TRITON

v1.0

Generated by Doxygen 1.8.13

Contents

1 Bug List **1**

2 Hierarchical Index **3**

2.1 Class Hierarchy 3

3 Class Index **5**

3.1 Class List 5

4 File Index **7**

4.1 File List 7

5 Class Documentation **9**

5.1 ConfigUtils::arguments< T > Struct Template Reference 9

5.1.1 Detailed Description 10

5.1.2 Member Data Documentation 10

5.1.2.1 checkpoint_id 10

5.1.2.2 const_mann 10

5.1.2.3 courant 10

5.1.2.4 dem_filename 11

5.1.2.5 domain_decomposition 11

5.1.2.6 extbc_bctype 11

5.1.2.7 extbc_dir 11

5.1.2.8 extbc_file 11

5.1.2.9 extbc_fname 11

5.1.2.10 extbc_x1_loc 11

5.1.2.11 extbc_x2_loc 12
5.1.2.12 extbc_y1_loc 12
5.1.2.13 extbc_y2_loc 12
5.1.2.14 factor_interval_domain_decomposition 12
5.1.2.15 gpu_direct_flag 12
5.1.2.16 h_infile 12
5.1.2.17 hextra 12
5.1.2.18 hydrograph_filename 13
5.1.2.19 input_format 13
5.1.2.20 it_count 13
5.1.2.21 max_value_print_option 13
5.1.2.22 n_infile 13
5.1.2.23 num_extbc 13
5.1.2.24 num_runoffs 13
5.1.2.25 num_sources 14
5.1.2.26 observation_loc_file 14
5.1.2.27 observation_x_loc 14
5.1.2.28 observation_y_loc 14
5.1.2.29 outfile_pattern 14
5.1.2.30 output_format 14
5.1.2.31 output_option 14
5.1.2.32 print_interval 15
5.1.2.33 print_option 15
5.1.2.34 qx_infile 15
5.1.2.35 qy_infile 15
5.1.2.36 runoff_filename 15
5.1.2.37 runoff_map 15
5.1.2.38 sim_duration 15
5.1.2.39 sim_start_time 16
5.1.2.40 src_loc_file 16

5.1.2.41 src_x_loc 16
5.1.2.42 src_y_loc 16
5.1.2.43 time_increment_fixed 16
5.1.2.44 time_series_flag 16
5.1.2.45 time_step 16
5.2 SuperTimer::ci_less Struct Reference 17
5.2.1 Detailed Description 17
5.2.2 Member Function Documentation 17
5.2.2.1 operator()() 17
5.3 DemFile::dem_file< T > Class Template Reference 18
5.3.1 Detailed Description 19
5.3.2 Constructor & Destructor Documentation 19
5.3.2.1 dem_file() [1/3] 19
5.3.2.2 dem_file() [2/3] 19
5.3.2.3 dem_file() [3/3] 20
5.3.3 Member Function Documentation 20
5.3.3.1 get_cell_size() 20
5.3.3.2 get_ncols() 20
5.3.3.3 get_no_data_value() 21
5.3.3.4 get_nrows() 21
5.3.3.5 get_xll_corner() 21
5.3.3.6 get_yll_corner() 21
5.3.3.7 load_header_from_dem_file_ascii() 21
5.3.3.8 load_header_from_dem_file_binary() 22
5.3.3.9 set_cell_size() 22
5.3.3.10 set_ncols() 22
5.3.3.11 set_no_data_value() 23
5.3.3.12 set_nrows() 23
5.3.3.13 set_xll_corner() 23
5.3.3.14 set_yll_corner() 23

5.4 ExtBC::extBC< T > Class Template Reference 24
5.4.1 Detailed Description 25
5.4.2 Constructor & Destructor Documentation 25
5.4.2.1 extBC() [1/2] 25
5.4.2.2 extBC() [2/2] 25
5.4.3 Member Function Documentation 25
5.4.3.1 check_extreme_extbc() 25
5.4.3.2 convert_to_secs() 27
5.4.3.3 create_involved_cells() 27
5.4.3.4 get_num_rows() 27
5.4.3.5 get_rows() 28
5.4.3.6 get_var1_at() 28
5.4.3.7 get_var2_at() 28
5.4.3.8 load_from_file() 29
5.4.3.9 set_num_rows() 29
5.4.4 Member Data Documentation 29
5.4.4.1 extreme_cols 29
5.4.4.2 extreme_rows 30
5.4.4.3 i_cols 30
5.4.4.4 i_rows 30
5.4.4.5 location 30
5.4.4.6 ncells 30
5.4.4.7 ncells_local 30
5.5 Hydrograph::hydrograph< T > Class Template Reference 31
5.5.1 Detailed Description 31
5.5.2 Constructor & Destructor Documentation 31
5.5.2.1 hydrograph() [1/2] 32
5.5.2.2 hydrograph() [2/2] 32
5.5.3 Member Function Documentation 32
5.5.3.1 convert_rate_hr_to_secs() 32

5.5.3.2 convert_rate_mm_to_m() 32
5.5.3.3 convert_time_hr_to_secs() 32
5.5.3.4 get_flow_at() 33
5.5.3.5 get_num_inflow_rows() 33
5.5.3.6 get_num_inflows() 33
5.5.3.7 get_rows() 34
5.5.3.8 get_time_at() 34
5.5.3.9 load_from_file() 34
5.5.3.10 set_num_flow_rows() 34
5.5.3.11 set_num_sources() 35
5.6 Matrix::matrix< T > Class Template Reference 35
5.6.1 Detailed Description 37
5.6.2 Constructor & Destructor Documentation 38
5.6.2.1 matrix() [1/4] 38
5.6.2.2 matrix() [2/4] 38
5.6.2.3 matrix() [3/4] 38
5.6.2.4 matrix() [4/4] 39
5.6.2.5 ∼matrix() 39
5.6.3 Member Function Documentation 39
5.6.3.1 add_ghost_cells() 39
5.6.3.2 begin() 39
5.6.3.3 copy_elevation_into_ghost_cells() 40
5.6.3.4 copy_value_into_ghost_cells() 40
5.6.3.5 get_address_at() 40
5.6.3.6 get_data() 41
5.6.3.7 get_dims_2d() 41
5.6.3.8 get_ghost_ncols() 41
5.6.3.9 get_ghost_nrows() 42
5.6.3.10 get_num_cols() 42
5.6.3.11 get_num_rows() 42

5.6.3.12 get_value() [1/3] 42
5.6.3.13 get_value() [2/3] 43
5.6.3.14 get_value() [3/3] 43
5.6.3.15 is_inbounds() 44
5.6.3.16 load_from_ascii_file() [1/3] 44
5.6.3.17 load_from_ascii_file() [2/3] 44
5.6.3.18 load_from_ascii_file() [3/3] 45
5.6.3.19 load_from_binary_file() [1/2] 45
5.6.3.20 load_from_binary_file() [2/2] 45
5.6.3.21 operator()() [1/2] 46
5.6.3.22 operator()() [2/2] 46
5.6.3.23 operator*() [1/2] 46
5.6.3.24 operator*() [2/2] 47
5.6.3.25 operator*=() 47
5.6.3.26 operator+() [1/2] 47
5.6.3.27 operator+() [2/2] 48
5.6.3.28 operator+=() [1/2] 48
5.6.3.29 operator+=() [2/2] 48
5.6.3.30 operator=() 48
5.6.3.31 pow() 49
5.6.3.32 remove_ghost_cells() 49
5.6.3.33 resize() 49
5.6.3.34 set_infinite_walls() 49
5.6.3.35 set_size() 50
5.6.3.36 set_value() [1/3] 50
5.6.3.37 set_value() [2/3] 50
5.6.3.38 set_value() [3/3] 51
5.6.3.39 square() 51
5.6.3.40 zero_fill() 51
5.6.3.41 zero_fill_int() 51

5.7 SuperTimer::ci_less::nocase_compare Struct Reference 52
5.7.1 Detailed Description 52
5.7.2 Member Function Documentation 52
5.7.2.1 operator()() 52
5.8 Output::output< T > Class Template Reference 52
5.8.1 Detailed Description 53
5.8.2 Constructor & Destructor Documentation 54
5.8.2.1 output() 54
5.8.2.2 ∼output() 54
5.8.3 Member Function Documentation 54
5.8.3.1 average() 54
5.8.3.2 get_mat_path() 54
5.8.3.3 init() 55
5.8.3.4 init_time_series() 56
5.8.3.5 output_cfg() 56
5.8.3.6 output_time_series() 56
5.8.3.7 write_domain_decomposition() 57
5.8.3.8 write_output() 57
5.8.3.9 write_output_ascii_parallel() 58
5.8.3.10 write_output_ascii_sequential() 58
5.8.3.11 write_output_binary_parallel() 58
5.8.3.12 write_output_binary_sequential() 59
5.8.3.13 write_times() 59
5.8.4 Member Data Documentation 60
5.8.4.1 cur_proc_data_size 60
5.8.4.2 displs 60
5.8.4.3 recvcounts 60
5.8.4.4 total_data_arr 60
5.8.4.5 total_data_size 60
5.9 MpiUtils::partition_data_t Struct Reference 60

5.9.1 Detailed Description 61
5.9.2 Constructor & Destructor Documentation 61
5.9.2.1 partition_data_t() [1/4] 61
5.9.2.2 partition_data_t() [2/4] 61
5.9.2.3 partition_data_t() [3/4] 62
5.9.2.4 partition_data_t() [4/4] 62
5.9.3 Member Data Documentation 62
5.9.3.1 cols 62
5.9.3.2 cols_ini 63
5.9.3.3 part_dims 63
5.9.3.4 rows 63
5.9.3.5 rows_ini 63
5.9.3.6 size 63
5.10 SuperTimer::super_timer Class Reference 63
5.10.1 Detailed Description 64
5.10.2 Constructor & Destructor Documentation 64
5.10.2.1 super_timer() 64
5.10.3 Member Function Documentation 64
5.10.3.1 add_new_timer() 64
5.10.3.2 get_current_date() 65
5.10.3.3 get_custom_time() 65
5.10.3.4 get_hostname() 65
5.10.3.5 get_total_time() 66
5.10.3.6 reset() 66
5.10.3.7 restart() 66
5.10.3.8 start() 66
5.10.3.9 stop() 67
5.11 Triton::triton< T > Class Template Reference 67
5.11.1 Detailed Description 67
5.11.2 Constructor & Destructor Documentation 67
5.11.2.1 triton() 67
5.11.2.2 ∼triton() 68
5.11.3 Member Function Documentation 68
5.11.3.1 initialize() 68
5.11.3.2 simulate() 68

6 File Documentation 69

6.1 config_utils.h File Reference 69

6.1.1 Detailed Description 70

6.1.2 Function Documentation 70

6.1.2.1 args() 71

6.1.2.2 argsd() 71

6.1.2.3 file_content_to_string() 71

6.1.2.4 get_args() 72

6.1.2.5 get_root_dir() 72

6.1.2.6 parse_cfg() 72

6.1.2.7 parse_extbc_file() 73

6.1.2.8 parse_src_location() 73

6.1.2.9 read_and_parse_checkpoint_partition() 74

6.2 constants.h File Reference 74

6.2.1 Detailed Description 76

6.2.2 Macro Definition Documentation 77

6.2.2.1 ASCII_DIR 77

6.2.2.2 BALANCING_MPI_TIME 77

6.2.2.3 BCINDEXSTART 77

6.2.2.4 BCNROWSVARS 77

6.2.2.5 BCRELATIVEINDEX 77

6.2.2.6 BCTYPE 77

6.2.2.7 BIN_COL_ID 77

6.2.2.8 BIN_DEFAULT_HEADER_SIZE 78

6.2.2.9 BIN_DIR 78

6.2.2.10 BIN_ROW_ID 78

6.2.2.11 BLUE 78

6.2.2.12 CFG_DIR 78

6.2.2.13 COMPUTE_TIME 78

6.2.2.14 DASH 78

6.2.2.15 DEFAULT_CFG 78
6.2.2.16 DEM 79
6.2.2.17 DEM_CELL_SIZE_LINE 79
6.2.2.18 DEM_HEADER_SIZE 79
6.2.2.19 DEM_NCOLS_LINE 79
6.2.2.20 DEM_NODATA_VALUE_LINE 79
6.2.2.21 DEM_NROWS_LINE 79
6.2.2.22 DEM_XLL_CORNER_LINE 79
6.2.2.23 DEM_YLL_CORNER_LINE 79
6.2.2.24 DT 80
6.2.2.25 EPS12 80
6.2.2.26 ERROR 80
6.2.2.27 EXTBCV1 80
6.2.2.28 EXTBCV2 80
6.2.2.29 FT3_TO_M3_FACTOR 80
6.2.2.30 FT_TO_M_FACTOR 80
6.2.2.31 G 80
6.2.2.32 GHOST_CELL_PADDING 81
6.2.2.33 GRAY 81
6.2.2.34 GREEN 81
6.2.2.35 H 81
6.2.2.36 HALO 81
6.2.2.37 HOUR_TO_SEC_FACTOR 81
6.2.2.38 HYGT 81
6.2.2.39 HYGV 81
6.2.2.40 IN 82
6.2.2.41 INPUT_DIR 82
6.2.2.42 IO_TIME 82
6.2.2.43 MAX_VALUE 82
6.2.2.44 MAXH 82

6.2.2.45 MM_TO_M_FACTOR 82
6.2.2.46 MPI_DATA_TYPE 82
6.2.2.47 MPI_TIME 82
6.2.2.48 N 83
6.2.2.49 OBSERVATION_LOCATION 83
6.2.2.50 OK 83
6.2.2.51 OUTPUT_DIR 83
6.2.2.52 QX 83
6.2.2.53 QY 83
6.2.2.54 RED 83
6.2.2.55 RESET 83
6.2.2.56 RESIZE_TIME 84
6.2.2.57 RHSH0 84
6.2.2.58 RHSH1 84
6.2.2.59 RHSQX0 84
6.2.2.60 RHSQX1 84
6.2.2.61 RHSQY0 84
6.2.2.62 RHSQY1 84
6.2.2.63 RUNID 84
6.2.2.64 RUNIN 85
6.2.2.65 SEC_TO_HOUR_FACTOR 85
6.2.2.66 SIMULATION_TIME 85
6.2.2.67 SQRTG 85
6.2.2.68 SQRTH 85
6.2.2.69 SRC_LOCATION 85
6.2.2.70 SRCP 85
6.2.2.71 THREAD_BLOCK 85
6.2.2.72 TIME_SERIES_DIR 86
6.2.2.73 TIMER_NSECS 86
6.2.2.74 TIMER_SECS 86

6.2.2.75 TOTAL_TIME 86
6.2.2.76 TYPE_DYNAMIC 86
6.2.2.77 TYPE_STATIC 86
6.2.2.78 USE_HALO 86
6.2.2.79 USE_MATRIX 86
6.2.2.80 WARN 87
6.2.2.81 YELLOW 87
6.2.3 Typedef Documentation 87
6.2.3.1 char_t 87
6.2.3.2 dims_t 87
6.2.3.3 sources_list_t 87
6.2.3.4 string_vector 87
6.2.3.5 ull 87
6.2.3.6 value_t 88
6.3 dem_utils.h File Reference 88
6.3.1 Detailed Description 89
6.4 extbc.h File Reference 89
6.4.1 Detailed Description 90
6.5 inflow.h File Reference 90
6.5.1 Detailed Description 91
6.6 kernels.h File Reference 91
6.6.1 Detailed Description 92
6.7 main.cpp File Reference 93
6.7.1 Detailed Description 93
6.7.2 Function Documentation 94
6.7.2.1 main() 94
6.8 matrix.h File Reference 94
6.8.1 Detailed Description 95
6.9 mpi_utils.h File Reference 96
6.9.1 Detailed Description 97

6.9.2 Function Documentation 98
6.9.2.1 create_local_dims() 98
6.9.2.2 exchange() 99
6.9.2.3 scatter_exchange() 99
6.9.2.4 scatter_exchange_int() 100
6.10 output.h File Reference 100
6.10.1 Detailed Description 101
6.11 string_utils.h File Reference 102
6.11.1 Detailed Description 103
6.11.2 Function Documentation 103
6.11.2.1 down_char() 103
6.11.2.2 is_numeric() 104
6.11.2.3 itoa() 104
6.11.2.4 itos() 104
6.11.2.5 split() [1/2] 106
6.11.2.6 split() [2/2] 106
6.11.2.7 tolower() 107
6.11.2.8 toupper() 107
6.11.2.9 up_char() 107
6.11.2.10 vecstr_to_vecflt() 108
6.11.2.11 vecstr_to_vecint() 108
6.12 supertimer.h File Reference 108
6.12.1 Detailed Description 109
6.13 triton.h File Reference 110
6.13.1 Detailed Description 111

Index **113**

Chapter 1

Bug List

File config_utils.h
: No known bugs.

File constants.h
: No known bugs.

File dem_utils.h
: No known bugs.

File extbc.h
: No known bugs.

File inflow.h
: No known bugs.

File kernels.h
: No known bugs.

File main.cpp
: No known bugs.

File matrix.h
: No known bugs.

File mpi_utils.h
: No known bugs.
: No known bugs.

File output.h
: No known bugs.

File string_utils.h
: No known bugs.

File supertimer.h
: No known bugs.

