Optimization Methods

0.1.0

Generated by Doxygen 1.8.17

1 Todo List 1

2 Namespace Index 3

2.1 Namespace List . . . 3

3 Hierarchical Index 5

3.1 Class Hierarchy . . . 5

4 Class Index 7

4.1 Class List . . . 7

5 Namespace Documentation 9

5.1 om_common Namespace Reference . . . 9

5.1.1 Detailed Description . . . 9

5.2 om_differentiation Namespace Reference . . . 9

5.2.1 Detailed Description . . . 10

5.3 om_differentiation_traits Namespace Reference . . . 10

5.3.1 Detailed Description . . . 10

5.4 om_test_functions Namespace Reference . . . 10

5.4.1 Detailed Description . . . 11

5.4.2 Function Documentation . . . 11

5.4.2.1 beale_function() . . . 11

5.4.2.2 create_rao_test_collection() . . . 12

5.4.2.3 fletcher_powell_helical_valley() . . . 12

5.4.2.4 freudenstein_roth_function() . . . 13

5.4.2.5 non_linear_function() . . . 13

5.4.2.6 powell_badly_scaled_function() . . . 14

5.4.2.7 powell_function() . . . 14

5.4.2.8 quadratic_function() . . . 15

5.4.2.9 rosenbrock_parabolic_valley() . . . 15

5.4.2.10 wood_function() . . . 16

5.4.3 Variable Documentation . . . 16

5.4.3.1 pi . . . 16

5.5 om_test_helpers Namespace Reference . . . 17

5.5.1 Detailed Description . . . 17

5.6 om_types Namespace Reference . . . 17

5.6.1 Detailed Description . . . 18

5.6.2 Typedef Documentation . . . 18

5.6.2.1 f_line_minimiser_t . . . 18

5.6.2.2 f_scalar_t . . . 18

5.6.2.3 f_vector_t . . . 18

5.6.2.4 matrix_t . . . 19

5.6.2.5 sptr_t . . . 19

5.6.2.6 vector_arg_t . . . 19

5.6.2.7 vector_t 20

5.7 om_unconstrained_methods Namespace Reference 20

5.7.1 Detailed Description 20

5.8 om_unconstrained_methods::om_conjugate_gradient Namespace Reference 20

5.8.1 Detailed Description 21

5.9 om_unconstrained_methods::om_line_methods Namespace Reference 21

5.9.1 Detailed Description 21

5.10 om_unconstrained_methods::om_quasi_newton Namespace Reference 21

5.10.1 Detailed Description 21

5.11 om_unconstrained_methods::om_steepest_descent Namespace Reference 22

5.11.1 Detailed Description 22

5.12 om_unconstrained_methods::om_zero_order Namespace Reference 22

5.12.1 Detailed Description 22

5.13 om_utilities Namespace Reference 22

5.13.1 Detailed Description 23

5.13.2 Function Documentation 23

5.13.2.1 fib() 23

5.13.2.2 iqerp() 23

5.13.2.3 lerp() 25

5.13.2.4 sign() 26

6 Class Documentation 27

6.1 om_unconstrained_methods::om_line_methods::brent_method< fp_type, typename > Class Template Reference 27

6.1.1 Detailed Description 27

6.1.2 Constructor & Destructor Documentation 28

6.1.2.1 brent_method() [1/2] 28

6.1.2.2 brent_method() [2/2] 28

6.1.3 Member Function Documentation 28

6.1.3.1 operator()() 28

6.1.3.2 operator=() 29

6.2 om_unconstrained_methods::om_quasi_newton::broyden_fletcher_goldfarb_shanno_method< fp_↩ type > Class Template Reference 29

6.2.1 Detailed Description 30

6.2.2 Constructor & Destructor Documentation 30

6.2.2.1 broyden_fletcher_goldfarb_shanno_method() 30

6.2.3 Member Function Documentation 31

6.2.3.1 minimize() 31

6.3 om_utilities::cartesian_basis_vectors< fp_type, typename > Struct Template Reference 31

6.3.1 Detailed Description 31

6.4 om_differentiation::central_difference< order, fp_type, typename > Struct Template Reference . . . 32

6.4.1 Detailed Description 32

6.5 om_differentiation::central_difference< 0, fp_type > Struct Template Reference 32

6.6 om_differentiation::central_difference< 1, fp_type > Struct Template Reference 33
6.7 om_differentiation_traits::central_difference_trait< fp_type > Struct Template Reference 33
6.7.1 Detailed Description 33
6.8 om_common::closest_to< count, fp_type, typename, type > Struct Template Reference 33
6.8.1 Detailed Description 34
6.9 om_common::closest_to< 2, fp_type > Struct Template Reference 34
6.10 om_common::closest_to< 3, fp_type > Struct Template Reference 34
6.11 om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base< fp_type, typename > Class Template Reference 34
6.11.1 Detailed Description 35
6.11.2 Constructor & Destructor Documentation 35
6.11.2.1 conjugate_gradient_base() [1/2] 35
6.11.2.2 conjugate_gradient_base() [2/2] 36
6.11.3 Member Function Documentation 36
6.11.3.1 operator=() 36
6.11.3.2 set_arg_tolerance() 37
6.11.3.3 set_fun_tolerance() 37
6.11.3.4 set_grad_tolerance() 37
6.11.3.5 set_max_iterations() 38
6.12 om_unconstrained_methods::om_quasi_newton::davidon_fletcher_powell_method< fp_type > Class Template Reference 38
6.12.1 Detailed Description 39
6.12.2 Constructor & Destructor Documentation 39
6.12.2.1 davidon_fletcher_powell_method() 39
6.12.3 Member Function Documentation 39
6.12.3.1 minimize() 39
6.13 om_differentiation::divided_difference< order, fp_type, typename, type > Struct Template Reference 40
6.13.1 Detailed Description 40
6.14 om_differentiation::divided_difference< 0, fp_type > Struct Template Reference 41
6.15 om_differentiation::divided_difference< 1, fp_type > Struct Template Reference 41
6.16 om_differentiation::divided_difference< 2, fp_type > Struct Template Reference 41
6.17 om_unconstrained_methods::om_line_methods::fibonacci_method< fp_type, typename > Class Template Reference 41
6.17.1 Detailed Description 42
6.17.2 Constructor & Destructor Documentation 42
6.17.2.1 fibonacci_method() [1/2] 42
6.17.2.2 fibonacci_method() [2/2] 43
6.17.3 Member Function Documentation 43
6.17.3.1 operator()() 43
6.17.3.2 operator=() 43
6.18 om_unconstrained_methods::om_conjugate_gradient::fletcher_reeves_method< fp_type > Class Template Reference 44
6.18.1 Detailed Description 44

6.18.2 Constructor & Destructor Documentation 45
6.18.2.1 fletcher_reeves_method() 45
6.18.3 Member Function Documentation 45
6.18.3.1 minimize() 45
6.19 om_differentiation::forward_difference< order, fp_type, typename > Struct Template Reference . . 46
6.19.1 Detailed Description 46
6.20 om_differentiation::forward_difference< 0, fp_type > Struct Template Reference 46
6.21 om_differentiation::forward_difference< 1, fp_type > Struct Template Reference 47
6.22 om_differentiation_traits::forward_difference_trait< fp_type > Struct Template Reference 47
6.22.1 Detailed Description 47
6.23 om_common::furthest_from< count, fp_type, typename, type > Struct Template Reference 47
6.23.1 Detailed Description 48
6.24 om_common::furthest_from< 2, fp_type > Struct Template Reference 48
6.25 om_common::furthest_from< 3, fp_type > Struct Template Reference 48
6.26 om_unconstrained_methods::om_line_methods::golden_section_method< fp_type, typename > Class Template Reference 49
6.26.1 Detailed Description 49
6.26.2 Constructor & Destructor Documentation 49
6.26.2.1 golden_section_method() [1/2] 49
6.26.2.2 golden_section_method() [2/2] 50
6.26.3 Member Function Documentation 50
6.26.3.1 operator()() 50
6.26.3.2 operator=() 51
6.27 om_unconstrained_methods::om_conjugate_gradient::hestenes_stiefel_method< fp_type > Class Template Reference 51
6.27.1 Detailed Description 52
6.27.2 Constructor & Destructor Documentation 52
6.27.2.1 hestenes_stiefel_method() 52
6.27.3 Member Function Documentation 52
6.27.3.1 minimize() 52
6.28 om_common::max_arg< count, fp_type, typename, type > Struct Template Reference 53
6.28.1 Detailed Description 53
6.29 om_common::max_arg< 2, fp_type > Struct Template Reference 54
6.30 om_common::max_arg< 3, fp_type > Struct Template Reference 54
6.31 om_common::min_arg< count, fp_type, typename, type > Struct Template Reference 54
6.31.1 Detailed Description 54
6.32 om_common::min_arg< 2, fp_type > Struct Template Reference 55
6.33 om_common::min_arg< 3, fp_type > Struct Template Reference 55
6.34 om_test_helpers::minimizer_helper< fp_type > Struct Template Reference 55
6.35 om_unconstrained_methods::om_zero_order::nelder_mead_method< fp_type > Class Template Reference 55
6.35.1 Detailed Description 56
6.35.2 Constructor & Destructor Documentation 56

6.35.2.1 nelder_mead_method() [1/2] 56
6.35.2.2 nelder_mead_method() [2/2] 57
6.35.3 Member Function Documentation 57
6.35.3.1 minimize() 57
6.35.3.2 operator=() 58
6.35.3.3 set_contraction_rho() 58
6.35.3.4 set_converge_tolerance() 58
6.35.3.5 set_expansion_rho() 59
6.35.3.6 set_max_iterations() 59
6.35.3.7 set_reflection_rho() 59
6.35.3.8 set_shrinkage_rho() 60
6.36 om_unconstrained_methods::om_conjugate_gradient::polak_ribiere_method< fp_type > Class Template Reference 60
6.36.1 Detailed Description 60
6.36.2 Constructor & Destructor Documentation 61
6.36.2.1 polak_ribiere_method() 61
6.36.3 Member Function Documentation 61
6.36.3.1 minimize() 61
6.37 om_unconstrained_methods::om_zero_order::powell_conjugate_method< fp_type > Class Template Reference 62
6.37.1 Detailed Description 62
6.37.2 Constructor & Destructor Documentation 63
6.37.2.1 powell_conjugate_method() [1/2] 63
6.37.2.2 powell_conjugate_method() [2/2] 63
6.37.3 Member Function Documentation 63
6.37.3.1 minimize() 63
6.37.3.2 operator=() 64
6.37.3.3 set_converge_tolerance() 64
6.37.3.4 set_max_iterations() 64
6.38 om_unconstrained_methods::om_line_methods::powell_method< fp_type, typename > Class Template Reference 65
6.38.1 Detailed Description 65
6.38.2 Constructor & Destructor Documentation 66
6.38.2.1 powell_method() [1/3] 66
6.38.2.2 powell_method() [2/3] 66
6.38.2.3 powell_method() [3/3] 67
6.38.3 Member Function Documentation 67
6.38.3.1 operator()() 67
6.38.3.2 operator=() 67
6.39 om_unconstrained_methods::om_quasi_newton::quasi_newton_base< fp_type, typename > Class Template Reference 68
6.39.1 Detailed Description 68
6.39.2 Constructor & Destructor Documentation 69

6.39.2.1 quasi_newton_base() [1/2] 69

6.39.2.2 quasi_newton_base() [2/2] 69

6.39.3 Member Function Documentation 70

6.39.3.1 operator=() 70

6.39.3.2 set_arg_tolerance() 70

6.39.3.3 set_fun_tolerance() 70

6.39.3.4 set_grad_tolerance() 71

6.39.3.5 set_max_iterations() 71

6.40 om_utilities::random_vectors_from_guess< fp_type, distribution, typename > Struct Template Reference 71

6.40.1 Detailed Description 72

6.41 om_utilities::range< fp_type, typename > Class Template Reference 72

6.41.1 Detailed Description 73

6.41.2 Constructor & Destructor Documentation 73

6.41.2.1 range() [1/4] 73

6.41.2.2 range() [2/4] 73

6.41.2.3 range() [3/4] 74

6.41.2.4 range() [4/4] 75

6.41.3 Member Function Documentation 75

6.41.3.1 high() 75

6.41.3.2 low() 75

6.41.3.3 low_high() 76

6.41.3.4 operator=() [1/2] 76

6.41.3.5 operator=() [2/2] 76

6.41.3.6 spread() 77

6.42 om_unconstrained_methods::om_steepest_descent::steepest_descent_method< fp_type > Class Template Reference 77

6.42.1 Detailed Description 78

6.42.2 Constructor & Destructor Documentation 79

6.42.2.1 steepest_descent_method() [1/2] 79

6.42.2.2 steepest_descent_method() [2/2] 79

6.42.3 Member Function Documentation 80

6.42.3.1 minimize() 80

6.42.3.2 operator=() 80

6.42.3.3 set_arg_tolerance() 80

6.42.3.4 set_fun_tolerance() 81

6.42.3.5 set_grad_tolerance() 81

6.42.3.6 set_max_iterations() 81

Index **83**

Chapter 1

Todo List

Member om_test_functions::freudenstein_roth_function (vector_arg_t< fp_type > const &args)

Check if the minimiser and local_minimiser are correct!!

Member om_test_functions::powell_badly_scaled_function (vector_arg_t< fp_type > const &args)

Check the validity of minimiser!!!

