

Optimization Methods

0.1.0

Generated by Doxygen 1.8.17

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 Namespace Documentation	7
4.1 om_test_functions Namespace Reference	7
4.1.1 Detailed Description	8
4.1.2 Function Documentation	8
4.1.2.1 powell_function()	8
4.1.2.2 quadratic_function()	8
4.1.2.3 rosenbrock_parabolic_valley()	9
4.1.3 Variable Documentation	9
4.1.3.1 pi	9
5 Class Documentation	11
5.1 om_unconstrained_methods::om_line_methods::brent_method< fp_type, typename > Class Template Reference	11
5.1.1 Detailed Description	11
5.1.2 Constructor & Destructor Documentation	12
5.1.2.1 brent_method() [1/2]	12
5.1.2.2 brent_method() [2/2]	12
5.1.3 Member Function Documentation	12
5.1.3.1 operator()()	12
5.1.3.2 operator=()	13
5.2 om_unconstrained_methods::om_quasi_newton::broyden_fletcher_goldfarb_shanno_method< fp_type > Class Template Reference	13
5.2.1 Detailed Description	14
5.2.2 Constructor & Destructor Documentation	14
5.2.2.1 broyden_fletcher_goldfarb_shanno_method()	14
5.2.3 Member Function Documentation	15
5.2.3.1 minimize()	15
5.3 om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base< fp_type, typename > Class Template Reference	15
5.3.1 Detailed Description	16
5.3.2 Constructor & Destructor Documentation	16
5.3.2.1 conjugate_gradient_base() [1/2]	16
5.3.2.2 conjugate_gradient_base() [2/2]	17
5.3.3 Member Function Documentation	17
5.3.3.1 operator=()	17

5.3.3.2	set_arg_tolerance()	18
5.3.3.3	set_fun_tolerance()	18
5.3.3.4	set_grad_tolerance()	18
5.3.3.5	set_max_iterations()	19
5.4	om_unconstrained_methods::om_quasi_newton::davidon_fletcher_powell_method< fp_type > Class Template Reference	19
5.4.1	Detailed Description	19
5.4.2	Constructor & Destructor Documentation	20
5.4.2.1	davidon_fletcher_powell_method()	20
5.4.3	Member Function Documentation	20
5.4.3.1	minimize()	20
5.5	om_unconstrained_methods::om_line_methods::fibonacci_method< fp_type, typename > Class Template Reference	21
5.5.1	Detailed Description	21
5.5.2	Constructor & Destructor Documentation	22
5.5.2.1	fibonacci_method() [1/2]	22
5.5.2.2	fibonacci_method() [2/2]	22
5.5.3	Member Function Documentation	22
5.5.3.1	operator()()	22
5.5.3.2	operator=()	23
5.6	om_unconstrained_methods::om_conjugate_gradient::fletcher_reeves_method< fp_type > Class Template Reference	23
5.6.1	Detailed Description	24
5.6.2	Constructor & Destructor Documentation	24
5.6.2.1	fletcher_reeves_method()	24
5.6.3	Member Function Documentation	25
5.6.3.1	minimize()	25
5.7	om_unconstrained_methods::om_line_methods::golden_section_method< fp_type, typename > Class Template Reference	25
5.7.1	Detailed Description	26
5.7.2	Constructor & Destructor Documentation	26
5.7.2.1	golden_section_method() [1/2]	26
5.7.2.2	golden_section_method() [2/2]	26
5.7.3	Member Function Documentation	27
5.7.3.1	operator()()	27
5.7.3.2	operator=()	27
5.8	om_unconstrained_methods::om_conjugate_gradient::hestenes_stiefel_method< fp_type > Class Template Reference	28
5.8.1	Detailed Description	28
5.8.2	Constructor & Destructor Documentation	28
5.8.2.1	hestenes_stiefel_method()	28
5.8.3	Member Function Documentation	29
5.8.3.1	minimize()	29

