# My Project

Generated by Doxygen 1.9.4

# Chapter 1

# Hierarchical Index

## 1.1 Class Hierarchy

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

# 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

# Class Documentation

## 4.1 Area Class Reference

Class containing information about the given rectangular area.

`#include <Definitions.h>`

### Public Member Functions

- bool **is_in** (vector< double > x)

  *Answers the question "is the point x in this area?".*
- Area (vector< double > lefts, vector< double > rights)

  *Constructor.*

### Public Attributes

- int **dim**
- vector< double > **left_borders**
- vector< double > **right_borders**

### 4.1.1 Detailed Description

Class containing information about the given rectangular area.

### 4.1.2 Constructor & Destructor Documentation

#### 4.1.2.1 Area()

```
Area::Area (
            vector< double > lefts,
            vector< double > rights )  [inline]
```

Constructor.

**Parameters**

| | |
|---|---|
| *lefts* | contains left borders of the rectangular area |
| *rights* | contains right borders of the rectangular area |

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

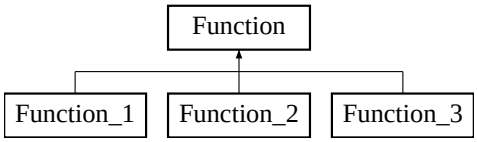
- Definitions.h
- Definitions.cpp

## 4.2 Function Class Reference

Class implementing computing of function's value, gradient and Gesse's matrix in the given point. @detailed There are three hardcoded functions. The user is choosing one of them, implementation involved polymorphism.

`#include <Functions.h>`

Inheritance diagram for Function:




### Public Member Functions

- virtual double eval (vector< double > x)

  *Computes function's value in the given point.*
- virtual vector< double > eval_gr (vector< double > x)

  *Computes function's gradient in the given point.*
- virtual vector< vector< double > > eval_Gesse (vector< double > x)

  *Computes function's Gesse's matrix in the given point.*

### 4.2.1 Detailed Description

Class implementing computing of function's value, gradient and Gesse's matrix in the given point. @detailed There are three hardcoded functions. The user is choosing one of them, implementation involved polymorphism.

### 4.2.2 Member Function Documentation

#### 4.2.2.1 eval()

```
double Function::eval (
            vector< double > x )  [virtual]
```

Computes function's value in the given point.

**Parameters**

| | |
|---|---|
| *x* | is the given point |

**Returns**

Function's value

Reimplemented in Function_1, Function_2, and Function_3.

#### 4.2.2.2 eval_Gesse()

```
vector< vector< double > > Function::eval_Gesse (
            vector< double > x )  [virtual]
```

Computes function's Gesse's matrix in the given point.

**Parameters**

| | |
|---|---|
| *x* | is the given point |

**Returns**

Function's Gesse's matrix

Reimplemented in Function_1, Function_2, and Function_3.

#### 4.2.2.3 eval_gr()

```
vector< double > Function::eval_gr (
            vector< double > x )  [virtual]
```

Computes function's gradient in the given point.

**Parameters**

| | |
|---|---|
| *x* | is the given point |

**Returns**

Function's gradient

Reimplemented in Function_1, Function_2, and Function_3.

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

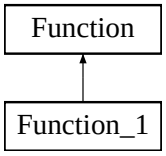
- Functions.h
- Functions.cpp

## 4.3 Function_1 Class Reference

f(x, y) = sin(x)∗cos(y)

`#include <Functions.h>`

Inheritance diagram for Function_1:




### Public Member Functions

- double eval (vector< double > x)

  *Computes function's value in the given point.*
- vector< double > eval_gr (vector< double > x)

  *Computes function's gradient in the given point.*
- vector< vector< double > > eval_Gesse (vector< double > x)

  *Computes function's Gesse's matrix in the given point.*

### 4.3.1 Detailed Description

f(x, y) = sin(x)∗cos(y)

### 4.3.2 Member Function Documentation

#### 4.3.2.1 eval()

```
double Function_1::eval (
            vector< double > x )  [inline], [virtual]
```

Computes function's value in the given point.

**Parameters**

| | |
|---|---|
| *x* | is the given point |

**Returns**

Function's value

Reimplemented from Function.

#### 4.3.2.2 eval_Gesse()

```
vector< vector< double > > Function_1::eval_Gesse (
            vector< double > x )  [virtual]
```

Computes function's Gesse's matrix in the given point.

**Parameters**

| | |
|---|---|
| *x* | is the given point |

**Returns**

Function's Gesse's matrix

Reimplemented from Function.

