

spheroidal\_lib

0.1.0

Generated by Doxygen 1.9.1



<b>1 spheroidal_cpp</b>	<b>1</b>
1.1 Description	1
1.2 Getting Started	1
1.2.1 Dependencies	1
1.3 Authors	2
1.4 Version History	2
1.5 Acknowledgments	2
<b>2 Namespace Index</b>	<b>3</b>
2.1 Namespace List	3
<b>3 Hierarchical Index</b>	<b>5</b>
3.1 Class Hierarchy	5
<b>4 Class Index</b>	<b>7</b>
4.1 Class List	7
<b>5 File Index</b>	<b>9</b>
5.1 File List	9
<b>6 Namespace Documentation</b>	<b>11</b>
6.1 gsl Namespace Reference	11
6.1.1 Function Documentation	13
6.1.1.1 arange()	13
6.1.1.2 arrayfun() [1/8]	13
6.1.1.3 arrayfun() [2/8]	13
6.1.1.4 arrayfun() [3/8]	13
6.1.1.5 arrayfun() [4/8]	14
6.1.1.6 arrayfun() [5/8]	14
6.1.1.7 arrayfun() [6/8]	15
6.1.1.8 arrayfun() [7/8]	15
6.1.1.9 arrayfun() [8/8]	15
6.1.1.10 eye()	15
6.1.1.11 fft() [1/4]	15
6.1.1.12 fft() [2/4]	16
6.1.1.13 fft() [3/4]	16
6.1.1.14 fft() [4/4]	16
6.1.1.15 ifft() [1/4]	16
6.1.1.16 ifft() [2/4]	16
6.1.1.17 ifft() [3/4]	17
6.1.1.18 ifft() [4/4]	17
6.1.1.19 leggauss() [1/2]	17
6.1.1.20 leggauss() [2/2]	17
6.1.1.21 linspace() [1/2]	17

6.1.1.22 linspace() [2/2]	18
6.1.1.23 meshgrid() [1/2]	18
6.1.1.24 meshgrid() [2/2]	18
6.2 gsl::complex_literals Namespace Reference	18
6.2.1 Function Documentation	18
6.2.1.1 operator""_i()	18
<b>7 Class Documentation</b>	<b>19</b>
7.1 gsl::ccolumn_view Class Reference	19
7.1.1 Detailed Description	19
7.1.2 Constructor & Destructor Documentation	19
7.1.2.1 ccolumn_view() [1/2]	20
7.1.2.2 ccolumn_view() [2/2]	20
7.2 gsl::cmatrix Class Reference	20
7.2.1 Constructor & Destructor Documentation	21
7.2.1.1 cmatrix() [1/8]	21
7.2.1.2 cmatrix() [2/8]	22
7.2.1.3 cmatrix() [3/8]	22
7.2.1.4 cmatrix() [4/8]	22
7.2.1.5 cmatrix() [5/8]	22
7.2.1.6 cmatrix() [6/8]	22
7.2.1.7 cmatrix() [7/8]	23
7.2.1.8 cmatrix() [8/8]	23
7.2.1.9 ~cmatrix()	23
7.2.2 Member Function Documentation	23
7.2.2.1 calloc()	23
7.2.2.2 clear()	23
7.2.2.3 column()	23
7.2.2.4 free()	24
7.2.2.5 get()	24
7.2.2.6 get_gsl_ptr()	24
7.2.2.7 ncols()	24
7.2.2.8 nrows()	24
7.2.2.9 operator()() [1/2]	24
7.2.2.10 operator()() [2/2]	25
7.2.2.11 operator=() [1/2]	25
7.2.2.12 operator=() [2/2]	25
7.2.2.13 print()	25
7.2.2.14 reshape()	25
7.2.2.15 resize()	25
7.2.2.16 row()	26
7.2.2.17 set()	26

7.2.2.18 size()	26
7.2.2.19 submatrix()	26
7.2.3 Friends And Related Function Documentation	26
7.2.3.1 ccolumn_view	26
7.2.3.2 cmatrix_view	26
7.2.3.3 crow_view	27
7.2.3.4 matrix	27
7.2.3.5 operator*	27
7.2.4 Member Data Documentation	27
7.2.4.1 gmat	27
7.3 gsl::cmatrix_view Class Reference	27
7.3.1 Detailed Description	28
7.3.2 Constructor & Destructor Documentation	28
7.3.2.1 cmatrix_view() [1/2]	28
7.3.2.2 cmatrix_view() [2/2]	28
7.3.3 Member Function Documentation	28
7.3.3.1 get_gsl_ptr()	28
7.3.3.2 operator cmatrix()	29
7.3.3.3 operator=() [1/2]	29
7.3.3.4 operator=() [2/2]	29
7.3.3.5 print()	29
7.3.4 Member Data Documentation	29
7.3.4.1 gmat_view	29
7.4 gsl::column_view Class Reference	30
7.4.1 Detailed Description	30
7.4.2 Constructor & Destructor Documentation	30
7.4.2.1 column_view() [1/2]	30
7.4.2.2 column_view() [2/2]	30
7.5 gsl::complex Class Reference	31
7.5.1 Detailed Description	31
7.5.2 Constructor & Destructor Documentation	32
7.5.2.1 complex() [1/4]	32
7.5.2.2 complex() [2/4]	32
7.5.2.3 complex() [3/4]	32
7.5.2.4 complex() [4/4]	32
7.5.3 Member Function Documentation	32
7.5.3.1 abs()	32
7.5.3.2 abs2()	32
7.5.3.3 arg()	33
7.5.3.4 imag()	33
7.5.3.5 operator*=( ) [1/2]	33
7.5.3.6 operator*=( ) [2/2]	33

7.5.3.7 operator+=( ) [ 1 / 2 ] . . . . .	33
7.5.3.8 operator+=( ) [ 2 / 2 ] . . . . .	33
7.5.3.9 operator-( ) . . . . .	33
7.5.3.10 operator-=( ) [ 1 / 2 ] . . . . .	34
7.5.3.11 operator-=( ) [ 2 / 2 ] . . . . .	34
7.5.3.12 operator/=( ) [ 1 / 2 ] . . . . .	34
7.5.3.13 operator/=( ) [ 2 / 2 ] . . . . .	34
7.5.3.14 operator=( ) . . . . .	34
7.5.3.15 print( ) . . . . .	34
7.5.3.16 real( ) . . . . .	34
7.5.3.17 set( ) [ 1 / 2 ] . . . . .	35
7.5.3.18 set( ) [ 2 / 2 ] . . . . .	35
7.5.4 Friends And Related Function Documentation . . . . .	35
7.5.4.1 complex_ref . . . . .	35
7.5.4.2 operator* . . . . .	35
7.5.4.3 operator+ . . . . .	35
7.5.4.4 operator- . . . . .	35
7.5.4.5 operator/ . . . . .	36
7.5.4.6 operator== . . . . .	36
7.6 gsl::complex_ref Class Reference . . . . .	36
7.6.1 Detailed Description . . . . .	37
7.6.2 Constructor & Destructor Documentation . . . . .	37
7.6.2.1 complex_ref( ) [ 1 / 4 ] . . . . .	37
7.6.2.2 complex_ref( ) [ 2 / 4 ] . . . . .	37
7.6.2.3 complex_ref( ) [ 3 / 4 ] . . . . .	37
7.6.2.4 complex_ref( ) [ 4 / 4 ] . . . . .	37
7.6.3 Member Function Documentation . . . . .	37
7.6.3.1 imag( ) . . . . .	38
7.6.3.2 operator complex( ) . . . . .	38
7.6.3.3 operator=( ) [ 1 / 2 ] . . . . .	38
7.6.3.4 operator=( ) [ 2 / 2 ] . . . . .	38
7.6.3.5 real( ) . . . . .	38
7.6.4 Member Data Documentation . . . . .	38
7.6.4.1 dat . . . . .	39
7.7 gsl::crow_view Class Reference . . . . .	39
7.7.1 Detailed Description . . . . .	39
7.7.2 Constructor & Destructor Documentation . . . . .	39
7.7.2.1 crow_view( ) [ 1 / 2 ] . . . . .	39
7.7.2.2 crow_view( ) [ 2 / 2 ] . . . . .	40
7.7.3 Member Function Documentation . . . . .	40
7.7.3.1 reshape( ) . . . . .	40
7.8 gsl::cvector Class Reference . . . . .	40

7.8.1 Detailed Description	42
7.8.2 Constructor & Destructor Documentation	42
7.8.2.1 cvector() [1/6]	42
7.8.2.2 cvector() [2/6]	42
7.8.2.3 cvector() [3/6]	42
7.8.2.4 cvector() [4/6]	42
7.8.2.5 cvector() [5/6]	42
7.8.2.6 cvector() [6/6]	43
7.8.2.7 ~cvector()	43
7.8.3 Member Function Documentation	43
7.8.3.1 calloc()	43
7.8.3.2 clear()	43
7.8.3.3 free()	43
7.8.3.4 get()	43
7.8.3.5 get_gsl_ptr()	44
7.8.3.6 norm()	44
7.8.3.7 operator>() [1/2]	44
7.8.3.8 operator>() [2/2]	44
7.8.3.9 operator+=()	44
7.8.3.10 operator-=()	44
7.8.3.11 operator=() [1/2]	45
7.8.3.12 operator=() [2/2]	45
7.8.3.13 print()	45
7.8.3.14 resize()	45
7.8.3.15 set()	45
7.8.3.16 size()	45
7.8.3.17 subvector()	46
7.8.3.18 view()	46
7.8.4 Friends And Related Function Documentation	46
7.8.4.1 ccolumn_view	46
7.8.4.2 cmatrix	46
7.8.4.3 crow_view	46
7.8.4.4 cvector_view	46
7.8.4.5 operator* [1/4]	47
7.8.4.6 operator* [2/4]	47
7.8.4.7 operator* [3/4]	47
7.8.4.8 operator* [4/4]	47
7.8.4.9 operator+ [1/4]	47
7.8.4.10 operator+ [2/4]	47
7.8.4.11 operator+ [3/4]	48
7.8.4.12 operator+ [4/4]	48
7.8.4.13 operator- [1/4]	48

7.8.4.14 operator- [2/4]	48
7.8.4.15 operator- [3/4]	48
7.8.4.16 operator- [4/4]	48
7.8.4.17 operator==	49
7.8.4.18 vector	49
7.8.5 Member Data Documentation	49
7.8.5.1 gvec	49
7.9 gsl::cvector_view Class Reference	49
7.9.1 Detailed Description	50
7.9.2 Constructor & Destructor Documentation	50
7.9.2.1 cvector_view() [1/2]	50
7.9.2.2 cvector_view() [2/2]	50
7.9.3 Member Function Documentation	50
7.9.3.1 get_gsl_ptr()	50
7.9.3.2 operator cvector()	51
7.9.3.3 operator=() [1/2]	51
7.9.3.4 operator=() [2/2]	51
7.9.3.5 print()	51
7.9.4 Member Data Documentation	51
7.9.4.1 gvec_view	51
7.10 gsl::matrix Class Reference	52
7.10.1 Detailed Description	53
7.10.2 Constructor & Destructor Documentation	53
7.10.2.1 matrix() [1/7]	53
7.10.2.2 matrix() [2/7]	53
7.10.2.3 matrix() [3/7]	54
7.10.2.4 matrix() [4/7]	54
7.10.2.5 matrix() [5/7]	54
7.10.2.6 matrix() [6/7]	54
7.10.2.7 matrix() [7/7]	54
7.10.2.8 ~matrix()	54
7.10.3 Member Function Documentation	55
7.10.3.1 calloc()	55
7.10.3.2 clear()	55
7.10.3.3 column()	55
7.10.3.4 free()	55
7.10.3.5 get()	55
7.10.3.6 get_gsl_ptr()	56
7.10.3.7 load_csv()	56
7.10.3.8 ncols()	56
7.10.3.9 nrows()	56
7.10.3.10 operator>() [1/2]	56



7.10.3.11 operator>() [2/2]	56
7.10.3.12 operator=() [1/2]	57
7.10.3.13 operator=() [2/2]	57
7.10.3.14 print()	57
7.10.3.15 print_csv()	57
7.10.3.16 reshape()	57
7.10.3.17 resize()	57
7.10.3.18 row()	58
7.10.3.19 set()	58
7.10.3.20 size()	58
7.10.3.21 submatrix()	58
7.10.4 Friends And Related Function Documentation	58
7.10.4.1 cmatrix	58
7.10.4.2 column_view	58
7.10.4.3 matrix_view	59
7.10.4.4 operator*	59
7.10.4.5 row_view	59
7.10.5 Member Data Documentation	59
7.10.5.1 gmat	59
7.11 gsl::matrix_view Class Reference	59
7.11.1 Detailed Description	60
7.11.2 Constructor & Destructor Documentation	60
7.11.2.1 matrix_view() [1/2]	60
7.11.2.2 matrix_view() [2/2]	60
7.11.3 Member Function Documentation	60
7.11.3.1 get_gsl_ptr()	60
7.11.3.2 operator matrix()	61
7.11.3.3 operator=() [1/2]	61
7.11.3.4 operator=() [2/2]	61
7.11.3.5 print()	61
7.11.4 Member Data Documentation	61
7.11.4.1 gmat_view	61
7.12 gsl::row_view Class Reference	62
7.12.1 Detailed Description	62
7.12.2 Constructor & Destructor Documentation	62
7.12.2.1 row_view() [1/2]	62
7.12.2.2 row_view() [2/2]	62
7.12.3 Member Function Documentation	63
7.12.3.1 reshape()	63
7.13 gsl::vector Class Reference	63
7.13.1 Detailed Description	64
7.13.2 Constructor & Destructor Documentation	64

7.13.2.1 vector() [1/5]	64
7.13.2.2 vector() [2/5]	65
7.13.2.3 vector() [3/5]	65
7.13.2.4 vector() [4/5]	65
7.13.2.5 vector() [5/5]	65
7.13.2.6 ~vector()	65
7.13.3 Member Function Documentation	65
7.13.3.1 calloc()	65
7.13.3.2 clear()	66
7.13.3.3 free()	66
7.13.3.4 get()	66
7.13.3.5 get_gsl_ptr()	66
7.13.3.6 norm()	66
7.13.3.7 operator>() [1/2]	66
7.13.3.8 operator>() [2/2]	67
7.13.3.9 operator+=()	67
7.13.3.10 operator-=()	67
7.13.3.11 operator=() [1/2]	67
7.13.3.12 operator=() [2/2]	67
7.13.3.13 print()	67
7.13.3.14 resize()	68
7.13.3.15 set()	68
7.13.3.16 size()	68
7.13.3.17 subvector()	68
7.13.3.18 view()	68
7.13.4 Friends And Related Function Documentation	68
7.13.4.1 column_view	68
7.13.4.2 cvector	69
7.13.4.3 matrix	69
7.13.4.4 operator* [1/6]	69
7.13.4.5 operator* [2/6]	69
7.13.4.6 operator* [3/6]	69
7.13.4.7 operator* [4/6]	69
7.13.4.8 operator* [5/6]	70
7.13.4.9 operator* [6/6]	70
7.13.4.10 operator+ [1/4]	70
7.13.4.11 operator+ [2/4]	70
7.13.4.12 operator+ [3/4]	70
7.13.4.13 operator+ [4/4]	70
7.13.4.14 operator- [1/4]	71
7.13.4.15 operator- [2/4]	71
7.13.4.16 operator- [3/4]	71