Chapter 2

Namespace Index

2.1 Namespace List

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

om_common
Contains some commonly used measures . . . 9

om_differentiation
Contains some numerical differentiation functors . . . 9

om_differentiation_traits
Contains traits tested for numerical differentiation . . . 10

om_test_functions
Some classical test functions (designed by Rao) . . . 10

om_test_helpers
Contains test helpers . . . 17

om_types
Contains some types used throughout the whole library . . . 17

om_unconstrained_methods
Contains some well-known methods for unconstrained optimisation . . . 20

om_unconstrained_methods::om_conjugate_gradient
Contains conjugate-gradient methods . . . 20

om_unconstrained_methods::om_line_methods
Contains one-dimensional line methods . . . 21

om_unconstrained_methods::om_quasi_newton
Contains Quasi-Newton methods . . . 21

om_unconstrained_methods::om_steepest_descent
Contains steepest-descent method . . . 22

om_unconstrained_methods::om_zero_order
Contains zero-order methods . . . 22

om_utilities
Contains some commonly used utilities . . . 22

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

om_unconstrained_methods::om_line_methods::brent_method< fp_type, typename > 27
om_utilities::cartesian_basis_vectors< fp_type, typename > 31
om_differentiation::central_difference< order, fp_type, typename > 32
om_differentiation::central_difference< 0, fp_type > 32
om_differentiation::central_difference< 1, fp_type > 33
om_differentiation_traits::central_difference_trait< fp_type > 33
om_common::closest_to< count, fp_type, typename, type > 33
om_common::closest_to< 2, fp_type > 34
om_common::closest_to< 3, fp_type > 34
om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base< fp_type, typename > 34
om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base< double > 34
  om_unconstrained_methods::om_conjugate_gradient::fletcher_reeves_method< fp_type > 44
  om_unconstrained_methods::om_conjugate_gradient::hestenes_stiefel_method< fp_type > 51
  om_unconstrained_methods::om_conjugate_gradient::polak_ribiere_method< fp_type > 60
om_differentiation::divided_difference< order, fp_type, typename, type > 40
om_differentiation::divided_difference< 0, fp_type > 41
om_differentiation::divided_difference< 1, fp_type > 41
om_differentiation::divided_difference< 2, fp_type > 41
om_unconstrained_methods::om_line_methods::fibonacci_method< fp_type, typename > 41
om_differentiation::forward_difference< order, fp_type, typename > 46
om_differentiation::forward_difference< 0, fp_type > 46
om_differentiation::forward_difference< 1, fp_type > 47
om_differentiation_traits::forward_difference_trait< fp_type > 47
om_common::furthest_from< count, fp_type, typename, type > 47
om_common::furthest_from< 2, fp_type > 48
om_common::furthest_from< 3, fp_type > 48
om_unconstrained_methods::om_line_methods::golden_section_method< fp_type, typename > 49
om_common::max_arg< count, fp_type, typename, type > 53
om_common::max_arg< 2, fp_type > 54
om_common::max_arg< 3, fp_type > 54
om_common::min_arg< count, fp_type, typename, type > 54
om_common::min_arg< 2, fp_type > 55
om_common::min_arg< 3, fp_type > 55
om_test_helpers::minimizer_helper< fp_type > 55
om_unconstrained_methods::om_zero_order::nelder_mead_method< fp_type > 55

om_unconstrained_methods::om_zero_order::powell_conjugate_method< fp_type > 62

om_unconstrained_methods::om_line_methods::powell_method< fp_type, typename > 65

om_unconstrained_methods::om_quasi_newton::quasi_newton_base< fp_type, typename > 68

om_unconstrained_methods::om_quasi_newton::quasi_newton_base< double > 68

- om_unconstrained_methods::om_quasi_newton::broyden_fletcher_goldfarb_shanno_method< fp_type > 29
- om_unconstrained_methods::om_quasi_newton::davidon_fletcher_powell_method< fp_type > 38

om_utilities::random_vectors_from_guess< fp_type, distribution, typename > 71

om_utilities::range< fp_type, typename > 72

om_utilities::range< double > 72

om_unconstrained_methods::om_steepest_descent::steepest_descent_method< fp_type > 77

Chapter 4

Class Index

4.1 Class List

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

om_unconstrained_methods::om_line_methods::brent_method< fp_type, typename >

Brent method object 27

om_unconstrained_methods::om_quasi_newton::broyden_fletcher_goldfarb_shanno_method< fp_type >

Broyden-Fletcher-Goldfarb-Shanno method object 29

om_utilities::cartesian_basis_vectors< fp_type, typename >

Cartesian basis vectors functor 31

om_differentiation::central_difference< order, fp_type, typename >

Central difference functor 32

om_differentiation::central_difference< 0, fp_type > 32

om_differentiation::central_difference< 1, fp_type > 33

om_differentiation_traits::central_difference_trait< fp_type >

Central difference trait 33

om_common::closest_to< count, fp_type, typename, type >

Closest_to functor 33

om_common::closest_to< 2, fp_type > 34

om_common::closest_to< 3, fp_type > 34

om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base< fp_type, typename >

Conjugate-gradient base class 34

om_unconstrained_methods::om_quasi_newton::davidon_fletcher_powell_method< fp_type >

Davidon-Fletcher-Powell method object 38

om_differentiation::divided_difference< order, fp_type, typename, type >

Divided difference functor 40

om_differentiation::divided_difference< 0, fp_type > 41

om_differentiation::divided_difference< 1, fp_type > 41

om_differentiation::divided_difference< 2, fp_type > 41

om_unconstrained_methods::om_line_methods::fibonacci_method< fp_type, typename >

Fibonacci method object 41

om_unconstrained_methods::om_conjugate_gradient::fletcher_reeves_method< fp_type >

Fletcher-Reeves method object 44

om_differentiation::forward_difference< order, fp_type, typename >

Forward difference functor 46

om_differentiation::forward_difference< 0, fp_type > 46

om_differentiation::forward_difference< 1, fp_type > 47

om_differentiation_traits::forward_difference_trait< fp_type >

Forward difference trait 47

om_common::furthest_from< count, fp_type, typename, type >
Furthest_from functor 47
om_common::furthest_from< 2, fp_type > 48
om_common::furthest_from< 3, fp_type > 48
om_unconstrained_methods::om_line_methods::golden_section_method< fp_type, typename >
Golden section method object 49
om_unconstrained_methods::om_conjugate_gradient::hestenes_stiefel_method< fp_type >
Hestenes-Stiefel method object 51
om_common::max_arg< count, fp_type, typename, type >
Max_arg functor returns argument at which a function takes maximum value 53
om_common::max_arg< 2, fp_type > 54
om_common::max_arg< 3, fp_type > 54
om_common::min_arg< count, fp_type, typename, type >
Min_arg functor retuns argument as which a function takes minimum value 54
om_common::min_arg< 2, fp_type > 55
om_common::min_arg< 3, fp_type > 55
om_test_helpers::minimizer_helper< fp_type >
Helper for optimisation methods 55
om_unconstrained_methods::om_zero_order::nelder_mead_method< fp_type >
Nelder-Mead method object 55
om_unconstrained_methods::om_conjugate_gradient::polak_ribiere_method< fp_type >
Polak-Ribiere method object 60
om_unconstrained_methods::om_zero_order::powell_conjugate_method< fp_type >
Powell conjugate method object 62
om_unconstrained_methods::om_line_methods::powell_method< fp_type, typename >
Powell method object 65
om_unconstrained_methods::om_quasi_newton::quasi_newton_base< fp_type, typename >
Quasi-Newton base class 68
om_utilities::random_vectors_from_guess< fp_type, distribution, typename >
Random vectors from guess functor 71
om_utilities::range< fp_type, typename >
Represents a one dimensional range 72
om_unconstrained_methods::om_steepest_descent::steepest_descent_method< fp_type >
Steepest descent method object 77

Chapter 5

Namespace Documentation

5.1 om_common Namespace Reference

Contains some commonly used measures.

Classes

- struct closest_to

  closest_to functor
- struct closest_to< 2, fp_type >
- struct closest_to< 3, fp_type >
- struct furthest_from

  furthest_from functor
- struct furthest_from< 2, fp_type >
- struct furthest_from< 3, fp_type >
- struct max_arg

  max_arg functor returns argument at which a function takes maximum value
- struct max_arg< 2, fp_type >
- struct max_arg< 3, fp_type >
- struct min_arg

  min_arg functor retuns argument as which a function takes minimum value
- struct min_arg< 2, fp_type >
- struct min_arg< 3, fp_type >

5.1.1 Detailed Description

Contains some commonly used measures.

5.2 om_differentiation Namespace Reference

Contains some numerical differentiation functors.

Classes

- struct central_difference

  central difference functor
- struct central_difference< 0, fp_type >
- struct central_difference< 1, fp_type >
- struct divided_difference

  Divided difference functor.
- struct divided_difference< 0, fp_type >
- struct divided_difference< 1, fp_type >
- struct divided_difference< 2, fp_type >
- struct forward_difference

  forward difference functor
- struct forward_difference< 0, fp_type >
- struct forward_difference< 1, fp_type >

5.2.1 Detailed Description

Contains some numerical differentiation functors.

5.3 om_differentiation_traits Namespace Reference

Contains traits tested for numerical differentiation.

Classes

- struct central_difference_trait

  central difference trait
- struct forward_difference_trait

  forward difference trait

5.3.1 Detailed Description

Contains traits tested for numerical differentiation.

5.4 om_test_functions Namespace Reference

Some classical test functions (designed by Rao)

Functions

- template<typename fp_type >
  fp_type rosenbrock_parabolic_valley (vector_arg_t< fp_type > const &args)
  Rosenbrock's parabolic valley test function.
- template<typename fp_type >
  fp_type quadratic_function (vector_arg_t< fp_type > const &args)
  Quadratic test function.
- template<typename fp_type >
  fp_type powell_function (vector_arg_t< fp_type > const &args)
  Powell's quadratic test function.
- template<typename fp_type >
  fp_type fletcher_powell_helical_valley (vector_arg_t< fp_type > const &args)
  Fletcher and Powell's helical valley test function.
- template<typename fp_type >
  fp_type non_linear_function (vector_arg_t< fp_type > const &args)
  Non-linear test function of 3 variables.
- template<typename fp_type >
  fp_type freudenstein_roth_function (vector_arg_t< fp_type > const &args)
  Freudenstein and Roth test function.
- template<typename fp_type >
  fp_type powell_badly_scaled_function (vector_arg_t< fp_type > const &args)
  Powell's badly scaled test function.
- template<typename fp_type >
  fp_type beale_function (vector_arg_t< fp_type > const &args)
  Beale's test function.
- template<typename fp_type >
  fp_type wood_function (vector_arg_t< fp_type > const &args)
  Wood's test function.
- template<typename fp_type >
  std::vector< sptr_t< minimizer_helper< fp_type > > > create_rao_test_collection ()
  Create a rao test collection object.

Variables

- template<typename fp_type >
  constexpr fp_type pi {3.14159265359}
  Pi definition used in the Rao test functons.

5.4.1 Detailed Description

Some classical test functions (designed by Rao)

5.4.2 Function Documentation

5.4.2.1 beale_function()

```
template<typename fp_type >
fp_type om_test_functions::beale_function (
            vector_arg_t< fp_type > const & args )
```

Beale's test function.

initial guess = (1.0,1.0), minimiser = (3.0,0.5)

Template Parameters

fp_type	

Parameters

args	

Returns

fp_type

5.4.2.2 create_rao_test_collection()

```
template<typename fp_type >
std::vector<sptr_t<minimizer_helper<fp_type> > > om_test_functions::create_rao_test_collection
( )
```

Create a rao test collection object.

Template Parameters

fp_type	fp_type is a floating-point template parameter

Returns

std::vector<sptr_t<minimizer_helper<fp_type>>>

5.4.2.3 fletcher_powell_helical_valley()

```
template<typename fp_type >
fp_type om_test_functions::fletcher_powell_helical_valley (
            vector_arg_t< fp_type > const & args )
```

Fletcher and Powell's helical valley test function.

initial guess = (-1.0,0.0,0.0), minimiser = (1.0,0.0,0.0)

Template Parameters

fp_type	fp_type is a floating-point template parameter

Parameters

args	function arguments

Returns

fp_type

5.4.2.4 freudenstein_roth_function()

```
template<typename fp_type >
fp_type om_test_functions::freudenstein_roth_function (
            vector_arg_t< fp_type > const & args )
```

Freudenstein and Roth test function.

initial guess = (0.5,-2.0), minimiser = (5.0,4.0), local_minimiser = (11.41..., -0.8968)

Todo Check if the minimiser and local_minimiser are correct!!

Template Parameters

fp_type	fp_type is a floating-point template parameter

Parameters

args	function arguments

Returns

fp_type

5.4.2.5 non_linear_function()

```
template<typename fp_type >
fp_type om_test_functions::non_linear_function (
            vector_arg_t< fp_type > const & args )
```

Non-linear test function of 3 variables.

initial guess = (0.0,1.0,2.0), minimiser = (1.0,1.0,1.0)

Template Parameters

fp_type	fp_type is a floating-point template parameter

Parameters

args	function arguments

Returns

fp_type

5.4.2.6 powell_badly_scaled_function()

```
template<typename fp_type >
fp_type om_test_functions::powell_badly_scaled_function (
            vector_arg_t< fp_type > const & args )
```

Powell's badly scaled test function.

initial guess = (0.0,1.0), minimiser = (1.098...*10^-5,9.106...)

Todo Check the validity of minimiser!!!

Template Parameters

fp_type	

Parameters

args	

Returns

fp_type

5.4.2.7 powell_function()

```
template<typename fp_type >
fp_type om_test_functions::powell_function (
            vector_arg_t< fp_type > const & args )
```

Powell's quadratic test function.

initial guess = (3.0,-1.0,0.0,1.0), minimiser = (0.0,0.0,0.0,0.0)

Template Parameters

fp_type	fp_type is a floating-point template parameter

Parameters

args	function arguments

Returns

fp_type

5.4.2.8 quadratic_function()

```
template<typename fp_type >
fp_type om_test_functions::quadratic_function (
             vector_arg_t< fp_type > const & args )
```

Quadratic test function.

initial guess = (0.0,0.0), minimiser = (1.0,3.0)

Template Parameters

fp_type	fp_type is a floating-point template parameter

Parameters

args	function arguments

Returns

fp_type

5.4.2.9 rosenbrock_parabolic_valley()

```
template<typename fp_type >
fp_type om_test_functions::rosenbrock_parabolic_valley (
             vector_arg_t< fp_type > const & args )
```

Rosenbrock's parabolic valley test function.

initial guess = (-1.2,1.0), minimiser = (1.0,1.0)

Template Parameters

fp_type	fp_type is a floating-point template parameter

Parameters

args	arguments of the function

Returns

fp_type

5.4.2.10 wood_function()

```
template<typename fp_type >
fp_type om_test_functions::wood_function (
            vector_arg_t< fp_type > const & args )
```

Wood's test function.

initial guess = (-3.0,-1.0,-3.0,-1.0), minimiser = (1.0,1.0,1.0,1.0)

Template Parameters

fp_type	fp_type is a floation-point template parameter

Parameters

args	function arguments

Returns

fp_type

5.4.3 Variable Documentation

5.4.3.1 pi

```
template<typename fp_type >
constexpr fp_type om_test_functions::pi {3.14159265359}  [constexpr]
```

Pi definition used in the Rao test functons.