#### 4.3.2.3 eval_gr()

```
vector< double > Function_1::eval_gr (
            vector< double > x )  [virtual]
```

Computes function's gradient in the given point.

**Parameters**

| | |
|---|---|
| *x* | is the given point |

**Returns**

Function's gradient

Reimplemented from Function.

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

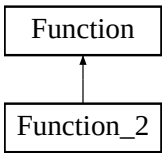
- Functions.h
- Functions.cpp

## 4.4 Function_2 Class Reference

f(x, y) = 20∗exp(sin((x+y)/20))+xy

`#include <Functions.h>`

Inheritance diagram for Function_2:




### Public Member Functions

- double eval (vector< double > x)

  *Computes function's value in the given point.*
- vector< double > eval_gr (vector< double > x)

  *Computes function's gradient in the given point.*
- vector< vector< double > > eval_Gesse (vector< double > x)

  *Computes function's Gesse's matrix in the given point.*

### 4.4.1 Detailed Description

f(x, y) = 20∗exp(sin((x+y)/20))+xy

### 4.4.2 Member Function Documentation

#### 4.4.2.1 eval()

```
double Function_2::eval (
            vector< double > x )  [inline], [virtual]
```

Computes function's value in the given point.

**Parameters**

| | |
|---|---|
| *x* | is the given point |

**Returns**

Function's value

Reimplemented from Function.

#### 4.4.2.2 eval_Gesse()

```
vector< vector< double > > Function_2::eval_Gesse (
            vector< double > x )  [virtual]
```

Computes function's Gesse's matrix in the given point.

**Parameters**

| | |
|---|---|
| *x* | is the given point |

**Returns**

Function's Gesse's matrix

Reimplemented from Function.

#### 4.4.2.3 eval_gr()

```
vector< double > Function_2::eval_gr (
            vector< double > x )  [virtual]
```

Computes function's gradient in the given point.

**Parameters**

| | |
|---|---|
| *x* | is the given point |

**Returns**

Function's gradient

Reimplemented from Function.

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

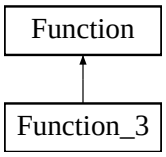
- Functions.h
- Functions.cpp

## 4.5 Function_3 Class Reference

f(x, y) = (1-x)^2 + 100(y-x^2)^2

`#include <Functions.h>`

Inheritance diagram for Function_3:




## Public Member Functions

- double eval (vector< double > x)

  *Computes function's value in the given point.*
- vector< double > eval_gr (vector< double > x)

  *Computes function's gradient in the given point.*
- vector< vector< double > > eval_Gesse (vector< double > x)

  *Computes function's Gesse's matrix in the given point.*

### 4.5.1 Detailed Description

f(x, y) = (1-x)^2 + 100(y-x^2)^2

### 4.5.2 Member Function Documentation

#### 4.5.2.1 eval()

```
double Function_3::eval (
            vector< double > x )  [inline], [virtual]
```

Computes function's value in the given point.

**Parameters**

| | |
|---|---|
| *x* | is the given point |

**Returns**

Function's value

Reimplemented from Function.

#### 4.5.2.2 eval_Gesse()

```
vector< vector< double > > Function_3::eval_Gesse (
            vector< double > x )  [virtual]
```

Computes function's Gesse's matrix in the given point.

**Parameters**

| | |
|---|---|
| *x* | is the given point |

**Returns**

Function's Gesse's matrix

Reimplemented from Function.

#### 4.5.2.3 eval_gr()

```
vector< double > Function_3::eval_gr (
            vector< double > x )  [virtual]
```

Computes function's gradient in the given point.

**Parameters**

| | |
|---|---|
| *x* | is the given point |

**Returns**

Function's gradient

Reimplemented from Function.

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

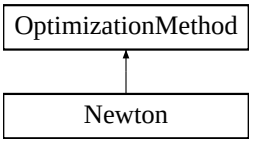
- Functions.h
- Functions.cpp

## 4.6 Newton Class Reference

Class containing Newton's optimization method @detailed That involves computing Gesse's matrixes and gradients.

```
#include <OptimizationMethods.h>
```

Inheritance diagram for Newton:




## Public Member Functions

- double get_alpha_alt (vector< double > x_n)

  *Function that computes the best step length in the direction p_n.*
- **Newton** (Function *F, vector< double > x_0_, vector< double > l_border, vector< double > r_border, int SC_var=1, double eps_=0.001)

  *Constructor @detailed See OptimizationMethod constructor for more.*
- vector< double > **iteration** (vector< double > &x_, double alpha)
- vector< double > optimize ()

  *Main function.*

## Public Attributes

- vector< double > **p_n**

### 4.6.1 Detailed Description

Class containing Newton's optimization method @detailed That involves computing Gesse's matrixes and gradients.