File triton.h
: No known bugs.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

ConfigUtils::arguments< T > 9
SuperTimer::ci_less 17
ExtBC::extBC< T > 24
Hydrograph::hydrograph< T > 31
Matrix::matrix< T > 35
    DemFile::dem_file< T > 18
Matrix::matrix< int > 35
SuperTimer::ci_less::nocase_compare 52
Output::output< T > 52
MpiUtils::partition_data_t 60
SuperTimer::super_timer 63
Triton::triton< T > 67

Chapter 3

Class Index

3.1 Class List

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

ConfigUtils::arguments< T > 9
SuperTimer::ci_less 17
DemFile::dem_file< T > 18
ExtBC::extBC< T > 24
Hydrograph::hydrograph< T > 31
Matrix::matrix< T > 35
SuperTimer::ci_less::nocase_compare 52
Output::output< T > 52
MpiUtils::partition_data_t 60
SuperTimer::super_timer 63
Triton::triton< T > 67

Chapter 4

File Index

4.1 File List

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

config_utils.h
    Header containing the ConfigUtils class 69
constants.h
    Header containing the Constants class 74
dem_utils.h
    Header containing the DemFile class 88
extbc.h
    Header containing the ExtBC class 89
inflow.h
    Header containing the Hydrograph class 90
kernels.h
    Header containing the Kernels class 91
main.cpp
    Main file containing the driver 93
matrix.h
    Header containing the Matrix class 94
mpi_utils.h
    Header containing the MpiUtils class 96
mpi_utils_old.h . . . . **??**
output.h
    Header containing the Output class 100
string_utils.h
    Header containing the StringUtils class 102
supertimer.h
    Header containing the SuperTimer class 108
triton.h
    Header containing the Triton class 110

Chapter 5

Class Documentation

5.1 ConfigUtils::arguments$<$ T $>$ Struct Template Reference

```
#include <config_utils.h>
```

Public Attributes

- bool time_increment_fixed
- bool time_series_flag
- bool gpu_direct_flag
- int checkpoint_id
- int num_sources
- int num_runoffs
- int num_extbc
- int it_count
- int factor_interval_domain_decomposition
- T time_step
- T sim_start_time
- T sim_duration
- T print_interval
- T courant
- T const_mann
- T hextra
- std::string outfile_pattern
- std::string hydrograph_filename
- std::string runoff_filename
- std::string print_option
- std::string max_value_print_option
- std::string input_format
- std::string output_format
- std::string output_option
- std::string dem_filename
- std::string src_loc_file
- std::string runoff_map
- std::string observation_loc_file
- std::string extbc_file
- std::string extbc_dir

- std::string h_infile
- std::string qx_infile
- std::string qy_infile
- std::string n_infile
- std::string domain_decomposition
- std::vector< T > src_x_loc
- std::vector< T > src_y_loc
- std::vector< T > observation_x_loc
- std::vector< T > observation_y_loc
- std::vector< T > extbc_x1_loc
- std::vector< T > extbc_y1_loc
- std::vector< T > extbc_x2_loc
- std::vector< T > extbc_y2_loc
- std::vector< int > extbc_bctype
- std::vector< std::string > extbc_fname

5.1.1 Detailed Description

template< typename T>
struct ConfigUtils::arguments< T >

< Structure to contain all arguments extracted from configuration (cfg) file.

5.1.2 Member Data Documentation

5.1.2.1 checkpoint_id

```
template<typename T>
int ConfigUtils::arguments< T >::checkpoint_id
```

Use for hot start. If 0 then that means a clean start. Greater than 0 means start from that specific checkpoint.

5.1.2.2 const_mann

```
template<typename T>
T ConfigUtils::arguments< T >::const_mann
```

Constant manning value to use in every cell in case of no external manning file is provided.

5.1.2.3 courant

```
template<typename T>
T ConfigUtils::arguments< T >::courant
```

Represents Courant number.

5.1.2.4 dem_filename

```
template<typename T>
std::string ConfigUtils::arguments< T >::dem_filename
```

Directory of the DEM file to use.

5.1.2.5 domain_decomposition

```
template<typename T>
std::string ConfigUtils::arguments< T >::domain_decomposition
```

Domain decomposition. Options are static or dynamic. Static by default

5.1.2.6 extbc_bctype

```
template<typename T>
std::vector<int> ConfigUtils::arguments< T >::extbc_bctype
```

Contains all external boundary condition type serially.

5.1.2.7 extbc_dir

```
template<typename T>
std::string ConfigUtils::arguments< T >::extbc_dir
```

Parent directory of the External boundary condition files.

5.1.2.8 extbc_file

```
template<typename T>
std::string ConfigUtils::arguments< T >::extbc_file
```

Directory of the External boundary condition file to use.

5.1.2.9 extbc_fname

```
template<typename T>
std::vector<std::string> ConfigUtils::arguments< T >::extbc_fname
```

Contains all external boundary condition file name serially.

5.1.2.10 extbc_x1_loc

```
template<typename T>
std::vector<T> ConfigUtils::arguments< T >::extbc_x1_loc
```

Vector to hold all the Longitude value of the starting cell of an external boundary condition.

5.1.2.11 extbc_x2_loc

```
template<typename T>
std::vector<T> ConfigUtils::arguments< T >::extbc_x2_loc
```

Vector to hold all the Longitude value of the ending cell of an external boundary condition.

5.1.2.12 extbc_y1_loc

```
template<typename T>
std::vector<T> ConfigUtils::arguments< T >::extbc_y1_loc
```

Vector to hold all the Latitude value of the starting cell of an external boundary condition.

5.1.2.13 extbc_y2_loc

```
template<typename T>
std::vector<T> ConfigUtils::arguments< T >::extbc_y2_loc
```

Vector to hold all the Latitude value of the ending cell of an external boundary condition.

5.1.2.14 factor_interval_domain_decomposition

```
template<typename T>
int ConfigUtils::arguments< T >::factor_interval_domain_decomposition
```

Factor applied to the print interval time to check for domain decomposition.

5.1.2.15 gpu_direct_flag

```
template<typename T>
bool ConfigUtils::arguments< T >::gpu_direct_flag
```

Flag to allow GPU-Direct use. True = Use GPU-Direct, False = Don't use GPU-Direct.

5.1.2.16 h_infile

```
template<typename T>
std::string ConfigUtils::arguments< T >::h_infile
```

Initial water depth file directory.

5.1.2.17 hextra

```
template<typename T>
T ConfigUtils::arguments< T >::hextra
```

Represents a the minimum water depth tolerance

5.1.2.18 hydrograph_filename

```
template<typename T>
std::string ConfigUtils::arguments< T >::hydrograph_filename
```

Directory of the Hygrograph file to use.

5.1.2.19 input_format

```
template<typename T>
std::string ConfigUtils::arguments< T >::input_format
```

Expected input file format. BIN for binary file or ASC for ascii file.

5.1.2.20 it_count

```
template<typename T>
int ConfigUtils::arguments< T >::it_count
```

The total number of iterations up to a specific point. 0 in case of a clean start, greater than 0 otherwise.

5.1.2.21 max_value_print_option

```
template<typename T>
std::string ConfigUtils::arguments< T >::max_value_print_option
```

Use to determine maximum value of each cells output types. h to output just the h (depth).

5.1.2.22 n_infile

```
template<typename T>
std::string ConfigUtils::arguments< T >::n_infile
```

Directory of the manning file to use.

5.1.2.23 num_extbc

```
template<typename T>
int ConfigUtils::arguments< T >::num_extbc
```

The total number of External boundary cells group. Each group can contain one or multiple cells.

5.1.2.24 num_runoffs

```
template<typename T>
int ConfigUtils::arguments< T >::num_runoffs
```

The total number of Runoffs.

5.1.2.25 num_sources

```
template<typename T>
int ConfigUtils::arguments< T >::num_sources
```

The total number of flow locations in Hygrograph. If there are no flow locations then 0 is allowed.

5.1.2.26 observation_loc_file

```
template<typename T>
std::string ConfigUtils::arguments< T >::observation_loc_file
```

Directory of the file that contains the information of all cells to observe and generate time series output.

5.1.2.27 observation_x_loc

```
template<typename T>
std::vector<T> ConfigUtils::arguments< T >::observation_x_loc
```

Vector to hold all the Longitude value of all the observation cells.

5.1.2.28 observation_y_loc

```
template<typename T>
std::vector<T> ConfigUtils::arguments< T >::observation_y_loc
```

Vector to hold all the Latitude value of all the observation cells.

5.1.2.29 outfile_pattern

```
template<typename T>
std::string ConfigUtils::arguments< T >::outfile_pattern
```

Output file directory and name pattern.

5.1.2.30 output_format

```
template<typename T>
std::string ConfigUtils::arguments< T >::output_format
```

Expected output file format. BIN for binary file or ASC for ascii file.

5.1.2.31 output_option

```
template<typename T>
std::string ConfigUtils::arguments< T >::output_option
```

Strategy to use for outputting into files. PAR for parallel outputs or SEQ for sequential outputs. PAR saves each MPI partitions subdomain in separate files and SEQ saves the whole domain into one file.

5.1.2.32 print_interval

```
template<typename T>
T ConfigUtils::arguments< T >::print_interval
```

Use for outputting files. After every defined print interval time, the program will save outputs in an external file.

5.1.2.33 print_option

```
template<typename T>
std::string ConfigUtils::arguments< T >::print_option
```

Use to determine output types. h to output just the h (depth), huv to output all h (depth),u and v (velocities).

5.1.2.34 qx_infile

```
template<typename T>
std::string ConfigUtils::arguments< T >::qx_infile
```

Initial flux in x direction file directory.

5.1.2.35 qy_infile

```
template<typename T>
std::string ConfigUtils::arguments< T >::qy_infile
```

Initial flux in y direction file directory.

5.1.2.36 runoff_filename

```
template<typename T>
std::string ConfigUtils::arguments< T >::runoff_filename
```

Directory of the Runoff file to use.

5.1.2.37 runoff_map

```
template<typename T>
std::string ConfigUtils::arguments< T >::runoff_map
```

Directory of the Runoff map to use.

5.1.2.38 sim_duration

```
template<typename T>
T ConfigUtils::arguments< T >::sim_duration
```

Finishing time point of a simulation. Regardless of the starting point, simulation always ends at this point.

5.1.2.39 sim_start_time

```
template<typename T>
T ConfigUtils::arguments< T >::sim_start_time
```

Starting time point of a simulation. Usually 0 for a new simulation.

5.1.2.40 src_loc_file

```
template<typename T>
std::string ConfigUtils::arguments< T >::src_loc_file
```

Directory of the file that contains the information of all flow locations.

5.1.2.41 src_x_loc

```
template<typename T>
std::vector<T> ConfigUtils::arguments< T >::src_x_loc
```

Vector to hold all the Longitude value of all the flow locations serially.

5.1.2.42 src_y_loc

```
template<typename T>
std::vector<T> ConfigUtils::arguments< T >::src_y_loc
```

Vector to hold all the Latitude value of all the flow locations serially.

5.1.2.43 time_increment_fixed

```
template<typename T>
bool ConfigUtils::arguments< T >::time_increment_fixed
```

Flag to indicate time step size characteristics. True = Constant time step size, False = Variable time step size.

5.1.2.44 time_series_flag

```
template<typename T>
bool ConfigUtils::arguments< T >::time_series_flag
```

Flag to allow time series output. True = Output time series, False = Don't output time series.

5.1.2.45 time_step

```
template<typename T>
T ConfigUtils::arguments< T >::time_step
```

Indicates the time step size. Time step size determines the time for the next computation.

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

- config_utils.h

5.2 SuperTimer::ci_less Struct Reference

```
#include <supertimer.h>
```

Classes

- struct nocase_compare

Public Member Functions

- bool operator() (const std::string &s1, const std::string &s2) const

  It compares to string. If first string is less than second string, it returns true. Compare happens one by one char.

5.2.1 Detailed Description

< Structure to compare to string.

5.2.2 Member Function Documentation

5.2.2.1 operator()()

```
bool SuperTimer::ci_less::operator() (
            const std::string & s1,
            const std::string & s2 ) const  [inline]
```

It compares to string. If first string is less than second string, it returns true. Compare happens one by one char.

Parameters

s1	First string
s2	Second string

Returns

True or False

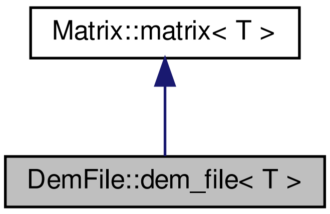
The documentation for this struct was generated from the following file:

- supertimer.h

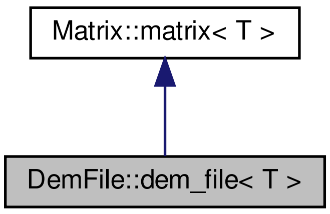
5.3 DemFile::dem_file< T > Class Template Reference

```
#include <dem_utils.h>
```

Inheritance diagram for DemFile::dem_file< T >:



Collaboration diagram for DemFile::dem_file< T >:



Public Member Functions

- dem_file ()

  Constructor. Takes no argument.
- dem_file (int rows, int cols)

  Constructor. Takes number of rows and columns as argument.
- dem_file (Matrix::matrix< T > const &m)

  Constructor. Takes a Matrix object as argument.
- int get_nrows () const

  To get number of rows in DEM domain.
- int get_ncols () const

  To get number of columns in DEM domain.
- T get_xll_corner () const

  To get the X coordinate of the origin.
- T get_yll_corner () const

To get the Y coordinate of the origin.

- T get_cell_size () const

    To get the size of each cell.
- int get_no_data_value () const

    To get the default value if no data.
- void set_nrows (int row)

    To set number of rows in DEM domain.
- void set_ncols (int col)

    To set number of columns in DEM domain.
- void set_xll_corner (T xll)

    To set X coordinate of the origin.
- void set_yll_corner (T yll)

    To set Y coordinate of the origin.
- void set_cell_size (T cell_size)

    To set size of a cell.
- void set_no_data_value (int no_data_value)

    To set default value in case of no data.
- void load_header_from_dem_file_ascii (std::string filename)

    Extract header information from a Ascii DEM file.
- void load_header_from_dem_file_binary (std::string filename)

    Extract header information from a Binary DEM file.

5.3.1 Detailed Description

template<class T>
class DemFile::dem_file< T >

< To process and store DEM data. It extends the base Matrix class.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 dem_file() [1/3]

```
template<class T>
DemFile::dem_file< T >::dem_file ( )  [inline]
```

Constructor. Takes no argument.

5.3.2.2 dem_file() [2/3]

```
template<class T>
DemFile::dem_file< T >::dem_file (
                int rows,
                int cols )  [inline]
```

Constructor. Takes number of rows and columns as argument.

Parameters

rows	Number of rows
cols	Number of columns

5.3.2.3 dem_file() [3/3]

```
template<class T>
DemFile::dem_file< T >::dem_file (
            Matrix::matrix< T > const & m )  [inline]
```

Constructor. Takes a Matrix object as argument.

Parameters

m	Matrix object

5.3.3 Member Function Documentation

5.3.3.1 get_cell_size()

```
template<typename T >
T DemFile::dem_file< T >::get_cell_size ( ) const
```

To get the size of each cell.

Returns

The cell size

5.3.3.2 get_ncols()

```
template<typename T >
int DemFile::dem_file< T >::get_ncols ( ) const
```

To get number of columns in DEM domain.

Returns

Number of columns

5.3.3.3 get_no_data_value()

```
template<typename T >
int DemFile::dem_file< T >::get_no_data_value ( ) const
```

To get the default value if no data.

Returns

No data value

5.3.3.4 get_nrows()

```
template<typename T >
int DemFile::dem_file< T >::get_nrows ( ) const
```

To get number of rows in DEM domain.

Returns

Number of rows

5.3.3.5 get_xll_corner()

```
template<typename T >
T DemFile::dem_file< T >::get_xll_corner ( ) const
```

To get the X coordinate of the origin.

Returns

The X coordinate

5.3.3.6 get_yll_corner()

```
template<typename T >
T DemFile::dem_file< T >::get_yll_corner ( ) const
```

To get the Y coordinate of the origin.

Returns

The Y coordinate

5.3.3.7 load_header_from_dem_file_ascii()

```
template<typename T >
void DemFile::dem_file< T >::load_header_from_dem_file_ascii (
             std::string filename )
```

Extract header information from a Ascii DEM file.

Parameters

filename	Ascii file name

5.3.3.8 load_header_from_dem_file_binary()

```
template<typename T >
void DemFile::dem_file< T >::load_header_from_dem_file_binary (
             std::string filename )
```

Extract header information from a Binary DEM file.

Parameters

filename	Binary file name

5.3.3.9 set_cell_size()

```
template<typename T >
void DemFile::dem_file< T >::set_cell_size (
             T cell_size )
```

To set size of a cell.

Parameters

cell_size	Cell size

5.3.3.10 set_ncols()

```
template<typename T >
void DemFile::dem_file< T >::set_ncols (
             int col )
```

To set number of columns in DEM domain.

Parameters

col	Number of columns

5.3.3.11 set_no_data_value()

```
template<typename T >
void DemFile::dem_file< T >::set_no_data_value (
             int no_data_value )
```

To set default value in case of no data.

Parameters

no_data_value	Deafult value

5.3.3.12 set_nrows()

```
template<typename T >
void DemFile::dem_file< T >::set_nrows (
             int row )
```

To set number of rows in DEM domain.

