each cell of the grid (mean of the x of all the pairs inside)

· const cd::vectorptr m\_mean\_y

vector with the y of each cell of the grid (mean of the y of all the pairs inside)

· size\_t m\_x0

index of the position where to evaluate gammaisoptr

 $\bullet \quad \text{std::shared\_ptr} < \textit{VariogramFunction} > \textit{m\_gammaisoptr} \\$ 

pointer to the variogram function

### 6.12.1 Detailed Description

functor to pass to the optimizer that contains the wls to be minimized

#### 6.12.2 Constructor & Destructor Documentation

#### 6.12.2.1 TargetFunction()

#### constructor

#### **Parameters**

empiricvariogram	a shared pointer to the empiric variogram
squaredweights	a shared pointer to the squared weights
mean_x	a shared pointer to the vector of the abscissas of the centers
mean_y	a shared pointer to the vector of the ordinates of the centers
хО	the index of the position x0
id	the name of the variogram of your choice

#### 6.12.3 Member Function Documentation

32 Class Documentation

#### 6.12.3.1 operator()() [1/2]

#### **Parameters**

params	a vector containing the previous value of the parameters of the function (lambda1, lambda2, phi,	
	sigma, etc.)	

#### 6.12.3.2 operator()() [2/2]

#### **Parameters**

params	a vector containing the previous value of the parameters of the function (lambda1, lambda2, phi, sigma, etc.)
grad	a vector containing the previous value of the gradient which is updated at each iteration

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

- · variogramfit.hpp
- · variogramfit.cpp

## 6.13 LocallyStationaryModels::Tolerances Struct Reference

collects all the tolerances and the constants used inside the code

```
#include <tolerances.hpp>
```

#### **Static Public Attributes**

• static constexpr double anchor\_tolerance = 1e-6

value of the noise to be added to the anchor points grid to prevent out of domain points

• static double **pi** = 4 \* std::atan(1.)

default value of pi

• static constexpr double min\_determinant = 1e-12

minimum threshold below which the determinant of a matrix is considered to be 0

static constexpr double param\_epsilon = 1e-6

optimization termination condition parameter epsilon

• static constexpr double **param\_max\_iterations** = 1000000

optimization termination condition parameter max\_iterations

- static constexpr double min\_norm = 1e-12
  - minimun threshold below which the norm of a vector is considered to be 0
- static constexpr double infinity = 1e12
  - huge value to be considered as infinite when returning inf would cause troubles
- static constexpr double n\_deltas = 1000
  - number of delta between min\_delta and max\_delta to perform cross-validation
- static constexpr double gradient\_step = 10e-8
  - step for the numerical computation of the gradient

#### 6.13.1 Detailed Description

collects all the tolerances and the constants used inside the code

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

· tolerances.hpp

## 6.14 LocallyStationaryModels::VariogramFunction Class Reference

Inheritance diagram for LocallyStationaryModels::VariogramFunction:



#### **Public Member Functions**

virtual double operator() (const cd::vector &params, const double &x, const double &y)=0
 return f(params, x, y)

#### **Protected Member Functions**

• double **compute\_anisotropic\_h** (const double &lambda1, const double &lambda2, const double &phi, const double &x, const double &y)

convert the isotropic variogram in the equivalent anisotropic one calculating the norm of the spatial lag rotated and expanded according to the eigenvalues and eigenvector of the anisotropy matrix

#### 6.14.1 Member Function Documentation

34 Class Documentation

#### 6.14.1.1 operator()()

return f(params, x, y)

 $Implemented \ in \ Locally Stationary Models:: Exponential, \ Locally Stationary Models:: Matern, \ Locally Stationary Models:: Matern Nu Fixed, \ and \ Locally Stationary Models:: Gaussian.$ 

The documentation for this class was generated from the following files:

- · variogramfunctions.hpp
- · variogramfunctions.cpp

## **Chapter 7**

## **File Documentation**

## 7.1 anchor.hpp

```
1 // Copyright (C) Luca Crippa <luca7.crippa@mail.polimi.it>
2 // Copyright (C) Giacomo De Carlo <giacomo.decarlo@mail.polimi.it>
4 #ifndef LOCALLY_STATIONARY_MODELS_ANCHOR
5 #define LOCALLY_STATIONARY_MODELS_ANCHOR
7 #include "traits.hpp"
9 namespace LocallyStationaryModels {
13 class Anchor {
14 private:
        cd::matrixptr m_data;
1.5
16
        double m_n_pieces;
        double m_width = 0;
        double m_height = 0;
19
         double m_piece_width = 0;
2.0
        double m_piece_height = 0;
21
        double m_origin_x = 0;
22
        double m_origin_y = 0;
27
         Eigen::VectorXi find_indeces()
2.8
29
              size_t n = m_data->rows();
30
             m_origin_x = (m_data->col(0)).minCoeff() * (1 - Tolerances::anchor_tolerance);
m_origin_y = (m_data->col(1)).minCoeff() * (1 - Tolerances::anchor_tolerance);
31
32
             m_width = (m_data->col(0)).maxCoeff() * (1 + Tolerances::anchor_tolerance) - m_origin_x;
m_height = (m_data->col(1)).maxCoeff() * (1 + Tolerances::anchor_tolerance) - m_origin_y;
34
35
             m_piece_width = m_width / m_n_pieces;
m_piece_height = m_height / m_n_pieces;
36
37
38
39
              // fill a vector with the position of each point
              Eigen::VectorXi result(n);
41
              for (size_t i = 0; i < n; ++i) {</pre>
42
                   \verb"cd::vector coordinates = \verb"m_data->row"(i)";
                   result(i) = ceil((coordinates(0) - m_origin_x) / m_piece_width)
43
44
                        + m_n_pieces * floor((coordinates(1) - m_origin_y) / m_piece_height);
46
              return result;
47
48
49 public:
        Anchor(const cd::matrixptr& data, const double& n_pieces)
55
             : m_data(data)
56
             , m_n_pieces(n_pieces) {};
58
63
         const cd::matrix find_anchorpoints()
64
             size_t n = m_data->rows();
Eigen::VectorXi indeces = find_indeces();
65
68
             // build a new vector without duplicates
69
             std::vector<size_t> positions;
             for (size_t i = 0; i < n; ++i) {
    size_t pos = indeces(i);
    if (std::find(positions.begin(), positions.end(), pos) == positions.end())</pre>
70
71
72
                        positions.push_back(pos);
```

```
}
75
76
           // fill a new matrix with the coordinates of each anchorpoins
77
           cd::matrix anchorpos(positions.size(), m_data->cols());
78
           for (size_t i = 0; i < anchorpos.rows(); ++i) {
    size_t I = positions[i];</pre>
79
                anchorpos(i, 0) = m_origin_x
80
81
                    + (I - floor((I * (Î - Tolerances::anchor_tolerance)) / m_n_pieces) * m_n_pieces) *
82
                    - m_piece_width / 2;
                anchorpos(i, 1) = m_origin_y + ceil((I * (1 - Tolerances::anchor_tolerance)) / m_n_pieces) *
83
       m_piece_height
84
                    - m_piece_height / 2;
85
86
           return anchorpos;
87
88
       std::pair<double, double> get_origin() const { return std::make_pair(m_origin_x, m_origin_y); }
92
       std::pair<double, double> get_tiles_dimensions() const { return std::make_pair(m_piece_width,
96
       m_piece_height); }
     // class Anchor
98 } // namespace LocallyStationaryModels
99
100 #endif // LOCALLY_STATIONARY_MODELS_ANCHOR
```