7.13.4.17 operator- [4/4]	71
7.13.4.18 operator==	71
7.13.4.19 row_view	71
7.13.4.20 vector_view	71
7.13.5 Member Data Documentation	72
7.13.5.1 gvec	72
7.14 gsl::vector_view Class Reference	72
7.14.1 Detailed Description	73
7.14.2 Constructor & Destructor Documentation	73
7.14.2.1 vector_view() [1/2]	73
7.14.2.2 vector_view() [2/2]	73
7.14.3 Member Function Documentation	73
7.14.3.1 get_gsl_ptr()	73
7.14.3.2 operator vector()	73
7.14.3.3 operator=() [1/2]	74
7.14.3.4 operator=() [2/2]	74
7.14.3.5 print()	74
7.14.4 Member Data Documentation	74
7.14.4.1 gvec_view	74
<b>8 File Documentation</b>	<b>75</b>
8.1 grid_functions.h File Reference	75
8.1.1 Function Documentation	75
8.1.1.1 gl_grid()	75
8.1.1.2 spharm_grid_size_ord()	76
8.1.1.3 spharm_grid_size_tot()	76
8.2 legendre_otc.h File Reference	76
8.2.1 Function Documentation	76
8.2.1.1 cont_frac()	76
8.2.1.2 Dlegendre_otc() [1/2]	77
8.2.1.3 Dlegendre_otc() [2/2]	77
8.2.1.4 geti()	77
8.2.1.5 legendre_otc() [1/2]	77
8.2.1.6 legendre_otc() [2/2]	77
8.3 spheroidal_analysis.h File Reference	78
8.3.1 Function Documentation	78
8.3.1.1 spheroidal_analysis()	78
8.4 /home/jspainhour/spheroidal_cpp/include/yawg/cmatrix.h File Reference	78
8.5 /home/jspainhour/spheroidal_cpp/include/yawg/complex.h File Reference	78
8.6 /home/jspainhour/spheroidal_cpp/include/yawg/core.h File Reference	79
8.7 /home/jspainhour/spheroidal_cpp/include/yawg/cvector.h File Reference	79
8.8 /home/jspainhour/spheroidal_cpp/include/yawg/fft.h File Reference	80

8.9 /home/jspainhour/spheroidal_cpp/include/yawg/matrix.h File Reference	80
8.10 /home/jspainhour/spheroidal_cpp/include/yawg/utils.hpp File Reference	81
8.11 /home/jspainhour/spheroidal_cpp/include/yawg/vector.h File Reference	82
8.12 /home/jspainhour/spheroidal_cpp/README.md File Reference	82
8.13 /home/jspainhour/spheroidal_cpp/tests/yawg/test_fft.cpp File Reference	82
8.13.1 Macro Definition Documentation	83
8.13.1.1 CATCH_CONFIG_MAIN	83
8.13.2 Function Documentation	83
8.13.2.1 TEST_CASE() [1/4]	83
8.13.2.2 TEST_CASE() [2/4]	83
8.13.2.3 TEST_CASE() [3/4]	83
8.13.2.4 TEST_CASE() [4/4]	84
8.14 /home/jspainhour/spheroidal_cpp/tests/yawg/test_utils.cpp File Reference	84
8.14.1 Macro Definition Documentation	84
8.14.1.1 CATCH_CONFIG_MAIN	84
8.14.2 Function Documentation	84
8.14.2.1 TEST_CASE() [1/8]	85
8.14.2.2 TEST_CASE() [2/8]	85
8.14.2.3 TEST_CASE() [3/8]	85
8.14.2.4 TEST_CASE() [4/8]	85
8.14.2.5 TEST_CASE() [5/8]	85
8.14.2.6 TEST_CASE() [6/8]	85
8.14.2.7 TEST_CASE() [7/8]	86
8.14.2.8 TEST_CASE() [8/8]	86
8.15 /home/jspainhour/spheroidal_cpp/tests/yawg/test_wrapper.cpp File Reference	86
8.15.1 Macro Definition Documentation	87
8.15.1.1 CATCH_CONFIG_MAIN	87
8.15.2 Function Documentation	87
8.15.2.1 TEST_CASE() [1/20]	87
8.15.2.2 TEST_CASE() [2/20]	87
8.15.2.3 TEST_CASE() [3/20]	87
8.15.2.4 TEST_CASE() [4/20]	87
8.15.2.5 TEST_CASE() [5/20]	88
8.15.2.6 TEST_CASE() [6/20]	88
8.15.2.7 TEST_CASE() [7/20]	88
8.15.2.8 TEST_CASE() [8/20]	88
8.15.2.9 TEST_CASE() [9/20]	88
8.15.2.10 TEST_CASE() [10/20]	88
8.15.2.11 TEST_CASE() [11/20]	89
8.15.2.12 TEST_CASE() [12/20]	89
8.15.2.13 TEST_CASE() [13/20]	89
8.15.2.14 TEST_CASE() [14/20]	89

8.15.2.15 TEST_CASE() [15/20]	89
8.15.2.16 TEST_CASE() [16/20]	89
8.15.2.17 TEST_CASE() [17/20]	90
8.15.2.18 TEST_CASE() [18/20]	90
8.15.2.19 TEST_CASE() [19/20]	90
8.15.2.20 TEST_CASE() [20/20]	90
8.16 /home/jspainhour/spheroidal_cpp/tests/yawg/test_wrapper_math.cpp File Reference	90
8.16.1 Macro Definition Documentation	91
8.16.1.1 CATCH_CONFIG_MAIN	91
8.16.2 Function Documentation	91
8.16.2.1 TEST_CASE() [1/6]	91
8.16.2.2 TEST_CASE() [2/6]	91
8.16.2.3 TEST_CASE() [3/6]	91
8.16.2.4 TEST_CASE() [4/6]	92
8.16.2.5 TEST_CASE() [5/6]	92
8.16.2.6 TEST_CASE() [6/6]	92
8.17 /home/jspainhour/spheroidal_cpp/tests/yawg/test_wrapper_view.cpp File Reference	92
8.17.1 Macro Definition Documentation	93
8.17.1.1 CATCH_CONFIG_MAIN	93
8.17.2 Function Documentation	93
8.17.2.1 TEST_CASE() [1/10]	93
8.17.2.2 TEST_CASE() [2/10]	93
8.17.2.3 TEST_CASE() [3/10]	93
8.17.2.4 TEST_CASE() [4/10]	93
8.17.2.5 TEST_CASE() [5/10]	94
8.17.2.6 TEST_CASE() [6/10]	94
8.17.2.7 TEST_CASE() [7/10]	94
8.17.2.8 TEST_CASE() [8/10]	94
8.17.2.9 TEST_CASE() [9/10]	94
8.17.2.10 TEST_CASE() [10/10]	94



# Chapter 1

## spheroidal\_cpp

### 1.1 Description

`spheroidal` is a library for constructing and manipulating spheroidal harmonics, evaluating boundary integral operators, and other operations necessary for solving boundary integral equations on spheroids. Additionally, this repository contains example files to demonstrate usage and test files to demonstrate correctness.

Additionally, this repository contains Yet Another Wrapper for GSL, or `yawg`. This library contains a lightweight interface for the GNU Scientific Library, or GSL. Its intentions are to simplify usage of these functions, such as with a `gsl::vector` class that automatically handles memory allocation and pointer management.

The full documentation is built with `doxygen`, and can be constructed by performing the `make docs` command.

### 1.2 Getting Started

Both the `spheroidal` and `yawg` libraries are built using CMake. To build from the command line, run

```
mkdir build
cd build
cmake ..
make (<specific target>)
```

An extensive library of testing functions and assertions is contained in the `tests` subfolder. These tests can also be built and executed using CMake.

#### 1.2.1 Dependencies

Requires the following prerequisites, along with the version used during testing:

- C++11
- CMake v3.22.1
- GNU Scientific Library v2.7.1
- Doxygen v1.9.1 (optional)
- Catch2 v3.0.1

## 1.3 Authors

[Jacob Spainhour](@jcs15c)

[Leo Crowder](@lcrowder)

## 1.4 Version History

- 0.1
  - Initial Release

## 1.5 Acknowledgments

We would like to acknowledge the existence of other GSL C++ wrappers, and hope that ours is comparable in utility.

The following are existing C++ gsl wrapper classes we have found.

- GSLwrap: <https://gslwrap.sourceforge.net/>
- ccgsl: <https://ccgsl.sourceforge.net/>
- GSL-lib: <https://github.com/johanjoensson/GSL-lib>
- ROOT: <https://root.cern/root/html606/index.html>



## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">gsl</a> . . . . .	11
<a href="#">gsl::complex_literals</a> . . . . .	18



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

gsl::cmatrix . . . . .	20
gsl::cmatrix_view . . . . .	27
gsl::complex_ref . . . . .	36
gsl::cvector . . . . .	40
gsl::cvector_view . . . . .	49
gsl::ccolumn_view . . . . .	19
gsl::crow_view . . . . .	39
gsl_complex	
gsl::complex . . . . .	31
gsl::matrix . . . . .	52
gsl::matrix_view . . . . .	59
gsl::vector . . . . .	63
gsl::vector_view . . . . .	72
gsl::column_view . . . . .	30
gsl::row_view . . . . .	62



## Chapter 4

# Class Index

### 4.1 Class List

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

<a href="#">gsl::ccolumn_view</a>	
A subclass of <a href="#">cvector_view</a> for non-stride-1 cvector	19
<a href="#">gsl::cmatrix</a>	20
<a href="#">gsl::cmatrix_view</a>	
A wrapper class for <a href="#">gsl_matrix_complex_view</a>	27
<a href="#">gsl::column_view</a>	
A subclass of <a href="#">vector_view</a> for non-stride-1 vectors	30
<a href="#">gsl::complex</a>	
Wrapper class for <a href="#">gsl_complex</a> structs	31
<a href="#">gsl::complex_ref</a>	
Stores a reference to a <a href="#">gsl::complex</a> object	36
<a href="#">gsl::crow_view</a>	
A subclass of <a href="#">cvector_view</a> for stride-1 vectors	39
<a href="#">gsl::cvector</a>	
A wrapper class for <a href="#">gsl_vector_complex</a>	40
<a href="#">gsl::cvector_view</a>	
A wrapper class for <a href="#">gsl_vector_complex_view</a>	49
<a href="#">gsl::matrix</a>	
A wrapper class for <a href="#">gsl_matrix</a>	52
<a href="#">gsl::matrix_view</a>	
A wrapper class for <a href="#">gsl_matrix_view</a>	59
<a href="#">gsl::row_view</a>	
A subclass of <a href="#">vector_view</a> for stride-1 vectors	62
<a href="#">gsl::vector</a>	
A wrapper class for <a href="#">gsl_vector</a>	63
<a href="#">gsl::vector_view</a>	
A wrapper class for <a href="#">gsl_vector_view</a>	72



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">grid_functions.h</a>	75
<a href="#">legendre_otc.h</a>	76
<a href="#">spheroidal_analysis.h</a>	78
/home/jspainhour/spheroidal_cpp/include/yawg/ <a href="#">cmatrix.h</a>	78
/home/jspainhour/spheroidal_cpp/include/yawg/ <a href="#">complex.h</a>	78
/home/jspainhour/spheroidal_cpp/include/yawg/ <a href="#">core.h</a>	79
/home/jspainhour/spheroidal_cpp/include/yawg/ <a href="#">cvector.h</a>	79
/home/jspainhour/spheroidal_cpp/include/yawg/ <a href="#">fft.h</a>	80
/home/jspainhour/spheroidal_cpp/include/yawg/ <a href="#">matrix.h</a>	80
/home/jspainhour/spheroidal_cpp/include/yawg/ <a href="#">utils.hpp</a>	81
/home/jspainhour/spheroidal_cpp/include/yawg/ <a href="#">vector.h</a>	82
/home/jspainhour/spheroidal_cpp/tests/yawg/ <a href="#">test_fft.cpp</a>	82
/home/jspainhour/spheroidal_cpp/tests/yawg/ <a href="#">test_utils.cpp</a>	84
/home/jspainhour/spheroidal_cpp/tests/yawg/ <a href="#">test_wrapper.cpp</a>	86
/home/jspainhour/spheroidal_cpp/tests/yawg/ <a href="#">test_wrapper_math.cpp</a>	90
/home/jspainhour/spheroidal_cpp/tests/yawg/ <a href="#">test_wrapper_view.cpp</a>	92





## Chapter 6

# Namespace Documentation

### 6.1 gsl Namespace Reference

#### Namespaces

- [complex\\_literals](#)

#### Classes

- class [cmatrix](#)
- class [cmatrix\\_view](#)  
*A wrapper class for `gsl_matrix_complex_view`.*
- class [complex](#)  
*Wrapper class for `gsl_complex` structs.*
- class [complex\\_ref](#)  
*Stores a reference to a `gsl::complex` object.*
- class [cvector](#)  
*A wrapper class for `gsl_vector_complex`.*
- class [cvector\\_view](#)  
*A wrapper class for `gsl_vector_complex_view`.*
- class [crow\\_view](#)  
*A subclass of `cvector_view` for stride-1 vectors.*
- class [ccolumn\\_view](#)  
*A subclass of `cvector_view` for non-stride-1 cvecs.*
- class [matrix](#)  
*A wrapper class for `gsl_matrix`.*
- class [matrix\\_view](#)  
*A wrapper class for `gsl_matrix_view`.*
- class [vector](#)  
*A wrapper class for `gsl_vector`.*
- class [vector\\_view](#)  
*A wrapper class for `gsl_vector_view`.*
- class [row\\_view](#)  
*A subclass of `vector_view` for stride-1 vectors.*
- class [column\\_view](#)  
*A subclass of `vector_view` for non-stride-1 vectors.*

## Functions

- [complex operator""\\_i](#) (long double y)  
*User defined literal overload for complex numbers.*
- [gsl::cvector fft](#) ([gsl::cvector](#) &&x)  
*Compute in-place fft of a [gsl::\(c\)vector](#).*
- [gsl::cvector fft](#) (const [gsl::cvector](#) &x)  
*Compute fft of a [gsl::\(c\)vector](#).*
- [gsl::cvector ifft](#) ([gsl::cvector](#) &&x)  
*Compute in-place inverse fft of a [gsl::\(c\)vector](#).*
- [gsl::cvector ifft](#) (const [gsl::cvector](#) &x)  
*Compute inverse fft of a [gsl::\(c\)vector](#).*
- [gsl::cmatrix fft](#) ([gsl::cmatrix](#) &&x, int dim=1)  
*Compute in-place 1D fft of each column/row of a [gsl::\(c\)matrix](#).*
- [gsl::cmatrix fft](#) (const [gsl::cmatrix](#) &x, int dim=1)  
*Compute 1D fft of each column/row of a [gsl::\(c\)matrix](#).*
- [gsl::cmatrix ifft](#) ([gsl::cmatrix](#) &&x, int dim=1)  
*Compute in-place 1D inverse fft of each column/row of a [gsl::\(c\)matrix](#).*
- [gsl::cmatrix ifft](#) (const [gsl::cmatrix](#) &x, int dim=1)  
*Compute 1D inverse fft of each column/row of a [gsl::\(c\)matrix](#).*
- void [leggauss](#) (size\_t n, [gsl::vector](#) &x, [gsl::vector](#) &w, double a=-1.0, double b=1.0)  
*Store Gauss-Legendre quadrature nodes and weights.*
- [vector leggauss](#) (size\_t n, double a=-1.0, double b=1.0)  
*Get a vector Gauss-Legendre quadrature nodes.*
- [gsl::vector linspace](#) (double a, double b, size\_t N=100)  
*Get a [gsl::vector](#) of evenly spaced points on the interval [a, b] (inclusive)*
- [gsl::cvector linspace](#) ([gsl::complex](#) a, [gsl::complex](#) b, size\_t n)  
*Complex version of linspace.*
- [gsl::vector arange](#) (double a, double b, double step=1.0)
- void [meshgrid](#) (const [gsl::vector](#) &x, const [gsl::vector](#) &y, [gsl::matrix](#) &X, [gsl::matrix](#) &Y)  
*Store 2D grid coordinates based on 1D input [gsl::vectors](#).*
- void [meshgrid](#) (const [gsl::cvector](#) &x, const [gsl::cvector](#) &y, [gsl::cmatrix](#) &X, [gsl::cmatrix](#) &Y)
- [gsl::matrix eye](#) (size\_t n)  
*Return the nxn identity matrix.*
- template<typename Lambda >  
[gsl::vector arrayfun](#) (Lambda &&func, const [gsl::vector](#) &x)  
*Apply lambda function to each element of a [gsl::vector](#), akin to MATLAB arrayfun.*
- template<typename Lambda >  
[gsl::vector arrayfun](#) (Lambda &&func, [gsl::vector](#) &&x)  
*Move version of arrayfun for vectors.*
- template<typename Lambda >  
[gsl::cvector arrayfun](#) (Lambda &&func, const [gsl::cvector](#) &x)  
*Copy version of arrayfun for complex vectors.*
- template<typename Lambda >  
[gsl::cvector arrayfun](#) (Lambda &&func, [gsl::cvector](#) &&x)  
*Move version of arrayfun for complex vectors.*
- template<typename Lambda >  
[gsl::matrix arrayfun](#) (Lambda &&func, const [gsl::matrix](#) &x)  
*Apply lambda function to each element of a [gsl::matrix](#), akin to MATLAB arrayfun.*
- template<typename Lambda >  
[gsl::matrix arrayfun](#) (Lambda &&func, [gsl::matrix](#) &&x)  
*Move version of arrayfun.*

- `template<typename Lambda >`  
`gsl::cmatrix arrayfun` (`Lambda &&func`, `const gsl::cmatrix &x`)  
*Copy version of arrayfun for complex matrices.*
- `template<typename Lambda >`  
`gsl::cmatrix arrayfun` (`Lambda &&func`, `gsl::cmatrix &&x`)  
*Move version of arrayfun for complex matrices.*

## 6.1.1 Function Documentation

### 6.1.1.1 `arange()`

```
gsl::vector gsl::arange (
    double a,
    double b,
    double step = 1.0 )
```

### 6.1.1.2 `arrayfun()` [1/8]

```
template<typename Lambda >
gsl::cmatrix gsl::arrayfun (
    Lambda && func,
    const gsl::cmatrix & x )
```

Copy version of arrayfun for complex matrices.