Template Parameters

fp_type	fp_type is a floating-point template parameter

5.5 om_test_helpers Namespace Reference

Contains test helpers.

Classes

- struct minimizer_helper

  Helper for optimisation methods.

5.5.1 Detailed Description

Contains test helpers.

5.6 om_types Namespace Reference

Contains some types used throughout the whole library.

Typedefs

- template< typename T >
  using sptr_t = std::shared_ptr< T >

  Alias for shared_ptr< T>
- template< typename T >
  using vector_arg_t = Eigen::Matrix< T, Eigen::Dynamic, 1 >

  Alias for 1D matrix = vector.
- template< typename T >
  using vector_t = Eigen::Matrix< T, Eigen::Dynamic, 1 >

  Alias for 1D matrix = vector.
- template< typename T >
  using f_scalar_t = std::function< T(T)>

  One dimensional scalar function.
- template< typename T >
  using f_vector_t = std::function< T(vector_arg_t< T >)>

  One dimensional vector function.
- template< typename T >
  using matrix_t = Eigen::Matrix< T, Eigen::Dynamic, Eigen::Dynamic >

  Alias for Eigen matrix.
- template< typename fp_type >
  using f_line_minimiser_t = std::function< std::tuple< fp_type, fp_type, std::size_t, std::size_t >(f_scalar_t< fp_type > &&)>

  Line method functor type.
- template< typename T >
  using **constraints_t** = std::vector< std::pair< f_vector_t< T >, constraint_t > >

Enumerations

- enum **one_dim_line_search_method** { **GoldenSection**, **Powell** }
- enum **constraint_t** { **Equality**, **LessThenZero** }

5.6.1 Detailed Description

Contains some types used throughout the whole library.

5.6.2 Typedef Documentation

5.6.2.1 f_line_minimiser_t

```
template<typename fp_type >
using om_types::f_line_minimiser_t = typedef std::function<std::tuple<fp_type, fp_type, std↩
::size_t, std::size_t>( f_scalar_t<fp_type> &&)>
```

Line method functor type.

Template Parameters

fp_type	

5.6.2.2 f_scalar_t

```
template<typename T >
using om_types::f_scalar_t = typedef std::function<T(T)>
```

One dimensional scalar function.

Template Parameters

T	

5.6.2.3 f_vector_t

```
template<typename T >
using om_types::f_vector_t = typedef std::function<T(vector_arg_t<T>)>
```

One dimensional vector function.

Template Parameters

T	

5.6.2.4 matrix_t

```
template<typename T >
using om_types::matrix_t = typedef Eigen::Matrix<T, Eigen::Dynamic, Eigen::Dynamic>
```

Alias for Eigen matrix.

Template Parameters

T	

5.6.2.5 sptr_t

```
template<typename T >
using om_types::sptr_t = typedef std::shared_ptr<T>
```

Alias for shared_ptr<T>

Template Parameters

T	

5.6.2.6 vector_arg_t

```
template<typename T >
using om_types::vector_arg_t = typedef Eigen::Matrix<T, Eigen::Dynamic, 1>
```

Alias for 1D matrix = vector.

Template Parameters

T	

5.6.2.7 vector_t

```
template<typename T >
using om_types::vector_t = typedef Eigen::Matrix<T, Eigen::Dynamic, 1>
```

Alias for 1D matrix = vector.

Template Parameters

T	

5.7 om_unconstrained_methods Namespace Reference

Contains some well-known methods for unconstrained optimisation.

Namespaces

- om_conjugate_gradient

  Contains conjugate-gradient methods.
- om_line_methods

  Contains one-dimensional line methods.
- om_quasi_newton

  Contains Quasi-Newton methods.
- om_steepest_descent

  Contains steepest-descent method.
- om_zero_order

  Contains zero-order methods.

5.7.1 Detailed Description

Contains some well-known methods for unconstrained optimisation.

5.8 om_unconstrained_methods::om_conjugate_gradient Namespace Reference

Contains conjugate-gradient methods.

Classes

- class conjugate_gradient_base

  Conjugate-gradient base class.
- class fletcher_reeves_method

  Fletcher-Reeves method object.
- class hestenes_stiefel_method

  Hestenes-Stiefel method object.
- class polak_ribiere_method

  Polak-Ribiere method object.

5.8.1 Detailed Description

Contains conjugate-gradient methods.

5.9 om_unconstrained_methods::om_line_methods Namespace Reference

Contains one-dimensional line methods.

Classes

- class brent_method

  Brent method object.
- class fibonacci_method

  Fibonacci method object.
- class golden_section_method

  Golden section method object.
- class powell_method

  Powell method object.

5.9.1 Detailed Description

Contains one-dimensional line methods.

5.10 om_unconstrained_methods::om_quasi_newton Namespace Reference

Contains Quasi-Newton methods.

Classes

- class broyden_fletcher_goldfarb_shanno_method

  Broyden-Fletcher-Goldfarb-Shanno method object.
- class davidon_fletcher_powell_method

  Davidon-Fletcher-Powell method object.
- class quasi_newton_base

  Quasi-Newton base class.

5.10.1 Detailed Description

Contains Quasi-Newton methods.

5.11 om_unconstrained_methods::om_steepest_descent Namespace Reference

Contains steepest-descent method.

Classes

- class steepest_descent_method

  Steepest descent method object.

5.11.1 Detailed Description

Contains steepest-descent method.

5.12 om_unconstrained_methods::om_zero_order Namespace Reference

Contains zero-order methods.

Classes

- class nelder_mead_method

  Nelder-Mead method object.
- class powell_conjugate_method

  Powell conjugate method object.

5.12.1 Detailed Description

Contains zero-order methods.

5.13 om_utilities Namespace Reference

Contains some commonly used utilities.

Classes

- struct cartesian_basis_vectors

  Cartesian basis vectors functor.
- struct random_vectors_from_guess

  Random vectors from guess functor.
- class range

  Represents a one dimensional range.

Functions

- double fib (std::size_t n)

  fib function
- template<typename fp_type >
  fp_type iqerp (fp_type x0, fp_type x1, fp_type x2, fp_type y0, fp_type y1, fp_type y2)

  Inverse quadratic interpolation among points (x0,y0),(x1,y1),(x2,y2)
- template<typename fp_type >
  fp_type lerp (fp_type x0, fp_type x1, fp_type y0, fp_type y1)

  Linear interpolation between points (x0,y0) and (x1,y1)
- template<typename fp_type >
  fp_type sign (fp_type x)

  Signum function.

5.13.1 Detailed Description

Contains some commonly used utilities.

5.13.2 Function Documentation

5.13.2.1 fib()

```
double om_utilities::fib (
            std::size_t n )
```

fib function

Parameters

n	number of values from Fibonacci sequence

Returns

double

5.13.2.2 iqerp()

```
template<typename fp_type >
fp_type om_utilities::iqerp (
            fp_type x0,
            fp_type x1,
            fp_type x2,
            fp_type y0,
            fp_type y1,
            fp_type y2 )
```

Inverse quadratic interpolation among points (x0,y0),(x1,y1),(x2,y2)

Template Parameters

fp_type	

Parameters

x0	first value
x1	second value
x2	third value
y0	first function value
y1	second function value
y2	third function value

Returns

fp_type

5.13.2.3 lerp()

```
template<typename fp_type >
fp_type om_utilities::lerp (
            fp_type x0,
            fp_type x1,
            fp_type y0,
            fp_type y1 )
```

Linear interpolation between points (x0,y0) and (x1,y1)

Template Parameters

fp_type	

Parameters

x0	first value
x1	second value
y0	first function value
y1	second function value

Returns

fp_type

5.13.2.4 sign()

```
template<typename fp_type >
fp_type om_utilities::sign (
            fp_type x )
```

Signum function.

Template Parameters

fp_type	fp_type is a floating-point template parameter

Parameters

x	value

Returns

fp_type

Chapter 6

Class Documentation

6.1 om_unconstrained_methods::om_line_methods::brent_method< fp_type, typename > Class Template Reference

Brent method object.

```
#include <om_brent.hpp>
```

Public Types

- typedef fp_type **value_type**

Public Member Functions

- brent_method (range< fp_type > const &range, fp_type tolerance=1e-5, std::size_t max_iters=1000)

  Construct a new brent method object.
- brent_method (brent_method const &copy)

  Copy constructor of a brent method object.
- brent_method & operator= (brent_method const &copy)

  Assignment operator of a brent method object.
- std::tuple< fp_type, fp_type, std::size_t, std::size_t > operator() (f_scalar_t< fp_type > &&fun) const

  Functor of a brent method object.

6.1.1 Detailed Description

template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_point<fp_type>::value>::type>
class om_unconstrained_methods::om_line_methods::brent_method< fp_type, typename >

Brent method object.

Template Parameters

fp_type	fp_type id a floating-point template parameter
std::enable_if<	std::is_floating_point<fp_type>::value>::type

6.1.2 Constructor & Destructor Documentation

6.1.2.1 brent_method() [1/2]

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
om_unconstrained_methods::om_line_methods::brent_method< fp_type, typename >::brent_method (
            range< fp_type > const & range,
            fp_type tolerance = 1e-5,
            std::size_t max_iters = 1000 )  [inline]
```

Construct a new brent method object.

Parameters

range	range of the minimiser
tolerance	tolerance of the minimiser
max_iters	maximum number of iterations

6.1.2.2 brent_method() [2/2]

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
om_unconstrained_methods::om_line_methods::brent_method< fp_type, typename >::brent_method (
            brent_method< fp_type, typename > const & copy )  [inline]
```

Copy constructor of a brent method object.

Parameters

copy	copy is the object which we want to make a copy of

6.1.3 Member Function Documentation

6.1.3.1 operator()()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
std::tuple<fp_type, fp_type, std::size_t, std::size_t> om_unconstrained_methods::om_line_methods::brent_metho
fp_type, typename >::operator() (
            f_scalar_t< fp_type > && fun ) const  [inline]
```

Functor of a brent method object.

Parameters

fun	objective function

Returns

std::tuple<fp_type, fp_type, std::size_t, std::size_t>

6.1.3.2 operator=()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
brent_method& om_unconstrained_methods::om_line_methods::brent_method< fp_type, typename >↩
::operator= (
            brent_method< fp_type, typename > const & copy )  [inline]
```

Assignment operator of a brent method object.

Parameters

copy	

Returns

brent_method&

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

- include/unconstrained_methods/one_dim/om_brent.hpp

6.2 om_unconstrained_methods::om_quasi_newton::broyden_fletcher_goldfarb_shanno_method< fp_type > Class Template Reference

Broyden-Fletcher-Goldfarb-Shanno method object.

```
#include <om_broyden_fletcher_goldfarb_shanno.hpp>
```

Inheritance diagram for om_unconstrained_methods::om_quasi_newton::broyden_fletcher_goldfarb_shanno_method< fp_type >:

Collaboration diagram for om_unconstrained_methods::om_quasi_newton::broyden_fletcher_goldfarb_shanno_method< fp_type >:

Public Member Functions

- broyden_fletcher_goldfarb_shanno_method (f_line_minimiser_t< fp_type > const &line_search_minimiser, std::size_t const &max_iters=100, fp_type arg_tol=1e-4, fp_type grad_tol=1e-4, fp_type fun_tol=1e-4)

  Construct a new broyden fletcher goldfarb shanno method object.
- std::tuple< vector_t< fp_type >, fp_type, std::size_t > minimize (f_vector_t< fp_type > objective, vector_arg_t< fp_type > const &init_guess) const

  Function method that minimises the objective function.

Additional Inherited Members

6.2.1 Detailed Description

template<typename fp_type = double>
class om_unconstrained_methods::om_quasi_newton::broyden_fletcher_goldfarb_shanno_method< fp_type >

Broyden-Fletcher-Goldfarb-Shanno method object.

Template Parameters

fp_type	fp+type is a floating-point template parameter

6.2.2 Constructor & Destructor Documentation

6.2.2.1 broyden_fletcher_goldfarb_shanno_method()

```
template<typename fp_type = double>
om_unconstrained_methods::om_quasi_newton::broyden_fletcher_goldfarb_shanno_method< fp_type
>::broyden_fletcher_goldfarb_shanno_method (
            f_line_minimiser_t< fp_type > const & line_search_minimiser,
            std::size_t const & max_iters = 100,
            fp_type arg_tol = 1e-4,
            fp_type grad_tol = 1e-4,
            fp_type fun_tol = 1e-4 )  [inline]
```

Construct a new broyden fletcher goldfarb shanno method object.

Parameters

line_search_minimiser	line method to be used in finding the minimiser
max_iters	maximum number of iterations
arg_tol	tolerance for stopping criteria
grad_tol	tolerance for gradient
fun_tol	tolerance for a value of objective function

6.2.3 Member Function Documentation

6.2.3.1 minimize()

```
template<typename fp_type >
std::tuple< om_unconstrained_methods::om_quasi_newton::vector_t< fp_type >, fp_type, std↩
::size_t > om_unconstrained_methods::om_quasi_newton::broyden_fletcher_goldfarb_shanno_method<
fp_type >::minimize (
            f_vector_t< fp_type > objective,
            vector_arg_t< fp_type > const & init_guess ) const
```

Function method that minimises the objective function.