## 7.2 grid.hpp

```
1 // Copyright (C) Luca Crippa <luca7.crippa@mail.polimi.it>
2 // Copyright (C) Giacomo De Carlo <giacomo.decarlo@mail.polimi.it>
4 #ifndef LOCALLY_STATIONARY_MODELS_GRID
5 #define LOCALLY_STATIONARY_MODELS_GRID
7 #include "traits.hpp"
9 namespace LocallyStationaryModels {
13 class Grid {
14 private:
      cd::gridfunction m_f;
15
       cd::matrixIptr m_g = std::make_shared<cd::matrixI>(0, 0);
      cd::vectorptr m_normh
17
1.8
          = nullptr;
19
      cd::vectorptr m_mean_x
20
          = nullptr;
21
      cd::vectorptr m_mean_y
           = nullptr;
      double m_epsilon;
24
30
      void build_normh(const cd::matrixptr& data);
31
32 public:
      Grid(const std::string& id, const double& epsilon);
39
43
44
      void build_grid(const cd::matrixptr& data, const size_t& n_angles, const size_t& n_intervals);
51
52
      const cd::matrixIptr get_grid() const;
56
       const cd::vectorptr get_normh() const;
       const cd::vectorptr get_x() const;
68
       const cd::vectorptr get_y() const;
69 }; // class Grid
70 } // namespace LocallyStationaryModels
72 #endif // LOCALLY_STATIONARY_MODELS_GRID
```

## 7.3 gridfunctions.hpp

```
1 // Copyright (C) Luca Crippa <luca7.crippa@mail.polimi.it>
2 // Copyright (C) Giacomo De Carlo <giacomo.decarlo@mail.polimi.it>
3
4 #ifndef LOCALLY_STATIONARY_MODELS_GRID_FUNCTIONS
5 #define LOCALLY_STATIONARY_MODELS_GRID_FUNCTIONS
6
7 #include "traits.hpp"
8
9 namespace LocallyStationaryModels {
14 namespace gf {
23    cd::matrixIptr pizza(
```

7.4 kernel.hpp 37

## 7.4 kernel.hpp

```
1 // Copyright (C) Luca Crippa <luca7.crippa@mail.polimi.it>
2 // Copyright (C) Giacomo De Carlo <giacomo.decarlo@mail.polimi.it>
4 #ifndef LOCALLY_STATIONARY_MODELS_KERNEL
5 #define LOCALLY_STATIONARY_MODELS_KERNEL
7 #include "traits.hpp"
9 namespace LocallyStationaryModels {
13 class Kernel {
14 private:
1.5
       double m_epsilon;
16
       cd::kernelfunction m f;
      cd::matrixptr m_k = std::make_shared<cd::matrix>(0, 0);
18
19 public:
25
       Kernel(const std::string& id, const double& epsilon);
26
30
      Kernel();
31
       double operator()(const cd::vector& x, const cd::vector& y) const;
36
43
       void build_kernel(const cd::matrixptr& data, const cd::matrixptr& anchorpoints);
44
       void build_simple_kernel(const cd::matrixptr& coordinates);
49
50
56
       void build_simple_kernel(const cd::matrixptr& coordinates, const double& epsilon);
61
       const cd::matrixptr get_kernel() const;
62 }; // class Kernel
63 } // namespace LocallyStationaryModels
65 #endif // LOCALLY_STATIONARY_MODELS_KERNEL
```

## 7.5 kernelfunctions.hpp

```
1 // Copyright (C) Luca Crippa <luca7.crippa@mail.polimi.it>
2 // Copyright (C) Giacomo De Carlo <giacomo.decarlo@mail.polimi.it>
4 #ifndef LOCALLY_STATIONARY_MODELS_KERNEL_FUNCTIONS
5 #define LOCALLY_STATIONARY_MODELS_KERNEL_FUNCTIONS
7 #include "traits.hpp"
9 namespace LocallyStationaryModels {
14 namespace kf {
18
       double gaussian(const cd::vector& x, const cd::vector& y, const double& epsilon);
19
2.3
       double identity(const cd::vector& x, const cd::vector& y, const double& epsilon);
24
29
      cd::kernelfunction make_kernel(const std::string& id);
30 } // namespace kf
31 } // namespace LocallyStationaryModels
33 #endif // LOCALLY_STATIONARY_MODELS_KERNEL_FUNCTION
```