5.9 om_test_helpers::minimizer_helper< T > Struct Template Reference	29
5.10 om_unconstrained_methods::om_zero_order::nelder_mead_method< fp_type > Class Template Reference	30
5.10.1 Detailed Description	30
5.10.2 Constructor & Destructor Documentation	30
5.10.2.1 nelder_mead_method() [1/2]	31
5.10.2.2 nelder_mead_method() [2/2]	31
5.10.3 Member Function Documentation	31
5.10.3.1 minimize()	31
5.10.3.2 operator=()	32
5.10.3.3 set_contraction_rho()	32
5.10.3.4 set_converge_tolerance()	32
5.10.3.5 set_expansion_rho()	33
5.10.3.6 set_max_iterations()	33
5.10.3.7 set_reflection_rho()	33
5.10.3.8 set_shrinkage_rho()	34
5.11 om_unconstrained_methods::om_conjugate_gradient::polak_ribiere_method< fp_type > Class Template Reference	34
5.11.1 Detailed Description	35
5.11.2 Constructor & Destructor Documentation	36
5.11.2.1 polak_ribiere_method()	36
5.11.3 Member Function Documentation	36
5.11.3.1 minimize()	36
5.12 om_unconstrained_methods::om_zero_order::powell_conjugate_method< fp_type > Class Template Reference	37
5.12.1 Detailed Description	37
5.12.2 Constructor & Destructor Documentation	38
5.12.2.1 powell_conjugate_method() [1/2]	38
5.12.2.2 powell_conjugate_method() [2/2]	38
5.12.3 Member Function Documentation	38
5.12.3.1 minimize()	38
5.12.3.2 operator=()	39
5.12.3.3 set_converge_tolerance()	39
5.12.3.4 set_max_iterations()	39
5.13 om_unconstrained_methods::om_line_methods::powell_method< fp_type, typename > Class Template Reference	40
5.13.1 Detailed Description	40
5.13.2 Constructor & Destructor Documentation	41
5.13.2.1 powell_method() [1/3]	41
5.13.2.2 powell_method() [2/3]	41
5.13.2.3 powell_method() [3/3]	42
5.13.3 Member Function Documentation	42
5.13.3.1 operator()()	42

5.13.3.2 operator=()	42
5.14 om_unconstrained_methods::om_quasi_newton::quasi_newton_base< fp_type, typename > Class Template Reference	43
5.14.1 Detailed Description	43
5.14.2 Constructor & Destructor Documentation	44
5.14.2.1 quasi_newton_base() [1/2]	44
5.14.2.2 quasi_newton_base() [2/2]	44
5.14.3 Member Function Documentation	45
5.14.3.1 operator=()	45
5.14.3.2 set_arg_tolerance()	45
5.14.3.3 set_fun_tolerance()	45
5.14.3.4 set_grad_tolerance()	46
5.14.3.5 set_max_iterations()	46
5.15 om_unconstrained_methods::om_steepest_descent::steepest_descent_method< fp_type > Class Template Reference	46
5.15.1 Detailed Description	47
5.15.2 Constructor & Destructor Documentation	47
5.15.2.1 steepest_descent_method() [1/2]	47
5.15.2.2 steepest_descent_method() [2/2]	48
5.15.3 Member Function Documentation	48
5.15.3.1 minimize()	48
5.15.3.2 operator=()	49
5.15.3.3 set_arg_tolerance()	49
5.15.3.4 set_fun_tolerance()	49
5.15.3.5 set_grad_tolerance()	50
5.15.3.6 set_max_iterations()	50
Index	51

Chapter 1

Namespace Index

1.1 Namespace List

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

om_test_functions	Some classical test functions (designed by Rao)	7
-----------------------------------	---	---