### 6.1.1.3 `arrayfun()` [2/8]

```
template<typename Lambda >
gsl::cvector gsl::arrayfun (
    Lambda && func,
    const gsl::cvector & x )
```

Copy version of arrayfun for complex vectors.

### 6.1.1.4 `arrayfun()` [3/8]

```
template<typename Lambda >
gsl::matrix gsl::arrayfun (
    Lambda && func,
    const gsl::matrix & x )
```

Apply lambda function to each element of a `gsl::matrix`, akin to MATLAB `arrayfun`.

**Parameters**

<i>func</i>	Lambda function to apply to each element
<i>x</i>	Input matrix

**Note**

This function is not optimized for speed, but rather for convenience, as there is overhead in using templated Lambda functions. If speed is necessary, add the function to `gsl_utils` specifically.

**Returns**

Matrix of same size as `x` with `func` applied to each element

**6.1.1.5 `arrayfun()` [4/8]**

```
template<typename Lambda >
gsl::vector gsl::arrayfun (
    Lambda && func,
    const gsl::vector & x )
```

Apply lambda function to each element of a `gsl::vector`, akin to MATLAB `arrayfun`.

**Parameters**

<i>func</i>	Lambda function to apply to each element
<i>x</i>	Input matrix

**Note**

This function is not optimized for speed, but rather for convenience, as there is overhead in using templated Lambda functions. If speed is necessary, add the function to `gsl_utils` specifically.

**Returns**

Matrix of same size as `x` with `func` applied to each element

**6.1.1.6 `arrayfun()` [5/8]**

```
template<typename Lambda >
gsl::cmatrix gsl::arrayfun (
    Lambda && func,
    gsl::cmatrix && x )
```

Move version of `arrayfun` for complex matrices.

#### 6.1.1.7 arrayfun() [6/8]

```
template<typename Lambda >
gsl::cvector gsl::arrayfun (
    Lambda && func,
    gsl::cvector && x )
```

Move version of arrayfun for complex vectors.

#### 6.1.1.8 arrayfun() [7/8]

```
template<typename Lambda >
gsl::matrix gsl::arrayfun (
    Lambda && func,
    gsl::matrix && x )
```

Move version of arrayfun.

#### 6.1.1.9 arrayfun() [8/8]

```
template<typename Lambda >
gsl::vector gsl::arrayfun (
    Lambda && func,
    gsl::vector && x )
```

Move version of arrayfun for vectors.

#### 6.1.1.10 eye()

```
gsl::matrix gsl::eye (
    size_t n )
```

Return the nxn identity matrix.

#### 6.1.1.11 fft() [1/4]

```
gsl::cmatrix gsl::fft (
    const gsl::cmatrix & x,
    int dim = 1 )
```

Compute 1D fft of each column/row of a gsl::(c)matrix.

**6.1.1.12 `fft()` [2/4]**

```
gsl::cvector gsl::fft (
    const gsl::cvector & x )
```

Compute fft of a `gsl::(c)vector`.

**6.1.1.13 `fft()` [3/4]**

```
gsl::cmatrix gsl::fft (
    gsl::cmatrix && x,
    int dim = 1 )
```

Compute in-place 1D fft of each column/row of a `gsl::(c)matrix`.

**6.1.1.14 `fft()` [4/4]**

```
gsl::cvector gsl::fft (
    gsl::cvector && x )
```

Compute in-place fft of a `gsl::(c)vector`.

**6.1.1.15 `ifft()` [1/4]**

```
gsl::cmatrix gsl::ifft (
    const gsl::cmatrix & x,
    int dim = 1 )
```

Compute 1D inverse fft of each column/row of a `gsl::(c)matrix`.

**6.1.1.16 `ifft()` [2/4]**

```
gsl::cvector gsl::ifft (
    const gsl::cvector & x )
```

Compute inverse fft of a `gsl::(c)vector`.

**6.1.1.17** `ifft()` [3/4]

```
gsl::cmatrix gsl::ifft (
    gsl::cmatrix && x,
    int dim = 1 )
```

Compute in-place 1D inverse fft of each column/row of a `gsl::(c)matrix`.

**6.1.1.18** `ifft()` [4/4]

```
gsl::cvector gsl::ifft (
    gsl::cvector && x )
```

Compute in-place inverse fft of a `gsl::(c)vector`.

**6.1.1.19** `leggauss()` [1/2]

```
vector gsl::leggauss (
    size_t n,
    double a = -1.0,
    double b = 1.0 )
```

Get a vector Gauss-Legendre quadrature nodes.

**6.1.1.20** `leggauss()` [2/2]

```
void gsl::leggauss (
    size_t n,
    gsl::vector & x,
    gsl::vector & w,
    double a = -1.0,
    double b = 1.0 )
```

Store Gauss-Legendre quadrature nodes and weights.

**6.1.1.21** `linspace()` [1/2]

```
gsl::vector gsl::linspace (
    double a,
    double b,
    size_t N = 100 )
```

Get a `gsl::vector` of evenly spaced points on the interval [a, b] (inclusive)

### 6.1.1.22 linspace() [2/2]

```
gsl::cvector gsl::linspace (
    gsl::complex a,
    gsl::complex b,
    size_t n )
```

Complex version of linspace.

### 6.1.1.23 meshgrid() [1/2]

```
void gsl::meshgrid (
    const gsl::cvector & x,
    const gsl::cvector & y,
    gsl::cmatrix & X,
    gsl::cmatrix & Y )
```

Complex version of [gsl::meshgrid](#)

### 6.1.1.24 meshgrid() [2/2]

```
void gsl::meshgrid (
    const gsl::vector & x,
    const gsl::vector & y,
    gsl::matrix & X,
    gsl::matrix & Y )
```

Store 2D grid coordinates based on 1D input gsl::vectors.

## 6.2 gsl::complex\_literals Namespace Reference

### Functions

- [complex operator""\\_i](#) (long double y)  
*User defined literal overload for complex numbers.*

### 6.2.1 Function Documentation

#### 6.2.1.1 operator""\_i()

```
complex gsl::complex_literals::operator""_i (
    long double y ) [inline]
```

User defined literal overload for complex numbers.



## Chapter 7

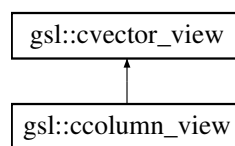
# Class Documentation

### 7.1 gsl::ccolumn\_view Class Reference

A subclass of [cvector\\_view](#) for non-stride-1 cectors.

```
#include <cvector.h>
```

Inheritance diagram for `gsl::ccolumn_view`:



#### Public Member Functions

- [ccolumn\\_view](#) (`gsl_vector_complex_view` [gvec\\_view](#))  
*Construct [ccolumn\\_view](#) from existing `cvector` view.*
- [ccolumn\\_view](#) (`const cvector &``v`)  
*Construct [ccolumn\\_view](#) from `cvector`.*

#### Additional Inherited Members

##### 7.1.1 Detailed Description

A subclass of [cvector\\_view](#) for non-stride-1 cectors.

##### 7.1.2 Constructor & Destructor Documentation

### 7.1.2.1 ccolumn\_view() [1/2]

```
gsl::ccolumn_view::ccolumn_view (
    gsl_vector_complex_view gvec_view ) [inline]
```

Construct [ccolumn\\_view](#) from existing cvector view.

### 7.1.2.2 ccolumn\_view() [2/2]

```
gsl::ccolumn_view::ccolumn_view (
    const cvector & v ) [inline]
```

Construct [ccolumn\\_view](#) from cvector.

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

- [/home/jspainhour/spheroidal\\_cpp/include/yawg/cvector.h](#)

## 7.2 gsl::cmatrix Class Reference

```
#include <cmatrix.h>
```

### Public Member Functions

- [cmatrix](#) ()  
*Construct empty matrix.*
- [cmatrix](#) (size\_t n, size\_t m)  
*Construct zero matrix of size n x m.*
- [cmatrix](#) (const gsl\_matrix\_complex \*gmat\_other)  
*Construct new [gsl::matrix](#) from [gsl\\_matrix](#).*
- [cmatrix](#) (const cvector &v)  
*Construct new n x 1 [gsl::cmatrix](#) from a [gsl::cvector](#).*
- [cmatrix](#) (const [cmatrix](#) &M, size\_t n, size\_t m)  
*Copy constructor creating n x m complex matrix.*
- [cmatrix](#) (const [cmatrix](#) &M)
- [cmatrix](#) ([cmatrix](#) &&M)
- [cmatrix](#) (const [matrix](#) &M)  
*Construct new [gsl::cmatrix](#) from [gsl::matrix](#).*
- [cmatrix](#) & operator= (const [cmatrix](#) &gvec\_other)
- [cmatrix](#) & operator= ([cmatrix](#) &&gvec\_other)
- [~cmatrix](#) ()
- [complex\\_ref operator\(\)](#) (size\_t i, size\_t j)  
*Return a reference to the element at position (i,j)*
- void [set](#) (size\_t i, size\_t j, [complex](#) z)
- [complex](#) [get](#) (size\_t i, size\_t j) const  
*Return a const reference to the element at position (i,j)*
- const [complex\\_ref operator\(\)](#) (size\_t i, size\_t j) const

- `size_t size () const`
- `size_t nrows () const`
- `size_t ncols () const`
- `gsl_matrix_complex * get_gsl_ptr () const`  
*Access the pointer to the underlying `gsl_matrix_complex`.*
- `void resize (size_t n, size_t m)`  
*Resize the `gsl::cmatrix`, setting elements to zero.*
- `void clear ()`  
*Clear the `gsl::cmatrix`, free underlying memory.*
- `cmatrix reshape (size_t n, size_t m) const`  
*Return a new  $n \times m$  `gsl::cmatrix` with same elements.*
- `void print (FILE *out=stdout) const`  
*Pretty-print the complex matrix to file stream.*
- `cmatrix_view submatrix (size_t i, size_t j, size_t n, size_t m)`  
*Return a view to a submatrix of the complex matrix.*
- `crow_view row (size_t i)`  
*Return a view to a row of the complex matrix.*
- `ccolumn_view column (size_t j)`  
*Return a view to a column of the complex matrix.*

## Protected Member Functions

- `void free ()`  
*Private function to free allocated memory.*
- `void calloc (size_t n, size_t m)`  
*Private function to (continuously) allocate memory.*

## Protected Attributes

- `gsl_matrix_complex * gmat`

## Friends

- class `matrix`
- class `cmatrix_view`
- class `crow_view`
- class `ccolumn_view`
- `cmatrix operator*` (const `cmatrix` &A, const `cmatrix` &B)

## 7.2.1 Constructor & Destructor Documentation

### 7.2.1.1 `cmatrix()` [1/8]

```
gsl::cmatrix::cmatrix ( )
```

Construct empty matrix.

### 7.2.1.2 `cmatrix()` [2/8]

```
gsl::cmatrix::cmatrix (
    size_t n,
    size_t m )
```

Construct zero matrix of size  $n \times m$ .

### 7.2.1.3 `cmatrix()` [3/8]

```
gsl::cmatrix::cmatrix (
    const gsl_matrix_complex * gmat_other )
```

Construct new [gsl::matrix](#) from `gsl_matrix`.

### 7.2.1.4 `cmatrix()` [4/8]

```
gsl::cmatrix::cmatrix (
    const cvector & v )
```

Construct new  $n \times 1$  [gsl::cmatrix](#) from a [gsl::cvector](#).

### 7.2.1.5 `cmatrix()` [5/8]

```
gsl::cmatrix::cmatrix (
    const cmatrix & M,
    size_t n,
    size_t m )
```

Copy constructor creating  $n \times m$  complex matrix.

### 7.2.1.6 `cmatrix()` [6/8]

```
gsl::cmatrix::cmatrix (
    const cmatrix & M )
```

### 7.2.1.7 cmatrix() [7/8]

```
gsl::cmatrix::cmatrix (
    cmatrix && M )
```

### 7.2.1.8 cmatrix() [8/8]

```
gsl::cmatrix::cmatrix (
    const matrix & M )
```

Construct new [gsl::cmatrix](#) from [gsl::matrix](#).

### 7.2.1.9 ~cmatrix()

```
gsl::cmatrix::~~cmatrix ( )
```

## 7.2.2 Member Function Documentation

### 7.2.2.1 calloc()

```
void gsl::cmatrix::calloc (
    size_t n,
    size_t m ) [protected]
```

Private function to (continuously) allocate memory.

### 7.2.2.2 clear()

```
void gsl::cmatrix::clear ( )
```

Clear the [gsl::cmatrix](#), free underlying memory.

### 7.2.2.3 column()

```
ccolumn_view gsl::cmatrix::column (
    size_t j )
```

Return a view to a column of the complex matrix.

#### 7.2.2.4 free()

```
void gsl::cmatrix::free ( ) [protected]
```

Private function to free allocated memory.

#### 7.2.2.5 get()

```
complex gsl::cmatrix::get (
    size_t i,
    size_t j ) const
```

Return a const reference to the element at position (i,j)

#### 7.2.2.6 get\_gsl\_ptr()

```
gsl_matrix_complex* gsl::cmatrix::get_gsl_ptr ( ) const [inline]
```

Access the pointer to the underlying gsl\_matrix\_complex.

#### 7.2.2.7 ncols()

```
size_t gsl::cmatrix::ncols ( ) const
```

#### 7.2.2.8 nrows()

```
size_t gsl::cmatrix::nrows ( ) const
```

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

```
complex_ref gsl::cmatrix::operator() (
    size_t i,
    size_t j )
```

Return a reference to the element at position (i,j)

### 7.2.2.10 operator>() [2/2]

```
const complex_ref gsl::cmatrix::operator() (
    size_t i,
    size_t j ) const
```

### 7.2.2.11 operator=() [1/2]

```
cmatrix& gsl::cmatrix::operator= (
    cmatrix && gvec_other )
```

### 7.2.2.12 operator=() [2/2]

```
cmatrix& gsl::cmatrix::operator= (
    const cmatrix & gvec_other )
```

### 7.2.2.13 print()

```
void gsl::cmatrix::print (
    FILE * out = stdout ) const
```

Pretty-print the complex matrix to file stream.

### 7.2.2.14 reshape()

```
cmatrix gsl::cmatrix::reshape (
    size_t n,
    size_t m ) const
```

Return a new  $n \times m$  [gsl::cmatrix](#) with same elements.

### 7.2.2.15 resize()

```
void gsl::cmatrix::resize (
    size_t n,
    size_t m )
```

Resize the [gsl::cmatrix](#), setting elements to zero.

#### 7.2.2.16 row()

```
crow_view gsl::cmatrix::row (
    size_t i )
```

Return a view to a row of the complex matrix.

#### 7.2.2.17 set()

```
void gsl::cmatrix::set (
    size_t i,
    size_t j,
    complex z )
```

#### 7.2.2.18 size()

```
size_t gsl::cmatrix::size ( ) const
```

#### 7.2.2.19 submatrix()

```
cmatrix_view gsl::cmatrix::submatrix (
    size_t i,
    size_t j,
    size_t n,
    size_t m )
```

Return a view to a submatrix of the complex matrix.