Parameters

row	Number of rows

5.3.3.13 set_xll_corner()

```
template<typename T >
void DemFile::dem_file< T >::set_xll_corner (
             T xll )
```

To set X coordinate of the origin.

Parameters

xll	X coordinate

5.3.3.14 set_yll_corner()

```
template<typename T >
void DemFile::dem_file< T >::set_yll_corner (
             T yll )
```

To set Y coordinate of the origin.

Parameters

yll	Y coordinate

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

- dem_utils.h

5.4 ExtBC::extBC< T > Class Template Reference

```
#include <extbc.h>
```

Public Member Functions

- extBC ()
  Constructor. Takes no argument.
- extBC (std::string filename, int bctype)
  Constructor. Takes filename containing boundary condition and boundary condition type. Reads from files and push each row in a vector and construct data.
- void load_from_file (std::string filename, int bctype)
  Reads from files and push each row in a vector and construct data.
- int check_extreme_extbc (std::vector< int > e_cols, std::vector< int > e_rows, int ncols, int nrows)
  It checks for extreme boundary condition and calculates the number of cells in that boundary condition.
- void create_involved_cells (std::vector< int > e_cols, std::vector< int > e_rows, int ncols, int nrows, int bctype)
  It calculates involved cells corresponding to a boundary condition.
- std::vector< std::vector< T > > get_rows ()
  It returns all data saved from boundary condition file.
- T get_var1_at (int index)
  It calculates vector at a specific index and return that vecors 0 indexed value.
- T get_var2_at (int index)
  It calculates vector at a specific index and return that vecors 1 indexed value.
- int get_num_rows ()
  Use to get number of rows in boundary condition data.
- void set_num_rows (int rows)
  Use to set number of rows in boundary condition data.
- void convert_to_secs ()
  It converts hour data to seconds.

Public Attributes

- int ncells
- int location
- int ncells_local
- std::vector< int > extreme_rows
- std::vector< int > extreme_cols
- std::vector< int > i_cols
- std::vector< int > i_rows

5.4.1 Detailed Description

template<class T>
class ExtBC::extBC< T >

< To process and store data related to external boundary condition.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 extBC() [1/2]

```
template<class T >
ExtBC::extBC< T >::extBC ( )
```

Constructor. Takes no argument.

5.4.2.2 extBC() [2/2]

```
template<class T >
ExtBC::extBC< T >::extBC (
             std::string filename,
             int bctype )
```

Constructor. Takes filename containing boundary condition and boundary condition type. Reads from files and push each row in a vector and construct data.

Parameters

filename	File name
bctype	Boundary condition type

5.4.3 Member Function Documentation

5.4.3.1 check_extreme_extbc()

```
template<typename T >
int ExtBC::extBC< T >::check_extreme_extbc (
             std::vector< int > e_cols,
             std::vector< int > e_rows,
             int ncols,
             int nrows )
```

It checks for extreme boundary condition and calculates the number of cells in that boundary condition.

Parameters

e_cols	Extreme columns vector
e_rows	Extreme rows vector
ncols	Number of columns
nrows	Number of rows

Returns

Number of cells in that boundary condition

5.4.3.2 convert_to_secs()

```
template<typename T >
void ExtBC::extBC< T >::convert_to_secs ( )
```

It converts hour data to seconds.

5.4.3.3 create_involved_cells()

```
template<typename T >
void ExtBC::extBC< T >::create_involved_cells (
             std::vector< int > e_cols,
             std::vector< int > e_rows,
             int ncols,
             int nrows,
             int bctype )
```

It calculates involved cells corresponding to a boundary condition.

Parameters

e_cols	Extreme columns vector
e_rows	Extreme rows vector
ncols	Number of columns
nrows	Number of rows
bctype	Boundary condition type

5.4.3.4 get_num_rows()

```
template<typename T >
int ExtBC::extBC< T >::get_num_rows ( )
```

Use to get number of rows in boundary condition data.

Returns

Number of rows

5.4.3.5 get_rows()

```
template<typename T >
std::vector< std::vector< T > > ExtBC::extBC< T >::get_rows ( )
```

It returns all data saved from boundary condition file.

Returns

Boundary condition data

5.4.3.6 get_var1_at()

```
template<typename T >
T ExtBC::extBC< T >::get_var1_at (
              int index )
```

It calculates vector at a specific index and return that vecors 0 indexed value.

Parameters

index	Data vector's index

Returns

value

5.4.3.7 get_var2_at()

```
template<typename T >
T ExtBC::extBC< T >::get_var2_at (
              int index )
```

It calculates vector at a specific index and return that vecors 1 indexed value.

Parameters

index	Data vector's index

Returns

value

5.4.3.8 load_from_file()

```
template<typename T >
void ExtBC::extBC< T >::load_from_file (
             std::string filename,
             int bctype )
```

Reads from files and push each row in a vector and construct data.

Parameters

filename	File name
bctype	Boundary condition type

5.4.3.9 set_num_rows()

```
template<typename T >
void ExtBC::extBC< T >::set_num_rows (
             int rows )
```

Use to set number of rows in boundary condition data.

Parameters

rows	Number of rows

5.4.4 Member Data Documentation

5.4.4.1 extreme_cols

```
template<class T>
std::vector<int> ExtBC::extBC< T >::extreme_cols
```

Extreme columns

5.4.4.2 extreme_rows

```
template<class T>
std::vector<int> ExtBC::extBC< T >::extreme_rows
```

Extreme rows

5.4.4.3 i_cols

```
template<class T>
std::vector<int> ExtBC::extBC< T >::i_cols
```

Involved columns.

5.4.4.4 i_rows

```
template<class T>
std::vector<int> ExtBC::extBC< T >::i_rows
```

Involved rows.

5.4.4.5 location

```
template<class T>
int ExtBC::extBC< T >::location
```

0–> westBoundary 1–>northBoundary 2–>eastBoundary 3–> southBoundary

5.4.4.6 ncells

```
template<class T>
int ExtBC::extBC< T >::ncells
```

Number of cells of a boundary condition.

5.4.4.7 ncells_local

```
template<class T>
int ExtBC::extBC< T >::ncells_local
```

Number of cells of a boundary condition in a subdomain.

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

- extbc.h

5.5 Hydrograph::hydrograph< T > Class Template Reference

```
#include <inflow.h>
```

Public Member Functions

- hydrograph ()

  Constructor. Takes no argument.
- hydrograph (std::string filename)

  Constructor. Takes filename as an argument to construct the object.
- void load_from_file (std::string filename)

  It reads content from a hydrograph file and construct data.
- std::vector< std::vector< T > > get_rows ()

  To get all the contents in each rows of hydrograph file.
- T get_flow_at (int index, int source_num)

  It calculates flow value at a specific row index for a specific flow location number.
- T get_time_at (int index)

  It calculates time at a specific row index.
- int get_num_inflow_rows ()

  To get number of inflow rows.
- int get_num_inflows ()

  To get number of inflows.
- void convert_time_hr_to_secs ()

  It converts all data time values from hour to second.
- void convert_rate_hr_to_secs ()

  It converts all rate values from hour to second.
- void convert_rate_mm_to_m ()

  It converts all mm values to m.
- void set_num_flow_rows (int rows)

  It sets number of flow rows.
- void set_num_sources (int sources)

  It sets number of inflow locations.

5.5.1 Detailed Description

template<class T>
class Hydrograph::hydrograph< T >

< To process and store hydrograph input files.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 hydrograph() [1/2]

```
template<class T >
Hydrograph::hydrograph< T >::hydrograph ( )
```

Constructor. Takes no argument.

5.5.2.2 hydrograph() [2/2]

```
template<class T >
Hydrograph::hydrograph< T >::hydrograph (
             std::string filename )
```

Constructor. Takes filename as an argument to construct the object.

Parameters

filename	Input file name

5.5.3 Member Function Documentation

5.5.3.1 convert_rate_hr_to_secs()

```
template<typename T >
void Hydrograph::hydrograph< T >::convert_rate_hr_to_secs ( )
```

It converts all rate values from hour to second.

5.5.3.2 convert_rate_mm_to_m()

```
template<typename T >
void Hydrograph::hydrograph< T >::convert_rate_mm_to_m ( )
```

It converts all mm values to m.

5.5.3.3 convert_time_hr_to_secs()

```
template<typename T >
void Hydrograph::hydrograph< T >::convert_time_hr_to_secs ( )
```

It converts all data time values from hour to second.

5.5.3.4 get_flow_at()

```
template<typename T >
T Hydrograph::hydrograph< T >::get_flow_at (
            int index,
            int source_num )
```

It calculates flow value at a specific row index for a specific flow location number.

Parameters

index	Row index
source_num	Flow location serial number

Returns

Flow value

5.5.3.5 get_num_inflow_rows()

```
template<typename T >
int Hydrograph::hydrograph< T >::get_num_inflow_rows ( )
```

To get number of inflow rows.

Returns

Inflow rows count

5.5.3.6 get_num_inflows()

```
template<typename T >
int Hydrograph::hydrograph< T >::get_num_inflows ( )
```

To get number of inflows.

Returns

Inflows count

5.5.3.7 get_rows()

```
template<typename T >
std::vector< std::vector< T > > Hydrograph::hydrograph< T >::get_rows ( )
```

To get all the contents in each rows of hydrograph file.

Returns

All input rows.

5.5.3.8 get_time_at()

```
template<typename T >
T Hydrograph::hydrograph< T >::get_time_at (
             int index )
```

It calculates time at a specific row index.

Parameters

index	Row index

Returns

Time value

5.5.3.9 load_from_file()

```
template<typename T >
void Hydrograph::hydrograph< T >::load_from_file (
             std::string filename )
```

It reads content from a hydrograph file and construct data.

Parameters

filename	Input file name

5.5.3.10 set_num_flow_rows()

```
template<typename T >
```

```
void Hydrograph::hydrograph< T >::set_num_flow_rows (
             int rows )
```

It sets number of flow rows.

Parameters

rows	Number of rows

5.5.3.11 set_num_sources()

```
template<typename T >
void Hydrograph::hydrograph< T >::set_num_sources (
             int sources )
```

It sets number of inflow locations.

Parameters

sources	inflow location count

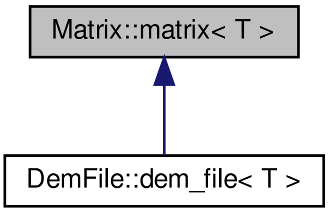
The documentation for this class was generated from the following file:

- inflow.h

5.6 Matrix::matrix< T > Class Template Reference

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

Inheritance diagram for Matrix::matrix< T >:



Public Member Functions

- matrix ()
  Constructor.
- matrix (int rows, int cols)
  Constructor. Creates a matrix of given size.
- matrix (int rows, int cols, T ∗∗arr)
  Constructor. Creates a matrix of giver size and 2D array.
- matrix (matrix< T > const &m)
  Constructor. Creates a matrix from another matrix.
- ∼matrix ()
  Destruction. Releases allocated memory.
- T & operator() (int row, int col)
  Operator to create matrix by address and given size.
- T operator() (int row, int col) const
  Operator to create matrix by given size.
- matrix & operator= (matrix< T > m)
  Assignement operator to copy a matrix object into another.
- matrix & operator∗= (T value)
  It multiplies each cell of a matrix by a constant value and creates a copy.
- matrix & operator∗ (T value)
  It multiplies each cell of a matrix by a constant value.
- matrix & operator+ (T value)
  It adds a constant value with each cell of a matrix.
- matrix & operator+ (matrix const &m)
  It adds corresponding cells value of two different matrix.
- matrix & operator+= (matrix const &m)
  It adds corresponding cells value of two different matrix and creates a copy.
- matrix & operator+= (T value)
  It adds a constant value with each cell of a matrix and create a copy.
- matrix< T > & operator∗ (matrix const &m)
  It multiply corresponding cells value of two different matrix.
- T ∗ get_data () const
  Get data from the matrix.
- T ∗ begin ()
  Get beginning address of data.
- T ∗ get_address_at (int row, int col)
  It calculates address of a specific position of data.
- int get_num_rows () const
  Gets the total number of rows.
- int get_num_cols () const
  Gets the total number of columns.
- int get_ghost_nrows () const
  Gets the number of ghost rows in each boundary.
- int get_ghost_ncols () const
  Gets the number of ghost columns in each boundary.
- void set_size (int rows, int cols)
  Sets the number of rows and columns of a Matrix.
- void resize (int rows, int cols)
  It resizes previous matrix in a new dimension.
- void set_value (int row, int col, T value)

It sets value in a particular cell.
- void set_value (std::pair< int, int > cell, T value)

  It sets value in a particular cell.
- void set_value (int index, T value)

  It sets value in a particular cell.
- T get_value (int row, int col)

  Gets value from a particular cell.
- T get_value (std::pair< int, int >)

  Gets value from a particular cell.
- T get_value (int index)

  Gets value from a particular cell.
- void add_ghost_cells (int grows, int gcols, T value)

  It adds ghost rows and columns in each boundary.
- void remove_ghost_cells ()

  It removes ghost cells from the domain.
- void copy_value_into_ghost_cells ()

  It copies values from boundary cells of domain into ghost cells.
- void copy_elevation_into_ghost_cells (std::vector< int > irows, std::vector< int > icols, int ncells, int location)

  It copies the elevation of boundary cells values into ghost cells.
- void set_infinite_walls ()

  Put infinite walls in boundary cells.
- bool is_inbounds (int row, int col)

  It calculates if a cell in inside boundary or not.
- void zero_fill ()

  Fill whole matrix with 0 as a floting point number.
- void zero_fill_int ()

  Fill whole matrix with 0 as a integer number.
- void pow (T e)

  Change value of each cell as a base with a power.
- void square ()

  Change value of each cell by its square.
- void load_from_ascii_file (std::string &filepath)

  Load values into matrix from an ascii file.
- void load_from_ascii_file (int rows, int cols, std::string &filepath)

  Load values into matrix from an ascii file.
- void load_from_ascii_file (int rows, int cols, std::string &filepath, int header_size)

  Load values into matrix from an ascii file.
- void load_from_binary_file (int rows, int cols, std::string &filepath)

  Load values into matrix from a binary file.
- void load_from_binary_file (int rows, int cols, std::string &filepath, int header_size)

  Load values into matrix from a binary file.
- std::pair< int, int > get_dims_2d (std::string &filepath)

  It calculates dimension of an ascii file.

5.6.1 Detailed Description

template<class T>
class Matrix::matrix< T >

< Matrix class to process 2D grid data structure.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 matrix() [1/4]

```
template<class T >
Matrix::matrix< T >::matrix ( )
```

Constructor.

5.6.2.2 matrix() [2/4]

```
template<class T >
Matrix::matrix< T >::matrix (
             int rows,
             int cols )
```

Constructor. Creates a matrix of given size.

Parameters

rows	Number of rows
cols	Number of columns

5.6.2.3 matrix() [3/4]

```
template<class T>
Matrix::matrix< T >::matrix (
             int rows,
             int cols,
             T ** arr )
```

Constructor. Creates a matrix of giver size and 2D array.

Parameters

rows	Number of rows
cols	Number of columns
arr	2d Array

5.6.2.4 matrix() [4/4]

```
template<class T>
Matrix::matrix< T >::matrix (
             matrix< T > const & m )
```

Constructor. Creates a matrix from another matrix.

Parameters

m	Giver matrix

5.6.2.5 ~matrix()

```
template<class T >
Matrix::matrix< T >::~matrix ( )
```

Destruction. Releases allocated memory.

5.6.3 Member Function Documentation

5.6.3.1 add_ghost_cells()

```
template<typename T>
void Matrix::matrix< T >::add_ghost_cells (
             int grows,
             int gcols,
             T value )
```

It adds ghost rows and columns in each boundary.

Parameters

grows	Number of ghost rows
grows	Number of ghost columns
value	Value of each ghost cell

5.6.3.2 begin()

```
template<typename T >
T * Matrix::matrix< T >::begin ( )
```

Get beginning address of data.

Returns

Pointer of first position

5.6.3.3 copy_elevation_into_ghost_cells()

```
template<typename T >
void Matrix::matrix< T >::copy_elevation_into_ghost_cells (
             std::vector< int > irows,
             std::vector< int > icols,
             int ncells,
             int location )
```

It copies the elevation of boundary cells values into ghost cells.

Parameters

irows	Index of boundary cells row
icols	Index of boundary cells column
ncells	Number of cells
location	Position of the boundary

5.6.3.4 copy_value_into_ghost_cells()

```
template<typename T >
void Matrix::matrix< T >::copy_value_into_ghost_cells ( )
```

It copies values from boundary cells of domain into ghost cells.

5.6.3.5 get_address_at()

```
template<typename T >
T * Matrix::matrix< T >::get_address_at (
             int row,
             int col )
```

It calculates address of a specific position of data.

Parameters

row	Row number of cell
col	Column number of cell

Returns

Pointer of the position

5.6.3.6 get_data()

```
template<typename T >
T * Matrix::matrix< T >::get_data ( ) const
```

Get data from the matrix.

Returns

Pointer of array

5.6.3.7 get_dims_2d()

```
template<typename T >
std::pair< int, int > Matrix::matrix< T >::get_dims_2d (
             std::string & filepath )
```

It calculates dimension of an ascii file.