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

om_unconstrained_methods::om_line_methods::brent_method< fp_type, typename >	11
om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base< fp_type, typename >	15
om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base< double >	15
om_unconstrained_methods::om_conjugate_gradient::fletcher_reeves_method< fp_type >	23
om_unconstrained_methods::om_conjugate_gradient::hestenes_stiefel_method< fp_type >	28
om_unconstrained_methods::om_conjugate_gradient::polak_ribiere_method< fp_type >	34
om_unconstrained_methods::om_line_methods::fibonacci_method< fp_type, typename >	21
om_unconstrained_methods::om_line_methods::golden_section_method< fp_type, typename >	25
om_test_helpers::minimizer_helper< T >	29
om_unconstrained_methods::om_zero_order::nelder_mead_method< fp_type >	30
om_unconstrained_methods::om_zero_order::powell_conjugate_method< fp_type >	37
om_unconstrained_methods::om_line_methods::powell_method< fp_type, typename >	40
om_unconstrained_methods::om_quasi_newton::quasi_newton_base< fp_type, typename >	43
om_unconstrained_methods::om_quasi_newton::quasi_newton_base< double >	43
om_unconstrained_methods::om_quasi_newton::broyden_fletcher_goldfarb_shanno_method< fp_↵ type >	13
om_unconstrained_methods::om_quasi_newton::davidon_fletcher_powell_method< fp_type >	19
om_unconstrained_methods::om_steepest_descent::steepest_descent_method< fp_type >	46

Chapter 3

Class Index

3.1 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	11
om_unconstrained_methods::om_quasi_newton::broyden_fletcher_goldfarb_shanno_method< fp_type >	
Broyden-Fletcher-Goldfarb-Shanno method object	13
om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base< fp_type, typename >	
Conjugate-gradient base class	15
om_unconstrained_methods::om_quasi_newton::davidon_fletcher_powell_method< fp_type >	
Davidon-Fletcher-Powell method object	19
om_unconstrained_methods::om_line_methods::fibonacci_method< fp_type, typename >	
Fibonacci method object	21
om_unconstrained_methods::om_conjugate_gradient::fletcher_reeves_method< fp_type >	
Fletcher-Reeves method object	23
om_unconstrained_methods::om_line_methods::golden_section_method< fp_type, typename >	
Golden section method object	25
om_unconstrained_methods::om_conjugate_gradient::hestenes_stiefel_method< fp_type >	
Hestenes-Stiefel method object	28
om_test_helpers::minimizer_helper< T >	29
om_unconstrained_methods::om_zero_order::nelder_mead_method< fp_type >	
Nelder-Mead method object	30
om_unconstrained_methods::om_conjugate_gradient::polak_ribiere_method< fp_type >	
Polak-Ribiere method object	34
om_unconstrained_methods::om_zero_order::powell_conjugate_method< fp_type >	
Powell conjugate method object	37
om_unconstrained_methods::om_line_methods::powell_method< fp_type, typename >	
Powell method object	40
om_unconstrained_methods::om_quasi_newton::quasi_newton_base< fp_type, typename >	
Quasi-Newton base class	43
om_unconstrained_methods::om_steepest_descent::steepest_descent_method< fp_type >	
Steepest descent method object	46

Chapter 4

Namespace Documentation

4.1 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 function.
- `template<typename fp_type >`
`fp_type powell_function (vector_arg_t< fp_type > const &args)`
Powell's quadratic function.
- `template<typename fp_type >`
`fp_type fletcher_powell_helical_valley (vector_arg_t< fp_type > const &args)`
- `template<typename fp_type >`
`fp_type non_linear_function (vector_arg_t< fp_type > const &args)`
- `template<typename fp_type >`
`fp_type freudenstein_roth_function (vector_arg_t< fp_type > const &args)`
- `template<typename fp_type >`
`fp_type powell_badly_scaled_function (vector_arg_t< fp_type > const &args)`
- `template<typename fp_type >`
`fp_type beale_function (vector_arg_t< fp_type > const &args)`
- `template<typename fp_type >`
`fp_type wood_function (vector_arg_t< fp_type > const &args)`
- `template<typename fp_type >`
`std::vector< sptr_t< minimizer_helper< fp_type > > > create_rao_test_collection ()`

Variables

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

4.1.1 Detailed Description

Some classical test functions (designed by Rao)

4.1.2 Function Documentation

4.1.2.1 powell_function()

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

Powell's quadratic function.

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

Template Parameters

<i>fp_type</i>	<i>fp_type</i> is a floating-point template parameter
----------------	---

Parameters

<i>args</i>	function arguments
-------------	--------------------

Returns

fp_type

4.1.2.2 quadratic_function()

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

Quadratic function.