### 7.2.3 Friends And Related Function Documentation

#### 7.2.3.1 ccolumn\_view

```
friend class ccolumn_view [friend]
```

#### 7.2.3.2 cmatrix\_view

```
friend class cmatrix_view [friend]
```



### 7.2.3.3 crow\_view

```
friend class crow_view [friend]
```

### 7.2.3.4 matrix

```
friend class matrix [friend]
```

### 7.2.3.5 operator\*

```
cmatrix operator* (
    const cmatrix & A,
    const cmatrix & B ) [friend]
```

## 7.2.4 Member Data Documentation

### 7.2.4.1 gmat

```
gsl_matrix_complex* gsl::cmatrix::gmat [protected]
```

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

- /home/jspainhour/spheroidal\_cpp/include/yawg/[cmatrix.h](#)

## 7.3 gsl::cmatrix\_view Class Reference

A wrapper class for gsl\_matrix\_complex\_view.

```
#include <cmatrix.h>
```

### Public Member Functions

- [operator cmatrix](#) () const  
*"Dereferences" a [matrix\\_view](#) into independent [gsl::cmatrix](#) object*
- [cmatrix\\_view](#) (gsl\_matrix\_complex\_view [gmat\\_view](#))  
*Construct a view of a [gsl::cmatrix](#) through another [cmatrix\\_view](#).*
- [cmatrix\\_view](#) (const [cmatrix](#) &m)  
*Construct a view of the given [gsl::cmatrix](#).*
- [cmatrix\\_view](#) & [operator=](#) (const [cmatrix](#) &v)  
*Assignment to a complex matrix view from a complex matrix.*
- [cmatrix\\_view](#) & [operator=](#) ([cmatrix\\_view](#) v)  
*Assignment to a complex matrix view from another complex matrix view.*
- void [print](#) (FILE \*out=stdout) const  
*Pretty-print the viewed complex matrix to file stream.*
- const gsl\_matrix\_complex \* [get\\_gsl\\_ptr](#) () const  
*Return a constant pointer to the underlying [gsl\\_matrix\\_complex](#).*

## Protected Attributes

- `gsl_matrix_complex_view` [gmat\\_view](#)

### 7.3.1 Detailed Description

A wrapper class for `gsl_matrix_complex_view`.

Stores a `gsl_matrix_complex_view` and uses it to access original member data.

### 7.3.2 Constructor & Destructor Documentation

#### 7.3.2.1 `cmatrix_view()` [1/2]

```
gsl::cmatrix_view::cmatrix_view (  
    gsl_matrix_complex_view gmat_view )    [inline]
```

Construct a view of a [gsl::cmatrix](#) through another [cmatrix\\_view](#).

#### 7.3.2.2 `cmatrix_view()` [2/2]

```
gsl::cmatrix_view::cmatrix_view (  
    const cmatrix & m )    [inline]
```

Construct a view of the given [gsl::cmatrix](#).

### 7.3.3 Member Function Documentation

#### 7.3.3.1 `get_gsl_ptr()`

```
const gsl_matrix_complex* gsl::cmatrix_view::get_gsl_ptr ( ) const    [inline]
```

Return a constant pointer to the underlying `gsl_matrix_complex`.

### 7.3.3.2 operator cmatrix()

```
gsl::cmatrix_view::operator cmatrix ( ) const [inline]
```

"Dereferences" a [matrix\\_view](#) into independent [gsl::cmatrix](#) object

### 7.3.3.3 operator=() [1/2]

```
cmatrix_view& gsl::cmatrix_view::operator= (
    cmatrix_view v )
```

Assignment to a complex matrix view from another complex matrix view.

### 7.3.3.4 operator=() [2/2]

```
cmatrix_view& gsl::cmatrix_view::operator= (
    const cmatrix & v )
```

Assignment to a complex matrix view from a complex matrix.

### 7.3.3.5 print()

```
void gsl::cmatrix_view::print (
    FILE * out = stdout ) const
```

Pretty-print the viewed complex matrix to file stream.

## 7.3.4 Member Data Documentation

### 7.3.4.1 gmat\_view

```
gsl_matrix_complex_view gsl::cmatrix_view::gmat_view [protected]
```

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

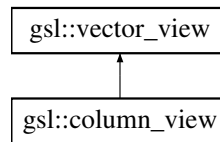
- [/home/jspainhour/spheroidal\\_cpp/include/yawg/cmatrix.h](#)

## 7.4 gsl::column\_view Class Reference

A subclass of [vector\\_view](#) for non-stride-1 vectors.

```
#include <vector.h>
```

Inheritance diagram for `gsl::column_view`:



### Public Member Functions

- [column\\_view](#) ([gsl\\_vector\\_view](#) [gvec\\_view](#))  
Construct [column\\_view](#) from existing vector view.
- [column\\_view](#) (const [vector](#) &v)  
Construct [column\\_view](#) from vector.

### Additional Inherited Members

#### 7.4.1 Detailed Description

A subclass of [vector\\_view](#) for non-stride-1 vectors.

#### 7.4.2 Constructor & Destructor Documentation

##### 7.4.2.1 column\_view() [1/2]

```
gsl::column_view::column_view (
    gsl_vector_view gvec_view ) [inline]
```

Construct [column\\_view](#) from existing vector view.

##### 7.4.2.2 column\_view() [2/2]

```
gsl::column_view::column_view (
    const vector & v ) [inline]
```

Construct [column\\_view](#) from vector.

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

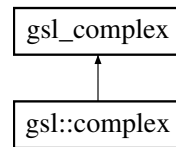
- `/home/jspainhour/spheroidal_cpp/include/yawg/vector.h`

## 7.5 gsl::complex Class Reference

Wrapper class for gsl\_complex structs.

```
#include <complex.h>
```

Inheritance diagram for gsl::complex:



### Public Member Functions

- `complex` ()
- `complex` (double x)
- `complex` (double re, double im)
- `complex` (gsl\_complex z)
- double `real` () const
- double `imag` () const
- double `abs` () const
- double `abs2` () const
- double `arg` () const
- `complex` & `operator+=` (`complex` gsl\_complex\_other)
- `complex` & `operator-=` (`complex` gsl\_complex\_other)
- `complex` & `operator*=` (`complex` gsl\_complex\_other)
- `complex` & `operator/=` (`complex` gsl\_complex\_other)
- `complex` & `operator+=` (double x)
- `complex` & `operator-=` (double x)
- `complex` & `operator*=` (double x)
- `complex` & `operator/=` (double x)
- `complex` & `operator=` (`complex` gsl\_complex\_other)
- void `set` (double re, double im)
- void `set` (`complex` z)
- `complex operator-` () const
- void `print` () const

### Friends

- class `complex_ref`
- `complex operator+` (`complex` a, `complex` b)
- `complex operator-` (`complex` a, `complex` b)
- `complex operator*` (`complex` a, `complex` b)
- `complex operator/` (`complex` a, `complex` b)
- bool `operator==` (`complex` a, `complex` b)

#### 7.5.1 Detailed Description

Wrapper class for gsl\_complex structs.

Inherits `double dat[2]` from `gsl_complex` and provides a number of convenience functions.

## 7.5.2 Constructor & Destructor Documentation

### 7.5.2.1 `complex()` [1/4]

```
gsl::complex::complex ( ) [inline]
```

### 7.5.2.2 `complex()` [2/4]

```
gsl::complex::complex (
    double x ) [inline]
```

### 7.5.2.3 `complex()` [3/4]

```
gsl::complex::complex (
    double re,
    double im ) [inline]
```

### 7.5.2.4 `complex()` [4/4]

```
gsl::complex::complex (
    gsl_complex z ) [inline]
```

## 7.5.3 Member Function Documentation

### 7.5.3.1 `abs()`

```
double gsl::complex::abs ( ) const [inline]
```

### 7.5.3.2 `abs2()`

```
double gsl::complex::abs2 ( ) const [inline]
```

### 7.5.3.3 arg()

```
double gsl::complex::arg ( ) const [inline]
```

### 7.5.3.4 imag()

```
double gsl::complex::imag ( ) const [inline]
```

### 7.5.3.5 operator\*=( ) [1/2]

```
complex& gsl::complex::operator*= (
    complex gsl_complex_other )
```

### 7.5.3.6 operator\*=( ) [2/2]

```
complex& gsl::complex::operator*= (
    double x )
```

### 7.5.3.7 operator+=( ) [1/2]

```
complex& gsl::complex::operator+= (
    complex gsl_complex_other )
```

### 7.5.3.8 operator+=( ) [2/2]

```
complex& gsl::complex::operator+= (
    double x )
```

### 7.5.3.9 operator-()

```
complex gsl::complex::operator- ( ) const [inline]
```

#### 7.5.3.10 operator-=( ) [1/2]

```
complex& gsl::complex::operator-= (
    complex gsl_complex_other )
```

#### 7.5.3.11 operator-=( ) [2/2]

```
complex& gsl::complex::operator-= (
    double x )
```

#### 7.5.3.12 operator/=( ) [1/2]

```
complex& gsl::complex::operator/= (
    complex gsl_complex_other )
```

#### 7.5.3.13 operator/=( ) [2/2]

```
complex& gsl::complex::operator/= (
    double x )
```

#### 7.5.3.14 operator=( )

```
complex& gsl::complex::operator= (
    complex gsl_complex_other )
```

#### 7.5.3.15 print()

```
void gsl::complex::print ( ) const
```

#### 7.5.3.16 real()

```
double gsl::complex::real ( ) const [inline]
```



#### 7.5.3.17 set() [1/2]

```
void gsl::complex::set (  
    complex z ) [inline]
```

#### 7.5.3.18 set() [2/2]

```
void gsl::complex::set (  
    double re,  
    double im ) [inline]
```

### 7.5.4 Friends And Related Function Documentation

#### 7.5.4.1 complex\_ref

```
friend class complex_ref [friend]
```

#### 7.5.4.2 operator\*

```
complex operator* (  
    complex a,  
    complex b ) [friend]
```

#### 7.5.4.3 operator+

```
complex operator+ (  
    complex a,  
    complex b ) [friend]
```

#### 7.5.4.4 operator-

```
complex operator- (  
    complex a,  
    complex b ) [friend]
```

#### 7.5.4.5 operator/

```
complex operator/ (
    complex a,
    complex b ) [friend]
```

#### 7.5.4.6 operator==

```
bool operator== (
    complex a,
    complex b ) [friend]
```

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

- [/home/jspainhour/spheroidal\\_cpp/include/yawg/complex.h](#)

## 7.6 gsl::complex\_ref Class Reference

Stores a refernce to a [gsl::complex](#) object.

```
#include <complex.h>
```

### Public Member Functions

- [operator complex](#) () const  
*"Dereferences" a [complex\\_ref](#) into independent [gsl::complex](#) object*
- [complex\\_ref](#) ([complex\\_ref](#) &z)  
*Constructs a reference to another [complex\\_ref](#) object.*
- [complex\\_ref](#) ([gsl\\_complex](#) \*z)  
*Constructs a reference to a [gsl\\_complex](#) struct.*
- [complex\\_ref](#) ([complex](#) &z)  
*Constructs a reference to a [gsl::complex](#) object.*
- [complex\\_ref](#) & [operator=](#) ([complex](#) z)  
*Assigns values of [gsl::complex](#) object to the reference.*
- [complex\\_ref](#) & [operator=](#) ([complex\\_ref](#) z)  
*Assigns values of one [complex\\_ref](#) object to another.*
- double [real](#) () const
- double [imag](#) () const

### Protected Member Functions

- [complex\\_ref](#) ()

### Protected Attributes

- double \* [dat](#)

## 7.6.1 Detailed Description

Stores a reference to a [gsl::complex](#) object.

This class is necessary to communicate between `gsl_complex` and [gsl::complex](#) so that overloads of `()` work [gsl::cvector](#) and [gsl::cmatrix](#).

Implementation heavily inspired by `ccgsl` ( <https://ccgsl.sourceforge.net/> )

## 7.6.2 Constructor & Destructor Documentation

### 7.6.2.1 `complex_ref()` [1/4]

```
gsl::complex_ref::complex_ref ( ) [inline], [protected]
```

### 7.6.2.2 `complex_ref()` [2/4]

```
gsl::complex_ref::complex_ref (
    complex_ref & z ) [inline]
```

Constructs a reference to another `complex_ref` object.

### 7.6.2.3 `complex_ref()` [3/4]

```
gsl::complex_ref::complex_ref (
    gsl_complex * z ) [inline]
```

Constructs a reference to a `gsl_complex` struct.

### 7.6.2.4 `complex_ref()` [4/4]

```
gsl::complex_ref::complex_ref (
    complex & z ) [inline]
```

Constructs a reference to a [gsl::complex](#) object.

## 7.6.3 Member Function Documentation

#### 7.6.3.1 imag()

```
double gsl::complex_ref::imag ( ) const [inline]
```

#### 7.6.3.2 operator complex()

```
gsl::complex_ref::operator complex ( ) const [inline]
```

"Dereferences" a `complex_ref` into independent `gsl::complex` object

#### 7.6.3.3 operator=() [1/2]

```
complex_ref& gsl::complex_ref::operator= (
    complex z ) [inline]
```

Assigns values of `gsl::complex` object to the reference.

#### 7.6.3.4 operator=() [2/2]

```
complex_ref& gsl::complex_ref::operator= (
    complex_ref z ) [inline]
```

Assigns values of one `complex_ref` object to another.

#### Note

This is an alternative to the default assignment operator, which does not work for unknown reasons.

#### 7.6.3.5 real()

```
double gsl::complex_ref::real ( ) const [inline]
```

### 7.6.4 Member Data Documentation

#### 7.6.4.1 dat

```
double* gsl::complex_ref::dat [protected]
```

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

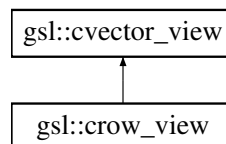
- [/home/jspainhour/spheroidal\\_cpp/include/yawg/complex.h](#)

## 7.7 gsl::crow\_view Class Reference

A subclass of [cvector\\_view](#) for stride-1 vectors.

```
#include <cvector.h>
```

Inheritance diagram for `gsl::crow_view`:



### Public Member Functions

- [crow\\_view](#) ([gsl\\_vector\\_complex\\_view](#) [gvec\\_view](#))  
*Construct [crow\\_view](#) from existing [cvector](#) view, checking that stride is 1.*
- [crow\\_view](#) (const [cvector](#) &v)  
*Construct [crow\\_view](#) from [cvector](#), checking that stride is 1.*
- [cmatrix\\_view](#) [reshape](#) (size\_t n, size\_t m)  
*Return a [cmatrix](#) view out of the elements of the row.*

### Additional Inherited Members

#### 7.7.1 Detailed Description

A subclass of [cvector\\_view](#) for stride-1 vectors.

#### 7.7.2 Constructor & Destructor Documentation

##### 7.7.2.1 crow\_view() [1/2]

```
gsl::crow_view::crow_view (
    gsl_vector_complex_view gvec_view ) [inline]
```

Construct [crow\\_view](#) from existing [cvector](#) view, checking that stride is 1.

### 7.7.2.2 `crow_view()` [2/2]

```
gsl::crow_view::crow_view (
    const cvector & v ) [inline]
```

Construct [crow\\_view](#) from [cvector](#), checking that stride is 1.

## 7.7.3 Member Function Documentation

### 7.7.3.1 `reshape()`

```
cmatrix\_view gsl::crow_view::reshape (
    size_t n,
    size_t m )
```

Return a [cmatrix](#) view out of the elements of the row.