## 7.6 kriging.hpp

```
1 // Copyright (C) Luca Crippa <luca7.crippa@mail.polimi.it>
2 // Copyright (C) Giacomo De Carlo <giacomo.decarlo@mail.polimi.it>
3
4 #ifndef LOCALLY_STATIONARY_MODELS_KRIGING
5 #define LOCALLY_STATIONARY_MODELS_KRIGING
6
```

```
7 #include "smooth.hpp"
8 #include "traits.hpp"
9 #include "variogramfit.hpp"
1.0
11 namespace LocallyStationaryModels {
15 class Predictor {
16 private:
       std::shared_ptr<VariogramFunction> m_gammaisoptr;
17
18
       cd::vectorptr m_z = nullptr;
19
       Smt m smt;
20
       double m_b;
21
       cd::vectorptr m means = nullptr;
       cd::matrixptr m_data = nullptr;
22
23
28
       cd::vectorind build_neighbourhood(const cd::vector& pos) const;
29
       cd::vectorind build_neighbourhood(const size_t& pos) const;
30
       cd::vector build eta(cd::vector& params, cd::vectorind& neighbourhood) const;
36
37
43
       std::pair<cd::vector, double> build_etakriging(const cd::vector& params, const cd::vector& pos)
44
45 public:
       Predictor(
54
55
           const std::string& id, const cd::vectorptr& z, const Smt& mysmt, const double& b, const
       cd::matrixptr& data);
59
60
       template <typename Input, typename Output> Output predict_mean(const Input& pos) const;
64
65
       template <typename Input, typename Output> Output predict_z(const Input& pos) const;
69
70 }; // class Predictor
71 } // namespace LocallyStationaryModels
73 #endif // LOCALLY_STATIONARY_MODELS_KRIGING
```

## 7.7 samplevar.hpp

```
1 // Copyright (C) Luca Crippa <luca7.crippa@mail.polimi.it>
2 // Copyright (C) Giacomo De Carlo <giacomo.decarlo@mail.polimi.it>
4 #ifndef LOCALLY_STATIONARY_MODELS_SAMPLEVAR
5 #define LOCALLY STATIONARY MODELS SAMPLEVAR
7 #include "grid.hpp"
8 #include "kernel.hpp"
9 #include "traits.hpp"
10
11 namespace LocallyStationaryModels {
15 class SampleVar {
16 private:
       cd::matrixptr m_variogram = nullptr;
18
       cd::matrixptr m_denominators = nullptr;
19
       cd::matrixptr m_squaredweights = nullptr;
2.0
       Kernel m_kernel;
21
       Grid m_grid;
       size_t m_n_angles;
22
23
       size_t m_n_intervals;
24
28
       void build_squaredweights();
29
30 public:
       SampleVar(const std::string& kernel id, const size t& n angles, const size t& n intervals, const
38
       double& epsilon);
39
43
       SampleVar();
44
51
       void build_samplevar(const cd::matrixptr& data, const cd::matrixptr& anchorpoints, const
       cd::vectorptr& z);
52
56
       const cd::matrixptr get_variogram() const;
60
       const cd::matrixptr get_denominators() const;
64
       const cd::matrixptr get_squaredweights() const;
68
       const cd::vectorptr get_x() const;
72
       const cd::vectorptr get_y() const;
const cd::matrixptr get_kernel() const;
76
       const cd::matrixIptr get_grid() const;
       const cd::vectorptr get_normh() const;
85 }; // class SampleVar
86 } // namespace LocallyStationaryModels
87
88 #endif // LOCALLY_STATIONARY_MODELS_SAMPLEVAR
```

7.8 smooth.hpp 39

### 7.8 smooth.hpp

```
1 // Copyright (C) Luca Crippa <luca7.crippa@mail.polimi.it>
2 // Copyright (C) Giacomo De Carlo <giacomo.decarlo@mail.polimi.it>
4 #ifndef LOCALLY_STATIONARY_MODELS_SMOOTH
5 #define LOCALLY_STATIONARY_MODELS_SMOOTH
7 #include "kernel.hpp"
8 #include "traits.hpp"
10 namespace LocallyStationaryModels {
15 class Smt {
       cd::matrixptr m_solutions = nullptr;
17
18
       cd::matrixptr m_anchorpos = nullptr;
19
2.0
       Kernel m kernel;
21
22
       double m_optimal_delta = 0;
23
29
       double smooth_value(const size_t& pos, const size_t& n) const;
30
       double smooth_value(const cd::vector& pos, const size_t& n) const;
37
38
39 public:
49
       Smt(const cd::matrixptr& solutions, const cd::matrixptr& anchorpos, const double& min_delta,
50
            const double& max_delta, const std::string& kernel_id);
5.8
       Smt(const cd::matrixptr& solutions, const cd::matrixptr& anchorpos, const double delta,
59
           const std::string& kernel_id);
       Smt();
63
70
       template <class Input> cd::vector smooth_vector(const Input& pos) const
71
72
            cd::vector result(m_solutions->cols());
           for (size_t i = 0; i < m_solutions->cols(); ++i) {
    result(i) = smooth_value(pos, i);
7.3
74
75
76
           return result;
77
       } ;
78
82
       const cd::matrixptr get_solutions() const;
       double get_optimal_delta() const;
86
       const cd::matrixptr get_anchorpos() const;
90
91 }; // class Smt
92 } // namespace LocallyStationaryModels
94 #endif // LOCALLY STATIONARY MODELS SMOOTH
```