### 4.6.2 Member Function Documentation

#### 4.6.2.1 get_alpha_alt()

```
double Newton::get_alpha_alt (
            vector< double > x_n )
```

Function that computes the best step length in the direction p_n.

**Parameters**

| | |
|---|---|
| *x_n* | The point of current iteration |

**Returns**

The fraction of p_n with which the next iteration will be optimal

#### 4.6.2.2 optimize()

```
vector< double > Newton::optimize ( )  [virtual]
```

Main function.

**Returns**

The point of last iteration

Reimplemented from OptimizationMethod.

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

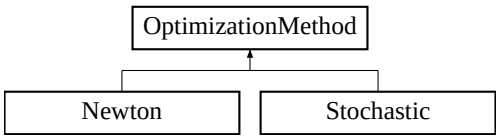
- OptimizationMethods.h
- OptimizationMethods.cpp

## 4.7 OptimizationMethod Class Reference

Optimizing class containing realization of determined or stochastic method. @detailed Different methods are being applied using polymorphism.

```
#include <OptimizationMethods.h>
```

Inheritance diagram for OptimizationMethod:




### Public Member Functions

- OptimizationMethod (Function *F, vector< double > x_0_, vector< double > l_border, vector< double > r_border, int SC_var=1, double eps=0.001)

  *Constructor.*
- virtual vector< double > optimize ()

  *Main function.*

### Public Attributes

- Area **D**
- Function * **f**
- StopCriterion * **SC**
- vector< double > **x_0**
- double **answer**

### 4.7.1 Detailed Description

Optimizing class containing realization of determined or stochastic method. @detailed Different methods are being applied using polymorphism.

### 4.7.2 Constructor & Destructor Documentation

#### 4.7.2.1 OptimizationMethod()

```
OptimizationMethod::OptimizationMethod (
            Function * F,
            vector< double > x_0_,
            vector< double > l_border,
            vector< double > r_border,
            int SC_var = 1,
            double eps = 0.001 )
```

Constructor.

**Parameters**

| | |
|---|---|
| *F* | is one of the hardcoded functions (see Functions.h for more) |
| *x_0_* | starting point |
| *SC_var* | current variant of stop criterion (see StopCriteria.h for more) |

### 4.7.3 Member Function Documentation

#### 4.7.3.1 optimize()

```
virtual vector< double > OptimizationMethod::optimize ( )  [inline], [virtual]
```

Main function.

**Returns**

The point of last iteration

Reimplemented in Newton, and Stochastic.

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

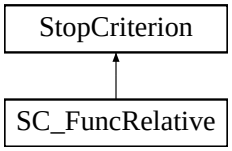
- OptimizationMethods.h
- OptimizationMethods.cpp

## 4.8 SC_FuncRelative Class Reference

Class that stops iterations if the absolute difference between current and previous values related to the current value is less than eps.

`#include <StopCriteria.h>`

Inheritance diagram for SC_FuncRelative:




### Additional Inherited Members

### 4.8.1 Detailed Description

Class that stops iterations if the absolute difference between current and previous values related to the current value is less than eps.

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

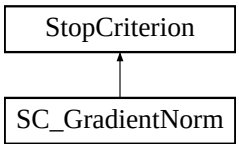
- StopCriteria.h
- StopCriteria.cpp

## 4.9 SC_GradientNorm Class Reference

Class that stops iterations if the norm of the gradient in the current point is less than eps.

`#include <StopCriteria.h>`

Inheritance diagram for SC_GradientNorm:




### Additional Inherited Members

### 4.9.1 Detailed Description

Class that stops iterations if the norm of the gradient in the current point is less than eps.

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

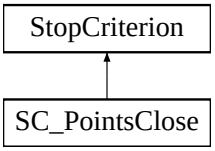
- StopCriteria.h
- StopCriteria.cpp

## 4.10 SC_PointsClose Class Reference

Class that stops iterations if the norm of the difference between current and previous points are less than eps.

`#include <StopCriteria.h>`

Inheritance diagram for SC_PointsClose:




### Additional Inherited Members

### 4.10.1 Detailed Description

Class that stops iterations if the norm of the difference between current and previous points are less than eps.

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

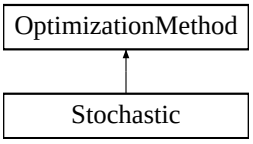
- StopCriteria.h
- StopCriteria.cpp

## 4.11 Stochastic Class Reference

Class containing stochastic optimization method @detailed That involves finding random points in general area and locally.