Parameters

filepath	File name

Returns

Rows and columns

5.6.3.8 get_ghost_ncols()

```
template<typename T >
int Matrix::matrix< T >::get_ghost_ncols ( ) const
```

Gets the number of ghost columns in each boundary.

Returns

Number of columns

5.6.3.9 get_ghost_nrows()

```
template<typename T >
int Matrix::matrix< T >::get_ghost_nrows ( ) const
```

Gets the number of ghost rows in each boundary.

Returns

Number of rows

5.6.3.10 get_num_cols()

```
template<typename T >
int Matrix::matrix< T >::get_num_cols ( ) const
```

Gets the total number of columns.

Returns

Number ofcolumns

5.6.3.11 get_num_rows()

```
template<typename T >
int Matrix::matrix< T >::get_num_rows ( ) const
```

Gets the total number of rows.

Returns

Number of rows

5.6.3.12 get_value() [1/3]

```
template<typename T >
T Matrix::matrix< T >::get_value (
              int row,
              int col )
```

Gets value from a particular cell.

Parameters

row	Row index
col	Colum index

Returns

Value of that cell

5.6.3.13 get_value() [2/3]

```
template<typename T >
T Matrix::matrix< T >::get_value (
             std::pair< int, int > cell )
```

Gets value from a particular cell.

Parameters

cell	Cell index in pair

Returns

Value of that cell

5.6.3.14 get_value() [3/3]

```
template<typename T >
T Matrix::matrix< T >::get_value (
             int index )
```

Gets value from a particular cell.

Parameters

index	Cell index

Returns

Value of that cell

5.6.3.15 is_inbounds()

```
template<typename T >
bool Matrix::matrix< T >::is_inbounds (
             int row,
             int col )
```

It calculates if a cell in inside boundary or not.

Parameters

row	Row index
col	Column index

Returns

Bound status

5.6.3.16 load_from_ascii_file() [1/3]

```
template<typename T >
void Matrix::matrix< T >::load_from_ascii_file (
             std::string & filepath )
```

Load values into matrix from an ascii file.

Parameters

filepath	File name

5.6.3.17 load_from_ascii_file() [2/3]

```
template<typename T >
void Matrix::matrix< T >::load_from_ascii_file (
             int rows,
             int cols,
             std::string & filepath )
```

Load values into matrix from an ascii file.

Parameters

rows	Number of rows
cols	Number of columns
filepath	File name

5.6.3.18 load_from_ascii_file() [3/3]

```
template<typename T >
void Matrix::matrix< T >::load_from_ascii_file (
             int rows,
             int cols,
             std::string & filepath,
             int header_size )
```

Load values into matrix from an ascii file.

Parameters

rows	Number of rows
cols	Number of columns
filepath	File name
header_size	Number of headers

5.6.3.19 load_from_binary_file() [1/2]

```
template<typename T >
void Matrix::matrix< T >::load_from_binary_file (
             int rows,
             int cols,
             std::string & filepath )
```

Load values into matrix from a binary file.

Parameters

rows	Number of rows
cols	Number of columns
filepath	File name

5.6.3.20 load_from_binary_file() [2/2]

```
template<typename T >
void Matrix::matrix< T >::load_from_binary_file (
             int rows,
             int cols,
             std::string & filepath,
             int header_size )
```

Load values into matrix from a binary file.

Parameters

rows	Number of rows
cols	Number of columns
filepath	File name
header_size	Number of headers

5.6.3.21 operator()() [1/2]

```
template<typename T >
T & Matrix::matrix< T >::operator() (
             int row,
             int col )
```

Operator to create matrix by address and given size.

Parameters

rows	Number of rows
cols	Number of columns

5.6.3.22 operator()() [2/2]

```
template<typename T >
T Matrix::matrix< T >::operator() (
             int row,
             int col ) const
```

Operator to create matrix by given size.

Parameters

rows	Number of rows
cols	Number of columns

5.6.3.23 operator*() [1/2]

```
template<typename T>
matrix< T > & Matrix::matrix< T >::operator* (
             T value )
```

It multiplies each cell of a matrix by a constant value.

Parameters

value	Contant multiplier

5.6.3.24 operator∗() [2/2]

```
template<typename T>
matrix< T > & Matrix::matrix< T >::operator∗ (
              matrix< T > const & m )
```

It multiply corresponding cells value of two different matrix.

Parameters

m	Matrix

5.6.3.25 operator∗=()

```
template<typename T>
matrix< T > & Matrix::matrix< T >::operator∗= (
              T value )
```

It multiplies each cell of a matrix by a constant value and creates a copy.

Parameters

value	Contant multiplier

5.6.3.26 operator+() [1/2]

```
template<class T>
matrix& Matrix::matrix< T >::operator+ (
              T value )
```

It adds a constant value with each cell of a matrix.

Parameters

value	Contant addition value

5.6.3.27 operator+() [2/2]

```
template<typename T>
matrix< T > & Matrix::matrix< T >::operator+ (
            matrix< T > const & m )
```

It adds corresponding cells value of two different matrix.

Parameters

m	Matrix

5.6.3.28 operator+=() [1/2]

```
template<typename T >
matrix< T > & Matrix::matrix< T >::operator+= (
            matrix< T > const & m )
```

It adds corresponding cells value of two different matrix and creates a copy.

Parameters

m	Matrix

5.6.3.29 operator+=() [2/2]

```
template<typename T>
matrix< T > & Matrix::matrix< T >::operator+= (
            T value )
```

It adds a constant value with each cell of a matrix and create a copy.

Parameters

value	Contant addition value

5.6.3.30 operator=()

```
template<typename T>
matrix< T > & Matrix::matrix< T >::operator= (
            matrix< T > m )
```

Assignement operator to copy a matrix object into another.

Parameters

m	Matrix object

5.6.3.31 pow()

```
template<typename T>
void Matrix::matrix< T >::pow (
             T e )
```

Change value of each cell as a base with a power.

Parameters

e	Power

5.6.3.32 remove_ghost_cells()

```
template<typename T >
void Matrix::matrix< T >::remove_ghost_cells ( )
```

It removes ghost cells from the domain.

5.6.3.33 resize()

```
template<typename T >
void Matrix::matrix< T >::resize (
             int rows,
             int cols )
```

It resizes previous matrix in a new dimension.

Parameters

rows	Number of rows
cols	Number of columns

5.6.3.34 set_infinite_walls()

```
template<typename T >
```

```
void Matrix::matrix< T >::set_infinite_walls ( )
```

Put infinite walls in boundary cells.

5.6.3.35 set_size()

```
template<typename T >
void Matrix::matrix< T >::set_size (
             int rows,
             int cols )
```

Sets the number of rows and columns of a Matrix.

Parameters

rows	Number of rows
cols	Number of columns

5.6.3.36 set_value() [1/3]

```
template<typename T>
void Matrix::matrix< T >::set_value (
             int row,
             int col,
             T value )
```

It sets value in a particular cell.

Parameters

row	Row index
col	Column index
value	Value to set

5.6.3.37 set_value() [2/3]

```
template<typename T>
void Matrix::matrix< T >::set_value (
             std::pair< int, int > cell,
             T value )
```

It sets value in a particular cell.

Parameters

cell	Cell index in pair
value	Value to set

5.6.3.38 set_value() [3/3]

```
template<typename T>
void Matrix::matrix< T >::set_value (
             int index,
             T value )
```

It sets value in a particular cell.

Parameters

index	Cell index
value	Value to set

5.6.3.39 square()

```
template<typename T >
void Matrix::matrix< T >::square ( )
```

Change value of each cell by its square.

5.6.3.40 zero_fill()

```
template<typename T >
void Matrix::matrix< T >::zero_fill ( )
```

Fill whole matrix with 0 as a floting point number.

5.6.3.41 zero_fill_int()

```
template<typename T >
void Matrix::matrix< T >::zero_fill_int ( )
```

Fill whole matrix with 0 as a integer number.

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

- matrix.h

5.7 SuperTimer::ci_less::nocase_compare Struct Reference

```
#include <supertimer.h>
```

Public Member Functions

- bool operator() (const unsigned char &c1, const unsigned char &c2) const

  It compares to char. If first char is less than second char, it returns true.

5.7.1 Detailed Description

$<$ Structure to compare to char.

5.7.2 Member Function Documentation

5.7.2.1 operator()()

```
bool SuperTimer::ci_less::nocase_compare::operator() (
            const unsigned char & c1,
            const unsigned char & c2 ) const  [inline]
```

It compares to char. If first char is less than second char, it returns true.

Parameters

c1	First char
c2	Second char

Returns

True or False

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

- supertimer.h

5.8 Output::output$<$ T $>$ Class Template Reference

```
#include <output.h>
```

Public Member Functions

- output ()

  Constructor.
- ∼output ()

  Destructor. Releases any allocated memory.
- void init (int rows, int cols, int rank, int size, std::string project_dir, std::string outfile_pattern, int time_series↩ _flag, std::string cfg_content, std::string output_option)

  It initializes anything related to outputs in file.
- void init_time_series (int observation_loc_size, Constants::sources_list_t observation_cells)

  It initializes time series outputs in a file.
- void write_output (Matrix::matrix< T > &h_arr, Matrix::matrix< T > &qx_arr, Matrix::matrix< T > &qy_arr, std::string output_format, std::string print_option, int print_id, int it_count, T simtime, T average_dt, Matrix↩ ::matrix< T > &max_value_h, std::string max_value_print_option)

  It calculates which data to output in file. Also prints checkpoint id.
- void write_output_ascii_sequential (Matrix::matrix< T > &arr, std::string what_mat, int print_id, T simtime)

  It outputs a specific data array's full domain in a single ascii file.
- void write_output_ascii_parallel (Matrix::matrix< T > &arr, std::string what_mat, int print_id, T simtime)

  It outputs a specific data array's sub domain in a ascii file. All subdomain outputs seperately in different file.
- void write_output_binary_sequential (Matrix::matrix< T > &arr, std::string what_mat, int print_id, T simtime)

  It outputs a specific data array's full domain in a single binary file.
- void write_output_binary_parallel (Matrix::matrix< T > &arr, std::string what_mat, int print_id, T simtime)

  It outputs a specific data array's sub domain in a binary file. All subdomain outputs seperately in different file.
- std::string get_mat_path (std::string what, std::string root_dir, std::string subdir, int print_id, std::string extension)

  It calculates output file name.
- void output_time_series (std::string what_mat, int print_id, T simtime)

  It outputs time series data in a file.
- void output_cfg (T simtime, int print_id, T average_dt, int it_count)

  It calculates content of updated configuration and outputs it in a file.
- void write_times (SuperTimer::super_timer st, int print_id)

  It calculates all custom timer value and output them.
- double average (double a[ ], int n)

  It calculates average time of each timer for all MPI processes.
- void write_domain_decomposition (MpiUtils::partition_data_t pd, int print_id)

  It writes the evolution of subdomain dimensions if dynamic load balancing is enabled.

Public Attributes

- int cur_proc_data_size = 0
- int ∗ recvcounts = NULL
- int total_data_size = 0
- int ∗ displs = NULL
- T ∗ total_data_arr = NULL

5.8.1 Detailed Description

template<class T>
class Output::output< T >

< Ths class handles all data outputs in file.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 output()

```
template<class T>
Output::output< T >::output ( )  [inline]
```

Constructor.

5.8.2.2 ∼output()

```
template<class T >
Output::output< T >::∼output ( )
```

Destructor. Releases any allocated memory.

5.8.3 Member Function Documentation

5.8.3.1 average()

```
template<typename T >
double Output::output< T >::average (
             double a[ ],
             int n )
```

It calculates average time of each timer for all MPI processes.

Parameters

a	Time array
n	Size

Returns

Average value

5.8.3.2 get_mat_path()

```
template<typename T >
std::string Output::output< T >::get_mat_path (
             std::string what,
             std::string root_dir,
             std::string subdir,
             int print_id,
             std::string extension )
```

It calculates output file name.

Parameters

what	Data type
subdir	Output format directory
print_id	Current checkpoint id
extension	File extension

Returns

File name

5.8.3.3 init()

```
template<typename T >
void Output::output< T >::init (
             int rows,
             int cols,
             int rank,
             int size,
             std::string project_dir,
             std::string outfile_pattern,
             int time_series_flag,
             std::string cfg_content,
             std::string output_option )
```

It initializes anything related to outputs in file.

Parameters

rows	Rows in subdomain
cols	Columns in subdomain
rank	Current sub domain id
size	Number of sub domains
project_dir	Main project directory
outfile_pattern	Output file name pattern
time_series_flag	Flag to output time series or not
cfg_content	Contents of input cfg file
output_option	Determines how to write data

5.8.3.4 init_time_series()

```
template<typename T >
void Output::output< T >::init_time_series (
             int observation_loc_size,
             Constants::sources_list_t observation_cells )
```

It initializes time series outputs in a file.

Parameters

observation_loc_size	Number of cells
observation_cells	All cell index

5.8.3.5 output_cfg()

```
template<typename T >
void Output::output< T >::output_cfg (
             T simtime,
             int print_id,
             T average_dt,
             int it_count )
```

It calculates content of updated configuration and outputs it in a file.

Parameters

simtime	Current time of simulation
print_id	Current checkpoint id
average←_dt	Average time step size from the last output
it_count	Total number of iterations so far

5.8.3.6 output_time_series()

```
template<typename T >
void Output::output< T >::output_time_series (
             std::string what_mat,
             int print_id,
             T simtime )
```

It outputs time series data in a file.

Parameters

what_mat	Data type
print_id	Current checkpoint id
simtime	Current time of simulation

5.8.3.7 write_domain_decomposition()

```
template<typename T >
void Output::output< T >::write_domain_decomposition (
            MpiUtils::partition_data_t pd,
            int print_id )
```

It writes the evolution of subdomain dimensions if dynamic load balancing is enabled.

Parameters

pd	Partition data
print←_id	Print output id

5.8.3.8 write_output()

```
template<typename T >
void Output::output< T >::write_output (
            Matrix::matrix< T > & h_arr,
            Matrix::matrix< T > & qx_arr,
            Matrix::matrix< T > & qy_arr,
            std::string output_format,
            std::string print_option,
            int print_id,
            int it_count,
            T simtime,
            T average_dt,
            Matrix::matrix< T > & max_value_h,
            std::string max_value_print_option )
```

It calculates which data to output in file. Also prints checkpoint id.

Parameters

h_arr	Water depth data
qx_arr	Discharge in x direction data
qy_arr	Discharge in y direction data
output_format	Format of output files
print_option	Which data to write
print_id	Current checkpoint id
it_count	Number of iterations so far
simtime	Current time of simulation
average_dt	Average time step size from the last output
max_value_h	Max value of water depth data
max_value_print_option	Which max value data to write

5.8.3.9 write_output_ascii_parallel()

```
template<typename T >
void Output::output< T >::write_output_ascii_parallel (
             Matrix::matrix< T > & arr,
             std::string what_mat,
             int print_id,
             T simtime )
```

It outputs a specific data array's sub domain in a ascii file. All subdomain outputs seperately in different file.

Parameters

arr	Subdomain data
what_mat	Data type
print_id	Current checkpoint id
simtime	Current time of simulation

5.8.3.10 write_output_ascii_sequential()

```
template<typename T >
void Output::output< T >::write_output_ascii_sequential (
             Matrix::matrix< T > & arr,
             std::string what_mat,
             int print_id,
             T simtime )
```

It outputs a specific data array's full domain in a single ascii file.

Parameters

arr	Subdomain data
what_mat	Data type
print_id	Current checkpoint id
simtime	Current time of simulation

5.8.3.11 write_output_binary_parallel()

```
template<typename T >
void Output::output< T >::write_output_binary_parallel (
             Matrix::matrix< T > & arr,
             std::string what_mat,
             int print_id,
             T simtime )
```

It outputs a specific data array's sub domain in a binary file. All subdomain outputs seperately in different file.

Parameters

arr	Subdomain data
what_mat	Data type
print_id	Current checkpoint id
simtime	Current time of simulation

5.8.3.12 write_output_binary_sequential()

```
template<typename T >
void Output::output< T >::write_output_binary_sequential (
             Matrix::matrix< T > & arr,
             std::string what_mat,
             int print_id,
             T simtime )
```

It outputs a specific data array's full domain in a single binary file.

Parameters

arr	Subdomain data
what_mat	Data type
print_id	Current checkpoint id
simtime	Current time of simulation

5.8.3.13 write_times()

```
template<typename T >
void Output::output< T >::write_times (
             SuperTimer::super_timer st,
             int print_id )
```

It calculates all custom timer value and output them.

Parameters

st	Timer object

5.8.4 Member Data Documentation

5.8.4.1 cur_proc_data_size

```
template<class T>
int Output::output< T >::cur_proc_data_size = 0
```

Number of cells in current subdomain

5.8.4.2 displs

```
template<class T>
int* Output::output< T >::displs = NULL
```

Position array to hold each sub domains starting point in main domain

5.8.4.3 recvcounts

```
template<class T>
int* Output::output< T >::recvcounts = NULL
```

Array to hold every subdomains cell count

5.8.4.4 total_data_arr

```
template<class T>
T* Output::output< T >::total_data_arr = NULL
```

Main domains data or collection data of every subdomain

5.8.4.5 total_data_size

```
template<class T>
int Output::output< T >::total_data_size = 0
```

Number of cells in main domain

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

- output.h

5.9 MpiUtils::partition_data_t Struct Reference

```
#include <mpi_utils.h>
```

Public Member Functions

- partition_data_t ()

    Constructor.
- partition_data_t (int s, int r, int c)

    Constructor.
- partition_data_t ()

    Constructor.
- partition_data_t (int s, int r, int c, std::string t, int ri, int ci)

    Constructor.