### 7.9 tolerances.hpp

```
1 // Copyright (C) Luca Crippa <luca7.crippa@mail.polimi.it>
2 // Copyright (C) Giacomo De Carlo <giacomo.decarlo@mail.polimi.it>
4 #ifndef LOCALLY_STATIONARY_MODELS_TOLERANCES
5 #define LOCALLY_STATIONARY_MODELS_TOLERANCES
7 namespace LocallyStationaryModels {
11 struct Tolerances {
      static constexpr double anchor_tolerance = 1e-6;
13
      inline static double pi = 4 * std::atan(1.);
15
      static constexpr double min_determinant = 1e-12;
      static constexpr double param_epsilon = 1e-6;
19
21
       static constexpr double param_max_iterations = 1000000;
23
      static constexpr double min_norm = 1e-12;
      static constexpr double infinity = 1e12;
2.5
      static constexpr double n_deltas = 1000;
      static constexpr double gradient_step = 10e-8;
30 }; // struct Tolerances
31 } // namespace LocallyStationaryModels
33 #endif // LOCALLY STATIONARY MODELS TOLERANCES
```

## 7.10 traits.hpp

```
1 // Copyright (C) Luca Crippa <luca7.crippa@mail.polimi.it>
2 // Copyright (C) Giacomo De Carlo <giacomo.decarlo@mail.polimi.it>
3
4 #ifndef LOCALLY_STATIONARY_MODELS_TRAITS
```

```
5 #define LOCALLY_STATIONARY_MODELS_TRAITS
7 #include <algorithm>
8 #include <cfloat>
9 #include <cmath>
10 #include <functional>
11 #include <iostream>
12 #include <memory>
13 #include <omp.h>
14 #include <string>
15 #include <vector>
16
17 #include "Eigen/Dense"
18 #include "tolerances.hpp"
19
20 namespace LocallyStationaryModels {
25 namespace cd {
                      // defining basic types
26
                        using vector = Eigen::VectorXd;
                        using matrix = Eigen::MatrixXd;
29
                        using matrixI = Eigen::MatrixXi;
30
                        using vectorptr = std::shared_ptr<vector>;
                       using matrixptr = std::shared_ptr<matrix>;
using matrixIptr = std::shared_ptr<matrixI>;
31
32
33
                       using vectorind = std::vector<size_t>;
                        // defining function types
35
36
                        using kernelfunction = std::function<double(const vector&, const vector&, const double&)>;
37
                        using \ gridfunction = std::function < matrixIptr(const \ matrixptr\&, \ const \ size\_t\&, \ const \ size\_t\&
                        double&)>;
38
39
         } // namespace cd
40 } // namespace LocallyStationaryModels
42 #endif // LOCALLY_STATIONARY_MODELS_TRAITS
```