`#include <OptimizationMethods.h>`

Inheritance diagram for Stochastic:




### Public Member Functions

- vector< double > iteration (vector< double > x, double &delta)

  *Function computing the next iteration.*
- vector< double > optimize ()

  *Main function.*
- Stochastic (Function *F, vector< double > x_0_, vector< double > l_border, vector< double > r_border, int SC_var=1, double eps_=0.001, double alpha_=0.2, double p_=0.5)

  *Constructor.*

## Public Attributes

- double **alpha**
- double **p**

### 4.11.1 Detailed Description

Class containing stochastic optimization method @detailed That involves finding random points in general area and locally.

### 4.11.2 Constructor & Destructor Documentation

#### 4.11.2.1 Stochastic()

```
Stochastic::Stochastic (
            Function * F,
            vector< double > x_0_,
            vector< double > l_border,
            vector< double > r_border,
            int SC_var = 1,
            double eps_ = 0.001,
            double alpha_ = 0.2,
            double p_ = 0.5 )  [inline]
```

Constructor.

**Parameters**

| | |
|---|---|
| *alpha* | is delta's iterational multiplier @detailed See OptimizationMethod constructor for more |

### 4.11.3 Member Function Documentation

#### 4.11.3.1 iteration()

```
vector< double > Stochastic::iteration (
            vector< double > x,
            double & delta )
```

Function computing the next iteration.

**Parameters**

| | |
|---|---|
| *delta* | is the local square's rib length. Where local square is an area in which the local search will be done with the probability p |

#### 4.11.3.2 optimize()

```
vector< double > Stochastic::optimize ( )  [virtual]
```

Main function.

**Returns**

The point of last iteration

Reimplemented from OptimizationMethod.

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

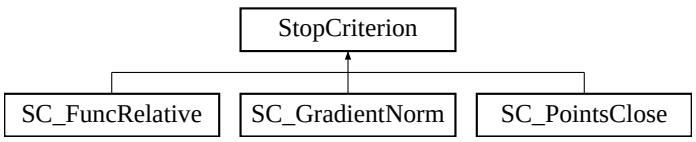
- OptimizationMethods.h
- OptimizationMethods.cpp

## 4.12 StopCriterion Class Reference

Class containing a criterion with which iterations are being stopped. @detailed There are three of them. They are implemented using polymorphism.

```
#include <StopCriteria.h>
```

Inheritance diagram for StopCriterion:




### Public Member Functions

- virtual bool do_we_stop (vector< vector< double > > &a, Function *F, int iteration_number)

  *Main function.*
- StopCriterion (double ep=0.001, int bo=1000)

  *Constructor.*

## Public Attributes

- double **eps**

## Protected Attributes

- int **bound**

### 4.12.1 Detailed Description

Class containing a criterion with which iterations are being stopped. @detailed There are three of them. They are implemented using polymorphism.

### 4.12.2 Constructor & Destructor Documentation

#### 4.12.2.1 StopCriterion()

```
StopCriterion::StopCriterion (
            double ep = 0.001,
            int bo = 1000 )  [inline]
```

Constructor.

**Parameters**

| | |
|---|---|
| *bo* | is the upper bound of iterations number |

### 4.12.3 Member Function Documentation

#### 4.12.3.1 do_we_stop()

```
virtual bool StopCriterion::do_we_stop (
            vector< vector< double > > & a,
            Function * F,
            int iteration_number )  [inline], [virtual]
```

Main function.

**Parameters**

| | |
|---|---|
| *a* | contains the trajectory of iterations |
| *F* | is a function that is being minimized |

**Returns**

true if iterations are to be stopped, false otherwise

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

- StopCriteria.h

# Chapter 5

# File Documentation

## 5.1 Definitions.h

```
1 #pragma once
2
3 #include <iostream>
4 #include "Eigen/QR"
5 #include <vector>
6 #include <cmath>
7 #include "R64M.hpp"
8 #include "Functions.h"
9
10 using namespace std;
14 vector<double> operator-(vector<double> x, vector<double> y);
15
16 vector<double> operator-(vector<double> x);
17
18 vector<double> operator+(vector<double> x, vector<double> y);
19
20 vector<double> operator*(vector<vector<double» a, vector<double> y);
21
22 vector<double> operator*(double c, vector<double> x);
23
27 double norm(vector<double> x);
31 double f_0(vector<double>& x);
32
36 class Area {
37 public:
38     int dim;
39     vector<double> left_borders;
40     vector<double> right_borders;
44     bool is_in(vector<double> x);
50     Area(vector<double> lefts, vector<double> rights) : left_borders(lefts), right_borders(rights),
51         dim(rights.size() == lefts.size() ? rights.size() : 0) {}
52 };
```