Parameters

objective	objective function
init_guess	initial guess

Returns

std::tuple<vector_t<fp_type>, fp_type, std::size_t>

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

- include/unconstrained_methods/multi_dim/quasi_newton/om_broyden_fletcher_goldfarb_shanno.hpp

6.3 om_utilities::cartesian_basis_vectors< fp_type, typename > Struct Template Reference

Cartesian basis vectors functor.

```
#include <om_utilities.hpp>
```

Public Member Functions

- std::vector< vector_t< fp_type > > **operator()** (std::size_t const &dimension) const

6.3.1 Detailed Description

**template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_point<fp_type>::value>::type>
struct om_utilities::cartesian_basis_vectors< fp_type, typename >**

Cartesian basis vectors functor.

Template Parameters

fp_type	fp_type is a floating-point template parameter
std::enable_if<	std::is_floating_point<fp_type>::value>::type

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

- include/utilities/om_utilities.hpp

6.4 om_differentiation::central_difference< order, fp_type, typename > Struct Template Reference

central difference functor

```
#include <om_differentiation.hpp>
```

6.4.1 Detailed Description

template<std::size_t order, typename fp_type, typename = typename std::enable_if< std::is_floating_point<fp_type>↩ ::value>::type>
struct om_differentiation::central_difference< order, fp_type, typename >

central difference functor

Template Parameters

order	order of difference
fp_type	fp_type is a floating-point template parameter
std::enable_if<	std::is_floating_point<fp_type>::value>::type

order = 0,order = 1 currently supported

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

- include/utilities/om_differentiation.hpp

6.5 om_differentiation::central_difference< 0, fp_type > Struct Template Reference

Public Member Functions

- vector_t< fp_type > **operator()** (f_vector_t< fp_type > fun, vector_arg_t< fp_type > const &args) const

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

- include/utilities/om_differentiation.hpp

6.6 om_differentiation::central_difference< 1, fp_type > Struct Template Reference

Public Member Functions

- vector_t< fp_type > **operator()** (f_vector_t< fp_type > fun, vector_arg_t< fp_type > const &args) const

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

- include/utilities/om_differentiation.hpp

6.7 om_differentiation_traits::central_difference_trait< fp_type > Struct Template Reference

central difference trait

```
#include <om_differentiation_traits.hpp>
```

Static Public Attributes

- static constexpr fp_type **step_size** = 10e-7

6.7.1 Detailed Description

template<typename fp_type>
struct om_differentiation_traits::central_difference_trait< fp_type >

central difference trait

Template Parameters

fp_type	fp_type is a floating-point template parameter

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

- include/utilities/om_differentiation_traits.hpp

6.8 om_common::closest_to< count, fp_type, typename, type > Struct Template Reference

closest_to functor

```
#include <om_common.hpp>
```

6.8.1 Detailed Description

template< std::size_t count, typename fp_type = double, typename = typename std::enable_if< count >= 2 && count <= 3, ↩ ::type>
struct om_common::closest_to< count, fp_type, typename, type >

closest_to functor

Template Parameters

count	number of points
fp_type	fp_type is a floating-point template parameter

count = 2,count = 3 is currently supported

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

- include/utilities/om_common.hpp

6.9 om_common::closest_to< 2, fp_type > Struct Template Reference

Public Member Functions

- fp_type **operator()** (fp_type const &target, fp_type const &x1, fp_type const &x2) const

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

- include/utilities/om_common.hpp

6.10 om_common::closest_to< 3, fp_type > Struct Template Reference

Public Member Functions

- fp_type **operator()** (fp_type const &target, fp_type const &x1, fp_type const &x2, fp_type const &x3) const

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

- include/utilities/om_common.hpp

6.11 om_unconstrained_methods::om_conjugate_gradient::conjugate↩_gradient_base< fp_type, typename > Class Template Reference

Conjugate-gradient base class.

```
#include <om_conjugate_gradient_base.hpp>
```

Collaboration diagram for om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base< fp_↩type, typename >:

Public Member Functions

- conjugate_gradient_base (f_line_minimiser_t< fp_type > const &line_search_minimiser, std::size_t const &max_iters=100, fp_type arg_tol=1e-4, fp_type grad_tol=1e-4, fp_type fun_tol=1e-4)

  Construct a new conjugate gradient base object.
- conjugate_gradient_base (conjugate_gradient_base const &copy)

  Construct a new conjugate gradient base object.
- conjugate_gradient_base & operator= (conjugate_gradient_base const &copy)

  Assignment operator of a conjugate gradient base object.
- void set_arg_tolerance (fp_type arg_tol)

  Set the stopping criteria tolerance object.
- void set_fun_tolerance (fp_type fun_tol)

  Set the fun tolerance object.
- void set_grad_tolerance (fp_type grad_tol)

  Set the grad tolerance object.
- void set_max_iterations (std::size_t const &iters)

  Set the max iterations object.

Protected Attributes

- fp_type **arg_tol_**
- fp_type **grad_tol_**
- fp_type **fun_tol_**
- std::size_t **max_iters_**
- f_line_minimiser_t< fp_type > **lsm_**

6.11.1 Detailed Description

template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_point<fp_type>::value>::type> class om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base< fp_type, typename >

Conjugate-gradient base class.

Template Parameters

fp_type	fp_type is a floating-point template parameter
std::enable_if<	std::is_floating_point<fp_type>::value>::type

6.11.2 Constructor & Destructor Documentation

6.11.2.1 conjugate_gradient_base() [1/2]

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
```

```
om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base< fp_type, typename
>::conjugate_gradient_base (
            f_line_minimiser_t< fp_type > const & line_search_minimiser,
            std::size_t const & max_iters = 100,
            fp_type arg_tol = 1e-4,
            fp_type grad_tol = 1e-4,
            fp_type fun_tol = 1e-4 )  [inline], [explicit]
```

Construct a new conjugate gradient base object.

Parameters

line_search_minimiser	line method to be used in finding the minimiser
max_iters	maximum number of iterations
arg_toltolerance	for a stopping criteria
grad_tol	tolerance for gradient
fun_tol	tolerance for a value of objective function

6.11.2.2 conjugate_gradient_base() [2/2]

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base< fp_type, typename
>::conjugate_gradient_base (
            conjugate_gradient_base< fp_type, typename > const & copy )  [inline]
```

Construct a new conjugate gradient base object.

Parameters

copy	copy is the object which we want to make a copy of

6.11.3 Member Function Documentation

6.11.3.1 operator=()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
conjugate_gradient_base& om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base<
fp_type, typename >::operator= (
            conjugate_gradient_base< fp_type, typename > const & copy )  [inline]
```

Assignment operator of a conjugate gradient base object.

Parameters

copy	

Returns

conjugate_gradient_base&

6.11.3.2 set_arg_tolerance()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
void om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base< fp_type, typename
>::set_arg_tolerance (
             fp_type arg_tol )  [inline]
```

Set the stopping criteria tolerance object.

Parameters

arg_tol	tolerance for a stopping criteria

6.11.3.3 set_fun_tolerance()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
void om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base< fp_type, typename
>::set_fun_tolerance (
             fp_type fun_tol )  [inline]
```

Set the fun tolerance object.

Parameters

fun_tol	tolerance for a value of objective function

6.11.3.4 set_grad_tolerance()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
void om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base< fp_type, typename
```

```
>::set_grad_tolerance (
            fp_type grad_tol )  [inline]
```

Set the grad tolerance object.

Parameters

grad_tol	tolerance for gradient

6.11.3.5 set_max_iterations()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
void om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base< fp_type, typename
>::set_max_iterations (
            std::size_t const & iters )  [inline]
```

Set the max iterations object.

Parameters

iters	maximum number of iterations

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

- include/unconstrained_methods/multi_dim/conjugate_gradient/om_conjugate_gradient_base.hpp

6.12 om_unconstrained_methods::om_quasi_newton::davidon_↩ fletcher_powell_method< fp_type > Class Template Reference

Davidon-Fletcher-Powell method object.

```
#include <om_davidon_fletcher_powell.hpp>
```

Inheritance diagram for om_unconstrained_methods::om_quasi_newton::davidon_fletcher_powell_method< fp_↩ type >:

Collaboration diagram for om_unconstrained_methods::om_quasi_newton::davidon_fletcher_powell_method< fp↩ _type >:

Public Member Functions

- davidon_fletcher_powell_method (f_line_minimiser_t< fp_type > const &line_search_minimiser, std::size_t const &max_iters=100, fp_type arg_tol=1e-4, fp_type grad_tol=1e-4, fp_type fun_tol=1e-4)

  Construct a new davidon fletcher powell method object.
- std::tuple< vector_t< fp_type >, fp_type, std::size_t > minimize (f_vector_t< fp_type > objective, vector↩ _arg_t< fp_type > const &init_guess) const

  Function method that minimises the objective function.

Additional Inherited Members

6.12.1 Detailed Description

template<typename fp_type = double>
class om_unconstrained_methods::om_quasi_newton::davidon_fletcher_powell_method< fp_type >

Davidon-Fletcher-Powell method object.

Template Parameters

fp_type	fp_type is a floating-point template parameter

6.12.2 Constructor & Destructor Documentation

6.12.2.1 davidon_fletcher_powell_method()

```
template<typename fp_type = double>
om_unconstrained_methods::om_quasi_newton::davidon_fletcher_powell_method< fp_type >::davidon_fletcher_powell
(
            f_line_minimiser_t< fp_type > const & line_search_minimiser,
            std::size_t const & max_iters = 100,
            fp_type arg_tol = 1e-4,
            fp_type grad_tol = 1e-4,
            fp_type fun_tol = 1e-4 )  [inline]
```

Construct a new davidon fletcher powell method object.

Parameters

line_search_minimiser	line method to be used in finding the minimiser
max_iters	maximum number of iterations
arg_tol	tolerance for stopping criteria
grad_tol	tolerance for gradient
fun_tol	tolerance for a value of objective function

6.12.3 Member Function Documentation

6.12.3.1 minimize()

```
template<typename fp_type >
std::tuple< om_unconstrained_methods::om_quasi_newton::vector_t< fp_type >, fp_type, std↩
::size_t > om_unconstrained_methods::om_quasi_newton::davidon_fletcher_powell_method< fp_type
```

```
>::minimize (
            f_vector_t< fp_type > objective,
            vector_arg_t< fp_type > const & init_guess ) const
```

Function method that minimises the objective function.

Parameters

objective	objective function
init_guess	initial guess

Returns

std::tuple<vector_t<fp_type>, fp_type, std::size_t>

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

- include/unconstrained_methods/multi_dim/quasi_newton/om_davidon_fletcher_powell.hpp

6.13 om_differentiation::divided_difference< order, fp_type, typename, type > Struct Template Reference

Divided difference functor.

```
#include <om_differentiation.hpp>
```

6.13.1 Detailed Description

template<std::size_t order, typename fp_type = double, typename = typename std::enable_if<order >= 0 && order <= 3, ::type> struct om_differentiation::divided_difference< order, fp_type, typename, type >

Divided difference functor.

Template Parameters

order	order of difference
fp_type	fp_type is a floating-point template parameter

order = 0, order = 1, order = 2, order = 3 currently supported

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

- include/utilities/om_differentiation.hpp

6.14 om_differentiation::divided_difference< 0, fp_type > Struct Template Reference

Public Member Functions

- fp_type **operator()** (f_scalar_t< fp_type > fun, fp_type const &arg) const

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

- include/utilities/om_differentiation.hpp

6.15 om_differentiation::divided_difference< 1, fp_type > Struct Template Reference

Public Member Functions

- fp_type **operator()** (f_scalar_t< fp_type > fun, std::tuple< fp_type, fp_type > const &arg) const

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

- include/utilities/om_differentiation.hpp

6.16 om_differentiation::divided_difference< 2, fp_type > Struct Template Reference

Public Member Functions

- fp_type **operator()** (f_scalar_t< fp_type > fun, std::tuple< fp_type, fp_type, fp_type > const &arg) const

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

- include/utilities/om_differentiation.hpp

6.17 om_unconstrained_methods::om_line_methods::fibonacci_↩ method< fp_type, typename > Class Template Reference

Fibonacci method object.

```
#include <om_fibonacci.hpp>
```

Public Types

- typedef fp_type **value_type**

Public Member Functions

- fibonacci_method (range< fp_type > const &range, fp_type tolerance=1e-5, std::size_t max_iters=1000)

  Construct a new fibonacci method object.
- fibonacci_method (fibonacci_method const &copy)

  Copy constructor of a fibonacci method object.
- fibonacci_method & operator= (fibonacci_method const &copy)

  Assignment operator of a fibonacci method object.
- std::tuple< fp_type, fp_type, std::size_t, std::size_t > operator() (f_scalar_t< fp_type > &&fun) const

  Functor of a fibonacci method object.

6.17.1 Detailed Description

template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_point<fp_type>::value>::type> class om_unconstrained_methods::om_line_methods::fibonacci_method< fp_type, typename >

Fibonacci method object.

Template Parameters

fp_type	fp_type is floating-point template parameter
std::enable_if<	std::is_floating_point<fp_type>::value>::type

6.17.2 Constructor & Destructor Documentation

6.17.2.1 fibonacci_method() [1/2]

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
om_unconstrained_methods::om_line_methods::fibonacci_method< fp_type, typename >::fibonacci_method
(
            range< fp_type > const & range,
            fp_type tolerance = 1e-5,
            std::size_t max_iters = 1000 )  [inline]
```

Construct a new fibonacci method object.

Parameters

range	range of the minimiser
tolerance	tolerance of the minimiser
max_iters	maximum number of iterations

6.17.2.2 fibonacci_method() [2/2]

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
om_unconstrained_methods::om_line_methods::fibonacci_method< fp_type, typename >::fibonacci_method
(
             fibonacci_method< fp_type, typename > const & copy )  [inline]
```

Copy constructor of a fibonacci method object.

Parameters

copy	copy is the object which we want to make a copy of

6.17.3 Member Function Documentation

6.17.3.1 operator()()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
std::tuple<fp_type, fp_type, std::size_t, std::size_t> om_unconstrained_methods::om_line_methods::fibonacci_m
fp_type, typename >::operator() (
             f_scalar_t< fp_type > && fun ) const  [inline]
```

Functor of a fibonacci method object.