## 7.11 variogramfit.hpp

```
1 // Copyright (C) Luca Crippa <luca7.crippa@mail.polimi.it>
2 // Copyright (C) Giacomo De Carlo <giacomo.decarlo@mail.polimi.it>
4 #ifndef LOCALLY_STATIONARY_MODELS_GRADIENT
5 #define LOCALLY STATIONARY MODELS GRADIENT
7 #include "LBFGS/LBFGSB.h"
8 #include "traits.hpp"
9 #include "variogramfunctions.hpp"
10
11 namespace LocallyStationaryModels {
15 struct TargetFunction {
        const cd::matrixptr m_empiricvariogram;
16
17
        const cd::matrixptr m_squaredweights;
       const cd::vectorptr
18
19
           m_mean_x;
2.0
       const cd::vectorptr
21
           m_mean_y;
       size_t m_x0;
22
23
       std::shared_ptr<VariogramFunction> m_gammaisoptr;
24
34
        TargetFunction(const cd::matrixptr& empiricvariogram, const cd::matrixptr& squaredweights,
35
            const cd::vectorptr& mean_x, const cd::vectorptr& mean_y, const size_t& x0, const std::string&
36
42
        double operator()(const cd::vector& params, cd::vector& grad);
        double operator()(const cd::vector& params);
48
49 }; // struct TargetFunction
50
55 class Opt {
56 private:
        cd::matrixptr m_empiricvariogram;
5.8
        cd::matrixptr m_squaredweights;
59
        cd::vectorptr m_mean_x;
60
        cd::vectorptr m_mean_y;
61
       std::string m_id;
cd::vector m_initialparameters;
62
63
        cd::vector m_lowerbound;
        cd::vector m_upperbound;
6.5
        cd::matrixptr m_solutions = nullptr;
66
        cd::vector findonesolution(const size_t& pos) const;
71
73 public:
```

```
Opt(const cd::matrixptr& empiricvariogram, const cd::matrixptr& squaredweights, const cd::vectorptr& mean_x,

const cd::vectorptr& mean_y, const std::string& id, const cd::vector& initialparameters,

const cd::vector& lowerbound, const cd::vector& upperbound);

void findallsolutions();

cd::matrixptr get_solutions() const;

y; // class Opt

100 } // namespace LocallyStationaryModels

101

102 #endif // LOCALLY_STATIONARY_MODELS_GRADIENT
```

## 7.12 variogramfunctions.hpp

```
1 // Copyright (C) Luca Crippa <luca7.crippa@mail.polimi.it>
2 // Copyright (C) Giacomo De Carlo <giacomo.decarlo@mail.polimi.it>
4 #ifndef LOCALLY_STATIONARY_MODES_VARIOGRAMFUNCTIONS
5 #define LOCALLY_STATIONARY_MODES_VARIOGRAMFUNCTIONS
7 #include "traits.hpp"
9 namespace LocallyStationaryModels {
10 class VariogramFunction {
11 protected:
      double compute_anisotropic_h(
          const double& lambda1, const double& lambda2, const double& phi, const double& x, const double&
18
19 public:
     VariogramFunction() = default;
20
       virtual double operator() (const cd::vector& params, const double& x, const double& y) = 0;
25 }; // class VariogramFunction
26
27 class Exponential : public VariogramFunction {
28 public:
29
     Exponential() = default;
       double operator()(const cd::vector& params, const double& x, const double& y) override;
35 }; // class Exponential
37 class Matern : public VariogramFunction {
38 public:
39
      Matern() = default;
       double operator()(const cd::vector& params, const double& x, const double& y) override;
46 }; // class Matern
48 class MaternNuFixed : public VariogramFunction {
49 private:
      double m nu = 0.5;
50
51 public:
     MaternNuFixed(const double& nu)
53
           : m_nu(nu) {};
59
      double operator()(const cd::vector& params, const double& x, const double& y) override;
60 }; // class MaternNuFixed
61
62 class Gaussian : public VariogramFunction {
63 public:
      Gaussian() = default;
69
       double operator()(const cd::vector& params, const double& x, const double& y) override;
70 }; // class Gaussian
71
76 std::shared ptr<VariogramFunction> make variogramiso(const std::string& id);
77 } // namespace LocallyStationaryModels
79 #endif // LOCALLY_STATIONARY_MODES_VARIOGRAM_FUNCTIONS
```