Public Attributes

- int size
- int rows
- int cols
- std::vector< Constants::dims_t > part_dims
- int rows_ini
- int cols_ini

5.9.1 Detailed Description

< Structure to contain all subdomains row and column dimension.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 partition_data_t() [1/4]

```
MpiUtils::partition_data_t::partition_data_t ( )  [inline]
```

Constructor.

5.9.2.2 partition_data_t() [2/4]

```
MpiUtils::partition_data_t::partition_data_t (
              int s,
              int r,
              int c )  [inline]
```

Constructor.

Parameters

s	Number of sub domains
r	Number of rows
c	Number of columns

5.9.2.3 partition_data_t() [3/4]

```
MpiUtils::partition_data_t::partition_data_t ( )  [inline]
```

Constructor.

5.9.2.4 partition_data_t() [4/4]

```
MpiUtils::partition_data_t::partition_data_t (
             int s,
             int r,
             int c,
             std::string t,
             int ri,
             int ci )  [inline]
```

Constructor.

Parameters

s	Number of sub domains
r	Number of rows
c	Number of columns
p	MPI partition type
ri	Number of initial rows
ci	Number of initial columns

5.9.3 Member Data Documentation

5.9.3.1 cols

```
int MpiUtils::partition_data_t::cols
```

Number of columns in main domain.

5.9.3.2 cols_ini

`int MpiUtils::partition_data_t::cols_ini`

Number of initial columns.

5.9.3.3 part_dims

`std::vector< Constants::dims_t > MpiUtils::partition_data_t::part_dims`

Vector containing all subdomains dimension.

5.9.3.4 rows

`int MpiUtils::partition_data_t::rows`

Number of rows in main domain.

5.9.3.5 rows_ini

`int MpiUtils::partition_data_t::rows_ini`

Number of initial rows.

5.9.3.6 size

`int MpiUtils::partition_data_t::size`

Number of sub domains.

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

- mpi_utils.h
- mpi_utils_old.h

5.10 SuperTimer::super_timer Class Reference

`#include <supertimer.h>`

Public Member Functions

- super_timer ()

    Constructor.
- void start (std::string category)

    It starts a custom timer.
- void stop (std::string category)

    It stops a custom timer.
- void restart (std::string category)

    It restarts a custom timer (set to zero)
- void reset ()

    It resets all timers.
- double get_total_time ()

    It calculates total time of every timer.
- double get_custom_time (std::string category)

    It calculates time value of a specific timer.
- std::string get_current_date ()

    It calculates current date.
- std::string get_hostname ()

    It calculates host name.
- int add_new_timer (std::string category)

    It helps to add a new timer.

5.10.1 Detailed Description

< Custom timer class to compute time for different operation.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 super_timer()

```
SuperTimer::super_timer::super_timer ( )
```

Constructor.

5.10.3 Member Function Documentation

5.10.3.1 add_new_timer()

```
int SuperTimer::super_timer::add_new_timer (
            std::string category )
```

It helps to add a new timer.

Parameters

category	Timer name

Returns

Timer id

5.10.3.2 get_current_date()

```
std::string SuperTimer::super_timer::get_current_date ( )
```

It calculates current date.

Returns

Current date

5.10.3.3 get_custom_time()

```
double SuperTimer::super_timer::get_custom_time (
             std::string category )
```

It calculates time value of a specific timer.

Parameters

category	Timer name

Returns

Time value

5.10.3.4 get_hostname()

```
std::string SuperTimer::super_timer::get_hostname ( )
```

It calculates host name.

Returns

Host name

5.10.3.5 get_total_time()

```
double SuperTimer::super_timer::get_total_time ( )
```

It calculates total time of every timer.

Returns

Time value

5.10.3.6 reset()

```
void SuperTimer::super_timer::reset ( )
```

It resets all timers.

5.10.3.7 restart()

```
void SuperTimer::super_timer::restart (
             std::string category )
```

It restarts a custom timer (set to zero)

Parameters

category	Timer name

5.10.3.8 start()

```
void SuperTimer::super_timer::start (
             std::string category )
```

It starts a custom timer.

Parameters

category	Timer name

5.10.3.9 stop()

```
void SuperTimer::super_timer::stop (
             std::string category )
```

It stops a custom timer.

Parameters

category	Timer name

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

- supertimer.h

5.11 Triton::triton< T > Class Template Reference

```
#include <triton.h>
```

Public Member Functions

- triton (int argc, char ∗argv[ ])

  Constructor.
- ∼triton ()

  Destructor. Releases any allocated memory.
- void initialize (int rank_, int size_)

  It initializes the simulation.
- void simulate ()

  It starts the simulation. It is the main simulation fuction.

5.11.1 Detailed Description

template<class T>
class Triton::triton< T >

< Main class to perform the simulation.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 triton()

```
template<class T >
Triton::triton< T >::triton (
             int argc,
             char * argv[ ] )
```

Constructor.

Parameters

argc	Number of arguments
argv	Arguments

5.11.2.2 ∼triton()

```
template<class T >
Triton::triton< T >::∼triton ( )
```

Destructor. Releases any allocated memory.

5.11.3 Member Function Documentation

5.11.3.1 initialize()

```
template<typename T >
void Triton::triton< T >::initialize (
             int rank_,
             int size_ )
```

It initializes the simulation.

Parameters

rank←_	Subdomain id
size←_	Number of subdomain

5.11.3.2 simulate()

```
template<typename T >
void Triton::triton< T >::simulate ( )
```

It starts the simulation. It is the main simulation fuction.

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

- triton.h

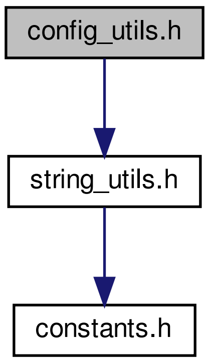
Chapter 6

File Documentation

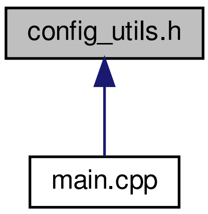
6.1 config_utils.h File Reference

Header containing the ConfigUtils class.

`#include "string_utils.h"`
Include dependency graph for config_utils.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct ConfigUtils::arguments< T >

Functions

- std::string ConfigUtils::argsd (std::string x, std::map< std::string, std::string > y, std::string d)

  It calculates the corresponding value of each attribute name from the contents of the configuration (cfg) file.
- std::string ConfigUtils::args (std::string x, std::map< std::string, std::string > y)

  It calculates the corresponding value of each attribute name from the contents of the configuration (cfg) file without any default value.
- std::map< std::string, std::string > ConfigUtils::parse_cfg (std::string cfg_content)

  It extracts the whole configuration string and constructs an attribute key-value mapping.
- std::map< std::string, std::string > ConfigUtils::parse_src_location (std::string filename, int type)

  It extracts each flow location and observation cells Longitude and Latitude value and constructs a (x,y) location mapping.
- std::map< std::string, std::string > ConfigUtils::parse_extbc_file (std::string filename, std::string dir)

  It extracts each external boundary condition file and constructs an attribute key-value mapping.
- template<typename T >
  arguments< T > ConfigUtils::get_args (std::string cfg)

  It calculates all argument values and constructs struct arguments object.
- std::string ConfigUtils::file_content_to_string (std::string filepath)

  It reads a configuration (cfg) file and constructs a string of the whole file.
- std::string ConfigUtils::get_root_dir (const char *path)

  It computes the root directory from the full path, shortening it out when a backslash is found.
- void ConfigUtils::read_and_parse_checkpoint_partition (std::string project_dir, int *dyn_rows, int checkpoint↩
  _id)

  It reads the number of rows from the output file when checkpoint is enabled.

6.1.1 Detailed Description

Header containing the ConfigUtils class.

This contains the subroutines and eventually any macros, constants, etc. needed for ConfigUtils class

Author

Mario Morales Hernandez
Md Bulbul Sharif
Tigstu T. Dullo
Sudershan Gangrade
Alfred Kalyanapu
Sheikh Ghafoor
Shih-Chieh Kao
Katherine J. Evans

Bug No known bugs.

6.1.2 Function Documentation

6.1.2.1 args()

```
std::string ConfigUtils::args (
            std::string x,
            std::map< std::string, std::string > y )
```

It calculates the corresponding value of each attribute name from the contents of the configuration (cfg) file without any default value.

Parameters

x	attibute name
y	contents of cfg file

Returns

The corresponding value

6.1.2.2 argsd()

```
std::string ConfigUtils::argsd (
            std::string x,
            std::map< std::string, std::string > y,
            std::string d )
```

It calculates the corresponding value of each attribute name from the contents of the configuration (cfg) file.

Parameters

x	attibute name
y	contents of cfg file
d	default value

Returns

The corresponding value

6.1.2.3 file_content_to_string()

```
std::string ConfigUtils::file_content_to_string (
            std::string filepath )
```

It reads a configuration (cfg) file and constructs a string of the whole file.

Parameters

filepath	file to read

Returns

Contents as a string

6.1.2.4 get_args()

```
template<typename T >
arguments< T > ConfigUtils::get_args (
             std::string cfg )
```

It calculates all argument values and constructs struct arguments object.

Parameters

cfg	file to parse

Returns

The arguments object contating all argument

6.1.2.5 get_root_dir()

```
std::string ConfigUtils::get_root_dir (
             const char * path )
```

It computes the root directory from the full path, shortening it out when a backslash is found.

Parameters

path	The full path

Returns

A string with the project directory

6.1.2.6 parse_cfg()

```
std::map< std::string, std::string > ConfigUtils::parse_cfg (
             std::string cfg_content )
```

It extracts the whole configuration string and constructs an attribute key-value mapping.

Parameters

cfg_content	cfg file content

Returns

Attribute key value mapping

6.1.2.7 parse_extbc_file()

```
std::map< std::string, std::string > ConfigUtils::parse_extbc_file (
            std::string filename,
            std::string dir )
```

It extracts each external boundary condition file and constructs an attribute key-value mapping.

Parameters

filename	file to parse
dir	parent directory of filename

Returns

Attribute key value mapping

6.1.2.8 parse_src_location()

```
std::map< std::string, std::string > ConfigUtils::parse_src_location (
            std::string filename,
            int type )
```

It extracts each flow location and observation cells Longitude and Latitude value and constructs a (x,y) location mapping.

Parameters

filename	file to parse
type	determine flow location of observation

Returns

(x,y) location mapping

6.1.2.9 read_and_parse_checkpoint_partition()

```
void ConfigUtils::read_and_parse_checkpoint_partition (
             std::string project_dir,
             int * dyn_rows,
             int checkpoint_id )
```

It reads the number of rows from the output file when checkpoint is enabled.

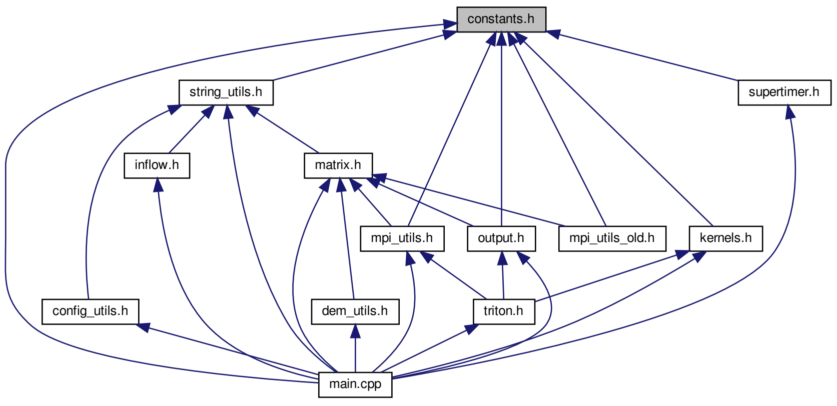
Parameters

project_dir	String containing the project directory
dyn_rows	Array of size "number of ranks" that will contain the number of rows
checkpoint←_id	Checkpoint id

6.2 constants.h File Reference

Header containing the Constants class.

This graph shows which files directly or indirectly include this file:



Macros

- #define MPI_DATA_TYPE MPI_DOUBLE
- #define MAX_VALUE DBL_MAX
- #define INPUT_DIR "input"
- #define OUTPUT_DIR "output"
- #define CFG_DIR "cfg"
- #define BIN_DIR "bin"

- #define ASCII_DIR "asc"
- #define TIME_SERIES_DIR "series"
- #define DEFAULT_CFG "case4.cfg"
- #define GHOST_CELL_PADDING 1
- #define USE_MATRIX 0
- #define USE_HALO 1
- #define SRC_LOCATION 0
- #define OBSERVATION_LOCATION 1
- #define DEM_NCOLS_LINE 1
- #define DEM_NROWS_LINE 2
- #define DEM_XLL_CORNER_LINE 3
- #define DEM_YLL_CORNER_LINE 4
- #define DEM_CELL_SIZE_LINE 5
- #define DEM_NODATA_VALUE_LINE 6
- #define DEM_HEADER_SIZE 6
- #define BIN_ROW_ID 0
- #define BIN_COL_ID 1
- #define BIN_DEFAULT_HEADER_SIZE 2
- #define H 0
- #define QX 1
- #define QY 2
- #define N 3
- #define DEM 4
- #define MAXH 5
- #define RHSH0 6
- #define RHSH1 7
- #define RHSQX0 8
- #define RHSQX1 9
- #define RHSQY0 10
- #define RHSQY1 11
- #define SQRTH 12
- #define HALO 13
- #define DT 14
- #define HYGT 15
- #define HYGV 16
- #define RUNIN 17
- #define EXTBCV1 18
- #define EXTBCV2 19
- #define SRCP 0
- #define RUNID 1
- #define BCRELATIVEINDEX 2
- #define BCTYPE 3
- #define BCINDEXSTART 4
- #define BCNROWSVARS 5
- #define TIMER_NSECS 0
- #define TIMER_SECS 1
- #define G 9.81
- #define SQRTG 3.132091953
- #define EPS12 1e-12
- #define FT3_TO_M3_FACTOR 0.028316847
- #define FT_TO_M_FACTOR 0.3048
- #define SEC_TO_HOUR_FACTOR 0.000277778
- #define HOUR_TO_SEC_FACTOR 3600.0
- #define MM_TO_M_FACTOR 0.001
- #define THREAD_BLOCK 256

- #define TOTAL_TIME "total_time"
- #define SIMULATION_TIME "simulation_time"
- #define COMPUTE_TIME "compute_time"
- #define MPI_TIME "mpi_time"
- #define IO_TIME "io_time"
- #define RESIZE_TIME "resize_time"
- #define BALANCING_MPI_TIME "balancing_mpi_time"
- #define TYPE_STATIC "static"
- #define TYPE_DYNAMIC "dynamic"
- #define RESET "\033[0m"
- #define RED "\033[31m"
- #define GREEN "\033[32m"
- #define YELLOW "\033[33m"
- #define BLUE "\033[34m"
- #define GRAY "\033[90m"
- #define OK GREEN << "[OK] " << RESET
- #define WARN YELLOW << "[!!] " << RESET
- #define ERROR RED << "[ERROR] " << RESET
- #define IN GRAY << "[..] " << RESET
- #define DASH BLUE << "[--] " << RESET
- #define **WRITE_PERFORMANCE** 0

Typedefs

- typedef std::pair< int, int > Constants::dims_t
- typedef std::vector< std::string > Constants::string_vector
- typedef std::string::value_type Constants::char_t
- typedef std::vector< std::pair< int, int > > Constants::sources_list_t
- typedef unsigned long long Constants::ull
- typedef double value_t

6.2.1 Detailed Description

Header containing the Constants class.

This contains the subroutines and eventually any macros, constants, etc. needed for Constants class

Author

Mario Morales Hernandez
Md Bulbul Sharif
Tigstu T. Dullo
Sudershan Gangrade
Alfred Kalyanapu
Sheikh Ghafoor
Shih-Chieh Kao
Katherine J. Evans

Bug No known bugs.

6.2.2 Macro Definition Documentation

6.2.2.1 ASCII_DIR

```
#define ASCII_DIR "asc"
```

Deafult folder name containing ascii files.

6.2.2.2 BALANCING_MPI_TIME

```
#define BALANCING_MPI_TIME "balancing_mpi_time"
```

Timer to get time needed for resizing and re-balancing

6.2.2.3 BCINDEXSTART

```
#define BCINDEXSTART 4
```

Boundary condition's start index array position in vector.

6.2.2.4 BCNROWSVARS

```
#define BCNROWSVARS 5
```

Boundary condition's number of rows variable array position in vector.

6.2.2.5 BCRELATIVEINDEX

```
#define BCRELATIVEINDEX 2
```

Boundary cells index array after domain decomposition position in vector.

6.2.2.6 BCTYPE

```
#define BCTYPE 3
```

Boundary condition cells type array position in vector.

6.2.2.7 BIN_COL_ID

```
#define BIN_COL_ID 1
```

Second number or index 1 in a binary output file represents number of columns.

6.2.2.8 BIN_DEFAULT_HEADER_SIZE

```
#define BIN_DEFAULT_HEADER_SIZE 2
```

Number of headers in a binary output file.