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

Template Parameters

<i>fp_type</i>	<i>fp_type</i> is a floating-point template parameter
----------------	---

Parameters

<i>args</i>	function arguments
-------------	--------------------

Returns

fp_type

4.1.2.3 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

<i>fp_type</i>	<i>fp_type</i> is a floating-point template parameter
----------------	---

Parameters

<i>args</i>	arguments of the function
-------------	---------------------------

Returns

fp_type

4.1.3 Variable Documentation

4.1.3.1 pi

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

Pi definition used in the Rao test functions.

Template Parameters

<i>fp_type</i>	<i>fp_type</i> is a floating-point template parameter
----------------	---

Chapter 5

Class Documentation

5.1 `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 constructor of a brent method object.
- `brent_method & operator= (brent_method const ©)`
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.

5.1.1 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

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

5.1.2 Constructor & Destructor Documentation

5.1.2.1 `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

<i>range</i>	range of the minimiser
<i>tolerance</i>	tolerance of the minimiser
<i>max_iters</i>	maximum number of iterations

5.1.2.2 `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

<i>copy</i>	copy is the object which we want to make a copy of
-------------	--

5.1.3 Member Function Documentation

5.1.3.1 `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

<i>fun</i>	objective function
------------	--------------------

Returns

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

5.1.3.2 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

<i>copy</i>	
-------------	--

Returns

brent_method&

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

- include/unconstrained_methods/one_dim/om_brent.hpp

5.2 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_t< fp_type > const &init_guess) const
Function method that minimises the objective function.

Additional Inherited Members

5.2.1 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

<i>fp_type</i>	fp+type is a floating-point template parameter
----------------	--

5.2.2 Constructor & Destructor Documentation

5.2.2.1 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

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

5.2.3 Member Function Documentation

5.2.3.1 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

<i>objective</i>	objective function
<i>init_guess</i>	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

5.3 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 ©)
 - Construct a new conjugate gradient base object.
- [conjugate_gradient_base](#) & operator= (conjugate_gradient_base const ©)

- 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_](#)

5.3.1 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

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

5.3.2 Constructor & Destructor Documentation

5.3.2.1 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

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

5.3.2.2 `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

<i>copy</i>	copy is the object which we want to make a copy of
-------------	--

5.3.3 Member Function Documentation

5.3.3.1 `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

<i>copy</i>	
-------------	--

Returns

`conjugate_gradient_base&`

5.3.3.2 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

<i>arg_tol</i>	tolerance for a stopping criteria
----------------	-----------------------------------

5.3.3.3 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

<i>fun_tol</i>	tolerance for a value of objective function
----------------	---

5.3.3.4 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

<i>grad_tol</i>	tolerance for gradient
-----------------	------------------------

5.3.3.5 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

<i>iters</i>	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

5.4 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

5.4.1 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

<i>fp_type</i>	<i>fp_type</i> is a floating-point template parameter
----------------	---

5.4.2 Constructor & Destructor Documentation

5.4.2.1 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

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

5.4.3 Member Function Documentation

5.4.3.1 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

<i>objective</i>	objective function
<i>init_guess</i>	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

5.5 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 constructor of a fibonacci method object.
- [fibonacci_method](#) & [operator=](#) ([fibonacci_method](#) const ©)
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.

5.5.1 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

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

5.5.2 Constructor & Destructor Documentation

5.5.2.1 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

<i>range</i>	range of the minimiser
<i>tolerance</i>	tolerance of the minimiser
<i>max_iters</i>	maximum number of iterations

5.5.2.2 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

<i>copy</i>	copy is the object which we want to make a copy of
-------------	--

5.5.3 Member Function Documentation

5.5.3.1 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

<i>fun</i>	objective function
------------	--------------------

Returns

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

5.5.3.2 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

<i>copy</i>	
-------------	--

Returns

[fibonacci_method&](#)

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

- include/unconstrained_methods/one_dim/om_fibonacci.hpp

5.6 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_t< fp_type > const &init_guess) const
Function method that minimises the objective function.

Additional Inherited Members

5.6.1 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

<i>fp_type</i>	fp_type is a floating-point template parameter
----------------	--

5.6.2 Constructor & Destructor Documentation

5.6.2.1 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

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

5.6.3 Member Function Documentation

5.6.3.1 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

<i>objective</i>	objective function
<i>init_guess</i>	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

5.7 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 constructor of a golden section method object.
- [golden_section_method](#) & [operator=](#) ([golden_section_method](#) const ©)
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.