# Index

Anchor	LocallyStationaryModels::kf, 11
LocallyStationaryModels::Anchor, 13	
	Kernel
build_grid	LocallyStationaryModels::Kernel, 19
LocallyStationaryModels::Grid, 17	
build_kernel	LocallyStationaryModels::Anchor, 13
LocallyStationaryModels::Kernel, 19	Anchor, 13
build_samplevar	get_origin, 14
LocallyStationaryModels::SampleVar, 26	get_tiles_dimensions, 14
build_simple_kernel	LocallyStationaryModels::cd, 9
LocallyStationaryModels::Kernel, 19, 20	LocallyStationaryModels::Exponential, 14
	operator(), 15
gaussian	LocallyStationaryModels::Gaussian, 15
LocallyStationaryModels::kf, 11	operator(), 15
get_anchorpos	LocallyStationaryModels::gf, 9
LocallyStationaryModels::Smt, 29	make_grid, 10
get_denominators	pizza, 10
LocallyStationaryModels::SampleVar, 26	LocallyStationaryModels::Grid, 16
get_grid	build_grid, 17
LocallyStationaryModels::Grid, 17	get_grid, 17
LocallyStationaryModels::SampleVar, 26	get_normh, 17
get_kernel	get_x, 18
LocallyStationaryModels::Kernel, 20	get_y, 18
LocallyStationaryModels::SampleVar, 26	Grid, 17
get_normh	LocallyStationaryModels::Kernel, 18
LocallyStationaryModels::Grid, 17	build_kernel, 19
LocallyStationaryModels::SampleVar, 27	build_simple_kernel, 19, 20
get_optimal_delta	get_kernel, 20
LocallyStationaryModels::Smt, 29	Kernel, 19
get_origin	operator(), 20
LocallyStationaryModels::Anchor, 14	LocallyStationaryModels::kf, 10
get_solutions	gaussian, 11
LocallyStationaryModels::Opt, 23	identity, 11
LocallyStationaryModels::Smt, 29	make_kernel, 11
get_squaredweights	LocallyStationaryModels::Matern, 21
LocallyStationaryModels::SampleVar, 27	operator(), 21
get_tiles_dimensions	LocallyStationaryModels::MaternNuFixed, 21
LocallyStationaryModels::Anchor, 14	operator(), 22
get_variogram	LocallyStationaryModels::Opt, 22
LocallyStationaryModels::SampleVar, 27	get_solutions, 23
get_x	Opt, 23
LocallyStationaryModels::Grid, 18	LocallyStationaryModels::Predictor, 24
LocallyStationaryModels::SampleVar, 27	Predictor, 24
	LocallyStationaryModels::SampleVar, 25
get_y LocallyStationaryModels::Grid, 18	build_samplevar, 26
LocallyStationaryModels::SampleVar, 27	get_denominators, 26
· · · · · · · · · · · · · · · · · · ·	get_grid, 26
Grid	get kernel, 26
LocallyStationaryModels::Grid, 17	get_kernel, 26 get_normh, 27
identity	get_normin, 27 get_squaredweights, 27
·················	yer squareuwelyilis, 4/

44 INDEX

```
get_variogram, 27
     get_x, 27
    get_y, 27
     SampleVar, 25
LocallyStationaryModels::Smt, 28
     get anchorpos, 29
     get_optimal_delta, 29
     get_solutions, 29
     smooth vector, 30
     Smt, 28, 29
LocallyStationaryModels::TargetFunction, 30
     operator(), 31, 32
     TargetFunction, 31
LocallyStationaryModels::Tolerances, 32
LocallyStationaryModels::VariogramFunction, 33
     operator(), 33
make_grid
     LocallyStationaryModels::gf, 10
make kernel
     LocallyStationaryModels::kf, 11
operator()
     LocallyStationaryModels::Exponential, 15
     LocallyStationaryModels::Gaussian, 15
     LocallyStationaryModels::Kernel, 20
     LocallyStationaryModels::Matern, 21
     LocallyStationaryModels::MaternNuFixed, 22
     LocallyStationaryModels::TargetFunction, 31, 32
     LocallyStationaryModels::VariogramFunction, 33
Opt
     LocallyStationaryModels::Opt, 23
pizza
     LocallyStationaryModels::gf, 10
Predictor
     LocallyStationaryModels::Predictor, 24
SampleVar
     LocallyStationaryModels::SampleVar, 25
smooth_vector
     LocallyStationaryModels::Smt, 30
Smt
     LocallyStationaryModels::Smt, 28, 29
TargetFunction
     LocallyStationaryModels::TargetFunction, 31
```