6.2.2.9 BIN_DIR

```
#define BIN_DIR "bin"
```

Default folder name containing binary files.

6.2.2.10 BIN_ROW_ID

```
#define BIN_ROW_ID 0
```

First number or index 0 in a binary output file represents number of rows.

6.2.2.11 BLUE

```
#define BLUE "\033[34m"
```

Blue Color

6.2.2.12 CFG_DIR

```
#define CFG_DIR "cfg"
```

Default folder name containing all configuration (cfg) files.

6.2.2.13 COMPUTE_TIME

```
#define COMPUTE_TIME "compute_time"
```

Timer to get computation time.

6.2.2.14 DASH

```
#define DASH BLUE << "[--] " << RESET
```

Other Message 2

6.2.2.15 DEFAULT_CFG

```
#define DEFAULT_CFG "case4.cfg"
```

Deafult configuration (cfg) file name.

6.2.2.16 DEM

```
#define DEM 4
```

DEM array position in vector.

6.2.2.17 DEM_CELL_SIZE_LINE

```
#define DEM_CELL_SIZE_LINE 5
```

Line 5 in DEM file represents cell size.

6.2.2.18 DEM_HEADER_SIZE

```
#define DEM_HEADER_SIZE 6
```

Number of headers in a DEM input file.

6.2.2.19 DEM_NCOLS_LINE

```
#define DEM_NCOLS_LINE 1
```

Line 1 in DEM file represents number of columns.

6.2.2.20 DEM_NODATA_VALUE_LINE

```
#define DEM_NODATA_VALUE_LINE 6
```

Line 6 in DEM file represents the input values to be NoData in the output raster.

6.2.2.21 DEM_NROWS_LINE

```
#define DEM_NROWS_LINE 2
```

Line 2 in DEM file represents number of rows.

6.2.2.22 DEM_XLL_CORNER_LINE

```
#define DEM_XLL_CORNER_LINE 3
```

Line 3 in DEM file represents X coordinate of the origin (by center or lower left corner of the cell).

6.2.2.23 DEM_YLL_CORNER_LINE

```
#define DEM_YLL_CORNER_LINE 4
```

Line 4 in DEM file represents Y coordinate of the origin (by center or lower left corner of the cell).

6.2.2.24 DT

```
#define DT 14
```

Reduction values container array when calculating min time step size, position in vector.

6.2.2.25 EPS12

```
#define EPS12 1e-12
```

Tolerance e-12.

6.2.2.26 ERROR

```
#define ERROR RED << "[ERROR] " << RESET
```

Error Message

6.2.2.27 EXTBCV1

```
#define EXTBCV1 18
```

External boundary condition's first variable array position in vector.

6.2.2.28 EXTBCV2

```
#define EXTBCV2 19
```

External boundary condition's second variable array position in vector.

6.2.2.29 FT3_TO_M3_FACTOR

```
#define FT3_TO_M3_FACTOR 0.028316847
```

Factor to convert feet cube to meter cube.

6.2.2.30 FT_TO_M_FACTOR

```
#define FT_TO_M_FACTOR 0.3048
```

Factor to convert feet to meter.

6.2.2.31 G

```
#define G 9.81
```

Gravitational acceleration.

6.2.2.32 GHOST_CELL_PADDING

```
#define GHOST_CELL_PADDING 1
```

Number of extra row and column to use besides each domain border.

6.2.2.33 GRAY

```
#define GRAY "\033[90m"
```

Gray Color

6.2.2.34 GREEN

```
#define GREEN "\033[32m"
```

Green Color

6.2.2.35 H

```
#define H 0
```

Water depth array position in vector.

6.2.2.36 HALO

```
#define HALO 13
```

Halo cells array position in vector.

6.2.2.37 HOUR_TO_SEC_FACTOR

```
#define HOUR_TO_SEC_FACTOR 3600.0
```

Factor to convert hour to second.

6.2.2.38 HYGT

```
#define HYGT 15
```

Time of flow values array position in vector.

6.2.2.39 HYGV

```
#define HYGV 16
```

Flow values array position in vector.

6.2.2.40 IN

```
#define IN GRAY << "[..] " << RESET
```

Other Message 1

6.2.2.41 INPUT_DIR

```
#define INPUT_DIR "input"
```

Deafult folder name containing all input files.

6.2.2.42 IO_TIME

```
#define IO_TIME "io_time"
```

Timer to get time needed for outputting in file.

6.2.2.43 MAX_VALUE

```
#define MAX_VALUE DBL_MAX
```

Maximum value of a floating-point number. It can be DBL_MAX or FLT_MAX.

6.2.2.44 MAXH

```
#define MAXH 5
```

Max values of water depth array position in vector.

6.2.2.45 MM_TO_M_FACTOR

```
#define MM_TO_M_FACTOR 0.001
```

Factor to convert mili meter to meter.

6.2.2.46 MPI_DATA_TYPE

```
#define MPI_DATA_TYPE MPI_DOUBLE
```

Represents MPI floating-point number. It can be MPI_DOUBLE or MPI_FLOAT.

6.2.2.47 MPI_TIME

```
#define MPI_TIME "mpi_time"
```

Timer to get all MPI operation time.

6.2.2.48 N

```
#define N 3
```

Manning array position in vector.

6.2.2.49 OBSERVATION_LOCATION

```
#define OBSERVATION_LOCATION 1
```

Define to use observation cells.

6.2.2.50 OK

```
#define OK GREEN << "[OK] " << RESET
```

Success Message

6.2.2.51 OUTPUT_DIR

```
#define OUTPUT_DIR "output"
```

Deafult folder name containing all output files.

6.2.2.52 QX

```
#define QX 1
```

Flux X array position in vector.

6.2.2.53 QY

```
#define QY 2
```

Flux Y array position in vector.

6.2.2.54 RED

```
#define RED "\033[31m"
```

Red Color

6.2.2.55 RESET

```
#define RESET "\033[0m"
```

Black Color

6.2.2.56 RESIZE_TIME

```
#define RESIZE_TIME "resize_time"
```

Timer to get time needed for resizing and re-balancing

6.2.2.57 RHSH0

```
#define RHSH0 6
```

Partial water depth 1 array position in vector.

6.2.2.58 RHSH1

```
#define RHSH1 7
```

Partial water depth 2 array position in vector.

6.2.2.59 RHSQX0

```
#define RHSQX0 8
```

Partial flux X 1 array position in vector.

6.2.2.60 RHSQX1

```
#define RHSQX1 9
```

Partial flux X 2 array position in vector.

6.2.2.61 RHSQY0

```
#define RHSQY0 10
```

Partial flux Y 1 array position in vector.

6.2.2.62 RHSQY1

```
#define RHSQY1 11
```

Partial flux Y 2 array position in vector.

6.2.2.63 RUNID

```
#define RUNID 1
```

Runoff id array position in vector.

6.2.2.64 RUNIN

```
#define RUNIN 17
```

Runoff intensity array position in vector.

6.2.2.65 SEC_TO_HOUR_FACTOR

```
#define SEC_TO_HOUR_FACTOR 0.000277778
```

Factor to convert second to hour.

6.2.2.66 SIMULATION_TIME

```
#define SIMULATION_TIME "simulation_time"
```

Timer to get only the simulation time.

6.2.2.67 SQRTG

```
#define SQRTG 3.132091953
```

Square root of Gravitational acceleration.

6.2.2.68 SQRTH

```
#define SQRTH 12
```

Square root of water depth array position in vector.

6.2.2.69 SRC_LOCATION

```
#define SRC_LOCATION 0
```

Define to use flow locations.

6.2.2.70 SRCP

```
#define SRCP 0
```

Flow locations index array position in vector.

6.2.2.71 THREAD_BLOCK

```
#define THREAD_BLOCK 256
```

Thread block size to use in CUDA.

6.2.2.72 TIME_SERIES_DIR

```
#define TIME_SERIES_DIR "series"
```

Deafult folder name containing time series outputs.

6.2.2.73 TIMER_NSECS

```
#define TIMER_NSECS 0
```

To use nano second in Timer.

6.2.2.74 TIMER_SECS

```
#define TIMER_SECS 1
```

To use second in Timer.

6.2.2.75 TOTAL_TIME

```
#define TOTAL_TIME "total_time"
```

Timer to get total runtime of the program.

6.2.2.76 TYPE_DYNAMIC

```
#define TYPE_DYNAMIC "dynamic"
```

Domain decomposition type: dynamic

6.2.2.77 TYPE_STATIC

```
#define TYPE_STATIC "static"
```

Domain decomposition type: static

6.2.2.78 USE_HALO

```
#define USE_HALO 1
```

Use only halo rows bundle when performing MPI halo exchange.

6.2.2.79 USE_MATRIX

```
#define USE_MATRIX 0
```

Use the whole matrix when performing MPI halo exchange.

6.2.2.80 WARN

```
#define WARN YELLOW << "[!!] " << RESET
```

Warning Message

6.2.2.81 YELLOW

```
#define YELLOW "\033[33m"
```

Yellow Color

6.2.3 Typedef Documentation

6.2.3.1 char_t

```
typedef std::string::value_type Constants::char_t
```

Custom type used for string utility.

6.2.3.2 dims_t

```
typedef std::pair<int, int> Constants::dims_t
```

Custom type to define dimension. The first number represents the rows and the second number is the columns.

6.2.3.3 sources_list_t

```
typedef std::vector<std::pair<int, int> > Constants::sources_list_t
```

Custom vector type that contains each cell's index pair. The first number is the row index and the second number is the column index.

6.2.3.4 string_vector

```
typedef std::vector<std::string> Constants::string_vector
```

Custom vector type that represents a vector of strings.

6.2.3.5 ull

```
typedef unsigned long long Constants::ull
```

Custom data type to hold large number.

6.2.3.6 value_t

```
typedef double value_t
```

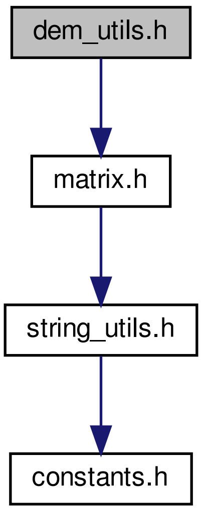
Data type to represent floating-point number. It can be double or float.

6.3 dem_utils.h File Reference

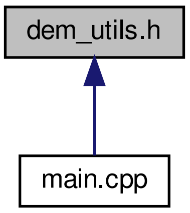
Header containing the DemFile class.

```
#include "matrix.h"
```

Include dependency graph for dem_utils.h:



This graph shows which files directly or indirectly include this file:



Classes

- class DemFile::dem_file< T >

6.3.1 Detailed Description

Header containing the DemFile class.

This contains the subroutines and eventually any macros, constants, etc. needed for DemFile class

Author

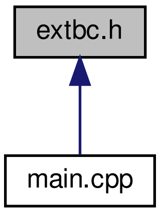
Mario Morales Hernandez
Md Bulbul Sharif
Tigstu T. Dullo
Sudershan Gangrade
Alfred Kalyanapu
Sheikh Ghafoor
Shih-Chieh Kao
Katherine J. Evans

Bug No known bugs.

6.4 extbc.h File Reference

Header containing the ExtBC class.

This graph shows which files directly or indirectly include this file:



Classes

- class ExtBC::extBC< T >

6.4.1 Detailed Description

Header containing the ExtBC class.

This contains the subroutines and eventually any macros, constants, etc. needed for ExtBC class

Author

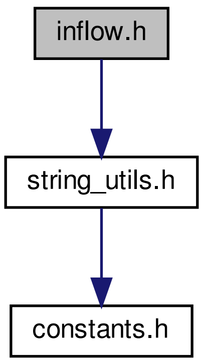
Mario Morales Hernandez
Md Bulbul Sharif
Tigstu T. Dullo
Sudershan Gangrade
Alfred Kalyanapu
Sheikh Ghafoor
Shih-Chieh Kao
Katherine J. Evans

Bug No known bugs.

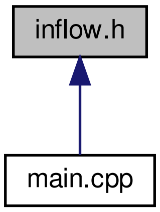
6.5 inflow.h File Reference

Header containing the Hydrograph class.

`#include "string_utils.h"`
Include dependency graph for inflow.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Hydrograph::hydrograph< T >

6.5.1 Detailed Description

Header containing the Hydrograph class.

This contains the subroutines and eventually any macros, constants, etc. needed for Hydrograph class

Author

Mario Morales Hernandez
Md Bulbul Sharif
Tigstu T. Dullo
Sudershan Gangrade
Alfred Kalyanapu
Sheikh Ghafoor
Shih-Chieh Kao
Katherine J. Evans

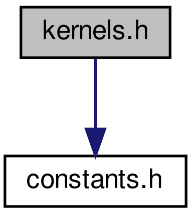
Bug No known bugs.

6.6 kernels.h File Reference

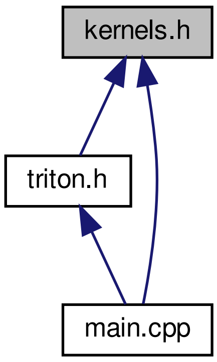
Header containing the Kernels class.

```
#include "constants.h"
```

Include dependency graph for kernels.h:



This graph shows which files directly or indirectly include this file:



6.6.1 Detailed Description

Header containing the Kernels class.

This contains the subroutines and eventually any macros, constants, etc. needed for Kernels class

Author

Mario Morales Hernandez
Md Bulbul Sharif
Tigstu T. Dullo
Sudershan Gangrade
Alfred Kalyanapu
Sheikh Ghafoor
Shih-Chieh Kao
Katherine J. Evans

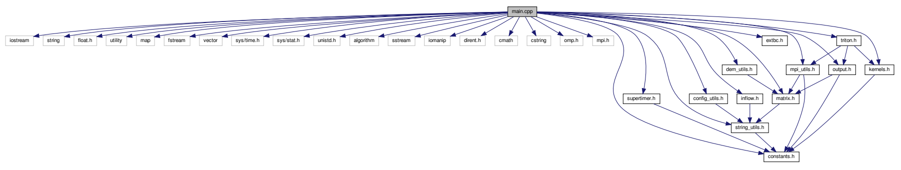
Bug No known bugs.

6.7 main.cpp File Reference

Main file containing the driver.

```
#include <iostream>
#include <string>
#include <float.h>
#include <utility>
#include <map>
#include <fstream>
#include <vector>
#include <sys/time.h>
#include <sys/stat.h>
#include <unistd.h>
#include <algorithm>
#include <sstream>
#include <iomanip>
#include <dirent.h>
#include <cmath>
#include <cstring>
#include <omp.h>
#include "mpi.h"
#include "constants.h"
#include "supertimer.h"
#include "string_utils.h"
#include "mpi_utils.h"
#include "config_utils.h"
#include "inflow.h"
#include "extbc.h"
#include "matrix.h"
#include "dem_utils.h"
#include "output.h"
#include "triton.h"
#include "kernels.h"
```

Include dependency graph for main.cpp:



Functions

- int main (int argc, char ∗argv[ ])

    Main function. This is the main function of the program.

6.7.1 Detailed Description

Main file containing the driver.

This contains the subroutines and eventually any macros, constants, etc. needed for the driver

Author

Mario Morales Hernandez
Md Bulbul Sharif
Tigstu T. Dullo
Sudershan Gangrade
Alfred Kalyanapu
Sheikh Ghafoor
Shih-Chieh Kao
Katherine J. Evans

Bug No known bugs.

6.7.2 Function Documentation

6.7.2.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Main function. This is the main function of the program.

Parameters

argc	Argument count
argv	Pointer array which points to each argument passed to the program. The program runs with cfg filename and number of threads (only for OpenMP version)

Returns

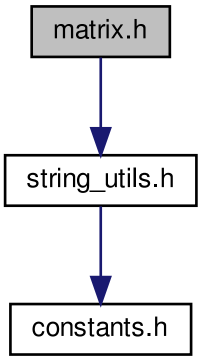
0

6.8 matrix.h File Reference

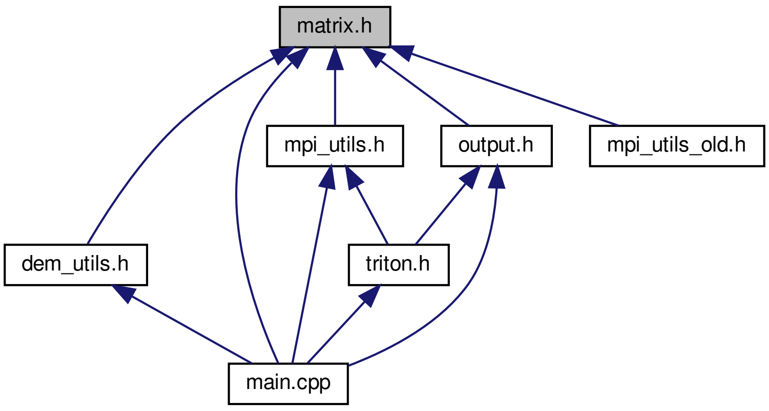
Header containing the Matrix class.

```
#include "string_utils.h"
```

Include dependency graph for matrix.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Matrix::matrix< T >

6.8.1 Detailed Description

Header containing the Matrix class.