Parameters

fun	objective function

Returns

std::tuple<fp_type, fp_type, std::size_t, std::size_t>

6.17.3.2 operator=()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
fibonacci_method& om_unconstrained_methods::om_line_methods::fibonacci_method< fp_type, typename
>::operator= (
             fibonacci_method< fp_type, typename > const & copy )  [inline]
```

Assignment operator of a fibonacci method object.

Parameters

copy	

Returns

fibonacci_method&

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

- include/unconstrained_methods/one_dim/om_fibonacci.hpp

6.18 om_unconstrained_methods::om_conjugate_gradient::fletcher_reeves_method< fp_type > Class Template Reference

Fletcher-Reeves method object.

```
#include <om_fletcher_reeves.hpp>
```

Inheritance diagram for om_unconstrained_methods::om_conjugate_gradient::fletcher_reeves_method< fp_type >:

Collaboration diagram for om_unconstrained_methods::om_conjugate_gradient::fletcher_reeves_method< fp_type >:

Public Member Functions

- fletcher_reeves_method (f_line_minimiser_t< fp_type > const &line_search_minimiser, std::size_t const &max_iters=100, fp_type arg_tol=1e-4, fp_type grad_tol=1e-4, fp_type fun_tol=1e-4)

  Construct a new fletcher reeves method object.
- std::tuple< vector_t< fp_type >, fp_type, std::size_t > minimize (f_vector_t< fp_type > objective, vector_arg_t< fp_type > const &init_guess) const

  Function method that minimises the objective function.

Additional Inherited Members

6.18.1 Detailed Description

template<typename fp_type = double>
class om_unconstrained_methods::om_conjugate_gradient::fletcher_reeves_method< fp_type >

Fletcher-Reeves method object.

Template Parameters

fp_type	fp_type is a floating-point template parameter

6.18.2 Constructor & Destructor Documentation

6.18.2.1 fletcher_reeves_method()

```
template<typename fp_type = double>
om_unconstrained_methods::om_conjugate_gradient::fletcher_reeves_method< fp_type >::fletcher_reeves_method
(
            f_line_minimiser_t< fp_type > const & line_search_minimiser,
            std::size_t const & max_iters = 100,
            fp_type arg_tol = 1e-4,
            fp_type grad_tol = 1e-4,
            fp_type fun_tol = 1e-4 )  [inline], [explicit]
```

Construct a new fletcher reeves method object.

Parameters

line_search_minimiser	line method to be used in finding the minimiser
max_iters	maximum number of iterations
arg_tol	tolerance for stopping criteria
grad_tol	tolerance for gradient
fun_tol	tolerance for a value of objective function

6.18.3 Member Function Documentation

6.18.3.1 minimize()

```
template<typename fp_type >
std::tuple< om_unconstrained_methods::om_conjugate_gradient::vector_t< fp_type >, fp_type,
std::size_t > om_unconstrained_methods::om_conjugate_gradient::fletcher_reeves_method< fp_↩
type >::minimize (
            f_vector_t< fp_type > objective,
            vector_arg_t< fp_type > const & init_guess ) const
```

Function method that minimises the objective function.

Parameters

objective	objective function
init_guess	initial guess

Returns

std::tuple<vector_t<fp_type>, fp_type, std::size_t>

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

- include/unconstrained_methods/multi_dim/conjugate_gradient/om_fletcher_reeves.hpp

6.19 om_differentiation::forward_difference< order, fp_type, typename > Struct Template Reference

forward difference functor

```
#include <om_differentiation.hpp>
```

6.19.1 Detailed Description

template<std::size_t order, typename fp_type, typename = typename std::enable_if< std::is_floating_point<fp_type>↩ ::value>::type>
struct om_differentiation::forward_difference< order, fp_type, typename >

forward difference functor

Template Parameters

order	order of difference
fp_type	
std::enable_if<	std::is_floating_point<fp_type>::value>::type

order = 0,order = 1 currently supported

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

- include/utilities/om_differentiation.hpp

6.20 om_differentiation::forward_difference< 0, fp_type > Struct Template Reference

Public Member Functions

- vector_t< fp_type > **operator()** (f_vector_t< fp_type > fun, vector_arg_t< fp_type > const &args) const

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

- include/utilities/om_differentiation.hpp

6.21 om_differentiation::forward_difference< 1, fp_type > Struct Template Reference

Public Member Functions

- vector_t< fp_type > **operator()** (f_vector_t< fp_type > fun, vector_arg_t< fp_type > const &args) const

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

- include/utilities/om_differentiation.hpp

6.22 om_differentiation_traits::forward_difference_trait< fp_type > Struct Template Reference

forward difference trait

```
#include <om_differentiation_traits.hpp>
```

Static Public Attributes

- static constexpr fp_type **step_size** = 10e-6

6.22.1 Detailed Description

template<typename fp_type>
struct om_differentiation_traits::forward_difference_trait< fp_type >

forward difference trait

Template Parameters

fp_type	fp_type is a floating-point template parameter

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

- include/utilities/om_differentiation_traits.hpp

6.23 om_common::furthest_from< count, fp_type, typename, type > Struct Template Reference

furthest_from functor

```
#include <om_common.hpp>
```

6.23.1 Detailed Description

template<std::size_t count, typename fp_type = double, typename = typename std::enable_if<count >= 2 && count <= 3, ↩ ::type>
struct om_common::furthest_from< count, fp_type, typename, type >

furthest_from functor

Template Parameters

count	number of points to measure the distance from
fp_type	fp_type is a floating-point template parameter

currently count = 2, count = 3 is supported

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

- include/utilities/om_common.hpp

6.24 om_common::furthest_from< 2, fp_type > Struct Template Reference

Public Member Functions

- fp_type **operator()** (fp_type const &target, fp_type const &x1, fp_type const &x2) const

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

- include/utilities/om_common.hpp

6.25 om_common::furthest_from< 3, fp_type > Struct Template Reference

Public Member Functions

- fp_type **operator()** (fp_type const &target, fp_type const &x1, fp_type const &x2, fp_type const &x3) const

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

- include/utilities/om_common.hpp

6.26 om_unconstrained_methods::om_line_methods::golden_section_method< fp_type, typename > Class Template Reference

Golden section method object.

```
#include <om_golden_section.hpp>
```

Public Types

- typedef fp_type **value_type**

Public Member Functions

- golden_section_method (range< fp_type > const &range, fp_type tolerance=1e-5, std::size_t max_iters=1000)

    Construct a new golden section method object.
- golden_section_method (golden_section_method const &copy)

    Copy constructor of a golden section method object.
- golden_section_method & operator= (golden_section_method const &copy)

    Assignment operator of a golden section method object.
- std::tuple< fp_type, fp_type, std::size_t, std::size_t > operator() (f_scalar_t< fp_type > &&fun) const

    Functor of a golden section method object.

6.26.1 Detailed Description

template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_point<fp_type>::value>::type>
class om_unconstrained_methods::om_line_methods::golden_section_method< fp_type, typename >

Golden section method object.

Template Parameters

fp_type	fp_type is floating point template parameter
std::enable_if<	std::is_floating_point<fp_type>::value>::type

6.26.2 Constructor & Destructor Documentation

6.26.2.1 golden_section_method() [1/2]

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_point<fp_type>::value>::type>
```

```
om_unconstrained_methods::om_line_methods::golden_section_method< fp_type, typename >::golden_section_method
(
             range< fp_type > const & range,
             fp_type tolerance = 1e-5,
             std::size_t max_iters = 1000 )  [inline]
```

Construct a new golden section method object.

Parameters

range	range of the minimiser
tolerance	tolerance of minimiser
max_iters	maximum number of iterations

6.26.2.2 golden_section_method() [2/2]

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
om_unconstrained_methods::om_line_methods::golden_section_method< fp_type, typename >::golden_section_method
(
             golden_section_method< fp_type, typename > const & copy )  [inline]
```

Copy constructor of a golden section method object.

Parameters

copy	copy is the object which we want to make a copy of

6.26.3 Member Function Documentation

6.26.3.1 operator()()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
std::tuple<fp_type, fp_type, std::size_t, std::size_t> om_unconstrained_methods::om_line_methods::golden_sect
fp_type, typename >::operator() (
             f_scalar_t< fp_type > && fun ) const  [inline]
```

Functor of a golden section method object.

Parameters

fun	objective function

Returns

std::tuple<fp_type, fp_type, std::size_t, std::size_t>

6.26.3.2 operator=()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
golden_section_method& om_unconstrained_methods::om_line_methods::golden_section_method< fp_↩
type, typename >::operator= (
            golden_section_method< fp_type, typename > const & copy )  [inline]
```

Assignment operator of a golden section method object.

Parameters

copy	

Returns

golden_section_method&

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

- include/unconstrained_methods/one_dim/om_golden_section.hpp

6.27 om_unconstrained_methods::om_conjugate_gradient::hestenes_stiefel_method< fp_type > Class Template Reference

Hestenes-Stiefel method object.

```
#include <om_hestenes_stiefel.hpp>
```

Inheritance diagram for om_unconstrained_methods::om_conjugate_gradient::hestenes_stiefel_method< fp_type >:

Collaboration diagram for om_unconstrained_methods::om_conjugate_gradient::hestenes_stiefel_method< fp_type >:

Public Member Functions

- hestenes_stiefel_method (f_line_minimiser_t< fp_type > const &line_search_minimiser, std::size_t const &max_iters=100, fp_type arg_tol=1e-4, fp_type grad_tol=1e-4, fp_type fun_tol=1e-4)

  Construct a new hestenes stiefel method object.
- std::tuple< vector_t< fp_type >, fp_type, std::size_t > minimize (f_vector_t< fp_type > objective, vector_arg_t< fp_type > const &init_guess) const

  Function method that minimises the objective function.

Additional Inherited Members

6.27.1 Detailed Description

template<typename fp_type = double>
class om_unconstrained_methods::om_conjugate_gradient::hestenes_stiefel_method< fp_type >

Hestenes-Stiefel method object.

Template Parameters

fp_type	fp_type is a floating-point template parameter

6.27.2 Constructor & Destructor Documentation

6.27.2.1 hestenes_stiefel_method()

```
template<typename fp_type = double>
om_unconstrained_methods::om_conjugate_gradient::hestenes_stiefel_method< fp_type >::hestenes_stiefel_method
(
            f_line_minimiser_t< fp_type > const & line_search_minimiser,
            std::size_t const & max_iters = 100,
            fp_type arg_tol = 1e-4,
            fp_type grad_tol = 1e-4,
            fp_type fun_tol = 1e-4 )  [inline], [explicit]
```

Construct a new hestenes stiefel method object.

Parameters

line_search_minimiser	line method to be used in finding the minimiser
max_iters	maximum number of iterations
arg_tol	tolerance for stopping criteria
grad_tol	tolerance for gradient
fun_tol	tolerance for a value of objective function

6.27.3 Member Function Documentation

6.27.3.1 minimize()

```
template<typename fp_type >
std::tuple< om_unconstrained_methods::om_conjugate_gradient::vector_t< fp_type >, fp_type,
std::size_t > om_unconstrained_methods::om_conjugate_gradient::hestenes_stiefel_method< fp↩
_type >::minimize (
             f_vector_t< fp_type > objective,
             vector_arg_t< fp_type > const & init_guess ) const
```

Function method that minimises the objective function.

Parameters

objective	objective function
init_guess	initial guess

Returns

std::tuple<vector_t<fp_type>, fp_type, std::size_t>

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

- include/unconstrained_methods/multi_dim/conjugate_gradient/om_hestenes_stiefel.hpp

6.28 om_common::max_arg< count, fp_type, typename, type > Struct Template Reference

max_arg functor returns argument at which a function takes maximum value

```
#include <om_common.hpp>
```

6.28.1 Detailed Description

template<std::size_t count, typename fp_type = double, typename = typename std::enable_if<count >= 2 && count <= 3, ↩ ::type>
struct om_common::max_arg< count, fp_type, typename, type >

max_arg functor returns argument at which a function takes maximum value

Template Parameters

count	number of arguments
fp_type	fp_type is a floating-point template argument

count = 2,count = 3 currently supported

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

- include/utilities/om_common.hpp

6.29 om_common::max_arg< 2, fp_type > Struct Template Reference

Public Member Functions

- std::pair< fp_type, fp_type > **operator()** (f_scalar_t< fp_type > fun, fp_type const &first, fp_type const &second) const

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

- include/utilities/om_common.hpp

6.30 om_common::max_arg< 3, fp_type > Struct Template Reference

Public Member Functions

- std::pair< fp_type, fp_type > **operator()** (f_scalar_t< fp_type > fun, fp_type const &first, fp_type const &second, fp_type const &third) const

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

- include/utilities/om_common.hpp

6.31 om_common::min_arg< count, fp_type, typename, type > Struct Template Reference

min_arg functor retuns argument as which a function takes minimum value

```
#include <om_common.hpp>
```

6.31.1 Detailed Description

template<std::size_t count, typename fp_type = double, typename = typename std::enable_if<count >= 2 && count <= 3, ↩ ::type>
struct om_common::min_arg< count, fp_type, typename, type >

min_arg functor retuns argument as which a function takes minimum value

Template Parameters

count	number of arguments
fp_type	fp_type is a floating-point template parameter

count = 2,count = 3 currently supported

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

- include/utilities/om_common.hpp

6.32 om_common::min_arg< 2, fp_type > Struct Template Reference

Public Member Functions

- std::pair< fp_type, fp_type > **operator()** (f_scalar_t< fp_type > fun, fp_type const &first, fp_type const &second) const

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

- include/utilities/om_common.hpp

6.33 om_common::min_arg< 3, fp_type > Struct Template Reference

Public Member Functions

- std::pair< fp_type, fp_type > **operator()** (f_scalar_t< fp_type > fun, fp_type const &first, fp_type const &second, fp_type const &third) const

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

- include/utilities/om_common.hpp

6.34 om_test_helpers::minimizer_helper< fp_type > Struct Template Reference

Helper for optimisation methods.