5.7.1 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

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

5.7.2 Constructor & Destructor Documentation

5.7.2.1 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

<i>range</i>	range of the minimiser
<i>tolerance</i>	tolerance of minimiser
<i>max_iters</i>	maximum number of iterations

5.7.2.2 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

<i>copy</i>	copy is the object which we want to make a copy of
-------------	--

5.7.3 Member Function Documentation

5.7.3.1 `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

<i>fun</i>	objective function
------------	--------------------

Returns

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

5.7.3.2 `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

<i>copy</i>	
-------------	--

Returns

`golden_section_method&`

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

- `include/unconstrained_methods/one_dim/om_golden_section.hpp`

5.8 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_t< fp_type > const &init_guess) const

Function method that minimises the objective function.

Additional Inherited Members

5.8.1 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

<i>fp_type</i>	<i>fp_type</i> is a floating-point template parameter
----------------	---

5.8.2 Constructor & Destructor Documentation

5.8.2.1 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

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

5.8.3 Member Function Documentation

5.8.3.1 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

<i>objective</i>	objective function
<i>init_guess</i>	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

5.9 om_test_helpers::minimizer_helper< T > Struct Template Reference

Collaboration diagram for om_test_helpers::minimizer_helper< T >:

5.10 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 ©)
Construct a new nelder mead method object.
- [nelder_mead_method](#) & [operator=](#) ([nelder_mead_method](#) const ©)
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_t< fp_type > const &init_guess) const
Function method that minimises the objective function.

5.10.1 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

<i>fp_type</i>	fp_type is a floating-point template parameter
----------------	--

5.10.2 Constructor & Destructor Documentation

5.10.2.1 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

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

5.10.2.2 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

<i>copy</i>	copy is the object which we want to make a copy of
-------------	--

5.10.3 Member Function Documentation

5.10.3.1 minimize()

```
template<typename fp_type >
std::tuple< om_zero_order::vector_t< fp_type >, fp_type, std::size_t > om_unconstrained_methods::om_zero_order::
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

<i>objective</i>	objective function
<i>init_guess</i>	initial guess

Returns

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

5.10.3.2 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

<i>copy</i>	
-------------	--

Returns

`nelder_mead_method&`

5.10.3.3 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

<i>value</i>	value of contraction rho
--------------	--------------------------

5.10.3.4 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

<i>converge_tol</i>	tolerance for convergance
---------------------	---------------------------

5.10.3.5 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

<i>value</i>	value of expansion rho
--------------	------------------------

5.10.3.6 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

<i>iters</i>	maximum number of iterations
--------------	------------------------------

5.10.3.7 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

<i>value</i>	value of reflection rho
--------------	-------------------------

5.10.3.8 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

<i>value</i>	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

5.11 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_t< fp_type > const &init_guess) const

Function method that minimises the objective function.

Additional Inherited Members

5.11.1 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

<i>fp_type</i>	<i>fp_type</i> is a floating-point template parameter
----------------	---

5.11.2 Constructor & Destructor Documentation

5.11.2.1 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

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

5.11.3 Member Function Documentation

5.11.3.1 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

<i>objective</i>	objective function
<i>init_guess</i>	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

5.12 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 constructor a new powell conjugate method object.
- [powell_conjugate_method](#) &operator= ([powell_conjugate_method](#) const ©)
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_t< fp_type > const &init_guess) const
Function method that minimises the objective function.

5.12.1 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

<i>fp_type</i>	fp_type is a floating-point template parameter
----------------	--

5.12.2 Constructor & Destructor Documentation

5.12.2.1 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

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

5.12.2.2 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

<i>copy</i>	copy is the object which we want to make a copy of
-------------	--

5.12.3 Member Function Documentation

5.12.3.1 minimize()

```
template<typename fp_type >
std::tuple< om_zero_order::vector_t< fp_type >, fp_type, std::size_t > om_unconstrained_methods::om_zero_order::
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

<i>objective</i>	objective function
<i>init_guess</i>	initial guess

Returns

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

5.12.3.2 `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

<i>copy</i>	
-------------	--

Returns

`powell_conjugate_method&`

5.12.3.3 `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

<i>converge_tol</i>	tolerance for convergance
---------------------	---------------------------

5.12.3.4 `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

<i>iters</i>	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

5.13 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 constructor of a new powell method object.
- [powell_method](#) & [operator=](#) ([powell_method](#) const ©)
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.