This contains the subroutines and eventually any macros, constants, etc. needed for Matrix class

Author

Mario Morales Hernandez
Md Bulbul Sharif
Tigstu T. Dullo
Sudershan Gangrade
Alfred Kalyanapu
Sheikh Ghafoor
Shih-Chieh Kao
Katherine J. Evans

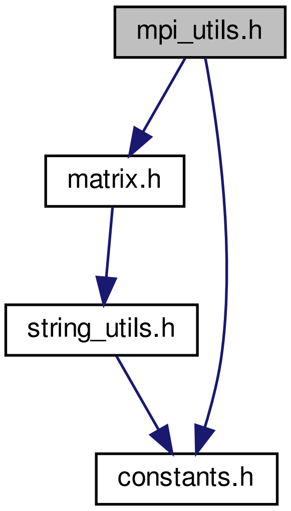
Bug No known bugs.

6.9 mpi_utils.h File Reference

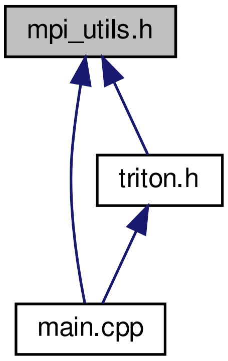
Header containing the MpiUtils class.

```
#include "matrix.h"
#include "constants.h"
```

Include dependency graph for mpi_utils.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct MpiUtils::partition_data_t

Functions

- Constants::dims_t MpiUtils::create_local_dims (int globalrows, int globalcols, int rank, int size)

    It calculates each subdomain's number of rows and columns.

- template<typename T >
  void MpiUtils::exchange (T ∗local, int lrows, int lcols, int rank, int size, int type)

    It performs MPI halo exchanges between multiple MPI processes.

- template<typename T >
  Matrix::matrix< T > MpiUtils::scatter_exchange (T ∗global, partition_data_t pd, int rank)

    It performs initial domain scattering and partitioning between multiple MPI processes.

- Matrix::matrix< int > MpiUtils::scatter_exchange_int (int ∗global, partition_data_t pd, int rank)

    It performs initial domain scattering and partitioning between multiple MPI processes for integer data type.

6.9.1 Detailed Description

Header containing the MpiUtils class.

This contains the subroutines and eventually any macros, constants, etc. needed for MpiUtils class

Author

Mario Morales Hernandez
Md Bulbul Sharif
Tigstu T. Dullo
Sudershan Gangrade
Alfred Kalyanapu
Sheikh Ghafoor
Shih-Chieh Kao
Katherine J. Evans

Bug No known bugs.

This contains the subroutines and eventually any macros, constants, etc. needed for MpiUtils class

Author

Mario Morales Hernandez
Md Bulbul Sharif
Tigstu T. Dullo
Sudershan Gangrade
Alfred Kalyanapu
Sheikh Ghafoor
Shih-Chieh Kao
Katherine J. Evans

Bug No known bugs.

6.9.2 Function Documentation

6.9.2.1 create_local_dims()

```
Constants::dims_t MpiUtils::create_local_dims (
            int globalrows,
            int globalcols,
            int rank,
            int size )
```

It calculates each subdomain's number of rows and columns.

Parameters

globalrows	Row count of main domain
globalcols	Column count of main domain
rank	Current sub domain id
size	Total number of sub domain

Returns

Row and column dimension

6.9.2.2 exchange()

```
template<typename T >
void MpiUtils::exchange (
             T * local,
             int lrows,
             int lcols,
             int rank,
             int size,
             int type )
```

It performs MPI halo exchanges between multiple MPI processes.

Parameters

local	Data to use for halo exchange
lrows	Number of rows
lcols	Number of columns
rank	Current MPI process id
size	Total number of MPI processes
type	Data type (Only halo data/Full domain data)

6.9.2.3 scatter_exchange()

```
template<typename T >
Matrix::matrix< T > MpiUtils::scatter_exchange (
             T * global,
             partition_data_t pd,
             int rank )
```

It performs initial domain scattering and partitioning between multiple MPI processes.

Parameters

global	Data of the whole domain
pd	Partitioning information
rank	Current MPI process id

Returns

Subdomain data

6.9.2.4 scatter_exchange_int()

```
Matrix::matrix< int > MpiUtils::scatter_exchange_int (
            int * global,
            partition_data_t pd,
            int rank )
```

It performs initial domain scattering and partitioning between multiple MPI processes for integer data type.

Parameters

global	Data of the whole domain
pd	Partitioning information
rank	Current MPI process id

Returns

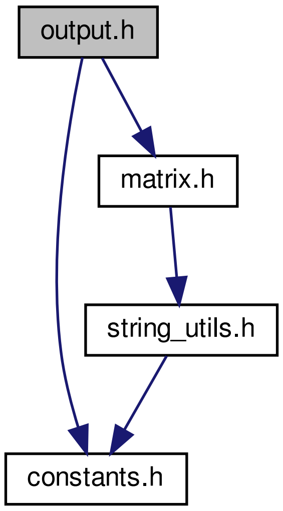
Subdomain data

6.10 output.h File Reference

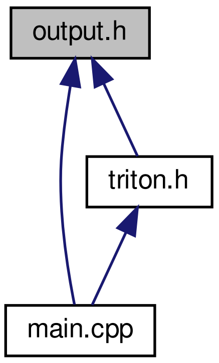
Header containing the Output class.

```
#include "constants.h"
#include "matrix.h"
```

Include dependency graph for output.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Output::output< T >

Functions

- template<typename T >
  std::ostream & **Output::operator**<< (std::ostream &out, Matrix::matrix< T > &M)

6.10.1 Detailed Description

Header containing the Output class.

This contains the subroutines and eventually any macros, constants, etc. needed for Output class

Author

Mario Morales Hernandez
Md Bulbul Sharif
Tigstu T. Dullo
Sudershan Gangrade
Alfred Kalyanapu
Sheikh Ghafoor
Shih-Chieh Kao
Katherine J. Evans

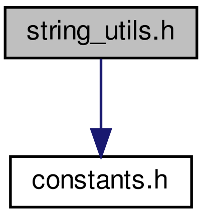
Bug No known bugs.

6.11 string_utils.h File Reference

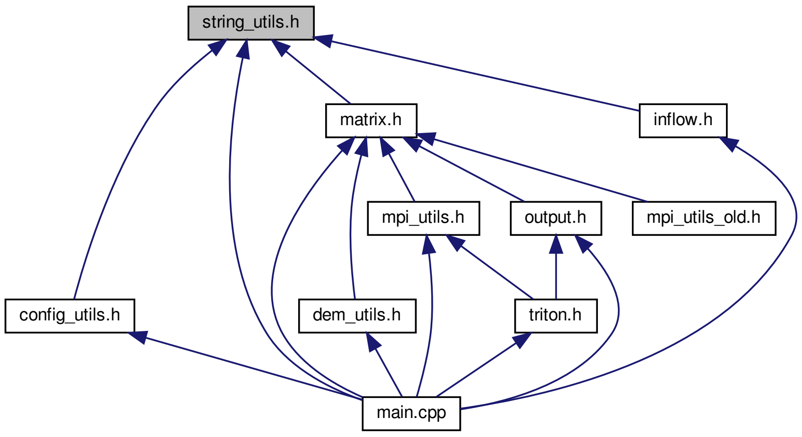
Header containing the StringUtils class.

```
#include "constants.h"
```

Include dependency graph for string_utils.h:



This graph shows which files directly or indirectly include this file:



Functions

- Constants::char_t StringUtils::up_char (Constants::char_t ch)

  It capitalizes a char_t type variable.
- std::string StringUtils::toupper (const std::string &src)

  It capitalizes every char of a string.
- Constants::char_t StringUtils::down_char (Constants::char_t ch)

  It converts a char_t type variable into lower case.
- std::string StringUtils::tolower (const std::string &src)

  It converts every char of a string into lower case.

- bool StringUtils::is_numeric (const std::string &str)

  It determines a string is a numeric number or not.
- Constants::string_vector & StringUtils::split (const std::string &s, char delim, Constants::string_vector &elems)

  It splits a string by a char delimeter.
- Constants::string_vector StringUtils::split (const std::string &s, char delim)

  It splits a string by a char delimeter.
- std::vector< int > StringUtils::vecstr_to_vecint (std::vector< std::string > vs)

  It converts every element of a string vector into an integer vector.
- template<typename T >
  std::vector< T > StringUtils::vecstr_to_vecflt (Constants::string_vector vs)

  It converts every element of a string vector into an floating point vector.
- std::string StringUtils::itoa (int i)

  It converts a integer to string.
- std::string StringUtils::itos (int num)

  It converts a integer to string.

6.11.1 Detailed Description

Header containing the StringUtils class.

This contains the subroutines and eventually any macros, constants, etc. needed for StringUtils class

Author

Mario Morales Hernandez
Md Bulbul Sharif
Tigstu T. Dullo
Sudershan Gangrade
Alfred Kalyanapu
Sheikh Ghafoor
Shih-Chieh Kao
Katherine J. Evans

Bug No known bugs.

6.11.2 Function Documentation

6.11.2.1 down_char()

```
Constants::char_t StringUtils::down_char (
            Constants::char_t ch )
```

It converts a char_t type variable into lower case.

Parameters

ch	char_t type variable

Returns

Lower case value

6.11.2.2 is_numeric()

```
bool StringUtils::is_numeric (
             const std::string & str )
```

It determines a string is a numeric number or not.

Parameters

src	String

Returns

True or False

6.11.2.3 itoa()

```
std::string StringUtils::itoa (
             int i )
```

It converts a integer to string.

Parameters

i	Integer number

Returns

String

6.11.2.4 itos()

```
std::string StringUtils::itos (
             int num )
```

It converts a integer to string.

Parameters

num	Integer number

Returns

String

6.11.2.5 split() [1/2]

```
Constants::string_vector & StringUtils::split (
             const std::string & s,
             char delim,
             Constants::string_vector & elems )
```

It splits a string by a char delimeter.

Parameters

s	String
delim	Char delimeter
elems	String vector

Returns

Splited string vector

6.11.2.6 split() [2/2]

```
Constants::string_vector StringUtils::split (
             const std::string & s,
             char delim )
```

It splits a string by a char delimeter.

Parameters

s	String
delim	Char delimeter

Returns

Splited string vector

6.11.2.7 tolower()

```
std::string StringUtils::tolower (
             const std::string & src )
```

It converts every char of a string into lower case.

Parameters

src	String

Returns

Lower case string

6.11.2.8 toupper()

```
std::string StringUtils::toupper (
             const std::string & src )
```

It capitalizes every char of a string.

Parameters

src	String

Returns

Capitalized string

6.11.2.9 up_char()

```
Constants::char_t StringUtils::up_char (
             Constants::char_t ch )
```

It capitalizes a char_t type variable.

Parameters

ch	char_t type variable

Returns

Capitalized value

6.11.2.10 vecstr_to_vecflt()

```
template<typename T >
std::vector< T > StringUtils::vecstr_to_vecflt (
            Constants::string_vector vs )
```

It converts every element of a string vector into an floating point vector.

Parameters

vs	String vector

Returns

Floating point vector

6.11.2.11 vecstr_to_vecint()

```
std::vector< int > StringUtils::vecstr_to_vecint (
            std::vector< std::string > vs )
```

It converts every element of a string vector into an integer vector.

Parameters

vs	String vector

Returns

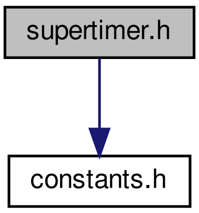
Integer vector

6.12 supertimer.h File Reference

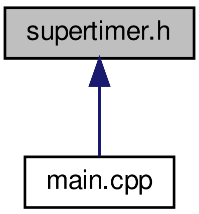
Header containing the SuperTimer class.

```
#include "constants.h"
```

Include dependency graph for supertimer.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct SuperTimer::ci_less
- struct SuperTimer::ci_less::nocase_compare
- class SuperTimer::super_timer

6.12.1 Detailed Description

Header containing the SuperTimer class.

This contains the subroutines and eventually any macros, constants, etc. needed for SuperTimer class

Author

Mario Morales Hernandez
Md Bulbul Sharif
Tigstu T. Dullo
Sudershan Gangrade
Alfred Kalyanapu
Sheikh Ghafoor
Shih-Chieh Kao
Katherine J. Evans

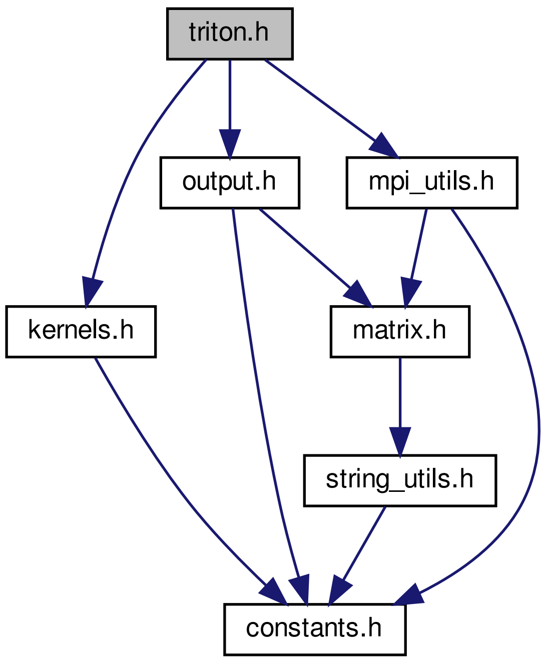
Bug No known bugs.

6.13 triton.h File Reference

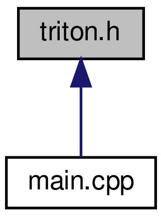
Header containing the Triton class.

```
#include "kernels.h"
#include "output.h"
#include "mpi_utils.h"
```

Include dependency graph for triton.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Triton::triton< T >

6.13.1 Detailed Description

Header containing the Triton class.

This contains the subroutines and eventually any macros, constants, etc. needed for Triton class

Author

Mario Morales Hernandez
Md Bulbul Sharif
Tigstu T. Dullo
Sudershan Gangrade
Alfred Kalyanapu
Sheikh Ghafoor
Shih-Chieh Kao
Katherine J. Evans

Bug No known bugs.

Index

∼matrix
    Matrix::matrix, 39
∼output
    Output::output, 54
∼triton
    Triton::triton, 68

ASCII_DIR
    constants.h, 77
add_ghost_cells
    Matrix::matrix, 39
add_new_timer
    SuperTimer::super_timer, 64
args
    config_utils.h, 70
argsd
    config_utils.h, 71
average
    Output::output, 54

BALANCING_MPI_TIME
    constants.h, 77
BCINDEXSTART
    constants.h, 77
BCNROWSVARS
    constants.h, 77
BCRELATIVEINDEX
    constants.h, 77
BCTYPE
    constants.h, 77
BIN_COL_ID
    constants.h, 77
BIN_DEFAULT_HEADER_SIZE
    constants.h, 77
BIN_DIR
    constants.h, 78
BIN_ROW_ID
    constants.h, 78
BLUE
    constants.h, 78
begin
    Matrix::matrix, 39

CFG_DIR
    constants.h, 78
COMPUTE_TIME
    constants.h, 78
char_t
    constants.h, 87
check_extreme_extbc
    ExtBC::extBC, 25
checkpoint_id
    ConfigUtils::arguments, 10
cols
    MpiUtils::partition_data_t, 62
cols_ini
    MpiUtils::partition_data_t, 62
config_utils.h, 69
    args, 70
    argsd, 71
    file_content_to_string, 71
    get_args, 72
    get_root_dir, 72
    parse_cfg, 72
    parse_extbc_file, 73
    parse_src_location, 73
    read_and_parse_checkpoint_partition, 74
ConfigUtils::arguments
    checkpoint_id, 10
    const_mann, 10
    courant, 10
    dem_filename, 10
    domain_decomposition, 11
    extbc_bctype, 11
    extbc_dir, 11
    extbc_file, 11
    extbc_fname, 11
    extbc_x1_loc, 11
    extbc_x2_loc, 11
    extbc_y1_loc, 12
    extbc_y2_loc, 12
    factor_interval_domain_decomposition, 12
    gpu_direct_flag, 12
    h_infile, 12
    hextra, 12
    hydrograph_filename, 12
    input_format, 13
    it_count, 13
    max_value_print_option, 13
    n_infile, 13
    num_extbc, 13
    num_runoffs, 13
    num_sources, 13
    observation_loc_file, 14
    observation_x_loc, 14
    observation_y_loc, 14
    outfile_pattern, 14
    output_format, 14
    output_option, 14

print_interval, 14
print_option, 15
qx_infile, 15
qy_infile, 15
runoff_filename, 15
runoff_map, 15
sim_duration, 15
sim_start_time, 15
src_loc_file, 16
src_x_loc, 16
src_y_loc, 16
time_increment_fixed, 16
time_series_flag, 16
time_step, 16
ConfigUtils::arguments< T >, 9
const_mann
ConfigUtils::arguments, 10
constants.h, 74
ASCII_DIR, 77
BALANCING_MPI_TIME, 77
BCINDEXSTART, 77
BCNROWSVARS, 77
BCRELATIVEINDEX, 77
BCTYPE, 77
BIN_COL_ID, 77
BIN_DEFAULT_HEADER_SIZE, 77
BIN_DIR, 78
BIN_ROW_ID, 78
BLUE, 78
CFG_DIR, 78
COMPUTE_TIME, 78
char_t, 87
DASH, 78
DEFAULT_CFG, 78
DEM_CELL_SIZE_LINE, 79
DEM_HEADER_SIZE, 79
DEM_NCOLS_LINE, 79
DEM_NODATA_VALUE_LINE, 79
DEM_NROWS_LINE, 79
DEM_XLL_CORNER_LINE, 79
DEM_YLL_CORNER_LINE, 79
DEM, 78
dims_t, 87
DT, 79
EPS12, 80
ERROR, 80
EXTBCV1, 80
EXTBCV2, 80
FT3_TO_M3_FACTOR, 80
FT_TO_M_FACTOR, 80
G, 80
GHOST_CELL_PADDING, 80
GRAY, 81
GREEN, 81
H, 81
HALO, 81
HOUR_TO_SEC_FACTOR, 81
HYGT, 81
HYGV, 81
INPUT_DIR, 82
IO_TIME, 82
IN, 81
MAX_VALUE, 82
MAXH, 82
MM_TO_M_FACTOR, 82
MPI_DATA_TYPE, 82
MPI_TIME, 82
N, 82
OBSERVATION_LOCATION, 83
OUTPUT_DIR, 83
OK, 83
QX, 83
QY, 83
RESET, 83
RESIZE_TIME, 83
RED, 83
RHSH0, 84
RHSH1, 84
RHSQX0, 84
RHSQX1, 84
RHSQY0, 84
RHSQY1, 84
RUNID, 84
RUNIN, 84
SEC_TO_HOUR_FACTOR, 85
SIMULATION_TIME, 85
SQRTG, 85
SQRTH, 85
SRC_LOCATION, 85
SRCP, 85
sources_list_t, 87
string_vector, 87
THREAD_BLOCK, 85
TIME_SERIES_DIR, 85
TIMER_NSECS, 86
TIMER_SECS, 86
TOTAL_TIME, 86
TYPE_DYNAMIC, 86
TYPE_STATIC, 86
USE_HALO, 86
USE_MATRIX, 86
ull, 87
value_t, 87
WARN, 86
YELLOW, 87
convert_rate_hr_to_secs
Hydrograph::hydrograph, 32
convert_rate_mm_to_m
Hydrograph::hydrograph, 32
convert_time_hr_to_secs
Hydrograph::hydrograph, 32
convert_to_secs
ExtBC::extBC, 27
copy_elevation_into_ghost_cells
Matrix::matrix, 40
copy_value_into_ghost_cells