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

- [/home/jspainhour/spheroidal\\_cpp/include/yawg/cvector.h](#)

## 7.8 `gsl::cvector` Class Reference

A wrapper class for `gsl_vector_complex`.

```
#include <cvector.h>
```

### Public Member Functions

- [cvector](#) ()  
*Construct empty vector.*
- [cvector](#) (size\_t n)  
*Construct zero vector of size n.*
- [cvector](#) (const [gsl\\_vector\\_complex](#) \*gvec\_other)  
*Construct new [gsl::cvector](#) from [gsl\\_vector\\_complex](#).*
- [cvector](#) (const [cvector](#) &gvec\_other)
- [cvector](#) ([cvector](#) &&gvec\_other)
- [cvector](#) (const [vector](#) &vec)  
*Construct new [gsl::cvector](#) from [gsl::vector](#).*
- [cvector](#) & operator= (const [cvector](#) &gvec\_other)
- [cvector](#) & operator= ([cvector](#) &&gvec\_other)
- [~cvector](#) ()
- [complex\\_ref operator\(\)](#) (size\_t i)  
*Return a reference to the element at position (i,j)*
- void [set](#) (size\_t i, [complex](#) z)
- const [complex\\_ref operator\(\)](#) (size\_t i) const

- `complex get (size_t i) const`
- `size_t size () const`
- `gsl_vector_complex * get_gsl_ptr () const`  
*Access the pointer to the underlying gsl\_vector\_complex.*
- `void resize (size_t n)`  
*Resize the `gsl::cvector`, setting elements to zero.*
- `void clear ()`  
*Clear the `gsl::cvector`, free underlying memory.*
- `void print (FILE *out=stdout) const`  
*Pretty-print the complex vector to file stream.*
- `double norm () const`  
*Return the 2-norm of the vector.*
- `cvector & operator+= (const cvector &gvec_other)`
- `cvector & operator-= (const cvector &gvec_other)`
- `cvector_view subvector (size_t offset, size_t size)`  
*Return a view to a subvector of the vector.*
- `cvector_view view ()`  
*Return a view to the entire vector.*

## Protected Member Functions

- `void free ()`  
*Private function to free allocated memory.*
- `void calloc (size_t n)`  
*Private function to (continuously) allocate memory.*

## Protected Attributes

- `gsl_vector_complex * gvec`

## Friends

- class `vector`
- class `cvector_view`
- class `crow_view`
- class `ccolumn_view`
- class `cmatrix`
- `cvector operator* (complex a, const cvector &v)`
- `cvector operator* (complex a, cvector &&v)`
- `cvector operator* (const cvector &v, complex a)`
- `cvector operator* (cvector &&v, complex a)`
- `cvector operator+ (const cvector &v1, const cvector &v2)`
- `cvector operator+ (cvector &&v1, const cvector &v2)`
- `cvector operator+ (const cvector &v1, cvector &&v2)`
- `cvector operator+ (cvector &&v1, cvector &&v2)`
- `cvector operator- (const cvector &v1, const cvector &v2)`
- `cvector operator- (cvector &&v1, const cvector &v2)`
- `cvector operator- (const cvector &v1, cvector &&v2)`
- `cvector operator- (cvector &&v1, cvector &&v2)`
- `bool operator== (const cvector &v1, const cvector &v2)`

### 7.8.1 Detailed Description

A wrapper class for `gsl_vector_complex`.

Stores and operates on a pointer to a `gsl_vector_complex`.

### 7.8.2 Constructor & Destructor Documentation

#### 7.8.2.1 `cvector()` [1/6]

```
gsl::cvector::cvector ( )
```

Construct empty vector.

#### 7.8.2.2 `cvector()` [2/6]

```
gsl::cvector::cvector (
    size_t n ) [explicit]
```

Construct zero vector of size `n`.

#### 7.8.2.3 `cvector()` [3/6]

```
gsl::cvector::cvector (
    const gsl_vector_complex * gvec_other )
```

Construct new [gsl::cvector](#) from `gsl_vector_complex`.

#### 7.8.2.4 `cvector()` [4/6]

```
gsl::cvector::cvector (
    const cvector & gvec_other )
```

#### 7.8.2.5 `cvector()` [5/6]

```
gsl::cvector::cvector (
    cvector && gvec_other )
```



### 7.8.2.6 `cvector()` [6/6]

```
gsl::cvector::cvector (
    const vector & vec )
```

Construct new [gsl::cvector](#) from [gsl::vector](#).

### 7.8.2.7 `~cvector()`

```
gsl::cvector::~~cvector ( )
```

## 7.8.3 Member Function Documentation

### 7.8.3.1 `calloc()`

```
void gsl::cvector::calloc (
    size_t n ) [protected]
```

Private function to (continuously) allocate memory.

### 7.8.3.2 `clear()`

```
void gsl::cvector::clear ( )
```

Clear the [gsl::cvector](#), free underlying memory.

### 7.8.3.3 `free()`

```
void gsl::cvector::free ( ) [protected]
```

Private function to free allocated memory.

### 7.8.3.4 `get()`

```
complex gsl::cvector::get (
    size_t i ) const
```

#### 7.8.3.5 get\_gsl\_ptr()

```
gsl_vector_complex* gsl::cvector::get_gsl_ptr ( ) const [inline]
```

Access the pointer to the underlying `gsl_vector_complex`.

#### 7.8.3.6 norm()

```
double gsl::cvector::norm ( ) const [inline]
```

Return the 2-norm of the vector.

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

```
complex_ref gsl::cvector::operator() (
    size_t i )
```

Return a reference to the element at position (i,j)

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

```
const complex_ref gsl::cvector::operator() (
    size_t i ) const
```

#### 7.8.3.9 operator+=()

```
cvector& gsl::cvector::operator+= (
    const cvector & gvec_other )
```

#### 7.8.3.10 operator-=()

```
cvector& gsl::cvector::operator-= (
    const cvector & gvec_other )
```

#### 7.8.3.11 operator=() [1/2]

```
cvector& gsl::cvector::operator= (
    const cvector & gvec_other )
```

#### 7.8.3.12 operator=() [2/2]

```
cvector& gsl::cvector::operator= (
    cvector && gvec_other )
```

#### 7.8.3.13 print()

```
void gsl::cvector::print (
    FILE * out = stdout ) const
```

Pretty-print the complex vector to file stream.

#### 7.8.3.14 resize()

```
void gsl::cvector::resize (
    size_t n )
```

Resize the [gsl::cvector](#), setting elements to zero.

#### 7.8.3.15 set()

```
void gsl::cvector::set (
    size_t i,
    complex z )
```

#### 7.8.3.16 size()

```
size_t gsl::cvector::size ( ) const
```

#### 7.8.3.17 subvector()

```
cvector_view gsl::cvector::subvector (
    size_t offset,
    size_t size )
```

Return a view to a subvector of the vector.

#### 7.8.3.18 view()

```
cvector_view gsl::cvector::view ( )
```

Return a view to the entire vector.

### 7.8.4 Friends And Related Function Documentation

#### 7.8.4.1 ccolumn\_view

```
friend class ccolumn_view [friend]
```

#### 7.8.4.2 cmatrix

```
friend class cmatrix [friend]
```

#### 7.8.4.3 crow\_view

```
friend class crow_view [friend]
```

#### 7.8.4.4 cvector\_view

```
friend class cvector_view [friend]
```

#### 7.8.4.5 operator\* [1/4]

```
cvector operator* (
    complex a,
    const cvector & v ) [friend]
```

#### 7.8.4.6 operator\* [2/4]

```
cvector operator* (
    complex a,
    cvector && v ) [friend]
```

#### 7.8.4.7 operator\* [3/4]

```
cvector operator* (
    const cvector & v,
    complex a ) [friend]
```

#### 7.8.4.8 operator\* [4/4]

```
cvector operator* (
    cvector && v,
    complex a ) [friend]
```

#### 7.8.4.9 operator+ [1/4]

```
cvector operator+ (
    const cvector & v1,
    const cvector & v2 ) [friend]
```

#### 7.8.4.10 operator+ [2/4]

```
cvector operator+ (
    const cvector & v1,
    cvector && v2 ) [friend]
```

**7.8.4.11 operator+ [3/4]**

```
cvector operator+ (  
    cvector && v1,  
    const cvector & v2 ) [friend]
```

**7.8.4.12 operator+ [4/4]**

```
cvector operator+ (  
    cvector && v1,  
    cvector && v2 ) [friend]
```

**7.8.4.13 operator- [1/4]**

```
cvector operator- (  
    const cvector & v1,  
    const cvector & v2 ) [friend]
```

**7.8.4.14 operator- [2/4]**

```
cvector operator- (  
    const cvector & v1,  
    cvector && v2 ) [friend]
```

**7.8.4.15 operator- [3/4]**

```
cvector operator- (  
    cvector && v1,  
    const cvector & v2 ) [friend]
```

**7.8.4.16 operator- [4/4]**

```
cvector operator- (  
    cvector && v1,  
    cvector && v2 ) [friend]
```

#### 7.8.4.17 operator==

```
bool operator== (
    const cvector & v1,
    const cvector & v2 ) [friend]
```

#### 7.8.4.18 vector

```
friend class vector [friend]
```

### 7.8.5 Member Data Documentation

#### 7.8.5.1 gvec

```
gsl\_vector\_complex\* gsl::cvector::gvec [protected]
```

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

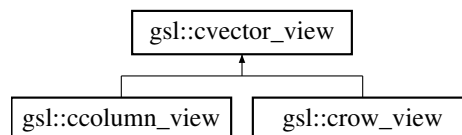
- [/home/jspainhour/spheroidal\\_cpp/include/yawg/cvector.h](/home/jspainhour/spheroidal_cpp/include/yawg/cvector.h)

## 7.9 gsl::cvector\_view Class Reference

A wrapper class for [gsl\\_vector\\_complex\\_view](#).

```
#include <cvector.h>
```

Inheritance diagram for [gsl::cvector\\_view](#):



### Public Member Functions

- [operator cvector](#) () const  
*"Dereferences" a [cvector\\_view](#) into independent [gsl::cvector](#) object*
- [cvector\\_view](#) ([gsl\\_vector\\_complex\\_view](#) [gvec\\_view](#))  
*Construct a view of a [gsl::cvector](#) through another [cvector\\_view](#).*
- [cvector\\_view](#) (const [cvector](#) &v)  
*Construct a view of the given [gsl::cvector](#).*
- [cvector\\_view](#) & [operator=](#) (const [cvector](#) &v)  
*Assignment to a [cvector](#) view from a [cvector](#).*
- [cvector\\_view](#) & [operator=](#) ([cvector\\_view](#) v)  
*Assignment to a [cvector](#) view from another [cvector](#) view.*
- void [print](#) (FILE \*out=stdout) const  
*Pretty-print the viewed vector to file stream.*
- const [gsl\\_vector\\_complex](#) \* [get\\_gsl\\_ptr](#) () const  
*Return a constant pointer to the underlying [gsl\\_vector\\_complex](#).*

## Protected Attributes

- `gsl_vector_complex_view` [gvec\\_view](#)

### 7.9.1 Detailed Description

A wrapper class for `gsl_vector_complex_view`.

Stores a `gsl_vector_complex_view` and uses it to access original member data.

### 7.9.2 Constructor & Destructor Documentation

#### 7.9.2.1 `cvector_view()` [1/2]

```
gsl::cvector_view::cvector_view (  
    gsl_vector_complex_view gvec_view )    [inline]
```

Construct a view of a [gsl::cvector](#) through another [cvector\\_view](#).

#### 7.9.2.2 `cvector_view()` [2/2]

```
gsl::cvector_view::cvector_view (  
    const cvector & v )    [inline]
```

Construct a view of the given [gsl::cvector](#).

### 7.9.3 Member Function Documentation

#### 7.9.3.1 `get_gsl_ptr()`

```
const gsl_vector_complex* gsl::cvector_view::get_gsl_ptr ( ) const    [inline]
```

Return a constant pointer to the underlying `gsl_vector_complex`.



### 7.9.3.2 operator cvector()

```
gsl::cvector_view::operator cvector ( ) const [inline]
```

"Dereferences" a [cvector\\_view](#) into independent [gsl::cvector](#) object

### 7.9.3.3 operator=() [1/2]

```
cvector_view& gsl::cvector_view::operator= (
    const cvector & v )
```

Assignment to a cvector view from a cvector.

### 7.9.3.4 operator=() [2/2]

```
cvector_view& gsl::cvector_view::operator= (
    cvector_view v )
```

Assignment to a cvector view from another cvector view.

### 7.9.3.5 print()

```
void gsl::cvector_view::print (
    FILE * out = stdout ) const
```

Pretty-print the viewed vector to file stream.

## 7.9.4 Member Data Documentation

### 7.9.4.1 gvec\_view

```
gsl_vector_complex_view gsl::cvector_view::gvec_view [protected]
```

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

- [/home/jspainhour/spheroidal\\_cpp/include/yawg/cvector.h](#)

## 7.10 gsl::matrix Class Reference

A wrapper class for gsl\_matrix.

```
#include <matrix.h>
```

### Public Member Functions

- [matrix](#) ()  
*Construct empty matrix.*
- [matrix](#) (size\_t n, size\_t m)  
*Construct zero matrix of size n x m.*
- [matrix](#) (const gsl\_matrix \*gsl\_mat)  
*Construct new [gsl::matrix](#) from [gsl\\_matrix](#).*
- [matrix](#) (const [vector](#) &v)  
*Construct new n x 1 [gsl::matrix](#) from [gsl::vector](#).*
- [matrix](#) (const [matrix](#) &M, size\_t n, size\_t m)  
*Copy constructor creating n x m matrix.*
- [matrix](#) (const [matrix](#) &M)
- [matrix](#) ([matrix](#) &&M)
- [matrix](#) & [operator=](#) (const [matrix](#) &M)
- [matrix](#) & [operator=](#) ([matrix](#) &&M)
- [~matrix](#) ()
- double & [operator\(\)](#) (size\_t i, size\_t j)
- void [set](#) (size\_t i, size\_t j, double val)
- double [operator\(\)](#) (size\_t i, size\_t j) const
- double [get](#) (size\_t i, size\_t j) const
- size\_t [size](#) () const
- size\_t [nrows](#) () const
- size\_t [ncols](#) () const
- gsl\_matrix \* [get\\_gsl\\_ptr](#) () const  
*Access the pointer to the underlying [gsl\\_matrix](#).*
- void [resize](#) (size\_t n, size\_t m)  
*Resize the [gsl::matrix](#), setting elements to zero.*
- void [clear](#) ()  
*Clear the [gsl::matrix](#), free underlying memory.*
- [matrix](#) [reshape](#) (size\_t n, size\_t m) const  
*Return a new n x m [gsl::matrix](#) with same elements.*
- void [print](#) (FILE \*out=stdout) const  
*Pretty-print the matrix to file stream.*
- void [print\\_csv](#) (FILE \*out=stdout) const  
*Print the matrix to file stream in CSV format.*
- void [load\\_csv](#) (FILE \*in=stdin)  
*Load the matrix from a file stream in CSV format.*
- [matrix\\_view](#) [submatrix](#) (size\_t i, size\_t j, size\_t n, size\_t m)  
*Return a view to a submatrix of the matrix.*
- [row\\_view](#) [row](#) (size\_t i)  
*Return a view to a row of the matrix.*
- [column\\_view](#) [column](#) (size\_t j)  
*Return a view to a column of the matrix.*

## Protected Member Functions

- void `free` ()  
*Private function to free allocated memory.*
- void `calloc` (size\_t n, size\_t m)  
*Private function to (continuously) allocate memory.*

## Protected Attributes

- gsl\_matrix \* `gmat`

## Friends

- class `cmatrix`
- class `matrix_view`
- class `row_view`
- class `column_view`
- `matrix operator*` (const `matrix` &A, const `matrix` &B)

### 7.10.1 Detailed Description

A wrapper class for `gsl_matrix`.

Stores and operates on a pointer to a `gsl_matrix`.

### 7.10.2 Constructor & Destructor Documentation

#### 7.10.2.1 `matrix()` [1/7]

```
gsl::matrix::matrix ( )
```

Construct empty matrix.

#### 7.10.2.2 `matrix()` [2/7]

```
gsl::matrix::matrix (
    size_t n,
    size_t m )
```

Construct zero matrix of size n x m.

### 7.10.2.3 `matrix()` [3/7]

```
gsl::matrix::matrix (
    const gsl_matrix * gsl_mat )
```

Construct new `gsl::matrix` from `gsl_matrix`.

### 7.10.2.4 `matrix()` [4/7]

```
gsl::matrix::matrix (
    const vector & v )
```

Construct new  $n \times 1$  `gsl::matrix` from `gsl::vector`.

### 7.10.2.5 `matrix()` [5/7]

```
gsl::matrix::matrix (
    const matrix & M,
    size_t n,
    size_t m )
```

Copy constructor creating  $n \times m$  matrix.