```
#include <om_test_helpers.hpp>
```

Collaboration diagram for om_test_helpers::minimizer_helper< fp_type >:

6.35 om_unconstrained_methods::om_zero_order::nelder_mead_↩ method< fp_type > Class Template Reference

Nelder-Mead method object.

```
#include <om_nelder_mead.hpp>
```

Public Member Functions

- nelder_mead_method (std::size_t const &max_iters=80, fp_type convergence_tol=10e-4, fp_type reflection←↩ _rho=0.5, fp_type expansion_rho=1.5, fp_type contraction_rho=0.25, fp_type shrinkage_rho=0.5)

  Construct a new nelder mead method object.
- nelder_mead_method (nelder_mead_method const &copy)

  Construct a new nelder mead method object.
- nelder_mead_method & operator= (nelder_mead_method const &copy)

  Assignment operator of a nelder mead method object.
- void set_max_iterations (std::size_t const &iters)

  Set the max iterations object.
- void set_converge_tolerance (fp_type converge_tol)

  Set the converge tolerance object.
- void set_reflection_rho (fp_type value)

  Set the reflection rho object.
- void set_expansion_rho (fp_type value)

  Set the expansion rho object.
- void set_contraction_rho (fp_type value)

  Set the contraction rho object.
- void set_shrinkage_rho (fp_type value)

  Set the shrinkage rho object.
- std::tuple< vector_t< fp_type >, fp_type, std::size_t > minimize (f_vector_t< fp_type > objective, vector←↩ _arg_t< fp_type > const &init_guess) const

  Function method that minimises the objective function.

6.35.1 Detailed Description

template<typename fp_type = double>
class om_unconstrained_methods::om_zero_order::nelder_mead_method< fp_type >

Nelder-Mead method object.

Template Parameters

fp_type	fp_type is a floating-point template parameter

6.35.2 Constructor & Destructor Documentation

6.35.2.1 nelder_mead_method() [1/2]

```
template<typename fp_type = double>
om_unconstrained_methods::om_zero_order::nelder_mead_method< fp_type >::nelder_mead_method (
              std::size_t const & max_iters = 80,
              fp_type convergence_tol = 10e-4,
              fp_type reflection_rho = 0.5,
              fp_type expansion_rho = 1.5,
              fp_type contraction_rho = 0.25,
              fp_type shrinkage_rho = 0.5 )  [inline]
```

Construct a new nelder mead method object.

Parameters

max_iters	maximum number of iterations
convergence_tol	tolerance for convergence
reflection_rho	reflection rho
expansion_rho	expansion rho
contraction_rho	contraction rho
shrinkage_rho	shrinkage rho

6.35.2.2 nelder_mead_method() [2/2]

```
template<typename fp_type = double>
om_unconstrained_methods::om_zero_order::nelder_mead_method< fp_type >::nelder_mead_method (
            nelder_mead_method< fp_type > const & copy )  [inline]
```

Construct a new nelder mead method object.

Parameters

copy	copy is the object which we want to make a copy of

6.35.3 Member Function Documentation

6.35.3.1 minimize()

```
template<typename fp_type >
std::tuple< om_zero_order::vector_t< fp_type >, fp_type, std::size_t > om_unconstrained_methods::om_zero_ord
fp_type >::minimize (
            f_vector_t< fp_type > objective,
            vector_arg_t< fp_type > const & init_guess ) const
```

Function method that minimises the objective function.

Parameters

objective	objective function
init_guess	initial guess

Returns

std::tuple<vector_t<fp_type>, fp_type, std::size_t>

6.35.3.2 operator=()

```
template<typename fp_type = double>
nelder_mead_method& om_unconstrained_methods::om_zero_order::nelder_mead_method< fp_type >↩
::operator= (
            nelder_mead_method< fp_type > const & copy )  [inline]
```

Assignment operator of a nelder mead method object.

Parameters

copy	

Returns

nelder_mead_method&

6.35.3.3 set_contraction_rho()

```
template<typename fp_type = double>
void om_unconstrained_methods::om_zero_order::nelder_mead_method< fp_type >::set_contraction↩
_rho (
            fp_type value )  [inline]
```

Set the contraction rho object.

Parameters

value	value of contraction rho

6.35.3.4 set_converge_tolerance()

```
template<typename fp_type = double>
void om_unconstrained_methods::om_zero_order::nelder_mead_method< fp_type >::set_converge_↩
tolerance (
            fp_type converge_tol )  [inline]
```

Set the converge tolerance object.

Parameters

converge_tol	tolerance for convergance

6.35.3.5 set_expansion_rho()

```
template<typename fp_type = double>
void om_unconstrained_methods::om_zero_order::nelder_mead_method< fp_type >::set_expansion_rho
(
             fp_type value )  [inline]
```

Set the expansion rho object.

Parameters

value	value of expansion rho

6.35.3.6 set_max_iterations()

```
template<typename fp_type = double>
void om_unconstrained_methods::om_zero_order::nelder_mead_method< fp_type >::set_max_iterations
(
             std::size_t const & iters )  [inline]
```

Set the max iterations object.

Parameters

iters	maximum number of iterations

6.35.3.7 set_reflection_rho()

```
template<typename fp_type = double>
void om_unconstrained_methods::om_zero_order::nelder_mead_method< fp_type >::set_reflection_↩
rho (
             fp_type value )  [inline]
```

Set the reflection rho object.

Parameters

value	value of reflection rho

6.35.3.8 set_shrinkage_rho()

```
template<typename fp_type = double>
void om_unconstrained_methods::om_zero_order::nelder_mead_method< fp_type >::set_shrinkage_rho
(
            fp_type value )  [inline]
```

Set the shrinkage rho object.

Parameters

value	value of shrinkage rho

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

- include/unconstrained_methods/multi_dim/zero_order/om_nelder_mead.hpp

6.36 om_unconstrained_methods::om_conjugate_gradient::polak_ribiere_method< fp_type > Class Template Reference

Polak-Ribiere method object.

`#include <om_polak_ribiere.hpp>`

Inheritance diagram for om_unconstrained_methods::om_conjugate_gradient::polak_ribiere_method< fp_type >:

Collaboration diagram for om_unconstrained_methods::om_conjugate_gradient::polak_ribiere_method< fp_type >:

Public Member Functions

- polak_ribiere_method (f_line_minimiser_t< fp_type > const &line_search_minimiser, std::size_t const &max_iters=100, fp_type arg_tol=1e-4, fp_type grad_tol=1e-4, fp_type fun_tol=1e-4)

  Construct a new polak ribiere method object.
- std::tuple< vector_t< fp_type >, fp_type, std::size_t > minimize (f_vector_t< fp_type > objective, vector_arg_t< fp_type > const &init_guess) const

  Function method that minimises the objective function.

Additional Inherited Members

6.36.1 Detailed Description

template<typename fp_type = double>
class om_unconstrained_methods::om_conjugate_gradient::polak_ribiere_method< fp_type >

Polak-Ribiere method object.

Template Parameters

fp_type	fp_type is a floating-point template parameter

6.36.2 Constructor & Destructor Documentation

6.36.2.1 polak_ribiere_method()

```
template<typename fp_type = double>
om_unconstrained_methods::om_conjugate_gradient::polak_ribiere_method< fp_type >::polak_ribiere_method
(
            f_line_minimiser_t< fp_type > const & line_search_minimiser,
            std::size_t const & max_iters = 100,
            fp_type arg_tol = 1e-4,
            fp_type grad_tol = 1e-4,
            fp_type fun_tol = 1e-4 )  [inline], [explicit]
```

Construct a new polak ribiere method object.

Parameters

line_search_minimiser	line method to be used in finding the minimiser
max_iters	maximum number of iterations
arg_tol	tolerance for stopping criteria
grad_tol	tolerance for gradient
fun_tol	tolerance for a value of objective function

6.36.3 Member Function Documentation

6.36.3.1 minimize()

```
template<typename fp_type >
std::tuple< om_unconstrained_methods::om_conjugate_gradient::vector_t< fp_type >, fp_type,
std::size_t > om_unconstrained_methods::om_conjugate_gradient::polak_ribiere_method< fp_type
>::minimize (
            f_vector_t< fp_type > objective,
            vector_arg_t< fp_type > const & init_guess ) const
```

Function method that minimises the objective function.

Parameters

objective	objective function
init_guess	initial guess

Returns

std::tuple<vector_t<fp_type>, fp_type, std::size_t>

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

- include/unconstrained_methods/multi_dim/conjugate_gradient/om_polak_ribiere.hpp

6.37 om_unconstrained_methods::om_zero_order::powell_conjugate_method< fp_type > Class Template Reference

Powell conjugate method object.

```
#include <om_powell_conjugate.hpp>
```

Public Member Functions

- powell_conjugate_method (f_line_minimiser_t< fp_type > const &line_search_minimiser, std::size_t const &max_iters=50, fp_type convergence_tol=10e-4)

  Construct a new powell conjugate method object.
- powell_conjugate_method (powell_conjugate_method const &copy)

  Copy constructor a new powell conjugate method object.
- powell_conjugate_method & operator= (powell_conjugate_method const &copy)

  Assignment operator of a powell conjugate method object.
- void set_max_iterations (std::size_t const &iters)

  Set the max iterations object.
- void set_converge_tolerance (double converge_tol)

  Set the converge tolerance object.
- std::tuple< vector_t< fp_type >, fp_type, std::size_t > minimize (f_vector_t< fp_type > objective, vector_arg_t< fp_type > const &init_guess) const

  Function method that minimises the objective function.

6.37.1 Detailed Description

template<typename fp_type = double>
class om_unconstrained_methods::om_zero_order::powell_conjugate_method< fp_type >

Powell conjugate method object.

Template Parameters

fp_type	fp_type is a floating-point template parameter

6.37.2 Constructor & Destructor Documentation

6.37.2.1 powell_conjugate_method() [1/2]

```
template<typename fp_type = double>
om_unconstrained_methods::om_zero_order::powell_conjugate_method< fp_type >::powell_conjugate_method
(
            f_line_minimiser_t< fp_type > const & line_search_minimiser,
            std::size_t const & max_iters = 50,
            fp_type convergence_tol = 10e-4 )  [inline]
```

Construct a new powell conjugate method object.

Parameters

line_search_minimiser	line method to be used in finding the minimiser
max_iters	maximum number of iterations
convergence_tol	tolerance for convergance

6.37.2.2 powell_conjugate_method() [2/2]

```
template<typename fp_type = double>
om_unconstrained_methods::om_zero_order::powell_conjugate_method< fp_type >::powell_conjugate_method
(
            powell_conjugate_method< fp_type > const & copy )  [inline]
```

Copy constructor a new powell conjugate method object.

Parameters

copy	copy is the object which we want to make a copy of

6.37.3 Member Function Documentation

6.37.3.1 minimize()

```
template<typename fp_type >
std::tuple< om_zero_order::vector_t< fp_type >, fp_type, std::size_t > om_unconstrained_methods::om_zero_ord
fp_type >::minimize (
            om_zero_order::f_vector_t< fp_type > objective,
            om_zero_order::vector_arg_t< fp_type > const & init_guess ) const
```

Function method that minimises the objective function.

Parameters

objective	objective function
init_guess	initial guess

Returns

std::tuple<vector_t<fp_type>, fp_type, std::size_t>

6.37.3.2 operator=()

```
template<typename fp_type = double>
powell_conjugate_method& om_unconstrained_methods::om_zero_order::powell_conjugate_method<
fp_type >::operator= (
             powell_conjugate_method< fp_type > const & copy )  [inline]
```

Assignment operator of a powell conjugate method object.

Parameters

copy	

Returns

powell_conjugate_method&

6.37.3.3 set_converge_tolerance()

```
template<typename fp_type = double>
void om_unconstrained_methods::om_zero_order::powell_conjugate_method< fp_type >::set_converge↩
_tolerance (
             double converge_tol )  [inline]
```

Set the converge tolerance object.

Parameters

converge_tol	tolerance for convergance

6.37.3.4 set_max_iterations()

```
template<typename fp_type = double>
```

```
void om_unconstrained_methods::om_zero_order::powell_conjugate_method< fp_type >::set_max_↩
iterations (
            std::size_t const & iters )  [inline]
```

Set the max iterations object.

Parameters

iters	maximum number of iterations

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

- include/unconstrained_methods/multi_dim/zero_order/om_powell_conjugate.hpp

6.38 om_unconstrained_methods::om_line_methods::powell_method< fp_type, typename > Class Template Reference

Powell method object.

```
#include <om_powell.hpp>
```

Public Types

- typedef fp_type **value_type**

Public Member Functions

- powell_method (range< fp_type > const &range, fp_type tolerance=1e-5, std::size_t max_ites=1000)

  Construct a new powell method object.
- powell_method (range< fp_type > const &range, fp_type step, fp_type max_step, fp_type tolerance=1e-5, std::size_t max_ites=1000)

  Construct a new powell method object.
- powell_method (powell_method const &copy)

  Copy constructor of a new powell method object.
- powell_method & operator= (powell_method const &copy)

  Assignment operator of a powell method object.
- std::tuple< fp_type, fp_type, std::size_t, std::size_t > operator() (f_scalar_t< fp_type > &&fun) const

  Functor of a powell method object.

6.38.1 Detailed Description

template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_point<fp_type>::value>::type> class om_unconstrained_methods::om_line_methods::powell_method< fp_type, typename >

Powell method object.

Template Parameters

fp_type	fp_type is floating-point template parameter
std::enable_if<	std::is_floating_point<fp_type>::value>::type

6.38.2 Constructor & Destructor Documentation

6.38.2.1 powell_method() [1/3]

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
om_unconstrained_methods::om_line_methods::powell_method< fp_type, typename >::powell_method (
            range< fp_type > const & range,
            fp_type tolerance = 1e-5,
            std::size_t max_ites = 1000 )  [inline]
```

Construct a new powell method object.