Matrix::matrix, 40
courant
ConfigUtils::arguments, 10
create_involved_cells
ExtBC::extBC, 27
create_local_dims
mpi_utils.h, 98
cur_proc_data_size
Output::output, 60

DASH
constants.h, 78
DEFAULT_CFG
constants.h, 78
DEM_CELL_SIZE_LINE
constants.h, 79
DEM_HEADER_SIZE
constants.h, 79
DEM_NCOLS_LINE
constants.h, 79
DEM_NODATA_VALUE_LINE
constants.h, 79
DEM_NROWS_LINE
constants.h, 79
DEM_XLL_CORNER_LINE
constants.h, 79
DEM_YLL_CORNER_LINE
constants.h, 79
DEM
constants.h, 78
dem_file
DemFile::dem_file, 19, 20
dem_filename
ConfigUtils::arguments, 10
dem_utils.h, 88
DemFile::dem_file
dem_file, 19, 20
get_cell_size, 20
get_ncols, 20
get_no_data_value, 20
get_nrows, 21
get_xll_corner, 21
get_yll_corner, 21
load_header_from_dem_file_ascii, 21
load_header_from_dem_file_binary, 22
set_cell_size, 22
set_ncols, 22
set_no_data_value, 22
set_nrows, 23
set_xll_corner, 23
set_yll_corner, 23
DemFile::dem_file< T >, 18
dims_t
constants.h, 87
displs
Output::output, 60
domain_decomposition
ConfigUtils::arguments, 11
down_char
string_utils.h, 103
DT
constants.h, 79

EPS12
constants.h, 80
ERROR
constants.h, 80
EXTBCV1
constants.h, 80
EXTBCV2
constants.h, 80
exchange
mpi_utils.h, 99
ExtBC::extBC< T >, 24
ExtBC::extBC
check_extreme_extbc, 25
convert_to_secs, 27
create_involved_cells, 27
extBC, 25
extreme_cols, 29
extreme_rows, 29
get_num_rows, 27
get_rows, 28
get_var1_at, 28
get_var2_at, 28
i_cols, 30
i_rows, 30
load_from_file, 29
location, 30
ncells, 30
ncells_local, 30
set_num_rows, 29
extBC
ExtBC::extBC, 25
extbc.h, 89
extbc_bctype
ConfigUtils::arguments, 11
extbc_dir
ConfigUtils::arguments, 11
extbc_file
ConfigUtils::arguments, 11
extbc_fname
ConfigUtils::arguments, 11
extbc_x1_loc
ConfigUtils::arguments, 11
extbc_x2_loc
ConfigUtils::arguments, 11
extbc_y1_loc
ConfigUtils::arguments, 12
extbc_y2_loc
ConfigUtils::arguments, 12
extreme_cols
ExtBC::extBC, 29
extreme_rows
ExtBC::extBC, 29

FT3_TO_M3_FACTOR
constants.h, 80

FT_TO_M_FACTOR
    constants.h, 80
factor_interval_domain_decomposition
    ConfigUtils::arguments, 12
file_content_to_string
    config_utils.h, 71

G
    constants.h, 80
GHOST_CELL_PADDING
    constants.h, 80
GRAY
    constants.h, 81
GREEN
    constants.h, 81
get_address_at
    Matrix::matrix, 40
get_args
    config_utils.h, 72
get_cell_size
    DemFile::dem_file, 20
get_current_date
    SuperTimer::super_timer, 65
get_custom_time
    SuperTimer::super_timer, 65
get_data
    Matrix::matrix, 41
get_dims_2d
    Matrix::matrix, 41
get_flow_at
    Hydrograph::hydrograph, 32
get_ghost_ncols
    Matrix::matrix, 41
get_ghost_nrows
    Matrix::matrix, 41
get_hostname
    SuperTimer::super_timer, 65
get_mat_path
    Output::output, 54
get_ncols
    DemFile::dem_file, 20
get_no_data_value
    DemFile::dem_file, 20
get_nrows
    DemFile::dem_file, 21
get_num_cols
    Matrix::matrix, 42
get_num_inflow_rows
    Hydrograph::hydrograph, 33
get_num_inflows
    Hydrograph::hydrograph, 33
get_num_rows
    ExtBC::extBC, 27
    Matrix::matrix, 42
get_root_dir
    config_utils.h, 72
get_rows
    ExtBC::extBC, 28
    Hydrograph::hydrograph, 33
get_time_at
    Hydrograph::hydrograph, 34
get_total_time
    SuperTimer::super_timer, 65
get_value
    Matrix::matrix, 42, 43
get_var1_at
    ExtBC::extBC, 28
get_var2_at
    ExtBC::extBC, 28
get_xll_corner
    DemFile::dem_file, 21
get_yll_corner
    DemFile::dem_file, 21
gpu_direct_flag
    ConfigUtils::arguments, 12

H
    constants.h, 81
h_infile
    ConfigUtils::arguments, 12
HALO
    constants.h, 81
HOUR_TO_SEC_FACTOR
    constants.h, 81
HYGT
    constants.h, 81
HYGV
    constants.h, 81
hextra
    ConfigUtils::arguments, 12
hydrograph
    Hydrograph::hydrograph, 31, 32
Hydrograph::hydrograph
    convert_rate_hr_to_secs, 32
    convert_rate_mm_to_m, 32
    convert_time_hr_to_secs, 32
    get_flow_at, 32
    get_num_inflow_rows, 33
    get_num_inflows, 33
    get_rows, 33
    get_time_at, 34
    hydrograph, 31, 32
    load_from_file, 34
    set_num_flow_rows, 34
    set_num_sources, 35
Hydrograph::hydrograph< T >, 31
hydrograph_filename
    ConfigUtils::arguments, 12

i_cols
    ExtBC::extBC, 30
i_rows
    ExtBC::extBC, 30
INPUT_DIR
    constants.h, 82
IO_TIME
    constants.h, 82
IN

constants.h, 81
inflow.h, 90
init
Output::output, 55
init_time_series
Output::output, 55
initialize
Triton::triton, 68
input_format
ConfigUtils::arguments, 13
is_inbounds
Matrix::matrix, 43
is_numeric
string_utils.h, 104
it_count
ConfigUtils::arguments, 13
itoa
string_utils.h, 104
itos
string_utils.h, 104

kernels.h, 91

load_from_ascii_file
Matrix::matrix, 44, 45
load_from_binary_file
Matrix::matrix, 45
load_from_file
ExtBC::extBC, 29
Hydrograph::hydrograph, 34
load_header_from_dem_file_ascii
DemFile::dem_file, 21
load_header_from_dem_file_binary
DemFile::dem_file, 22
location
ExtBC::extBC, 30

MAX_VALUE
constants.h, 82
MAXH
constants.h, 82
MM_TO_M_FACTOR
constants.h, 82
MPI_DATA_TYPE
constants.h, 82
MPI_TIME
constants.h, 82
main
main.cpp, 94
main.cpp, 93
main, 94
matrix
Matrix::matrix, 38
matrix.h, 94
Matrix::matrix
∼matrix, 39
add_ghost_cells, 39
begin, 39
copy_elevation_into_ghost_cells, 40
copy_value_into_ghost_cells, 40
get_address_at, 40
get_data, 41
get_dims_2d, 41
get_ghost_ncols, 41
get_ghost_nrows, 41
get_num_cols, 42
get_num_rows, 42
get_value, 42, 43
is_inbounds, 43
load_from_ascii_file, 44, 45
load_from_binary_file, 45
matrix, 38
operator∗, 46, 47
operator∗=, 47
operator(), 46
operator+, 47
operator+=, 48
operator=, 48
pow, 49
remove_ghost_cells, 49
resize, 49
set_infinite_walls, 49
set_size, 50
set_value, 50, 51
square, 51
zero_fill, 51
zero_fill_int, 51
Matrix::matrix< T >, 35
max_value_print_option
ConfigUtils::arguments, 13
mpi_utils.h, 96
create_local_dims, 98
exchange, 99
scatter_exchange, 99
scatter_exchange_int, 99
MpiUtils::partition_data_t, 60
cols, 62
cols_ini, 62
part_dims, 63
partition_data_t, 61, 62
rows, 63
rows_ini, 63
size, 63

N
constants.h, 82
n_infile
ConfigUtils::arguments, 13
ncells
ExtBC::extBC, 30
ncells_local
ExtBC::extBC, 30
num_extbc
ConfigUtils::arguments, 13
num_runoffs
ConfigUtils::arguments, 13
num_sources
ConfigUtils::arguments, 13

OBSERVATION_LOCATION
constants.h, 83
OUTPUT_DIR
constants.h, 83
observation_loc_file
ConfigUtils::arguments, 14
observation_x_loc
ConfigUtils::arguments, 14
observation_y_loc
ConfigUtils::arguments, 14
OK
constants.h, 83
operator∗
Matrix::matrix, 46, 47
operator∗=
Matrix::matrix, 47
operator()
Matrix::matrix, 46
SuperTimer::ci_less, 17
SuperTimer::ci_less::nocase_compare, 52
operator+
Matrix::matrix, 47
operator+=
Matrix::matrix, 48
operator=
Matrix::matrix, 48
outfile_pattern
ConfigUtils::arguments, 14
output
Output::output, 54
output.h, 100
Output::output
∼output, 54
average, 54
cur_proc_data_size, 60
displs, 60
get_mat_path, 54
init, 55
init_time_series, 55
output, 54
output_cfg, 56
output_time_series, 56
recvcounts, 60
total_data_arr, 60
total_data_size, 60
write_domain_decomposition, 57
write_output, 57
write_output_ascii_parallel, 58
write_output_ascii_sequential, 58
write_output_binary_parallel, 58
write_output_binary_sequential, 59
write_times, 59
Output::output< T >, 52
output_cfg
Output::output, 56
output_format
ConfigUtils::arguments, 14
output_option
ConfigUtils::arguments, 14
output_time_series
Output::output, 56

parse_cfg
config_utils.h, 72
parse_extbc_file
config_utils.h, 73
parse_src_location
config_utils.h, 73
part_dims
MpiUtils::partition_data_t, 63
partition_data_t
MpiUtils::partition_data_t, 61, 62
pow
Matrix::matrix, 49
print_interval
ConfigUtils::arguments, 14
print_option
ConfigUtils::arguments, 15

QX
constants.h, 83
qx_infile
ConfigUtils::arguments, 15
QY
constants.h, 83
qy_infile
ConfigUtils::arguments, 15

RESET
constants.h, 83
RESIZE_TIME
constants.h, 83
RED
constants.h, 83
RHSH0
constants.h, 84
RHSH1
constants.h, 84
RHSQX0
constants.h, 84
RHSQX1
constants.h, 84
RHSQY0
constants.h, 84
RHSQY1
constants.h, 84
RUNID
constants.h, 84
RUNIN
constants.h, 84
read_and_parse_checkpoint_partition
config_utils.h, 74
recvcounts
Output::output, 60
remove_ghost_cells
Matrix::matrix, 49
reset

SuperTimer::super_timer, 66
resize
Matrix::matrix, 49
restart
SuperTimer::super_timer, 66
rows
MpiUtils::partition_data_t, 63
rows_ini
MpiUtils::partition_data_t, 63
runoff_filename
ConfigUtils::arguments, 15
runoff_map
ConfigUtils::arguments, 15

SEC_TO_HOUR_FACTOR
constants.h, 85
SIMULATION_TIME
constants.h, 85
SQRTG
constants.h, 85
SQRTH
constants.h, 85
SRC_LOCATION
constants.h, 85
SRCP
constants.h, 85
scatter_exchange
mpi_utils.h, 99
scatter_exchange_int
mpi_utils.h, 99
set_cell_size
DemFile::dem_file, 22
set_infinite_walls
Matrix::matrix, 49
set_ncols
DemFile::dem_file, 22
set_no_data_value
DemFile::dem_file, 22
set_nrows
DemFile::dem_file, 23
set_num_flow_rows
Hydrograph::hydrograph, 34
set_num_rows
ExtBC::extBC, 29
set_num_sources
Hydrograph::hydrograph, 35
set_size
Matrix::matrix, 50
set_value
Matrix::matrix, 50, 51
set_xll_corner
DemFile::dem_file, 23
set_yll_corner
DemFile::dem_file, 23
sim_duration
ConfigUtils::arguments, 15
sim_start_time
ConfigUtils::arguments, 15
simulate
Triton::triton, 68
size
MpiUtils::partition_data_t, 63
sources_list_t
constants.h, 87
split
string_utils.h, 106
square
Matrix::matrix, 51
src_loc_file
ConfigUtils::arguments, 16
src_x_loc
ConfigUtils::arguments, 16
src_y_loc
ConfigUtils::arguments, 16
start
SuperTimer::super_timer, 66
stop
SuperTimer::super_timer, 66
string_utils.h, 102
down_char, 103
is_numeric, 104
itoa, 104
itos, 104
split, 106
tolower, 106
toupper, 107
up_char, 107
vecstr_to_vecflt, 108
vecstr_to_vecint, 108
string_vector
constants.h, 87
super_timer
SuperTimer::super_timer, 64
SuperTimer::ci_less, 17
operator(), 17
SuperTimer::ci_less::nocase_compare, 52
operator(), 52
SuperTimer::super_timer, 63
add_new_timer, 64
get_current_date, 65
get_custom_time, 65
get_hostname, 65
get_total_time, 65
reset, 66
restart, 66
start, 66
stop, 66
super_timer, 64
supertimer.h, 108

THREAD_BLOCK
constants.h, 85
TIME_SERIES_DIR
constants.h, 85
TIMER_NSECS
constants.h, 86
TIMER_SECS
constants.h, 86

TOTAL_TIME
    constants.h, 86
TYPE_DYNAMIC
    constants.h, 86
TYPE_STATIC
    constants.h, 86
time_increment_fixed
    ConfigUtils::arguments, 16
time_series_flag
    ConfigUtils::arguments, 16
time_step
    ConfigUtils::arguments, 16
tolower
    string_utils.h, 106
total_data_arr
    Output::output, 60
total_data_size
    Output::output, 60
toupper
    string_utils.h, 107
triton
    Triton::triton, 67
triton.h, 110
Triton::triton
    ∼triton, 68
    initialize, 68
    simulate, 68
    triton, 67
Triton::triton< T >, 67

USE_HALO
    constants.h, 86
USE_MATRIX
    constants.h, 86
ull
    constants.h, 87
up_char
    string_utils.h, 107

value_t
    constants.h, 87
vecstr_to_vecflt
    string_utils.h, 108
vecstr_to_vecint
    string_utils.h, 108

WARN
    constants.h, 86
write_domain_decomposition
    Output::output, 57
write_output
    Output::output, 57
write_output_ascii_parallel
    Output::output, 58
write_output_ascii_sequential
    Output::output, 58
write_output_binary_parallel
    Output::output, 58
write_output_binary_sequential
    Output::output, 59
write_times
    Output::output, 59

YELLOW
    constants.h, 87

zero_fill
    Matrix::matrix, 51
zero_fill_int
    Matrix::matrix, 51