### 7.10.2.6 `matrix()` [6/7]

```
gsl::matrix::matrix (
    const matrix & M )
```

### 7.10.2.7 `matrix()` [7/7]

```
gsl::matrix::matrix (
    matrix && M )
```

### 7.10.2.8 `~matrix()`

```
gsl::matrix::~~matrix ( )
```

## 7.10.3 Member Function Documentation

### 7.10.3.1 calloc()

```
void gsl::matrix::calloc (
    size_t n,
    size_t m ) [protected]
```

Private function to (continuously) allocate memory.

### 7.10.3.2 clear()

```
void gsl::matrix::clear ( )
```

Clear the [gsl::matrix](#), free underlying memory.

### 7.10.3.3 column()

```
column_view gsl::matrix::column (
    size_t j )
```

Return a view to a column of the matrix.

### 7.10.3.4 free()

```
void gsl::matrix::free ( ) [protected]
```

Private function to free allocated memory.

### 7.10.3.5 get()

```
double gsl::matrix::get (
    size_t i,
    size_t j ) const
```

#### 7.10.3.6 get\_gsl\_ptr()

```
gsl_matrix* gsl::matrix::get_gsl_ptr ( ) const [inline]
```

Access the pointer to the underlying `gsl_matrix`.

#### 7.10.3.7 load\_csv()

```
void gsl::matrix::load_csv (
    FILE * in = stdin )
```

Load the matrix from a file stream in CSV format.

#### 7.10.3.8 ncols()

```
size_t gsl::matrix::ncols ( ) const
```

#### 7.10.3.9 nrows()

```
size_t gsl::matrix::nrows ( ) const
```

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

```
double& gsl::matrix::operator() (
    size_t i,
    size_t j )
```

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

```
double gsl::matrix::operator() (
    size_t i,
    size_t j ) const
```

### 7.10.3.12 operator=() [1/2]

```
matrix& gsl::matrix::operator= (
    const matrix & M )
```

### 7.10.3.13 operator=() [2/2]

```
matrix& gsl::matrix::operator= (
    matrix && M )
```

### 7.10.3.14 print()

```
void gsl::matrix::print (
    FILE * out = stdout ) const
```

Pretty-print the matrix to file stream.

### 7.10.3.15 print\_csv()

```
void gsl::matrix::print_csv (
    FILE * out = stdout ) const
```

Print the matrix to file stream in CSV format.

### 7.10.3.16 reshape()

```
matrix gsl::matrix::reshape (
    size_t n,
    size_t m ) const
```

Return a new  $n \times m$  [gsl::matrix](#) with same elements.

### 7.10.3.17 resize()

```
void gsl::matrix::resize (
    size_t n,
    size_t m )
```

Resize the [gsl::matrix](#), setting elements to zero.

#### 7.10.3.18 row()

```
row_view gsl::matrix::row (
    size_t i )
```

Return a view to a row of the matrix.

#### 7.10.3.19 set()

```
void gsl::matrix::set (
    size_t i,
    size_t j,
    double val )
```

#### 7.10.3.20 size()

```
size_t gsl::matrix::size ( ) const
```

#### 7.10.3.21 submatrix()

```
matrix_view gsl::matrix::submatrix (
    size_t i,
    size_t j,
    size_t n,
    size_t m )
```

Return a view to a submatrix of the matrix.

### 7.10.4 Friends And Related Function Documentation

#### 7.10.4.1 cmatrix

```
friend class cmatrix [friend]
```

#### 7.10.4.2 column\_view

```
friend class column_view [friend]
```



### 7.10.4.3 matrix\_view

```
friend class matrix_view [friend]
```

### 7.10.4.4 operator\*

```
matrix operator* (
    const matrix & A,
    const matrix & B ) [friend]
```

### 7.10.4.5 row\_view

```
friend class row_view [friend]
```

## 7.10.5 Member Data Documentation

### 7.10.5.1 gmat

```
gsl_matrix* gsl::matrix::gmat [protected]
```

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

- /home/jspainhour/spheroidal\_cpp/include/yawg/matrix.h

## 7.11 gsl::matrix\_view Class Reference

A wrapper class for gsl\_matrix\_view.

```
#include <matrix.h>
```

### Public Member Functions

- [operator matrix](#) () const  
*"Dereferences" a [matrix\\_view](#) into independent [gsl::matrix](#) object*
- [matrix\\_view](#) (gsl\_matrix\_view [gmat\\_view](#))  
*Construct a view of a [gsl::matrix](#) through another [matrix\\_view](#).*
- [matrix\\_view](#) (const [matrix](#) &m)  
*Construct a view of the given [gsl::matrix](#).*
- [matrix\\_view](#) & [operator=](#) (const [matrix](#) &M)  
*Assignment to a matrix view from a matrix.*
- [matrix\\_view](#) & [operator=](#) ([matrix\\_view](#) Mv)  
*Assignment to a matrix view from another matrix view.*
- void [print](#) (FILE \*out=stdout) const  
*Pretty-print the viewed matrix to file stream.*
- const gsl\_matrix \* [get\\_gsl\\_ptr](#) () const  
*Return a constant pointer to the underlying [gsl\\_matrix](#).*

## Protected Attributes

- `gsl_matrix_view` [gmat\\_view](#)

### 7.11.1 Detailed Description

A wrapper class for `gsl_matrix_view`.

Stores a `gsl_matrix_view` and uses it to access original member data.

### 7.11.2 Constructor & Destructor Documentation

#### 7.11.2.1 `matrix_view()` [1/2]

```
gsl::matrix_view::matrix_view (  
    gsl_matrix_view gmat_view ) [inline]
```

Construct a view of a [gsl::matrix](#) through another [matrix\\_view](#).

#### 7.11.2.2 `matrix_view()` [2/2]

```
gsl::matrix_view::matrix_view (  
    const matrix & m ) [inline]
```

Construct a view of the given [gsl::matrix](#).

### 7.11.3 Member Function Documentation

#### 7.11.3.1 `get_gsl_ptr()`

```
const gsl_matrix* gsl::matrix_view::get_gsl_ptr ( ) const [inline]
```

Return a constant pointer to the underlying `gsl_matrix`.

### 7.11.3.2 operator matrix()

```
gsl::matrix_view::operator matrix ( ) const [inline]
```

"Dereferences" a [matrix\\_view](#) into independent [gsl::matrix](#) object

### 7.11.3.3 operator=() [1/2]

```
matrix\_view& gsl::matrix_view::operator= (
    const matrix & M )
```

Assignment to a matrix view from a matrix.

### 7.11.3.4 operator=() [2/2]

```
matrix\_view& gsl::matrix_view::operator= (
    matrix\_view Mv )
```

Assignment to a matrix view from another matrix view.

### 7.11.3.5 print()

```
void gsl::matrix_view::print (
    FILE * out = stdout ) const
```

Pretty-print the viewed matrix to file stream.

## 7.11.4 Member Data Documentation

### 7.11.4.1 gmat\_view

```
gsl\_matrix\_view gsl::matrix_view::gmat_view [protected]
```

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

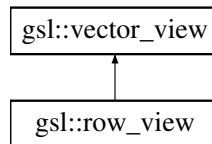
- [/home/jspainhour/spheroidal\\_cpp/include/yawg/matrix.h](#)

## 7.12 gsl::row\_view Class Reference

A subclass of [vector\\_view](#) for stride-1 vectors.

```
#include <vector.h>
```

Inheritance diagram for `gsl::row_view`:



### Public Member Functions

- [row\\_view](#) ([gsl\\_vector\\_view](#) [gvec\\_view](#))  
Construct [row\\_view](#) from existing vector view, checking that stride is 1.
- [row\\_view](#) (const [vector](#) &v)  
Construct [row\\_view](#) from vector, checking that stride is 1.
- [matrix\\_view](#) [reshape](#) (size\_t n, size\_t m)  
Return a matrix view out of the elements of the row.

### Additional Inherited Members

#### 7.12.1 Detailed Description

A subclass of [vector\\_view](#) for stride-1 vectors.

#### 7.12.2 Constructor & Destructor Documentation

##### 7.12.2.1 row\_view() [1/2]

```
gsl::row_view::row_view (
    gsl_vector_view gvec_view ) [inline]
```

Construct [row\\_view](#) from existing vector view, checking that stride is 1.

##### 7.12.2.2 row\_view() [2/2]

```
gsl::row_view::row_view (
    const vector & v ) [inline]
```

Construct [row\\_view](#) from vector, checking that stride is 1.

## 7.12.3 Member Function Documentation

### 7.12.3.1 reshape()

```
matrix_view gsl::row_view::reshape (
    size_t n,
    size_t m )
```

Return a matrix view out of the elements of the row.

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

- /home/jspainhour/spheroidal\_cpp/include/yawg/vector.h

## 7.13 gsl::vector Class Reference

A wrapper class for gsl\_vector.

```
#include <vector.h>
```

### Public Member Functions

- [vector](#) ()  
*Construct empty vector.*
- [vector](#) (size\_t n)  
*Construct zero vector of size n.*
- [vector](#) (const gsl\_vector \*gvec\_other)  
*Construct new [gsl::vector](#) from [gsl\\_vector](#).*
- [vector](#) (const [vector](#) &gvec\_other)
- [vector](#) ([vector](#) &&gvec\_other)
- [vector](#) & [operator=](#) (const [vector](#) &gvec\_other)
- [vector](#) & [operator=](#) ([vector](#) &&gvec\_other)
- [~vector](#) ()
- double & [operator\(\)](#) (size\_t i)
- void [set](#) (size\_t i, double val)
- double [operator\(\)](#) (size\_t i) const
- double [get](#) (size\_t i) const
- size\_t [size](#) () const
- gsl\_vector \* [get\\_gsl\\_ptr](#) () const  
*Access the pointer to the underlying [gsl\\_vector](#).*
- void [resize](#) (size\_t n)  
*Resize the [gsl::vector](#), setting elements to zero.*
- void [clear](#) ()  
*Clear the [gsl::vector](#), free underlying memory.*
- void [print](#) (FILE \*out=stdout) const  
*Pretty-print the vector to file stream.*
- double [norm](#) () const  
*Return the 2-norm of the vector.*
- [vector](#) & [operator+=](#) (const [vector](#) &gvec\_other)
- [vector](#) & [operator-=](#) (const [vector](#) &gvec\_other)
- [vector\\_view](#) [subvector](#) (size\_t offset, size\_t [size](#))  
*Return a view to a subvector of the vector.*
- [vector\\_view](#) [view](#) ()  
*Return a view to the entire vector.*

## Protected Member Functions

- void `free` ()  
*Private function to free allocated memory.*
- void `calloc` (size\_t n)  
*Private function to (continuously) allocate memory.*

## Protected Attributes

- gsl\_vector \* `gvec`

## Friends

- class `cvector`
- class `vector_view`
- class `row_view`
- class `column_view`
- class `matrix`
- `vector operator*` (double a, const `vector` &v)
- `vector operator*` (double a, `vector` &&v)
- `vector operator*` (const `vector` &v, double a)
- `vector operator*` (`vector` &&v, double a)
- `cvector operator*` (complex a, const `vector` &v)
- `cvector operator*` (const `vector` &v, complex a)
- `vector operator+` (const `vector` &v1, const `vector` &v2)
- `vector operator+` (`vector` &&v1, const `vector` &v2)
- `vector operator+` (const `vector` &v1, `vector` &&v2)
- `vector operator+` (`vector` &&v1, `vector` &&v2)
- `vector operator-` (const `vector` &v1, const `vector` &v2)
- `vector operator-` (`vector` &&v1, const `vector` &v2)
- `vector operator-` (const `vector` &v1, `vector` &&v2)
- `vector operator-` (`vector` &&v1, `vector` &&v2)
- bool `operator==` (const `vector` &v1, const `vector` &v2)

### 7.13.1 Detailed Description

A wrapper class for `gsl_vector`.

Stores and operates on a pointer to a `gsl_vector`.

### 7.13.2 Constructor & Destructor Documentation

#### 7.13.2.1 `vector()` [1/5]

```
gsl::vector::vector ( )
```

Construct empty vector.

### 7.13.2.2 vector() [2/5]

```
gsl::vector::vector (
    size_t n ) [explicit]
```

Construct zero vector of size n.

### 7.13.2.3 vector() [3/5]

```
gsl::vector::vector (
    const gsl_vector * gvec_other )
```

Construct new [gsl::vector](#) from `gsl_vector`.

### 7.13.2.4 vector() [4/5]

```
gsl::vector::vector (
    const vector & gvec_other )
```

### 7.13.2.5 vector() [5/5]

```
gsl::vector::vector (
    vector && gvec_other )
```

### 7.13.2.6 ~vector()

```
gsl::vector::~~vector ( )
```

## 7.13.3 Member Function Documentation

### 7.13.3.1 calloc()

```
void gsl::vector::calloc (
    size_t n ) [protected]
```

Private function to (continuously) allocate memory.

#### 7.13.3.2 clear()

```
void gsl::vector::clear ( )
```

Clear the [gsl::vector](#), free underlying memory.

#### 7.13.3.3 free()

```
void gsl::vector::free ( ) [protected]
```

Private function to free allocated memory.

#### 7.13.3.4 get()

```
double gsl::vector::get (
    size_t i ) const
```

#### 7.13.3.5 get\_gsl\_ptr()

```
gsl_vector* gsl::vector::get_gsl_ptr ( ) const [inline]
```

Access the pointer to the underlying `gsl_vector`.

#### 7.13.3.6 norm()

```
double gsl::vector::norm ( ) const [inline]
```

Return the 2-norm of the vector.

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

```
double& gsl::vector::operator() (
    size_t i )
```



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

```
double gsl::vector::operator() (
    size_t i ) const
```

#### 7.13.3.9 operator+=()

```
vector& gsl::vector::operator+= (
    const vector & gvec_other )
```

#### 7.13.3.10 operator-=()

```
vector& gsl::vector::operator-= (
    const vector & gvec_other )
```

#### 7.13.3.11 operator=() [1/2]

```
vector& gsl::vector::operator= (
    const vector & gvec_other )
```

#### 7.13.3.12 operator=() [2/2]

```
vector& gsl::vector::operator= (
    vector && gvec_other )
```

#### 7.13.3.13 print()

```
void gsl::vector::print (
    FILE * out = stdout ) const
```

Pretty-print the vector to file stream.

#### 7.13.3.14 `resize()`

```
void gsl::vector::resize (
    size_t n )
```

Resize the [gsl::vector](#), setting elements to zero.

#### 7.13.3.15 `set()`

```
void gsl::vector::set (
    size_t i,
    double val )
```

#### 7.13.3.16 `size()`

```
size_t gsl::vector::size ( ) const
```

#### 7.13.3.17 `subvector()`

```
vector_view gsl::vector::subvector (
    size_t offset,
    size_t size )
```

Return a view to a subvector of the vector.

#### 7.13.3.18 `view()`

```
vector_view gsl::vector::view ( )
```

Return a view to the entire vector.