Parameters

range	range of the minimiser
tolerance	tolerance of the minimiser
max_ites	maximum number of iterations

6.38.2.2 powell_method() [2/3]

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
om_unconstrained_methods::om_line_methods::powell_method< fp_type, typename >::powell_method (
            range< fp_type > const & range,
            fp_type step,
            fp_type max_step,
            fp_type tolerance = 1e-5,
            std::size_t max_ites = 1000 )  [inline]
```

Construct a new powell method object.

Parameters

range	range of the minimiser
step	size of the step of the minimiser
max_step	maximum size of the step of the minimiser
tolerance	tolerance of the minimiser
max_ites	maximum number of iterations

6.38.2.3 powell_method() [3/3]

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
om_unconstrained_methods::om_line_methods::powell_method< fp_type, typename >::powell_method (
            powell_method< fp_type, typename > const & copy )  [inline]
```

Copy constructor of a new powell method object.

Parameters

copy	

6.38.3 Member Function Documentation

6.38.3.1 operator()()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
std::tuple<fp_type, fp_type, std::size_t, std::size_t> om_unconstrained_methods::om_line_methods::powell_meth
fp_type, typename >::operator() (
            f_scalar_t< fp_type > && fun ) const  [inline]
```

Functor of a powell method object.

Parameters

fun	objective function

Returns

std::tuple<fp_type, fp_type, std::size_t, std::size_t>

6.38.3.2 operator=()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
powell_method& om_unconstrained_methods::om_line_methods::powell_method< fp_type, typename >↩
::operator= (
            powell_method< fp_type, typename > const & copy )  [inline]
```

Assignment operator of a powell method object.

Parameters

copy	

Returns

powell_method&

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

- include/unconstrained_methods/one_dim/om_powell.hpp

6.39 om_unconstrained_methods::om_quasi_newton::quasi_newton_↩ base< fp_type, typename > Class Template Reference

Quasi-Newton base class.

```
#include <om_quasi_newton_base.hpp>
```

Collaboration diagram for om_unconstrained_methods::om_quasi_newton::quasi_newton_base< fp_type, typename >:

Public Member Functions

- quasi_newton_base (f_line_minimiser_t< fp_type > const &line_search_minimiser, std::size_t const &max↩ _iters=100, fp_type arg_tol=1e-4, fp_type grad_tol=1e-4, fp_type fun_tol=1e-4)

  Construct a new quasi newton base object.
- quasi_newton_base (quasi_newton_base const &copy)

  Construct a new quasi newton base object.
- quasi_newton_base & operator= (quasi_newton_base const &copy)

  Assignment operator of a quasi newton base object.
- void set_max_iterations (std::size_t const &iters)

  Set the max iterations object.
- void set_arg_tolerance (fp_type arg_tol)

  Set the stopping criteria tolerance object.
- void set_fun_tolerance (fp_type fun_tol)

  Set the fun tolerance object.
- void set_grad_tolerance (fp_type grad_tol)

  Set the grad tolerance object.

Protected Attributes

- fp_type **arg_tol_**
- fp_type **grad_tol_**
- fp_type **fun_tol_**
- std::size_t **max_iters_**
- f_line_minimiser_t< fp_type > **lsm_**

6.39.1 Detailed Description

template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_point<fp_type>::value>::type> class om_unconstrained_methods::om_quasi_newton::quasi_newton_base< fp_type, typename >

Quasi-Newton base class.

Template Parameters

fp_type	fp_type is a floating-point template parameter
std::enable_if<	std::is_floating_point<fp_type>::value>::type

6.39.2 Constructor & Destructor Documentation

6.39.2.1 quasi_newton_base() [1/2]

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
om_unconstrained_methods::om_quasi_newton::quasi_newton_base< fp_type, typename >::quasi_newton_base
(
            f_line_minimiser_t< fp_type > const & line_search_minimiser,
            std::size_t const & max_iters = 100,
            fp_type arg_tol = 1e-4,
            fp_type grad_tol = 1e-4,
            fp_type fun_tol = 1e-4 )  [inline]
```

Construct a new quasi newton base object.

Parameters

line_search_minimiser	line method to be used in finding the minimiser
max_iters	maximum number of iterations
arg_tol	tolerance for stopping criteria
grad_tol	tolerance for gradient
fun_tol	tolerance for a value of objective function

6.39.2.2 quasi_newton_base() [2/2]

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
om_unconstrained_methods::om_quasi_newton::quasi_newton_base< fp_type, typename >::quasi_newton_base
(
            quasi_newton_base< fp_type, typename > const & copy )  [inline]
```

Construct a new quasi newton base object.

Parameters

copy	copy is the object which we want to make a copy of

6.39.3 Member Function Documentation

6.39.3.1 operator=()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
quasi_newton_base& om_unconstrained_methods::om_quasi_newton::quasi_newton_base< fp_type,
typename >::operator= (
            quasi_newton_base< fp_type, typename > const & copy )  [inline]
```

Assignment operator of a quasi newton base object.

Parameters

copy	

Returns

quasi_newton_base&

6.39.3.2 set_arg_tolerance()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
void om_unconstrained_methods::om_quasi_newton::quasi_newton_base< fp_type, typename >::set_↩
arg_tolerance (
            fp_type arg_tol )  [inline]
```

Set the stopping criteria tolerance object.

Parameters

arg_tol	tolerance for stopping criteria

6.39.3.3 set_fun_tolerance()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
void om_unconstrained_methods::om_quasi_newton::quasi_newton_base< fp_type, typename >::set_↩
fun_tolerance (
            fp_type fun_tol )  [inline]
```

Set the fun tolerance object.

Parameters

fun_tol	tolerance for a value of objective function

6.39.3.4 set_grad_tolerance()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
void om_unconstrained_methods::om_quasi_newton::quasi_newton_base< fp_type, typename >::set_↩
grad_tolerance (
            fp_type grad_tol )  [inline]
```

Set the grad tolerance object.

Parameters

grad_tol	tolerance for gardient

6.39.3.5 set_max_iterations()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
void om_unconstrained_methods::om_quasi_newton::quasi_newton_base< fp_type, typename >::set_↩
max_iterations (
            std::size_t const & iters )  [inline]
```

Set the max iterations object.

Parameters

iters	maximum number of iterations

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

- include/unconstrained_methods/multi_dim/quasi_newton/om_quasi_newton_base.hpp

6.40 om_utilities::random_vectors_from_guess< fp_type, distribution, typename > Struct Template Reference

Random vectors from guess functor.

```
#include <om_utilities.hpp>
```

Public Member Functions

- std::vector< vector_t< fp_type > > **operator()** (std::size_t N, vector_t< fp_type > const &init_guess)

6.40.1 Detailed Description

template<typename fp_type = double, template< typename > typename distribution = std::normal_distribution, typename = typename std::enable_if_t<std::is_floating_point<fp_type>::value>>
struct om_utilities::random_vectors_from_guess< fp_type, distribution, typename >

Random vectors from guess functor.

Template Parameters

fp_type	fp_type is a floating-point template parameter
distribution	distribution of random generator
std::enable_if_t<std::is_floating_point<fp_type>::value>	

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

- include/utilities/om_utilities.hpp

6.41 om_utilities::range< fp_type, typename > Class Template Reference

Represents a one dimensional range.

```
#include <om_utilities.hpp>
```

Public Member Functions

- range (fp_type const &low, fp_type const &high)
  Construct a new range object.
- range ()
  Construct a new range object.
- range (range< fp_type > const &copy)
  Construct a new range object.
- range< fp_type > & operator= (range< fp_type > const &copy)
  Copy assignment operator of a range object.
- range (range< fp_type > &&other)
  Move constructor of a range object.
- range< fp_type > & operator= (range< fp_type > &&other)
  Move assignment of a range object.
- const fp_type & low () const
  Returns low end of the range.
- const fp_type & high () const
  Returns high end of the range.
- std::pair< fp_type, fp_type > low_high () const
  Returns a pair of low high end of the range.
- fp_type spread () const
  Returns a spread between high and low end of the range.

6.41.1 Detailed Description

**template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_point<fp_type>::value>::type>
class om_utilities::range< fp_type, typename >**

Represents a one dimensional range.

Template Parameters

fp_type	fp_type is a floating-point template parameter
std::enable_if<	std::is_floating_point<fp_type>::value>::type

6.41.2 Constructor & Destructor Documentation

6.41.2.1 range() [1/4]

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
om_utilities::range< fp_type, typename >::range (
             fp_type const & low,
             fp_type const & high )  [inline]
```

Construct a new range object.

Parameters

low	low value of a range
high	high value of a range

6.41.2.2 range() [2/4]

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
om_utilities::range< fp_type, typename >::range ( )  [inline]
```

Construct a new range object.

6.41.2.3 range() [3/4]

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
om_utilities::range< fp_type, typename >::range (
            range< fp_type > const & copy )  [inline]
```

Construct a new range object.

Parameters

copy	copy is the object which we want to make a copy of

6.41.2.4 range() [4/4]

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↵
point<fp_type>::value>::type>
om_utilities::range< fp_type, typename >::range (
            range< fp_type > && other )  [inline]
```

Move constructor of a range object.

Parameters

other	

6.41.3 Member Function Documentation

6.41.3.1 high()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↵
point<fp_type>::value>::type>
const fp_type& om_utilities::range< fp_type, typename >::high ( ) const  [inline]
```

Returns high end of the range.

Returns

fp_type const&

6.41.3.2 low()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↵
point<fp_type>::value>::type>
const fp_type& om_utilities::range< fp_type, typename >::low ( ) const  [inline]
```

Returns low end of the range.

Returns

fp_type const&

6.41.3.3 low_high()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
std::pair<fp_type, fp_type> om_utilities::range< fp_type, typename >::low_high ( ) const
[inline]
```

Returns a pair of low high end of the range.

Returns

std::pair<fp_type, fp_type>

6.41.3.4 operator=() [1/2]

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
range<fp_type>& om_utilities::range< fp_type, typename >::operator= (
            range< fp_type > && other )  [inline]
```

Move assignment of a range object.

Parameters

other	

Returns

range<fp_type>&

6.41.3.5 operator=() [2/2]

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
range<fp_type>& om_utilities::range< fp_type, typename >::operator= (
            range< fp_type > const & copy )  [inline]
```

Copy assignment operator of a range object.

Parameters

copy	

Returns

range<fp_type>&

6.41.3.6 spread()

```
template<typename fp_type = double, typename = typename std::enable_if< std::is_floating_↩
point<fp_type>::value>::type>
fp_type om_utilities::range< fp_type, typename >::spread ( ) const  [inline]
```

Returns a spread between high and low end of the range.

Returns

fp_type

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

- include/utilities/om_utilities.hpp

6.42 om_unconstrained_methods::om_steepest_descent::steepest_descent_method< fp_type > Class Template Reference

Steepest descent method object.

```
#include <om_steepest_descent.hpp>
```

Public Member Functions

- steepest_descent_method (f_line_minimiser_t< fp_type > const &line_search_minimiser, std::size_t const &max_iters=100, fp_type arg_tol=1e-4, fp_type grad_tol=1e-4, fp_type fun_tol=1e-4)

  Construct a new steepest descent method object.
- steepest_descent_method (steepest_descent_method const &copy)

  Copy constructor of a steepest descent method object.
- steepest_descent_method & operator= (steepest_descent_method const &copy)

  Assignment operator of a steepest descent method object.
- void set_arg_tolerance (fp_type arg_tol)

  Set the stopping criteria tolerance object.
- void set_fun_tolerance (fp_type fun_tol)

  Set the fun tolerance object.
- void set_grad_tolerance (fp_type grad_tol)

  Set the grad tolerance object.
- void set_max_iterations (std::size_t const &iters)

  Set the max iterations object.
- std::tuple< vector_t< fp_type >, fp_type, std::size_t > minimize (f_vector_t< fp_type > objective, vector_arg_t< fp_type > const &init_guess) const

  Function method that minimises the objective function.

6.42.1 Detailed Description

template< typename fp_type = double >
class om_unconstrained_methods::om_steepest_descent::steepest_descent_method< fp_type >

Steepest descent method object.

Template Parameters

fp_type	fp_type is a floating-point template parameter

6.42.2 Constructor & Destructor Documentation

6.42.2.1 steepest_descent_method() [1/2]

```
template<typename fp_type = double>
om_unconstrained_methods::om_steepest_descent::steepest_descent_method< fp_type >::steepest_descent_method
(
            f_line_minimiser_t< fp_type > const & line_search_minimiser,
            std::size_t const & max_iters = 100,
            fp_type arg_tol = 1e-4,
            fp_type grad_tol = 1e-4,
            fp_type fun_tol = 1e-4 )  [inline], [explicit]
```

Construct a new steepest descent method object.

Parameters

line_search_minimiser	line method to be used in finding the minimiser
max_iters	maximum number of iterations
arg_tol	tolerance for stopping criteria
grad_tol	tolerance for gradient
fun_tol	tolerance for a value of objective function

6.42.2.2 steepest_descent_method() [2/2]

```
template<typename fp_type = double>
om_unconstrained_methods::om_steepest_descent::steepest_descent_method< fp_type >::steepest_descent_method
(
            steepest_descent_method< fp_type > const & copy )  [inline]
```

Copy constructor of a steepest descent method object.

Parameters

copy	copy is the object which we want to make a copy of

6.42.3 Member Function Documentation

6.42.3.1 minimize()

```
template<typename fp_type = double>
std::tuple< om_unconstrained_methods::om_steepest_descent::vector_t< fp_type >, fp_type,
std::size_t > om_unconstrained_methods::om_steepest_descent::steepest_descent_method< fp_type
>::minimize (
            f_vector_t< fp_type > objective,
            vector_arg_t< fp_type > const & init_guess ) const
```

Function method that minimises the objective function.

Parameters

objective	objective function
init_guess	initial guess

Returns

std::tuple<vector_t<fp_type>, fp_type, std::size_t>

6.42.3.2 operator=()

```
template<typename fp_type = double>
steepest_descent_method& om_unconstrained_methods::om_steepest_descent::steepest_descent_method<
fp_type >::operator= (
            steepest_descent_method< fp_type > const & copy ) [inline]
```

Assignment operator of a steepest descent method object.