5.13.1 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

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

5.13.2 Constructor & Destructor Documentation

5.13.2.1 `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

<i>range</i>	range of the minimiser
<i>tolerance</i>	tolerance of the minimiser
<i>max_ites</i>	maximum number of iterations

5.13.2.2 `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

<i>range</i>	range of the minimiser
<i>step</i>	size of the step of the minimiser
<i>max_step</i>	maximum size of the step of the minimiser
<i>tolerance</i>	tolerance of the minimiser
<i>max_ites</i>	maximum number of iterations

5.13.2.3 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

<i>copy</i>	
-------------	--

5.13.3 Member Function Documentation

5.13.3.1 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

<i>fun</i>	objective function
------------	--------------------

Returns

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

5.13.3.2 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

5.14 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 ©)
Construct a new quasi newton base object.
- [quasi_newton_base](#) & operator= ([quasi_newton_base](#) const ©)
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_**

5.14.1 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

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

5.14.2 Constructor & Destructor Documentation

5.14.2.1 `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

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

5.14.2.2 `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

<i>copy</i>	copy is the object which we want to make a copy of
-------------	--

5.14.3 Member Function Documentation

5.14.3.1 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

<i>copy</i>	
-------------	--

Returns

[quasi_newton_base&](#)

5.14.3.2 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

<i>arg_tol</i>	tolerance for stopping criteria
----------------	---------------------------------

5.14.3.3 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

<i>fun_tol</i>	tolerance for a value of objective function
----------------	---

5.14.3.4 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

<i>grad_tol</i>	tolerance for gardient
-----------------	------------------------

5.14.3.5 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

<i>iters</i>	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

5.15 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 constructor of a steepest descent method object.
- [steepest_descent_method](#) & operator= (steepest_descent_method const ©)
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_t< fp_type > const &init_guess) const
Function method that minimises the objective function.

5.15.1 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

<i>fp_type</i>	<i>fp_type</i> is a floating-point template parameter
----------------	---

5.15.2 Constructor & Destructor Documentation

5.15.2.1 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

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

5.15.2.2 `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

<i>copy</i>	copy is the object which we want to make a copy of
-------------	--

5.15.3 Member Function Documentation

5.15.3.1 `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

<i>objective</i>	objective function
<i>init_guess</i>	initial guess

Returns

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

5.15.3.2 `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

<i>copy</i>	
-------------	--

Returns

`steepest_descent_method&`

5.15.3.3 `set_arg_tolerance()`

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

Set the stopping criteria tolerance object.

Parameters

<i>arg_tol</i>	tolerance for stopping criteria
----------------	---------------------------------

5.15.3.4 `set_fun_tolerance()`

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

Set the fun tolerance object.

Parameters

<i>fun_tol</i>	tolerance for a value of function
----------------	-----------------------------------

5.15.3.5 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

<i>grad_tol</i>	tolerance for gradient
-----------------	------------------------

5.15.3.6 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

<i>iters</i>	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

brent_method
om_unconstrained_methods::om_line_methods::brent_method<
fp_type, typename >, 12

broyden_fletcher_goldfarb_shanno_method
om_unconstrained_methods::om_quasi_newton::broyden_fletcher_goldfarb_shanno_method<
fp_type >, 14

conjugate_gradient_base
om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base<
fp_type, typename >, 16, 17

davidon_fletcher_powell_method
om_unconstrained_methods::om_quasi_newton::davidon_fletcher_powell_method<
fp_type >, 20

fibonacci_method
om_unconstrained_methods::om_line_methods::fibonacci_method<
fp_type, typename >, 22

fletcher_reeves_method
om_unconstrained_methods::om_conjugate_gradient::fletcher_reeves_method<
fp_type >, 24