### 7.13.4 Friends And Related Function Documentation

#### 7.13.4.1 `column_view`

```
friend class column_view [friend]
```

#### 7.13.4.2 cvector

```
friend class cvector [friend]
```

#### 7.13.4.3 matrix

```
friend class matrix [friend]
```

#### 7.13.4.4 operator\* [1/6]

```
cvector operator* (  
    complex a,  
    const vector & v ) [friend]
```

#### 7.13.4.5 operator\* [2/6]

```
cvector operator* (  
    const vector & v,  
    complex a ) [friend]
```

#### 7.13.4.6 operator\* [3/6]

```
vector operator* (  
    const vector & v,  
    double a ) [friend]
```

#### 7.13.4.7 operator\* [4/6]

```
vector operator* (  
    double a,  
    const vector & v ) [friend]
```

**7.13.4.8 operator\* [5/6]**

```
vector operator* (
    double a,
    vector && v ) [friend]
```

**7.13.4.9 operator\* [6/6]**

```
vector operator* (
    vector && v,
    double a ) [friend]
```

**7.13.4.10 operator+ [1/4]**

```
vector operator+ (
    const vector & v1,
    const vector & v2 ) [friend]
```

**7.13.4.11 operator+ [2/4]**

```
vector operator+ (
    const vector & v1,
    vector && v2 ) [friend]
```

**7.13.4.12 operator+ [3/4]**

```
vector operator+ (
    vector && v1,
    const vector & v2 ) [friend]
```

**7.13.4.13 operator+ [4/4]**

```
vector operator+ (
    vector && v1,
    vector && v2 ) [friend]
```

**7.13.4.14 operator- [1/4]**

```
vector operator- (
    const vector & v1,
    const vector & v2 ) [friend]
```

**7.13.4.15 operator- [2/4]**

```
vector operator- (
    const vector & v1,
    vector && v2 ) [friend]
```

**7.13.4.16 operator- [3/4]**

```
vector operator- (
    vector && v1,
    const vector & v2 ) [friend]
```

**7.13.4.17 operator- [4/4]**

```
vector operator- (
    vector && v1,
    vector && v2 ) [friend]
```

**7.13.4.18 operator==**

```
bool operator== (
    const vector & v1,
    const vector & v2 ) [friend]
```

**7.13.4.19 row\_view**

```
friend class row_view [friend]
```

**7.13.4.20 vector\_view**

```
friend class vector_view [friend]
```

## 7.13.5 Member Data Documentation

### 7.13.5.1 gvec

```
gsl_vector* gsl::vector::gvec [protected]
```

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

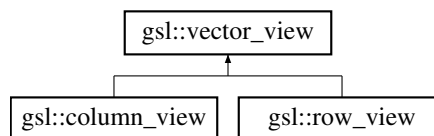
- /home/jspainhour/spheroidal\_cpp/include/yawg/[vector.h](#)

## 7.14 gsl::vector\_view Class Reference

A wrapper class for `gsl_vector_view`.

```
#include <vector.h>
```

Inheritance diagram for `gsl::vector_view`:



### Public Member Functions

- [operator vector](#) () const  
*"Dereferences" a [vector\\_view](#) into independent [gsl::vector](#) object*
- [vector\\_view](#) ([gsl\\_vector\\_view](#) [gvec\\_view](#))  
*Construct a view of a [gsl::vector](#) through another [vector\\_view](#).*
- [vector\\_view](#) (const [vector](#) &*v*)  
*Construct a view of the given [gsl::vector](#).*
- [vector\\_view](#) & [operator=](#) (const [vector](#) &*v*)  
*Assignment to a vector view from a vector.*
- [vector\\_view](#) & [operator=](#) ([vector\\_view](#) *v*)  
*Assignment to a vector view from a vector view.*
- void [print](#) (FILE \**out=stdout*) const  
*Pretty-print the viewed vector to file stream.*
- const [gsl\\_vector](#) \* [get\\_gsl\\_ptr](#) () const  
*Return a constant pointer to the underlying [gsl\\_vector](#).*

### Protected Attributes

- [gsl\\_vector\\_view](#) [gvec\\_view](#)

### 7.14.1 Detailed Description

A wrapper class for `gsl_vector_view`.

Stores a `gsl_vector_view` and uses it to access original member data.

### 7.14.2 Constructor & Destructor Documentation

#### 7.14.2.1 vector\_view() [1/2]

```
gsl::vector_view::vector_view (  
    gsl_vector_view gvec_view ) [inline]
```

Construct a view of a [gsl::vector](#) through another [vector\\_view](#).

#### 7.14.2.2 vector\_view() [2/2]

```
gsl::vector_view::vector_view (  
    const vector & v ) [inline]
```

Construct a view of the given [gsl::vector](#).

### 7.14.3 Member Function Documentation

#### 7.14.3.1 get\_gsl\_ptr()

```
const gsl_vector* gsl::vector_view::get_gsl_ptr ( ) const [inline]
```

Return a constant pointer to the underlying `gsl_vector`.

#### 7.14.3.2 operator vector()

```
gsl::vector_view::operator vector ( ) const [inline]
```

"Dereferences" a [vector\\_view](#) into independent [gsl::vector](#) object

#### 7.14.3.3 operator=() [1/2]

```
vector_view& gsl::vector_view::operator= (
    const vector & v )
```

Assignment to a vector view from a vector.

#### 7.14.3.4 operator=() [2/2]

```
vector_view& gsl::vector_view::operator= (
    vector_view v )
```

Assignment to a vector view from a vector view.

#### 7.14.3.5 print()

```
void gsl::vector_view::print (
    FILE * out = stdout ) const
```

Pretty-print the viewed vector to file stream.

### 7.14.4 Member Data Documentation

#### 7.14.4.1 gvec\_view

```
gsl_vector_view gsl::vector_view::gvec_view [protected]
```

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

- [/home/jspainhour/spheroidal\\_cpp/include/yawg/vector.h](/home/jspainhour/spheroidal_cpp/include/yawg/vector.h)



## Chapter 8

# File Documentation

### 8.1 grid\_functions.h File Reference

```
#include <yawg/matrix.h>
```

#### Functions

- int [spharm\\_grid\\_size\\_ord](#) (int p, int &nu, int &nv)  
*Computes grid size of spheroidal harmonics grid given the order.*
- int [spharm\\_grid\\_size\\_tot](#) (int ntot, int &nu, int &nv)  
*Computes grid size of spheroidal harmonics grid given the total number of points.*
- void [gl\\_grid](#) (size\_t p, [gsl::matrix](#) &U, [gsl::matrix](#) &V)  
*Populates the matrices  $U$  and  $V$  with a spheroidal harmonics grid of order  $p$ .*

#### 8.1.1 Function Documentation

##### 8.1.1.1 gl\_grid()

```
void gl_grid (  
    size_t p,  
    gsl::matrix & U,  
    gsl::matrix & V )
```

Populates the matrices  $U$  and  $V$  with a spheroidal harmonics grid of order  $p$ .

### 8.1.1.2 spharm\_grid\_size\_ord()

```
int spharm_grid_size_ord (
    int p,
    int & nu,
    int & nv )
```

Computes grid size of spheroidal harmonics grid given the order.

### 8.1.1.3 spharm\_grid\_size\_tot()

```
int spharm_grid_size_tot (
    int ntot,
    int & nu,
    int & nv )
```

Computes grid size of spheroidal harmonics grid given the total number of points.

## 8.2 legendre\_otc.h File Reference

```
#include <yawg/vector.h>
#include <yawg/matrix.h>
```

### Functions

- [gsl::vector cont\\_frac](#) (int n, int m, [gsl::vector](#) u)
- int [geti](#) (int n, int m)
- void [legendre\\_otc](#) (int p, [gsl::vector](#) u, [gsl::matrix](#) &P)
- void [legendre\\_otc](#) (int p, [gsl::vector](#) u, [gsl::matrix](#) &P, [gsl::matrix](#) &Q)
- void [Dlegendre\\_otc](#) (int p, [gsl::vector](#) u, [gsl::matrix](#) &P, [gsl::matrix](#) &dP)
- void [Dlegendre\\_otc](#) (int p, [gsl::vector](#) u, [gsl::matrix](#) &P, [gsl::matrix](#) &Q, [gsl::matrix](#) &dP, [gsl::matrix](#) &dQ)

### 8.2.1 Function Documentation

#### 8.2.1.1 cont\_frac()

```
gsl::vector cont_frac (
    int n,
    int m,
    gsl::vector u )
```

### 8.2.1.2 Dlegendre\_otc() [1/2]

```
void Dlegendre_otc (
    int p,
    gsl::vector u,
    gsl::matrix & P,
    gsl::matrix & dP )
```

### 8.2.1.3 Dlegendre\_otc() [2/2]

```
void Dlegendre_otc (
    int p,
    gsl::vector u,
    gsl::matrix & P,
    gsl::matrix & Q,
    gsl::matrix & dP,
    gsl::matrix & dQ )
```

### 8.2.1.4 geti()

```
int geti (
    int n,
    int m )
```

### 8.2.1.5 legendre\_otc() [1/2]

```
void legendre_otc (
    int p,
    gsl::vector u,
    gsl::matrix & P )
```

### 8.2.1.6 legendre\_otc() [2/2]

```
void legendre_otc (
    int p,
    gsl::vector u,
    gsl::matrix & P,
    gsl::matrix & Q )
```

## 8.3 spheroidal\_analysis.h File Reference

```
#include <vector>
#include <gsl_wrapper/core.h>
```

### Functions

- [gsl::vector spheroidal\\_analysis \(gsl::vector f\)](#)

### 8.3.1 Function Documentation

#### 8.3.1.1 spheroidal\_analysis()

```
gsl::vector spheroidal_analysis (
    gsl::vector f )
```

## 8.4 /home/jspainhour/spheroidal\_cpp/include/yawg/cmatrix.h File Reference

```
#include <yawg/complex.h>
#include <gsl/gsl_matrix.h>
#include <stdio.h>
```

### Classes

- class [gsl::cmatrix](#)
- class [gsl::cmatrix\\_view](#)  
*A wrapper class for gsl\_matrix\_complex\_view.*

### Namespaces

- [gsl](#)

## 8.5 /home/jspainhour/spheroidal\_cpp/include/yawg/complex.h File Reference

```
#include <gsl/gsl_complex.h>
#include <gsl/gsl_complex_math.h>
```

## Classes

- class [gsl::complex](#)  
*Wrapper class for `gsl_complex` structs.*
- class [gsl::complex\\_ref](#)  
*Stores a reference to a [gsl::complex](#) object.*

## Namespaces

- [gsl](#)
- [gsl::complex\\_literals](#)

## Functions

- complex [gsl::complex\\_literals::operator""\\_i](#) (long double y)  
*User defined literal overload for complex numbers.*

## 8.6 /home/jspainhour/spheroidal\_cpp/include/yawg/core.h File Reference

```
#include <yawg/complex.h>
#include <yawg/vector.h>
#include <yawg/cvector.h>
#include <yawg/matrix.h>
#include <yawg/cmatrix.h>
```

## 8.7 /home/jspainhour/spheroidal\_cpp/include/yawg/cvector.h File Reference

```
#include <yawg/complex.h>
#include <gsl/gsl_vector_complex.h>
#include <gsl/gsl_blas.h>
#include <stdio.h>
```

## Classes

- class [gsl::cvector](#)  
*A wrapper class for `gsl_vector_complex`.*
- class [gsl::cvector\\_view](#)  
*A wrapper class for `gsl_vector_complex_view`.*
- class [gsl::crow\\_view](#)  
*A subclass of [cvector\\_view](#) for stride-1 vectors.*
- class [gsl::ccolumn\\_view](#)  
*A subclass of [cvector\\_view](#) for non-stride-1 cvecs.*

## Namespaces

- [gsl](#)

## 8.8 /home/jspainhour/spheroidal\_cpp/include/yawg/fft.h File Reference

```
#include <yawg/core.h>
```

## Namespaces

- [gsl](#)

## Functions

- [gsl::cvector gsl::fft](#) ([gsl::cvector](#) &&x)  
*Compute in-place fft of a [gsl::\(c\)vector](#).*
- [gsl::cvector gsl::fft](#) (const [gsl::cvector](#) &x)  
*Compute fft of a [gsl::\(c\)vector](#).*
- [gsl::cvector gsl::ifft](#) ([gsl::cvector](#) &&x)  
*Compute in-place inverse fft of a [gsl::\(c\)vector](#).*
- [gsl::cvector gsl::ifft](#) (const [gsl::cvector](#) &x)  
*Compute inverse fft of a [gsl::\(c\)vector](#).*
- [gsl::cmatrix gsl::fft](#) ([gsl::cmatrix](#) &&x, int dim=1)  
*Compute in-place 1D fft of each column/row of a [gsl::\(c\)matrix](#).*
- [gsl::cmatrix gsl::fft](#) (const [gsl::cmatrix](#) &x, int dim=1)  
*Compute 1D fft of each column/row of a [gsl::\(c\)matrix](#).*
- [gsl::cmatrix gsl::ifft](#) ([gsl::cmatrix](#) &&x, int dim=1)  
*Compute in-place 1D inverse fft of each column/row of a [gsl::\(c\)matrix](#).*
- [gsl::cmatrix gsl::ifft](#) (const [gsl::cmatrix](#) &x, int dim=1)  
*Compute 1D inverse fft of each column/row of a [gsl::\(c\)matrix](#).*

## 8.9 /home/jspainhour/spheroidal\_cpp/include/yawg/matrix.h File Reference

```
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_blas.h>
#include <stdio.h>
```

## Classes

- class [gsl::matrix](#)  
*A wrapper class for [gsl\\_matrix](#).*
- class [gsl::matrix\\_view](#)  
*A wrapper class for [gsl\\_matrix\\_view](#).*

## Namespaces

- [gsl](#)

## 8.10 /home/jspainhour/spheroidal\_cpp/include/yawg/utils.hpp File Reference

```
#include <yawg/core.h>
#include <utility>
```

## Namespaces

- [gsl](#)

## Functions

- void [gsl::leggauss](#) (size\_t n, [gsl::vector](#) &x, [gsl::vector](#) &w, double a=-1.0, double b=1.0)  
*Store Gauss-Legendre quadrature nodes and weights.*
- vector [gsl::leggauss](#) (size\_t n, double a=-1.0, double b=1.0)  
*Get a vector Gauss-Legendre quadrature nodes.*
- [gsl::vector](#) [gsl::linspace](#) (double a, double b, size\_t N=100)  
*Get a [gsl::vector](#) of evenly spaced points on the interval [a, b] (inclusive)*
- [gsl::cvector](#) [gsl::linspace](#) ([gsl::complex](#) a, [gsl::complex](#) b, size\_t n)  
*Complex version of linspace.*
- [gsl::vector](#) [gsl::arange](#) (double a, double b, double step=1.0)
- void [gsl::meshgrid](#) (const [gsl::vector](#) &x, const [gsl::vector](#) &y, [gsl::matrix](#) &X, [gsl::matrix](#) &Y)  
*Store 2D grid coordinates based on 1D input [gsl::vectors](#).*
- void [gsl::meshgrid](#) (const [gsl::cvector](#) &x, const [gsl::cvector](#) &y, [gsl::cmatrix](#) &X, [gsl::cmatrix](#) &Y)
- [gsl::matrix](#) [gsl::eye](#) (size\_t n)  
*Return the nxn identity matrix.*
- template<typename Lambda >  
[gsl::vector](#) [gsl::arrayfun](#) (Lambda &&func, const [gsl::vector](#) &x)  
*Apply lambda function to each element of a [gsl::vector](#), akin to MATLAB arrayfun.*
- template<typename Lambda >  
[gsl::vector](#) [gsl::arrayfun](#) (Lambda &&func, [gsl::vector](#) &&x)  
*Move version of arrayfun for vectors.*
- template<typename Lambda >  
[gsl::cvector](#) [gsl::arrayfun](#) (Lambda &&func, const [gsl::cvector](#) &x)  
*Copy version of arrayfun for complex vectors.*
- template<typename Lambda >  
[gsl::cvector](#) [gsl::arrayfun](#) (Lambda &&func, [gsl::cvector](#) &&x)  
*Move version of arrayfun for complex vectors.*
- template<typename Lambda >  
[gsl::matrix](#) [gsl::arrayfun](#) (Lambda &&func, const [gsl::matrix](#) &x)  
*Apply lambda function to each element of a [gsl::matrix](#), akin to MATLAB arrayfun.*
- template<typename Lambda >  
[gsl::matrix](#) [gsl::arrayfun](#) (Lambda &&func, [gsl::matrix](#) &&x)  
*Move version of arrayfun.*

- `template<typename Lambda >`  
`gsl::cmatrix gsl::arrayfun` (Lambda &&func, const `gsl::cmatrix` &x)  
*Copy version of arrayfun for complex matrices.*
- `template<typename Lambda >`  
`gsl::cmatrix gsl::arrayfun` (Lambda &&func, `gsl::cmatrix` &&x)  
*Move version of arrayfun for complex matrices.*

## 8.11 /home/jspainhour/spheroidal\_cpp/include/yawg/vector.h File Reference

```
#include <gsl/gsl_vector.h>
#include <gsl/gsl_blas.h>
#include <stdio.h>
```

### Classes

- class `gsl::vector`  
*A wrapper class for `gsl_vector`.*
- class `gsl::vector_view`  
*A wrapper class for `gsl_vector_view`.*
- class `gsl::row_view`  
*A subclass of `vector_view` for stride-1 vectors.*
- class `gsl::column_view`  
*A subclass of `vector_view` for non-stride-1 vectors.*

### Namespaces

- `gsl`

## 8.12 /home/jspainhour/spheroidal\_cpp/README.md File Reference

## 8.13 /home/jspainhour/spheroidal\_cpp/tests/yawg/test\_fft.cpp File Reference

```
#include <yawg/core.h>
#include <yawg/utils.hpp>
#include <yawg/fft.h>
#include <catch2/catch_test_macros.hpp>
#include <catch2/matchers/catch_matchers_floating_point.hpp>
```