Parameters

copy	

Returns

steepest_descent_method&

6.42.3.3 set_arg_tolerance()

```
template<typename fp_type = double>
void om_unconstrained_methods::om_steepest_descent::steepest_descent_method< fp_type >::set_←
arg_tolerance (
            fp_type arg_tol ) [inline]
```

Set the stopping criteria tolerance object.

Parameters

arg_tol	tolerance for stopping criteria

6.42.3.4 set_fun_tolerance()

```
template<typename fp_type = double>
void om_unconstrained_methods::om_steepest_descent::steepest_descent_method< fp_type >::set_↩
fun_tolerance (
             fp_type fun_tol )  [inline]
```

Set the fun tolerance object.

Parameters

fun_tol	tolerance for a value of function

6.42.3.5 set_grad_tolerance()

```
template<typename fp_type = double>
void om_unconstrained_methods::om_steepest_descent::steepest_descent_method< fp_type >::set_↩
grad_tolerance (
             fp_type grad_tol )  [inline]
```

Set the grad tolerance object.

Parameters

grad_tol	tolerance for gradient

6.42.3.6 set_max_iterations()

```
template<typename fp_type = double>
void om_unconstrained_methods::om_steepest_descent::steepest_descent_method< fp_type >::set_↩
max_iterations (
             std::size_t const & iters )  [inline]
```

Set the max iterations object.

Parameters

iters	maximum number of iterations

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

- include/unconstrained_methods/multi_dim/steepest_descent/om_steepest_descent.hpp

Index

beale_function
om_test_functions, 11
brent_method
om_unconstrained_methods::om_line_methods::brent_method<
fp_type, typename >, 28
broyden_fletcher_goldfarb_shanno_method
om_unconstrained_methods::om_quasi_newton::broyden_fletcher_goldfarb_shanno_method<
fp_type >, 30

conjugate_gradient_base
om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base<
fp_type, typename >, 35, 36
create_rao_test_collection
om_test_functions, 12

davidon_fletcher_powell_method
om_unconstrained_methods::om_quasi_newton::davidon_fletcher_powell_method<
fp_type >, 39

f_line_minimiser_t
om_types, 18
f_scalar_t
om_types, 18
f_vector_t
om_types, 18
fib
om_utilities, 23
fibonacci_method
om_unconstrained_methods::om_line_methods::fibonacci_method<
fp_type, typename >, 42, 43
fletcher_powell_helical_valley
om_test_functions, 12
fletcher_reeves_method
om_unconstrained_methods::om_conjugate_gradient::fletcher_reeves_method<
fp_type >, 45
freudenstein_roth_function
om_test_functions, 13

golden_section_method
om_unconstrained_methods::om_line_methods::golden_section_method<
fp_type, typename >, 49, 50

hestenes_stiefel_method
om_unconstrained_methods::om_conjugate_gradient::hestenes_stiefel_method<
fp_type >, 52
high
om_utilities::range< fp_type, typename >, 75

iqerp
om_utilities, 23

lerp
om_utilities, 25
low
om_utilities::range< fp_type, typename >, 75
low_high
om_utilities::range< fp_type, typename >, 75
matrix_t
om_types, 19
minimize
om_unconstrained_methods::om_conjugate_gradient::fletcher_reeve
fp_type >, 45
om_unconstrained_methods::om_conjugate_gradient::hestenes_stief
fp_type >, 52
om_unconstrained_methods::om_conjugate_gradient::polak_ribiere_
fp_type >, 61
om_unconstrained_methods::om_quasi_newton::broyden_fletcher_g
fp_type >, 31
om_unconstrained_methods::om_quasi_newton::davidon_fletcher_p
fp_type >, 39
om_unconstrained_methods::om_steepest_descent::steepest_desce
fp_type >, 80
om_unconstrained_methods::om_zero_order::nelder_mead_method
fp_type >, 57
om_unconstrained_methods::om_zero_order::powell_conjugate_met
fp_type >, 63

nelder_mead_method
om_unconstrained_methods::om_zero_order::nelder_mead_method
fp_type >, 56, 57
non_linear_function
om_test_functions, 13

om_common, 9
om_common::closest_to< 2, fp_type >, 34
om_common::closest_to< 3, fp_type >, 34
om_common::closest_to< count, fp_type, typename,
type >, 33
om_common::furthest_from< 2, fp_type >, 48
om_common::furthest_from< 3, fp_type >, 48
om_common::furthest_from< count, fp_type, typename,
type >, 47
om_common::max_arg< 2, fp_type >, 54
om_common::max_arg< 3, fp_type >, 54
om_common::max_arg< count, fp_type, typename,
type >, 53
om_common::min_arg< 2, fp_type >, 55
om_common::min_arg< 3, fp_type >, 55
om_common::min_arg< count, fp_type, typename, type
>, 54

om_differentiation, 9
om_differentiation::central_difference< 0, fp_type >, 32
om_differentiation::central_difference< 1, fp_type >, 33
om_differentiation::central_difference< order, fp_type, typename >, 32
om_differentiation::divided_difference< 0, fp_type >, 41
om_differentiation::divided_difference< 1, fp_type >, 41
om_differentiation::divided_difference< 2, fp_type >, 41
om_differentiation::divided_difference< order, fp_type, typename, type >, 40
om_differentiation::forward_difference< 0, fp_type >, 46
om_differentiation::forward_difference< 1, fp_type >, 47
om_differentiation::forward_difference< order, fp_type, typename >, 46
om_differentiation_traits, 10
om_differentiation_traits::central_difference_trait< fp_type >, 33
om_differentiation_traits::forward_difference_trait< fp_type >, 47
om_test_functions, 10
    beale_function, 11
    create_rao_test_collection, 12
    fletcher_powell_helical_valley, 12
    freudenstein_roth_function, 13
    non_linear_function, 13
    pi, 16
    powell_badly_scaled_function, 14
    powell_function, 14
    quadratic_function, 15
    rosenbrock_parabolic_valley, 15
    wood_function, 16
om_test_helpers, 17
om_test_helpers::minimizer_helper< fp_type >, 55
om_types, 17
    f_line_minimiser_t, 18
    f_scalar_t, 18
    f_vector_t, 18
    matrix_t, 19
    sptr_t, 19
    vector_arg_t, 19
    vector_t, 19
om_unconstrained_methods, 20
om_unconstrained_methods::om_conjugate_gradient, 20
om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base< fp_type, typename >, 34
    conjugate_gradient_base, 35, 36
    operator=, 36
    set_arg_tolerance, 37
    set_fun_tolerance, 37
    set_grad_tolerance, 37
    set_max_iterations, 38
om_unconstrained_methods::om_conjugate_gradient::fletcher_reeves_method< fp_type >, 44
    fletcher_reeves_method, 45
    minimize, 45
om_unconstrained_methods::om_conjugate_gradient::hestenes_stiefel_m
fp_type >, 51
    hestenes_stiefel_method, 52
    minimize, 52
om_unconstrained_methods::om_conjugate_gradient::polak_ribiere_meth
fp_type >, 60
    minimize, 61
    polak_ribiere_method, 61
om_unconstrained_methods::om_line_methods, 21
om_unconstrained_methods::om_line_methods::brent_method< fp_type, typename >, 27
    brent_method, 28
    operator(), 28
    operator=, 29
om_unconstrained_methods::om_line_methods::fibonacci_method< fp_type, typename >, 41
    fibonacci_method, 42, 43
    operator(), 43
    operator=, 43
om_unconstrained_methods::om_line_methods::golden_section_method<
fp_type, typename >, 49
    golden_section_method, 49, 50
    operator(), 50
    operator=, 51
om_unconstrained_methods::om_line_methods::powell_method< fp_type, typename >, 65
    operator(), 67
    operator=, 67
    powell_method, 66, 67
om_unconstrained_methods::om_quasi_newton, 21
om_unconstrained_methods::om_quasi_newton::broyden_fletcher_goldfa
fp_type >, 29
    broyden_fletcher_goldfarb_shanno_method, 30
    minimize, 31
om_unconstrained_methods::om_quasi_newton::davidon_fletcher_powell_
fp_type >, 38
    davidon_fletcher_powell_method, 39
    minimize, 39
om_unconstrained_methods::om_quasi_newton::quasi_newton_base< fp_type, typename >, 68
    operator=, 70
    quasi_newton_base, 69
    set_arg_tolerance, 70
    set_fun_tolerance, 70
    set_grad_tolerance, 71
    set_max_iterations, 71
om_unconstrained_methods::om_steepest_descent, 22
om_unconstrained_methods::om_steepest_descent::steepest_descent_m
fp_type >, 77
    minimize, 80
    operator=, 80
    set_arg_tolerance, 80
    set_fun_tolerance, 81
    set_grad_tolerance, 81
    set_max_iterations, 81
    steepest_descent_method, 79
om_unconstrained_methods::om_zero_order, 22
om_unconstrained_methods::om_zero_order::nelder_mead_method<

fp_type >, 55
minimize, 57
nelder_mead_method, 56, 57
operator=, 58
set_contraction_rho, 58
set_converge_tolerance, 58
set_expansion_rho, 59
set_max_iterations, 59
set_reflection_rho, 59
set_shrinkage_rho, 60
om_unconstrained_methods::om_zero_order::powell_conjugate_method<
fp_type >, 62
minimize, 63
operator=, 64
powell_conjugate_method, 63
set_converge_tolerance, 64
set_max_iterations, 64
om_utilities, 22
fib, 23
iqerp, 23
lerp, 25
sign, 25
om_utilities::cartesian_basis_vectors< fp_type, typename >, 31
om_utilities::random_vectors_from_guess< fp_type, distribution, typename >, 71
om_utilities::range< fp_type, typename >, 72
high, 75
low, 75
low_high, 75
operator=, 76
range, 73, 75
spread, 77
operator()
om_unconstrained_methods::om_line_methods::brent_method<
fp_type, typename >, 28
om_unconstrained_methods::om_line_methods::fibonacci_method<
fp_type, typename >, 43
om_unconstrained_methods::om_line_methods::golden_section_method<
fp_type, typename >, 50
om_unconstrained_methods::om_line_methods::powell_method<
fp_type, typename >, 67
operator=
om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base<
fp_type, typename >, 36
om_unconstrained_methods::om_line_methods::brent_method<
fp_type, typename >, 29
om_unconstrained_methods::om_line_methods::fibonacci_method<
fp_type, typename >, 43
om_unconstrained_methods::om_line_methods::golden_section_method<
fp_type, typename >, 51
om_unconstrained_methods::om_line_methods::powell_method<
fp_type, typename >, 67
om_unconstrained_methods::om_quasi_newton::quasi_newton_base<
fp_type, typename >, 70
om_unconstrained_methods::om_steepest_descent::steepest_descent_method<
fp_type >, 80
om_unconstrained_methods::om_zero_order::nelder_mead_method<

fp_type >, 58
om_unconstrained_methods::om_zero_order::powell_conjugate_met
fp_type >, 64
om_utilities::range< fp_type, typename >, 76

pi
om_test_functions, 16
polak_ribiere_method
om_unconstrained_methods::om_conjugate_gradient::polak_ribiere_
fp_type >, 61
powell_badly_scaled_function
om_test_functions, 14
powell_conjugate_method
om_unconstrained_methods::om_zero_order::powell_conjugate_met
fp_type >, 63
powell_function
om_test_functions, 14
powell_method
om_unconstrained_methods::om_line_methods::powell_method<
fp_type, typename >, 66, 67

quadratic_function
om_test_functions, 15
quasi_newton_base
om_unconstrained_methods::om_quasi_newton::quasi_newton_base
fp_type, typename >, 69

range
om_utilities::range< fp_type, typename >, 73, 75
rosenbrock_parabolic_valley
om_test_functions, 15

set_arg_tolerance
om_unconstrained_methods::om_conjugate_gradient::conjugate_gra
fp_type, typename >, 37
om_unconstrained_methods::om_quasi_newton::quasi_newton_base
fp_type, typename >, 70
om_unconstrained_methods::om_steepest_descent::steepest_desce
fp_type >, 80
set_contraction_rho
om_unconstrained_methods::om_zero_order::nelder_mead_method
fp_type >, 58
set_converge_tolerance
om_unconstrained_methods::om_zero_order::nelder_mead_method
fp_type >, 58
om_unconstrained_methods::om_zero_order::powell_conjugate_met
fp_type >, 64
set_expansion_rho
om_unconstrained_methods::om_zero_order::nelder_mead_method
fp_type >, 59
set_fun_tolerance
om_unconstrained_methods::om_conjugate_gradient::conjugate_gra
fp_type, typename >, 37
om_unconstrained_methods::om_quasi_newton::quasi_newton_base
fp_type, typename >, 70
om_unconstrained_methods::om_steepest_descent::steepest_desce
fp_type >, 81
set_grad_tolerance

om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base< fp_type, typename >, 37
om_unconstrained_methods::om_quasi_newton::quasi_newton_base< fp_type, typename >, 71
om_unconstrained_methods::om_steepest_descent::steepest_descent_method< fp_type >, 81
set_max_iterations
om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base< fp_type, typename >, 38
om_unconstrained_methods::om_quasi_newton::quasi_newton_base< fp_type, typename >, 71
om_unconstrained_methods::om_steepest_descent::steepest_descent_method< fp_type >, 81
om_unconstrained_methods::om_zero_order::nelder_mead_method< fp_type >, 59
om_unconstrained_methods::om_zero_order::powell_conjugate_method< fp_type >, 64
set_reflection_rho
om_unconstrained_methods::om_zero_order::nelder_mead_method< fp_type >, 59
set_shrinkage_rho
om_unconstrained_methods::om_zero_order::nelder_mead_method< fp_type >, 60
sign
om_utilities, 25
spread
om_utilities::range< fp_type, typename >, 77
sptr_t
om_types, 19
steepest_descent_method
om_unconstrained_methods::om_steepest_descent::steepest_descent_method< fp_type >, 79

vector_arg_t
om_types, 19
vector_t
om_types, 19

wood_function
om_test_functions, 16