golden_section_method
om_unconstrained_methods::om_line_methods::golden_section_method<
fp_type, typename >, 26

hestenes_stiefel_method
om_unconstrained_methods::om_conjugate_gradient::hestenes_stiefel_method<
fp_type >, 28

minimize
om_unconstrained_methods::om_conjugate_gradient::fletcher_reeves_method<
fp_type >, 25
om_unconstrained_methods::om_conjugate_gradient::hestenes_stiefel_method<
fp_type >, 29
om_unconstrained_methods::om_conjugate_gradient::polak_ribiere_method<
fp_type >, 36
om_unconstrained_methods::om_quasi_newton::broyden_fletcher_goldfarb_shanno_method<
fp_type >, 15
om_unconstrained_methods::om_quasi_newton::davidon_fletcher_powell_method<
fp_type >, 20
om_unconstrained_methods::om_steepest_descent::steepest_descent_method<
fp_type >, 48
om_unconstrained_methods::om_zero_order::nelder_mead_method<
fp_type >, 31
om_unconstrained_methods::om_zero_order::powell_conjugate_method<
fp_type >, 38

nelder_mead_method
om_unconstrained_methods::om_zero_order::nelder_mead_method<
fp_type >, 30, 31

om_test_functions, 7

powell_function, 8

quadratic_function, 8

om_test_helpers::minimizer_helper< T >, 29

om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base<
fp_type, typename >, 15

conjugate_gradient_base, 16, 17

operator=, 17

set_arg_tolerance, 17

set_fun_tolerance, 18

set_grad_tolerance, 18

set_max_iterations, 18

om_unconstrained_methods::om_conjugate_gradient::fletcher_reeves_method<
fp_type >, 23

fletcher_reeves_method, 24

minimize, 25

om_unconstrained_methods::om_conjugate_gradient::hestenes_stiefel_method<
fp_type >, 28

hestenes_stiefel_method, 28

minimize, 29

om_unconstrained_methods::om_conjugate_gradient::polak_ribiere_method<
fp_type >, 34

minimize, 36

polak_ribiere_method, 36

hestenes_stiefel_method<
om_unconstrained_methods::om_line_methods::brent_method<
fp_type, typename >, 11

brent_method, 12

operator(), 12

operator=, 13

om_unconstrained_methods::om_line_methods::fibonacci_method<
fp_type, typename >, 21

fibonacci_method, 22

operator(), 22

operator=, 23

om_unconstrained_methods::om_line_methods::golden_section_method<
fp_type, typename >, 25

golden_section_method, 26

operator(), 27

operator=, 27

om_unconstrained_methods::om_line_methods::powell_method<
fp_type, typename >, 40

operator(), 42

operator=, 42

powell_method, 41, 42

om_unconstrained_methods::om_quasi_newton::broyden_fletcher_goldfarb_shanno_method, 14

set_grad_tolerance
 om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base<
 fp_type, typename >, [18](#)
 om_unconstrained_methods::om_quasi_newton::quasi_newton_base<
 fp_type, typename >, [46](#)
 om_unconstrained_methods::om_steepest_descent::steepest_descent_method<
 fp_type >, [49](#)

set_max_iterations
 om_unconstrained_methods::om_conjugate_gradient::conjugate_gradient_base<
 fp_type, typename >, [18](#)
 om_unconstrained_methods::om_quasi_newton::quasi_newton_base<
 fp_type, typename >, [46](#)
 om_unconstrained_methods::om_steepest_descent::steepest_descent_method<
 fp_type >, [50](#)
 om_unconstrained_methods::om_zero_order::nelder_mead_method<
 fp_type >, [33](#)
 om_unconstrained_methods::om_zero_order::powell_conjugate_method<
 fp_type >, [39](#)

set_reflection_rho
 om_unconstrained_methods::om_zero_order::nelder_mead_method<
 fp_type >, [33](#)

set_shrinkage_rho
 om_unconstrained_methods::om_zero_order::nelder_mead_method<
 fp_type >, [34](#)

steepest_descent_method
 om_unconstrained_methods::om_steepest_descent::steepest_descent_method<
 fp_type >, [47](#), [48](#)