### Macros

- `#define CATCH_CONFIG_MAIN`



## Functions

- [TEST\\_CASE](#) ("gsl::fft", "[\[gsl::fft\]](#)")
- [TEST\\_CASE](#) ("gsl::fft with non-power-of-2 input size", "[\[gsl::fft\]](#)")
- [TEST\\_CASE](#) ("gsl::fft on 3x4 matrix", "[\[gsl::fft\]](#)")
- [TEST\\_CASE](#) ("gsl::ifft on 3x4 matrix", "[\[gsl::ifft\]](#)")

### 8.13.1 Macro Definition Documentation

#### 8.13.1.1 CATCH\_CONFIG\_MAIN

```
#define CATCH_CONFIG_MAIN
```

### 8.13.2 Function Documentation

#### 8.13.2.1 TEST\_CASE() [1/4]

```
TEST_CASE (
    "gsl::fft on 3x4 matrix" ,
    "" \[gsl::fft\] )
```

#### 8.13.2.2 TEST\_CASE() [2/4]

```
TEST_CASE (
    "gsl::fft with non-power-of-2 input size" ,
    "" \[gsl::fft\] )
```

#### 8.13.2.3 TEST\_CASE() [3/4]

```
TEST_CASE (
    "gsl::fft" ,
    "" \[gsl::fft\] )
```

#### 8.13.2.4 TEST\_CASE() [4/4]

```
TEST_CASE (
    "gsl::ifft on 3x4 matrix" ,
    "" [gsl::ifft] )
```

### 8.14 /home/jspainhour/spheroidal\_cpp/tests/yawg/test\_utils.cpp File Reference

```
#include <yawg/core.h>
#include <yawg/utils.hpp>
#include <catch2/catch_test_macros.hpp>
#include <catch2/matchers/catch_matchers_floating_point.hpp>
#include <gsl/gsl_math.h>
#include <cmath>
```

#### Macros

- `#define` [CATCH\\_CONFIG\\_MAIN](#)

#### Functions

- [TEST\\_CASE](#) ("linspace", "[linspace]")
- [TEST\\_CASE](#) ("arange", "[arange]")
- [TEST\\_CASE](#) ("leggauss", "[leggauss]")
- [TEST\\_CASE](#) ("leggauss return", "[leggauss]")
- [TEST\\_CASE](#) ("meshgrid", "[meshgrid]")
- [TEST\\_CASE](#) ("eye", "[eye]")
- [TEST\\_CASE](#) ("arrayfun vector", "[arrayfun]")
- [TEST\\_CASE](#) ("arrayfun matrix", "[arrayfun]")

### 8.14.1 Macro Definition Documentation

#### 8.14.1.1 CATCH\_CONFIG\_MAIN

```
#define CATCH_CONFIG_MAIN
```

### 8.14.2 Function Documentation

#### 8.14.2.1 TEST\_CASE() [1/8]

```
TEST_CASE (
    "arange" ,
    "" [arange] )
```

#### 8.14.2.2 TEST\_CASE() [2/8]

```
TEST_CASE (
    "arrayfun matrix" ,
    "" [arrayfun] )
```

#### 8.14.2.3 TEST\_CASE() [3/8]

```
TEST_CASE (
    "arrayfun vector" ,
    "" [arrayfun] )
```

#### 8.14.2.4 TEST\_CASE() [4/8]

```
TEST_CASE (
    "eye" ,
    "" [eye] )
```

#### 8.14.2.5 TEST\_CASE() [5/8]

```
TEST_CASE (
    "leggauss return" ,
    "" [leggauss] )
```

#### 8.14.2.6 TEST\_CASE() [6/8]

```
TEST_CASE (
    "leggauss" ,
    "" [leggauss] )
```

#### 8.14.2.7 TEST\_CASE() [7/8]

```
TEST_CASE (
    "linspace" ,
    "" [linspace] )
```

#### 8.14.2.8 TEST\_CASE() [8/8]

```
TEST_CASE (
    "meshgrid" ,
    "" [meshgrid] )
```

### 8.15 /home/jspainhour/spheroidal\_cpp/tests/yawg/test\_wrapper.cpp File Reference

```
#include <yawg/core.h>
#include <catch2/catch_test_macros.hpp>
```

#### Macros

- `#define CATCH_CONFIG_MAIN`

#### Functions

- `TEST_CASE` ("gsl::vector constructors", "[gsl::vector]")
- `TEST_CASE` ("gsl::complex constructors", "[gsl::complex]")
- `TEST_CASE` ("gsl::cvector constructors", "[gsl::cvector]")
- `TEST_CASE` ("gsl::vector -> gsl::cvector conversion", "[gsl::vector][gsl::cvector]")
- `TEST_CASE` ("gsl::matrix -> gsl::cmatrix conversion", "[gsl::matrix][gsl::cmatrix]")
- `TEST_CASE` ("gsl::vector assignment operators", "[gsl::vector]")
- `TEST_CASE` ("gsl::cvector assignment operators", "[gsl::cvector]")
- `TEST_CASE` ("gsl::vector resize", "[gsl::vector]")
- `TEST_CASE` ("gsl::cvector resize", "[gsl::cvector]")
- `TEST_CASE` ("gsl::vector element access", "[gsl::vector]")
- `TEST_CASE` ("gsl::cvector element access", "[gsl::cvector]")
- `TEST_CASE` ("gsl::vector print", "[gsl::vector][gsl::cvector][gsl::matrix][gsl::cmatrix]")
- `TEST_CASE` ("gsl::matrix constructors", "[gsl::matrix]")
- `TEST_CASE` ("gsl::cmatrix constructors", "[gsl::cmatrix]")
- `TEST_CASE` ("gsl::matrix assignment", "[gsl::matrix]")
- `TEST_CASE` ("gsl::cmatrix assignment operators", "[gsl::cmatrix]")
- `TEST_CASE` ("gsl::matrix element access", "[gsl::matrix]")
- `TEST_CASE` ("gsl::cmatrix element access", "[gsl::cmatrix]")
- `TEST_CASE` ("gsl::matrix resize", "[gsl::matrix]")
- `TEST_CASE` ("gsl::cmatrix resize", "[gsl::cmatrix]")

## 8.15.1 Macro Definition Documentation

### 8.15.1.1 CATCH\_CONFIG\_MAIN

```
#define CATCH_CONFIG_MAIN
```

## 8.15.2 Function Documentation

### 8.15.2.1 TEST\_CASE() [1/20]

```
TEST_CASE (
    "gsl::cmatrix assignment operators" ,
    "" [gsl::cmatrix] )
```

### 8.15.2.2 TEST\_CASE() [2/20]

```
TEST_CASE (
    "gsl::cmatrix constructors" ,
    "" [gsl::cmatrix] )
```

### 8.15.2.3 TEST\_CASE() [3/20]

```
TEST_CASE (
    "gsl::cmatrix element access" ,
    "" [gsl::cmatrix] )
```

### 8.15.2.4 TEST\_CASE() [4/20]

```
TEST_CASE (
    "gsl::cmatrix resize" ,
    "" [gsl::cmatrix] )
```

#### 8.15.2.5 TEST\_CASE() [5/20]

```
TEST_CASE (
    "gsl::complex constructors" ,
    "" [gsl::complex] )
```

#### 8.15.2.6 TEST\_CASE() [6/20]

```
TEST_CASE (
    "gsl::cvector assignment operators" ,
    "" [gsl::cvector] )
```

#### 8.15.2.7 TEST\_CASE() [7/20]

```
TEST_CASE (
    "gsl::cvector constructors" ,
    "" [gsl::cvector] )
```

#### 8.15.2.8 TEST\_CASE() [8/20]

```
TEST_CASE (
    "gsl::cvector element access" ,
    "" [gsl::cvector] )
```

#### 8.15.2.9 TEST\_CASE() [9/20]

```
TEST_CASE (
    "gsl::cvector resize" ,
    "" [gsl::cvector] )
```

#### 8.15.2.10 TEST\_CASE() [10/20]

```
TEST_CASE (
    "gsl::matrix -> gsl::cmatrix conversion" ,
    "" [gsl::matrix][gsl::cmatrix] )
```

**8.15.2.11 TEST\_CASE()** [11/20]

```
TEST_CASE (
    "gsl::matrix assignment" ,
    "" [gsl::matrix] )
```

**8.15.2.12 TEST\_CASE()** [12/20]

```
TEST_CASE (
    "gsl::matrix constructors" ,
    "" [gsl::matrix] )
```

**8.15.2.13 TEST\_CASE()** [13/20]

```
TEST_CASE (
    "gsl::matrix element access" ,
    "" [gsl::matrix] )
```

**8.15.2.14 TEST\_CASE()** [14/20]

```
TEST_CASE (
    "gsl::matrix resize" ,
    "" [gsl::matrix] )
```

**8.15.2.15 TEST\_CASE()** [15/20]

```
TEST_CASE (
    "gsl::vector -> gsl::cvector conversion" ,
    "" [gsl::vector][gsl::cvector] )
```

**8.15.2.16 TEST\_CASE()** [16/20]

```
TEST_CASE (
    "gsl::vector assignment operators" ,
    "" [gsl::vector] )
```

**8.15.2.17 TEST\_CASE()** [17/20]

```
TEST_CASE (
    "gsl::vector constructors" ,
    "" [gsl::vector] )
```

**8.15.2.18 TEST\_CASE()** [18/20]

```
TEST_CASE (
    "gsl::vector element access" ,
    "" [gsl::vector] )
```

**8.15.2.19 TEST\_CASE()** [19/20]

```
TEST_CASE (
    "gsl::vector print" ,
    "" [gsl::vector][gsl::ector][gsl::matrix][gsl::cmatrix] )
```

**8.15.2.20 TEST\_CASE()** [20/20]

```
TEST_CASE (
    "gsl::vector resize" ,
    "" [gsl::vector] )
```

## 8.16 /home/jspainhour/spheroidal\_cpp/tests/yawg/test\_wrapper\_↵ math.cpp File Reference

```
#include <yawg/core.h>
#include <yawg/utils.hpp>
#include <catch2/catch_test_macros.hpp>
#include <catch2/matchers/catch_matchers_floating_point.hpp>
#include <cmath>
```

**Macros**

- #define [CATCH\\_CONFIG\\_MAIN](#)



## Functions

- `TEST_CASE` ("gsl::complex methods", "[gsl::complex]")
- `TEST_CASE` ("gsl::complex assignment operators", "[gsl::complex]")
- `TEST_CASE` ("gsl::complex operators", "[gsl::complex]")
- `TEST_CASE` ("gsl::vector addition", "[gsl::vector]")  
*Use Catch2 to test addition of gsl::vectors.*
- `TEST_CASE` ("gsl::vector and gsl::cvector addition", "[gsl::vector][gsl::cvector]")  
*Use Catch2 to test addition of gsl::vectors and gsl::cvectors.*
- `TEST_CASE` ("gsl::vector and gsl::cvector scalar multiplication", "[gsl::vector][gsl::cvector]")  
*Use Catch2 to test scalar multiplication of gsl::vectors and gsl::cvectors.*

## 8.16.1 Macro Definition Documentation

### 8.16.1.1 CATCH\_CONFIG\_MAIN

```
#define CATCH_CONFIG_MAIN
```

## 8.16.2 Function Documentation

### 8.16.2.1 TEST\_CASE() [1/6]

```
TEST_CASE (
    "gsl::complex assignment operators" ,
    " [gsl::complex] )
```

### 8.16.2.2 TEST\_CASE() [2/6]

```
TEST_CASE (
    "gsl::complex methods" ,
    " [gsl::complex] )
```

### 8.16.2.3 TEST\_CASE() [3/6]

```
TEST_CASE (
    "gsl::complex operators" ,
    " [gsl::complex] )
```

#### 8.16.2.4 TEST\_CASE() [4/6]

```
TEST_CASE (
    "gsl::vector addition" ,
    "" [gsl::vector] )
```

Use Catch2 to test addition of `gsl::vectors`.

#### 8.16.2.5 TEST\_CASE() [5/6]

```
TEST_CASE (
    "gsl::vector and gsl::cvector addition" ,
    "" [gsl::vector][gsl::cvector] )
```

Use Catch2 to test addition of `gsl::vectors` and `gsl::cvector`s.

#### 8.16.2.6 TEST\_CASE() [6/6]

```
TEST_CASE (
    "gsl::vector and gsl::cvector scalar multiplication" ,
    "" [gsl::vector][gsl::cvector] )
```

Use Catch2 to test scalar multiplication of `gsl::vectors` and `gsl::cvector`s.

## 8.17 /home/jspainhour/spheroidal\_cpp/tests/yawg/test\_wrapper\_↵ view.cpp File Reference

```
#include <yawg/core.h>
#include <catch2/catch_test_macros.hpp>
```

### Macros

- #define `CATCH_CONFIG_MAIN`

### Functions

- `TEST_CASE` ("gsl::vector view methods", "[gsl::vector\_view]")
- `TEST_CASE` ("gsl::cvector view methods", "[gsl::cvector\_view]")
- `TEST_CASE` ("gsl::matrix view methods", "[gsl::matrix\_view]")
- `TEST_CASE` ("gsl::cmatrix view methods", "[gsl::cmatrix\_view]")
- `TEST_CASE` ("gsl::matrix column and row view methods", "[gsl::matrix][gsl::row\_view][gsl::column\_view]")
- `TEST_CASE` ("gsl::cmatrix column and row view methods", "[gsl::cmatrix][gsl::crow\_view][gsl::ccolumn\_view]")
- `TEST_CASE` ("gsl::vector view edge cases", "[gsl::vector][gsl::vector\_view]")
- `TEST_CASE` ("gsl::cvector view edge cases", "[gsl::cvector][gsl::cvector\_view]")
- `TEST_CASE` ("gsl::matrix view edge cases", "[gsl::matrix][gsl::matrix\_view]")
- `TEST_CASE` ("gsl::cmatrix view edge cases", "[gsl::cmatrix][gsl::cmatrix\_view]")

## 8.17.1 Macro Definition Documentation

### 8.17.1.1 CATCH\_CONFIG\_MAIN

```
#define CATCH_CONFIG_MAIN
```

## 8.17.2 Function Documentation

### 8.17.2.1 TEST\_CASE() [1/10]

```
TEST_CASE (
    "gsl::cmatrix column and row view methods" ,
    "" [gsl::cmatrix][gsl::crow_view][gsl::ccolumn_view] )
```

### 8.17.2.2 TEST\_CASE() [2/10]

```
TEST_CASE (
    "gsl::cmatrix view edge cases" ,
    "" [gsl::cmatrix][gsl::cmatrix_view] )
```

### 8.17.2.3 TEST\_CASE() [3/10]

```
TEST_CASE (
    "gsl::cmatrix view methods" ,
    "" [gsl::cmatrix_view] )
```

### 8.17.2.4 TEST\_CASE() [4/10]

```
TEST_CASE (
    "gsl::cvector view edge cases" ,
    "" [gsl::cvector][gsl::cvector_view] )
```

**8.17.2.5 TEST\_CASE()** [5/10]

```
TEST_CASE (
    "gsl::cvector view methods" ,
    "" [gsl::cvector_view] )
```

**8.17.2.6 TEST\_CASE()** [6/10]

```
TEST_CASE (
    "gsl::matrix column and row view methods" ,
    "" [gsl::matrix][gsl::row_view][gsl::column_view] )
```

**8.17.2.7 TEST\_CASE()** [7/10]

```
TEST_CASE (
    "gsl::matrix view edge cases" ,
    "" [gsl::matrix][gsl::matrix_view] )
```

**8.17.2.8 TEST\_CASE()** [8/10]

```
TEST_CASE (
    "gsl::matrix view methods" ,
    "" [gsl::matrix_view] )
```

**8.17.2.9 TEST\_CASE()** [9/10]

```
TEST_CASE (
    "gsl::vector view edge cases" ,
    "" [gsl::vector][gsl::vector_view] )
```

**8.17.2.10 TEST\_CASE()** [10/10]

```
TEST_CASE (
    "gsl::vector view methods" ,
    "" [gsl::vector_view] )
```