## 5.2 Functions.h

```
1 #pragma once
2
3 #include <vector>
4 #include <cmath>
5
6 using namespace std;
11 class Function {
12 public:
18     virtual double eval(vector<double> x);
24     virtual vector<double> eval_gr(vector<double> x);
30     virtual vector<vector<double» eval_Gesse(vector<double> x);
31 };
35 class Function_1 : public Function {
36 public:
42     double eval(vector<double> x) { return sin(x[0]) * cos(x[1]); };
48     vector<double> eval_gr(vector<double> x);
54     vector<vector<double» eval_Gesse(vector<double> x);
55 };
56
```

```
60 class Function_2 : public Function {
61 public:
67     double eval(vector<double> x) { return 20. * exp(sin((x[0] + x[1]) / 20.)) + x[0] * x[1]; };
73     vector<double> eval_gr(vector<double> x);
79     vector<vector<double>> eval_Gesse(vector<double> x);
80 };
81
82
86 class Function_3 : public Function {
87 public:
93     double eval(vector<double> x) { return (1. - x[0]) * (1. - x[0]) + 100. * (x[1] - x[0] * x[0]) *
       (x[1] - x[0] * x[0]); };
99     vector<double> eval_gr(vector<double> x);
105     vector<vector<double>> eval_Gesse(vector<double> x);
106 };
```

## 5.3 OptimizationMethods.h

```
1 #pragma once
2 #include "Definitions.h"
3 #include "StopCriteria.h"
9 class OptimizationMethod {
10 public:
11     Area D;
12     Function* f;
13     StopCriterion* SC;
14     vector<double> x_0;
15     double answer;
22     OptimizationMethod(Function* F, vector<double> x_0_, vector<double> l_border,
23         vector<double> r_border, int SC_var = 1, double eps = 0.001);
28     virtual vector<double> optimize() { return x_0; };
29 };
30
35 class Newton : public OptimizationMethod {
36 public:
37     vector<double> p_n;
43     double get_alpha_alt(vector<double> x_n);
48     Newton(Function* F, vector<double> x_0_, vector<double> l_border, vector<double> r_border, int SC_var
       = 1, double eps_ = 0.001) :p_n(), OptimizationMethod(F, x_0_, l_border, r_border, SC_var, eps_) {}
49     vector<double> iteration(vector<double>& x_, double alpha){return x_ + alpha * p_n;}
54     vector<double> optimize();
55 };
56
61 class Stochastic : public OptimizationMethod {
62 public:
63     double alpha;
64     double p;
69     vector<double> iteration(vector<double> x, double& delta);
74     vector<double> optimize();
80     Stochastic(Function* F, vector<double> x_0_, vector<double> l_border, vector<double> r_border,
81         int SC_var = 1, double eps_ = 0.001, double alpha_ = 0.2, double p_ = 0.5) :
82         alpha(alpha_), p(p_), OptimizationMethod(F, x_0_, l_border, r_border, SC_var, eps_) {}
83 };
```

## 5.4 R64M.hpp

```
1 #define _CRT_SECURE_NO_WARNINGS
2 #pragma once
3
4 void  rninit   (unsigned long long iufir);
5 void  rnrest   ();
6 void  rnconst  (unsigned long long iufir);
7 //void  rnconfix (unsigned nmb);
8
9 unsigned long long rnfirst ();
10 unsigned long long rnlast  ();
11 //unsigned long rnconrd ();
12
13 double  rnunif ();
14 //double  rnexp  ();
15 //double  rnnorm ( );
```

## 5.5 StopCriteria.h

```
1 #pragma once
```

```
2
3 #include <iostream>
4 #include <vector>
5 #include "Functions.h"
6 #include "Definitions.h"
7
8 using namespace std;
13 class StopCriterion {
14 protected:
15     int bound;
16 public:
17     double eps;
24     virtual bool do_we_stop(vector<vector<double>& a, Function* F, int iteration_number) { return true;
       };
29     StopCriterion(double ep = 0.001, int bo = 1000) : eps(ep), bound(bo) {}
30 };
34 class SC_GradientNorm : public StopCriterion {
41     bool do_we_stop(vector<vector<double>& a, Function* F, int iteration_number);
42 };
46 class SC_PointsClose : public StopCriterion {
53     bool do_we_stop(vector<vector<double>& a, Function* F, int iteration_number);
54 };
58 class SC_FuncRelative : public StopCriterion {
65     bool do_we_stop(vector<vector<double>& a, Function* F, int iteration_number);
66 };
```

# Index